Spatial Statistics CW

CID: 02044064

March 17, 2025

Data: Project 7
Variable: Rainfall

Q1-3

The predictions at Imperial and UCL are 68.28501 and 66.89304 respectively. The corresponding prediction errors are 439.6416 and 564.8904.

Q4

The prediction error is greater at UCL than at Imperial because UCL is further away from its nearest weather station. Greater covariance means lower prediction error, and covariance decays with distance, so we expect prediction error to increase with distance.

Q5

After changing $\beta = 1.5$ to 2 in my code, my prediction for Imperial becomes 69.01248. The prediction error is 681.9502.

Q6

The prediction error is higher when $\beta = 2$ than when $\beta = 1.5$. This makes sense because the covariance kernel is $\rho(s,t) = \sigma^2 \exp(-\beta||t-s||^2)$ so larger β means that covariance decays faster for the same distance, and lower covariance means greater prediction errors.

Q7

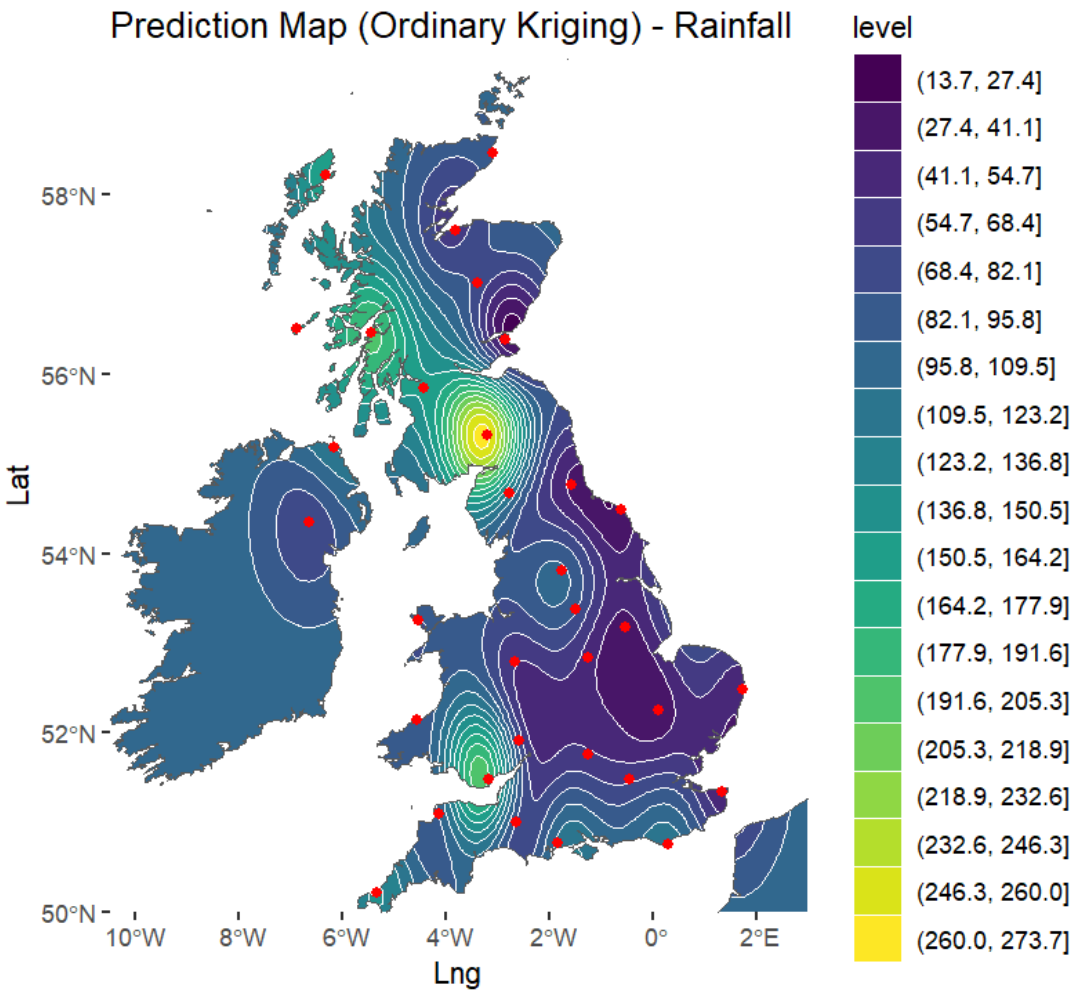
Using ordinary Kriging, the prediction for Imperial is 68.19851. The prediction error is 439.7467.

