Introdução à Biometria utilizando 

Leandro R. Monteiro

José Louvise Gomes-Jr (colaborador)

Prefácio

Esta apostila foi criada para a utilização na disciplina LCA 2613 Biometria, na Pós Graduação em Ecologia e Recursos Naturais da Universidade Estadual do Norte Fluminense, situada em Campos dos Goytacazes, Estado do Rio de Janeiro, Brasil. Este material está sendo disponibilizado gratuitamente de modo a contribuir para a disseminação da plataforma R entre estudantes e usuários de estatística aplicada à biologia em países de língua portuguesa. Consideramos a disponibilização deste texto uma pequena retribuição e adição ao esforço de inúmeros pesquisadores e programadores direcionado ao R-system no sentido de criar uma plataforma de software livre para análises estatísticas que possa ser utilizada em qualquer sistema operacional.

O texto foi produzido em LaTeX , utilizando o *front end* Kile (http://kile.sourceforge.net), no sistema Poseidon Linux (http://poseidon.furg.br). A versão do R utilizada para a produção de figuras e saídas com resultados foi a 2.0.1, de modo que pequenas diferenças podem ser encontradas em relação a versões mais recentes. Os conjuntos de dados utilizados e disponibilizados em conjunto são referenciados no texto quando originados de outras fontes. Quando nenhuma referência é feita à fonte original, os dados correspondem a uma pequena parte de dados relativos à pesquisas realizadas pelo grupo dos autores.

Este texto deverá ser atualizado permanentemente, à medida que recebermos sugestões e críticas, tanto dos nossos próprios estudantes quanto de leitores que venham a ter contato com este material.

Versão 1-2006

Laboratório de Ciências Ambientais, CBB, Universidade Estadual do Norte Fluminense, Av. Alberto Lamego 2000, Campos dos Goytacazes, RJ, cep 28013-600, Brazil. (lrmont@uenf.br).

Sumário

1 O ambiente R.

O R é um conjunto integrado de programas para manipulação de dados, cálculo e visualização de gráficos. Entre outras coisas ele possui:

- Ferramentas eficientes para manipulação e armazenamento de dados; um conjunto de operadores para cálculos matriciais;
- uma grande e coerente coleção integrada de funções para análise de dados;
- ferramentas gráficas para análise e visualização dos dados, seja impressos ou na tela do computador;
- uma linguagem computacional simples, bem desenvolvida e efetiva, que inclui condicionais, alças, funções recursivas definidas pelo usuário, e funções para entrada e saída de dados

Assim, o termo ambiente *(environment)* pretende caracterizá-lo como um sistema completamente planejado e coerente, ao contrário de uma aglomeração de ferramentas muito específicas e inflexíveis, como é o caso freqüentemente de outros pacotes estatísticos. Apesar de existirem algumas opções de interfaces gráficas para a interação com o ambiente R, como o pacote R-commander, o diferencial do pacote é a flexibilidade obtida a partir da elaboração de funções para resolver problemas específicos, o que deve ser realizado dentro da interface de comandos.

1.1 Páginas de internet sobre o R-system.

Algumas páginas na internet podem ser encontradas para melhor compreensão de como funciona o sistema R como, indicado abaixo:

- http://r.project.org - A *home page* do projeto R. A partir desta página web podemos acessar todas as páginas relacionadas abaixo.
- http://cran.br.r-project.org - O servidor mirror (espelho) brasileiro (UFPR). Existem espelhos também em outras instituições do Brasil. Estes podem ser acessados a partir de um *browser* como Internet Explorer ou Mozilla ou a partir do próprio ambiente computacional. A partir dos sites CRAN (Comprehensive R Archive Network), podemos baixar arquivos de ajuda e pacotes adicionais que extendem as funções do sistema base.
- http://r.project.org/mail - r-help é a lista mais apropriada para usuários. Esta página contém uma ferramenta de busca nos arquivos que facilita a obtenção de respostas baseada em dúvidas anteriores.
- http://cran.r-project.org/other-docs.html - documentos de ajuda e tutoriais em formato pdf. Alguns dos arquivos são muito grandes e compreensivos, outros são guias rápidos para a realização de tarefas específicas. Outra opção de ajuda para tarefas no R é o seu próprio sitema de ajuda em formato html ou no formato próprio do programa. Neste sistema podem ser encontradas funções e esclarecimento de dúvidas para execução de comandos, assim como exemplos, os quais podem ser mais esclarecedores que o texto explicativo.

1.2 Iniciando e terminando o R

Quando se utiliza um programa baseado em comandos como o R, encontramos um alerta de espera para um comando *(prompt)*. O alerta demonstrado neste programa é o '>'. Isto significa que o programa está esperando um comando para realizar alguma tarefa. É neste espaço que vamos escrever as instruções para o programa, sempre observando os detalhes de sintaxe da linguagem R. É sempre importante prestar atenção ao escrever comandos que o programa diferencia letras maiúsculas e minúsculas (portanto, não podemos chamar a função `read.table` escrevendo `Read.table`). Símbolos como , . / ; apresentam funções específicas dentro da linguagem R, o que significa que uma vírgula mal colocada pode fazer com que o programa não compreenda os comandos e deixe de executar a tarefa. Após escrever um comando, devemos teclar *(enter)* para que o programa execute. Caso o comando digitado esteja incompleto, o resultado é um novo *prompt* com o sinal (+). Isto quer dizer que você deve escrever algo para completar o comando antes que ele possa executar. Dependendo do sistema operacional sendo utilizado, encontraremos maneiras diferentes de iniciar o programa. Como ainda não estamos tão avançados a ponto de poder ensinar apenas a utilização dentro de um sistema Linux, provavelmente a maior parte dos alunos utilizará o Windows como sistema operacional. Uma vez realizada a instalação do pacote a partir do arquivo executável baixado de algum site espelho brasileiro dentre os listados abaixo (e se tudo correu bem), encontraremos um ícone do R na área de trabalho do sistema. A partir deste atalho entraremos no ambiente R e poderemos começar a escrever os comandos operacionais.

Para sair do R digite apenas

```
>q()
```

Esta função é em geral seguida por uma pergunta `(save workspace image? y/n/c)`. Se a resposta dada for y (sim), o programa salvará os conjuntos de dados armazenados na memória, bem como o histórico de comandos (o que facilita muito o processo de relembrar comandos anteriormente utilizados e a repetição de análises). É sempre interessante manter um registro das análises feitas gravando os comandos (manual ou automaticamente) em um arquivo texto chamado arquivo de *script*, caso seja necessário repetir uma análise em outro computador ou caso o arquivo com a história de comandos seja apagado da memória acidentalmente.

1.3 Comandos de Ajuda do R.

Aqui estão algumas funções que auxiliarão na pesquisa de funções de comando no R:

a) Para a pesquisa de uma função particular: EXEMPLO (LM) LINEAR MODEL:

```
>help(lm)
```

enter para confirmar. Após isto abrirá uma janela com o comando desejado. Neste caso é necessário saber exatamente o nome da função, caso contrário o programa retorna uma mensagem de erro.

b) Para disponibilizar um exemplo:

```
>example(lm)
```

c)Um sistema de ajuda em formato html também pode ser iniciado usando:

```
>help.start()
```

d)Uma pesquisa no sistema de ajuda para documentos pode ser feita utilizando uma dada expressão do nome, do título ou palavras chaves:

```
>help.search(lm)
```

Existem ainda outras maneiras de conseguir ajuda dentro e fora do ambiente do programa (a partir da lista de discussão r-help na internet). Uma vez que o funcionamento do programa é compreendido, o processo de conseguir ajuda se torna fácil e automático.

1.4 Diretório de Trabalho

Para verificação do diretório de trabalho sendo utilizado pelo R, utilize o seguinte comando:

```
>getwd
```

Se o diretório desejado não estiver sendo utilizado pelo R para leitura e gravação dos arquivos siga estes passos:

```
File-
Change dir...
```

e escolha o diretório desejado através do *browse*. Para verificar o conteúdo de um determinado diretório (para lembrar os nomes dos arquivos, por exemplo, usamos o comando `dir`

```
>dir()
```

Que produz uma listagem dos arquivos e subdiretórios dentro do diretório de trabalho. O comando `ls()` pode ser usado para verficar os nomes dos objetos que estão armazenados dentro do R. O conjunto de objetos armazenados é chamado de *workspace*. Assim que a função for aplicada o nome dos arquivos em uso devem aparecer

```
>ls() [1] 'ap' 'apf' 'apm'
```

Neste caso, os objetos armazenados durante a sessão são os nomeados acima. Caso esta seja a primeira sessão, nenhum objeto deve aparecer na lista.

1.5 Objetos no R

O R trabalha executando funções sobre objetos. As funções devem ser sempre acompanhadas pelo nome do objeto dentro de parênteses. A referência às variáveis a serem analisadas estatisticamente é em geral realizada a partir de vetores. Um vetor dentro do R é uma *string* de caracteres com os mesmos atributos intrínsecos ou modos. Estes modos podem ser *numeric, complex, logical, character e raw*. Por exemplo, um vetor **a** numérico correspondendo à sequência de números inteiros de 5 a 10 pode ser definido como:

```
> a<-5:10
> a
[1] 5 6 7 8 9 10
```

Onde o operador de atribuição é determinado pela combinação de sinais <- ou ->, dependendo da direção de atribuição, de modo que

```
>5:10->a
```

e

```
>a<-5:10
```

são equivalentes.

Quando escrevemos o nome de um objeto no *prompt* o resultado é uma listagem do conteúdo do objeto, como visto no exemplo acima.

Quando queremos referenciar um elemento ou um grupo de elementos pertencente a um vetor, podemos utilizar os elementos de indexação dentro de colchetes. Por exemplo, para listar o terceiro elemento do vetor **a**, escrevemos

```
> a[3]
[1] 7
```

Ou se quisermos listar os 3 primeiros elementos do vetor **a** escrevemos

```
> a[1:3]
[1] 5 6 7
```

Os objetos apresentam diferentes classes, sendo as mais importantes os modos de vetores vistos anteriomente ou objetos compostos como um conjunto de objetos (vetores) menores comor *"list"* (listas de objetos menores geralmente com resultados de funções de análises estatísticas), *"matrix"* (matrizes que permitem a utilização de operações algébricas apropriadas), *"factor"* (uma decomposição ortogonal de uma variável categórica utilizada em modelos estatísticos) e *"data.frame"* (os conjuntos de dados ou alguns resultados de análises podem ser data frames). Entraremos em contato com estes tipos de objetos à medida que progredirmos na utilização do ambiente R.

1.6 Carregando arquivos de dados

Existem várias maneiras de ler um conjunto de dados externos ao R, um modo interessante e simples de se fazer a leitura é a utilização do comando,*'read.table'*, que cria uma tabela *(data frame)* dentro do R com dados lidos a partir de um arquivo em texto simples. A esta função, podem ser atribuídos uma série de argumentos que possibilitam dizer ao programa como ler o arquivo (se existem nomes de variáveis ou objetos, qual é o separador decimal, qual é o separador de campos, etc...). Por exemplo, se o arquivo a ser lido apresenta os nomes das variáveis na primeira linha, adicionamos o argumento `'header=TRUE'` ou `'header=T'`.

Imagine um arquivo contido num diretório chamado
texttt work, chamado de `exemplo1.txt` que pretendemos carregar para a área de trabalho. O arquivo, em formato texto pode ser aberto por qualquer programa que trabalhe com arquivos em formato ASCII (Notepad, por exemplo, mas existem opções melhores em software livre, como o ConText - http://www.context.cx/), ou qualquer planilha eletrônica que permita a gravação de arquivos em modo texto. O conteúdo do arquivo exemplo1.txt está listado abaixo:

```
   X   Y
5.14 4.4
4.73 3.95
4.52 3.83
5.23 4.41
5.21 4.34
```

Para carregar o arquivo como um objeto R, escrevemos o comando

```
>ex1<-read.table('exemplo1.txt', header=T)
```

onde, ex1 passa a ser o nome do arquivo a ser manipulado agora no ambiente do R; `header=T` indica que o cabeçalho (nome das variáveis) também será lido pelo comando e anexado ao conjunto de dados. É importante atentar para o fato que sempre que nos referimos a caracteres não numéricos no R (como nomes de arquivos), precisamos utilizar haspas. Para verificar se o arquivo

foi carregado corretamente pelo R, escreva o nome do objeto ao qual o conteúdo do arquivo foi atribuído.

```
> ex1
     X   Y
 1 5.14 4.40
 2 4.73 3.95
 3 4.52 3.83
 4 5.23 4.41
 5 5.21 4.34
```

Se o conjunto de dados for listado corretamente, ele está pronto para ser trabalhado no R. É importante ressaltar que o separador decimal *default* da função é o ponto. Caso os decimais no seu arquivo estejam separados por vírgula, deve-se adicionar um argumento `dec = ','` à função `read.table`, logo após o argumento `header` (ou substituir as vírgulas por pontos no arquivo texto).

Uma vez que o conjunto de dados encontra-se como um objeto dentro do *workspace*, podemos referenciar uma variável (vetor) específico com o símbolo $. Por exemplo, se quisermos listar a variável X do objeto `ex1` escrevemos

```
> ex1$X
[1] 5.14 4.73 4.52 5.23 5.21
```

Temos como resultado a listagem apenas da variável X.

Quando vamos utilizar um determinado objeto repetidamente, podemos anexá-lo à memória do programa, de modo a evitar referenciar o nome do objeto antes da variável. O comando a ser utilizado neste caso é o `'attach()'`. Com este comando, um objeto é anexado para uma referência mais rápida a uma determinada variável.

```
> attach(ex1)
> X
[1] 5.14 4.73 4.52 5.23 5.21
```

Percebemos que não é mais necessário escrever o nome do objeto `ex1` para listar a variável X. Para desanexar um arquivo utilizamos o comando

```
>detach(ex1)
```

Assim, o programa estará livre para anexar outro conjunto de dados para análise. É interessante não anexar vários arquivos ao mesmo tempo se houver variáveis com o mesmo nome. Caso isto aconteça, o objeto anexado por último será utilizado nas referências. Para remover um objeto do *workspace* utilizamos o comando `rm(object)`. Por exemplo, para remover o objeto a criado anteriormente, escrevemos

```
>rm(a)
```

1.7 Realizando operações algébricas:

a) Soma

```
>1+1
[1] 2
```

b)Subtração

```
>14-6
[1] 8
```

c) Multiplicação

```
>3*4
[1] 12
```

d) Divisão

```
>47/11
[1] 4.272727
```

e) Potenciação

```
>4^2
[1] 16
```

f) Logaritmização (utilizamos `log()` para logaritmos naturais e `log10()` para logaritmos decimais. Outras bases podem ser utilizadas com `logb(x,base)`.

```
> log(1000)
[1] 6.907755
> log10(1000)
[1] 3
```

e) Radiciação (raiz quadrada)

```
> sqrt(4)
[1] 2
```

f) Módulo (valor absoluto de um número)

```
> abs(-3)
[1] 3
```

Os mesmos operadores podem ser utilizados para realizar operações com variáveis (se a classe das mesmas for `array`, se forem da classe `matrix`, os operadores são ligeiramente diferentes). Se quisermos multiplicar a variável X do objeto `ex1` por 2, simplesmente escrevemos o comando:

```
> X2
<-ex1$X*2
> X2
[1] 10.28 9.46 9.04 10.46 10.42
```

Atribuímos então à variável `X2` os elementos multiplicados da variável X. Podemos ainda realizar operações com duas variáveis. Por exemplo:

```
> XY<-ex1$X*ex1$Y
> XY
[1] 22.6160 18.6835 17.3116 23.0643 22.6114
```

Obtemos assim a multiplicação dos elementos de uma variável pelos elementos correspondentes do outro. A multiplicação de matrizes requer a definição do seguinte operador: (%*%). A multiplicação de matrizes é realizada multiplicando linhas por colunas e somando os resultados. A ordem das matrizes deve ser compatível para tanto, de modo que o número de colunas na primeira matriz deve ser igual ao número de linhas na segunda matriz. Por exemplo, podemo multiplicar uma matriz 2x3 por uma matriz 3x4 (resultando em uma matriz 2x4), mas não podemos multiplicar uma matriz 3x4 por uma matriz 2x3. Podemos transformar a tabela *(data frame)* `ex1` em uma matriz **A** usando o comando `as.matrix()`.E podemos obter uma matriz de produtos cruzados (importante para ocálculo das matrizes de covariância ou correlação) multiplicando a transposta de **A** por **A**.

```
A<-as.matrix(ex1)
 > t(A)%*%A
  X    Y
 X 123.7199 104.2868
 Y 104.2868 87.9151
```

Outros comandos estão ainda disponíveis para álgebra de matrizes como o cálculo de matrizes inversas (`solve()`), determinantes (`det()`), autovalores e autovetores (`eigen()`). Estes podem ser encontrados nos arquivos de ajuda correspondentes.

1.8 Salvando objetos no R

Uma das maneiras mais simples de salvar um objeto no R (dados modificados ou resultados de análises) é pedir para listar na tela (escrevendo o nome do objeto), copiando e colando diretamente em um arquivo texto. Caso o objeto seja grande demais, é possível utilizar a função `'write.table()'` onde os argumentos a serem passados são o nome do objeto e o nome do arquivo onde o texto será salvo. Por exemplo, o comando

```
> write.table(log(A),file="a.txt")
```

escreve em um arquivo que chamamos `'a.txt'`, a matriz **A** definida na seção anterior. O mesmo pode ser realizado com qualquer tipo de objeto como sumários de análises ou tabelas de distribuição. Os gráficos também são salvos com facilidade em diversos formato, a partir do sistema de menus na interface gráfica do windows, ou também podem ser copiados para a área de transferência para serem editados em outros programas.

2 Carregando pacotes adicionais

Uma das características que torna o sistema R tão interessante é a possibilidade de adicionar novas funções ao pacote base, as quais representam contribuições de pesquisadores em diversas áreas para facilitar a utilização de determinados métodos. Estes pacotes adicionais podem ser baixados como arquivos compactados a partir de algum dos espelhos CRAN ou pode ser baixado e instalado a partir da interface gráfica de usuário disponível para plataformas Windows (O propósito das páginas espelho é aumentar a velocidade de trnsferência de dados, de modo que é interessante sempre escolher um espelho localizado dentro do Brasil). Uma vez que um pacote adicional esteja instalado é possível requerer ao programa a utilização do pacote a partir do menu da interface gráfica ou utilizando os comandos `library()` ou `require()`. Quando o sistema é iniciado, apenas alguns pacotes que fazem parte da base do programa são carregados, o que permite economizar tempo e memória do computador. Deste modo precisamos carregar apenas os pacotes adicionais que serão necessários para aquela sessão específica. Por exemplo, se quisermos utilizar a função de reamostragem que se encontra dentro do pacote `boot`, devemos, antes de executar qualquer comando, carregar o pacote da seguinte maneira:

```
> require(boot)
Loading required package:
boot [1] TRUE
```

3 Funções gráficas no R

3.1 A função plot()

A função gráfica geral mais utilizada e flexível do R é a `plot()`. O resultado desta função depende do modo de vetores utilizados como argumento ou da classe de objetos. Quando trabalhamos com vetores numéricos, como as variáveis `X` e `Y` do `ex1`, o resultado será um diagrama de dispersão para as duas coordenadas.

```
> attach(ex1) > plot(X,Y)
```

O resultado será um gráfico como a Figura 1

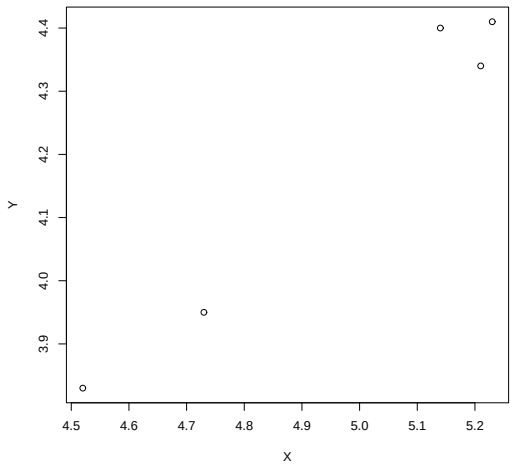

Figura 1:

Por outro lado, se as variáveis sendo utilizadas forem categóricas (modo *character*), o resultado será um tipo de gráfico de barras com as frequências de ocorrência. Um arquivo de exemplo contendo duas variáveis categóricas é o `"categ.txt"`, com 30 observações em que os indivíduos de uma espécie animal qualquer são classificados como machos ou fêmeas, e uma variável Idade, onde os indivíduos são classificados como Jovens ou Adultos. Um pequena parte do arquivo original pode ser vista abaixo:

```
Sexo  Idade
macho  Jovem
macho  Jovem
macho  Adulto
macho  Adulto
femea  Jovem
femea  Jovem
femea  Adulto
femea  Adulto
```

Carregamos o arquivo como um objeto do R e fazemos um gráfico das proporções de machos e fêmeas da seguinte maneira (Figura 2):

```
>cat<-read.table("categ.txt",header=T)
> attach(cat)
>plot(Sexo)
```

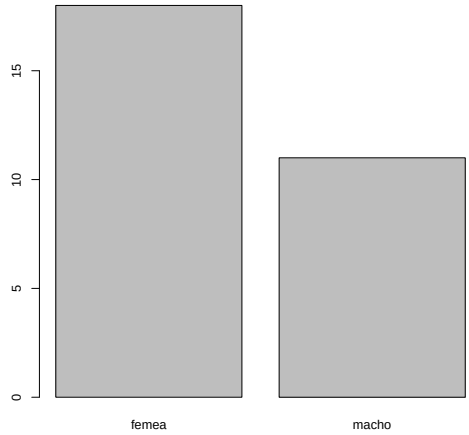

Figura 2:

Se por outro lado estamos interessados na relação de frequências dos diferentes sexos nas diferentes idades, podemos fazer o seguinte gráfico (Figura 3):

```
> plot(Sexo,Idade)
```

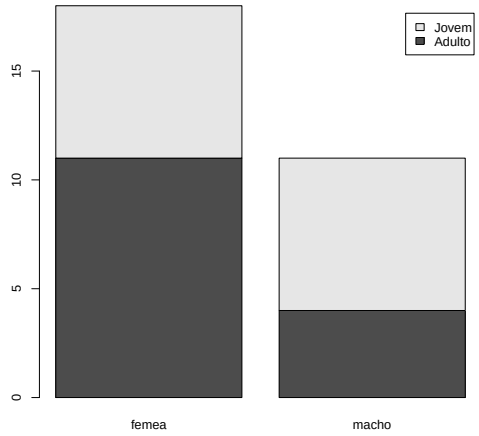

Figura 3:

Esta figura é equivalente a uma tabela de contingência, mostrando que as proporções relativas não são as mesmas para sexo e idade. Existe uma maior proporção de machos jovens que adultos, ao passo que o número de fêmeas adultas é maior que fêmeas jovens.

A função `plot()` apresenta uma série de argumentos que permitem a modificação dos atributos gráficos, como cores, forma (símbolos) e tamanho dos pontos (no diagrama de dispersão), títulos dos gráficos e dos eixos, entre outros. Além deste comando de nível alto *(high level plotting* - que gera o gráfico principal), é possível utilizar comandos de nível baixo *(low level plotting)*, que se sobrepõem ao gráfico principal, permitindo a adição de linhas, pontos adicionais, texto, legendas eixos adicionais.

Com alguns comandos adicionais podemos gerar um gráfico de dispersão baseado no ex1 com alguns elementos a mais que a Figura 1. Por exemplo (Figura 4):

```
> plot(X,Y,xlab="Variável X",ylab="Variável Y",main="Exemplo 1",cex=1.5)
> points(X[1:3],Y[1:3],pch=16,cex=1.5)
```

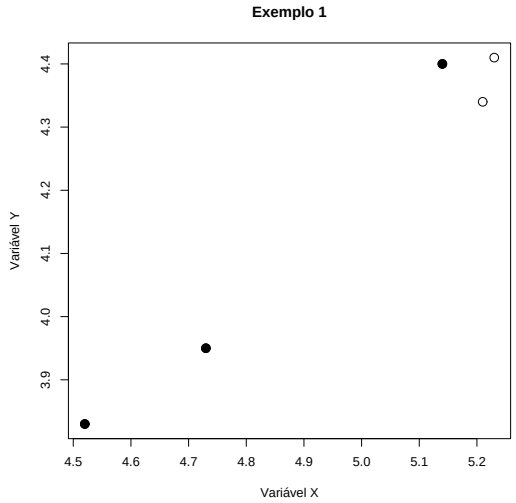

Figura 4:

A primeira linha de comandos corresponde aos comandos de nível alto que geram o gráfico principal, colocando um título principal e nomes nos eixos diferentes dos nomes das variáveis. O parâmetro `cex` *(character expansion)*, permite o aumento no tamanho dos símbolos em uma vez e meia. A segunda linha de comandos (escrita após a execução do primeiro comando) corresponde a um comando de nível baixo para mostrar as três primeiras observações do objeto ex1 como um círculo fechado *(pch=16)*. As cores e os símbolos que podem ser utilizados na geração de gráficos são referenciados a partir de números em um código específico. Estes podem ser encontrados nos arquivos de ajuda ou nos textos (existem vários) sobre a produção e modificação de gráficos no R.

3.2 Histogramas e distribuições de frequência contínuas

A função `"hist"` calcula uma distribuição de frequência para variáveis contínuas e gera um histograma de um conjunto de dados. Parâmetros adicionais da função permitem a determinação da escala a ser utilizada no eixo vertical (se as frequências absolutas ou relativas) e o número de intervalos. Como vimos em classe, a forma do histograma depende do número de intervalos utilizados na discretização da distribuição. Este número pode ser atribuído arbitrariamente pelo usuário ou pode ser obtido por uma função. A opção default para geração do histograma é a

função "Sturges" que calcula o número de intervalos ideal baseada no intervalo de variação. Outras opções são as funções 'Scott' (baseia-se no desvio padrão da distribuição normal ou 'FD' (baseia-se no intervalo interquartil). Para exemplificar esta função, vamos utilizar um conjunto de dados utilizado como exercício na apostila teórica: as distâncias interorbitais de pombos (extraído de Sokal, R.R. & Rohlf, F.J. 1995. Biometry: the principles and practice of statistics in biological research. W.H. Freeman and Co., New York). O conjunto de dados (com uma variável e 40 observações) encontra-se no arquivo "pombos.txt"e deve ser primeiramente carregado como objeto.

```
> pombos<-read.table("pombos.txt")
```

Podemos gerar e gravar a distribuição de frequência com o comando:

```
> pombohist<-hist(pombos$V1)
> pombohist
$breaks
[1] 9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5
$counts
[1] 1 3 7 8 8 10 2 1
$intensities
[1] 0.04999999 0.15000000 0.35000000 0.40000000 0.40000000 0.50000000 0.10000000
[8] 0.05000000
$density
[1] 0.04999999 0.15000000 0.35000000 0.40000000 0.40000000 0.50000000 0.10000000
[8] 0.05000000
$mids
[1] 9.75 10.25 10.75 11.25 11.75 12.25 12.75 13.25
$xname [1]
"pombos$V1"$equidist
[1] TRUE
attr(,"class")
[1] "histogram"
```

Ao executar a função `hist` criamos um objeto da classe *histogram*, que é uma *list*com os componentes listados acima. O componente *breaks* é um vetor com os limites dos intervalos, *counts* é um vetor com as frequências absolutas, `density` contém as frequências relativas, assim por diante. Podemos visualizar o histograma usando a função `'plot()'` para um objeto da classe histogram ou `'hist()'` para um objeto da classe numeric (caso não seja necessário gravar a distribuição de frequência como um objeto à parte para consulta dos números). Vamos então visualizar o gráfico relativo ao histograma (Figura 5):

```
>plot(pombohist)
```

Podemos ainda modificar o eixo vertical de frequências absolutas para relativas utilizando o parâmetro `'freq=F'` e o número de intervalos com o parâmetro `'breaks = ?'`. Uma alternativa ao histograma é o gráfico de densidades, que é mais apropriado à visualização de distribuições contínuas, visto que torna desnecessária a divisão da variável contínua em intervalos discretos, o que minimiza a perda de informação. Para comparar o histograma com o gráfico de densidades, podemos utilizar a função `'density()'` para criar um objeto da classe *density* contendo as

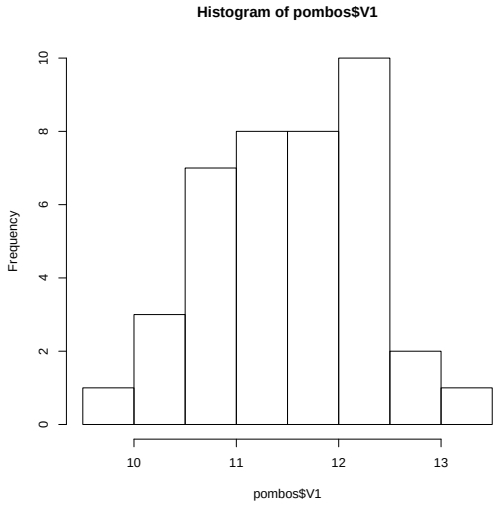

Figura 5:

estimativas de núcleo para as densidades (frequências relativas) ao longo da variável.

```
> pombodens<-density(pombos$V1)
```

Podemos visualizar a curva de densidade sobre o histograma gerando primeiro o gráfico de histograma com um comando de nível alto e adicionando a linha de densidade (Figura 6)com um comando de nível baixo:

```
>hist(pombos$V1,freq=F)
>lines(pombodens)
```

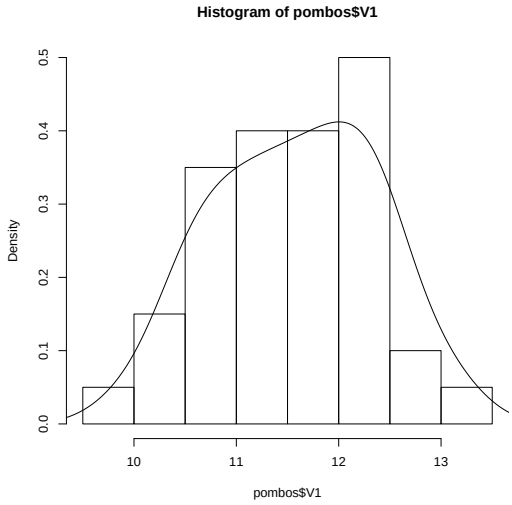

Figura 6:

Podemos também gerar um gráfico de um objeto da classe *density* sem a necessidade do histograma.

