Métodos de Otimização em Finanças com R

Paulo Henrique Sales Guimarães

Universidade Federal de Lavras

paulo.guimaraes@des.ufla.br

29 de setembro de 2016

Métodos de Otimização em Finanças com R

Software R

O R foi criado originalmente por Ross Ihaka e por Robert Gentleman na Universidade de Auckland, Nova Zelândia, desenvolvido por um esforço colaborativo por pessoas de todo o mundo. Sendo ao mesmo tempo uma linguagem de programação e um ambiente para computação estatística e gráfica.
Trata-se de uma linguagem de programação especializada em computação com dados. Algumas das suas principais características são o seu caráter gratuito e a sua disponibilidade para uma gama bastante variada de sistemas operacionais.

Características

- *Software* Livre (gratuito)

Características

- *Software* Livre (gratuito)
- Código Fonte Aberto

Características

- *Software* Livre (gratuito)
- Código Fonte Aberto
- Possui Diversas GUI

Características

- *Software* Livre (gratuito)
- Código Fonte Aberto
- Possui Diversas GUI
- Software Colaborativo (pacotes)

Características

- *Software* Livre (gratuito)
- Código Fonte Aberto
- Possui Diversas GUI
- Software Colaborativo (pacotes)
- Linguagem de Programação

Características

- *Software* Livre (gratuito)
- Código Fonte Aberto
- Possui Diversas GUI
- Software Colaborativo (pacotes)
- Linguagem de Programação
- Manipulação e Análise de Dados

Características

- *Software* Livre (gratuito)
- Código Fonte Aberto
- Possui Diversas GUI
- Software Colaborativo (pacotes)
- Linguagem de Programação
- Manipulação e Análise de Dados
- Usado em Vários Sistemas Operacionais (UNIX,WINDOWS,etc)

Instalação do R

1. Primeiro Passo
   - O programa de instalação do R pode ser baixado a partir do site: The R Project for Statistical Computing.

Instalação do R

1. Primeiro Passo
   - O programa de instalação do R pode ser baixado a partir do site: The R Project for Statistical Computing.
   - Devemos escolher o *CRAN Mirrors* (por exemplo, Universidade Federal do Paraná.

Instalação do R

1. Primeiro Passo
   - O programa de instalação do R pode ser baixado a partir do site: The R Project for Statistical Computing.
   - Devemos escolher o *CRAN Mirrors* (por exemplo, Universidade Federal do Paraná.
   - Sistema Operacional (por exemplo, *Windows*)

Instalação do R

1. Primeiro Passo
   - O programa de instalação do R pode ser baixado a partir do site: The R Project for Statistical Computing.
   - Devemos escolher o *CRAN Mirrors* (por exemplo, Universidade Federal do Paraná.
   - Sistema Operacional (por exemplo, *Windows*)
   - Install R for the first time (última versão: R.3.3.1 for *Windows*).

Instalação do R

1. Primeiro Passo
   - O programa de instalação do R pode ser baixado a partir do site: The R Project for Statistical Computing.
   - Devemos escolher o *CRAN Mirrors* (por exemplo, Universidade Federal do Paraná.
   - Sistema Operacional (por exemplo, *Windows*)
   - Install R for the first time (última versão: R.3.3.1 for *Windows*).
2. Segundo Passo
   - Abra o programa de instalação do R que acabou de baixar (selecione o idioma).

Instalação do R

1. Primeiro Passo
   - O programa de instalação do R pode ser baixado a partir do site: The R Project for Statistical Computing.
   - Devemos escolher o *CRAN Mirrors* (por exemplo, Universidade Federal do Paraná.
   - Sistema Operacional (por exemplo, *Windows*)
   - Install R for the first time (última versão: R.3.3.1 for *Windows*).
2. Segundo Passo
   - Abra o programa de instalação do R que acabou de baixar (selecione o idioma).
   - Clique em Avançar e depois no final Concluir.

Software R

Ao iniciar o R abrirá automaticamente o Console que é a janela na qual os comandos são digitados. Internamente ao Console, se encontra o *prompt*, conforme figura abaixo, que é um sinal indicador de que o programa está pronto para receber comando.

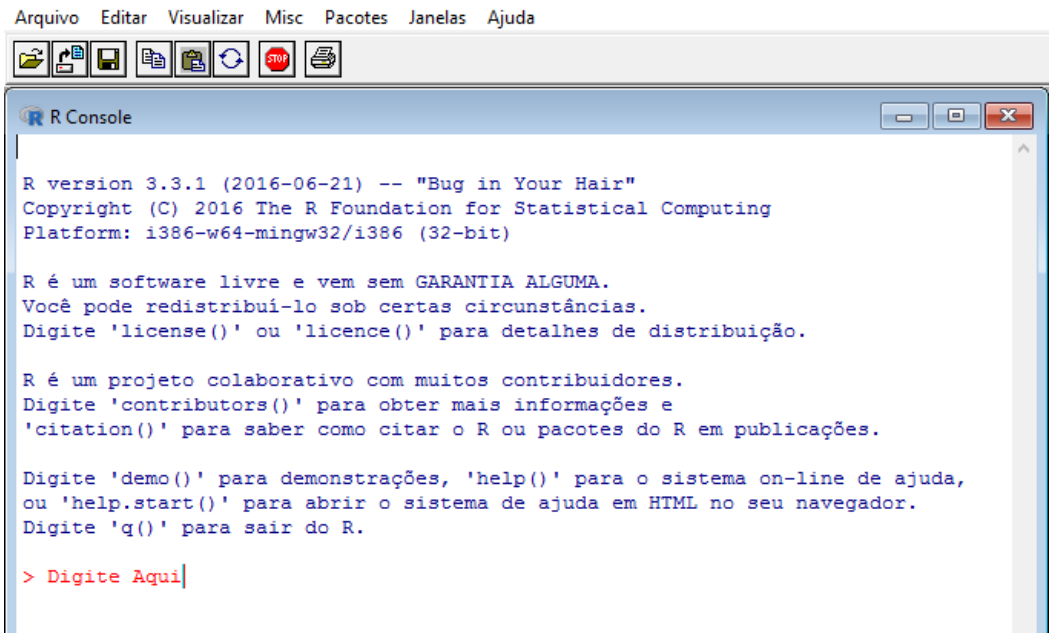


Figura: Interface do R - Console e Prompt

Software R

Uma forma mais fácil de inserir os comandos no R é por meio do *script* (Arquivo + Novo Script + Janela (Dividir Lado a Lado).

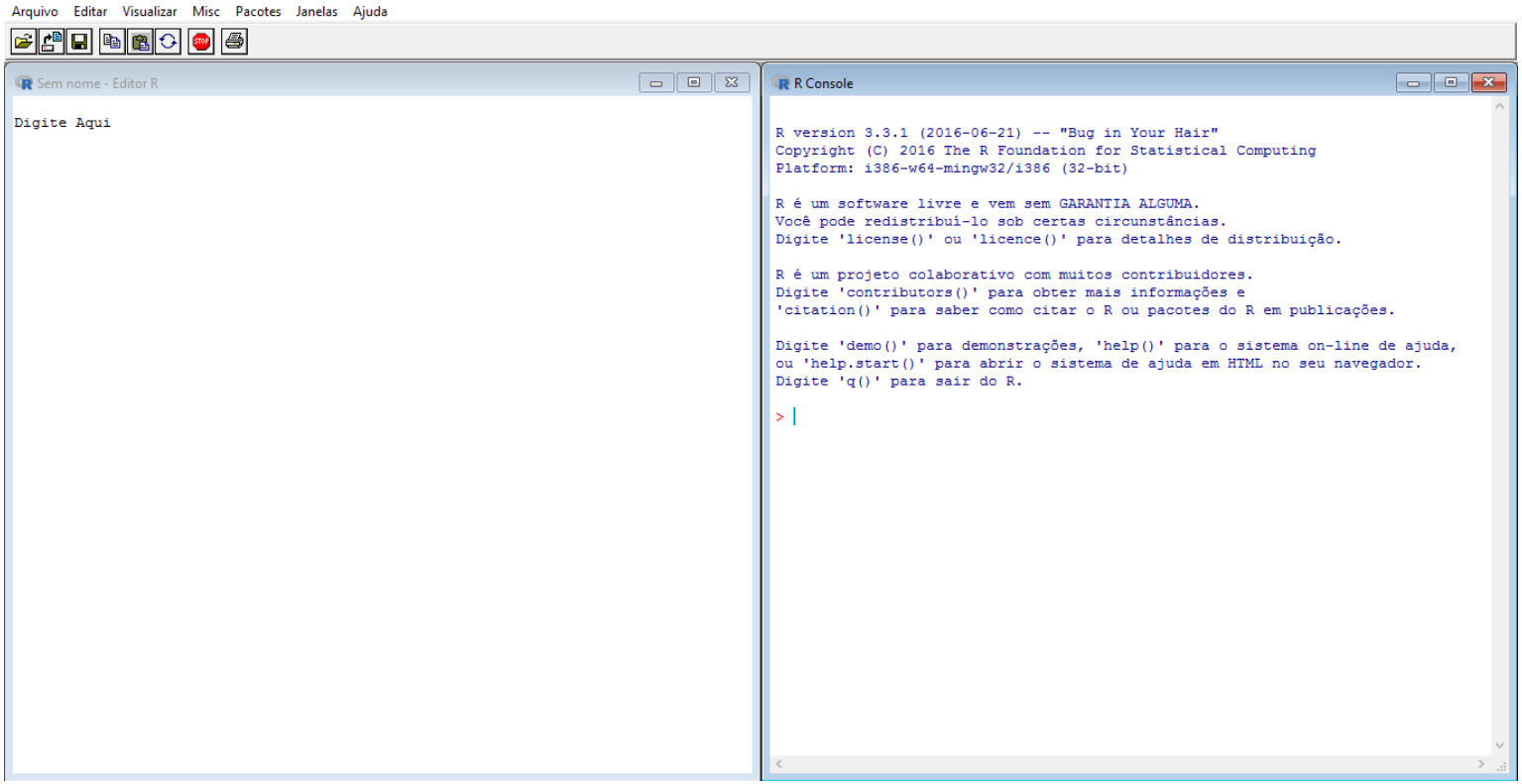


Figura: Interface do R - Script

Interface do R

R é uma linguagem de expressões com regras e sintaxe muito simples.
Faz distinção entre maiúsculas e minúsculas, de modo que os caracteres “x” e “X” são interpretados como sendo diferentes.

Tipos de Dados

Basicamente tem-se quatro tipos de dados no R: númericos, caracteres, lógicos e números complexos.

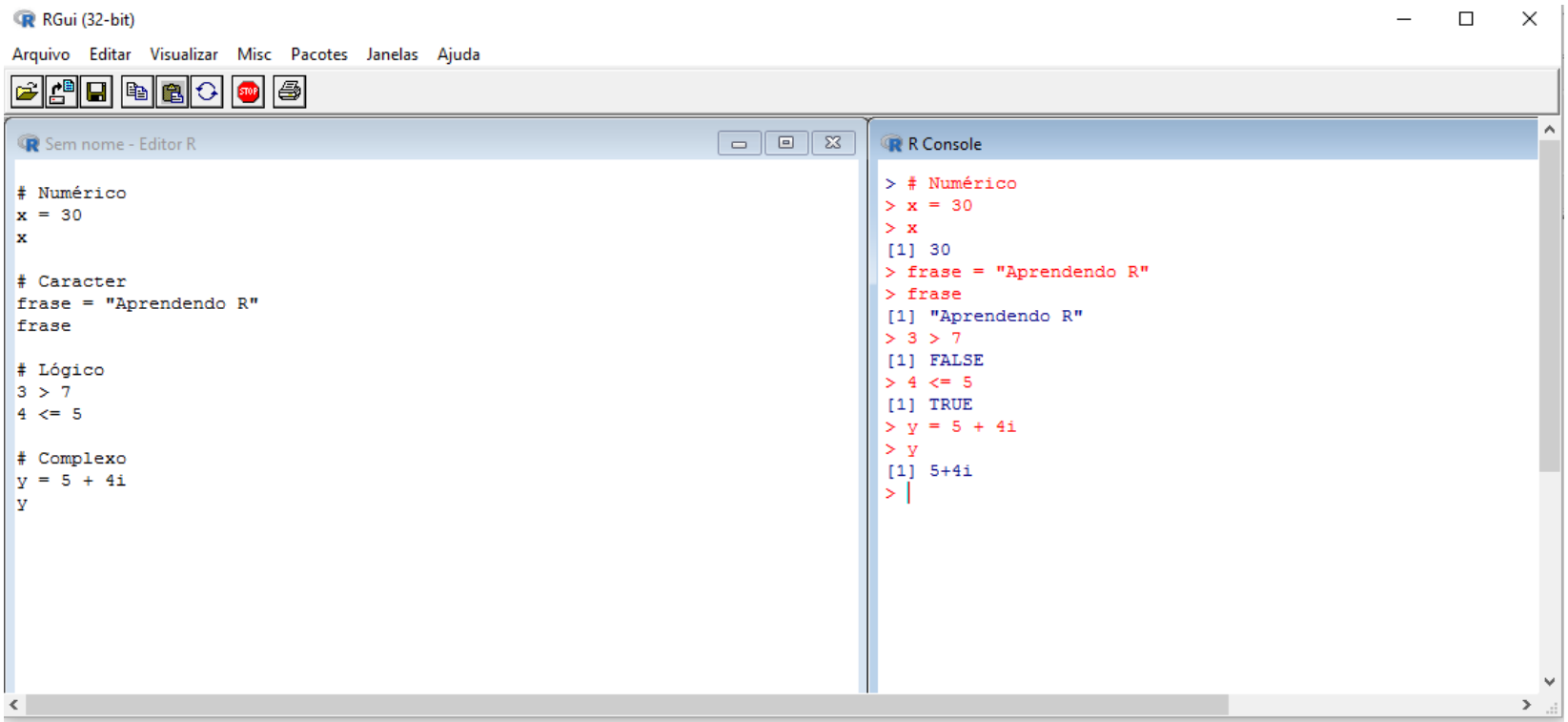


Figura: Tipos de Dados

Comandos Básicos

No R é possível fazer uma atribuição de valores de várias formas, conforme o exemplo:

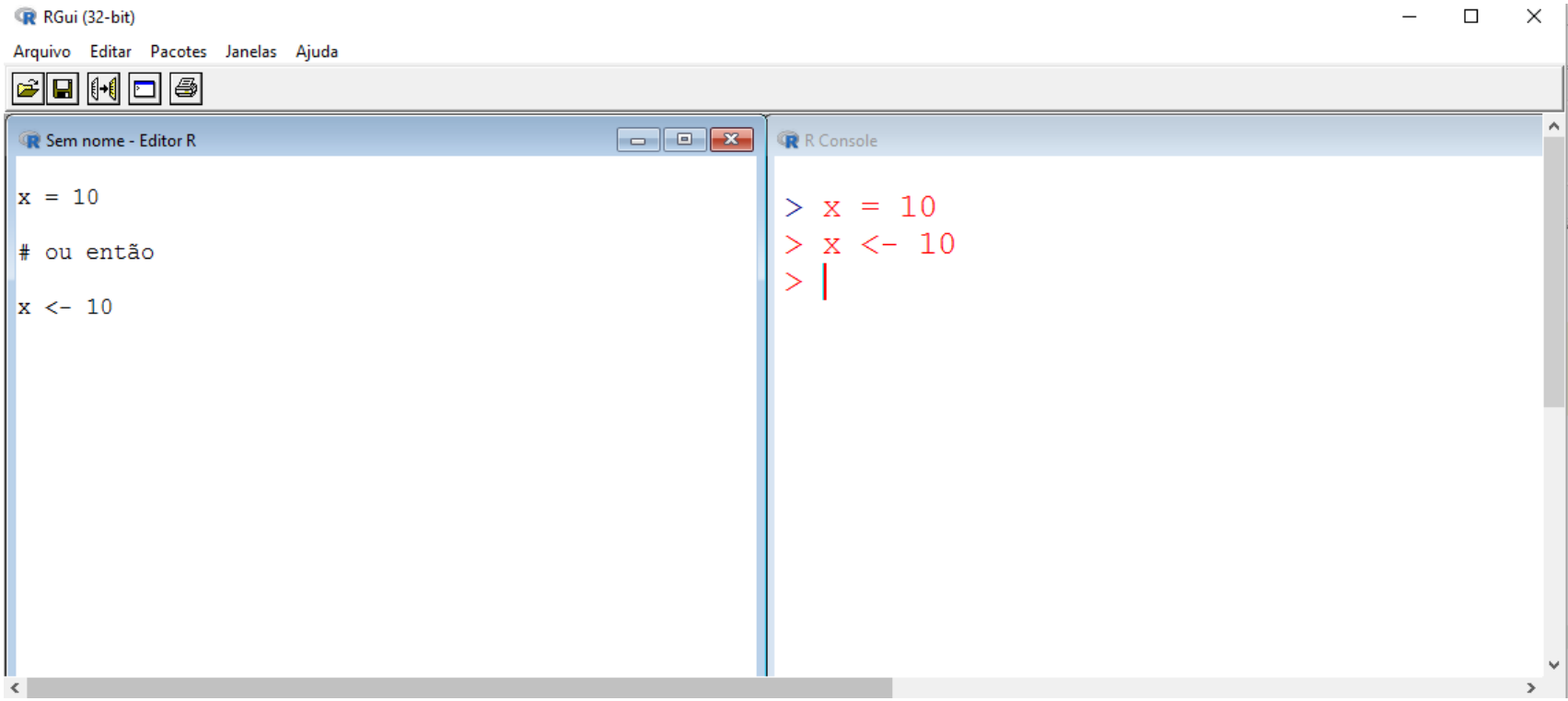


Figura: Atribuição de Valores

Comandos Básicos

O R pode funcionar como uma calculadora executando operações matemáticas.

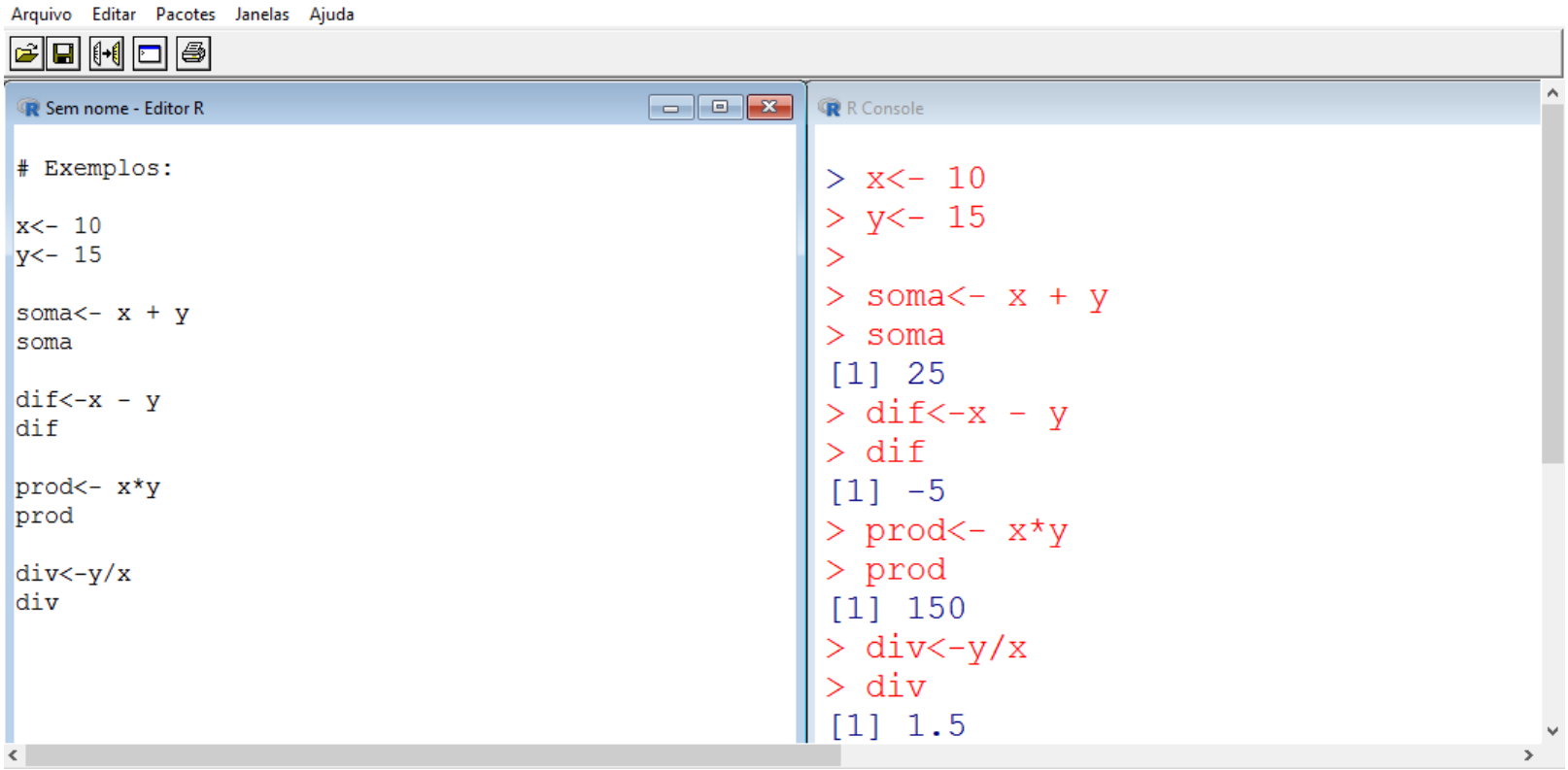


Figura: Atribuição de Valores

Tabela: Operadores Básicos

Símbolo	Descrição
<	Menor
>	Maior
<=	Menor Igual
>=	Maior Igual
==	Comparação
&	E (and)
|	Ou (or)
!	Não
!=	Diferente
TRUE ou 1	Valor verdadeiro (booleano)
FALSE ou 0	Valor falso (booleano)

Comandos Básicos

Na próxima tabela são apresentadas algumas outras operações possíveis:

Tabela: Funções Matemáticas Básicas

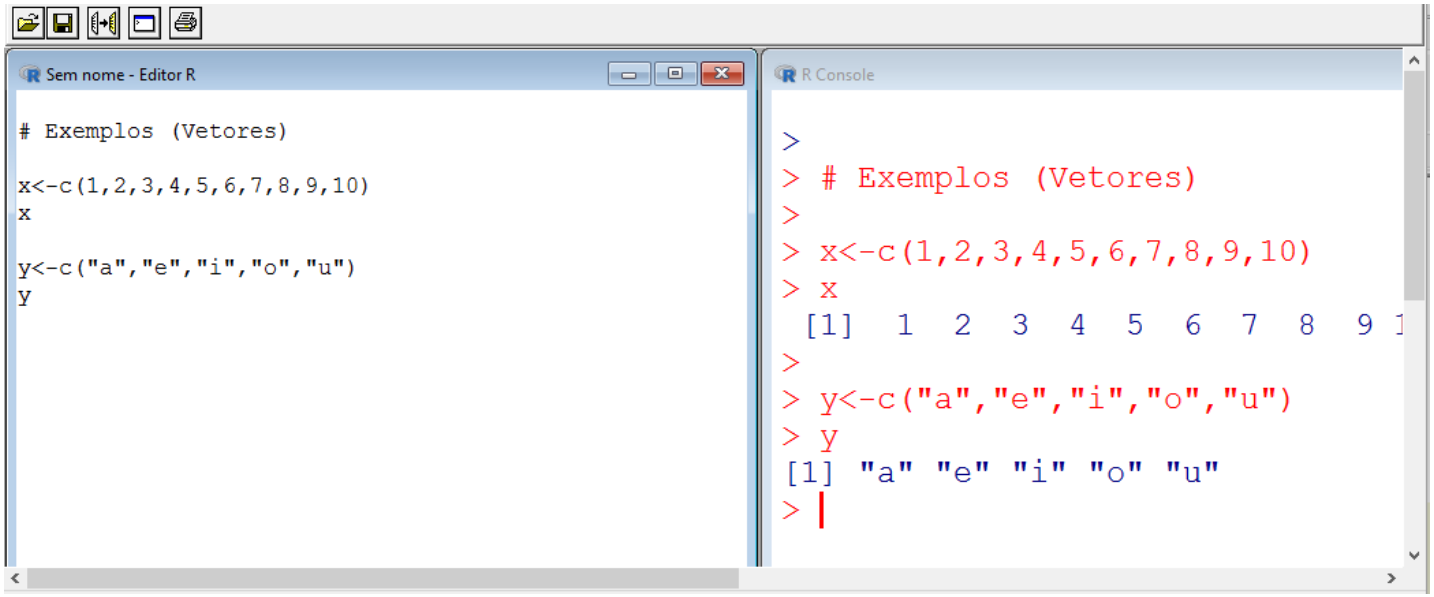
Função	Descrição
abs(x)	Valor absoluto de x
ceiling(x)	Arredondamento de x para o maior valor
exp(x)	Exponencial elevado a x
floor(x)	Arredondamento de x para o menor valor
length(x)	Número de elementos do vetor x
log(x)	Logaritmo natural de x
log(x,a)	Logaritmo de x com base a
max(x)	Seleciona o maior valor de x
min(x)	Menor valor de x
sqrt(x)	Raiz quadrada de x

Vetores

Vetores são conjuntos de dados unidimensionais. Sua principal utilidade é poder armazenar diversos dados em forma de lista e aplicar funções e operações sobre todos os dados pertencentes a determinado vetor.

Para criar um vetor devemos utilizar o comando c().