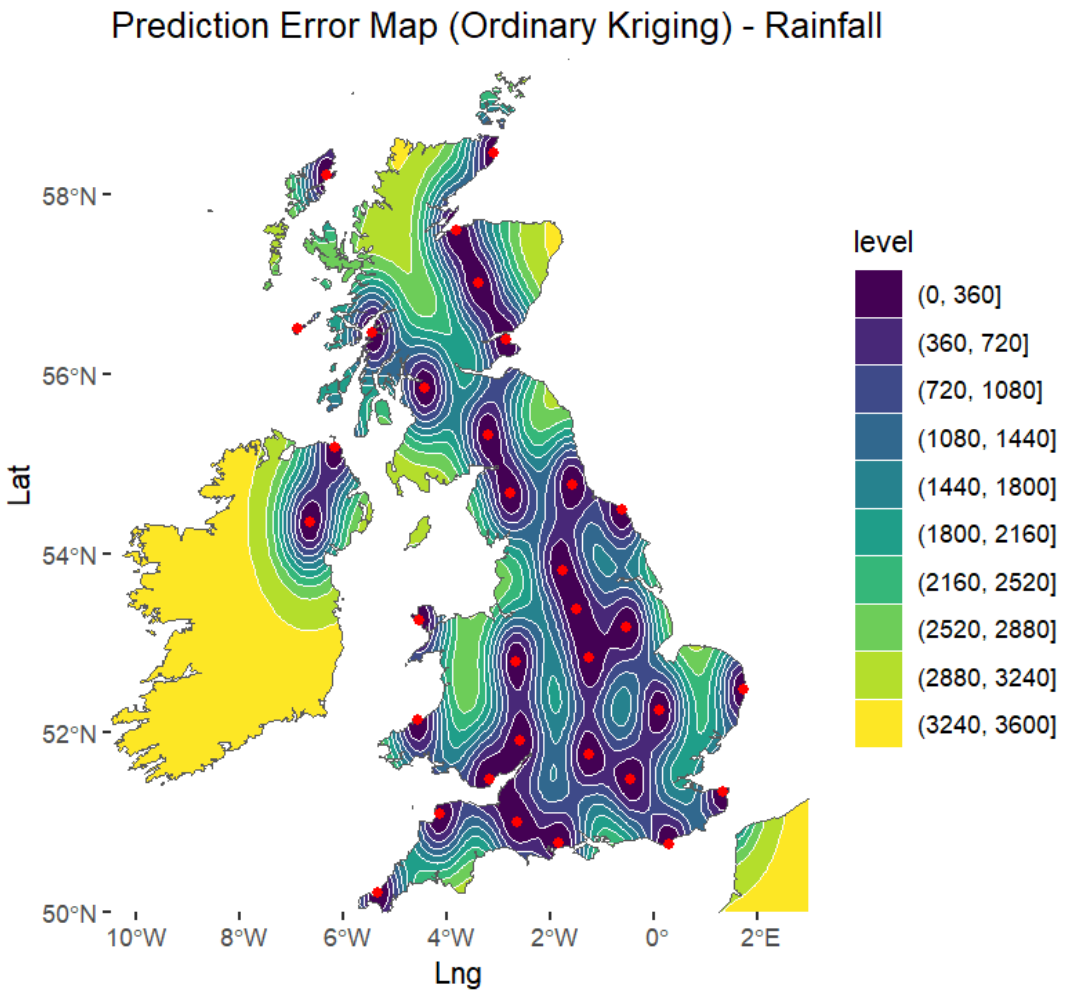
Q8

Ordinary Kriging results in a greater prediction error as we assume the mean is unknown, so there is more uncertainty than simple Kriging, where the mean function is assumed to be known.

Q9

My code is based on lab 1b Q3.


Prediction Map (Ordinary Kriging) - Rainfall
level
(13.7, 27.4]
(27.4, 41.1]
(41.1, 54.7]
(54.7, 68.4]
(68.4, 82.1]
(82.1, 95.8]
(95.8, 109.5]
(109.5, 123.2]
(123.2, 136.8]
(136.8, 150.5]
(150.5, 164.2]
(164.2, 177.9]
(177.9, 191.6]
(191.6, 205.3]
(205.3, 218.9]
(218.9, 232.6]
(232.6, 246.3]
(246.3, 260.0]
(260.0, 273.7]
Lat
58°N
56°N
54°N
52°N
50°N
10°W
8°W
6°W
4°W
2°W
0°
2°E
Lng



