Introdução ao Scilab Para Análise Estatística

Carlos A. Cinquetti e Ricardo G. Silva

11 de junho de 2006

Sumário

Capítulo 1

Recursos Básicos

1.1 Operações Elementares

Todas as operações são feitas a partir do prompt do Scilab, como ilustrado abaixo:

```
-->2+3
 ans  = 5.
-->2*3
 ans  = 6.
-->10/2
 ans  = 5.
-->10-2
 ans  = 8.
```

Caso fôssemos criar variáveis ou constantes a partir das operações básicas, faríamos:

```
-->a=5+2
 a  = 7.
-->b=2*3
 b  = 6.
-->c=8/4
 c  = 2.
```

As variáveis ficariam armazenadas no Scilab, permitindo-nos operar com elas, como no exemplo abaixo:

```
-->A=a+b*2
 A  = 19.
-->A=(a+b)*2
 A  = 26.
```

1.2 Variáveis Especiais

Números irreais ou complexos, assim alguns números especiais, são criados no Scilab a partir de algumas funções especiais, como nos casos mostrados na Tabela 1.1 abaixo:

Tabela 1.1: Variáveis Especiais

Número π	$\%pi$
Complexo $\sqrt{-1}$	$\%i$
Constante Trigonométrica e	$\%e$
Infinito	$\%inf$
NotANumber (Missing)	$\%nan$
Constante Boleana	$\%t$ e $\%f$

Ilustremos o uso destas variáveis com algumas operações simples.

```
-->a=5+2*%i
a  = 5. + 2.i
-->b= 4-3*%i
 b  = 4. - 3.i
-->a*b
 ans  = 26. - 7.i
```

1.3 Comentários e Pontuação

Comentários no arquivo Scilab, feitos à título de organizar o raciocínio, ou qualquer fim, devem ser precedidos do símbolo de porcentagem, %. Por exemplo:

```
% observe que, no Scilab, como em muitas calculadoras, as
operações $a+b*c$ difere de $(a+b)*c$
```

A linha em questão não será, portanto, tomada como parte das operações matemáticas. É um recurso especialmente útil nos arquivos de programação.

Outro recurso é a pontuação final, indicando se desejas ou não exibir o produto na tela. Ilustrando:

```
-->Y=[1 2; 3 4]
 Y  =
!   1.    2. !
!   3.    4. !
```

Se não queres visualizá-la, ponha ponto e vírgula no final:

```
 -->Y=[1 2; 3 4];
```

É desejável quando estamos operando com transformações longas, notadamente com matrizes ou séries, permitindo nos concentrar nas simples operações lógicas, deixando de lado a computação numérica, que podemos deixar para o final.

Capítulo 2

Polinomios, Vetores, Matrizes e Listas

2.1 Polinômios

Seguem alguns exemplos de definição de Polinomios no Scilab

```
-->x=poly(0, 'x')
x  =
    x

-->p=1 -3*x +x^2
 p  =
                2
    1 - 3x + x
```

2.2 Vetores

Um vetor y qualquer é definido pelas variáveis que o compõe. No Scilab, faríamos:

```
y=[1 2 3 4]';
```

Note que o ' após o colchete define a transposta de um vetor definido em [..]. Como a forma normal é o vetor coluna, y do exemplo seria um vetor linha. Vamos criar dois outros vetores e fazer algumas operações com eles.

```
ct=[1  1  1  1]';
```

O Scilab tem funções específicas para gerar alguns vetores ou matrizes especiais. No caso dos vetores, podemos gerá-los pelas funções para sequências do Scilab. Tomemos alguns exemplos de vetores de constantes:

```
z=ones(1:4)
z2=3*ones(1:3)
```

```
    z2  =
    !   3.    3.    3. !

    z3=zeros(1:5)
    z3  =
!   0.    0.    0.    0.    0. !
```

Note que, dentro do parênteses, definimos a quantidade de termos do vetor, ou da seqüencia, e fora do parênteses os termos constantes que o compõe.

