Day11 exercise solutions

Ali Movasati, Isabelle Caroline Rose Cretton, Tristan Koning

Nov. 29th, 2024

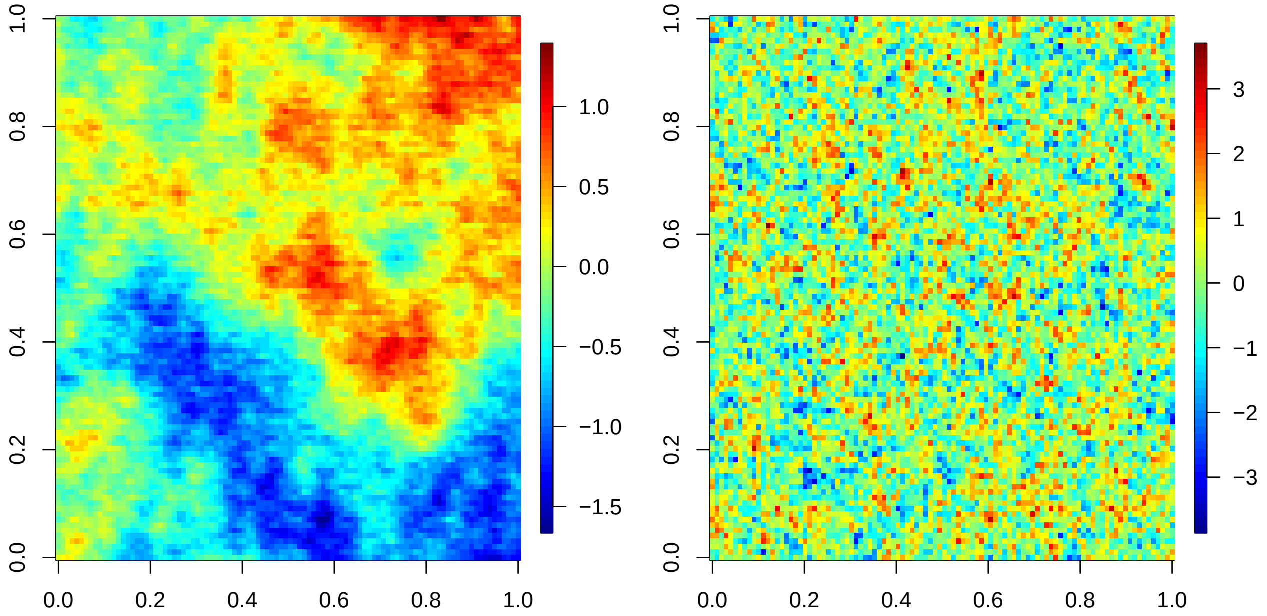
```
# Set global code chunk options
knitr::opts_chunk$set(
  echo = TRUE,
  warning = FALSE,
  message = FALSE,
  fig.width = 10,
  fig.height = 6
)
```

```
# load required libraries
library("skimr")
library("dplyr")
library("magrittr")
library("ggplot2")
library("survival")
library("survminer")
library("fields")

# define functions
`%notin%` <- Negate(`%in%`)
```

Problem 1

1.A)

```
load("/Users/alimos313/Documents/studies/phd/university/courses/stat-modelling/StatModelEx/day11/data/sp

par(mfrow=c(1,2))
image.plot(sim1)
image.plot(sim2)
```



```
# summary statistics
summary(as.vector(sim1))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
## -1.64094 -0.55859 -0.05631 -0.13214  0.28576  1.37183
```

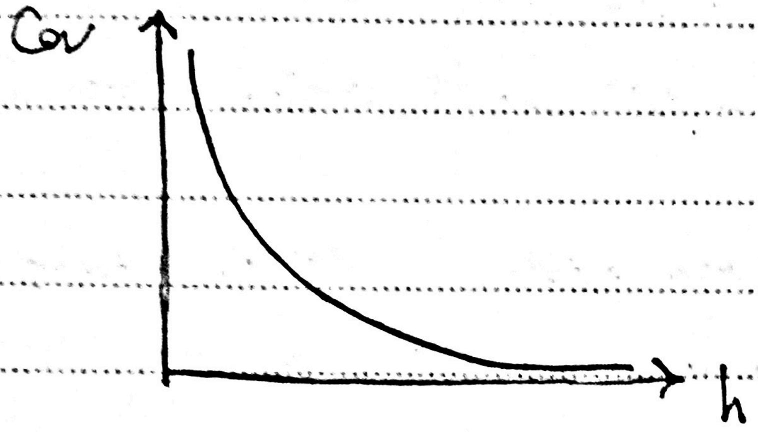
```
summary(as.vector(sim2))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
## -3.80355 -0.63321  0.01862  0.01869  0.68658  3.64528
```

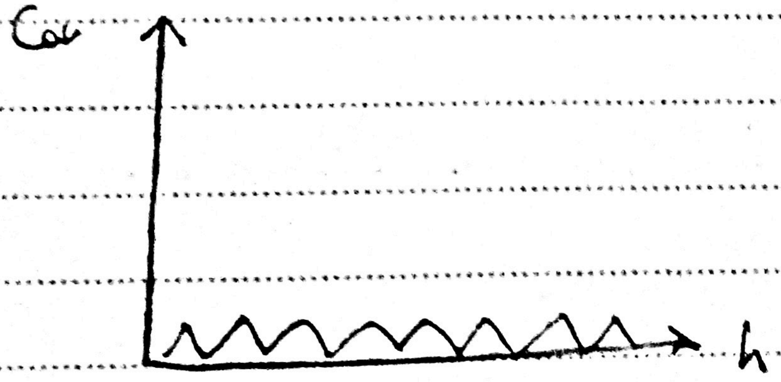
1.B)

subject: date:

