Instituto Nacional de Pesquisas da Amazônia

EXERCÍCIOS PARA O CURSO DE ESTATÍSTICA

WILLIAM E. MAGNUSSON
bill@inpa.gov.br

CÓDIGOS DO R:
Victor Lemes Landeiro -- vllandeiro@gmail.com

Última modificação em: 15 de outubro de 2008

AULA PRÁTICA 1

Coloquem seus dados de medidas de folhas em uma planilha do R com duas variáveis: “tamanho" e “amostra” (1-30).

Para passar os dados pro R:

```
temp<-data.frame()  ## cria uma data frame vazia (temp = temporário)
folhas<-edit(temp)                ### abre o editor de tabelas do R.
```

Coloque seus dados

Agora vamos salvar nossos dados em um arquivo txt.

```
write.table(folhas,file="folhas.txt")
```

PARTE 1

O primeiro passo para calcular o erro padrão das médias é calcular a média para os valores de cada amostra. Use a função tapply() para calcular as médias para cada amostra, salve o resultado num objeto chamado **medias**.

```
medias<-tapply(folhas$tamanho,folhas$amostra,mean) # médias por "amostra"
```

Podemos fazer o mesmo usando colchetes para acessar as variáveis:

```
medias<-tapply(folhas[,1],folhas[,2],mean)  # médias por "amostra"
```

Ou colocando o nome das variáveis dentro dos colchetes

```
medias<-
tapply(folhas[,"tamanho"],folhas[,"amostra"],mean)  # médias por "amostra"
```

Veja os valores das médias:

```
medias
```

Agora que calculamos a média de cada amostra nós podemos calcular o desvio padrão das médias, usando a função sd() (sd= desvio padrão). Dê o nome de **erro5.obs** (erro observado) ao resultado.

```
erro5.obs<-sd(medias)    ## erro padrão das médias observado
erro5.obs
```

Agora calcule 30 estimativas do erro padrão nas 30 amostras usando a fórmula mágica dos estatísticos, que é o desvio padrão em cada amostra dividido pela raiz quadrada do número de observações (no caso 5). Dê o nome de **erro5magico** aos valores de erro padrão calculados com a fórmula mágica.

Primeiro calculamos o desvio padrão em cada amostra:

```
desvios<-tapply(folhas[,"tamanho"],folhas[,"amostra"],sd)
```

Então dividimos o desvio padrão de cada amostra pela raiz quadrada de 5.

```
erros5magico<-desvios/sqrt(5)
```

Agora vamos ver quantos erros padrão da média, calculados com a fórmula mágica, são menores que o erro padrão da média observado (erro padrão "verdadeiro").

Use a função sort() para colocar as estimativas de erro calculadas usando a fórmula mágica dos estatísticos em ordem e conte quantos foram menores que o erro verdadeiro:

```
sort(erros5magico)
```

Para facilitar a contagem podemos usar as funções do R. A função which() mostra quais são menores que o erro verdadeiro e a função length() conta quantos foram menores:

Use a função which() para ver quais valores são menores que o observado.

```
which(erros5magico<erro5.obs)          ## valores que são menores que o observado
```

Para contar quantos foram menores use a função length().

```
length(which(erros5magico<erro5.obs))    ## quantos foram menores
```

Anote em papel quantos erros padrão "mágicos" foram menores que o erro padrão "real"?

Agora, calcule e anote o valor do desvio absoluto médio, do desvio padrão geral e do erro padrão (para as 150 medidas) com a fórmula mágica. Note que o erro padrão é muito diferente dos outros calculados porque o erro padrão é dependente do tamanho da amostra.

Para calcular o desvio absoluto médio vamos calcular primeiro o desvio da média e depois transformar em valores absolutos.

```
desvio<-folhas[,"tamanho"]-mean(folhas[,"tamanho"])
desvio.absoluto<-sum(abs(desvio)) / 150  ## soma valores absolutos / n
```

Para calcular o desvio padrão geral!

```
desv.geral<-sd(folhas[,1]) ###desvio padrão geral do tamanho das folhas
erro.150<-desv.geral/sqrt(150)  ## erro padrão para as 150 medidas
```

Vamos aleatorizar os dados de medidas de folhas para parecer que nossa amostragem foi feita ao acaso. Ou seja, vamos mudar a posição dos valores de tamanho, mas manter a posição das amostras inalterada. Para isso vamos criar um objeto vetor com 150 valores amostrados ao acaso. Usando a função sample() podemos criar esse vetor:

```
sample(1:150)  ### não esqueça de salvar: aleatorio<-sample(1:150)
```

