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) y (a.2)

Es casi directo pero no tanto.

En ambos casos, tanto $x(j), w(k)$ como $z(j), w(k)$ son variables aleatorias independientes. $z(j)$ es una combinación lineal de variables aleatorias gaussianas independientes, y $w(k)$ es u ruido blanco gaussiano. Por lo tanto, se cumple:

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

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

Demostración de (a.3)

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

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

La esperanza de $x(j)$ es cero, ya que $x(j)$ es un vector aleatorio gaussiano con media cero. La esperanza de $v(k)$ es también cero, ya que $v(k)$ es un ruido blanco gaussiano centrado.

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

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

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

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 $v(k)$. 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

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

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

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

Apartado (b)

Considerando:

1. $\hat{x}(k/j)$: el estimador lineal de menor error cuadrático medio (LMMSE) de $x(k)$ basado en $z(0), \ldots, z(j)$.
2. $\hat{x}(k/j-1)$: es el estimador LMMSE de $x(k)$ basado en $z(0), \ldots, z(j-1)$.
3. $K(k,j)$: es la ganancia del filtro de Kalman en el instante k basado en la información hasta $z(j)$.
4. $\hat{z}(j/j-1)$: es la predicción del sistema en el instante j basado en la información hasta $z(j-1)$.

Se pide demostrar:

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

El LMMSE de $x(k)$ basado en $z(0), \ldots, z(j)$ se puede expresar como la predicción más la corrección del error de predicción:

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

Se puede simplificar si tenemos en cuenta que $\hat{z}(k/j) = \hat{z}(k/j-1)$, ya que la predicción en el instante k basada en la información hasta $z(j)$ es la misma que la predicción basada en $z(j-1)$. Sustituimos esto en la expresión anterior:

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

No, es parecido a como lo tienes hecho en teoría para el filtro.

Factorizo:

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

Agrupo términos y simplifico; teniendo en cuenta que $\hat{z}(j/j-1) = \hat{z}(j-1/j-1)$,:

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

Apartado (c)

La innovación es la diferencia entre la observación actual y la predicción de la observación. La observación es $z(k)$, y la predicción se obtiene a partir de la estimación del estado anterior.

La predicción de la observación en el instante k basada en la información hasta $k-1$ es:

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

donde $\hat{x}(k/k-1)$ es la predicción del estado en el instante k basada en la información hasta $k-1$.

La innovación se define como:

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

Sustituyendo se llega a que el proceso de innovación es:

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

Para las demostraciones que se piden:

1. **Gaussiano:** Si $\tilde{z}(k)$ y $\hat{x}(k/k-1)$ son gaussianos; 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:** Si $\tilde{z}(k/k-1)$ es blanco, su covarianza para $j \neq i$ es:

$$E\left[\tilde{z}_i(k/k-1)\tilde{z}_j(k/k-1)^T\right] = E\left[\left(z_i(k) - H_i(k)\hat{x}(k/k-1)\right)\left(z_j(k) - H_j(k)\hat{x}(k/k-1)\right)^T\right]$$

Expandimos esto y simplificamos teniendo en cuenta que $\hat{x}(k/k-1)$ y $z(k)$ son independientes, lo que significa que (demostrado en el ejercicio 1)

No

$$E\left[H_i(k)\hat{x}_i(k/k-1)z_j(k)^T\right] = E\left[z_i(k)H_j(k)\hat{x}_j(k/k-1)^T\right] = 0.$$

Como además $E[z(k)z(k)^T] = \Pi(k)$, donde $\Pi(k)$ es la matriz de covarianza de la observación en el instante k:

$$E\left[\tilde{z}_i(k/k-1)\tilde{z}_j(k/k-1)^T\right] = \Pi(k) - E\left[H_i(k)\hat{x}_i(k/k-1)H_j(k)\hat{x}_j(k/k-1)^T\right]$$

Apartado (d)

El algoritmo de filtrado de Kalman es un algoritmo 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 inicial y la actualización posterior.

Predicción: 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 $k-1$.

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.

Actualización: 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.

Ahora se actualiza el 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.

Por último, con la covarianza del Error de Estimación:

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

No. Se pide que demuestre cómo se obtienen cada una de las expresiones que intervienen en el algoritmo

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.

