A continuación, se presentan ejemplos de código en R para realizar las pruebas mencionadas, junto con las condiciones que deben verificarse antes de aplicar cada prueba.

1. Prueba t de Student

Código en R:

```
# Supongamos que tenemos dos grupos de datos: grupo1 y grupo2
grupo1 <- c(5, 6, 7, 8, 9)
grupo2 <- c(6, 7, 8, 9, 10)
# Prueba t de Student
t.test(grupo1, grupo2)
```

Condiciones a probar:

- Normalidad:

```
shapiro.test(grupo1)  # Para grupo1
shapiro.test(grupo2)  # Para grupo2
```

- Homogeneidad de varianzas:

```
var.test(grupo1, grupo2)
```

2. ANOVA (Análisis de Varianza)

Código en R:

```
# Supongamos que tenemos un data frame con tres grupos
data <- data.frame(
  grupo = rep(c("A", "B", "C"), each = 10),
  valores = c(rnorm(10, mean = 5), rnorm(10, mean = 6), rnorm(10, mean = 7))
)
# ANOVA
anova_result <- aov(valores ~ grupo, data = data)
summary(anova_result)
```

Condiciones a probar:

- Normalidad:

```
shapiro.test(residuals(anova_result))
```

- Homogeneidad de varianzas:

```
library(car)

leveneTest(valores ~ grupo, data = data)
```

3. Prueba de Chi-cuadrado

Código en R:

```
# Supongamos que tenemos una tabla de contingencia

tabla <- matrix(c(10, 20, 30, 20, 15, 25), nrow = 2)

chisq.test(tabla)
```

Condiciones a probar:

- Tamaño de muestra adecuado: cada celda de la tabla debe tener al menos 5 observaciones esperadas.
- Independencia de las observaciones: asegurarse de que los datos no estén relacionados.

4. Regresión Lineal

Código en R:

```
# Supongamos que tenemos un data frame con variables independientes y dependientes

data <- data.frame(

  x = c(1, 2, 3, 4, 5),

  y = c(2, 3, 5, 7, 11)

)

# Regresión lineal

modelo <- lm(y ~ x, data = data)

summary(modelo)
```

Condiciones a probar:

- Linealidad: graficar los residuos.

```
plot(modelo)
```

- Normalidad de los errores:

```
shapiro.test(residuals(modelo))
```

- Homocedasticidad:

```
library(car)
ncvTest(modelo)
```

- Independencia de los errores: verificar mediante el gráfico de residuos.

5. Pruebas No Paramétricas (ejemplo: Mann-Whitney)

Código en R:

```
# Supongamos que tenemos dos grupos de datos
grupo1 <- c(5, 6, 7, 8, 9)
grupo2 <- c(6, 7, 8, 9, 10)

# Prueba de Mann-Whitney
wilcox.test(grupo1, grupo2)
```

Condiciones a probar:

- Los datos deben ser independientes.
- Los datos deben ser ordinales o continuos.

1. Multicolinealidad

Para evaluar la multicolinealidad, puedes usar el paquete car para calcular el VIF (Factor de Inflación de la Varianza).

```
# Instalar y cargar el paquete car

install.packages("car")

library(car)

# Ajustar el modelo de regresión logística

modelo <- glm(am ~ wt + hp + drat, data = mtcars, family = binomial)

# Calcular el VIF

vif(modelo)
```

Condición a probar: Un VIF mayor a 5 o 10 indica problemas de multicolinealidad.

2. Información incompleta

Para verificar la información incompleta, puedes usar la función summary() para observar el número de observaciones y la estructura de los datos.

```
# Resumen del modelo

summary(modelo)

# Verificar datos faltantes

sum(is.na(mtcars))
```

Condición a probar: Asegúrate de que no haya valores NA en los predictores utilizados en el modelo.