Se quisermos operar com vetores dados por uma seqüência que progridem numa seqüencia definida, dentro de um intervalo estabelecio, faríamos:

```
-->v=[5:1:10]
 v  =
!   5.    6.    7.    8.    9.    10. !

-->v1=[5:-.5:1]
 v1  =
!   5.    4.5    4.    3.5    3.    2.5    2.    1.5    1. !
```

Geramos, desta forma, tendências temporais, ou outra série temporal como seqüencia definida, as quais são muito úteis na análise estatística e econométrica.

2.3 Matrizes

Criam-se matrizes de forma semelhante, usando o operador ; para inserir linhas novas. Convém, também, seguir a convenção matemática, usando maiúscula para matrizes:

```
A=[1 2; 3 4]; B=[5 6;7 8]; C=[9 10;11 12];
```

Finalmente, podemos criar uma matrix composta pelas demais matrizes ou vetores, definidos acima. Por exemplo:

```
-->D=[A B C]
 D  =
!   1.    2.    5.    6.    9.     10. !

!   3.    4.    7.   8.    11.     12. !
```

Podemos transformá-la numa outra matriz, E, 3x4, fazendo:

```
E=matrix(D,3,4)
```

Podemos, finalmente, transformar E numa matriz identidade:

```
F=eye(E)
```

Uma matriz identidade $3x3$ poderia ser assim gerada:

```
-->Z=eye(3,3) z  =

!   1.    0.    0. !

!   0.    1.    0. !

!   0.    0.    1. !
```

Finalmente, dada uma matriz A

```
-->A=[1 2; 3 4]
 A  =
!   1.    2. !

!   3.    4. !
```

As funções abaixo, nos dão:

```
-->size(A); \\as dimensões da matriz

-->det(A); \\seu determinante

-->diag(A);\\sua diagonal
```

2.4 Acessando Vetores e Matrizes

Como visto, o Scilab toma uma matriz pelo nome associado a ela, ficando sua dimensão numérica e espacial registrada na memória do programa. Quando desejamos nos certificar do conteúdo de tal matriz, trazendo-a à tela, usamos a função evstr(.):

```
-->evstr(E)
 ans  =
!   1.    4.    6.    11. !

!   3.    5.    8.    10. !

!   2.    7.    9.    12. !
```

Suponha que tivéssemos uma matriz muito grande, cuja exibição na tela ocuparia enorme espaço, quando nos interessa avaliar apenas algumas de suas colunas, ou linhas. Neste caso, usamos os seguintes operadores:

- $M(1,:)$ Toda linha 1 da matriz M;
- $M(2,:)$ Toda linha 2;
- $M(:,1)$ toda coluna 1 e
- $M(:,2)$ Toda coluna 2.

Os dois pontos tudo e a vírgula denota linha, quando antes dela, e coluna, quando posterior a ela.

Se queres apenas os dados da primeira linha, segunda coluna seria (1,2). Abaixo exemplificamos como procedemos no caso da matrix tbl61.

Para o caso de vetores, tomemos por ilustração o vetor y definido acima. Os operadores

```
-->y(:)    //acessa todos seus elementos
 ans  =

!   1. ! !   2. ! !   3. ! !   4. !


-->y(2:4)  //apenas do segundo ao quarto elemento
 ans  =

!   2.    3.    4. !
```

2.5 Operações com Vetores e Matrizes

Suponha que tenhamos um vetor x e uma matriz A e queremos obter um novo vetor e matriz correspondente ao quadrado de ambos. Fazemos:

```
x2=(x.^2);     A2=(A.^2);
```

Como $x2$ é definido a partir de uma transformação de x, a função que opera tal transformação é definida entre parênteses. Outra observação importante: usamos o ponto antes do operador exponencial, ^, pois estamos transformando uma variável, x, e não um número.

