- Tutorial
- Exercícios
- Apostila

4. Tutoriais de Análise Exploratória de Dados

Conferindo Data Frames

A primeira coisa que temos que fazer quando criamos um objeto de dados é verificá-lo cuidadosamente em busca de erros e incoerências. Neste tutorial simulamos duas situações bem comuns: variáveis de fatores com códigos errados e valores NA que na verdade são zeros.

Vamos usar uma planilha com dados fictícios, que você baixa aqui.

Lendo a planilha com `read.table` ou `read.csv2`

```
aves.c <- read.table("aves_cerrado.csv", row.names=1, header=T, sep=";",
dec=",", as.is=T)
aves.c <- read.csv2("aves_cerrado.csv", row.names=1, as.is=T)
```

Verificação inicial do *data frame*

```
head(aves.c)
tail(aves.c)
str(aves.c)
summary(aves.c)
```

Com isso já vemos um problema: os NAs são de fato observações faltantes? Verificando a planilha descobrimos que na verdade são pontos em que não houve avistamento. Portanto, o valor correto é zero Quais são os registros com este problema? Vamos verificar para os urubus

```
aves.c[aves.c$urubu==NA,]
```

Isto não funciona. O teste lógico Para NA é feito pela função `is.na`:

```
is.na(aves.c)
is.na(aves.c$urubu)
```

Que retorna um vetor lógico que usamos para indexar o data frame ou um de seus vetores:

```
aves.c[is.na(aves.c$urubu),]
aves.c[is.na(aves.c$urubu)|is.na(aves.c$carcara)|is.na(aves.c$seriema),]
```

Vamos guardar este pedaço da planilha num objeto temporário

```
temp1 <-
aves.c[is.na(aves.c$urubu)|is.na(aves.c$carcara)|is.na(aves.c$seriema),]
```

E agora vamos corrigir estes valores, que pode ser feito de três maneiras:

```
aves.c$urubu[is.na(aves.c$urubu)] <- 0
aves.c[is.na(aves.c$urubu),2] <- 0
aves.c[is.na(aves.c[,2]), 2] <- 0
```

Continuando, para as outras aves:

```
aves.c$carcara[is.na(aves.c$carcara)] <- 0
aves.c$seriema[is.na(aves.c$seriema)] <- 0
```

Deu certo? Vamos verificar, comparando linhas que agora são zero com o pedaço antigo do objeto que guardamos:

```
aves.c[aves.c$urubu==0|aves.c$carcara==0|aves.c$seriema==0,]
temp1
```

Agora vamos verificar os valores da coluna que será um fator

```
table(aves.c$fisionomia)
```

Há um erro de digitação no código de fisionomia de uma parcela. Corrigindo:

```
aves.c$fisionomia[aves.c$fisionomia=="ce"] <- "Ce"
table(aves.c$fisionomia)
```

Convertendo para fator, que ordenamos da fisionomia mais aberta para a menos:

```
aves.c$fisionomia <- factor(aves.c$fisionomia, levels=c("CL","CC","Ce"))
```

Ufa! Verificando tudo novamente:

```
str(aves.c)
summary(aves.c)
```

Parece que agora está ok.

Média, Mediana e Quantis

Vamos usar o mesmo arquivo do tutorial anterior para explorar as estatísticas descritivas básicas. Começando pela média, e mediana:

```
mean(aves.c[,2:4])
```

Este comando ainda funciona, mas está “condenado” (*deprecated*) desde a verão 2.14.0. Em versões posteriores ele irá gerar um erro. Use a maneira recomendada pelo aviso, que é aplicar as funções a cada elemento do *data frame*, que é uma lista:

```
sapply(aves.c[,2:4],mean)
sapply(aves.c[,2:4],median)
```

No caso de *data frames* você consegue o mesmo com a função `apply`, indicando no segundo argumento a margem a aplicar a função (1 para linhas e 2 para colunas):

```
apply(aves.c[,2:4],2,median)
```

Calcule também a média truncada:

```
apply(aves.c[,2:4], 2, mean, trim=0.1)
```

Há muita diferença entre essas três medidas de tendência central? Como você as explicaria? Agora calcule os quantis para o número de avistamentos de urubus. O padrão da função `quantile` são quartis, como na função `summary`:

```
quantile(aves.c$urubu) ## O mesmo que o retornado pelo summary
summary(aves.c$urubu)
```

Mas você pode mudar para qualquer quantil:

```
quantile(aves.c$urubu, probs= seq(from=0,to=1,by=0.1))
```

Por fim, obtenha quartis, médias e medianas de uma vez para todas as variáveis, com o comando:

```
summary(aves.c[,2:4])
```

Exploração de uma Variável Categórica

Vamos usar um conjunto de dados de um inventário de árvores, que você baixa aqui. Leia com atenção a descrição deste conjunto de dados.

