Se tiene lo siguiente para el log-likelihood:

$$\ell_I(\mathbf{p}; \mathcal{X}) = \sum_{b \in \mathcal{B}} \ln P(\mathbf{X}_b = \mathbf{x}_b \mid \mathbf{p})$$

Notar que tiene complejidad polinomial. Por lo tanto, se proponen distintos métodos para obtener el log-likelihood (y en consecuencia, su variación).

Las pruebas se harán con las instancias del paper con 2 candidatos y 2 grupos entre las 20 semillas.

```
library(jsonlite)
library(ggplot2)

# Directorio con las instancias
json_dir <- "/Users/daniel/ecological-inference-elections/instances"
json_files <- list.files(json_dir, pattern = "G2.*I2.*\\.json$", full.names = TRUE)

instances <- list()

# Loop through each filtered JSON file
for (file in json_files) {
    filename <- basename(file)
    match <- regmatches(filename,
        regexec("G([0-9]+)_.*I([0-9]+)_.*seed([0-9]+)", filename))

    if (length(match[[1]]) == 4) {
        G <- as.integer(match[[1]][2])
        I <- as.integer(match[[1]][3])
        seed <- as.integer(match[[1]][4])

        # Load JSON data
        data <- fromJSON(file)

        # Ensure required fields exist
        if (!("X" %in% names(data)) || !("W" %in% names(data)) || !("p" %in% names(data))) {
            next # Skip file if missing data
        }

        # Store the q array
        instances[[seed]] <- data
    }
}
```

Método exacto

Se obtiene el log-likelihood de forma *"cerrada"*

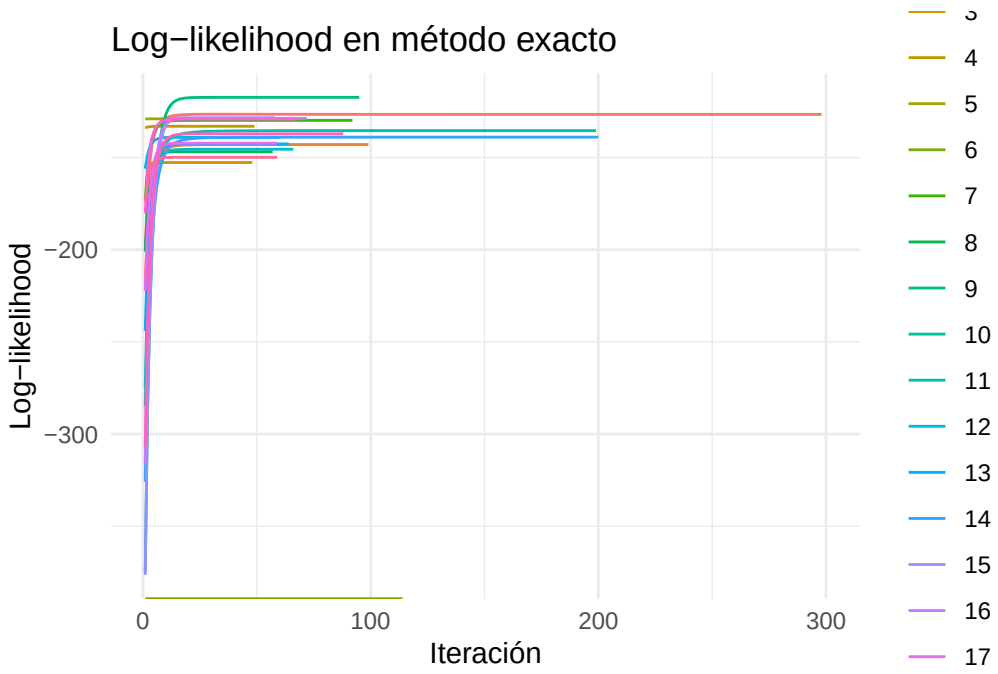
```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(tidyr)
library(fastei)
library(ggplot2)

result_matrix <- matrix(NA, nrow = 300, ncol = 20)

for (i in 1:20) {
    # Corre el algoritmo
    result <- run_em(X = instances[[i]]$X, W = instances[[i]]$W, method = "exact", maxiter =

    # Log-likelihood es un array de tamaño 'maxiter'
    logLik_values <- result$logLik
    len <- length(logLik_values)

    # Si hizo menos de 300 iteraciones, se llenan con `NA`
    result_matrix[1:len, i] <- logLik_values
}
```

```
Maximum iterations reached without convergence.
```

```r
# Pasar los resultados a dataframe para poder graficarlo
df <- as.data.frame(result_matrix)
df$iteracion <- 1:300
df <- tidyr::pivot_longer(df, cols = -iteracion, names_to = "instancia", values_to = "valor")

# Lo convertimos a float
df$instancia <- as.numeric(sub("V", "", df$instancia))

ggplot(df, aes(x = iteracion, y = valor, color = factor(instancia), group = instancia)) +
  geom_line() +
  labs(
    title = "Log-likelihood en método exacto",
    x = "Iteración",
    y = "Log-likelihood",
    color = "Instancia"
  ) +
  theme_minimal()
```



