GUIA RÁPIDO DO USUÁRIO R.

por Conrado Oliveira em um projeto do prof. Walmes M. Zeviani. Tradução não oficial de R Reference Card (Tom Short). UFPR 05-03-2013.

PEDINDO AJUDA

A maioria das funções do R tem um documento online.

help(tópico): documento do tópico.

?tópico: idem.

help.search("tópico"):procura a ajuda do sistema.

apropos("tópico"):encontra o nome de todos os objetos

help.start():inicia a ajuda na versão em HTML.

str("tópico"):mostra a estrutura do tópico no R.

summary(a):mostra o resumo de 'a'. Geralmente um resumo estatístico. Mas há diferentes operações para diferentes classes de 'a'.

ls():lista os objetos criados no console.

ls("padrão"):procura e lista objetos com determinado padrão.

ls.str(): Lista os objetos criados com suas respectivas estruturas.

dir("diretório"): lista arquivos de determinado diretório.

methods(a):mostra os métodos S3 do 'a'.

methods(class=class(a)):Lista todos os métodos capazes de lidar com objetos da classe 'a'.

IMPORTANDO E EXPORTANDO

load():carrega datasets(dados) escritos com *save.*

data(x):carrega dados específicos.

library(x):carrega um pacote de dados.

read.table(file): lê um arquivo no formato de tabela e cria a partir dele um *dataframe;*O separador padrão **sep=""** é qualquer espaço em branco; Use **header=TRUE** para ler a primeira linha como cabeçalho das colunas; Use **as.is=TRUE** para prevenir vetores de caracteres serem transformados em *factors;* Use **comment.char=""** para prevenir que # seja transformada em um inicializador de comentários; Use **skip=n** para pular n linhas antes de ler o arquivo; verifique na função *help* opções para nomear linhas, tratamentos nulos e outros.

read.csv(“nome_do_arquivo”,header=TRUE): idem, mas com opções padrões para leitura de arquivos delimitados por vírgulas.

read.delim(“nome_do_arquivo”,header=TRUE):idem ao **read.table(),**mas com opções padrões para a leitura de arquivos delimitados por *tab.*

read.fwf(arquivo,widths,header=FALSE,sep=””,as.is=FALSE): ler uma tabela com *width* com formatos fixos, dentro de um “*data.frame”. widths* é um vetor com números inteiros.

save(arquivo,...): salva o objeto especificado na plataforma binária independente XDR.

cat(...,arquivo=””,sep=””): mostra a saída no console dos argumentos depois de transformá-los em caracteres; **sep** será como os argumentos serão separados.

print(a,...):mostra a saída no console do objeto ‘a’.

format(x,...):formata o objeto em um *print* bonitinho.

write.table(x,file=””,row.names=TRUE,col.names=TRUE,sep=””): mostra a saída de ‘x’ depois de converter para um *data frame.* Caso *quote* seja *TRUE,* colunas de caracteres ou fatores estarão entre aspas (“”); **sep** é o campo separador; **eol** é como será a quebra de linha.; **na** é o que estará no lugar de valores não-encontrados; use **col.names=NA** para adicionar um espaço em branco no lugar do cabeçalho.

sink(arquivo): redireciona todas as saídas do console subsequentes para um arquivo.

A maioria dos funções que importam ou exportam tem um argumento **arquivo(file).** Isso pode ser frequentemente um nome em caractere ou uma conexão. **file=””** significa uma importação ou exportação padrão. O R faz conexões com arquivos, *pipes,* arquivos zipados e variáveis R.

No Windows, **description=’clipboard’** pode ser usado com uma conexão de arquivo. Para ler uma tabela do Excel, **use: x<-read.delim(“clipboard”)**.

Para escrever uma tabela em Excel, use: **write.table(x,”clipboard”,sep”\t”,col.names=NA).**

Para interação com banco de dados, veja pacotes **RODBC, DBI, RMySQL, RPgSQL** e **ROracle.**

Veja pacotes XML, hdf5, netCDF para ler outros formatos de arquivos.