3.3 Gráficos de Barras e distribuições de frequência descontínuas ou categóricas.

Quando trabalhamos com variáveis descontínuas ou categóricas, não devemos utilizar a função "hist()". Existem funções específicas para trabalhar estes tipos de variáveis. Primeiramente, precisamos apresentar a função 'table()', a qual gera uma tabela de contingência, um objeto de classe *table*, que sumariza a distribuição de frequência e permite a geração de gráficos correspondentes. No caso de variáveis categóricas, podemos exemplificar com o conjunto de dados 'cat' utilizado anteriormente e que já se encontra como um objeto no *workspace* do R.

```
> table(cat)
 Idade
 Sexo Adulto Jovem
 femea 11 7
 macho 4 7
```

Os gráficos correspondentes para este tipo de distribuição já foram abordados anteriormente. Para o caso de distribuições de variáveis merísticas ou descontínuas, utilizaremos um conjunto de dados alternativo que encontra-se no arquivo 'vanellus.txt', contendo contagens de indivíduos de *Vanellus chilensis* (a ave quero-quero) em uma área de estuário, com 285 observações individuais. Carregamos o arquivo e calculamos a tabela com a distribuição de frequência:.

```
> van<-read.table("Vanellus.txt",header=T)
> table(van)
van
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 24 28 29 31
62 17 51 17 19 17 14 13 10 8 4 9 4 5 3 1 4 2 1 4 1 1 1 2 2 1 33 36
38 43 52 53 55 58 61 98 1 1 2 2 1 1 1 1 1 1
```

Nas linhas acima encontramos as classes de frequência na parte superior e as frequências na parte inferior. Para gerar um gráfico desta distribuição, a melhor alternativa é a função 'barplot()', que pode ser utilizada para gerar um gráfico a partir de uma tabela como a calculada acima. Neste caso, os comandos deveriam ser:

```
>barplot(table(van$N), xlab= "Número de Indivíduos", ylab="Frequência",
main=expression( "Ocorrência de"* italic( "Vanellus chilensis") *
"em um Estuário"))
```

Repare que para misturar caracteres em fonte normal em itálico em um título é necessário utilizar a função "expression()", a qual permite a geração de gráficos com fórmulas complicadas no título ou nos eixos. O gráfico gerado com o comando acima seria como o mostrado na Figura 7.

Outros tipos de gráficos serão vistos à medida que progredimos nas análises.

4 Estatística Descritiva

4.1 Funções numéricas

Como visto anteriormente, o R executa funções sobre objetos utilizando comandos. Os vetores (variáveis analisadas estatisticamente) podem ser usados nas expressões aritméticas, nas quais as

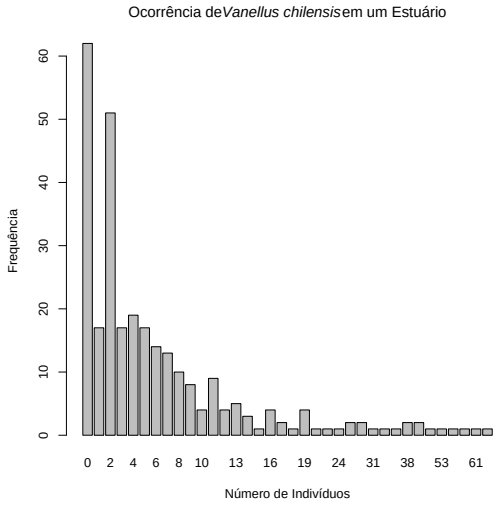

Figura 7:

operações podem ser feitas elemento por elemento. Os operadores elementares da aritmética já foram vistos anteriormente e algumas outras funções também são acrescentadas a estas como, `log()`, `exp()`, `sin()`, `cos()`, `tan()`, `sqrt()`. Qualquer estatística pode ser implementada diretamente como uma função no R, bastando entender como escrever a função que queremos. Utilizando como exemplo, o objeto ex1 utilizado anteriormente, podemos calcular a média deste com o seguinte cálculo:

```
> attach(ex1)
> sum(X)/length(X)
[1] 4.966
```

Onde a função `sum()` executa o somatório dos elementos do vetor X e a função `length()` conta quantos elementos existem no vetor. Podemos ainda definir uma função geral para o cálculo da média sempre que quisermos, usando a função `function()`, da maneira exemplificada abaixo:

```
> media<-function(x)sum(x)/length(x)
> media(X)
[1] 4.966
> media(Y)
[1] 4.186
```

Uma vez definida a função, esta passa a existir como um objeto no workspace e pode ser chamada para executar sobre qualquer variável. A definição de funções personalizadas é interessante quando precisamos calcular uma estatística que não esteja previamente programada no R. No caso da média, já existe uma função (`mean()`) que pode ser utilizada normalmente:

```
> mean(X)
[1] 4.966
```

Na tabela abaixo encontramos as principais funções utilizadas para o cálculo de estatísticas descritivas no R.

Função	*Estatística Calculada*
`mean(X)`	Média de X
`var (X)`	Variância de X
`sd (X)`	Desvio padrão de X
`median (X)`	Mediana de X
`min (X) e max (X)`	Valores mínimo de máximo de X
`range`	Valores mínimo e máximo (em um mesmo conjunto de resultados)
`IQR`	Intervalo interquartis (diferença entre os quartis inferior e superior)

Podemos conferir os valores calculados para a variável X no exemplo 1:

```
> var(X)
[1] 0.10353
> sd(X)
[1] 0.3217608
> median(X)
[1] 5.14
> min(X)
[1] 4.52
> max(X)
[1] 5.23
> range(X)
[1] 4.52 5.23
> IQR(X)
[1] 0.48
```

O cálculo de médias ponderadas pode ser efetuado através da função `weighted.mean(x,w)`, onde definimos os valores a serem utilizados no cálculo da média (x) e os pesos a serem atribuídos a cada valor (w). Vamos construir um vetor contendo as médias das variáveis X e Y do exemplo 1 usando a função `c(obj1,obj2, etc...)`, que combina objetos em um novo vetor a ser formado. Definimos também o vetor w com os pesos a serem atribuídos às diferentes médias, sendo que o peso de X será o dobro do peso de Y:

`> mex1<-c(mean(X),mean(Y)) > w<-c(2,1) > weighted.mean(mex1,w) [1] 4.706`

Se por outro lado quisermos atribuir mais peso à média de Y, podemos inverter o vetor de pesos:

`> w<-c(1,2) > weighted.mean(mex1,w) [1] 4.446`

Podemos reparar que o valor de resultado é um pouco menor, visto que a média de Y é menor que a média de X. Apesar de não existir uma função específica para o cálculo do coeficiente de variação, é muito simples calcular ou definir uma função para o cálculo repetitivo desta estatística dentro do ambiente R, como vimos anteriormente. Basta escrever a função da seguinte maneira:

```
> cv<-function(x)(sd(x)/mean(x))*100
> cv(X)
[1] 6.479275
> cv(Y)
[1] 6.56539
```

```
> cv(pombo$V1)
[1] 6.882348
> cv(van$N)
[1] 163.9570
```

Note que o coeficiente de variação para o número de aves por evento de observação (`van$N`) é muito grande, sugerindo que não devemos utilizar média e desvio padrão para descrever esta distribuição. Podemos obter um sumário numérico descritivo para qualquer variável ou grupo de variáveis, usando a função `summary()`. Este comando retorna os valores do mínimo, quartil inferior (1st Qu, mediana, média, quartil superior (3rd Qu) e máximo. Se o argumento principal da função for um *data.frame* com mais de uma variável, os sumários são produzidos para cada variável, como demonstrado abaixo:

```
> summary(ex1)
       X               Y
 Min.   :4.520   Min.   :3.830
 1st Qu.:4.730   1st Qu.:3.950
 Median :5.140   Median :4.340
 Mean   :4.966   Mean   :4.186
 3rd Qu.:5.210   3rd Qu.:4.400
 Max.   :5.230   Max.   :4.410
```

Se estivermos interessados em percentis específicos, podemos utilizar a função `quantile()`, (quantis e percentis são sinônimos), especificando além do vetor de dados, quais percentis devem ser calculados. Por exemplo, se quisermos calcular os percentis de 10%, 25%, 50%, 75% e 90%, devemos escrever:

```
> quantile(X,c(0.1,0.25,0.5,0.75,0.90))
10% 25% 50% 75% 90%
4.604 4.730 5.140 5.210 5.222
```

Este comando será muito útil e importante para determinarmos valores críticos de estatísticas e intervalos de confiança para distribuições de estatísticas a partir de procedimentos de reamostragem. Uma medida de dispersão não paramétrica que pode ser utilizada para a descrição de distribuições amostrais em conjunto com a mediana e quantis é o desvio absoluto mediano (median absolute deviation). Esta é a mediana dos desvios absolutos em relação à mediana, multiplicada por uma constante. calculada como:

$$mad = M\left(X_i - M\left(X\right)\right) \times 1,4826$$

onde M representa a mediana e a constante 1,4826 é utilizada para colocar esta medida de dispersão na mesma escala do desvio padrão. A constante é definida de modo que, caso a distribuição de X seja perfeitamente normal, o desvio padrão e a mad apresentarão o mesmo valor.

```
> mad(X)
[1] 0.133434
> sd(X)
[1] 0.3217608
```

Podemos comparar com uma variável com maior número amostral para observar que os dois valores tendem a se aproximar mais

```
> mad(pombo$V1)
[1] 0.96369
> sd(pombo$V1)
[1] 0.797492
```

4.2 Funções gráficas (boxplot)

Até o momento, vimos como gerar sumários descritivos de variáveis aleatórias a partir de estatísticas numéricas e gráficos mostrando distribuições de frequência, como o histograma e o diagrama de barras. Uma ferramenta gráfica muito importante para a descrição e principalmente para a comparação de distribuições associadas a grupos diferentes é o gráfico de boxplot este gráfico mostra a distribuição de acordo com estatísticas descritivas não paramétricas, como a exemplificado na Figura 8.

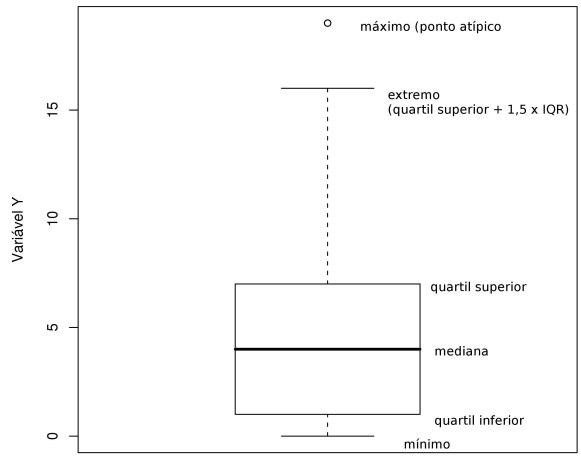

Figura 8: Boxplot ilustrativo (`boxplot(van$N[50:200], ylab="Variável Y"`)

É possível estabelecer uma série de parâmetros para a função `boxplot()`, que permitem uma personalização do gráfico de acordo com a necessidade. A linha horizontal mais escura marca sempre a posição da mediana. A caixa se estende no intervalo interquartil. A linha tracejada (whiskers) pode marcar, diferentes estatísticas, dependendo do argumento range para a função `boxplot()`. Quando um valor positivo de range é utilizado (o default é 1.5), a linha tracejada marcará valores extremos, não maiores que range x intervalo interquartil. A partir destes ponto extremo, as observações com valores mais altos são marcadas com pontos, como valores atípicos. Caso não tenhamos interesse em pontos atípicos, utilizamos range = 0, e a linha tracejada marcará os pontos de máximo e mínimo da amostra. Podemos comparar distribuições de variáveis ou de grupos diferentes em uma mesma variável utilizando os diagramas boxplot. Quando queremos comparar variáveis que se encontrem dentro de um mesmo data frame, ou mesmo em data frames

diferentes, simplesmente escrevemos os nomes das variáveis a serem utilizadas no gráfico, uma após a outra, como os primeiros argumentos:

```
> boxplot(X,Y,names=c("Variável X","Variável Y"),ylab="Variáveis")
```

Lembrando que o *data.frame* ex1 já está anexado (`attach()`), não sendo necessário utilizar o nome do objeto principal. Este comando produzirá um gráfico como a Figura 9.

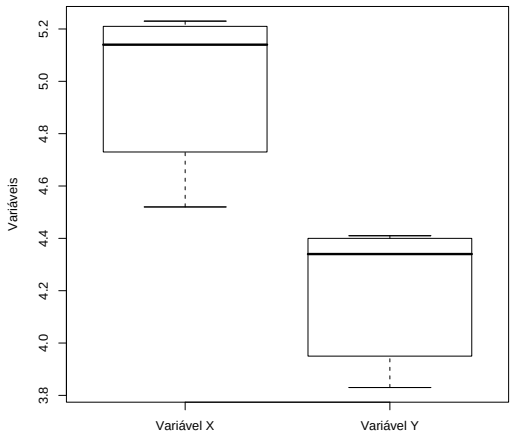

Figura 9:

Quando o propósito do gráfico é comparar a distribuição de dois ou mais grupos diferentes para uma mesma variável, podemos utilizar a função `boxplot()` para fazer um gráfico de uma função. Para exemplificar este comando, vamos utilizar um novo conjunto de dados a ser utilizado como exemplo: o comprimento padrão de machos e fêmeas de *Poecilia vivipara* de uma mesma população. Este conjunto de dados encontra-se no arquivo poec-boxp.txt e deve ser carregado no *workspace* da maneira conhecida:

```
> poecb<-read.table("poec-boxp.txt",header=T)
```

Este aquivo contem duas variáveis, a primeira coluna (CP) contém os comprimentos dos indivíduos e a segunda coluna (S) contém o sexo dos indivíduos. Para gerar o gráfico boxplot comparando os comprimentos de machos e fêmeas vamos utilizar os seguintes comandos:

```
> boxplot(poecb$CP~poecb$S, names=c("Fêmeas","Machos"),
ylab="Comprimento padrão")
```

Note que ao invés de simplesmente colocar o nome de variáveis utilizamos simplesmente uma fórmula (poecb$CP~poecb$S), que significa comprimento=sexo (vamos ver novamente este tipo de fórmulas quando quisermos analisar modelos lineares). O gráfico produzido encontra-se na Figura 10.

O mesmo gráfico poderia ser obtido diretamente sem o auxílio de uma fórmula, com os seguintes comandos:

```
> boxplot(poecb$CP[1:60], poecb$CP[61:120], names=c("Fêmeas","Machos"),
ylab="Comprimento padrão")
```

já que sabemos que os 60 primeiros indivíduos do conjunto de dados são fêmeas e os últimos

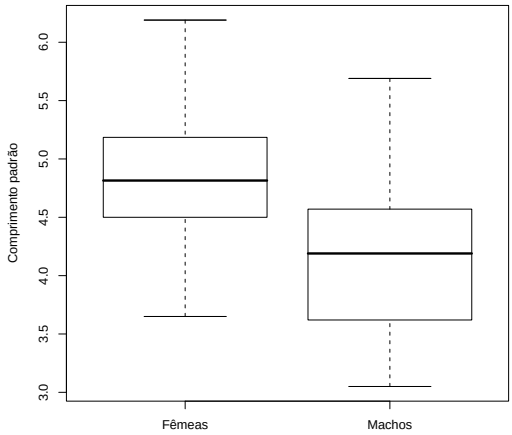

Figura 10: Diagrama *boxplot* comparando o comprimento de machos e fêmeas

60 são machos.

Para uma comparação mais eficiente é possível utilizar o argumento `notch` para gerar uma chanfradura em cada caixa que representa um intervalo de confiança para as medianas. quando as chanfraduras nas caixas relativas aos diferentes grupos não se sobrepõem, temos "fortes evidências"de que as medianas são diferentes (Chambers, J. M., Cleveland, W. S., kleiner, B. and Tukey, P. A. 1983. Graphical Methods for Data Analysis. Wadsworth Brooks/Cole). Podemos então repetir os comandos para comparar o comprimento de machos e fêmeas usando o argumento `notch` desta vez:

```
> boxplot(poecb$CP poecb$S, names=c("Fêmeas","Machos"),
ylab="Comprimento padrão",notch=T)
```

O gráfico gerado a partir deste comando encontra-se na Figura 11.

Podemos perceber que as chanfraduras não se sobrepõem no exemplo acima. o que pode ser considerado um "teste"gráfico para as medianas. Como veremos adiante, nem sempre é necessário um teste numérico e um valor de probabilidade para inferir a significacia de uma diferença É importante conhecer todas as opções que podemos personalizar neste e em outros gráficos para que possamos aproveitar todos os recursos disponíveis do programa e gerar gráficos que mostrem a informação de maneira eficiente.

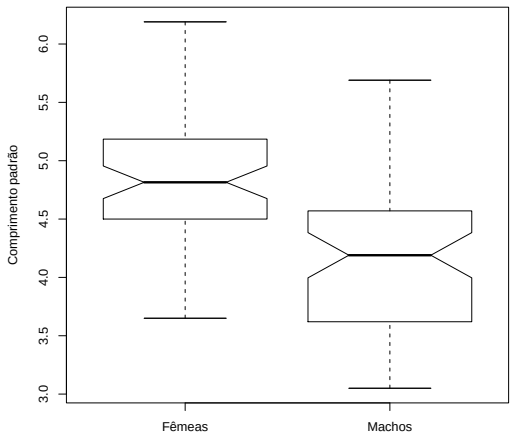

Figura 11: Diagrama *boxplot* comparando comprimentos de machos e fêmeas, mostrando os intervalos de confiança para as medianas nas chanfraduras das caixas.

5 Distribuições de probabilidade no R

O ambiente R apresenta uma série de distribuições teóricas de probabilidade prontas como funções que podem ser utilizadas para o cálculo de probabilidades, o cálculo de percentis associados a probabilidades e a geração de números aleatórios (importante para trabalharmos com simulações). O R possui 18 funções de probabilidade. Nesta apostila examinaremos apenas as mais utilizadas por nós. Algumas distribuições e suas respectivas funções estão listadas na tabela abaixo:

Distribuição	Função	Argumentos adicionais	Argumentos *default*	Acesso à ajuda
Binomial	`binom`	`size(n), prob(p)`		`help(Binomial)`
Poisson	`pois`	λ (média)		`help(Poisson)`
Normal	`norm`	`mean(`μ`)`, `sd(`σ`)`	`mean=0, sd=1`	`help(Normal)`
Student t	`t`	`df` (graus de liberdade)		`help(TDist)`
F	`f`	`df1,df2`		`help(FDist)`
Qui-quadrado	`chisq`	`df`		`help(Chisquare)`
Uniforme	`unif`	`min(a),max(b)`	`min=0, max=1`	`help(Uniform)`

Cada uma destas distribuições deve ser utilizada com um prefixo (d, p, q ou r) para determinar qual vai ser o valor de resultado. Se utilizamos `dfunção(x,...)`, teremos como resultado a densidade para os determinados valores de `x`. Isto é útil se quisermos contruir um gráfico ou se quisermos saber a probabilidade dentro de um determinado intervalo (para funções contínuas) ou determinado valor (para funções discretas). Se utilizamos `pfunção(q,...)`, onde q representa um percentil da distribuição, o resultado é a probabilidade (área sob a curva no caso de funções contínuas) de encontrar um valor igual ou menor que o percentil opservado. Com o prefixo `qfunção(p,...)`, o resultado da função é o percentile referente à probabilidade p especificada como primeiro argumento. As funções utilizadas com os prefixos p e q podem ser utilizadas como alternativa às tabelas estatísticas encontradas em livros, gerando inclusive resultados mais satisfatórios, pois permitem o cálculo de valores de probabilidade exatos, ao invés do resultado mais utilizado em testes estatísticos ($P < 0.05$). Por último, o prefixo r (`rfunção(n,...)`) retorna números aleatórios baseados na respectiva função de probabilidade, o que pode ser utilizado para gerar conjuntos de dados que são amostras de parâmetros conhecidos. Estes números aleatórios são muito utilizados em simulações. Mas podem ser úteis para o usuário de estatística comum, como veremos em alguns exemplos a seguir. Vamos examinar algumas distribuições em maior detalhe e exemplificar a utilização das funções.

5.1 Distribuição binomial

A distribuição binomial pode ser acessada no ambiente R a partir das funções `dbinom()`, `pbinom()`, `qbinom()` e `rbinom()`. Os argumentos a serem utilizados variam de acordo com o prefixo, visto que se queremos saber uma determinada densidade ou probabilidade, precisamos fornecer o valor da variável aleatória a ser estimado. Por outro lado, se queremos saber o percentil (ou valor da variável aleatória) referente a uma determinada probabilidade, ou gerar números aleatórios, precisamos fornecer, além dos argumentos normais, o número de amostras que devem ser geradas.

Vamos examinar alguns exemplos para ver melhor como funciona. Como vimos em um exemplo na aula teórica, se estamos trabalhando com uma hipótese de que uma determinada população apresenta frequências de ocorrências de machos e fêmeas iguais ($p = q = 0{,}5$), podemos calcular a probabilidade de obter uma amostra de 17 indivíduos com 3 machos e 14 fêmeas. Estabelecendo que o nosso evento de interesse é o número de machos, podemos obter a probabilidade exata de uma amostra de 17 observações e 3 machos com o prefixo d:

```
> dbinom(3,size=17,0.5)
[1] 0.005187988
```

Por outro lado, se quisermos saber a probabilidade de obter uma amostra com 3 ou menos machos (mais extrema que o observado) usamos o prefixo p

```
> pbinom(3,size=17,0.5)
[1] 0.006362915
```

Para calcular o número de machos por amostra que corresponderia a uma determinada probabilidade, utilizamos o prefixo q

```
> qbinom(0.05,size=17,prob=0.5)
[1] 5
```

Quer dizer que uma amostra com 5 (ou menos) machos e 12 fêmeas apresentaria uma probabilidade de 0.05 aproximadamente de ocorrer. Na verdade, a probabilidade exata de amostras com 5 ou menos machos seria 0,0717, mas como o resultado tem que ser um número exato, o resultado é arredondado (o resultado será 5 para probabilidades fornecidas entre 0,03 e 0,07. Se quisermos utilizar a distribuição binomial para simular uma amostragem aleatória, podemos utilizar o prefixo r, que significa *random numbers* (números aleatórios).Para isso devemos especificar quantas amostras queremos tomar (`n`) e o número total de observações por amostras (`size`).

```
> x<-rbinom(1000,size=17,prob=0.5)
```

Com este comando criamos um objeto x representando um conjunto de 1000 amostras em que foram contados os números de machos em cada. Um gráfico de barras mostrando a distribuição amostral de x (Figura 12).

Esta amostra aleatória simulada pode ser utilizada de várias maneiras. Uma delas é a comparação com a amostra observada. Se observarmos a distribuição de frequência das 1000 amostras:

```
> table(x)
x
3 4 5 6 7 8 9 10 11 12 13 14 15
6 15 53 112 139 154 187 159 100 44 26 4 1
```

podemos perceber que das 1000 amostras simuladas, apenas 6 apresentaram 3 machos (nenhuma amostra apresentou menos de 3 machos). Podemos, baseados nesta simulação, atribuir a probabilidade de uma amostra como esta como 6/1000=0,006. Se aumentarmos o número de simulações, nos aproximamos da probabilidade teórica da distribuição. Este tipo de procedimento se encaixa em um conjunto de técnicas chamadas de métodos de Monte Carlo, onde a simulação computacional com modelos estatísticos gera distribuições que podem ser utilizadas para o teste de hipóteses estatísticas comparando com dados reais.

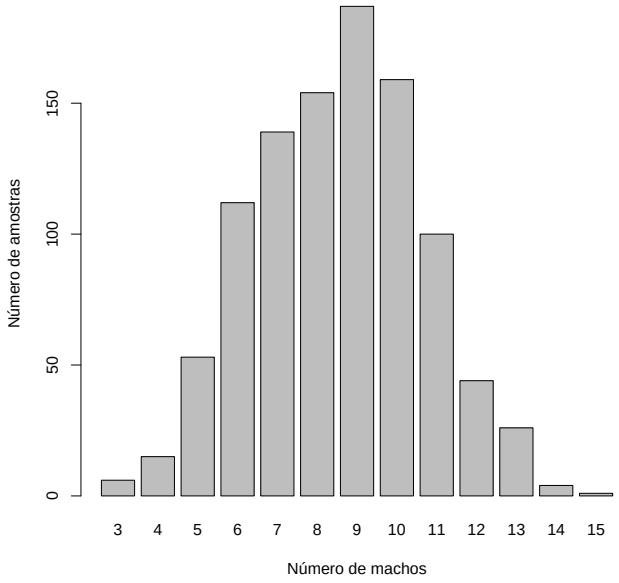

Figura 12: Gráfico de barras para 1000 amostras simuladas pela função `rbinom`

5.2 Distribuição normal

Podemos acessar a distribuição normal no ambiente R pelas funções `dnorm`, `pnorm`, `qnorm` e `rnorm`, da mesma maneira que utilizamos a distribuição binomial. As três primeiras funções podem ser utilizadas em substituição à tabela z, visto que nos permite o cálculo de probabilidades com mais precisão (mais casas decimais) que a tabela. O teste de Kolmogorov-Smirnov pode ser acessado através da função "`ks-test()`", onde precisamos determinar a variável original e a distribuição teórica a ser utilizada na comparação. Por exemplo, para comparar a distribuição observada nos dados dos pombos com a distribuição normal, escrevemos:

```
> ks.test(pombos$V1,"pnorm",mean=mean(pombos$V1),sd=sd(pombos$V1))
Aviso em ks.test(pombos$V1, "pnorm", mean = mean(pombos$V1), sd = sd(pombos$V1))
:
 cannot compute correct p-values with ties
 One-sample Kolmogorov-Smirnov test
data:  pombos$V1
D = 0.0801, p-value = 0.9598
alternative hypothesis:  two.sided
```

Repare que é necessário fornecer os parâmetros média e desvio padrão da distribuição normal como argumentos adicionais à função. O valor de probabilidade é alto, de modo que não devemos rejeitar a hipótese nula aqui de que a distribuição observada se assemelha à normal teórica esperada. Se tentarmos ajustar a distribuição normal à contagem de indivíduos em `van$N`, temos que escrever:

```
> ks.test(van$N,"pnorm",mean=mean(van$N),sd=sd(van$N))
```

```
 One-sample Kolmogorov-Smirnov test
data:  van$N
D = 0.271, p-value < 2.2e-16
alternative hypothesis:  two.sided
```

Como esperado para uma distribuição amostral tão assimétrica, encontramos uma probabilidade muito baixa (menor que 0,00000000000000022) de observar tal distribuição amostral a partir de uma população com distribuição normal. Se quisermos ajustar esta população a uma distribuição de Poisson, escrevemos:

```
> ks.test(van$N,"ppois",lambda=mean(van$N))
 One-sample Kolmogorov-Smirnov test
data:  van$N
D = 0.4307, p-value < 2.2e-16
alternative hypothesis:  two.sided
```

A função dnorm(x,...) pode ser também utilizada para construir um gráfico com a curva normal para uma distribuição de qualquer variável contínua x. Contruir um gráfico para uma função é um procedimento que pode ser realizado por vários caminhos, uma vez que se compreenda o as operações que estão sendo realizadas. Um maneira econômica é utilizar a função `curve()`, que utiliza a função `plot()`, mas deve ser utilizada especificamente para desenhar curvas atribuídas a funções. Estas curvas podem ser contruídas em um gráfico sozinhas ou adicionadas a um gráfico pré-existente. Se quisermos gerar um gráfico com a curva normal padrão, podemos simplesmente escrever:

```
> curve(dnorm(x),from=-3,to=3,lwd=3)
```

E o gráfico é gerado como na Figura 13.