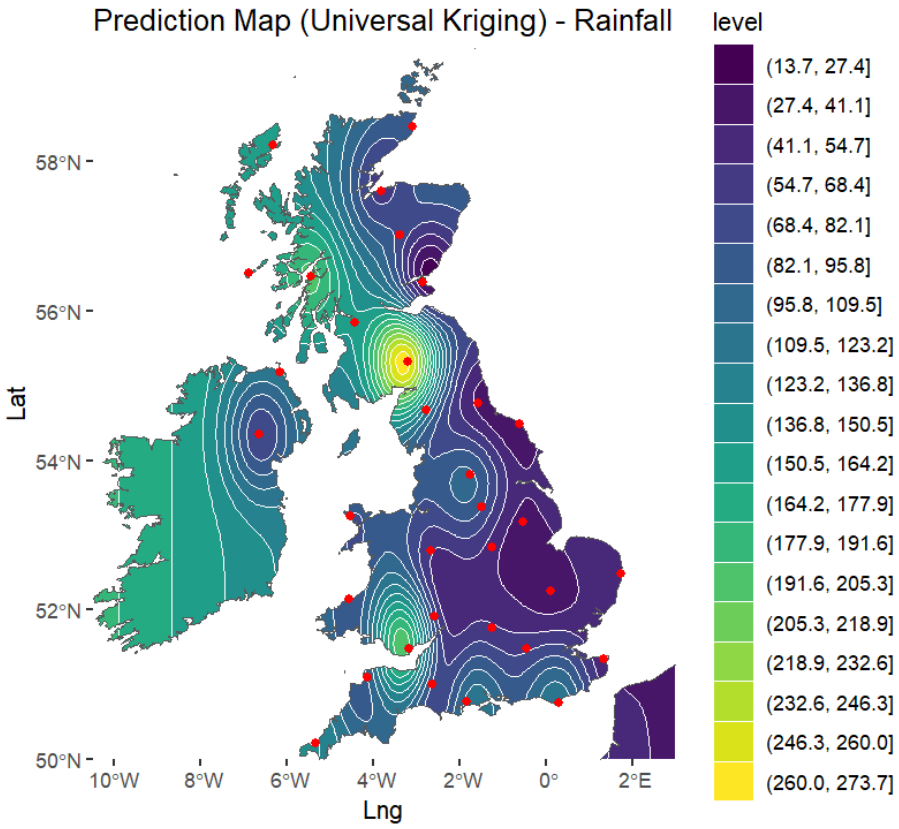
Q10

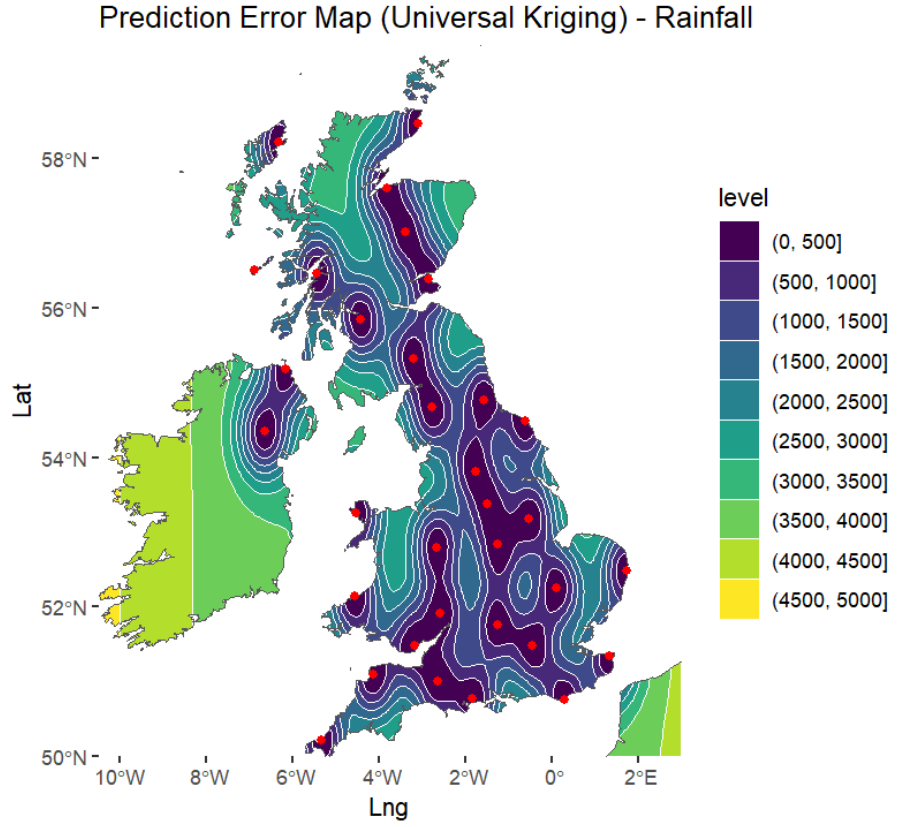
(a)

Using universal Kriging, the prediction for Imperial is 69.04911. The prediction error is 439.8867.

(b)

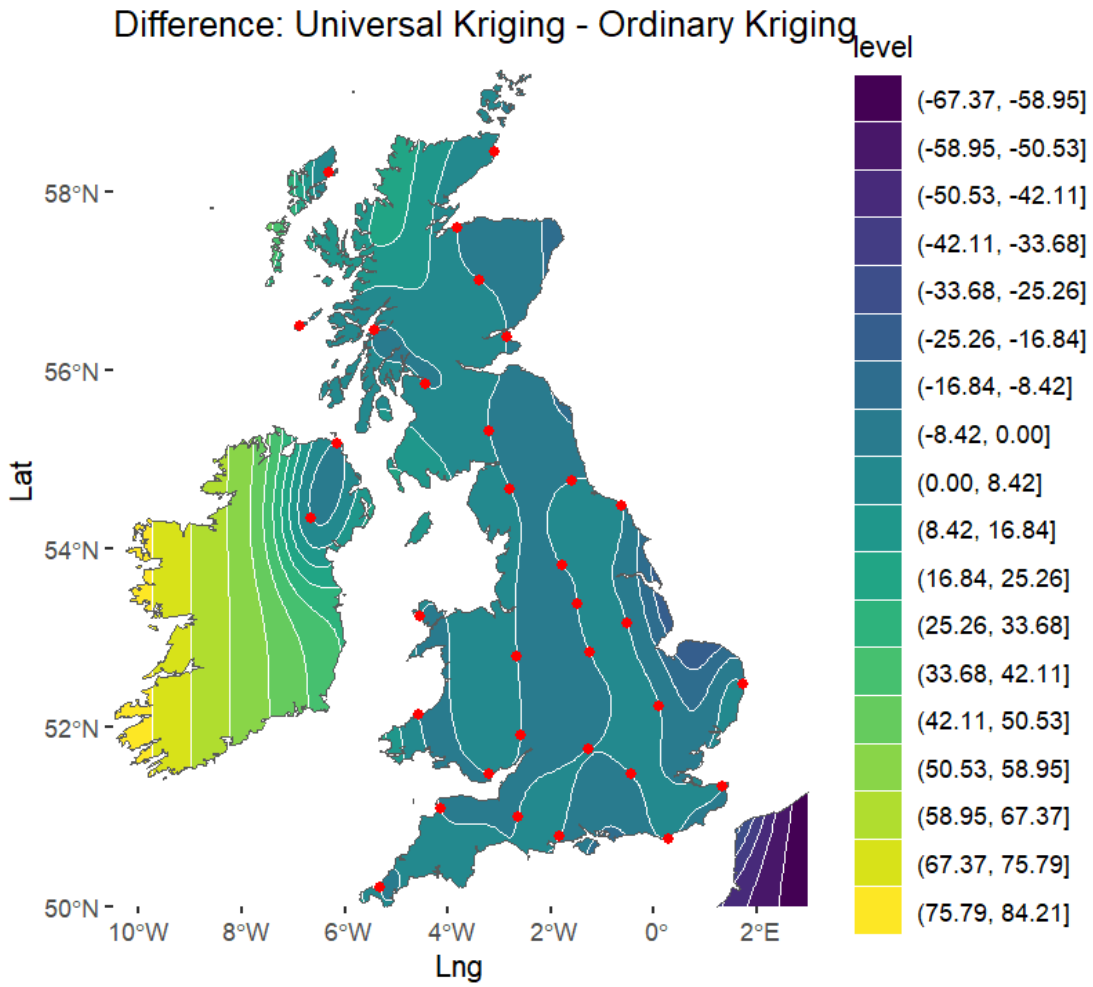
The prediction and prediction error maps for universal Kriging are shown on the next two pages.


