---

title: “2.0” author: “Vantablack” date: “2025-04-21” output: html_document

Agrupación K-Means

En el enfoque de partición de R, las observaciones se dividen en grupos K y se reorganizan para formar los grupos más cohesivos posibles de acuerdo con un criterio dado. Hay dos métodos: K-medias y partición alrededor de mediodes PAM. En este artículo, basado en el capítulo 16 de R in Action, Second Edition, el autor Rob Kabaco analiza la agrupación de K-means. El algoritmo k-means podría utilizarse en el contexto del mantenimiento industrial. Supongamos que tienes un conjunto de datos históricos sobre el rendimiento y la vida útil de diferentes componentes de maquinaria en una planta industrial. Estos datos pueden incluir variables como la temperatura de funcionamiento, la vibración, el consumo de energía, la carga de trabajo, entre otros. El objetivo sería utilizar el algoritmo k-means para identi car grupos o clus ters de componentes que tengan un comportamiento similar en términos de su rendimiento y vida útil. Esto puede ayudar a los ingenieros y operarios de man tenimiento a entender mejor el comportamiento de los componentes y tomar decisiones más informadas sobre el mantenimiento preventivo o la sustitución de los mismos. El proceso podría seguir los siguientes pasos: Preparación de datos: Reunir y preparar los datos históricos relevantes de los componentes de maquinaria. Esto puede incluir la recopilación de variables, como la temperatura, la vibración, el consumo de energía, etc., para cada componente en diferentes momentos. Selección de características: Identi car las características relevantes para el análisis. Puedes realizar un análisis de correlación o utilizar el conocimiento experto para seleccionar las variables más signi cativas que pueden inuir en el rendimiento y la vida útil de los componentes. Normalización de datos: Normalizar los datos para asegurarse de que todas las características tengan la misma escala y rango. Esto es importante para que todas las variables tengan una inuencia equitativa en el algoritmo k means. Elección del número de clusters (k): Determinar el número de clusters que deseas encontrar. Puedes utilizar métodos como el método del codo (elbow method) o la silueta (silhouette) para evaluar diferentes valores de k y selec cionar el más adecuado. Aplicación del algoritmo k-means: Ejecutar el algoritmo k-means en los datos normalizados, agrupando los componentes en los k clusters determinados. El algoritmo asignará cada componente al cluster más cercano en función de las características seleccionadas. Análisis e interpretación de los resultados: Analizar los resultados del clus tering para identi car patrones y comportamientos similares entre los com ponentes. Puedes examinar las características de cada cluster para entender qué factores pueden estar relacionados con un mayor rendimiento o una vida útil más larga. Esto puede ayudar a tomar decisiones de mantenimiento más precisas y e cientes. Programación en R 11 Atención que este es solo un ejemplo de cómo se podría aplicar el algoritmo k means en el mantenimiento industrial. La selección de variables, la normalización de datos y la interpretación de los resultados pueden variar según el contexto y los objetivos especí cos del estudio de mantenimiento. No es lo mismo utilizar esta técnica para mantenimiento en la industria alimenticia que en la industria nuclear. Cada campo de aplicación debería respetar los procedimiento Hazop sugerido por las normas y buenas prácticas recomendadas

```
# Cargamos las librerías necesarias
library(tidyverse)  # Para manipulación de datos
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(cluster)    # Para análisis de clustering
library(factoextra) # Para visualización de clusters
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
library(gridExtra)  # Para organizar gráficos
```