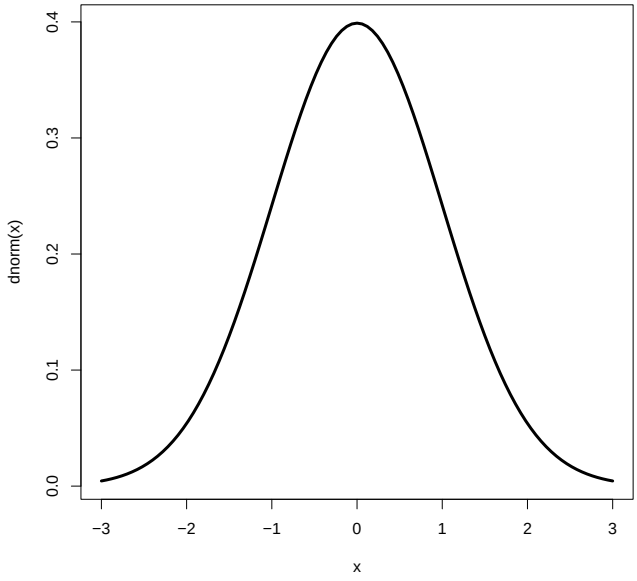

Figura 13: Curva normal padrão ($\mu = 0$, $\sigma = 1$)

Mas ao invés de fazer um gráfico sozinho, podemos estar interessados em gerar um gráfico com a curva normal esperada sobreposta ao histograma de uma variável observada. Usando como exemplo os dados dos pombos (que já está gravado no *workspace como um data.frame*), escrevemos os seguintes comandos (repare que o argumento `add=T` na função `curve()` permite que a mesma se comporte como uma comando de nível baixo e seja adicionada a um gráfico pré-existente - Figura 14):

```
> mean(pombos$V1)
[1] 11.5875
> sd(pombos$V1)
[1] 0.797492
> hist(pombos$V1,xlab="Distância interorbital de pombos", ylab="Densidade",freq=F,
main= NULL)
> curve(dnorm(x,mean=11.5875,sd=0.797492),lwd=3,add=T)
```

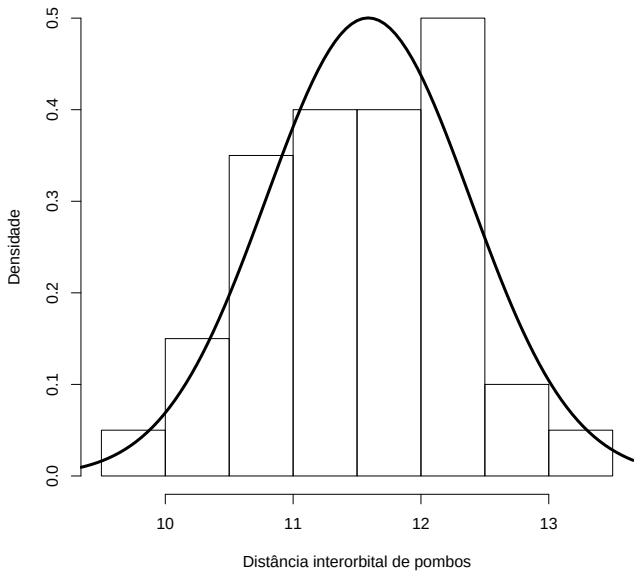

Figura 14: Histograma para a distância interorbital de pombos com a curva normal esperada sobreposta

Para calcularmos a probabilidade de uma determinada observação apresentar um valor igual ou maior que x, usamos o prefixo p na função normal e encontramos a probabilidade como em uma tabela estatística, a partir da função de distribuição normal cumulativa (a função `pnorm()` corresponde à integral da função `dnorm()`):

```
> pnorm(1.96)
[1] 0.9750021
```

O resultado mostra que a área abaixo da curva para a região menor ou igual a 1.96 desvios padrões acima da média equivale a 97,5% do total. Como esta distribuição é simétrica, se quisermos saber a probabilidade de uma observação ser igual ou maior que 1.96 desvios acima da média,

devemos escrever:

```
> pnorm(-1.96)
[1] 0.02499790
```

O resultado mostra que 2,5% da curva encontram-se abaixo de 1.96 desvios. Para calcular a probabilidade associada a um determinado intervalo, simplesmente escrevemos comandos que diminuam uma da outra. Por exemplo, a proporção da área abaixo da curva entre no intervalo entre -1.96 e 1.96 corresponde a:

```
> pnorm(1.96)-pnorm(-1.96)
[1] 0.9500042
```

Ou seja, o intervalo de 95% de confiança. Mesmo que estejamos interessados na distribuição normal de uma variável não padronizada (com média e desvio padrão diferentes de 0 e 1), podemos calcular estes intervalos de probabilidade utilizando os parâmetros `mean=` e `sd=` na função normal. Por exemplo, para a distância interorbital dos pombos, sabemos que a média amostral é 11,5875mm e o desvio padrão amostral é 0,797492mm. Podemos (na escala original da variável) calcular a probabilidade de obter um valor tão diferente da média (igual ou menor) que 10mm.

```
> pnorm(10,mean=11.5875,sd=0.797492)
[1] 0.02326158
```

para calcular valores de percentis específicos baseados na distribuição normal, usamos o prefixo q, dizendo qual a probabilidade associada a este percentil. Por exemplo, na distribuição dos pombos citada acima, o valor da variável que corresponde a 0,025 da distribuição esperada é:

```
> qnorm(0.025,mean=11.5875,sd=0.797492)
[1] 10.02444
```

Se quisermos saber o valor da variável correspondente a 97.5% da distribuição teórica temos:

```
> qnorm(0.975,mean=11.5875,sd=0.797492)
[1] 13.15056
```

Podemos dizer então que esperamos que haja 95% de chance de amostrar um indivíduo da população original de pombos cuja distância interorbital esteja no intervalo entre 10,02 e 13,15 (os limites inferior e superior do intervalo de confiança para a população).

O prefixo r permite a geração de números aleatórios baseados na distribuição normal. Se quisermos gerar uma amostra com 1000 indivíduos baseada na média e no desvio padrão da amostra dos pombos (como se estes fossem parâmetros da população *virtual* original, escrevemos:

```
> x<-rnorm(1000,mean=11.5875,sd=0.797492)
> hist(x)
```

e o gráfico pode ser visto na Figura 15.

Esta função de simulação pode ser utilizada até mesmo dentro de modelos estocásticos mais complexos em que há a necessidade de gerar números aleatórios em uma função normal.

5.3 Outras distribuições

Como mostrado no início desta parte, existem várias outras distribuições teóricas prontas no ambiente R. Podemos utilizar as mesmas para o calculo de probabilidades associadas a estatísticas inferenciais, como a distribuição t de Student, F e *Qui-quadrado*. Estas funções servem como alternativas interessantes às tabelas estatísticas encontradas em livros, visto que permitem o cálculo

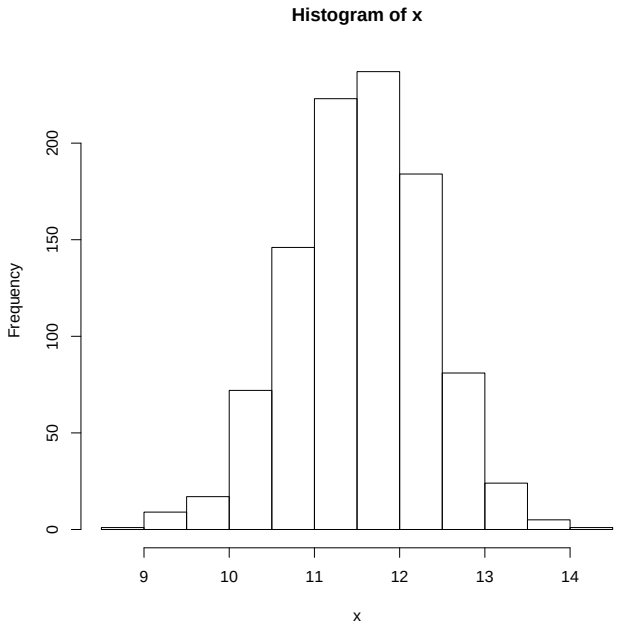

Figura 15: Histograma para uma amostra simulada de 1000 observações baseada na função normal

de probabilidades exatas. É claro que a maioria dos testes estatísticos inferenciais que realizaremos já trazem estas funções embutidas de modo apropriado, mas caso seja necessário conferir um resultado obtido fora do R, ou gerar um teste personalizado, devemos nos referenciar diretamente a estas distribuições. Algumas destas distribuições serão utilizadas novamente nos momentos apropriados.

5.4 Amostragem aleatória e simulação

A distribuição uniforme não apresenta um interesse direto para estatísticas inferenciais, no entanto, é muito útil para a geração de números aleatórios como em um sorteio de números. Os argumentos principais para esta distribuição são o número de observações e o intervalo a ser amostrado (o *default* fica entre 0 e 1). Se quisermos gerar um sorteio de 10 números entre 0 e 100 escrevemos:

```
>x<-runif(10,min=0,max=100)
> x
[1] 65.115452 3.894268 60.213677 7.734489 92.179522 92.084185 77.232314
[8] 73.094990 37.257027 47.839611
```

Por outro lado, o comando `sample()` pode ser mais geral para alguns tipo de sorteio. Imagine que temos uma comunidade onde estamos estudando a diversidade de espécies. Em uma amostra original encontramos as seguintes proporções de espécies (baseado nas abundâncias relativas):

sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9	sp10
10%	5%	7%	12%	13%	24%	9%	5%	7%	%8

Criamos uma comunidade virtual (de espécies) utilizando estas proporções, de modo que a probabilidade de em uma amostragem aleatória da comunidade, a probabilidade de observar a espécie 1 seja 10/100=0,1. Esta população virtual será um objeto no *workspace* do R, o qual chamaremos `comun`:

```
>comun<-c(rep("sp1",10), rep("sp2",5), rep("sp3",7), rep("sp4",12), rep("sp5",13),
rep("sp6",24), rep("sp7",9), rep("sp8",5), rep("sp9",7), rep("sp10",8))
> table(comun)
comun
 sp1 sp10 sp2 sp3 sp4 sp5 sp6 sp7 sp8 sp9
 10 8 5 7 12 13 24 9 5 7
```

O comando `rep(obj,n)` repete o objeto ou valor determinado n vezes e a função combine (`c()`) adiciona todas as repetições a um único vetor. Simularemos agora uma amostra de 10 indivíduos desta comunidade, utilizando a função `sample()`:

```
> comun.samp<-sample(comun,10,replace=T)
> table(comun.samp)
comun.samp
 sp1 sp10 sp2 sp4 sp5 sp6 sp7
 1 2 1 1 2 2 1
```

É possível notar que o número de espécies (riqueza) na amostra com 10 indivíduos é menor que o existente na comunidade *virtual* original. Isto acontece porque o tamanho amostral é pequeno. Se quisermos comparar a riqueza de espécies em duas localidades precisamos nos preocupar em padronizar o esforço amostral. Este procedimento é chamado curva de rarefação e permite a comparação de riquezas (ou índices de diversidade e dominância) em locais com amostras de diferentes tamanhos. Se reamostrarmos a mesma comunidade com amostras de diferentes tamanhos (1000 vezes em cada), encontraremos uma curva parecida com a Figura 16.

É claro que a curva que encontramos é um sorteio para cada tamanho amostral. Por outro lado, percebemos aqui que se estivéssemos comparando uma amostra de 10 indivíduos em um local com uma amostra de 50 indivíduos em outro, encontraríamos valores diferentes para a riqueza de espécies, causada por uma diferença no tamanho das amostras. Com pouco conhecimento de programação, é possível gerar um processo recursivo para criar os intervalos de confiança mostrados na figura, baseados em reamostragens a partir da população *virtual* original (a qual será baseada em uma grande amostra de uma comunidade). Estes intervalos de confiança podem ser utilizados para a comparação dos índices de diversidade ou riqueza entre locais.

5.5 Ajuste de distribuições a amostras

Como vimos na parte teórica, a primeira maneira de comparar um conjunto de dados com uma expectativa teórica é gerando um gráfico com as distribuições observada e esperada. Já vimos anteriormente como comparar uma distribuição de uma variável contínua (histograma) com a distribuição normal esperada. Podemos comparar dados com outros tipos de distribuição. Por exemplo, o objeto `"van"`, que já se encontra no *workspace*, contém uma variável (`"N"`) com contagens de indivíduos de uma espécie de ave em um estuário. Podemos construir um gráfico de barras

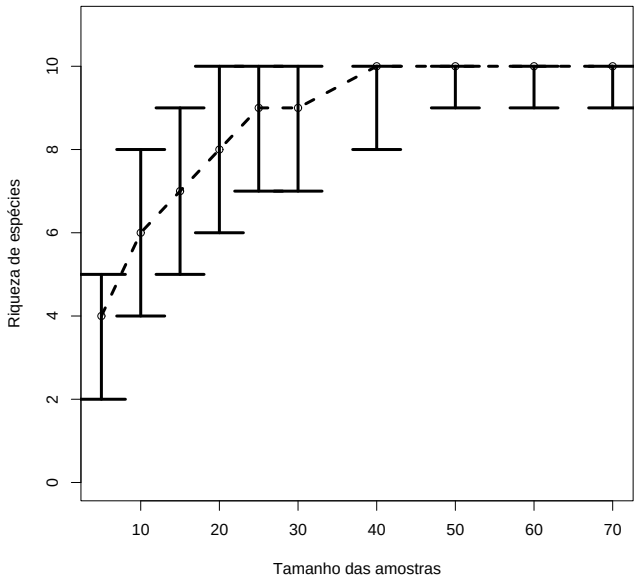

Figura 16: Curva de rarefação para amostras de diferentes tamanhos, com 1000 repetições em cada.

com a distribuição desta variável e adicionar uma linha que corresponde à distribuição esperada se a variável seguir a expectativa da distribuição de Poisson da seguinte maneira:

```
> barplot(table(van$N),xlab="Número de indivíduos",ylab="Frequência")
> x<-0:62
> lines(x,dpois(x,6.936842)*285,type="s",lwd=3)
```

Onde 6.936842 é a média de indivíduos por observação na variável N (o argumento que define a forma da distribuição de Poisson) e 285 é o número de observações na variável (necessário para transformar as probabilidades em frequências esperadas). O gráfico produzido é igual à Figura 17.

No caso da distribuição normal, sabemos que uma ferramenta importante para a comparação com a distribuição normal é o gráfico de quantis esperados e observados. Este gráfico para a distribuição da distância interorbital de pombos pode ser criado a partir da função gráfica `qqnorm()`. Esta função especificamente compara a distribuição de uma variável com a distribuição normal. Se quisermos comparar a distribuição de duas variáveis diferentes, podemos utilizar a variante `qqplot()`. Para construir o gráfico de quantis para a variável dos pombos escrevemos:

```
> qqnorm(pombos$V1)
```

E o resultado é a figura 18.

Podemos adicionar uma linha reta entre o primeiro e o terceiro quartil com o comando `qqline`:

```
> qqline(pombos$V1,lwd=3)
```

gerando um gráfico como na Figura 19.

Esta linha reta auxilia na identificação do tipo de desvio da normalidade. Podemos reparar que a distribuição de pontos assume ligeiramente uma forma de S, indicando uma distribuição assimétrica. Ainda com relação à normalidade, as estatísticas de assimetria e curtose (g_1 e g_2) não

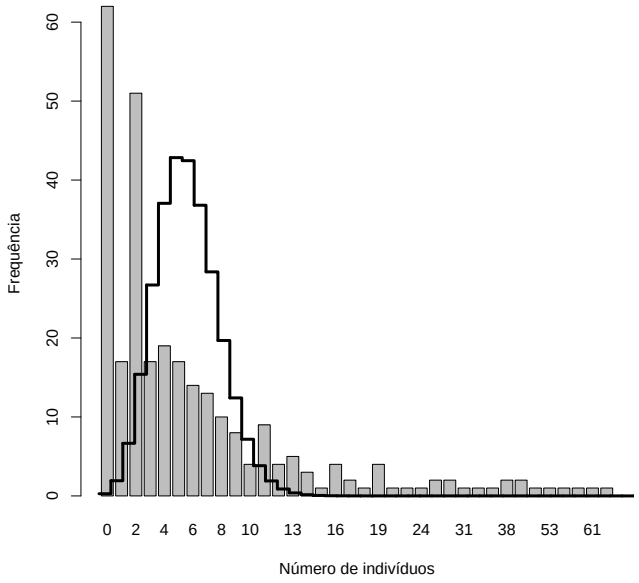

Figura 17: Comparação entre frequências observadas e esperadas para o número de indivíduos de *Vanellus chilensis* em um estuário de acordo com a distribuição de Poisson.

possuem funções específicas no R. No entanto, podemos utilizar o nosso conhecimento acumulado com relação à criação de funções personalizadas no R e as fórmulas fornecidas na apostila teórica para criar funções que estimem a assimetria e a curtose de uma amostra. Para o cálculo da assimetria podemos utilizar a seguinte função:

```
gamma1<-function(x) {
n=length(x)
m=mean(x)
s=sd(x)
g=(n*sum((x-m)^3 ))/((n-1)*(n-2)*s^3)
return(g)}
```

Criamos então uma função que chamamos gamma1() para calcular a assimetria de uma variável. Podemos testar na distância interorbital de pombos da seguinte maneira:

```
> gamma1(pombos$V1)
[1] -0.08869748
```

Perceba que neste exemplo definimos uma função já utilizando alguns elementos de programação, onde definimos uma entrada (variável x), alguns valores a serem utilizados (m, n e sd) e definimos o valor que será retornado após todos os cálculos usando a função interna "`return()`". O cálculo da curtose (gamma2) pode ser definido similarmente, mas deixamos como exercício para treinar a elaboração de funções mais complicadas. É possível utilizar testes de ajuste (goodness of fit) para comparar numericamente distribuições amostrais com distribuições teóricas (ou duas distribuições amostrais). Para comparar a distribuição de uma variável contínua com uma distribuição normal teórica temos dois testes disponíveis no R: o teste de Shapiro-Wilk e o

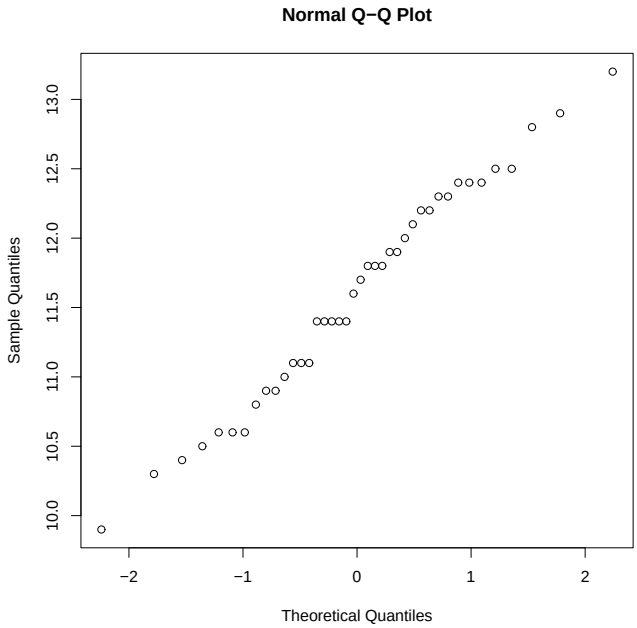

Figura 18: Gráfico de quantis para a distância interorbital dos pombos.

Komogorov-Smirnov (mais geral, este teste pode ser utilizado para comparar dados com qualquer distribuição ou distribuições amostrais diferentes). O teste de Shapiro-Wilk é acessado através da função `shapiro.test()`:

```
>shapiro.test(pombos$V1)
 Shapiro-Wilk normality test
data:  pombos$V1
W = 0.9807, p-value = 0.7144
```

Mais detalhes e referências podem ser encontradas no respectivo arquivo de ajuda. Neste teste, o valor de p (`p-value`) indica a probabilidade de observar uma amostra aleatória tão diferente do esperado. Se neste caso a probabilidade é de 0,714, dizemos que a chance de observar tal amostra é muito alta e não podemos rejeitar a hipótese nula de que a variável apresente uma distribuição diferente da normal. O teste de Shapiro é mais robusto que outros e deve ser utilizado para testes de normalidade.

Se estivermos interessados em comparar observados e esperados em distribuições teóricas, precisamos utilizar o teste de Qui-quadrado (função `chisq.test()`). Por exemplo, para comparar a distribuição da variável van$N com a distribuição teórica esperada pelo modelo de Poisson, temos que tabelar esta distribuição e calcular as probabilidades associadas. Primeiro criando um vetor com as contagens observadas para cada número de indivíduos:

```
> tab<-as.vector(table(van$N))
> tab
 [1] 62 17 51 17 19 17 14 13 10 8 4 9 4 5 3 1 4 2 1 4 1 1 1 2 2
[26] 1 1 1 2 2 1 1 1 1 1 1
```

Calculamos então as probabilidades esperadas para cada contagem destas pela distribuição de

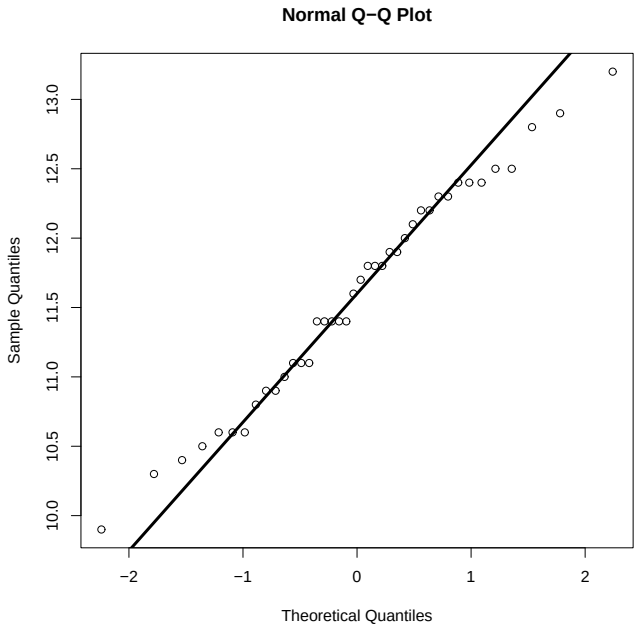

Figura 19: Gráfico de quantis para a distância interorbital dos pombos com uma linha reta.

Poisson:

```
> p<-dpois(as.numeric(dimnames(table(van$N))[[1]]),mean(van$N))
```

Veja que é necessário calcular as probabilidades baseadas na própria tabela de distribuição original (modificando o modo dos caracteres para numéricos) para que haja uma compatibilidade entre os vetores a serem comparados. A função `dpois()` permite o cálculo das probabilidades para cada um dos números de indivíduos quando fornecemos a média de van$N como o parâmetro λ da distribuição. Calculamos então o teste de Qui-quadrado da seguinte maneira:

```
> chisq.test(tab,p=p,rescale.p=T)
 Chi-squared test for given probabilities
data:  tab
X-squared = 1.253998e+72, df = 35, p-value < 2.2e-16
Mensagem de aviso:
Chi-squared approximation may be incorrect in:  chisq.test(tab, p = p, rescale.p=
T)
```

O aviso acima refere-se ao fato das probabilidades calculadas a partir do modelo de Poisson serem baseadas na média da distribuição, apresentando portanto um grau de liberdade a menos que o calculado para o teste normal (nclasses-2).Na prática, a diferença entre a probabilidade associada a um valor de Qui-quadrado tão grande será mínima para 35 ou 34 graus de liberdade. O argumento `rescale.p` também é importante para a função, pois como a nossa tabela não contempla todos os valores possíveis da variável aleatória, o somatório das probabilidades não é igual a 1 (no nosso caso é 0,9999963). O argumento cuida para que as probabilidades sejam recalculadas de acordo, para que o somatório seja 1. É claro que podemos realizar os cálculo necessários com muita facilidade atribuindo uma função personalizada (sabendo que a fórmula

para o cálculo da estatística é $\sum (obs - esp)^2/esp$):

```
> chi2<-sum(((tab - p*285)^2) / (p*285))
> chi2
[1] 1.254003e+72
```

Encontramos a probabilidade associada ao valor encontrado:

```
> 1-pchisq(chi2,34)
[1] 0
```

Como vimos anteriormente, o valor de probabilidade é tão baixo que o programa arredonda para 0. O mesmo processo descrito anteriormente pode ser utilizado para qualquer distribuição de probabilidade teórica ou na comparação entre diferentes distribuições amostrais.

6 Limites de confiança e testes de hipóteses

Já vimos como utilizar as distribuições de probabilidade para encontrar quantiles e probabilidades associadas. Para gerar limites de confiança para um estimador qualquer, podemos obter estes valores a partir das funções apropriadas a partir do R. Aproveitando o exemplo da distância interorbital dos pombos utilizado anteriormente, temos os seguintes valores para a média e o erro padrão: $\bar{Y} = 11,5875; s_{\bar{Y}} = 0,797492/\sqrt{40} = 0,12609$. Usando a distribuição t de Student, (`qt()`), encontramos que, para um intervalo de confiança de 95%, o erro padrão deve ser multiplicado pelo seguinte coeficiente:

```
> qt(0.975,df=39)
[1] 2.022691
```

Veja que para conseguir o percentil correto da distribuição visando o intervalo de confiança, precisamos colocar a probabilidade bicaudal (0,975). O seguinte programa permite calcular intervalos de confiança segundo a distribuição t de Student para intervalos de 95% de confiança:

```
conf95<-function(x) {
m=mean(x)
sd=sd(x)
n=length(x)
sm=sd/sqrt(n)
l1=m-qt(0.975,df=n-1)*sm
l2=m+qt(0.975,df=n-1)*sm
int=list("L1"=l1,"Média"=m,"L2"=l2)
return(int) }
```

A função `conf95()` pode ser então usada para o cálculo de intervalos de confiança para a média de qualquer variável x. Por exemplo:

```
> conf95(pombos$V1)
$L1
[1] 11.33245
$Média
[1] 11.5875
$L2
[1] 11.84255
```

É importante lembrar que este intervalo de confiança permite a inferência sobre a incerteza da estimação. Se o propósito do intervalo for descrever a amostra ou a população, devemos utilizar os percentiles (e o diagrama boxplot) ou intervalos baseados no desvio padrão (se sabemos que a população e a amostra são normais). A adição de intervalos de confiança a gráficos já prontos pode ser realizada a partir da função `arrows()`, que normalmente adiciona flechas a gráficos, mas pode ser personalizada para um tipo de *whisker* mostrando os limites máximo e mínimo.

6.1 Limites de confiança baseados em reamostragem

Vamos conhecer agora um pacote que será utilizado constantemente em todas as partes do curso. O pacote `boot`, o qual já é instalado junto com o pacote base do R, e facilita a realização de mais

de um tipo de reamostragem (*bootstrap* paramétrico, não paramétrico, permutação). O primeiro passo é carregar o pacote (biblioteca) `boot` da seguinte maneira:

```
> require(boot)
```

ou

```
> library(boot)
```

Uma vez carregado o pacote, a função mais importante dentro deste é a função `boot()`, a qual utilizaremos para realizar a reamostragem e o cálculo da estatística de interesse. A função `boot()` apresenta uma série de argumentos: `boot(data, statistic, R, sim= "ordinary", stype= "i")`. Além destes outros podem ser verificados no arquivo de ajuda correspondente. Para o nosso propósito, os argumentos mais importantes são `data` (o conjunto de dados (*data.frame*) a ser analisado, statistic (a função definindo a estatística a ser calculada em cada uma das reamostragens), `R` (o número de reamostragens a realizadas, são sorteadas R amostras com tamanho igual ao n original), `sim` (o tipo de reamostragem a ser realizado = "ordinary"é o bootstrap comum com reposição; "permutation"deve ser utilizado para análises de permutação sem reposição; outros tipos podem ser encontrados na ajuda), e `stype` é o significado do segundo argumento que deve ser utilizado na função, definindo quais observações devem ser reamostradas (veremos a importância disto adiante). O resultado da aplicação da função `boot()` é um objeto da classe *boot* (uma lista), onde os valores de interesse são t0 (o valor da estatística baseado na amostra original), e t (contendo os R valores da estatística baseados na reamostragem). O pacote `boot` apresenta funções que permitem a extração direta de intervalos de confiança (`boot.ci()`) e a geração de gráficos com os resultados (`plot.boot()`).

Vamos examinar agora um exemplo para começar a entender como funciona o processo. O passo mais importante (e talvez mais difícil) é definir a função que será utilizada no cálculo da estatística. Para isso, temos que criar uma função com dois argumentos: o conjunto de dados a ser analisado e o índice das observações que serão reamostradas. Já vimos anteriormente que o índice de uma observação em um vetor é determinado por um número ou um conjunto de números entre colchetes. Desta maneira, quando queremos referenciar o terceiro elemento de um vetor X, escrevemos X[3]. Se queremos referenciar os elementos de 3 a 6 de um vetor X, escrevemos X[3:6], e assim por diante. Este índice deve ser definido como um dos argumentos utilizados pela função de *bootstrap* de modo a especificar quais grupos de observações serão reamostrados. A aplicação mais importante desta indexação é diferenciar a reamostragem de *bootstrap* (com reposição) da reamostragem com permutação (sem reposição). No nosso exemplo, vamos calcular um intervalo de confiança baseado na reamostragem para a média das fêmeas de *Poecilia vivipara* contidas no *data.frame* `poecb` utilizado anteriormente para gerar gráficos *boxplot*. Primeiro vamos remover os dados relativos às fêmeas (os 60 primeiros) e criar um vetor separado:

```
> fem<-poecb$CP[1:60]
```

Este será o nosso conjunto de dados original. Agora vamos criar a função que permitirá ao programa recalcular a média para cada reamostragem:

```
> media<-function(x,i) mean(x[i])
```

Esta função é um pouco diferente das outras que fizemos aqui, pois apresenta dois argumentos (x,i), determinando a variável x e o índice vetorial i. Se tentarmos calcular a média de uma variável simplesmente colocando o argumento x recebemos uma mensagem de erro. Por exemplo, tentamos calcular a média do vetor fem definido anteriormente:

```
> media(fem)
Erro em mean(x[i]) :  argumento "i"ausente, sem padrão
```

Por outro lado, se definirmos i encontramos:

```
> media(fem,1:60)
[1] 4.852167
```

Que equivale a utilizar a função `mean()` sem definir o índice:

```
> mean(fem)
[1] 4.852167
```

A função `media()` precisa ser definida desta maneira para que possamos utilizá-la como estatística de reamostragem. O segundo argumento da estatística a ser calculada é definido pelo parâmetro `stype`, cujo valor *default* corresponde ao índice [i]. Outros tipos estão disponíveis, como frequências [f] ou pesos[w], de modo a realizar bootstraps em estruturas de dados diferentes dos *data.frames* usuais. Para o nosso propósito, não precisamos alterar o valor *default*. Se quisermos calcular uma reamostragem *bootstrap* do vetor `fem`, escrevemos os seguintes comandos:

```
> bootfem<-boot(fem,media,1000)
```

Esta função cria um objeto (`bootfem`) da classe *boot* que corresponde a uma lista de valores. Destes, os mais importantes são as 1000 médias calculadas por reamostragem que encontram-se no vetor `bootfem$t`. A partir deste vetor, podemos estimar uma média, mediana e um intervalo de confiança, gerar histogramas e boxplots, assim como comparar estes intervalos entre grupos ou variáveis. Se quisermos visualizar um summário do objeto podemos escrever:

```
> bootfem
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = fem, statistic = media, R = 1000)
Bootstrap Statistics :
 original bias std.  error
t1* 4.852167 0.005603 0.06890374
```

Este resumo nos mostra a estatística original e o erro padrão da estatística calculado pelo processo de reamostragem. Se quisermos calcular intervalos de confiança, usamos:

```
> boot.ci(bootfem)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates
CALL :
boot.ci(boot.out = bootfem)
Intervals :
Level Normal Basic
95% ( 4.712, 4.982 ) ( 4.710, 4.977 )
Level Percentile BCa
95% ( 4.728, 4.994 ) ( 4.719, 4.984 )
Calculations and Intervals on Original Scale
Mensagem de aviso:
bootstrap variances needed for studentized intervals in:  boot.ci(bootfem)
```

O intervalo Normal é baseado na expectativa da distribuição normal, usando o desvio padrão das estatísticas reamostradas. O Percentile corresponde aos percentis de 0,025 e 0,975. O Basic é calculado a partir dos intervalos básicos de *bootstrap*, cujas fórmulas podem ser encontradas em Davison and Hinkley (1997 - Bootstrap Methods and Their Application, Chapter 5. Cambridge University Press).

