Introdução ao R

Apresentação desenvolvida e cedida por Isaías V. Prestes

O que é o R?

Um ambiente que integra diversos programas para computação estatística e composição de gráficos.

- permite manipulação e armazenamento de dados.
- oferece um conjunto de operadores para cálculos sobre vetores e matrizes.
- coloca à disposição do usuário uma grande coleção de ferramentas para análise de dados.
- comunicação por meio de uma linguagem simples e eficaz, similar à linguagem S.

Instalação do R

O R pode ser obtido a partir de algum "*mirror*" em seu site oficial, ***The R Project for Statistical Computing***:

http://www.r-project.org

Para plataforma Windows 9x/2k/XP, diretamente em

http://gauss.est.ufpr.br/CRAN/bin/windows/base

Ajuda do R?

Se foi feita a instalação completa do R, é muito provável que esteja disponível uma vasta documentação própria do R.

- Perguntas freqüentes do R
- Perguntas freqüentes do R para ambiente Windows
- Manuais em *Portable Document Format* (PDF)
- Ajuda online via comando help( ).
- Ajuda em HTML (com Introdução, Busca, Pacotes, Linguagem R, Instalação e Administração, etc.)

Um comando importante – fundamental – é o help( ):

```
help( FUNCAO )
help( "ASSUNTO" )
```

Ajuda para lidar com o R?

• R por Exemplos, de Ajay Shah (inglês)
http://www.mayin.org/ajayshah/KB/R/index.html

• Tutorial de Introdução ao R (UFPR)
http://www.est.ufpr.br/Rtutorial/

• Using R for psychological research, de William Revelle (Northwestern University)
http://personality-project.org/r/r.guide.html

• R_STAT · Linguagem de programação R – Lista de discussão
http://br.groups.yahoo.com/group/R_STAT/

Operações Básicas

Operador	Descrição	Exemplo
`+`	Adição	`2 + 2`
`-`	Subtração	`2 - 2`
`*`	Multiplicação	`2 * 2`
`/`	Divisão	`2 / 2`
`^`	Potenciação	`2 ^ 2`

Atribuir valor a um objeto (variável):

```
Resp <- 2 * 2 + 1
2.5 * (1.71 + 1^0) -> Resp
```

O separador decimal é o ponto "." e não vírgula ","

Operações Básicas

Funções Matemáticas Básicas:

Valor Absoluto `abs(x)`	Raiz Quadrada `sqrt(x)`	Exponencial `exp(x)`
Seno `sin(x)`	Cosseno `cos(x)`	Tangente `tan(x)`
Arco Seno `asin(x)`	Arco-cosseno `acos(x)`	Arco Tangente `atan(x)`
Logaritmo `log(x,base)`	Log. Natural `log(x)`	Log. Base 10 `log10(x)`
Arredondamento `round(x)`	Truncagem `trunc(x)`	Inteiro Inferior `floor(x)`

Operadores relacionais

Consideremos `x` e `y` variáveis numéricas:

Operador	Descrição	Exemplo
<=	Menor igual a	`x <= y`
<	Menor que	`x < y`
>=	Maior igual a	`x >= y`
>	Maior que	`x > y`
!=	Diferente de	`x != y`
==	Igual a	`x == y`

Operadores lógicos

Consideremos **x** e **y** variáveis numéricas:

Operador	Descrição	Exemplo
&	'E' lógico	`(x < y)&(x==x)`
&&	'E' lógico	`(x < y)&&(x==x)`
|	'OU' lógico	`(x < y)|(x==x)`
||	'OU' lógico	`(x < y)||(x==x)`
!	Negação	`!(x < y)&(x==x)`

xor(expr,expr)	'OU' exclusivo	`X <- 2`
		`xor(x==2,x==3)`
		`TRUE`

Vetores

Para criarmos um vetor, utilizamos a seguinte função:

array(data = DADOS, dim = NÚMERO DE CASELAS, dimnames = NOMES)

Onde

data: dados para o vetor
dim: dimensão do vetor, número de caselas.
dimnames: um conjunto contendo o nome de cada dimensão.