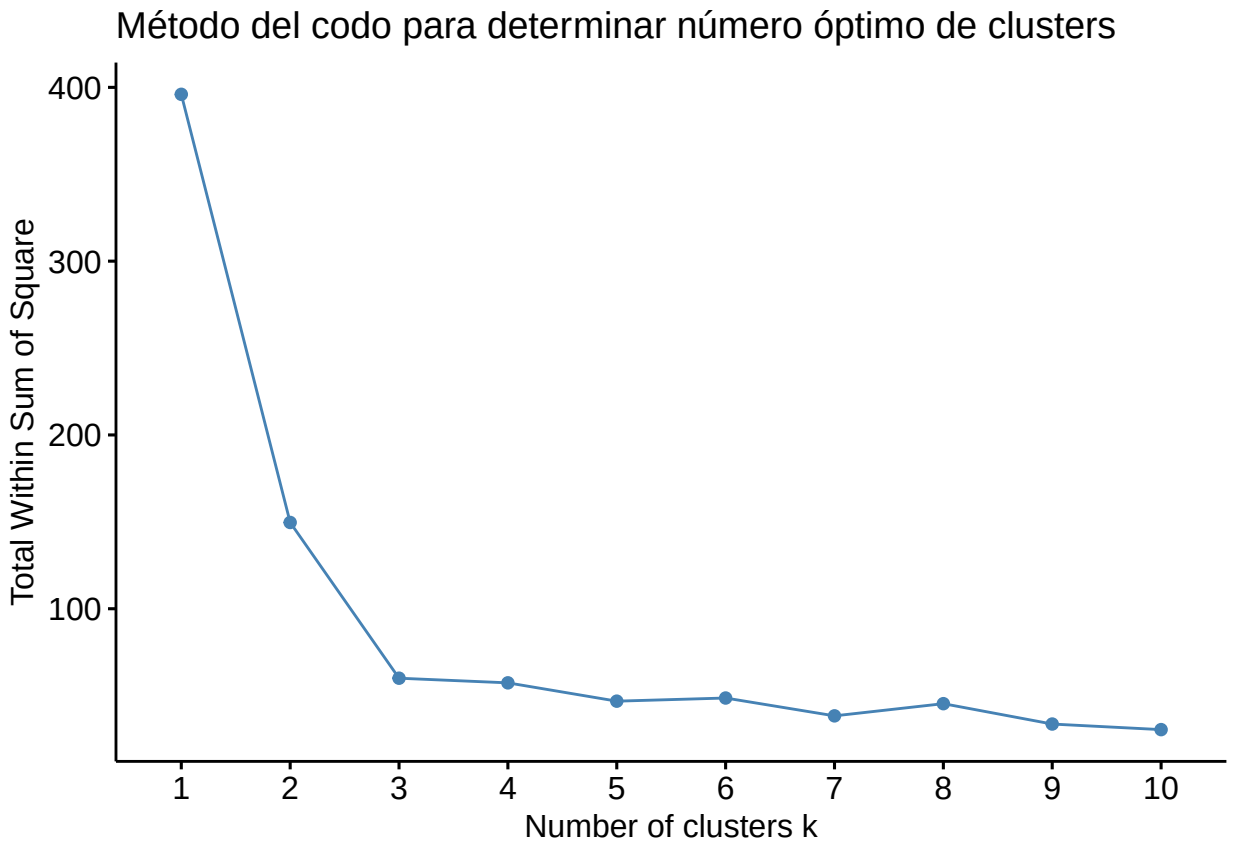
```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
# 1. Generamos datos simulados de componentes industriales
set.seed(123)  # Para reproducibilidad

# Simulamos 100 componentes con 5 variables:
# - temperatura: temperatura de operación en °C
# - vibracion: nivel de vibración (mm/s)
# - energia: consumo de energía (kW)
# - carga: porcentaje de carga de trabajo (%)
# - vida_util: tiempo de vida útil en horas

n_componentes <- 100

# Creamos tres grupos de componentes con características diferentes
# Grupo 1: Componentes de alta durabilidad (baja temperatura, baja vibración)
# Grupo 2: Componentes de durabilidad media
# Grupo 3: Componentes de baja durabilidad (alta temperatura, alta vibración)

grupo1 <- data.frame(
  id = paste0("C", 1:35),
  temperatura = rnorm(35, mean = 50, sd = 5),
  vibracion = rnorm(35, mean = 2, sd = 0.3),
  energia = rnorm(35, mean = 10, sd = 1),
  carga = rnorm(35, mean = 60, sd = 5),
  vida_util = rnorm(35, mean = 12000, sd = 800)
)

grupo2 <- data.frame(
  id = paste0("C", 36:70),
  temperatura = rnorm(35, mean = 65, sd = 6),
  vibracion = rnorm(35, mean = 4, sd = 0.5),
  energia = rnorm(35, mean = 15, sd = 1.5),
  carga = rnorm(35, mean = 75, sd = 6),
  vida_util = rnorm(35, mean = 8000, sd = 700)
)

grupo3 <- data.frame(
  id = paste0("C", 71:100),
  temperatura = rnorm(30, mean = 80, sd = 7),
  vibracion = rnorm(30, mean = 6, sd = 0.8),
  energia = rnorm(30, mean = 20, sd = 2),
  carga = rnorm(30, mean = 90, sd = 7),
  vida_util = rnorm(30, mean = 5000, sd = 600)
)
```

