Curso de Ox

DME - UFRJ

Autor:
Joaquim Henriques V. Neto

Departamento de Métodos Estatísticos - IM
Universidade Federal do Rio de Janeiro

2007

Sumário

Introdução: O que é...

Ox é uma linguagem de programação matricial orientada a objetos com uma extensa biblioteca de funções matemáticas e estatísticas.

- Duas versões
  - Console (*free* para uso acadêmico)
  - Profissional

- *Ox console* × *OxEdit*

OxEdit

OxEdit é um editor de textos com opções úteis para a programação em Ox.

Algumas características do *OxEdit* são:

- Coloração usada para distinguir comandos, constantes e comentários, o que facilita a leitura do código e a identificação de erros.
- Facilita a introdução ou a exclusão de comentários no código do programa.
- Roda programas Ox facilmente e exibe os resultados do programa numa nova janela de título *Ox Output*.

OxEdit

OxEdit é um editor de textos com opções úteis para a programação em Ox.

Algumas características do *OxEdit* são:

- Coloração usada para distinguir comandos, constantes e comentários, o que facilita a leitura do código e a identificação de erros.
- Facilita a introdução ou a exclusão de comentários no código do programa.
- Roda programas Ox facilmente e exibe os resultados do programa numa nova janela de título *Ox Output*.

OxEdit

OxEdit é um editor de textos com opções úteis para a programação em Ox.

Algumas características do *OxEdit* são:

- Coloração usada para distinguir comandos, constantes e comentários, o que facilita a leitura do código e a identificação de erros.
- Facilita a introdução ou a exclusão de comentários no código do programa.
- Roda programas Ox facilmente e exibe os resultados do programa numa nova janela de título *Ox Output*.