Exemplo:

```
# Vetor de tamanho 10 repleto de zeros.
X <- array(0,10)
# Abaixo um vetor de tamanho 8 repetindo a seqüência
# {0,1,2,3} duas vezes.
x <- array(data=c(0,1,2,3) , dim=8)
# De forma mais sintética e multidimensional:
x <- array( c(0,1,2,3) , dim=c(2,2) )
# Um vetor formado por uma seqüência propriamente dita:
x <- array( seq(1,10,0.5))
```

Matrizes

Para criarmos uma matriz, utilizamos a seguinte função:

matrix(data = DADOS, nrow = L, ncol = C, byrow = FALSE, dimnames = NULL)

Onde

data: dados para a matriz
nrow: número de linhas da matriz
ncol: número de colunas da matriz
byrow: TRUE ou FALSE, preencher a matriz 'por linhas'.
dimnames: uma lista contendo o nome de cada dimensão.

Exemplo:

```
      # Matriz de {0,1} por linha com rótulos.
X <- matrix(data = c(0,1), nrow = 2, ncol = 2, byrow
= TRUE, dimnames = list(c('LIN1','LIN2'),c('COL1','COL2')))
      # Abaixo um vetor de tamanho 8 repetindo a seqüência
Y <- matrix(c(76,56,0,1),2,2,byrow=FALSE,dimnames=
list(c('Deputados','Senadores'),c('Honestos','Desonestos')))
```

Data Frames

O que é um *DataFrame*? Um objeto tal do R, com linhas e colunas (dimensional), capaz de comportar colunas de números ou palavras (strings).

Para criarmos um dataframe, utilizamos a seguinte função:

```
data.frame(DADOS)
```

Exemplo:

```
Notas <- c(2.3,1.5,4.5,5.9)
Nomes <- c('Silva','Mengueli','Soares','Barry
R. James')
Estado <- c('REC','REC','REC','REC')
Notas.Zeal <- data.frame(Notas,Nomes,Estado)
is.data.frame(Notas.Zeal)
```

Listas

O que é uma lista? Um objeto tal do R que comporta diferentes tipos de dados (número, NaN, palavras) e objetos (vetor, matriz, por exemplo).

Para criarmos uma lista, utilizamos a seguinte função:

```
list(DADOS_OBJETOS)
```

Exemplo:

```
x <- list(nome="Isaias", idade=25, sexo=1,
universidade="UFRGS")
y <- list(nome="Ergon",idade=125,sexo=1,
universidade="York University")
z <- list(nome="Ramelle", idade=31, sexo=0,
universidade="Stanford University")
Data.estudantes <- rbind(x,y,z)
list(666,"abobora",”Barry R. James",Data.estudantes)
```

Abrir um banco de dados

Um arquivo de um banco de dados pode ser carregado para um Data Frame no R por meio da função read.table:

read.table(file , header = FALSE , sep = "“ , quote = "\"", dec = ".", row.names, col.names, na.strings = "NA", fill = TRUE, blank.lines.skip = TRUE)

Onde

file : caminho e nome do arquivo de dados
header : o arquivo contém rótulos das variáveis {TRUE,FALSE}
sep : caractere separador dos dados {“,”,”;”,”\t”,” “}
quote : define o caractere para ‘strings’.
dec : separador de decimal {“.”,”,”}
row.names : conjunto com nome das linhas
col.names : conjunto com nome das colunas
fill : aplica espaços em branco se linhas de diferente tamanho
blank.lines.skip : ignorar linhas em branco {TRUE,FALSE}

Abrir um banco de dados

Exemplos:

```
# Obtendo os dados; enviando para um DataFrame
dados <- read.table(file='D:/Centro_de_Estudos/
Amostragem_II/soja-ord.csv',header=TRUE,sep = ";",
quote="\"", dec=",")

# Obtendo os dados de um arquivo CSV
dados <- read.csv2(file='D:/Centro_de_Estudos/
Amostragem_II/soja-ord.csv‘)
```