Prediction Map (Universal Kriging) - Rainfall
level
(13.7, 27.4]
(27.4, 41.1]
(41.1, 54.7]
(54.7, 68.4]
(68.4, 82.1]
(82.1, 95.8]
(95.8, 109.5]
(109.5, 123.2]
(123.2, 136.8]
(136.8, 150.5]
(150.5, 164.2]
(164.2, 177.9]
(177.9, 191.6]
(191.6, 205.3]
(205.3, 218.9]
(218.9, 232.6]
(232.6, 246.3]
(246.3, 260.0]
(260.0, 273.7]
58°N
56°N
54°N
52°N
50°N
Lat
10°W
8°W
6°W
4°W
2°W
0°
2°E
Lng


Prediction Error Map (Universal Kriging) - Rainfall
level
(0, 500]
(500, 1000]
(1000, 1500]
(1500, 2000]
(2000, 2500]
(2500, 3000]
(3000, 3500]
(3500, 4000]
(4000, 4500]
(4500, 5000]
58°N
56°N
54°N
52°N
50°N
Lat
10°W
8°W
6°W
4°W
2°W
0°
2°E
Lng

(c)

Map of difference between ordinary and universal Kriging predictions:

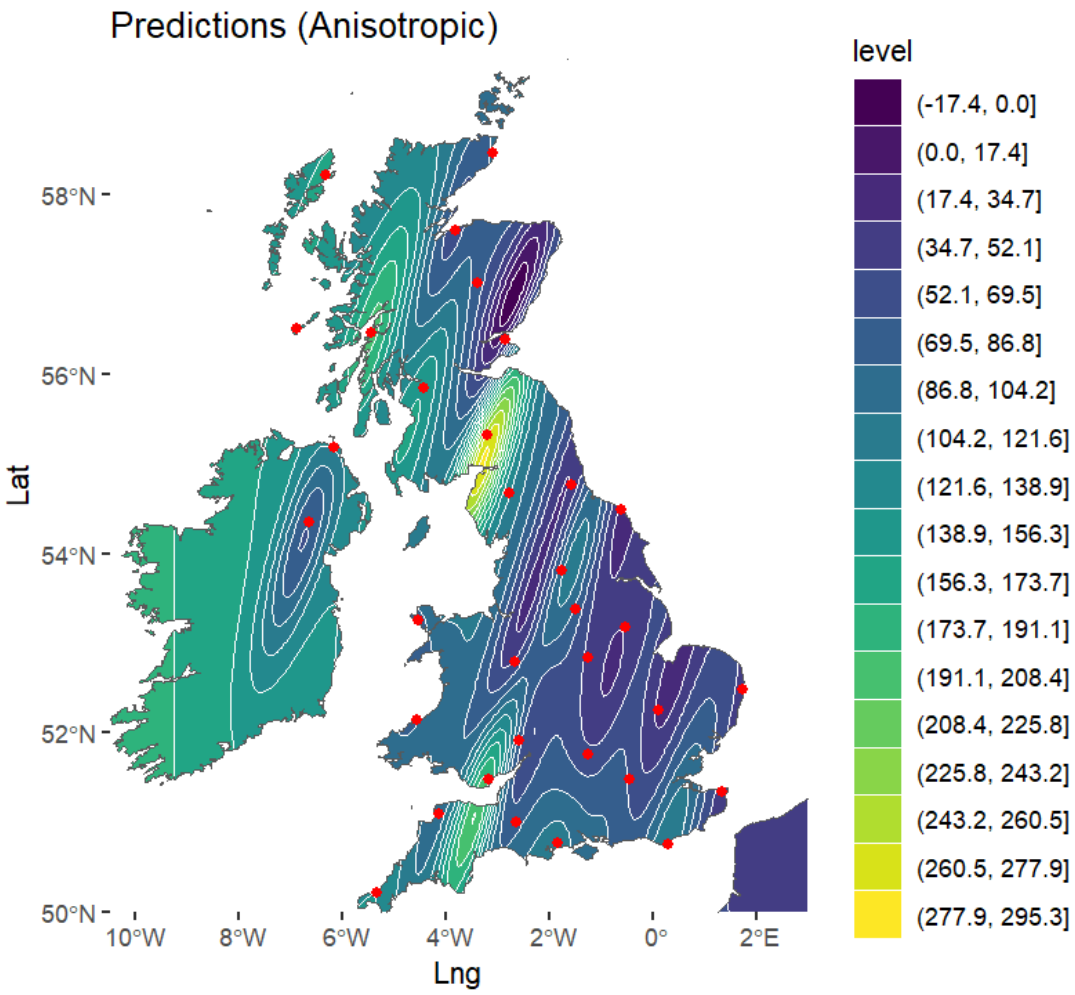


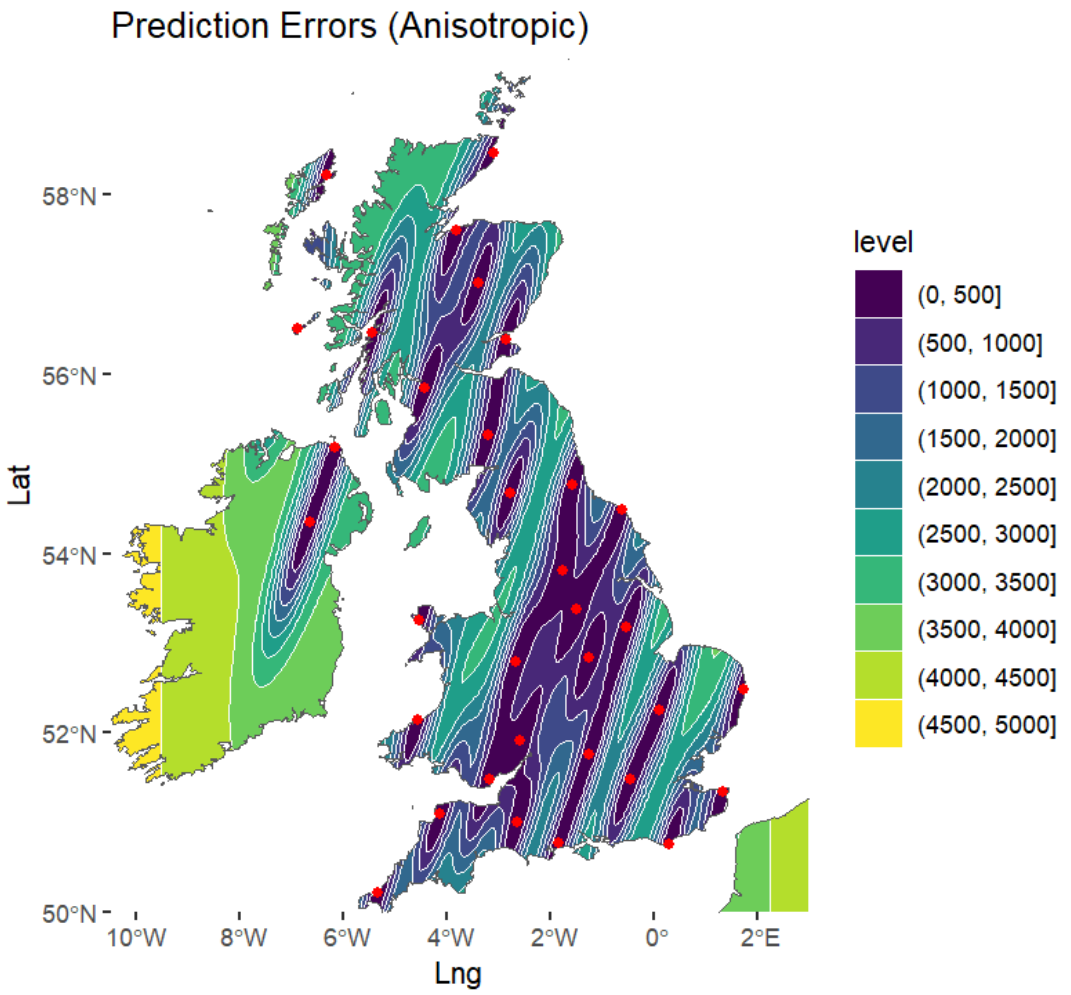
(d)

Unlike ordinary Kriging, the mean function in universal Kriging is no longer constant and varies with longitude since $E(X_t) = \alpha_1 + \alpha_2 \text{lon}(t)$ for rainfall. From the previous plot, we see that, on average, the predictions decrease as longitude increases.

Q11

The covariance function in Q9 is isotropic: the predictions and prediction errors depend on the distance between the sampling locations and the location of interest. However, in Q11, the covariance function is anisotropic because every difference vector $t - s$ is rotated 30 degrees clockwise followed by stretching by factor 2 in the new x direction i.e. 120 bearing and factor 1/2 in the new y direction i.e. 030 bearing, so covariance decays slower in the 030 bearing direction and decays faster in the 120 bearing direction.


Predictions (Anisotropic)
Lat
58°N
56°N
54°N
52°N
50°N
10°W
8°W
6°W
4°W
2°W
0°
2°E
Lng
level
(-17.4, 0.0]
(0.0, 17.4]
(17.4, 34.7]
(34.7, 52.1]
(52.1, 69.5]
(69.5, 86.8]
(86.8, 104.2]
(104.2, 121.6]
(121.6, 138.9]
(138.9, 156.3]
(156.3, 173.7]
(173.7, 191.1]
(191.1, 208.4]
(208.4, 225.8]
(225.8, 243.2]
(243.2, 260.5]
(260.5, 277.9]
(277.9, 295.3]



