Ejercicio 1

Apartado (a)

Para todo el ejercicio nos basaremos en la propiedad de las covarianzas y su relación con las variables. la covarianza es una medida de la dependencia entre las variables aleatorias. Por lo tanto, la covarianza de dos variables aleatorias independientes es 0. Esta propiedad es fácilmente demostrable. Suponiendo

$$X_i, X_j$$

como dos variables independientes:

La esperanza de un producto de variables aleatorias es el producto de las esperanzas de las variables aleatorias.

$$E(X_i X_j) = E(X_i)E(X_j)$$

Que se puede escribir como:

$$cov(X_i, X_j) = E(X_i X_j) - E(X_i)E(X_j) = 0$$

Demostración de (a.1)

El estado $x(j)$ en el instante j es independiente del ru
(siendo $j, k \geq 0$ y $j \leq k$). La ecuación de estado

$$x(k+1) = \Phi(k+1, k)x(k) + \Gamma($$

muestra que $x(k+1)$ es una combinación lineal de $x($
fundamental de la covarianza es que la covarianza ent
cero (visto previamente)

No, por hipótesis lo que es independiente es x_0 y los ruidos. Entonces no puedes usar eso. Tienes que expresar x(k+1) recursivamente hasta llegar a expresarlo en función de x_0 y al calcular la esperanza del producto te sale en función de x_0 y w que ya si son independientes. Razonamiento similar tienes que hacer para las cuatro propiedades que se piden que demuestres.

Por lo tanto, la covarianza entre $x(j)$ y $w(k)$ es $E[x(j)w^T(k)] = 0$ para $j, k \geq 0$ y $j \leq k$, ya que $x(j)$ no depende de $w(k)$ en virtud del modelo de espacio de estados. Esto demuestra que $E[x(j)w^T(k)] = 0$:

$$E[x(j)w^T(k)] = cov(x(j), w(k)) = 0$$

Demostración de (a.2)

Similarmente, considerando la ecuación de observaciones $z(k) = H(k)x(k) + v(k)$, observamos que $z(j)$ en el instante j es independiente del ruido aditivo $w(k)$ en el instante k, ya que $z(j)$ es una combinación lineal de $x(j)$ y $v(j)$, y $w(k)$ y es un ruido blanco gaussiano. Por lo tanto, se cumple:

$$E[z(j)w^T(k)] = cov(x(j), w(k)) = 0$$

Demostración de (a.3)

De nuevo, como en el caso anterior, $x(j), v(k)$ son variables aleatorias independientes, por lo que:

$$E[x(j)v^T(k)] = E[x(j)]E[v^T(k)]$$

Si desarrollamos la expresión anterior de $x(j)$, y tenemos en cuenta que la media de $x(j)$ y $v(k)$ es cero:

$$cov\big(x(j), v(k)\big) = E[x(j)v^T(k)] - E[x(j)]E[v^T(k)]$$

Dado que $E[x(j)v^T(k)] = cov\big(x(j), v(k)\big) + E[x(j)]E[v^T(k)]$:

$$cov\big(x(j), v(k)\big) = cov\big(x(j), v(k)\big) + E[x(j)]E[v^T(k)] - E[x(j)]E[v^T(k)]$$

Al estar en ambos términos de la ecuación, se simplifica:

$$cov\big(x(j), v(k)\big) = 0$$

Y por lo tanto, se concluye que $E[x(j)v^T(k)] = 0$.

Demostración de (a.4)

En este caso, $z(j), v(k)$ son variables aleatorias independientes solo si se cumple que $j \neq k$. Si $j = k$, entonces $z(j), v(k)$ no son independientes, ya que $z(j)$ depende de $x(j)$ que a su vez depende de $v(k)$ a través del término de ruido $w(k)$.

La esperanza de $z(j)$es cero, ya que $z(j)$ es una combinación lineal de variables aleatorias gaussianas con media cero. La esperanza de $v(k)$ es también cero, ya que $v(k)$ es un ruido blanco gaussiano centrado. Por lo que:

$$cov\big(z(j), v(k)\big) = E[z(j)v^T(k)] - E[z(j)]E[v^T(k)]$$

$$cov\big(z(j), v(k)\big) = E[z(j)]E[v^T(k)] - E[z(j)]E[v^T(k)]$$

$$cov\big(z(j), v(k)\big) = 0 \text{ si solo si } j \neq k$$

La dependencia de $x(j)$ de $w(k)$ a través de la ecuación de estado no afecta la independencia entre $z(j)$ y $v(k)$ cuando $j \neq k$, ya que $w(k)$ es independiente de $v(k)$ y, por lo tanto, de $z(j)$ en esos casos.

Apartado (b)

La relación que se quiere demostrar es:

$$\hat{x}(k/j) = \hat{x}(k/j-1) + K(k,j)[z(j) - \hat{z}(j/j-1)], \quad j > 0$$

Primero, expresamos $\hat{x}(k/j)$ y $\hat{z}(j/j-1)$ como funciones lineales de las observaciones correspondientes:

$$\hat{x}(k/j) = X_j \sum_{i=0}^{j} K\,(k,i)z(i) + K(k,j)\hat{z}(j/j-1)$$

$$\hat{z}(j/j-1) = H(j)\hat{x}(j/j-1)$$

Ahora se debe aplicar el Lema de Proyección Ortogonal (LPO) para obtener las ecuaciones de Wiener-Hopf. La idea clave es que el error de predicción es ortogonal al espacio de observaciones pasadas, por lo que la covarianza se simplifica y se convierte en una combinación lineal:

$$E[x(k)z^T(\sigma)] = \sum_{i=0}^{k} K\,(k,i)E[z(i)z^T(\sigma)],\ \sigma = 0, \dots,$$

$$E[z(k)z^T(\sigma)] = \sum_{i=0}^{k-1} F\,(k,i)E[z(i)z^T(\sigma)],\ \sigma = 0, \dots, k-1$$

Estas ecuaciones son las ecuaciones de Wiener-Hopf que describen la relación entre las observaciones y los filtros óptimos. Ahora, para $\sigma = 0, \dots, k-1$, podemos expresar $E[x(k)z^T(\sigma)]$ como:

$$E[x(k)z^T(\sigma)] = \sum_{i=0}^{k-1} \big(K(k,i) + K(k,k)F(k,i)\big)\, E[z(i)z^T(\sigma)]$$

Te comenté que era similar a la demostración hecha en los apuntes para el filtro, pero lo tienes que ajustar para lo que estás demostrando.

Y utilizando esta expresión para $E[x(k)z^T(\sigma)]$ en la relación $\hat{x}(k/j)$ y $\hat{z}(j$ llega a la expresión buscada:

$$\hat{x}(k/j) = \hat{x}(k/j-1) + K(k,j)[z(j) - \hat{z}(j/j-1)]$$

Apartado (c)

Dado el modelo de espacio de estados:

$$\begin{aligned} x(k+1) &= \Phi(k+1,k)x(k) + \Gamma(k+1,k)w(k) \quad \text{para } k \geq 0, \quad x(0) = x_0 \\ z(k) &= H(k)x(k) + v(k) \quad \text{para } k \geq 0 \end{aligned}$$

La predicción de la observación en el instante k basada en la información hasta $k-1$ se obtiene aplicando la ecuación de estado a la estimación del es

$$\hat{z}(k/k-1) = H(k)\hat{x}(k/k-1)$$

No, este no es el razonamiento. Mezclas el predictor de la observación, \widehat{z}(k/k-1), y el predictor de la estimación del estado, \widehat{x}(k/k-1). Aquí se pide que demuestres que el predictor de la observación, \widehat{z}(k/k-1), tiene la expresión H(k)\widehat{x}(k/k-1)

Esta ecuación se deriva utilizando el filtrado óptimo y la ecuació que proporciona la mejor estimación lineal de $x(k)$ basada en to anteriores $z(0), \dots, z(k-1)$. La relación fundamental es:

$$\hat{x}(k/k-1) = \Phi(k,k-1)\hat{x}(k-1/k-1)$$

Donde $\Phi(k, k-1)$ es la matriz de transición entre los instantes $k-1$ y k. Al aplicar esta relación sucesivamente, se llega a la ecuación de la predicción de la observación.

Para determinar el proceso de innovación, $\tilde{z}(k/k-1)$, primero, recordemos que la innovación es la diferencia entre la observación actual y la predicción de la observación. En este contexto, la observación es $z(k)$, y la predicción se obtiene a partir de la estimación del estado anterior.

La innovación se define como:

$$\tilde{z}(k/k-1) = z(k) - \hat{z}(k/k-1)$$

Entonces, el proceso de innovación es:

$$\tilde{z}(k/k-1) = z(k) - H(k)\hat{x}(k/k-1)$$

Ahora, demostremos que este proceso de innovación es gaussiano, centrado y blanco.

1. **Gaussiano:** Sabemos que $\tilde{z}(k)$ y $\hat{x}(k/k-1)$ son gaussianos, y la diferencia de dos variables gaussianas también es gaussiana. Por lo tanto, $\tilde{z}(k/k-1)$ es gaussiano.