Outras funções relevantes para importar dados e seus padrões:

- **read.csv** : cabeçalho, separador vírgula, decimal ponto.
- **read.csv2** : cabeçalho, separador ponto e vírgula, decimal vírgula.
- **read.delim** : cabeçalho, separador tabulador “\t”, decimal ponto.
- **read.delim2** : cabeçalho, separador tabulador “\t”, decimal vírgula.

Abrir um R-Script

Um arquivo de script pode ser carregado no R por meio do seguinte comando:

```
source(ARQUIVO)
```

Onde **ARQUIVO** é o caminho e o nome do arquivo com o script.

Convencionou-se atribuir a extensão .R ou .RData para arquivos do R.

Exemplo:

```
source('D:/Centro_de_Estudos/Amostragem_II/trab1.R')

# Considerando que o arquivo .R se encontra no
# diretório de trabalho
source(trab.R)
```

Gráficos

Gráficos de duas dimensões (x,y) podem ser criados no R por meio da função plot( ):

```
plot(DADOS_ABS, DADOS_ORD, col=COR, type=TIPO,
pch='CARACTERE', lwd=CODIGO, main='TITULO',
sub='TITULO_INFERIOR, xlab='NOME_X', ylab='NOME_Y')
```

Onde:

col : define a cor dos pontos 'plotados'.
type : define o tipo dos pontos.
pch : define um caractere especial para plotagem.
lwd : espessura da linha 'plotada'.
main : título do gráfico.
sub : título na parte inferior do gráfico.
xlab : rótulo do eixo das abscissas.
ylab : rótulo do eixo das ordenadas.

Gráficos

Gráficos de pizza podem ser criados por meio da função pie( ):

```
pie(DADOS, labels = names(x), edges = 200, radius =
0.8, col = NULL, main = NULL, ...)
```

Onde:

labels : são os nomes de cada 'fatia'.
edges : qualidade da circunferência do gráfico.
radius : tamanho do raio da circunferência do gráfico.
col : permite especificação de um vetor de cores
main : título do gráfico

Exemplo:

```
zeta <- rpois(5,10) ; zeta <- zeta/sum(zeta)
pie( zeta, labels=c('A','B','C','D','F'),
col=c(1,2,3,4,5), main='Coisas desconhecidas no meu
quarto')
```

Gráficos

Histogramas podem ser criados no R por meio da função hist( ):

```
hist(DADOS, breaks = NUMERO, freq = TRUE, col = COD,
border = NULL, main = paste("Histogram of" , xname),
xlim = range(breaks), ylim = NULL, xlab = xname, ylab,
axes = TRUE, labels = FALSE, nclass = NULL, ...)
```

Onde:

breaks : define o número de limites para fronteiras das classes.
freq : se TRUE temos freqüências, se FALSE temos prob.
col : cor(es) para as barras do histograma.
border : define cor da borda.
nclass : número de classes para o histograma.

Exemplos:

```
x <- rnorm(100) ; hist(x,breaks=3)
hist(x,breaks=c(-4,-2,0,3))
```

Estatística descritiva simples

Seja **x** um vetor, matriz, ou coluna de um dataframe, numérico:

Função	Descrição	Exemplo
`table(x)`	**Exibe tabela com freqüências dos elementos de x**	`table(c(2,2,1))` `table(x)`
`summary(x)`	**Resumo descritivo com mínimo, 1º quartil, moda, média, 3º quartil e máximo**	`summary(x)`
`min(x)`	**Valor mínimo de x**	`min(c(1,2,3))`
`max(x)`	**Valor máximo de x**	`max(c(0,1,0))`
`range(x)`	**Amplitude (mínimo-máximo)**	`range(c(0,1,10))`
`median(x)`	**Mediana**	`median(c(1,2,3,3))`

