MATH 8090: Geostatistics

Whitney Huang, Clemson University

11/18-20/2026

Contents

Toy Examples . . . 2
- Create Spatial Locations . . . 2
- Create spatial locations . . . 2
- Case 1: No Spatial Pattern . . . 2
- Case 2: A Smooth Spatial Image . . . 3

Using Variogram Cloud and (Binned) Variogram to Examine the Spatial Dependence . . . 4

Gaussian Processes: Covariance Functions and Their Realizations . . . 6
- Simulate 1D Realizations . . . 6
- Simulate 2D Realizations . . . 8

Spatial Interpolation . . . 9
- Predicting One Location . . . 9
- Predicting Multiple Locations . . . 11

Conditional Simulation and Uncertainty in Spatial Functionals . . . 13

Applied Spatial Modeling: Lake Conductivity (`spmodel` and `fields`) . . . 17
- Data and Exploratory Analysis . . . 17
- Nonspatial Regression: Establishing a Benchmark . . . 20
- Empirical Semivariogram of Regression Residuals . . . 22
- Spatial Linear Models: Exponential and Gaussian Covariance . . . 23
- Kriging Prediction: Trend and Spatial Correction . . . 26
- Decomposition of the Kriging Predictor . . . 27
- Visualizing the Kriging Decomposition . . . 28
- Prediction Uncertainty . . . 30
- Spatial Correction and Observed Locations . . . 31
- Spatial Model Diagnostics . . . 32
- Out-of-Sample Prediction Assessment . . . 34
- Complementary Spatial Modeling with `fields` . . . 37

Summary . . . 40
Applied Spatial Modeling: Rainfall in Paraná, Brazil (geoR) . . . 40
Data and Exploratory Analysis . . . 40
Exploring the Large-Scale Spatial Trend . . . 42
Empirical Semivariogram: Before and After Detrending . . . 45
Equal-Pair-Count Variogram Binning . . . 46
Directional Variograms and Isotropy . . . 48
Variogram-Based Covariance Estimation . . . 49
Maximum Likelihood Estimation . . . 51
Constructing a Spatial Prediction Grid . . . 53
Kriging Prediction: Trend and Spatial Correction . . . 54
Decomposing the Kriging Prediction . . . 55
Visualizing Trend, Spatial Correction, and Kriging Prediction . . . 56
Kriging Prediction Uncertainty . . . 57
Conditional Simulations . . . 59
Out-of-Sample Prediction Assessment . . . 61
Observed Versus Predicted Rainfall . . . 64
Summary and Discussion . . . 65

Toy Examples

We consider two spatial fields: one with independent observations and another with spatial dependence. In each case, we remove one observation (marked by ?) and consider how neighboring values might help predict it.

Create Spatial Locations

We first construct a regular 30×30 grid over the unit square.

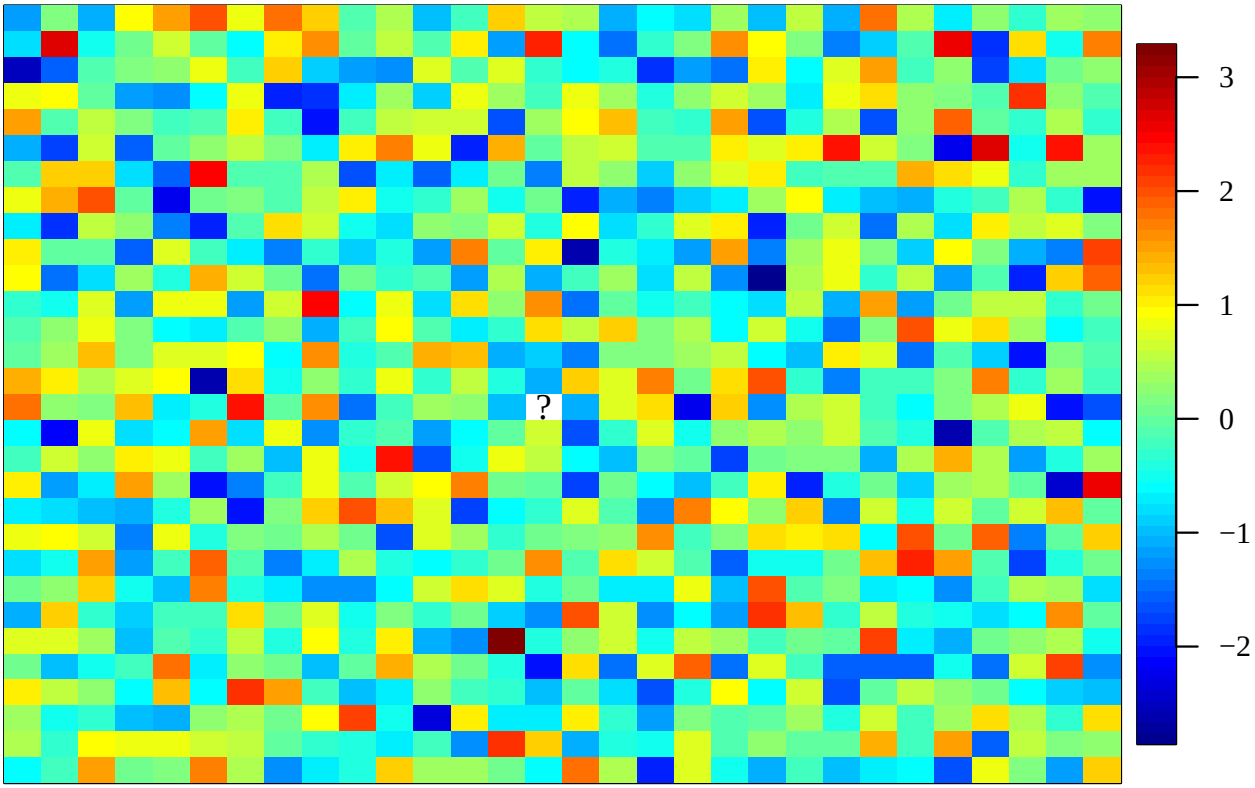
Create spatial locations

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

Case 1: No Spatial Pattern

We generate independent standard normal observations. Since there is no spatial dependence, nearby observations provide no more information about the missing value than distant ones.

```r
plot_style()
set.seed(123)

# Independent N(0,1) observations
y1 <- array(rnorm(N^2), dim = c(N, N))

# Remove one observation for prediction
y1[N / 2, N / 2] <- NA

library(fields)
fields::image.plot(xg, yg, y1, col = map_colors, xlab = "", ylab = "",
                   xaxt = "n", yaxt = "n", legend.mar = 2.5)
text(xg[N / 2], yg[N / 2], "?")
```

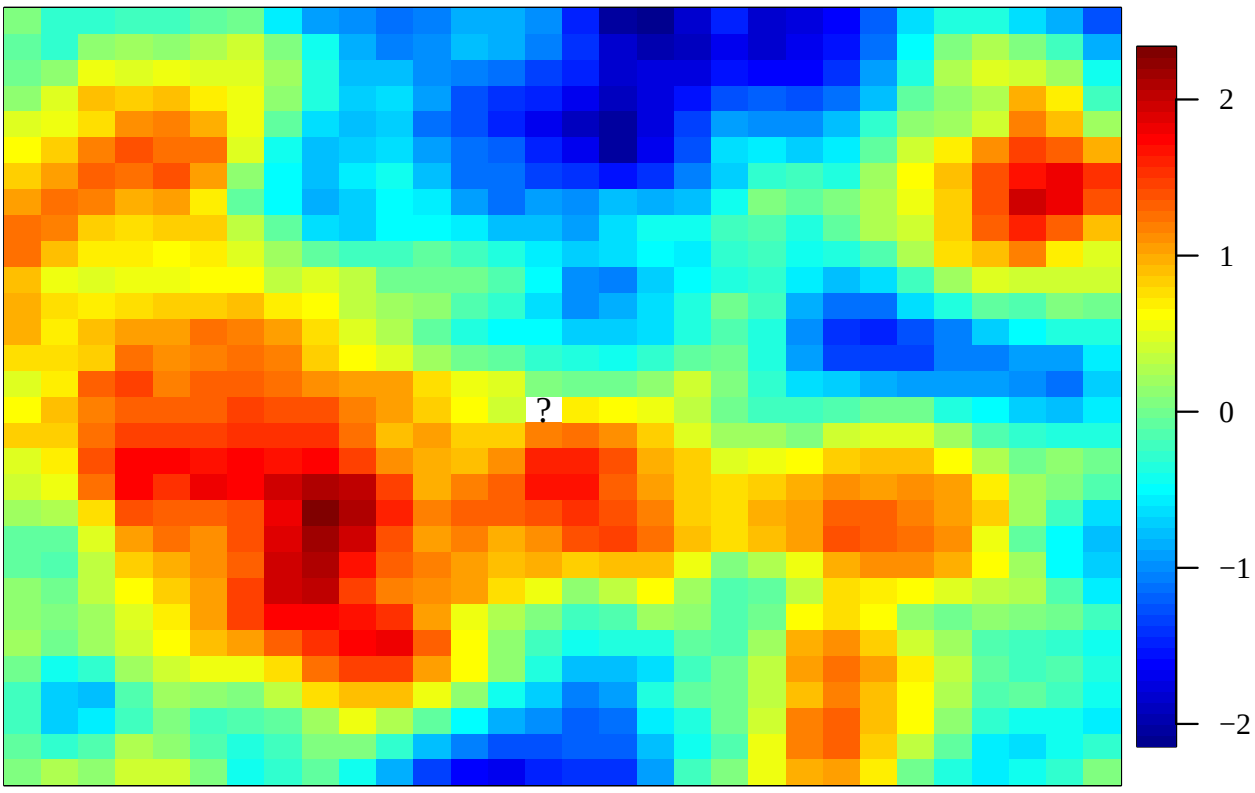


Case 2: A Smooth Spatial Image

We now simulate a Gaussian random field with a Matérn covariance function. Unlike Case 1, nearby locations tend to have similar values, making spatial prediction potentially more informative.

```r
library(MASS)

# Matérn covariance function
cov.Matern <- function(h, pars) {
  Matern(h, phi = pars[1], range = pars[2], smoothness = pars[3])
}

# Pairwise distances and covariance matrix
```

```r
dist <- rdist(locs)
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 1.5))

# Simulate a spatially correlated Gaussian field
set.seed(123)
y2 <- array(mvrnorm(n = 1, rep(0, N^2), Sigma_Matern), dim = c(N, N))

# Remove the same observation
y2[N / 2, N / 2] <- NA

plot_style()
fields::image.plot(xg, yg, y2, col = map_colors, xlab = "", ylab = "",
                   xaxt = "n", yaxt = "n", legend.mar = 2.5)
text(xg[N / 2], yg[N / 2], "?")
```



Key observation: Spatial dependence makes neighboring observations informative for predicting the missing value. The variogram provides a way to quantify this dependence.

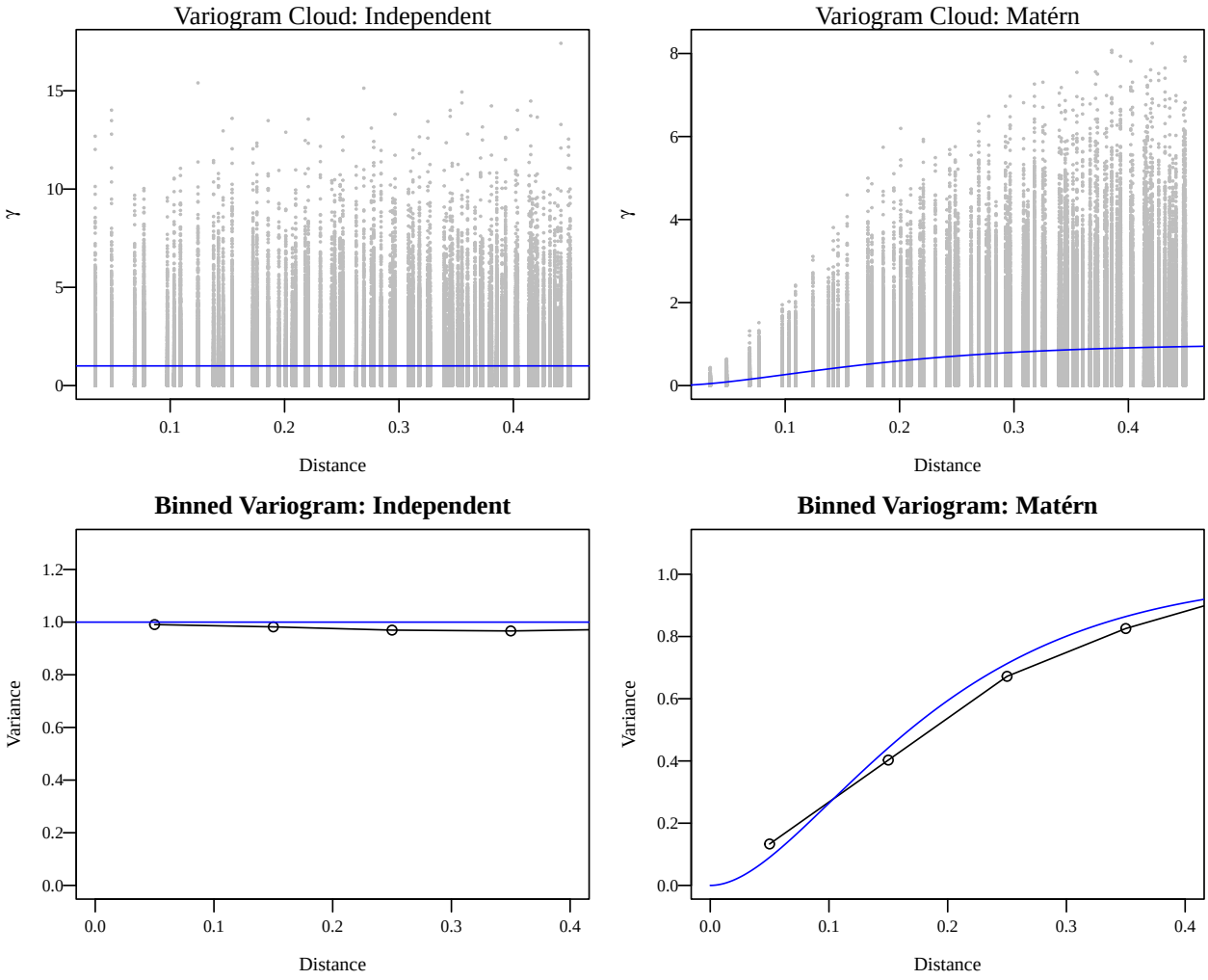
Using Variogram Cloud and (Binned) Variogram to Examine the Spatial Dependence

The empirical semivariance between observations at locations $\mathbf{s}_i$ and $\mathbf{s}_j$ is

$$\widehat{\gamma}_{ij} = \frac{1}{2}\{Y(\mathbf{s}_i) - Y(\mathbf{s}_j)\}^2.$$

The **variogram cloud** displays individual pairwise semivariances, while the **binned variogram** averages them over distance intervals. We compare both fields with their theoretical semivariograms (blue curves).

```r
gamma1 <- outer(c(y1), c(y1), FUN = "-")^2
gamma2 <- outer(c(y2), c(y2), FUN = "-")^2

# Use unique pairs within the distance cutoff
dist_cut <- 0.45
good <- which(upper.tri(dist) & dist <= dist_cut &
              !is.na(gamma1) & !is.na(gamma2))

# Theoretical Matérn semivariogram
dgrid <- seq(0, sqrt(2), 0.001)
gamma_theory <- 1 - cov.Matern(dgrid, c(1, 0.1, 1.5))

plot_style(mar = c(3.2, 3.3, 1.9, 0.7), mfrow = c(2, 2))

# Independent field: variogram cloud
plot(c(dist)[good], c(gamma1)[good] / 2, col = "gray",
     cex = 0.2, xlab = "Distance", ylab = expression(gamma))
abline(h = 1, col = "blue")
mtext("Variogram Cloud: Independent")

# Correlated field: variogram cloud
plot(c(dist)[good], c(gamma2)[good] / 2, col = "gray",
     cex = 0.2, xlab = "Distance", ylab = expression(gamma))
lines(dgrid, gamma_theory, col = "blue")
mtext("Variogram Cloud: Matérn")

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

# Correlated field: binned variogram
vargram2 <- vgram(locs, c(y2), N = 30)
plot(vargram2, ylim = c(0, 1.1), xlim = c(0, 0.4), main = "Binned Variogram: Matérn")
lines(dgrid, gamma_theory, col = "blue")
```



Interpretation:

- **Independent field:** The semivariogram is approximately constant at $\gamma(h) = 1$, indicating no spatial dependence at nonzero distances.
- **Matérn field:** Semivariance increases with distance, reflecting stronger similarity between nearby observations.
- **Binning:** Averaging pairwise semivariances reduces scatter and makes the underlying spatial dependence easier to identify.

These examples motivate fitting a parametric semivariogram model, which provides the covariance structure needed for **kriging**.

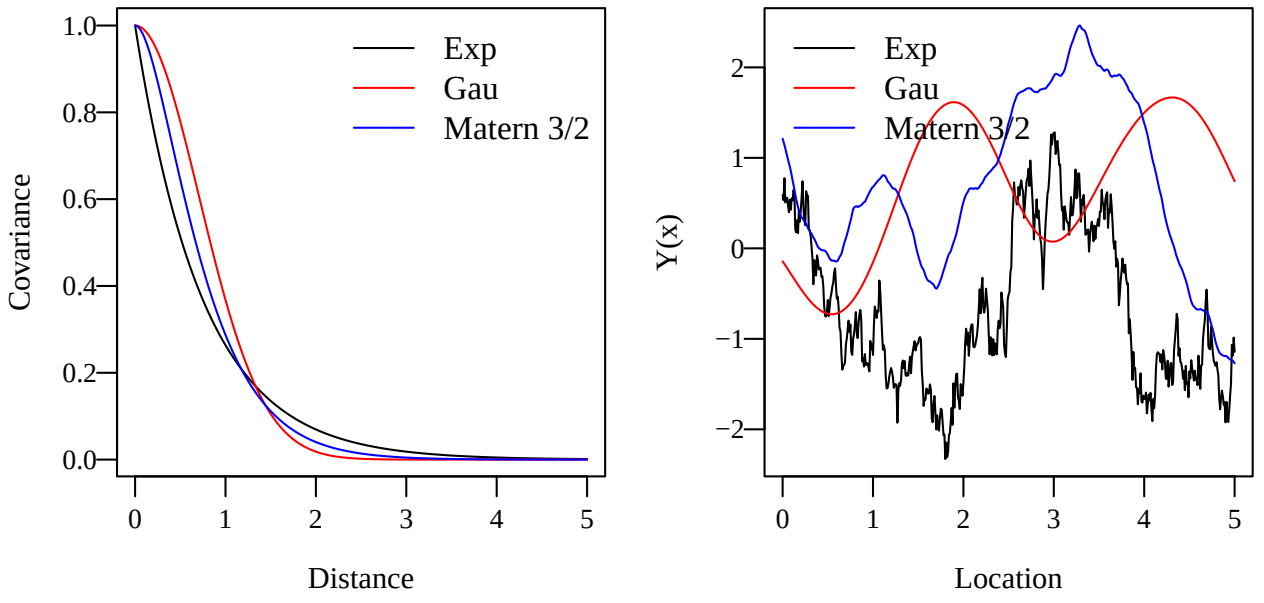
Gaussian Processes: Covariance Functions and Their Realizations

We compare three covariance functions and their Gaussian process realizations. The covariance function determines how dependence decreases with distance and influences the smoothness of the resulting process.