2. **Centrado:** La linealidad de la esperanza condicional implica que la diferencia $z(k) - H(k)\hat{x}(k/k-1)$ sigue siendo centrada, ya que $E[z(k)] - H(k)E[\hat{x}(k/k-1)] = 0$

3. **Blanco:** Para demostrar que $\tilde{z}(k/k-1)$ es un proceso blanco, usaremos la propiedad de que su esperanza debe ser cero en instantes de tiempo distintos. Entonces, calculamos la esperanza para $\tilde{z}(k/k-1)$ y para $\tilde{z}(j/j-1)$:

   Para $\tilde{z}(k/k-1)$:

$$\tilde{z}(k/k-1) = E[z(k) - H(k)\hat{x}(k$$
$$= E[z(k)] - E[H(k)\hat{x}(k/k -$$
$$= H(k)E[x(k)] - H(k)E[\hat{x}(k/$$

No, tienes que probar que E[\widetilde{z}(k/k-1)\widetilde{z}(j/j-1)^T]=0 para todo k<j. Esto se prueba teniendo en cuenta la ortogonalidad de la innovación y las obsevaciones.

Considerando el error de predicción $E[\hat{x}(k/k-1)] = E[x(k)]$:

$$\tilde{z}(k/k-1) = H(k)\big[E[x(k)] - E[x(k)]\big]$$

$$\tilde{z}(k/k-1) = 0$$

Lo que demuestra que $\tilde{z}(k/k-1)$ tiene una esperanza cero.

Por otra parte, para $\tilde{z}(j/j-1)$:

$$E[\tilde{z}(j/j-1)] = E[z(j) - H(j)\hat{x}(j/j-1)] = H(j)E[x(j)] - H(j)E[\hat{x}(j/j-1)]$$
$$= H(j)\big[E[x(j)] - E[x(j)]\big] = 0$$

Igualmente, se obtiene que $\tilde{z}(j/j-1)$ tiene una esperanza cero y por lo tanto, se demuestra que el proceso es blanco.

El cálculo de la matriz de covarianzas $\Pi(k)$ es:

$$\Pi(k) = H(k)P(k/k-1)H^T(k) + R(k)$$

Esta expresión representa la covarianza de $\tilde{z}(k/k-1)$ en el instante k.

Apartado (d)

El algoritmo de filtrado de Kalman es un procedimiento recursivo que se utiliza para estimar el estado de un sistema dinámico en tiempo real, basándose en las mediciones disponibles y en el modelo dinámico del sistema. El algoritmo se puede dividir en dos pasos principales: la predicción y la actualización.

Paso de Predicción: 1. **Predicción del Estado Estimado:**

$$\hat{x}(k|k-1) = \Phi(k,k-1)\hat{x}(k-1|k-1)$$

Donde $\hat{x}(k|k-1)$ es la predicción del estado en el tiempo k basada en la información hasta $k-1$, $\Phi(k,k-1)$ es la matriz de transición del sistema, y $\hat{x}(k-1|k-1)$ es el estado estimado en el tiempo $k-1$.

Esta predicción del estado se obtiene directamente aplicando la matriz de transición del sistema $\Phi(k,k-1)$ al estado estimado anterior $\hat{x}(k-1|k-1)$. La demostración es inmediata al multiplicar la matriz por el estado anterior.

2. **Predicción de la Covarianza del Error de Estimación:**

$$P(k|k-1) = \Phi(k,k-1)P(k-1|k-1)\Phi(k,k-1)^T + Q(k)$$

Donde $P(k|k-1)$ es la predicción de la covarianza del error de estimación en el tiempo k, $P(k-1|k-1)$ es la covarianza del error de estimación en el tiempo $k-1$, y $Q(k)$ es la matriz de covarianza del ruido de proceso en el tiempo k.

La predicción de la covarianza del error se deriva aplicando la matriz de transición y la covarianza del error anterior. La adición de $Q(k)$ se aplica para tener en cuenta la incertidumbre del ruido de proceso en el tiempo k.

Paso de Actualización: 1. **Cálculo de la Ganancia de Kalman:**

$$K(k) = P(k|k-1)H(k)^T[H(k)P(k|k-1)H(k)^T + R(k)]^{-1}$$

Donde $K(k)$ es la ganancia de Kalman en el tiempo k, $H(k)$ es la matriz de observación en el tiempo k, y $R(k)$ es la matriz de covarianza del ruido de medición en el tiempo k.

2. **Actualización del Estado Estimado:**

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)[z(k) - H(k)\hat{x}(k|k-1)]$$

Donde $\hat{x}(k|k)$ es el estado estimado en el tiempo k basado en la información hasta k, $z(k)$ es la medición en el tiempo k, y $H(k)$ es la matriz de observación en el tiempo k.

Demostración: La actualización del estado se deriva aplicando la ganancia de Kalman a la diferencia entre la medición real $z(k)$ y la predicción de la medición $H(k)\hat{x}(k|k-1)$.

Tienes que demostrar todas las expresiones del filtro de Kalman.

3. **Actualización de la Covarianza del Error de Estimación:**

$$P(k|k) = [I - K(k)H(k)]P(k|k-1)$$

Donde $P(k|k)$ es la covarianza del error de estimación en el tiempo k, I es la matriz de identidad, y $K(k)$ es la ganancia de Kalman en el tiempo k.

Apartado (e)

El algoritmo de suavizamiento en el filtro de Kalman se utiliza para mejorar las estimaciones de estado retrospectivamente, después de haber obtenido nuevas mediciones. El objetivo es actualizar las estimaciones pasadas a la luz de información adicional. A continuación, se presenta el algoritmo de suavizamiento punto fijo y la derivación de las matrices de covarianza de los errores de suavizamiento.

Consideremos el sistema definido en la Sección 1, y supongamos que el filtro óptimo y la matriz de covarianzas del error de filtrado son conocidos en cualquier instante de tiempo. El algoritmo de suavizamiento punto fijo proporciona el estimador óptimo del estado T(k) para un instante fijo k basado en el conjunto de observaciones {z(0), z(1), …, z(k)} a partir del estimador basado en k, donde j = k + 1, k + 2, …, siendo así un algoritmo recursivo hacia adelante.

Tienes que demostrar cómo se obtienen las expresiones del algoritmo de suavizamiento.

1. El suavizador punto fijo óptimo, $\hat{x}(k/j)$ viene dado por la relación
$\hat{x}(k/j) = \hat{x}(k/j-1) + K(k,j)[z(j) - H(j)\phi(j,j-1)\hat{x}(j-1)/(j-$
donde j = k + 1, k + 2, …, y la condición inicial es el filtro, $\hat{x}(k/k)$.

Esta relación se puede demostrar a partir de la ecuación de predicción del filtro de Kalman:

$$\hat{x}(k/k-1) = \Phi(k,k-1)\hat{x}(k-1/k-1)$$

Si aplicamos la corrección utilizando la medición $z(j)$ en el instante j:

$$\hat{x}(k/j) = \hat{x}(k/k-1) + K(k,j)[z(j) - H(j)\Phi(j,j-1)\hat{x}(j-1)]$$

Donde $K(k,j)$ es la ganancia de suavizado. Ahora, utilizamos recursivamente la misma relación para $\hat{x}(j-1/j-2)$, $\hat{x}(j-2/j-3)$, y así sucesivamente hasta llegar a $\hat{x}(k/j-1)$. Esto nos lleva a la forma final de la ecuación.

2. $K(k,j)$, la matriz de ganancia del suavizador punto fijo, verifica
$K(k,j) = L(k,j-1)\Phi^T(j,j-1)H^T(j)(H(j)P(j/j-1)H^T(j) + R(j)^{-1})$

donde $L(k,j)$ es obtenida recursivamente por la siguiente relación:

$$L(k,j) = L(k,j-1)\Phi^T(j,j-1)[I - K(j)H(j)]^T, \quad j = k+1, k+2, \ldots$$
$$L(k,k) = P(k/k)$$

Esta ecuación se deriva considerando la forma de la ganancia de suavizado en el algoritmo de suavizamiento punto fijo. La ganancia $K(k,j)$ se define en función de $L(k,j-1)$ y otros términos. Para demostrar esto, se puede utilizar una relación de recursión para $L(k,j)$.

3. La matriz de covarianzas del error, $P(k/j)$, verifica

$$P(k/j) = P(k/j-1) - K(k,j)H(j)P(j/j-1)H^T(j) + R(j)K^T(k,j)$$

donde j = k + 1, k + 2, … y la condición inicial es la matriz de covarianzas del error de filtrado $P(k/k)$.

La demostración de esta ecuación se puede realizar utilizando las propiedades del filtro de Kalman. La matriz de covarianzas del error $P(k/j)$ se actualiza basándose en la predicción $P(k/j-1)$ y la ganancia de suavizado $K(k,j)$, teniendo en cuenta la influencia de la medición $z(j)$ en el instante j. La actualización de la matriz de covarianzas sigue la estructura del filtro de Kalman.

El algoritmo de suavizamiento punto fijo utiliza la información futura para mejorar las estimaciones pasadas. La derivación de las matrices de covarianza de los errores de suavizamiento se basa en la propagación de las covarianzas a lo largo del tiempo y el uso de la ganancia de suavizamiento $J(k)$. Este algoritmo es útil cuando se dispone de información futura y se desea mejorar las estimaciones retrospectivamente.