3. Criterios de evaluación de modelos (AIC y BIC)

Después de ajustar el modelo, puedes obtener el AIC y BIC directamente desde el objeto del modelo.

```
# AIC y BIC

aic_value <- AIC(modelo)

bic_value <- BIC(modelo)
```

```
print(paste("AIC:", aic_value))

print(paste("BIC:", bic_value))
```

Condición a probar: Comparar AIC y BIC entre diferentes modelos para seleccionar el más parsimonioso.

4. Evaluación de residuos

Para evaluar los residuos, puedes usar la función residuals() y calcular la distancia de Cook.

```
# Residuos

residuos <- residuals(modelo, type = "deviance")

# Distancia de Cook

cooks_distance <- cooks.distance(modelo)

# Graficar residuos y distancia de Cook

par(mfrow = c(1, 2))

plot(residuos, main = "Residuos")

plot(cooks_distance, main = "Distancia de Cook", ylab = "Distancia de Cook")

abline(h = 4/(nrow(mtcars)-length(coef(modelo))), col = "red") # Línea de corte
```

Condición a probar: Identificar valores atípicos y casos influyentes. Valores de distancia de Cook mayores a 1 pueden ser problemáticos.

5. Condiciones para usar regresión logística

Para verificar la relación lineal entre los predictores y la respuesta transformada, puedes usar gráficos de probabilidad.

```
# Graficar la relación entre los predictores y la probabilidad de éxito

library(ggplot2)

# Graficar la probabilidad de éxito

ggplot(mtcars, aes(x = wt, y = am)) + geom_point() + geom_smooth(method = "glm", method.args = list(family = "binomial"), se = FALSE)
```

Condición a probar: Asegúrate de que la relación entre los predictores y la variable de respuesta sea lineal en la escala logit.

1. Pruebas de Hipótesis

a. Prueba t de Student

```
# Supongamos que tenemos dos grupos de datos
grupo1 <- c(5, 6, 7, 8, 9)
grupo2 <- c(6, 7, 8, 9, 10)
# Prueba t de Student
t.test(grupo1, grupo2)

# Condiciones a verificar:

# 1. Normalidad: Se puede usar la prueba de Shapiro-Wilk
shapiro.test(grupo1)
shapiro.test(grupo2)

# 2. Homogeneidad de varianzas: Se puede usar la prueba de Levene
library(car)
leveneTest(c(grupo1, grupo2) ~ factor(c(rep(1, length(grupo1)), rep(2, length(grupo2)))))
```

b. Prueba de Chi-Cuadrado

```
# Supongamos que tenemos una tabla de contingencia
tabla <- matrix(c(10, 20, 20, 30), nrow = 2)
# Prueba de chi-cuadrado
chisq.test(tabla)

# Condiciones a verificar:
# 1. Tamaño de muestra: Esperar al menos 5 observaciones en cada celda.
```

c. ANOVA

```
# Supongamos que tenemos tres grupos de datos
grupoA <- c(5, 6, 7)
grupoB <- c(6, 7, 8)
grupoC <- c(7, 8, 9)
# ANOVA
anova_result <- aov(c(grupoA, grupoB, grupoC) ~ factor(rep(1:3, each = 3)))
summary(anova_result)

# Condiciones a verificar:
# 1. Normalidad: Prueba de Shapiro-Wilk para cada grupo.
# 2. Homogeneidad de varianzas: Prueba de Levene.
```

2. Intervalos de Confianza

```
# Supongamos que tenemos un vector de datos
datos <- c(5, 6, 7, 8, 9)
# Calcular el intervalo de confianza para la media
mean_datos <- mean(datos)
sd_datos <- sd(datos)
n <- length(datos)
error_estandar <- sd_datos / sqrt(n)
nivel_confianza <- 0.95
z <- qnorm(1 - (1 - nivel_confianza) / 2)
limite_inferior <- mean_datos - z * error_estandar
limite_superior <- mean_datos + z * error_estandar
c(limite_inferior, limite_superior)
```

