###Curso de introdução ao R - v 1.2###

###Pavel Dodonov - pdodonov@gmail.com (doutorando em Ecologia e Recursos Naturais pela UFSCar)###

###Ministrado na UFSCar de São Carlos, 18-19/outubro/2014###

###Este curso foi escrito em forma de um script de R; junto com ele deve haver o arquivo dadosfogo.txt, usado nos exemplos. Este arquivo teoricamente está disponível em http://textuploader.com/af734

###Recomendo ir digitando o código (não copiando e colando, digitando) e buscando entender o que ele faz. Algumas coisas não são intuitivas porque este material foi preparado como material de suporte para uma aula presencial, não uma apostila independente. Mas pode ser que dê certo. :)

###Texto precedido por # (um ou mais) é entendido pelo R como comentários, não como linha de código.

```
#Facilita a vida se abrirmos um script!
#File, new script
#E podemos salvar o script, o que facilita mais ainda a vida!

#Para rodar uma linha do script: ctrl + R (no Mac - comand + R)

#Parte 1 - Números, funções e objetos
#Elementos abordados:+ / c() , = <- -> : rnorm() runif() rm

# (sustenido/jogo-da-velha) é o símbolo usado para fazer anotações no código.
Se escrever sem este símbolo, acontece isso (erro)

#Com o símbolo, não acontece nada

1
2
5

#Podemos fazer operações básicas com os números

1+5
1/5

#E se queremos trabalhar com mais de um número simultaneamente?
```

```
1 2 4
1, 2, 4

#Não funciona!

c(1,2,4)

#c() é uma função. O que está dentro dos parênteses são os argumentos desta função.
#No caso, a função está combinando os números 1, 2, 4 para que possamos trabalhar com eles ao mesmo tempo

c(1,2,4)/5

c(1,2,4)+c(4,5,7)

#Mas é claro que não vamos ficar escrevendo esses números toda vez que formos fazer alguma coisa com eles.
#Precisamos criar um objeto

#Um objeto pode ser um conjunto de números, um texto, um conjunto de textos, os resultados de uma análise...
#Enfim, um objeto pode ser essencialmente qualquer coisa

#E um objeto precisa de um nome!
#Recomendações pessoas de nomes:
    #Se queremos fazer algo bem rápido: a, b, d, e, f... Não c porque c é uma função!
    #Para análises mais completas: ter um sistema que te permita sempre achar o objeto mais importante
    #Exemplos: dados, dados.Ilha, modelo.Ilha, modelo1.Ilha...
    #Para objetos temporários, que não serão usados novamente: temp, temporario, foo, bar, foobar...

a=c(1,2,4)
b=c(4,5,7)

    #Evitar dar nomes que coincidam com nomes de funções ou objetos já existentes!
    #Na dúvida, teste
pi
data
```

```
Pi        #Pi é um nome que podemos usar sem sobrepôr a outro objeto ou função.

#Mas e se fizermos algo errado?

a=c(1,2,4)
b=c(4,5,7)
c=c(8,9,11)

c(1,2,6)          #continua funcionando - não sobrepôs a função
                  #mas agora temos uma função e um objeto que têm o mesmo nome, e isso
pode confundir

pi=c(1,4,5)       #Agora perdemos o valor do pi, que o R tinha salvo

#Solução:
rm(c)
rm(pi)

#a função rm remove os objetos que criamos.

#Diferentes forma de criar objetos
a=c(1,2,4)
a = c(1,2,4)
a = c ( 1 , 2 , 4 )
a <- c(1, 2, 4)
c(1, 2, 4) -> a

c(1, 2
,
4) ->
a

#É claro que existem limites...

a=
c
(
1,2,
4)
```

```
a

a=
c(
1,2,
4)


#Então não saiam quebrando linhas por aí sem motivo :)


#Espaços e quebras de linha não importam