Note que os valores de 1:150 estão desordenados. Agora vamos usar este vetor aleatório e desordenado para aleatorizar os dados de medidas de folhas, usando a função order().

```
tam.aleat<-folhas[ order(aleatorio)  ,"tamanho"] ## ou seja, a
```

posição dos valores de tamanho irá mudar de acordo com o vetor aleatório

Vamos adicionar os valores de tamanho de folhas aleatório na nossa tabela de dados. A função cbind() que significa "juntar colunas" faz isso.

```
folhas.novo<-cbind(folhas , tam.aleat)
folhas.novo
```

Veja que agora nós temos uma tabela com uma terceira coluna que possui os valores aleatorizados do tamanho das folhas.

Agora refaça os passos anteriores para calcular o erro padrão das médias dos valores aleatorizados do tamanho das folhas.

Agora vamos simular dados que possuem aproximadamente os mesmos parâmetros (média e desvio padrão) que os dados que coletamos em campo, mas de forma que agora eles sejam normais e independentes. Depois vamos refazer as análises acima usando este novo conjunto de dados.

Para gerar os novos dados vamos usar a função rnorm() do R. A função rnorm() cria um conjunto de dados aleatório, com distribuição normal, onde podemos especificar a média e o desvio padrão. Ou seja, vamos criar um conjunto de dados novo que possui média e desvio padrão iguais aos dados que coletamos em campo. Os argumentos usados na função rnorm() são: `rnorm(n, mean=0, sd=1)`, onde n é o número de valores, a média (mean) e o desvio padrão (sd).

rnorm(150, mean= M, sd= D) ## substitua M e D pelos valores de média e desvio calculados no arquivo original, e salve o resultado.

```
tamanhoNOVO<- rnorm(150, mean(folhas$tam), sd(folhas$tam))
```

Vejas os novos valores:

```
tamanhoNOVO
```

Junte estes novos valores em uma terceira coluna do arquivo folhas:

```
folhas3<-cbind(folhas,tamanhoNOVO)
```

Nós agora repetiremos o exercício anterior usando estes dados simulados. Calcule o erro padrão da média "observado", o erro padrão da média para cada amostra usando a fórmula mágica e veja quantos foram menores que o observado.

A média dos resultados da fórmula mágica é similar ao valor real? Parece que nós podemos acreditar na fórmula mágica dos estatísticos, embora ela possa não ser tão precisa para todos que queiram usá-la. Contudo, a fórmula dos estatísticos somente ajuda-nos se for usada com um desenho experimental correto para nossas questões. Pensem nisto de agora até a próxima aula.

AULA PRÁTICA 2

TESTE DE PERMUTAÇÃO GERADO PELO COMPUTADOR

Coloque os dados das medidas de altura das pessoas no R, indicando em uma coluna o sexo da pessoa (M ou H) e a outra com medidas de altura. Por razões que nós não vamos discutir agora, intercale homens e mulheres na coluna (H,M,H,M,H...). Chame o arquivo de **tamanho**.

```
temp<-data.frame()  #cria uma data frame vazia
tamanho<-edit(temp)        # Abre a planilha do R. Coloque os dados e feche.
```

Salve seus dados em um arquivo txt.

```
write.table(tamanho,file="tamanho.txt")
```

Confira sua tabela

```
tamanho
```

Nesta aula, nós vamos fazer o teste de permutação da última aula no computador. Este teste pode ser feito usando as funções `replicate()` e a função `sample()`. Nós usaremos esta permutação de forma que vocês possam ver a ligação direta entre o que o computador faz e o lançamento de moedas da aula passada.

Primeiro vamos calcular a dif dos dados originais, como vocês fizeram na aula e armazenar este valor num objeto chamado **difOBS**.

```
difOBS<- diff(tapply(tamanho$alturas,tamanho$sexo,mean))
```

Agora vamos realocar ao acaso os valores de alturas, usando a função `sample()`, enquanto a ordem de sexo permanece inalterada. Depois vamos calcular a média e a diff entre as médias dos valores aleatorizados e repetir isso 100 vezes (aleatorizando alturas e calculando a diff…).