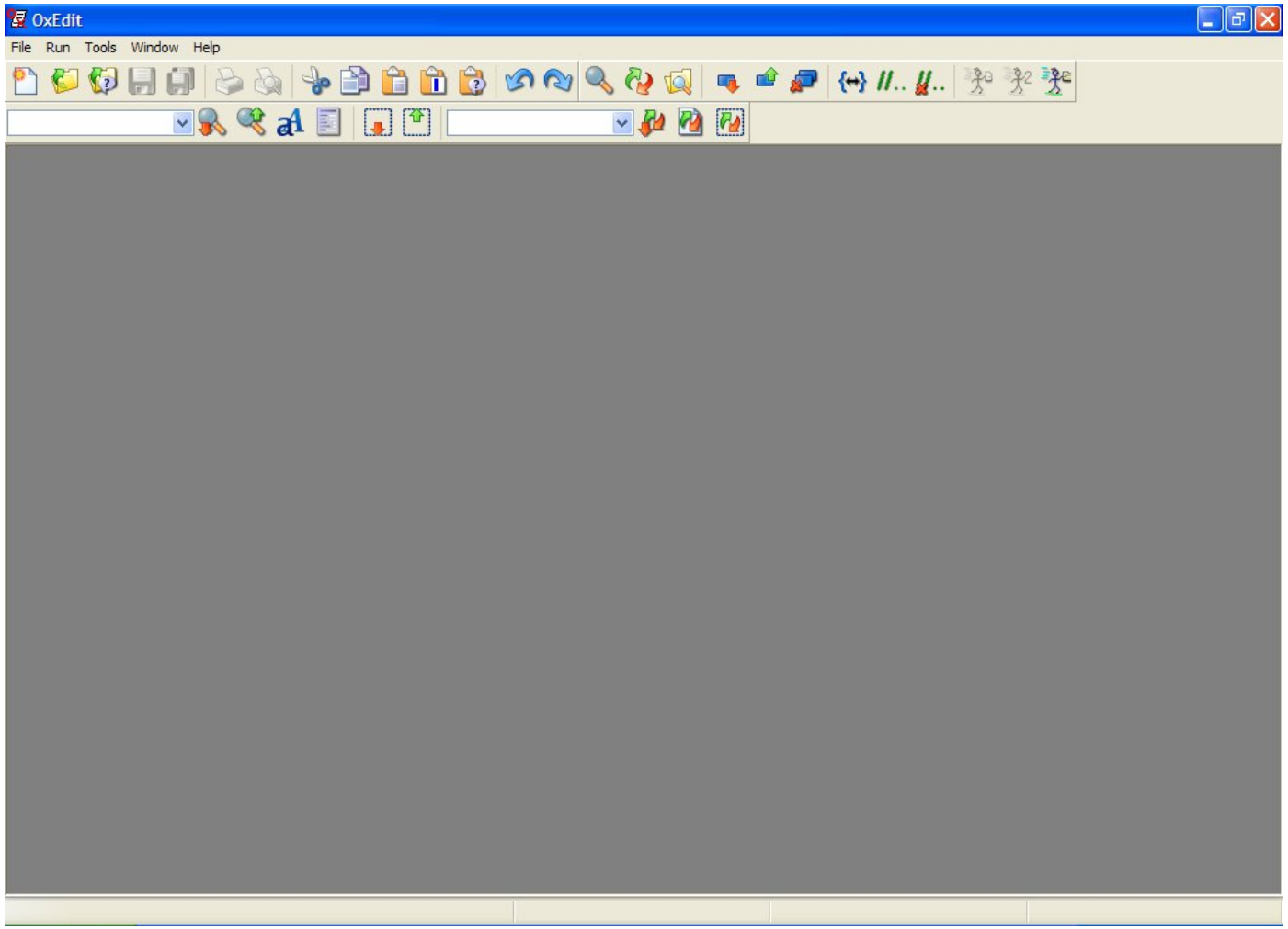

Figura: Janela do *OxEdit*.

Download

- http://www.doornik.com

- http://www.dme.ufrj.br/~henriqueneto

Pontos Fortes

- *Free* para uso acadêmico
- Disponível para diversas plataformas (sistemas operacionais)
- Velocidade
- Precisão
- Poder (boa seleção de funções matemáticas e estatísticas, rotinas de maximização, integração numérica, derivação, etc...)

Pontos Fortes

- *Free* para uso acadêmico
- Disponível para diversas plataformas (sistemas operacionais)
- Velocidade
- Precisão
- Poder (boa seleção de funções matemáticas e estatísticas, rotinas de maximização, integração numérica, derivação, etc...)

Pontos Fortes

- *Free* para uso acadêmico
- Disponível para diversas plataformas (sistemas operacionais)
- Velocidade
- Precisão
- Poder (boa seleção de funções matemáticas e estatísticas, rotinas de maximização, integração numérica, derivação, etc...)

Pontos Fortes

- *Free* para uso acadêmico
- Disponível para diversas plataformas (sistemas operacionais)
- Velocidade
- Precisão
- Poder (boa seleção de funções matemáticas e estatísticas, rotinas de maximização, integração numérica, derivação, etc...)

Pontos Fortes

- *Free* para uso acadêmico
- Disponível para diversas plataformas (sistemas operacionais)
- Velocidade
- Precisão
- Poder (boa seleção de funções matemáticas e estatísticas, rotinas de maximização, integração numérica, derivação, etc...)

Desvantagens

- Suporte gráfico
- Interatividade

Ajuda

O livro de referência do Ox é (Doornik, 2002). Grande parte de seu conteúdo está disponível na ajuda on line do Ox

(`http://www.nuff.ox.ac.uk/Users/Doornik/doc/ox/index.html`).

Este sistema de ajuda está no formato html e pode ser lido a partir de qualquer navegador. A ajuda do Ox companha também o software e pode ser consultada usando o menu "Help-Ox Help" do *OxEdit*.

Sentenças

Sentenças são comandos para realizar uma tarefa como, por exemplo:

- Cálculos,
- Declaração de Variáveis,
- Manipulação de Variáveis e etc...

Alguns Detalhes

- Os arquivos com códigos Ox têm extensão “.ox”
- Ox faz distinção entre letras minúsculas e maiúsculas.
- Uma sentença deve ser sempre finalizada com “;”.

Primeiro Programa

Primeiro Programa

```
#include <oxstd.h>

main(){
   print("Oi!");
}
```

Primeiro programa.ox

- A primeira linha inclui um arquivo de cabeçalho. A função deste arquivo (oxstd.h) é declarar algumas funções padrão do Ox que poderão ser usadas ao longo do programa.
- Este programa tem uma função, chamada `main`. Um programa Ox inicia a execução na função `main`. Sem esta função, nada irá acontecer.
- `print` é uma função padrão usada para exibir um resultado.