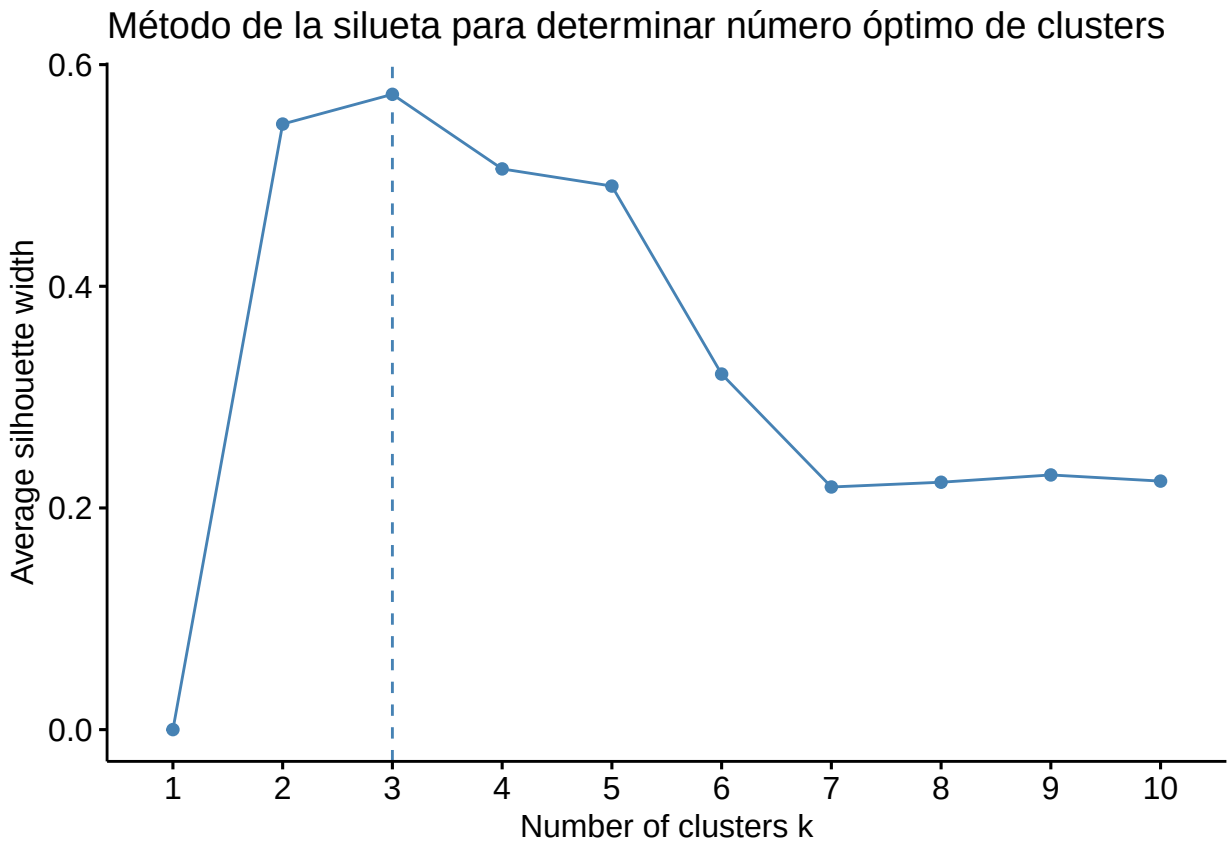
```r
# Combinamos los grupos
datos_componentes <- rbind(grupo1, grupo2, grupo3)

# Visualizamos los primeros registros
head(datos_componentes)
```

```
##   id temperatura vibracion   energia    carga vida_util
## 1 C1    47.19762  2.206592  9.508969 59.77486  12561.43
## 2 C2    48.84911  2.166175  7.690831 56.07548  11790.24
## 3 C3    57.79354  1.981426 11.005739 51.66029  10742.28
## 4 C4    50.35254  1.908211  9.290799 58.09887  10788.27
## 5 C5    50.64644  1.885859  9.311991 64.59498  10718.77
## 6 C6    58.57532  1.791588 11.025571 57.12327  11575.27
```

