Práctica 6 2024

Ignacio Fernández Contreras

ifcau3z@uma.es
Planificación de Proyectos y Análisis de Riesgos, E.T.S Informática.

1 Problema 1

Una de las medidas más potentes y sencillas de la confiabilidad del equipo es el tiempo medio entre fallas (MTBF), que permite al ingeniero de confiabilidad estimar aproximadamente las condiciones que rodean un sistema. Por ejemplo, para identificar un área de una planta para investigación, se destacarán los equipos o sistemas que tengan un MTBF deficiente, independientemente de la causa. Esto significa que las oportunidades se pueden identificar rápidamente sin necesidad de realizar una investigación profunda de cada sistema por adelantado. MTBF proporciona la base para otras funciones utilizadas para esbozar las condiciones de un sistema o instalación. Entre las que podemos destacar:

MTBF: Número de fallas/período de tiempo
MTTR: Tiempo de reparación/Número de fallas
MTTF: (Tiempo de detección – Tiempo de error)/Número de fallos
Tasa de fracaso: 1/MTBF
Función de confiabilidad: R=et donde t es el tiempo de interés
TTFE = 1-R en el momento de interés (probabilidad de que el equipo sobreviva hasta el momento de interés).
El TTFE de una falla se puede evaluar por componente, revisando el tiempo transcurrido desde la última falla de cada subsistema, o en todo el sistema suponiendo un inicio desde el tiempo "0".

La tasa de falla (FS) de la serie se realiza simplemente multiplicando cada R por cada componente del sistema, como FS = (R1)(R2)...(Rn). La tasa de falla en paralelo (FP) se calcula como FP = (R1 + R2) – (R1)(R2) para dos sistemas en paralelo, y $F = –(1–F)^n$ para tres o más sistemas idénticos en paralelo. Calcula R, FS y FP para el caso de un sistema de bomba paralela que se compone de una fuente de potencia entrante; un variador de frecuencia (VFD); dos motores eléctricos; dos acoplamientos; dos bombas; y dos válvulas. Después de observar el sistema por un periodo de 6000 horas al año (6000/año) el equipo proporciona la información siguiente:

Componente	MTBF(horas)	Horas desde el último fallo
Fuente de potencia	3000	1500
Variador de frecuencia (VFD)	10000	8000
Motor 1	12000	6800
Motor 2	4000	3800
Acoplamiento 1	3000	100
Acoplamiento 2	24000	3200
Válvula 1	N/A	60000
Válvulo 2	N/A	60000

(Las válvulas no se han reparado en 10 años)

```
# Datos del sistema
componentes <- data.frame(
  nombre = c("Fuente de potencia", "Variador de frecuencia (VFD)", "Motor 1", "Motor
      2",
             "Acoplamiento 1", "Acoplamiento 2", "V lvula 1", "V lvula 2"),
  MTBF = c(3000, 10000, 12000, 4000, 3000, 24000, NA, NA),
  horas_ultimo_fallo = c(1500, 8000, 6800, 3800, 100, 3200, 60000, 60000)
)

# Funci n para calcular la confiabilidad (R)
calcular_confiabilidad <- function(MTBF, tiempo) {
  if (!is.na(MTBF)) {
    lambda <- 1 / MTBF
    R <- exp(-lambda * tiempo)
  } else {
    R <- 1  # Si no se tiene el MTBF (por ejemplo, para las v lvulas), asumimos R=1
  }
  return(R)
}

# Aplicamos la funci n para calcular la confiabilidad de cada componente
componentes$confiabilidad <- mapply(calcular_confiabilidad, componentes$MTBF,
    componentes$horas_ultimo_fallo)

# Mostrar confiabilidad de cada componente
print(componentes)

# C lculo de la tasa de falla en serie (FS) como el producto de las confiabilidades
FS <- prod(componentes$confiabilidad, na.rm = TRUE)
cat("La tasa de falla en serie (FS) es:", FS, "\n")

# C lculo de la tasa de falla en paralelo para las v lvulas (en paralelo)
# Dado que hay dos v lvulas, calculamos la tasa de falla en paralelo para ellas
R_valvulas <- componentes$confiabilidad[7:8]
FP_valvulas <- sum(R_valvulas) - prod(R_valvulas)
cat("La tasa de falla en paralelo para las v lvulas (FP) es:", FP_valvulas, "\n")

# Ahora, considerando el sistema completo en paralelo, utilizamos la f rmula para 3
    componentes
# En este caso, si los componentes restantes est n en paralelo, calculamos la tasa
    de falla en paralelo para todo el sistema
componentes_restantes <- componentes$confiabilidad[1:6]  # Excluyendo las v lvulas

# Calculamos la tasa de falla en paralelo para los otros componentes
FP_restantes <- Reduce(function(x, y) x + y - x * y, componentes_restantes)

# La tasa total de fallo (FP) para el sistema completo es la combinaci n de las
    tasas de los componentes en serie y en paralelo
FP_total <- FP_restantes * FP_valvulas
cat("La tasa de falla total en paralelo (FP) para el sistema es:", FP_total, "\n")
```

```
> print(componentes)
                        nombre  MTBF horas_ultimo_fallo confiabilidad
1           Fuente de potencia  3000              1500     0.6065307
2 Variador de frecuencia (VFD) 10000              8000     0.4493290
3                      Motor 1 12000              6800     0.5674137
4                      Motor 2  4000              3800     0.3867410
5               Acoplamiento 1  3000               100     0.9672161
6               Acoplamiento 2 24000              3200     0.8751733
7                    Válvula 1    NA             60000     1.0000000
8                    Válvula 2    NA             60000     1.0000000
> cat("La tasa de falla en serie (FS) es:", FS, "\n")
La tasa de falla en serie (FS) es: 0.05062381
> cat("La tasa de falla en paralelo para las válvulas (FP) es:", FP_valvulas, "\n")
La tasa de falla en paralelo para las válvulas (FP) es: 1
> cat("La tasa de falla total en paralelo (FP) para el sistema es:", FP_total, "\n")
La tasa de falla total en paralelo (FP) para el sistema es: 0.9997648
>
```