Primeiro Programa

Primeiro Programa

```
#include <oxstd.h>

main(){
   print("Oi!");
}
```

Primeiro programa.ox

- A primeira linha inclui um arquivo de cabeçalho. A função deste arquivo (oxstd.h) é declarar algumas funções padrão do Ox que poderão ser usadas ao longo do programa.
- Este programa tem uma função, chamada `main`. Um programa Ox inicia a execução na função `main`. Sem esta função, nada irá acontecer.
- `print` é uma função padrão usada para exibir um resultado.

Primeiro Programa

Primeiro Programa

```
#include <oxstd.h>

main(){
   print("Oi!");
}
```

Primeiro programa.ox

- A primeira linha inclui um arquivo de cabeçalho. A função deste arquivo (oxstd.h) é declarar algumas funções padrão do Ox que poderão ser usadas ao longo do programa.
- Este programa tem uma função, chamada `main`. Um programa Ox inicia a execução na função `main`. Sem esta função, nada irá acontecer.
- `print` é uma função padrão usada para exibir um resultado.

Comentários

Para inserir comentários em um código Ox, há duas opções:

- // $\longrightarrow$ comentários numa mesma linha
- /* ... */ $\longrightarrow$ comentário de várias linhas

Comentarios.ox

Comandos e Operações Básicas: Exibindo Informações (print e println)

Os comandos `print` e `println` exibem resultados. A diferença entre os comandos é que cada `println` usa uma linha enquanto vários comandos `print` usam uma mesma linha.

print e println.ox

Declaração de variáveis (decl)

O comando `decl` declara variáveis.

OBS:

- Diferentemente do C, declarações não precisam ser feitas no início do bloco.
- A variável declarada é removida assim que o bloco no qual ela foi declarada finaliza.

Variáveis globais: São variáveis visíveis em qualquer parte do programa (funções). Devem ser declaradas antes da função `main`.

Constantes: São variáveis que não serão alteradas ao longo do programa. Devem ser declaradas antes da função `main`

Declaracao de variaveis.ox

Declaração de variáveis (decl)

O comando `decl` declara variáveis.

OBS:

- Diferentemente do C, declarações não precisam ser feitas no início do bloco.
- A variável declarada é removida assim que o bloco no qual ela foi declarada finaliza.

Variáveis globais: São variáveis visíveis em qualquer parte do programa (funções). Devem ser declaradas antes da função `main`.

Constantes: São variáveis que não serão alteradas ao longo do programa. Devem ser declaradas antes da função `main`

Declaracao de variaveis.ox

Declaração de variáveis (decl)

O comando `decl` declara variáveis.

OBS:

- Diferentemente do C, declarações não precisam ser feitas no início do bloco.
- A variável declarada é removida assim que o bloco no qual ela foi declarada finaliza.

Variáveis globais: São variáveis visíveis em qualquer parte do programa (funções). Devem ser declaradas antes da função `main`.

Constantes: São variáveis que não serão alteradas ao longo do programa. Devem ser declaradas antes da função `main`

Declaracao de variaveis.ox

Operações Básicas

Operações Básicas	
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Potenciação	^

Os números 0 e 1 no Ox representam também os valores lógicos "Verdadeiro" e "Falso", respectivamente.

Operações Lógicas	
Conjunção	&&
Disjunção	||
Igualdade	==
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Operacoes.ox

Vetores e Matrizes: Criação

A maneira mais usual de criar uma matriz é definindo seus elementos. No Ox podemos definir uma matriz introduzindo seus elementos entre "<" e ">". A separação de colunas é feita usando "," e a separação de linhas usando ";".

Alguns comandos especiais para criação de matrizes:

- `unit(5)` $\longrightarrow$ matriz Identidade 5×5.
- `zeros(3,2)` $\longrightarrow$ matriz nula 3×2.
- `ones(3,2)` $\longrightarrow$ matriz 3×2 com todas entradas iguais a 1.
- `constant(2,4,3)` $\longrightarrow$ matriz 3×2 com todas as entradas iguais a 2.
- `diag(<1,2,3>)` $\longrightarrow$ matriz 3×3 com $(1, 2, 3)$ na diagonal principal.
- `range(3,30,4)` $\longrightarrow$ cria uma sequência de números com incremento 4, iniciando no 3 e não ultrapassando 30. O resultado deste comando é um vetor linha.

