EJERCICIOS DE REPASO

Arreglos unidimensionales o vectores

1. Suma de elementos pares e impares

Escribe un programa que reciba un vector de números enteros y calcule la suma de los elementos pares y la suma de los elementos impares por separado.

```
sumar_pares_impares <- function(vec) {
  suma_pares <- 0
  suma_impares <- 0

  for (num in vec) {
    if (num %% 2 == 0) {
      suma_pares <- suma_pares + num
    } else {
      suma_impares <- suma_impares + num
    }
  }

  return(list(Suma_Pares = suma_pares, Suma_Impares = suma_impares))
}

# Ejemplo de uso
vector <- c(1, 2, 3, 4, 5, 6)
sumar_pares_impares(vector)
```

2. Contar números positivos, negativos y ceros

Dado un vector de números enteros, cuenta cuántos números son positivos, cuántos negativos y cuántos son ceros. Usa un for y if.

```
contar_elementos <- function(vec) {
  pos <- 0
  neg <- 0
  ceros <- 0

  for (num in vec) {
    if (num > 0) {
      pos <- pos + 1
    } else if (num < 0) {
      neg <- neg + 1
    } else {
      ceros <- ceros + 1
    }
  }

  return(list(Positivos = pos, Negativos = neg, Ceros = ceros))
}
```

```
# Ejemplo de uso
vector <- c(-3, -1, 0, 5, 7, 0, -2, 8)
contar_elementos(vector)
```

3. Máximo y mínimo en un vector
Dado un vector numérico, encuentra el valor máximo y el mínimo sin usar las funciones max() ni min(). Usa un for y un if.

```
encontrar_max_min <- function(vec) {
  maximo <- vec[1]
  minimo <- vec[1]

  for (num in vec) {
    if (num > maximo) {
      maximo <- num
    }
    if (num < minimo) {
      minimo <- num
    }
  }

  return(list(Maximo = maximo, Minimo = minimo))
}

# Ejemplo de uso
vector <- c(7, 2, 9, 4, 1, 10)
encontrar_max_min(vector)
```

4. Buscar un elemento en un vector
Solicita al usuario un número y verifica si está en un vector dado. Usa while.

```
buscar_elemento <- function(vec, valor) {
  i <- 1
  encontrado <- FALSE

  while (i <= length(vec)) {
    if (vec[i] == valor) {
      encontrado <- TRUE
      break
    }
    i <- i + 1
  }

  return(encontrado)
}

# Ejemplo de uso
```

```
vector <- c(3, 7, 2, 9, 5)
buscar_elemento(vector, 9)  # Devuelve TRUE
buscar_elemento(vector, 10) # Devuelve FALSE
```

5. Suma acumulativa de un vector

Dado un vector numérico, crea un nuevo vector donde cada posición i almacene la suma acumulativa de los elementos hasta i. Usa un for.

Ejemplo:

Si el vector es c(1, 3, 5, 7), el resultado debe ser c(1, 4, 9, 16).

```
suma_acumulativa <- function(vec) {
  acumulado <- numeric(length(vec))
  acumulado[1] <- vec[1]

  for (i in 2:length(vec)) {
    acumulado[i] <- acumulado[i-1] + vec[i]
  }

  return(acumulado)
}

# Ejemplo de uso
vector <- c(1, 3, 5, 7)
suma_acumulativa(vector) # Resultado: c(1, 4, 9, 16)
```

6. Contar elementos mayores a un umbral

Escribe un programa que reciba un vector y cuente cuántos valores son mayores a un número dado.

```
contar_mayores <- function(vec, umbral) {
  contador <- 0

  for (num in vec) {
    if (num > umbral) {
      contador <- contador + 1
    }
  }

  return(contador)
}

# Ejemplo de uso
vector <- c(10, 5, 8, 12, 3, 15)
contar_mayores(vector, 7)  # Resultado: 4
```

7. Invertir un vector

Dado un vector numérico, invierte su orden sin usar rev(). Usa for o while.

```
invertir_vector <- function(vec) {
```

```
  invertido <- numeric(length(vec))
  j <- length(vec)

  for (i in 1:length(vec)) {
    invertido[j] <- vec[i]
    j <- j - 1
  }

  return(invertido)
}

# Ejemplo de uso
vector <- c(1, 2, 3, 4, 5)
invertir_vector(vector) # Resultado: c(5, 4, 3, 2, 1)
```