2 Ejercicio 2

Repite el ejercicio anterior aumentando el MTBF entre un 10% y un 20%

```
> print(componentes)
                        nombre      MTBF horas_ultimo_fallo confiabilidad
1           Fuente de potencia  3430.342              1500     0.6457943
2 Variador de frecuencia (VFD) 11205.055              8000     0.4896994
3                      Motor 1 13429.293              6800     0.6026879
4                      Motor 2  4443.252              3800     0.4251856
5               Acoplamiento 1  3339.546               100     0.9704997
6               Acoplamiento 2 26951.256              3200     0.8880450
7                    Válvula 1        NA             60000     1.0000000
8                    Válvula 2        NA             60000     1.0000000
> cat("La tasa de falla en serie (FS) es:", FS, "\n")
La tasa de falla en serie (FS) es: 0.06984338
> cat("La tasa de falla en paralelo para las válvulas (FP) es:", FP_valvulas, "\n")
La tasa de falla en paralelo para las válvulas (FP) es: 1
> cat("La tasa de falla total en paralelo (FP) para el sistema es:", FP_total, "\n")
La tasa de falla total en paralelo (FP) para el sistema es: 0.9998637
```

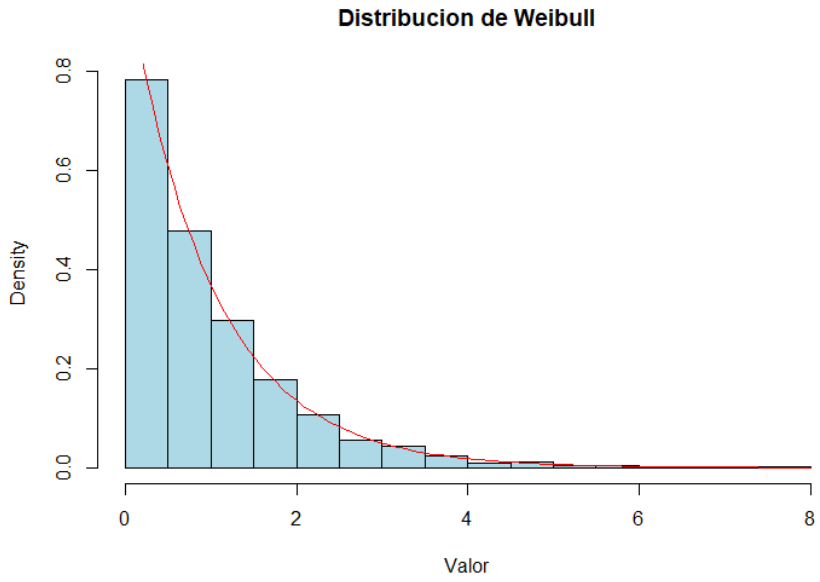
3 Ejercicio 3

En este ejercicio vamos a estudiar la Función de Densidad de Weibull que modela la distribución de fallos (en sistemas) cuando la tasa de fallos es proporcional a una potencia del tiempo

3.1 Apartado A: Usa la función weibull y dweibull (función de distribución) del paquete VaRES con una distribución continua de 1000 números en el intervalo [0.1]

```
# Generar 1000 n meros con distribuci n Weibull
set.seed(123)
n <- 1000
shape <- 1
scale <- 1
data_weibull <- rweibull(n, shape, scale)
```

