title: “Course on SAR-PolSAR Statistics” author: “Alejandro C. Frery and Antonio C. Medeiros” date: “Initial: 11 June 2017” output: html_document: default html_notebook: default

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

Parte I: Elementos de análisis estadístico de imágenes PolSAR usando R

A1 – Formación del dato SAR

Una escena es iluminada por un haz electromagnético. Cada “mancha” de la señal en la escena generará un pixel de imagen. La intensidad que cada mancha devuelve al sensor es el resultado de la suma coherente de los retornos individuales de los dispersores elementales.

Si $A_n e^{j\phi_n}$ es el retorno (complejo) del dispersor n, y si tenemos N dispersores en la célula iluminada, entonces el retorno total de la célula es

$$A = \sum_{n=1}^{N} A_n e^{j\phi_n}.$$

Así, el retorno (complejo) A puede descomponerse en sus partes real

$$\Re(A) = \sum_{n=1}^{N} A_n \cos\phi_n$$

e imaginaria

$$\Im(A) = \sum_{n=1}^{N} A_n \sin\phi_n.$$

Si en cada pixel observamos A, tenemos una imagen SAR en formato complejo.

Frecuentemente, en vez de A tomamos su amplitud o, más frecuentemente, su intensidad, que está dada por

$$I = \big(\Re(A)\big)^2 + \big(\Im(A)\big)^2.$$

Hasta ahora no podemos afirmar nada sobre las propiedades de I.

```
if(!require(shape)){install.packages("shape"); require("shape")}
```

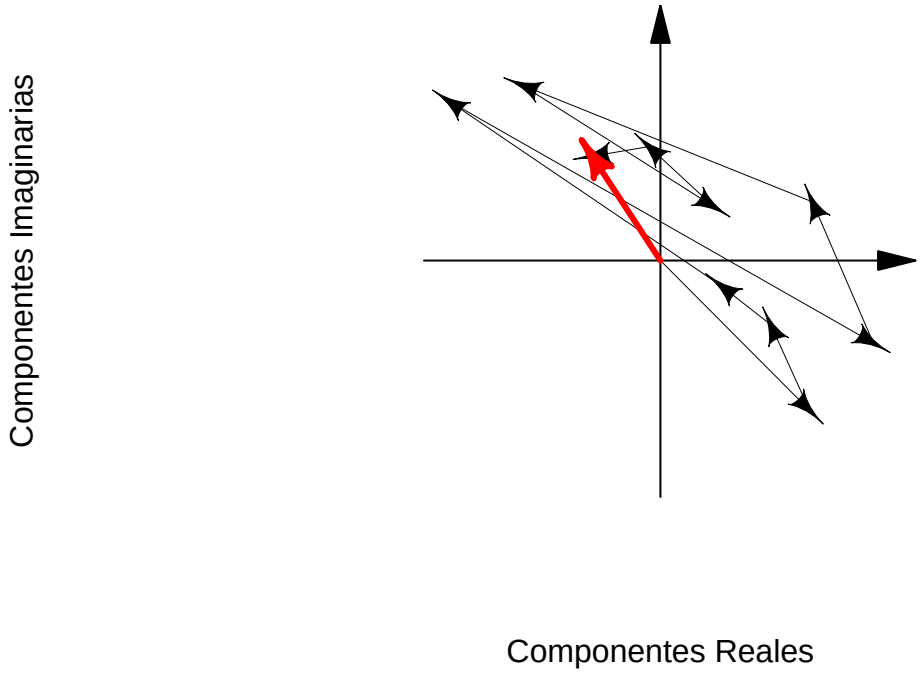
```
## Loading required package: shape
```

```
if(!require(wesanderson)){install.packages("wesanderson"); require("wesanderson")}
```

```
## Loading required package: wesanderson
```

```
set.seed(1234567890, kind="Mersenne-Twister")
factor.reduccion <- .5
### Gráfico de formación del speckle

N <- 10 # número de dispersores elementales
sigma2 <- 1
RAi <- c(0, rnorm(n = N, mean = 0, sd = sqrt(sigma2)))
IAi <- c(0, rnorm(n = N, mean = 0, sd = sqrt(sigma2)))
```

```
### Determinación de la caja para dibujar los ejes
rangex <- range(cumsum(RAi))
rangey <- range(cumsum(IAi))
maxaxis <- max(abs(c(rangex, rangey))) * factor.reduccion

### El área de dibujo
plot(c(-maxaxis, maxaxis), c(-maxaxis, maxaxis), type = "n", asp = 1,
     axes = FALSE,
     xlab = "Componentes Reales", ylab = "Componentes Imaginarias")
Arrows(x0 = -maxaxis, y0 = 0, x1 = maxaxis, y1 = 0, arr.type="triangle")
Arrows(y0 = -maxaxis, x0 = 0, y1 = maxaxis, x1 = 0, arr.type="triangle")

### Los vectores sumándose
Arrows(x0 = RAi[1:(N)], y0 = IAi[1:(N)], x1 = RAi[2:(N+1)], y1 = IAi[2:(N+1)], lwd=.3)

### El vector final
Arrows(x0 = 0, y0 = 0, x1 = RAi[N+1], y1 = IAi[N+1], lwd = 3, col = 2)
```




```
### Fin Gráfico de formación del speckle
```

El modelo básico que podemos tratar es el que describe la situación de tener muchos dispersores elementales ($N \to \infty$), que no se orientan de forma organizada, y que ninguno predomina sobre los otros. Esta es una situación razonable cuando modelamos, por ejemplo, pasto sobre una superficie sin relieve.

Bajo esta suposición, supongamos que las amplitudes retornadas A_n pueden ser descriptas por variables aleatorias independientes e idénticamente distribuídas. Supongamos también que las fases ϕ_n pueden ser descriptas por variables aleatorias independientes e idénticamente distribuídas y, más aún, que por no predominar ninguna fase podemos suponer que siguen una distribución uniforme en $(0, 2\pi]$.