1- Para aleatorizar as medidas de altura usamos o sample.

```
sample(tamanho$alturas) ## veja que a posição das alturas é alterada.
```

2- Calcular as médias com os valores de alturas aleatorizados.

```
medias<-tapply(sample(tamanho$alturas),tamanho$sexo,mean)
medias
```

3 - Calcular a diferença entre as médias usando.

```
medias["H"] - medias["M"]
```

Para facilitar o calculo da dif podemos a função do R chamada `diff()`. Note que o sinal fica invertido porque o R calcula a dif com base na ordem alfabética, mas isto não é um problema desde que todos os cálculos estão sendo feitos da mesma forma. Agora vamos usar a função diff() para calcular a diferença entre as médias:

```
diff(medias)
```

Veja abaixo que nós não precisamos salvar o resultado das médias, mas apenas usar a função tapply() dentro da função diff().

```
diff(tapply(sample(tamanho$alturas), tamanho$sexo,mean))
```

4 - Agora vamos repetir 100 vezes o comando acima usando a função `replicate()`. Ou seja, no passo acima nós calculamos **uma** dif com base nos dados de alturas aleatorizados (`sample(alturas)`) e agora nós vamos calcular 100 difs usando a função `replicate()`.

```
difPERMU<- replicate(100, diff(tapply(sample(tamanho$alturas), tamanho$sexo,mean)))
difPERMU        ### para ver os 100 valores de dif
```

Você pode ver o valor de difOBS, que deve ser o mesmo que calculamos na aula (o sinal pode mudar). Os valores esperados sob a hipótese nula estão no arquivo difPERMU. Conte quantos valores são maiores do que o valor observado. Este número corresponde à probabilidade (como uma porcentagem) de obter um valor tão ou mais extremo que o valor observado se a hipótese estivesse correta.

```
length(which(difPERMU>difOBS))        # Este foi um teste de uma cauda
```

Se quisermos fazer um teste de duas caudas precisamos usar os dois lados da distribuição de freqüências de dif e podemos fazer isso usando os valores absolutos de dif "abs(difPERMU)".

```
length(which(abs(difPERMU)>difOBS))  ## duas caudas
```

Use a função hist() para fazer um histograma de freqüências da distribuição dos valores de dif obtidos sob a hipótese nula (difPERMU).

```
hist(difPERMU)
```

AULA PRÁTICA 3

COMPARAÇÕES SIMPLES COM AJUDA DO COMPUTADOR

Agora que sabemos o que é um teste t, vamos usá-lo para ver se encontramos o mesmo resultado (probabilidade) que o nosso teste de permutação para os dados de altura dos estudantes na turma. Usaremos 100 amostras ao acaso de alturas para fazer repetidos testes *t*.

Primeiro o comando para teste t no R é t.test()

USE:

```
t.test(tamanho$alturas~ tamanho$sexo)          ## teste t
```

comparando alturas entre homens e mulheres, veja o valor de p (*p-value*).

Para **acessar** apenas o valor do p no resultado do teste t usamos **[[3]]**

```
p.obs<-t.test(tamanho$alturas~ tamanho$sexo)[[3]] ## [[3]]
```

para acessar o valor do p

Para fazer os testes repetidos em suas amostras ao acaso use o replicate:

```
replicate(100,t.test(sample(tamanho$alturas)~
tamanho$sexo))
```

Vamos salvar os valores de p em 100 aleatorizações

```
p.aleat<-replicate(100,t.test(sample(tamanho$alturas)~
tamanho$sexo)[[3]])
hist(p.aleat)
length(which(p.aleat<0.05))
```

Quantos testes vocês fizeram para chegar a um resultado significante?

```
which(p.aleat<0.05)
```

A teoria estatística diz que por volta de 1.5 deveria ter tido probabilidades menores do que 0.05. Quantos tiveram valores menores do que 0.1? Menores do que 0.2?

```
length(which(t.aleat<0.1))
length(which(t.aleat<0.2))
```

Agora vamos gerar valores aleatórios de alturas das pessoas de forma que estes valores tenham a mesma média e o mesmo desvio padrão das alturas que medimos na sala. Como estes valores novos são aleatórios nós esperamos que não exista diferença entre as alturas de

homens e mulheres. Para gerar estes valores vamos usar a função rnorm(x, mean, sd). Onde o x é o numero de valores que queremos gerar, mean e sd é a média e o desvio padrão que desejamos que esses valores tenham.

```
med<-mean(tamanho$alturas)  ## média das alturas
desv<-sd(tamanho$alturas) ## desvio padrão das alturas
alt.novo<-rnorm(10, med, desv)
```

Refaça os exercícios anteriores usando os dados novos `alt.novo`.