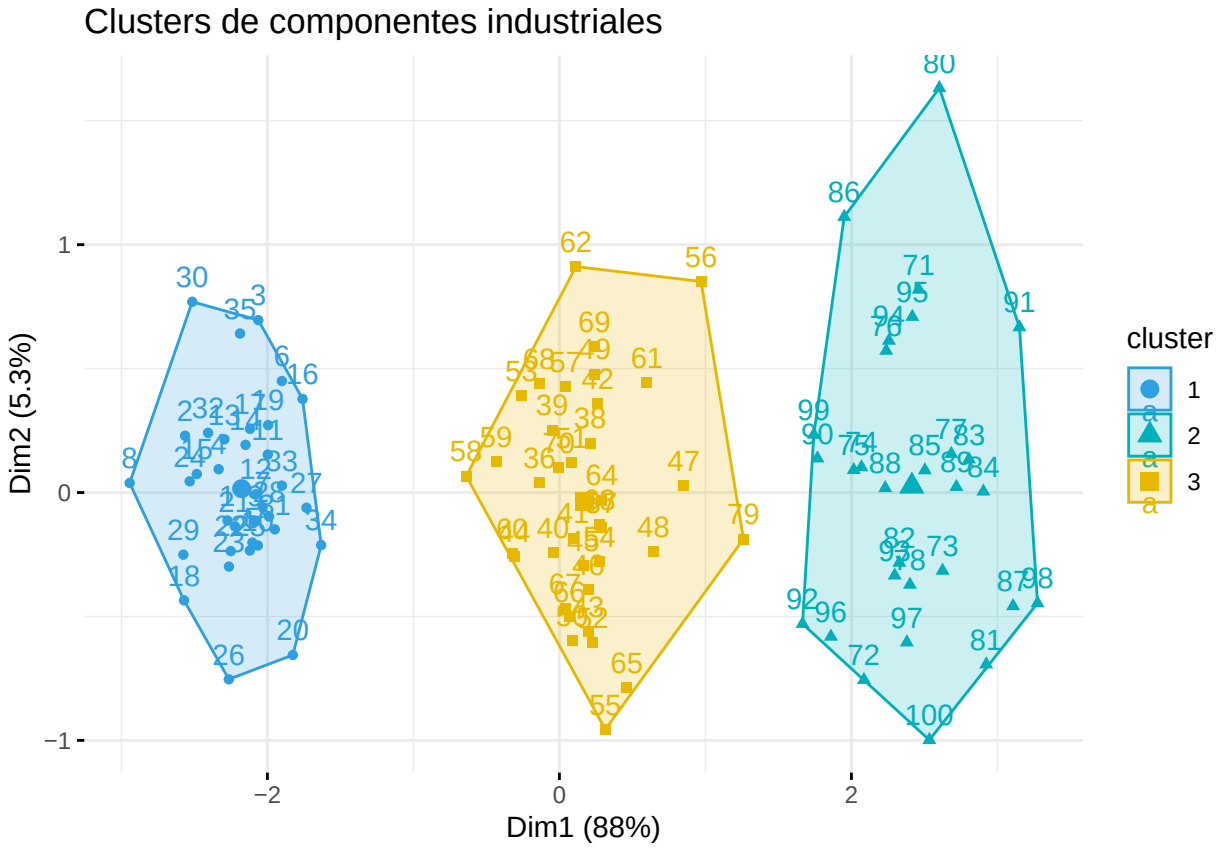
```r
# 2. Normalización de datos
# Guardamos la columma ID y vida útil (para análisis posterior)
componentes_id <- datos_componentes$id
vida_util_real <- datos_componentes$vida_util

# Seleccionamos solo las variables numéricas para clustering
datos_scaled <- datos_componentes %>%
  select(-id, -vida_util) %>%
  scale()

# 3. Determinar el número óptimo de clusters (método del codo y silueta)
# Método del codo
set.seed(123)
fviz_nbclust(datos_scaled, kmeans, method = "wss", k.max = 10) +
  labs(title = "Método del codo para determinar número óptimo de clusters")
```




```r
# Método de la silueta
set.seed(123)
fviz_nbclust(datos_scaled, kmeans, method = "silhouette", k.max = 10) +
  labs(title = "Método de la silueta para determinar número óptimo de clusters")
```




```r
# 4. Aplicar K-means con el número óptimo de clusters (k=3 en este caso)
set.seed(123)
k <- 3
kmeans_result <- kmeans(datos_scaled, centers = k, nstart = 25)

# 5. Análisis de resultados
# Añadimos los clusters asignados a nuestros datos originales
datos_componentes$cluster <- as.factor(kmeans_result$cluster)

# Visualización de clusters
fviz_cluster(list(data = datos_scaled, cluster = kmeans_result$cluster),
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             ellipse.type = "convex",
             ggtheme = theme_minimal(),
             main = "Clusters de componentes industriales")
```