Método multinomial

El método multinomial aproxima el log-likelihood de la siguiente forma:

$$\ell \approx \sum_{b \in \mathcal{B}} \log \left(\sum_{c \in \mathcal{C}} x_{bc} p_{gc} r_{bgc}^{-1} \right)$$

Notar que fijamos $g = 0$ de forma arbitraria, ya que se utiliza Probabilidades Totales.

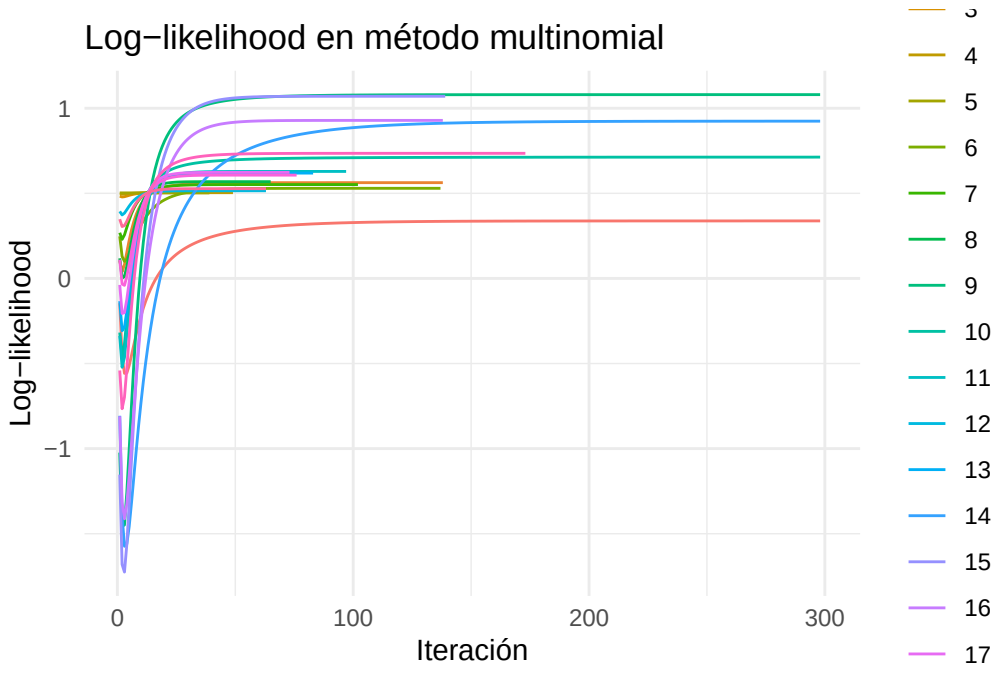
Así, se llega al siguiente resultado:

```
result_matrix <- matrix(NA, nrow = 300, ncol = 20)

for (i in 1:20) {
    # Corre el algoritmo
    result <- run_em(X = instances[[i]]$X, W = instances[[i]]$W, method = "mult", maxiter = 3

    # Log-likelihood es un array de tamaño 'maxiter'
    logLik_values <- result$logLik
    len <- length(logLik_values)

    # Si hizo menos de 300 iteraciones, se llenan con `NA`
    result_matrix[1:len, i] <- logLik_values
}
```

```
Maximum iterations reached without convergence.
Maximum iterations reached without convergence.
Maximum iterations reached without convergence.
Maximum iterations reached without convergence.
```

```
# Pasar los resultados a dataframe para poder graficarlo
df <- as.data.frame(result_matrix)
df$iteracion <- 1:300
df <- tidyr::pivot_longer(df, cols = -iteracion, names_to = "instancia", values_to = "valor")

# Lo convertimos a float
df$instancia <- as.numeric(sub("V", "", df$instancia))

ggplot(df, aes(x = iteracion, y = valor, color = factor(instancia), group = instancia)) +
  geom_line() +
  labs(
    title = "Log-likelihood en método multinomial",
    x = "Iteración",
    y = "Log-likelihood",
    color = "Instancia"
```

```
  ) +
  theme_minimal()
```