No Ox, a criação de vetores é idêncica à criação de matrizes. Ou seja, para criar um vetor linha, basta criar uma matriz com uma única linha e, para criar um vetor coluna, basta criar uma matriz com uma única coluna.

Criando matrizes.ox

Modificação

Alguns comandos especiais para modificar matrizes:

- `setdiagonal`
- `diagonalize`
- `setlower`
- `setupper`
- `vec`
- `shape`
- `reshape`
- `setbounds`

Para saber mais sobre estas funções, consulte a ajuda do Ox[1].

[1] http://www.nuff.ox.ac.uk/Users/Doornik/doc/ox/index.html

Indexação

Para fazer referência a uma linha, uma coluna ou um elemento de uma matriz, devemos lembrar que a indexação no Ox inicia no 0 e não no 1.

Exemplo: Consideremos a matriz

$$
\begin{array}{c}
\text{coluna 1} \quad \text{coluna 3} \\
\uparrow \qquad \uparrow \\
A = \begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{bmatrix} \begin{array}{l} \rightarrow \text{linha 0} \\ \rightarrow \text{linha 1} \\ \rightarrow \text{linha 2} \end{array} \\
\downarrow \qquad \downarrow \\
\text{coluna 0} \quad \text{coluna 2}
\end{array}
$$

- `A[1][3]` retorna o elemento da linha 2, coluna 4, ou seja, 24
- `A[0][0]` retorna o elemento da linha 1, coluna 1, ou seja, 11

OBS: No caso de um vetor linha, não é necessário fazer referência à linha zero, por exemplo: se $B = [3 \quad 7 \quad 1 \quad 6]$, então `B[1]=B[0][1]=7`.

Indexação

Para fazer referência a uma linha, uma coluna ou um elemento de uma matriz, devemos lembrar que a indexação no Ox inicia no 0 e não no 1.

Exemplo: Consideremos a matriz

$$
\begin{array}{c}
\text{coluna 1} \quad \text{coluna 3} \\
\uparrow \qquad \uparrow \\
A = \begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{bmatrix} \begin{array}{l} \rightarrow \text{linha 0} \\ \rightarrow \text{linha 1} \\ \rightarrow \text{linha 2} \end{array} \\
\downarrow \qquad \downarrow \\
\text{coluna 0} \quad \text{coluna 2}
\end{array}
$$

- `A[1][3]` retorna o elemento da linha 2, coluna 4, ou seja, 24
- `A[0][0]` retorna o elemento da linha 1, coluna 1, ou seja, 11

OBS: No caso de um vetor linha, não é necessário fazer referência à linha zero, por exemplo: se $B = [3 \;\; 7 \;\; 1 \;\; 6]$, então `B[1]=B[0][1]=7`.

Operações

Sejam `A` e `B` duas matrizes.

Operações básicas:

- `A+B` → soma de matrizes
- `A-B` → subtração de matrizes
- `A*B` → produto de matrizes

Operações elemento a elemento:

- `A.*B` → produto
- `A./B` → divisão
- `A.^3` → potenciação (eleva a 3 cada entrada da matriz)

Manipulação

Seja `A` uma matriz.

- `A'` → transposta da matriz `A`
- `dropc(A,2)` → matriz `A` sem a terceira coluna
- `dropr(A,0)` → matriz `A` sem a primeira linha
- `sumc(A)` → somatório por colunas (retorna vetor linha)
- `sumr(A)` → somatório por linhas (retorna vetor coluna)
- `prodc(A)` → produtório por colunas (retorna vetor linha)
- `prodr(A)` → produtório por linhas (retorna vetor coluna)
- `meanc(A)` → média por colunas (retorna vetor linha)
- `meanr(A)` → média por linhas (retorna vetor coluna)
- `varc(A)` → variância por colunas (retorna vetor linha)
- `varr(A)` → variância por linhas (retorna vetor coluna)

Manipulando matrizes.ox

Para juntar (concatenar) duas matrizes, podemos usar os conectivos "|" e "~".

- "|" → concatenação vertical.
- "~" → concatenação horizontal.