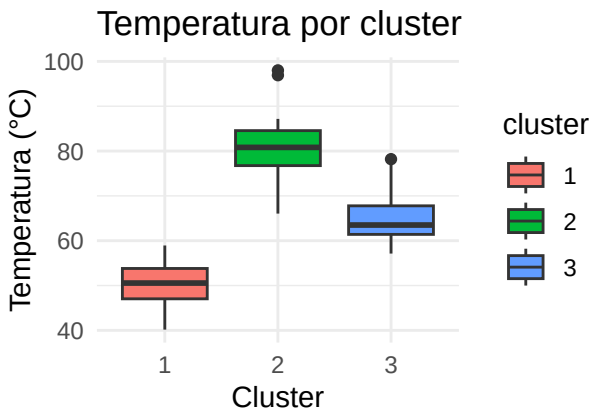

```
# Análisis de características por cluster
cluster_stats <- datos_componentes %>%
  group_by(cluster) %>%
  summarise(
    n = n(),
    temp_media = mean(temperatura),
    vibra_media = mean(vibracion),
    energia_media = mean(energia),
    carga_media = mean(carga),
    vida_util_media = mean(vida_util),
    sd_vida_util = sd(vida_util)
  )

print(cluster_stats)
```

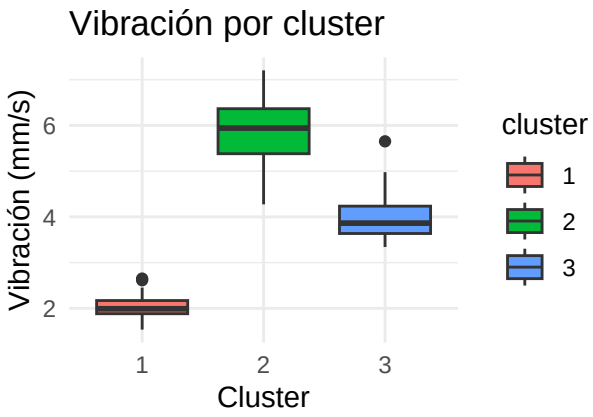
```
## # A tibble: 3 x 8
##   cluster     n temp_media vibra_media energia_media carga_media vida_util_media
##   <fct>   <int>      <dbl>       <dbl>         <dbl>       <dbl>           <dbl>
## 1 1          35       50.2        2.03          10.1        59.1          12015.
## 2 2          29       80.4        5.85          20.1        91.4           5137.
## 3 3          36       64.9        3.99          15.4        75.9           7978.
## # i 1 more variable: sd_vida_util <dbl>
```