El algoritmo de filtrado de Kalman se ejecuta de manera recursiva para cada nueva medición disponible, actualizando el estado estimado y la covarianza del error de estimación en cada paso. Este proceso proporciona una estimación óptima del estado del sistema, teniendo en cuenta las mediciones disponibles.

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 una vez se obtiene la información actual. Un estimador de suavizamiento históricamente usado es el suavizamiento de punto fijo:

1. Inicio definiendo N como el "rango" de tiempo utilizado para el suavizado. O sea, el conjunto de observaciones disponibles va de 0 a N.

$$\hat{x}(N/N) = \hat{x}(N/N)$$

$$P(N/N) = P(N/N)$$

2. Suavizamiento punto fijo:

$$\hat{x}(k/N) = \hat{x}(k/k) + J(k)[\hat{x}(k+1/N) - \hat{x}(k+1/k)]$$
$$P(k/N) = P(k/k) + J(k)[P(k+1/N) - P(k+1/k)]J(k)^T$$

$$J(k) = P(k/k)\Phi(k)^T P(k+1/k)^{-1}$$

Has cogido esto de un artículo que no parte del mismo modelo que aquí se pregunta.

Las matrices de covarianza de los errores de suavizamiento se actualizan e algoritmo según:

$$M(k) = P(k/k)\Phi(k)^T P(k+1/k)^{-1}$$

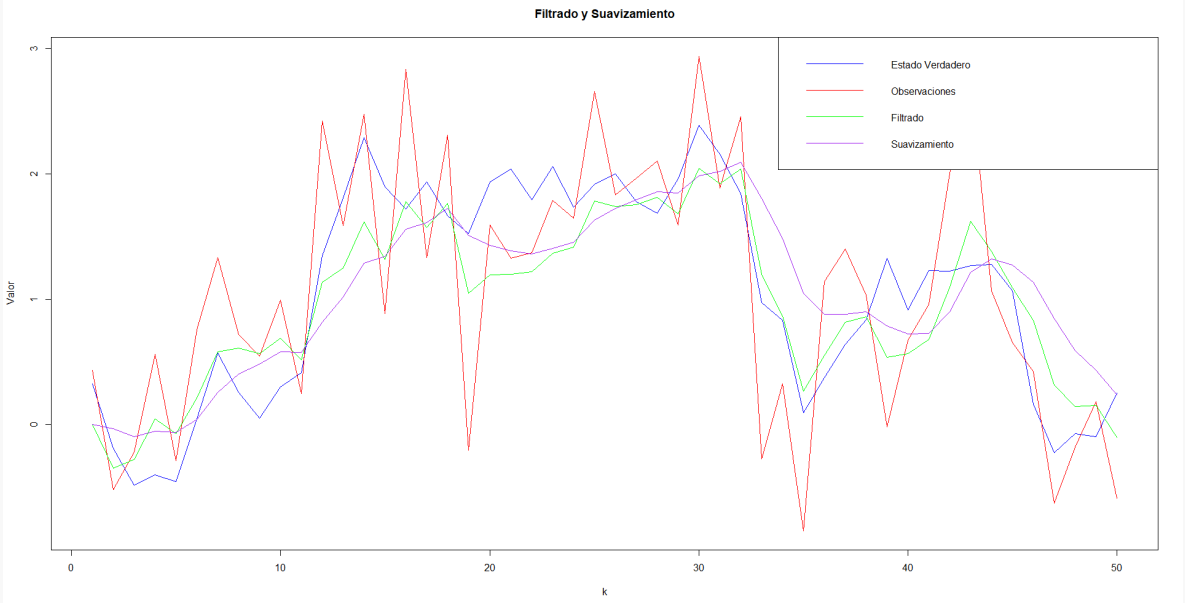
$$P(k/N) = P(k/k) + M(k)[P(k+1/N) - P(k+1/k)]M(k)^T \blacksquare$$

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.