Estatística descritiva simples

Função	Descrição	Exemplo
`mean(x)`	**Média de x**	`mean(c(0,pi,2))`
`var(x)`	**Variância corrigida de x**	`var(c(pi^exp(1),-pi^exp(1),0))`
`sd(x)`	**Desvio padrão de x**	`sd(x)== var(x)^0.5`
`sum(x)`	**Somatório dos elementos de x**	`sum(x-mean(x))`
`cumsum(x)`	**Somatório cumulativo de x**	`cumsum(x-mean(x))`
`stem(x)`	**Diagrama de folhas**	`stem(x)`
`cor(x1,x2)`	**Correlação**	`cor(x,3*x,method = "pearson"))`
`cov(x1,x2)`	**Covariância**	`cov(x,x^log(x),method = "spearman")`

Distribuições de Probabilidade

O R oferece em seu pacote 'Stats' funções para operarmos com as distribuições de probabilidade mais conhecidas.

Quatro destas funções merecem destaque:

d : calcular a densidade de probabilidade f(x) no ponto x
p : calcular a função distribuição de probabilidade F(x) no ponto
q : calcular o quantil correspondente a uma dada probabilidade
r : gerar números pseudo-aleatórios seguindo uma dada distribuição

Distribuições de Probabilidade

Onde devemos seguir o padrão:

d[DISTRIBUIÇÃO](ARGUMENTOS)
p[DISTRIBUIÇÃO](ARGUMENTOS)
q[DISTRIBUIÇÃO](ARGUMENTOS)
r[DISTRIBUIÇÃO](ARGUMENTOS)

Exemplos:

```
rbeta(10, shape1 = 1, shape2 = 2)
rcauchy(31, location = 0, scale = 1)
pnorm(3.996,0,1)
qchisq(p, df, lower.tail = TRUE)
qt(0.975,df=10)
qgamma(0.05,2,0.5) ; qchisq(0.05,4)
```

Distribuições de Probabilidade

Nome da distribuição	Nome no R
Beta	`beta`
Binomial	`binom`
Cauchy	`cauchy`
Qui-quadrado	`chisq`
Exponencial	`exp`
Gama	`gamma`
Geométrica	`geom`
Hipergeométrica	`hyper`
Log-normal	`lnorm`
Logística	`logis`

Distribuições de Probabilidade

Nome da distribuição	Nome no R
Multinomial	`multinom`
Binomial Negativa	`nbinom`
Normal	`norm`
Poisson	`pois`
T de Student	`t`
Uniforme	`unif`
Weibull	`weibull`

Estrutura condicional

Como na maioria das linguagens de programação, a linguagem do R oferece estruturas de controle.

Estruturas condicionais:

IF simples: `if( CONDICAO ) EXPRESSAO`

IF-ELSE:
```
if( CONDICAO ) { EXPR_PADRAO }
else { EXPR_ALTERNATIVA }
```

Exemplo:
```
if( ENE <= 30.0 ) {
      cat("Rejeitamos Ho") }
else {
      cat("Aceitamos Ho")
}
```

Estruturas de repetição

O uso do **FOR**:

```
for( VAR_CONTADOR in SEQUENCIA ) EXPRESSAO
```

Exemplo:

```
for ( i in 2:31 ) {
    THETA <- i
    if ( THETA <= 30 ) {
        cat( THETA, " elefantes não incomodam muita
gente...", " \n")
    }
    else {
        cat( THETA, " elefantes incomodam muita gente,
e muito mais", "\n")
    }
}
```

Estruturas de repetição

O uso do **WHILE**:

```
while( CONDICAO ) EXPRESSAO
```

Exemplo:

```
x <- 10
amostra <- 1:10
while ( x >= 5 ) {
      x <- sample(amostra,1)
      cat("Extraí o número ", x ,"\n")
}
```