#Algumas pessoas gostam de usar =, outras de usar <-... É questão de estilo.

1:10

a=1:10
b=11:20

# : é muito útil para criar sequências de números.

d=seq(0, 10, by=0.1)

d

# seq() permite criar sequências com espaçamento maior ou menor que 1.

#Funções têm argumentos, que definem o que exatamente a função vai fazer

rnorm(n=15, mean=1, sd=1)
rnorm(15, 1, 1)
#Se não damos os nomes dos argumentos, eles são usados na mesma ordem que aparecem na definição da função
rnorm

#Se não fornecemos um argumento, é usado o default - no caso 0 e 1
rnorm(15)

a=rnorm(15, 1,1)
b=runif(n=15, min=1, max=2)
```

```
a-b

#Se não sabemos o que uma função faz, podemos perguntar ao R!
?rnorm

#Embora às vezes o help não ajuda tanto assim...

#Às vezes perdemos conta dos objetos que criamos

ls()

#Para evitar que objetos de sessões passadas apareçam na nossa sessão atual e nos confundam mais ainda:

rm(list=ls())

ls()

#Pronto, não temos mais objetos.

###Fim da parte 1###

###Parte 2 - inserindo dados###
getwd()
setwd(F:\Pavel\2-Ensino\MINICURSOS\R-RioClaro-2014)
setwd("F:\Pavel\2-Ensino\MINICURSOS\R-RioClaro-2014")
setwd("F:/Pavel/2-Ensino/MINICURSOS/R-RioClaro-2014")
setwd("F:\\Pavel\\2-Ensino\\MINICURSOS\\R-RioClaro-2014")
getwd()

setwd(choose.dir())


read.table("dadosfogo.txt", sep="", dec=".")
read.table("dadosfogo.txt", sep="", dec=".", skip=3)
read.table("dadosfogo.txt", sep="", dec=".", skip=3, header=T)

read.table(file.choose(), sep="", dec=".", skip=3, header=T)
```

```
read.table("clipboard", sep="\t", dec=".", header=T)
#No mac: read.table(pipe("pbpaste"))

fogo=read.table("dadosfogo.txt", sep="", dec=".", skip=3, header=T)

fogo

fogo.errado=read.table("dadosfogo.txt", sep="", dec=".", skip=3)

fogo.errado

###Fim da parte 2###

###Parte 3 - verificando a estrutura dos dados###
str(fogo)
str(fogo.errado)

rm(fogo.errado)

head(fogo, n=5)
tail(fogo, n=5)

ncol(fogo)
nrow(fogo)
dim(fogo)

names(fogo)

names(fogo) = c("Transecto", "Parcela", "x", "y", "Severidade", "Tipo", "Nind")
colnames(fogo) = c("Transecto", "Parcela", "x", "y", "Severidade", "Tipo", "Nind")

#Como é um data.frame, tanto names() quanto colnames() pode ser usado.
#No caso de listas, apenas names() funciona para isso
#No caso de matrizes, apenas colnames()

###Parte 4 - tipos de objetos###
fogo$Tipo=as.character(fogo$Tipo)
str(fogo)
fogo$Tipo=as.factor(fogo$Tipo)
str(fogo)
```

```
fogo$Tipo=ordered(fogo$Tipo, levels=c("Pl", "Br", "AdBr", "Ad", "M"))
str(fogo)
fogo$Tipo=factor(fogo$Tipo, ordered=F)
str(fogo)



#O objeto fogo atualmente é um data.frame
#data.frames são conjuntos de vetores, todos com o mesmo comprimento, podendo serem de diferentes tipos

#Outro tipo de objeto que pode conter diferentes tipos de variáveis e coisas são as listas

a.lista = list(1:10, list(a=c(2,5), b=3:7), "oisouumapalavraqualquer")

a.lista

str(a.lista)
length(a.lista)

#Se quisermos, podemos transformar essa lista em um único vetor

a.vetor = unlist(a.lista)

a.vetor

length(a.vetor)
str(a.vetor)

#como tem um caracter na lista, ela se transformou inteira em caracteres

as.numeric(a.vetor)

#NAs introduced by coercion: a palavra "oisouumapalavraqualquer" não pode ser transformada em número

#NA: not available; NaN: Not a Number - por exemplo, divisão de zero por zero

5/0
0/0
```