CDF de Normal Multivariada

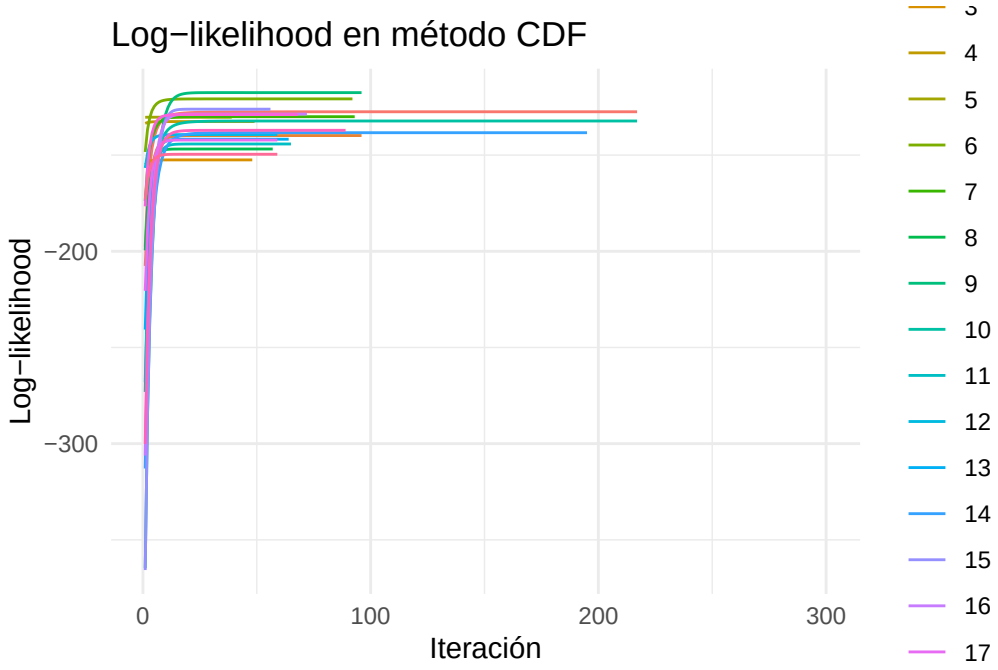
La aproximación se hace de la siguiente forma:

$$\ell \approx \sum_{b \in \mathcal{B}} = \log \left(\sum_{c \in \mathcal{C}} F_{bg}(\mathcal{A}_{bc}(\mathbf{x}_b)) \cdot p_{gc} \right)$$

De la misma forma anterior, se fija un grupo arbitrariamente.

```
result_matrix <- matrix(NA, nrow = 300, ncol = 20)

for (i in 1:20) {
    # Corre el algoritmo
    result <- run_em(X = instances[[i]]$X, W = instances[[i]]$W, method = "mvn_cdf", maxiter

    # Log-likelihood es un array de tamaño 'maxiter'
    logLik_values <- result$logLik
```

```
    len <- length(logLik_values)

    # Si hizo menos de 300 iteraciones, se llenan con `NA`
    result_matrix[1:len, i] <- logLik_values
}

# Pasar los resultados a dataframe para poder graficarlo
df <- as.data.frame(result_matrix)
df$iteracion <- 1:300
df <- tidyr::pivot_longer(df, cols = -iteracion, names_to = "instancia", values_to = "valor")

# Lo convertimos a float
df$instancia <- as.numeric(sub("V", "", df$instancia))

ggplot(df, aes(x = iteracion, y = valor, color = factor(instancia), group = instancia)) +
  geom_line() +
  labs(
    title = "Log-likelihood en método CDF",
    x = "Iteración",
    y = "Log-likelihood",
    color = "Instancia"
  ) +
  theme_minimal()
```