2.6 Listas

Capítulo 3

Arquivos

3.1 Diretório dos Dados

Ao operar com uma base de dados, ou manipulações numéricas, um primeiro passo é saber em que diretório o Scilab está operando, o que é feito com a função pwd:

```
-->pwd
    ans  = C:\WINDOWS\Desktop
```

Desejando mudar de diretório, use a função 'chdir'. Assim, supondo que seu arquivo esteja em `C:\cursos\IEconometria\Apostila`, faríamos:

```
>chdir(C:\cursos\IEconometria\Apostila)
     ans  =   0.
```

3.2 Importando Arquivos de Outros Programas

Na análise estatística, um trabalho comum seria a importação de arquivos de dados, sempre compostos numa matrix, onde cada coluna representa uma variável e as linhas as respectivas observações. São os chamados arquivos no tipo ascii. O Excel é, talvez, o mais comum, embora . armazenados num arquivo Excel, com extensão ".xls". A forma ".csv"e ".prn"são as mais utilizadas em bancos de dados de domínio público, mantidos por instituições públicas ou privadas. Para abrir tais arquivos no Scilab dê o seguinte comando:

```
M = excel2sci("C:\arquivos\alguma-coisa\filename.csv", ",")
```

. Outra opção é:

```
M = excel2sci("C:\arquivos\alguma-coisa\filename.csv", ":,:")
```

A despeito da função excel2sci remeter ao Excel, veja que ela se aplica a outros arquivos ascii, como o csv do exemplo. Na realidade, para o caso de arquivos Excel, convém transformá-los antes para "bloco de notas", o que pode ser feito ou abrindo-o e salvando no

Bloco de Notas (entre os programas acessórios do Windows), ou pedindo, no Excel, para salvá-lo como "txt" (com separador de vírgulas). Pelo mesmo caminho, é possível transformar um arquivo "txt" para "csv", pedindo que o Excel salve-o como 'csv' (com separador de vírgulas).

O Scilab não reconhecerá corretamente as variáveis se, no arquivo original, elas estavam definidas por um nome inserido na primeira linha de cada coluna. É preciso retirar toda a linha contendo o nome das variáveis. Abaixo, mostramos um caminho rápido para nomear as variáveis do arquivo importado.

Se o nome das variáveis já foram retirados, outro caminho é simplesmente colar. Suponha que os dados estejam no Excel, onde cada coluna está associada a uma variável, com a primeira linha indicando o nome desta. Corte tal linha, faça o *copy* e insira, via paste, no Scilab, para posterior associação dos valores. Isto é, terás que definir tais dados como uma matriz, com um nome e escrito na forma correspondente a uma matriz, como definido no capítulo anterior.

3.3 Nomeando as Variáveis

Como dissemos acima, será preciso associar variáveis e nomes, do arquivo importado ou copiado, pelo caminho específico do Scilab. Seguindo a notação matricial do programa, isto pode ser feita na forma:

```
-->nome da variável=nome da matrix dos dados(:,no da coluna).
```

que dá um determinado nome para toda coluna (no coluna) da matrix (nome). Supondo que estivéssemos trabalhando com o nosso arquivo tbl6, isto poderia ser feito da seguinte forma:

```
-->YEAR=tbl6(:,1); G=tbl6(:,2); PG=tbl6(:,3); Y=tbl6(:,4);
PNC=tbl6(:,5); PUC=tbl6(:,6); PPT=tbl6(:,7); PD=tbl6(:,8); PN
 tbl6(:,9); PS=tbl6(:,10); POP=tbl6(:,11);
```

Note que fizemos a transformação de todas as variáveis de uma só vez, usando o separador ; após a definição de cada nome.

3.4 Salvando e Avaliando Dados e Arquivos

Suponha que tenhamos criado a matrix:

```
-->x=[1 2 %pi;%e 3 4];
x  =

!   1.           2.    3.1415927 ! !   2.7182818    3.    4.!
```

Caso estivéssemos operando num arquivo de programação, no editor do Scilab, faríamos:

```
-->write('matx.dat',x) \\isto é(nome do arquivo.extensão, nome da
matrix
```

para armazená-lo com banco de dados específico ao programa, embora só seria necessário caso não quiséssemos operar futuramente com tal arquivo, de modo abreviado.

O caminho é outro quando estamos operando com bases de dados, das quais apenas algumas das variáveis podem nos interessar. Suponha que tívessemos criado a base de dados:

```
-->X=[1960 1961 1962 1963 1964; 129.7 131.3 137.1 141.6 148.6;
0.925 0.914 .919 .918 .949]'
 X  =
    1960.    129.7    0.925
    1961.    131.3    0.914
    1962.    137.1    0.919
    1963.    141.6    0.918
    1964.    148.6    0.949
```

Para salvá-la:

```
-->save('C:\DATA\dados.dat', X)
```

O arquivo data é do tipo binário, comum às formas de operações matemáticas no Scilab.