Exemplos:



Vetores

O comando seq() é bastante útil para a criação de vetores.
Exemplos:

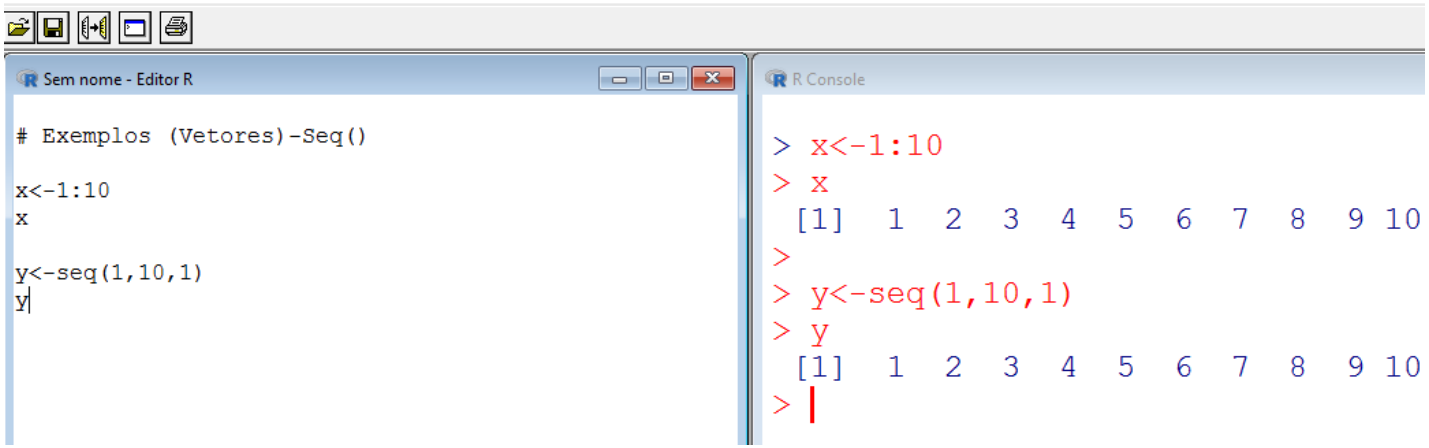


Figura: Comando seq()

Vetores

O comando rep() também pode ser utilizado para a criação de vetores.
Exemplos:

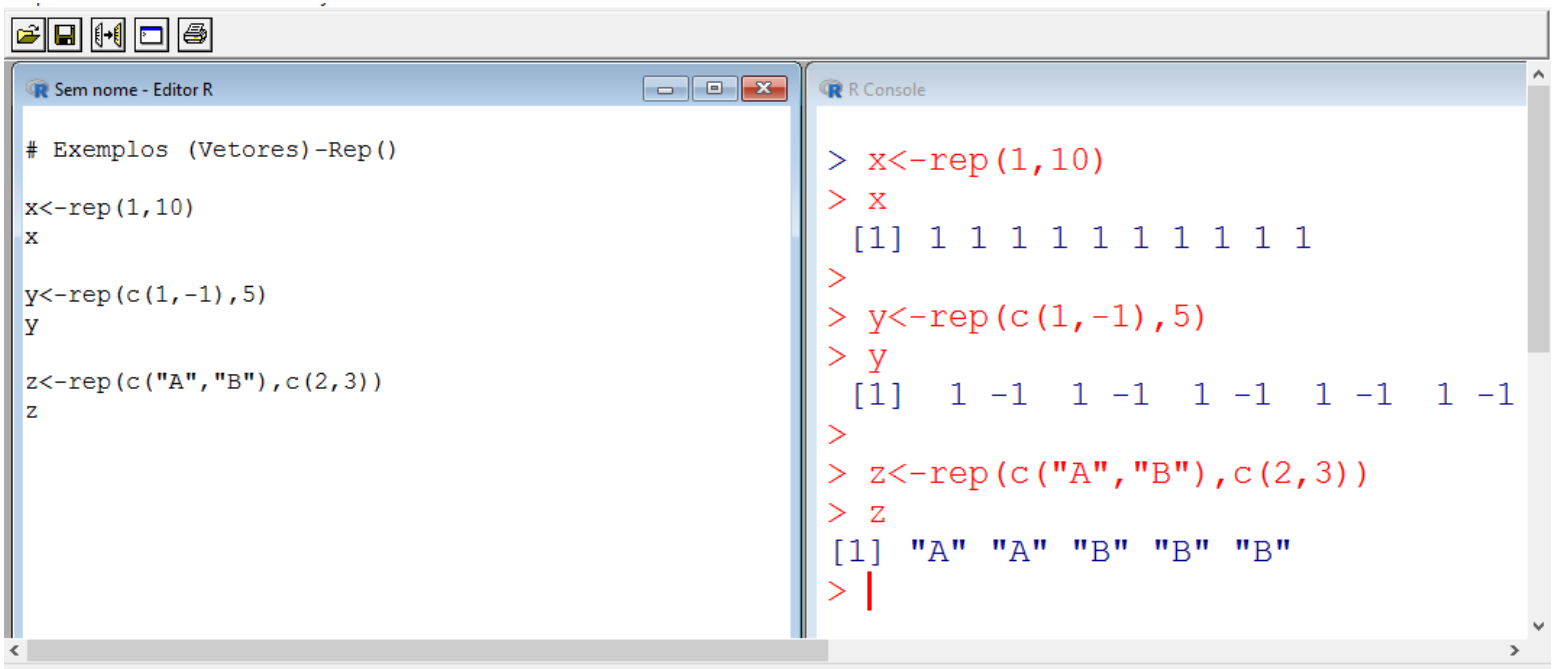


Figura: Comando req()

Matrizes

Assim como os vetores, há também várias maneiras de se criar matrizes no R.

A função matrix() recebe um vetor como argumento e o transforma em uma matriz de acordo com as dimensões especificadas.

Exemplos:

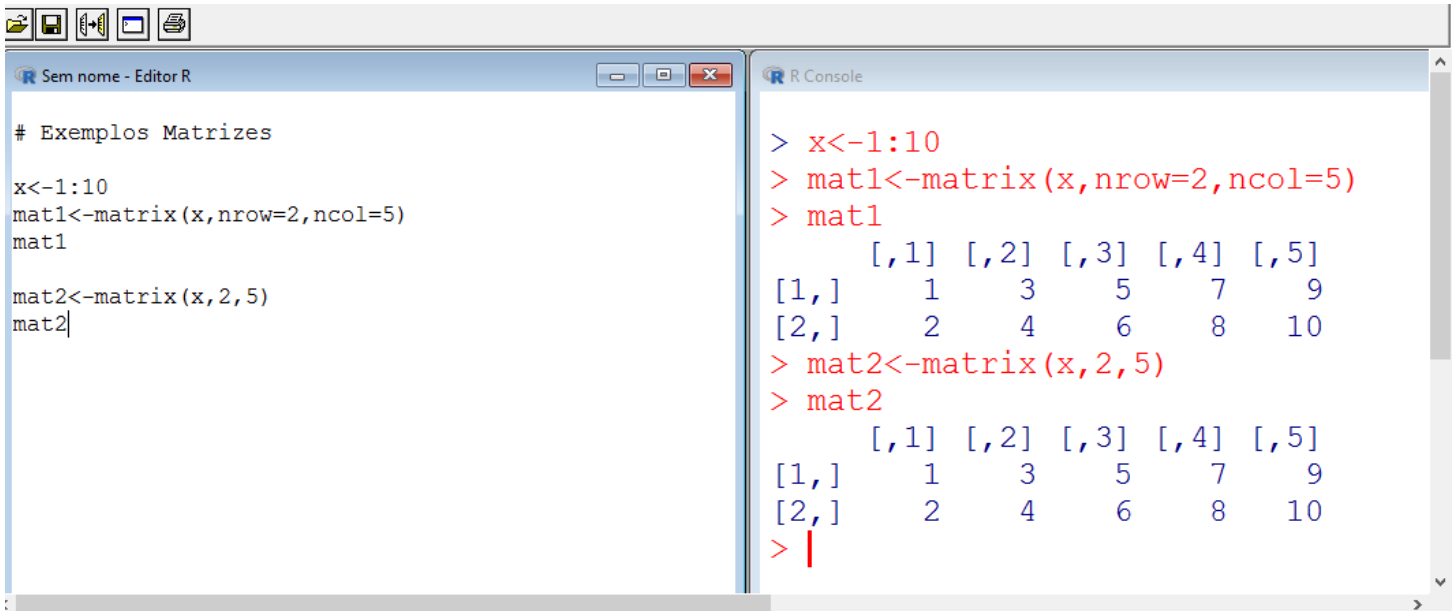


Figura: Matrizes

Matrizes

Algumas operações básicas com matrizes:

Tabela: Operações Básicas Matrizes

Função	Descrição
A%*%B	Produto matricial de A por B
t(A)	Transposta da matriz A
solve(A)	Inversa da matriz A
x=solve(A,b)	Resolve o sistema linear Ax=b
det(B)	Determinante de B
eigen(A)	Retorna os autovalores e autovetores de A

Matrizes

Alguns exemplos de operações com matrizes:

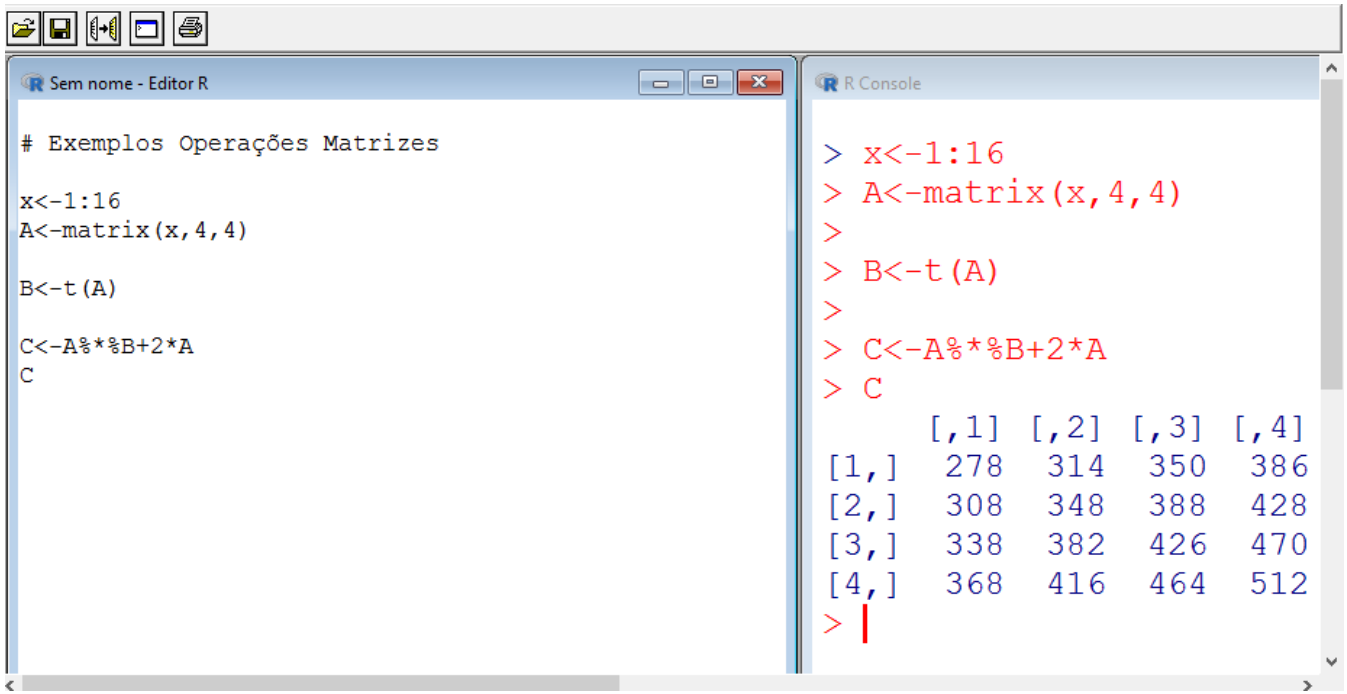


Figura: Exemplos de operações com matrizes

Dados Externos

Geralmente, trabalhamos com arquivos salvos sob forma de planilhas, tabelas, etc.Desta forma, precisamos que o R leia estes arquivos, transformando-os em um objeto.
Para que o R reconheça o conjunto de dados do arquivo é necessário que as colunas sejam separadas. Caso isso não ocorra o R não conseguirá separar as colunas e emitirá uma mensagem de erro. Um modo fácil de resolver este problema é salvar a planilha de dados com o formato (.csv) que utiliza virgula (,) como elemento separador das colunas ou no formato (.txt).

Dados Externos

Antes de iniciar a entrada de dados no R deve-se alterar o diretório no R em que a pasta de trabalho padrão com o arquivo de dados (.csv ou .txt) está salvo (Arquivo - Mudar Diretório).
Para verificar qual é o diretório use **getwd()** .
Outra forma de mudar o diretório é por meio do comando:setwd("C:/Documents and Settings/Administrador/Meus documentos/temp1").
Em seguida, devemos dar o comando para que o R carregue o arquivo (.csv ou .txt) no console de trabalho da forma:
dados<-read.table("exemplo.txt",h=T) ou
dados<-read.table("exemplo.csv",h=T).

Dados Externos

No R é muito comum trabalharmos com conjuntos de dados da forma de *Data Frame*.Estes possuem linhas e colunas, em que cada coluna pode armazenar elementos de diferentes tipos.
Assim *data.frame* é um tipo especial de lista, composta por vetores de mesmo tamanho, mas que podem ser de classes diferentes:

Dados Externos

O R permite acessar um banco de dados disponível na *web*.Esta tarefa é importante,pois facilita o acesso aos dados provenientes da *internet* sem a necessidade dos dados serem copiados para algum diretório e, posteriormente, carregados para o R.
Para fazer a leitura usamos: read.table("http://endereço").

Exemplo:read.table("http://www.leg.ufpr.br/ paulo-jus/dados/exemplo02.txt").

Funções

Uma das grandes vantagens do R é poder criar (programar) novas funções. Para fazer uma função são necessários as seguintes atribuições:
nome(da função)<-function(argumento1,...,argumento*n*).