Vamos explorar a variável categórica nome da espécie, com a função `table`:

```
caixeta <- read.csv("caixeta.csv", as.is=T)
names(caixeta)
table(caixeta$especie)
```

Qual a diferença no resultado se usamos o comando abaixo?

```
sort(table(caixeta$especie), decreasing=T)
```

O resultado da função `table` pode ser fornecido à função de gráficos de barras:

```
barplot(sort(table(caixeta$especie), decreasing=T))
barplot(table(caixeta$local))
```

Gráficos para uma Variável

Voltando ao objeto criado no tutorial Conferindo Data Frames, vamos criar em uma só página os quatro gráficos básicos de diagnóstico de uma variável numérica, para o número de avistamentos de urubus:

```
par(mfrow=c(2,2))
boxplot(aves.c$urubu)
hist(aves.c$urubu)
plot(density(aves.c$urubu))
stripchart(aves.c$urubu, method="stack")
par(mfrow=c(1,1))
```

O que acontece se você omite a primeira linha? E a última?

Variações do Histograma

Voltando ao objeto criado no tutorial Conferindo Data Frames, vamos fazer algumas variações de histogramas do número de avistamentos de urubus: Você pode acrescentar marcas (traços) indicando a posição de cada observação no eixo x.

```
## Histograma com os valores (funcao rug)
hist(aves.c$urubu)
rug(jitter(aves.c$urubu))
```

O que acontece se você omite a função `jitter` neste caso? Por que?

Agora vamos fazer um histograma re-escalonado de modo que as áreas das barras somem um. Com isto, podemos sobrepor ao histograma um ajuste não paramétrico de densidade probabilística, que também mantém área um:

```
hist(aves.c$urubu, prob=T)
lines( density(aves.c$urubu),col="blue" )
```

Também sobre este histograma podemos sobrepor a curva normal. Para os parâmetros da normal, usamos a média e o desvio-padrão da amostra.

```
hist(aves.c$urubu, prob=T)
curve(expr = dnorm(x,mean=mean(aves.c$urubu),sd=sd(aves.c$urubu)),add=T,
col="red")
```

Por fim, vamos sobrepor a curva empírica de densidade probabilística com a curva normal:

```
plot(density(aves.c$urubu),col="blue", ylim=c(0,0.08))
curve(expr = dnorm(x,mean=mean(aves.c$urubu),sd=sd(aves.c$urubu)),add=T,
col="red")
```

O que estes gráficos revelam sobre a distribuição do número de avistamentos de urubus neste estudo fictício?

table e aggregate

Usaremos o objeto `caixeta` criado no tutorial Exploração de uma Variável Categórica.

A relação entre duas ou mais variáveis categóricas pode ser explorada com tabelas cruzadas, por exemplo:

```
table(caixeta$especie,caixeta$local)
```

Quando temos uma variavel categórica (fator) e uma numérica, as funções `aggregate` e `tapply` são muito úteis. A função `aggregate` é o equivalente das tabelas dinâmicas das planilhas eletrônicas. Por exemplo, para obter do objeto `caixeta` um *data frame* com a altura média dos fustes de cada espécie de árvore por local você executa o comando:

```
caixeta.alt <- aggregate(caixeta$h,
by=list(local=caixeta$local,especie=caixeta$especie),
                         FUN=mean)
```

Consulte a ajuda da função `aggregate` e experimente outras combinações de fatores e funções, com este conjunto de dados.

xtabs

Crie um objeto com este arquivo e faça as seguintes tabulações:

```
xtabs(Freq~Sex+Survived, data=Titanic.df)
prop.table(xtabs(Freq~Sex+Survived, data=Titanic.df), margin=1)
xtabs(Freq~Class+Survived, data=Titanic.df)
prop.table(xtabs(Freq~Class+Survived, data=Titanic.df), margin=1)
```

Por que usamos a função `xtabs` neste caso e não a função `table`? P.ex:

```
table(Titanic.df$Sex,Titanic.df$Survived)
```

Fórmula Estatística em Gráficos

A função `xtabs` (tutorial anterior) usa a notação de fórmula estatística do R, que é:

variável dependente ~ variáveis preditoras

Esta notação foi criada para os modelos estatísticos, como a regressão linear, mas foi estendida para vários gráficos no R. Experimente os comandos abaixo em que esta notação é usada:

```
boxplot(urubu~fisionomia, data=aves.c)
plot(seriema~urubu, data=aves.c, subset=fisionomia=="Ce")
plot(seriema~urubu, data=aves.c, subset=fisionomia=="CC")
plot(seriema~urubu, data=aves.c, subset=fisionomia!="CL")
```

Que tipo de padrão ou diferenças estes gráficos podem revelar?