Desocupamos a memória do Scilab com tal arquivo fazendo:

```
-->clear X
```

que poderia ser aplicada, também, à matrix de nosso arquivo de programação, fazendo:

```
-->clear x
```

embora isto só tenha sentido caso estíssemos operando com "x" no ambiente próprio do Scilab; isto é, não aplicaríamos *clear* num arquivo de programação.

Para recarregar o arquivo, faríamos:

```
-->load('C:\DATA\dados.dat', 'no_linhas, no_colunas')
```

Isto é, no_linhas corresponderia ao número de linhas de nossa matrix do arquivo ascii recém criado.

Se fóssemos apenas ler o arquivo, sem carregá-lo, faríamos:

```
x=read('nome do arquivo.txt', no. linhas, no. colunas)
```

Todos os dados do arquivo tabl6, que estão na forma ascii, apareceriam na tela, que deixamos de exibir aqui.

O comando "load" não traz os dados à tela. Querendo vê-los, fazer:

```
-->evstr(X)
 ans  =
    1960.    129.7    0.925
    1961.    131.3    0.914
    1962.    137.1    0.919
    1963.    141.6    0.918
    1964.    148.6    0.949
```

Lembre-se, poderíamos associar as variáveis acima com caracteres, como:

```
Y=X(:,1) ; C=X(:,2); I=X(:,3)
```

Neste caso, para carregá-lo faríamos:

```
-->load('C:\DATA\dados.dat', 'y,C,I')
```

Numa análise estatísticas, podemos precisar de apenas um conjunto de variáveis de uma matriz de dados, e variáveis, contida em determinado arquivo. À título de exemplo, considere a matrix x criada acima, que nos interessaria apenas . eventualmente, quere tomar apenas algumas das variáveis

- xenw=read('x.dat',2,3)

```
 xenw  =
!   1.           2.    3.1415927 ! !   2.7182818    3.    4. !
```

Note que o comando read especifica o número de linhas e colunas.

Capítulo 4

Figuras

Em análise estatística usaríamos análise gráfica tanto para analisar a correlação entre dados, ou entre pontos previstos e pontos observados, quanto para simulações das propriedades matemáticas de alguns estimadores ou estatísticas. Nesta seção introduziremos a projeção de gráficos de funções, o que será útil tanto para os dois últimos tipos de análise gráfica em estatística. Nos capítulos Estatísticas Básicas e .. exporemos outros recursos gráficos, trantando dos respectivos problemas estatísticos.

Veremos, aqui, os comandos para a geração de gráficos mais comuns, em duas e três dimensões. As novas versões do Scilab, como a 3.0, já dispõe de uma interface gráfica para manipulação de gráficos, facilitando muito o trabalho. Como esta. porém não está totalmente implementada, utilizares o prompt de comando para executar os gráficos.

4.1 Gráfico de Funções em Duas Dimensões

Vamos iniciar o estudo de gráficos de funções no espaço dois. Para que o Scilab entenda que as variáveis x e y estão relacionadas por uma função, usamos deff (definir função) e especificamos:

```
-->deff("[y]=f(x)","y=sin(x)")
```

A função seno é uma das muitas já definidas no Scilab. Precisamos, agora, definir o domínio da função:

```
-->x=[0:0.1:10]*%pi/10;
```

Isto posto, solicitamos plotar a função f, como segue:

```
-->fplot2d(x,f)
```

cujo resultado está exposto na 4.1 abaixo.