AULA PRÁTICA 4

REPETIDOS TESTES

Apliquem o mesmo procedimento nas medidas de folhas que vocês fizeram no campo, comparando a amostra 1 com amostra 2, depois amostra 2 com amostra 3, amostra 3 com 4 e assim por diante até que obtenha 29 comparações. Anote quantas comparações foram significativas e quantas tentativas vocês precisaram para obter um resultado significativo?

Vamos ver como fazemos o teste t escolhendo os pares de amostras. Para fazer o teste t entre a amostra 1 e a amostra 2 nós precisamos acessar apenas os dados das amostras 1 e 2.

Então vamos acessar estas amostras e fazer teste t entre as amostras 1 e 2.

```
t.test(folhas[1:10,"tamanho"]~folhas[1:10,"amostra"])  ## Isso fará um teste t entre as amostras 1 e 2, pois as linhas 1 a 10 possuem os dados das amostras 1 e 2.
```

Vamos salvar o valor de p:

```
p.val<-t.test(folhas[1:10,"tamanho"]~folhas[1:10,"amostra"])[[3]]
```

Agora para comparar o resto das amostras mude os valores acima para fazer as comparações desejadas.

Ou seja, para comparar amostras 2 e 3 use: folhas[6:15,"tamanho"]~ folhas[6:15,"amostra"]

```
t.test(folhas[6:15,"tamanho"]~ folhas[6:15,"amostra"])
```

Depois use 11:20 para comparar amostras 3 e 4 e assim por diante.

Para agilizar o processo acima de ficar trocando os valores nós podemos com um looping fazer todos os teste de uma vez só, usando o comando for().

```
for(i in seq(1,145,5)){
p.val[i]<-
t.test(folhas[i:(i+9),"tamanho"]~folhas[i:(i+9),"amostra"])[[3]]
p.val<-na.omit(p.val)
}
```

Agora vamos ver os 29 valores de p.

```
p.val
```

Veja quantos foram menores que 0.05

```
length(which(p.val<0.05))
```

AULA PRÁTICA 5

ANOVA

PARTE 1

Use uma anova para comparar suas 30 amostras de folhas para determinar se há diferenças significativas entre elas. O comando para fazer anova é aov()

Primeiro vamos informar ao R que "amostra" é um **fator**.

```
amostra.F<-as.factor(folhas$amostra)## criamos o fator amostra
amostra.F
```

Agora para fazer a anova:

```
resu.aov<-aov(folhas$tamanho~amostra.F)
```

Ver o resultado:

```
resu.aov
```

Para ver o resultado detalhado da anova use a função summary():

```
summary(resu.aov)
```

A análise detectou sua pseudorepetição? Faça a mesma análise nos dados gerados aleatoriamente. A ANOVA detectou corretamente NENHUMA DIFERENÇA apesar das 435 comparações potenciais entre os indivíduos amostrados?

Agora vamos aleatorizar o fator amostra.F:

```
amostra.Faleat<-sample(amostra.F)
```

Faça a anova com os dados aleatorizados

```
aov.aleat<-aov(folhas$tamanho~amostra.Faleat)
aov.aleat
```

Resultado detalhado

```
summary(aov.aleat)
```

Nós também podemos aleatorizar dentro da função aov().

```
aov(folhas$tamanho~sample(amostra.F))## cada vez que você rodar
```

este comando a ANOVA será feita com uma aleatorização diferente

PARTE 2

Agora vamos comparar os resultados da anova com os resultados do teste t. Para fazer esta comparação vamos usar os dados das medidas de alturas das pessoas que medimos na sala de aula.

Para fazer o teste t:

```
resu.t<-t.test(alturas~sexo,var.equal=TRUE)
```

Para fazer a anova:

```
resu.aov<-aov(alturas~as.factor(sexo))
```

Agora compare os resultados:

```
resu.t
```

Anote o valor do **t**

```
summary(resu.aov)
```

Anote o valor do **F**

Qual a semelhança entre o t e o F? Pegue o valor do t e eleve ao quadrado. Qual a semelhança entre o t e o F?