Simulate 1D Realizations

We first compare the exponential, Gaussian (squared exponential), and Matérn covariance functions, followed by one simulated realization from each.

```r
# 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)

xx <- seq(0, 5, by = 0.01)

# Compare covariance decay with distance
cov_funs <- list(
  Exp = function(h) cov.exp(h, c(1, 0.75)),
  Gau = function(h) cov.doubleExp(h, c(1, 1)),
  "Matern 3/2" = function(h) cov.Matern(h, c(1, 0.4, 1.5))
)
cols <- c("black", "red", "blue")

plot_style(mfrow = c(1, 2))
plot(xx, cov_funs[[1]](xx), type = "l", col = cols[1],
     xlab = "Distance", ylab = "Covariance")
for (i in 2:3) lines(xx, cov_funs[[i]](xx), col = cols[i])
legend("topright", names(cov_funs), col = cols, lty = 1, bty = "n")

# Simulate one GP realization for each covariance model
D <- fields::rdist(matrix(xx, ncol = 1))
set.seed(1234)
sim <- lapply(cov_funs, function(f)
  MASS::mvrnorm(1, rep(0, length(xx)), f(D)))

plot(xx, sim[[1]], type = "l", col = cols[1],
     ylim = range(unlist(sim)), xlab = "Location", ylab = "Y(x)")
for (i in 2:3) lines(xx, sim[[i]], col = cols[i])
legend("topleft", names(cov_funs), col = cols, lty = 1, bty = "n")
```

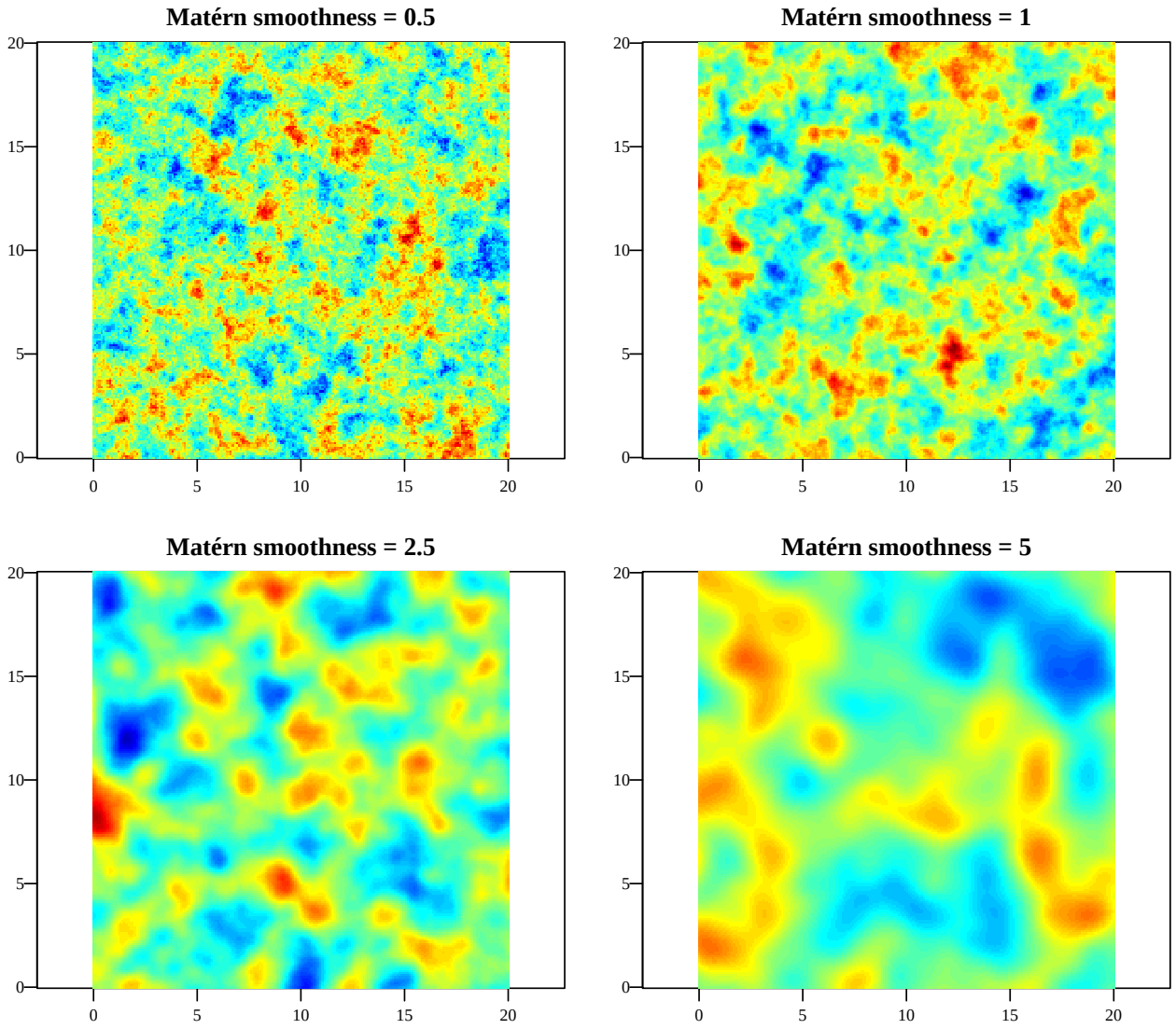



```r
par(mfrow = c(1, 1))
```

Simulate 2D Realizations

We now examine how the Matérn smoothness parameter ν affects spatial patterns. Larger ν produces smoother random fields, while smaller ν produces rougher spatial variation.

```
grid <- list(x = seq(0, 20, length.out = 200), y = seq(0, 20, length.out = 200))
nu <- c(0.5, 1, 2.5, 5)

# Set up Matérn covariance models with different smoothness
obj <- lapply(nu, function(v)
  matern.image.cov(grid = grid, theta = 0.5, smoothness = v, setup = TRUE))

# Simulate one spatial field for each model
set.seed(2021)
sim <- lapply(obj, sim.rf)

# Compare fields using a common color scale
plot_style(mar = c(2.5, 2.6, 2, 0.5), mfrow = c(2, 2))
for (i in seq_along(nu)) {
  image(grid$x, grid$y, sim[[i]], col = map_colors,
        zlim = c(-4.25, 4.25), xlab = "", ylab = "",
        main = paste0("Matérn smoothness = ", nu[i]), asp = 1)
}
```




```
par(mfrow = c(1, 1))
```

Interpretation: As ν increases from 0.5 to 5, the simulated fields become progressively smoother, with less abrupt local variation. The Matérn family therefore provides a flexible way to model spatial processes with different degrees of regularity.

Spatial Interpolation

We assume a zero-mean Gaussian process, $m(s) = 0$, with a known Matérn covariance function. Kriging uses observed values and their spatial dependence to predict unobserved locations.

Predicting One Location

For an unobserved location s_0, the conditional mean and variance are

$$\hat{y}_0 = k^\top \Sigma^{-1} \mathbf{y}, \qquad \mathrm{Var}(Y_0 \mid \mathbf{Y}) = \sigma^2 - k^\top \Sigma^{-1} k,$$

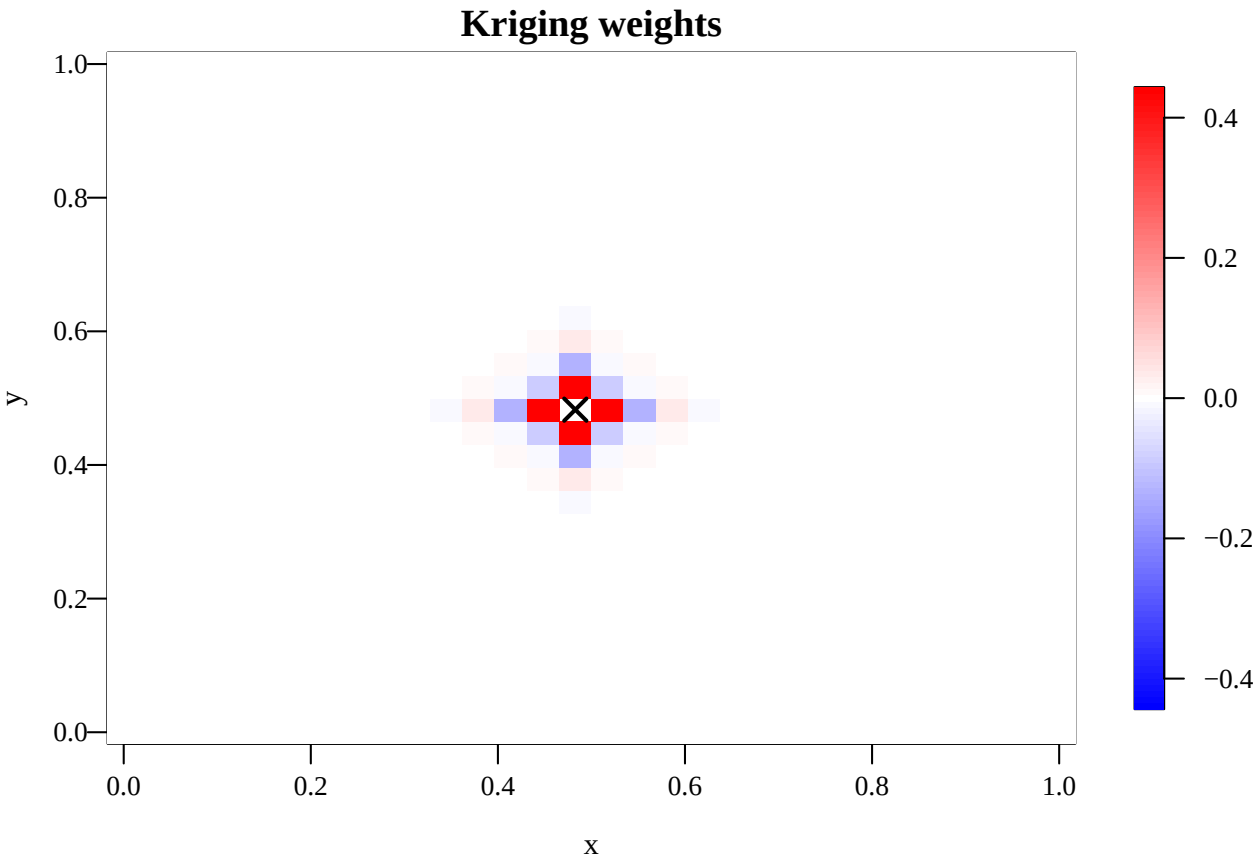
where k is the covariance vector between the prediction location and observed locations, and Σ is the covariance matrix of the observations.

We remove one observation and predict it using the remaining 899 locations. We also visualize the kriging weights to examine how neighboring observations contribute to the prediction.

```r
# Remove one observation near the center
idx <- (N / 2 - 1) * N + N / 2
y_true <- y2[idx]
y2[idx] <- NA
obs <- which(!is.na(y2))

# Partition the covariance matrix
Sigma <- Sigma_Matern[obs, obs]
k <- Sigma_Matern[obs, idx]
y <- c(y2)[obs]

# Kriging weights, prediction, and uncertainty
w <- solve(Sigma, k)
y0_hat <- sum(w * y)
var_y0_hat <- Sigma_Matern[idx, idx] - sum(k * w)

round(c(True_value = y_true, Prediction = y0_hat,
        Kriging_SD = sqrt(pmax(0, var_y0_hat))), 3)
```

```
## True_value Prediction Kriging_SD
##         NA      0.554      0.081
```

```r
# Map kriging weights to spatial locations
weight_map <- matrix(NA_real_, N, N)
weight_map[obs] <- w

# Diverging color scale centered at zero
lim <- max(abs(w))
cols <- colorRampPalette(c("blue", "white", "red"))(101)

plot_style()
par(mar = c(3.5, 3.5, 1.5, 2))

fields::image.plot(xg, yg, weight_map, col = cols,
                   zlim = c(-lim, lim),
                   xlab = "x", ylab = "y",
                   main = "Kriging weights",
                   legend.mar = 3, legend.width = 1)

points(xg[N / 2], yg[N / 2], pch = 4, cex = 1.5, lwd = 2)
```



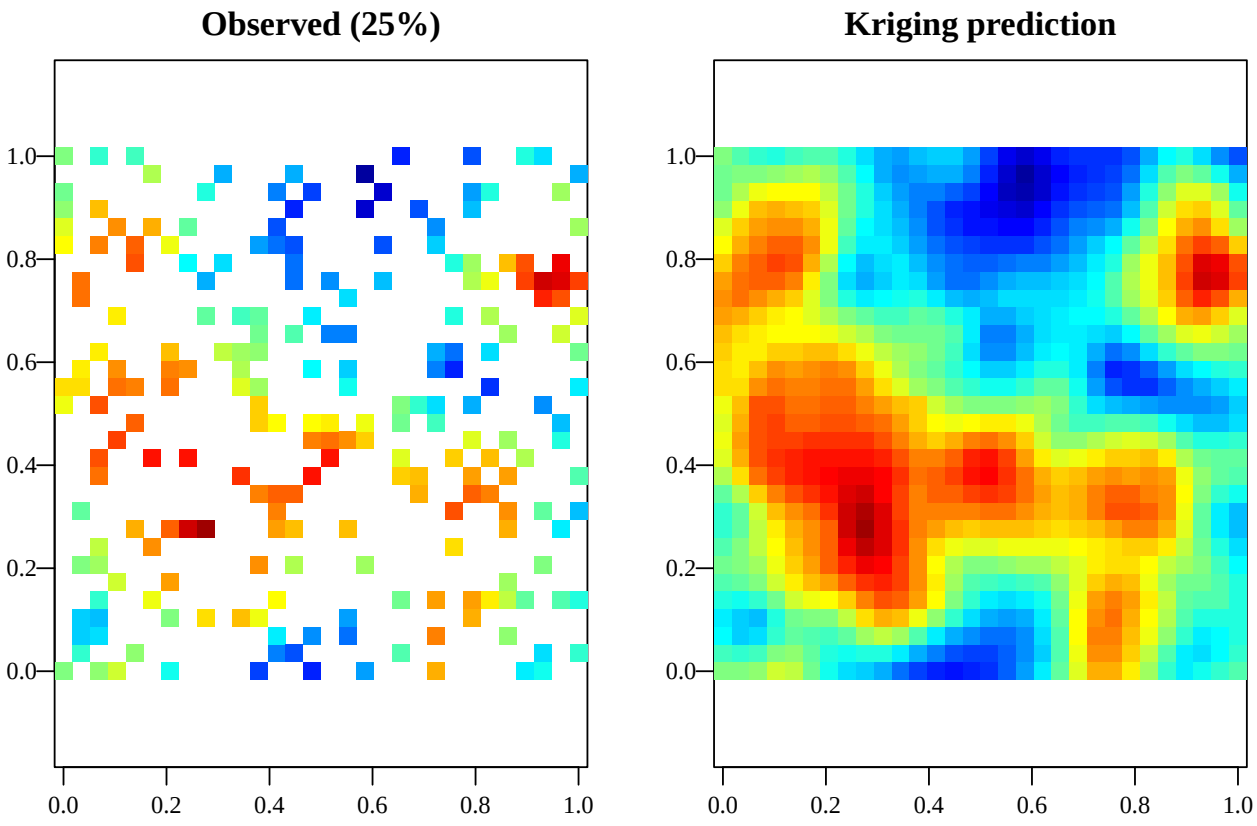
Interpretation: Kriging assigns weights according to spatial dependence. Nearby observations generally contribute more to predicting the missing value, while distant observations receive weights close to zero. Some weights may be negative because kriging accounts for correlations among observed locations.

Predicting Multiple Locations

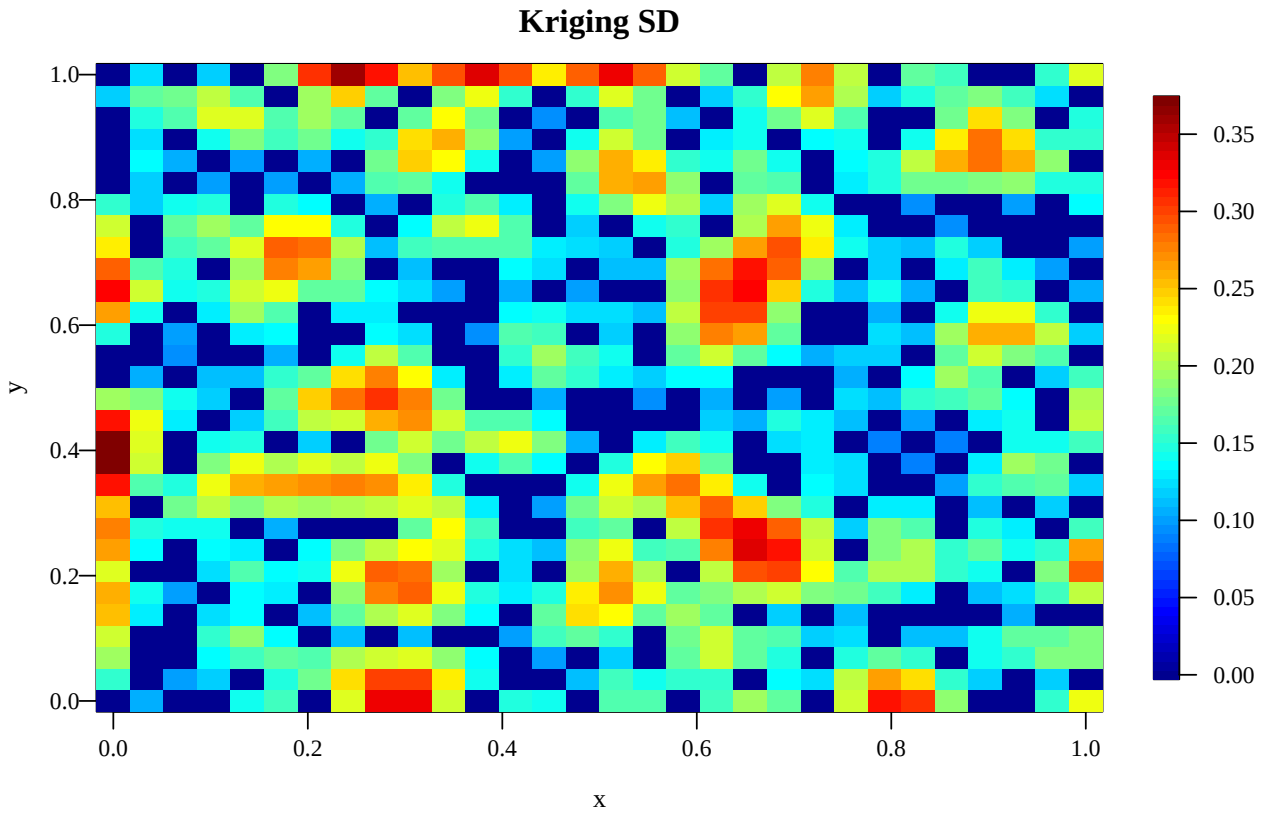
We now randomly remove 75% of the observations and use the remaining locations for kriging. The conditional mean reconstructs the spatial field, while the conditional standard deviation quantifies prediction uncertainty.

```
# Recover the original complete simulated field
set.seed(123)
y_full <- matrix(MASS::mvrnorm(1, rep(0, N^2), Sigma_Matern), N, N)

# Randomly remove 75% of observations
set.seed(123)
rm <- sample(seq_len(N^2), 0.75 * N^2)
obs <- setdiff(seq_len(N^2), rm)

y2_rm <- y_full
y2_rm[rm] <- NA

# Covariance blocks for kriging
Sigma <- Sigma_Matern[obs, obs]
k <- Sigma_Matern[obs, rm, drop = FALSE]
```

```r
# Kriging predictions and conditional standard deviations
R <- chol(Sigma)
alpha <- backsolve(R, forwardsolve(t(R), y_full[obs]))
v <- backsolve(R, forwardsolve(t(R), k))

y0_hat <- drop(crossprod(k, alpha))
con_sd <- sqrt(pmax(0, diag(Sigma_Matern)[rm] - colSums(k * v)))

# Complete field: retain observed values and predict missing values
y2_pred <- y2_rm
y2_pred[rm] <- y0_hat

# Prediction uncertainty is zero at observed locations
condSD <- matrix(0, N, N)
condSD[rm] <- con_sd

# Compare observed data and complete kriging prediction
common_zlim <- range(c(y_full, y2_pred))

plot_style(mar = c(2.8, 2.8, 2.1, 0.8), mfrow = c(1, 2))

image(xg, yg, y2_rm, col = map_colors, zlim = common_zlim,
      xlab = "", ylab = "", main = "Observed (25%)", asp = 1)

image(xg, yg, y2_pred, col = map_colors, zlim = common_zlim,
      xlab = "", ylab = "", main = "Kriging prediction", asp = 1)
```



```r
par(mfrow = c(1, 1))

# Map conditional standard deviation
plot_style(mar = c(3.5, 3.5, 2.5, 2))
fields::image.plot(xg, yg, condSD, col = map_colors,
                   xlab = "x", ylab = "y", main = "Kriging SD",
                   legend.mar = 3, legend.width = 1)
```



Interpretation: Kriging uses spatial dependence to reconstruct the missing values, producing a smoother field than the original realization. The conditional standard deviation is zero at observed locations and generally increases with distance from observations. Regions with fewer nearby observations therefore have greater prediction uncertainty.

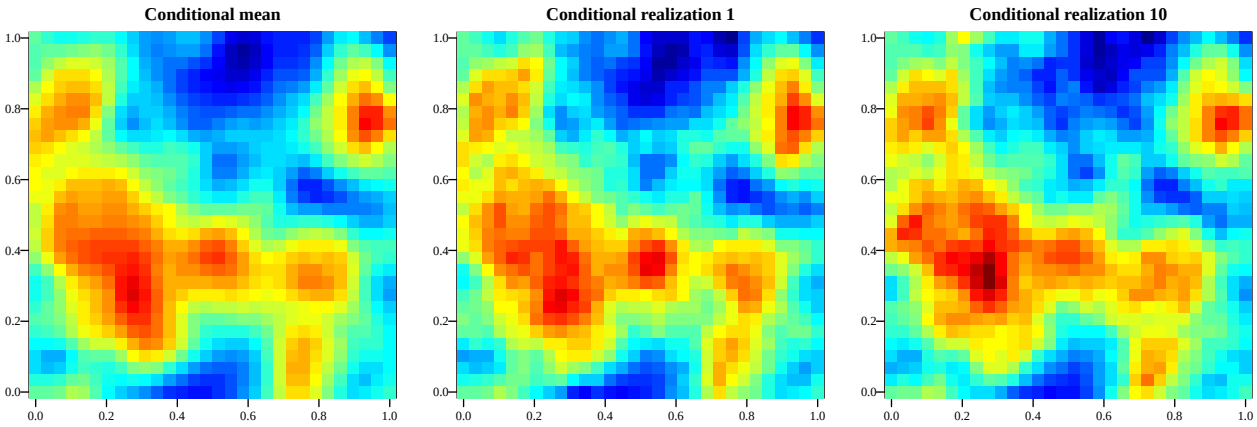
Conditional Simulation and Uncertainty in Spatial Functionals

Kriging provides the conditional mean and covariance of a Gaussian field. However, the kriging mean alone does not capture uncertainty in *spatial functionals*, such as exceedance areas or spatial maxima. To quantify this uncertainty, we simulate entire spatial fields conditional on the observed data.

