

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAFFAEL DA SILVA NAGEL

**Comparação de Alternativas de  
Implementação de Interfaces de Usuário  
para Aplicações em R**

Monografia apresentada como requisito  
parcial para a obtenção do grau de Bacharel  
em Ciência da Computação

Orientador: Prof. Dr. Ingrid Oliveira de  
Nunes

Porto Alegre  
2015

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Vice-Reitor: Prof. Rui Vicente Oppermann

Pró-Reitor de Graduação: Prof. Sérgio Roberto Kieling Franco

Diretor do Instituto de Informática: Prof. Luis da Cunha Lamb

Coordenador do Curso de Ciência de Computação: Prof. Raul Fernando Weber

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

*“..you’ve got to ask yourself one question:  
'Do I feel lucky?' Well, do ya, punk?”*

— CLINT EASTWOOD

## RESUMO

A estatística vem, cada vez mais, sendo utilizada por corporações e empresas como chave nas decisões estratégicas. Através de cálculos e testes é possível observar, acompanhar e até prever comportamentos referentes ao negócio. É neste cenário que a linguagem de programação R e seu ambiente de desenvolvimento tem se destacado. Por se tratar de uma linguagem com foco na aplicação de modelos estatísticos, o R traz um horizonte de possibilidades no que se refere a análise de dados, cálculos, modelagem e criação de gráficos.

A grande problemática do uso de R, e foco deste trabalho, é a criação de interfaces gráficas de usuário. Por ser R uma linguagem baseada em scripts, na qual linhas de comando são inseridas em um ambiente textual, a interação com usuários é bastante comprometida.

Neste trabalho são apresentadas duas formas de criação de sistemas estatísticos que se utilizam de R e apresentam interface gráfica. Através da criação de uma aplicação real são apresentadas as duas possíveis abordagens, a primeira utilizando apenas a linguagem R, e a segunda utilizando a linguagem de programação Java para definição da interface e fazendo uso de chamadas externas para execução de código em R. A partir destas implementações, as duas abordagens são comparadas quantitativa e qualitativamente e, a partir desta comparação, são identificadas suas respectivas vantagens, desvantagens, dificuldades e oportunidades.

**Palavras-chave:** R. JAVA. INTERFACE GRÁFICA DE USUARIO. DESENVOLVIMENTO DE SOFTWARE.

## **Comparison of User Interfaces Implementation Alternatives for Applications in R**

### **ABSTRACT**

Statistics comes increasingly being used by corporations and businesses as a key for strategic decisions. With statistical calculations and tests, it is possible to observe, monitor and even predict behaviors related to business. In this context, the R programming language and development environment have been receiving much attention. Because it is a language with a focus on the application of statistical models, R brings a horizon of possibilities with regard to data analysis, calculations, modeling and charting.

The major problem of the use of R, which this work addresses is the creation of graphical user interfaces. Because R is a language based on scripts, in which command lines are inserted in a textual environment, the interaction with users is not trivial.

In this work the two ways of creating statistical systems that use R and feature graphical interface are presented. Using a real application, the two approaches are presented, the first using only the R language, and the second using the Java language to the interface development and making use of external calls to R code. Using these implementations, the two approaches are compared quantitatively and qualitatively and, based on this comparison, their respective advantages, disadvantages, difficulties and opportunities are identified.

**Keywords:** SOFTWARE DEVELOPMENT, R, GRAPHICAL USER INTERFACE, JAVA.

## LISTA DE FIGURAS

Figura 3.1	Exemplo básico de interface em R. ....	16
Figura 3.2	Exemplo de interface em R com botões. ....	17
Figura 3.3	Ciclo de abordagem utilizando espera em variável de controle. ....	20
Figura 4.1	Interface criada em R. ....	25
Figura 4.2	Prompt de comandos que acompanha a execução da interface em R. ....	26
Figura 4.3	Interface desenvolvida em Java. ....	29
Figura 4.4	Diagrama de classes da aplicação desenvolvida em Java. ....	29
Figura 5.1	Gráfico de diferenças entre tempos de execução. ....	34
Figura A.1	Relatório - Sistema de Diagnóstico de Linearidade ....	40
Figura A.2	Relatório - Sistema de Diagnóstico de Linearidade ....	41
Figura A.3	Relatório - Sistema de Diagnóstico de Linearidade ....	42
Figura A.4	Relatório - Sistema de Diagnóstico de Linearidade ....	43