Sim1



Sim2

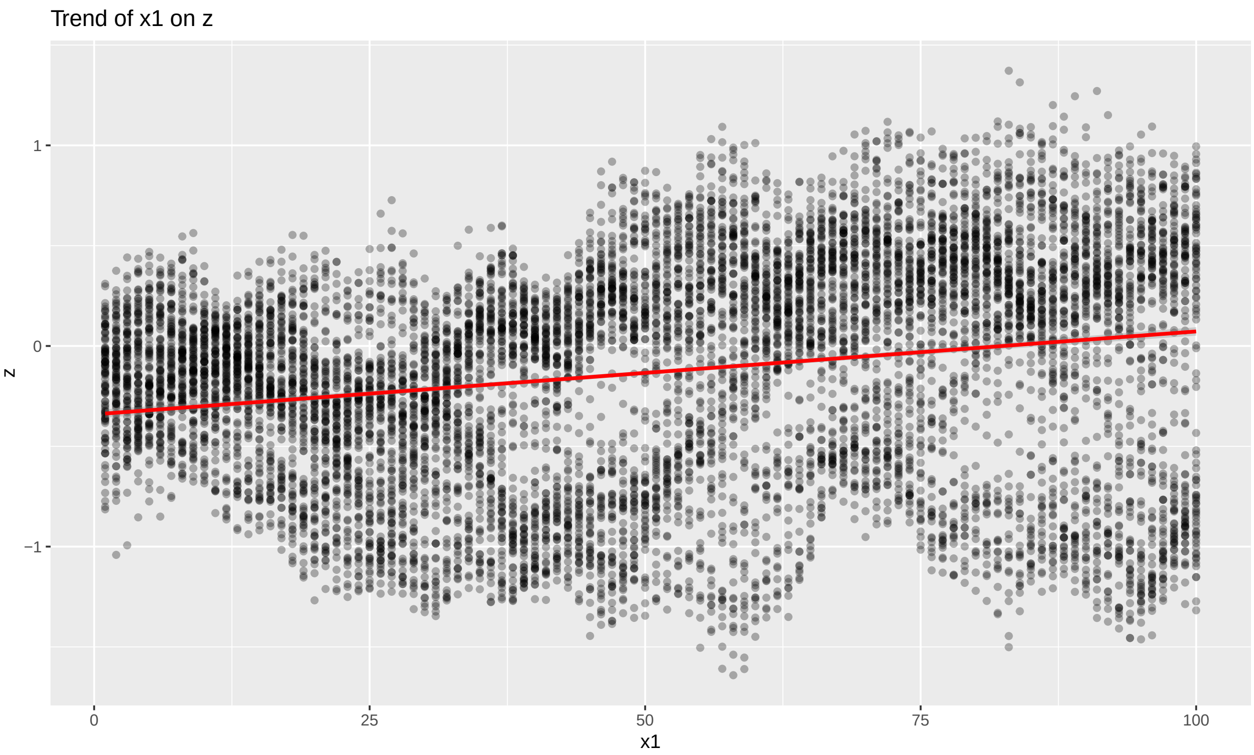


1.C)

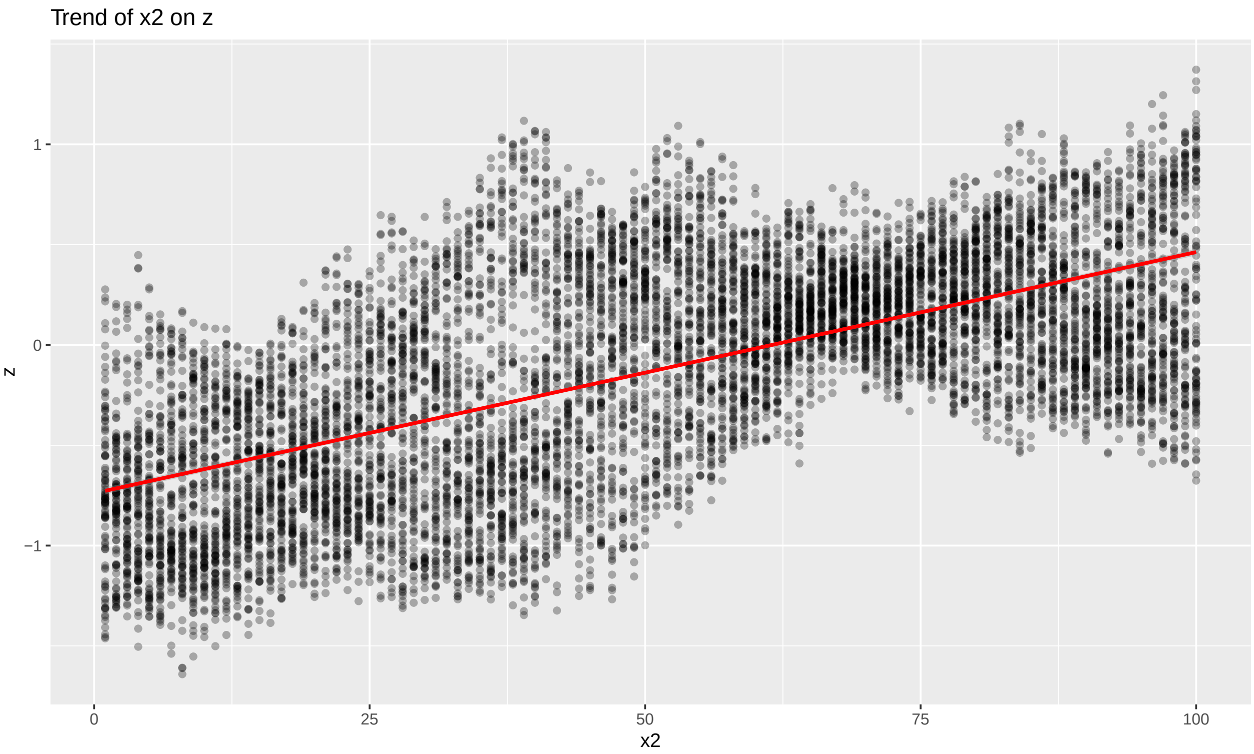
```r
coords <- expand.grid(x1 = 1:nrow(sim1), x2 = 1:ncol(sim1))
sim1_df <- data.frame(coords, z = as.vector(sim1))

sim1_lm <- lm(z ~ x1 + x2, data = sim1_df)
summary(sim1_lm)
```

```
##
## Call:
## lm(formula = z ~ x1 + x2, data = sim1_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02896 -0.30692  0.00469  0.27550  1.32699
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.947859   0.011015  -86.06   <2e-16 ***
## x1           0.004133   0.000143   28.91   <2e-16 ***
## x2           0.012020   0.000143   84.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4128 on 9997 degrees of freedom
## Multiple R-squared:  0.4415, Adjusted R-squared:  0.4414
## F-statistic:  3951 on 2 and 9997 DF,  p-value: < 2.2e-16
```

```r
ggplot(sim1_df, aes(x = x1, y = z)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "Trend of x1 on z", x = "x1", y = "z")
```