We return to the simulated Matérn field, treating its covariance parameters as known. For observed locations o and unobserved locations u,

$$Y_u \mid Y_o = y_o \sim N\left(\Sigma_{uo}\Sigma_{oo}^{-1}y_o,\ \Sigma_{uu} - \Sigma_{uo}\Sigma_{oo}^{-1}\Sigma_{ou}\right).$$

```r
# Partition locations into observed and missing
obs <- which(!is.na(y2_rm))
miss <- which(is.na(y2_rm))

S_oo <- Sigma_Matern[obs, obs, drop = FALSE]
S_uo <- Sigma_Matern[miss, obs, drop = FALSE]
S_uu <- Sigma_Matern[miss, miss, drop = FALSE]

# Conditional mean and covariance using Cholesky solves
R_oo <- chol(S_oo)
A <- backsolve(R_oo, forwardsolve(t(R_oo), t(S_uo)))

mu_u <- drop(S_uo %*%
  backsolve(R_oo, forwardsolve(t(R_oo), c(y2_rm)[obs])))

V_u <- S_uu - S_uo %*% A
V_u <- (V_u + t(V_u)) / 2
R_u <- chol(V_u + diag(1e-10, length(miss)))

# Generate conditional realizations
set.seed(2026)
B <- 100
sim_u <- matrix(mu_u, length(miss), B) +
  t(R_u) %*% matrix(rnorm(length(miss) * B), length(miss), B)

# Restore observed values in every realization
sim_full <- matrix(rep(c(y2_rm), B), N^2, B)
sim_full[miss, ] <- sim_u

# Compare the conditional mean and one simulated field
mean_field <- y2_rm
mean_field[miss] <- mu_u

plot_style(mar = c(2.8, 2.8, 2.1, 0.8), mfrow = c(1, 3))
zlim <- range(c(mean_field, sim_full[, 1], sim_full[, 10]))

image(xg, yg, mean_field, col = map_colors, zlim = zlim,
      xlab = "", ylab = "", main = "Conditional mean", asp = 1)

image(xg, yg, matrix(sim_full[, 1], N, N),
      col = map_colors, zlim = zlim,
      xlab = "", ylab = "", main = "Conditional realization 1", asp = 1)

image(xg, yg, matrix(sim_full[, 10], N, N),
      col = map_colors, zlim = zlim,
      xlab = "", ylab = "", main = "Conditional realization 10", asp = 1)
```




```
par(mfrow = c(1, 1))
```

Unlike the kriging mean, a conditional realization retains spatial variability at unobserved locations while reproducing the observed values. Repeated realizations provide a Monte Carlo approximation to the conditional distribution of spatial functionals.

Uncertainty in Spatial Functionals We consider three functionals defined over the spatial grid $\mathcal{S}$:

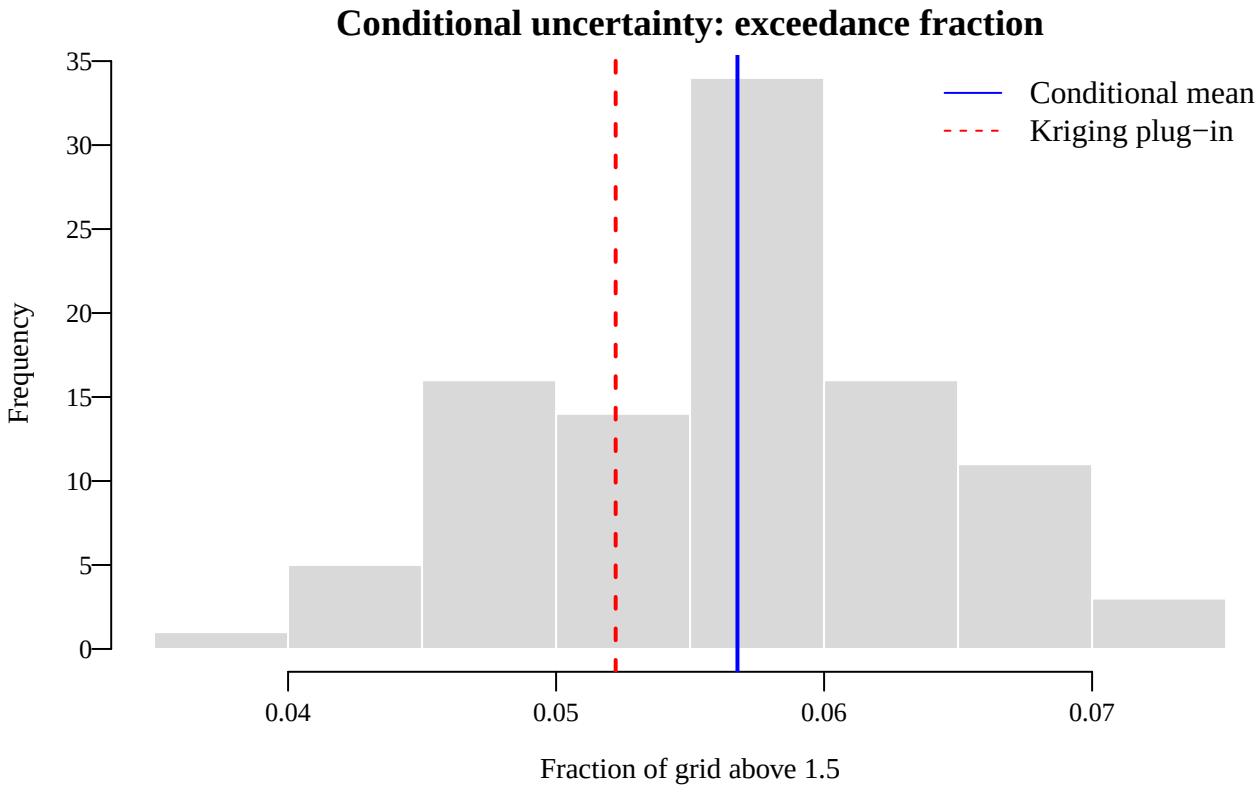
- **Spatial average:** $T_1(Y) = |\mathcal{S}|^{-1} \sum_{s \in \mathcal{S}} Y(s)$.
- **Exceedance fraction:** $T_2(Y) = |\mathcal{S}|^{-1} \sum_{s \in \mathcal{S}} I\{Y(s) > c\}$.
- **Spatial maximum:** $T_3(Y) = \max_{s \in \mathcal{S}} Y(s)$.

For each functional, we compare its value computed from the kriging mean field with its conditional Monte Carlo mean and 95% interval.

```
threshold <- 1.5

# Evaluate functionals across conditional realizations
functionals <- rbind(
  spatial_mean = colMeans(sim_full),
  exceedance_fraction = colMeans(sim_full > threshold),
  spatial_maximum = apply(sim_full, 2, max)
)

# Plug-in estimates based on the kriging mean field
plugin <- c(
  spatial_mean = mean(mean_field),
  exceedance_fraction = mean(mean_field > threshold),
  spatial_maximum = max(mean_field)
)

# Conditional Monte Carlo summaries
uq <- t(apply(functionals, 1, function(v)
  c(mean = mean(v), lower = unname(quantile(v, 0.025)),
    upper = unname(quantile(v, 0.975)))))

uq <- cbind(kriging_plugin = plugin[rownames(uq)], uq)
print(round(uq, 3))
```

```
##                       kriging_plugin  mean lower upper
## spatial_mean                   0.165 0.165 0.143 0.189
## exceedance_fraction            0.052 0.057 0.043 0.070
## spatial_maximum                2.108 2.245 2.108 2.602
```

```r
# Conditional distribution of exceedance fraction
plot_style(mar = c(3.5, 3.7, 1.7, 0.7))
hist(functionals["exceedance_fraction", ], breaks = 8,
     col = "gray85", border = "white",
     xlab = paste0("Fraction of grid above ", threshold),
     main = "Conditional uncertainty: exceedance fraction")

abline(v = uq["exceedance_fraction", "mean"],
       col = "blue", lwd = 2)
abline(v = uq["exceedance_fraction", "kriging_plugin"],
       col = "red", lwd = 2, lty = 2)

legend("topright", c("Conditional mean", "Kriging plug-in"),
       col = c("blue", "red"), lty = c(1, 2), bty = "n")
```



Interpretation: The spatial average is a linear functional, so its conditional expectation equals the average of the kriging mean field (up to Monte Carlo error). In contrast, exceedance fractions and spatial maxima are nonlinear, and evaluating them at the kriging mean generally differs from averaging their values over conditional realizations.

The simulation-based intervals quantify **conditional process uncertainty**, treating covariance parameters as known. They do not account for parameter estimation uncertainty.

Applied Spatial Modeling: Lake Conductivity (`spmodel` and `fields`)

We now move from kriging with known covariance parameters to spatial modeling in which the mean and covariance parameters are estimated from data. This application follows the framework of the 2024 ENVR short course by Ver Hoef, Zimmerman, and Dumelle, using the `lake` and `lake_preds` datasets from `ZVHdata`.

Our objectives are to:

1. Examine the mean relationship between conductivity, temperature, and lake origin.
2. Investigate residual spatial dependence.
3. Fit and compare spatial covariance models.
4. Decompose kriging predictions into regression trend and spatial correction.
5. Visualize prediction uncertainty and assess predictive performance.

Data and Exploratory Analysis

The response is log conductivity. Temperature and lake origin enter the mean model, while geographic coordinates determine spatial separation.

We first inspect the data and sampling locations. Euclidean distance-based models require projected coordinates with meaningful distance units.

```
library(sf)
library(spmodel)
library(fields)
library(ZVHdata)

data("lake", package = "ZVHdata")
data("lake_preds", package = "ZVHdata")

lake_df <- sf::st_drop_geometry(lake)
pred_df <- sf::st_drop_geometry(lake_preds)

stopifnot(
  !sf::st_is_longlat(lake),
  sf::st_crs(lake) == sf::st_crs(lake_preds)
)

xy <- sf::st_coordinates(lake)[, 1:2, drop = FALSE]
xy_pred <- sf::st_coordinates(lake_preds)[, 1:2, drop = FALSE]

# Assumes projected coordinates are in meters
xy_km <- xy / 1000
xy_pred_km <- xy_pred / 1000

z <- lake_df$log_cond

if (!exists("map_colors")) {
  map_colors <- fields::tim.colors(64)
}
if (!exists("plot_style")) {
  plot_style <- function(...) par(...)
}
```

```r
summary(lake_df[, c("log_cond", "temp", "origin")])
```

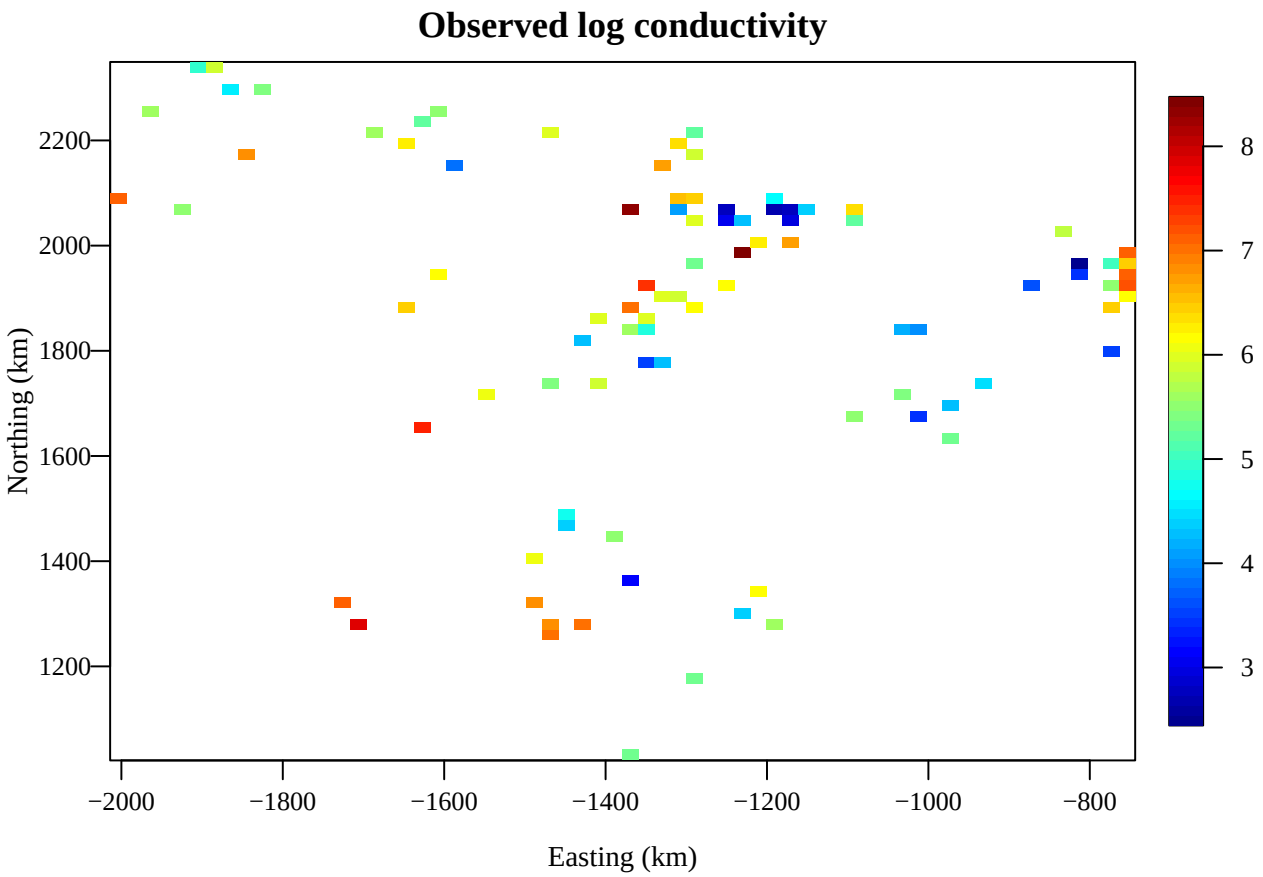
```
##     log_cond          temp           origin
##  Min.   :2.491   Min.   :-3.178   Length:102
##  1st Qu.:4.419   1st Qu.: 3.323   Class :character
##  Median :5.611   Median : 7.542   Mode  :character
##  Mean   :5.477   Mean   : 7.635
##  3rd Qu.:6.337   3rd Qu.: 9.852
##  Max.   :8.427   Max.   :23.484
```

```r
table(lake_df$origin)
```

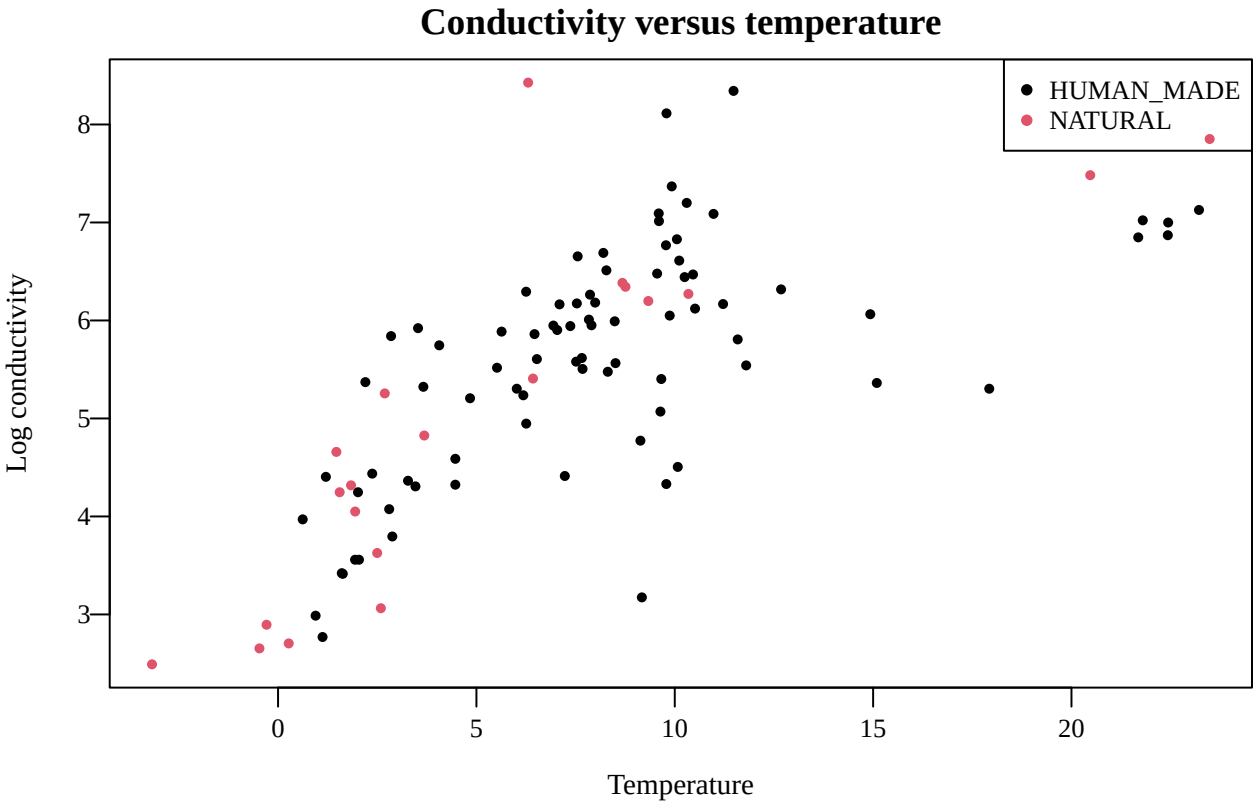
```
##
## HUMAN_MADE    NATURAL
##         82         20
```

The observed map provides an initial view of geographic variation. However, spatial patterns may arise from covariates, spatially correlated errors, or both.

```r
plot_style(mar = c(3.5, 3.8, 2, 1.2))

fields::quilt.plot(
  xy_km, z,
  col = map_colors,
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Observed log conductivity",
  legend.mar = 3
)
```




```
origin_group <- as.factor(lake_df$origin)
origin_col <- as.integer(origin_group)

plot_style(mar = c(4, 4, 2, 1))
plot(
  lake_df$temp, z,
  pch = 16, cex = 0.7,
  col = origin_col,
  xlab = "Temperature",
  ylab = "Log conductivity",
  main = "Conductivity versus temperature"
)

legend(
  "topright",
  legend = levels(origin_group),
  col = seq_along(levels(origin_group)),
  pch = 16, cex = 0.8
)
```



Interpretation. Examine whether conductivity changes systematically with temperature and whether the relationship differs by lake origin. A map of the raw response alone cannot establish residual spatial dependence.

Nonspatial Regression: Establishing a Benchmark

We begin with a conventional linear regression:

$$Y(s_i) = \beta_0 + \beta_1 \,\mathrm{Temp}(s_i) + \beta_{\mathrm{origin}(i)} + \varepsilon(s_i),$$

where the errors are assumed independent with constant variance.

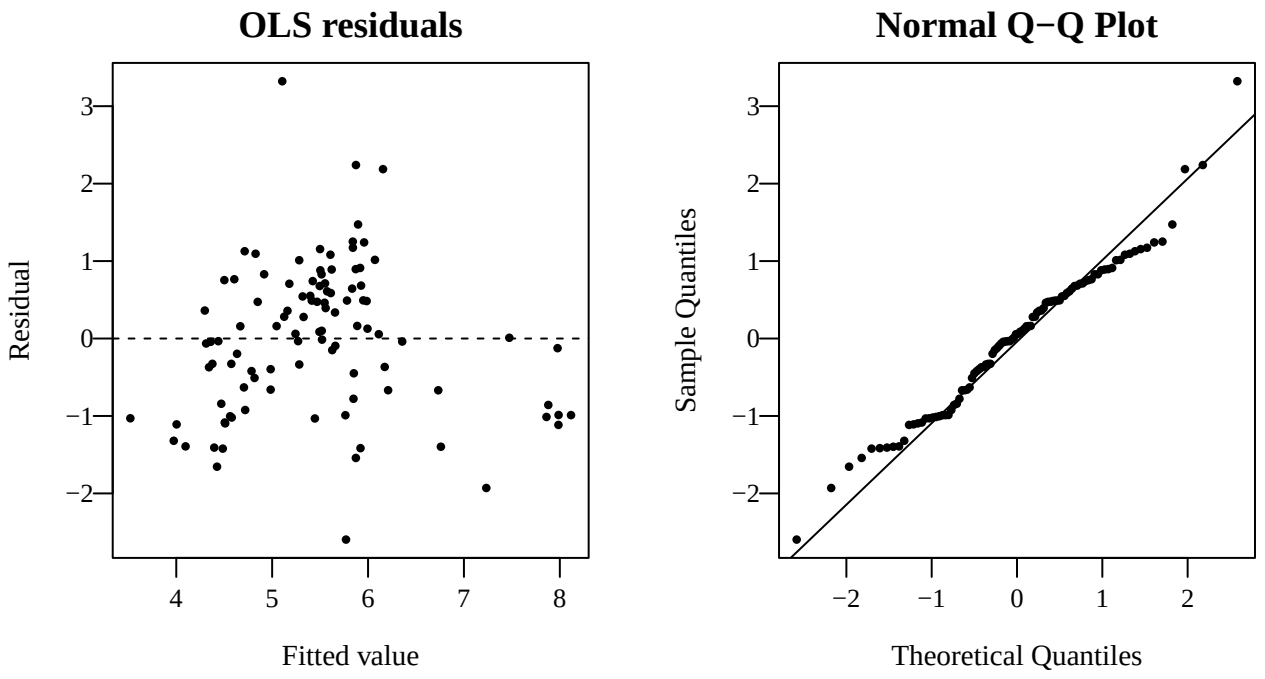
This model estimates the large-scale mean relationship without accounting for spatial correlation.

```
mean_formula <- log_cond ~ temp + origin

fit_lm <- lm(mean_formula, data = lake_df)
summary(fit_lm)
```

```
##
## Call:
## lm(formula = mean_formula, data = lake_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5961 -0.7506  0.0579  0.6694  3.3220
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.23743    0.18145  23.354  < 2e-16 ***
## temp           0.16711    0.01791   9.330 3.17e-15 ***
## originNATURAL -0.18553    0.24696  -0.751    0.454
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9703 on 99 degrees of freedom
## Multiple R-squared:  0.4875, Adjusted R-squared:  0.4771
## F-statistic: 47.08 on 2 and 99 DF,  p-value: 4.272e-15
```

```r
plot_style(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

plot(
  fitted(fit_lm), residuals(fit_lm),
  pch = 16, cex = 0.6,
  xlab = "Fitted value",
  ylab = "Residual",
  main = "OLS residuals"
)
abline(h = 0, lty = 2)

qqnorm(residuals(fit_lm), pch = 16, cex = 0.6)
qqline(residuals(fit_lm))
```




```r
par(mfrow = c(1, 1))
```

Interpretation. OLS can estimate the regression mean under appropriate exogeneity assumptions even when errors are spatially correlated. However, ignoring dependence can lead to misleading standard errors and inefficient spatial prediction.