Apartado (f)

Primero, para el modelo de espacio de estados dado:

$$x(k+1) = 0.95x(k) + w(k), \quad k \geq 0; \quad x(0) = x_0$$
$$z(k) = x(k) + v(k), \quad k \geq 0$$

Donde:

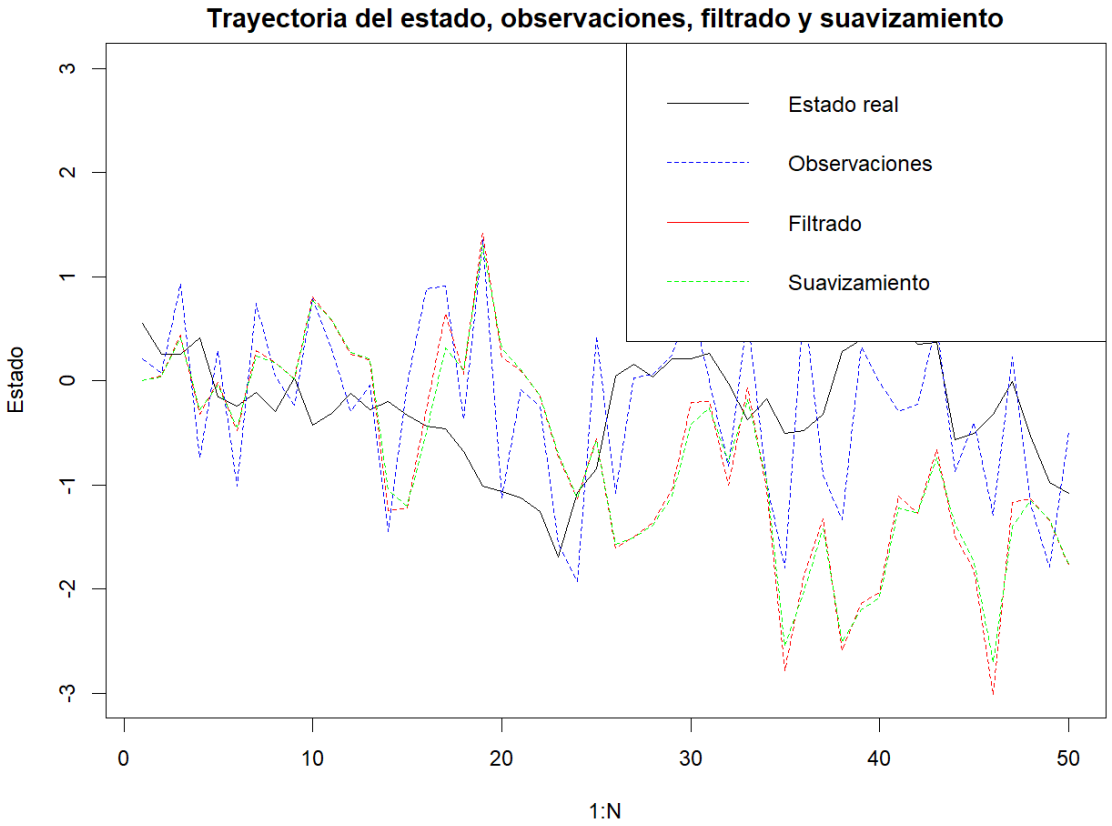
- x_0 *es una variable gaussiana con media cero y varianza* $P_0 = 1$.
- $\{w(k); k \geq 0\}$ *es una sucesión blanca gaussiana, centrada, con varianzas* $Q(k) = 0.1$ *para* $\forall k \geq 0$.
- $\{v(k); k \geq 0\}$ *es un proceso de ruido blanco gaussiano, centrado, con varianzas* $R(k) = 0.5$ *para* $\forall k \geq 0$.

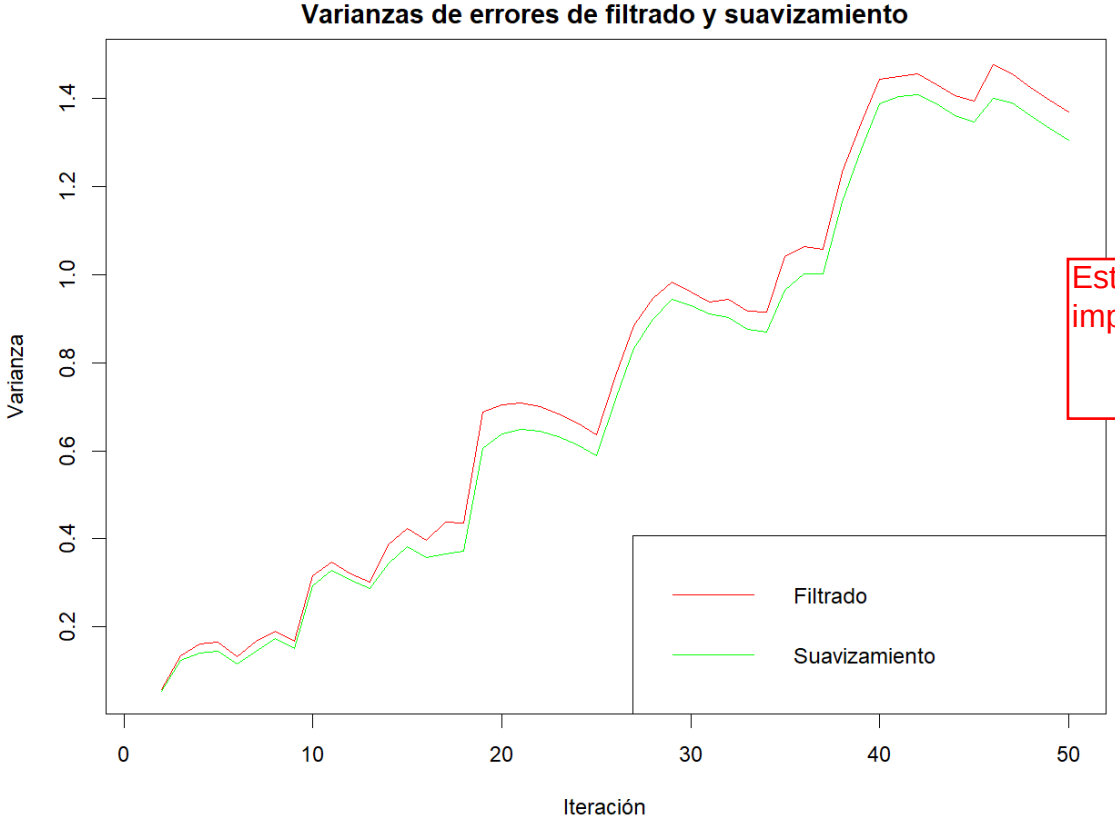
```
# Definir parámetros del sistema
n <- 2       # Dimensión del estado
N <- 50      # Número de iteraciones
x0 <- 0      # Media del estado inicial
P0 <- 1      # Varianza del estado inicial
```

```
phi <- 0.95  # Coeficiente del modelo del sistema
Q <- 0.1     # Varianza del ruido de proceso
p <- 0.5     # Probabilidad del ruido multiplicativo
R <- 0.5     # Varianza del ruido de medida

# Inicializar vectores y matrices
x <- numeric(N)  # Estado real
z <- numeric(N)  # Observaciones
x_filt <- numeric(N)  # Estimaciones de filtrado
x_smooth <- numeric(N)  # Estimaciones de suavizamiento
P_filt <- numeric(N)  # Varianzas de filtrado
P_smooth <- numeric(N)  # Varianzas de suavizamiento
K <- numeric(N)  # Ganancia de Kalman
J <- numeric(N)  # Ganancia de suavizamiento

# Generar observaciones inciertas
set.seed(50)
w <- rnorm(N, mean = 0, sd = sqrt(Q))
gamma <- rbinom(N, size = 1, prob = p)
v <- rnorm(N, mean = 0, sd = sqrt(R))
x[1] <- rnorm(1)
z[1] <- x[1] + sqrt(R) * rnorm(1)
x_filt[1] <- 0 # Estimación inicial
P_filt[1] <- 1

# Aplicar el ciclo computacional del filtro y el suavizador
for (k in 2:N) {
  # Modelo del sistema
  x[k] <- phi * x[k-1] + w[k]
  z[k] <- gamma[k] * x[k] + v[k]

  # Filtro de Kalman
  # Predicción
  x_filt[k] <- phi * x_filt[k-1]
  P_filt[k] <- phi^2 * P_filt[k-1] + Q

  # Actualización
  K[k] <- P_filt[k] / (P_filt[k] + gamma[k]^2 * R)
  x_filt[k] <- x_filt[k] + K[k] * (z[k] - gamma[k] * x_filt[k])
  P_filt[k] <- (1 - K[k] * gamma[k]) * P_filt[k]

  # Suavizador
  if (k >= 2) {
    J[k-1] <- P_filt[k-1] * phi * P_filt[k]
    x_smooth[k] <- x_filt[k] + J[k-1] * (x_smooth[k-1] - phi * x_filt[k])
    P_smooth[k] <- P_filt[k] + J[k-1] * (P_smooth[k-1] - P_filt[k]) *
J[k-1]

    # Cálculo de errores
```

Quién es J

```
    e_filt <- x_filt - x
    e_suav <- x_smooth - x

    # Cálculo de varianzas
    e_filt2 <- rep(0, N)
    e_suav2 <- rep(0, N)

    for (n in 1:N) {
      e_filt2[n] <- var(e_filt[1:n])
      e_suav2[n] <- var(e_suav[1:n])
    }
  }
}

# Gráfica 1: Trayectoria del estado, observaciones, estimaciones de
filtrado y suavizamiento
par(mfrow = c(1, 1))
par(mar = c(4, 4, 2, 1))  # Ajustar los márgenes

plot(1:N, x, type = "l", col = "black", ylim = c(-3, 3), ylab = "Estado",
     main = "Trayectoria del estado, observaciones, filtrado y
suavizamiento")
lines(1:N, z, col = "blue", lty = 2)
lines(1:N, x_filt, col = "red", lty = 2)
lines(1:N, x_smooth, col = "green", lty = 2)
legend("topright", legend = c("Estado real", "Observaciones", "Filtrado",
"Suavizamiento"),
       col = c("black", "blue", "red", "green"), lty = 1:2)

# Gráfica 2: Varianzas de los errores de filtrado y suavizamiento
par(mar = c(4, 4, 2, 1))  # Ajustar los márgenes

plot(1:N, e_filt2, type = "l", col = "red", ylab = "Varianza",
     xlab = "Iteración", main = "Varianzas de errores de filtrado y
suavizamiento")
lines(1:N, e_suav2, col = "green")
legend("bottomright", legend = c("Filtrado", "Suavizamiento"), col =
c("red", "green"), lty = 1)
```