Os objetos de classe *bootstrap* apresentam também um método gráfico para a função `plot()`, que consiste em colocar lado a lado um gráfico de histograma e um gráfico de quantis da distribuição *bootstrap* da estatística (mostrando o valor de t0 como uma linha tracejada vertical no histograma). Podemos ver o resultado deste na Figura 20.

```
> plot(bootfem)
```

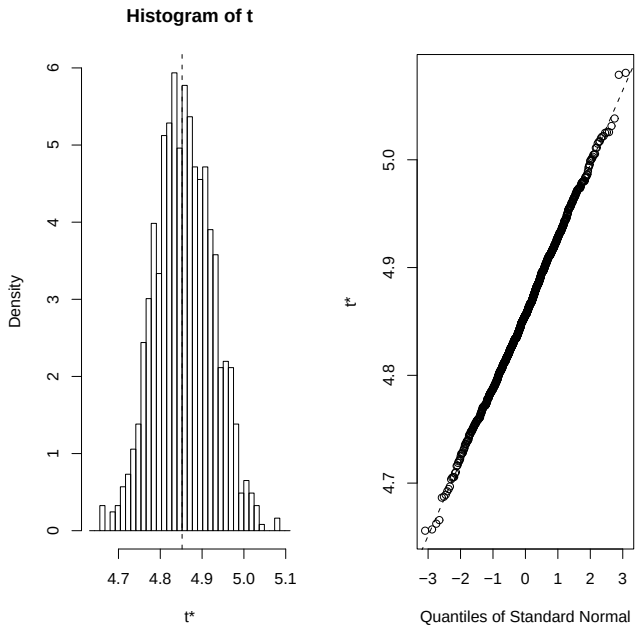

Figura 20: Resultado do método gráfico para objetos *bootstrap*

Se quisermos construir gráficos diferentes, podemos referenciar o conjunto de 1000 valores da estatística reamostrada como `bootfem$t`, ou seja, é o vetor `t` dentro do objeto `bootfem`. Por exemplo, podemos calcular o intervalo de confiança de 95% simplesmente usando a função `quantile()`:

```
> quantile(bootfem$t,c(0.025,0.975))
 2.5% 97.5%
4.727663 4.992704
```

Compare estes resultados com os Percentiles da função `boot.ci()` descrita acima. Podemos também criar um histograma, como na Figura 21.

```
> hist(bootfem$t,col="lightgray"
```

Ou um boxplot, como na Figura 22.

```
> boxplot(bootfem$t
```

Podemos repetir os mesmos cálculos para machos e comparar o intervalo obtido com o das

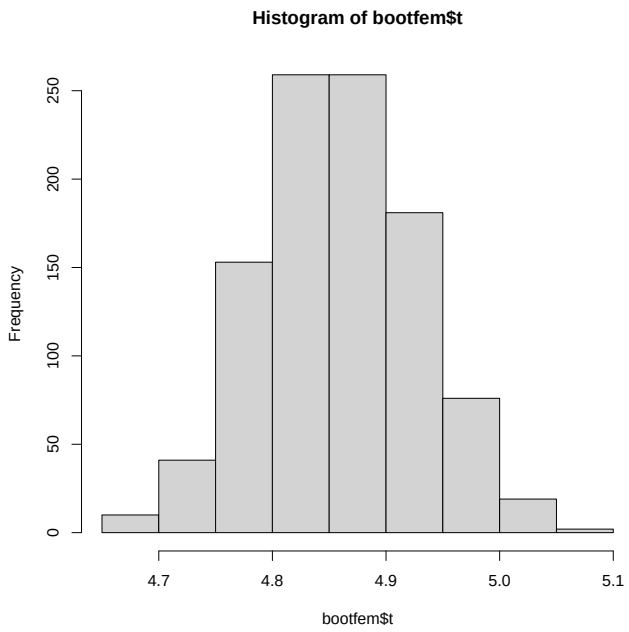

Figura 21: Histograma para reamostragem *bootstrap* da média.

fêmeas:

```
> mac<-poecb$CP[61:120]
> bootmac<-boot(mac,media,1000)
> boxplot(bootfem$t,bootmac$t,ylab="Comprimento padrão médio",names=c("Fêmeas","Machos"))
```

O gráfico obtido pode ser visto na Figura 23.

Agora vamos calcular um intervalo de confiança bootstrap para uma função com distribuição não conhecida, como é o caso do índice de dimorfismo sexual. Este índice é calculado como a razão entre a média do maior sexo e a média do menor sexo. No caso da comparação entre machos e fêmeas, criamos um *data.frame* com os vetores relativos aos dois sexos:

```
> sexdim<-data.frame(fem,mac)
> id<-function(x,i) mean(x$fem[i])/mean(x$mac[i])
> idboot<-boot(sexdim,id,1000)
```

A reamostragem acontece nos dois vetores (`fem` e `mac`) de modo independente. Podemos visualizar o resultado no boxplot da Figura 24.

```
> boxplot(idboot$t,ylab="Índice de dimorfismo sexual")
```

O intervalo de confiança de 95% e a mediana:

```
> quantile(idboot$t,c(0.025,0.5,0.975))
 2.5% 50% 97.5%
1.107970 1.161902 1.216083
```

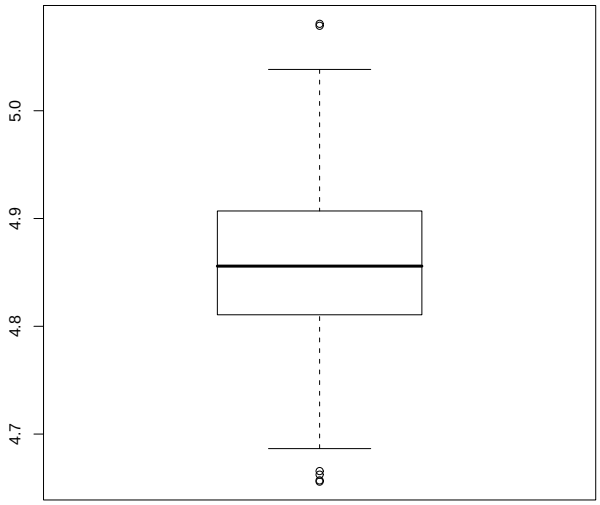

Figura 22: Boxplot para reamostragem *bootstrap* da média.

6.2 Testes de hipóteses utilizando reamostragem

Como podemos perceber anteriormente, a função que define a estatística a ser reamostada pode ser tão complexa ou simples quanto precisarmos que seja. Imagine que nosso interesse agora seja definir uma estatística que permita a comparação de médias entre dois grupos, digamos, os machos e fêmeas do exemplo anterior. Podemos definir uma estatística como a diferença entre as médias e reamostrar os dois grupos de modo a definir um intervalo de confiança para a estatística:

```
> dif<-function(x,i) mean(x$fem[i])-mean(x$mac[i])
> difboot<-boot(sexdim,dif,10000)
> quantile(difboot$t,c(0.025,0.975))
 2.5% 97.5%
0.4665000 0.8810125
```

Realizando 10000 reamostragens, temos um intervalo de confiança de 95% para esta estatística como mostrado acima. Percebemos que o valor do limite inferior ainda é maior que 0 (o valor esperado se não houvesse uma diferença real entre as médias). De fato, o valor mínimo observado entre as 10000 reamostragens foi 0.2291667, o que significa que a hipótese nula de igualdade entre médias deve ser rejeitada. Vamos olhar um segundo exemplo, em que os valores dentro dos vetores de machos e fêmeas serão obtidos como números aleatórios de uma distribuição normal com média 5 e desvio padrão 1.

```
> mac2<-rnorm(20,mean=5,sd=1)
> fem2<-rnorm(20,mean=5,sd=1)
> sexdim2<-data.frame(fem2,mac2)
> dif<-function(x,i) mean(x$fem2[i])-mean(x$mac2[i])
```

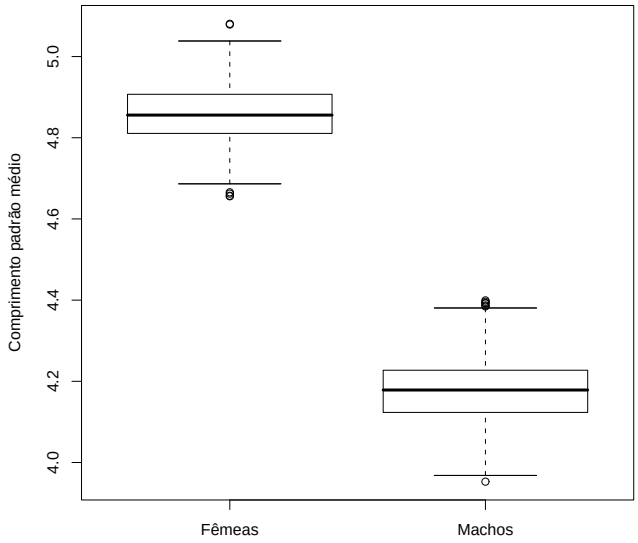

Figura 23: Boxplot comparando intervalos de confiança *bootstrap* da média do comprimento padrão.

```
> difboot2<-boot(sexdim2,dif,10000)
> quantile(difboot2$t,c(0.025,0.975))
 2.5% 97.5%
-0.88632734 0.08367873
```

Neste caso sabemos a que hipótese de igualdade entre as médias é verdadeira (dif=0). Como o intervalo de confiança de 95% incorpora o valor 0, devemos aceitar a hipótese nula. Uma outra maneira de realizar isto, seria construir um intervalo de confiança baseado em permutações para a estatística. Neste caso, o procedimento é completamente diferente. Temos que permutar (trocar de lugar) machos e fêmeas aleatoriamente (em um sorteio sem reposição) e calcular a diferença entre eles. As amostras permutadas têm uma expectativa de apresentar uma diferença média igual a 0, com uma variância determinada pela própria amostra. Este processo exige a definição da função de um modo completamente diferente.

```
> dif<-function(x,i) {
+ x[i]->data
+ g1<-data[1:60]
+ g2<-data[61:120]
+ d<-mean(g1)-mean(g2)
+ return(d) }
```

Esta função basicamente vai esperar a função `boot` criar um conjunto de dados permutado (`data`), o qual é posteriormente quebrado em duas metades (`g1 e g2`), cuja diferença média (`d`) é calculada e retornada, que correspondem à separação original de machos e fêmeas. Nesta reamostra, os indivíduos serão recolocados em posições diferentes de modo que nem todos os 60

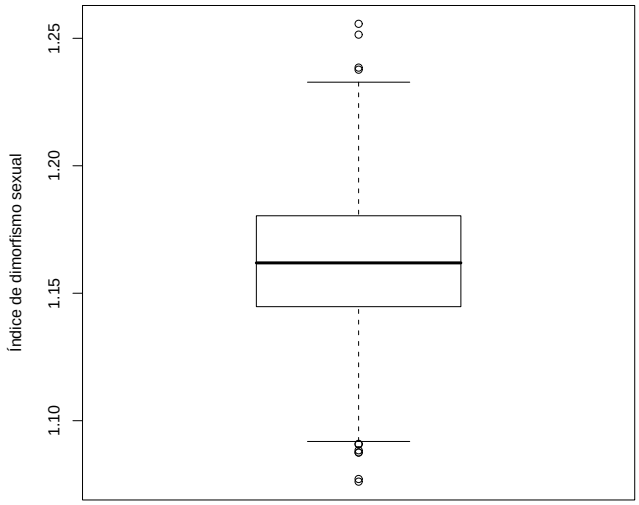

Figura 24: Boxplot mostrando intervalo de confiança *bootstrap* da média do comprimento padrão.

primeiros serão fêmeas (a expectativa neste caso é algo em torno de 50% de fêmeas entre os 60 primeiros). Ao repetir esta amostragem R vezes criaremos uma distribuição que corresponde à expectativa de diferença caso a hipótese nula de diferença média =0 seja verdadeira. Após isso, vamos comparar o valor observado com as diferenças calculadas a partir da permutação.

```
> permdim<-boot(poecb$CP,dif,10000,sim="permutation")
```

Agora não usamos mais a opção *default* de `sim="ordinary"`. Utilizamos a opção de permutação, visto que desejamos que todos os espécimes sejam recolocados (amostragem sem reposição) na amostra, mas desta vez misturando machos e fêmeas. A distribuição da estatística `dif` pode ser visualizada na Figura 25 obtida pelo comando abaixo:

```
hist(permdim$t, xlab="Diferença entre médias permutadas", main=NULL,
col="lightgray")
```

Sabemos que o valor original da função `dif` para este conjunto de dados é:

```
> dif(poecb,1:120)
[1] 0.6721667
```

e o intervalo de confiança de 99% para a distribuição da estatística após 10000 permutações aleatórias foi:

```
> quantile(permdim$t,c(0.005,0.995))
 0.5% 99.5%
-0.3248333 0.3245633
```

O qual não inclui o valor observado. Poderíamos desta maneira rejeitar a hipótese nula de igualdade entre as médias de machos e fêmeas ao nível de significância $P < 0{,}01$. Podemos calcular um valor exato baseado na distribuição. Os valores máximo e mínimo observados na

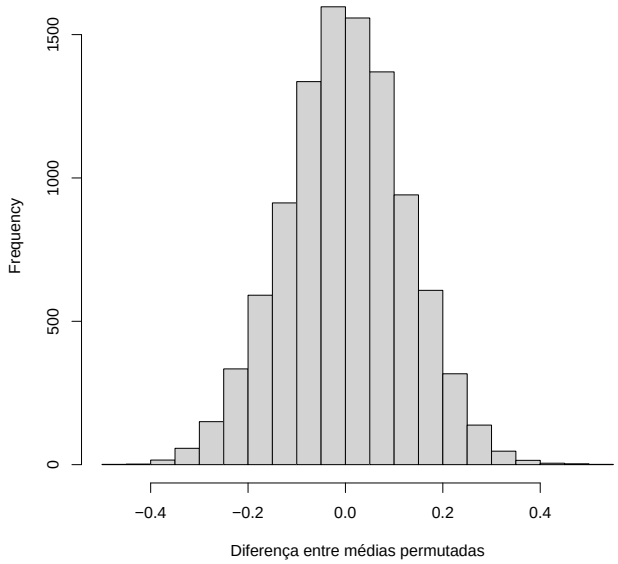

Figura 25: Histograma mostrando a distribuição da diferença entre médias permutadas.

reamostragem foram `max(permdim$t)= 0,5135` e `min(permdim$t)= -0,4732`, respectivamente, significando que não encontramos nenhum valor igual ou mais distante da expectativa da hipótese nula (`dif=0`). Podemos dizer que o valor de probabilidade observado seria ainda menor que 1/10000 (P $<$ 0,0001), o que corresponde a uma probabilidade muito baixa de encontrar um valor de `dif` como o observado.

Voltando ao exemplo da simulação de números aleatórios, usamos novamente os valores normais simulados para comparar os resultados em uma situação onde sabemos que a hipótese nula `dif=0` é verdadeira. Primeiro criamos o conjunto de dados simulado com números aleatórios normais:

```
> CP2<-rnorm(120,mean=5,sd=1)
```

Verificamos o funcionamento da função calculando o valor observado:

```
> dif(CP2,1:120)
[1] -0.1673237
```

Agora realizamos as permutações aleatórias

```
> permdif2<-boot(CP2,dif,10000,sim="permutation")
```

E verificamos o intervalo de confiança de 95% gerado:

```
> quantile(permdif2$t,c(0.025,0.975))
 2.5% 97.5%
-0.3113531 0.3145535
```

Podemos perceber por estes números que o valor observado encontra-se dentro do intervalo de confiança, o que já pode ser considerado como um teste da hipótese nula, que deve ser aceita neste caso. Vamos procurar descobrir agora a probabilidade exata de encontrar uma amostra tão desviante assim. Examinando os 10000 valores de dif permutados, encontramos 2935 iguais

ou maiores (em valor absoluto para um teste bicaudal) que o observado (dif=-0,1673237). Isto nos dá uma probabilidade de 2935/10000 (P = 0,2935) de encontrar uma amostra tão desviante caso a hipótese nula seja verdadeira. Com uma probabilidade tão alta temos que aceitar esta hipótese nula. O histograma para a distribuição da estatística permutada neste segundo exemplo encontra-se na Figura 26.

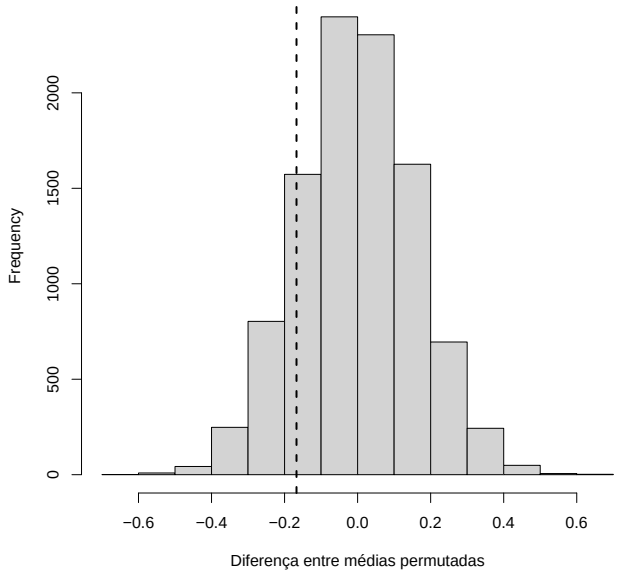

Figura 26: Histograma mostrando a distribuição da diferença entre médias permutadas no segundo exemplo. A linha tracejada vertical mostra a posição do valor observado.

Tendo algum domínio sobre estas ferramentas, podemos implantar qualquer função e qualquer estrutura de reamostragem. À medida que avançarmos nos modelos lineares veremos como realizar testes de *bootstrap* e permutação sobre diferentes estruturas de dados e como estimar parâmetros e testar hipóteses em modelos mais complexos.

6.3 Poder e planejamento amostral

O cálculo do poder é muito importante para o planejamento de um estudo. O R apresenta funções para o cálculo do poder de alguns testes pré-determinados, como análise de variância, teste t e teste de proporções. vamos utilizar como exemplo o poder a ser calculado para um teste t na comparação entre amostras. Em primeiro lugar, vamos testar a hipótese nula de que as duas amostras simuladas do exemplo de permutação anterior tenham vindo de uma mesma população ($H_0 : \mu_1 - \mu_2 = 0$). Sabemos que a hipótese nula é verdadeira, visto que os dados são simulados a partir de uma distribuição normal com média e desvio padrão conhecido. Para realizar um teste t, utilizamos a função `t-test()` onde podemos inserir argumentos de diversas maneiras. Tendo em vista que os dados encontram-se como subconjuntos do vetor CP2, vamos comparar os grupos referenciando os índices respectivos:

```
> t.test(CP2[1:60],CP2[61:120],var.equal=T)
```

Aqui determinamos que o grupo 1 encontra-se n vetor CP do 1 ao 60, enquanto o grupo 2 encontra-se no vetor CP de 61 a 120. O argumento var.equal=T determina que os grupos apresentam variâncias homogêneas (ainda não estamos vendo testes de premissas) e calcula uma variância combinada dentro dos grupos.

```
 Two Sample t-test
data:  CP2[1:60] and CP2[61:120]
t = -1.0524, df = 118, p-value = 0.2947
alternative hypothesis:  true difference in means is not equal to 0
95 percent confidence interval:
-0.4821604 0.1475129
sample estimates:
mean of x mean of y
4.935465 5.102789
```

Observamos que o valor de p (p-value) é muito parecido com o que encontramos anteriormente para o teste da mesma hipótese utilizando permutações (0,2935 - 0,2947). Com uma probabilidade de quase 0,3 de encontrarmos uma diferença média como esta, devemos aceitar a hipótese nula (como fizemos no teste de permutações). A hipótese alternativa de ($H_1 : \mu_1 \neq \mu_2$) determina que o teste realizado foi bicaudal. Se por algum motivo estivéssemos interessados apenas em $\mu_1 > \mu_2$ ou $\mu_1 < \mu_2$, o teste seria unicaudal (é possível determinar isto utilizando argumentos na função). Se o propósito desta análise for planejar o estudo, precisamos calcular o poder obtido e planejar o tamanho amostral de acordo com a função `power.t.test()`. Para isso é preciso fornecer os argumentos conhecidos e a função retorna o argumento a ser estimado. Como vimos na aula teórica, o poder é determinado pelo tamanho amostral (n em cada grupo), pela diferença entre as médias (*delta*), pelo desvio padrão dentro dos grupos (*sd*) e pelo nível de significância escolhido (*alpha*). Se quisermos calcular qualquer um destes parâmetros a partir de um poder pré-determinado, basta fornecer o poder como um argumento (*power*) e a função retorna o que estiver faltando. Para o exemplo acima, temos os seguintes argumentos: n = 60, delta = (4,935465-5.102789)= -0,167324, sd = sqrt((var(CP2[1:60])+var(CP2[61:120]))/2) = 0,8708, sig.level = 0,05. escrevemos então:

```
> power.t.test(n=60,delta=-0.167324,sd=0.8708)
 Two-sample t test power calculation
 n = 60
 delta = 0.167324
 sd = 0.8708
 sig.level = 0.05
 power = 0.179816
 alternative = two.sided
 NOTE: n is number in *each* group
```

O poder calculado é muito baixo. A probabilidade de cometer um erro do tipo II (β) neste caso é muito alta, pois $\beta = 1 - power = 0,82018$. O propósito de calcular o poder de um teste é auxiliar no planejamento do estudo. Se as características do sistema sendo estudado forem estas

mesmas (diferença entre as médias e variabilidade dentro dos grupos) podemos estimar qual seria o tamanho amostral necessário para alcançar um poder maior, digamos, de 0,8:

```
> power.t.test(delta=-0.167324,sd=0.8708,power=0.8)
 Two-sample t test power calculation
 n = 426.128
 delta = 0.167324
 sd = 0.8708
 sig.level = 0.05
 power = 0.8
 alternative = two.sided
 NOTE: n is number in *each* group
```

Neste caso, precisaríamos de 426 individuos por grupo para chegar ao poder desejado. Se quiséssemos um poder maior precisaríamos de amostras ainda maiores. É claro que isto tudo depende do nosso julgamento subjetivo em decidir que a diferença de 0,167 entre as médias é uma diferença biologicamente significante (independente da significância estatística). Podemos utilizar a função para calcular a diferença mínima estatisticamente significativa a ser detectada pelo presente tamanho amostral:

```
> power.t.test(n=60,sd=0.8708,power=0.8)
delta = 0.4490754
```

Para economizar espaço, colocamos apenas o resultado a ser estimado. A diferença real entre as médias teria que ser 0,449 para ser detectável com o poder determinado.

Quando podemos especificar melhor a hipótese alternativa, o teste apresenta um poder maior. Por exemplo, imagine que só temos interesse no resultado se $\mu_1 > \mu_2$, de modo que a hipótese nula passa a ser $H_0 : \mu_1 = \mu_2, \mu_1 < \mu_2$. O teste passa a ser unicaudal e o poder aumenta, como veremos:

```
> power.t.test(n=60,sd=0.8708,power=0.8,alternative="one.sided")
 Two-sample t test power calculation
 n = 60
 delta = 0.3975957
 sd = 0.8708
 sig.level = 0.05
 power = 0.8
 alternative = one.sided
 NOTE: n is number in *each* group
```

Neste caso, a diferença mínima detectável seria 0,3976, menor que a calculada para o teste bicaudal. Se repetirmos estes cálculos, veremos que o n mínimo também seria menor e assim por diante. Uma ferramenta interessante a ser utilizada neste planejamento é o gráfico com a curva de poder para diferentes tamanhos amostrais (ou qualquer outro parâmetro que possa ser modificado durante a realização do estudo).

```
> curve(power.t.test(n=x, delta=-0.167324, sd=0.8708)$power, from=10, to=1000,
lwd=2, xlab="Tamanho amostral", ylab="Poder do teste")
```

A figura produzida pelo comando acima encontra-se na Figura 27. Percebemos que após um determinado tamanho amostral (em torno de 500) a curva começa a estabilizar, de modo que a

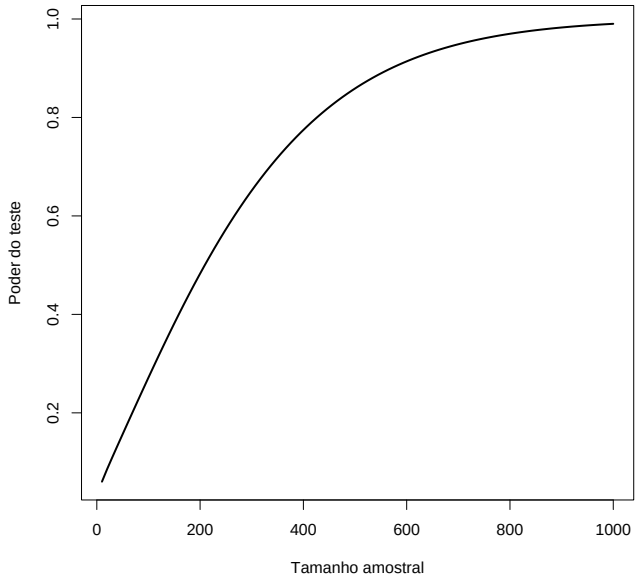

Figura 27: Curva mostrando o poder do teste t como função do tamanho amostral.

diferença no ganho de resolução entre amostrar 500 ou 1000 indivíduos é pequena (um aumento no poder em torno de 0,13). Esta informação também deve ser utilizada no planejamento de um estudo, visto que na maioria dos casos, cada observação realizada compreende um custo de dinheiro e tempo (em alguns casos de vidas também, dependendo do tipo de pesquisa a ser realizada) e o ideal é que o pesquisador otimize esta relação custo/benefício para que seja a menor possível.

A informação mais importante então a ser obtida em um estudo piloto é a variabilidade que pode ser esperada na variável a ser medida e a determinação da magnitude do efeito a ser estudado. Esta magnitude pode ser muito fácil ou muito difícil de calcular, mas é de suma importância procurar entender o que é e o que não é biologicamente significante. Outros parâmetros como o poder e o nível de significância desejados devem ser determinados de acordo com o objetivo de cada estudo, de modo a não ser conservador demais nem liberal demais. Uma solução de compromisso deve ser buscada e os valores usuais encontrados na literatura (alfa = 0,05 e poder = 0,8) não precisam ser seguidos à risca. Na verdade, o ideal é ajustar estes valores de acordo com os objetivos do estudo e a seriedade das consequências dos diferentes tipos de erro.

7 Modelos lineares no R

Os modelos lineares podem ser acessados no ambiente R a partir da função (`lm()`), que gera um objeto no R correspondente a uma lista com vários componentes do modelo linear em questão (coeficientes, resíduos, valores preditos, dentre outros). A estes objetos (os modelos lineares), encontram-se associados também modos especificos das funções (`plot()`) e (`summary()`), mostrando tabelas de análise de variância para o modelo linear.

A diferença entre modelos (regressão, análise de variância, análise de covariância) é determinada unicamente pela fórmula a ser utilizada na especificação do modelo, a qual é um dos argumentos da função. Para as análises mais simples, as fórmulas são muito fáceis de escrever. No entanto, à medida que o número de fatores (variáveis independentes) aumenta, é necessário especificar possíveis interações entre fatores, o que complica um pouco a determinação das fórmulas. O pacote R-commander, quando instalado, simplifica a tarefa de montar a equação, mas não resolve os modelos mais complicados, como análises de covariância com múltiplos fatores e covariáveis. Inicialmente veremos os modelos de regressão mais simples, seguidos por análises de variância e covariância. Veremos também como realizar reamostragens (*bootstrap* e permutações) para construir limites de confiança e testar hipóteses.

7.1 Regressão linear simples

Os modelos de regressão linear são especificados como uma equação da reta (`Y ~ X`), onde o símbolo $\sim$ assume a função de igualdade e os parâmetros do modelo linear não precisam ser especificados diretamente. A equação de regressão a ser ajustada a partir da especificação acima seria $Y = a + bX$. Uma vez calculado um objeto da classe modelo linear, podemos utilizar este objeto como argumento em uma série de funções que permitem a extração de mais informação do modelo (como a predição de observações) e a verificação de premissas, por meio de gráficos diagnósticos.