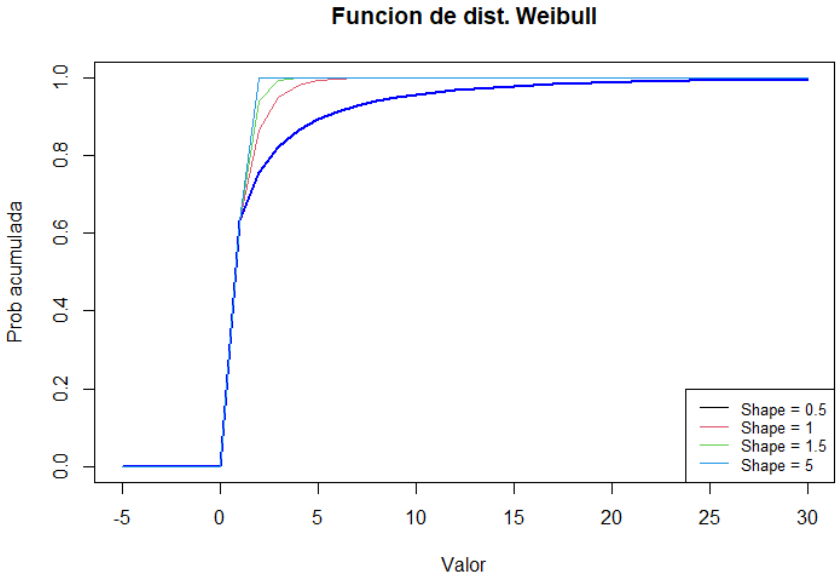
```
hist(data_weibull, probability = TRUE, main = "Distribuci n de Weibull", xlab = "
    Valor", col = "lightblue", border = "black")
curve(dweibull(x, shape, scale), col = "red", add = TRUE)
```



3.2 Apartado B: Dibuja la función de distribución de weibull en el intervalo [-5,30] con un incremento de 1 (factor de escala 1 y factor de forma 0.5, 1, 1.5 y 5).

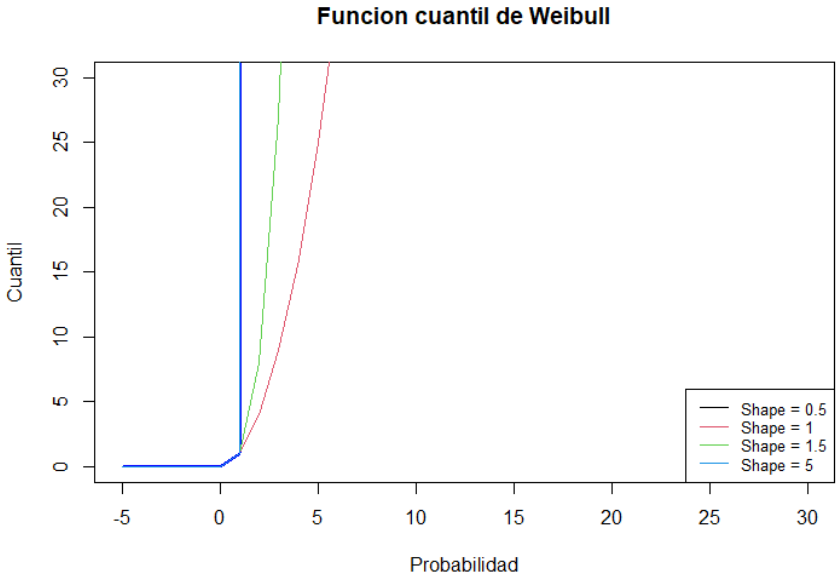
```
# Dibujar la funci n de distribuci n de Weibull con diferentes valores de forma
x <- seq(-5, 30, by = 1)  # Rango de -5 a 30
shape_values <- c(0.5, 1, 1.5, 5)

plot(x, pweibull(x, shape_values[1], scale), type = "l", col = "blue", lwd = 2,
     ylim = c(0, 1), ylab = "Probabilidad acumulada", xlab = "Valor", main = "
         Funci n de distribuci n Weibull")
for (i in 2:length(shape_values)) {
  lines(x, pweibull(x, shape_values[i], scale), col = i)
}
legend("bottomright", legend = paste("Shape =", shape_values), col = 1:length(
    shape_values), lty = 1, cex = 0.8)
```



3.3 Apartado C: Ídem con la función cuantil de weibull.

```
# Dibujar la funci n cuantil de Weibull para diferentes valores de shape
plot(x, qweibull(pweibull(x, shape_values[1], scale), shape_values[1], scale), type
    = "l", col = "blue", lwd = 2,
      ylim = c(0, 30), ylab = "Cuantil", xlab = "Probabilidad", main = "Funci n
          cuantil de Weibull")
for (i in 2:length(shape_values)) {
  lines(x, qweibull(pweibull(x, shape_values[i], scale), shape_values[i], scale),
       col = i)
}
legend("bottomright", legend = paste("Shape =", shape_values), col = 1:length(
    shape_values), lty = 1, cex = 0.8)
```