Por exemplo, se $A = \begin{bmatrix} 4 & 5 \\ 1 & 3 \end{bmatrix}$ e $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, então:

- `A|B` retorna

$$\left.\begin{matrix} 4 & 5 \\ 1 & 3 \end{matrix}\right\rangle A$$
$$\left.\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}\right\rangle B$$

- `A~B` retorna

$$\underbrace{\begin{matrix} 4 & 5 \\ 1 & 3 \end{matrix}}_{A} \quad \underbrace{\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}}_{B}$$

Manipulando matrizes 2.ox

Arquivos: Inclusão Usual

A grosso modo, podemos fazer referência ao conteúdo de dois tipos de arquivos:

- Arquivos criados pelo usuário
- Arquivos de sistema ou bibliotecas

Para incluir um arquivo criado pelo usuário em algum ponto do código Ox, é comum usar o comando

```
#include "nome do arquivo"
```

Já para incluir o conteúdo de um arquivo de sistema ou biblioteca, é comum usar o comando

```
#include <nome do arquivo>
```

OBS: Informando apenas o nome do arquivo, sem sua localização, o Ox irá procura-lo na pasta que contém o arquivo executado (pasta padrão). Para fazer referência a um arquivo em uma subpasta da pasta padrão, use a expressão abaixo no local do nome do arquivo.

```
./Subpasta 1/Subpasta 2/Nome do arquivo
```

Include.ox
Subarquivo.ox

Arquivos: Inclusão Usual

A grosso modo, podemos fazer referência ao conteúdo de dois tipos de arquivos:

- Arquivos criados pelo usuário
- Arquivos de sistema ou bibliotecas

Para incluir um arquivo criado pelo usuário em algum ponto do código Ox, é comum usar o comando

```
#include "nome do arquivo"
```

Já para incluir o conteúdo de um arquivo de sistema ou biblioteca, é comum usar o comando

```
#include <nome do arquivo>
```

OBS: Informando apenas o nome do arquivo, sem sua localização, o Ox irá procura-lo na pasta que contém o arquivo executado (pasta padrão). Para fazer referência a um arquivo em uma subpasta da pasta padrão, use a expressão abaixo no local do nome do arquivo.

```
./Subpasta 1/Subpasta 2/Nome do arquivo
```

Include.ox
Subarquivo.ox

Lendo e Salvando Variáveis em Arquivos

Para salvar uma variável em um arquivo, basta usar:

```
savemat(nome do arquivo, nome da variável)
```

O tipo de arquivo depende da extensão do nome do arquivo:

- .mat → ASCII
- .dat → ASCII com informações de leitura
- .in7 → PcGive 7
- .xls → Excel
- .wks and .wk1 → Lotus
- .fmt: arquivo matriz Gauss
- .dht: arquivo de dados Gauss
- qualquer outra extensão: arquivo .mat

Para ler informações basta usar `loadmat` ao invés de `savemat`.

savemat.ox

Gravação em arquivos texto (txt)

Para salvar informações em um arquivo texto é preciso antes abrir um arquivo para gravação usando o comando `fopen`.
Já para gravar as informações, podemos usar os comandos `fprint`, similar ao comando `print`, e `fprintln`, similar ao comando `println`.

Exemplo

```
decl fileteste=fopen("./Arquivos criados/
teste.txt","a");
fprintln(fileteste, "Anaceptilpó ", 3);
```

OBS:

- Se o arquivo não existir, o comando `fopen` cria o arquivo.
- A expressão `"a"` indica o modo de gravação *append*. Neste modo, se o arquivo `"teste.txt"` já existir, a informação será gravada no final do arquivo, sem substituir o conteúdo existente. Há também o modo `"w"`, *write*, que substitui o conteúdo no caso do arquivo já existir. Para outros modos de gravação, consulte a ajuda do Ox[2].

Teste.txt
Gravacao.ox

[2] http://www.nuff.ox.ac.uk/Users/Doornik/doc/ox/index.html

Gravação em arquivos texto (txt)

Para salvar informações em um arquivo texto é preciso antes abrir um arquivo para gravação usando o comando `fopen`.
Já para gravar as informações, podemos usar os comandos `fprint`, similar ao comando `print`, e `fprintln`, similar ao comando `println`.