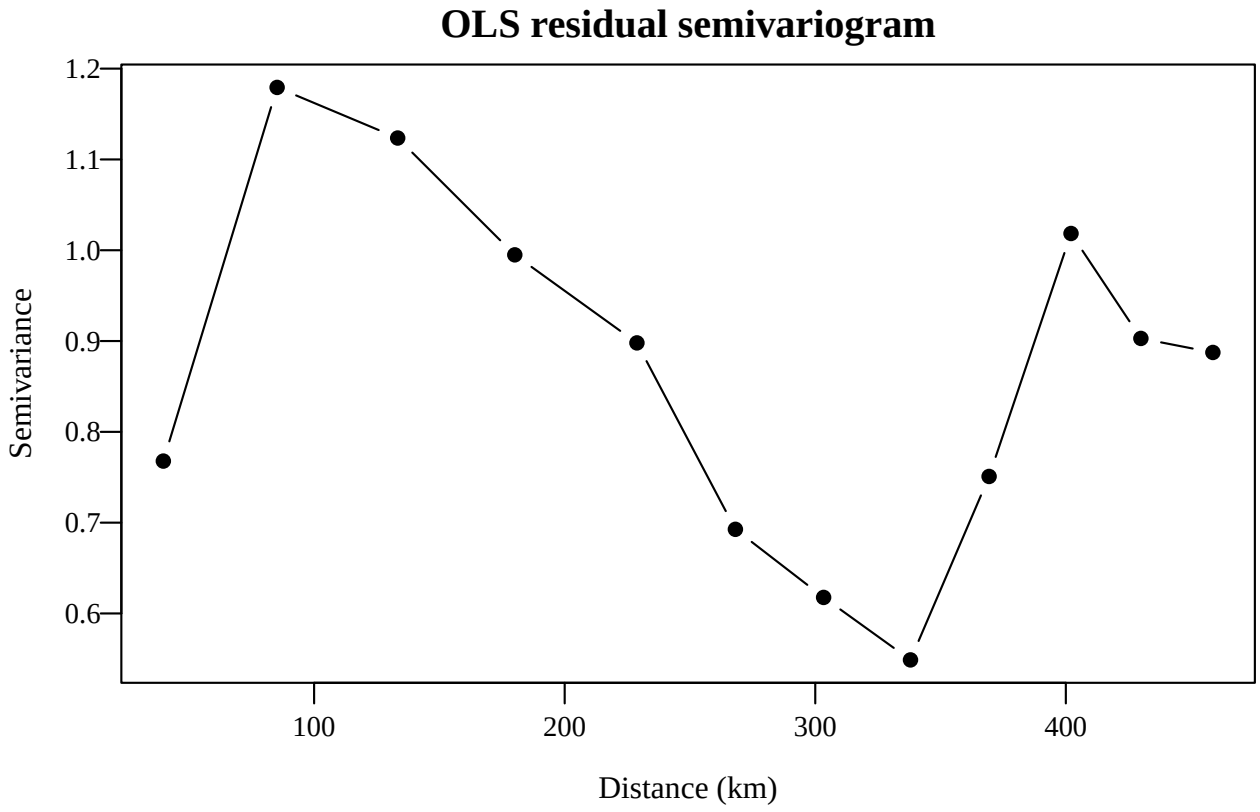
Empirical Semivariogram of Regression Residuals

The empirical semivariogram measures how residual dissimilarity changes with spatial separation:

$$\widehat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{(i,j)\in N(h)} \{e(s_i) - e(s_j)\}^2.$$

We restrict attention to distances below one-third of the maximum intersite distance. To avoid highly unbalanced distance bins, we use approximately equal numbers of location pairs per bin.

```
res_lm <- as.numeric(residuals(fit_lm))

D <- as.matrix(dist(xy_km))
idx <- which(upper.tri(D), arr.ind = TRUE)

h <- D[idx]
gamma_pair <- 0.5 *
  (res_lm[idx[, 1]] - res_lm[idx[, 2]])^2

max_lag <- max(h) / 3
keep <- is.finite(h) & h <= max_lag

h <- h[keep]
gamma_pair <- gamma_pair[keep]

breaks <- unique(as.numeric(quantile(
  h, probs = seq(0, 1, length.out = 13)
)))

bin <- cut(h, breaks = breaks, include.lowest = TRUE)

vg <- aggregate(
  cbind(distance = h, semivariance = gamma_pair),
  by = list(bin = bin),
  FUN = mean
)

vg$n_pairs <- as.numeric(table(bin))

plot_style(mar = c(4, 4, 2, 1))
plot(
  vg$distance, vg$semivariance,
  type = "b", pch = 16,
  xlab = "Distance (km)",
  ylab = "Semivariance",
  main = "OLS residual semivariogram"
)
```




```
print(vg)
```

```
##            bin  distance semivariance n_pairs
## 1  [3.04,64.1]  39.82355    0.7678120     196
## 2   (64.1,110]  85.23180    1.1793002     196
## 3    (110,155] 133.35156    1.1235791     196
## 4    (155,206] 180.08445    0.9949375     196
## 5    (206,250] 228.82784    0.8979477     195
## 6    (250,285] 268.12945    0.6926631     196
## 7    (285,320] 303.32758    0.6176748     196
## 8    (320,354] 338.00107    0.5488479     196
## 9    (354,386] 369.36427    0.7508866     195
## 10   (386,416] 402.06806    1.0184480     196
## 11   (416,445] 429.94491    0.9028686     196
## 12   (445,472] 458.68192    0.8874284     196
```

Interpretation. An increasing semivariogram suggests positive spatial correlation at shorter distances. The empirical variogram can guide covariance modeling, but it does not uniquely identify the covariance family. Equal pair counts also do not guarantee equal precision because pairs sharing locations are dependent.

Spatial Linear Models: Exponential and Gaussian Covariance

We now replace independent errors with a spatial covariance model:

$$\mathbf{Y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim N(0, \Sigma_\theta).$$

The covariance matrix describes spatially structured variation and nugget variation. We compare exponential and Gaussian covariance families while retaining the same mean model.

```
fit_exp <- spmodel::splm(
  mean_formula,
  data = lake,
  spcov_type = "exponential",
  estmethod = "reml"
)

fit_gau <- spmodel::splm(
  mean_formula,
  data = lake,
  spcov_type = "gaussian",
  estmethod = "reml"
)

summary(fit_exp)
```

```
##
## Call:
## spmodel::splm(formula = mean_formula, data = lake, spcov_type = "exponential",
##     estmethod = "reml")
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1958  0.2496  0.9623  1.4088  4.0847
##
## Coefficients (fixed):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.73445   41.55610   0.066    0.948
## temp           0.26577    0.02378  11.177   <2e-16 ***
## originNATURAL -0.06717    0.20837  -0.322    0.747
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Pseudo R-squared: 0.5638
##
## Coefficients (exponential spatial covariance):
##        de        ie     range
## 1.728e+03 5.062e-01 1.445e+09
```

```
summary(fit_gau)
```

```
##
## Call:
## spmodel::splm(formula = mean_formula, data = lake, spcov_type = "gaussian",
##     estmethod = "reml")
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.232  0.345  1.020  1.462  4.236
##
```

```
## Coefficients (fixed):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.62596    0.84787   3.097  0.00195 **
## temp           0.27383    0.02293  11.944  < 2e-16 ***
## originNATURAL -0.16097    0.21270  -0.757  0.44917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Pseudo R-squared: 0.6025
##
## Coefficients (gaussian spatial covariance):
##        de        ie     range
## 1.574e+00 6.154e-01 7.633e+05
```

```
spmodel::tidy(fit_exp)
```

```
## # A tibble: 3 x 5
##   term          estimate std.error statistic p.value
##   <chr>            <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)     2.73     41.6       0.0658   0.948
## 2 temp            0.266     0.0238   11.2      0
## 3 originNATURAL  -0.0672    0.208    -0.322    0.747
```

```
spmodel::tidy(fit_gau)
```

```
## # A tibble: 3 x 5
##   term          estimate std.error statistic p.value
##   <chr>            <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)      2.63     0.848      3.10  0.00195
## 2 temp             0.274    0.0229    11.9   0
## 3 originNATURAL   -0.161    0.213     -0.757 0.449
```

```
spmodel::glance(fit_exp)
```

```
## # A tibble: 1 x 10
##       n     p  npar value   AIC  AICc   BIC logLik deviance pseudo.r.squared
##   <int> <dbl> <int> <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>            <dbl>
## 1   102     3     3  261.  267.  267.  275.  -130.     99.0            0.564
```

```
spmodel::glance(fit_gau)
```

```
## # A tibble: 1 x 10
##       n     p  npar value   AIC  AICc   BIC logLik deviance pseudo.r.squared
##   <int> <dbl> <int> <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>            <dbl>
## 1   102     3     3  263.  269.  269.  277.  -131.     99.0            0.602
```

The spatial variance describes the magnitude of structured variation, the range controls the decay of spatial correlation, and the nugget represents microscale variation and/or measurement error.

The reported range parameter is not necessarily the practical correlation range. Its interpretation depends on the covariance function and parameterization.

```r
coef_compare <- data.frame(
  Term = names(coef(fit_lm)),
  OLS = as.numeric(coef(fit_lm)),
  Exponential = as.numeric(coef(fit_exp)),
  Gaussian = as.numeric(coef(fit_gau))
)

print(coef_compare, row.names = FALSE)
```

```
##           Term        OLS Exponential   Gaussian
##    (Intercept)  4.2374286  2.73445222  2.6259578
##           temp  0.1671092  0.26576803  0.2738290
##  originNATURAL -0.1855307 -0.06717235 -0.1609736
```

Discussion. Does accounting for spatial dependence substantially change the estimated temperature and origin effects? How might the estimated standard errors differ from those obtained under independence?

Because both spatial models use the same fixed-effect design, REML-based likelihood comparisons are appropriate for their covariance specifications. When comparing different mean formulas, ML should be used instead.

Kriging Prediction: Trend and Spatial Correction

The spatial linear model can be written as

$$Y(s) = \mathbf{x}(s)^\top \boldsymbol{\beta} + W(s) + \epsilon(s),$$

where $W(s)$ is a spatial process and $\epsilon(s)$ is the nugget.

For a new location s_0, the plug-in kriging predictor is

$$\widehat{Y}(s_0) = \underbrace{\mathbf{x}(s_0)^\top \widehat{\boldsymbol{\beta}}}_{\text{Regression trend}} + \underbrace{\mathbf{c}_0^\top \widehat{\Sigma}^{-1}(\mathbf{y} - X\widehat{\boldsymbol{\beta}})}_{\text{Spatial correction}}.$$

The regression trend represents the prediction based on temperature and lake origin alone. The spatial correction uses covariance-weighted residual information from observed locations.

For visualization, we use at most 600 prediction locations to reduce plotting time. The fitted spatial model itself is unchanged.

```r
set.seed(123)

if (nrow(lake_preds) > 600) {
  pred_id <- sort(sample(seq_len(nrow(lake_preds)), 600))
  lake_plot <- lake_preds[pred_id, ]
} else {
  lake_plot <- lake_preds
}

pred_plot_df <- sf::st_drop_geometry(lake_plot)
xy_plot <- sf::st_coordinates(lake_plot)[, 1:2, drop = FALSE]
xy_plot_km <- xy_plot / 1000
```

```r
# Request predictions and prediction intervals
pred_exp <- predict(
  fit_exp,
  newdata = lake_plot,
  se.fit = TRUE,
  interval = "prediction"
)

str(pred_exp, max.level = 1)
```

```
## List of 2
##  $ fit   : num [1:10, 1:3] 7.05 7.6 6.08 5.12 5.13 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ se.fit: Named num [1:10] 0.794 0.861 0.831 0.877 0.894 ...
##   ..- attr(*, "names")= chr [1:10] "1" "2" "3" "4" ...
```

With `se.fit = TRUE`, the prediction object is expected to contain a `fit` component and a `se.fit` component. When an interval is requested, `fit` contains fitted values and interval endpoints.

```r
stopifnot(
  is.list(pred_exp),
  all(c("fit", "se.fit") %in% names(pred_exp))
)

if (is.matrix(pred_exp$fit) ||
    is.data.frame(pred_exp$fit)) {
  pred_mean <- as.numeric(pred_exp$fit[, "fit"])
  pred_lower <- as.numeric(pred_exp$fit[, "lwr"])
  pred_upper <- as.numeric(pred_exp$fit[, "upr"])
} else {
  pred_mean <- as.numeric(pred_exp$fit)
  pred_lower <- NULL
  pred_upper <- NULL
}

pred_se <- as.numeric(pred_exp$se.fit)

stopifnot(
  length(pred_mean) == nrow(lake_plot),
  length(pred_se) == nrow(lake_plot)
)
```

Decomposition of the Kriging Predictor

We evaluate the fitted regression mean at each prediction location:

$$\widehat{m}(s_0) = \mathbf{x}(s_0)^\top \widehat{\boldsymbol{\beta}}.$$

The spatial correction is then

$$\widehat{W}(s_0) = \widehat{Y}(s_0) - \widehat{m}(s_0).$$

This decomposition separates the large-scale mean structure from residual spatial variation.

```
tt <- delete.response(terms(fit_lm))

X_pred <- model.matrix(
  tt,
  data = pred_plot_df,
  contrasts.arg = attr(
    model.matrix(fit_lm), "contrasts"
  ),
  xlev = fit_lm$xlevels
)

beta_hat <- coef(fit_exp)

X_pred <- X_pred[, names(beta_hat), drop = FALSE]

trend_pred <- as.numeric(X_pred %*% beta_hat)
spatial_correction <- pred_mean - trend_pred

stopifnot(isTRUE(all.equal(
  pred_mean,
  trend_pred + spatial_correction
)))
```

Visualizing the Kriging Decomposition

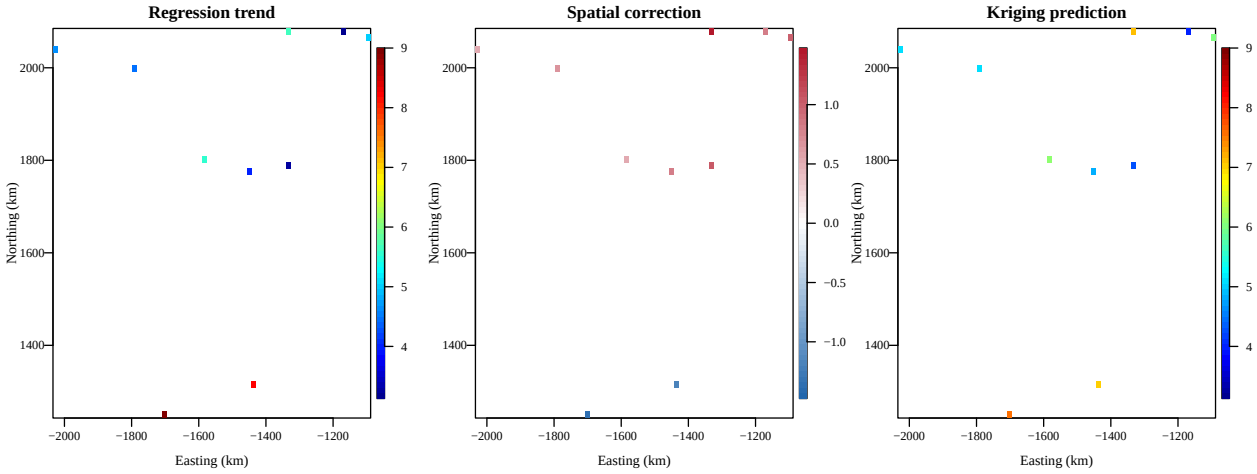
The following three panels illustrate how spatial prediction is constructed:

- The regression trend reflects the estimated effects of temperature and lake origin.
- The spatial correction adjusts this trend using observed residual patterns.
- The final kriging prediction combines both components.

The trend and total prediction share the same color scale, while the spatial correction uses a diverging scale centered at zero.

```
zlim_common <- range(
  c(trend_pred, pred_mean),
  finite = TRUE
)

cor_lim <- max(
  abs(spatial_correction),
  na.rm = TRUE
)

if (cor_lim == 0) cor_lim <- 1e-8

cor_colors <- colorRampPalette(
  c("#2166AC", "white", "#B2182B")
)(64)
```

```r
plot_style(mfrow = c(1, 3), mar = c(3.5, 3.5, 2, 1))

fields::quilt.plot(
  xy_plot_km, trend_pred,
  col = map_colors,
  zlim = zlim_common,
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Regression trend"
)

fields::quilt.plot(
  xy_plot_km, spatial_correction,
  col = cor_colors,
  zlim = c(-cor_lim, cor_lim),
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Spatial correction"
)

fields::quilt.plot(
  xy_plot_km, pred_mean,
  col = map_colors,
  zlim = zlim_common,
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Kriging prediction"
)
```




```r
par(mfrow = c(1, 1))
```

Interpretation. The regression trend captures systematic variation associated with the covariates. The spatial correction represents additional variation inferred from spatially correlated residuals.

Positive corrections indicate that the observations support higher conductivity than the regression trend predicts. Negative corrections indicate lower conductivity.

The correction depends on both the residual magnitudes and the fitted covariance structure.

Prediction Uncertainty

Kriging also quantifies uncertainty at unsampled locations.

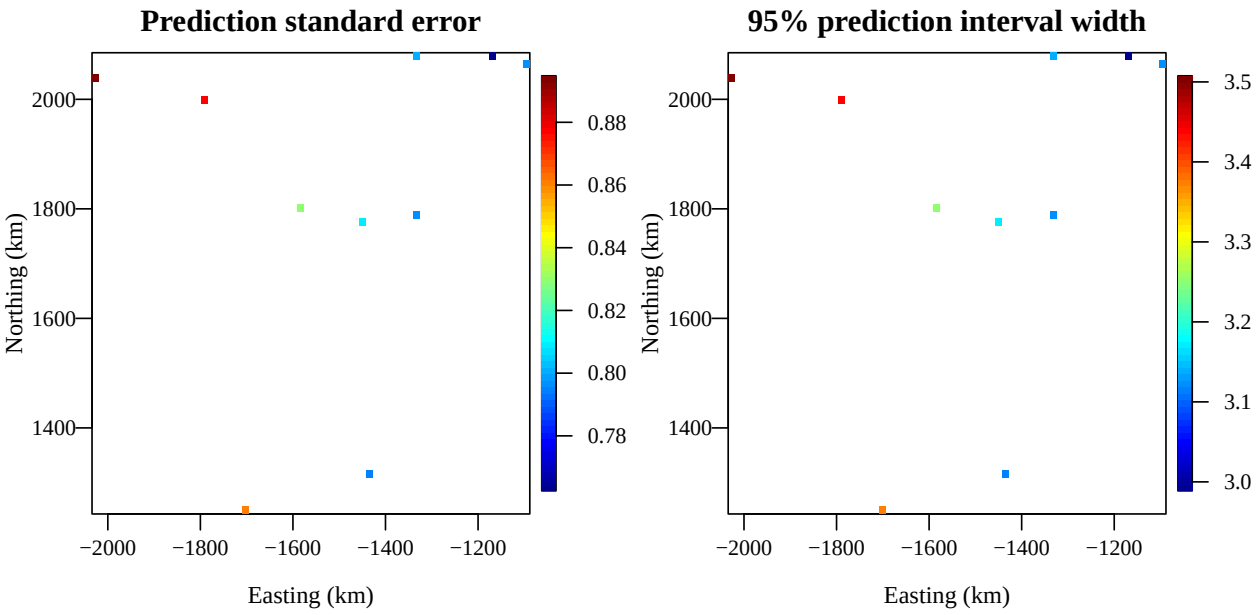
For a latent spatial process with known covariance parameters, the simple-kriging variance is

$$\mathrm{Var}\{W(s_0) \mid \mathbf{Y}\} = C(0) - \mathbf{c}_0^\top \Sigma^{-1} \mathbf{c}_0.$$

Universal kriging additionally accounts for uncertainty in estimating the regression coefficients. Prediction uncertainty for a new observed response may also include the nugget variance.