Appendix

Code for Q1-6:

```
library(readxl)
library(ggplot2)

setwd("[file path]")
my_data <- read_excel("Project7.xlsx")
my_data$rain

# Simple Kriging
N <- length(my_data$rain) # Number of weather stations
sigmasquared <- var(my_data$rain); beta <- 1.5 # Gaussian covariance parameters
mu <- mean(my_data$rain) # Use the sample mean
Sigma <- matrix(0,N,N) # covariance between locations
for (ii in 1:N) {
  for (jj in 1:N) {
    d <- sqrt((my_data$lon[ii]-my_data$lon[jj])^2+(my_data$lat[ii]-my_data$lat[jj])^2)
    Sigma[ii, jj] <- sigmasquared*exp(-beta*d^2) } }
```

```
invSig <- solve(Sigma) # inverse covariance
lat <- c(51.499, 51.525)  # Imperial and UCL latitudes
lon <- c(-0.175, -0.134)  # Imperial and UCL longitudes
num_points <- length(lat)

# Initialise prediction and error vectors
XP <- numeric(num_points)
XE <- numeric(num_points)
for (ii in 1:num_points) { # kriging procedure
  d <- sqrt((lon[ii]-my_data$lon)^2+(lat[ii]-my_data$lat)^2)
  K <- sigmasquared*exp(-beta*d^2) # K vector
  XP[ii] <- mu+K%*%invSig%*%(my_data$rain-mu) # predictions
  XE[ii] <- sigmasquared-K%*%invSig%*%K } # prediction errors

# Predictions
print(XP)

# Prediction errors
print(XE)
```

Code for Q7:

```
# Ordinary Kriging

N <- length(my_data$rain) # Number of weather stations
sigmasquared <- var(my_data$rain); beta <- 1.5 # Gaussian covariance parameters
Sigma <- matrix(0, N, N) # covariance matrix

# Compute covariance matrix Sigma
for (ii in 1:N) {
  for (jj in 1:N) {
    d <- sqrt((my_data$lon[ii] - my_data$lon[jj])^2 + (my_data$lat[ii] -
    ↪  my_data$lat[jj])^2)
    Sigma[ii, jj] <- sigmasquared * exp(-beta * d^2)
  }
}

invSig <- solve(Sigma) # Inverse of covariance matrix
One <- rep(1, N) # Vector of ones

# Initialise prediction and error vectors
XP <- numeric(num_points)
XE <- numeric(num_points)

# Ordinary Kriging procedure
for (ii in 1:num_points) {
  d <- sqrt((lon[ii] - my_data$lon)^2 + (lat[ii] - my_data$lat)^2)
  K <- sigmasquared * exp(-beta * d^2) # Covariance vector K

  # Compute prediction using Theorem 3.3 (BLUP)
  XP[ii] <- t(K) %*% invSig %*% my_data$rain +
    (1 - t(One) %*% invSig %*% K) / (t(One) %*% invSig %*% One) * (t(One) %*% invSig %*%
    ↪  my_data$rain)
```

```
  # Compute mean squared prediction error using Theorem 3.3
  XE[ii] <- sigmasquared - t(K) %*% invSig %*% K +
    (1 - t(One) %*% invSig %*% K)^2 / (t(One) %*% invSig %*% One)
}

# Prediction for Imperial
print(XP[1])

# Prediction error for Imperial
print(XE[1])
```

Code for Q9:

```
# Ordinary Kriging over the entire UK

# Number of weather stations
N <- length(my_data$lon)

# Parameters
sigma2 <- var(my_data$rain)
beta <- 1.5

# Compute covariance matrix Sigma
Sigma <- matrix(0, N, N)
for (ii in 1:N) {
  for (jj in 1:N) {
    d <- sqrt((my_data$lon[ii] - my_data$lon[jj])^2 + (my_data$lat[ii] -
    ↪  my_data$lat[jj])^2)
    Sigma[ii, jj] <- sigma2 * exp(-beta * d^2)
  }
}

invSig <- solve(Sigma)
invData <- my_data$rain
One <- rep(1, N)

# Define grid for interpolation
lat <- seq(50, 59.5, 0.05)
lon <- seq(-10.5, 3, 0.05)

latl <- length(lat)
lonl <- length(lon)

XP <- matrix(0, lonl, latl)
XE <- matrix(0, lonl, latl)

# Ordinary Kriging procedure
for (ii in 1:lonl) {
  for (jj in 1:latl) {
    K <- sigma2 * exp(-beta * ((lon[ii] - my_data$lon)^2 + (lat[jj] - my_data$lat)^2))

    # Compute prediction using Theorem 3.3 (BLUP)
    XP[ii, jj] <- t(K) %*% invSig %*% invData +
      (1 - t(One) %*% invSig %*% K) / (t(One) %*% invSig %*% One) * (t(One) %*% invSig
      ↪  %*% invData)
```

```
    # Compute mean squared prediction error using Theorem 3.3
    XE[ii, jj] <- sigma2 - t(K) %*% invSig %*% K +
      (1 - t(One) %*% invSig %*% K)^2 / (t(One) %*% invSig %*% One)
  }
}

# Download ocean shape data
sea <- ne_download(scale = 10, type = 'ocean', category = "physical", returnclass = "sf")

# Convert station locations into a data frame
stations <- data.frame(lon = my_data$lon, lat = my_data$lat)

# Flatten matrices for ggplot
data_pred <- cbind(expand.grid(Lng = lon, Lat = lat), P = c(XP))
data_err <- cbind(expand.grid(Lng = lon, Lat = lat), P = c(XE))

# Plot Kriging Predictions
ggplot(data_pred) +
  ggtitle("Prediction Map (Ordinary Kriging) - Rainfall") +
  geom_contour_filled(aes(Lng, Lat, z = P), bins = 20, color = "white", linewidth = 0) +
  guides(colour = "colorbar", size = "legend", shape = "legend") +
  geom_sf(data = sea, fill = "white") +
  coord_sf(ylim = c(50, 59.5), xlim = c(-10.5, 3), expand = FALSE) +
  geom_point(data = stations, aes(x = lon, y = lat), colour = "red")

# Plot Prediction Errors
ggplot(data_err) +
  ggtitle("Prediction Error Map (Ordinary Kriging) - Rainfall") +
  geom_contour_filled(aes(Lng, Lat, z = P), bins = 10, color = "white", linewidth = 0) +
  guides(colour = "colorbar", size = "legend", shape = "legend") +
  geom_sf(data = sea, fill = "white") +
  coord_sf(ylim = c(50, 59.5), xlim = c(-10.5, 3), expand = FALSE) +
  geom_point(data = stations, aes(x = lon, y = lat), colour = "red")
```