Con estas hipótesis, podemos suponer que vale el Teorema Central del Límite. Siendo así,

$$\big(\Re(A)\big), \big(\Im(A)\big) \to \mathcal{N}(\not{\nu}, \sigma^2 \not{\nu}/2),$$

que denota una variable aleatoria gaussiana bivariada de médias nulas, independientes, y de varianzas iguales a $\sigma^2/2$.

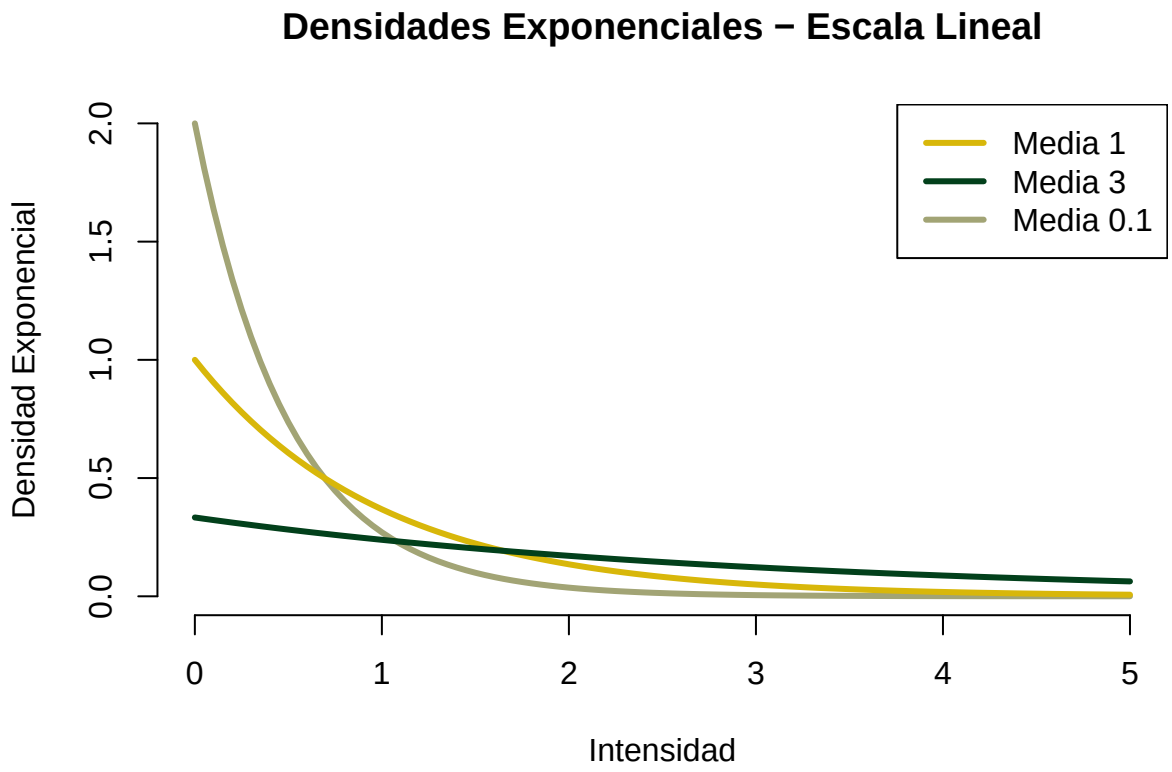
Con esto, la intensidad sigue una ley Exponencial de media σ^2, y podemos escribir

$$I \sim \sigma^2 Y,$$

en que $Y \sim E(1)$. La variable aleatoria Y se denomina speckle, y σ^2 retrodispersión.

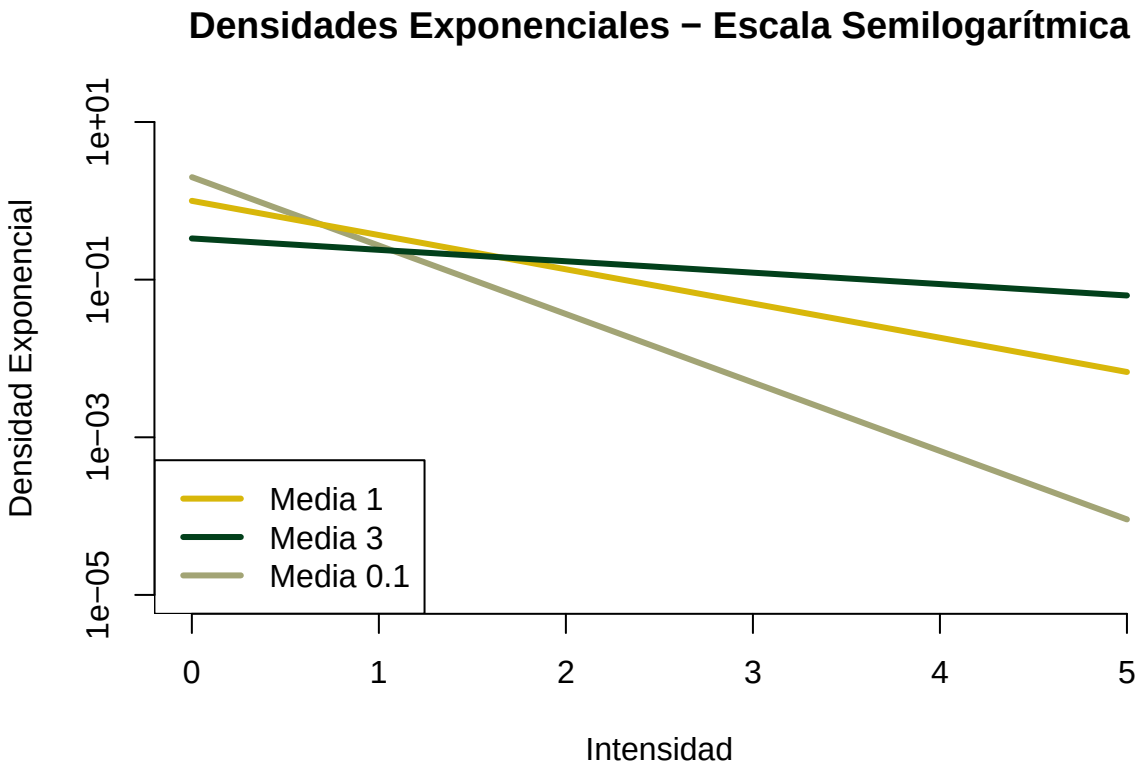
Veamos a seguir tres densidades con medias diferentes en escala lineal.

```
colores <- wes_palette("Cavalcanti", 3)

plot(c(0,5), c(0,2), type="n", axes = FALSE, xlab = "Intensidad", ylab = "Densidad Exponencial",
     main = "Densidades Exponenciales - Escala Lineal")
axis(1)
axis(2)
curve(dexp(x, rate = 2), col=colores[3], lwd=3, from=0, to=5, add=TRUE)
curve(dexp(x, rate = 1), col=colores[1], lwd=3, add=TRUE)
curve(dexp(x, rate = 1/3, ), col=colores[2], lwd=3, add=TRUE)
legend("topright", lwd = c(3,3,3), col = colores, legend = c("Media 1", "Media 3", "Media 0.1"))
```