```
log(0)
log(-1)

# Inf: infinito.

#Digamos que queremos um objeto com os dados separados por transecto...
#Podemos fazer uma matriz!

a=1:10
matrix(a, ncol=2)

#Percebam que ele preencheu por colunas
matrix(a, ncol=2, byrow=T)

#Percebam que agora ele preencheu por linhas!

fogo.sever.tr=matrix(fogo$Severidade,ncol=5)

str(fogo.sever.tr)

fogo.sever.tr

head(fogo.sever.tr)

colnames(fogo.sever.tr) = c("T1", "T2", "T3", "T4", "T5")
rownames(fogo.sever.tr) = 1:100

str(fogo.sever.tr)

dimnames(fogo.sever.tr)=list(Parcela=c(1:100), Transecto=c("T1", "T2", "T3", "T4",
"T5") )

str(fogo.sever.tr)

fogo.sever.tr



#Mas percebam que está tudo repetido? É porque a severidade ela mesma está repetida...
```

```
###Fim da parte 4###

###Parte 5 - indexação###

#É melhor pegar apenas os referente apenas um conjunto de valores de fireSeverity.

#Como cada variável tem 100 valores, podemos pegar simplesmente os 100 primeiros valores do vetor FireSeverity

fogo$Severidade[1:100]

fogo.sever.tr=matrix(fogo$Severidade[1:100],ncol=5)

fogo.sever.tr

#E se quisermos pegar as 100 primeiras linhas, mas para todas as variáveis?

fogo[1:100]

fogo[1:100,]

head(fogo)

#Podemos, por exemplo, querer todas as informações sobre os indivíduos adultos, sem a coluna Tipo

fogo.Ad=fogo[1:100,c(1:5,7)]
fogo.Ad=fogo[1:100,-6]

colnames(fogo)

fogo.Ad=fogo[1:100,c("Transecto", "Parcela", "x", "y", "Severidade", "Nind")]

str(fogo.Ad)

fogo.Ad=as.matrix(fogo.Ad)

str(fogo.Ad)

fogo.Ad[1:10,c("x", "y", "Nind")]
```

```
fogo.Ad[1:10]

#E se quisermos, por exemplo, pegar apenas as linhas ímpares?

#O R tem um comportamento interessante com vetores de verdadeiro/falso

a=1:100
a[c(T,F)]

a=runif(100,0,100)

a<50

a[a<50]

#Se quisermos saber quais os números que satisfazem a equação:

1:100[a<50]
(1:100)[a<50]

(1:length(a)) [a<50]

which(a<50)

a[a<50]

#No nosso objeto fogo... Podemos, por exemplo, querer as linhas correspondtes ao tipo
Br

fogo.Br=fogo[fogo$Tipo=="Br",]

fogo$Tipo=="Br"

#Seleciona apenas as linhas correspondentes a TRUE


fogo.Br=fogo[which(fogo$Tipo=="Br"),]

which(fogo$Tipo=="Br")
```

```
#Seleciona as linhas correspondentes a TRUE, pelos números

fogo.Br=subset(fogo, Tipo=="Br")

#E se quisermos, por exemplo, todos os indivíduos vivos, ou seja, exceto o tipo M?

fogo.Vivo=subset(fogo, Tipo!="M")

fogo.Vivo=fogo[fogo$Tipo!="M",]

#E quanto a listas?

fogo$Nind

fogo[["Nind"]]

fogo[[7]]


a.lista

a.lista[[2]]

a.lista[[2]]$a

a.lista[[2]][[1]]

#E se quisermos pegar mais de duas coisas da lista?

a.lista[[c(1,2)]]

a.lista[c(1,2)]

#Ou seja, tanto [[ ]] quanto [ ] se aplicam a listas e data.frames.

#Finalmente, pudemos mudar apenas parte dos valores...

#Por exemplo, decidimos que parcelas com FireSeverity < 0.4 têm FireSeverity = 0
```

```
fogo.mudado=fogo

fogo.mudado$FireSeverity[fogo.mudado$FireSeverity<0.4]=0



#Operações matemáticas e estatística descritiva

5+2
5*2
5/2
5^2
sqrt(5)
5%%2
5%/%2

adultos=subset(fogo,Tipo=="Ad")$Nind
brotos=subset(fogo, Tipo=="Br")$Nind

sum(adultos)
mean(adultos)
var(adultos)
sd(adultos)

mean(fogo.sever.tr)

mean(fogo.sever.tr, na.rm=T)


adultos+brotos

adultos>brotos

#E se quisermos ver quantas parcelas têm mais adultos que brotos?

length(which(adultos>brotos))

sum(adultos>brotos)
```