AULA PRÁTICA 6

TESTE DE TUKEY

Uma das limitações da ANOVA é que ela só diz para você que há diferenças. Mas ela não diz onde está a diferença. Se você quer determinar onde está a diferença, você tem que usar um teste mais fraco que faz mais comparações, o teste de Tukey. Contudo, estes testes são geralmente mais fracos do que a ANOVA e você pode não ser capaz de detectar onde a diferença está.

```
TukeyHSD(aov(tamanho~as.factor(amostra)))
```
Isso não faz sentido para 30 níveis

AULA PRÁTICA 7

REGRESSÃO LINEAR DE MÍNIMOS QUADRADOS

Nós vamos calcular uma regressão linear de mínimos quadrados para a relação entre a cobertura vegetal e a densidade de alguma espécie de animal.

cober	dens
5.3	189
3.2	159
14.5	475
12.4	302
17.3	476
18.9	581
20.1	629
21.2	639
21.4	572
24.2	714
24.2	702
24	684
24.9	677
26.2	858
27	831
27.3	741
28.2	867
27.6	930
29	883
29.8	874

Os dados da tabela acima estão na pasta do curso com o nome de **cobdens.txt**, importe o arquivo para o R.

Para fazer a regressão o comando é lm():

```
regress<-lm(dens~cober) #
```

lembre que os dados estão em uma dataframe, você terá que usar ***nome.dados*[,"dens"],** ou **nome.dados$dens** (use a forma que preferir).

Veja os coeficientes da regressão e anote o valor do ***a*** (intercepto) e do ***b*** para traçar a linha no gráfico. No resultado que o R mostra o coeficiente a é chamado de intercepto e o coeficiente b aparece com o nome da variável preditora (neste caso **cober**).

```
regress
```

Para ver resultado de talhado da regressão use a função summary():

```
summary(regress)
```

Para fazer o gráfico da regressão:

```
plot(cober,dens)
```

Agora vamos calcular os valores estimados da variável resposta usando a equação da reta (substitua os valores de a e b pelos valores que você anotou acima):

```
dens.esti<-a+b*cober
```

Agora que temos os valores estimados de densidade vamos colocar estes valores no gráfico com símbolos e cor diferentes.

```
points(cober,dens.esti, pch=8,col="red")
```

Agora vamos traçar a reta de regressão usando a função lines() a partir da equação da reta:

```
lines(cober, a+b*cober)
```

Não é preciso anotar os coeficientes todas as vezes que for traçar uma linha de regressão, basta usar a função abline() e colocar o resultado da regressão dentro da função abline:

```
plot(cober,dens)
abline(regress) ##
```

A linha captura a relação mostrada pelos pontos?

AULA PRÁTICA 8

REGRESSÃO COM DADOS TRANSFORMADOS

Os dados da tabela abaixo estão na pasta do curso com o nome de **biomdap.txt**, importe os dados para o R.

dap	biomassa
1.7	6.6
1.4	4.9
4.3	115.6
3.5	20.5
5.7	117.4
6.6	336.6
7.5	540.1
8.3	600.6
8.5	306.7
11.3	1271.2
11.3	1122.5
11.1	937.7
12.1	874.8
13.9	5349.3
14.9	4085
15.4	1657.8
16.8	5873.7
15.9	11021.4
18.3	6885
18.8	6308.9

Veja o gráfico da relação entre as variáveis:

```
plot(dap,biom)
```

Vamos transformar os dados em log:

```
biom.log<-log(biom)
dap.log<-log(dap)
```

Agora veja a relação entre as variáveis transformadas:

```
plot(dap.log,biom.log)
```

Agora vamos fazer uma regressão linear dos valores transformados em log:

```
resu<-lm(biom.log~dap.log)
resu
```

Anote os valores dos coeficientes a e b e substitua os valores **a** e **b** no comando abaixo para calcularmos os valores estimados de biomassa:

```
biom.est<- exp(a)*dap^b
```

Vamos plotar os pontos dos valores estimados:

```
points(dap,est,pch=4,col=4)
```

Agora vamos traçar a curva:

```
lines(dap,exp(a)*dap^b)
```

AULA PRÁTICA 9

REGRESSÃO NÃO LINEAR

Ao invés de transformar as variáveis em log, vamos fazer uma regressão não linear. No R a principal diferença entre modelos lineares e não lineares é que nós precisamos especificar a natureza exata da equação dos modelos não lineares como parte da fórmula do modelo. No lugar da função lm() nós usamos a função nls(). Então, ao invés de usarmos y~x como fórmula nós usaremos a formula `y~a*x^b` (função de potência), no nosso exemplo será y = biomassa e x = dap. No comando abaixo, a e b são letras, não substitua por números.

```
modelo<-nls(biom~a*dap^b,start=list(a=500,b=0.05))
```

Veja o resultado e anote os valores dos coeficientes a e b.

```
modelo
```

Agora vamos usar a equação para encontrar os valores estimados de biomassa. Substitua os valores **a** e **b** pelos valores que você anotou.

```
biomEST<-a*dap^b
```