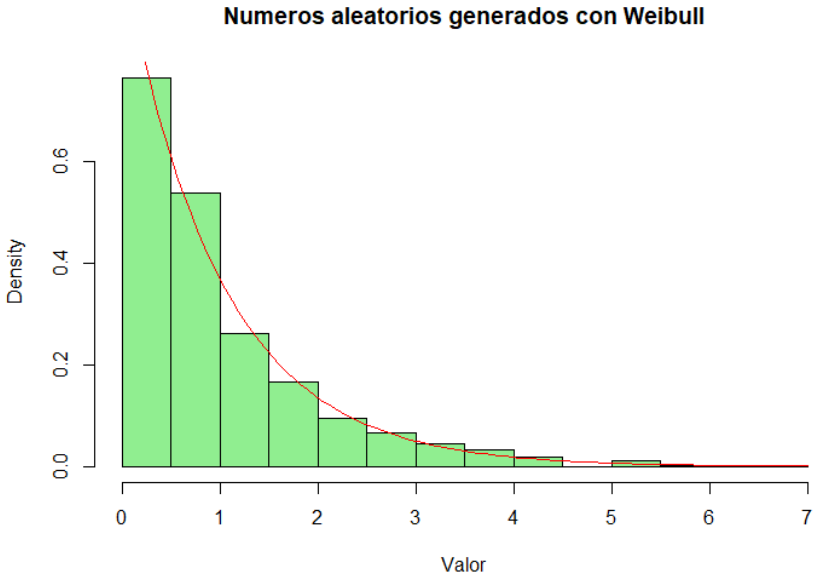


3.4 Apartado D: Ídem con la función de generación de números aleatorios de weibull.

```
# Generar 1000 n meros aleatorios de Weibull
data_random_weibull <- rweibull(n, shape = 1, scale = 1)

hist(data_random_weibull, probability = TRUE, main = "N meros Aleatorios Generados
    con Weibull",
     xlab = "Valor", col = "lightgreen", border = "black")

curve(dweibull(x, shape = 1, scale = 1), col = "red", add = TRUE)
```



4 Ejercicio 4

El análisis de fallas de máquinas corresponde a un conjunto de enfoques estadísticos utilizados para investigar el tiempo que tarda en ocurrir un evento de interés.
El tiempo hasta el evento, el análisis de fallas, el análisis de supervivencia, el análisis del historial de eventos, el análisis de confiabilidad, todos abordan el tiempo hasta que ocurre un evento de interés. Sin embargo, es posible que el tiempo de falla no se observe dentro del período de tiempo relevante, lo que produce las llamadas observaciones censuradas.
El paquete de supervivencia es la piedra angular de todo el ecosistema de análisis de supervivencia de R. El paquete en sí no solo es rico en características, sino que el objeto creado por la función Surv(), que contiene tiempo de falla e información de censura, es la estructura básica de datos de análisis de supervivencia en R.
Las variables en este conjunto de datos (un subconjunto del mismo lo podemos descargar del campus virtual) son:
lifetime: número de semanas que la máquina ha estado activa
broken: especifica si la máquina se rompió o no se ha roto aún durante las semanas correspondientes en actividad
PressureInd: el índice de presión se utiliza para cuantificar el flujo de líquido a través de las tuberías, ya que una caída repentina de presión puede indicar una fuga.
moistureInd : el índice de humedad es una medida de la humedad relativa del aire. Es importante realizar un seguimiento ya que la humedad excesiva puede crear moho y dañar el equipo.
temperature : el índice de temperatura de la máquina se calcula utilizando dispositivos de voltaje llamados termopares que traducen un cambio de voltaje en una medida de temperatura. Está grabado para evitar daños a los circuitos eléctricos, incendio o incluso explosión.
team: este indicador especifica qué equipo está utilizando la máquina. provider: este indicador especifica el nombre del fabricante de la máquina.
Calcula la predicción de fallo del sistema usando:

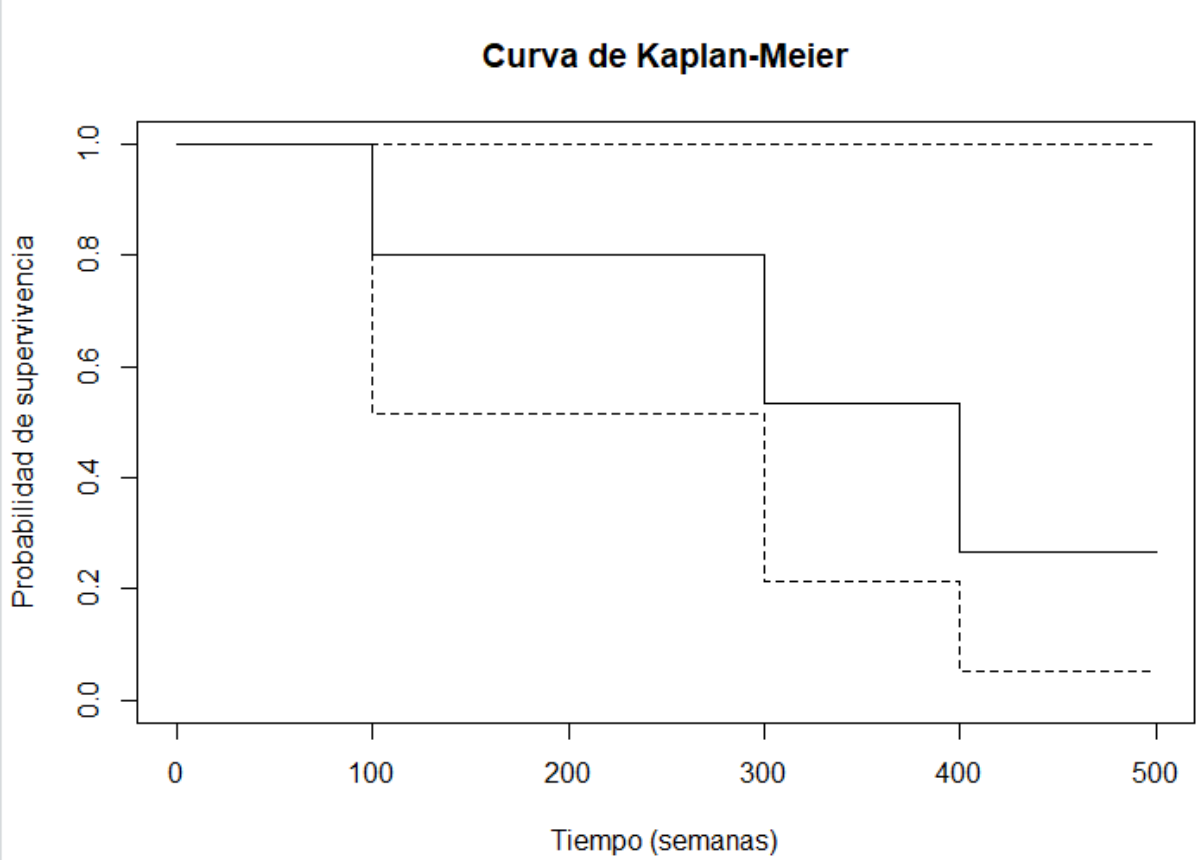
4.1 Apartado A: el análisis de Kaplan Meier

```
data <- data.frame(
  lifetime = c(100, 200, 300, 400, 500),
  broken = c(1, 0, 1, 1, 0),  # 1 indica que se rompi , 0 que no
  PressureInd = c(30, 40, 35, 45, 50),
  moistureInd = c(60, 65, 55, 70, 75),
  temperature = c(70, 80, 75, 85, 90),
  team = factor(c("A", "B", "C", "A", "B")),
  provider = factor(c("Provider1", "Provider2", "Provider1", "Provider2", "Provider1
      "))
)

# Crear objeto Surv para an lisis de supervivencia
surv_obj <- Surv(data$lifetime, data$broken)

# Aplicar el an lisis de Kaplan-Meier
fit_km <- survfit(surv_obj ~ 1)

# Graficar la curva de Kaplan-Meier
plot(fit_km, main="Curva de Kaplan-Meier", xlab="Tiempo (semanas)", ylab="
    Probabilidad de supervivencia")
```