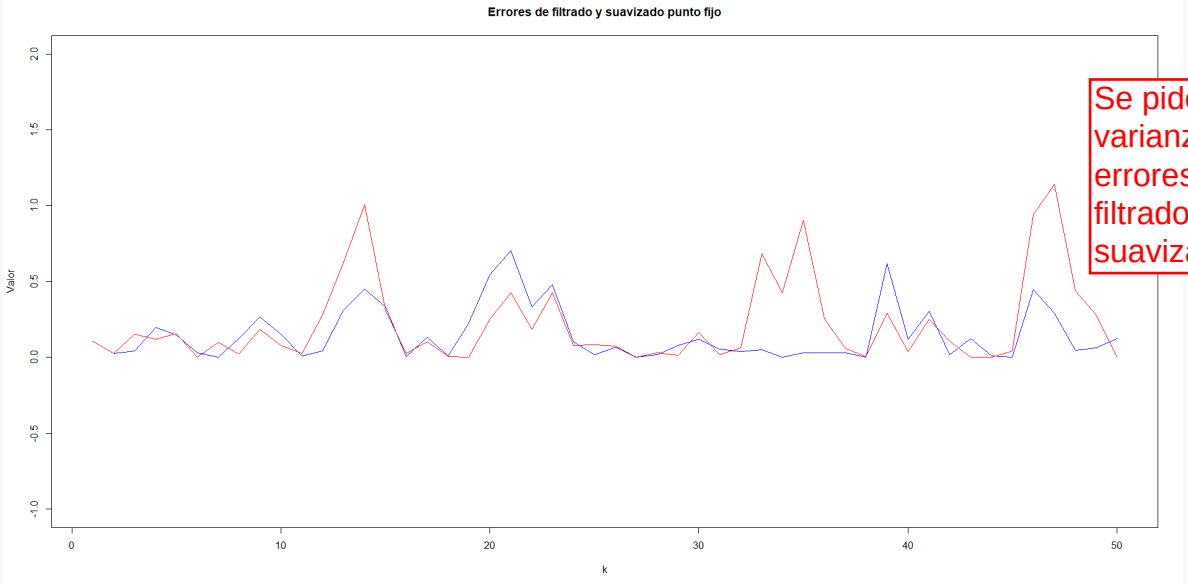
Apartado (f) y (g)

```
# Parámetros del modelo de espacio de estados
phi <- 0.95
Q <- 0.1
R <- 0.5
N_iter <- 50
N_suav <- 2 # Este parámetro se podría cambiar para N=1, N=2 o N=4

# Inicialización de variables
x <- numeric(N_iter)
z <- numeric(N_iter)x_hat_filt <- numeric(N_iter)

x_hat_suav <- numeric(N_iter)
P_filt <- numeric(N_iter)
P_suav <- numeric(N_iter)



# 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

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

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

  # 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]

  # Suavizador punto fijo
  M <- 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 * (x_hat_suav[k - 1] - phi *
x_hat_filt[k])
  P_suav[k] <- P_filt[k] - M^2 * P_filt[k]
}

e_filt <- x - x_hat_filt
e_suav <- x - x_hat_suav
e_filt2 <- e_filt^2
e_suav2 <- e_suav^2

# Gráfica de trayectoria del estado, observaciones, filtrado y
suavizamiento
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 = '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)
```



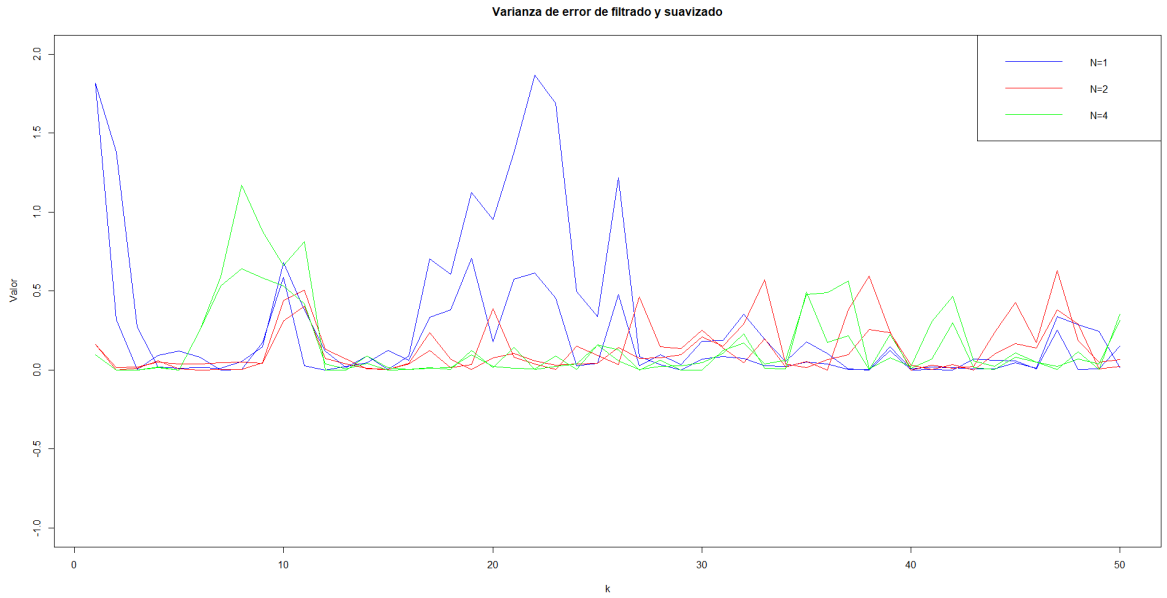

```
plot(1:N_iter, e_filt2, type = 'l', col = 'blue', ylim = c(-1,2),
     ylab = 'Valor', xlab = 'k', main = 'Errores de filtrado y suavizado
punto fijo')
lines(1:N_iter, e_suav2, col = 'red')
```