Estruturas de repetição

O uso do **REPEAT**:

```
repeat { EXPRESSAO }
```

Exemplo:

```
amostra <- 1:10
repeat {
      x <- sum( sample(amostra,2) )^2 ; print(x)
      # Um dia teremos que sair daqui!
      if ( x <= 20 ) break
} # fim do repeat
```

Criação de funções

O **R** permite que o usuário crie suas próprias funções. Vejamos o 'eidos' dessa criação:

```
nome_da_função <- function(arg1,arg2,...,argN) {
      conteúdo
      }
```

Exemplo:

```
varpop <- function(x) {
      xbar <- mean(x)
      n <- length(x)
      sum( (x - xbar )^2 ) / n } # Fim
varpop( MinhaPop )
```

E apenas começamos...

Testes de Hipóteses no R

Teste para média de uma população

O Teste T

O pacote 'Stats' do R oferece uma função capaz de aplicar o Teste-t sobre um ou dois vetores de dados.

```
t.test(x, y = NULL, alternative = c("two.sided", "less",
"greater"), mu = 0, paired = FALSE, var.equal = FALSE,
conf.level = 0.95, ...)
```

Onde:

x,y : vetores de dados
alternative : teste bilateral "two sided" ($H_1: \mu \neq \mu_0$)
teste à esquerda "less" ($H_1: \mu < \mu_0$)
teste à direita "greater" ($H_1: \mu > \mu_0$)
mu : média de H_0
paired : amostras pareadas {TRUE,FALSE}
var.equal : mesma variância {TRUE,FALSE}
conf.level : nível de confiança (1 - nível de significância) intervalo

Teste para média de uma população

1. Teste para uma amostra:

```
pulgas <- rnorm(31,5,3)

# Testando H0: mi = m0 = 10 vs. H1: mi < m0
t.test(pulgas,mu=10,alternative = "less")

# Testando H0: mi = m0 = 10 vs. H1: mi ≠ m0
t.test(pulgas,mu=10,alternative = "two.sided")

# Testando H0: mi = m0 = 10 vs. H1: mi < m0
t.test(pulgas,mu=10,alternative = "greater")
```

Teste para média de uma população

2. Teste para duas amostras:

```
x <- rnorm(15,mean=75,sd=10^0.5)
y <- rnorm(15,mean=82,sd=15^0.5)

# Testando H0: Mi(x)-Mi(y) = 0 vs. H1: Mi(x)-Mi(y) < 0
t.test(x,y,alternative = "less",var.equal=FALSE)

# Testando H0: Mi(x)-Mi(y) = 0 vs. H1: Mi(x)- Mi(y) ≠ 0
t.test(x,y,alternative = "two.sided", var.equal=FALSE)
t.test(x,y,alternative = "two.sided", var.equal=TRUE)

# Testando H0: Mi(x)-Mi(y) = 0 vs. H1: Mi(x)-Mi(y) > 0
t.test(x,y,mu=10,alternative = "greater",
  var.equal=FALSE)
```

Teste para média de uma população

3. Teste para 2 amostras com observações pareadas:

```
x <- rnorm(15,mean=40,sd=10^0.5)
y <- rnorm(20,mean=60,sd=12^0.5)

# Testando H0: Mi[d]= 0 vs. H1: Mi[d] < 0
t.test(x,y, paired = TRUE, alternative = "less")
t.test(x,y, paired = TRUE, alternative = "l")

# Testando H0: Mi[d]= 0 vs. H1: Mi[d] ≠ 0
t.test(x,y, paired = TRUE, alternative = "t")

# Testando H0: Mi[d]= 0 vs. H1: Mi[d] > 0
t.test(x,y, paired = TRUE, alternative = "g")
```

Teste para variância

O Teste F

O pacote 'Stats' do R oferece uma função capaz de aplicar o Teste-F sobre um ou dois vetores de dados.

```
var.test(x, y, ratio = 1, alternative =
c("two.sided", "less", "greater"), conf.level =
0.95, ...)
```