Refaça o gráfico de dap X biomassa.

```
plot(dap,biom)
```

Adicione os valores estimados a partir do modelo não linear:

```
points(dap,biomEST,col="red",pch=4)
```

Agora vamos inserir uma curva que representa os valores estimados de biomassa de acordo com o modelo não linear.

```
lines(dap,a*dap^b)
```

Agora vamos inserir a curva que nós havíamos calculado usando a regressão dos valores em log. No comando abaixo os valores de a e b **são os valores de a e b calculados no exercício anterior da AULA 6.**

```
lines(dap,exp(a)*dap^b, col=4)
```

AULA PRÁTICA 10

REGRESSÃO MÚLTIPLA

Nós faremos a regressão múltipla no computador primeiro de uma forma longa e depois de forma curta. A forma longa é para que vocês possam compreender o método, e a forma curta para que vocês não se chateiem fazendo da forma longa no futuro.

Coloquem os dados de plantas, animais e área no computador.

```
animais<-c(5,1,2,3,5,7)
arvores<-c(3,2,4,1,6,5)
area<-c(4,1,3,2,5,6)
```

PASSO 1

Forma longa

Calculem a regressão de animais na área. Anotem os parâmetros da regressão ("*a*" e "*b*").

Animais modelado pela área

```
aniXarea<-lm(animais~area)
```
faz a regressão animais~area#anote coeficientes *a* e *b*

```
aniXarea
coef(aniXarea)
coefi.ani<-coef(aniXarea)
```
extrai e salva os coeficientes *a* e *b* com a função coef

Calcule os valores esperados do número de animais para cada valor observado de área. Lembrem que a equação da reta é y= a + b*x. Portanto para calcular os valores estimados basta usar a equação. Ou seja, multiplicar cada valor de x pelo coeficiente b e somar o valor do coeficiente a.

```
coefi.ani[1]+(coefi.ani[2]*area)
```
a + b * x

```
aniXareaEST<-coefi.ani[1]+(coefi.ani[2]*area)
```
para salvar

Não é preciso calcular desta forma os valores estimados, pois ao fazer a regressão usando lm() o R já havia calculado os valores estimados. Para acessar os valores estimados dentro do resultado da regressão use `$fitted` após o nome do resultado da regressão.

```
aniXarea$fitted
```
extrai os valores estimados, use este método daqui em diante.

Faça o gráfico de aniXareaEst contra aniXarea$fitted para verificar que os valores ajustados que nós calculamos são os mesmos que o R calculou.

Agora faça a regressão do número de plantas nas áreas e **anote** os parâmetros da regressão ("a" e "b").

Árvores modelado pela área

```
arvXarea<-lm(arvores~area)  #anote coeficientes a e b
coefi.arv<-coef(arvXarea)
```

Salve os valores estimados de plantas para cada valor de área.

```
arvXareaEST<-arvXarea$fitted
```

Agora subtraia os valores de animais estimados pelo modelo do número observado de animais para obter o número de animais independente da área. Salve com nome de anarea.

```
anarea<-animais-aniXareaEST ### animais independente da área
```

Subtraia os valores de plantas estimados pelo modelo do número observado de plantas para obter o número de plantas independente da área. Salve com nome de arvarea.

```
arvarea<-arvores-arvXareaEST ## arvores independente da área
```

Faça uma regressão de anarea por arvaria. Anote os resultados e faça o gráfico de anaria contra arvaria.

```
longa<-lm(anarea~arvarea)
```

Anote o valor dos coeficientes a e b:

```
longa
```

Salve os resíduos:

```
parciais.longa<-longa$residuals
```

Faça o gráfico dos parciais:

```
plot(arvarea,anarea)
```

Compare com o gráfico com os valores originais:

```
windows()
plot(arvores,animais)
```

Repita este processo para obter a regressão parcial dos animais na área, independente do número de plantas, e o gráfico correspondente.