Exemplo

```
decl fileteste=fopen("./Arquivos criados/
teste.txt","a");
fprintln(fileteste, "Anaceptilpó ", 3);
```

OBS:

- Se o arquivo não existir, o comando `fopen` cria o arquivo.
- A expressão `"a"` indica o modo de gravação *append*. Neste modo, se o arquivo `"teste.txt"` já existir, a informação será gravada no final do arquivo, sem substituir o conteúdo existente. Há também o modo `"w"`, *write*, que substitui o conteúdo no caso do arquivo já existir. Para outros modos de gravação, consulte a ajuda do Ox[2].

Teste.txt
Gravacao.ox

[2] http://www.nuff.ox.ac.uk/Users/Doornik/doc/ox/index.html

Algumas Bibliotecas

Segue abaixo o uso de algumas bibliotecas importantes.

- `#include <oxstd.h>` → biblioteca padrão
- `#include <oxprob.h>` → biblioteca de probabilidade
- `#include <oxdraw.h>` → biblioteca de recursos gráficos
- `#include <oxfloat.h>` → permite usar `pi=3.141516...`

Algumas Bibliotecas

Segue abaixo o uso de algumas bibliotecas importantes.

- `#include <oxstd.h>` → biblioteca padrão
- `#include <oxprob.h>` → biblioteca de probabilidade
- `#include <oxdraw.h>` → biblioteca de recursos gráficos
- `#include <oxfloat.h>` → permite usar `pi=3.141516...`

Algumas Bibliotecas

Segue abaixo o uso de algumas bibliotecas importantes.

- `#include <oxstd.h>` → biblioteca padrão
- `#include <oxprob.h>` → biblioteca de probabilidade
- `#include <oxdraw.h>` → biblioteca de recursos gráficos
- `#include <oxfloat.h>` → permite usar `pi=3.141516...`

Algumas Bibliotecas

Segue abaixo o uso de algumas bibliotecas importantes.

- `#include <oxstd.h>` → biblioteca padrão
- `#include <oxprob.h>` → biblioteca de probabilidade
- `#include <oxdraw.h>` → biblioteca de recursos gráficos
- `#include <oxfloat.h>` → permite usar `pi=3.141516...`

Biblioteca de Probabilidade: Uso

Com o Ox é possivel obter informações sobre uma grande variedade de distribuições de probabilidade. Com o prefixo *dens* obtem-se a função densidade de probabilidade, com *prob* a função de distribuição, com *quan* quantis e com *ran* pode-se gerar valores das distribuições.

Para informações detalhadas sobre a manipulação de cada distibuição, consulte a ajuda do Ox[3].

[3] http://www.nuff.ox.ac.uk/Users/Doornik/doc/ox/index.html

function *density*		
`densbeta`		
Beta (a, b),	$\frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$	$0 < x < 1; a > 0, b > 0$
`densbinomial`		
Binomial(n,p)	$\binom{n}{x} p^x q^{n-x}$	$x = 0, 1, \ldots, n;\ 0 \leq p \leq 1$
`denscauchy`		
Cauchy,	$\left(\pi\left(1 + x^2\right)\right)^{-1}$	
`densexp`		
Exponential,	$\lambda e^{-\lambda x}$	$x > 0; \lambda > 0$
`densextremevalue`		
Extreme Value, (Type I or Gumbel)	$\exp\left[-e^{-(x-\alpha)/\beta}\right]$	$\beta > 0$
`densgamma`		
Gamma	$\frac{a^r}{\Gamma(r)} x^{r-1} e^{-ax}$	$x > 0; r > 0, a > 0$
`densgeometric`		
Geometric	pq^x	$x = 0, 1, \ldots;\ \mu > 0$

function *density*		
`densgh` Generalized hyperbolic, see (11.5)		
`densgig` Generalized inverse Gaussian, see (11.4)		
`denshypergeometric` Hypergeometric	$\binom{K}{x}\binom{M-K}{n-x}/\binom{M}{n}$	$x = 0, 1, \ldots, n$
Pr[x white balls | sample n without replacement from K white balls and M in total]		
`densinvgaussian` Inverse Gaussian,	$\left(\frac{\lambda}{2\pi x^3}\right)^{1/2} \exp\left[-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right]$	$x > 0;\ \lambda > 0, \mu > 0$
`denskernel` kernel,	see below	
`denslogarithmic` Logarithmic	$\frac{-\alpha^x}{x \log(1-\alpha)}$	$x = 1, 2 \ldots;\ 0 < \alpha < 1$
`denslogistic` Logistic,	$\left[1 + e^{-(x-\alpha)/\beta}\right]^{-1}$	$\beta > 0,$