Estas varianzas son imposibles

Apartado (g)

Se utiliza el código del apartado anterior, adaptándolo para los tres casos que se pide (N=1, N=2, N=4)

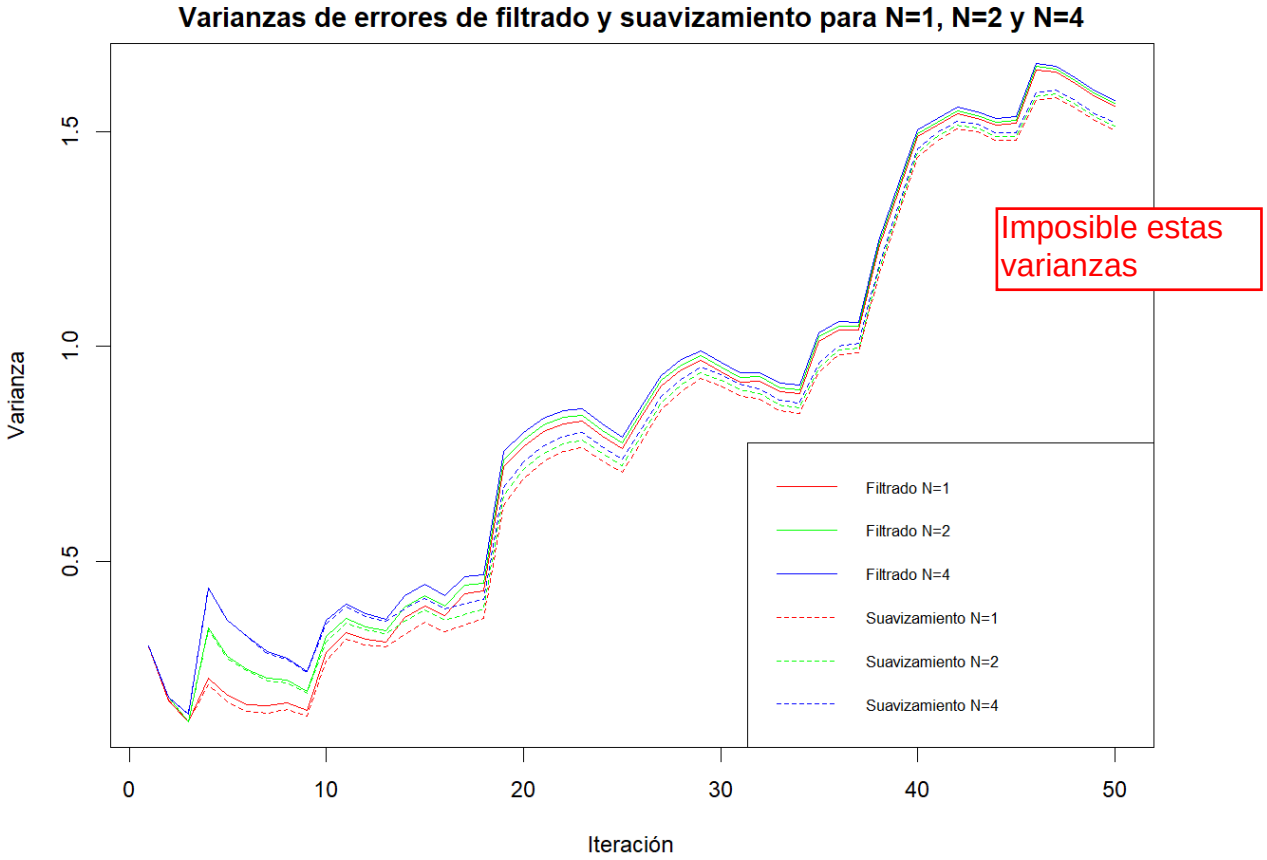
```
# Definir parámetros del sistema
n <- 2        # Dimensión del estado
N <- 50       # Número de iteraciones
x0 <- 0       # Media del estado inicial
P0 <- 1       # Varianza del estado inicial
phi <- 0.95   # Coeficiente del modelo del sistema
Q <- 0.1      # Varianza del ruido de proceso
p <- 0.5      # Probabilidad del ruido multiplicativo
R <- 0.5      # Varianza del ruido de medida

# Inicializar vectores y matrices
x <- numeric(N)  # Estado real
z <- numeric(N)  # Observaciones
x_filt1 <- numeric(N)  # Estimaciones de filtrado para N=1
x_filt2 <- numeric(N)  # Estimaciones de filtrado para N=2
x_filt4 <- numeric(N)  # Estimaciones de filtrado para N=4
x_smooth1 <- numeric(N)  # Estimaciones de suavizamiento para N=1
x_smooth2 <- numeric(N)  # Estimaciones de suavizamiento para N=2
x_smooth4 <- numeric(N)  # Estimaciones de suavizamiento para N=4
P_filt1 <- numeric(N)  # Varianzas de filtrado para N=1
P_filt2 <- numeric(N)  # Varianzas de filtrado para N=2
P_filt4 <- numeric(N)  # Varianzas de filtrado para N=4
P_smooth1 <- numeric(N)  # Varianzas de suavizamiento para N=1
P_smooth2 <- numeric(N)  # Varianzas de suavizamiento para N=2
P_smooth4 <- numeric(N)  # Varianzas de suavizamiento para N=4
K1 <- numeric(N)  # Ganancia de Kalman para N=1
K2 <- numeric(N)  # Ganancia de Kalman para N=2
K4 <- numeric(N)  # Ganancia de Kalman para N=4
J1 <- numeric(N)  # Ganancia de suavizamiento para N=1
J2 <- numeric(N)  # Ganancia de suavizamiento para N=2
J4 <- numeric(N)  # Ganancia de suavizamiento para N=4

# Generar observaciones inciertas
set.seed(50)
w <- rnorm(N, mean = 0, sd = sqrt(Q))
gamma <- rbinom(N, size = 1, prob = p)
v <- rnorm(N, mean = 0, sd = sqrt(R))
x[1] <- rnorm(1)
z[1] <- x[1] + sqrt(R) * rnorm(1)
x_filt1[1] <- 0  # Estimación inicial para N=1
x_filt2[1] <- 0  # Estimación inicial para N=2
x_filt4[1] <- 0  # Estimación inicial para N=4
P_filt1[1] <- 1
P_filt2[1] <- 1
```

```
P_filt4[1] <- 1
x_smooth1[1] <- 0  # Estimación inicial para N=1
x_smooth2[1] <- 0  # Estimación inicial para N=2
x_smooth4[1] <- 0  # Estimación inicial para N=4
P_smooth1[1] <- 1
P_smooth2[1] <- 1
P_smooth4[1] <- 1

# Aplicar el ciclo computacional del filtro y el suavizador para N=1
for (k in 2:N) {
  # Modelo del sistema
  x[k] <- phi * x[k-1] + w[k]
  z[k] <- gamma[k] * x[k] + v[k]

  # Filtro de Kalman
  # Predicción
  x_filt1[k] <- phi * x_filt1[k-1]
  P_filt1[k] <- phi^2 * P_filt1[k-1] + Q

  # Actualización
  K1[k] <- P_filt1[k] / (P_filt1[k] + gamma[k]^2 * R)
  x_filt1[k] <- x_filt1[k] + K1[k] * (z[k] - gamma[k] * x_filt1[k])
  P_filt1[k] <- (1 - K1[k] * gamma[k]) * P_filt1[k]

  # Suavizador
  if (k >= 2) {
    J1[k-1] <- P_filt1[k-1] * phi * P_filt1[k]
    x_smooth1[k] <- x_filt1[k] + J1[k-1] * (x_smooth1[k-1] - phi *
x_filt1[k])
    P_smooth1[k] <- P_filt1[k] + J1[k-1] * (P_smooth1[k-1] - P_filt1[k])
* J1[k-1]
  }
}

# Aplicar el ciclo computacional del filtro y el suavizador para N=2
for (k in 3:N) {
  # Modelo del sistema
  x[k] <- phi * x[k-1] + w[k]
  z[k] <- gamma[k] * x[k] + v[k]

  # Filtro de Kalman
  # Predicción
  x_filt2[k] <- phi * x_filt2[k-1]
  P_filt2[k] <- phi^2 * P_filt2[k-1] + Q

  # Actualización
  K2[k] <- P_filt2[k] / (P_filt2[k] + gamma[k]^2 * R)
  x_filt2[k] <- x_filt2[k] + K2[k] * (z[k] - gamma[k] * x_filt2[k])
  P_filt2[k] <- (1 - K2[k] * gamma[k]) * P_filt2[k]
```

```
  # Suavizador
  if (k >= 3) {
    J2[k-1] <- P_filt2[k-1] * phi * P_filt2[k]
    x_smooth2[k] <- x_filt2[k] + J2[k-1] * (x_smooth2[k-1] - phi *
x_filt2[k])
    P_smooth2[k] <- P_filt2[k] + J2[k-1] * (P_smooth2[k-1] - P_filt2[k])
* J2[k-1]
  }
}

# Aplicar el ciclo computacional del filtro y el suavizador para N=4
for (k in 4:N) {
  # Modelo del sistema
  x[k] <- phi * x[k-1] + w[k]
  z[k] <- gamma[k] * x[k] + v[k]

  # Filtro de Kalman
  # Predicción
  x_filt4[k] <- phi * x_filt4[k-1]
  P_filt4[k] <- phi^2 * P_filt4[k-1] + Q

  # Actualización
  K4[k] <- P_filt4[k] / (P_filt4[k] + gamma[k]^2 * R)
  x_filt4[k] <- x_filt4[k] + K4[k] * (z[k] - gamma[k] * x_filt4[k])
  P_filt4[k] <- (1 - K4[k] * gamma[k]) * P_filt4[k]

  # Suavizador
  if (k >= 4) {
    J4[k-1] <- P_filt4[k-1] * phi * P_filt4[k]
    x_smooth4[k] <- x_filt4[k] + J4[k-1] * (x_smooth4[k-1] - phi *
x_filt4[k])
    P_smooth4[k] <- P_filt4[k] + J4[k-1] * (P_smooth4[k-1] - P_filt4[k])
* J4[k-1]
  }
}

# Cálculo de errores
e_filt1 <- x_filt1 - x
e_filt2 <- x_filt2 - x
e_filt4 <- x_filt4 - x
e_suav1 <- x_smooth1 - x
e_suav2 <- x_smooth2 - x
e_suav4 <- x_smooth4 - x

# Cálculo de varianzas
e_filt1_var <- cumsum(e_filt1^2) / (1:N)
e_suav1_var <- cumsum(e_suav1^2) / (1:N)
e_filt2_var <- cumsum(e_filt2^2) / (1:N)
```