CRIANDO DADOS

c(...):pode se entender c de combinar. É uma função genérica que combina os argumentos em um vetor. Escrevendo **c(...,recursive=TRUE)**, combina várias listas em um só vetor.

from:to: gera uma sequencia. Os dois pontos(:) tem a prioridade na operação. Exemplo: 1:4+1, a saída no console é “2,3,4,5”.

seq(from,to):também gera uma sequência. **by=**específica o incremento. **length=**específica o tamanho desejado da sequência.

seq(along=x):gera uma sequência 1,2,...até x. ideal para o *loop* **for.**

rep(x,times): repete x vezes; use **each=**para repetir cada elemento x vezes. Exemplo: rep(c(1,2,3)2) -> 1 2 3 1 2 3. Exemplo 2: rep(c(1,2,3)each=2) -> 1 1 2 2 3 3

data.frame(): cria um *data frame* com os argumentos, nomeados ou não. Exemplo: data.frame(v=1:4,ch=c(“a”,”b”, ”c”,”d”),n=10). Vetores menores são reciclados para o tamanho dos maiores.

list(): cria uma lista com os argumentos. Exemplo: list(a=c(1,2),b=”hi”,c=3i)

array(x,dim=): cria um banco de dados com x. **Dim=** dita quais dimensões o banco de dados terá. Elementos menores serão reciclados para o tamanho desejado.

matrix(x,nrow=,ncol=):cria uma matrix; com **nrow=**número de linhas e **ncol=**número de colunas. Elementos menores serão reciclados.

factor(x,levels=):transforma um vetor em um fator.

g1(n,k,length=n*k,labels=1:n):gera determinado número de níveis para um fator, de acordo com o padrão desejado. **k** é o número de níveis e n o número de repetições.

expand.grid():cria um *data frame* de acordo com todas as combinações desejadas de vetores ou fatores.

rbind(...):combina os argumentos por linhas em uma matriz, *data frame* e entre outros.

cbind(...):combina os argumentos por colunas em uma matriz, *data frame* e entre outros.

FRAGMENTANDO E EXTRAINDO DADOS

Vetores:

x[n]: seleciona o elemento presente no **n** índice do vetor x.

x[-n]: seleciona todos os elementos do vetor **x**, exceto o elemento **n.**

x[1:n]: seleciona os primeiros **n** elementos do vetor x.

x[-(1:n)]: seleciona todos os elementos do vetor x até **n,** exceto o primeiro.

x[c(1,3,5)]:seleciona os elementos do vetor x, pelo índice indicado.

x[“nome”]:seleciona o elemento do vetor x, pelo nome indicado.

x[x>3]:seleciona todos os elementos do vetor x maiores que 3.

x[x>3 & x<5]:seleciona todos os elementos do vetor x entre 3 e 5.

x[x %in%c(“a”,”and”,”the”)]: diz quais elementos dentro de **c** estão em x.

Listas:

x[n]: seleciona o elemento **n** da lista x.

x[[“nome”]] ou x$nome: seleciona o elemento da lista x com aquele nome.

Matrizes:

x[i,j]:seleciona o elemento da matriz x da linha i e da coluna j.

x[i,]:seleciona a linha i da matriz x.

x[,j]:seleciona a coluna j da matriz x.

x[,c(1,3): seleciona a coluna 1 e 3 da matriz x.

x[“name”,]:seleciona a linha com determinado nome da matriz x.

Data Frames (os indexadores das matrizes também funcionam nos *data frames*):

x[[“nome”]] ou x$nome: seleciona a coluna do *data frame* x com determinado nome.

CONVERTENDO VARIAVÉIS

as.array(x), as.data.frame(x), as.numeric(x), as.logical(x), as.convert(x), as.character(x): converte o a variável x para o tipo desejado. Para a lista completa, use **method(as)**.

INFORMAÇÕES SOBRE A VARIÁVEL

is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x): testa se a variável é de determinado tipo. Para a lista completa, use **method(is)**.

length(x): número de elementos de x.

dim(x): informa a dimensão de x.

dimnames(x): informa ou recupera os nomes das colunas e linhas de x.

nrow(x):informa o número de linhas de x. **NROW(x)** é a mesma função, porém trata um vetor com uma matriz de uma linha só.

ncol(x) ou NCOL(x): idem ao nrow, porém informa o número de colunas de x.

class(x): informa a classe de x. **class(x)<-"minha classe"** Atribui uma classe a x.

unclass(x):remove a classe atribuída de x.

attr(x,which): informa ao usuário qual é tamanho do atributo **which** da variável x. O usuário também pode informar ao R qual o tamanho do atributo que ele deseja.

attributes(objeto): Atribuir valores a um objeto.