Para começar o exemplo sobre regressão, vamos carregar um novo conjunto de dados, contido no arquivo `"algas.txt"`. Este conjunto de dados(extraído de McPherson, G. 1990. *Statistics in scientific investigation: its basis, application and interpretation.* Springer-Verlag, New York) é composto por 20 observações organizadas em duas colunas com cabeçalho. A primeira coluna (`Abund`) corresponde à Abundância de invertebrados (número de indivíduos) encontrados associados a indivíduos da alga *Caulocystis cephalornitos*. A segunda coluna (`DW`) corresponde ao peso seco dos mesmos indivíduos da espécie de alga. Podemos carregar o arquivo e visualizar a relação entre as duas variáveis com os comandos (o gráfico de dispersão gerado encontra-se na figura 28.

```
> algas<-read.table("algas.txt",header=T)
> plot(algas$DW, algas$Abund, xlab="Peso seco(g)",ylab="Abundância
de invertebrados", cex=1.3,cex.lab=1.3,pch=16)
```

Se quisermos ajustar um modelo de regressão a este conjunto de dados, de modo que $Abund = a + bDW$, precisamos utilizar a função lm() para criar um modelo linear. Como explicado antes, o principal argumento para esta função é o modelo a ser ajustado. No nosso caso, este será `Abund` $\sim$ `DW`. O ajuste do modelo é realizado simplesmente escrevendo:

```
> lm.algas<-lm(Abund ~ DW,data=algas)
```

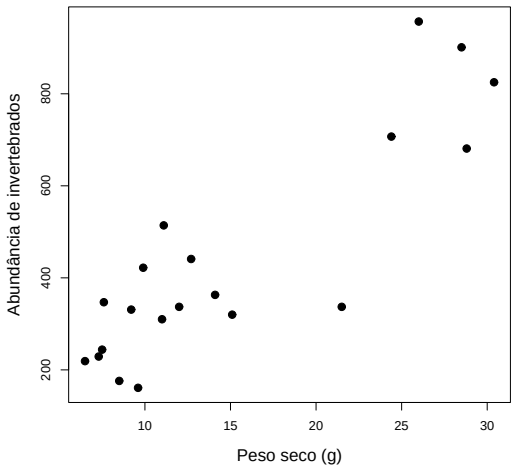

Figura 28: Diagrama de dispersão para o exemplo da alga *Caulocystis*

Conforme convencionado, a variável dependente é a Abundância de invertebrados e a independente é o peso seco das algas. Podemos visualizar os parâmetros estimados para a reta de regressão simplesmente escrevendo o nome do objeto:

```
> lm.algas
Call:
lm(formula = Abund ~ DW, data = algas)
Coefficients:
 (Intercept)    DW
    50.13     25.92
```

O coeficiente chamado (Intercept) corresponde ao a da reta de regressão, ao passo que o coeficiente identificado como DW (o nome da variável independente) corresponde ao b da reta de regressão (o coeficiente angular). Podemos adicionar esta reta de regressão no gráfico de dispersão utilizando a função `abline()` e o modelo linear como argumento (como a função `abline` é de nível mais baixo, é necessário que o gráfico da Figura 28 já esteja ativo no dispositivo gráfico), gerando um gráfico com o da Figura 29:

```
> abline(lm.algas,lwd=2)
```

7.2 Testes de significância em regressão

A informação sobre testes de significância e os erros associados aos parâmetros do modelo linear pode ser obtida com a função `summary()`:

```
> summary(lm.algas)
Call:  lm(formula = Abund ~ DW, data = algas)
Residuals:
      Min       1Q   Median       3Q      Max
 -270.363  -63.036   -5.421   71.259  233.007
Coefficients:
```

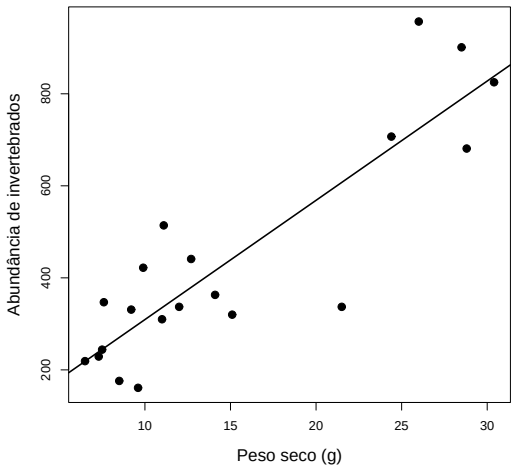

Figura 29: Diagrama de dispersão para o exemplo da alga *Caulocystis*, com a reta de regressão

```
            Estimate  Std.  Error  t value      Pr(>|t|)
 Intercept    50.129        57.428    0.873         0.394
 DW           25.918         3.363    7.707  4.15e-07 ***
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
Residual standard error:  120.4 on 18 degrees of freedom
Multiple R-Squared:  0.7674, Adjusted R-squared:  0.7545
F-statistic:  59.4 on 1 and 18 DF, p-value:  4.152e-07
```

Temos nestas tabelas uma distribuição dos resíduos, com quantis, intervalo e mediana, os valores estimados dos parâmetros com seus devidos erros e testes t para $H_0 : \beta = 0$ e $H_0 : \alpha = 0$. O intervalo de confiança para o coeficiente de regressão pode ser construído a partir do erro associado ao mesmo, usando a distribuição t, por exemplo, o valor crítico de t para os 18 graus de liberdade residuais do modelo, associado ao intervalo de confiança de 95% será:

```
> qt(0.975,18)
[1] 2.100922
```

E os limites de confiança para o coeficiente de regressão podem ser calculados como:

```
> c(25.918-2.100922*3.363,25.918+2.100922*3.363)
[1] 18.8526 32.9834
```

O erro padrão residual (raiz quadrada do quadrado médio residual) é mostrado na mesma saída, assim como o coeficiente de determinação e o coeficiente de determinação ajustado. A diferença entre os coeficientes de determinação normal e ajustado ocorre por causa de uma divisão pelos graus de liberdade. Enquanto o coeficiente de determinação é uma razão entre a soma de quadrados explicada pelo modelo e a soma de quadrados total ($r^2 = \sum \hat{y}^2 / \sum y^2$) , o coeficiente de determinação ajustado é calculado a partir de quadrados médios

$$r^2 = 1 - \left(\frac{\sum d^2_{Y \cdot X} / gl_r}{\sum y^2 / gl_t} \right)$$

.

Onde o numerador da fração é a soma de quadrados do resíduo dividida pelos graus de liberdade do resíduo e o denominador da fração é a variância da variável dependente. O coeficiente ajustado é importante para modelos com um grande número de variáveis independentes, permitindo a comparação de modelos com diferentes números de variáveis independentes. A estatística F mostrada também é um teste para a hipótese $H_0 : \beta = 0$ e podemos perceber comparando os valores de F e t (na linha do DW), que o primeiro é o quadrado do segundo (esta relação entre F e t é bastante conhecida).

A tabela de análise de variância para o modelo pode ser visualizada com o comando `anova()`:

```
> anova(lm.algas)
Analysis of Variance Table
Response:  Abund
                  Df   Sum Sq  Mean Sq        F value  Pr(>F)
 DW          1 860657  860657   59.398  4.152e-07 ***
 Residuals         18  260816    14490
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

A tabela de análise de variância permite uma visualização do teste da hipótese nula $H_0 : \beta = 0$, assim como a decomposição das somas de quadrados, que permitem uma verificação da importância relativa das variáveis independentes no modelo de regressão.

Um número de funções pode também ser utilizada para extrair informação de modelos lineares. Para o propósito da regressão linear, as mais utilizadas são `coef(lm)`, `fitted(lm)` e `residuals(lm)`. Estas funções serão importantes para a realização de reamostragens e o cálculo de intervalos de confiança bootstrap ou testes de hipóteses por permutação. A função para extrair os coeficientes do modelo pode ser usada como:

```
> coef(lm.algas)
(Intercept) DW
 50.12922 25.91785
```

Ou podemos acessar apenas um deles, indexando a função:

```
> coef(lm.algas)[2]
 DW
25.91785
```

A função `predict(lm)` pode ser utilizada para o cálculo de predições de valores individuais e de seus respectivos intervalos de confiança. Esta função é útil se quisermos adicionar intervalos de confiança ao gráfico da Figura 29. Partindo do princípio que o gráfico da Figura 29 encontra-se ativo, precisamos gerar uma nova variável independente com observações em sequência (entre o mínimo e o máximo da variável independente). É importante que esta nova variável esteja dentro de um data frame e que possua o mesmo nome da variável independente para que a função de predição possa encontrá-la:

```
> new.x<-data.frame(DW=seq(6.5,30.5,by=0.1))
```

Com isto podemos predizer os valores da variável dependente para cada valor da nova variável independente, assim como os limites superior e inferior dos intervalos de confiança:

```
> pred.alga<-predict(lm.algas,new.x,interval="confidence")
```

Esta função gera um objeto do tipo matriz, o que faz com que a referência às colunas dos limites de confiança geradas seja realizada por meio de indexação, ao invés dos nomes de variáveis como estamos acostumados nos data frames. Os argumentos utilizados acima são os seguintes: `predict(modelo,novos dados,tipo de intervalo)`. O intervalo de confiança *default* é o de 95%, mas é possível adicionar um argumento (`level=0.99`) que permite o cálculo do intervalo para qualquer nível de confiança. Uma vez gerada a matriz com os valores preditos e os limites de confiança, podemos adicionar estes como linhas ao gráfico usando os comandos:

```
> lines(new.x$DW,pred.alga[,2],lty=2,lwd=2)
> lines(new.x$DW,pred.alga[,3],lty=2,lwd=2)
```

O resultado será um gráfico como o da Figura 30.

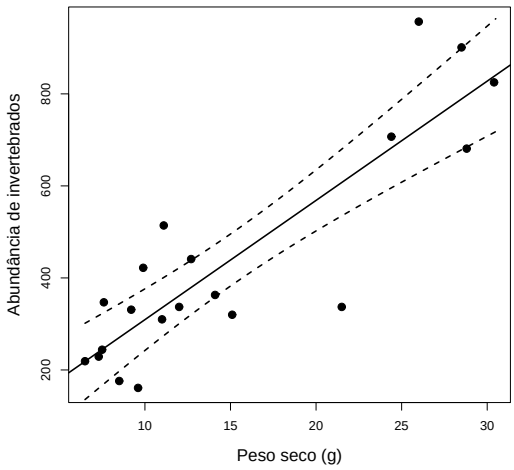

Figura 30: Diagrama de dispersão para o exemplo da alga *Caulocystis*, mostrando a reta de regressão e os intervalos de confiança de 95%.

7.3 Resíduos e testes diagnósticos

Uma série de gráficos estão disponíveis (na função `plot(lm)`) para o teste de premissas do modelo linear. Esta função gera quatro gráficos em série, de modo que pode ser interessante gerar um a um ou colocar todos os gráficos de uma vez. Para este fim, vamos mostrar como colocar múltiplos gráficos em uma única página. para isto, precisamos modificar os parâmetros `mfcol` ou `mfrow`. Estes parâmetros permitem a determinação de quantos gráficos devem aparecer por página, fazendo com que os gráficos gerados na sequência sejam adicionados nas colunas ou linhas pré-determinadas pelos parâmetros:

```
> par(mfrow=c(2,2),pch=16)
> plot(lm.algas)
```

Estes comandos geram um gráfico como o mostrado na Figura 31.

Nesta figura encontramos quatro gráficos diferentes que nos ajudam a verificar as premissas do modelo. O primeiro gráfico mostrado (Residuals vs Fitted) corresponde aos resíduos comparados

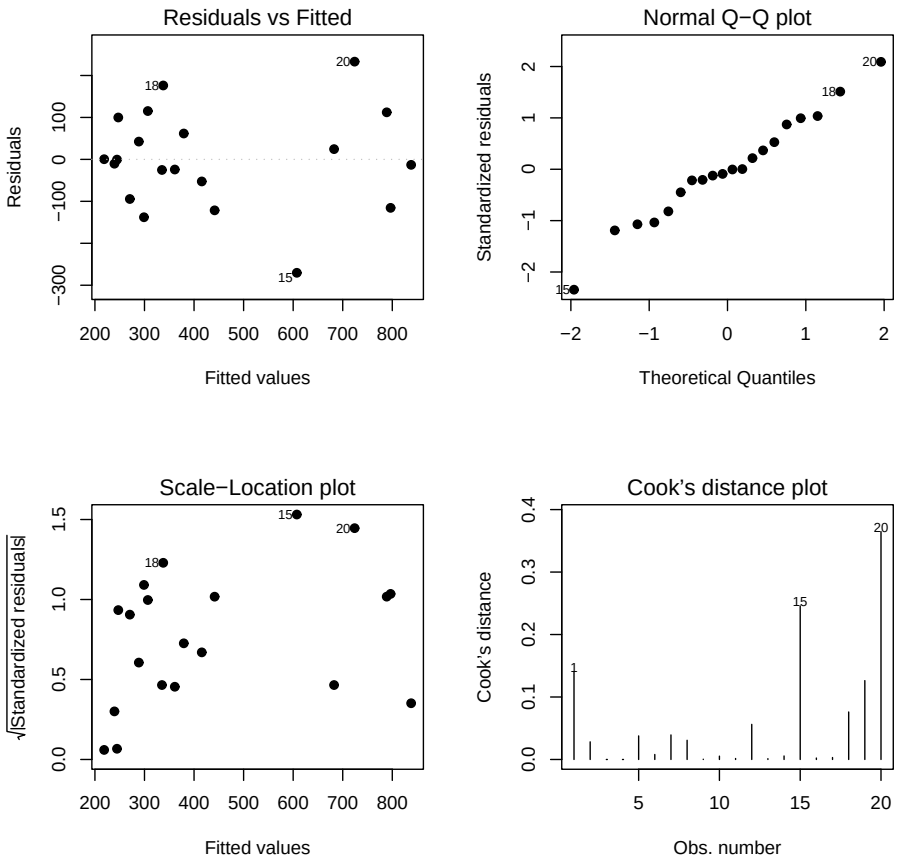

Figura 31: Gráficos diagnósticos para testes de premissas do modelo linear.

aos valores preditos pelo modelo. Neste gráfico é possível observar tendências nos resíduos, como não-linearidade ou mesmo heteroscedasticidade (esta, no entanto, será melhor verificada por outro gráfico). No exemplo da Figura 31, não encontramos nenhuma tendência clara e a distribuição dos resíduos parece simétrica em torno da linha. O segundo gráfico (Normal Q-Q plot) mostra o gráfico de quantis para os resíduos, onde podemos verificar a normalidade dos resíduos. No nosso exemplo, temos os pontos distribuídos de modo aproximadamente linear, o que sugere que os resíduos apresentam uma distribuição normal. O terceiro gráfico (Scale-Location Plot) mostra a distribuição da raiz do módulo dos resíduos em comparação aos valores preditos. Este gráfico permite a verificação da tendência de heteroscedasticidade em que a magnitude da variação dos resíduos está correlacionada com os valores preditos. Quando podemos observar uma correlação clara entre os módulos dos desvios e os valores preditos, os dados apresentam sérios desvios em relação às premissas do modelo. A utilidade destes gráficos será mais evidente nos próximos exemplos. O último gráfico mostra as distâncias de Cook para cada observação. A distância para cada observação é calculada segundo a fórmula:

$$D_i = \frac{\sum (\hat{Y}_j - \hat{Y}_{j(i)})^2}{p \times QMR}$$

Onde $\hat{Y}_j$ corresponde ao vetor de estimados para o modelo baseado na amostra total, $\hat{Y}_{j(i)}$ corresponde ao vetor de estimados para o modelo baseado na amostra removendo a observação i, p é o número de variáveis independentes e QMR é o quadrado médio do resíduo. A distância de Cook para uma determinada observação corresponde geometricamente a uma distância entre as retas de regressão estimadas para a amostra completa e para uma amostra em que falta a i-ésima observação. Isto permite identificar observações influentes em uma amostra, que podem ser revistas para a verificação de possíveis erros de medição ou tipográficos. Em geral, como a distância de Cook é distribuída aproximadamente como uma estatística F (podemos perceber na fórmula que a mesma é uma razão entre duas variâncias), uma regra informal é preocupar-se quando a distância para uma determinada observação for maior que 1. No nosso exemplo, não temos nenhuma observação tão influente assim, de modo que não precisamos nos preocupar com isto.

7.4 Transformações de variáveis

O exemplo que acabamos de ver mostra uma situação em que as premissas do modelo linear foram todas aceitas e o modelo foi bem ajustado. No entanto, encontraremos situações em que a estrutura da amostragem (não independente espacial ou temporalmente), a não-linearidade ou a estrutura do erro (seguindo uma distribuição diferente da normal) exigem a aplicação de técnicas especiais mais apropriadas. Para o caso de amostragens em séries temporais ou espacialmente distribuídas, existem métodos específicos de análise. Por outro lado, se a estrutura do erro não é normal, mas segue uma distribuição esperada conhecida (como Poisson ou Binomial), podemos utilizar modelos lineares generalizados (função `glm()`). Estes, no entanto, são mais complexos que os lineares gerais e não serão tratados nesta apostila. Uma opção que temos em algumas situações é transformar a variável dependente (ou ambas) de modo a linearizar a relação, normalizar o erro ou homogeneizar as variâncias. Uma série de transformações de dados está disponível no R, sendo que uma maneira interessante de determinar a transformação necessária para o conjunto de dados é a função `boxcox(lm)`. Esta função utiliza um procedimento de estimação por verossimilhança máxima de modo a calcular um valor de λ, que define o tipo de transformação mais apropriada para a variável dependente no modelo linear, da seguinte maneira:

$$t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ log(y) & \lambda = 0 \end{cases}$$

.

Esta transformação é definida de modo que um valor de $\lambda = 1$ determina que nenhuma transformação é necessária para este conjunto de dados, quando $\lambda = 0$ uma transformação logarítmica é a melhor opção, e quando $\lambda = 0,5$ a transformação raiz quadrada é mais indicada. Podemos acessar a transformação Box-Cox no R a partir da função `boxcox(lm)` disponível no pacote `MASS`. Vamos experimentar o procedimento no conjunto de dados que acabamos de analisar e ver o resultado. Primeiro é necessário carregar o pacote [`MASS`], o qual é instalado junto com o R-base:

```
> require(MASS)
```

```
Loading required package: MASS
[1] TRUE
```

Podemos agora utilizar a função boxcox() para calcular o λ estimado e visualizar o resultado graficamente, como na Figura 32

```
> par(mfrow=c(1,2))
> boxcox(lm.algas)
> boxcox(lm.algas,lambda=seq(-0.5,1.5,by=0.1))
```

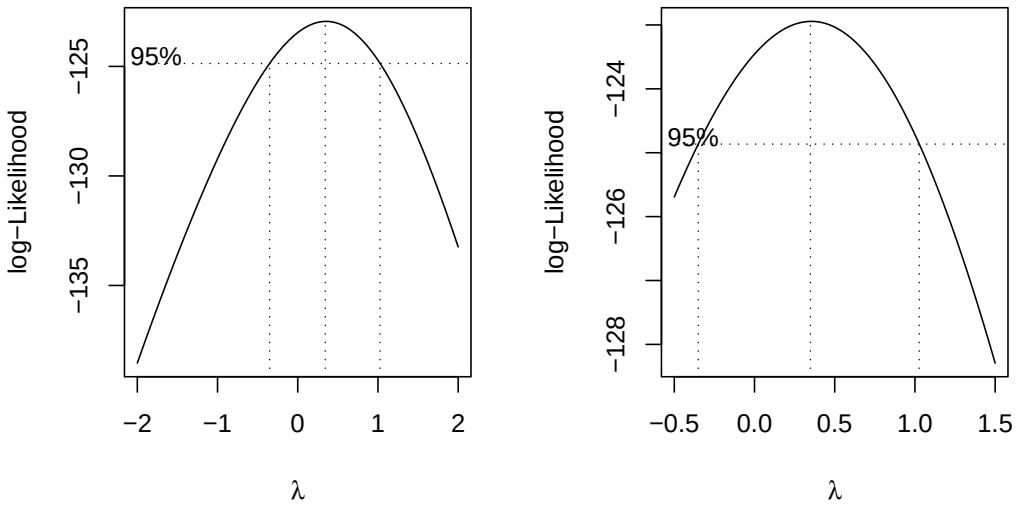

Figura 32: Estimação do parâmetro λ pelo método de Box-Cox. O gráfico da direita mostra uma versão ampliada da região de confiança.

Este gráfico sugere que o λ com maior verossimilhança seria um pouco menor que 0,5. No entanto, percebemos que o 1 encontr-ase dentro do intervalo de confiança de 95%, de modo que podemos escolher neste caso, não efetuar nenhuma transformação. É importante nunca efetuar transformações desnecessárias no conjunto de dados, visto que as mesmas dificultam um pouco a interpretação dos parâmetros, modificando a escala das variáveis. Como os gráficos diagnósticos para este exemplo já mostram linearidade da relação, normalidade e homoscedasticidade dos resíduos, é melhor optar por não modificar.

É interessante agora examinar um outro exemplo em que será interessante modificar a escala da variável dependente para linearizar a relação. Vamos carregar um conjunto de dados de exemplo chamado gemfish.txt, que apresenta duas variáveis e 242 observações, que correspondem a comprimentos de indivíduos (variável `Len`) de uma espécie de peixe australiana (*Rexea solandri*) e o conteúdo de mercúrio (mg/Kg) em tecidos (variável `Merc`). O arquivo relativo a este conjunto de dados foi obtido no site (`http://aerg.canberra.edu.au/envirostats/bm/introduction.htm`). Carregamos o arquivo e visualizamos a dispersão entre o conteúdo de mercúrio e o comprimento dos peixes:

```
> gemfish<-read.table("gemfish.txt",header=T)
> plot(gemfish$Len,gemfish$Merc, xlab="Comprimento (cm)",
ylab="Concentração de mercúrio em tecidos (mg/Kg)", pch=16,cex.lab=1.3)
```

O gráfico gerado é observado na Figura 33.

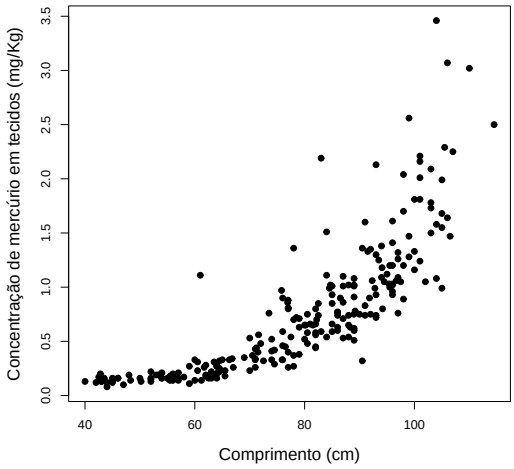

Figura 33: Diagrama de dispersão entre o Comprimento do corpo e a concentração de mercúrio em tecidos no peixe *Rexea solandri*.

Pelo diagrama de dispersão mostrado na Figura 33, percebemos que a relação entre as duas variáveis não é linear. Podemos ajustar um modelo linear aqui, mas já podemos imaginar que o resultado não será muito satisfatório:

```
> lm.gemfish<-lm(Merc ~ Len,gemfish)
> summary(lm.gemfish)
Call:
lm(formula = Merc ~ Len, data = gemfish)
Residuals:
      Min        1Q    Median        3Q       Max
 -0.77551  -0.22482  -0.06595   0.13670   1.98830
Coefficients:
              Estimate   Std.  Error  t value   Pr(>|t|)
 (Intercept)  -1.426346    0.105996    -13.46   <2e-16 ***
 Len           0.027866    0.001318     21.14   <2e-16 ***
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
Residual standard error:  0.364 on 240 degrees of freedom
Multiple R-Squared:  0.6506, Adjusted R-squared:  0.6492
F-statistic:  446.9 on 1 and 240 DF, p-value:  < 2.2e-16
```

O resultado é estatisticamente significativo, no entanto, os gráficos de diagnósticos mostram que as premissas do modelo linear não são atendidas (Figura 34):

```
> par(mfrow=c(2,2),pch=16)
```

```
> plot(lm.gemfish)
```

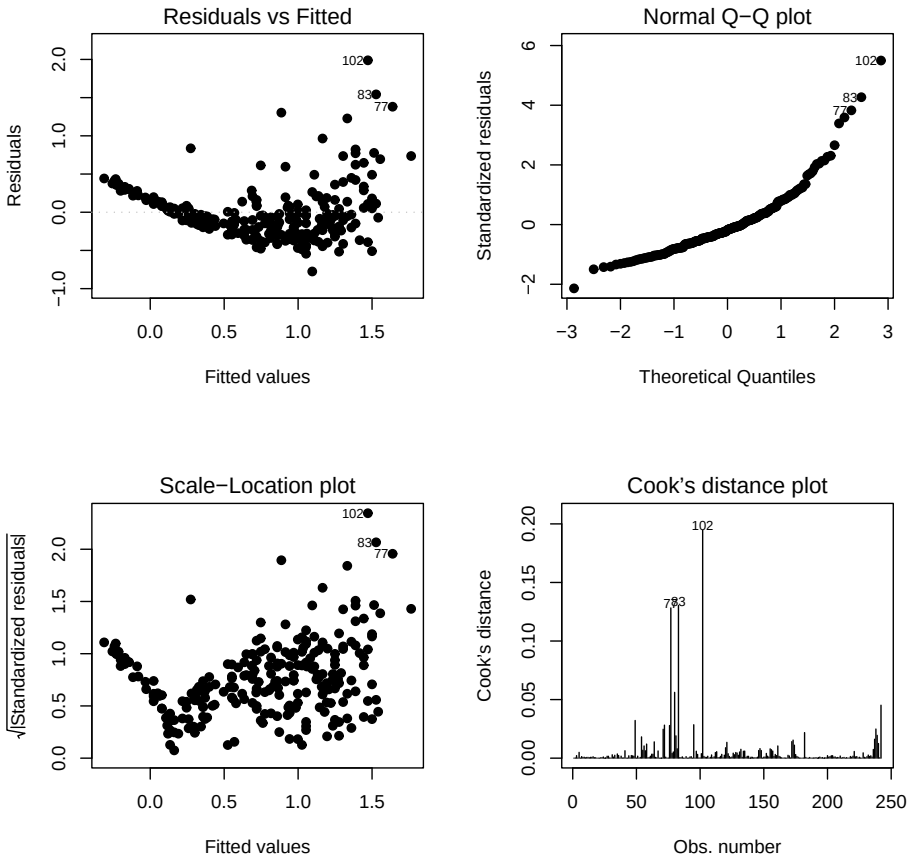

Figura 34: Gráficos de diagnóstico para o modelo `lm.gemfish`.

Na Figura 34, os resíduos mostram claramente um padrão de tendência não-linear. O gráfico normal mostra uma distribuição de resíduos com tendência assimétrica à direita (formando uma curva exponencial positiva). O gráfico Scale-Location mostra que a heteroscedasticidade não chega a ser um problema tão sério e as distâncias de Cook não apresentam nenhum valor grande demais. Chegamos à conclusão que este conjunto de dados apresenta um problema de linearidade. Podemos rodar o estimador de Box-Cox para ver qual seria a o valor de λ mais apropriado para realizar uma transformação da variável dependente:

```
> par(mfrow=c(1,2))
> boxcox(lm.gemfish)
> boxcox(lm.gemfish,lambda=seq(-0.2,0.1,by=0.01))
```

O resultado vemos na Figura 35.

Nesta figura, o intervalo de confiança para a estimativa do parâmetro λ inclui o 0, que significaria uma transformação logarítmica (apesar de não ser o valor com maior verossimilhança). Como a transformação logarítmica é bastante conhecida, é preferível utilizá-la, pois será mais fácil interpretar na escala decimal onde a variável foi medida. Quando temos um modelo linear em que

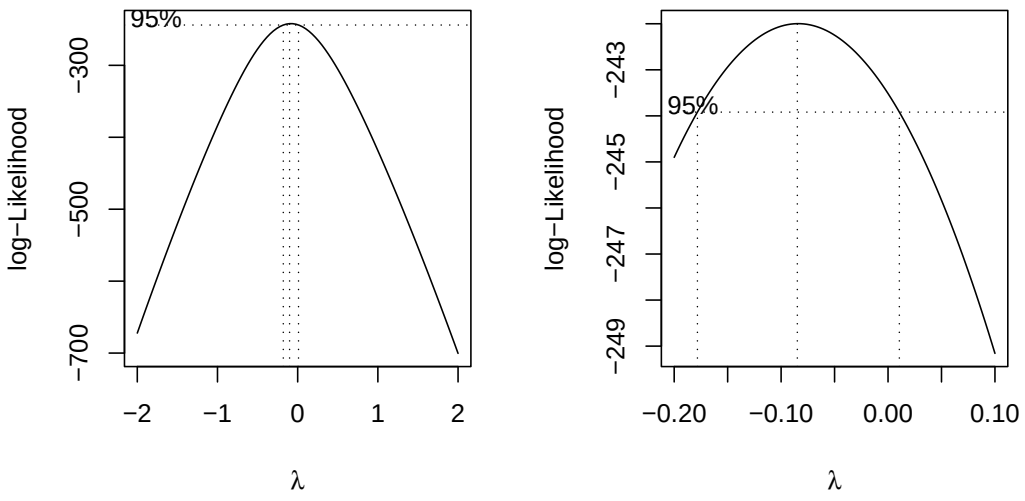

Figura 35: Estimação do parâmetro λ pelo método de Box-Cox para o modelo `lm.gemfish`.

a variável dependente foi logaritmizada (na base e), temos a seguinte equação:

$$\log(Y) = a + bX + \epsilon$$

Se transformarmos de volta à escala original, temos a seguinte relação:

$$Y = e^{(a+bX)} \times e^{\epsilon}$$

Neste caso, na escala original, o termo residual é multiplicativo e não aditivo. Quando este é o caso, a transformação logarítmica é adequada. por outro lado, se a relação for aditiva

$$Y = e^{(a+bX)} + \epsilon$$

O modelo não poderá ser linearizado pela transformação e devemos optar por um modelo não-linear. Na prática, não temos como saber se o resíduo é aditivo ou multiplicativo. A melhor opção é experimentar a transformação e ver se os resíduos passam a atender as premissas do modelo linear.

Vamos agora refazer o ajuste do modelo de regressão transformando a variável `Merc` em logaritmos naturais usando a função `log()`:

```
> lm.logfish<-lm(log(Merc) ~ Len,gemfish)
> summary(lm.logfish)
Call:
lm(formula = log(Merc) ~ Len, data = gemfish)
Residuals:
```

```
      Min        1Q    Median        3Q       Max
 -1.07019  -0.21346  -0.01326   0.18690   1.50932
Coefficients:
              Estimate   Std.  Error   t value   Pr(>|t|)
 (Intercept)  -4.166968    0.095181    -43.78   <2e-16 ***
 Len           0.045279    0.001184     38.25   <2e-16 ***
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
Residual standard error:  0.3268 on 240 degrees of freedom
Multiple R-Squared:  0.8591, Adjusted R-squared:  0.8585
F-statistic:  1463 on 1 and 240 DF, p-value:  < 2.2e-16
```

Comparando este ajuste com o prévio (sem a transformação), podemos notar uma diferença no coeficiente de determinação (que aumentou de 0,6506 para 0,8591), assim como uma diminuição do erro padrão residual. A reta de regressão para os dados transformados em logaritmos, assim como os intervalos de confiança de predição podem ser obtidos com os comandos abaixo e visualizados na Figura 36.