8. Filtrar números primos en un vector
Escribe una función que reciba un vector de números enteros y devuelva un nuevo vector solo con los números primos.

```
es_primo <- function(n) {
  if (n < 2) return(FALSE)
  for (i in 2:sqrt(n)) {
    if (n %% i == 0) return(FALSE)
  }
  return(TRUE)
}

filtrar_primos <- function(vec) {
  primos <- c()

  for (num in vec) {
    if (es_primo(num)) {
      primos <- c(primos, num)
    }
  }

  return(primos)
}

# Ejemplo de uso
vector <- c(2, 3, 4, 5, 10, 13, 17, 20)
filtrar_primos(vector) # Resultado: c(2, 3, 5, 13, 17)
```

9. Encontrar el segundo número mayor
Dado un vector numérico, encuentra el segundo número más grande sin usar sort(). Usa if y for.

```
segundo_maximo <- function(vec) {
  max1 <- -Inf
  max2 <- -Inf
```

```
  for (num in vec) {
    if (num > max1) {
      max2 <- max1
      max1 <- num
    } else if (num > max2 && num != max1) {
      max2 <- num
    }
  }

  return(max2)
}

# Ejemplo de uso
vector <- c(10, 5, 8, 12, 3, 15)
segundo_maximo(vector)  # Resultado: 12
```

10. Contar repeticiones de un número en un vector
Escribe un programa que reciba un vector y un número, y cuente cuántas veces aparece en el vector. Usa un for y un if.

```
contar_repeticiones <- function(vec, num) {
  contador <- 0

  for (elemento in vec) {
    if (elemento == num) {
      contador <- contador + 1
    }
  }

  return(contador)
}

# Ejemplo de uso
vector <- c(3, 5, 3, 8, 3, 10, 3)
contar_repeticiones(vector, 3)  # Resultado: 4
```

Arreglos bidimensionales o matrices

1. Suma de dos matrices
Escribe una función que reciba dos matrices del mismo tamaño y devuelva su suma sin usar +.

```
sumar_matrices <- function(A, B) {
  if (!all(dim(A) == dim(B))) {
    stop("Las matrices deben tener el mismo tamaño.")
  }

  resultado <- matrix(0, nrow = nrow(A), ncol = ncol(A))
```

```
  for (i in 1:nrow(A)) {
    for (j in 1:ncol(A)) {
      resultado[i, j] <- A[i, j] + B[i, j]
    }
  }

  return(resultado)
}

# Ejemplo de uso
A <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
B <- matrix(c(6, 5, 4, 3, 2, 1), nrow = 2)
sumar_matrices(A, B)
```

2. Encontrar el mayor y menor valor en una matriz
Dada una matriz numérica, encuentra el valor máximo y mínimo sin usar max() ni min().

```
encontrar_max_min <- function(mat) {
  maximo <- mat[1,1]
  minimo <- mat[1,1]

  for (i in 1:nrow(mat)) {
    for (j in 1:ncol(mat)) {
      if (mat[i, j] > maximo) {
        maximo <- mat[i, j]
      }
      if (mat[i, j] < minimo) {
        minimo <- mat[i, j]
      }
    }
  }

  return(list(Maximo = maximo, Minimo = minimo))
}

# Ejemplo de uso
M <- matrix(c(3, 7, 2, 9, 5, 1), nrow = 2)
encontrar_max_min(M)
```

3. Calcular la suma de cada fila y de cada columna
Dada una matriz numérica, devuelve dos vectores: uno con la suma de cada fila y otro con la suma de cada columna.

```
sumar_filas_columnas <- function(mat) {
  sum_filas <- numeric(nrow(mat))
  sum_columnas <- numeric(ncol(mat))
```

```
  for (i in 1:nrow(mat)) {
    sum_filas[i] <- sum(mat[i, ])
  }

  for (j in 1:ncol(mat)) {
    sum_columnas[j] <- sum(mat[, j])
  }

  return(list(Suma_Filas = sum_filas, Suma_Columnas = sum_columnas))
}

# Ejemplo de uso
M <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
sumar_filas_columnas(M)
```