We map the standard error and prediction interval width returned by the fitted model.

```
if (!is.null(pred_lower) && !is.null(pred_upper)) {
  interval_width <- pred_upper - pred_lower
} else {
  interval_width <- 2 * 1.96 * pred_se
}

plot_style(mfrow = c(1, 2), mar = c(3.5, 3.5, 2, 1))

fields::quilt.plot(
  xy_plot_km, pred_se,
  col = map_colors,
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Prediction standard error"
)

fields::quilt.plot(
  xy_plot_km, interval_width,
  col = map_colors,
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "95% prediction interval width"
)
```




```
par(mfrow = c(1, 1))
```

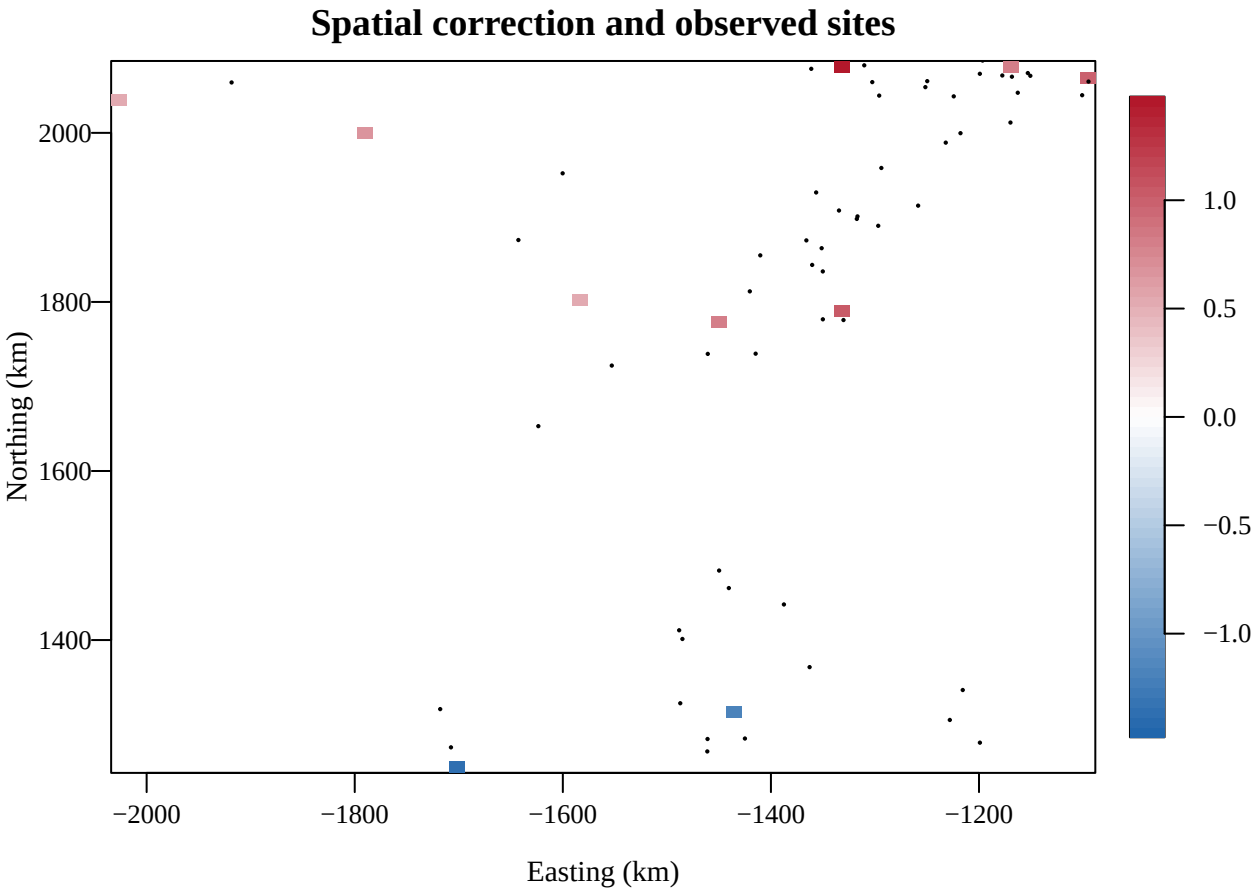
Interpretation. Prediction uncertainty depends on the configuration of sampling locations and the fitted covariance structure. Regions weakly correlated with observed sites may have larger uncertainty.

Importantly, a high predicted conductivity does not necessarily imply high prediction uncertainty.

Spatial Correction and Observed Locations

To visualize how observations inform kriging, we overlay sampling locations on the spatial correction map.

```
plot_style(mar = c(3.5, 3.8, 2, 1.2))

fields::quilt.plot(
  xy_plot_km, spatial_correction,
  col = cor_colors,
  zlim = c(-cor_lim, cor_lim),
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Spatial correction and observed sites"
)

points(xy_km, pch = 16, cex = 0.3)
```



Discussion. Do large positive or negative corrections occur near clusters of observations? Why should the spatial correction tend toward zero when a prediction location becomes weakly correlated with all observed sites?

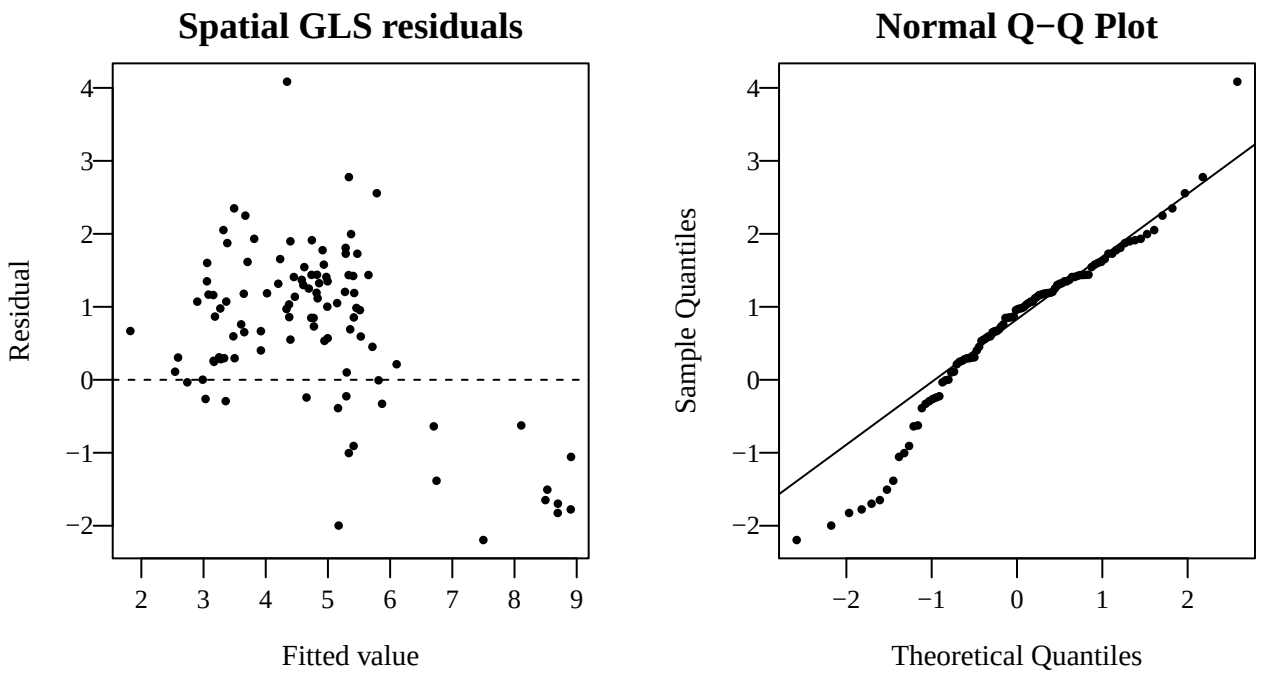
Spatial Model Diagnostics

Ordinary residuals from spatial GLS are generally correlated. Therefore, their empirical semivariogram does not necessarily become flat after fitting a valid spatial covariance model.

We first inspect the residuals for nonlinearity, heteroscedasticity, and unusual observations.

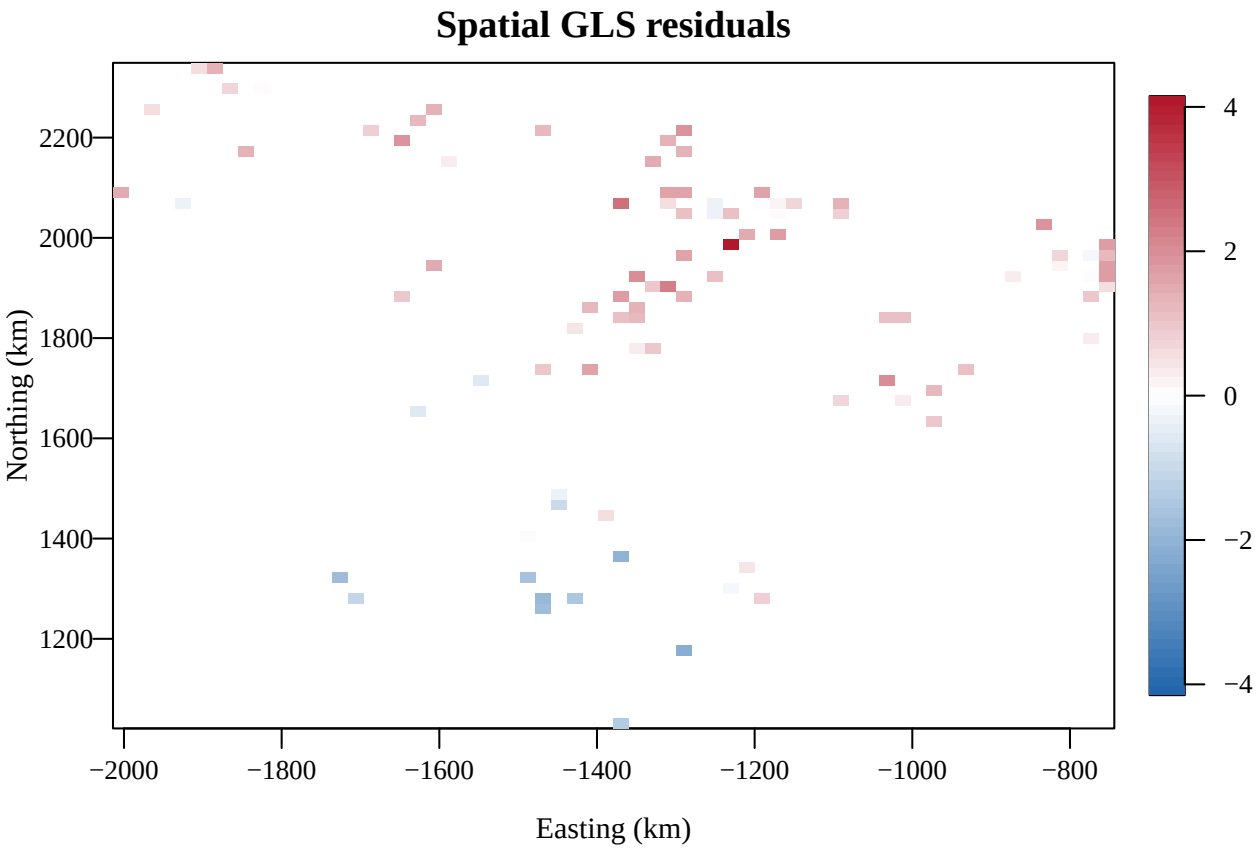
```
e_exp <- as.numeric(residuals(fit_exp))
f_exp <- as.numeric(fitted(fit_exp))

plot_style(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

plot(
  f_exp, e_exp,
  pch = 16, cex = 0.6,
  xlab = "Fitted value",
  ylab = "Residual",
  main = "Spatial GLS residuals"
)
abline(h = 0, lty = 2)
```

```r
qqnorm(e_exp, pch = 16, cex = 0.6)
qqline(e_exp)
```




```r
par(mfrow = c(1, 1))
```

```r
res_lim <- max(abs(e_exp), na.rm = TRUE)
if (res_lim == 0) res_lim <- 1e-8

plot_style(mar = c(3.5, 3.8, 2, 1.2))

fields::quilt.plot(
  xy_km, e_exp,
  col = cor_colors,
  zlim = c(-res_lim, res_lim),
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Spatial GLS residuals"
)
```



A more formal assessment of the covariance structure would use covariance-whitened residuals or simulation-based diagnostics. The ordinary residual map is primarily descriptive.

Out-of-Sample Prediction Assessment

The prediction maps illustrate the fitted model, but they do not establish whether spatial modeling improves predictive accuracy.

We therefore hold out 20% of the observed locations and compare OLS with exponential and Gaussian spatial models.

```
set.seed(123)

n <- nrow(lake)
train_id <- sample(seq_len(n), floor(0.8 * n))

lake_train <- lake[train_id, ]
lake_test <- lake[-train_id, ]

fit_lm_cv <- lm(
  mean_formula,
  data = sf::st_drop_geometry(lake_train)
)

fit_exp_cv <- spmodel::splm(
  mean_formula,
```

```
  data = lake_train,
  spcov_type = "exponential",
  estmethod = "reml"
)

fit_gau_cv <- spmodel::splm(
  mean_formula,
  data = lake_train,
  spcov_type = "gaussian",
  estmethod = "reml"
)

p_lm <- as.numeric(predict(
  fit_lm_cv,
  newdata = sf::st_drop_geometry(lake_test)
))

p_exp <- predict(
  fit_exp_cv,
  newdata = lake_test,
  se.fit = TRUE,
  interval = "prediction"
)

p_gau <- predict(
  fit_gau_cv,
  newdata = lake_test,
  se.fit = TRUE,
  interval = "prediction"
)
```

We extract the point predictions and their uncertainty using the same output structure as in the full-data analysis.

```
get_sp_pred <- function(object) {
  fit <- object$fit

  if (is.matrix(fit) || is.data.frame(fit)) {
    mean_pred <- as.numeric(fit[, "fit"])
    lower <- as.numeric(fit[, "lwr"])
    upper <- as.numeric(fit[, "upr"])
  } else {
    mean_pred <- as.numeric(fit)
    lower <- mean_pred - 1.96 * object$se.fit
    upper <- mean_pred + 1.96 * object$se.fit
  }

  list(
    mean = mean_pred,
    se = as.numeric(object$se.fit),
    lower = lower,
    upper = upper
  )
```

```
}

exp_cv <- get_sp_pred(p_exp)
gau_cv <- get_sp_pred(p_gau)

obs <- lake_test$log_cond

score <- function(obs, pred) {
  c(
    RMSE = sqrt(mean((obs - pred)^2)),
    MAE = mean(abs(obs - pred))
  )
}

scores <- rbind(
  OLS = score(obs, p_lm),
  Exponential = score(obs, exp_cv$mean),
  Gaussian = score(obs, gau_cv$mean)
)

round(scores, 3)
```

```
##              RMSE   MAE
## OLS         0.866 0.690
## Exponential 0.772 0.590
## Gaussian    0.780 0.573
```

We also assess the empirical coverage and average width of the 95% prediction intervals.

```
interval_score <- function(obs, lower, upper) {
  c(
    Coverage = mean(obs >= lower & obs <= upper),
    Mean_width = mean(upper - lower)
  )
}

interval_results <- rbind(
  Exponential = interval_score(
    obs, exp_cv$lower, exp_cv$upper
  ),
  Gaussian = interval_score(
    obs, gau_cv$lower, gau_cv$upper
  )
)

round(interval_results, 3)
```

```
##             Coverage Mean_width
## Exponential    0.952      3.183
## Gaussian       0.952      3.281
```

Interpretation. RMSE and MAE measure point prediction accuracy. Interval coverage and width assess predictive uncertainty.

A single random holdout split is illustrative rather than conclusive. Repeated random splits or spatially separated validation would provide a stronger evaluation.

Complementary Spatial Modeling with `fields`

The `fields` package provides alternative spatial-process and smoothing methods.

We fit a spatial process and a thin-plate spline to illustrate their different modeling approaches. These models use spatial coordinates only and therefore are not directly comparable with the covariate-adjusted `spmodel` fits.

```
fit_fields <- fields::spatialProcess(xy_km, z)
fit_tps <- fields::Tps(xy_km, z)

print(fit_fields)
```

```
## CALL:
## fields::spatialProcess(x = xy_km, y = z)
##
##  SUMMARY OF MODEL FIT:
##
##  Number of Observations:                    102
##  Degree of polynomial in fixed part:        1
##  Total number of parameters in fixed part:  3
##  sigma Process stan. dev:                   1.236
##  tau  Nugget stan. dev:                     0.1166
##  lambda   tau^2/sigma^2:                    0.008905
##  aRange parameter (in units of distance):   18.14
##  Approx.  degrees of freedom for curve      99.69
##     Standard Error of df estimate:          3.016
##  log Likelihood:                            -152.416627690494
##  log Likelihood REML:                       -170.02370367435
##
##  ESTIMATED COEFFICIENTS FOR FIXED PART:
##
##      estimate        SE    pValue
## d1  4.4710000 1.0850000 3.788e-05
## d2 -0.0012490 0.0005213 1.661e-02
## d3 -0.0003051 0.0005075 5.477e-01
##
##  COVARIANCE MODEL: stationary.cov
##   Covariance function:  Matern
##    Non-default covariance arguments and their values
## Covariance :
## [1] "Matern"
## smoothness :
## [1] 1
## aRange :
## [1] 18.13741
## onlyUpper :
## [1] FALSE
## distMat :
## [1] NA
```

```
## Nonzero entries in covariance matrix  10404
##
## SUMMARY FROM Max. Likelihood ESTIMATION:
## Parameters found from optim:
##       lambda       aRange
##  0.008904776 18.137409843
## Approx. confidence intervals for MLE(s)
##            lower95% upper95%
## lambda 1.475512e-05  5.37407
## aRange 1.323411e+01 24.85740
##
##  Note: MLEs for  tau and sigma found analytically from lambda
##
## Summary from estimation:
## lnProfileLike.FULL lnProfileREML.FULL        lnLike.FULL        lnREML.FULL
##      -1.524166e+02      -1.700237e+02                 NA                 NA
##             lambda                tau             sigma2             aRange
##       8.904776e-03       1.166236e-01       1.527390e+00       1.813741e+01
##             eff.df                GCV
##       9.969292e+01       4.443128e-01
```

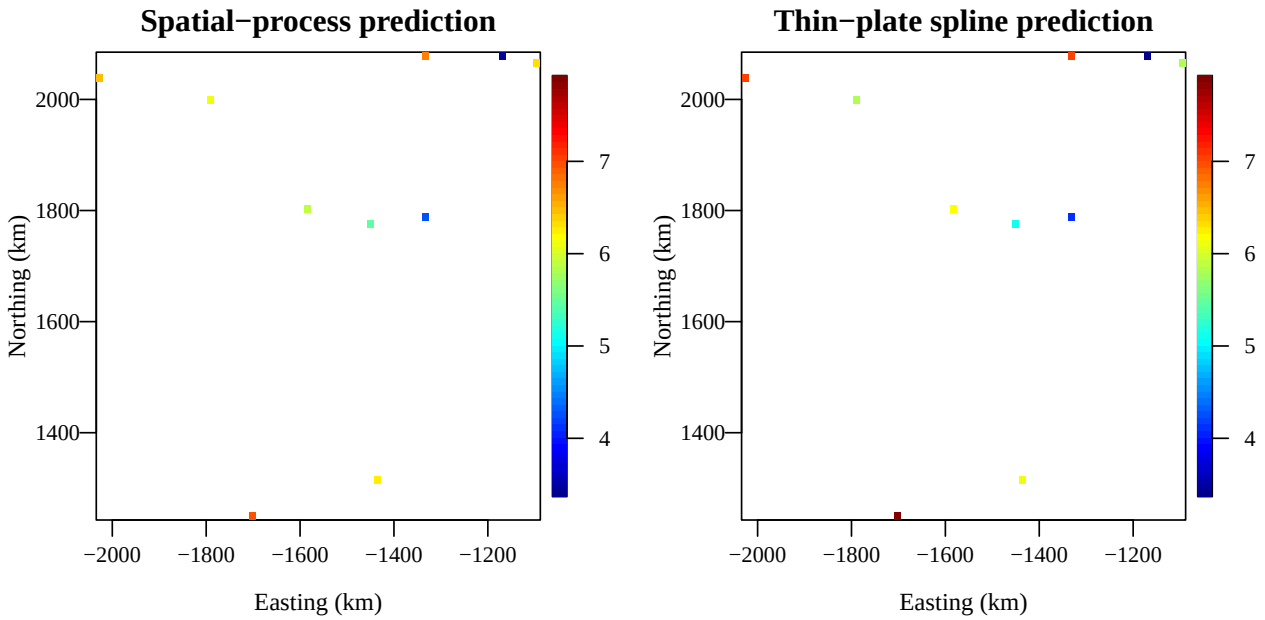
```
print(fit_tps)
```

```
## Call:
## fields::Tps(x = xy_km, Y = z)
##
##  Number of Observations:                102
##  Number of parameters in the null space 3
##  Parameters for fixed spatial drift     3
##  Model degrees of freedom:              62.3
##  Residual degrees of freedom:           39.7
##  GCV estimate for tau:                  0.6471
##  MLE for tau:                           0.563
##  MLE for sigma:                         7210
##  lambda                                 4.4e-05
##  User supplied sigma                    NA
##  User supplied tau^2                    NA
## Summary of estimates:
##                  lambda      trA      GCV    tauHat -lnLike Prof converge
## GCV        4.395858e-05 62.27831 1.075147 0.6470647     171.0681        1
## GCV.model            NA       NA       NA        NA           NA       NA
## GCV.one    4.395858e-05 62.27831 1.075147 0.6470647           NA        1
## RMSE                 NA       NA       NA        NA           NA       NA
## pure error           NA       NA       NA        NA           NA       NA
## REML       1.746985e-03 21.23250 1.403198 1.0540902     163.6306        3
```

The spatial-process model uses an explicit covariance structure. The thin-plate spline estimates a smooth surface through penalized fitting.