SELEÇÃO E MANIPULAÇÃO DE DADOS

which.max(x): retorna o índice do maior elemento de x.

which.min(x):retorna o índice do menor elemento de x.

rev(x):reverte a ordem dos elementos de x.

sort(x): ordena de ordem crescente os elementos de x.

rev(sort(x)):ordena de ordem decrescente os elementos de x.

cut(x,breaks): divide x em intervalos (fatores); **breaks** é o tamanho do intervalo das divisões.

match(x,y): ele cruza x com y. Ele retorna um vetor do tamanho de x, informando o índice em y no qual se encontram os elementos de x. Caso contrário, ele informa NA.

which(x==y):ele informa o índice, com um vetor, dos elementos que tanto em x e y estão na mesma posição e são iguais.

choose(n, k): computa a combinação de **k** elementos entre **n** repetições. choose(n, k)=n!/[(n-k)!k!].

na.omit(x):omite as observações NA. Se x for uma matriz ou *data frame* o na.omit omite a linha onde os NA se encontram.

na.fail(x): retorna uma mensagem de erro caso x contenho ao menos um NA.

unique(x): caso x seja um vetor ou uma matriz, unique(x) retorna um objeto similar porém com os elementos omissos duplicados.

table(x): transforma os valores de x em uma tabela.

subset(x,...):seleciona elementos de x de acordo com um critério. Se x é um *data frame,*a opção **select** dita quais variáveis que devem ser mantidos e quais devem ser descartados.

sample(x, size): retira aleatoriamente um determinado número de elementos (**size)** de x, sem repetição. Para conter repetição, deve-se selecionar a opção **replace=TRUE** dentro dos argumentos.

prop.table(x,margin=):Cria uma tabela de proporção. **margin=**é a soma das proporções de cada linha.

MATEMÁTICA

Funções trigonométricas:

sin,cos,tan,asin,acos,atan,atan2...

Logaritmo e exponencial:

log, log10(logaritmo na base 10.), **exp**

max(x): maior valor encontrando em x.

min(x): menor valor encontrado em x.

range(x): informa o menor e o maior valor. **range(x)=c(min(x), max(x))**

sum(x): informa o somatório dos elementos de x.

diff(x):informa qual a variação em cada elemento de x.

prod(x): produto de todos os elementos de x.

mean(x): média dos elementos de x.

median(x): mediana de x.

quantile(x,probs=): informa os quartis de acordo com a probabilidade correspondente.

wighted.mean(x,w): média (de x) ponderada (por w).

rank(x): ranqueia os elementos de x do menor para o maior.

var(x) ou cov(x): variância dos elementos de x (calculado sobre n-1). Caso x seja uma matriz ou um *data frame,* é calculada a variância-covariância da matriz.

sd(x): desvio padrão de x.

var(x, y) or cor(x, y):covariância entre x e y. Se x e y forem matrizes ou *data frame*, calcula-se a covariância entre suas colunas.

round(x, n): arredonda os elementos de x para n casas decimais.

log(x, n): logaritmo x na base n.

scale(x): padroniza um vetor/matriz subtraindo os valores de sua média e dividindo pelo desvio padrão.

pmin(x, y, …) e pmax(x,y,...): Mínimos e máximos paralelos. As funções analisam dois ou mais vetores e informam qual é o mínimo ou máximo naquela posição. Vetores menores são reciclados até o tamanho do maior.

cumsum(x): responde com um vetor de tamanho igual à x a soma acumulada do respectivo elemento de x. Exemplo: x<-c(1,2,3), cum(x)=1,3,6.

cumprod(x): idem a cumsum, porém usando o produto e não a soma.

cummin(x): informa qual foi o mínimo até o respectivo elemento de x.

cummax(x): informa qual foi o máximo até o respectivo elemento de x.

union(x,y),intersect(x,y),setdiff(x,y),setequal(x,y),is.element(el,set) : funções do tipo *set*.

Re(x): informa qual a parte real de um número complexo.

Im(x): informa qual a parte imaginária de um número complexo.

Mod(x) ou abs(x): informa o valor absoluto de x, o seu módulo.

Arg(x): ângulo em radianos de um número complexo.