4. Matriz identidad sin usar diag()
Escribe una función que genere una matriz identidad de tamaño n × n sin usar diag().

```
matriz_identidad <- function(n) {
  mat <- matrix(0, nrow = n, ncol = n)

  for (i in 1:n) {
    mat[i, i] <- 1
  }

  return(mat)
}

# Ejemplo de uso
matriz_identidad(4)
```

5. Transponer una matriz sin usar t()
Dada una matriz, implementa una función que intercambie las filas por las columnas sin usar t().

```
transponer_matriz <- function(mat) {
  transpuesta <- matrix(0, nrow = ncol(mat), ncol = nrow(mat))

  for (i in 1:nrow(mat)) {
    for (j in 1:ncol(mat)) {
      transpuesta[j, i] <- mat[i, j]
    }
  }

  return(transpuesta)
}

# Ejemplo de uso
```

```
M <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
transponer_matriz(M)
```

6. Contar cuántos valores negativos tiene una matriz
Dada una matriz de números, cuenta cuántos valores son negativos.

```
contar_negativos <- function(mat) {
  contador <- 0

  for (i in 1:nrow(mat)) {
    for (j in 1:ncol(mat)) {
      if (mat[i, j] < 0) {
        contador <- contador + 1
      }
    }
  }

  return(contador)
}

# Ejemplo de uso
M <- matrix(c(-1, 2, -3, 4, -5, 6), nrow = 2)
contar_negativos(M)
```

7. Intercambiar dos filas de una matriz
Escribe una función que reciba una matriz y dos índices de fila, e intercambie esas dos filas entre sí.

```
intercambiar_filas <- function(mat, fila1, fila2) {
  temp <- mat[fila1, ]
  mat[fila1, ] <- mat[fila2, ]
  mat[fila2, ] <- temp
  return(mat)
}

# Ejemplo de uso
M <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, byrow = TRUE)
intercambiar_filas(M, 1, 3)
```

8. Multiplicación de dos matrices sin usar %*%
Implementa la multiplicación de dos matrices A y B de tamaño compatible sin usar %*%.

```
multiplicar_matrices <- function(A, B) {
  if (ncol(A) != nrow(B)) {
    stop("No se pueden multiplicar matrices con estas dimensiones.")
  }

  resultado <- matrix(0, nrow = nrow(A), ncol = ncol(B))
```

```
  for (i in 1:nrow(A)) {
    for (j in 1:ncol(B)) {
      suma <- 0
      for (k in 1:ncol(A)) {
        suma <- suma + A[i, k] * B[k, j]
      }
      resultado[i, j] <- suma
    }
  }

  return(resultado)
}

# Ejemplo de uso
A <- matrix(c(1, 2, 3, 4), nrow = 2)
B <- matrix(c(5, 6, 7, 8), nrow = 2)
multiplicar_matrices(A, B)
```

9. Determinar si una matriz es simétrica
Una matriz es simétrica si es cuadrada y A[i, j] == A[j, i] para todos sus elementos. Implementa una función que lo verifique.

```
es_simetrica <- function(mat) {
  if (nrow(mat) != ncol(mat)) {
    return(FALSE)
  }

  for (i in 1:nrow(mat)) {
    for (j in 1:ncol(mat)) {
      if (mat[i, j] != mat[j, i]) {
        return(FALSE)
      }
    }
  }

  return(TRUE)
}

# Ejemplo de uso
M <- matrix(c(1, 2, 2, 3), nrow = 2)
es_simetrica(M)
```

10. Contar cuántos números pares e impares hay en una matriz
Escribe una función que reciba una matriz y devuelva cuántos valores pares e impares contiene.

```
contar_pares_impares <- function(mat) {
  pares <- 0
  impares <- 0
```

```
  for (i in 1:nrow(mat)) {
    for (j in 1:ncol(mat)) {
      if (mat[i, j] %% 2 == 0) {
        pares <- pares + 1
      } else {
        impares <- impares + 1
      }
    }
  }

  return(list(Pares = pares, Impares = impares))
}

# Ejemplo de uso
M <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2)
contar_pares_impares(M)
```