function *density*		
`denslogn`		
Lognormal,	$\frac{1}{x(2\pi)^{1/2}} \exp\left[-(\log x)^2/2\right]$	$x > 0$
`densmises`		
von Mises,	see (11.7) below	
`densnegbin`		
Negative Binomial	$\binom{k+x-1}{x} p^k q^x$	$x = 0, 1, \ldots;\ 0 < p \leq 1,\ k > 0$
`denspareto`		
Pareto	$ak^a x^{-(a+1)}$	$x \geq k > 0;\ a > 0$
`denspoisson`		
Poisson	$\frac{e^{-\mu}\mu^x}{x!}$	$x = 0, 1, \ldots;\ \mu > 0$
`densweibull`		
Weibull	$abx^{b-1} \exp\left(-ax^b\right)$	$x > 0;\ a > 0, b > 0$

function
density

`denskernel` arguments:

`itype`	kernel name	form	
'e'	Epanechnikov	$0.75(1-x^2)$	$\|x\|<1$
'b'	Biweight (Quartic)	$(15/16)(1-x^2)^2$	$\|x\|<1$
't'	Triangular	$1-\|x\|$	$\|x\|<1$
'g'	Gaussian (Normal)	$(2\pi)^{-1/2}\exp\left[-x^2/2\right]$	
'r'	Rectangular (Uniform)	0.5	$\|x\|<1$

If $X \sim GIG(\nu,\delta,\gamma)$ then it has a generalized inverse Gaussian density:

$$\frac{(\gamma/\delta)^{\nu}}{2K_{\nu}(\delta\gamma)}x^{\nu-1}\exp\left\{-\frac{1}{2}(\delta^2x^{-1}+\gamma^2x)\right\}, \quad \gamma,\delta\geq 0, \quad \nu\in\mathbb{R}, \quad x>0, \tag{11.4}$$

where K_{ν} is a modified Bessel function of the third kind.

function
density

The generalized hyperbolic distribution with $\mu = 0$, $GH(\nu, \delta, \gamma, \beta)$ has support on the real line. The density is :

$$\frac{(\gamma/\delta)^{\nu}}{\sqrt{2\pi}\alpha^{\nu-\frac{1}{2}}K_{\nu}(\delta\gamma)}\left\{\delta^2+x^2\right\}^{\frac{1}{2}(\nu-\frac{1}{2})}K_{\nu-\frac{1}{2}}\left(\alpha\left[\delta^2+x^2\right]^{1/2}\right)e^{\beta x}, \quad (11.5)$$

where $\alpha = \sqrt{\beta^2+\gamma^2}$. For $\mu \neq 0$ replace x by $x-\mu$.

Funções Específicas: Funções Matriciais

Funções Matriciais

Seja `A` uma matriz.

- `rows(A)` → número de linhas
- `columns(A)` → número de colunas
- `invert(A)` → inversa
- `invertsym(A)` → inversa, usada no caso de A ser simétrica (mais eficiente)
- `determinant(A)` → determinante
- `logdet(A,{" "})` → logaritmo do determinante (evita erros computacionais)
- `choleski(A)` → decomposição de choleski

Funções Matemáticas

As funções a seguir quando aplicadas em uma matriz, transformam cada elemento da matriz. Por exemplo, `log(A)` é uma matriz obtida aplicando o logaritmo em cada entrada de `A`.