Onde:

x,y : vetores de dados
ratio : a suposta razão entre σ_x e σ_y
alternative : teste bilateral "two sided" (H_1: $\sigma^2 \neq \sigma^2_0$)
teste à esquerda "less" (H_1: $\sigma^2 < \sigma^2_0$)
teste à direita "greater" (H_1: $\sigma^2 > \sigma^2_0$)
conf.level : nível de confiança (1 - nível de significância) intervalo

Teste para variância

Teste para duas variâncias (amostras):

```
x <- rnorm(12,5,3)
Y <- rnorm(12,5,5)

# Testando H0: sig2zero >= sig2um vs.
#          H1: sig2zero < sig2um
var.test(x,y,alternative = "less")

# Testando H0: sig2zero = sig2um vs.
#          H1: sig2zero ≠ sig2um
var.test(x,y,alternative = "two.sided")

# Testando H0: sig2zero <= sig2um vs.
#          H1: sig2zero > sig2um
var.test(x,y,alternative = "greater")
```

Teste para variância

1. Teste para uma variância:

```
one.lower.var.test <- function(x,sig2,significance) {
n <- length(x)
chi <- (n-1)*var(x)/sig2
PC <- qchisq(significance,df=(n-1))
p_value <- pchisq(chi,df=(n-1))
cat("        Teste para uma variância  \n")
cat("\n")
cat("Chi: ",chi," num gl = ",n-1," PC = ",PC,"\n")
cat("nível de significância: ",significance," p-valor =
  ",p_value,"\n")
cat("hipótese alternativa: sig^2 < sig2zero \n")
cat("\n")
cat("variância amostral: ",var(x),"\n")
cat("\n")
}
x <- rnorm(12,5,3) ; one.lower.var.test(x,sig2=12,0.05)
```

Teste para proporções

Teste Binomial Exato

Uma das funções oferecidas pelo R no sentido de se testar proporções é a binom.test():

```
binom.test(x, n, p = 0.5, alternative =
c("two.sided", "less", "greater"), conf.level = 0.95)
```

Onde:

x : número de sucessos ou um vetor de tamanho 2 com o número de sucessos e fracassos.
n : número de tentativas – ignorado se houver vetor tamanho 2
p : hipotético p_0
alternative : teste bilateral "two sided" (H_1: $p \neq p_0$)
teste à esquerda "less" (H_1: $p < p_0$)
teste à direita "greater" (H_1: $p > p_0$)
conf.level : nível de confiança (1 - nível de significância) intervalo

Exemplo:

```
x <- rbinom(30,1,0.3) ;
binom.test(sum(x),n=30,p=0.5,alternative="l")
```

Teste para proporções

Teste para Proporções Assintótico

O pacote "Stats" do R traz uma função que executa o teste para proporções assintótico. Esta função é a prop.test( ):

```
prop.test(x, n, p = NULL, alternative = c("two.sided",
"less", "greater"), conf.level = 0.95, correct = TRUE)
```

Onde:

- **x** : número de sucessos ou um vetor de tamanho 2 com o número de sucessos e fracassos.
- **n** : número de tentativas – ignorado se x matriz de dimensão 2
- **p** : um vetor com hipotéticos p_0
- **alternative** : teste bilateral "two sided" ($H_1: p \neq p_0$)
  teste à esquerda "less" ($H_1: p < p_0$)
  teste à direita "greater" ($H_1: p > p_0$)
- **conf.level** : nível de confiança (1 - nível de significância) intervalo
- **correct** : correção de continuidade de Yates