Listas

1. Crear una lista con diferentes tipos de datos
Crea una lista que contenga:
Un vector numérico.
Un vector de caracteres.
Una matriz.
Imprime la lista y accede a cada elemento individualmente.

```
mi_lista <- list(
  numeros = c(1, 2, 3, 4, 5),
  caracteres = c("a", "b", "c"),
  matriz = matrix(1:9, nrow = 3)
)

# Imprimir la lista completa
print(mi_lista)

# Acceder a elementos individuales
print(mi_lista$numeros)   # Acceder al vector numérico
print(mi_lista$matriz)    # Acceder a la matriz
```

2. Concatenar dos listas
Dadas dos listas, combínalas en una sola sin perder elementos.
Ejemplo:
Si lista1 = list(a=1, b=2) y lista2 = list(c=3, d=4),
el resultado debe ser list(a=1, b=2, c=3, d=4).

```
lista1 <- list(a = 1, b = 2)
lista2 <- list(c = 3, d = 4)
```

```
lista_combinada <- c(lista1, lista2)

print(lista_combinada)
```

3. Acceder y modificar elementos dentro de una lista
Dada una lista, accede al tercer elemento y modifícalo con un nuevo valor.
Ejemplo:
Si L = list(10, 20, 30), cambia el tercer elemento a 100.

```
L <- list(10, 20, 30)
# Modificar el tercer elemento
L[[3]] <- 100
print(L)
```

4. Calcular estadísticas básicas dentro de una lista
Crea una lista con tres vectores numéricos y calcula para cada vector:

Media
Mediana
Varianza
Devuelve una nueva lista con estos valores.

```
lista_vectores <- list(
  vec1 = c(1, 2, 3, 4, 5),
  vec2 = c(6, 7, 8, 9, 10),
  vec3 = c(11, 12, 13, 14, 15)
)

estadisticas <- lapply(lista_vectores, function(x) {
  list(
    media = mean(x),
    mediana = median(x),
    varianza = var(x)
  )
})

print(estadisticas)
```

5. Filtrar elementos de una lista
Dada una lista con diferentes vectores, devuelve una nueva lista que solo contenga los vectores que tienen más de 3 elementos.
Ejemplo:
Si L = list(c(1,2), c(3,4,5,6), c(7,8,9)),
el resultado debe ser list(c(3,4,5,6), c(7,8,9)).

```
lista_vectores <- list(
  c(1, 2),
  c(3, 4, 5, 6),
```

```
  c(7, 8, 9)
)

lista_filtrada <- Filter(function(x) length(x) > 3, lista_vectores)

print(lista_filtrada)
```

6. Convertir una lista en una matriz o dataframe
Dada una lista con tres vectores del mismo tamaño, conviértela en una matriz o un dataframe.
Ejemplo:
Si L = list(c(1,2,3), c(4,5,6), c(7,8,9)),
convierte L en la matriz:
Copiar
Editar
1 4 7
2 5 8
3 6 9

```
lista_datos <- list(
  c(1, 2, 3),
  c(4, 5, 6),
  c(7, 8, 9)
)

# Convertir a matriz
matriz_resultado <- do.call(cbind, lista_datos)
print(matriz_resultado)

# Convertir a dataframe
dataframe_resultado <- as.data.frame(lista_datos)
print(dataframe_resultado)
```

7. Aplicar una función a cada elemento de una lista
Dada una lista de números, usa lapply() o sapply() para elevar al cuadrado cada elemento.

```
lista_numeros <- list(1, 2, 3, 4, 5)

# Elevar cada número al cuadrado usando lapply()
cuadrados <- lapply(lista_numeros, function(x) x^2)

print(cuadrados)
```

8. Crear una lista anidada y acceder a elementos internos
Crea una lista dentro de otra lista y accede a un elemento interno.
Ejemplo:
Si L = list(a=1, b=list(c=2, d=3)),
accede al valor d dentro de la sublista b.

```
lista_anidada <- list(
  a = 1,
  b = list(c = 2, d = 3)
)

# Acceder al elemento "d" dentro de la sublista "b"
print(lista_anidada$b$d)
```

9. Ordenar una lista de vectores numéricos
Dada una lista con varios vectores numéricos, ordena cada vector dentro de la lista de menor a mayor.
Ejemplo:
Si L = list(c(5,2,9), c(3,8,1)),
devuelve list(c(2,5,9), c(1,3,8)).

```
lista_vectores <- list(
  c(5, 2, 9),
  c(3, 8, 1)
)

lista_ordenada <- lapply(lista_vectores, sort)

print(lista_ordenada)
```