Veamos ahora las mismas densidades en escala semilogarítmica.

```
plot(c(0,5), c(10^(-5),10), type="n", axes = FALSE, log="y", xlab = "Intensidad", ylab = "Densidad Expo
     main = "Densidades Exponenciales - Escala Semilogarítmica")
axis(1)
axis(2)
curve(dexp(x, rate = 2), col=colores[3], lwd=3, from=0.001, to=5, add=TRUE)
curve(dexp(x, rate = 1), col=colores[1], lwd=3, add=TRUE)
curve(dexp(x, rate = 1/3), col=colores[2], lwd=3, add=TRUE)
legend("bottomleft", lwd = c(3,3,3), col = colores, legend = c("Media 1", "Media 3", "Media 0.1"))
```



Es importante ver siempre las densidades de los modelos para datos SAR en ambas escalas. La escala semilogarítmica revela el comportamiento del modelo para valores muy grandes.

La relación señal-ruido de una ley exponencial de media σ^2 se puede medir por la recíproca del cuadrado del coeficiente de variación, que en la literatura SAR se denomina "número de looks". Como vale que si $Z \sim E(\sigma^2)$, entonces su esperanza y varianza están dadas por $\mathrm{E}(Z) = \sigma^2$ y $\mathrm{Var}(Z) = \sigma^4$, respectivamente; luego el coeficiente de variación es $\mathrm{CV}(Z) = \sigma^4/(\sigma^2)^2$ y el número de looks es 1.

Para mitigar el efecto del speckle, se suele promediar L observaciones (idealmente independientes) del mismo blanco. Así siendo, se tiene una observación en intensidad "multilook":

$$I^{(L)} = \frac{1}{L}\sum_{\ell=1}^{L} \sigma^2 Y_\ell = \sigma^2 \frac{1}{L}\sum_{\ell=1}^{L} Y_\ell.$$

Admitiendo que σ^2 permanece constante, podemos modelar $I^{(L)}$ como una variable aleatoria gama de media unitaria y parámetro de forma L, multiplicada por σ^2. Con esto, $I^{(L)}$ sigue una ley gama de media σ^2 y parámetro de forma L. Denotaremos esta situación $Z \sim \Gamma(\sigma^2, L)$. La varianza de Z es $\mathrm{Var}(Z) = \sigma^4/L$, con lo que el número de looks pasa a ser L.

En general, el número de looks no es conocido. Se puede estimar de varias formas, pero la más simple es como la razón entre el cuadrado de la media muestral y la varianza muestral, ambas calculadas sobre una muestra de datos donde valgan las hipótesis discutidas. A cualquier estimador del número de looks se lo denomina "número equivalente de looks".

A2 – El modelo multiplicativo y distribuciones de intensidad

Según el modelo multiplicativo elemental (para áreas sin textura), la intensidad se puede modelar con una distribución gama de media σ^2 y parámetro de forma L: $Z = \sigma^2 Y \sim \sigma^2\Gamma(1, L)$. Nos interesa σ^2, mientras que Y interfiere y, por ende, se considera ruido.

Diversos factores pueden hacer que σ^2 no se mantenga constante para todos los píxeles del área observada, aún cuando se trate de un mismo tipo de blanco de la escena (blanco homogéneo). En ese caso, se puede modelar σ^2 como otra variable aleatoria, independiente de Y.

A2 – El modelo multiplicativo y distribuciones de intensidad

T1 – Introducción

R é um programa livre e um ambiente computacional de exploração de gráficos e análise de dados (Chambers 6 , 2008). Ele é útil para qualquer pessoa que usa e interpreta dados, das seguintes maneiras:

- Super-calculadora;
- Ambiente com diversos pacotes estatísticos;
- Ferramenta de gráficos de alta qualidade;
- Linguagem de programação multi-uso.

O R é utilizado para explorar, analisar e compreender os dados epidemiológicos. Os dados podem ser importados de fontes como, por exemplo, o DATASUS 1 ou o IBGE 2 em planilhas de Excel.

Como Instalar o R

O R é um software livre para computação estatística e construção de gráficos que pode ser baixado e distribuído gratuitamente de acordo com a licença GNU. O R está disponível para as plataformas UNIX, Windows e MacOS.