4.2 Apartado B: el Modelo de riesgos proporcionales de Cox (Cox Proportional Hazards Model)

```
# Crear modelo de Cox Proportional Hazards
cox_model <- coxph(surv_obj ~ PressureInd + moistureInd + temperature + team +
    provider, data=data)

summary(cox_model)
plot(cox_model)
```

```
Call:
coxph(formula = surv_obj ~ PressureInd + moistureInd + temperature +
    team + provider, data = data)

  n= 5, number of events= 3

                       coef  exp(coef)   se(coef)      z Pr(>|z|)
PressureInd       -2.216e+00  1.090e-01  2.232e+03 -0.001    0.999
moistureInd       -1.299e+00  2.729e-01  1.621e+03 -0.001    0.999
temperature        0.000e+00  1.000e+00  2.232e+03  0.000    1.000
teamB              1.030e+00  2.802e+00  2.301e+04  0.000    1.000
teamC             -1.773e+01  1.990e-08  2.454e+04 -0.001    0.999
providerProvider2  3.562e+00  3.525e+01  1.728e+04  0.000    1.000

                  exp(coef) exp(-coef) lower .95 upper .95
PressureInd       1.090e-01  9.173e+00         0       Inf
moistureInd       2.729e-01  3.664e+00         0       Inf
temperature       1.000e+00  1.000e+00         0       Inf
teamB             2.802e+00  3.568e-01         0       Inf
teamC             1.990e-08  5.024e+07         0       Inf
providerProvider2 3.525e+01  2.837e-02         0       Inf

Concordance= 1  (se = 0 )
Likelihood ratio test= 6.8  on 6 df,   p=0.3
Wald test            = 0  on 6 df,   p=1
Score (logrank) test = 5.73  on 6 df,   p=0.5
```

4.3 Apartado C: el modelo de bosques aleatorios (Random Forests Model). Calcula la importancia de las variables

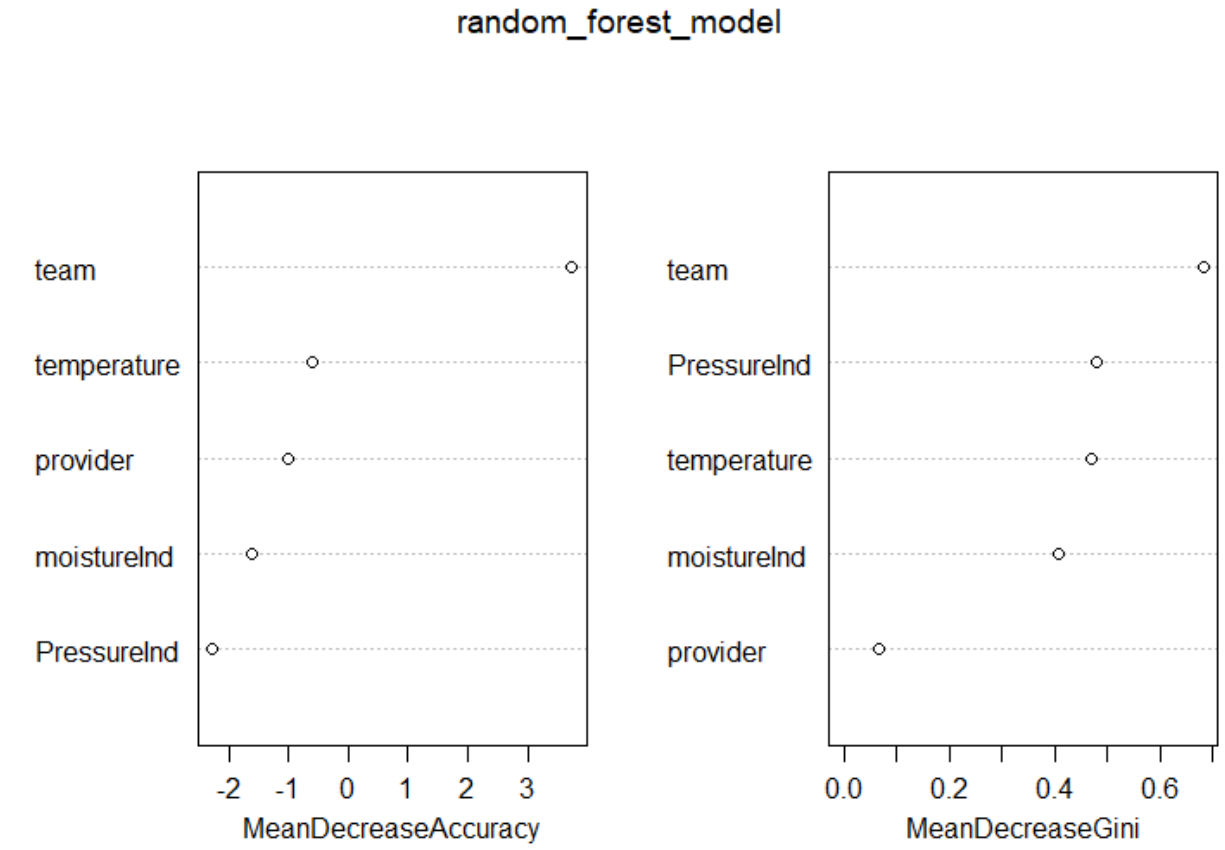
```
# Usar Random Forest para el an lisis
random_forest_model <- randomForest(as.factor(broken) ~ PressureInd + moistureInd +
    temperature + team + provider, data = data, importance = TRUE)
print(random_forest_model)
importance(random_forest_model)
varImpPlot(random_forest_model)
```

```
> print(random_forest_model)

Call:
 randomForest(formula = as.factor(broken) ~ PressureInd + moistureInd +      temperature + team + provider, data = data, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 40%
Confusion matrix:
  0 1 class.error
0 1 1   0.5000000
1 1 2   0.3333333
```

```
> importance(random_forest_model)
                     0          1 MeanDecreaseAccuracy MeanDecreaseGini
PressureInd -3.194383 -0.7306865            -2.275656       0.47926667
moistureInd -2.120044 -0.9052750            -1.612809       0.40680000
temperature -1.266940  0.4083163            -0.611188       0.46800000
team         3.795167  3.6178730             3.746989       0.68200000
provider    -1.001002 -1.0010015            -1.001002       0.06633333
```