Both approaches can produce spatial predictions, but their parameters and uncertainty have different interpretations.

```r
p_fields <- as.numeric(
  predict(fit_fields, xy_plot_km)
)

p_tps <- as.numeric(
  predict(fit_tps, xy_plot_km)
)

zlim_fields <- range(
  c(p_fields, p_tps), finite = TRUE
)

plot_style(mfrow = c(1, 2), mar = c(3.5, 3.5, 2, 1))

fields::quilt.plot(
  xy_plot_km, p_fields,
  col = map_colors,
  zlim = zlim_fields,
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Spatial-process prediction"
)

fields::quilt.plot(
  xy_plot_km, p_tps,
  col = map_colors,
  zlim = zlim_fields,
  xlab = "Easting (km)",
  ylab = "Northing (km)",
  main = "Thin-plate spline prediction"
)
```



```r
par(mfrow = c(1, 1))
```

Discussion. A spatial-process model provides a probabilistic description of spatial variation, while a thin-plate spline emphasizes flexible surface estimation. Which method is preferable depends on whether the primary objective is regression inference, spatial prediction with model-based uncertainty, or smoothing.

Summary

This application illustrates the complete spatial modeling workflow:

1. Model the large-scale mean using scientific covariates.
2. Examine residual spatial dependence using an empirical semivariogram.
3. Estimate covariance parameters with spatial linear models.
4. Decompose kriging predictions into regression trend and spatial correction.
5. Map predictions and quantify spatially varying uncertainty.
6. Evaluate predictive performance at held-out locations.
7. Compare parametric spatial modeling with flexible smoothing.

The central lesson is that spatial modeling does more than produce a smooth map. It combines covariate information, spatial dependence, and uncertainty quantification to support inference and prediction at unobserved locations.

Final discussion: Does accounting for spatial dependence improve estimation of the temperature–conductivity relationship, prediction at unsampled locations, or both?

Applied Spatial Modeling: Rainfall in Paraná, Brazil (`geoR`)

We revisit the Paraná rainfall data from `geoR` to illustrate a complete spatial modeling workflow, from exploratory analysis and covariance estimation to kriging prediction and uncertainty quantification.

The data contain dry-season (May–June) rainfall measurements from 143 weather stations in Paraná State, Brazil.

Our objectives are to:

1. Identify large-scale spatial trends in rainfall.
2. Examine spatial dependence after removing the trend.
3. Estimate spatial covariance parameters using variogram fitting and maximum likelihood.
4. Decompose kriging predictions into trend and spatial correction.
5. Map prediction uncertainty and generate conditional simulations.
6. Assess prediction accuracy at held-out locations.

Data and Exploratory Analysis

We begin by examining the observed rainfall and the geographic distribution of weather stations.

The coordinates supplied with `parana` are projected spatial coordinates. We retain their original distance units throughout the analysis.

```r
library(geoR)
library(fields)

data(parana)

coords <- parana$coords
z <- as.numeric(parana$data)
border <- parana$borders

map_colors <- fields::tim.colors(64)

plot_style <- function(...) {
  par(las = 1, mgp = c(2.4, 0.7, 0), ...)
}

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"
```

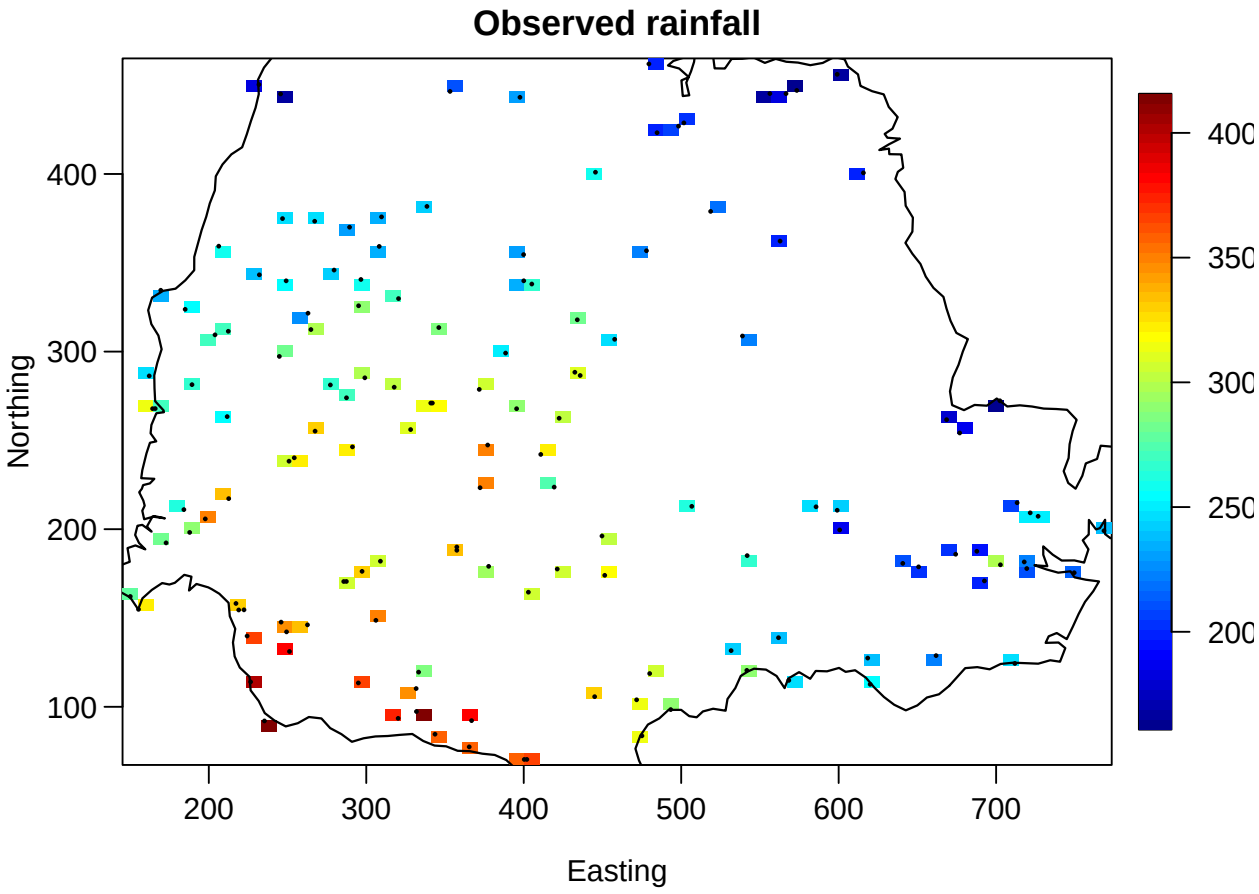
```r
summary(z)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   162.8   234.2   269.9   274.4   318.2   413.7
```

A spatial map helps identify broad geographic variation and possible local concentrations of high or low rainfall.

```r
plot_style(mar = c(3.5, 3.8, 2, 1))

fields::quilt.plot(
  coords, z,
```

```
  col = map_colors,
  xlab = "Easting",
  ylab = "Northing",
  main = "Observed rainfall",
  legend.mar = 3
)

lines(border, lwd = 1.2)
points(coords, pch = 16, cex = 0.35)
```



Interpretation. The map reveals spatial variation in rainfall, but this variation may reflect a large-scale trend, spatially correlated local fluctuations, or both.

Our next step is to separate these components.

Exploring the Large-Scale Spatial Trend

Consider the spatial regression model

$$Y(s) = m(s) + W(s) + \epsilon(s),$$

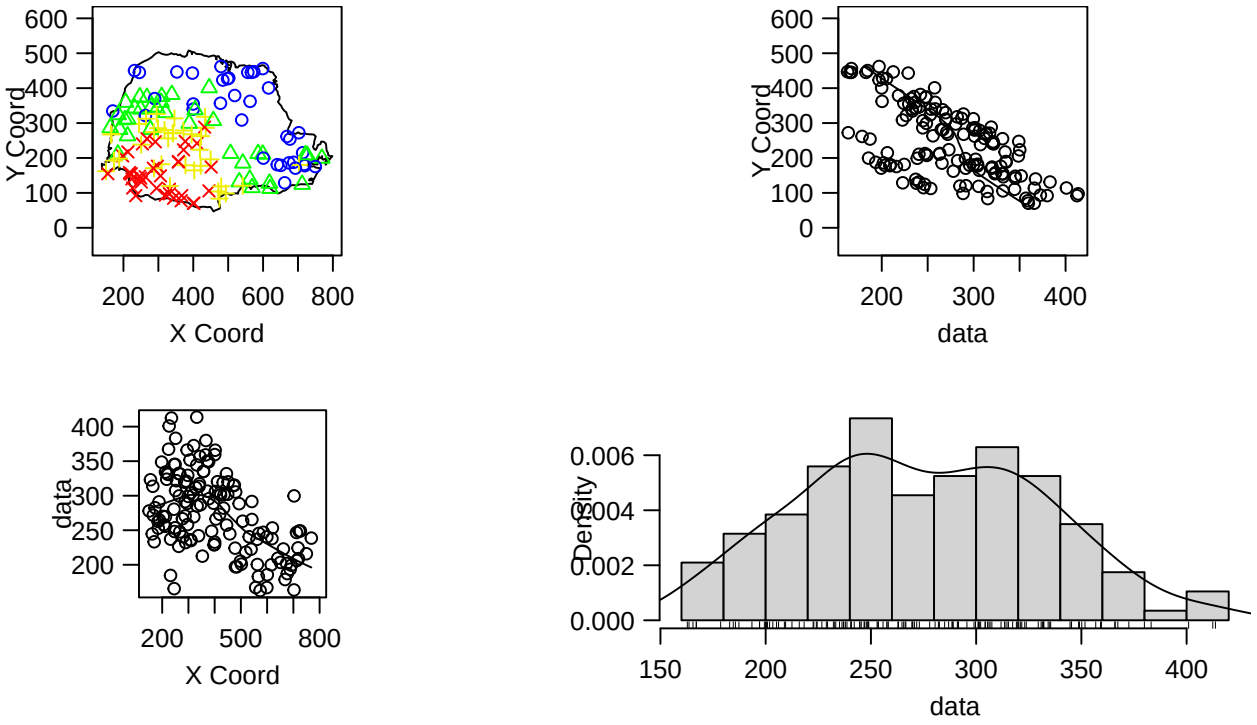
where $m(s)$ is the large-scale mean, $W(s)$ is a spatially correlated process, and $\epsilon(s)$ represents nugget variation.

A first-order spatial trend takes the form

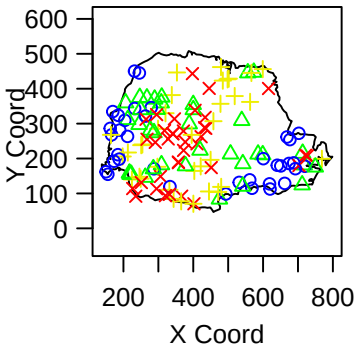
$$m(s) = \beta_0 + \beta_1 s_x + \beta_2 s_y.$$

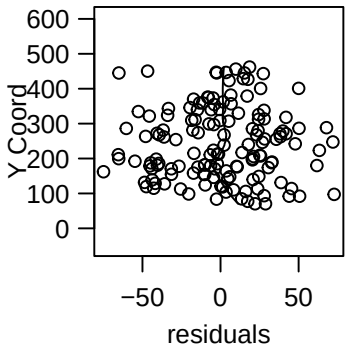
We first examine whether a linear trend provides a reasonable description of broad spatial variation.

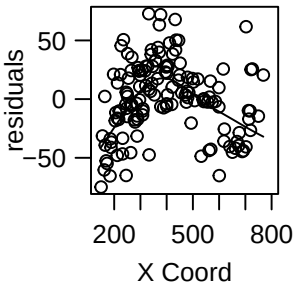
```
plot_style(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

plot(parana, lowess = TRUE)
```

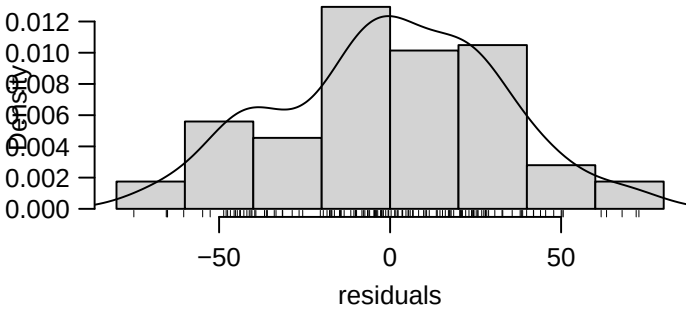



```
plot(parana, trend = "1st", lowess = TRUE)
```










```
par(mfrow = c(1, 1))
```

We can also estimate the trend directly using ordinary least squares.

```
rain_df <- data.frame(
  rainfall = z,
  east = coords[, 1],
  north = coords[, 2]
)

fit_lm <- lm(rainfall ~ east + north, data = rain_df)

summary(fit_lm)
```

```
##
## Call:
## lm(formula = rainfall ~ east + north, data = rain_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -74.926 -18.078   0.293  23.435  72.749
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 441.63635    9.70838   45.49   <2e-16 ***
## east         -0.19599    0.01620  -12.10   <2e-16 ***
## north        -0.36618    0.02566  -14.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 31.73 on 140 degrees of freedom
## Multiple R-squared:  0.6985, Adjusted R-squared:  0.6941
## F-statistic: 162.1 on 2 and 140 DF,  p-value: < 2.2e-16
```

Interpretation. The linear trend describes broad geographic changes in rainfall. However, the regression assumes independent errors and therefore does not account for residual spatial dependence.

The trend coefficients are useful for describing the mean structure, but conventional OLS standard errors may be misleading if the residuals are spatially correlated.

Empirical Semivariogram: Before and After Detrending

The empirical semivariogram measures the dissimilarity between observations separated by approximately distance h:

$$\widehat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{(i,j)\in N(h)} \{Y(s_i) - Y(s_j)\}^2.$$

A semivariogram computed from raw observations can reflect both large-scale trend and residual spatial dependence.

We therefore compare variograms before and after removing the first-order spatial trend.

We focus on shorter distances, where spatial dependence is usually most informative for kriging.

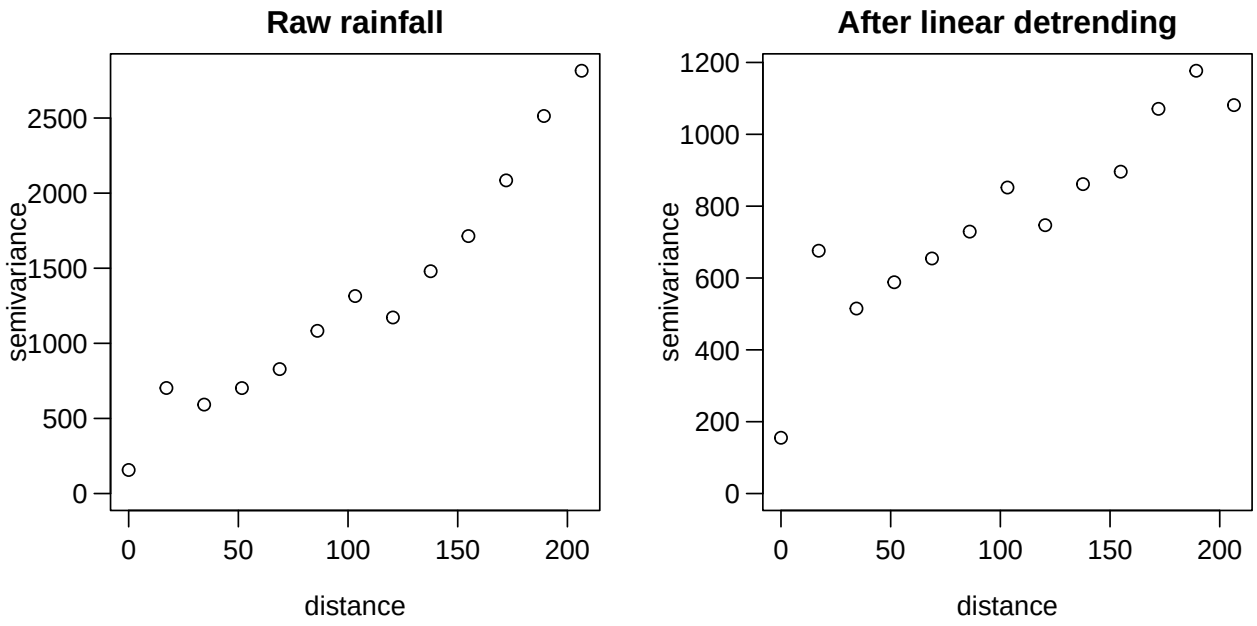
```
max_lag <- max(dist(coords)) / 3

vg_raw <- geoR::variog(
  parana,
  max.dist = max_lag,
  uvec = seq(0, max_lag, length.out = 13)
)
```

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

```
vg_trend <- geoR::variog(
  parana,
  trend = "1st",
  max.dist = max_lag,
  uvec = seq(0, max_lag, length.out = 13)
)
```

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

```
plot_style(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

plot(vg_raw, main = "Raw rainfall")
plot(vg_trend, main = "After linear detrending")
```




```
par(mfrow = c(1, 1))
```

Interpretation. Compare the magnitude and shape of the two semivariograms. If the raw variogram increases more strongly with distance, part of that increase may be attributable to the spatial trend.

Detrending is important because a nonconstant mean can distort the interpretation of the variogram.

Equal-Pair-Count Variogram Binning

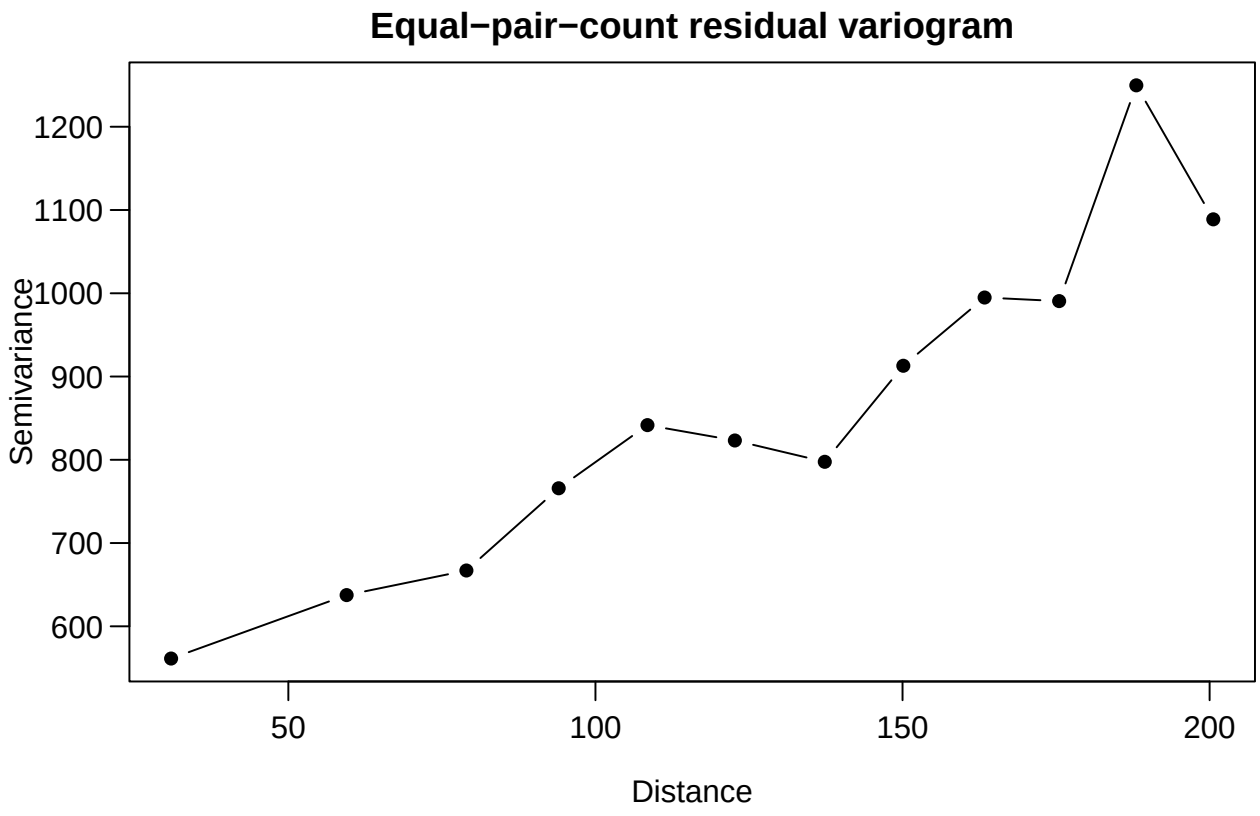
The conventional empirical variogram uses distance intervals. For irregularly distributed sampling locations, these intervals can contain very different numbers of pairs.