O R é case-sensitive, isto é, ele diferencia letras maiúsculas de minúsculas, portanto A é diferente de a. O separador de casas decimais é ponto ".". A vírgula é usada para separar argumentos (informações). Não é recomendado o uso de acentos em palavras (qualquer nome que for salvar em um computador, não só no R, evite usar acentos. Acentos são comandos usados em programação e podem causar erros, por exemplo, em documentos do word e excel).

Demonstrações

Algumas funções do R possuem demonstrações de uso. Estas demonstrações podem ser vistas usando a função demo(). Vamos ver algumas demonstrações de gráficos que podem ser feitos no R. Digite o seguinte na linha de comandos:
> demo(graphics) # Vai aparecer uma mensagem pedindo que você tecle Enter para prosseguir, depois clique na janela do gráfico para ir passando os exemplos.
> demo(persp)
> demo(image)

Pacotes do R

O R é um programa leve (ocupa pouco espaço e memória) e geralmente roda rápido, até em computadores não muito bons. Isso porque ao instalarmos o R apenas as configurações mínimas para seu funcionamento básico são instaladas (pacotes que vem na instalação "base"). Para realizar tarefas mais complicadas pode ser necessário instalar pacotes adicionais (packages).

Como usar um pacote do R

Não basta apenas instalar um pacote. Para usá-lo é necessário "carregar" o pacote sempre que você abrir o R e for usá-lo. Use a função library para rodar um pacote. Por exemplo: Digite library(vegan) na linha de comandos do R.

library(vegan) # Após isso as funcionalidades do vegan estarão prontas para serem usadas. Lembre-se que sempre que abrir o R será necessário carregar o pacote novamente.

Como citar o R, ou um pacote do R em publicações

No R existe um comando que mostra como citar o R ou um de seus pacotes. Veja como fazer:

citation() # Mostra como citar o R To cite R in publications use: R Development Core Team (2011). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/. Veja que na parte com o nome dos autores aparece "R development core team", isso está correto, cite o R desta forma. Algumas pessoas não sabem disso e citam o R com autor Anônimo, isto tira o crédito do time.

Para citar um pacote, por exemplo o vegan, basta colocar o nome do pacote entre aspas.

citation("vegan")

T2 – Lectura de datos

T3 – Operaciones basicas en R

Nociones de programación en R

Os comandos (scripts em linguagem R) devem ser explicitados na linha de comando (console) ou por meio de um editor de texto (a ser comentado depois). Veja os exemplos de operações matemáticas (Quadro 2):

Quadro 2: R como super-calculadora.

%Carregando pacote para tabela knitr & kableExtra

```
require(knitr)
```

```
## Loading required package: knitr
```

```
require(kableExtra)
```

```
## Loading required package: kableExtra
```

```
if(!require(data.table)){install.packag
  es("data.table"); require(data.table)}
```

```
## Loading required package: data.table
```

```
#Criando conjunto de dados

#OperacoesR <- data.table(Operador = c( "+", "-", "*", "/", "^", "abs", "exp", "log", "sqrt", "rnorm"),
#                         Descricao = c( "Adicao", "Subtracao", "Multiplicacao", "Divisao", "Exponenciac
#                         Exemplos = c( "5+4", "5-4", "5*4", "5/4", "5^4", "abs(-5)", "exp(5)", "log(exp
Tabela <- data.table(Operador = c("+", "-","*","/", "^", "abs", "exp", "log", "sqrt", "rnorm" ),
                     Descricao = c("Adicao", "Subtracao", "Multiplicacao", "Divisao", "Exponenciacao",
                     Exemplos = c("5+4", "5-4", "5*4", "5/4", "5^4", "abs(-5)", "exp(5)", "log(exp(5)",
```

```
## Warning in data.table(Operador = c("+", "-", "*", "/", "^", "abs", "exp", :
## Item 2 is of size 9 but maximum size is 10 (recycled leaving remainder of 1
## items)
```

```
kable(Tabela, booktabs = T)
```

Operador	Descricao	Exemplos
+	Adicao	5+4
-	Subtracao	5-4
	Multiplicacao	5*4
/	Divisao	5/4
^	Exponenciacao	5^4
abs	Valor Absoluto	abs(-5)
exp	log logaritmo (default 'e log natural)	exp(5)
log	Raiz quadrada	log(exp(5)
sqrt	sorteia 100.000 valores de uma Normal	sqrt(64)
rnorm	Adicao	rnorm(100000)

O R como calculadora

O forma de uso mais básica do R é usá-lo como calculadora. Os operadores matemáticos básicos são: + para soma, - subtração, * multiplicação, / divisão e ^ exponenciação. Digite as seguintes operações na linha de comandos do R:

%Carregando pacote para tabela knitr & kableExtra

```
 2+2
```

```
## [1] 4
```

```
 2*2
```

```
## [1] 4
```

```
 2/2
```

```
## [1] 1
```

```
 2-2
```

```
## [1] 0
```

```
 2^2
```

```
## [1] 4
```

Use parênteses para separar partes dos cálculos, por exemplo, para fazer a conta 4+16, dividido por 4, elevado ao quadrado:

```
 ((4+16)/4)^2
```

```
## [1] 25
```

Funções do R

O R tem diversas funções que podemos usar para fazer os cálculos desejados. O uso básico de uma função é escrever o nome da função e colocar os argumentos entre parênteses, por exemplo: função(argumentos). função especifica qual função irá usar e argumentos especifica os argumentos que serão avaliados pela função. Não se assuste com esses nomes, com um pouco de pratica eles se tornarão triviais.

Antes de usar uma função precisamos aprender como usá-la. Para isso vamos aprender como abrir e usar os arquivos de ajuda do R.