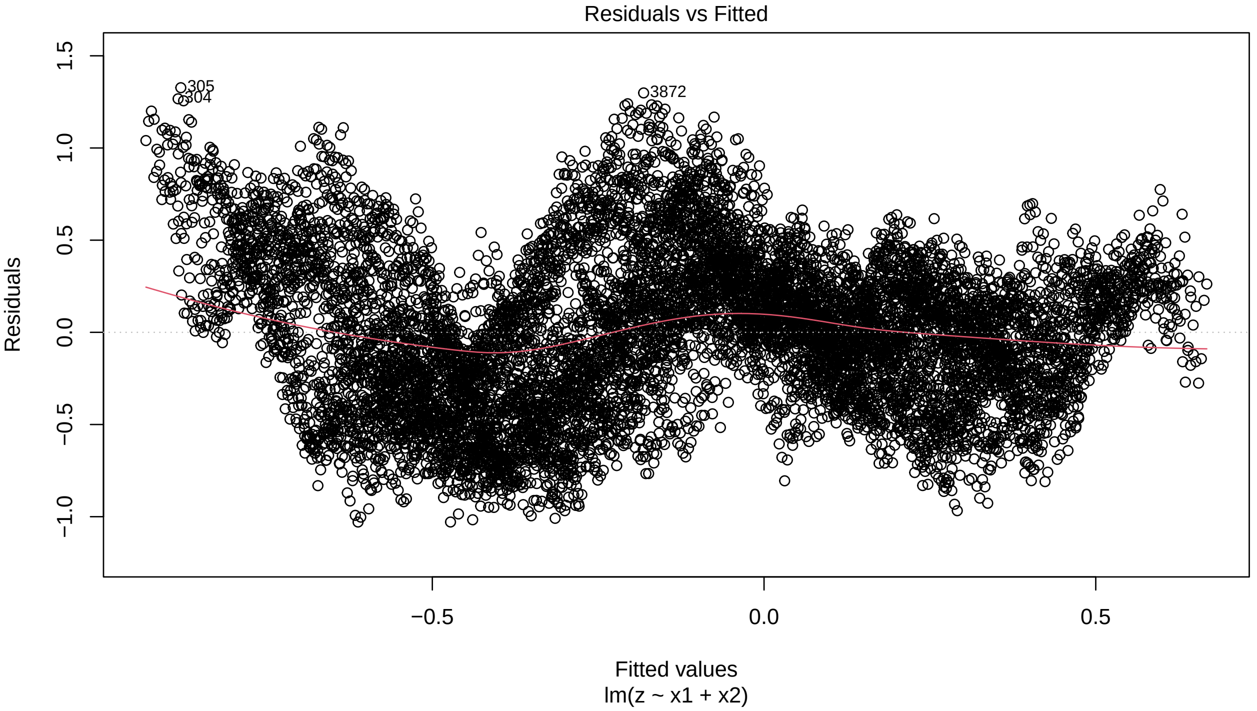
```r
ggplot(sim1_df, aes(x = x2, y = z)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "Trend of x2 on z", x = "x2", y = "z")
```