Igual que te he comentado antes.

```
e_suav2_var <- cumsum(e_suav2^2) / (1:N)
e_filt4_var <- cumsum(e_filt4^2) / (1:N)
e_suav4_var <- cumsum(e_suav4^2) / (1:N)

# Gráfica: Varianzas de los errores de filtrado y suavizamiento para N=1, N=2 y N=4
par(mfrow = c(1, 1))
par(mar = c(4, 4, 2, 1))  # Ajustar los márgenes

plot(1:N, e_filt1_var, type = "l", col = "red", lty = 1, ylab = "Varianza",
     xlab = "Iteración", main = "Varianzas de errores de filtrado y suavizamiento para N=1, N=2 y N=4")
lines(1:N, e_filt2_var, col = "green", lty = 1)
lines(1:N, e_filt4_var, col = "blue", lty = 1)
lines(1:N, e_suav1_var, col = "red", lty = 2)
lines(1:N, e_suav2_var, col = "green", lty = 2)
lines(1:N, e_suav4_var, col = "blue", lty = 2)
legend("bottomright", legend = c("Filtrado N=1", "Filtrado N=2", "Filtrado N=4", "Suavizamiento N=1", "Suavizamiento N=2", "Suavizamiento N=4"),
       col = c("red", "green", "blue", "red", "green", "blue"), lty = c(1, 1, 1, 2, 2, 2), cex=0.7)
```



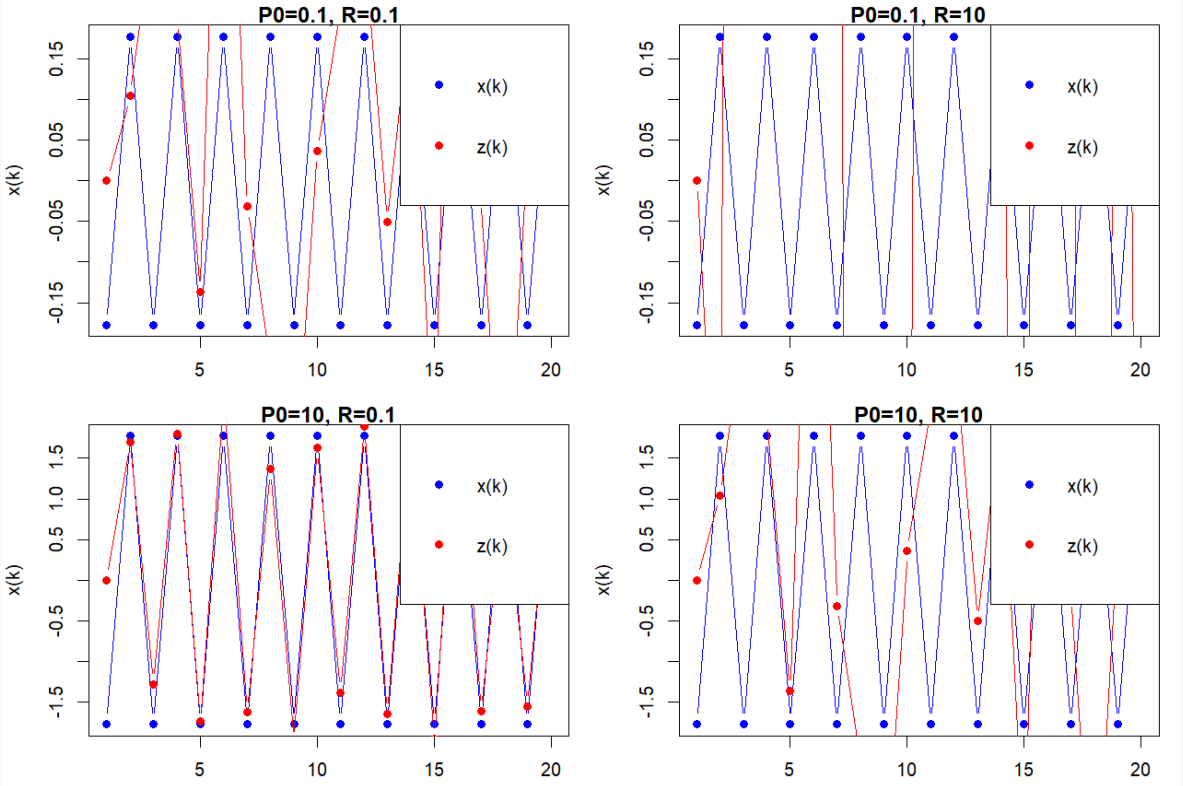
Se puede ver que en todos los casos la varianza del error de suavizamiento es menor que la varianza del error de filtrado.

Ejercicio 2

Apartado (a)

```
# Creo la función para simular x(k)
proceso_simulado <- function(P0, R, iteraciones) {
  set.seed(100)
  # Para garantizar la reproducibilidad del código y el proceso simulado, fijo una semilla determinada. Así se generan los mismos números "pseudoaleatorios" siempre.
  x <- numeric(iteraciones)
  z <- numeric(iteraciones)

  # Inicio x0 como una variable gaussiana con media cero y varianza P0
  x[1] <- sqrt(P0) * rnorm(1)

  for (k in 2:iteraciones) {
    # Actualizo el proceso x(k) según la relación dada
    x[k] <- (-1)^(2*k + 1) * x[k-1]
```

```
    # Y de aquí salen las observaciones z(k)
    z[k] <- x[k] + sqrt(R) * rnorm(1)
  }

  return(list(x = x, z = z))
}

# Defino ahora el número de iteraciones que quiero y los valores P0 y R.
# Como los tres parametros son variables es muy facil cambiar cualquiera
de los tres y generar nuevas gráficas
iteraciones <- 20
P0_values <- c(1, 10)
R_values <- c(0.1, 100)

# Creo los 4 gráficos con cada combinación de P0 y R
par(mfrow = c(length(P0_values), length(R_values)), mar = c(3, 4, 1, 1))

for (i in seq_along(P0_values)) {
  for (j in seq_along(R_values)) {
    # Simulamos el proceso para el bucle de valores de P0 (bucle i, 2
valores posibles) y R (bucle j, también 2 valores)
    simulated_data <- proceso_simulado(P0_values[i], R_values[j],
iteraciones)

    # Gráfico de las trayectorias de x(k) y z(k)
    plot(seq_len(iteraciones), simulated_data$x, type = "b", pch = 16,
         col = "blue", xlab = "k", ylab = "x(k)", main = sprintf("P0=%s,
R=%s", P0_values[i], R_values[j]))
    points(seq_len(iteraciones), simulated_data$z, type = "b", pch = 16,
col = "red")

    # Leyenda en los gráficos
    legend("topright", legend = c("x(k)", "z(k)"), col = c("blue",
"red"), pch = 16)
  }
}
```



Se puede cambiar los valores de P0 y R para ver cómo varían los resultados. Se ve que las observaciones del proceso son muy poco precisas cuando R se incrementa (siendo R la varianza, así que es razonable). En cambio, para varianzas muy pequeñas, las observaciones se aproximan mucho más a la realidad del proceso.