Funções

Exemplo:

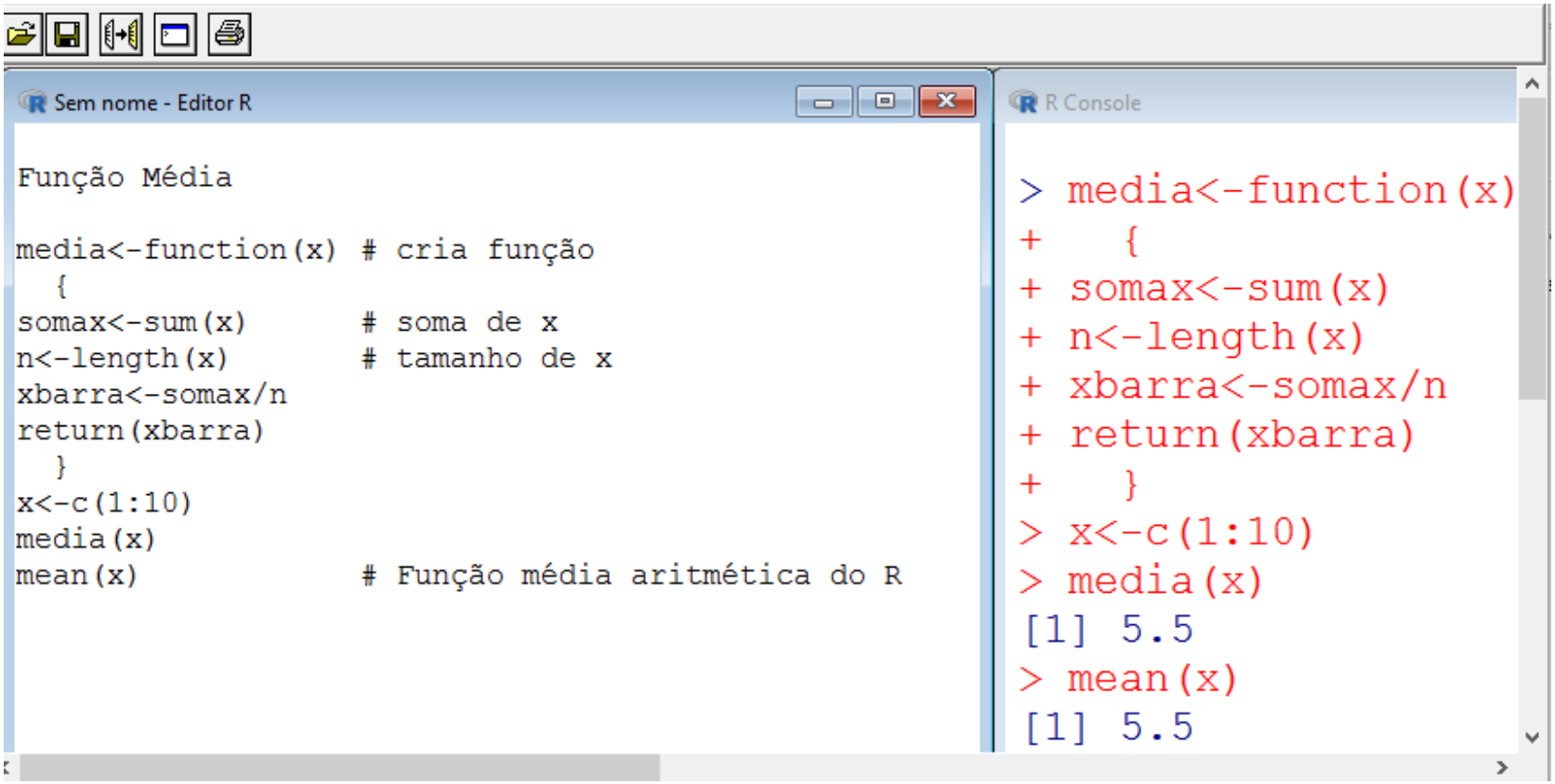


Figura: Função Média

Gráficos

Outra grande vantagem do R é a sua capacidade gráfica. Consegue plotar desde gráficos bidimensionais simples até gráficos tridimensionais mais complexos por meio de comandos bastante simples.
O comando básico é dado por meio da função plot ().

Gráficos

Exemplo:

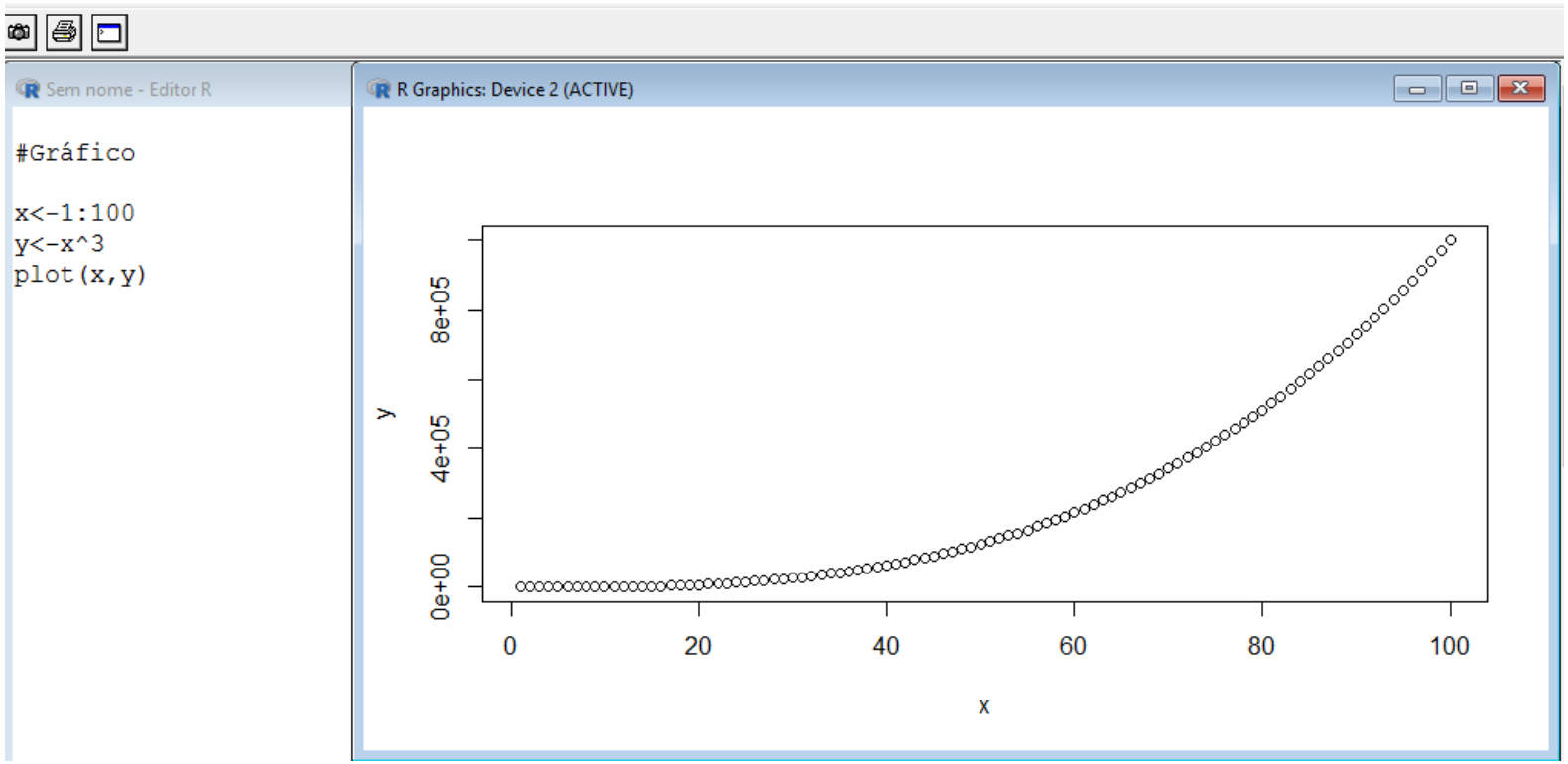


Figura: Exemplo de Gráfico Simples

Gráficos

Alguns comandos básicos para gráficos:

Tabela: Comandos básicos para gráficos

Argumentos Básicos	**Descrição**
pch( )	Muda o padrão dos pontos
lwd( )	Muda a largura das linhas
lty( )	Muda o estilo das linhas
main( )	Título do gráfico
xlab( )	Título do eixo x
ylab( )	Título do eixo y
text( )	Adiciona texto
points( )	Adiciona pontos
lines( )	Adiciona linhas

Gráficos

Exemplo:

Sem nome - Editor R

```
# Exemplo Gráfico

x<-1:100
y<-x^3
plot(x,y,xlab="Eixo x",ylab="Eixo y",
main="Aprendendo Gráficos")
lines(x,2*y,lwd=2,col="red")
points(x,x^4,pch=4)
text(80,100000,"Escrever um texto aqui")
```

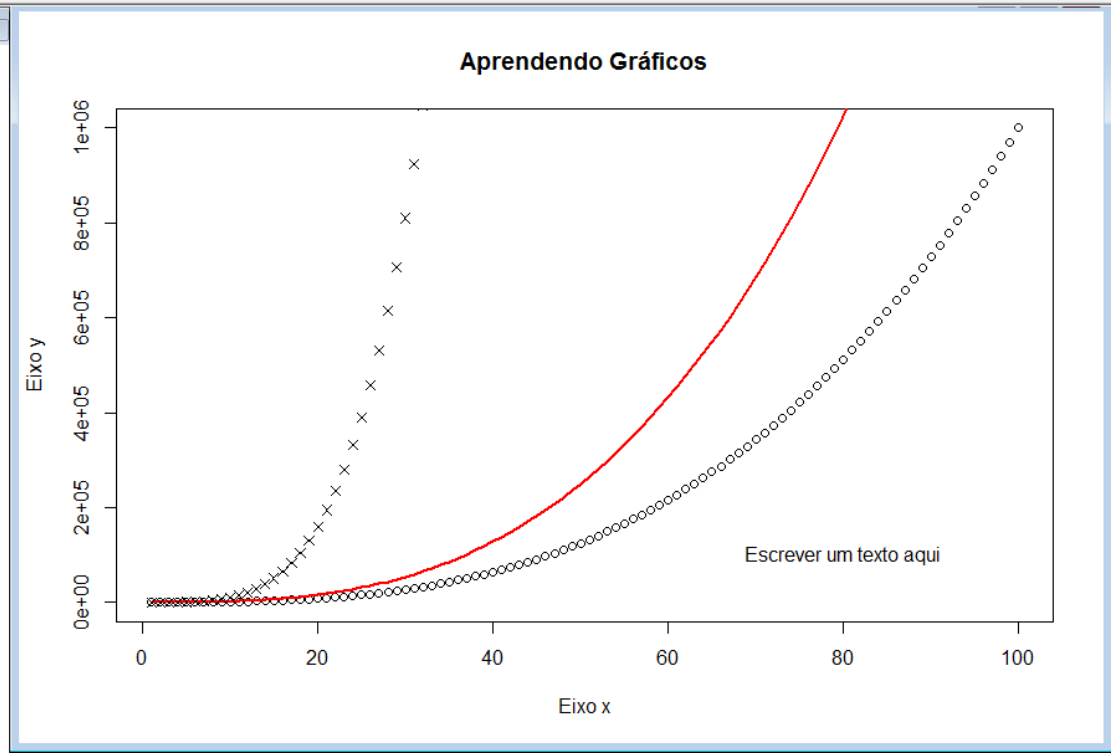


Figura: Exemplo de Gráfico

Ajuda do R (Help)

Em diversos casos precisamos de alguma função ou temos alguma dúvida em algum comando do R . Nestes casos existem duas formas básicas para descobrir uma função que faça aquilo que você deseja utilizando o *help* do R.
help(nome da função ou comando)ou
?(nome da função ou comando).

Ajuda do R (Help)

Geralmente, o arquivo de help do R possui 10 tópicos básicos:
1) *Description* - faz um resumo geral sobre o uso da função
2) *Usage* - mostra como a função pode ser utilizada e quais os argumentos que devem ser especificados
3) *Arguments* - explica o que é cada um dos argumentos
4) *Details* - explica alguns detalhes sobre o uso e aplicação da função
5) *Value* - mostra o que sai no *output* após usar a função (os resultados).

Ajuda do R (Help)

6) *Note* - algumas notas sobre a função
7) *Authors* - lista os autores da função (quem escreveu os códigos em R)
8) *References* - referências para os métodos utilizados
9) *See also* - mostra outras funções relacionadas que podem ser consultadas
10) *Examples* - exemplos do uso da função.

Instalação de Pacotes

Quando instalarmos o R apenas as configurações mínimas para seu funcionamento básico são instaladas (pacotes que vem na instalação "*base*"). Para realizar tarefas mais complicadas pode ser necessário instalar pacotes adicionais (*packages*).
Packages ou bibliotecas designam um conjunto de funções (comandos) e, ou, dados agregados.

Instalação de Pacotes

As funções básicas do R estão no pacote chamado *base*. Há outras bibliotecas já inclusas no R advindas da instalação padrão.
Os pacotes do R são desenvolvidos por usuários do próprio R.

Instalação de Pacotes

Para fazer a instalação de pacotes do R podemos fazer por meio do comando: Pacotes + Escolher espelho CRAN (por exemplo, Brazil,RJ) + Nome do Pacote)
Importante: Podemos fazer a instalação de pacotes também por meio *Install Package(s) from local zip files* (arquivo externo).

Carregando um Pacote

Carregando um pacote do R:

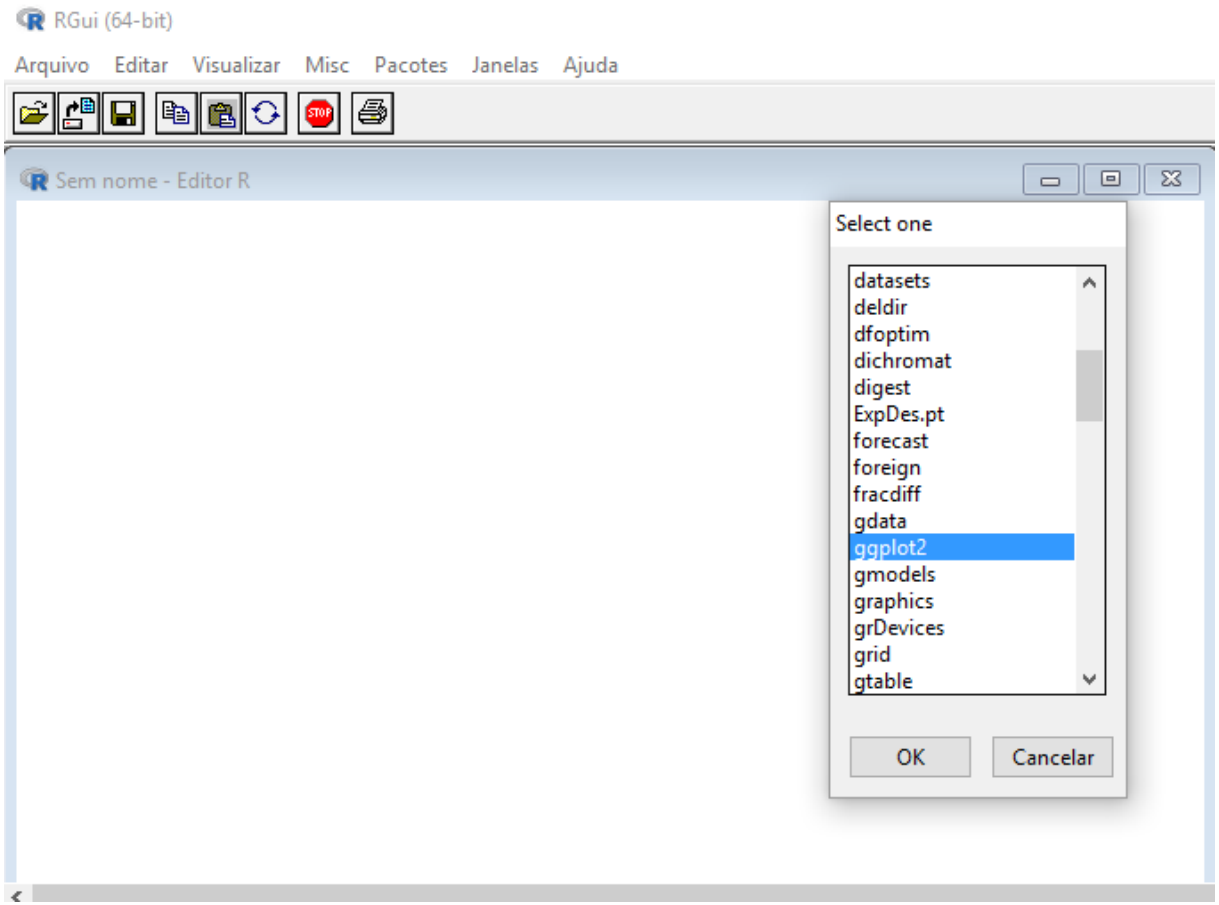


Figura: Carregando Pacote

Carregando um Pacote

Outra forma de carregar um pacote é por meio do comando *library* ou *require* + nome do pacote.
Exemplo:
library(MASS)
require(MASS).