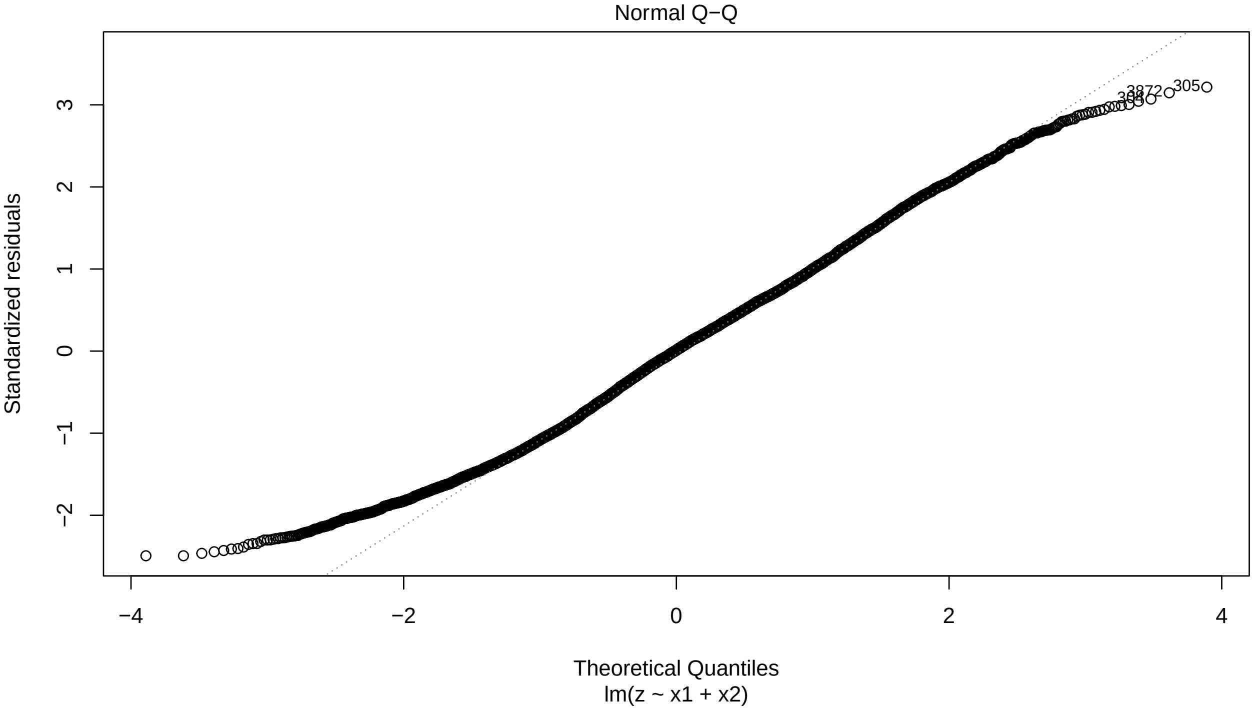


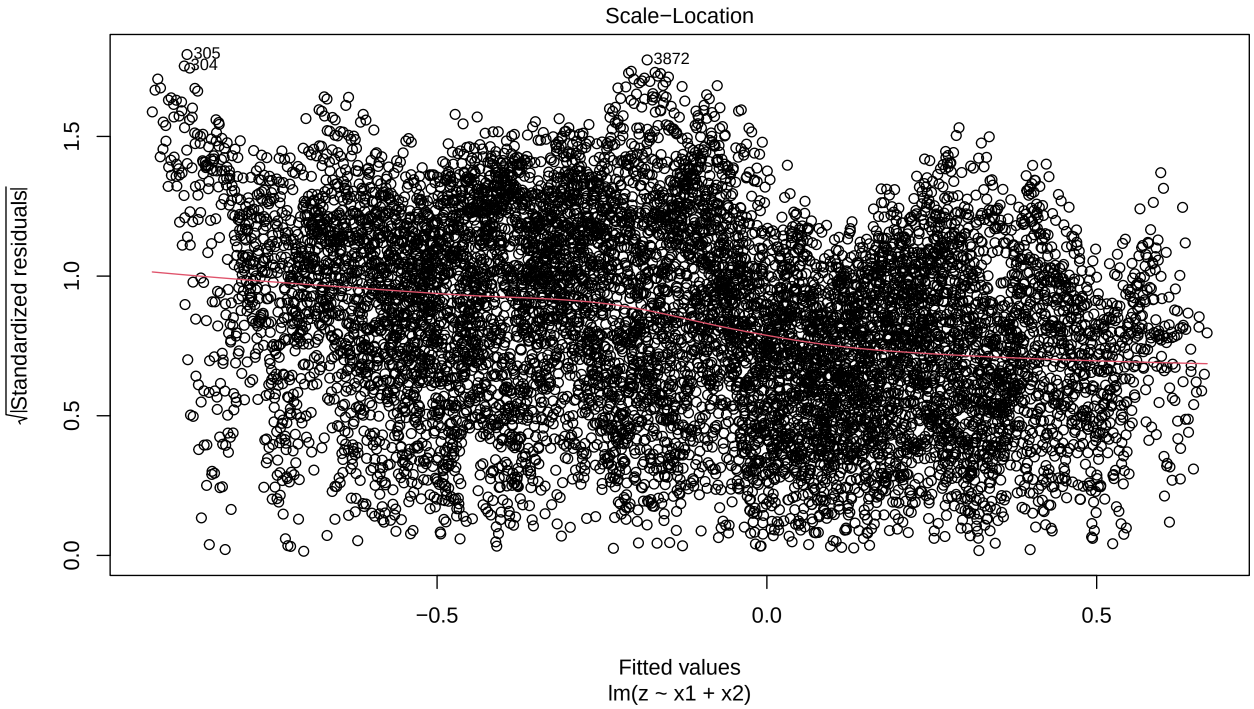
There appears to be a trend for both x1 and x2 that the higher these coordinate values go, so does the value of z. This is more pronounced on z. This makes sense as we seem to have a hotspot in the “top right corner” of the plotted image, and lower values in the “bottom left corner”.

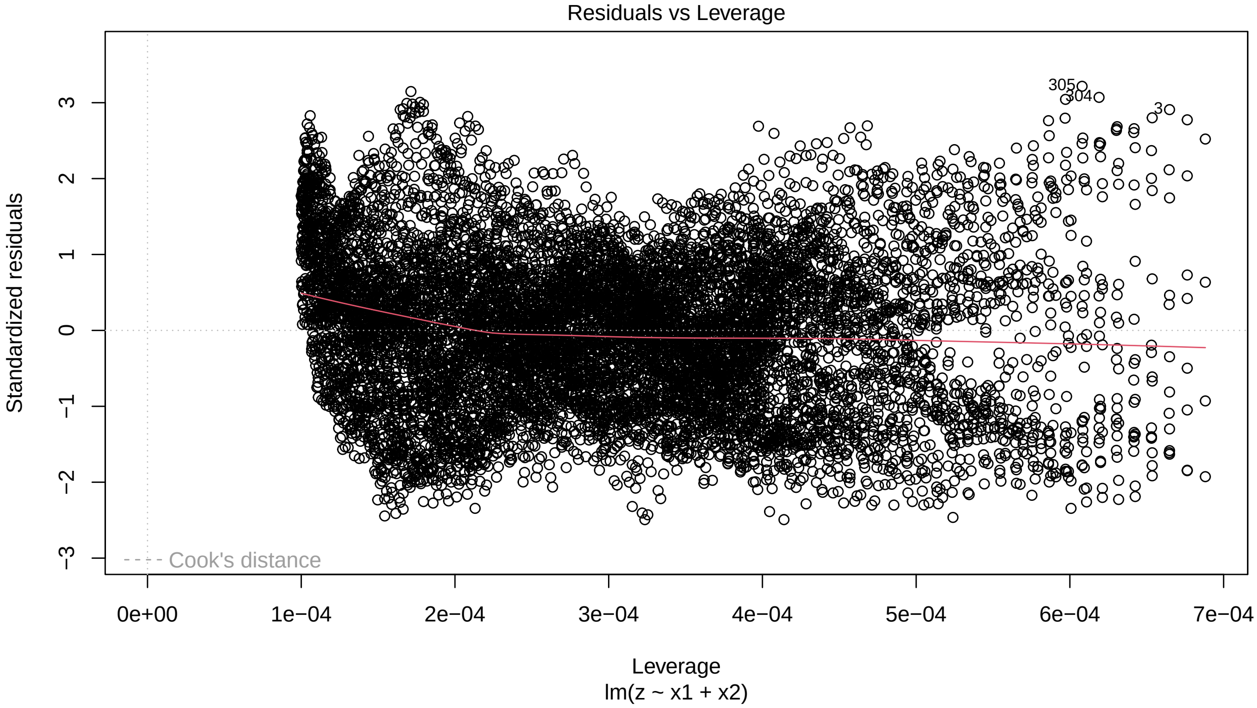
1.D)

```r
# Assumptions of the linear model
plot(sim1_lm)
```









The residuals vs fitted plot shows a pattern, which indicates that the linear model is not appropriate for this data. The Q-Q plot shows that the residuals are not normally distributed, as they deviate quite a bit. The scale-location plot shows that the residuals are homoscedastic, as there is no clear structure visible. The residuals vs leverage plot shows that there arent any high leverage points.