An alternative is to construct bins containing approximately equal numbers of location pairs.

This does not guarantee equal statistical precision, but it avoids bins with extremely small pair counts.

```
res_lm <- as.numeric(residuals(fit_lm))

D <- as.matrix(dist(coords))
idx <- which(upper.tri(D), arr.ind = TRUE)

h <- D[idx]
gamma_pair <- 0.5 *
  (res_lm[idx[, 1]] - res_lm[idx[, 2]])^2

keep <- is.finite(h) & h <= max_lag
h <- h[keep]
gamma_pair <- gamma_pair[keep]

# Equal-pair-count distance bins
breaks <- unique(as.numeric(quantile(
  h, probs = seq(0, 1, length.out = 13)
)))
```

```r
bin <- cut(h, breaks = breaks, include.lowest = TRUE)

vg_equal <- aggregate(
  cbind(distance = h, semivariance = gamma_pair),
  by = list(bin = bin),
  FUN = mean
)

vg_equal$n_pairs <- as.numeric(table(bin))

plot_style(mar = c(4, 4, 2, 1))

plot(
  vg_equal$distance,
  vg_equal$semivariance,
  type = "b", pch = 16,
  xlab = "Distance",
  ylab = "Semivariance",
  main = "Equal-pair-count residual variogram"
)
```




```r
print(vg_equal)
```

```
##            bin  distance semivariance n_pairs
## 1     [1,47.6]  30.90830     561.2874     365
## 2  (47.6,70.5]  59.48614     637.4941     365
## 3  (70.5,86.8]  78.99149     667.0167     364
```

```
## 4   (86.8,101]  94.01756     765.7929     365
## 5    (101,116] 108.46855     841.6396     364
## 6    (116,130] 122.69980     823.1897     365
## 7    (130,144] 137.33958     797.5816     364
## 8    (144,157] 150.12646     912.9271     365
## 9    (157,169] 163.36352     994.8958     364
## 10   (169,182] 175.50218     990.5843     364
## 11   (182,194] 188.06518    1249.7389     365
## 12   (194,206] 200.59748    1088.7999     365
```

Discussion. Equal-width bins offer a consistent distance resolution, whereas equal-pair-count bins adapt to the sampling geometry. Which approach better reveals the short-distance behavior of the semivariogram?

Because location pairs share observations, their contributions are dependent. Consequently, equal pair counts do not imply equal variogram uncertainty.

Directional Variograms and Isotropy

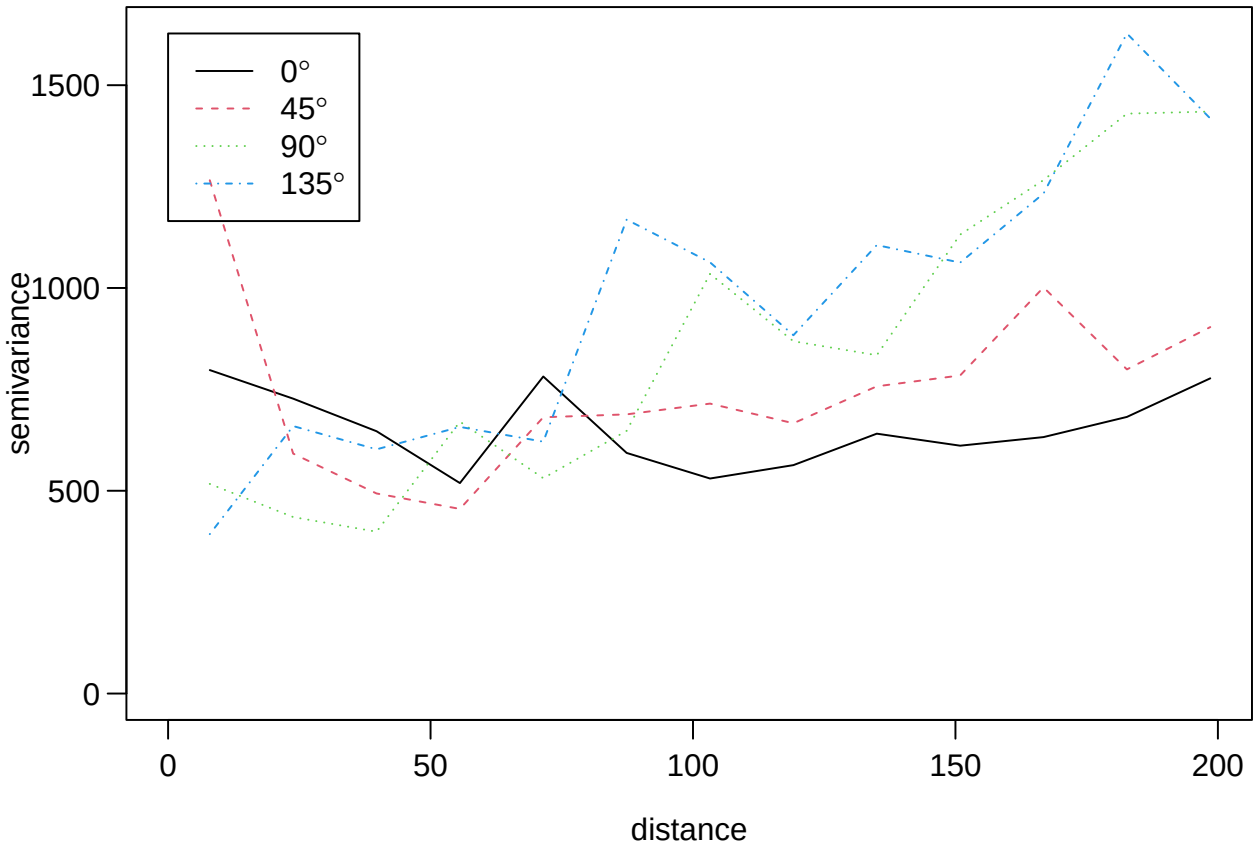
A stationary isotropic covariance model assumes that correlation depends only on separation distance, not direction.

We examine directional variograms to assess whether the observed spatial dependence appears similar across orientations.

```
vg_dir <- geoR::variog4(
  parana,
  trend = "1st",
  max.dist = max_lag
)
```

```
## 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_style(mar = c(4, 4, 2, 1))
plot(vg_dir)
```



Interpretation. Large systematic differences among directional variograms may suggest anisotropy. However, apparent differences can also arise from unequal sampling geometry, limited pairs, or remaining trend misspecification.

Directional variograms provide exploratory evidence rather than a definitive test of isotropy.

Variogram-Based Covariance Estimation

We now fit parametric covariance models to the detrended empirical variogram.

For a stationary spatial process,

$$\gamma(h) = \tau^2 + \sigma^2\{1 - \rho(h)\}, \qquad h > 0,$$

where τ^2 is the nugget variance, σ^2 is the spatial variance, and $\rho(h)$ is the spatial correlation function.

We compare an exponential covariance model with a Matérn model having smoothness parameter $\kappa = 1.5$.

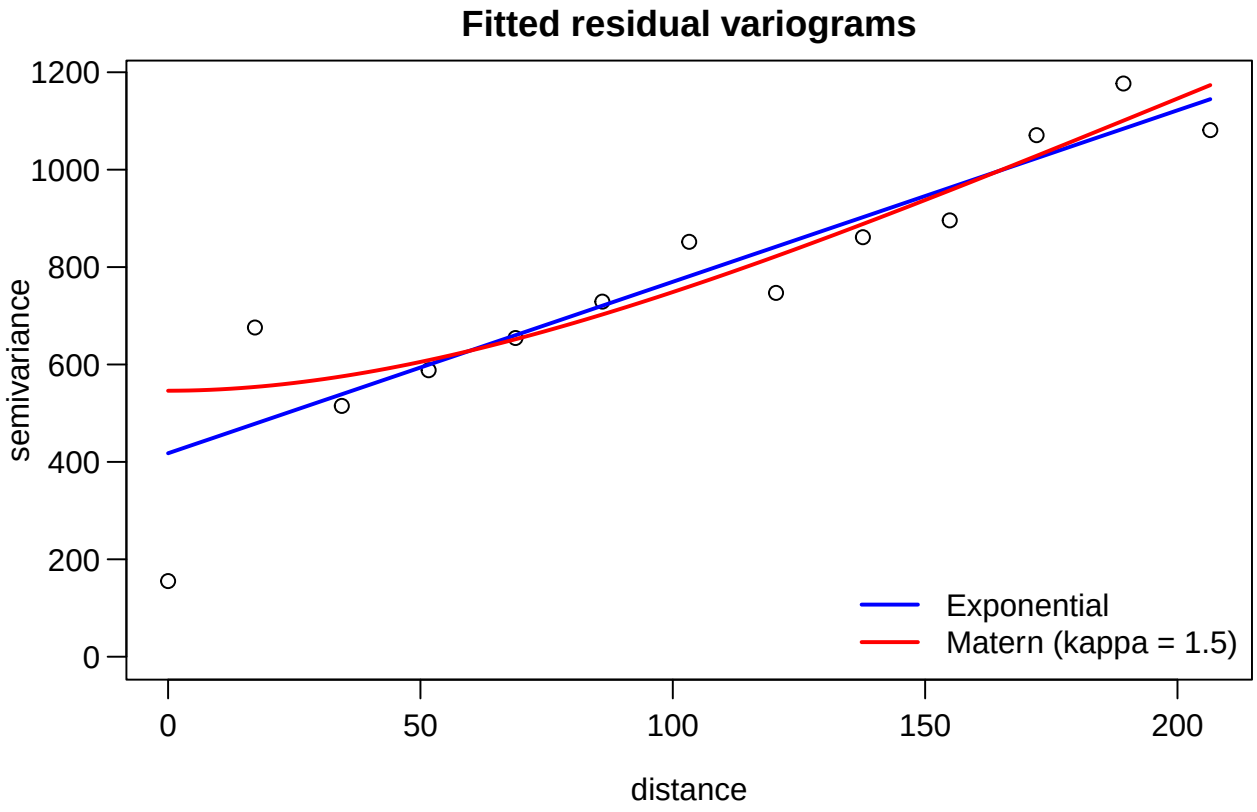
```
vg_fit_exp <- geoR::variofit(
  vg_trend,
  cov.model = "exponential",
  ini.cov.pars = c(1000, 50),
  nugget = 100,
  weights = "npairs"
)
```

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

```
vg_fit_mat <- geoR::variofit(
  vg_trend,
  cov.model = "matern",
  kappa = 1.5,
  fix.kappa = TRUE,
  ini.cov.pars = c(1000, 50),
  nugget = 100,
  weights = "npairs"
)
```

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

```
plot_style(mar = c(4, 4, 2, 1))
plot(vg_trend, main = "Fitted residual variograms")

lines(vg_fit_exp, col = "blue", lwd = 2)
lines(vg_fit_mat, col = "red", lwd = 2)

legend(
  "bottomright",
  legend = c("Exponential", "Matern (kappa = 1.5)"),
  col = c("blue", "red"),
  lty = 1, lwd = 2, bty = "n"
)
```



Interpretation. The fitted models differ primarily in their assumptions about spatial smoothness and the rate of correlation decay.

Variogram fitting is a useful exploratory estimation approach, but it does not use the full joint Gaussian likelihood. The results may depend on binning, weighting, and the selected distance range.

Maximum Likelihood Estimation

We now estimate the regression and covariance parameters jointly under a Gaussian spatial linear model:

$$\mathbf{Y} \sim N(X\boldsymbol{\beta}, \Sigma_\theta).$$

The log-likelihood is

$$\ell(\boldsymbol{\beta}, \theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_\theta| - \frac{1}{2}(\mathbf{y} - X\boldsymbol{\beta})^\top \Sigma_\theta^{-1}(\mathbf{y} - X\boldsymbol{\beta}).$$

We use the first-order spatial trend and compare exponential and Matérn covariance families.

```
fit_exp <- geoR::likfit(
  parana,
  trend = "1st",
  cov.model = "exponential",
  ini.cov.pars = c(1000, 50),
  nugget = 100,
  lik.method = "ML",
  messages = FALSE
```

```
)

fit_mat <- geoR::likfit(
  parana,
  trend = "1st",
  cov.model = "matern",
  kappa = 1.5,
  fix.kappa = TRUE,
  ini.cov.pars = c(1000, 50),
  nugget = 100,
  lik.method = "ML",
  messages = FALSE
)

print(fit_exp)
```

```
## 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
```

```
print(fit_mat)
```

```
## likfit: estimated model parameters:
##      beta0      beta1      beta2      tausq    sigmasq        phi
## "417.7055" " -0.1284" " -0.4139" "460.2353" "783.0931" " 86.7192"
## Practical Range with cor=0.05 for asymptotic range: 411.384
##
## likfit: maximised log-likelihood = -662.9
```

We summarize the estimated covariance parameters and compare likelihood-based criteria.

```
model_table <- data.frame(
  Model = c("Exponential", "Matern 1.5"),
  Spatial_variance = c(
    fit_exp$cov.pars[1],
    fit_mat$cov.pars[1]
  ),
  Range_parameter = c(
    fit_exp$cov.pars[2],
    fit_mat$cov.pars[2]
  ),
  Nugget = c(
    fit_exp$nugget,
    fit_mat$nugget
  ),
  LogLik = c(
    fit_exp$loglik,
    fit_mat$loglik
  )
```

```
)

model_table$AIC <- -2 * model_table$LogLik +
  2 * c(fit_exp$np, fit_mat$np)

print(round(
  model_table[, -1], 2
))
```

```
##   Spatial_variance Range_parameter Nugget  LogLik     AIC
## 1           785.69          184.39 385.52 -663.86 1339.72
## 2           783.09           86.72 460.24 -662.94 1337.87
```

Interpretation. Compare the fitted spatial variance, nugget, and range parameters. The range parameter controls the correlation decay but is not generally equal to the practical correlation range.

Likelihood comparisons are meaningful here because both models use the same observations and fixed-effect structure.

The likelihood-based model comparison complements, rather than replaces, examination of the empirical variogram and spatial prediction performance.

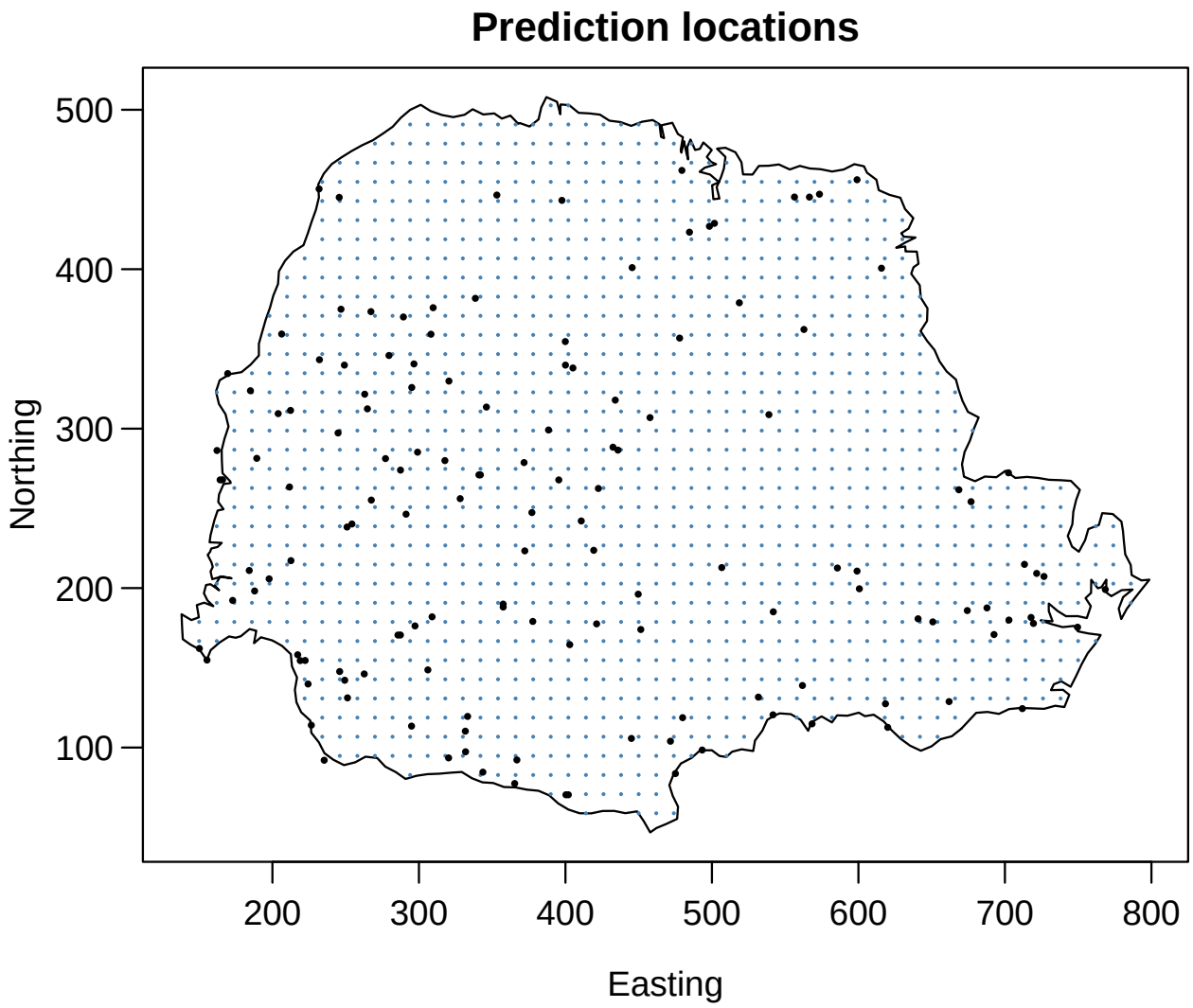
Constructing a Spatial Prediction Grid

We now predict rainfall across Paraná State.

We construct a regular grid and retain locations inside the state boundary. Using a moderate grid spacing keeps the computation suitable for a classroom R session.

```
pred_grid_all <- geoR::pred_grid(
  border, by = 12
)

pred_grid_in <- geoR::locations.inside(
  pred_grid_all, border
)

plot_style(mar = c(4, 4, 2, 1))

plot(
  border, type = "l",
  xlab = "Easting",
  ylab = "Northing",
  main = "Prediction locations"
)

points(
  pred_grid_in,
  pch = 16, col = "steelblue",
  cex = 0.25
)

points(
  coords,
```

```
  pch = 16, col = "black",
  cex = 0.45
)
```



Kriging Prediction: Trend and Spatial Correction

Under the fitted model,

$$Y(s) = m(s) + W(s) + \epsilon(s),$$

the kriging predictor at a new location s_0 can be written as

$$\widehat{Y}(s_0) = \underbrace{\mathbf{x}(s_0)^\top \widehat{\boldsymbol{\beta}}}_{\text{Regression trend}} + \underbrace{\mathbf{c}_0^\top \widehat{\Sigma}^{-1}(\mathbf{y} - X\widehat{\boldsymbol{\beta}})}_{\text{Spatial correction}}.$$

The trend captures broad geographic variation. The spatial correction uses the observed residuals and the estimated covariance structure to adjust predictions locally.

We use the exponential covariance model for the remaining illustrations.

```r
kc <- geoR::krige.control(
  obj.model = fit_exp,
  trend.d = "1st",
  trend.l = "1st"
)

krig <- geoR::krige.conv(
  geodata = parana,
  locations = as.matrix(pred_grid_in),
  krige = kc
)
```

```
## 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: Kriging performed using global neighbourhood
```

```r
pred_mean <- as.numeric(krig$predict)
pred_se <- sqrt(pmax(krig$krige.var, 0))

summary(pred_mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   163.9   213.3   245.0   255.7   298.8   388.7
```

```r
summary(pred_se)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   21.53   22.90   23.60   23.79   24.35   28.58
```

Decomposing the Kriging Prediction

The first-order trend is

$$\widehat{m}(s) = \widehat{\beta}_0 + \widehat{\beta}_1 s_x + \widehat{\beta}_2 s_y.$$

We evaluate this fitted mean at the prediction locations. The spatial correction is then the difference between the kriging prediction and the regression trend.

```r
beta_hat <- as.numeric(fit_exp$beta)

X_pred <- cbind(
  1,
  pred_grid_in[, 1],
  pred_grid_in[, 2]
)

trend_pred <- as.numeric(X_pred %*% beta_hat)
spatial_correction <- pred_mean - trend_pred

stopifnot(isTRUE(all.equal(
```

```
  pred_mean,
  trend_pred + spatial_correction
)))
```

The decomposition is exact for the computed predictor: the final kriging surface is the sum of the estimated trend and the spatial correction.