Conj(x): conjugado de um número complexo.

convolve(x,y): informa os diversos tipos de convolução entre duas sequências.

fft(x): Transformação Rápida de Fourier (Fast Fourier Transform – FFT) de um banco de dados (*array*).

mvfft(x): FFT de cada coluna de uma matriz.

filter(x,filter): aplica uma filtragem linear de uma série de dados constante ou para cada série separadamente de uma série multivariada.

Muitas das funções matemáticas do R tem um parâmetro base de **na.rm=FALSE**, ou seja, ela não remove os valores NA da operação, a não ser que o usuário indique o contrário, **na.rm=TRUE**.

MATRICES

t(x):matriz transposta.

diag(x) : diagonal principal de x.

a%*%b :multiplicação de matrizes a e b.

solve(a,b): informa a matriz x da seguinte equação: a%*%x=b

solve(a): matriz inversa de a.

rowsum(a) ou rowSums(a): soma dos elementos de cada linha da matriz a.

colsum(a) ou colSums(a): a soma dos elementos de cada coluna da matriz a.

rowMeans(a): tira a média de cada linha da matriz a.

colMeans(a): tira a média de cada coluna da matriz a.

PROCESSAMENTO DE DADOS AVANÇADOS

apply(x, INDEX, FUN=): obtêm um vetor, uma base de dados ou uma lista através do aplicando uma função com um índice (*index)* de x.

lapply(X,FUN): aplica a função (*fun*) para cada elemento de uma lista x.

tapply(X, INDEX,FUN): aplica a função (*fun*) para cada célula do banco de dados x, com um índice (*index*).

by(data, INDEX, FUN): aplica a função(*fun)* em uma *data frame (data)* dado um índice(*index).*

merge(a,b): junta dois *data frames* segundo o critério de colunas ou linhas em comum.

xtabs(a b,data=x): Cria uma tabela de contingência, através do cruzamento de fatores.

aggregate(x,by, FUN): divide um *data frame* x em subconjuntos, cria um sumário estatístico para cada um deles e retorna o resultado na forma mais conveniente. *By* é uma lista de grupos de elementos, cada um deles do tamanho de x.

stack(x,...): transforma dados que estão em diversas colunas em um *data frame* único com diversas colunas.

unstack(x,...): função inversa de *stack().*

reshape(x,...) : refaz um *data frame* que está em um formato 'largo'(*wide format)* com mensurações repetidas em diversas colunas no mesmo registro e no formato longo (*long format*) com mensurações repetidas em registros separados. Use **(direction='wide)** ou **(direction=long)**.

STRINGS *

**string* é uma série de caracteres.

paste(...): concatena vetores depois de transformá-los em caracteres; **sep=**será como as séries serão separadas (a definição padrão é um espaço em branco.);**collapse=**é uma opção para separar os resultados das séries.

substr(x,start,stop): Extrai ou substitui um string em um vetor de caracteres.

strsplit(x,split): divide x em substrings de acordo com o parâmetro *split.*

grep(pattern, x): procura por padrões (*patter*) em x. veja mais em **? regex.**

gsub(pattern, replacement, x): substitui os padrões encontrados por uma expressão pré-determinada. A função **sub()** tem o mesmo objetivo, porém só substitui uma única vez, a primeira.

tolower(x): reescreve a *string* x em letras minúsculas.

toupper(x):reescreve a *string* x em letras maiúsculas.

match(x, table): retorna um vetor com a posição que está em *table* do primeiro encontro dos elementos de x.

x %in%table: idem ao anterior, porém retorna um vetor lógico.

pmatch(x,table): encontra combinações parciais dos elementos de x entre os dos elementos de *table*.

nchar(x):número de caracteres de x.

DATA E HORA.

A classe *Date* tem a data mas sem o horário. Já *POSIXct* tem horário e a data, incluindo o fuso horário. Comparações **(*e.g.*>)**, **seq()** e **difftime()** são úteis. *Date* também permite somar e subtrair. *?DataTimeClasses* traz mais informações.

as.Date(s) e as.POISXct(s): converte para a respectiva classe. **format(dt)** para um *string*. O *string* padrão é “2013-03-21”. Isso é aceito como um segundo argumento para especificar o formato de conversão. Outros formatos comuns são:

%a, %A: Data abreviada, mas com o dia da semana por extenso.

%b,%B: Data abreviada, mas com o mês por extenso.

%d: Dia do mês(01-31).

%H: Hora(00-23).

%I: Hora(01-12).