#Podemos também querer ver onde temos adultos, brotos, adultos e brotos, ou adultos ou brotos

```
adultos>0
brotos>0

adultos>0 & brotos>0

adultos&0 | brotos>0


#E se quisermos ver de forma fácil o que é NA em um objeto?

fogo.sever.tr==NA
is.na(fogo.sever.tr)
sum(is.na(fogo.sever.tr))

#A somatória funciona porque um T ou TRUE é igual a 1 e um F ou FALSE é igual 0.

### Fim da parte 6!###

###Parte 7 - Gráficos simples###

#Esta é apenas uma introdução básica, bem básica mesma, sobre gráficos.
#Se tudo der certo pretendo ministrar um curso só sobre gráficos no futuro...


#eixo x é a ordem ao longo do objeto

plot(brotos~adultos)

#quantitade de brotos em função da quantidade de adultos

#E se quisermos ver a relação de número de brotos com a severidade do fogo?

#Vamos fazer um novo objeto antes

ls()

str(fogo.Br)
```

```
plot(fogo.Br$Nind ~ fogo.Br$Severidade)

plot(Nind ~ Severidade, data=fogo.Br)

points(Nind~Severidade, data=fogo.Ad)


plot(Nind ~ Severidade, data=fogo.Br)

points(Nind~Severidade, data=fogo.Ad, col="red")

#Podemos querer olhar como a intensidade do fogo muda ao longo de um transecto

ls()

str(fogo.sever.tr)

plot(fogo.sever.tr[,1])

plot(fogo.sever.tr[,1], type="l")

points(fogo.sever.tr[,2], type="l", col="red")
points(fogo.sever.tr[,3], type="l", col="green")
points(fogo.sever.tr[,4], type="l", col="blue")
points(fogo.sever.tr[,5], type="l", col="orange")

#Para ver diferentes coisas que você pode mudar no gráfico:

?par

#E se quiserem ter dois gráficos juntos?

par(mfrow=c(2,1) )

plot(fogo.sever.tr[,1], type="l")

#Mas agora podemos fazer uma outra função pra fazer as linhas

lines(fogo.sever.tr[,2], col="red")
```

```
lines(fogo.sever.tr[,3], col="green")
lines(fogo.sever.tr[,4], col="blue")
lines(fogo.sever.tr[,5], col="orange")


plot(brotos~adultos)

###Fim da parte 7###

###Parte 8 - modelos lineares###

#E se quisermos ver se existe uma relação nisso tudo?

mod.br.ad = lm(brotos~adultos)

summary(mod.br.ad)


plot(brotos~adultos)

new.data=min(adultos):max(adultos)

predicted.br.ad=predict(mod.br.ad,list(adultos=new.data))

lines(predicted.br.ad~new.data)

predicted.br.ad=predict(mod.br.ad)

lines(predicted.br.ad~adultos, col="red")

# A reta ficou irregular porque os originais não estão distrubuídos de ordem crescente e são menos regulares.

###Fim da parte 8###

###Parte 9 - Help do R###

?lm
??"linear model"
```