Code for Q10a-b:

```
library(readxl)
library(ggplot2)
library(rnaturalearth)
library(rnaturalearthdata)

# Load data
my_data <- read_excel("Project7.xlsx")

# Number of weather stations
N <- length(my_data$lon)

# Parameters
sigma2 <- var(my_data$rain)
beta <- 1.5

# Compute covariance matrix Sigma
Sigma <- matrix(0, N, N)
for (ii in 1:N) {
```

```
  for (jj in 1:N) {
    d <- sqrt((my_data$lon[ii] - my_data$lon[jj])^2 + (my_data$lat[ii] -
    ↪  my_data$lat[jj])^2)
    Sigma[ii, jj] <- sigma2 * exp(-beta * d^2)
  }
}

# Compute M matrix for Universal Kriging (linear trend with longitude)
M <- cbind(1, my_data$lon)  # Two columns: [1, lon]

# Precompute inverses for efficiency
invSig <- solve(Sigma)
invSigM <- invSig %*% M
MinvSigM <- solve(t(M) %*% invSigM)

# Define grid for interpolation
lat <- seq(50, 59.5, 0.05)
lon <- seq(-10.5, 3, 0.05)

latl <- length(lat)
lonl <- length(lon)

XP <- matrix(0, lonl, latl)
XE <- matrix(0, lonl, latl)

# Universal Kriging procedure
for (ii in 1:lonl) {
  for (jj in 1:latl) {
    # Compute covariance vector K
    K <- sigma2 * exp(-beta * ((lon[ii] - my_data$lon)^2 + (lat[jj] - my_data$lat)^2))

    # Mean function at the prediction location
    m_t0 <- c(1, lon[ii])

    # Universal Kriging predictor (Theorem 3.4)
    XP[ii, jj] <- t(K) %*% invSig %*% my_data$rain +
      t(m_t0 - t(M) %*% invSig %*% K) %*% MinvSigM %*% (t(M) %*% invSig %*% my_data$rain)

    # Mean squared prediction error (Theorem 3.4)
    XE[ii, jj] <- sigma2 - t(K) %*% invSig %*% K +
      t(m_t0 - t(M) %*% invSig %*% K) %*% MinvSigM %*% (m_t0 - t(M) %*% invSig %*% K)
  }
}

# Prediction at Imperial College London (Lat: 51.499, Lon: -0.175)
K_imperial <- sigma2 * exp(-beta * ((-0.175 - my_data$lon)^2 + (51.499 - my_data$lat)^2))
m_t0_imperial <- c(1, -0.175)

XP_imperial <- t(K_imperial) %*% invSig %*% my_data$rain +
  t(m_t0_imperial - t(M) %*% invSig %*% K_imperial) %*% MinvSigM %*% (t(M) %*% invSig %*%
  ↪  my_data$rain)

XE_imperial <- sigma2 - t(K_imperial) %*% invSig %*% K_imperial +
```

```
  t(m_t0_imperial - t(M) %*% invSig %*% K_imperial) %*% MinvSigM %*% (m_t0_imperial -
  ↪  t(M) %*% invSig %*% K_imperial)

cat("Rainfall Prediction at Imperial College London:", XP_imperial, "\n")
cat("Prediction Error at Imperial College London:", XE_imperial, "\n")

# Download ocean shape data
sea <- ne_download(scale = 10, type = 'ocean', category = "physical", returnclass = "sf")

# Convert station locations into a data frame
stations <- data.frame(lon = my_data$lon, lat = my_data$lat)

# Flatten matrices for ggplot
data_pred <- cbind(expand.grid(Lng = lon, Lat = lat), P = c(XP))
data_err <- cbind(expand.grid(Lng = lon, Lat = lat), P = c(XE))

# Plot Kriging Predictions
ggplot(data_pred) +
  ggtitle("Prediction Map (Universal Kriging) - Rainfall") +
  geom_contour_filled(aes(Lng, Lat, z = P), bins = 20, color = "white", linewidth = 0) +
  guides(colour = "colorbar", size = "legend", shape = "legend") +
  geom_sf(data = sea, fill = "white") +
  coord_sf(ylim = c(50, 59.5), xlim = c(-10.5, 3), expand = FALSE) +
  geom_point(data = stations, aes(x = lon, y = lat), colour = "red")

# Plot Prediction Errors
ggplot(data_err) +
  ggtitle("Prediction Error Map (Universal Kriging) - Rainfall") +
  geom_contour_filled(aes(Lng, Lat, z = P), bins = 10, color = "white", linewidth = 0) +
  guides(colour = "colorbar", size = "legend", shape = "legend") +
  geom_sf(data = sea, fill = "white") +
  coord_sf(ylim = c(50, 59.5), xlim = c(-10.5, 3), expand = FALSE) +
  geom_point(data = stations, aes(x = lon, y = lat), colour = "red")
```