Objetos do R (O que são?):

O que são os Objetos do R? Existem muitos tipos de objetos no R que só passamos a conhecê-los bem com o passar do tempo. Por enquanto vamos aprender os tipos básicos de objetos.
a) vetores: uma seqüência de valores numéricos ou de caracteres (letras, palavras).
b) matrizes: coleção de vetores em linhas e colunas, todos os vetores dever ser do mesmo tipo (numérico ou de caracteres).
c) dataframe: O mesmo que uma matriz, mas aceita vetores de tipos diferentes (numérico e caracteres). Geralmente nós guardamos nossos dados em objetos do tipo data frame, pois sempre temos variáveis numéricas e variáveis categóricas (por exemplo, largura do rio e nome do rio, respectivamente).
d) listas: conjunto de vetores, dataframes ou de matrizes. Não precisam ter o mesmo comprimento, é a forma que a maioria das funções retorna os resultados.
e) funções: as funções criadas para fazer diversos cálculos também são objetos do R.
No decorrer da apostila você verá exemplos de cada um destes objetos.

Como criar objetos

Objetos vetores com valores numéricos

Vamos criar um conjunto de dados de contém o número de espécies de aves (riqueza) coletadas em 10 locais. As riquezas são 22, 28, 37, 34, 13, 24, 39, 5, 33, 32.

```
aves<-c(22,28,37,34,13,24,39,5,33,32)
```

O comando <- (sinal de menor e sinal de menos) significa assinalar (assign). Indica que tudo que vem após este comando será salvo com o nome que vem antes. É o mesmo que dizer "salve os dados a seguir com o nome de aves".

A letra c significa concatenar (colocar junto). Entenda como "agrupe os dados entre parênteses dentro do objeto que será criado" neste caso no objeto aves.

Para ver os valores (o conteúdo de um objeto), basta digitar o nome do objeto na linha de comandos. aves

A função length fornece o número de observações (n) dentro do objeto.

```
length(aves)
```

```
## [1] 10
```

Objetos vetores com caracteres (letras, variáveis categóricas).

Também podemos criar objetos que contêm letras ou palavras ao invés de números. Porem, as letras ou palavras devem vir entre aspas " ".

```
letras<-c("a","b","c","da","edw")
7
```

```
## [1] 7
```

```
letras
```

```
## [1] "a"   "b"   "c"   "da"  "edw"
```

```
palavras<-c("Manaus","Boa Vista","Belém","Brasília")

palavras
```

```
## [1] "Manaus"    "Boa Vista" "Belém"     "Brasília"
```

Crie um objeto "misto", com letras e com números. Funciona? Esses números realmente são números?

Note a presença de aspas, isso indica que os números foram convertidos em caracteres. Evite criar vetores "mistos", a menos que tenha certeza do que está fazendo.

Operações com vetores

Podemos fazer diversas operações usando o objeto aves, criado acima.

```
max(aves) #valor máximo contido no objeto aves
```

```
## [1] 39
```

```
min(aves) #valor mínimo
```

```
## [1] 5
```

```
sum(aves) #Soma dos valores de aves
```

```
## [1] 267
```

```
aves^2 #...
```

```
##  [1]  484  784 1369 1156  169  576 1521   25 1089 1024
```

```
aves/10
```

```
##  [1] 2.2 2.8 3.7 3.4 1.3 2.4 3.9 0.5 3.3 3.2
```

Agora vamos usar o que já sabemos para calcular a média dos dados das aves.

```
n.aves<-length(aves)              # número de observações (n)
media.aves<-sum(aves)/n.aves                          #média
```

Para ver os resultados basta digitar o nome dos objetos que você salvou

```
n.aves # para ver o número de observações (n)
```

```
## [1] 10
```

```
media.aves # para ver a média
```

```
## [1] 26.7
```

Você não ficará surpreso em saber que o R já tem uma função pronta para calcular a média.

```
mean(aves)
```

```
## [1] 26.7
```

Acessar valores dentro de um objeto [colchetes]

Caso queira acessar apenas um valor do conjunto de dados use colchetes []. Isto é possível porque o R salva os objetos como vetores, ou seja, a sequencia na qual você incluiu os dados é preservada. Por exemplo, vamos acessar o quinto valor do objeto aves.

```
aves[5] # Qual o quinto valor de aves?
```

```
## [1] 13
```

```
palavras[3] # Qual a terceira palavra?
```

```
## [1] "Belém"
```

Para acessar mais de um valor use c para concatenar dentro dos colchetes [c(1,3,...)]:

```
aves[c(5,8,10)] # acessa o quinto, oitavo e décimo valores
```

```
## [1] 13  5 32
```

Para excluir um valor, ex: o primeiro, use:

```
aves[-1] # note que o valor 22, o primeiro do objeto aves, foi excluído
```

```
## [1] 28 37 34 13 24 39  5 33 32
```

Caso tenha digitado um valor errado e queira corrigir o valor, especifique a posição do valor e o novo valor. Por exemplo, o primeiro valor de aves é 22, caso estivesse errado, ex: deveria ser 100, basta alterarmos o valor da seguinte maneira.

```
aves[1]<-100 # O primeiro valor de aves deve ser 100
aves
```

```
##  [1] 100  28  37  34  13  24  39   5  33  32
```

```
aves[1]<-22 # Vamos voltar ao valor antigo
```

Gerar seqüências (usando : ou usando seq)

: (dois pontos) Dois pontos " : " é usado para gerar seqüências de um em um, por exemplo a seqüência de 1 a 10:

```
1:10 # O comando : é usado para especificar seqüências.
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
5:16 # Aqui a seqüência vai de 5 a 16
```

```
##  [1]  5  6  7  8  9 10 11 12 13 14 15 16
```

seq A função seq é usada para gerar seqüências especificando os intervalos. Vamos criar uma seqüência de 1 a 10 pegando valores de 2 em 2.

```
seq(1,10,2) #seq é a função para gerar seqüências, o default é em intervalos de 1.
```

```
## [1] 1 3 5 7 9
```