Para plotarmos duas funções juntas, basta colocarmos dentro dos colchetes:

```
plot2d([3.5*x x^2-3])
```

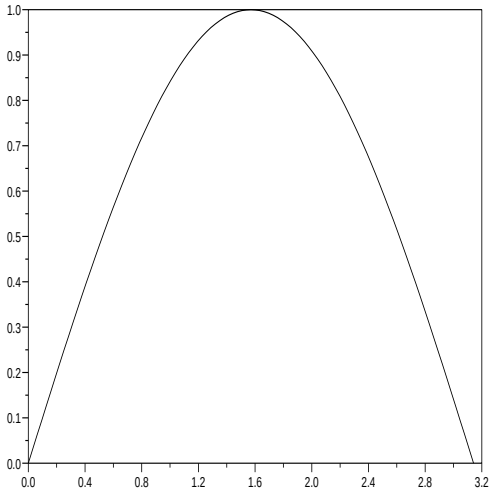

Figura 4.1: Plot em 2D da função Seno

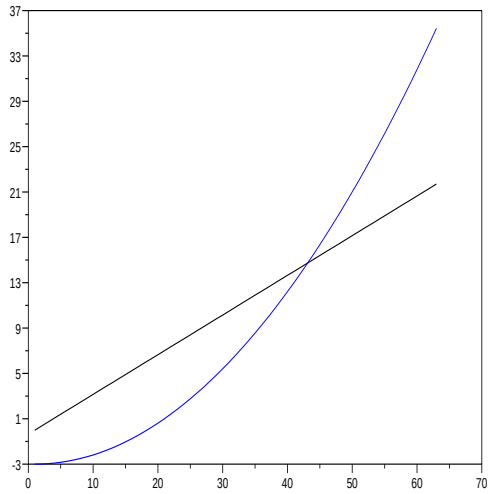

Figura 4.2: Plot em 2D da função Seno

O resultado: Vamos agora apenas utilizar valores de x, sem definir funções. Declarando

```
x=[0:0.1:2*%pi]';
```

Podemos plotar $sen(x)$, $sen(2x)$ e $sen(3x)$ juntos e dar nomes as curvas:

```
plot2d(x,[sin(x) sin(2*x) sin(3*x)],[1,2,3],leg="L1@L2@L3",nax=[2,10,2,10],rect=[0,-2
```

O resultado:

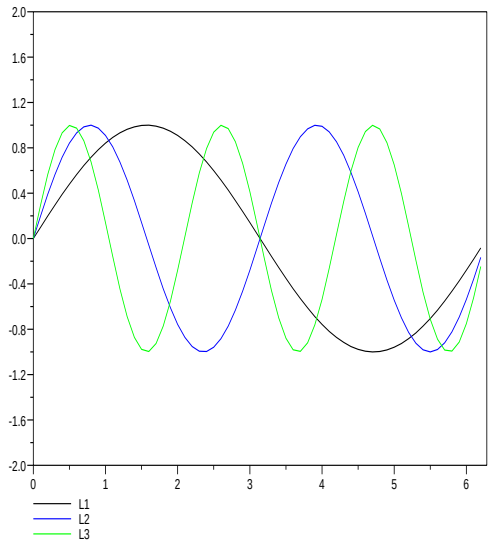

Figura 4.3: Plot em 2D da função Seno

r

r

r

4.2 Gráfico de Funções em 3 Dimensões

No capítulo 9, expomos os recursos gráficos do Scilab para análise de dados e para análise estatística mais geral.

Capítulo 5

Carregando Pacotes

Há uma série de pacotes que potencializam a execução de alguns funções no Scilab. Para carregar tais pacotes há uma caminho específico.

O primeiro é o download a partir da página do Scilab `http://scilabsoft.inria.fr/contribution/...` Ao abrir o arquivo zip, selecione todos os arquivos que o compõe e faça o extract para a pasta `c:\arquivosdeprograma\Scilab-30-RC1\macros`, supondo que seja este o nome da versão do Scila e o local em que está instalado no seu computador. Uma nova pasta será instalado no macros do Scilab com o nome do pacote instalado.

No menu do Scilab, selecione File, então a opção exec, e buscar o pacote recem instalado no caminho `c:\arquivosdeprograma\Scilab-30-RC1\macros\nomepacote`. Clique no arquivo 'builder' do pacote. O pacote será, então, instalado.

Embora instalado, para utilizar cada pacote será necessário carregá-lo toda vez que for utilizá-lo. Caminho: no menu do Scilab, selecione File, então exec, buscando o pacote. Ao clicar em cima dele, selecione seu "loader".