4.4 Apartado D: el modelo de maquina de soporte vectorial SVM

```
svm_model <- svm(as.factor(broken) ~ PressureInd + moistureInd + temperature + team
    + provider, data = data)
summary(svm_model)
svm_pred <- predict(svm_model, data)
```

```
# Evaluar el rendimiento del modelo
table(predicted = svm_pred, actual = data$broken)
```

```
> summary(svm_model)

Call:
svm(formula = as.factor(broken) ~ PressureInd + moistureInd + temperature + team + provider, data = data)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  5

 ( 3 2 )


Number of Classes:  2

Levels:
 0 1
```

```
> svm_pred <- predict(svm_model, data)
> table(predicted=svm_pred, actual=data$broken)
         actual
predicted 0 1
        0 1 0
        1 1 3
```

5 Ejercicio 6

Calcula la Función de distribución de Weibull usando predicción mediante técnicas de Aprendizaje automático (usa Perceptron y SVM). El conjunto de datos se muestra a continuación:

Time (Hours)	R(t) Lower 95.0 % Confidence Limit	R(t) Point Estimate	R(t) Upper 95.0% Confidence Limit
71.2	0.8563	0.9632	0.9970
95.3	0.7681	0.9057	0.9772
110.9	0.6925	0.8482	0.9468
123.4	0.6232	0.7907	0.9104
134.1	0.5581	0.7331	0.8698
143.8	0.4964	0.6756	0.8259
152.9	0.4375	0.6181	0.7793
162.4	0.3755	0.5548	0.7250
171.8	0.3164	0.4915	0.6680
186.9	0.2277	0.3903	0.5709
208.9	0.1218	0.2553	0.4291

Valores obtenidos a partir de los siguientes parámetros:

Parameter	Lower 95.0% Confidence Limit	Point Estimate	Upper 95.0% Confidence Limit
Shape Parameter(B)	1.85	2.97	4.77
Characteristic life	166.28	203.29	248.56

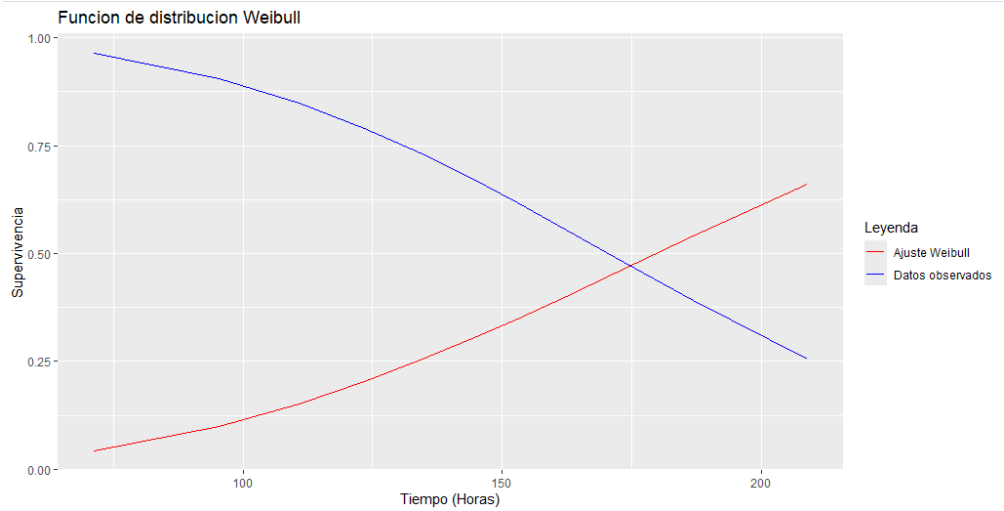
```
# Datos de supervivencia
data <- data.frame(
  Time = c(71.2, 95.3, 110.9, 123.4, 134.1, 143.8, 152.9, 162.4, 171.8, 186.9,
      208.9),
  R_Lower = c(0.8563, 0.7681, 0.6925, 0.6232, 0.5581, 0.4964, 0.4375, 0.3755,
      0.3164, 0.2277, 0.1218),
  R_Point = c(0.9632, 0.9057, 0.8482, 0.7907, 0.7331, 0.6756, 0.6181, 0.5548,
      0.4915, 0.3903, 0.2553),
  R_Upper = c(0.9970, 0.9772, 0.9468, 0.9104, 0.8698, 0.8259, 0.7793, 0.7250,
      0.6680, 0.5709, 0.4291)
)
```

