**Para saber mais: explorando as funções matemáticas**

O R oferece uma variedade de funções matemáticas que permitem realizar operações fundamentais com facilidade. Anteriormente, conhecemos as funções sum() e round(). Agora, vamos explorar outras quatro funções: abs(), sqrt(), exp(), e log().

**abs() (Valor Absoluto):**

A função abs() retorna o valor absoluto de um número, ou seja, a distância desse número até zero, sem considerar a direção. Ela converte números negativos em positivos e deixa números positivos inalterados.

**Exemplo abs():**

numero <- -10

resultado\_abs <- abs(numero)

print(resultado\_abs)

# Saída: 10

Copiar código

Aqui, abs() converte o número negativo -10 em seu equivalente positivo.

**sqrt() (Raiz Quadrada):**

A função sqrt() calcula a raiz quadrada de um número. É uma operação essencial em diversas áreas, incluindo estatística e cálculos de distância euclidiana.

Exemplo sqrt():

numero <- 25

resultado\_sqrt <- sqrt(numero)

print(resultado\_sqrt)

# Saída: 5

Copiar código

A função sqrt() calcula a raiz quadrada de 25, resultando em 5.

**exp() (Exponencial):**

A função exp() calcula o valor da função exponencial, onde o argumento representa o expoente. É útil em situações onde ocorre crescimento ou decaimento exponencial.

Exemplo exp():

expoente <- 2

resultado\_exp <- exp(expoente)

print(resultado\_exp)

# Saída: 7.389056

Copiar código

No exemplo, a função exp() eleva o número de Euler (aproximadamente 2.718) à potência de 2.

**log() (Logaritmo Natural):**

A função log() retorna o logaritmo natural de um número. Essa função é valiosa para analisar taxas de crescimento ou para reverter operações exponenciais.

**Exemplo log():**

numero <- 100

resultado\_log <- log(numero)

print(resultado\_log)

# Saída: 4.60517

Copiar código

No exemplo, a função log() calcula o logaritmo natural de 100, resultando em aproximadamente 4.60517.

Essas funções matemáticas são ferramentas essenciais para manipulação de dados e análise numérica no R, proporcionando uma base sólida para uma variedade de aplicações em ciência de dados e estatística.

**Para saber mais: explorando as funções estatísticas**

Anteriormente, explorarmos as funções mean(), median(), which.max(), e which.min(), onde aprimoramos nossa compreensão sobre medidas de tendência central e extremos, agora vamos explorar funções que podem nos ajudar a verificar a distribuição e variabilidade dos dados.

A seguir, vamos conhecer as funções mode(), sd(), e var().

**mode() (Moda):**

A função mode() é usada para encontrar a moda, que representa o valor mais frequente em um conjunto de dados. Enquanto mean() e median() fornecem medidas de tendência central, mode() é crucial para entender a concentração de valores em torno de um ponto específico.

Exemplo mode():

conjunto\_dados <- c(2, 3, 3, 5, 6, 6, 7, 8, 8, 8)

moda <- as.numeric(names(sort(table(conjunto\_dados), decreasing = **TRUE**)[1]))

print(moda)

# Saída: 8

Copiar código

**sd() (Desvio Padrão):**

O desvio padrão, calculado pela função sd(), é uma medida da dispersão ou variabilidade dos dados em relação à média. Ele indica quão longe os valores individuais estão da média, fornecendo uma visão mais detalhada da distribuição dos dados.

Exemplo sd():

notas <- c(75, 82, 88, 91, 95)

desvio\_padrao <- sd(notas)

print(desvio\_padrao)

# Saída: 6.557438

Copiar código

**var() (Variância):**

A função var() calcula a variância, que é o quadrado do desvio padrão. Assim como o desvio padrão, a variância fornece informações sobre a dispersão dos dados, destacando a amplitude das flutuações em relação à média.

Exemplo var():

alturas <- c(160, 165, 170, 175, 180)

variancia <- var(alturas)

print(variancia)

# Saída: 41.25

Copiar código

Essas funções matemáticas são fundamentais para entender a distribuição e variabilidade dos dados, fornecendo informações valiosas para análise estatística e tomada de decisões em análise de dados.

**Para saber mais: funções min() e max()**

As funções min() e max() são comumente utilizadas no R para encontrar os valores mínimo e máximo em um conjunto de dados, respectivamente. Elas são ferramentas essenciais para análise estatística, manipulação de dados e tomada de decisões em programação R.

A função min() retorna o valor mínimo de um vetor numérico ou de uma sequência de valores, enquanto a função max() retorna o valor máximo. Ambas as funções são aplicáveis ​​a uma variedade de objetos em R, incluindo vetores, matrizes e data frames.

Quando aplicadas a vetores ou matrizes, as funções min() e max() retornam o valor mínimo ou máximo de todos os elementos, respectivamente. Se aplicadas a data frames, elas retornam os valores mínimos ou máximos de cada coluna numérica.

Vamos explorar algumas situações práticas para entender como as funções min() e max() podem ser aplicadas:

1 - **Vetores:**

# Criando um vetor

vetor <- c(5, 10, 3, 8, 15)

# Encontrando o valor mínimo e máximo

minimo <- min(vetor)

maximo <- max(vetor)

# Imprimindo os resultados

print(minimo) # Saída: 3

print(maximo) # Saída: 15

Copiar código

2 - **Matrizes:**

# Criando uma matriz

matriz <- matrix(c(5, 10, 3, 8, 15, 2, 7, 12, 9), nrow = 3, ncol = 3)

# Encontrando o valor mínimo e máximo

minimo\_matriz <- min(matriz)

maximo\_matriz <- max(matriz)

# Imprimindo os resultados

print(minimo\_matriz) # Saída: 2

print(maximo\_matriz) # Saída: 15

Copiar código

3 - **Data frames:**

# Criando um data frame

dados <- data.frame(Idade = c(20, 25, 22, 30, 24, 28))

# Encontrando o valor mínimo e máximo das idades

min\_idade <- min(dados$Idade)

max\_idade <- max(dados$Idade)

# Imprimindo os resultados

print(min\_idade) # Saída esperada: 20

print(max\_idade) # Saída esperada: 30

Copiar código

**Paralelo com as funções which.min() e which.max():**

As funções which.min() e which.max() complementam as funções min() e max() ao retornar os índices dos valores mínimo e máximo, respectivamente. Como apresentado anteriormente em vídeo, elas são úteis quando é necessário saber a posição do valor mínimo ou máximo dentro de um vetor.

As funções min() e max() são fundamentais para encontrar valores extremos em conjuntos de dados, enquanto which.min() e which.max() fornecem informações adicionais sobre as posições desses valores dentro dos vetores. Combinadas, essas funções oferecem um poderoso conjunto de ferramentas para análise e manipulação de dados em R.