Capítulo 5

Modelado 2

En este modelo supondremos que las variables cercanas a la fuente puntual están correlacionadas, lo cual se asemeja más a la realidad, pero también es el modelo más complicado a la hora de realizar los cálculos, lo que representa su principal desventaja. Veremos si difiere de suponerlas independientes.

La hipótesis nula es la misma que en el modelo anterior: que las variables son independientes entre sí y que pertenecen a una misma distribución. La hipótesis alternativa es que las variables dentro del círculo están correlacionadas y que el centro del círculo es la fuente puntual, por lo tanto, la distribución conjunta es una función de distribución normal multivariada dada por Σ. Así, las hipótesis son:

$$\begin{aligned} H_0 &: \text{no existe una fuente puntual} \\ &vs \\ H_1 &: \text{ existe una fuente puntual} \end{aligned}$$

Covarianza paramétrica

Hug y González definen una función de covarianza al combinar una covarianza exponencial $R_1(s,t) = \sigma^2 e^{-\tau||\boldsymbol{t}-\boldsymbol{s}||^m}$ con el proceso de Wiener no estacionario cuya función de covarianza es $R_2(s,t) = \prod_{i=1}^{d} \text{mín}(t_i, s_i)$. Finalmente, se obtiene la función de covarianza paramétrica:

$$R(s_i, s_j) = \sigma^2 \exp\{-\tau[d(s_i, s_j)]\} \exp\{\text{mín}\,[(\delta + d(s_i, c))^a, (\delta + d(s_j, c))^a]\}$$

Algunas propiedades del comportamiento de los datos con respecto a la fuente puntual se describen mediante el parámetro a. El parámetro a controla el comportamiento de no estacionaridad. En particular, si $a = 0$, se obtiene la covarianza exponencial estacionaria. Cuando $a > 0$, la varianza aumenta a medida que la fuente puntual se aleja, y disminuye cuando $a < 0$. Este parámetro también controla la correlación entre observaciones con respecto a la fuente puntual.

La función de verosimilitud sera

$$
\begin{aligned}
L(Z) &= L(x_1, ..., x_n) = L_1(X_z)L_2(X_{z^c}) \\
&= L_1(\mu_z, \Sigma_z)L_2(\mu_{z^c}, \Sigma_{z^c})
\end{aligned}
$$

Para los puntos dentro de la zona

$$
L_1(\mu_z, \Sigma_z) = \frac{1}{(2\pi)^{n_Z} \mid \Sigma \mid^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}
$$

$$
\begin{aligned}
logL_1(\mu_z, \Sigma_z) &= log\left[\frac{1}{(2\pi)^{n_Z} \mid \Sigma \mid^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}\right] \\
&= log\left[\frac{1}{(2\pi)^{n_Z} \mid \Sigma \mid^{1/2}}\right] + log\left[exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}\right] \\
&= log[1] - log\left[(2\pi)^{n_Z} \mid \Sigma \mid^{1/2}\right] - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) \\
&= -(log((2\pi)^{n_Z/2}) + log(\mid \Sigma \mid^{1/2}) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) \\
&= -\frac{1}{2}\left[log((2\pi)^{n_Z}) + log(\mid \Sigma \mid) + (x-\mu)^T\Sigma^{-1}(x-\mu))\right]
\end{aligned}
$$

pero como

$$
R(s_i, s_j) = \sigma^2 \exp\{-\mathcal{T} d(s_i, s_j)\} \exp\{\text{mín}[(0{,}5 + d(s_i, c))^a, (0{,}5 + d(s_j, c))^a]\}
$$

Podemos obtener una matriz E tal que

$$
\Sigma = \sigma^2 E
$$

Dado que la matriz E es simétrica y cuadrada, se puede afirmar que es diagonalizable. Por lo tanto, aplicando el teorema de descomposición espectral, su determinante es igual al producto de sus eigenvalores. Así

$$
\mid \Sigma \mid = \mid \sigma^2 E \mid = \sigma^{2n_Z} \mid E \mid = \sigma^{2n_Z} \prod_{i=1}^{n_Z} \lambda_i
$$