Animais sem o efeito de árvores:

```
aniXarvo<-lm(animais~arvores)
coefi.ANI<-coef(aniXarvo)
```

```
aniXarvoEST<-aniXarvo$fitted
aniarvo<- animais -aniXarvoEST
```

Área sem o efeito de árvores.

```
areXarvo<-lm(area~arvores)
coefi.ANI<-coef(areXarvo)
areXarvoEST<-areXarvo$fitted
arearvo<- area-areXarvoEST
lm(aniarvo~arearvo) ## Veja e anote o coeficiente b
plot(arearvo,aniarvo)
```

Nós não podemos usar as probabilidades para as regressões parciais obtidas desta maneira (modo longo), porque nós superestimamos os graus de liberdade. Lembrem-se que os graus de liberdade são iguais aos números de observações menos o número de parâmetros estimados. Nesta análise, o computador não sabia que nós tínhamos previamente estimado parâmetros relacionados à área, e logo usou tantos graus de liberdade. Contudo, as inclinações das regressões parciais são estimadas corretamente (coeficiente b). Logo, nós poderemos estimar quantos animais aumentariam ou diminuiriam com um aumento de plantas se a área permanecesse constante. O mesmo se aplica para o efeito de área independente de plantas.

PASSO 2

Vocês não precisam seguir os passos acima sempre que forem fazer uma regressão múltipla. É possível fazer a regressão múltipla inserindo todas as variáveis em um único modelo de regressão.

Nós agora vamos fazer a regressão múltipla de forma curta para ver se dá o mesmo resultado.

Forma curta

Para fazer a regressão múltipla no R o comando também é lm(). A fórmula para regressão múltipla é: $Y = a + b_1*X\ 1+ b_2*X2$. Portanto, para fazermos a regressão múltipla nós precisamos usar a fórmula → $y{\sim}x_1+x_2$

```
curta<-lm(animais~arvores+area)    ### note o arvores + area
```

Para acessar os resíduos da regressão múltipla curta use:

```
parciais.curta<-curta$residuals### resíduos da variável resposta
```

Agora veja que os resíduos que calculamos da forma longa são os mesmos que calculamos da forma curta.

```
plot(parciais.curta,parciais.longa)
```
para ver que dá no mesmo (longo = curta)

```
summary(curta)
summary(longa)
```
veja que os resíduos são iguais, talvez os sinais sejam diferentes

No R a função `summary()`, produz uma tabela com os coeficientes da regressão parcial e os valores de probabilidade corretamente ajustados para os graus de liberdade. Os coeficientes das regressões parciais são os mesmos que vocês obtiveram usando o método longo? Vocês ainda têm um efeito negativo das plantas sobre os animais? Se não tiverem, vocês cometeram algum erro.

Vamos ver se os gráficos das regressões parciais são os mesmos que aqueles que vocês obtiveram pelo método longo. O pacote `car` possui a função `av.plots` que plota os parciais. A função funciona assim: `av.plots(modelo,variable="parcial desejado").`

A função av.plots possui uma facilidade de identificar os pontos no gráfico usando a função identify. **Pressione ESC** após usar a função. Então para plotar os parciais de árvores e animais use:

```
av.plots(lm(animais~arvores+area),variable="arvores")
```
Pressione ESC

Para conferir se o gráfico é o mesmo obtido da forma longa:

```
plot(arvarea,anarea)
```

AULA PRÁTICA 11

STEPWISE SELECTION

Coloque no R os dados que você gerou usando a função edit. Chame seu arquivo com os dados de **step**.

```
temp<-data.frame()
step<-edit(temp)
```

Agora use a função attach() para deixar as variáveis na sua tabela disponíveis apenas pelo nome.

```
attach(step)
```

Agora faça uma regressão múltipla que inclui todas as suas variáveis preditoras:

```
model1<-lm(resp~pre1+pre2+pre3+pre4+pre5+pre6+pre7+pre8+pre9+pre10)
```

Veja o resultado detalhado do seu modelo completo:

```
summary(model1)
```

A vamos refazer a nossa análise, mas retirando a variável que tem o maior valor de p no modelo anterior (i.e. retirar a variável menos significativa do modelo). Como exemplo, vamos supor que o preditor 5 foi o que teve o maior valor de p e vamos retirá-lo do modelo.

```
model2<-update(model1,~. - pre5) # excluímos o preditor 5
```

Veja o resultado detalhado do modelo 2:

```
summary(model2)
```

Continue este processo de retirar a variável preditora com maior valor de p até que seu modelo contenha apenas variáveis preditoras significativas (i.e. $p < 0.05$).

```
model3<-update(…)
...
```

No final desatache os dados:

```
detach(step)
```

AULA PRÁTICA 12

ANOVA COM MAIS DE 1 FATOR (MULTIFATORIAL)

Vamos planejar um desenho ortogonal para um experimento que envolve 5 fatores (a,b,c,d,e) com dois níveis para cada fator (0, 1) e duas réplicas. No R nós podemos usar o pacote agricolae para gerar desenhos ortogonais. A função fact.nk (desenho fatorial com k-fatores e n-níveis dos fatores), pode ser utilizada neste caso.

```
library(agricolae)
## fact.nk(level, factors, blocks)
```

level = número de níveis do fator; factors = número de fatores; blocks = número de réplicas.