Citação de Pacote

No R existe um comando que mostra como citar o R ou um de seus pacotes.
Exemplo:
citation("MASS")
citation()
R Development Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Alguns Pacotes para Otimização

- **NlcOptim**:Resolve problemas de otimização não-linear com restrições de desigualdade linear e não-linear da igualdade e, implementando um método sequencial de programação quadrática (SQP). Diponível em: **CRAN NlcOptim**.

Alguns Pacotes para Otimização

- **NlcOptim**:Resolve problemas de otimização não-linear com restrições de desigualdade linear e não-linear da igualdade e, implementando um método sequencial de programação quadrática (SQP). Diponível em: **CRAN NlcOptim**.
- **dfoptim**:Procedimentos de otimização livre de derivativos, contém implementações bastante eficientes R dos algoritmos de Nelder-Mead e Hooke-Jeeves.Diponível em: **CRAN dfoptim**.

Alguns Pacotes para Otimização

- **NlcOptim**:Resolve problemas de otimização não-linear com restrições de desigualdade linear e não-linear da igualdade e, implementando um método sequencial de programação quadrática (SQP). Diponível em: **CRAN NlcOptim**.
- **dfoptim**:Procedimentos de otimização livre de derivativos, contém implementações bastante eficientes R dos algoritmos de Nelder-Mead e Hooke-Jeeves.Diponível em: **CRAN dfoptim**.
- **maxLik**: Métodos de otimização de função de verossimilhança. Diponível em: **CRAN maxLik**.

Alguns Pacotes para Otimização

- **NlcOptim**:Resolve problemas de otimização não-linear com restrições de desigualdade linear e não-linear da igualdade e, implementando um método sequencial de programação quadrática (SQP). Diponível em: **CRAN NlcOptim**.
- **dfoptim**:Procedimentos de otimização livre de derivativos, contém implementações bastante eficientes R dos algoritmos de Nelder-Mead e Hooke-Jeeves.Diponível em: **CRAN dfoptim**.
- **maxLik**: Métodos de otimização de função de verossimilhança. Diponível em: **CRAN maxLik**.
- **limSolve**:Encontra o mínimo / máximo de uma função linear ou quadrática.Diponível em: **CRAN limSolve**.

Alguns Pacotes para Otimização

- **NMOF**:Fornece implementações de vários métodos de otimização. Diponível em: **CRAN NMOF**.

Alguns Pacotes para Otimização

- **NMOF**:Fornece implementações de vários métodos de otimização. Diponível em: **CRAN NMOF**.
- **RsoInp**: Utiliza o método do multiplicador de Lagrange aumentado.Diponível em: **CRAN RsoInp**.

Alguns Pacotes para Otimização

- **NMOF**:Fornece implementações de vários métodos de otimização. Diponível em: **CRAN NMOF**.
- **RsoInp**: Utiliza o método do multiplicador de Lagrange aumentado.Diponível em: **CRAN RsoInp**.
- **lpSolve**: Programação linear, inteira, problemas de atribuição e de transporte. Diponível em: **CRAN lpSolve**.

Alguns Pacotes para Otimização

- **NMOF**:Fornece implementações de vários métodos de otimização. Diponível em: **CRAN NMOF**.
- **RsoInp**: Utiliza o método do multiplicador de Lagrange aumentado.Diponível em: **CRAN RsoInp**.
- **lpSolve**: Programação linear, inteira, problemas de atribuição e de transporte. Diponível em: **CRAN lpSolve**.
- **nloptr**:Problemas de otimização não linear.Diponível em: **CRAN nloptr**.

Otimização em Finanças

Inúmeros problemas em Finanças envolvem encontrar mínimo e máximo de funções multidimensionais. Estes podem ser definidos como problemas de otimização.
No R existem três funções na instalação base para problemas de otimização: *optimize*( ) para problemas unidimensionais, *optim* ( ) para problemas multidimensionais, e *constrOptim*( ) para otimização com restrições lineares.

Função de Uma Variável

Exemplo 1: Otimizar a função $f(x) = xsen(4x)$.

- f = function(x) x*sin(4*x)

Função de Uma Variável

Exemplo 1: Otimizar a função $f(x) = xsen(4x)$.

- f = function(x) x*sin(4*x)
- curve(f,0,3)

Função de Uma Variável

Exemplo: Otimizar a função $f(x) = xsen(4x)$.

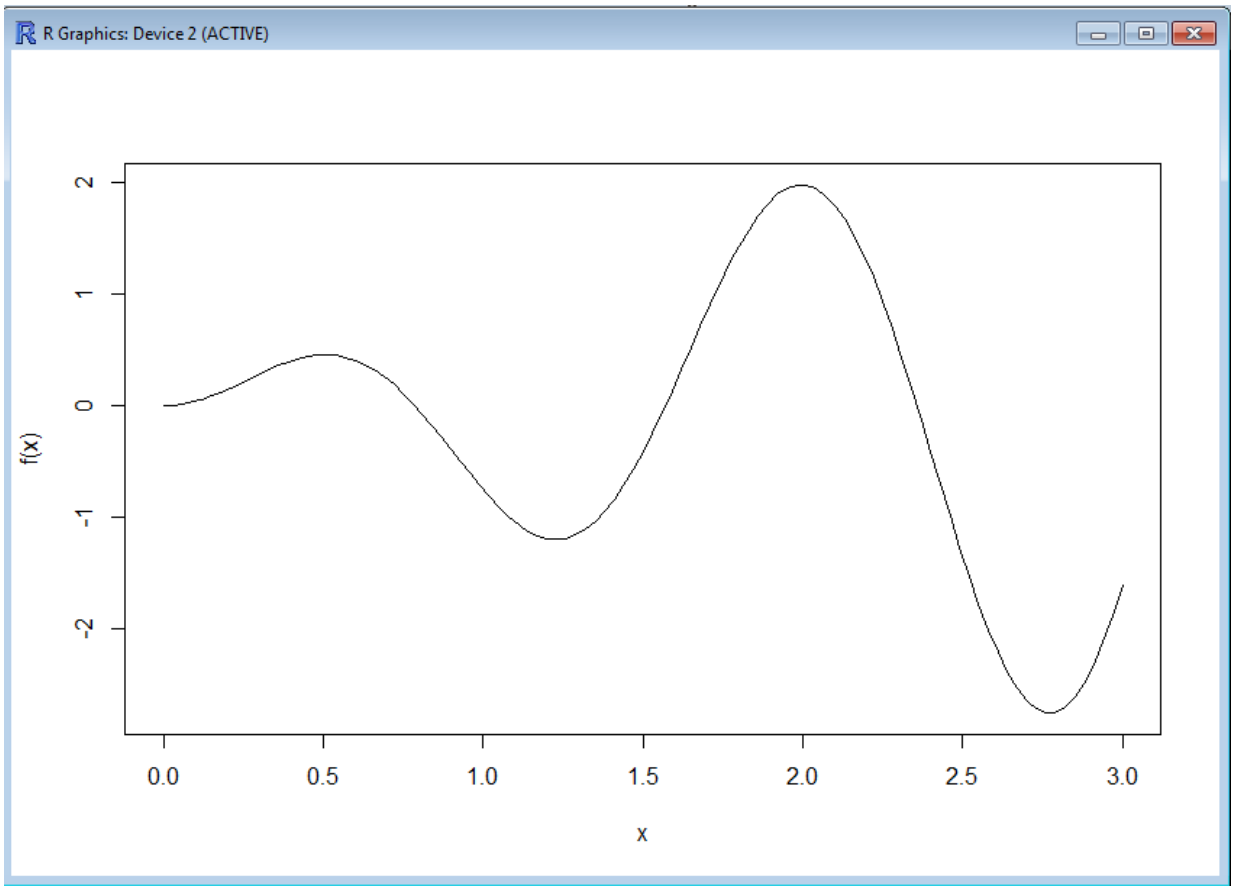


Figura: Gráfico função f(x)

Função de Uma Variável

Exemplo 1 : Otimizar a função $f(x) = xsen(4x)$.

- optimize(f,c(0,3))

Função de Uma Variável

Exemplo 1 : Otimizar a função $f(x) = xsen(4x)$.

- optimize(f,c(0,3))
- min1 = 1,228297

Função de Uma Variável

Exemplo 1 : Otimizar a função $f(x) = xsen(4x)$.

- optimize(f,c(0,3))
- min1 = 1,228297
- optimize(f,c(1.5,3))

Função de Uma Variável

Exemplo 1 : Otimizar a função $f(x) = xsen(4x)$.

- optimize(f,c(0,3))
- min1 = 1,228297
- optimize(f,c(1.5,3))
- min2 = 2,771403

Função de Uma Variável

Exemplo 1 : Otimizar a função $f(x) = xsen(4x)$.

- optimize(f,c(0,3))
- min1 = 1,228297
- optimize(f,c(1.5,3))
- min2 = 2,771403
- optimize(f,c(1,3),maximum=TRUE)

Função de Uma Variável

Exemplo 1: Otimizar a função $f(x) = xsen(4x)$.

- require(pracma)

Função de Uma Variável

Exemplo 1: Otimizar a função $f(x) = xsen(4x)$.

- require(pracma)
- f.mins = findmins(f,0,3)

Função de Uma Variável

Exemplo 1: Otimizar a função $f(x) = xsen(4x)$.

- require(pracma)
- f.mins = findmins(f,0,3)
- 1,228312 e 2,771382

Função de Uma Variável

Exemplo 1: Otimizar a função $f(x) = xsen(4x)$.

- require(pracma)
- f.mins = findmins(f,0,3)
- 1,228312 e 2,771382
- f(f.mins[1:2])

Função de Uma Variável

Exemplo 1: Otimizar a função $f(x) = xsen(4x)$.

- require(pracma)
- f.mins = findmins(f,0,3)
- 1,228312 e 2,771382
- f(f.mins[1:2])
- -1,203617 e -2,760177

Função de Várias Variáveis

Exemplo 2: Otimizar a função $f(x,y) = \frac{1}{x} + \frac{1}{y} + \frac{1-y}{y(1-x)} + \frac{1}{(1-x)(1-y)}$. (Hanna and Sandall, p. 191).

- x e y são frações molares (entre 0 e 1)

Função de Várias Variáveis

Exemplo 2: Otimizar a função $f(x,y) = \frac{1}{x} + \frac{1}{y} + \frac{1-y}{y(1-x)} + \frac{1}{(1-x)(1-y)}$. (Hanna and Sandall, p. 191).

- x e y são frações molares (entre 0 e 1)
- x = y = seq(0.1,0.9,0.01)

Função de Várias Variáveis

Exemplo 2: Otimizar a função $f(x,y) = \frac{1}{x} + \frac{1}{y} + \frac{1-y}{y(1-x)} + \frac{1}{(1-x)(1-y)}$. (Hanna and Sandall, p. 191).

- x e y são frações molares (entre 0 e 1)
- x = y = seq(0.1,0.9,0.01)
- z = outer(x,y,FUN=function(x,y) 1/x + 1/y +(1-y)/(y*(1-x1)) + 1/((1-x)*(1-y)))

Função de Várias Variáveis

Exemplo 2: Otimizar a função $f(x,y) = \frac{1}{x} + \frac{1}{y} + \frac{1-y}{y(1-x)} + \frac{1}{(1-x)(1-y)}$. (Hanna and Sandall, p. 191).

- x e y são frações molares (entre 0 e 1)
- x = y = seq(0.1,0.9,0.01)
- z = outer(x,y,FUN=function(x,y) 1/x + 1/y +(1-y)/(y*(1-x1)) + 1/((1-x)*(1-y)))
- persp(x,y,z,theta=45,phi=0,col="green")

Função de Várias Variáveis

Exemplo 2: Otimizar a função $f(x,y) = \frac{1}{x} + \frac{1}{y} + \frac{1-y}{y(1-x)} + \frac{1}{(1-x)(1-y)}$. (Hanna and Sandall, p. 191).