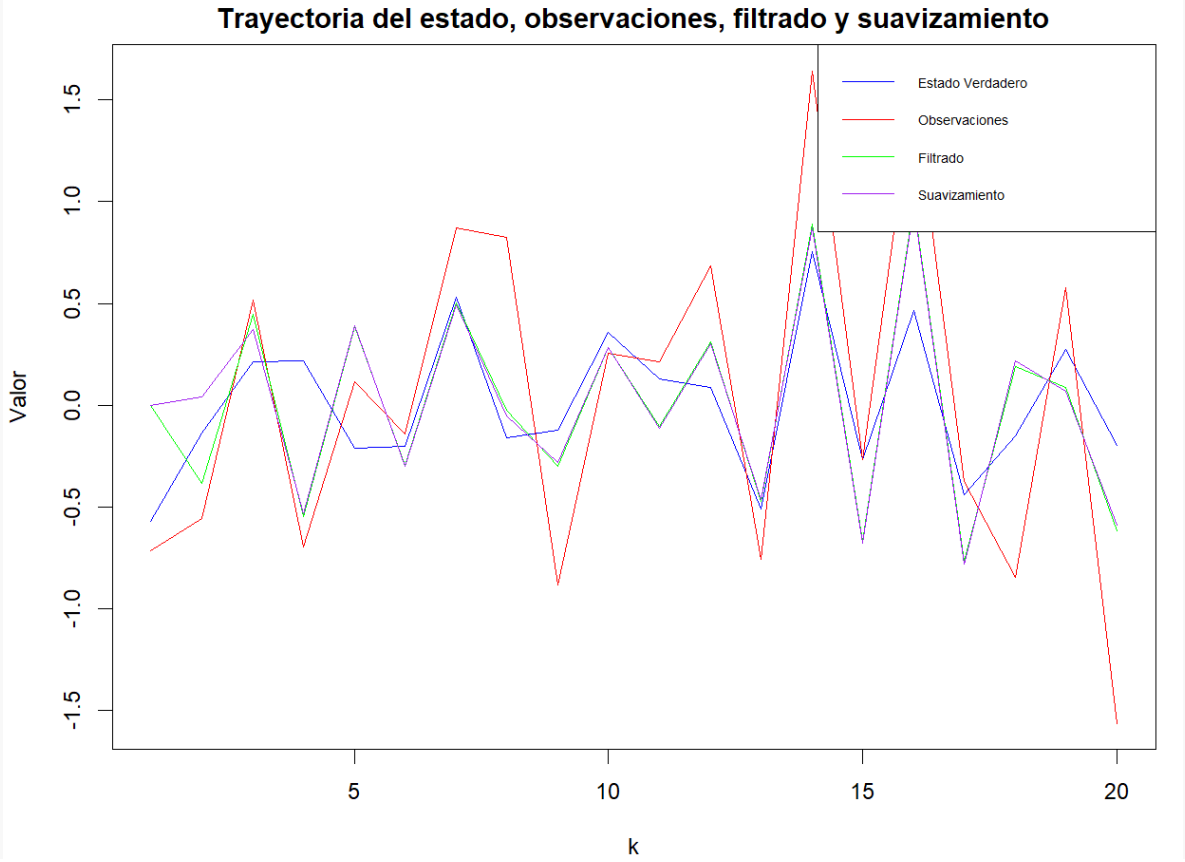
Apartado (b) y (c)

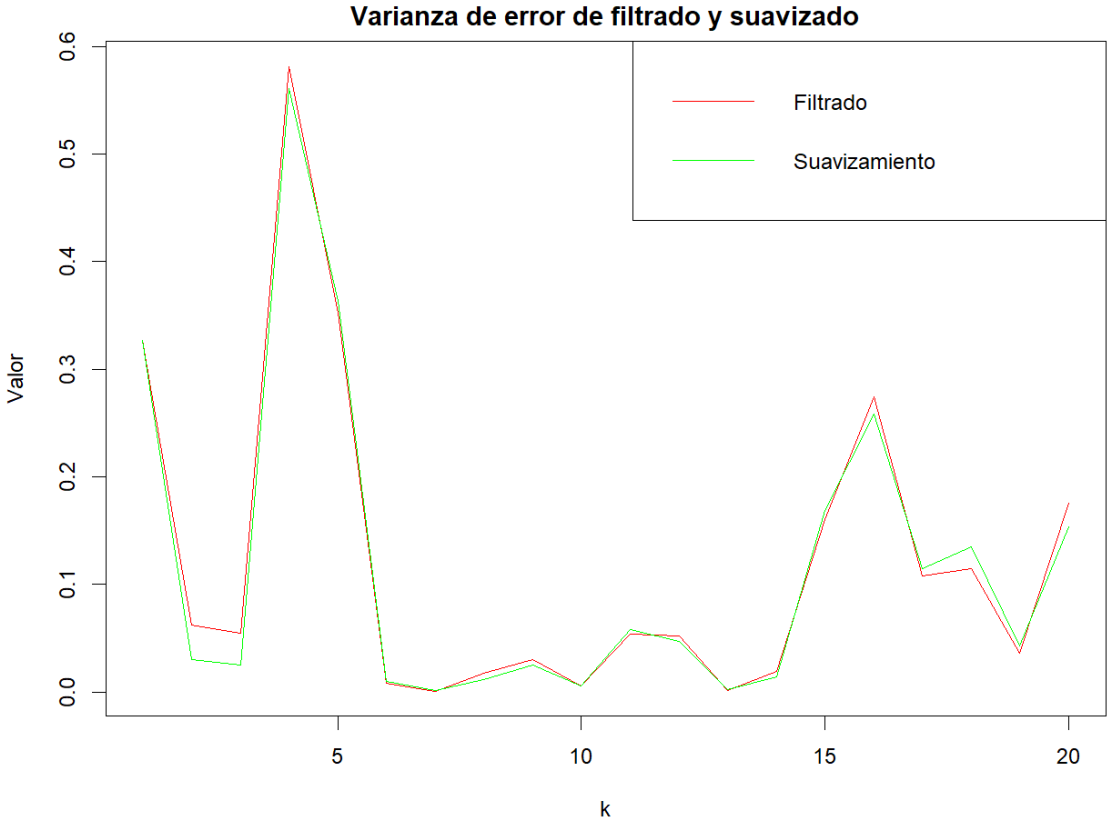
```
# Definir parámetros del sistema
n <- 2       # Dimensión del estado
N_iter <- 20  # Número de iteraciones
Q <- 0.1      # Varianza del ruido de proceso
R <- 0.5      # Varianza del ruido de medida
N_suav <- 2   # N para el suavizamiento

# Inicializar vectores y matrices
x <- numeric(N_iter)      # Estado real
z <- numeric(N_iter)      # Observaciones
x_hat_filt <- numeric(N_iter)  # Estimaciones de filtrado
x_hat_suav <- numeric(N_iter)  # Estimaciones de suavizamiento
P_filt <- numeric(N_iter)      # Varianzas de filtrado
P_suav <- numeric(N_iter)      # Varianzas de suavizamiento
K <- numeric(N_iter)       # Ganancia de Kalman
M <- numeric(N_iter)       # Ganancia de suavizamiento
```

```
# Condiciones iniciales
x[1] <- rnorm(1)  # Variable gaussiana con media cero y varianza P0 = 1
z[1] <- x[1] + sqrt(R) * rnorm(1)
x_hat_filt[1] <- 0  # Estimación inicial
P_filt[1] <- 1
x_hat_suav[1] <- 0  # Estimación inicial
P_suav[1] <- 1

# Bucle for para filtrado y suavizamiento
for (k in 2:N_iter) {
  # Modelo del sistema
  phi <- (-1)^((2*k)+1)
  x[k] <- phi * x[k - 1] + sqrt(Q) * rnorm(1)

  # Observación
  z[k] <- x[k] + sqrt(R) * rnorm(1)

  # Filtro de Kalman
  # Predicción
  x_hat_filt[k] <- phi * x_hat_filt[k - 1]
  P_filt[k] <- phi^2 * P_filt[k - 1] + Q

  # Ganancia de Kalman
  K[k] <- P_filt[k] / (P_filt[k] + R)

  # Actualización de la estimación y la covarianza
  x_hat_filt[k] <- x_hat_filt[k] + K[k] * (z[k] - x_hat_filt[k])
  P_filt[k] <- (1 - K[k]) * P_filt[k]

  # Suavizador punto fijo
  if (k >= N_suav) {
    M[k] <- P_filt[k - N_suav + 1] * phi / (P_filt[k - N_suav + 1] *
phi^2 + Q)
    x_hat_suav[k] <- x_hat_filt[k] + M[k] * (x_hat_suav[k - 1] - phi *
x_hat_filt[k])
    P_suav[k] <- P_filt[k] + M[k] / P_filt[k]
  }
}

# Trayectoria y observaciones
plot(1:N_iter, x, type = 'l', col = 'blue', ylim = c(min(x, z,
x_hat_filt, x_hat_suav), max(x, z, x_hat_filt, x_hat_suav)),
     ylab = 'Valor', xlab = 'k', main = 'Trayectoria del estado,
observaciones, filtrado y suavizamiento')
lines(1:N_iter, z, col = 'red')
lines(1:N_iter, x_hat_filt, col = 'green')
lines(1:N_iter, x_hat_suav, col = 'purple')
legend("topright", legend = c("Estado Verdadero", "Observaciones",
```

```
"Filtrado", "Suavizamiento"), col = c("blue", "red", "green", "purple"),
lty = 1, cex=0.6)

# Varianzas de errores de filtrado y suavizamiento
e_filt <- x_hat_filt - x
e_suav <- x_hat_suav - x
e_filt2 <- e_filt^2
e_suav2 <- e_suav^2

plot(1:N_iter, e_filt2, typ                    im = c(-1, 2),
     ylab = 'Valor', xlab =                    e error de filtrado y
suavizado')
lines(1:N_iter, e_suav2, co
legend("topright", legend =                    iento"), col =
c("red", "green"), lty = 1)
```