```
> new.x<-data.frame(Len=seq(40,120,by=1))
> pred.logfish<-predict(lm.logfish,new.x,interval="confidence")
> plot(gemfish$Len,log(gemfish$Merc), xlab="Comprimento (cm)",
ylab="log(Concentração de mercúrio)", cex.lab=1.3,pch=16)
> abline(lm.logfish,lwd=2)
> lines(new.x$Len,pred.logfish[,3],lty=2,lwd=2)
> lines(new.x$Len,pred.logfish[,2],lty=2,lwd=2)
```

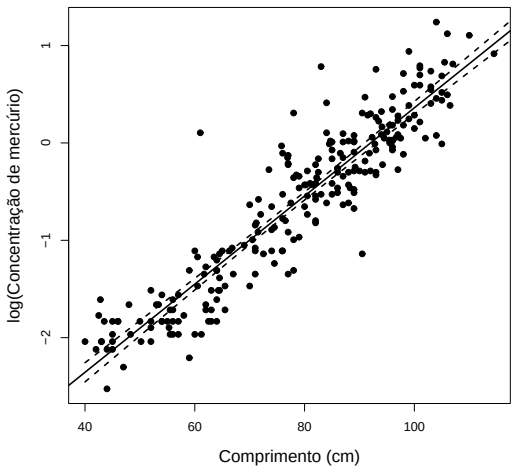

Figura 36: Diagrama de dispersão mostrando a concentração de mercúrio em tecidos (logaritmizada) com o tamanho dos indivíduos.

Por outro lado, se quisermos fazer um gráfico com a curva estimada na escala original, podemos transformar de volta as predições, assim como os intervalos de confiança para a escala decimal, usando a função `exp()`. O resultado do código abaixo pode ser visualizado na Figura 37:

```
> plot(gemfish$Len,gemfish$Merc, xlab="Comprimento (cm)",
ylab="log(Concentração de mercúrio)", cex.lab=1.3,pch=16)
> lines(new.x$Len,exp(pred.logfish[,2]),lty=2,lwd=2)
> lines(new.x$Len,exp(pred.logfish[,3]),lty=2,lwd=2)
> lines(new.x$Len,exp(pred.logfish[,1]),lwd=2)
```

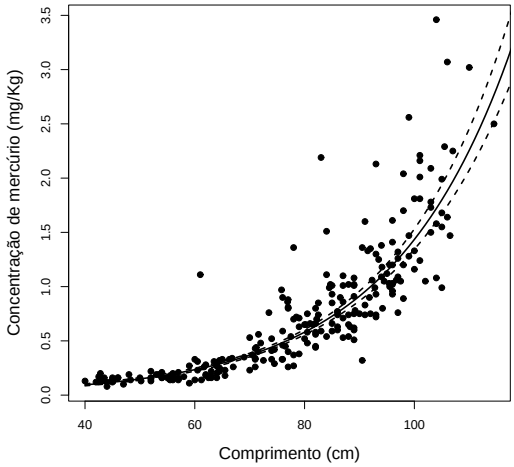

Figura 37: Diagrama de dispersão mostrando a curva de regressão do modelo ajustado em logaritmos, transformada de volta para a escala decimal original.

A Figura 36 mostra a dispersão do logaritmo natural (base e) da concentração de mercúrio em relação ao comprimento dos peixes. A reta de regressão com valores preditos apresenta a seguinte forma:

$$\log(Merc) = -4,166968 + 0,045279 \times Len$$

A Figura 37 mostra a dispersão dos valores de concentração de mercúrio na escala original em relação ao comprimento dos peixes. A curva mostrada no gráfico apresenta a seguinte forma:

$$Merc = e^{(-4,166968+0,045279 \times Len)}$$

Onde a transformação da escala logarítmica para a original foi realizada pela função `exp()` nos preditos e limites de confiança.

O gráfico de diagnósticos para este modelo obtido pelo comando `plot(lm.logfish)` (Figura 38) mostra que a logaritmização linearizou a relação entre as variáveis e normalizou os resíduos, fazendo com que os dados atendessem às premissas do modelo linear. A transformação de variáveis deve ser realizada com cautela, principalmente com relação à interpretação direta dos parâmetros e à comparação de modelos. Em algumas situações, a melhor opção é utilizar modelos apropriados como o linear generalizado. O modelo linear da variável transformada pode ser mais difícil de interpretar, visto que o coeficiente de regressão nos mostra uma taxa de aumento no logaritmo natural da concentração de mercúrio nos tecidos em relação ao aumento no tamanho corporal dos

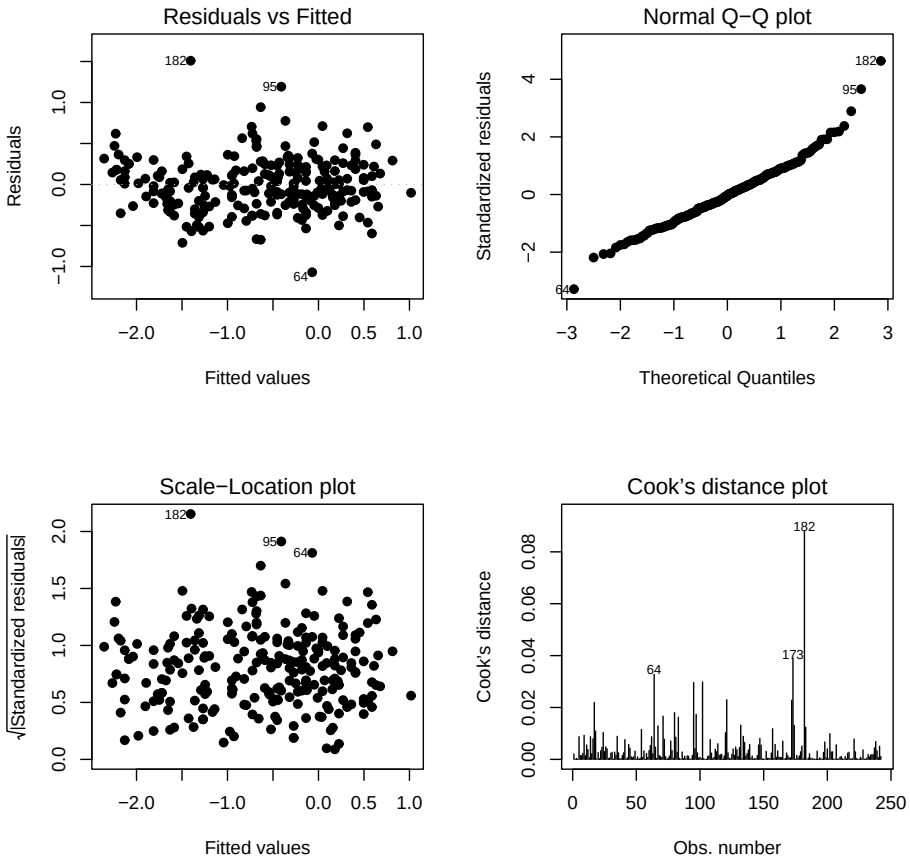

Figura 38: Gráficos de diagnóstico para o modelo `lm.logfish`.

peixes (não é uma relação muito direta) e não temos como inferir mais nada sobre o processo biológico responsável pelo acúmulo de mercúrio nos tecidos dos peixes maiores. No entanto, se o propósito do modelo estatístico for simplesmente a predição de valores de concentração de mercúrio a partir do comprimento dos peixes (com a aplicação em planejamento ambiental ou de saúde pública), o modelo com a variável transformada funciona muito bem. Existem situações em que é necessário também modificar a escala da variável independente. Neste caso, não temos como inferir a transformação ótima a partir do método de Box-Cox, e temos que tomar um cuidado especial se estivermos realizando uma regressão múltipla, pois as escalas de diferentes variáveis independentes podem requerer diferentes tranformações e a comparação de coeficientes de regressão pode ser difícil.

7.5 Coeficientes de correlação

Quando o propósito da análise é simplesmente calcular a magnitude da associação entre um par de variáveis, o método de regressão é irrelevante e o objetivo principal passa a ser estimar um parâmetro da distribuição normal bivariada: o coeficiente de correlação. Repare que, apesar da

estrita relação entre a reta de regressão e um coeficiente de correlação, o objetivo ou a pergunta do estudo deve determinar qual dos dois será utilizada. Para a análise de regressão, estamos interessados em estimar β, o coeficiente de regressão paramétrico que rege o aumento na variável dependente em relação ao aumento na variável independente. Quando não assumimos uma relação funcional entre as variáveis, não existe um β paramétrico a ser estimado, mas é possível que as duas variáveis distribuam-se de modo conjunto, segundo a distribuição bivariada normal, que é regida pelas médias (μ_{Y1} e μ_{Y2}) e desvios padrões (σ_{Y1} e σ_{Y2}) das duas variáveis e pelo coeficiente de correlação (ρ) que determina a força da associação entre as mesmas. Esta associação não necessariamente surge a partir da influência de uma sobre a outra, mas deve ser o produto de suas respectivas relações funcionais com uma terceira variável, que pode ser um fator mensurável ou latente (o que é a base do que conhecemos como análise multivariada).

A função para o cálculo do coeficiente de correlação no R é o comando cor(), onde os argumentos principais são as variáveis (Y_1 e Y_2) e o método (Pearson, Spearman ou Kendall). O método *default* é o de Pearson (produto-momento). Para calcularmos o coeficiente de correlação das variáveis no conjunto de dados das algas escrevemos:

```
> cor(algas$Abund,algas$DW)
[1] 0.8760333
```

Para calcular o coeficiente de correlação não paramétrico de Spearman

```
> cor(algas$Abund,algas$DW,method="spearman")
[1] 0.7897706
```

Note que o coeficiente *rho* de Spearman corresponde simplesmente a uma correlação de Pearson substituindo os valores das variáveis por postos (ranks):

```
> cor(rank(algas$Abund),rank(algas$DW))
[1] 0.7897706
```

Se quisermos calcular uma matriz de correlação entre p variáveis, simplesmente usamos como argumento o nome do data frame ou matriz cujas colunas queremos correlacionar:

```
> cor(algas)
             Abund          DW
 Abund   1.0000000   0.8760333
 DW      0.8760333   1.0000000
```

A função `cor.test()` permite a realização de um teste de significância para a hipótese H_0 : $\rho = 0$, usando os mesmos argumentos que a função `cor()` (com a diferença que a função de teste não permite o uso de um data frame ou matriz como argumento:

```
> cor.test(algas$Abund,algas$DW)
Pearson's product-moment correlation
data:  algas$Abund and algas$DW
t = 7.707, df = 18, p-value = 4.152e-07
alternative hypothesis:  true correlation is not equal to 0
95 percent confidence interval:
0.7079638 0.9501976
sample estimates:
cor
0.8760333
```

Podemos perceber pelo arquivo de saída, que é possível utilizar argumentos para modificar parâmetros do teste, como o método de correlação, a hipótese alternativa (bicaudal, unicaudal maior ou menor que 0) e o nível de confiança para o intervalo a ser estimado (só para o coeficiente produto-momento de Pearson).

```
> cor.test(algas$Abund,algas$DW,method="spearman")
Spearman's rank correlation rho
data:  algas$Abund and algas$DW
S = 280, p-value = 4.656e-05
alternative hypothesis:  true rho is not equal to 0
sample estimates:
rho
0.7897706
Mensagem de aviso:
p-values may be incorrect due to ties in:  cor.test.default(algas$Abund,
algas$DW, method = "spearman")
```

Além destas alternativas não-paramétricas, os limites de confiança e testes baseados em reamostragem podem ser realizados tanto para os coeficientes de regressão quanto para os coeficientes de correlação, como veremos na próxima seção.

7.6 Poder e erros do tipo II em regressão e correlação

Apesar de não existir nenhuma função que permita o cálculo do poder e erro do tipo II para uma análise de regressão já realizada, é possível programar uma função específica para o cálculo do poder para um teste de correlação baseado nas fórmulas disponíveis em livros texto como o de Zar (1984). Uma função para o cálculo do poder (com argumentos r = correlação observada, a = alfa desejado e n = tamanho da amostra) poderia ficar assim:

```
poder.cor<-function(r,a,n)
z.r<-(1/2)*log((1+r)/(1-r))
ra<- -qt(p=(a/2),df=(n-2))/(sqrt(n-2+(qt(p=(a/2),df=(n-2)))^2))
z.a<-(1/2)*log((1+ra)/(1-ra))
z.b<-(z.r-z.a)*(sqrt(n-3))
beta<-pnorm(-abs(z.b))
poder<-1-beta
return(list("beta"=beta,"poder"=poder))
```

As funções internas à `poder.cor()` calculam as transformadas de Fisher para a o valor observado de r e o valor crítico para o α desejado e $v = n - 2$ graus de liberdade. Para isso usamos a função de distribuição t (`qt`). O cálculo do z.b (Z_β) permite o cálculo do erro do tipo II e do poder a partir da função de distribuição normal (`pnorm()`). Mais detalhes podem ser encontrados em Zar (1984) e na apostila teórica. A função acima foi definida de modo a calcular o poder para o teste bicaudal ($H_1 : \rho \neq 0$). O resultado desta função para o exemplo das algas seria:

```
> poder.cor(0.87603,0.05,20)
$beta
[1] 0.0001391852
```

```
$poder
[1] 0.9998608
```

É possível construir uma curva de poder utilizando esta função, mas esta pode gerar resultados um tanto inesperados, tendo em vista que é o valor do r crítico tende a se modificar com os graus de liberdade e devemos considerar a curva apenas para os tamanhos amostrais cujos r críticos sejam menores que o ρ da H_1. Por exemplo, se para queremos planejar um estudo para encontrar uma associação entre variáveis, de modo que a magnitude do coeficiente de correlação esperado seja aproximadamente $\rho = 0,6$ e o teste de significância será realizado com $\alpha = 0,05$. Sabemos que, segundo a tabela dos r críticos, que apenas a partir de $n = 12$ ($v = 10$) o r crítico passa a ser menor que 0,6. De modo que só nos interessa então saber o poder para tamanhos amostrais a partir de 12. A Figura 39 mostra uma curva de poder calculada usando a função especificada:

```
> plot(x,poder.cor(0.6,0.05,x)$poder,type="l",lwd=2,xlab="Tamanho
amostral",ylab=expression(plain(Poder)~ ~ rho==0.6~ ~ alpha==0.05),cex.lab=1.3)
```

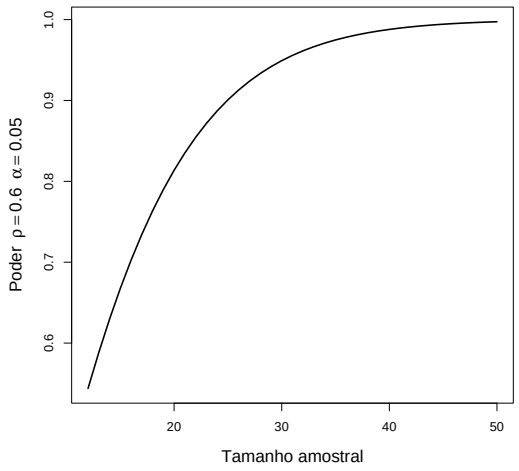

Figura 39: Curva de poder para diferentes tamanhos amostrais send a magnitude esperada para o coeficiente $\rho = 0,6$ e a probabilidade de erro do tipo I $\alpha = 0,05$.

7.7 Intervalos de confiança e testes baseados em reamostragem

Os procedimentos no R para o cálculo de intervalos de confiança *bootstrap* são semelhantes ao que vimos anteriormente. O grande segredo está na definição da função a ser reamostrada. Para começar com o exemplo mais simples, vamos calcular intervalos de confiança *bootstrap* para o coeficiente de correlação para o conjunto de dados das algas. Definir a função é muito fácil, visto que a própria função de correlação do R (`cor()`) já permite a extração desta informação:

```
> require(boot)
Loading required package:  boot
[1] TRUE
> ccor<-function(x,i) cor(x$Abund[i],x$DW[i])
```

```
> corboot<-boot(algas,ccor,10000)
> corboot
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = algas, statistic = ccor, R = 10000)
Bootstrap Statistics :
       original          bias   std.  error
 t1*  0.8760333  -0.005445315   0.06728858
```

E os intervalos de confiança:

```
> boot.ci(corboot)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates
CALL :
boot.ci(boot.out = corboot)
Intervals :
 Level        Normal              Basic
 95%    ( 0.7496, 1.0134 )  ( 0.7990, 1.0344 )
 Level      Percentile            BCa
 95%    ( 0.7177, 0.9531 )  ( 0.6133, 0.9430 )
Calculations and Intervals on Original Scale
```

É claro que não faz sentido encontrar um coeficiente maior que 1, de modo que preferimos utilizar o intervalo de confiança dos percentis. O histograma da distribuição do coeficiente de correlação mostra uma distribuição assimétrica à esquerda, como vemos na Figura 40:

```
> hist(corboot$t,xlab="Coeficiente de correlação",ylab="Frequência em
reamostragens",main=NULL,cex.lab=1.3)
```

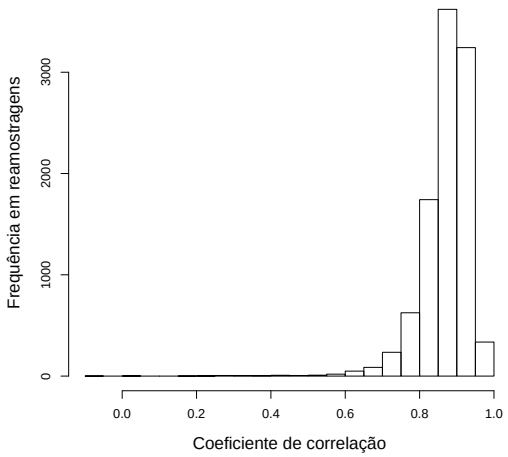

Figura 40: Histograma mostrando a distribuição dos coeficientes de correlação para 10.000 reamostragens *bootstrap* do conjunto de dados das algas.

Quando a estatística não apresenta distribuição normal nas reamostragens, não devemos utilizar a distribuição normal ou a distribuição t para gerar os limites de confiança ou testar hipóteses. A distribuição das estatísticas reamostradas pode ser utilizada para testar qualquer hipótese relativa ao coeficiente de correlação. Por exemplo, para a hipótese nula ($H_0 : \rho = 0$), podemos calcular a probabilidade de acordo com o número de reamostragens que apresentaram coeficientes de correlação menores ou iguais a 0:

```
> sum(corboot$t <= 0)/(corboot$R)
[1] 1e-04
```

Encontramos então, a probabilidade de 0,0001 (apenas 1 em 10000 reamostragens apresentou um coeficiente menor que 0). Uma outra maneira de testar a hipótese ($H_0 : \rho = 0$) é realizando uma permutação das observações em uma das variáveis, de modo que não seja esperado encontrar correlação significativa. Para isto, devemos redefinir a função a ser reamostrada de modo a realizar a permutação apenas em uma delas:

```
> ccor<-function(x,i) cor(x$Abund[i],x$DW)
> corperm<-boot(algas,ccor,10000)
> corperm<-boot(algas,ccor,10000,sim="permutation")
```

Na definição da função, só devemos indexar a variável a ser permutada (neste exemplo, Abund) e na hora de especificar os argumentos da função `boot` é importante definir `sim="permutation"` para que a reamostragem seja realizada sem reposição. O histograma da distribuição dos coeficientes reamostrados com permutação mostra uma grande diferença em relação ao histograma da distribuição *bootstrap* (Figura 41)

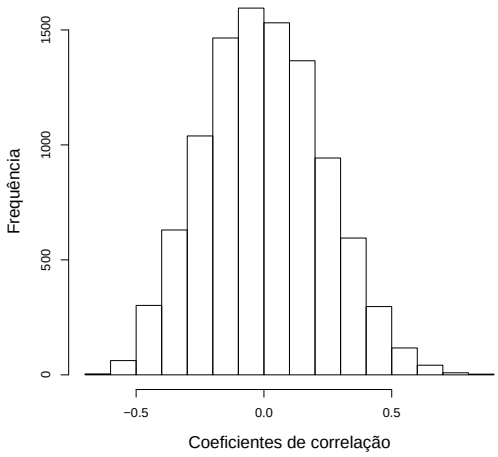

Figura 41: Histograma mostrando a distribuição dos coeficientes de correlação para 10.000 reamostragens com permutação do conjunto de dados das algas.

Podemos testar a hipótese nula $H_0 : \rho = 0$ calculando quantos coeficientes maiores ou iguais ao observado foram encontrados nas reamostragens. Para um teste bicaudal, devemos considerar apenas o módulo do coeficiente:

```
> corperm$t0
```

```
[1] 0.8760333
> sum(abs(corperm$t) >= corperm$t0)/corperm$R
[1] 0
```

Como nenhuma reamostragem mostrou um coeficiente tão alto quanto o observado, o valor de P é ainda menor que 0,0001. Podemos verificar isto pelo mínimo e máximo da distribuição das amostras permutadas:

```
> list(min(corperm$t),max(corperm$t))
[[1]]
[1] -0.6161695
[[2]]
[1] 0.8671825
```

Os mesmos intervalos e testes podem ser utilizados para o coeficiente de regressão, precisamos apenas redefinir a função que será reamostrada. No caso dos intervalos de confiança *bootstrap*, definimos a função da seguinte maneira:

```
> breg<-function(x,i) coef(lm(x$Abund[i] ~ x$DW[i]))[2]
> bboot<-boot(algas,breg,10000)
> bboot
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = algas, statistic = breg, R = 10000)
Bootstrap Statistics :
      original        bias  std.  error
 t1*  25.91785  -0.1828422     3.682918
```

Repare que extraímos do modelo linear (função `lm()`) apenas a informação que queremos (função `coef()`): o segundo coeficiente de regressão (b). Se não determinarmos o índice [2], a função irá calcular os dois parâmetros da equação da reta. Visualizando a distribuição do coeficiente de regressão bootstrap pelos percentiles de 0,025 e 0,975 e o diagrama boxplot (Figura 42):

```
> quantile(bboot$t,c(0.025,0.5,0.975))
2.5% 50% 97.5%
18.01195 25.86006 32.70028
> boxplot(bboot$t,ylab="Coeficiente de regressão",cex.lab=1.3)
```

Comparando estes limites com os obtidos pela distribuição normal e o erro padrão do coeficiente de regressão calculado pelo modelo linear (18.8526 e 32.9834), percebemos que a diferença é pequena, assim como é pequena a diferença entre a mediana estimada pelo bootstrap e o coeficiente original calculado (25,91785). Isto acontece porque este conjunto de dados já atende as premissas do modelo linear e não esperamos encontrar muita diferença entre os diferentes métodos de estimação.

Em nenhuma das reamostragens encontramos coeficientes de regressão menores ou iguais a 0, de modo que a hipótese nula ($H_0 : \beta = 0$) pode ser rejeitada. Vamos mostrar agora como seria o teste de permutações para esta hipótese nula:

```
> breg<-function(x,i) coef(lm(x$Abund[i] ~ x$DW))[2]
> permreg<-boot(algas,breg,10000,sim="permutation")
```

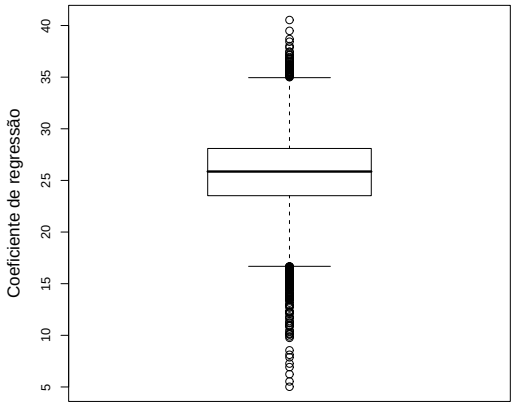

Figura 42: Diagrama boxplot para a distribuição dos coeficientes de regressão *bootstrap*.

A distribuição dos coeficientes de regressão permutados pode ser observada na Figura 43.

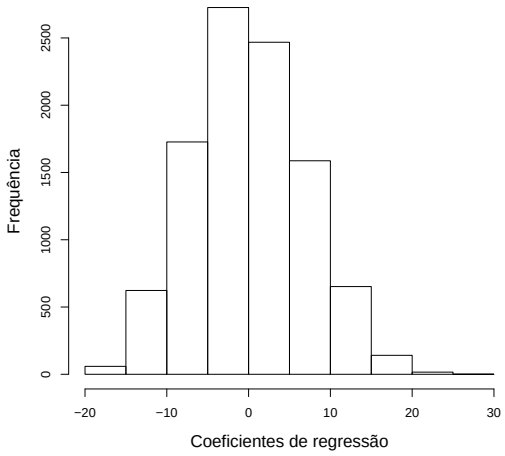

Figura 43: Histograma para a distribuição dos coeficientes de regressão permutados.

O valor de probabilidade pode ser calculado como:

```
> sum(abs(permreg$t) >= permreg$t0)/permreg$R
[1] 0
```

Nenhum valor absoluto dos coeficientes de regressão permutados foi igual ou maior que o observado. O que significa que $P < 0{,}0001$. É possível realizar a permutação ou o intervalo *bootstrap* para o parâmetro de intercepto, mas o interesse biológico neste parâmetro é menor. O intervalo *bootstrap* também pode ser utilizado para a comparação dos coeficientes em diferentes retas de regressão.

7.8 Análise de variância

Os modelos de análise de variância simples podem ser trabalhados também a partir da função `lm()`, simplesmente indicando um fator (vetor com níveis de uma variável categórica) como variável independente. No entanto, existem outras funções que também realizam análises de variância para casos específicos (como a `aov()` para amostras balanceadas) ou modelos lineares mais complexos, como análises de modelos II, modelos mistos ou hierárquicas (função `lme()` do pacote `nlme`).

7.8.1 Análise de variância de classificação simples com efeitos fixos (One-Way ANOVA)

Para iniciar nossa análise vamos carregar um conjunto de dados novo a ser analisado. O nome do arquivo é `macroinvert.txt`, contendo 80 observações realizadas em quadrados de 30x30 cm amostrados no sedimento até uma profundidade de 10 cm em 8 sítios diferentes (10 amostras por sítio), de modo que os invertebrados bentônicos foram triados e contados. Mais informação sobre os diferentes sítios (estes variam de acordo com a distância antes e depois de um ponto de descarga de esgoto no Rio Crackenback, na Austrália) pode ser encontrada no próprio arquivo e no site `http://aerg.canberra.edu.au/envirostats/bm/workbooks.htm`. Abrindo o arquivo em um editor de texto comum, encontraremos mais informação sobre o conjunto de dados em comentários. O R considera qualquer linha iniciada pelo símbolo # como um comentário e ignora os caracteres. Desta maneira, podemos gravar informação sobre o conjunto de dados dentro de um arquivo sem que esta informação seja lida pelo R. Basicamente, Os sítios foram definidos ao longo do Rio da seguinte maneira:

Site 1: 1 km acima da cidade de Thredbo;

Site 2: 1 km abaixo da cidade;

Site 3: 1.5 km abaixo da cidade (ponto de despejo de esgoto);

Site 4: 0.2 km abaixo do ponto de despejo;

Site 5: 1 km abaixo do ponto de despejo;

Site 6: 3 km abaixo do ponto de despejo;

Site 7: 4.5 km abaixo do ponto de despejo;

Site 8: 8 km abaixo do ponto de despejo.

Carregando o arquivo:

```
> invert<-read.table("macroinvert.txt",header=T)
```

Vemos que este arquivo apresenta duas variáveis (`Abm` = abundância de macroinvertebrados e `Site` = local onde as amostras foram coletadas). Sempre que carregamos um conjunto de dados, o primeiro passo é examinar a distribuição para tentar identificar possíveis problemas antes mesmo de realizarmos as análises. Quando pretendemos comparar uma variável dependente contínua (ou merística como é o caso) com uma variável independente categórica, a ferramenta gráfica mais apropriada é o boxplot, como vemos na Figura 44:

```
> boxplot(invert$Abm~invert$Site, xlab="Sítios de coleta",ylab= "Abundância de
Macroinvertebrados ",cex.lab=1.3)
```

Neste tipo de figura, devemos procurar por observações atípicas (outliers), assimetria nas caixas e heteroscedasticidade (caixas de diferentes tamanhos). Podemos perceber que no nosso caso parece que alguns sítios apresentam mais variabilidade que outros. Adiante teremos a oportunidade de verificar se esta aparente quebra de premissa é muito séria. Devemos ajustar um modelo linear

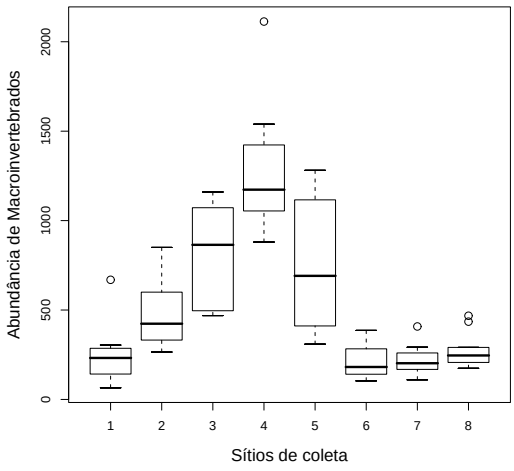

Figura 44: Abundância de macroinvertebrados em sítios do Rio Crackenback, Austrália

usando a função `lm()`, mas é importante que digamos ao programa que a variável `Site` é um fator e não uma variável numérica merística (os sítios estão numerados de 1 a 8 de acordo com sua posição no rio). Caso deixemos de fatorizar a variável `Site`, a função de modelo linear tentaria realizar uma regressão. Podemos ajustar o modelo linear da seguinte maneira:

```
lm.invert<-lm(Abm~factor(Site),data=invert)
```

Utilizando a função `factor()` para transformar a variável independente. O resultado pode ser visualizado com as funções `summary()` e `anova()`:

```
> summary(lm.invert)
Call:
lm(formula = Abm ~ factor(Site), data = invert)
Residuals:
Min 1Q Median 3Q Max
-469.20 -107.60 -31.00 90.58 838.90
Coefficients:
                Estimate  Std.  Error  t value  Pr(>|t|)
 (Intercept)     248.90      73.54      3.385    0.00116   **
 factor(Site)2   229.90     104.00      2.211    0.03024    *
 factor(Site)3   570.00     104.00      5.481   5.92e-07  ***
 factor(Site)4  1025.20     104.00      9.858   5.39e-15  ***
 factor(Site)5   529.30     104.00      5.089   2.76e-06  ***
 factor(Site)6   -38.30     104.00     -0.368    0.71375
 factor(Site)7   -29.20     104.00     -0.281    0.77969
 factor(Site)8    29.40     104.00      0.283    0.77822
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
Residual standard error:  232.5 on 72 degrees of freedom
```

```
Multiple R-Squared:  0.7279, Adjusted R-squared:  0.7014
F-statistic:  27.51 on 7 and 72 DF, p-value:  < 2.2e-16
> anova(lm.invert)
Analysis of Variance Table
Response:  Abm
               Df    Sum Sq    Mean Sq  F value    Pr(>F)
 factor(Site)   7   10414757   1487822   27.512   < 2.2e-16   ***
 Residuals     72   3893704    54079
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

Agora vamos examinar e interpretar estes resultados. O coeficiente estimado para o intercept corresponde a um valor arbitrário (a média do primeiro nível, o Site 1) que é utilizado como base de comparação. Os coeficientes para os níveis subsequentes correspondem a diferenças entre a média de cada nível e a média utilizada como base. Por exemplo, a média do Site 2 é 229.9 unidades maior que a média do Site 1, e a média do Site 4 (200 metros abaixo do ponto de despejo de esgotos) é 1025.20 unidades maior que a média do Site 1. Estes coeficientes correspondem aos efeitos (fixos) do modelo I de análise de variância. Além dos efeitos, a função `summary()` mostra a estatística F e o valor de P associado. Esta é, no entanto, melhor interpretada a partir da tabela de análise de variância mostrada com a função `anova()`. O valor de F observado é grande o bastante para que a hipótese nula ($H_0 : \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = \alpha_7 = \alpha_8$) seja aceita. Antes de seguir com os desdobramentos da análise (comparações múltiplas para modelo I ANOVA) precisamos examinar os gráficos diagnósticos obtidos a partir da função `plot()` e observados na Figura 45.

```
>par(mfrow=c(2,2),pch=16)
>plot(lm.invert)
```