## LISTA DE TABELAS

Tabela 5.1 Métricas de software.....	31
Tabela 5.2 Medidas de processamento.....	32
Tabela 5.3 Medidas de memória .....	32
Tabela 5.4 Tabela Descritiva .....	33
Tabela 5.5 Teste de Shapiro-Wilk .....	33
Tabela 5.6 Vantagens e desvantagens.....	36

## **LISTA DE ABREVIATURAS E SIGLAS**

GUI	Graphical User Interface
CLI	Command-line Interface
API	Application Programming Interface

## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	<b>10</b>
<b>1.1 Problema</b> .....	<b>11</b>
<b>1.2 Solução Proposta</b> .....	<b>11</b>
<b>1.3 Estrutura do documento</b> .....	<b>12</b>
<b>2 A LINGUAGEM R</b> .....	<b>13</b>
<b>3 INTERFACES EM LINGUAGEM R</b> .....	<b>15</b>
<b>3.1 Criando interfaces utilizando apenas linguagem R</b> .....	<b>16</b>
3.1.1 Interação .....	18
3.1.2 Chamadas de Funções em R .....	20
<b>3.2 Criando interfaces utilizando Java com chamadas externas à linguagem R</b> .....	<b>21</b>
<b>3.3 Considerações Finais</b> .....	<b>22</b>
<b>4 ESTUDO DE CASO</b> .....	<b>23</b>
<b>4.1 Requisitos do sistema de Diagnóstico de Linearidade</b> .....	<b>23</b>
<b>4.2 Implementação utilizando apenas linguagem R</b> .....	<b>24</b>
<b>4.3 Implementação utilizando interface em linguagem Java</b> .....	<b>27</b>
<b>5 DISCUSSÃO</b> .....	<b>30</b>
<b>5.1 Análise Quantitativa</b> .....	<b>30</b>
<b>5.2 Análise Qualitativa</b> .....	<b>33</b>
<b>5.3 Limitações do Trabalho</b> .....	<b>36</b>
<b>5.4 Considerações Finais</b> .....	<b>36</b>
<b>6 CONCLUSÃO</b> .....	<b>37</b>
<b>REFERÊNCIAS</b> .....	<b>38</b>
<b>A SISTEMA DE DIAGNÓSTICO DE LINEARIDADE</b> .....	<b>40</b>

## 1 INTRODUÇÃO

O mundo moderno vem sendo objeto de profundas e aceleradas transformações econômicas, tecnológicas, políticas e sociais, o que leva os ambientes que rodeiam as tomadas de decisões empresariais a ficarem cada dia mais exigentes. É neste contexto que cresce a importância da Estatística, a ciência exata que fornece métodos para a coleta, organização, descrição, análise e interpretação de dados visando a tomada de decisão. Pois, sendo a empresa uma pequena startup ou uma gigante corporação, o correto entendimento dos dados provenientes de seu meio de atuação tem a mesma importância vital ao negócio, principalmente com o surgimento de grandes bases de dados e os conceitos de “Big Data” (WU et al., 2013).

Com o crescimento e difusão da estatística temos, conseqüentemente, um aumento da importância das linguagens voltadas a aplicação de métodos estatísticos. Entre tais linguagens encontra-se a linguagem de programação R (IHAKA; GENTLEMAN, 1996), que por ser a única com código aberto de suas pares (SAS (DELWICHE; SLAUGHTER, 2012) e SPSS (FIELD, 2013)) tem se destacado e apresentado grande aceitação.

A linguagem R e seu ambiente de desenvolvimento fornecem uma ampla variedade de aplicações estatísticas, tais como modelagem linear e não-linear, testes estatísticos clássicos, análises de séries temporais, etc., além de técnicas gráficas, as quais são um dos pontos fortes desta linguagem, de forma a possibilitar a criação de gráficos complexos de maneira bastante simples. Porém, o grande diferencial desta linguagem de programação está na liberdade de criação e distribuição de bibliotecas de funções, chamadas “pacotes”. Pacotes escritos para a linguagem R acrescentam algoritmos avançados, gráficos coloridos e texturizados e técnicas de mineração para vasculhar bancos de dados mais a fundo, e qualquer usuário pode melhorar ou criar um novo pacote e distribuí-lo globalmente. Aproximadamente 1,6 mil pacotes estão disponíveis em apenas um dos muitos repositórios dedicados à R.