A função seq funciona assim:

seq(from = 1, to = 10, by = 2), seqüência(de um, a dez, em intervalos de 2)

```
seq(1,100,5) #seqüência de 1 a 100 em intervalos de 5
```

```
##  [1]  1  6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96
```

```
seq(0.01,1,0.02)
```

```
##  [1] 0.01 0.03 0.05 0.07 0.09 0.11 0.13 0.15 0.17 0.19 0.21 0.23 0.25 0.27
## [15] 0.29 0.31 0.33 0.35 0.37 0.39 0.41 0.43 0.45 0.47 0.49 0.51 0.53 0.55
## [29] 0.57 0.59 0.61 0.63 0.65 0.67 0.69 0.71 0.73 0.75 0.77 0.79 0.81 0.83
## [43] 0.85 0.87 0.89 0.91 0.93 0.95 0.97 0.99
```

Gerar repetições (rep)

rep Vamos usar a função rep para repetir algo n vezes.

```
rep(5,10) # repete o valor 5 dez vezes
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5
```

A função rep funciona assim:

rep(x, times=y) # rep(repita x, y vezes) # onde x é o valor ou conjunto de valores que deve ser repetido, e times é o número de vezes)

```
rep(2,5)
```

```
## [1] 2 2 2 2 2
```

```
rep("a",5) # repete a letra "a" 5 vezes
```

```
## [1] "a" "a" "a" "a" "a"
```

```
rep(1:4,2) # repete a seqüência de 1 a 4 duas vezes
```

```
## [1] 1 2 3 4 1 2 3 4
```

```
rep(1:4,each=2) # note a diferença ao usar o comando each=2
```

```
## [1] 1 1 2 2 3 3 4 4
```

```
rep(c("A","B"),5) # repete A e B cinco vezes.
```

```
##  [1] "A" "B" "A" "B" "A" "B" "A" "B" "A" "B"
```

```
rep(c("A","B"),each=5) # repete A e B cinco vezes.
```

```
##  [1] "A" "A" "A" "A" "A" "B" "B" "B" "B" "B"
```

```
rep(c("Três","Dois","Sete","Quatro"),c(3,2,7,4)) # Veja que neste
```

```
##  [1] "Três"   "Três"   "Três"   "Dois"   "Dois"   "Sete"   "Sete"
##  [8] "Sete"   "Sete"   "Sete"   "Sete"   "Sete"   "Quatro" "Quatro"
## [15] "Quatro" "Quatro"
```

caso a primeira parte do comando indica as palavras que devem ser repetidas e a segunda parte indica quantas vezes cada palavra deve ser repetida

Gerar dados aleatórios

runif (Gerar dados aleatórios com distribuição uniforme)

runif(n, min=0, max=1) # gera uma distribuição uniforme com n valores,

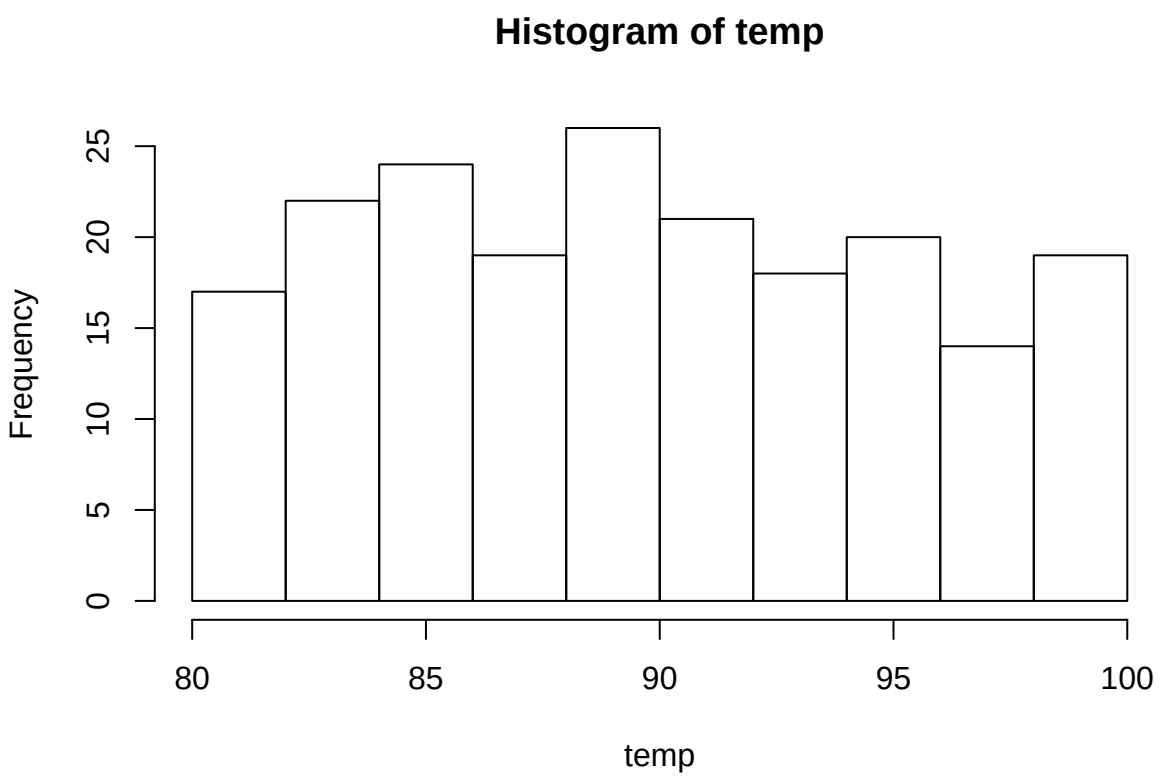
começando em min e terminando em max.

```
runif(200,80,100) # gera 200 valores que vão de um mínimo de 80 até um máximo de 100
```

```
##   [1] 94.94730 84.33717 85.17504 84.08975 89.54754 99.87803 96.07227
##   [8] 81.34506 85.68188 83.21437 94.59126 95.85362 84.96833 94.30739
##  [15] 99.01407 81.21132 99.58919 94.97883 93.38998 84.64158 96.29336
##  [22] 85.53583 85.97113 86.90559 95.96831 99.04142 89.23510 88.29744
##  [29] 97.42144 81.41111 87.30135 80.74547 93.49869 82.76904 80.16308
##  [36] 83.54718 98.49152 96.41023 85.97911 99.40680 92.14278 84.71753
```

```
##  [43] 80.22651 90.75236 85.44035 95.87915 98.57008 92.41261 97.19749
##  [50] 81.31842 85.29289 83.48113 85.50733 88.81695 88.73008 95.35502
##  [57] 95.51636 99.89529 98.00522 99.93613 94.62865 90.85935 92.44002
##  [64] 94.51279 85.76192 83.08418 82.37612 83.58956 84.09752 88.37485
##  [71] 86.44958 84.31305 87.97899 83.27165 87.85617 96.87889 92.93604
##  [78] 84.13157 89.96328 83.72010 99.66509 94.50012 96.13849 90.57087
##  [85] 98.90981 99.55632 99.53490 90.77749 93.81703 86.57303 94.97492
##  [92] 81.00698 91.74935 80.93459 98.62894 83.88563 99.35110 97.40529
##  [99] 97.88867 95.28298 95.49745 98.81533 99.85181 89.48135 83.05897
## [106] 90.94843 89.95016 84.45748 84.63365 97.33555 83.24940 95.20247
## [113] 93.97544 91.38961 82.75753 86.86866 87.05701 86.65106 99.50087
## [120] 86.19347 80.93310 89.10498 98.26766 94.32521 91.83623 82.90964
## [127] 83.27627 83.38104 90.30106 94.21363 95.90079 83.36505 87.91971
## [134] 92.38076 98.34256 83.20907 80.29098 88.47544 81.24279 90.67614
## [141] 96.36069 90.06263 80.81654 90.61257 97.91266 85.65199 99.19205
## [148] 83.25204 92.78204 89.21725 91.03039 99.33278 89.59767 97.48376
## [155] 86.00604 92.93748 98.53382 87.43853 98.07449 84.86855 84.99066
## [162] 97.48741 98.67066 96.82324 96.94507 87.35571 87.03720 98.60651
## [169] 99.45189 88.46982 92.17650 80.46076 84.27640 99.93883 89.63745
## [176] 92.10525 85.92257 92.94571 91.50621 84.74671 95.78387 91.26716
## [183] 80.83173 92.96557 96.44124 97.60199 95.16361 91.10949 87.71409
## [190] 86.66692 97.20501 94.24198 92.04262 94.83969 80.69776 98.52884
## [197] 82.20603 95.02185 90.22252 94.63878
```

```r
temp<-runif(200,80,100)
hist(temp) # Faz um histograma de freqüências dos valores
```