1.E)

« comments »

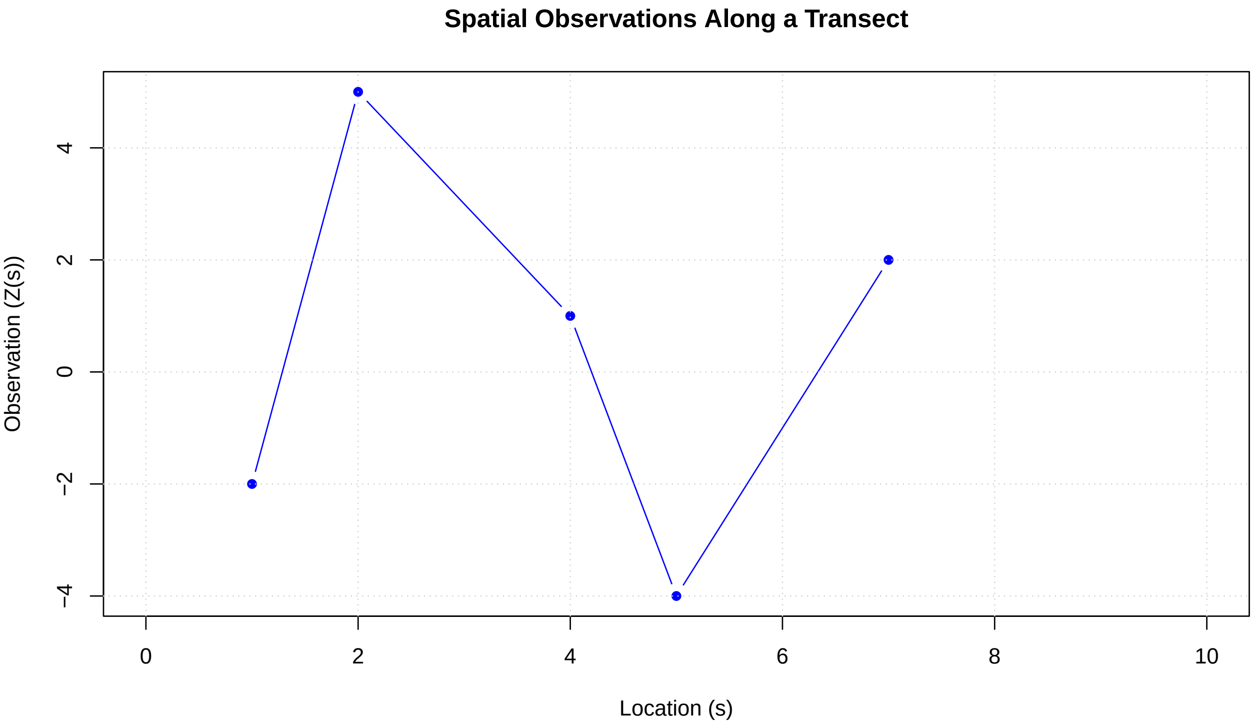
A spatial regression model like kriging or spatial mixed models could be used: Z(s) = + S(s) + Where:

Z(s): Spatial process : Mean trend S(s): Spatial random effect : Independent error term

Problem 2

2.A)

```
transect <- read.table(file = "/Users/alimos313/Documents/studies/phd/university/courses/stat-modelling


# Plot the data
plot(
  transect$s, transect$Z.s,
  type = "b",                      # Connect points with lines ("b" = both points and lines)
  pch = 16,                        # Use filled circles for points
  col = "blue",                    # Color of points and lines
  xlim = c(0, 10),                 # Set x-axis range to [0, 10]
  ylim = range(transect$Z.s),          # Automatically adjust y-axis range
  xlab = "Location (s)",           # Label for x-axis
  ylab = "Observation (Z(s))",     # Label for y-axis
  main = "Spatial Observations Along a Transect"
)

# Optionally add a grid for better readability
grid()
```



```
skim(transect)
```

Table 1: Data summary

Name	transect

Number of rows	5
Number of columns	2
Column type frequency:	
numeric	2
Group variables	None

Variable type: numeric

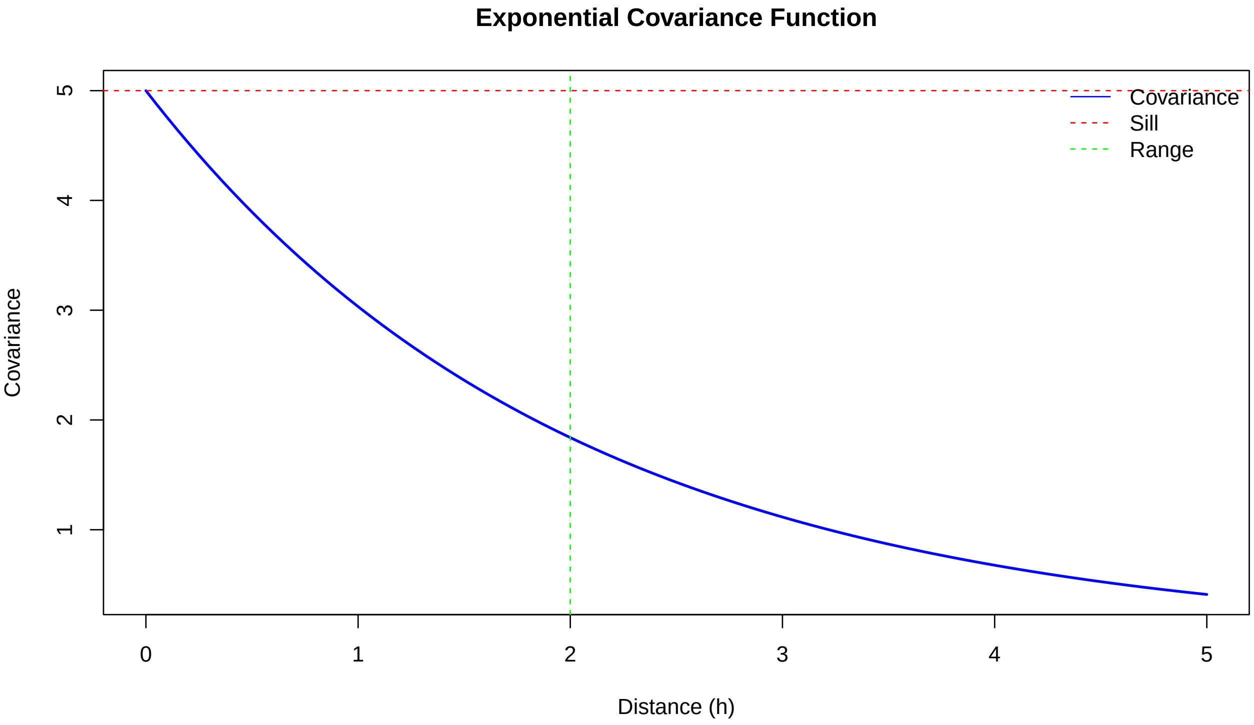
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
s	0	1	3.8	2.39	1	2	4	5	7	
Z.s	0	1	0.4	3.51	-4	-2	1	2	5	

```r
head(transect)
```

```
##   s Z.s
## 1 1  -2
## 2 2   5
## 3 4   1
## 4 5  -4
## 5 7   2
```