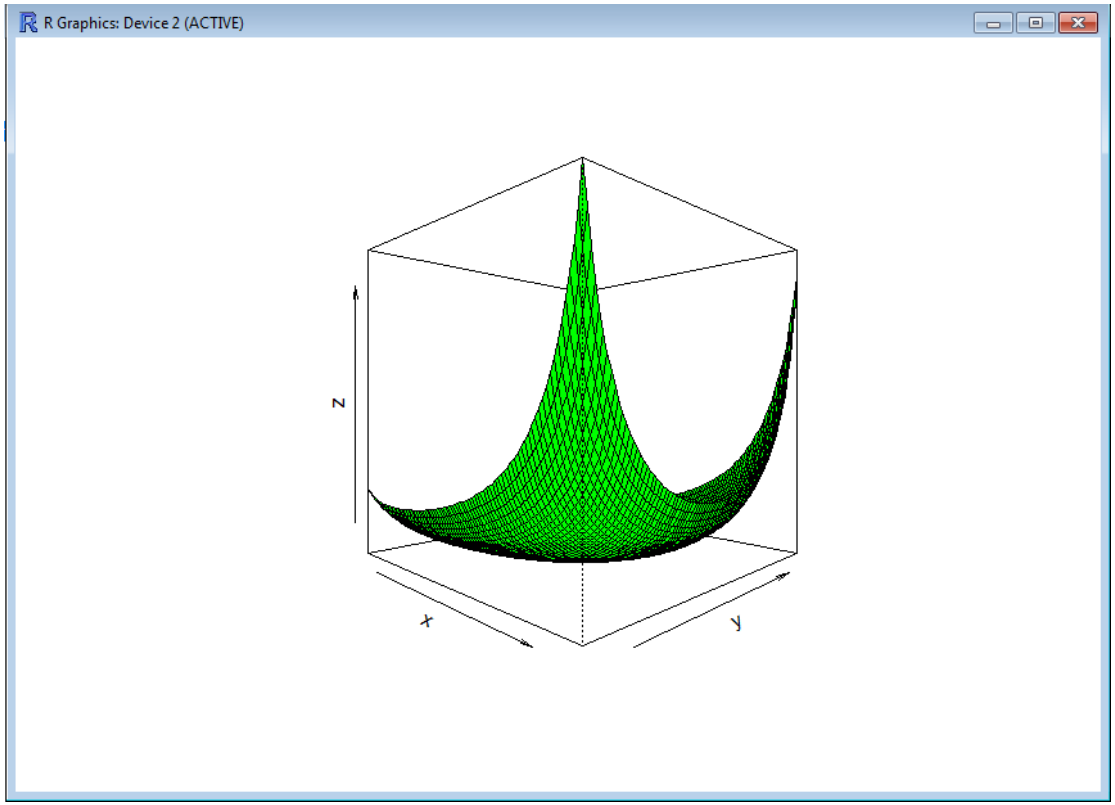


Figura: Gráfico função f(x,y)

Função de Várias Variáveis

Sempre que se puder razoavelmente assumir que a função objetivo é suave ou pelo menos diferenciável, pode-se utilizar o método "BFGS"ou "L-BFGS-B", aplicando "Nelder-Mead"apenas em outros casos, e apenas para tarefas de baixa dimensionalidade.
Todos estes são métodos da função *optim*. Se a função objetivo é não suave, nenhuma destas abordagens podem ser bem sucedidas.

Função de Várias Variáveis

optim(par, fn, gr = NULL, ..., method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent"),lower = -Inf, upper = Inf, control = list(), hessian = FALSE)

Função de Várias Variáveis

Exemplo 2: Otimizar a função $f(x,y) = \frac{1}{x} + \frac{1}{y} + \frac{1-y}{y(1-x)} + \frac{1}{(1-x)(1-y)}$.

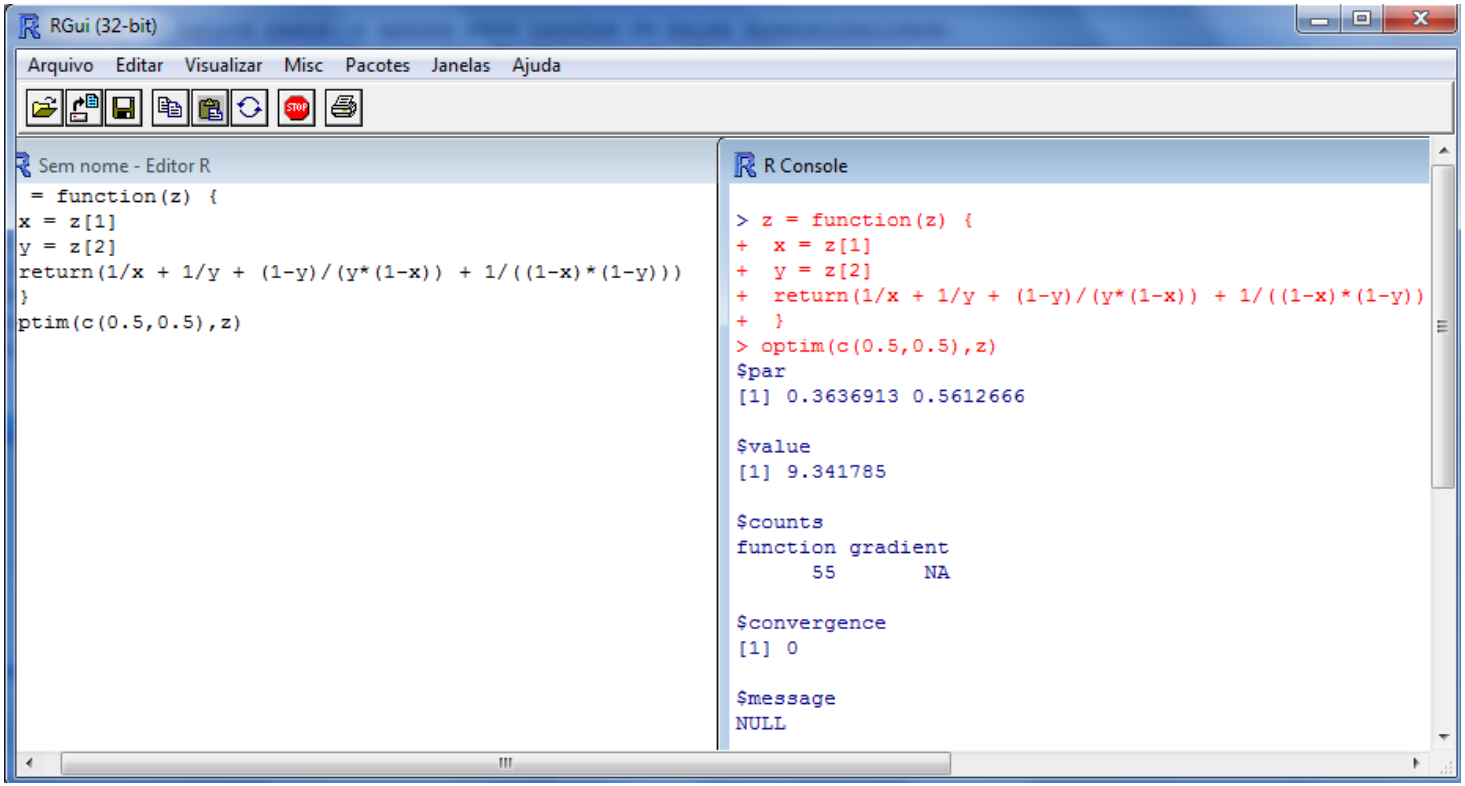


Figura: Otimização f(x,y)

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x, y) = 100\left(y - x^2\right)^2 + (1 - x)^2$ (Rosenbrock “Banana function").

- x = y = seq(-1.2,1,0.1)

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ (Rosenbrock “Banana function").

- x = y = seq(-1.2,1,0.1)
- z = outer(x,y,FUN=function(x,y) $100 * (y - x * x)^2 + (1 - x)^2$)

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ (Rosenbrock “Banana function").

- x = y = seq(-1.2,1,0.1)
- z = outer(x,y,FUN=function(x,y) $100 * (y - x * x)^2 + (1 - x)^2$)
- persp(x,y,z,theta=150,col="blue")

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ (Rosenbrock “Banana function").

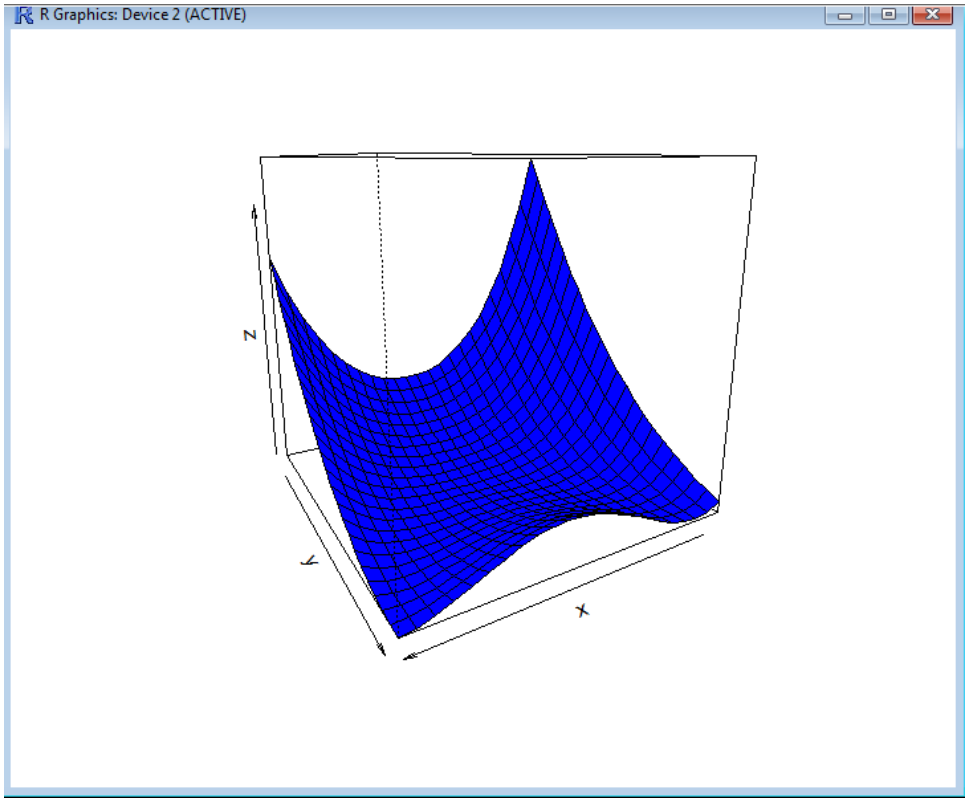


Figura: Otimização z(x,y)

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ (Rosenbrock “Banana function").

- optim(c(-1.2,1),z)

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ (Rosenbrock “Banana function").

- optim(c(-1.2,1),z)
- z = outer(x,y,FUN=function(x,y) $100 * (y - x * x)^2 + (1-x)^2$)

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ (Rosenbrock “Banana function").

- optim(c(-1.2,1),z)
- z = outer(x,y,FUN=function(x,y) $100 * (y - x * x)^2 + (1-x)^2$)
- persp(x,y,z,theta=150,col="blue")

Função de Várias Variáveis

Exemplo 3: Otimizar a função $z(x, y) = 100\left(y - x^2\right)^2 + (1 - x)^2$ (Rosenbrock “Banana function").

Sem nome - Editor R

```
z = function(z) { # Rosenbrock Banana function
x = z[1]
y = z[2]
 100*(y - x*x)^2 + (1 - x)^2
}

optim(c(-1.2,1),z)

optim(c(-1.2,1),z,"BFGS")

 gr = function(z) { ## Gradiente de z
 x = z[1]
 y = z[2]
 c(-400*x*(y-x*x)-2*(1-x),200 *(y - x*x))
 }

optim(c(-1.2,1),z,gr,method = "BFGS")
```

R Console

```
$value
[1] 9.594955e-18

$counts
function gradient
     110       43

$convergence
[1] 0

$message
NULL

> z = function(z) { # Rosenbrock Banana function
+ x = z[1]
+ y = z[2]
+  100*(y - x*x)^2 + (1 - x)^2
+ }
> optim(c(-1.2,1),z)
$par
[1] 1.000260 1.000506

$value
[1] 8.825241e-08

$counts
function gradient
     195       NA

$convergence
[1] 0

$message
NULL
```

Figura: Otimização z(x,y)

Otimização com restrições

Na base do R tem a fução *constrOptim* que permite minimizar uma função com *p* parâmetros desconhecidos sujeito a *k* restrições de desigualdade lineares. Ele usa a função *optim* para fazer a maior parte do cálculo, mas acrescenta uma barreira logarítmica para impor as restrições.

constrOptim(theta, f, grad, ui, ci, mu = 1e-04, control = list(),method = if(is.null(grad)) "Nelder-Mead"else "BFGS",outer.iterations = 100, outer.eps = 1e-05, ...,hessian = FALSE)

Otimização com restrições

Exemplo 4:Minimizar $z(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ sujeito a: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

Podendo ser rearranjado da forma: $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} -1 \\ -1 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Função de Várias Variáveis

Exemplo 4:Minimizar $z(x, y) = 100\left(y - x^2\right)^2 + (1 - x)^2$.