?%%
?"%%"
?"["

#Algumas coisas precisar ser usadas entre aspas

help(lm)
help.search("linear model")

help(package="stats")

###Fim da parte 9###

###Parte 10 - Instalando e usando pacotes###

#Pacotes são conjuntos de funções e objetos que não vêm com a instalação base do R
#Existem MUITOS pacotes. Mas normalmente usamos apenas alguns poucos.

#Podem ser baixados da internet ou instalados de zip files

Se quisermos instalar dois pacotes ao mesmo tempo, da internet:
install.packages(c("mgcv", "nlme"))

library(mgcv)
require(mgcv)

detach("package:mgcv")

#Útil quando se trabalha com diferentes pacotes que têm funções do mesmo nome.

help(package="mgcv")

###Fim da parte 10###

###Exercício para casa (ou não)!###

library(mgcv)

```
fogo.Ad=subset(fogo,Tipo=="Ad")

modelo=gamm(Nind~s(Severidade, fx=F, k=-1), data=fogo.Ad,
cor=corSpher(form=~x+y, nugget=F), method="REML")

#Usando summary e outras coisas de indexação que aprenderam hoje, encontrar o valor de significância do s(Severidade)
#Escrever um código para extrair este valor automaticamente e o salvar em um objeto
#Fazer o gráfico de Nind~Severidade e adicionar a linha predita pelo modelo




###Resolução do exercício###


summary(modelo)
str(modelo)
names(modelo)

summary(modelo$lme)
summary(modelo$gam)

str(modelo$gam)

str(summary(modelo$gam))

summary(modelo$gam)$s.table

str(summary(modelo$gam)$s.table)

signif=summary(modelo$gam)$s.table[,"p-value"]


new.data=seq(min(fogo.Ad$Severidade, na.rm=T), max(fogo.Ad$Severidade, na.rm=T),
0.01)

predicted=predict(modelo$gam, list(Severidade=new.data), type="response")
```

```
plot(Nind~Severidade, data=fogo.Ad)
lines(predicted~new.data)

###Fim da resolução do exercício###

###Parte 11 - exportando texto###

#Digamos que a gente quer salvar como txt o nosso objeto fogo.Ad

?write.table
write.table(x=fogo.Ad, file="fogo_Ad.txt", quote=F, row.names=F, col.names=T,
sep="\t")

#E se quisermos salvar o resultado da análise?

summary(modelo$gam)

write.table(summary(modelo$gam), file="modelo_fogo_Ad.txt")

write(summary(modelo$gam), file="modelo_fogo_Ad.txt")

unlist(summary(modelo$gam))

capture.output(summary(modelo$gam))

temp=capture.output(summary(modelo$gam))

write(temp, file="modelo_fogo_Ad.txt")

###Fim da parte 11###

###Parte 12 - Exportando figuras###


plot(Nind~Severidade, data=fogo.Ad)
lines(predicted~new.data)

png(filename="fogo_Ad.png", res=300, height=20, width=20, unit="cm")

plot(Nind~Severidade, data=fogo.Ad)
```

```
lines(predicted~new.data)

dev.off()


pdf(file="fogo_Ad.pdf", width=7, height=7)

plot(Nind~Severidade, data=fogo.Ad)
lines(predicted~new.data)

dev.off()


###Fim da parte 12 ###

###Parte 13 - loops###

#Fizemos a análise para Nind de um tipo, mas temos cinco tipos.
#Podemos fazer as cinco análises uma por uma.
#Mas, e se tivéssemos 50 variáveis?
#É melhor pedir ao R fazer as 50 análises automaticamente (e ir dormir! rs)

modelos=list()
tipos=unique(fogo$Tipo)
Ntipos = length(tipos)
tipos

i=1

for(i in 1:Ntipos) {

        print(i)

        tipo=tipos[i]
        fogo.temp=subset(fogo, Tipo==tipo)
        modelo.temp=gamm(Nind~s(Severidade, fx=F, k=-1), data=fogo.temp,
cor=corSpher(form=~x+y, nugget=F), method="REML")
        signif=summary(modelo.temp$gam)$s.table[,"p-value"]
        output.temp=capture.output(summary(modelo.temp$gam))
```

```
		nome.arq=paste("modelo_", tipo, ".txt", sep="")
		write(output.temp, file=nome.arq)

		new.data=seq(min(fogo.Ad$Severidade, na.rm=T), max(fogo.Ad$Severidade,
na.rm=T), 0.01)
		predicted=predict(modelo.temp$gam, list(Severidade=new.data),
type="response")

		nome.arq=paste("figura_", tipo, ".png", sep="")

		png(filename=nome.arq, res=300, height=20, width=20, unit="cm")

		plot(Nind~Severidade, data=fogo.temp)
		lines(predicted~new.data, lty=ifelse(signif<0.05,1,2))

		dev.off()

		}


	###Fim da parte 13###

	###Parte 14 - bonus - amigo secreto no R###


	participantes = c("Aragorn", "Boromir", "Gandalf", "Frodo", "Sam", "Pippin", "Merry",
"Elrond", "Gimli", "Legolas", "Bilbo")
	presenteado = sample(presenteador)
	while (any(presenteado==presenteador)) {
	presenteado = sample(presenteador)
	}

	for (i in 1:length(presenteado)) {
	write.table(presenteado[i],file=paste(participantes[i],".txt"))
	}

	###Fim da parte 14###

	print("Muito obrigado pela atenção e bom almoço! :)")
```