Donde los λ_i son los eigenvalores de la matriz E
asi

$$
\begin{aligned}
logL_1(\mu_z, \Sigma_z) &= -\frac{1}{2}\left[log((2\pi)^{n_Z}) + log(\mid \Sigma \mid) + (x-\mu)^T\Sigma^{-1}(x-\mu))\right] \\
&= -\frac{1}{2}\left[n_Z * log((2\pi)) + log(\sigma^2) + \sum_{i=1}^{n_z} log(\lambda_i) + (x-\mu)^T\Sigma^{-1}(x-\mu))\right]
\end{aligned}
$$

Importar librerías

```r
library(readxl)
library(minqa)
```

Importar datos

```r
    # Read the Excel file
    df <- read_excel("C:/Users/HP/Desktop/Desk/jupyter/database/pole.xls
      ", col_names = FALSE)

    # Extract data from columns
    valor1 <- df[[3]]
    valor2 <- df[[4]]
    x <- df[[1]]
    y <- df[[2]]
```

Cálculo de la distancia euclidiana entre dos puntos

```r
    distancia <- function(x1, y1, x2, y2) {
    return(sqrt((x1 - x2)^2 + (y1 - y2)^2))
    }
```

Cálculo de distancias desde un punto específico a todos los demás puntos

```r
    distancias_al_punto <- function(x, y, j) {
  d <- matrix(NA, nrow = length(x), ncol = 2)
  N <- length(x)

  for (i in 1:N) {
    d[i, 1] <- distancia(x[j], y[j], x[i], y[i])
    d[i, 2] <- i
  }

  return(d)
}
```

Determinar puntos dentro o fuera de un circulo

```r
dentro_fuera <- function(punto, r, X, Y) {
  dentro <- vector()
  fuera <- vector()
  N <- length(X)
  Distancia <- distancias_al_punto(X, Y, punto)
```

```
  for (i in 1:N) {
    if (Distancia[i, 1] <= r) {
      dentro <- c(dentro, Distancia[i, 2])
    } else {
      fuera <- c(fuera, Distancia[i, 2])
    }
  }

  return(list(dentro = dentro, fuera = fuera))
}
```

matriz de covarianza

```
    cov <- function(s_1x, s_1y, s_2x, s_2y, cx, cy, tao, a) {
  mini <- min((0.5 + distancia(s_1x, s_1y, cx, cy))^a, (0.5 + distancia(s_2x, s_2y, cx, cy))^a)
  return(exp(-tao * distancia(s_1x, s_1y, s_2x, s_2y) + mini))
}
```

```
    matriz_cov <- function(x, y, cx, cy, tao, a) {
  n <- length(x)
  cov_1 <- matrix(NA, nrow = n, ncol = n)

  for (i in 1:n) {
    for (j in 1:n) {
      cov_1[i, j] <- cov(x[i], y[i], x[j], y[j], cx, cy, tao, a)
    }
  }

  return(cov_1)
}
```

distribución multivariada

```
Distri_Multivariada <- function(sigma, covarianza, x, mu) {
  n <- length(x)
  mu <- as.matrix(mu)
  x <- as.matrix(x)

  # Calcular la descomposici n espectral
  espectral <- eigen(covarianza)

  # Obtener los eigenvalores y eigenvectores
  eigenvalores <- espectral$values
  eigenvectores <- espectral$vectors

  # Calcular el determinante a partir de los eigenvalores
  logdet <- sum(log(eigenvalores))

  inverza <- solve(covarianza)
```

```
  termino_cuadratico <- t(x - mu) %*% inverza %*% (x - mu)

  return(-1/2 * (n*log(sigma^2) + logdet) - 1/(2 * sigma^2) * termino_
      cuadratico)
}
```

```
Algoritmo_Medias <- function(valor){
  n <- length(valor)
  media <- mean(valor)
  MU <- rep(media, n)

  return (MU)
}
```

Selección de elementos en una lista con índices específicos

```

list_selec <- function(valor, puntos) {
  X <- vector()

  for (i in puntos) {
    X <- c(X, valor[i])
  }

  return(X)
}
```