PDF de Normal Multivariada

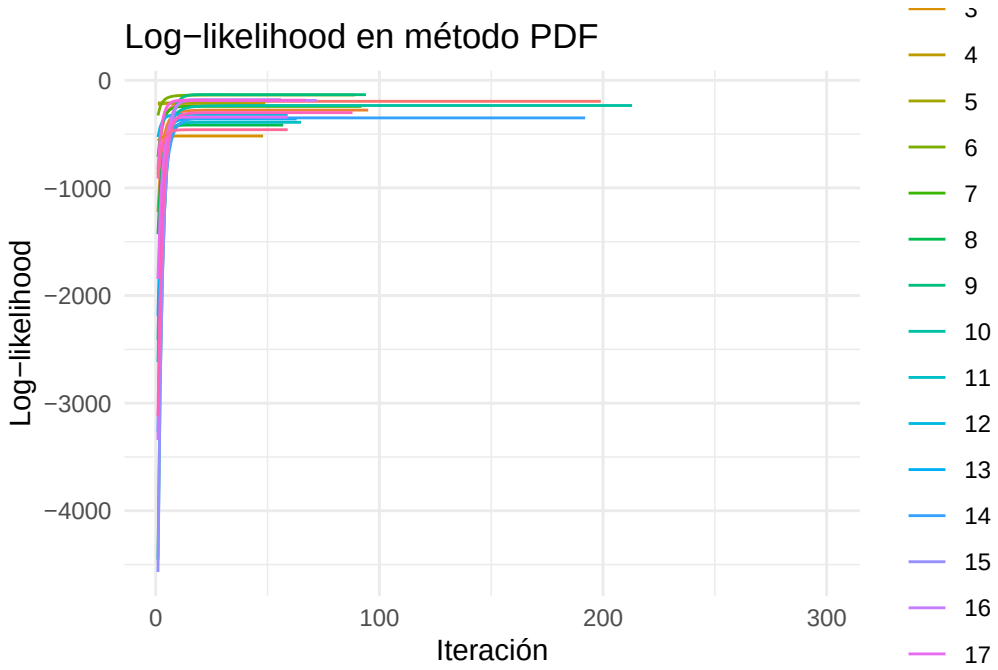
En este caso, la aproximación es la siguiente:

$$\ell \approx= \sum_{b \in \mathcal{B}} \log \left(\sum_{c \in \mathcal{C}} \sqrt{(2\pi)^{C-1} |\Sigma_b^g| \exp(-0.5 \cdot (\mathbf{x}_b - \mathbf{e}_c - {}_b^g)^T (\Sigma_b^g)^{-1}) (\mathbf{x}_b - \mathbf{e}_c - {}_b^g) \cdot p_{gc}} \right)$$

Notar que, se tendría que calcular de forma adicional el término de la raíz. Sin embargo, no hay tanta pérdida de eficiencia ya que al tener la inversa de la matriz de covarianza (que es positiva y simétrica), bastaría con multiplicar los elementos de su diagonal y sacar su inverso multiplicativo (ya que es la matriz inversa).

```
result_matrix <- matrix(NA, nrow = 300, ncol = 20)

for (i in 1:20) {
    # Corre el algoritmo
    result <- run_em(X = instances[[i]]$X, W = instances[[i]]$W, method = "mvn_pdf", maxiter

    # Log-likelihood es un array de tamaño 'maxiter'
    logLik_values <- result$logLik
```

```r
    len <- length(logLik_values)

    # Si hizo menos de 300 iteraciones, se llenan con `NA`
    result_matrix[1:len, i] <- logLik_values
}

# Pasar los resultados a dataframe para poder graficarlo
df <- as.data.frame(result_matrix)
df$iteracion <- 1:300
df <- tidyr::pivot_longer(df, cols = -iteracion, names_to = "instancia", values_to = "valor")

# Lo convertimos a float
df$instancia <- as.numeric(sub("V", "", df$instancia))

ggplot(df, aes(x = iteracion, y = valor, color = factor(instancia), group = instancia)) +
  geom_line() +
  labs(
    title = "Log-likelihood en método PDF",
    x = "Iteración",
    y = "Log-likelihood",
    color = "Instancia"
  ) +
  theme_minimal()
```



Hit and Run

Por último, el método `hnr` es el que es más costoso computacionalmente ya que no *"arrastra"* tantos cálculos. En específico, se tiene lo siguiente:

$$\ell = Q(\mathbf{p}\ ;\ \mathbf{p}^{\text{old}}) - \mathbb{E}[\ln P(\mathcal{Z} \mid \mathcal{X}\ ;\ \mathbf{p}^{\text{old}}) \mid \mathcal{X}\ ;\ \mathbf{p}^{(\text{old})}]$$

El valor $\mathbf{Q}$ es cerrado, donde

$$Q(\mathbf{p}\ ;\ \mathbf{p}^{\text{old}}) = \sum_{b\in\mathcal{B},\ g\in\mathcal{G}} w_{bg} \sum_{c\in\mathcal{C}} q_{bgc} \ln(p_{gc})$$

$$+ \sum_{b\in\mathcal{B}\ g\in\mathcal{G}} a_{w_{bg}} - \sum_{b\in\mathcal{B},\ g\in\mathcal{G}\ c\in\mathcal{C}} \sum_{k=1}^{w_{bg}} a_k \binom{w_{bg}}{k} q_{bgc}^{k} (1-q_{bgc})^{w_{bg}-k}$$