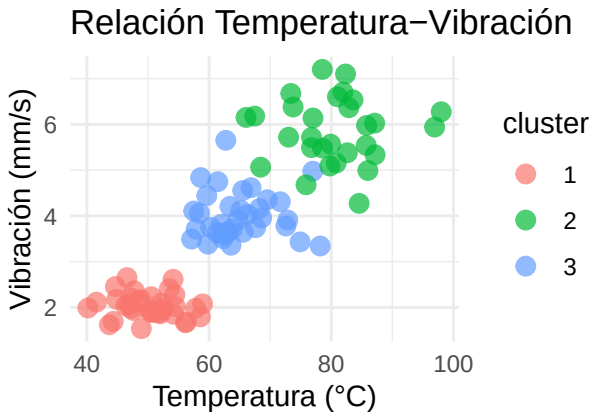
```
# Visualizamos las distribuciones de vida útil por cluster
ggplot(datos_componentes, aes(x = cluster, y = vida_util, fill = cluster)) +
  geom_boxplot() +
  labs(title = "Distribución de vida útil por cluster",
       x = "Cluster",
```

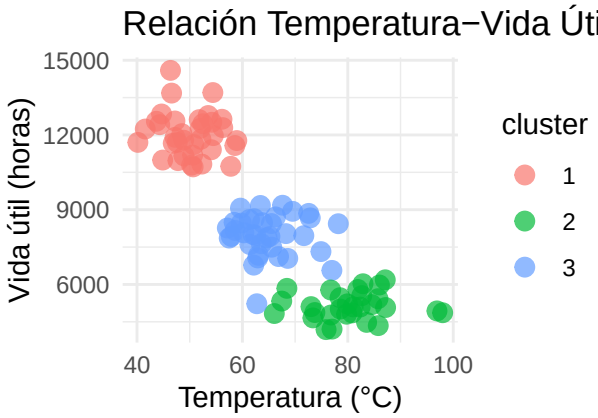
```
       y = "Vida útil (horas)") +
  theme_minimal()
```


```
# Gráficos para variables críticas por cluster
p1 <- ggplot(datos_componentes, aes(x = cluster, y = temperatura, fill = cluster)) +
  geom_boxplot() +
  labs(title = "Temperatura por cluster", x = "Cluster", y = "Temperatura (°C)") +
  theme_minimal()

p2 <- ggplot(datos_componentes, aes(x = cluster, y = vibracion, fill = cluster)) +
  geom_boxplot() +
  labs(title = "Vibración por cluster", x = "Cluster", y = "Vibración (mm/s)") +
  theme_minimal()

p3 <- ggplot(datos_componentes, aes(x = temperatura, y = vibracion, color = cluster)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Relación Temperatura-Vibración", x = "Temperatura (°C)", y = "Vibración (mm/s)") +
  theme_minimal()

p4 <- ggplot(datos_componentes, aes(x = temperatura, y = vida_util, color = cluster)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Relación Temperatura-Vida Útil", x = "Temperatura (°C)", y = "Vida útil (horas)") +
  theme_minimal()

grid.arrange(p1, p2, p3, p4, ncol = 2)
```










```r
# 6. Predicción de componentes - Ejemplo práctico
# Simulamos un nuevo componente
nuevo_componente <- data.frame(
  temperatura = 75,
  vibracion = 5.5,
  energia = 18,
  carga = 85
)

# Normalizamos usando la misma escala que los datos originales
# Primero obtenemos los parámetros de normalización de los datos originales
medias <- colMeans(datos_componentes[, c("temperatura", "vibracion", "energia", "carga")])
desviaciones <- apply(datos_componentes[, c("temperatura", "vibracion", "energia", "carga")], 2, sd)

# Normalizamos el nuevo componente
nuevo_componente_scaled <- scale(nuevo_componente, center = medias, scale = desviaciones)

# Calculamos las distancias a cada centro de cluster
distancias <- apply(kmeans_result$centers, 1, function(centro) {
  sqrt(sum((nuevo_componente_scaled - centro)^2))
})

cluster_predicho <- which.min(distancias)

cat("El nuevo componente probablemente pertenece al cluster:", cluster_predicho, "\n")
```

```
## El nuevo componente probablemente pertenece al cluster: 2
```

```r
cat("Vida útil estimada (basada en la media del cluster):",
    round(cluster_stats$vida_util_media[cluster_predicho], 2), "horas\n")
```

```
## Vida útil estimada (basada en la media del cluster): 5137.05 horas
```

```r
# 7. Conclusiones y recomendaciones para mantenimiento
cat("\nRecomendaciones de mantenimiento por cluster:\n")
```

```
##
## Recomendaciones de mantenimiento por cluster:
```

```r
for (i in 1:k) {
  cat("Cluster", i, ":\n")
  cat("  • Componentes:", cluster_stats$n[i], "\n")
  cat("  • Vida útil media:", round(cluster_stats$vida_util_media[i], 0), "horas\n")
  cat("  • Características principales: Temperatura media =",
      round(cluster_stats$temp_media[i], 1), "°C, Vibración media =",
      round(cluster_stats$vibra_media[i], 1), "mm/s\n")

  # Recomendaciones específicas según características del cluster
  if (cluster_stats$vida_util_media[i] > 10000) {
    cat("  • Recomendación: Mantenimiento preventivo cada",
        round(cluster_stats$vida_util_media[i] * 0.7, 0), "horas\n")
  } else if (cluster_stats$vida_util_media[i] > 7000) {
    cat("  • Recomendación: Mantenimiento preventivo cada",
        round(cluster_stats$vida_util_media[i] * 0.6, 0), "horas y monitoreo mensual\n")
  } else {
    cat("  • Recomendación: Mantenimiento preventivo cada",
        round(cluster_stats$vida_util_media[i] * 0.5, 0),
        "horas y monitoreo semanal. Considerar reemplazo temprano.\n")
  }
  cat("\n")
}
```

```
## Cluster 1 :
##   • Componentes: 35
##   • Vida útil media: 12015 horas
##   • Características principales: Temperatura media = 50.2 °C, Vibración media = 2 mm/s
##   • Recomendación: Mantenimiento preventivo cada 8410 horas
##
## Cluster 2 :
##   • Componentes: 29
##   • Vida útil media: 5137 horas
##   • Características principales: Temperatura media = 80.4 °C, Vibración media = 5.9 mm/s
##   • Recomendación: Mantenimiento preventivo cada 2569 horas y monitoreo semanal. Considerar reemplaz
##
## Cluster 3 :
##   • Componentes: 36
##   • Vida útil media: 7978 horas
##   • Características principales: Temperatura media = 64.9 °C, Vibración media = 4 mm/s
##   • Recomendación: Mantenimiento preventivo cada 4787 horas y monitoreo mensual
```