Esto no son las varianzas, son errores. Pero para medir la bondad de los estimadores debemos usar las varianzas, que en nuestro caso coincide con P(k/k) o P(k/k+N) dependiendo que sea filtro o suavizamiento





Para calcular las varianzas de errores de filtrado y suavizado para N=1, N=2 y N=4 podemos modificar el script anterior:

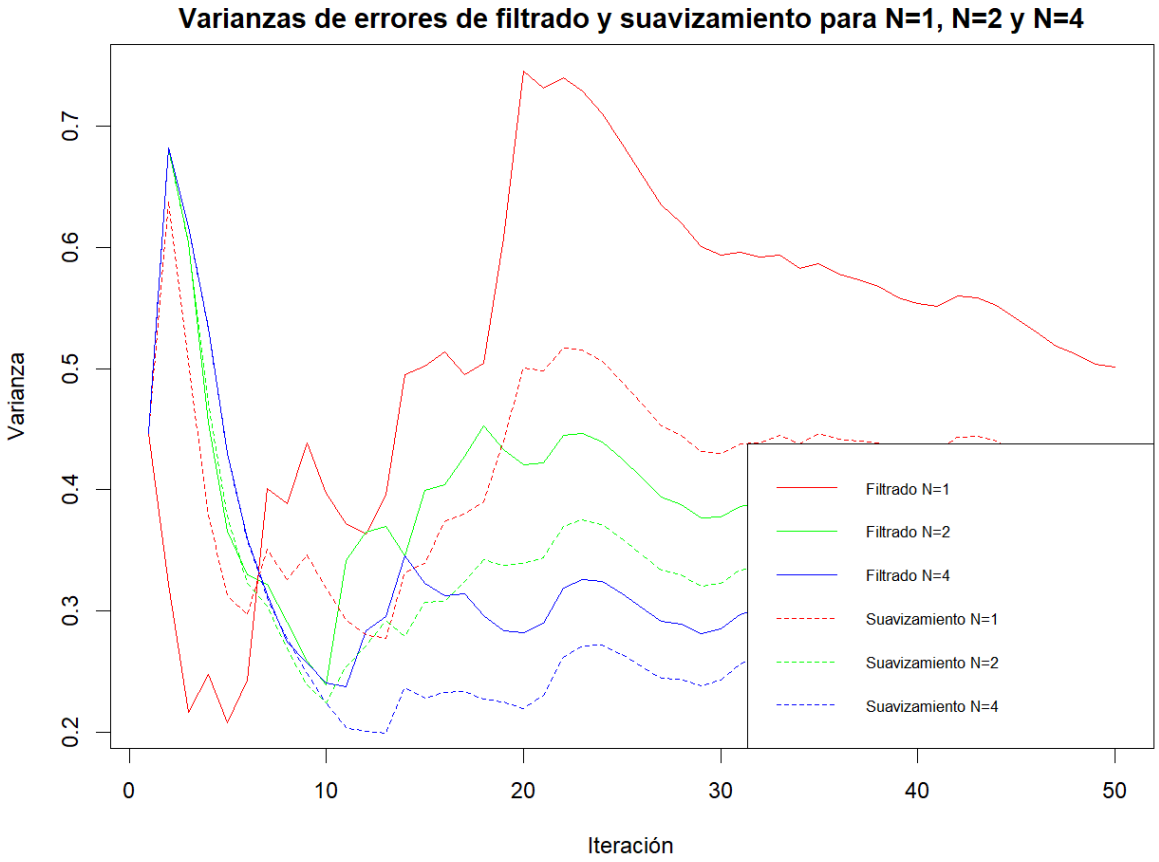
```
# Definir parámetros del sistema
n <- 2       # Dimensión del estado
N_iter <- 20  # Número de iteraciones
Q <- 0.1      # Varianza del ruido de proceso
R <- 0.5      # Varianza del ruido de medida
N_suav <- 2   # N para el suavizamiento

# Inicializar vectores y matrices
x <- numeric(N_iter)      # Estado real
z <- numeric(N_iter)      # Observaciones
x_hat_filt <- numeric(N_iter)  # Estimaciones de filtrado
x_hat_suav <- numeric(N_iter)  # Estimaciones de suavizamiento
P_filt <- numeric(N_iter)      # Varianzas de filtrado
P_suav <- numeric(N_iter)      # Varianzas de suavizamiento
K <- numeric(N_iter)       # Ganancia de Kalman
M <- numeric(N_iter)       # Ganancia de suavizamiento

# Condiciones iniciales
x[1] <- rnorm(1)  # Variable gaussiana con media cero y varianza P0 = 1
z[1] <- x[1] + sqrt(R) * rnorm(1)
x_hat_filt[1] <- 0  # Estimación inicial
P_filt[1] <- 1
```

```
x_hat_suav[1] <- 0  # Estimación inicial
P_suav[1] <- 1
x_filt1 <- numeric(N)  # Estimaciones de filtrado para N=1
x_filt2 <- numeric(N)  # Estimaciones de filtrado para N=2
x_filt4 <- numeric(N)  # Estimaciones de filtrado para N=4
x_smooth1 <- numeric(N)  # Estimaciones de suavizamiento para N=1
x_smooth2 <- numeric(N)  # Estimaciones de suavizamiento para N=2
x_smooth4 <- numeric(N)  # Estimaciones de suavizamiento para N=4
P_filt1 <- numeric(N)  # Varianzas de filtrado para N=1
P_filt2 <- numeric(N)  # Varianzas de filtrado para N=2
P_filt4 <- numeric(N)  # Varianzas de filtrado para N=4
P_smooth1 <- numeric(N)  # Varianzas de suavizamiento para N=1
P_smooth2 <- numeric(N)  # Varianzas de suavizamiento para N=2
P_smooth4 <- numeric(N)  # Varianzas de suavizamiento para N=4
K1 <- numeric(N)  # Ganancia de Kalman para N=1
K2 <- numeric(N)  # Ganancia de Kalman para N=2
K4 <- numeric(N)  # Ganancia de Kalman para N=4
J1 <- numeric(N)  # Ganancia de suavizamiento para N=1
J2 <- numeric(N)  # Ganancia de suavizamiento para N=2
J4 <- numeric(N)  # Ganancia de suavizamiento para N=4
x_filt1[1] <- 0  # Estimación inicial para N=1
x_filt2[1] <- 0  # Estimación inicial para N=2
x_filt4[1] <- 0  # Estimación inicial para N=4
P_filt1[1] <- 1
P_filt2[1] <- 1
P_filt4[1] <- 1
x_smooth1[1] <- 0  # Estimación inicial para N=1
x_smooth2[1] <- 0  # Estimación inicial para N=2
x_smooth4[1] <- 0  # Estimación inicial para N=4
P_smooth1[1] <- 1
P_smooth2[1] <- 1
P_smooth4[1] <- 1

# Bucle for para filtrado y suavizamiento
for (k in 2:N_iter) {
  # Modelo del sistema
  phi <- (-1)^((2*k)+1)
  x[k] <- phi * x[k - 1] + sqrt(Q) * rnorm(1)

  # Observación
  z[k] <- x[k] + sqrt(R) * rnorm(1)

  # Filtro de Kalman
  # Predicción
  x_filt1[k] <- phi * x_filt1[k - 1]
  P_filt1[k] <- phi^2 * P_filt1[k - 1] + Q

  # Ganancia de Kalman
  K[k] <- P_filt1[k] / (P_filt1[k] + R)
```

```
  # Actualización de la estimación y la covarianza
  x_filt1[k] <- x_filt1[k] + K[k] * (z[k] - x_filt1[k])
  P_filt1[k] <- (1 - K[k]) * P_filt1[k]

  # Suavizador punto fijo
  if (k >= 2) {
    M[k] <- P_filt1[k - N_suav + 1] * phi / (P_filt1[k - N_suav + 1] *
phi^2 + Q)
    x_smooth1[k] <- x_filt1[k] + M[k] * (x_smooth1[k - 1] - phi *
x_filt1[k])
    P_smooth1[k] <- P_filt1[k] + M[k] / P_filt1[k]
  }
}

# Bucle for para filtrado y suavizamiento para N=2
for (k in 3:N_iter) {
  # Modelo del sistema
  phi <- (-1)^((2*k)+1)
  x[k] <- phi * x[k - 1] + sqrt(Q) * rnorm(1)

  # Observación
  z[k] <- x[k] + sqrt(R) * rnorm(1)

  # Filtro de Kalman
  # Predicción
  x_filt2[k] <- phi * x_filt2[k - 1]
  P_filt2[k] <- phi^2 * P_filt2[k - 1] + Q

  # Ganancia de Kalman
  K[k] <- P_filt2[k] / (P_filt2[k] + R)

  # Actualización de la estimación y la covarianza
  x_filt2[k] <- x_filt2[k] + K[k] * (z[k] - x_filt2[k])
  P_filt2[k] <- (1 - K[k]) * P_filt2[k]

  # Suavizador punto fijo
  if (k >= 3) {
    M[k] <- P_filt2[k - N_suav + 1] * phi / (P_filt2[k - N_suav + 1] *
phi^2 + Q)
    x_smooth2[k] <- x_filt2[k] + M[k] * (x_smooth2[k - 1] - phi *
x_filt2[k])
    P_smooth2[k] <- P_filt2[k] + M[k] / P_filt2[k]
  }
}

# Bucle for para filtrado y suavizamiento para N=4
for (k in 4:N_iter) {
  # Modelo del sistema
```

```
  phi <- (-1)^((2*k)+1)
  x[k] <- phi * x[k - 1] + sqrt(Q) * rnorm(1)

  # Observación
  z[k] <- x[k] + sqrt(R) * rnorm(1)

  # Filtro de Kalman
  # Predicción
  x_filt4[k] <- phi * x_filt4[k - 1]
  P_filt4[k] <- phi^2 * P_filt4[k - 1] + Q

  # Ganancia de Kalman
  K[k] <- P_filt4[k] / (P_filt4[k] + R)

  # Actualización de la estimación y la covarianza
  x_filt4[k] <- x_filt4[k] + K[k] * (z[k] - x_filt4[k])
  P_filt4[k] <- (1 - K[k]) * P_filt4[k]

  # Suavizador punto fijo
  if (k >= 4) {
    M[k] <- P_filt4[k - N_suav + 1] * phi / (P_filt4[k - N_suav + 1] *
phi^2 + Q)
    x_smooth4[k] <- x_filt4[k] + M[k] * (x_smooth4[k - 1] - phi *
x_filt4[k])
    P_smooth4[k] <- P_filt4[k] + M[k] / P_filt4[k]
  }
}

# Cálculo de errores
e_filt1 <- x_filt1 - x
e_filt2 <- x_filt2 - x
e_filt4 <- x_filt4 - x
e_suav1 <- x_smooth1 - x
e_suav2 <- x_smooth2 - x
e_suav4 <- x_smooth4 - x

# Cálculo de varianzas
e_filt1_var <- cumsum(e_filt1^2) / (1:N)
e_suav1_var <- cumsum(e_suav1^2) / (1:N)
e_filt2_var <- cumsum(e_filt2^2) / (1:N)
e_suav2_var <- cumsum(e_suav2^2) / (1:N)
e_filt4_var <- cumsum(e_filt4^2) / (1:N)
e_suav4_var <- cumsum(e_suav4^2) / (1:N)

# Gráfica: Varianzas de los errores de filtrado y suavizamiento para N=1,
N=2 y N=4
par(mfrow = c(1, 1))
par(mar = c(4, 4, 2, 1))  # Ajustar los márgenes
```

```
plot(1:N, e_filt1_var, type = "l", col = "red", lty = 1, ylab =
"Varianza",
     xlab = "Iteración", main = "Varianzas de errores de filtrado y
suavizamiento para N=1, N=2 y N=4")
lines(1:N, e_filt2_var, col = "green", lty = 1)
lines(1:N, e_filt4_var, col = "blue", lty = 1)
lines(1:N, e_suav1_var, col = "red", lty = 2)
lines(1:N, e_suav2_var, col = "green", lty = 2)
lines(1:N, e_suav4_var, col = "blue", lty = 2)
legend("bottomright", legend = c("Filtrado N=1", "Filtrado N=2",
"Filtrado N=4", "Suavizamiento N=1", "Suavizamiento N=2", "Suavizamiento
N=4"),
       col = c("red", "green", "blue", "red", "green", "blue"), lty =
c(1, 1, 1, 2, 2, 2), cex=0.7)
```