5.1 Calcular la función de distribución Weibull

```
# Par metros de Weibull (Punto Estimado)
shape_param <- 2.97
scale_param <- 203.29

# Calcular la funci n de distribuci n Weibull
data$Weibull_Fit <- pweibull(data$Time, shape = shape_param, scale = scale_param)

# Visualizar los datos ajustados
ggplot(data, aes(x = Time)) +
  geom_line(aes(y = R_Point, color = "Datos Observados")) +
  geom_line(aes(y = Weibull_Fit, color = "Ajuste Weibull")) +
  labs(title = "Funci n de Distribuci n Weibull", x = "Tiempo (Horas)", y = "
      Supervivencia") +
  scale_color_manual(name = "Leyenda", values = c("Datos Observados" = "blue", "
      Ajuste Weibull" = "red"))
```

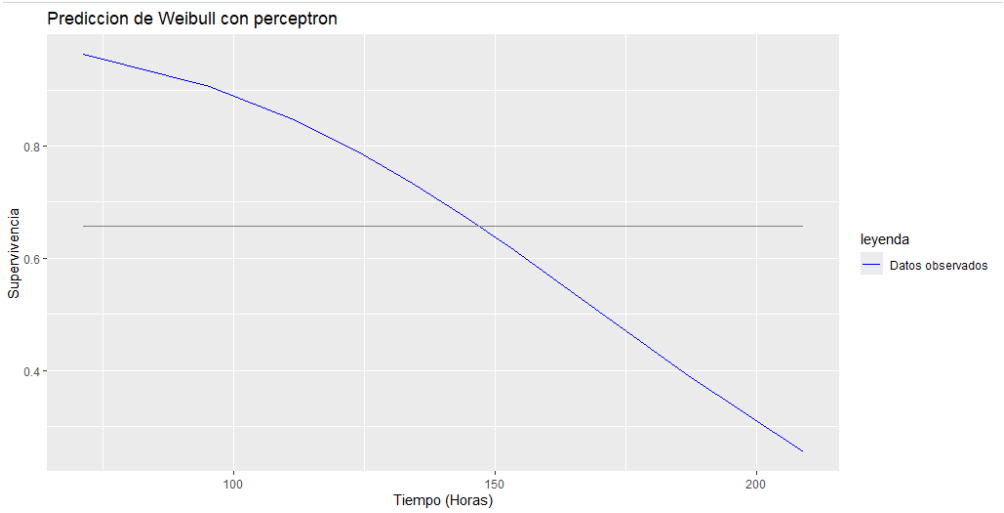


5.2 Predicción utilizando Perceptrón

```
# Crear el modelo de Perceptr n
perceptron_model <- nnet(
  R_Point ~ Time,
  data = data,
  size = 1,   # N mero de neuronas en la capa oculta
  linout = TRUE
)

# Predicci n usando el modelo
data$Perceptron_Pred <- predict(perceptron_model, newdata = data)

# Graficar resultados
ggplot(data, aes(x = Time)) +
  geom_line(aes(y = R_Point, color = "Datos Observados")) +
  geom_line(aes(y = Perceptron_Pred, color = "Predicci n Perceptr n")) +
  labs(title = "Predicci n de Weibull con Perceptr n", x = "Tiempo (Horas)", y = "
      Supervivencia") +
  scale_color_manual(name = "Leyenda", values = c("Datos Observados" = "blue", "
      Predicci n Perceptr n" = "green"))
```



5.3 Predicción utilizando SVM

```
# Crear el modelo SVM
svm_model <- svm(R_Point ~ Time, data = data, kernel = "radial")

# Predicci n usando el modelo SVM
data$SVM_Pred <- predict(svm_model, newdata = data)

# Graficar resultados
ggplot(data, aes(x = Time)) +
  geom_line(aes(y = R_Point, color = "Datos Observados")) +
  geom_line(aes(y = SVM_Pred, color = "Predicci n SVM")) +
  labs(title = "Predicci n de Weibull con SVM", x = "Tiempo (Horas)", y = "
      Supervivencia") +
  scale_color_manual(name = "Leyenda", values = c("Datos Observados" = "blue", "
      Predicci n SVM" = "purple"))
```