rnorm

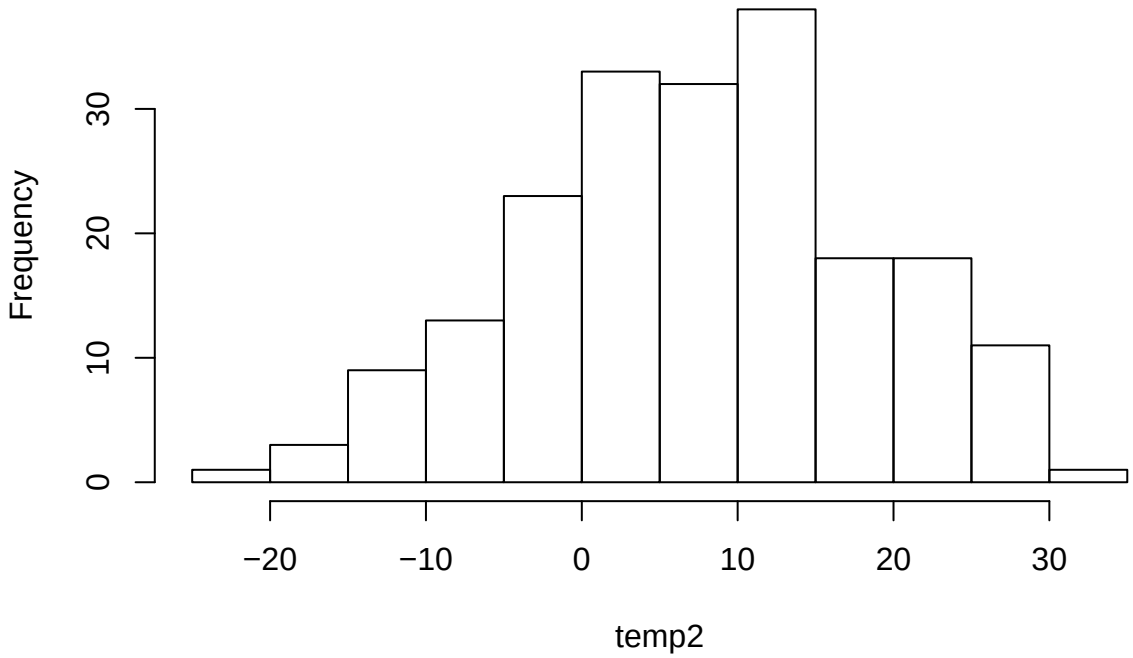
(Gerar dados aleatórios com distribuição normal)