10. Contar la cantidad de elementos en cada vector dentro de una lista
Dada una lista con diferentes vectores, devuelve un vector con la cantidad de elementos de cada vector.
Ejemplo:
Si L = list(c(1,2,3), c(4,5), c(6,7,8,9)),
el resultado debe ser c(3,2,4).

```
lista_vectores <- list(
  c(5, 2, 9),
  c(3, 8, 1)
)

lista_ordenada <- lapply(lista_vectores, sort)

print(lista_ordenada)
```

Función apply( )

1. Sumar los valores de cada fila de una matriz
Dada una matriz numérica, usa apply() para calcular la suma de cada fila.

```
# Crear una matriz de ejemplo
matriz <- matrix(1:12, nrow = 4, ncol = 3)

# Sumar los valores de cada fila
```

```
suma_filas <- apply(matriz, 1, sum)

print(suma_filas)
```

2. Calcular la media de cada columna de una matriz
Dada una matriz numérica, usa apply() para calcular la media de cada columna.

```
# Crear una matriz de ejemplo
matriz <- matrix(1:12, nrow = 4, ncol = 3)

# Calcular la media de cada columna
media_columnas <- apply(matriz, 2, mean)

print(media_columnas)
```

3. Encontrar el máximo de cada fila de una matriz
Dada una matriz, usa apply() para devolver un vector con el máximo valor de cada fila.

```
# Crear una matriz de ejemplo
matriz <- matrix(sample(1:100, 12), nrow = 4)

# Encontrar el máximo de cada fila
max_filas <- apply(matriz, 1, max)

print(max_filas)
```

4. Calcular la varianza de cada columna de una matriz
Usa apply() para calcular la varianza de cada columna.

```
# Crear una matriz de ejemplo
matriz <- matrix(rnorm(12), nrow = 4)

# Calcular la varianza de cada columna
varianza_columnas <- apply(matriz, 2, var)

print(varianza_columnas)
```

5. Contar cuántos valores negativos tiene cada fila
Dada una matriz numérica, usa apply() para contar cuántos valores negativos tiene cada fila.

```
# Crear una matriz con valores positivos y negativos
set.seed(123)
matriz <- matrix(rnorm(12, mean = 0, sd = 5), nrow = 4)
```

```
# Contar cuántos valores negativos tiene cada fila
negativos_filas <- apply(matriz, 1, function(x) sum(x < 0))

print(negativos_filas)
```

6. Normalizar una matriz por filas
Dada una matriz, usa apply() para restarle a cada fila su media y dividirla entre su desviación estándar.

```
# Crear una matriz de ejemplo
set.seed(123)
matriz <- matrix(rnorm(12), nrow = 4)

# Normalizar por filas (restar la media y dividir por la desviación estándar)
normalizar_filas <- apply(matriz, 1, function(x) (x - mean(x)) / sd(x))

print(normalizar_filas)
```

7. Aplicar una función personalizada a cada fila
Dada una matriz, usa apply() con una función personalizada que reste el mínimo de cada fila a todos sus elementos.

```
# Crear una matriz de ejemplo
matriz <- matrix(sample(1:50, 12), nrow = 4)

# Restar el mínimo de cada fila a todos sus elementos
restar_min_filas <- apply(matriz, 1, function(x) x - min(x))

print(restar_min_filas)
```

8. Convertir una lista de vectores en una matriz
Dada una lista con vectores del mismo tamaño, usa sapply() o apply() para convertirla en una matriz.

```
# Crear una lista con vectores del mismo tamaño
lista_vectores <- list(
  c(1, 2, 3),
  c(4, 5, 6),
  c(7, 8, 9)
)

# Convertir la lista en una matriz
matriz_convertida <- do.call(cbind, lista_vectores)

print(matriz_convertida)
```

9. Contar los elementos pares en cada columna
Dada una matriz, usa apply() para contar cuántos números pares hay en cada columna.

```
# Crear una matriz con números enteros
matriz <- matrix(sample(1:20, 12, replace = TRUE), nrow = 4)

# Contar cuántos números pares hay en cada columna
contar_pares <- apply(matriz, 2, function(x) sum(x %% 2 == 0))

print(contar_pares)
```