Esta fórmula pode ainda incluir um fator condicionante, que aplica a relação proposta dentro de cada nível dos condicionais:

variável dependente ~ variáveis preditoras | variáveis condicionantes

O pacote `lattice` implementa esta idéia para gráficos:

```
library(lattice)
xyplot(seriema~urubu|fisionomia, data= aves.c)
```

Qual a diferença entre este gráfico e os três últimos obtidos com os comandos anteriores?

O quarteto de Anscombe

O pacote `datasets` tem vários conjuntos de dados que se tornaram clássicos da estatística. Vamos analisar um dos mais usados para treino de análises exploratórias. Como o R já carrega o `datasets` por padrão, precisamos apenas fazer uma cópia do objeto para nossa área de trabalho [1)]:

```
data(anscombe)#carrega para a area de trabalho
ls() #agora o objeto está no workspace
```

Este objeto é composto de 4 pares de variáveis, nomeadas `x1` a `x4` (variáveis independentes ou preditoras) e `y1` a `y4` (variáveis dependentes ou resposta):

```
names(anscombe)
```

Compare as médias de cada uma das variáveis. Para isto, use a função `apply` para aplicar a função `mean` a cada coluna[2)] deste data frame:

```
apply(anscombe[1:4], MARGIN=2, FUN=mean)
apply(anscombe[5:8], 2, mean)
```

Faça o mesmo para obter as variâncias:

```
apply(anscombe[1:4], 2, var)
apply(anscombe[5:8], 2, var)
```

A pergunta principal para este conjunto de dados é se há relação entre cada variável `x` e `y`. Isso pode ser testado com o coeficiente de correlação de Pearson, que vai de zero (nenhuma correlação) a um (positivo ou negativo, correlação perfeita).

```
with(anscombe,cor(x1,y1))
with(anscombe,cor(x2,y2))
with(anscombe,cor(x3,y3))
with(anscombe,cor(x4,y4))
```

O que se pode concluir até aqui? Faça os gráficos de dispersão:

```
par(mfrow=c(2,2)) # 4 graficos em uma janela
plot(y1~x1, data=anscombe)
plot(y2~x2, data=anscombe)
plot(y3~x3, data=anscombe)
plot(y4~x4, data=anscombe)
par(mfrow=c(1,1))
```

Você pode acrescentar as linhas de regressão linear [3]:

```
par(mfrow=c(2,2)) # 4 graficos em uma janela
plot(y1~x1, data=anscombe,
     xlim=range(anscombe[,1:4]),ylim=range(anscombe[,5:8]))
abline(lm(y1~x1, data=anscombe))
plot(y2~x2, data=anscombe,
     xlim=range(anscombe[,1:4]),ylim=range(anscombe[,5:8]))
abline(lm(y2~x2, data=anscombe))
plot(y3~x3, data=anscombe,
     xlim=range(anscombe[,1:4]),ylim=range(anscombe[,5:8]))
abline(lm(y3~x3, data=anscombe))
plot(y4~x4, data=anscombe,
     xlim=range(anscombe[,1:4]),ylim=range(anscombe[,5:8]))
abline(lm(y4~x4, data=anscombe))
par(mfrow=c(1,1))
```

Este conjunto de dados foi criado pelo estatístico Frank Anscombe para demonstrar a importância da análise visual de dados, veja aqui.

Leia mais sobre análise exploratória de dados

Artigo com um protocolo de análise exploratória de dados:

- http://onlinelibrary.wiley.com/doi/10.1111/j.2041-210X.2009.00001.x/pdf

Capítulo do livro “Analysing Ecological Data” sobre análise exploratória de dados:

- zuur_cap_4_exploration_statistics.pdf

1)
isto não é estritamente necessário, mas facilita.
2)
argumento `MARGIN=2`
3)
descreve o valor esperado de uma variável dependente como uma função linear de uma variável

independente, veja na unidade sobre modelos lineares