rnorm(n, mean=0, sd=1) # gera n valores com distribuição uniforme, com média 0 e desvio padrão 1.

```r
rnorm(200,0,1) # 200 valores com média 0 e desvio padrão 1
```

```
##   [1] -0.853754727 -0.909345198 -0.212789527  0.153313988  0.154954105
##   [6] -1.520323692  0.303641350 -1.090559160  0.996578836  0.071446376
##  [11]  1.237814999  0.071322377 -0.573850069  1.624010815  0.021846724
##  [16]  0.428014381  0.289638798 -1.533816774  1.305469657  0.063491268
##  [21] -0.049908327 -0.858982496  0.636906699  0.109853071 -0.626986489
##  [26]  1.785821876 -0.252307702 -1.118517123  0.720185892 -0.373806503
##  [31] -0.280559645 -0.910573072 -1.204116080 -1.087435133  1.750948808
##  [36]  0.028607977  0.034291670 -0.826208877 -0.769554384 -1.021136763
##  [41] -0.524209462 -1.397778234 -0.351636346 -0.252949355 -0.574797032
##  [46] -0.136692217 -0.414997841  0.024625593  1.157077906  0.782869463
##  [51] -0.692906244  0.884908596  0.485586187  0.602377749  0.326033580
##  [56]  0.450320651 -1.127972967  0.799748732  1.305405130 -0.049967278
##  [61]  0.646556270  0.211900402 -0.962120724  0.961622686 -0.790104798
##  [66]  0.083559633 -0.996039266  0.107219221  0.118881628 -0.026198835
##  [71]  0.381441744  1.219314842  0.838728095  0.248335096 -0.638764766
##  [76]  2.098434142  0.437695267  0.229082397 -1.669313322  1.324132788
##  [81] -1.084006420  1.016830657  0.684392825  0.952295983 -0.221401796
##  [86]  0.278398060 -0.040087268  1.236006838 -1.020600974  0.061213772
##  [91]  0.496411002 -0.277738541  0.403100941  0.056102563  0.782870383
##  [96]  0.833335904  1.408460823  0.407045616 -1.146499509 -1.037694487
## [101]  2.030900411  0.105792301  1.320425303 -0.604081168 -1.081019593
## [106]  0.784052632 -2.045333811  2.098949008  0.982895128  0.057617270
## [111] -0.093621389  0.897987055 -1.458774254  1.509639297 -0.450120283
## [116]  0.674717245 -0.411852875 -1.010396446 -0.224746852 -1.026314555
## [121]  1.178751357 -0.270216618  1.644356681  0.570897587 -0.863090929
## [126]  1.242586151 -0.990754314 -0.663738694 -1.394836493  0.848727033
## [131]  1.324067441 -1.201946669  1.436767304  0.870523690 -0.876997874
## [136] -0.864806997 -0.048458120  0.376363813  0.818451314 -0.171659877
## [141] -0.890740197 -0.283909779 -0.003297283  0.158358470 -0.303476373
## [146]  0.354957424 -0.441440756  2.340107626  0.662009181  0.320397372
## [151] -0.153325602 -0.543082829 -1.126994356  1.156976420  0.747006857
## [156]  0.071774966  0.080388342 -0.439061791 -1.938157333 -0.767822571
## [161] -0.404045158  0.121281592 -0.351576826  1.327363925  1.205675122
## [166] -1.150670331  0.062621415  0.260331825  0.054429404 -0.132601882
## [171] -0.360311042  0.946920725  0.099619543 -1.967490953 -0.070401591
## [176] -1.483231497  0.342066204 -1.766906740  0.903195326 -0.064333179
## [181]  1.739945184  0.150538093  0.490015477  0.034951169 -0.711838022
## [186] -1.095045887 -1.179200055  0.252284577 -0.192829872 -0.159587375
## [191] -2.606500880  0.656972517  0.516206196 -1.511552091  1.617341891
## [196]  0.985156689 -0.217232309 -1.731071181 -0.393925249 -0.381109143
```

```r
temp2<-rnorm(200,8,10) # 200 valores com média 8 e desvio padrão 10
hist(temp2) # Faz um histograma de frequências dos valores
```



Veja o help da função ?Distributions para conhecer outras formar de gerar dados aleatórios com diferentes distribuições.

Fazer amostras aleatórias

A função sample A função sample é utilizada para realizar amostras aleatórias e funciona assim: sample(x, size=1, replace = FALSE)# onde x é o conjunto de dados do qual as amostras serão retiradas, size é o número de amostras e replace é onde você indica se a amostra deve ser feita com reposição (TRUE) ou sem reposição (FALSE).

```
sample(1:10,5) # tira 5 amostras com valores entre 1 e 10
```

```
## [1] 10  5  3  9  1
```

Como não especificamos o argumento replace o padrão é considerar que a amostra é sem reposição (= FALSE).

sample(1:10,15) # erro, amostra maior que o conjunto de valores, temos 10 valores (1 a 10)

portanto não é possível retirar 15 valores sem repetir nenhum!

```
sample(1:10,15,replace=TRUE) # agora sim!
```

```
##  [1] 10  9  9 10  3  5  8  4  7  4  5 10  4  2  7
```

Com a função sample nós podemos criar varios processos de amostragem aleatória. Por exemplo, vamos criar uma moeda e "jogá-la" para ver quantas caras e quantas coroas saem em 10 jogadas.

```
moeda<-c("CARA","COROA") # primeiro criamos a moeda
```

sample(moeda,10)

ops! Esqueci de colocar replace=TRUE

```r
sample(moeda,10,replace=TRUE) # agora sim
```

```
##  [1] "CARA"  "COROA" "COROA" "COROA" "CARA"  "COROA" "CARA"  "COROA"
##  [9] "CARA"  "CARA"
```

T4 – Visualizacion de imagenes, mejoria de contraste

T5 – Analisis estadistico descriptivo

T6 – Analisis estadistico cuantitativo

A3 – Filtros reductores de speckle

A4 – Evaluacion de filtros

Parte II: PolSARpro

T7 – Introduccion

PolSARpro Version 5.1

T8 – Lectura de datos

A5 – Datos polarimetricos: distribuciones

A6 – Descomposicion H-alfa

A7 – Clasificacion

T9 – Operaciones en PolSARpro: filtrado de speckle, descomposiciones polarimetricas, clasificacion

T10 – Exportacion

A8 – Tests de hipotesis y deteccion de cambios