Se pide la varianza de los errores de filtrado y suavizamiento

Aquí se ha realizado únicamente contemplando N=2 (es decir, la primera parte del apartado g). Para N=1 y N=4; la manera más eficiente sería definir un vector c(1,2,4) en la variable N_suav; y posteriormente adaptar el bucle for para que vaya guardando los valores de las tres N. Pero también se puede hacer cambiando a mano el valor en la variable N_suav y guardando los outputs con diferentes variables; y después, representarlas todas. Se decide esta opción.

```
plot(1:N_iter, e_filt21, type = 'l', col = 'blue', ylim = c(-1,2),
     ylab = 'Valor', xlab = 'k', main = 'Varianza de error de filtrado y
suavizado')
lines(1:N_iter, e_suav21, col = 'blue')
lines(1:N_iter, e_filt22, col = 'red')
lines(1:N_iter, e_suav22, col = 'red')
lines(1:N_iter, e_filt24, col = 'green')
lines(1:N_iter, e_suav24, col = 'green')
legend("topright", legend = c("N=1", "N=2", "N=4"), col = c("blue",
"red", "green"), lty = 1)
```



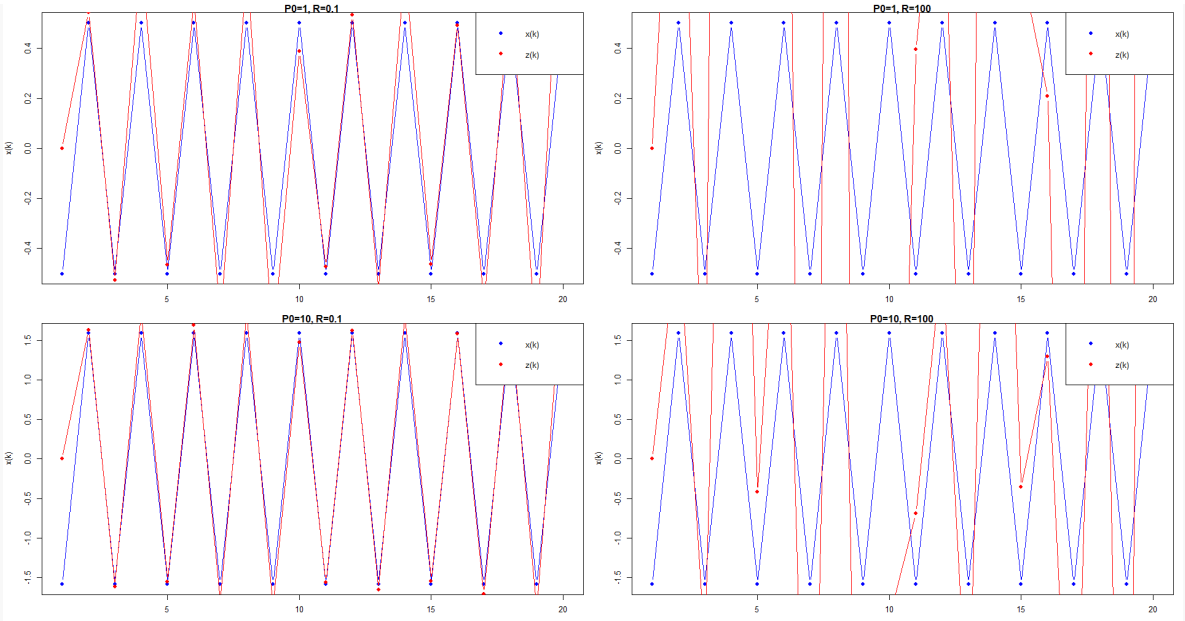
Se puede ver que la varianza de error es mayor cuando N=1, lo cual es esperado. Al aumentar el número de iteraciones, se consiguen más puntos de datos que ayudan a normalizar los resultados del proceso. Igualmente hay bastante variabilidad, pero si siguiésemos aumentando el número de N, iría decayendo.