2.B)

```
# Define the exponential covariance function
exp.cov <- function(h, theta2, theta3) {
  theta2 * exp(-h / theta3)
}

# Parameters
theta2 <- 5    # Sill
theta3 <- 2    # Range
h <- seq(0, 5, length.out = 100)  # Distances from 0 to 5

# Compute covariance values
cov_values <- exp.cov(h, theta2, theta3)

# Plot
plot(
  h, cov_values,
  type = "l",                # Line plot
  col = "blue",              # Line color
  lwd = 2,                   # Line width
  xlab = "Distance (h)",     # X-axis label
  ylab = "Covariance",       # Y-axis label
  main = "Exponential Covariance Function"
)

# Highlight key parameters
abline(h = theta2, col = "red", lty = 2)  # Sill (horizontal line)
abline(v = theta3, col = "green", lty = 2)  # Range (vertical line)
legend("topright", legend = c("Covariance", "Sill", "Range"),
       col = c("blue", "red", "green"), lty = c(1, 2, 2), bty = "n")
```



« comments »

Sill (2=5 2=5): The red dashed line at 5 corresponds to the sill. It represents the maximum covariance value. Range (3=2 3=2): The green dashed line at h=2h=2 indicates the range, the distance at which the covariance drops to approximately 2 e−1 1.84 2 e−1 1.84. Nugget (1=0 1=0): Since there's no nugget effect, the covariance starts at the sill (cov(0)= 2=5cov(0)= 2=5).

2.C)

```r
# Function to calculate pairwise Euclidean distances
dist.matrix <- function(x, y) {
  # Use outer() to calculate pairwise absolute differences
  outer(x, y, function(a, b) abs(a - b))
}

# Locations along the transect
locations <- c(1, 2, 4, 5, 7)

# Calculate the distance matrix
DIST.MAT <- dist.matrix(locations, locations)

# Print the distance matrix
print(DIST.MAT)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    3    4    6
## [2,]    1    0    2    3    5
## [3,]    3    2    0    1    3
## [4,]    4    3    1    0    2
## [5,]    6    5    3    2    0
```

2.D)

```r
# Transect locations
locations <- c(1, 2, 4, 5, 7)

# Calculate the distance matrix
DIST.MAT <- dist.matrix(locations, locations)

# Parameters for the covariance function
theta2 <- 5    # Sill
theta3 <- 2    # Range

# Calculate the covariance matrix
SIGMA <- exp.cov(DIST.MAT, theta2, theta3)

# Print the covariance matrix
print(SIGMA)
```

```
##           [,1]     [,2]     [,3]      [,4]      [,5]
## [1,] 5.0000000 3.032653 1.115651 0.6766764 0.2489353
## [2,] 3.0326533 5.000000 1.839397 1.1156508 0.4104250
## [3,] 1.1156508 1.839397 5.000000 3.0326533 1.1156508
## [4,] 0.6766764 1.115651 3.032653 5.0000000 1.8393972
## [5,] 0.2489353 0.410425 1.115651 1.8393972 5.0000000
```

2.E)

```
# Observed locations
locations <- c(1, 2, 4, 5, 7)

# New locations where prediction is needed
snew <- seq(0, 10, length.out = 5) # For example, 5 evenly spaced points between 0 and 10

# Calculate distances between observed points and new locations
DIST.OBS.NEW <- dist.matrix(locations, snew)

# Parameters for the covariance function
theta2 <- 5     # Sill
theta3 <- 2     # Range

# Calculate the covariance matrix sigma
sigma <- exp.cov(DIST.OBS.NEW, theta2, theta3)

# Print results
cat("New locations (snew):\n")
```

```
## New locations (snew):
```

```
print(snew)
```

```
## [1]  0.0  2.5  5.0  7.5 10.0
```

```
cat("\nDistances between observed locations and snew (DIST.OBS.NEW):\n")
```

```
##
## Distances between observed locations and snew (DIST.OBS.NEW):
```

```
print(DIST.OBS.NEW)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1  1.5    4  6.5    9
## [2,]    2  0.5    3  5.5    8
## [3,]    4  1.5    1  3.5    6
## [4,]    5  2.5    0  2.5    5
## [5,]    7  4.5    2  0.5    3
```

```
cat("\nCovariance matrix (sigma):\n")
```