%j:Dia do ano(001-366).

%m:Mês(01-12).

%M:Minuto(00-59).

%p: indica AM/PM.

%s: Segundos em números decimais (0-61).

%U:Semana do ano(00-53). Considera domingo como o primeiro dia da semana.

$w: Dia da semana.(0-6, sendo 0 o domingo).

$W: Semana do ano(00-53). Considera a segunda-feira como o primeiro dia da semana.

%y: Ano, sem considerar o século (00-99). Não use.

%Y: Ano, com o século.

%z(saída somente): Fuso horário a partir de Greenwich; -0800 são 8 horas a leste de Greenwich.

% **Z(saída somente):** Fuso horário no formato de *string.*

PLOTANDO GRÁFICOS

plot(a): cria um gráfico com os valores de a no eixo y e no eixo x a sua respectiva posição.

plot(a,b): cria um gráfico em que os valores de a estarão no eixo x e os valores de b no eixo y.

hist(x): cria um histograma de frequência de x.

barplot(x): cria um histograma com os valores de x; use **horiz=TRUE** para criar barras horizontais .

dotchart(x): caso x seja um *data frame*, cria um 'gráfico de pontos' (*dot plot)*; dispõe os pontos linha por linha e coluna por coluna.

pie(x): cria um gráfico de setores (o famoso gráfico de pizza).

boxplot(x): cria um gráfico do tipo box-plot.

sunflowerplot(x, y): idem ao **plot()** mas os pontos com coordenadas similares são representados por um desenho de flor, e o número de pétalas representa o número de pontos.

stripplot(x): plota os valores de x em um linha (uma alternativa para o **boxplot()** para pequenas amostras).

coplot(x~y | z): gráfico que analisa a covariância de x e y para cada valor ou intervalo dos valores de z.

interaction.plot(f1,f2,y): se f1 e f2 são fatores, plota a media de y (no eixo y) comos respectivo valor de f1(no eixo x) e f2 (curvas diferenciais); há a possibilidade de mudar a opção **fun** que permite escolher outra função estatística, como a média, por exemplo.

matplot(x,y): plota a covariancia da primeira coluna de x *versus*a primeira coluna de y, depois a segunda coluna de x com a segunda coluna de y e assim sucessivamente.

fourfoldplot(x): Cria uma tabela de contigência que permite a visualização da relação entre duas variáveis dicotômicas em uma ou várias populações. x deve ser um *array* de **dim=c(2,2,k),** ou uma matriz de **dim=c(2,2)** caso k seja igual a 1.

assocplot(x): Cria um gráfico do tipo Cohen-Friendly, cujo demonstra as diferenças em relação a independência das linhas e colunas em uma tabela de contingência.

mosaicplot(x): cria um gráfico do tipo mosaico com os valores residuais de uma regressão linear logarítmica em uma tabela de contigência.

pairs(x): caso x seja uma matriz ou um *data frame*, visualiza todas as possíveis bivariações entre as colunas de x.

plot.ts(): caso x seja um objeto do tipo "*ts*", essa função plota x com seu respectivo tempo, porém x deve ser multivariado, mas suas séries devem ter a mesma frequência e tempo.

qqnorm(x): plota os quantis de x usando como base a curva normal.

qqplot(x, y): plota os quantis de y respectivamente com seus pares em x.

contour(x, y, z): cria um gráfico de contornos (dados são interpolados para criar o desenho), x e y devem ser vetores e z deve ser uma matriz de **dim=(c(lenght(x), length(y))**. x e y podem estar omitidos.

filled.contour(x,y,z): idem ao anterior, porém a área entre os contornos são coloridos e há legenda das cores.

image(x,y,z): idem ao **contour(),** porém com cores.

persp(x,y,z): idem ao **contour()** porém com perspectiva.

stars(x): caso x seja uma matriz ou um *data frame*, desenha-se um gráfico com segmentos ou uma estrela na qual cada fila de x é representada e as colunas são do tamanho dos segmentos.

symbols(x, y,...): desenha de acordo com as coordenadas de x e y símbolos (círculos, quadrados, retângulos, estrelas, termômetros ou 'boxplot') e seus respectivos tamanhos e cores são dados por argumentos suplementares.

termplot(mod.obj): plota os efeitos (parciais) do modelo de regressão (mod.obj).