Code for Q10c:

```
# Map of differences between universal and ordinary Kriging predictions

# Define a vector of ones for the mean correction term
One <- rep(1, N)

# Ordinary Kriging procedure
XO <- matrix(0, lonl, latl)
XE <- matrix(0, lonl, latl)

for (ii in 1:lonl) {
  for (jj in 1:latl) {
    d <- sqrt((lon[ii] - my_data$lon)^2 + (lat[jj] - my_data$lat)^2)
    K <- sigma2 * exp(-beta * d^2)  # Covariance vector K

    # Compute prediction using Theorem 3.3 (BLUP)
    XO[ii, jj] <- t(K) %*% invSig %*% my_data$rain +
      (1 - t(One) %*% invSig %*% K) / (t(One) %*% invSig %*% One) * (t(One) %*% invSig
      ↪  %*% my_data$rain)
```

```
    # Compute mean squared prediction error using Theorem 3.3
    XE[ii, jj] <- sigma2 - t(K) %*% invSig %*% K +
      (1 - t(One) %*% invSig %*% K)^2 / (t(One) %*% invSig %*% One)
  }
}

# Compute difference between Universal and Ordinary Kriging predictions
X_diff <- XP - XO

# Flatten matrix for ggplot
data_diff <- cbind(expand.grid(Lng = lon, Lat = lat), Diff = c(X_diff))

# Plot the difference
ggplot(data_diff) +
  ggtitle("Difference: Universal Kriging - Ordinary Kriging") +
  geom_contour_filled(aes(Lng, Lat, z = Diff), bins = 20, color = "white", linewidth = 0)
  ↪  +
  guides(colour = "colorbar", size = "legend", shape = "legend") +
  geom_sf(data = sea, fill = "white") +
  coord_sf(ylim = c(50, 59.5), xlim = c(-10.5, 3), expand = FALSE) +
  geom_point(data = stations, aes(x = lon, y = lat), colour = "red")
```

Code for Q11:

```
library(readxl)
library(ggplot2)
library(rnaturalearth)
library(rnaturalearthdata)
library(reshape2)

# Load data
my_data <- read_excel("Project7.xlsx")

# Number of weather stations
N <- length(my_data$lon)

# Define parameters
beta <- 1.5
sigmasquared <- var(my_data$rain)
rho <- 2
theta <- -pi / 6

# Define transformation matrices
D <- matrix(c(rho, 0, 0, 1 / rho), nrow = 2)
R <- matrix(c(cos(theta), -sin(theta), sin(theta), cos(theta)), nrow = 2)
A <- D %*% R

# Compute covariance matrix Sigma
Sigma <- matrix(0, N, N)
for (ii in 1:N) {
  for (jj in 1:N) {
    delta <- A %*% matrix(c(my_data$lon[ii] - my_data$lon[jj], my_data$lat[ii] -
    ↪  my_data$lat[jj]), nrow = 2)
    d_squared <- sum(delta^2)
```

```
    Sigma[ii, jj] <- sigmasquared * exp(-beta * d_squared)
  }
}

# Define auxiliary variables
One <- rep(1, N)
Lon <- my_data$lon

# Define grid for prediction
lat <- seq(50, 59.5, 0.05)
lon <- seq(-10.5, 3, 0.05)
latl <- length(lat)
lonl <- length(lon)

# Initialise prediction matrices
XP <- matrix(0, lonl, latl)
XE <- matrix(0, lonl, latl)

# Compute inverse of Sigma
invSig <- solve(Sigma)

# Loop through grid points for predictions
for (ii in 1:lonl) {
  for (jj in 1:latl) {
    delta <- A %*% rbind(lon[ii] - my_data$lon, lat[jj] - my_data$lat)
    d_squared <- colSums(delta^2)
    K <- sigmasquared * exp(-beta * d_squared)

    M <- cbind(One, Lon)
    M_invSig_M <- solve(t(M) %*% invSig %*% M)
    M_invSig_K <- t(M) %*% invSig %*% K

    m_t0 <- c(1, lon[ii])

    x <- solve(rbind(cbind(Sigma, One, Lon), cbind(t(One), 0, 0), cbind(t(Lon), 0, 0)),
    ↪  c(K, 1, lon[ii]))
    alpha <- x[1:N]

    XP[ii, jj] <- t(K) %*% invSig %*% my_data$rain +
      t(m_t0 - M_invSig_K) %*% M_invSig_M %*% (t(M) %*% invSig %*% my_data$rain)
    XE[ii, jj] <- sigmasquared - t(K) %*% invSig %*% K +
      t(m_t0 - M_invSig_K) %*% M_invSig_M %*% (m_t0 - M_invSig_K)
  }
}

# Prepare data for plotting
data_pred <- cbind(expand.grid(Lng = lon, Lat = lat), P = c(XP))
data_err <- cbind(expand.grid(Lng = lon, Lat = lat), P = c(XE))

# Include sample points
FPlat <- numeric(latl*lonl)
FPlat[1:length(my_data$lat)]<- my_data$lat
FPlon <- numeric(latl*lonl)
FPlon[1:length(my_data$lon)]<- my_data$lon
```

```r
# Plot predictions
ggplot(data_pred) +
  ggtitle("Predictions (Anisotropic)") +
  geom_contour_filled(aes(Lng, Lat, z = P), bins = 20, color = "white", linewidth = 0) +
  guides(colour = "colorbar", size = "legend", shape = "legend") +
  geom_sf(data = sea, fill = "white") +
  coord_sf(ylim = c(50, 59.5), xlim = c(-10.5, 3), expand = FALSE) +
  geom_point(aes(FPlon, FPlat), colour = "red")

# Plot prediction errors
ggplot(data_err) +
  ggtitle("Prediction Errors (Anisotropic)") +
  geom_contour_filled(aes(Lng, Lat, z = P), bins = 10, color = "white", linewidth = 0) +
  guides(colour = "colorbar", size = "legend", shape = "legend") +
  geom_sf(data = sea, fill = "white") +
  coord_sf(ylim = c(50, 59.5), xlim = c(-10.5, 3), expand = FALSE) +
  geom_point(aes(FPlon, FPlat), colour = "red")
```