Análisis e interpretación Este código implementa un flujo de trabajo completo para utilizar K-means en el contexto de mantenimiento industrial:

Creación de datos simulados: Se generan datos de 100 componentes con variables relevantes para el man-

tenimiento (temperatura, vibración, energía, carga, vida útil). Normalización: Se escalan las variables para que tengan la misma influencia en el algoritmo. Determinación del número óptimo de clusters: Utilizando el método del codo y el método de la silueta para identificar el número óptimo de clusters. Aplicación de K-means: Se aplica el algoritmo con 3 clusters (que es el que se definió en la simulación). Análisis de resultados: Se analizan las características de cada cluster, mostrando las estadísticas descriptivas y las distribuciones de las variables clave. Predicción para nuevos componentes: Se muestra cómo predecir a qué cluster pertenecería un nuevo componente y estimar su vida útil. Recomendaciones de mantenimiento: Basadas en las características de cada cluster, se generan recomendaciones específicas para la programación del mantenimiento preventivo.

Este enfoque permite a los ingenieros de mantenimiento:

Identificar patrones de comportamiento entre los componentes Desarrollar estrategias de mantenimiento personalizadas para cada grupo Predecir la vida útil esperada de nuevos componentes Optimizar los intervalos de mantenimiento preventivo Reducir el tiempo de inactividad no planificado

El análisis de clusters ayuda a pasar de un enfoque reactivo de mantenimiento a uno más proactivo y basado en datos, mejorando la eficiencia operativa y reduciendo los costos de mantenimiento.