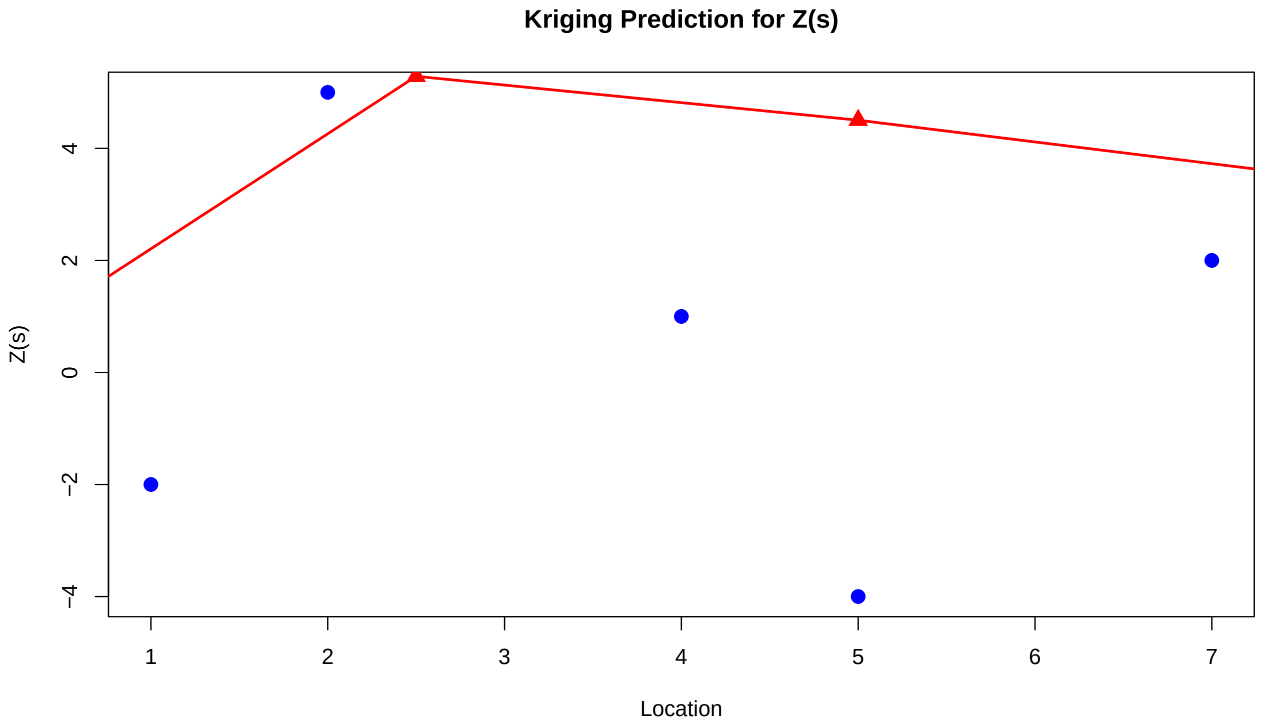
```
##
## Covariance matrix (sigma):
```

```
print(sigma)
```

```
##           [,1]      [,2]      [,3]      [,4]       [,5]
## [1,] 3.0326533 2.3618328 0.6766764 0.1938710 0.05554498
## [2,] 1.8393972 3.8940039 1.1156508 0.3196393 0.09157819
## [3,] 0.6766764 2.3618328 3.0326533 0.8688697 0.24893534
## [4,] 0.4104250 1.4325240 5.0000000 1.4325240 0.41042499
## [5,] 0.1509869 0.5269961 1.8393972 3.8940039 1.11565080
```

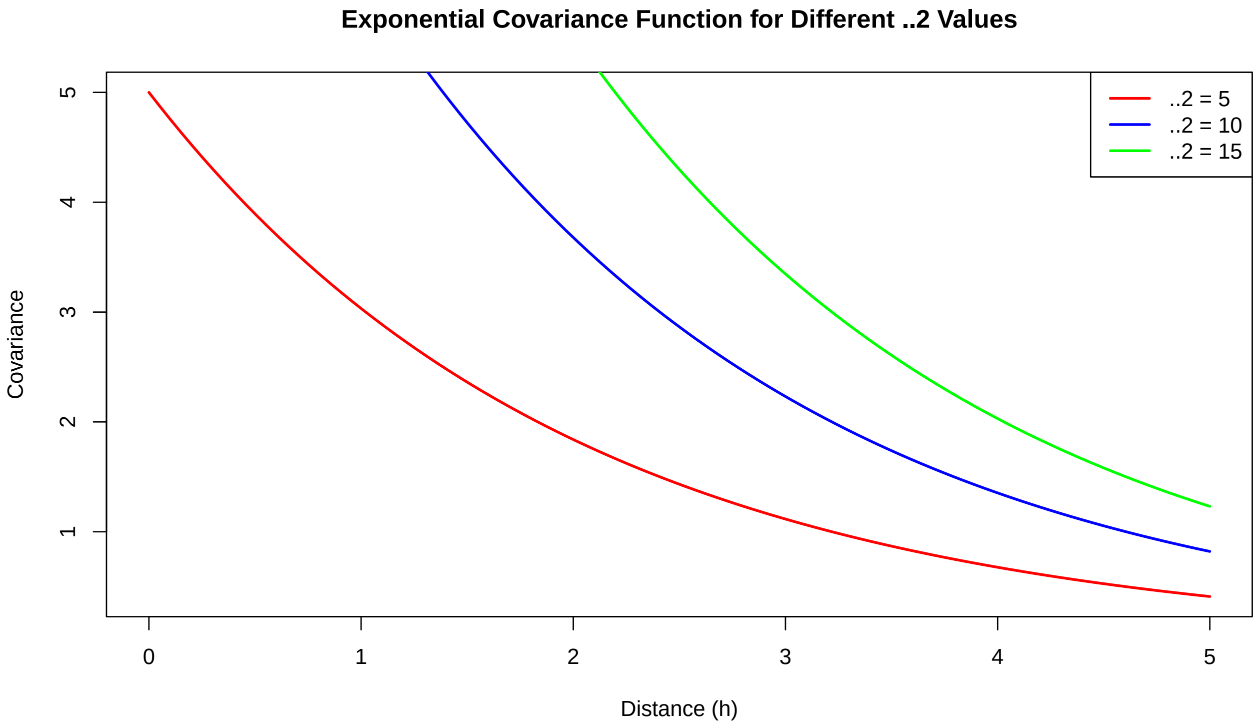
2.F)

```r
# Observed locations and corresponding values
locations <- c(1, 2, 4, 5, 7)
Z_obs <- c(-2, 5, 1, -4, 2)  # Observed values of Z at locations s

# New locations where prediction is needed
snew <- seq(0, 10, length.out = 5) # Example: 5 evenly spaced points between 0 and 10

# Calculate pairwise distances
DIST.OBS.NEW <- dist.matrix(locations, snew)
DIST.OBS.OBS <- dist.matrix(locations, locations)

# Parameters for the covariance function
theta2 <- 5    # Sill
theta3 <- 2    # Range

# Calculate covariance matrices
C_OBS_NEW <- exp.cov(DIST.OBS.NEW, theta2, theta3)  # Covariance vector between observed and new
C_OBS_OBS <- exp.cov(DIST.OBS.OBS, theta2, theta3)  # Covariance matrix for observed locations

# Solve kriging system to calculate weights
C_OBS_OBS_inv <- solve(C_OBS_OBS)  # Inverse of the covariance matrix for observed points
weights <- C_OBS_NEW %*% C_OBS_OBS_inv  # Kriging weights

# Kriging prediction for the new locations
Z_pred <- weights %*% Z_obs  # Predicted values for the new locations

# Plot the observed and predicted values
plot(locations, Z_obs, col = "blue", pch = 16, cex = 1.5, xlab = "Location", ylab = "Z(s)",
     main = "Kriging Prediction for Z(s)")

# Plot the predicted values
lines(snew, Z_pred, col = "red", lwd = 2)

# Optionally, plot points for predicted locations
points(snew, Z_pred, col = "red", pch = 17, cex = 1.5)
```