Obtención del Círculo con Máxima Log-Verosimilitud

```
    L_Z_dentro <- function(punto, r, x, y, valor) {
  dentroFuera <- dentro_fuera(punto, r, x, y)
  dentro <- dentroFuera$dentro
  Valor_Dentro <- list_selec(valor, dentro)
  corx <- list_selec(x, dentro)
  cory <- list_selec(y, dentro)
  cx <- x[punto]
  cy <- y[punto]

  # mu <- Algoritmo_Medias(cx, cy, Valor_Dentro, corx, cory)
  mu <- Algoritmo_Medias(Valor_Dentro)
  f <- function(x_1) {
    t <- matriz_cov(corx, cory, cx, cy, x_1[2], x_1[3])

    if  (!is.na(det(t)) && det(t) > 0.001 ) {
      return(-Distri_Multivariada(x_1[1], t, Valor_Dentro, mu))
    }
  }


```

```
  initial_guess <- c(1, 2, -1.5)
  result <- nlminb(initial_guess, lower=c(.0001, .0001,-3),f)


  return(list(-result$objective, result$par))
}
```

```
    Circulos <- function( radio, num_puntos, x, y){
  circulos <- list()
  n <- length(x)
  k <- 0

  for(i in 1:n){
    q <- length(dentro_fuera( i, radio, x, y)$dentro)
    if ( num_puntos <= q ){
      k <- 1+ k
      circulos[[k]] <- i
    }
  }
  return (circulos)
}
```

Obtención del Círculo con Máxima Log-Verosimilitud

```
    L <- function(x, y, valor, radio) {
  Lo <- list()
  m <- 0
  a <- list(radio)
  circulos <- Circulos( radio, 25, x, y)

  for (i in circulos) {
    for (j in a) {
      tryCatch({
        m <- m + 1
        parametros <- L_Z_dentro(i, j, x, y, valor)
        Lo[[m]] <- c(parametros[1], i, j, parametros[[2]][1],parametros
           [[2]][2],parametros[[2]][3])
      }, error = function(e) {
        #cat("Error en la iteraci n", m, ": ", conditionMessage(e), "\n
           ")
      })
    }
  }
  return(Lo)
}
```

```
suppressWarnings(lovero<-L(x, y, valor2,3))


# Convertir la lista a un marco de datos
mi_marco_datos <- do.call(rbind, lovero)
mm   <-as.data.frame(mi_marco_datos)
```

```
# Asignar nombres a las columnas
colnames(mm) <- c("log␣ve", "punto", "radio", "sigma", "tao", "a")

# Extraer los n meros de las listas y convertirlos en un vector
    num rico
 mm$`log ve` <- sapply( mm$`log ve`, function(x) x[[1]])

# Ordenar el data frame por la columna "Edad"
df_ordenado <- mm[order( mm$`log ve`), ]
# Obtener el  ltimo  elemento de la lista
ultimo_elemento <- tail(df_ordenado, n = 1)
ultimo_elemento
```

```
estadistico <- list()
    for(i in 1:20){

    conjunto_de_datos <- rnorm(length(valor2), mean=mean(valor2), sd=sd(
        valor2))

    # Mezclar el conjunto de datos
    conjunto_de_datos <- sample(conjunto_de_datos)

    suppressWarnings(lovero<-L(x, y, conjunto_de_datos,3))

    # Convertir la lista a un marco de datos
    mi_marco_datos <- do.call(rbind, lovero)
    mm   <-as.data.frame(mi_marco_datos)

    # Asignar nombres a las columnas
    colnames(mm) <- c("log␣ve", "punto", "radio", "sigma", "tao", "a")

    # Extraer los n meros de las listas y convertirlos en un vector
        num rico
     mm$`log ve` <- sapply( mm$`log ve`, function(x) x[[1]])

    # Ordenar el data frame por la columna "Edad"
    df_ordenado <- mm[order( mm$`log ve`), ]
    # Obtener el  ltimo  elemento de la lista
    ultimo_elemento <- tail(df_ordenado, n = 1)
    estadistico[i] <-ultimo_elemento$`log ve`

    print(ultimo_elemento)
                }
```