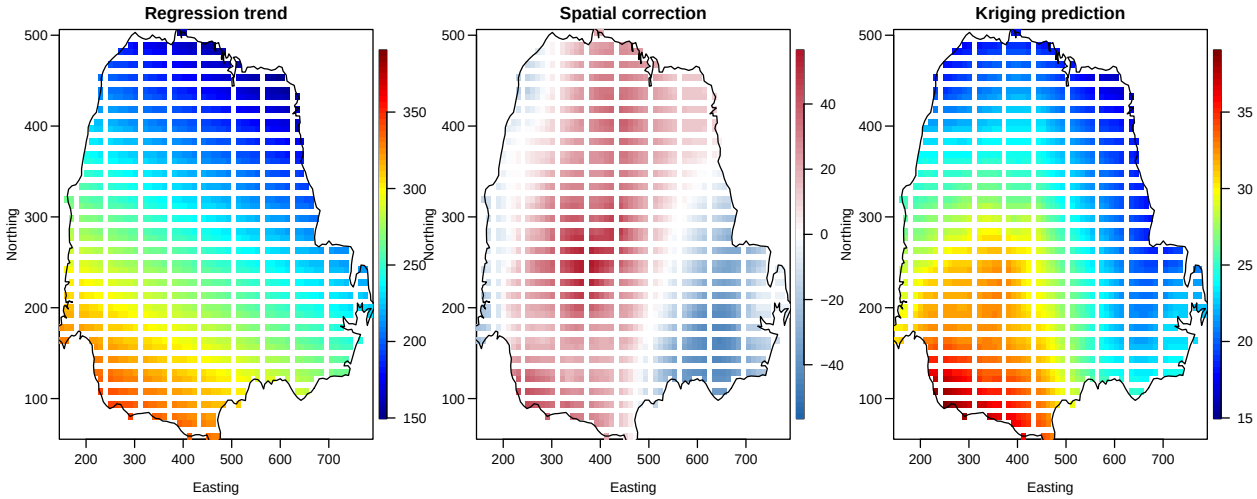
Visualizing Trend, Spatial Correction, and Kriging Prediction

The next figure is the central illustration of this application.

We display the regression trend, the spatial correction, and the final kriging prediction. The trend and final prediction share a common color scale; the correction uses a diverging scale centered at zero.

```
zlim_common <- range(
  c(trend_pred, pred_mean), finite = TRUE
)

cor_lim <- max(
  abs(spatial_correction), na.rm = TRUE
)

if (cor_lim == 0) cor_lim <- 1e-8

cor_colors <- colorRampPalette(
  c("#2166AC", "white", "#B2182B")
)(64)

plot_style(
  mfrow = c(1, 3),
  mar = c(3.5, 3.5, 2, 1)
)

fields::quilt.plot(
  pred_grid_in, trend_pred,
  col = map_colors,
  zlim = zlim_common,
  xlab = "Easting",
  ylab = "Northing",
  main = "Regression trend"
)
lines(border)

fields::quilt.plot(
  pred_grid_in, spatial_correction,
  col = cor_colors,
  zlim = c(-cor_lim, cor_lim),
  xlab = "Easting",
  ylab = "Northing",
  main = "Spatial correction"
)
lines(border)
```

```
fields::quilt.plot(
  pred_grid_in, pred_mean,
  col = map_colors,
  zlim = zlim_common,
  xlab = "Easting",
  ylab = "Northing",
  main = "Kriging prediction"
)
lines(border)
```




```
par(mfrow = c(1, 1))
```

Interpretation. The regression trend describes broad geographic variation in rainfall, whereas the spatial correction captures local departures from that trend.

Positive corrections indicate locations where nearby observations support higher rainfall than the trend predicts. Negative corrections indicate lower rainfall.

The final kriging surface combines both sources of information.

Kriging Prediction Uncertainty

Kriging also provides a prediction variance.

For a latent spatial process with known parameters, the simple-kriging conditional variance is

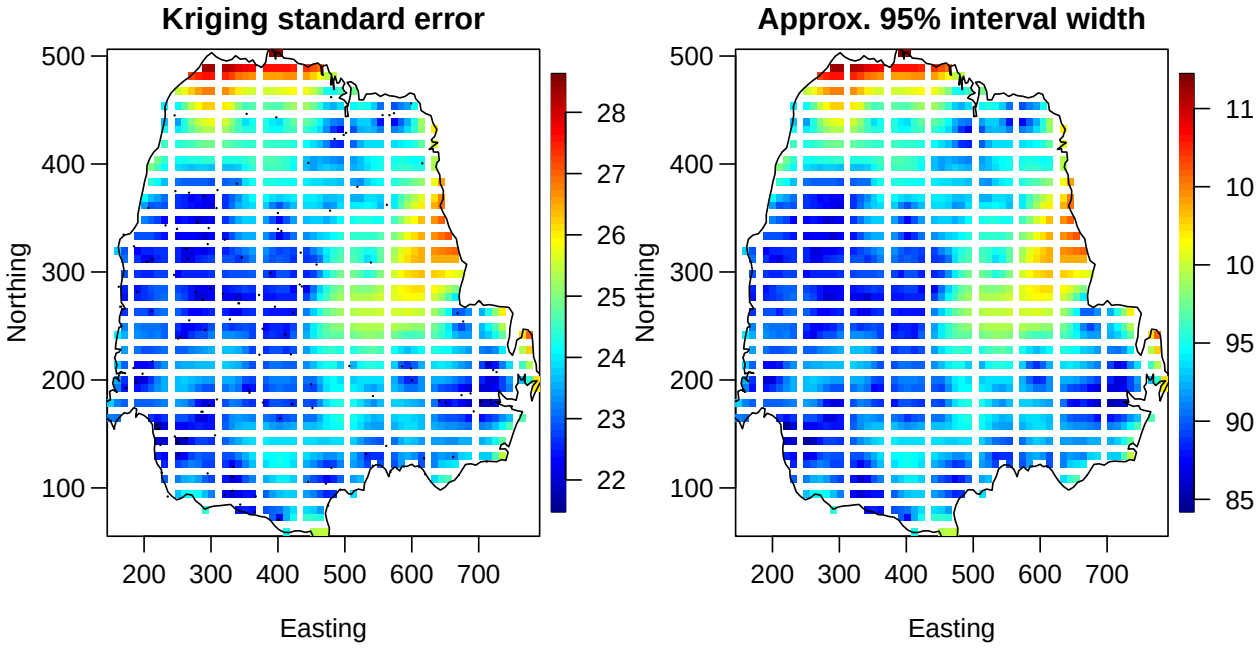
$$\mathrm{Var}\{W(s_0) \mid \mathbf{Y}\} = C(0) - \mathbf{c}_0^\top \Sigma^{-1} \mathbf{c}_0.$$

With an unknown regression mean, universal kriging includes an additional term reflecting estimation of the trend.

The `geoR` prediction variance is computed under the fitted covariance parameters and the specified kriging model. It does not fully incorporate uncertainty in estimated covariance parameters.

```
interval_width <- 2 * 1.96 * pred_se

plot_style(
```

```
  mfrow = c(1, 2),
  mar = c(3.5, 3.5, 2, 1)
)

fields::quilt.plot(
  pred_grid_in, pred_se,
  col = map_colors,
  xlab = "Easting",
  ylab = "Northing",
  main = "Kriging standard error"
)
lines(border)
points(coords, pch = 16, cex = 0.2)

fields::quilt.plot(
  pred_grid_in, interval_width,
  col = map_colors,
  xlab = "Easting",
  ylab = "Northing",
  main = "Approx. 95% interval width"
)
lines(border)
```




```
par(mfrow = c(1, 1))
```

Interpretation. Prediction uncertainty depends on the sampling geometry and the estimated spatial covariance.

Uncertainty is often smaller near observed locations and larger in sparsely sampled regions, although the exact pattern also depends on the trend model and nugget.

A location with high predicted rainfall does not necessarily have high prediction uncertainty.

The intervals shown are pointwise Gaussian plug-in intervals, not simultaneous confidence bands.

Conditional Simulations

A kriging prediction provides a conditional mean surface. It does not show the full range of plausible spatial rainfall patterns.

Conditional simulation generates realizations from the fitted spatial model given the observations.

These realizations preserve spatial variability that is smoothed out by the kriging predictor.

For computational efficiency, we generate only four conditional realizations.

```
set.seed(123)

oc <- geoR::output.control(
  simulations.predictive = TRUE,
  n.predictive = 4
)

krig_sim <- geoR::krige.conv(
  geodata = parana,
  locations = as.matrix(pred_grid_in),
  krige = kc,
  output = oc
)
```

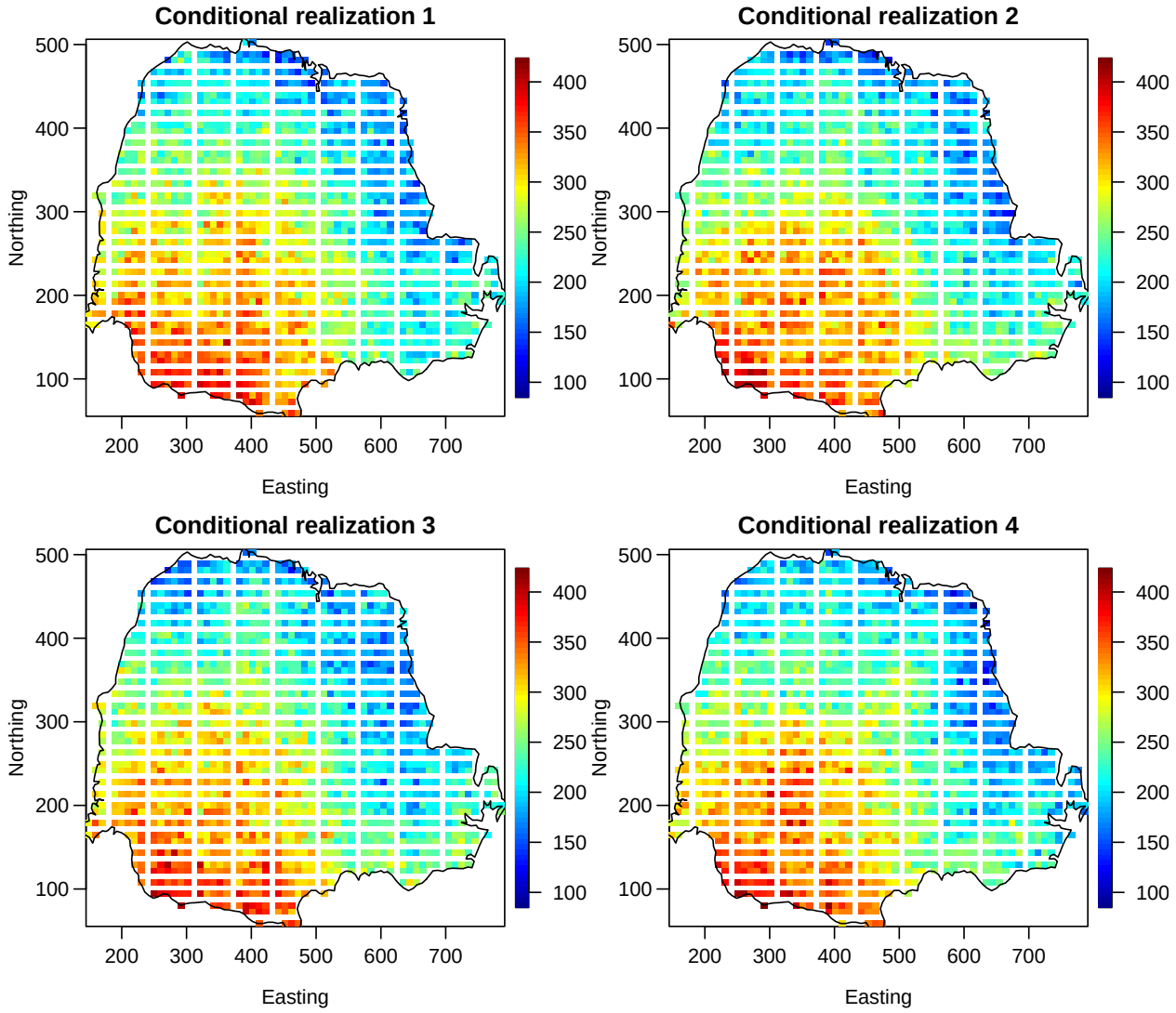
```
## krige.conv: results will be returned only for prediction locations inside the borders
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: sampling from the predictive distribution (conditional simulations)
## krige.conv: Kriging performed using global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

```
# Extract conditional simulations
sim <- krig_sim$simulations.predictive

# Check dimensions
dim(sim)
```

```
## NULL
```

```
sim <- krig_sim$simulation

if (is.null(dim(sim))) {
  sim <- matrix(sim, ncol = 1)
}

n_show <- min(4, ncol(sim))
sim_lim <- range(sim[, seq_len(n_show), drop = FALSE])

plot_style(
  mfrow = c(2, 2),
  mar = c(3.5, 3.5, 2, 1)
)

for (i in seq_len(n_show)) {
  fields::quilt.plot(
```

```
    pred_grid_in, sim[, i],
    col = map_colors,
    zlim = sim_lim,
    xlab = "Easting",
    ylab = "Northing",
    main = paste("Conditional realization", i)
  )
  lines(border)
}
```




```
par(mfrow = c(1, 1))
```

Interpretation. The conditional realizations share the same fitted mean and covariance model but display different plausible spatial patterns.

Unlike the kriging mean, conditional simulations retain spatial variability. They are useful when scientific questions concern nonlinear spatial functionals, such as maximum rainfall or the area exceeding a specified threshold.

Out-of-Sample Prediction Assessment

We now evaluate whether spatial modeling improves prediction at unobserved sites.

We randomly hold out 20% of the weather stations and fit the model using the remaining observations.

We compare:

- Linear regression with a first-order spatial trend.
- Universal kriging with the same trend and an exponential covariance.

Both methods use the same training observations and are evaluated at the same held-out locations.

```
set.seed(123)

# 80/20 train-test split
n <- length(z)
train_id <- sample(seq_len(n), size = floor(0.8 * n))
test_id <- setdiff(seq_len(n), train_id)

train_coords <- as.matrix(coords[train_id, , drop = FALSE])
test_coords <- as.matrix(coords[test_id, , drop = FALSE])

train_z <- z[train_id]
test_z <- z[test_id]

# Create geoR training data
train_geo <- geoR::as.geodata(
  data.frame(
    east = train_coords[, 1],
    north = train_coords[, 2],
    rainfall = train_z
  ),
  coords.col = 1:2,
  data.col = 3
)

# Nonspatial linear regression
train_df <- data.frame(
  rainfall = train_z,
  east = train_coords[, 1],
  north = train_coords[, 2]
)

test_df <- data.frame(
  east = test_coords[, 1],
  north = test_coords[, 2]
)

fit_lm_cv <- lm(
  rainfall ~ east + north,
  data = train_df
)
```

```r
pred_lm <- as.numeric(
  predict(fit_lm_cv, newdata = test_df)
)

# Fit spatial model using training data only
fit_geo_cv <- geoR::likfit(
  train_geo,
  trend = "1st",
  cov.model = "exponential",
  ini.cov.pars = fit_exp$cov.pars,
  nugget = fit_exp$nugget,
  lik.method = "ML",
  messages = FALSE
)

# Universal kriging
kc_cv <- geoR::krige.control(
  obj.model = fit_geo_cv,
  trend.d = "1st",
  trend.l = "1st"
)

pred_geo <- geoR::krige.conv(
  geodata = train_geo,
  locations = test_coords,
  krige = kc_cv
)
```

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```

```r
# Predictions and kriging standard errors
pred_krig <- as.numeric(pred_geo$predict)
se_krig <- sqrt(pmax(as.numeric(pred_geo$krige.var), 0))

# Compare predictive performance
rmse <- function(obs, pred) {
  sqrt(mean((obs - pred)^2))
}

mae <- function(obs, pred) {
  mean(abs(obs - pred))
}

results <- data.frame(
  Model = c("Linear regression", "Universal kriging"),
  RMSE = c(
    rmse(test_z, pred_lm),
    rmse(test_z, pred_krig)
  ),
  MAE = c(
    mae(test_z, pred_lm),
    mae(test_z, pred_krig)
```

```
  )
)

print(results)
```

```
##                Model     RMSE      MAE
## 1 Linear regression 29.54319 23.26965
## 2 Universal kriging 21.11818 17.65245
```

```
# Approximate 95% prediction interval coverage
lower <- pred_krig - 1.96 * se_krig
upper <- pred_krig + 1.96 * se_krig

coverage <- mean(test_z >= lower & test_z <= upper)

cat("Kriging 95% interval coverage:", coverage, "\n")
```

```
## Kriging 95% interval coverage: 0.9655172
```

The following metrics summarize prediction accuracy:

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \widehat{y}_i)^2},$$

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - \widehat{y}_i|.$$

```
score <- function(obs, pred) {
  c(
    RMSE = sqrt(mean((obs - pred)^2)),
    MAE = mean(abs(obs - pred))
  )
}

scores <- rbind(
  OLS = score(test_z, pred_lm),
  Kriging = score(test_z, pred_krig)
)

round(scores, 3)
```

```
##           RMSE    MAE
## OLS     29.543 23.270
## Kriging 21.118 17.652
```

We also examine approximate 95% prediction interval coverage and average interval width for the kriging model.

```r
lower <- pred_krig - 1.96 * se_krig
upper <- pred_krig + 1.96 * se_krig

coverage <- mean(
  test_z >= lower & test_z <= upper
)

mean_width <- mean(upper - lower)

data.frame(
  Coverage = coverage,
  Mean_interval_width = mean_width
)
```

```
##    Coverage Mean_interval_width
## 1 0.9655172            97.23715
```

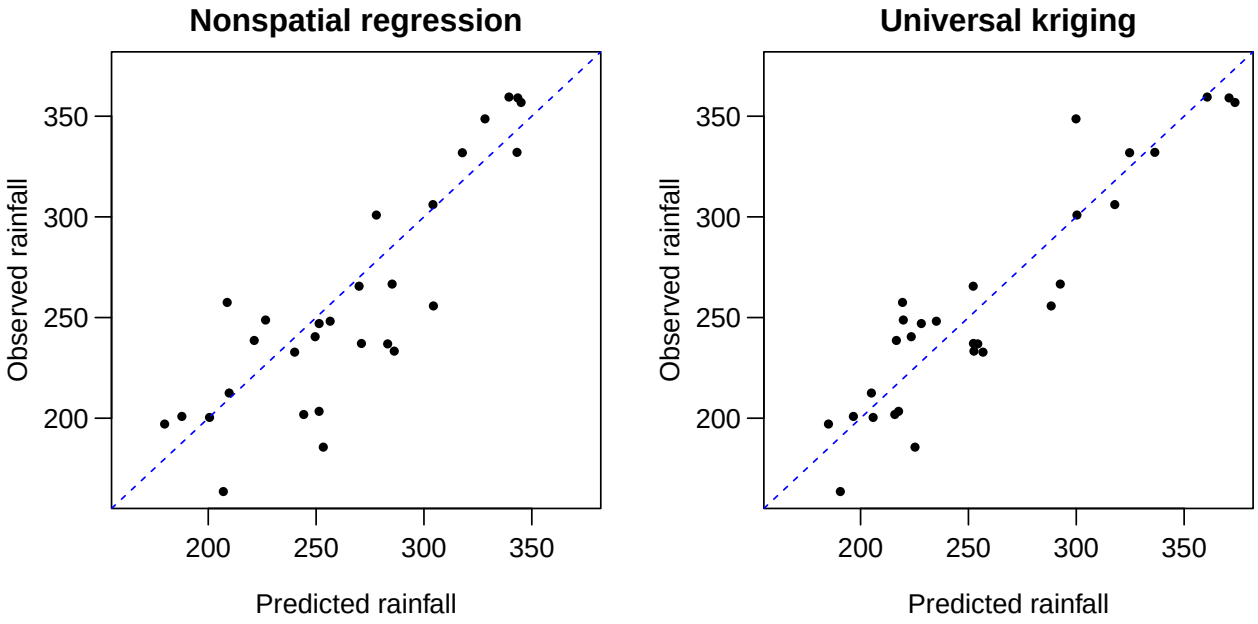
Interpretation. Lower RMSE and MAE indicate better point prediction. Coverage measures how often held-out observations fall within the predicted intervals, while interval width measures their precision.

A single random split is illustrative rather than conclusive. Repeated splits or spatially separated validation would provide a stronger assessment of predictive performance.

Observed Versus Predicted Rainfall

A final scatterplot provides a visual comparison of predictive performance.

```r
plot_style(mfrow = c(1, 2), mar = c(4, 4, 2, 1))

lims <- range(c(test_z, pred_lm, pred_krig))

plot(
  pred_lm, test_z,
  pch = 16, cex = 0.75,
  xlim = lims, ylim = lims,
  xlab = "Predicted rainfall",
  ylab = "Observed rainfall",
  main = "Nonspatial regression"
)
abline(0, 1, col = "blue", lty = 2)

plot(
  pred_krig, test_z,
  pch = 16, cex = 0.75,
  xlim = lims, ylim = lims,
  xlab = "Predicted rainfall",
  ylab = "Observed rainfall",
  main = "Universal kriging"
)
abline(0, 1, col = "blue", lty = 2)
```




```
par(mfrow = c(1, 1))
```

Summary and Discussion

This application illustrates how spatial modeling progresses from descriptive analysis to probabilistic prediction.

Key lessons

1. A spatial trend and residual spatial dependence represent different components of variation.
2. Detrending is important when interpreting empirical variograms.
3. Variogram fitting and maximum likelihood provide complementary approaches to covariance estimation.
4. Universal kriging combines a regression trend with a spatial correction.
5. Kriging uncertainty depends on both the fitted covariance and sampling geometry.
6. Conditional simulations represent plausible spatial realizations rather than only the conditional mean.
7. Out-of-sample assessment is necessary to evaluate predictive performance.

Final discussion question

Suppose our scientific interest is the probability that rainfall exceeds a critical threshold over a large region.

Would the kriging mean surface alone be sufficient?

Why might conditional simulations be more appropriate for estimating the distribution of regional exceedance area or maximum rainfall?