Exemplo:

```
x <- rbinom(30,1,0.3) ; prop.test(sum(x),n=30, p=0.5)
```

Teste para proporções

Exemplos:

```
# Dados de Fleiss(1981), p. 139
# H0: Hipótese nula de que as quatro populações amostradas
# apresentam a mesma proporção de fumantes. H0:p1=p2=p3=p4.
# H1: a proporção de fumantes é diferente em uma população
# pelo menos.

fumantes <- c( 83, 90, 129, 70 )
pacientes <- c( 86, 93, 136, 82 )
prop.test(fumantes, pacientes)

# Testar H0: p1=p2 vs. H1: p2 < p1 (atenção em H1)
amo <- c(20,23) ; ene <- c(40,39)
prop.test(amo, ene,alternative="less",correct=TRUE)

# Testar H0: p1=p2 vs. H1: p2 > p1 (atenção em H1)
amo <- c(20,23) ; ene <- c(40,39)
prop.test(amo, ene,alternative="greater",correct=TRUE)
```

O Poder

O Teste T

Podemos calcular o poder de um teste T aplicado sobre uma e duas amostras.

```
power.t.test(n = NULL, delta = NULL, sd = 1, sig.level =
0.05, power = NULL, type = c("two.sample", "one.sample",
"paired"), alternative = c("two.sided", "one.sided"),
strict = FALSE)
```

Onde:

- **n :** tamanho da amostra (observações caso 'pareado')
- **delta :** a verdadeira diferença entre médias
- **sd :** desvio padrão
- **sig.level :** nível de significância
- **type :** tipo de teste T
- **alternative :** teste unilateral ou bilateral
- **strict :** se TRUE, o PODER considera a probabilidade de rejeição na direção oposta do verdadeiro efeito, recomendado para bilateral.

O Poder

Exemplos:

```
power.t.test(power=0.95,delta=5,sd=10,
sig.level=0.05,
alternative="one.sided",type="one.sample")

power.t.test(n = 30, delta = 2,type='one.sample')

power.t.test(n = 30, delta = 2,type='two.sample')

power.t.test(n = 10, delta = 2, type='two.sample',
alternative='one.sided', stric=TRUE)

power.t.test(n = 10, delta = 2, type='two.sample',
alternative='two.sided', stric=TRUE)
```

O Poder

O Teste para Proporções

Podemos calcular o poder do teste para proporções do R aplicado sobre duas amostras.

```
power.prop.test(n = NULL, p1 = NULL, p2 = NULL,
sig.level = 0.05, power = NULL, alternative =
c("two.sided", "one.sided"), strict = FALSE)
```

Onde:

n : tamanho da amostra
p1: probabilidade no grupo 1
p2 : probabilidade no grupo 2
sig.level : nível de significância
alternative : teste unilateral ou bilateral
strict : se TRUE, o PODER considera a probabilidade de rejeição na direção oposta do verdadeiro efeito, recomendado para bilateral.

O Poder

Exemplos:

```
power.prop.test(n = 15, p1 = .50, p2 = .75)

power.prop.test(p1 = .50, p2 = .75, power = .90)

power.prop.test(n = 31, p1 = .5, power = .90)
```

Testes de Normalidade

A verificação da Normalidade de uma amostra ou população, pode ser realizada no R se utilizando, pelo menos, 2 testes:

Teste de Kolmogorov-Smirnov

```
ks.test(x, y, ..., alternative = c("two.sided", "less",
"greater"), exact = NULL)
```

Onde:

x : amostra
y : um vetor de dados ou a distribuição (hipótese) seguida pelos dados.
exact : aplicar teste exato => TRUE

Exemplo:

```
x <- rnorm(45,10,2)
ks.test(x,"pnorm",10,2,exact=TRUE)
ks.test(x,"pexp",1/10,exact=TRUE)
```

Testes de Normalidade

Teste de Shapiro-Wilk

shapiro.test(x)

Onde:

x : amostra

Exemplo:

```
x <- rnorm(45,10,2)
shapiro.test(x)

x <- rpois(10,5) ; shapiro.test(x)
x <- rpois(10,15) ; shapiro.test(x)
 x <- rpois(10,25) ; shapiro.test(x)
x <- rpois(10,31) ; shapiro.test(x)
```