3. Regresión Lineal

```
# Supongamos que tenemos dos variables
x <- c(1, 2, 3, 4, 5)
y <- c(2, 3, 5, 7, 11)

# Modelo de regresión lineal
modelo <- lm(y ~ x)
summary(modelo)

# Condiciones a verificar:
# 1. Linealidad: Graficar los residuos.
# 2. Homocedasticidad: Graficar los residuos vs. valores ajustados.
# 3. Normalidad de los errores: Prueba de Shapiro-Wilk sobre los residuos.
```

4. Análisis de Varianza (ANOVA)

```
# Ya se mostró en el ejemplo de ANOVA anterior.
```

5. Pruebas No Paramétricas

a. Prueba de Mann-Whitney

```
# Supongamos que tenemos dos grupos de datos
grupo1 <- c(5, 6, 7, 8, 9)
grupo2 <- c(6, 7, 8, 9, 10)

# Prueba de Mann-Whitney
wilcox.test(grupo1, grupo2)

# Condiciones a verificar:
# 1. Independencia de las observaciones.
```

b. Prueba de Kruskal-Wallis

```
# Supongamos que tenemos tres grupos de datos
grupoA <- c(5, 6, 7)
grupoB <- c(6, 7, 8)
grupoC <- c(7, 8, 9)

# Prueba de Kruskal-Wallis
kruskal.test(c(grupoA, grupoB, grupoC) ~ factor(rep(1:3, each = 3)))

# Condiciones a verificar:
# 1. Independencia de las observaciones.
```

Resumen de Condiciones

- **Normalidad**: Verificar mediante la prueba de Shapiro-Wilk.
- **Homogeneidad de varianzas**: Verificar mediante la prueba de Levene.
- **Independencia**: Asegurarse de que las observaciones no estén correlacionadas.

Estas pruebas y sus condiciones son esenciales para garantizar la validez de los resultados en análisis estadísticos.

Prueba de Normalidad (Shapiro-Wilk):

```
# Ajustar el modelo de regresión
modelo <- lm(requisitos ~ stakeholders, data = datos)

# Obtener los residuos
residuos <- modelo$residuals

# Prueba de normalidad
shapiro.test(residuos)
```

Condiciones a probar: Los residuos deben ser independientes y la muestra debe ser representativa.

Homocedasticidad:

Graficar residuos vs. valores ajustados

plot(modelo$fitted.values, residuos)

abline(h = 0, col = "red") # Línea horizontal en 0

Condiciones a probar: No debe haber patrones en el gráfico; la variabilidad de los residuos debe ser constante.

Independencia de Observaciones:

Condiciones a probar: Asegurarse de que las observaciones se seleccionaron aleatoriamente y no son parte de una serie temporal. Esto no se puede verificar directamente con un código, pero se debe tener en cuenta al recolectar los datos.

Detección de Valores Atípicos:

Graficar residuos para detectar valores atípicos

boxplot(residuos)

Condiciones a probar: Evaluar el contexto de los datos; los valores atípicos deben ser analizados para determinar su impacto.

Validación Cruzada:

library(caret) # Cargar la librería caret para validación cruzada

Dividir los datos en conjunto de entrenamiento y prueba

set.seed(123) # Para reproducibilidad

trainIndex <- createDataPartition(datos$requisitos, p = .8, list = FALSE, times = 1)

```
datosTrain <- datos[trainIndex, ]
datosTest <- datos[-trainIndex, ]

# Ajustar el modelo en el conjunto de entrenamiento
modelo_train <- lm(requisitos ~ stakeholders, data = datosTrain)

# Predecir en el conjunto de prueba
predicciones <- predict(modelo_train, newdata = datosTest)

# Calcular el error cuadrático medio (MSE)
mse <- mean((predicciones - datosTest$requisitos)^2)
print(mse)
```

Condiciones a probar: Asegurarse de que ambos conjuntos (entrenamiento y prueba) sean representativos.