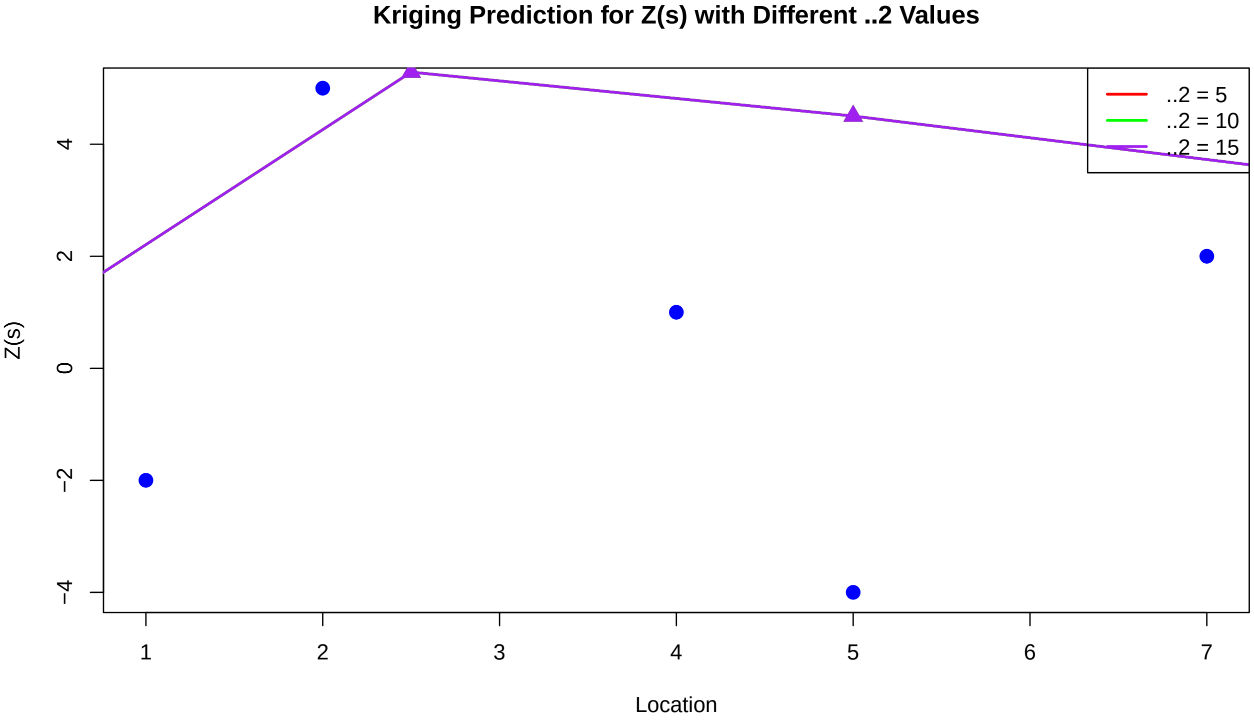


2.G)

```
# Define parameters for the covariance function
theta2_values <- c(5, 10, 15)  # Different sill values to compare
theta3 <- 2                    # Range value

# Generate distances between 0 and 5
h <- seq(0, 5, length.out = 100)

# Plot the covariance function for different theta2 values
plot(h, exp.cov(h, theta2_values[1], theta3), type = "l", col = "red", lwd = 2,
     xlab = "Distance (h)", ylab = "Covariance",
     main = "Exponential Covariance Function for Different  2 Values")
lines(h, exp.cov(h, theta2_values[2], theta3), col = "blue", lwd = 2)
lines(h, exp.cov(h, theta2_values[3], theta3), col = "green", lwd = 2)
legend("topright", legend = paste(" 2 =", theta2_values), col = c("red", "blue", "green"), lwd = 2)
```



```
# Observed locations and values
locations <- c(1, 2, 4, 5, 7)
Z_obs <- c(-2, 5, 1, -4, 2)

# New prediction locations
snew <- seq(0, 10, length.out = 5)


# Create a plot to store the results
plot(locations, Z_obs, col = "blue", pch = 16, cex = 1.5, xlab = "Location", ylab = "Z(s)",
     main = "Kriging Prediction for Z(s) with Different  2 Values")

# Loop to calculate and plot results for different theta2 values
for (theta2 in theta2_values) {

  # Calculate covariance matrices for the given theta2
  C_OBS_NEW <- exp.cov(DIST.OBS.NEW, theta2 = theta2, theta3 = theta3)  # Covariance between observed a
  C_OBS_OBS <- exp.cov(DIST.OBS.OBS, theta2 = theta2, theta3 = theta3)  # Covariance matrix for observe

  # Solve the kriging system
  C_OBS_OBS_inv <- solve(C_OBS_OBS)  # Inverse of the covariance matrix for observed points
  weights <- C_OBS_NEW %*% C_OBS_OBS_inv  # Kriging weights

  # Kriging prediction for the new locations
  Z_pred <- weights %*% Z_obs  # Predicted values for the new locations

  # Plot the predicted values as a line
  lines(snew, Z_pred, col = ifelse(theta2 == 5, "red", ifelse(theta2 == 10, "green", "purple")), lwd = 

  # Optionally, plot points for predicted locations
  points(snew, Z_pred, col = ifelse(theta2 == 5, "red", ifelse(theta2 == 10, "green", "purple")), pch = 
}

# Add a legend to the plot
legend("topright", legend = paste(" 2 =", theta2_values), col = c("red", "green", "purple"), lwd = 2)
```



Effect on the covariance function: As 2 2 increases, the covariance between locations increases, implying a stronger spatial correlation.

Effect on kriging predictions: We do not see a difference on kriging predictions. It's likely because the range of the predicted values for different 2 2 values isn't significantly different. This can happen if the locations where the predictions are made (the snew locations) are not very far apart, or if the model's sensitivity to 2 2 is low for the given parameters.