Os seguintes parâmetros são comuns a vários gráficos:

add=FALSE: Caso seja TRUE, o parâmetro sobrepõe uma plotagem sobre uma já existente.

axes=TRUE: Caso seja FALSE, o parâmetro não desenha os eixos e a caixa (bloxplot).

type='p': específica o tipo de plotagem. **'p'**=pontos; **'l'**=linhas; **'b'**=pontos conectados por linhas; **'o'**=idem ao anterior, mas os pontos estão sobrepostos a linha;**'h'**=linhas verticais;**'s'**=degrau, na qual o dado é representado no topo da linha vertical; **'S'**=idem ao **'s'**, porém o dado é representado na base da linha vertical.

xlim, ylim=específica os limites inferiores e superiores do eixos. Exemplo: **xlim=c(1,10), ylim=range(x).**

main=atribui um título ao gráfico.

sub=incluí um subtítulo.

OUTROS COMANDOS GRÁFICOS

points(x, y): adiciona pontos no gráfico (a opção **type=**pode ser usada)

lines(x, y): adiciona linhas no gráfico.

text(x, y, labels,...): adiciona um texto dado no campo **labels**, nas coordenadas (x,y); um uso comum é **plot(x, y, type='n'); text(x,y nomes).**

mtext(text, side=3, line=0): adiciona um texto dado no campo **text,** na margem especificada em **side** (ver **axis(),** abaixo); **line** específica a linha que partirá o texto.

segments(x0, y0, x1, y1, angle=30, code=2): desenha segmentos de linha a partir do ponto(x0,y0), até (x1,y1).

arrows(x0, y0, x1, y1, angle=30, code=2): desenha flechas no ponto(x0,y0) caso o **code=1;** se **code=2,** as flechas irão no ponto(x1,y1); e as flechas irão em ambas caso **code=3.**

abline(a,b): desenha um linha com inclinação b, no intercepto a.

abline(h=y): desenha uma linha horizontal no eixo y.

abline(v=x): desenha uma linha vertical no eixo x.

abline(lm.obj): desenha a linha de regressão dada por **lm.obj.**

rect(x1, y1, x2, y2): desenha um retângulo no qual os limites da esquerda, da direita, inferior e superior são, respectivamente, x1, y1, x2, y2.

polygon(x, y): desenha um polígono ligando os pontos dados em x e y**.**

legend(x, y ,legend): adiciona uma legenda no ponto(x,y) com os símbolos dados no campo **legend.**

title(): adiciona um título. Tem a opcção de adicionar um subtítulo**.**

axis(side, vect): adiciona um eixo inferior (**side=1**), na esquerda (**side=2**), superior (**side=3**) ou na direita (**side=4**); **vect**, é opcional, e direciona onde na abscissa (ou ordenada) partirá o eixo.

rug(x): desenha pequenas linhas representando os dados x no eixo x.

locator(n, type='n',...): retorna as coordenadas correspondentes pedidas pelo usuário ao clicar (n vezes) no gráfico. Também desenha símbolos (**type='p'**) ou linhas (**type=1**) respeitando os parâmetros do gráfico. Por padrão, **type='n'.**

PARÂMETROS GRÁFICOS

As funções a seguir podem ser configuradas a partir de **par**(...); a maioria pode ser usada como parâmetro gráfico.

adj: controla a formatação do texto (0 – formatação à esquerda; 0.5 centrada; 1 à direita).

bg: específica a cor de fundo do gráfico (ex.: **bg="red", bg="blue",...).** Há mais de 657 cores disponíveis. Para ver a lista, função **color()**.

bty: define o tipo de caixa desenhada em volta da plotagem. Os valores permitidos são: "o", "1", "7", "c","u" ou"]". O desenho se parece com o respectivo caractere. se bty='n', nenhuma caixa será desenhada.

cex: um valor controla o tamanho dos textos e símbolos de acordo com seus respectivos padrões. Os parâmetros a seguir tem o mesmo controle para os números nos eixos, **cex.axis,** o rótulo do eixo, **cex.lab,**o título, **cex.main,** e o subtítulo, **cex.sub.**

col: controla a cor dos símbolos e das linhas. Usar o nome das cores: **"red"**(vermelho)**, "blue"** (azul) e etc. Veja em **colors()** ou em "#RRGGBB"**,** a lista completa; **rgb()**, **hsv()**, **gray()**, e **rainbow()** ; para as funções **cex,** há: **col.axis, col.lab, col.main, col.sub.**

font: número inteiro que controla a fonte. 1-normal, 2-*itálico*, 3-**negrito**, 4-***negrito* e *itálico***. Para as funções **cex,** há: **font.axis, font.lab, font.main, font.sub.**

las: número inteiro que controla a orientação dos rótulos dos eixos. 0-paralelo ao eixo, 1-horizontal, 2-perpendicular ao eixo, 3- vertical.