Concatenación de vectores y matrices

1. Unir dos vectores en una matriz con cbind()
Dado dos vectores A y B, únelos en una matriz como columnas usando cbind().

```
# Definir vectores
A <- c(1, 2, 3, 4)
B <- c(5, 6, 7, 8)

# Unir los vectores en columnas
matriz_cbind <- cbind(A, B)

print(matriz_cbind)
```

2. Unir dos vectores en una matriz con rbind()
Dado dos vectores A y B, únelos en una matriz como filas usando rbind().

```
# Definir vectores
A <- c(1, 2, 3, 4)
B <- c(5, 6, 7, 8)

# Unir los vectores en filas
matriz_rbind <- rbind(A, B)

print(matriz_rbind)
```

3. Agregar una nueva columna a una matriz con cbind()
Dada una matriz M, agrega un nuevo vector V como una nueva columna.

```
# Crear matriz inicial
M <- matrix(1:9, nrow = 3, byrow = TRUE)

# Nuevo vector columna
V <- c(10, 11, 12)
```

```
# Agregar columna a la matriz
M_nueva <- cbind(M, V)

print(M_nueva)
```

4. Agregar una nueva fila a una matriz con rbind()
Dada una matriz M, agrega un nuevo vector V como una nueva fila.

```
# Crear matriz inicial
M <- matrix(1:9, nrow = 3, byrow = TRUE)

# Nueva fila
V <- c(10, 11, 12)

# Agregar fila a la matriz
M_nueva <- rbind(M, V)

print(M_nueva)
```

5. Fusionar dos dataframes con cbind()
Dado dos dataframes con la misma cantidad de filas, únelos por columnas usando cbind().

```
# Crear dos dataframes
df1 <- data.frame(A = c(1, 2, 3), B = c("a", "b", "c"))
df2 <- data.frame(C = c(4, 5, 6), D = c("d", "e", "f"))

# Unir dataframes por columnas
df_combinado <- cbind(df1, df2)

print(df_combinado)
```

6. Fusionar dos dataframes con rbind()
Dado dos dataframes con las mismas columnas, únelos por filas usando rbind().

```
# Crear dos dataframes con las mismas columnas
df1 <- data.frame(A = c(1, 2), B = c("a", "b"))
df2 <- data.frame(A = c(3, 4), B = c("c", "d"))

# Unir dataframes por filas
df_combinado <- rbind(df1, df2)

print(df_combinado)
```

7. Crear una matriz a partir de múltiples vectores con cbind()
Dado tres vectores X, Y, Z, únelos para formar una matriz.

```
# Definir vectores
X <- c(1, 2, 3)
```

```
Y <- c(4, 5, 6)
Z <- c(7, 8, 9)

# Unir vectores en una matriz (por columnas)
matriz_cbind <- cbind(X, Y, Z)

print(matriz_cbind)
```

8. Crear una matriz a partir de múltiples vectores con rbind()
Dado tres vectores X, Y, Z, únelos para formar una matriz donde cada vector sea una fila.

```
# Definir vectores
X <- c(1, 2, 3)
Y <- c(4, 5, 6)
Z <- c(7, 8, 9)

# Unir vectores en una matriz (por filas)
matriz_rbind <- rbind(X, Y, Z)

print(matriz_rbind)
```

9. Unir múltiples listas en una matriz con cbind()
Dadas tres listas con elementos del mismo tamaño, únelas en una matriz con cbind().

```
# Definir listas con elementos del mismo tamaño
lista1 <- list(c(1, 2, 3))
lista2 <- list(c(4, 5, 6))
lista3 <- list(c(7, 8, 9))

# Convertir listas en una matriz
matriz_cbind <- cbind(unlist(lista1), unlist(lista2), unlist(lista3))

print(matriz_cbind)
```

10. Unir múltiples listas en una matriz con rbind()
Dadas tres listas con elementos del mismo tamaño, únelas en una matriz con rbind().

```
# Definir listas con elementos del mismo tamaño
lista1 <- list(c(1, 2, 3))
lista2 <- list(c(4, 5, 6))
lista3 <- list(c(7, 8, 9))

# Convertir listas en una matriz
matriz_rbind <- rbind(unlist(lista1), unlist(lista2), unlist(lista3))

print(matriz_rbind)
```