Capítulo 6

Estatísticas Básicas

Há uma série de funções no Scilab para o cálculo de estatísticas ou parâmetros de uma ou mais variáveis, como ilustrado na tabela abaixo. No help do Scilab podem ser encontrados outras funções, assim como um maior detalhamento de alguns procedimentos específicos para cálculo de algumas destas funções.

Para ilustrar, tomemos duas séries (vetores) gerados arbitrariamente:

```
-->a=[26:-3:2];

-->b=[2:3:26];
```

E também nossa matríx E,

```
E=matrix(D,4,3)
 E  =
!   1.    5.    9.  !

!   3.    7.    11. !
```

Tabela 6.1: Funções para Estatísticas Básicas

mean(e)	média de e
geomean(e)	média geométrica
variance(e)	variance de e
variance(e,2)	variância
$st_deviation(e)$	desvio padrão
covar(e,a, fre)	covariância, com freq para frequência
correl (e,a, fre)	correlação entre a e b
moment(e,2)	momentos, com ,.. indicando o momento específico
center(e)	e centrado; desvios da média
quart(e)	quartis
perctl(e)	percentis

```
!   2.    6.    10. !

!   4.    8.    12. !
```

Na ilustração abaixo, damos um nome para cada variável, necessário quando queremos operar matematicamente com tais resultados.

```
-->m=mean(b)
 m  =
    14.

-->varb=variance(b)
 varb  =

    67.5

-->varE=variance(b)
 varE  =
    67.5

-->stde=st_deviation(e)
 stde  =
    8.2158384
```

Capítulo 7

Variáveis Aleatórias

Gerar variáveis aleatória muitas vezes é necessário, principalmente quando desejá-se realizar simulações do tipo Monte Carlos. No Scilab, iremos utilizar uma caixa de ferramentas chamada Grocer. Esta pode ser obtida na Internet, no endereço `http://scilabsoft.inria.fr/`, em Contributions –> Download –> Data Analysis And Statistics. Este comando deve ser instalado seguindo instruções expostas na seção Carregando Pacotes. O comando básico é:

```
Y=grand(m, n, dist_type [,p1,...,pk])
```

onde m é a quantidade de elementos de cada geração aleatória e n a quantidade de gerações aleatórias, $dist_type$ é o tipo de distribuição que se deseja gerar e $[, p1, ..., pk]$ são os parâmetros da dist. requerida. À título de exemplo, geremos 50 pontos de uma distribuição Normal, com média zero e variância unitária:

```
-->Y=grand(100,1,'nor',0,1)
```

e fazer o seu histograma, utilizando 10 barras:

```
--> histplot(10,Y)
```

O resultado esta na figura 7.1. Vamos, agora, gerar 1000 pontos de apenas uma geração aleatória de uma normal com média 10 e variância 4.2 e colocar em outro histograma, agora com 50 bins:

```
-->Y=grand(100,1,'nor',10,4.2);

-->histplot(50,Y)
```

O resultado, na figura 7.2.

Finalmente, como exemplo, vamos introduzir o gráfico de uma distribuição, gamma, bastante usada em inferência bayesiana como priori para a variância de um modelo. O comando:

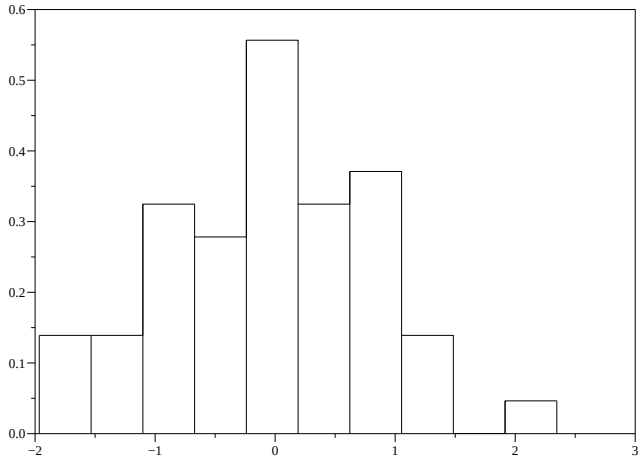

Figura 7.1: Distribuição Normal
h

```
Y=grand(500,1,'gam',5,50)
```

As distribuições disponíveis estão listadas abaixo. Quanto aos parâmetros, estes podem ser vistos utilizando o comando help do Scilab (help grand).