lty: controla o tipo de linha, podendo ser um *string* ou um número inteiro. 1 ou 'solid': sólido, 2 ou 'dashed': travessão, 3 ou 'dotted': pontilhado, 4 ou 'dotdash': linha pontilhada intercalada por pontos, 5 ou 'longdash':travessão alongado, 6 ou 'twodash': travessão duplo, ou um string com até oito caracteres (entre "0" e "9"), que específica, alternativamente, o comprimento da linha, em pontos ou *pixels*, o desenho dos elementos e do espaço em branco. Por exemplo, **lty="44",** terá o mesmo efeito que "**lty"=2.**

lwd: número que controla a largura da linha, padrão igual a 1.

mar: um vetor com 4 números que controla o espaço entre os eixos e as bordas do gráfico **c(bottom, left, top, right),** o padrão para os valores são c(5.1, 4.1, 4.1, 2.1).

mfcol: um vetor com dimensões **c(nr,nc),** no qual, nr=número de linhas e nc=número de colunas, que projeta no gráfico de colunas uma grade de acordo com os parâmetros.

mfrow: idem, mas o gráfico é plotado em linhas.

pch: controla o tipo de símbolo, sendo um inteiro entre 1 e 25, ou um caractere entre aspas.

(faltou símbolos)

ps: inteiro que controla o tamanho em pontos dos textos e símbolos.

pty: caractere que específica o tipo da região a ser plotada: **"s"=**quadrado, "m"=máxima".

tck: valor que específica o comprimento do símbolo de pontuação nos eixos como uma fração da menor largura ou altura da plotagem. Se **tck=1** uma grade é desenhada.

tcl: valor que específica a comprimento dos símbolos de pontuação nos eixos como uma fração da altura das linhas de textos (padrão, **tcl=-0.5).**

xaxt: Se **xaxt='n'**, o eixo x é definido, porém não é desenhado (útil em conjunção do tipo **axis(side=1,...)).**

yaxt: Se **yaxt='n',** o eixo é definido, porém não desenhado (útil em conjunção do tipo **axis(side=2,...)).**

GRÁFICOS LATTICE

xyplot(y~x): plotagem bivariável (com muitas funcionalidades).

barchart(y~x): histograma com os valores de y em relação aos valores de x.

dotplot(y~x): cria um gráfico de pontos, dispõe os valores linha por linha e coluna por coluna.

densityplot(~x): histograma de frequências de x.

histogram(~x): histograma dos valores de x.

bwplot(y, x): *Box-plot*. “A caixa com bigodes”.

qqmath(~x): quantis de x em relação os valores esperados da distribuição teórica.

stripplot(y~x): plotagem com dimensão única, sendo x um número e y um fator.

qq(y~x): quantis para comparar duas distribuições, sendo o x obrigatoriamente um número e y podendo ser um número, um caractere ou um fator, porém deve ter dois níveis.

splom(~x): matriz com plotagem bivariada.

parallel(~x): plotagem com as coordenadas paralelas.

levelplot(z~x*y | g1*g2): plotagem colorida com os valores de z nas coordenadas dadas nos campos y e x (x, y e z devem ter o mesmo comprimento).

wireframe(z~x*y | g1*g2): plotagem em 3d.

cloud(z~x*y | g1*g2): plotagem em 3d disperso.

Nas fórmulas do sistema Lattice, **y x | g1*g2** tem combinações para que tanto g1 e g2 sejam plotados em painéis separados. As funções Lattice usam a maioria dos argumentos dos gráficos bases do R. Além disso, o Lattice possui a função **data=**o *data frame* para as fórmulas das variáveis, e **substet=**para subconjuntos. Use **panel=** para definir o painel de funções padrão (ver **apropos(“panel”)** e **?lines).** As funções Lattice retornam como objetos e não tem o gráfico reproduzido automaticamente. Assim, deve-se usar **print(xyplot(...)** dentro das funções para que seja reproduzido automaticamente os gráficos. Use **lattice.theme** e **lset** para mudar as configurações do Lattice.