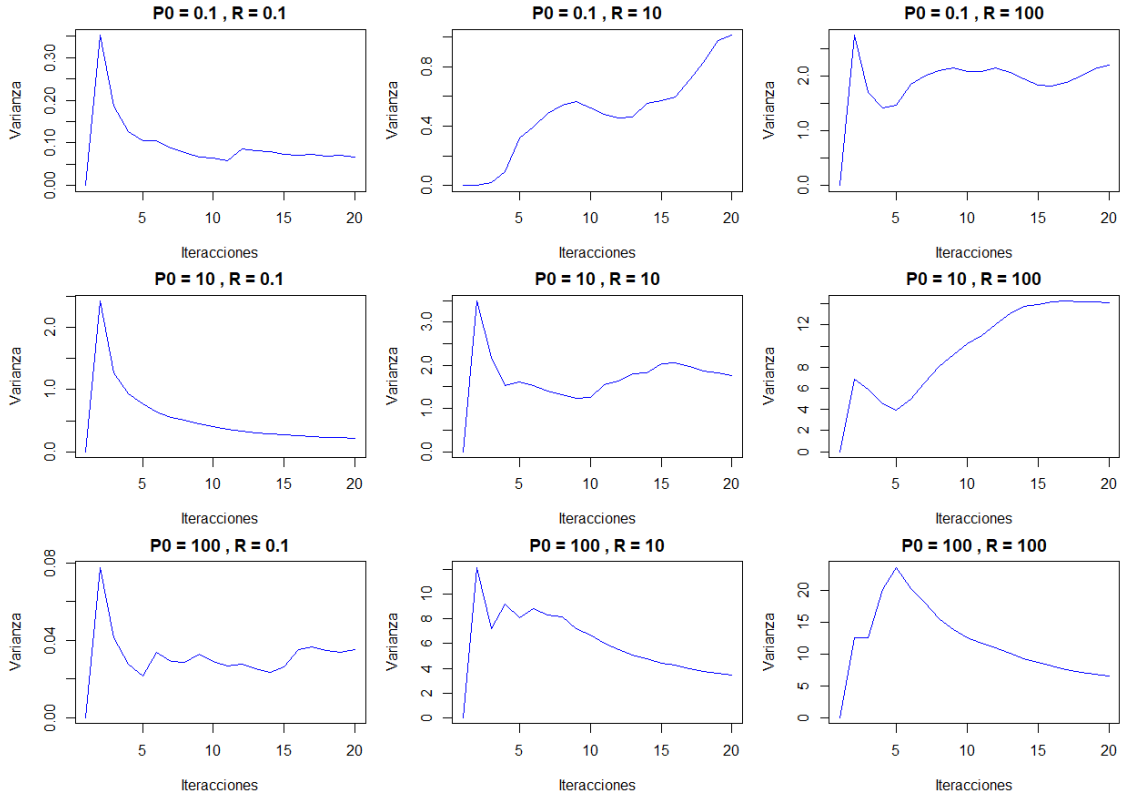


Apartado (d)

```
# Defino los parámetros del modelo
set.seed(50)
Q <- 0.1
N_iter <- 20
N_suav <- 2

# Valores de P0 y R a probar
P0_values <- c(0.1, 10, 100)
```

```
R_values <- c(0.1, 10, 100)

# Inicialización de e_filt2 fuera de los bucles
e_filt2 <- matrix(0, nrow = N_iter, ncol = length(P0_values) *
length(R_values))

# Bucles para iterar sobre P0 y R
for (i in 1:length(P0_values)) {
  for (j in 1:length(R_values)) {
    # Inicialización de variables
    x <- numeric(N_iter)
    z <- numeric(N_iter)
    x_hat_filt <- numeric(N_iter)
    P_filt <- numeric(N_iter)

    # Condiciones iniciales
    x[1] <- rnorm(1)
    z[1] <- x[1] + sqrt(R_values[j]) * rnorm(1)
    x_hat_filt[1] <- 0
    P_filt[1] <- P0_values[i]

    # Bucle for para filtrado
    for (k in 2:N_iter) {
      # Evolución del estado verdadero
      phi <- (-1)^((2*k)+1)
      x[k] <- phi * x[k - 1] + sqrt(Q) * rnorm(1)

      # Observación
      z[k] <- x[k] + sqrt(R_values[j]) * rnorm(1)

      # Filtro de Kalman
      x_hat_filt[k] <- phi * x_hat_filt[k - 1]
      P_filt[k] <- phi^2 * P_filt[k - 1] + Q

      # Ganancia de Kalman
      K <- P_filt[k] / (P_filt[k] + R_values[j])

      # Actualización de la estimación y la covarianza
      x_hat_filt[k] <- x_hat_filt[k] + K * (z[k] - x_hat_filt[k])
      P_filt[k] <- (1 - K) * P_filt[k]

      # Cálculo de errores
      e_filt <- x_hat_filt - x

      # Almacenar varianzas en e_filt2
      e_filt2[k, (i - 1) * length(R_values) + j] <- var(e_filt[1:k])
    }
  }
```

```
}

# Gráfica de varianzas de errores de filtrado
par(mfrow = c(length(P0_values), length(R_values)), mar = c(4, 4, 2, 1),
oma = c(0, 0, 2, 0))
for (i in 1:length(P0_values)) {
  for (j in 1:length(R_values)) {
    plot(e_filt2[, (i - 1) * length(R_values) + j], type = 'l', col =
'blue',
         ylab = 'Varianza', xlab = 'Iteracciones', main = paste('P0 =',
P0_values[i], ', R =', R_values[j]))
  }
}
```



La varianza inicial del estado ($P0$) representa la incertidumbre en el estado inicial. Si $P0$ es mayor, significa que estamos menos seguros sobre el estado inicial, y esto puede resultar en mayores varianzas de los errores de filtrado a lo largo del tiempo. La varianza del ruido de medición (R) afecta directamente la ponderación de las observaciones en el filtro de Kalman. Si R es mayor, las observaciones tienen menos influencia en las actualizaciones del filtro, y esto podría resultar en mayores varianzas de los errores de filtrado.

En la gráfica se puede ver que los valores más bajos de varianza vienen dados cuando la varianza inicial y R son bajos, y se ve cómo aumentan progresivamente cuando se incrementa la varianza inicial y, especialmente, cuando se incrementa el valor de R.