Funções Matemáticas

- `exp(A)` $\longrightarrow$ função exponencial $e^{(\cdot)}$
- `log(A)` $\longrightarrow$ logaritmo natural
- `log10(A)` $\longrightarrow$ logaritmo na base 10
- `factorial(A)` $\longrightarrow$ fatorial
- `min(A)` $\longrightarrow$ menor valor
- `max(A)` $\longrightarrow$ maior valor
- `gammafact(A)` $\longrightarrow$ função gamma $\Gamma(.)$
- `ceil(A)` $\longrightarrow$ menor inteiro maior ou igual
- `floor(A)` $\longrightarrow$ maior inteiro menor ou igual

Funções: Criação e Chamada

Para definir uma função, escreve-se o nome da função seguido de uma lista de argumentos, separados por vírgulas, entre parênteses. Exemplo:

Combinação de *n* tomado *p* a *p*

```
funexibecombinacao(argn, argp){
   decl combina=factorial(argn)/(factorial(argn-argp)*factorial(argp));
   println("Exibindo:  ", combina);
}
```

Chamando uma função

```
funexibecombinacao(4, 2);
```

OBS:

- A lista de argumentos pode ser vazia.
- Chamadas de função recursivas são permitidas.
- Uma função deve ser declarada antes de ser chamada e o número de argumentos na chamada deve coincidir com o numero de argumentos na declaração. Assim, as funções devem ser declaradas antes da função `main`.

Funções: Criação e Chamada

Para definir uma função, escreve-se o nome da função seguido de uma lista de argumentos, separados por vírgulas, entre parênteses. Exemplo:

Combinação de *n* tomado *p* a *p*

```
funexibecombinacao(argn, argp){
   decl combina=factorial(argn)/(factorial(argn-argp)*factorial(argp));
   println("Exibindo:  ", combina);
}
```

Chamando uma função

```
funexibecombinacao(4, 2);
```

OBS:

- A lista de argumentos pode ser vazia.
- Chamadas de função recursivas são permitidas.
- Uma função deve ser declarada antes de ser chamada e o número de argumentos na chamada deve coincidir com o numero de argumentos na declaração. Assim, as funções devem ser declaradas antes da função `main`.

Para que a função retorne um valor, utilize o comando `return( );` no final da função. Exemplo:

Função que retorna um valor

```
funguardacombinacao(argn, argp){
   decl combina=factorial(argn)/(factorial(argn-argp)*factorial(argp));
   return(combina);
}
```

Assim,

```
decl valorguardado=funguardacombinacao(4,2);
```

armazena a combinação de 4 tomado 2 a 2 na variável `valorguardado`.

OBS:

- O valor de um argumento pode ser modificado no corpo da função. Para não permitir esta modificação, pode-se usar a expressão `const` ao definir um argumento. Ao usar uma função que não altera um determinado argumento, use `const` sempre (maior eficiência).

Funcoes.ox

Para que a função retorne um valor, utilize o comando `return( );` no final da função. Exemplo:

Função que retorna um valor

```
funguardacombinacao(argn, argp){
   decl combina=factorial(argn)/(factorial(argn-argp)*factorial(argp));
   return(combina);
}
```

Assim,

```
decl valorguardado=funguardacombinacao(4,2);
```

armazena a combinação de 4 tomado 2 a 2 na variável `valorguardado`.

OBS:

- O valor de um argumento pode ser modificado no corpo da função. Para não permitir esta modificação, pode-se usar a expressão `const` ao definir um argumento. Ao usar uma função que não altera um determinado argumento, use `const` sempre (maior eficiência).

Funcoes.ox

Ciclos e Condições: if - else if - else

O comando `if` é usado para validar uma condição e, mediante o resultado, executar o código correspondente.

Múltiplas condições podem ser encadeadas com o comando `else if`.

O comando `else` é usado para executar um bloco de código caso as condições especificadas por `if` e `else if` forem falsas.

if - else if - else.ox

for - while - do while

Os comandos `for`, `while` e `do while` são usados para repetir um bloco de comandos várias vezes.

- O comando `for` cria um ciclo com base em um contador. O ciclo é interrompido assim que uma determinada condição que depende do contador se torna falsa.
- O comando `while` cria um ciclo com base em uma condição. O ciclo é interrompido quando a condição se torna falsa.
- O comando `do while` é similar ao comando while. A diferença é que com `do while` o bloco de comandos deve ser executado pelo menos uma vez, enquanto com `while` o bloco pode não ser executado.

for - while - do while.ox

Referências Bibliográficas

Doornik, Jurgen A.
(2002)
Object-Oriented Matrix Programming Using Ox, 3rd edn.
London: Timberlake Consultants Press and Oxford.

FIM!