OTIMIZANDO E AJUSTANDO MODELOS.

optim(par, fn, method=c(“Nelder-Mead”, “BFGS”, “CG”, “L-BFGS-B, “SANN”): tem como finalidade otimizar. Sendo **par** o valor inicial e **fn** a função para a otimização (geralmente para minimizar).

nlm(f, p): utiliza o algoritmo Newton-type, começando com o valor p, para minimizar a função f.

lm(fórmula): ajusta os modelos lineares. **fórmula** geralmente se compõe na forma **termA+termB+...;** use **I(x*y) + I (x+y)** para termos de base de modelos não lineares.

glm (fórmula, family=) ajusta modelos lineares generalizados, especificando de acordo com o símbolo descrito do preditor linear e da descrição do erro da distribuição. **family** é a descrição do erro da distribuição e liga a função a ser usada no modelo. Ver ?**family.**

nls(fórmula): estimativa do modelo não linear dos mínimos quadrados a partir de modelos não lineares.

approx(x, y=): interpolação linear dada por pontos; x pode ter uma estrutura de plotagem igual a xy.

spline(x, y=): interpolação spline cúbica.

loess(fórmula): ajusta uma superfície polinomial usando técnicas locais.

A maioria das fórmulas base de função de modelagem têm vários argumentos em comum; **data=**o *data frame* da fórmula das variáveis; **subset=** um subconjunto de variáveis para adequação; **na.action=** ação para valores indisponíveis; "**na**.**fail**", "**na**.**omit"** ou uma função. A seguir, algumas funções para adequação de modelos:

predict(fit,...): previsões a partir do **fit** baseado nos dados escritos.

df.residual(fit): retorna o número residual de graus de liberdade.

coef(fit): retorna os coeficientes estimados (as vezes, com um erro padrão).

residuals(fit): retorna os valores residuais.

deviance(fit): retorna o desvio.

fitted(fit): retorna os valores já ajustados.

logLink(fit): calcula o logaritmo da probabilidade e o número de parâmetros.

AIC(fit): calcula o teste de verificação de informação de Akaike (AIC).

ESTATÍSTICAS

aov(formula): análise de variância do modelo.

anova(fit,...): tabella de análise de variância (ou desvios) para um ou mais modelos ajustados.

density(x): centro de densidade estimada de x.

binom.test(), pairwise.t.test(), power.t.test(), prop.test(), t.test(), ...: usar **help.search("test")**

DISTRIBUIÇÕES

RNORM(n, mean=0, sd=1): Gaussiana (normal).

rexp(n, rate=1): exponencial.

rgamma(n, shape, scale=1): gamma

rpois(n, lambda): Poisson.

rweibull(n, shape, scale=1): Weibull

rcauchy(n, location=0, scale=1): Cauchy

rbeta(n, shape1, shape2): beta.

rt(n, df): 'Student' (*t*).

rf(n, df1, df2): Fisher-Snedecor (F) (x^2).

rchisq(n, df): Pearson.

rbinom(n,size, prob): binomial.

rgeon(n, prob): geométrica.

rhyper(nn, m, n, k): hipergeométrica.

rlogis(n, location=0, scale=1): logistical.

rlnorm(n, meanlog=0, sdlog=1): lognormal.

rnbinom(n, size, prob): binomial negativa.

runif(n, min=0, max=1): uniforme.

rwilcox(nn, m, n): rsignrank(nn, n) estatística de Wilcoxon.

Todas essas funções podem ser substituídas pelas letras **d, p** ou **q** para mudar, respectivamente, para a função de densidade da probabilidade (**dfunc(x,...)),** para a probabilidade acumulada (pfun(x,...)), e para o valor do **quantil (qfunc(p,...)**, com 0<p<1);

PROGRAMANDO

function(arglist) expr

return(valor)

if(condição) expr

if(condição) cons.expr else alt.expr

for (var in seq) expr

while(condição) expr

repeat expr

break: para o condicional

next

Use chaves {} para abrir e fechar a função.

ifelse(test, yes, no): um valor com o mesmo formato de **test,** com os elementos sim ou não (**yes, no).**

do.call(nome da função,argumentos) executa a função a partir dos argumentos dados.