Então para criarmos o desenho com 5 fatores, 2 níveis e 2 réplicas:

```
fact.nk(2,5,2)
```

Salve o desenho:

```
desenho<-fact.nk(2,5,2)
```

desenho do experimento, veja que são necessários 64 plots para fazer este experimento.

Verifique que o desenho criado é ortogonal calculando a correlação entre cada fator. Lembre que para ser ortogonal a correlação entre dois fatores deve ser 0.

```
cor(desenho[,3:7])
```

matriz de correlação entre os fatores

Vamos supor que este experimento tinha o objetivo de avaliar o crescimento de plantas em relação a estes 5 fatores. Então vamos criar 64 valores de resposta aleatórios, para representar a hipótese nula de que nenhum dos fatores afeta o crescimento de plantas.

```
cres<-rnorm(64, mean=5, sd=2)
```

Vamos adicionar os valores de crescimento em nossa dataframe do desenho experimental e dar o nome exp "experimento":

```
exp<-cbind(desenho,cres)
```

Veja a tabela:

```
exp
```

Agora vamos fazer uma anova para avaliar o efeito destes fatores sobre o crescimento. O sinal de vezes "*" no comando abaixo indica que nós queremos incluir as interações entre os fatores.

```
resu.aov<-aov(exp$cres~exp$A*exp$B*exp$C*exp$D*exp$E)
```

Veja o resultado detalhado e anote quantos valores de p (Pr(>F)) foram significativos:

```
summary(resu.aov)
```

	Df	SumSq	MeanSq	F	Pr(>F)	
dat$A	1	1.292	1.292	0.3553	0.5534	<---valor de p para o fator A

dat$B	1	0.115	0.115	0.0315	0.8597	<---valor de p para o fator B
dat$C	1	0.726	0.726	0.1998	0.6566	<---valor de p para o fator C
dat$D	1	0.055	0.055	0.0151	0.9025	<---valor de p para o fator D
dat$E	1	0.353	0.353	0.0972	0.7564	<---valor de p para o fator E
Residuals	58	210.837	3.635			

AULA PRÁTICA 13

ANÁLISE DE CAMINHOS

Para fazer este exercício vamos usar os dados de lagostins que está na pasta do curso. Estes dados são os mesmos da página XXX do livro Estatística sem matemática.

Importe os dados:

```
lagostins<-read.table("lagostins.txt",header=T)
```

Para calcular o b padronizado, vamos transformar os dados usando a função decostand() do vegan: Anote os valores do coeficiente b em cada análise.

```
d.pad<-decostand(lagostins,"standardize")
lm(d.pad$lagostins~d.pad$poluição+d.pad$peixes+d.pad$fitoplancton)
lm(d.pad$poluição~d.pad$peixes)
lm(d.pad$poluição~d.pad$fitoplancton)
```

AULA 14

ESCALONAMENTO MULTIDIMENSSIONAL NÃO MÉTRICO (NMDS)

Coloque seus dados no R em forma de vetores:

```
cabeça<-c(...)
pescoço<- c(...)
peito<- c(...)
cintura<- c(...)
quadril<- c(...)
coxa<- c(...)
sexo<- c(...)
```

Agora vamos juntar os vetores com os valores de medidas em uma dataframe, exceto o objeto sexo:

```
medidas<-data.frame(cabeça,pescoço,peito,cintura,quadril,coxa)
```

Para fazer o NMDS precisamos do pacote vegan:

```
library(vegan)
```

O comando para fazer o NMDS é metaMDS. O argumento "euclid" indica que usaremos a distancia euclidiana e o argumento autotransform indica que não queremos que o R faça nenhuma transformação nos dados.

```
resu.MDS<-metaMDS(medidas,"euclid",autotransform=FALSE)
```

Para fazer o gráfico do NMDS: pch=sexo indica que os símbolos do gráfico serão as letras do sexo:

```
plot(resu.MDS$points,pch=sexo)
```

Transforme os dados dividindo pelo total da linha. Para isso vamos usar a função decostand() do vegan

```
med.pad<-decostand(medidas,"total")
```

Agora refaça o NMDS usando os dados padronizados e depois faça o gráfico.