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
```

```r
  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 parámetros son variables es muy fácil 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)
  }
}
```

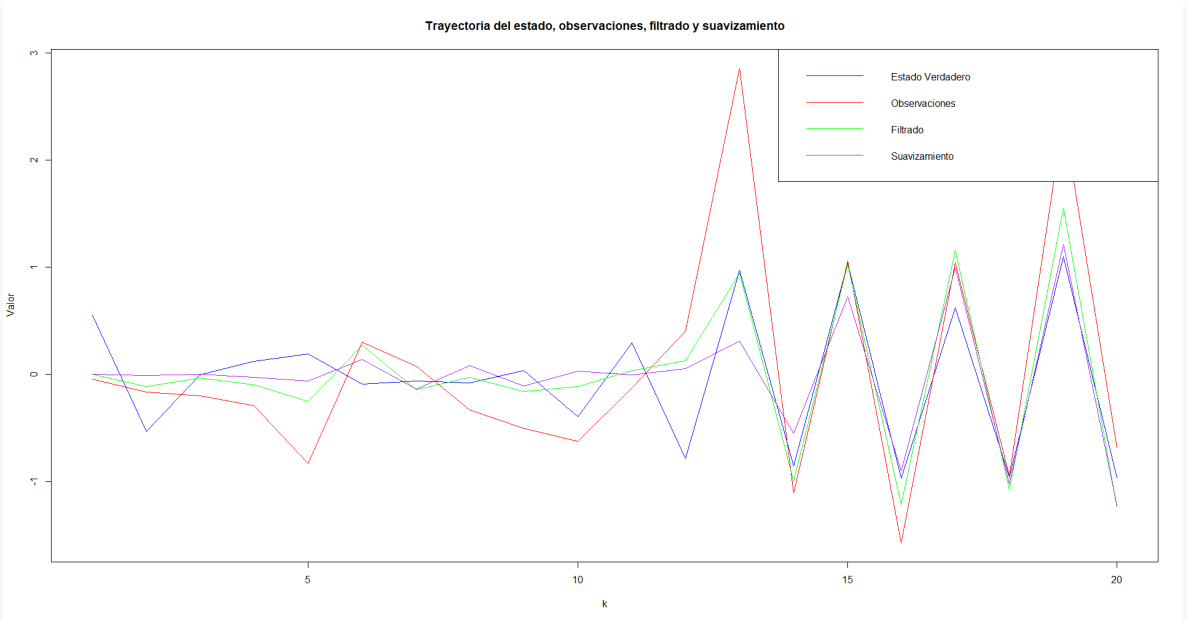


Se puede cambiar los valores de P0 y R para ver cómo varían los resultados. Se ve que las observaciones del proceso son muy poco precisas cuando R se incrementa (siendo R la varianza, así que es razonable). En cambio, para varianzas muy pequeñas, las observaciones se aproximan mucho más a la realidad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

Apartado (b) y apartado (c)

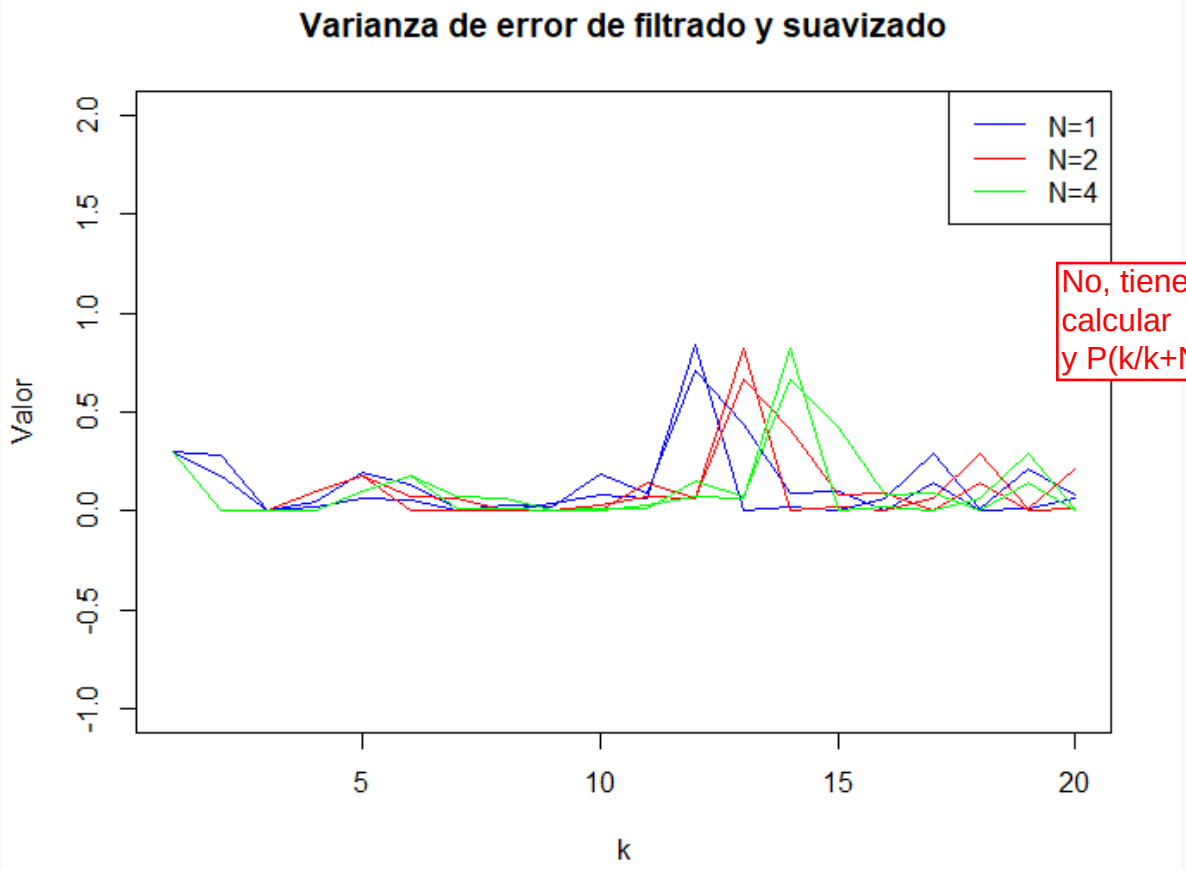
```
# Defino los parámetros del modelo
# De igual manera que en el apartado 2a, creo una semilla para garantizar la reproducibilidad.
set.seed(50)
Q <- 0.1
R <- 0.5
N_iter <- 20
N_suav <- 2

# Inicio las variables de variables
x <- numeric(N_iter)
z <- numeric(N_iter)
x_hat_filt <- numeric(N_iter)
x_hat_suav <- numeric(N_iter)
P_filt <- numeric(N_iter)
P_suav <- numeric(N_iter)

# 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
```

```
# Bucle for para filtrado y suavizamiento
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) * 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)

  # 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]

  # Suavizador punto fijo
  M <- 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 * (x_hat_suav[k - 1] - phi *
x_hat_filt[k])
  P_suav[k] <- P_filt[k] - M^2 * P_filt[k]
}

# Gráfica de trayectoria del estado, observaciones, filtrado y
suavizamiento
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)
```



De nuevo se ha hecho para N=2. Como en el ejercicio 1g, la manera más eficiente sería definir un vector c(1,2,4) para hacer también N=1 y N=4 y posteriormente adaptar el bucle for para que vaya guardando los valores de las tres N. Definiendo a mano N=1, N=2 y N=4 y guardando los outputs con diferentes variables; y después, representándolo sale el gráfico:

```
plot(1:N_iter, e_filt21, type = 'l', col = 'blue', ylim = c(-1,2),
     ylab = 'Valor', xlab = 'k', main = 'Varianza de error de filtrado y
suavizado')
lines(1:N_iter, e_suav21, col = 'blue')
lines(1:N_iter, e_filt22, col = 'red')
lines(1:N_iter, e_suav22, col = 'red')
lines(1:N_iter, e_filt24, col = 'green')
lines(1:N_iter, e_suav24, col = 'green')
legend("topright", legend = c("N=1", "N=2", "N=4"), col = c("blue",
"red", "green"), lty = 1)
```

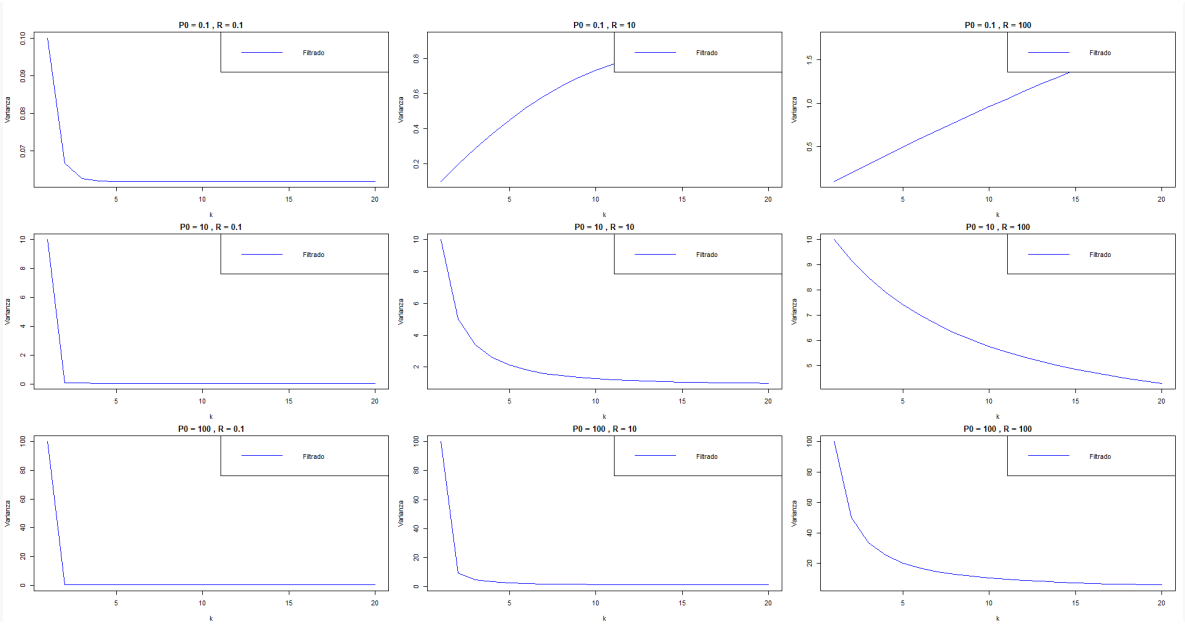


No, tienes que calcular P(k/k) y P(k/k+N)

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)

# Inicio el panel de gráficos (para mostrar varias gráficas en una)
par(mfrow = c(length(P0_values), length(R_values)), mar = c(4, 4, 2, 1),
oma = c(0, 0, 2, 0))

# 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]
    }

    # Gráfica de varianzas de errores de filtrado
    plot(1:N_iter, P_filt, type = 'l', col = 'blue', ylim =
c(min(P_filt), max(P_filt)),
         ylab = 'Varianza', xlab = 'k', main = paste('P0 =',
P0_values[i], ', R =', R_values[j]))
    legend("topright", legend = c("Filtrado"), col = c("blue"), lty = 1)
  }
}
```



Se puede ver el cambio de tendencia del filtrado al cambiar el parámetro, suavizándose las curvas cuando aumenta R. Además, se puede ver un cambio en el comportamiento de la función en relación a los parámetros, pasando de decreciente a creciente si P=0.1 y R aumenta.