- beta:

  ```
  Y=grand(m,n,'bet',A,B);
  ```

- binomial:

  ```
  Y=grand(m,n,'bin',N,p);
  ```

- binomial negativa:

  ```
  Y=grand(m,n,'nbn',N,p);
  ```

- chi-quadrada:

  ```
  Y=grand(m,n,'chi', Df);
  ```

- chi-quadrada não-central:

  ```
  Y=grand(m,n,'nch',Df,Xnon);
  ```

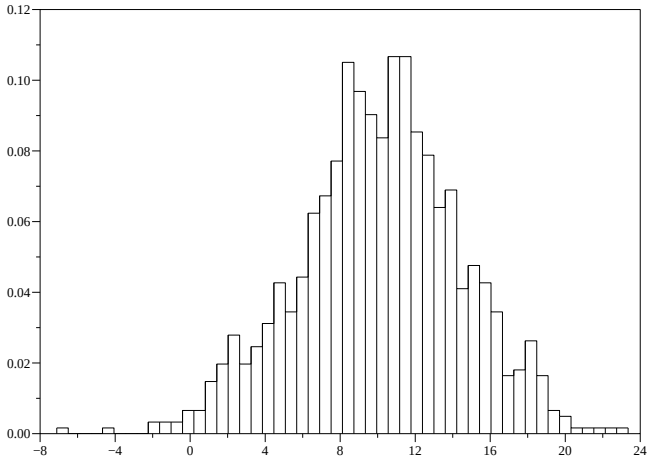

Figura 7.2: Distribuição Normal

- exponencial:

  ```
  Y=grand(m,n,'exp',Av);
  ```

- Distribuição F:

  ```
  Y=grand(m,n,'f',Dfn,Dfd);
  ```

- Distribuição F não central:

  ```
  Y=grand(m,n,'nf',Dfn,Dfd,Xnon);
  ```

- Gauss Laplace (normal):

  ```
  Y=grand(m,n,'nor',Av,Sd);
  ```

- gaussian multivariada:

  ```
  Y=grand(n,'mn',Mean,Cov);
  ```

- geométrica:

  ```
  Y=grand(m,n,'geom', p);
  ```

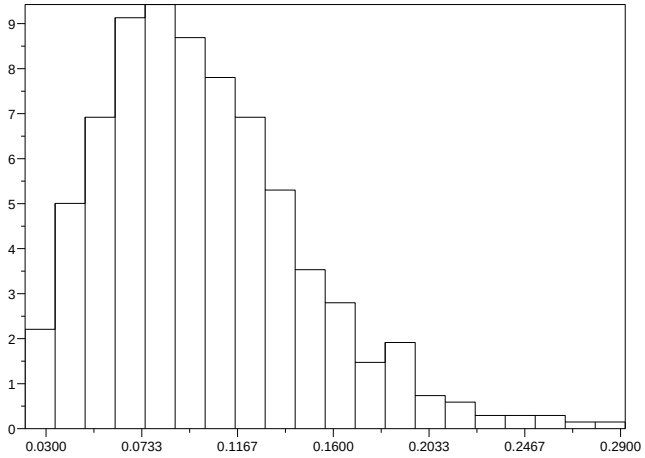

Figura 7.3: Distribuição Gamma(5,50)

- Markoviana:

  ```
  Y=grand(n,'markov',P,x0);
  ```

- multinomial:

  ```
  Y=grand(n,'mul',nb,P);
  ```

- Poisson :

  ```
  Y=grand(m,n,'poi',mu);
  ```

- Permutações aleatórias:

  ```
  Y=grand(n,'prm',vect);
  ```

- uniforme (0,1):

  ```
  Y=grand(m,n,'def');
  ```

- variações da uniforme.

