- Tutorial
- Exercícios
- Apostila

2. Tutoriais de Funções Matemáticas no R

Os 10 mandamentos do R Agora que já tem alguma experiência com a linguagem, veja os 10 mandamentos do R! Ao final do curso revisite esses mandamentos e avalie quantos pecados deve confessar aos monitores..

- **dezmanda**

Operações com Vetores

Estude os resultados dos comandos nos três exemplos abaixo, para entender as operações com vetores.

```
a <- 1:10
b <- -(1:10)
sum(a+b)
```

```
a <- seq(from=0, to=1, length = 9)
b <- seq(from=0, to=0.5, length = 9)
a/b
b/a
1+b/a
(1+b)/a
```

```
a <- rep( c(1,2), each=3 )
b <- rep( c(3,4), 6)
a+b
```

Cálculo da Média

Uma pessoa em dieta anotou seu peso mensalmente durante um ano e obteve os valores:

```
pesos <- c(78.4, 79.8, 76.0, 75.3, 77.4, 78.6, 77.9, 78.8, 79.2, 75.2, 75.0,
79.4)
```

A diferença de peso entre um mês e o consecutivo é obtida pela função `diff`:

```

```

```
pesos.dif <- diff(pesos)
```

Que tem a particularidade de retornar como resultado um vetor de tamanho igual ao comprimento do vetor de entrada, menos uma posição.

```
length(pesos)
length(pesos.dif)
```

Vamos calcular o valor mínimo, máximo e médio do peso:

```
max(pesos)
min(pesos)
mean(pesos)
```

O valor mínimo e máximo também é obtido com:

```
range(pesos)
```

E mesmo que não tivéssemos a função `mean` poderíamos calcular a média com:

```
sum(pesos)/length(pesos)
```

Valores Faltantes e Não-Numéricos

Calcule a média do logaritmo na base 10 das diferença de peso, obtidas no tutorial anterior ("Cálculo da Média"):

```
mean( log(pesos.dif, base=10) )
```

Este comando retorna um aviso e um resultado não numérico, `NaN`, pois não existem logaritmos de números negativos. `NaN` significa *Not a Number*, e serve para alertar que o usuário tentou realizar uma operação matemática não definida numericamente, como dividir por zero,

```
pesos.dif
log(pesos.dif, base=10)
```

Basta um valor faltante (`NA`, que significa *Not Available*) ou não numérico (`NaN`) para que operações numéricas retornem `NA` e `NaN`, respectivamente. Verifique:

```
(falta.um <- c(rep(10,9),NA))
mean(falta.um)
```

```
sqrt(-1:9)
mean(sqrt(-1:9))
```

Muitas funções têm um argumento lógico `na.rm` que, se declarado verdadeiro, desconsidera os valores `NA` e `NaN`. Verifique:

```
mean(falta.um, na.rm=TRUE)
var(sqrt(-1:9), na.rm=TRUE)
```

Índice de Shannon

O índice de diversidade de Shannon é dado pela fórmula:

$$H´=-\sum \ p_i \ ln p_i$$

onde p_i é a proporção de indivíduos da espécie i em relação ao total de indivíduos. Construa um objeto de abundâncias de espécies em uma comunidade e calcule o valor do índice, usando objetos intermediários, e verificando-os:

```
abund <- c(13, 23, 29, 29, 30, 48, 72, 76, 83, 84, 86, 97,102, 229, 343)
tot <- sum(abund)
tot
pi <- abund/tot
pi
log.pi <- log(pi)
log.pi
pi.log.pi <- pi*log.pi
pi.log.pi
H <- - sum(pi.log.pi)
H
```

O mesmo cálculo pode ser feito em uma única linha com:

```
-sum( abund/sum(abund) * log( abund/sum(abund) ) )
```

Compare os resultados.

Onde foi parar o pi?

No tutorial anterior criamos um objeto `pi`, mas este é nome que o R usa para armazenar o número π[1)]!!!

Mas não se preocupe. O objeto `pi` original pertence ao pacote *base* e continua lá.