Errores Cuadráticos Medios (MSE):

```
# Calcular MSE para el conjunto de entrenamiento
mse_train <- mean(modelo$residuals^2)
print(mse_train)
```

Condiciones a probar: Se debe tener un conjunto de datos suficientemente grande para obtener estimaciones confiables.

1. Bootstrapping

Código en R:

```
# Supongamos que tenemos un vector de datos llamado 'datos'
set.seed(123)  # Para reproducibilidad
n <- length(datos)
B <- 1000  # Número de remuestras
```

```
bootstrap_samples <- replicate(B, sample(datos, n, replace = TRUE))
bootstrap_means <- colMeans(bootstrap_samples)

# Calcular el intervalo de confianza
ci <- quantile(bootstrap_means, c(0.025, 0.975))
print(ci)
```

Condiciones a probar:

- No se requiere que los datos sigan una distribución normal.
- Se recomienda que la muestra sea representativa de la población.

2. Pruebas de Permutaciones

Código en R:

```
# Supongamos que tenemos dos grupos de datos: grupoA y grupoB
set.seed(123)
nA <- length(grupoA)
nB <- length(grupoB)
observed_diff <- mean(grupoA) - mean(grupoB)

# Función para calcular la diferencia de medias
perm_test <- function(x, y) {
 combined <- c(x, y)
 permuted <- sample(combined)
 new_x <- permuted[1:nA]
 new_y <- permuted[(nA + 1):(nA + nB)]
 return(mean(new_x) - mean(new_y))}

# Generar permutaciones
B <- 1000
```

```
perm_diffs <- replicate(B, perm_test(grupoA, grupoB))
p_value <- mean(abs(perm_diffs) >= abs(observed_diff))
print(p_value)
```

Condiciones a probar:

- Los datos deben ser intercambiables bajo la hipótesis nula.
- Se recomienda que las muestras sean pequeñas debido a la carga computacional.

3. Prueba Exacta de Fisher

Código en R:

```
# Supongamos que tenemos una tabla de contingencia llamada 'tabla'
tabla <- matrix(c(10, 20, 20, 30), nrow = 2)
fisher_test_result <- fisher.test(tabla)
print(fisher_test_result)
```

Condiciones a probar:

- Los datos deben ser categóricos.
- Se debe tener en cuenta que la prueba es más adecuada para muestras pequeñas.

4. Simulación de Monte Carlo

Código en R:

```
# Supongamos que tenemos dos grupos de datos: grupoA y grupoB
set.seed(123)
nA <- length(grupoA)
nB <- length(grupoB)
observed_diff <- mean(grupoA) - mean(grupoB)

# Función para calcular la diferencia de medias
monte_carlo_test <- function(x, y, R) {
```

```
  combined <- c(x, y)
  perm_diffs <- numeric(R)
  for (i in 1:R) {
    permuted <- sample(combined)
    new_x <- permuted[1:nA]
    new_y <- permuted[(nA + 1):(nA + nB)]
    perm_diffs[i] <- mean(new_x) - mean(new_y)
  }
  return(perm_diffs)
}

# Generar simulaciones
R <- 1000
perm_diffs <- monte_carlo_test(grupoA, grupoB, R)
p_value <- mean(abs(perm_diffs) >= abs(observed_diff))
print(p_value)
```

Condiciones a probar:

- Los datos deben ser intercambiables bajo la hipótesis nula.
- Se recomienda que las muestras sean grandes para obtener una buena aproximación.

Resumen de Condiciones

- **Bootstrapping**: Muestra representativa, no requiere normalidad.
- **Pruebas de Permutaciones**: Intercambiabilidad de datos, muestras pequeñas.
- **Prueba Exacta de Fisher**: Datos categóricos, adecuada para muestras pequeñas.
- **Simulación de Monte Carlo**: Intercambiabilidad de datos, adecuada para muestras grandes.