Capítulo 8

Análise de Regressão

A grande vantagem do Scilab, sobre softwares específicos de estatística, como o R, são seus recursos para computação matemática. O processo é mais lento, mas fazer estimações definindo as operações algébricas com os dados permite uma aprendizagem mais segura, além de permitir o estudioso a alterar as funções sempre que problemas inesperados surjam. Finalmente, no caso de análise de Regressão, que trataremos nesta seção, uma vez feito o primeiro programa, ele poderá ser usado em todos os demais problemas, com pequenas alterações, se necessárias.

Ainda assim, não podemos desprezar as funções já construídas que permitam uma solução mais rápida para análise de Regressão. Com efeito, estruturamos esta seção começando com uma programação para análise de regressão no Scilab e, a seguir, apresentamos funções existentes para tratar dos problemas típicos de Regressão.

8.1 Mínimos Quadrados Ordinários: Programação

Estando dentro do Editor do Scilab, o primeiro passo é carregar os dados, quer importando arquivos, como descrito acima, ou simplesmente definindo as variáveis, como no exmplo abaixo.

```
y=[74 15 46 45 85 32 65 98 54 87 21 26 45 56]'; //Variável
dependente

x=[1 1 1 1 1 1 1 1 1 1 1 1 1 1; 12 23 32 45 45 96 45 32 65 96 36
76 34 56; 67 98 34 23 56 67 78 57 34 98 45 34 54 65; 54 58 24 51
68 26 35 20 1 41 36 85 58 46]';  //variaveis independentes
```

Precisamos definir as dimensões do vetor y, nossa variável independente, e da matrix x de nossas variáveis independentes.

```
n=length(y); // Numero de elementos
```

```
y [n k]=size(x);//Numero delinhas (observações) e de colunas
(variáveis)
```

Suponha que desejássemos transformar nossas variáveis para logarítmo, pelo conhecimento prévio que temos do problema.

```
y1=log(y);
x1=log(x(:,2:4)); //Tranformando os dados, menos a
constante. Note que podemos transformar uma grande quantidade de
variaveis de uma só vez

z=[x(:,1) x1]; //Nova matriz de Variaveis independentes
```

Algumas estatisticas descritivas

```
mm=mean(y1)
vv=st_deviation(y1)

Loop para calcular o vetor de médias da var independents
for j=1:4
m(j)=mean(z(:,j));
end
meds=m

Loop para os desvio-padrões
for j=1:4
st(j)=st_deviation(z(:,j));
end stdevs=st
```

Cálculo dos coeficientes betas

```
zzinv=inv(z'*z);//Calculo da Inversa

zy=(z'*y1);//Calculo de x'y

betas=zzinv*zy
```

Análise da Variância(ANOVA)

```
yhat=z*betas;//Valores ajustados y chapéu (previsto)

e=y1-yhat;//erros da regressão
```

```
sqt=y1'*y1;//Soma dos quadrados totais

sqr=e'*e;//Calculo da somado quadrado dos residuos SQR

sqe=yhat'*yhat;//soma dos quadrados explicados

msqt=sqt/(n-1);//sqt médios

msqr=sqr/(n-k);//sqr médios

msqe=sqe/k-1;//sqe médios
```

Teste dos Betas

```
I=eye(4,4)//matriz identidade com dimensão do número de
coeficientes

covarb=msqe*I*zzinv;//matriz variância-covariância dos betas

varb=diag(covarb); //diagonal:variância dos betas

stdb=sqrt(varb); //dp dos betas

tbetas=betas.\stdb; //número de dp dos betas; tem distribuição
t-student
```

Qualidade e Teste do Moodelo

```
//R2 quadrados
R2=sqe/sqt

F=(sqe/(k-1))/(sqr/(n-k)) //Estatistica F

R2j=1 - (msqr/msqt)// R2 Ajustado

//Critérios de Informação de Akaike
CIA = 1 + log(2*\%\pi) + + log(sqr/n) + 2*k/n
```

8.2 Mínimos Quadrados Ordinários

8.3 Mínimos Quadrados Ponderados

Capítulo 9

Análise Estatística Exploratória

9.1 Diagrama de Dispersão

9.2 Diagrama de Dispersão: Recursos Avançados