Como o R busca qualquer objeto chamado na ordem estabelecida no *search path* e como o seu workspace está na frente do pacote *base*, ele irá usar o novo objeto `pi` se você digitar apenas o nome do objeto, pois este será o primeiro objeto com este nome que ele encontrará. Verifique o caminho de busca (*search path*) com:

```
search()
```

Sua área de trabalho (*workspace*) tem o nome padrão `.GlobalEnv`. Qual a posição dela no caminho de busca? Agora chame o objeto pi de seu *workspace*:

```
pi
```

E chame o verdadeiro número π explicitando o ambiente (pacote) onde ele está:

```
base::pi
```

Amplitude de uma Amostra Normal

Vamos simular uma amostra de 10 valores, tomados de uma distribuição normal com média = 10 e desvio-padrão = 2,5:

```
normal.1 <- rnorm(10, mean = 10, sd = 2.5)
```

Calcule mínimo e máximo e a amplitude destes valores:

```
range(normal.1)
diff( range(normal.1) )
```

O que acontece à medida que aumentamos o tamanho da amostra? Verifique para simulações de 100, 1000 e 10000 valores:

```
diff( range ( rnorm(100, mean = 10, sd = 0.5) ) )
diff( range ( rnorm(1000, mean = 10, sd = 0.5) ) )
diff( range ( rnorm(10000, mean = 10, sd = 0.5) ) )
```

Qui-quadrado na unha

Um ecólogo estimou a proporção de cinco tipos de itens alimentares para uma espécie de ave em uma área, que foi de 60%, 28%, 9%, 2,5% e 0,5%. No mesmo local, amostrou ao acaso eventos de alimentação desta ave, contando quantos eventos foram de consumo de cada um dos itens. Os resultados das contagens foram 544, 285, 117, 54, 12, respectivamente.

Crie objetos com estes valores:

```
disponivel <- c(60,28,9,2.5,0.5)
consumido <- c(544,285,117,54,12)
```

Será que esta espécie tem preferência por algum item? Se isso não acontecer, espera-se que 60% do total de itens consumidos sejam do primeiro tipo, e assim por diante. O total de itens consumidos é:

```
tot.itens <- sum(consumido)
```

E os valores esperados pela hipótese de falta de preferência serão:

```
esperado <- tot.itens*disponivel/100
```

O que resulta em um desvio de:

```
desvio <- consumido - esperado
desvio
```

Para testar esta diferença, calculamos o valor do Qui-quadrado, que é a somatória de cada desvio elevado ao quadrado e dividido pelo respectivo valor esperado:

```
d.quad <- desvio^2/esperado
qui2 <- sum(d.quad)
qui2
```

Qual a chance de um valor de Qui-quadrado maior ou igual a este ocorrer por acaso (ou seja, mesmo que não haja preferência)? Como são cinco itens alimentares, temos quatro graus de liberdade, e o que queremos saber é a probabilidade do Qui-quadrado sob a hipótese nula ter um valor igual ou maior ao observado. Obtemos isto com a função de probabilidade acumulada da distribuição de Qui-quadrado:

```
pchisq(q=qui2, df=4, lower.tail=FALSE)
```

A probabilidade é baixíssima, o que indica que o consumo não foi na proporção da disponibilidade dos itens alimentares. Usamos o argumento `lower.tail=FALSE` para que a função retorne a probabilidade da parte que resta da distribuição, após este valor[2)], que está representado em vermelho na figura abaixo:

Extras

Para reproduzir a figura acima digite os comandos abaixo. Para entender, veja a ajuda da função `curve`.

```
## Faz o grafico da funcao Qui-quadrado com 4 graus de liberdade,
## veja ajuda funcao curve
curve(dchisq(x, df=4),0,70, xlab="Qui-quadrado, 4 g.l.", ylab="Densidade
probabilística")
## Sobrepoe uma linha vermelha a partir
##do valor calculado do Qui-quadrado
curve(dchisq(x, df=4), 56.93, 70, add=T, col="red", lwd=2)
```

1)
razão entre o perímetro e o diâmetro do círculo

2)
mais rigorosamente, da área sob a curva a partir deste valor, isto é, a integral da função de densidade probabilística de Qui-quadrado deste valor até +∞