Sem nome - Editor R

```
fr = function(z) {
x = z[1]
y = z[2]
100*(y - x* x)^2 + (1 - x)^2
 }
gr <- function(z) { # gradiente
x = z[1]
y = z[2]
c(-400 * x * (y - x* x) - 2*(1 - x),
200*(y - x* x))
 }

constrOptim(c(-1.2,0.9), fr, gr, ui=rbind(c(-1,0),c(0,-1)),
ci=c(-1,-1))
```

R Console

```
+ x = z[1]
+ y = z[2]
+ 100*(y - x* x)^2 + (1 - x)^2
+  }
> gr <- function(z) { # gradiente
+ x = z[1]
+ y = z[2]
+ c(-400 * x * (y - x* x) - 2*(1 - x),
+ 200*(y - x* x))
+  }
>
> constrOptim(c(-1.2,0.9), fr, gr, ui=rbind(c(-1,0),c(0,-1)),
+ ci=c(-1,-1))
$par
[1] 0.9999761 0.9999521

$value
[1] 5.734117e-10

$counts
function gradient
     297       94

$convergence
[1] 0

$message
NULL

$outer.iterations
[1] 12

$barrier.value
[1] -0.0001999195

>
```

Figura: Otimização z(x,y)

Programação Linear

Há inúmeros pacotes no R que permitem trabalhar com a programação linear.
Exemplos de pacotes:**intpoint**, **lpSolve**,**Rglpk**,e **Rsymphony**.

Programação Linear

O pacote **intpoint** de programação linear utiliza o método do ponto interior e gráfico (duas dimensões)para resolver problemas.
Exemplo 1: max Z = $x_1 + 2x_2$
sujeito a:
$x_1 + x_2 = 1$
$x_2 = 4$
$x_1, x_2 \geq 0.$

Programação Linear

C:\Users\Felipe\Desktop\PH\carteira.R - Editor R

```
require(intpoint)

c=c(3,2)
M1=c(2,-1)
bM1=-2
m1=c(1,2)
bm1=8
solve2dlp(t=1, m=m1, bm=bm1, M=M1, bM=bM1, c=c, z=0, ip=0)
```

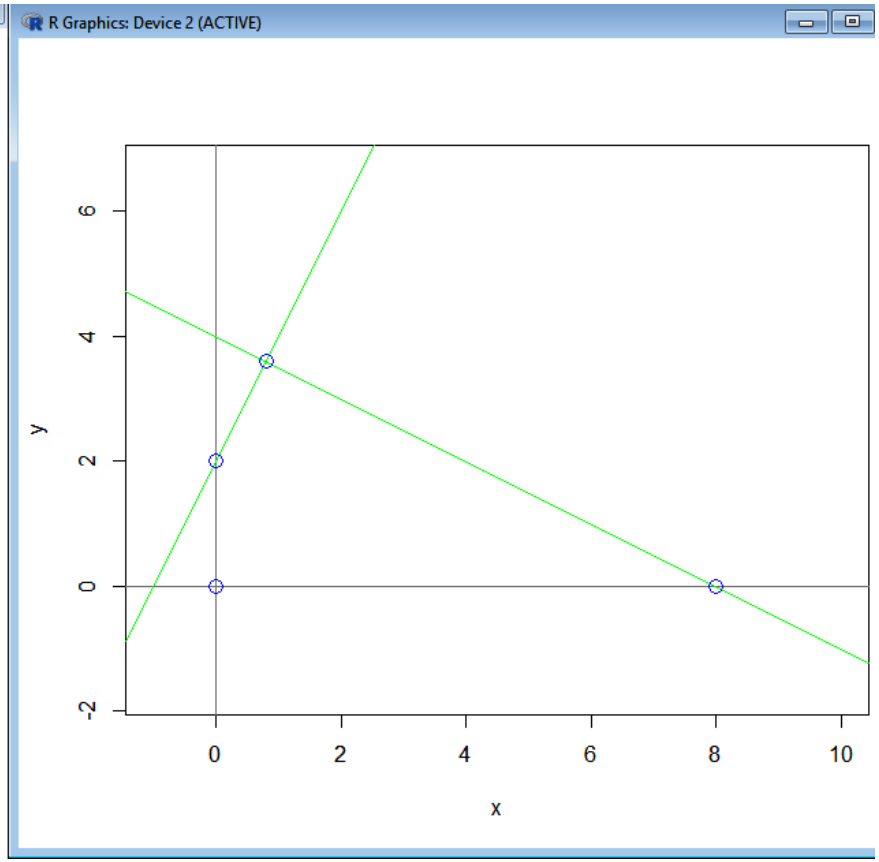


Figura: Método Gráfico

Programação Linear

Exemplo 2: max Z = $500m_1 + 400m_2$
sujeito a:
$20m_1 + 20m_2 \leq 100$
$5m_1 + 30m_2 \leq 50$
$15m_1 + 7m_2 \leq 60$
$m_1, m_2 \geq 0.$

Programação Linear

Utilizando o pacote **lpSolve** com a função **lp** temos que definir:

Programação Linear

Utilizando o pacote **lpSolve** com a função **lp** temos que definir:

- Função objetivo

Programação Linear

Utilizando o pacote **lpSolve** com a função **lp** temos que definir:

- Função objetivo
- Matriz de restrições

Programação Linear

Utilizando o pacote **lpSolve** com a função **lp** temos que definir:

- Função objetivo
- Matriz de restrições
- Tipos de desigualdades

Programação Linear

Utilizando o pacote **lpSolve** com a função **lp** temos que definir:

- Função objetivo
- Matriz de restrições
- Tipos de desigualdades
- Lado direito das desigualdades.

Programação Linear

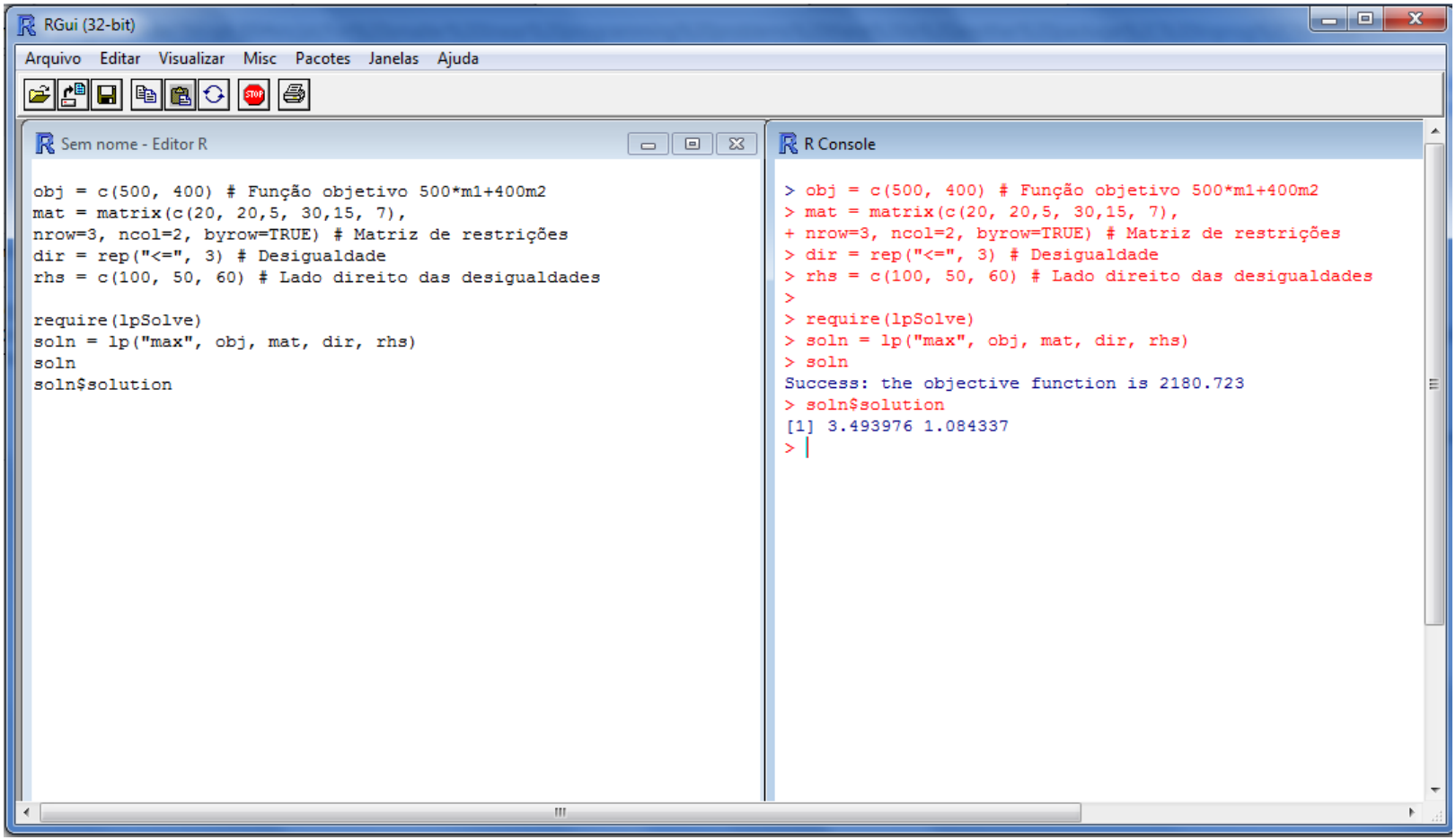


Figura: Programação linear no R

Programação Linear

Exemplo 3:
Uma empresa de investimentos gerencia recursos de terceiros por meio da escolha de carteiras de investimento para diversos clientes, baseados em *bonds* de diversas empresas. Um de seus clientes exige que:

- Não mais de 25% do total seja aplicado em um único investimento.

Programação Linear

Exemplo 3:
Uma empresa de investimentos gerencia recursos de terceiros por meio da escolha de carteiras de investimento para diversos clientes, baseados em *bonds* de diversas empresas. Um de seus clientes exige que:

- Não mais de 25% do total seja aplicado em um único investimento.
- Mais de 50% do total deve ser aplicado em títulos de maturidade de mais de 10 anos.

Programação Linear

Exemplo 3:
Uma empresa de investimentos gerencia recursos de terceiros por meio da escolha de carteiras de investimento para diversos clientes, baseados em *bonds* de diversas empresas. Um de seus clientes exige que:

- Não mais de 25% do total seja aplicado em um único investimento.
- Mais de 50% do total deve ser aplicado em títulos de maturidade de mais de 10 anos.
- O total aplicado em títulos de alto risco deve ser no máximo de 50% do total investido.

Programação Linear

Tabela: Títulos Selecionados

Títulos	**Retorno Anual**	**Vencimento (anos)**	**Risco**
1	8,7%	15	Muito baixo
2	9,5%	12	Regular
3	12,0%	8	Alto
4	9,0%	7	Baixo
5	13,0%	11	Alto
6	20,0%	5	Muito alto

Programação Linear

Variáveis de Decisão:

– P1 – Parcela do total aplicado no título do tipo 1
– P2 – Parcela do total aplicado no título do tipo 2
– P3 – Parcela do total aplicado no título do tipo 3
– P4 – Parcela do total aplicado no título do tipo 4
– P5 – Parcela do total aplicado no título do tipo 5
– P6 – Parcela do total aplicado no título do tipo 6

Programação Linear

- Função objetivo:

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento
  $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento
  $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$
- Restrição de aplicação

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento
  $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$
- Restrição de aplicação
  $P_{1,2,3,4,5,6} \leqslant 0,25$

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento
  $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$
- Restrição de aplicação
  $P_{1,2,3,4,5,6} \leqslant 0,25$
- Restrição para o mínimo aplicado

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento
  $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$
- Restrição de aplicação
  $P_{1,2,3,4,5,6} \leqslant 0,25$
- Restrição para o mínimo aplicado
  $P_1 + P_2 + P_5 \geqslant 0,5$

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento
  $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$
- Restrição de aplicação
  $P_{1,2,3,4,5,6} \leqslant 0,25$
- Restrição para o mínimo aplicado
  $P_1 + P_2 + P_5 \geqslant 0,5$
- Restrição para o máximo aplicado

Programação Linear

- Função objetivo:
  Max $0,087P_1 + 0,095P_2 + 0,12P_3 + 0,09P_4 + 0,13P_5 + 0,2P_6$
- Restrição de orçamento
  $P_1 + P_2 + P_3 + P_4 + P_5 + P_6 = 1$
- Restrição de aplicação
  $P_{1,2,3,4,5,6} \leqslant 0,25$
- Restrição para o mínimo aplicado
  $P_1 + P_2 + P_5 \geqslant 0,5$
- Restrição para o máximo aplicado
  $P_3 + P_5 + P_6 \leqslant 0,5$

Programação Linear

Sem nome - Editor R

```
require(lpSolve)
obj = c(0.087,0.095,0.12,0.09,0.13,0.2)
mat = matrix(c(1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,0,1,1,1,0,0,0,0,0,1,0,0,0,0,
0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1),nrow = 9, ncol = 6,byrow=T)
mat
dir = c("=",">=",rep("<=",7))
rhs = c(1,0.5,0.5,rep(0.25,6))

ret = lp("max",obj,mat,dir,rhs)
ret
ret$solution
```

R Console

```
> require(lpSolve)
> obj = c(0.087,0.095,0.12,0.09,0.13,0.2)
> mat = matrix(c(1,1,1,1,1,1,1,1,0,0,1,0,0,0,1,0,1,1,1,0,0,0
+ 0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1),nrow = 9,
> mat
      [,1] [,2] [,3] [,4] [,5] [,6]
 [1,]    1    1    1    1    1    1
 [2,]    1    1    0    0    1    0
 [3,]    0    0    1    0    1    1
 [4,]    1    0    0    0    0    0
 [5,]    0    1    0    0    0    0
 [6,]    0    0    1    0    0    0
 [7,]    0    0    0    1    0    0
 [8,]    0    0    0    0    1    0
 [9,]    0    0    0    0    0    1
> dir = c("=",">=",rep("<=",7))
> rhs = c(1,0.5,0.5,rep(0.25,6))
>
> ret = lp("max",obj,mat,dir,rhs)
> ret
Success: the objective function is 0.12875
> ret$solution
[1] 0.00 0.25 0.00 0.25 0.25 0.25
```

Figura: Retorno da carteira

Programação Inteira

Um problema de Programação Inteira é um modelo de programação linear no qual algumas ou todas as variáveis do problema pertencem ao conjunto dos números inteiros.
Exemplo:Considere o problema da mochila que pode ser resolvido usando a programação inteira. Seja um vendedor ambulante quer encher a mochila de capacidade de peso 105 com itens que têm os seguintes valores e pesos, de modo a maximizar o seu lucro:

Programação Inteira

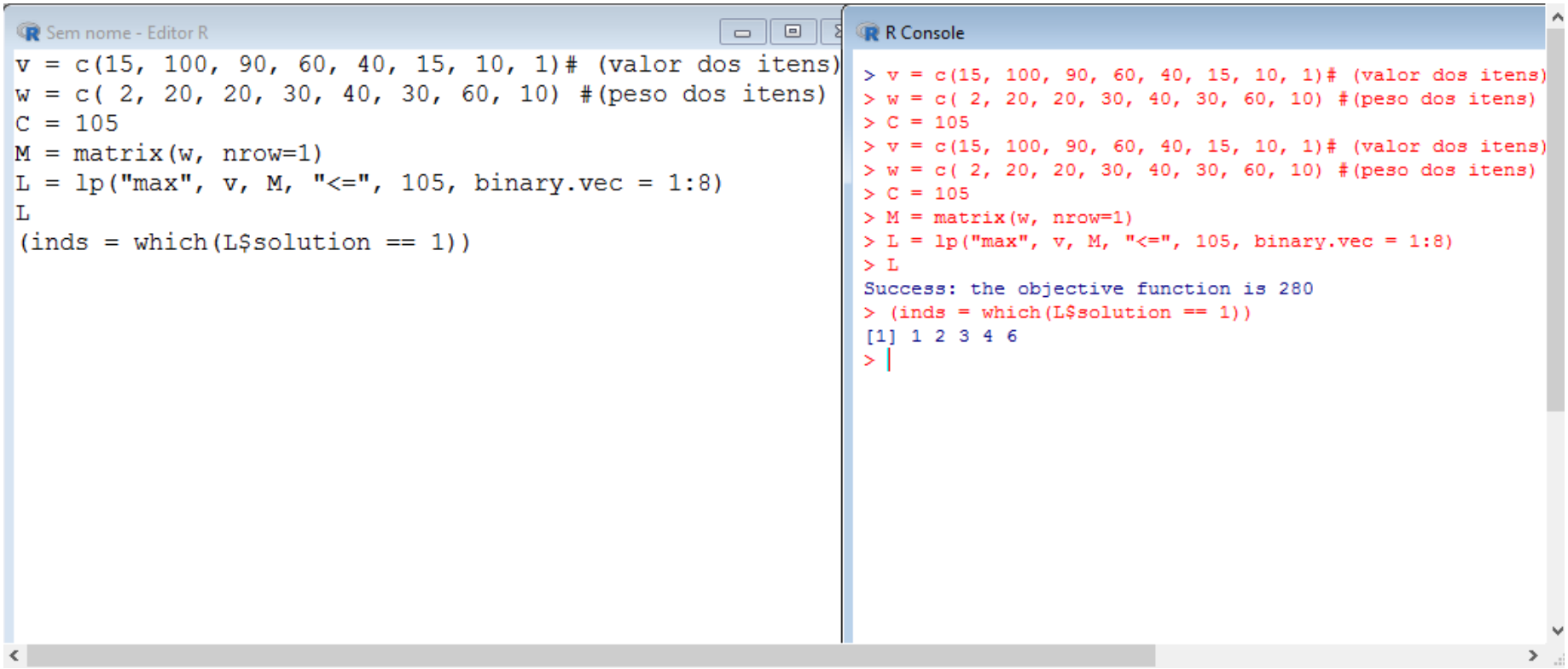


Figura: Programação inteira no R

Programação Quadrática

A programação quadrática visa otimizar uma função quadrática de várias variáveis sujeitas a restrições lineares sobre essas. Isto é, desejamos minimizar a função:

$$f(x) = -d^T x + \frac{1}{2} x^T D x$$
$$\text{sujeito a: } A^T x \geq x_0,$$

em que D é uma matriz de coeficientes quadráticos, d um vector de coeficientes linear, A uma matriz de restrições, e x_0 um vetor de valores de restrição.

Programação Quadrática

Exemplo 4: f(x) = $f(x) = \frac{1}{2}x_1^2 + x_2^2 - x_1x_2 - 2x_1 - 6x_2$
sujeito a: $x_1 + x_2 \leqslant 2$
$-x_1 + 2x_2 \leqslant 2$
$2x_1 + x_2 \leqslant 3$
$0 \leq x_1, 0 \leq x_2.$

Programação Quadrática

No R podemos utilizar o pacote **quadprog**.

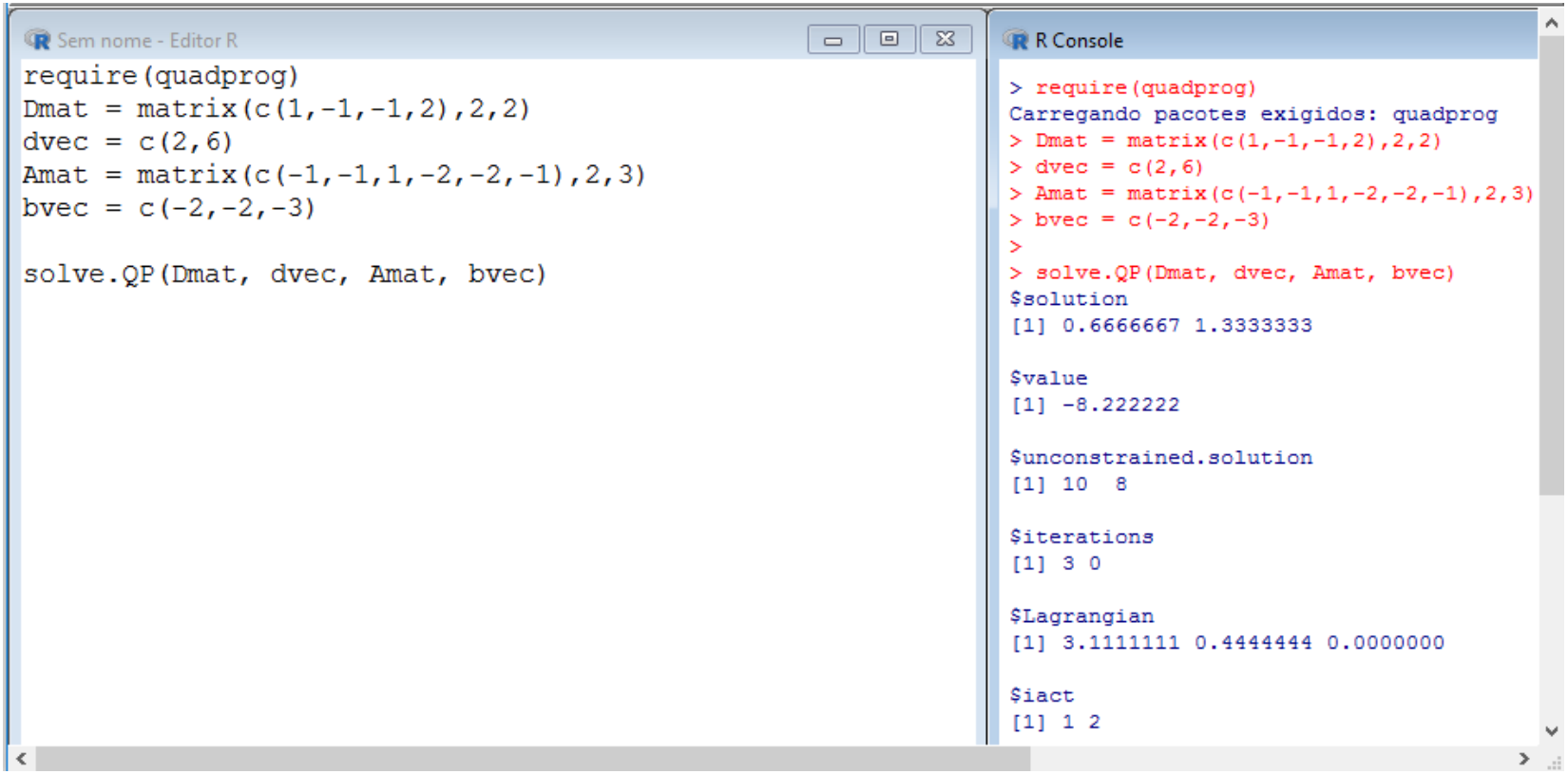


Figura: Programação quadrática no R

Otimização de Portfólio (Carteiras)

A teoria moderna do portfólio, ou simplesmente teoria do portfólio, busca entender como investidores racionais irão usar o princípio da diversificação para otimizar as suas carteiras de investimentos, e como um ativo arriscado deve ser precificado.
O trabalho pioneiro na área de otimização de portfólio foi à proposição do modelo média-variância por Markowitz (1952).A teoria do portfólio estabelece que decisões relacionadas à seleção de investimentos devam ser tomadas com base na relação risco-retorno.

Otimização de Portfólio (Carteiras)

De acordo com o modelo Markowitz (1952):
1 - O retorno da carteira é a combinação ponderada da proporção de retorno dos ativos que a constituem.
2 - A volatilidade da carteira é uma função da correlação ρ dos ativos componentes. A alteração na volatilidade é não-linear com a mudanças na ponderação dos ativos componentes.
A principal motivação para o desenvolvimento destes modelos se relaciona à redução do risco a que o investidor está exposto, por meio da diversificação ou balanceamento da carteira.

Otimização de Portfólio (Carteiras)

Se formos variando o nível de retorno desejado, e a cada vez resolvendo o problema de minimização da variância do portfólio, será gerada uma fronteira que possui, para cada nível de retorno, o portfólio com a menor variância.A parte superior desta fronteira é chamada fronteira eficiente. A próxima figura traz um exemplo de uma fronteira eficiente composta por sete ativos financeiros.

Otimização de Portfólio (Carteiras)

Para construir este gráfico ver **aqui**.

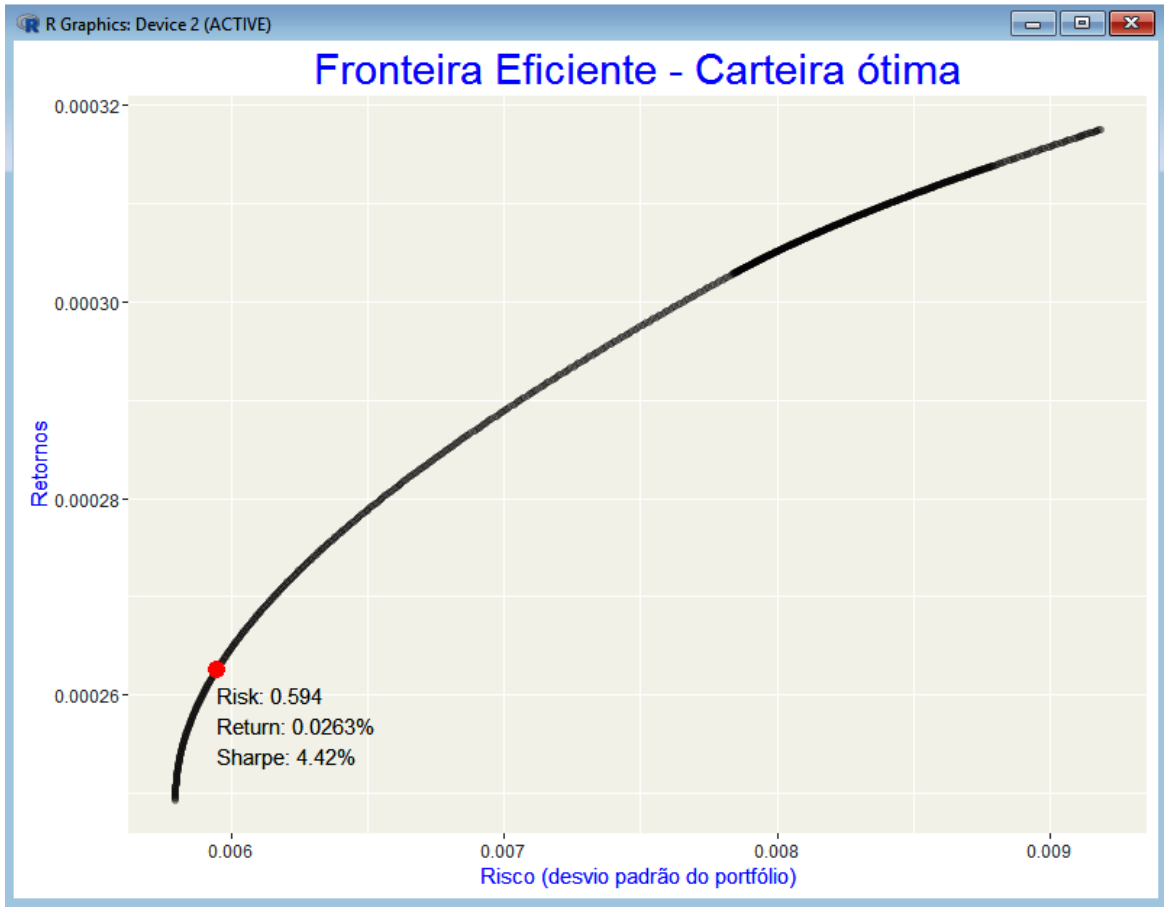


Figura: Fronteira Eficiente

Referências

BLOOMFIELD(2014)

Using R for Numerical Analysis in Science and Engineering,University of Minnesota Minneapolis, USA.

CORNUEJOLS,G.,*TÜTÜNCÜ*,R.(2006)

Optimization Methods in Finance,Carnegie Mellon University, Pittsburgh, USA.

LACHTERMACHER(2006)

Pesquisa Operacional na Tomada de Decisões,Rio de Janeiro:Campus.

MARKOWITZ(1952)

Portfolio selection. The Journal of Finance, v. 7, n. 1, p. 77-91, 1952.