O exame da Figura 45 sugere que o principal problema deste conjunto de dados pode ser a heteroscedasticidade, tendo em vista que a relação entre resíduos e médias (*Fitted values*) parece linear e os resíduos parecem apresentar uma distribuição aproximadamente normal. O teste geral de hipótese realizado na tabela de ANOVA é relativamente robusto com relação à heteroscedasticidade, no entanto, as comparações entre níveis (comparações múltiplas) podem ficar prejudicadas, tendo em vista que é irreal neste caso tentar estimar um erro padrão residual (o erro padrão comum a todos os grupos). Existem testes específicos para lidar com amostras heteroscedásticas ou podemos tentar uma transformação, como veremos adiante. Podemos verificar numericamente se a tendência visível de heteroscedasticidade é confirmada por algum teste. Existem vários testes sugeridos para verificar a homoscedasticidade ds níveis na análise de variância. Vários deles são muito sensíveis à quebra de outras premissas como a normalidade, por exemplo. Um teste sugerido por Faraway (2002, *Practical regression and anova using R* - apostila encontrada no site do R) é o de Levene, onde simplesmente realizamos uma análise de variância nos valores absolutos dos resíduos. Segundo Faraway, só precisaremos nos preocupar com este problema se o resultado for significativo ao nível de 1%. Os comandos para executar este teste no R são:

```
> anova(lm(abs(lm.invert$res)~factor(invert$Site)))
Analysis of Variance Table
Response:  abs(lm.invert$res)
```

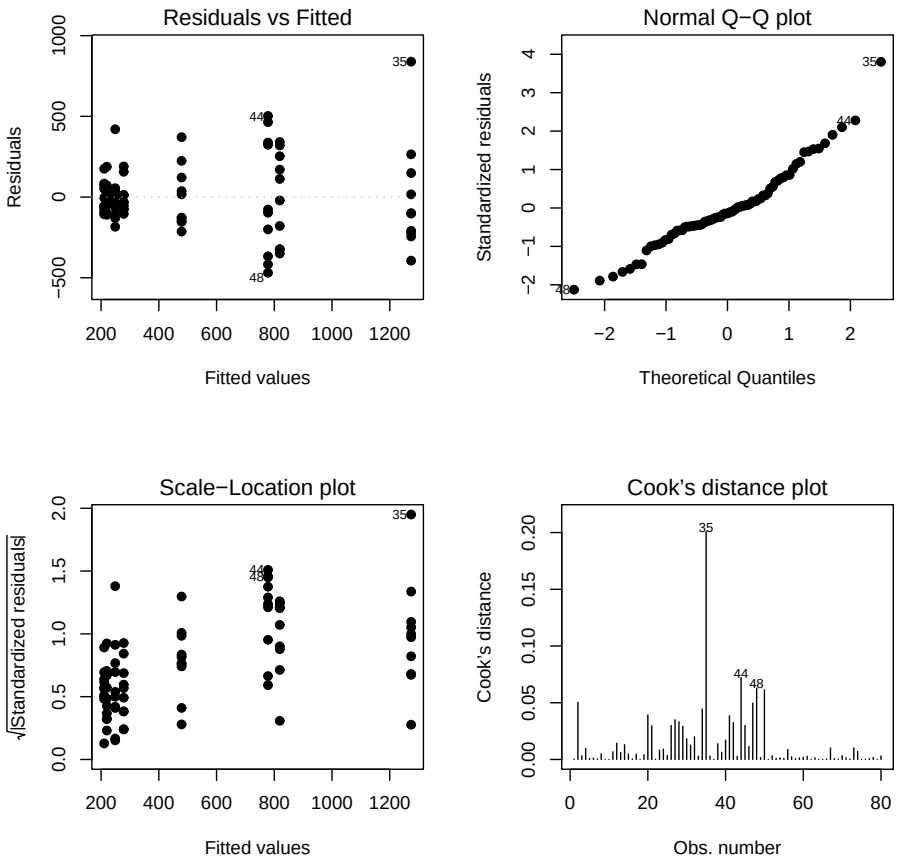

Figura 45: Gráficos diagnósticos para o modelo de análise de variância entre sítios.

```
                       Df   Sum Sq   Mean Sq  F value    Pr(>F)
 factor(invert$Site)    7   689736    98534    6.3764   7.647e-06  ***
 Residuals             72  1112605    15453
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

Como observamos uma significância estatística neste resultado, devemos rejeitar a hipótese nula de que as variâncias sejam as mesmas em todos os níveis do fator.

Podemos buscar uma transformação para as variáveis de modo a homogeneizar as variâncias. Os dados originados por contagens são conhecidamente heteroscedásticos e geralmente são transformados em logaritmos ou raiz quadrada. Podemos utilizar a função boxcox do pacote MASS para ver qual transformação seria mais recomendada neste caso (Figura 46):

```
> boxcox(lm.invert,lambda=seq(-0.5,0.5,by=0.01))
```

Este gráfico mostra claramente que o intervalo de confiança de 95% para a estimativa de λ inclui o 0, sugerindo que uma transformação logarítmica apresentaria mais verossimilhança que a raiz quadrada ($\lambda = 0,5$). Vamos então refazer a análise transformando a variável dependente Amb em logaritmos naturais. Não é o caso do nosso exemplo, mas quando temos valores na variável

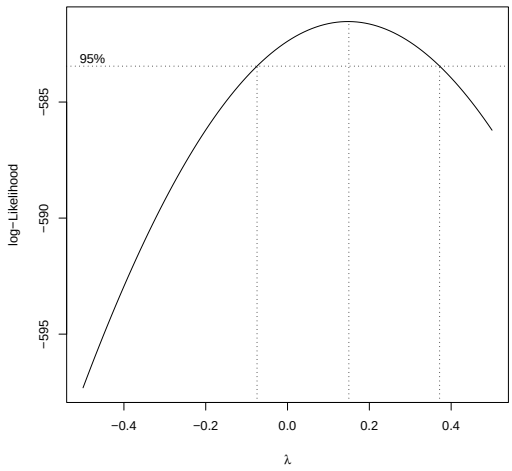

Figura 46: Intervalo de confiança de 95% para a estimativa de λ.

dependente iguais a 0, é interessante utilizar a transformação $Y' = \log(Y + 1)$, de modo a evitar valores indefinidos quando Y = 0. A distribuição da variável de abundância logaritmizada pode ser vista na Figura 47.

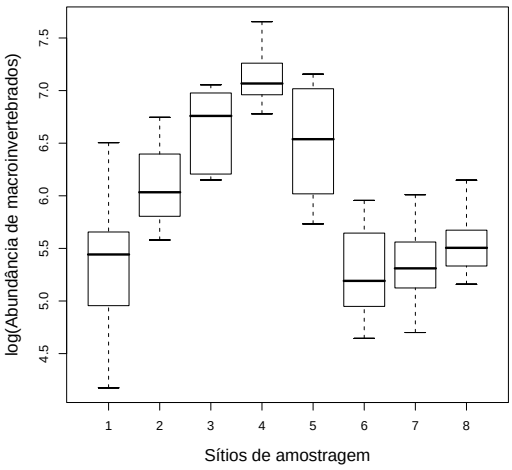

Figura 47: Diagrama boxplot para a variável de abundância logaritmizada.

Refazendo a análise para a variável transformada teremos:

```
> lm.loginv<-lm(log(Abm)~factor(Site),data=invert)
> anova(lm.loginv)
Analysis of Variance Table
Response:  log(Abm)
```

```
                Df  Sum Sq  Mean Sq  F value    Pr(>F)
 factor(Site)   7  35.854   5.122    27.976   < 2.2e-16  ***  --
 Residuals      72  13.182   0.183
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

A diferença entre os valores da estatística F é pequena, mas desta vez, as variâncias são homoscedásticas, como podemos visualizar nos gráficos diagnósticos (Figura 48) e no teste de Levene para os dados transformados, onde a hipótese nula de homoscedasticidade não é rejeitada:

```
>plot(lm.loginv)
> anova(lm(abs(lm.loginv$res)~factor(invert$Site)))
Analysis of Variance Table
Response:  abs(lm.loginv$res)
                     Df   Sum Sq   Mean Sq  F value  Pr(>F)
 factor(invert$Site)   7   0.5462   0.0780   1.4042   0.2170
 Residuals            72   4.0008   0.0556
```

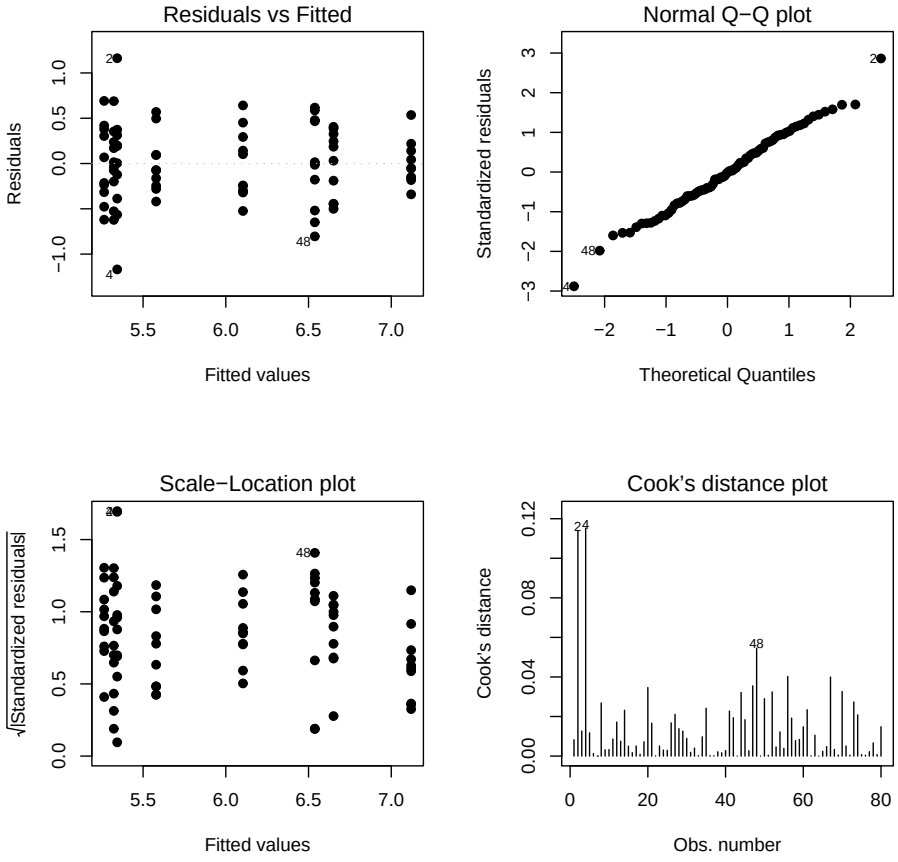

Figura 48: Gráficos diagnósticos para o modelo com a abundância logaritmizada.

Podemos agora prosseguir com as comparações múltiplas, visto que, em unidades logarítmicas podemos calcular um erro padrão residual comum a todos os níveis. Neste ponto precisamos

decidir, dependendo do delineamento experimental e estatístico do projeto, qual seria o método mais adequado para as comparações. Se o estudo for planejado desde o princípio de modo a exigir a comparação de apenas algumas poucas comparações (por exemplo, comparar o suposto "controle"do sítio 1 com os outros sítios), pode ser mais poderoso utilizar a diferença mínima significante (LSD) baseada na distribuição t de Student e corrigir o nível de significância a ser adotado pelo método de Bonferroni ou de Dunn-Sidak. O R possui uma função para comparações múltiplas chamada `TukeyHSD()` que calcula intervalos de confiança para as diferenças baseados na distribuição do *studentized range* (q), e cujo argumento principal é um objeto criado pela função `aov()`. Tecnicamente, esta função (assim como a `aov()`) deve ser utilizada apenas para delineamentos estaítísticos balanceados, no entanto, os autores incorporaram uma correção para o tamanho amostral que produz intervalos de confiança razoáveis para delineamento pouco desbalanceados. No nosso exemplo, o delineamento é balanceado (10 observações por nível do fator), de modo que podemos utilizar esta função normalmente. O resultado da função `TukeyHSD()` é uma matriz em que as linhas correspondem a todas as comparações possíveis (1-2,1-3,1-4,etc...), ao passo que as colunas correspondem à distância (diferença) entre os dois níveis, os limites inferior e superior do intervalo de confiança. Quando os limites de confiança sobrepõem-se ao 0, quer dizer que a diferença entre os dois níveis não é estatísticamente diferente de 0 (ao nível de significância desejado), ao passo que se o intervalo de confiança não inclui o 0, a diferença é significante. Podemos utilizar o comando:

```
> TukeyHSD(aov(log(invert$Abm)~factor(invert$Site)))
Tukey multiple comparisons of means
95% family-wise confidence level
Fit:  aov(formula = log(invert$Abm) ~ factor(invert$Site))
$"factor(invert$Site)"
         diff         lwr         upr
 2-1  0.76034124  0.16296125  1.35772122
 3-1  1.30754269  0.71016270  1.90492267
 4-1  1.77671973  1.17933975  2.37409972
 5-1  1.19445166  0.59707168  1.79183165
```

Repare que apenas as quatro primeiras comparações de 28 possíveis. É importante lembrar que estes valores de diferenças e intervalos estão em logaritmos, e por ser esta uma função não linear, os logaritmos das diferenças não são iguais às diferenças dos logaritmos. O importante aqui é determinar quais sítios são diferentes de quais. Nas quatro comparações mostradas, as diferenças são significativas, tendo em vista que nenhum intervalo compreende o 0. Os intervalos de confiança são melhor visualizados a partir da função plot() em conjunto com `TukeyHSD()` (Figura 49):

```
> plot(TukeyHSD(aov(log(invert$Abm)~factor(invert$Site))))
```

O padrão de diferenças não significativas mostra claramente a existência de dois grupos: um formado pelos sítios 1,6,7 e 8, outro formado pelos sítios 3,4 e 5. O sítio 2 assume uma posição intermediária, não sendo estatisticamente diferente do 3, do 5 e do 8. A interpretação deste resultado sugere que a partir do sítio 2 (primeiro abaixo da cidade) já existe algum impacto no rio. O despejo do esgoto seria um fator bastante significativo, aumentando muito a abundância de invertebrados, mas a partir de 3 km abaixo do local de despejo de esgoto, os efeitos já não são mais significativos. É claro que no estudo original, o pesquisador (David Tiller) se preocupou em

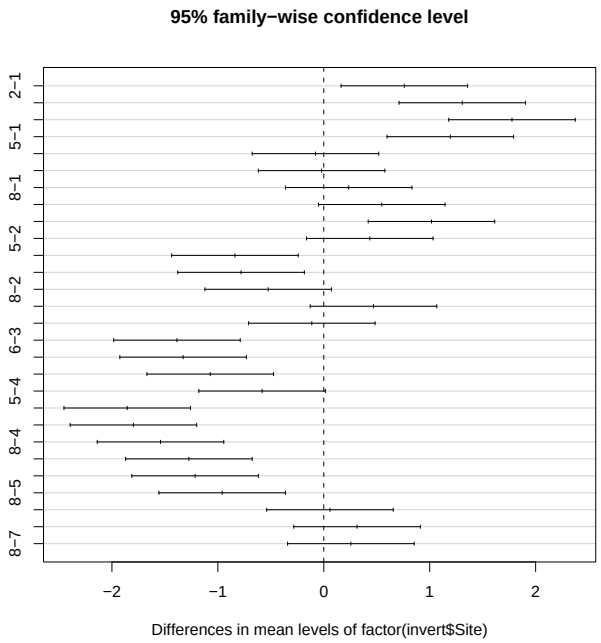

Figura 49: Intervalos de confiança para comparações múltiplas.

amostrar estes locais todos os meses durante três anos de modo a verificar a existência de variações temporais neste padrão de diferenças. Outro ponto importante é a validade externa deste estudo. Claro que, sem uma quantidade de repetições espaciais em outros rios onde a mesma situação ocorra, é impossível discernir o efeito do esgoto de qualquer outro fator que possa estar causando esta variação na abundância de macroinvertebrados sem ser detectado (isto também foi feito pelo grupo de pesquisa do autor, mas com um delineamento bem mais complicado e com um número maior de variáveis). Existe um pacote específico para comparações múltiplas no R (`multcomp`), que oferece uma maior variedade de métodos aplicáveis a um maior número de delineamentos experimentais. Este deve ser buscado se o usuário necessitar de mais opções para suas análises.

7.8.2 Análise de variância de classificação simples com efeitos aleatórios

Vamos agora examinar um delineamento modelo II ANOVA, em que o fator apresenta efeitos aleatórios. Neste caso, o desdobramento de interesse é o cálculo do componente de variância atribuível a cada fator. Este tipo de modelo pode ser ajustado pela função `lm()` ou pela função `lme()` do pacote `nlme - Linear and non linear mixed effects models`. Esta função é muito mais complicada que a função lm(), mas pode ser a melhor opção disponível para um grande número de delineamentos com modelos mistos (aqueles em que há uma mistura de fatores fixos e aleatórios), como veremos adiante.

Vamos carregar o conjunto de dados contendo comprimentos de asas de abelhas operárias de 4 colônias diferentes:

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

Neste arquivo encontramos duas variáveis: CA e Col, contendo os comprimentos das asas e os

agrupamentos por colônias, respectivamente. Como grupos são definidos pela variável Col usando caracteres alfanuméricos, o R reconhece automaticamente esta variável como sendo um fator, não sendo necessário definir isto na fórmula. Ajustando o modelo linear para este conjunto de dados:

```
> lm.bee<-lm(CA~Col,data=bee)
```

temos o seguinte resultado:

```
> anova(lm.bee)
Analysis of Variance Table
Response:  CA
            Df   Sum Sq   Mean Sq  F value     Pr(>F)
 Col         3  0.52155  0.17385   6.1798  0.001691 **
 Residuals  36  1.01275  0.02813
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

Obviamente, antes de analisar e interpretar estes resultados, precisamos checar os gráficos diagnósticos e o diagrama boxplot para a distribuição da variável dependente dentro dos grupos (Figura 50).

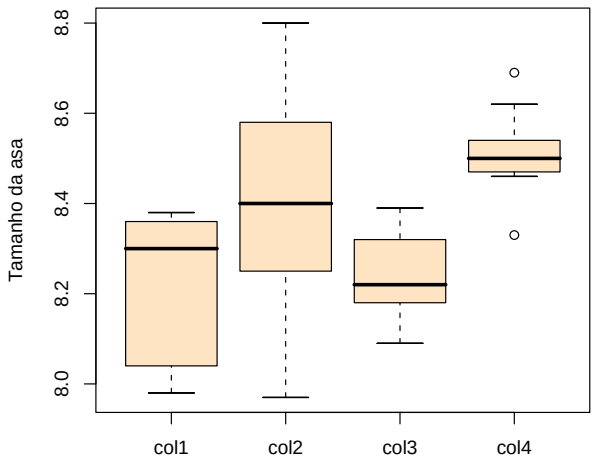

Figura 50: Diagrama boxplot para o comprimento das asas de abelhas operárias em 4 colônias.

Apesar das diferenças de tamanho entre caixas sugerirem uma certa heteroscedasticidade, os gráficos diagnósticos e o teste de Levene mostram que as variâncias dos resíduos são homogêneas entre os grupos (resultados não mostrados aqui). Uma vez que a tabela de análise de variância para este fator mostra um resultado significativo, inferimos a existência de um componente de variância (σ_A^2) diferente de 0. O desdobramento natural para um modelo II ANOVA é a estimativa do componente de variância para o termo de efeitos aleatórios. Como não temos nenhuma função específica para este, precisamos utilizar a informação fornecida pelo ajuste do modelo linear para esta estimativa. Sabemos que a tabela de análise de variância criada com a função `anova()`, a

partir de um modelo linear ajustado é uma lista de onde podemos retirar a informação necessária. Por exemplo, os quadrados médios estão armazenados em um vetor (`anova(lm.bee)$Mean`), onde o primeiro elemento é o quadrado médio explicado e o segundo elemento é o quadrado médio do resíduo:

```
> anova(lm.bee)$Mean
[1] 0.17384917 0.02813194
```

Como a fórmula para o componente de variância para amostras balanceadas seria:

$$s_A^2 = \frac{QM_E - QM_D}{n}$$

Onde QM_E representa o quadrado médio entre grupos, QM_D representa o quadrado médio residual e n representa o número de observações por grupo. O nosso cálculo do componente seria:

```
> (anova(lm.bee)$Mean[1]-anova(lm.bee)$Mean[2])/10
[1] 0.01457172
```

Da mesma maneira podemos calcular o coeficiente de correlação intraclasse:

$$t = \frac{s_A^2}{s_A^2 + s^2}$$

```
> vc<-(anova(lm.bee)$Mean[1]-anova(lm.bee)$Mean[2])/10
> vc/(vc+anova(lm.bee)$Mean[2])
[1] 0.3412288
```

Se quisermos gerar intervalos de confiança para estas estimativas do componente de variância podemos utilizar as fórmulas disponíveis nos livros texto (apenas para amostras balanceadas), ou realizar reamostragens para gerar um intervalo de confiança *bootstrap*. Precisamos definir uma função para o cálculo do componente de variação (ou do coeficiente de correlação intraclasse) a ser utilizada na função `boot()`.

```
> vc<-function(x,i)
+ lmx<-lm(x$CA[i]~x$Col[i])
+ vc<-(anova(lmx)$Mean[1]-anova(lmx)$Mean[2])/10
+ return(vc)
>boot.vc<-boot(bee,vc,1000,strata=bee$Col)
> hist(boot.vc$t)
> quantile(boot.vc$t,c(0.025,0.975))
2.5% 97.5%
0.00585274 0.03229858
```

É importante definir no caso do bootstrap que a reamostragem será realizada dentro dos grupos, o que é realizado pelo argumento `strata` da função `boot()`. O histograma com a distribuição pode ser encontrado na Figura 51

O mesmo processo pode ser utilizado para o cálculo de intervalos de confiança *bootstrap* para o coeficiente de correlação intraclasse.

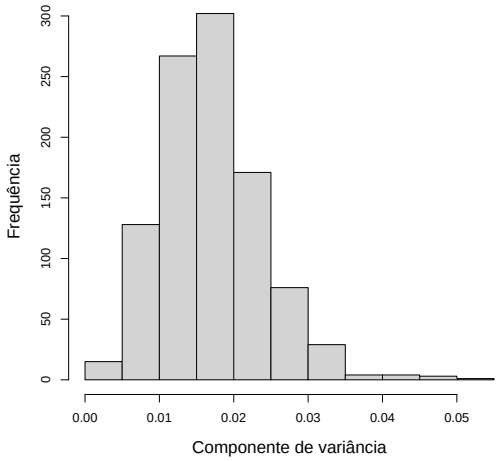

Figura 51: Histograma mostrando a distribuição dos componentes de variância reamostrados.

7.8.3 Testes de permutação para análises de variância

Muitas vezes, quando realizamos análises de variância, o interesse maior está no teste de hipótese que na estimação de efeitos ou componentes de variação. Neste caso, o método de reamostragem mais utilizado será o de permutação. Para realizar testes de permutação para um modelo de análise de variância, a estatística de interesse a ser calculada nas amostras permutadas será a estatística F, que pode ser acessada como parte integrante da lista gerada pela função `anova()`. Vamos definir a função abaixo:

```
> fstat<-function(x,i) anova(lm(x$CA[i]~x$Col))$F[1]
> boot.f<-boot(bee,fstat,10000,sim="permutation")
> hist(boot.f$t)
> sum(boot.f$t >= boot.f$t0)/10000
[1] 0.0019
```

Podemos perceber, como esperado para esta estatística, que o histograma gerado é bastante assimétrico (Figura 52).

Em 10000 reamostragens, observamos 19 valores de F tão altos ou mais que observado (uma probabilidade inferida de erro do tipo I um pouco maior que a do teste paramétrico, onde $P = 0{,}001691$).

7.8.4 Análise de variância com dois fatores (Two-Way)

Os delineamentos para este tipo de análise começam a se complicar um pouco mais, visto que precisaremos analisar os efeitos de dois fatores separadamente e possivelmente sua interação. Vamos começar com um exemplo em que os dois fatores são fixos. O arquivo de dados drosop.txt contém um conjunto de dados com comprimentos de asas de moscas (*Drosophila pseudoobscura*) criadas em laboratório com duas temperaturas diferentes (15 e 19 graus C) e duas densidades diferentes (10 e 100 indivíduos por garrafa). O arquivo possui três colunas (CA, Temp, Dens), contendo

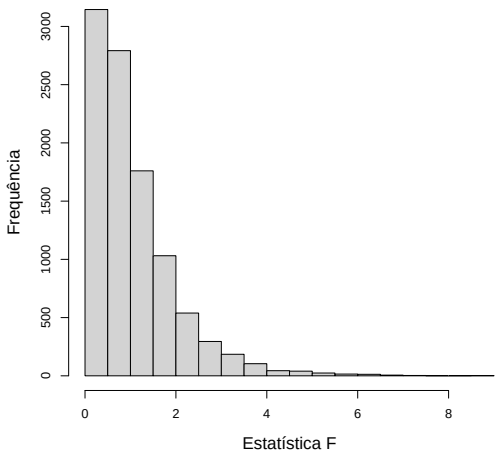

Figura 52: Histograma mostrando a distribuição da estatística F para as amostras permutadas.

os comprimentos das asas, a temperatura e a densidade para cada indivíduo medido (total de 40 indivíduos).

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

E o diagrama boxplot separando as observações de acordo com os fatores pode ser observado na Figura 53.

```
> boxplot(CA~Temp+Dens,xlab="Temp.Dens",ylab="Comprimento da asa",cex.lab=1.3)
```

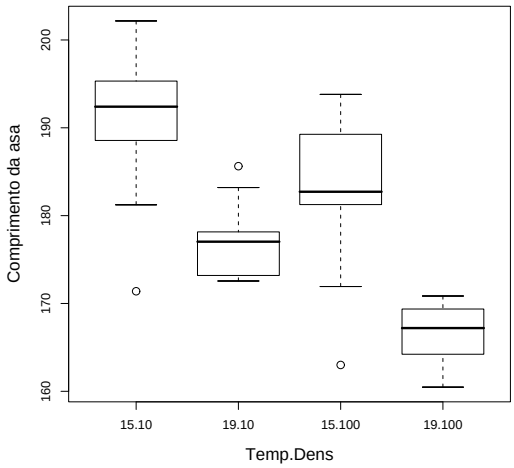

Figura 53: Diagrama boxplot mostrando o tamanho das asas de acordo com os fatores Temperatura e Densidade.

Os modelos lineares a serem ajustados quando temos mais de uma variável independente podem determinar na fórmula a existência de interações ou de que maneira estas interações serão tratadas:

1 `Y~A` - Análise de variância simples (fator A)

2 `Y~A+B+A:B` - Análise de variância dupla com termo de interação (fatores A e B)

3 `Y~A*B` - Mesmo que 2

3 `Y~A/B` - Análise hierárquica (mas a função `lm()` calcula a estatística de modo errado)

O ajuste do modelo de classificação dupla pode ser obtido com os seguintes comandos (como só temos dois níveis em cada um, não precisamos transformar as variáveis independentes em fatores):

```
> lm.dros<-lm(drosop$CA ~ drosop$Temp*drosop$Dens)
> anova(lm.dros)
Analysis of Variance Table
Response:  CA
            Df   Sum Sq   Mean Sq  F value      Pr(>F)
 Temp        1  2076.34  2076.34  43.6437  1.084e-07 ***
 Dens        1   834.94   834.94  17.5500  0.0001729 ***
 Temp:Dens   1    17.49    17.49   0.3676     0.5481018
 Residuals  36  1712.69    47.57
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

A tabela de análise de variância mostra claramente que os dois fatores são significativos, mas o termo de interação não é. Os gráficos diagnósticos e o teste de homoscedasticidade (não mostrados aqui) mostram que as premissas principais são atendidas. Como a interação não é significativa, cada um dos fatores é significativo e apresenta um efeito independente do outro (se tivéssemos mais de dois níveis nos fatores, precisaríamos realizar uma análise de comparações múltiplas). Por outro lado, se as interações fossem significativas, não poderíamos interpretar os resultados dos efeitos de modo independente. Um gráfico importante para a interpretação dos efeitos e suas interações pode ser gerado com a função `interaction.plot()`, como na Figura 54:

```
> interaction.plot(Temp,Dens,CA,cex.lab=1.3,lwd=2)
```

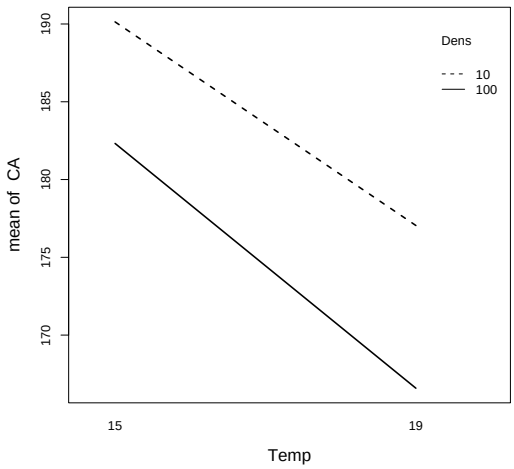

Figura 54: Gráfico de interação entre os fatores Temperatura e Densidade.

A Figura 54 mostra claramente que os efeitos dos fatores sobre a média da variável dependente não interagem. As moscas criadas a 15 graus apresentam médias maiores que as moscas criadas a 19 graus, independente da densidade em que foram mantidas. As moscas criadas com 10 indivíduos por garrafa apresentam médias maiores que as moscas criadas com 100 indivíduos por garrafa, independente da temperatura em que foram mantidas.