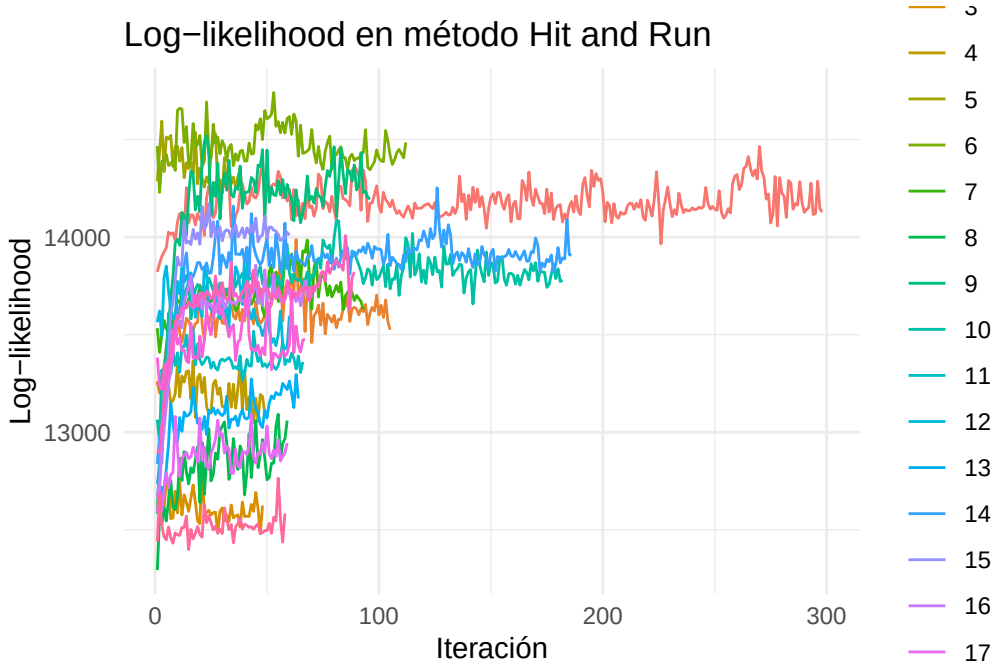
Donde a_k es la función logaritmo de la función gamma desplazado en una unidad.

Por el otro lado, la esperanza se aproxima con *importance sampling*:

$$\mathbb{E}[\ln P(\mathcal{Z} \mid \mathcal{X}\ ;\ \mathbf{p}^{(old)}) \mid \mathcal{X}\ ;\ \mathbf{p}^{\text{old}}] \approx \sum_{b\in\mathcal{B}} \sum_{z\in\mathcal{S}_b} g_b(\mathbf{z},\ \mathbf{p}^{\text{old}}) \ln(g_b(\mathbf{z}\ ,\ \mathbf{p}^{\text{old}}))$$

Notar que este método es el que más tiempo toma al ocupar 1000 samples y un step-size de 3000. Casi todo el computo se va en simular los samples.

```
result_matrix <- matrix(NA, nrow = 300, ncol = 20)

for (i in 1:20) {
    # Corre el algoritmo
    result <- run_em(X = instances[[i]]$X, W = instances[[i]]$W, method = "hnr", maxiter = 3

    # Log-likelihood es un array de tamaño 'maxiter'
    logLik_values <- result$logLik
    len <- length(logLik_values)

    # Si hizo menos de 300 iteraciones, se llenan con `NA`
    result_matrix[1:len, i] <- logLik_values
}

# Pasar los resultados a dataframe para poder graficarlo
df <- as.data.frame(result_matrix)
df$iteracion <- 1:300
df <- tidyr::pivot_longer(df, cols = -iteracion, names_to = "instancia", values_to = "valor"

# Lo convertimos a float
df$instancia <- as.numeric(sub("V", "", df$instancia))

ggplot(df, aes(x = iteracion, y = valor, color = factor(instancia), group = instancia)) +
  geom_line() +
  labs(
    title = "Log-likelihood en método Hit and Run",
    x = "Iteración",
    y = "Log-likelihood",
    color = "Instancia"
  ) +
  theme_minimal()
```


Log−likelihood en método Hit and Run
Log−likelihood
14000
13000
0
100
200
300
Iteración
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17