Vamos examinar agora um modelo misto de análise de variância, onde um dos fatores é fixo e o outro é aleatório. O conjunto de dados a ser utilizado corresponde a contagens de abundância de formigas realizadas em pares de parcelas dentro de uma floresta, onde uma foi aleatoriamente selecionada para ser queimada periodicamente (3 em 3 anos) durante 14 anos. O conjunto de dados e a descrição detalhada do problema pode ser encontrada em `http://aerg.canberra.edu.au/envirostats/bm/workbooks.htm`. A abundância utilizada foi calculada 2 anos depois da última queimada. O arquivo contendo o conjunto de dados (`ants.txt`) apresenta 3 colunas: Abant = abundância de formigas; Fogo = um fator com 2 níveis (BURNT e UNBURNT), indicando quais parcelas foram queimadas e quais não foram; Plot = um fator com 6 níveis, cada um correspondendo a um par de parcelas. Cada Plot foi amostrado 4 vezes com 1 m^2 de serapilheira em cada subamostra, totalizando 48 observações. O delineamento aleatorizado em blocos permite uma análise do efeito (fogo) em diferentes replicações (parcelas com diferentes características de solo e cobertura de dossel). O fator Fogo é uma variável independente de efeito fixo, ao passo que o fator Plot é uma variável de efeitos aleatórios, sendo que os blocos encontram-se arranjados hierarquicamente dentro da variável Fogo. Este é um caso típico de modelo misto de análise de variância e sabemos que o cálculo da estatística F para o teste do efeito fixo (Fogo) deve ser realizado dividindo seu quadrado médio pelo quadrado médio do fator aleatório (Plot). Se utilizarmos as funções `lm()` ou `aov()`, mesmo indicando uma relação hierárquica entre os fatores, o cálculo do F será realizado dividindo o quadrado médio de cada fator pelo quadrado médio do resíduo.

```
> ants<-read.table("ants.txt",header=T)
> ants$Plot<-as.factor(ants$Plot)
> lm.ants<-lm(Abant~Fogo/Plot,data=ants)
> anova(lm.ants)
Analysis of Variance Table
Response:  Abant
            Df   Sum Sq   Mean Sq   F value      Pr(>F)
 Fogo        1  1108384  1108384  247.3987  < 2.2e-16 ***
 Fogo:Plot  10   361964    36196    8.0793   1.163e-06 ***
 Residuals  36   161285     4480
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

Repare que, novamente tivemos que tansformar em fator uma variável categórica cujos valores eram numéricos (usando o comando `as.factor()`). Na tabela de análise de variância acima, o valor de F para o fator fixo (Fogo) não deveria ser calculado como a razão entre o quadrado médio do fator fixo e o quadrado médio do resíduo, mas sim dividindo o quadrado médio do fator fixo pelo quadrado médio do fator aleatório, o que reduziria consideravelmente o valor da estatística (1108384/36196 = 30,62173). Utilizando a informação fornecida na tabela é possível modificar os

valores de F e P, calculando a probabilidade associada ao `F` observado:

```
> pf(30.62173,df1=1,df2=10,lower.tail=F)
[1] 0.0002497253
```

Uma outra maneira de realizar esta análise é a partir da função `lme()`, do pacote `nlme`. Este pacote é mais complicado de utilizar, mas é uma alternativa mais elegante ao problema. Os argumentos principais a serem definidos são: `lme(fixed,data,random)`, onde `fixed` corresponde a uma equação com dois lados contendo a parte fixa do modelo, `data` corresponde ao data.frame a ser utilizado e `random` corresponde à parte aleatória do modelo (nesta parte definimos de que maneira os fatores encontram-se agrupados e aninhados). O primeiro passo é usar uma função para criar uma variável com os efeitos aleatórios que serão utilizados no modelo:

```
> ants$bl<-getGroups(ants,~1|Fogo/Plot,level=2)
```

Podemos então construir um modelo misto (veja os detalhes no arquivo de ajuda)

```
> lme.ants<-lme(Abant~Fogo,data=ants,random=~1|bl)
> lme.ants
Linear mixed-effects model fit by REML
Data:  ants
Log-restricted-likelihood:  -272.2662
Fixed:  Abant ~ Fogo
(Intercept) FogoUNBURNT
417.6667 -303.9167
Random effects:
Formula:  ~1 | bl
          (Intercept)  Residual
 StdDev:    89.04533   66.93394
Number of Observations:  48
Number of Groups:  12
> anova(lme.ants)
              numDF  denDF   F-value   p-value
 (Intercept)     1      36   93.62370   <.0001
 Fogo            1      10   30.62136    2e-04
```

Nesta tabela de análise de variância vemos que o F do fator fixo foi calculado corretamente. O efeito do fator fixo é mostrado como a média do primeiro nível no intercept (BURNT = 417.6667) e a diferença do segundo nível em relação ao primeiro (FogoUNBURNT = 417.6667-303.9167 = 113.75). O desvio padrão do efeito aleatório (Plot) é mostrado também, como o desvio referente ao intercept (89.04533). O componente de variância relativo ao fator Plot (s^2_{Plot}) pode ser calculado como o quadrado deste desvio:

```
> 89.04533^2
[1] 7929.071
```

O desvio padrão do resíduo pode ser transformado no quadrado médio do resíduo:

```
> 66.93394^2
[1] 4480.152
```

E o coeficiente de correlação intraclasse para o fator Plot pode ser calculado como:

```
> 7929.071/(7929.071+4480.152)
[1] 0.638966
```

Ou seja, existe uma considerável quantidade de variação entre parcelas, mas ainda assim o efeito fixo das queimadas é maior sobre a abundância das formigas. Os gráficos são bastante explicativos neste exemplo. O boxplot para os efeitos é encontrado na Figura 55 e o gráfico de interação na Figura 56:

```
> boxplot(Abant~Fogo/Plot,ylab="Abundância de formigas",cex.lab=1.3)
> interaction.plot(Plot,Fogo,Abant,cex.lab=1.3)
```

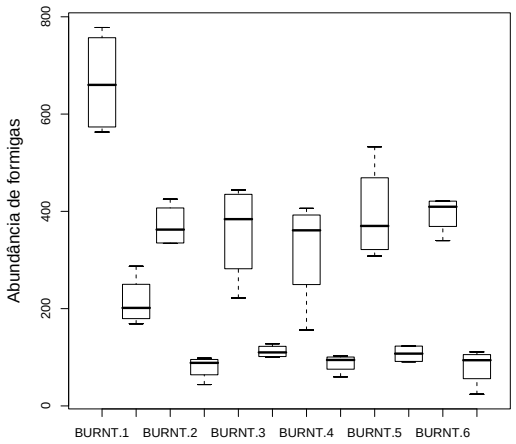

Figura 55: Boxplot para os efeitos Fogo e Plot.

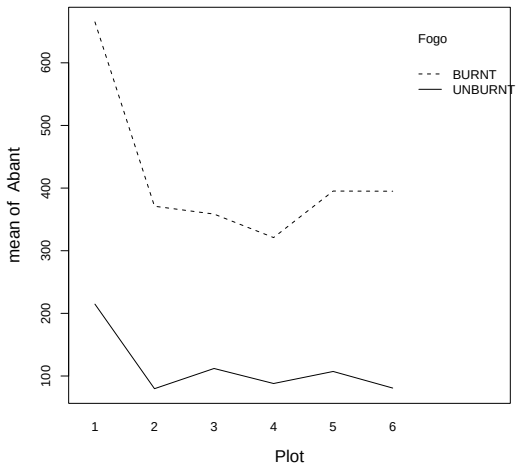

Figura 56: Gráfico de interação para os efeitos Fogo e Plot.

Mesmo que os modelos hierárquicos como este não apresentem possibilidade de interação, os gráficos boxplot e de interação mostram claramente que, independente do bloco de parcela, os locais queimados apresentaram consistentemente valores mais altos de abundância.

7.8.5 Poder e tamanho amostral em análises de variância

O cálculo do poder para análises de variância balanceadas (os delineamentos balanceados são mais poderosos, de modo que ninguém irá planejar um delineamento desbalanceado propositalmente) pode ser realizado diretamente pela função `power.anova.test()`, ou a partir do conhecimento sobre a distribuição não central F. O parâmetro que determina a não centralidade é chamado λ e pode ser calculado como:

$$\lambda = n\frac{\sum(\mu_j - \mu)^2}{\sigma^2}$$

Onde n corresponde ao tamanho das amostras (por grupo, μ_j corresponde à média de cada grupo, μ corresponde à média geral e σ^2 corresponde à variância dentro dos grupos. Este parâmetro também é chamado tamanho do efeito (*effect size*). O parâmeto λ pode ser utilizado como o argumento `ncp` (*non centrality parameter*) na função `pf()` que permite o cálculo de probabilidades para uma distribuição F não central. A distribuição F não central corresponde à distribuição esperada para a razão de variâncias se a hipótese alternativa for verdadeira (definindo a distribuição de H_1). O poder da análise de variância irá corresponder à proporção da curva da distribuição F não central igual ou menor que o F crítico. Vamos examinar um exemplo: em uma análise qualquer, temos quatro grupos diferentes com as seguintes médias: 10,10.5,10.5 e 11. Estas médias foram calculadas a partir de uma amostra inicial de 10 indivíduos por grupo e a variância dentro dos grupos foi calculada como sendo igual a 1. O valor do parâmetro de não centralidade seria então:

```
> x<-c(10,10.5,10.5,11)
> xm<-x-mean(x)
> (10*sum(xm^2))/1
[1] 5
```

O valor crítico de F para estas amostras seria

```
> qf(0.05,3,36,lower.tail=F)
[1] 2.866266
```

E a proporção da distribuição não central acima deste valor crítico seria:

```
> pf(2.866266,3,36,ncp=5,lower.tail=F)
[1] 0.3991676
```

Este valor corresponde ao poder da análise. Se utilizarmos a função `power.anova.test()` podemos confirmar este resultado e ainda buscar qual seria o tamanho de amostras adequado a este sistema. Os argumentos da função de poder são: o número de grupos, o tamanho das amostras por grupo, a variância entre grupos, a variância dentro dos grupos, o nível de significância (*default* = 0,05) e o poder. O parâmetro que não for preenchido entre os argumentos será calculado pela função, que produz uma lista dos parâmetros:

```
> power.anova.test(groups=4,n=10,between.var=var(c(10,10.5,10.5,11)),within.var=1)
Balanced one-way analysis of variance power calculation
groups = 4
n = 10
between.var = 0.1666667
within.var = 1
```

```
sig.level = 0.05
power = 0.3991677
NOTE: n is number in each group
```

Repare que o poder calculado é exatamente o mesmo que calculamos "manualmente"usando a distribuição F não central. Podemos ainda calcular a curva de poder relativo ao tamanho amostral para este exemplo e verificar qual seria o tamanho amostral necessário para ter um poder adequado:

```
> curve(power.anova.test(groups=4,n=x,between.var=var(c(10,10.5,10.5,11)),
within.var=1)$power,from=10,to=100,ylab="Poder",
xlab="Tamanho amostral",cex.lab=1.3,lwd=2)
```

Que pode ser visualizada na Figura 57.

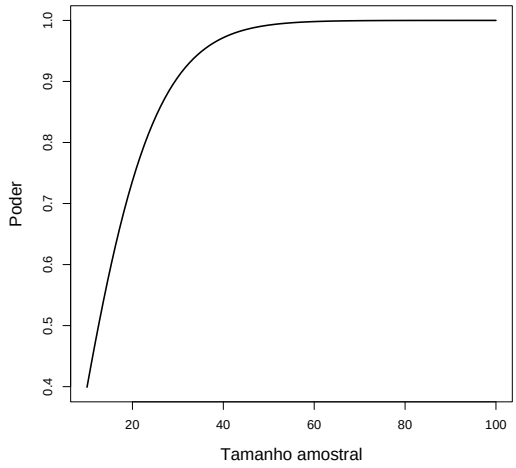

Figura 57: Poder e tamanho amostral para o exemplo.

Esta curva de poder sugere que a partir de 23 indivíduos por amostras passaríamos a ter um poder maior que 0,8. Os outros parâmetros cuja relação com o poder pode ser interessante verificar seriam provavelmente as diferenças entre médias e a variância dentro dos grupos.

7.9 Análise de covariância

Os modelos de análise de covariância (ANCOVA) são analisados a partir da função `lm()` ou `aov()`, assim como os outros modelos lineares que vimos anteriormente. Este tipo de modelo linear será utilizado com o propósito de comparar equações de regressão entre diferentes grupos (determinados por uma variável categórica ou fator), ou com o propósito de realizar um controle estatístico sobre a influência de uma variável contínua (covariável) sobre a variável dependente quando analisando as diferenças entre grupos. Um cuidado especial deve ser tomado na função a ser utilizada como argumento, pois o resultado na tabela de análise de variância pode diferir de acordo com a ordem dos termos (fatores e covariáveis). O arquivo a ser utilizado inicialmente como exemplo contém o conjunto de dados disponível na apostila teórica, com a fecundidade e o tamanho corporal de fêmeas de peixe, separadas em dois grupos diferentes (A e B). O primeiro passo será carregar o arquivo com os dados e visualizar os conjuntos de dados (Figura 58):

```
> pfec<-read.table("poecfec.txt",header=T)
> attach(pfec)
> plot(SL,Fec, cex.lab=1.3)
> points(SL[1:15],Fec[1:15],pch=16)
> legend(locator(1),c("A","B"),pch=c(16,1))
```

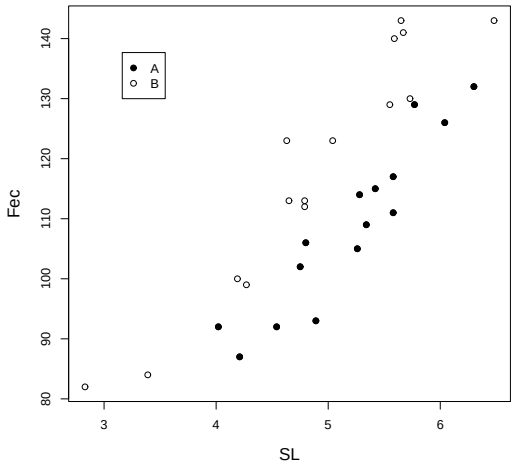

Figura 58: Diagrama de dispersão mostrando a relação entre fecundidade e comprimento padrão para os dois grupos.

Repare que introduzimos aqui um comando para fazer a legenda que aparece no gráfico (`legend()`). Este comando recebe como argumentos a posição desejada da legenda (em coordenadas x e y, ou usando a função `locator()`), o vetor com os nomes dos grupos na ordem que devem aparecer e o vetor com as características dos caracteres utilizados para o reconhecimento dos grupos (no nosso caso, o tipo de caractere (`pch`). A função `locator` permite a utilização do mouse para posicionar a legenda no gráfico sem que seja necessário especificar previamente as coordenadas.

O ajuste do modelo linear deve ser realizado com a função `lm()`, especificando a seguinte equação do modelo:

```
> lm.fec1<-lm(Fec~SL*Pop)
> anova(lm.fec1)
Analysis of Variance Table
Response:  Fec
            Df   Sum Sq   Mean Sq   F value      Pr(>F)
 SL          1   6572.9    6572.9   197.8604  1.148e-13 ***
 Pop         1   1800.9    1800.9    54.2101  8.088e-08 ***
 SL:Pop      1  0.01005   0.01005    0.0003       0.9863
 Residuals  26    863.7      33.2
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

Repare que é importante colocar a covariável (SL) na frente do fator (Pop) para que as somas de quadrados corretas sejam calculadas. Na tabela de análise de variância, o termo SL corresponde ao teste da regressão entre Fecundidade e tamanho do corpo, o termo Pop corresponde ao teste de comparação dos interceptos da regressão (equivalente ao das médias ajustadas) e o termo de interação corresponde ao teste de igualdade entre coeficientes de regressão nos dois grupos. Como o termo de interação não foi estatisticamente significativo, podemos considerar que as retas de regressão são paralelas e podemos seguir interpretando o teste de interceptos, o qual foi significativo, indicando que as retas são paralelas, mas com diferentes interceptos.

Se trocarmos a ordem das variáveis independentes no modelo, colocando o fator na frente da covariável, as somas de quadrados são calculadas de outra maneira, por exemplo:

```
> lm.fec2<-lm(Fec~Pop*SL)
> anova(lm.fec2)
Analysis of Variance Table
Response:  Fec
            Df   Sum Sq       Mean Sq       F value      Pr(>F)
 Pop         1    700.8         700.8       21.0968  9.839e-05 ***
 SL          1   7672.9        7672.9      230.9738  1.896e-14 ***
 Pop:SL      1  0.01005  0.01005 7 0.0003    0.9863
 Residuals  26    863.7         33.2
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

Neste caso, a soma de quadrados relativa ao fator Pop é calculada sem ajuste relativo à covariável, de modo que a estatística F é menor que a observada no modelo anterior. De fato, a soma de quadrados relativa ao fator Pop no modelo `lm.fec2` é calculada sem a correção dos valores de acordo com a covariável (a mesma que seria obtida a partir de uma análise de variância de classificação simples). A soma de quadrados relativa à covariável também é diferente no segundo modelo, pois esta se refere agora à regressão combinada dentro dos grupos (utilizando o coeficiente de regressão comum).

Se o resultado da interação fosse estatisticamente significativo, precisaríamos calcular uma reta de regressão separada para cada grupo. Isto poderia ser realizado com o seguinte modelo:

```
> lm.fec3<-lm(Fec~Pop/SL)
```

```
> summary(lm.fec3)
Call:
lm(formula = Fec ~ Pop/SL)
Residuals:
     Min       1Q   Median       3Q      Max
 -9.7026  -4.2484  -0.2679   4.6916   9.7952
Coefficients:
             Estimate  Std.  Error  t value    Pr(>|t|)
 (Intercept)    3.952      12.424      0.318       0.753
 PopB          15.522      14.767      1.051       0.303
 PopA:SL       20.194       2.379      8.489    5.71e-09 ***
 PopB:SL       20.244       1.606     12.606    1.40e-12 ***
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
Residual standard error:  5.764 on 26 degrees of freedom
Multiple R-Squared:  0.9065, Adjusted R-squared:  0.8957
F-statistic:  84.02 on 3 and 26 DF, p-value:  1.669e-13
```

Neste modelo, uma regressão separada para cada grupo foi calculada e os coeficientes de regressão para cada grupo estão listados nas linhas `PopA:SL` e `PopB:SL` da tabela acima. Como aceitamos a hipótese de que as retas apresentem a mesma inclinação, a diferença entre os coeficientes não deve ser considerada.

Podemos utilizar a função `anova()` também para a comparação dos termos de resíduo de diferentes modelos, com a adição ou remoção de termos. Este procedimento nos permite inferir se a adição de um determinado termo em um modelo diminui significativamente a soma de quadrados do resíduo. Vamos ajustar uma série de modelos e compará-los:

```
> lm.fec1<-lm(Fec~Pop)
> lm.fec2<-lm(Fec~SL+Pop)
> lm.fec3<-lm(Fec~SL*Pop)
> anova(lm.fec1,lm.fec2,lm.fec1)
Analysis of Variance Table
Model 1:  Fec ~ Pop
Model 2:  Fec ~ SL + Pop
Model 3:  Fec ~ SL * Pop
    Res.Df     RSS  Df  Sum of Sq          F          Pr(>F)
 1      28  8536.7
 2      27   863.7   1     7672.9   230.9738   1.896e-14 ***
 3      26   863.7   1    0.01005     0.0003          0.9863
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

O primeiro modelo ajustado (lm.fec1) é uma análise de variância de classificação simples. O segundo modelo ajustado é uma análise de covariância assumindo que os grupos apresentam retas de regressão paralelas (sem termo de interação). O terceiro modelo inclui um termo de interação entre o fator e a covariável (como vimos, a expressão `SL*Pop` se desdobra em `SL+Pop+SL:Pop`). A tabela de análise de variância montada compara as somas de quadrados residuais entre modelos.

A diferença entre o primeiro e o segundo modelo é muito grande. A adição da covariável no modelo linear diminui a soma de quadrados residual de 8536 para 863. O termo de interação, no entanto, não reduz a soma de quadrados residual de maneira significativa, mostrando que as retas são paralelas. Esta função é particularmente importante quando pretendemos comparar modelos com um maior número de fatores e covariáveis e buscamos o modelo mais simples possível cortando variáveis e termos de interação que não contribuam significativamente para a explicação da variância.

A conclusão e interpretação final é que os dois grupos apresentam relações significativas entre fecundidade e tamanho, com o mesmo coeficiente de regressão, mas com diferentes interceptos, como vemos na Figura 59

```
> lm1<-lm(Fec~SL,subset=1:15)
> lm2<-lm(Fec~SL,subset=16:30)
> plot(SL,Fec, cex.lab=1.3)
> points(SL[1:15],Fec[1:15],pch=16)
> abline(lm1,lwd=2)
> abline(lm2,lwd=2,lty=2)
> legend(locator(1),c("A","B"),pch=c(16,1),lty=c(1,2))
```

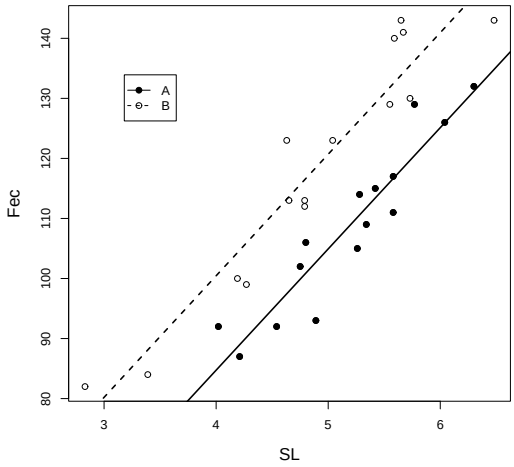

Figura 59: Diagrama de dispersão mostrando a relação entre fecundidade e comprimento padrão para os dois grupos, com as respectivas retas de regressão.

Nesta legenda inserimos mais uma característica, além do tipo de caractere, para mostrar diferenças entre os grupos: o tipo de linha (`lty`) utilizado para cada uma das regressões.

Uma vez que chegamos a um modelo satisfatório, devemos conferir os gráficos diagnósticos (Figura 60):

```
> par(mfrow=c(2,2),pch=16)
> plot(lm.fec2)
```

Nos gráficos diagnósticos podemos perceber que as premissas do modelo linear estão sendo razoavelmente atendidas pelos resíduos, de modo que a ajuste do modelo pode ser considerado

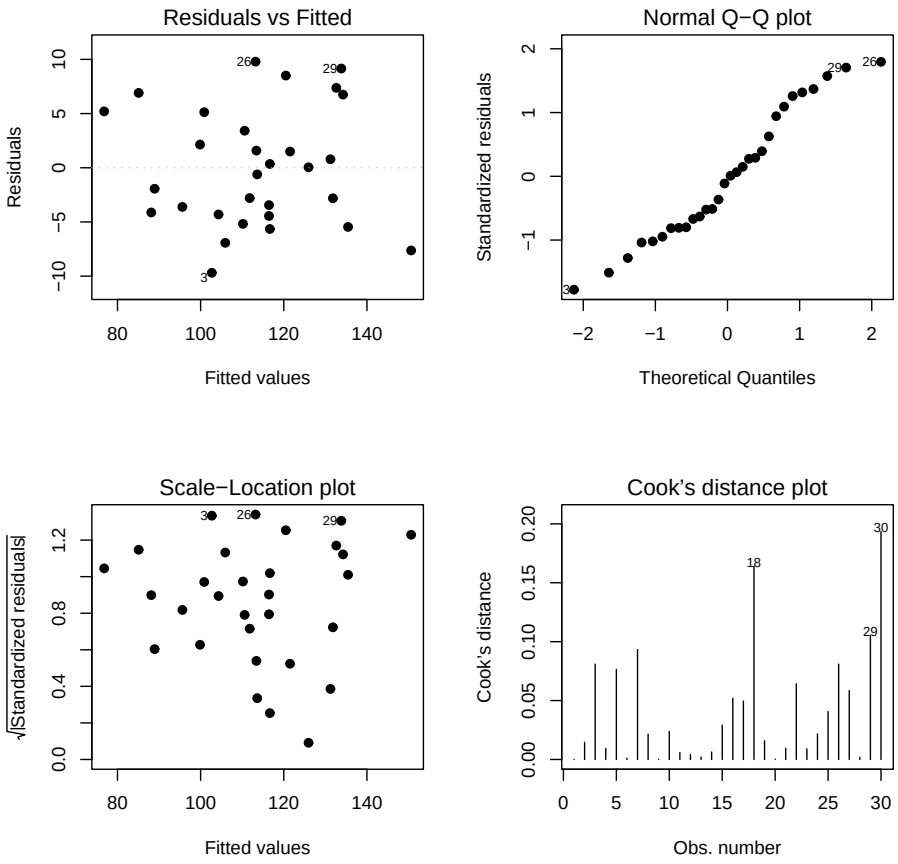

Figura 60: Gráficos diagnósticos para o modelo de análise de covariância.

satisfatório. Apesar das amostras serem relativamente pequenas, este conjunto de dados foi gerado por simulação (`rnorm`) a partir de parâmetros de duas populações reais. Isto significa que apesar dos pequenos desvios (principalmente com relação à normalidade), sabemos neste caso que todas as premissas são atendidas pela população.

Um outro exemplo, que ilustra a utilidade do controle estatístico da influência de uma covariável, vamos examinar a diferença na abundância de invertebrados no fital de algas de duas espécies diferentes, controlando os diferentes tamanhos médios das espécies. O arquivo algas2.txt (extraído de McPherson, G. 1990. *Statistics in scientific investigation: its basis, application and interpretation.* Springer-Verlag, New York) expande o conjunto de dados trabalhado anteriormente na seção sobre regressão, onde buscamos uma equação de regressão entre abundância de macroinvertebrados no fital e o peso seco em uma espécie de alga. Agora vamos comparar esta relação entre duas espécies diferentes:

```
> alg2<-read.table("algas2.txt",header=T)
```

Este arquivo possui 24 observações e três variáveis: o peso seco das algas (DW), a abundância de invertebrados (Ab) e a espécie de alga à qual os indivíduos pertencem (SP). A relação entre a variável dependente (Ab) e a covariável (DW) nos diferentes grupos pode ser vista na Figura 61:

```
> attach(alg2)
> plot(DW,Ab,cex.lab=1.4)
> points(DW[1:12],Ab[1:12],pch=16)
> legend(locator(1),c("Ccep","Sver"),pch=c(16,1))
```

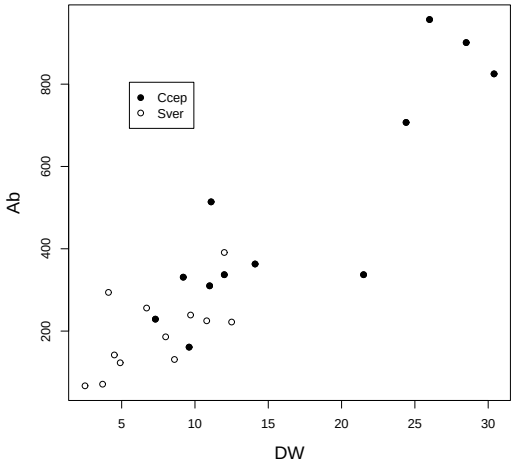

Figura 61: Diagrama de dispersão entre peso seco (DW) e abundância de invertebrados (Ab) para duas espécies de algas.

Parece claro que a abundância de invertebrados no grupo Ccep é maior que no grupo Sver, no entanto, esta maior abundância parece ser causada diretamente pelo maior tamanho dos indivíduos. Um modelo de análise de covariância pode controlar o efeito do tamanho dos indivíduos sobre a abundância:

```
> lm.alg<-lm(Ab~DW*SP)
> anova(lm.alg)
Analysis of Variance Table
Response:  Ab
            Df   Sum Sq   Mean Sq  F value      Pr(>F)
 DW          1  1218600  1218600  97.0661  4.061e-09 ***
 SP          1     4858     4858   0.3869       0.5409
 DW:SP       1    15596    15596   1.2423       0.2783
 Residuals  20   251086    12554
--
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1
```

A partir da tabela de análise de variância acima, vemos que as duas espécies apresentam retas de regressão estatisticamente similares entre abundância de invertebrados e peso seco. A interação não significativa mostra que os coeficientes de regressão são iguais e o efeito do fator SP não significativo indica que os interceptos também são os mesmos. Podemos então plotar a reta de regressão comum às duas espécies no gráfico e dizer que a única relação significativa no modelo é a regressão entre abundância e peso seco (Figura 62).

```
> lm.algr<-lm(Ab∼DW)
> plot(DW,Ab,cex.lab=1.4)
> points(DW[1:12],Ab[1:12],pch=16)
> abline(lm.algr,lwd=2)
> legend(locator(1),c("Ccep","Sver"),pch=c(16,1))
```

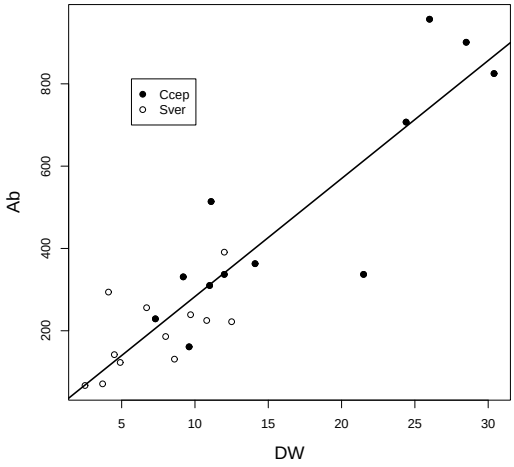

Figura 62: Diagrama de dispersão e reta de regressão entre peso seco (DW) e abundância de invertebrados (Ab) para duas espécies de algas.

8 Considerações finais

Acreditamos que o ambiente R seja o ponto central de uma revolução na maneira pela qual os usuários de estatística encaram a visualização e tratamento dos dados. A principal transição a ser realizada pelos alunos é habituar-se com a utilização de comandos ao invés de programas mais amigáveis baseados em menus. Esta é uma solução de compromisso entre facilidade e flexibilidade que é velha conhecida dos usuários que em algum momento procuraram realizar a transição entre os sistemas operacionais da Microsoft e o Linux. Apesar de interfaces amigáveis, como o R-commander, estarem disponíveis para a realização de análises mais comuns, é desejável que o usuário busque algum conhecimento sobre programação e um aprofundamento na estatística para poder aproveitar melhor a gama de recursos oferecida pelo ambiente.

As funções e gráficos apresentados nesta apostila representam uma parte muito pequena de todos os recursos disponíveis. Mesmo as funções apresentadas ao longo do texto podem apresentar argumentos e modos de personalização das análises não abordados aqui. A grande quantidade de textos em diversas línguas com introduções gerais ou explicações mais detalhadas e específicas facilita muito a iniciação do usuário neste ambiente. É fundamental que o usuário do R acostume-se com os arquivos de ajuda do programa, que constituem a fonte principal a ser utilizada. A teoria por trás dos métodos deve ser buscada em livros texto de estatística (alguns são repetidamente citados nos arquivos de ajuda). As respostas para a maior parte dos problemas comuns (e muitos bastante complexos) pode ser encontrada em buscas nos arquivos do diretório de e-mail R-help (http://www.r-project.org/mail.html). Mesmo se com toda a ajuda disponível, a resposta a um problema não for encontrada, é possível encaminhar uma pergunta ao diretório de e-mail R-help e contar com o auxílio dos próprios programadores ou autores dos pacotes. Este, no entanto, deve ser considerado um último recurso se nenhum dos outros meios funcionou, de modo a evitar a repetição de perguntas e respostas, congestionando ainda mais esta lista eletrônica.