Dada esta motivação para linguagens de programação voltadas à estatística, em particular a linguagem R, nas próximas seções serão apresentados o problema e a solução propostos neste trabalho. Na seção 1.1, temos a descrição do problema, para o qual é apresentada uma proposta de solução na seção 1.2. Na seção 1.3 descreve-se a estrutura do documento. Adiante neste trabalho são apresentadas duas possíveis alternativas de implementação e uma comparação, quantitativa e qualitativa, destas.

## 1.1 Problema

Apesar das inúmeras qualidades da linguagem R, esta não foi criada para ser utilizada por usuários que não possuem conhecimento em estatística e programação. O perfil principal dos usuários de R engloba na sua maioria estatísticos e programadores, os quais utilizam seus conhecimentos para criar funções e algoritmos através de um ambiente textual semelhante ao *prompt* de comando. Assim temos o problema de o poder de uma linguagem bastante robusta ficar concentrado nas mãos de poucos, o que acaba por criar um gargalo no uso efetivo da linguagem para a tomada de decisões.

Com objetivo de solucionar o problema de difusão da linguagem R são desenvolvidas interfaces gráficas de usuário. Assim mesmo que não tenham conhecimentos estatísticos ou de programação os usuários podem obter as vantagens do uso da linguagem. Tais interfaces são criadas por desenvolvedores experientes que, por estarem familiarizados com linguagens de programação de propósito geral (e.g. Java, C++), as implementam nestas linguagens. Tal abordagem força o uso de R a ser realizado de forma indireta, através de chamadas externas. Entretanto, existe na própria linguagem R um conjunto de ferramentas que permite a implementação destas GUIs sem a necessidade de utilização de qualquer outra linguagem.

Dadas estas duas alternativas de implementação, resta saber se entre elas existe distinção, ou seja, se uma pode ser considerada a mais adequada, ou se a escolha depende apenas da preferência do programador.

## 1.2 Solução Proposta

A fim de responder tal questionamento, este estudo visa realizar uma análise quantitativa e qualitativa do ponto de vista da engenharia de software, para verificar qual apresenta melhor qualidade, complexidade, desempenho, escalabilidade, portabilidade ou outras características que podem vir a diferenciá-las.

Através da aplicação prática de cada um das abordagens, e com base em conceitos da engenharia de software, busca-se obter dados relevantes às análises propostas de forma que se possa chegar a um conclusão. Para a construção do estudo de caso mencionado é utilizado um sistema real, com requisitos que compreendem, entre outros, a necessidade de aplicação de métodos estatísticos e a utilização de uma interface gráfica amigável aos usuários.

### **1.3 Estrutura do documento**

Este trabalho está organizado de forma a inicialmente apresentar, no capítulo 2, uma introdução à linguagem R, suas aplicações, sua evolução e o estado atual de suas interfaces gráficas. Passando então, no capítulo 3, à explanação sobre as duas alternativas para implementar interfaces, onde são descritas a implementação utilizando somente linguagem R e a implementação utilizando interfaces em Java que rodam código em R através de chamadas externas.

Ambas as abordagens serão apresentadas no capítulo 4, através da criação de um sistema real, cujos requisitos também serão definidos no mesmo capítulo. Um estudo comparativo (qualitativo e quantitativos), realizado a partir do ponto de vista da engenharia de software é apresentado no capítulo 5. Neste capítulo, que consiste na principal contribuição deste trabalho, são expostas, com o auxílio de métricas de software, as vantagens e desvantagens da utilização de cada uma das abordagens.

Também, está presente neste trabalho um capítulo de conclusão 6, no qual serão descritos os aprendizados obtidos no decorrer da elaboração do mesmo, juntamente com os benefícios e o legado que este trabalho traz à comunidade de desenvolvimento em linguagem R.

## 2 A LINGUAGEM R

Baseada na premiada linguagem de computação estatística S, desenvolvida por John Chambers na Bell Laboratories (BECKER; CHAMBERS, 1984), a linguagem R foi desenvolvida pelos professores da universidade de Auckland – Nova Zelândia, Ross Ihaka e Robert Gentleman em 1993 e publicada em 1996 no jornal de estatística computacional e gráfica (IHAKA; GENTLEMAN, 1996). O nome ‘R’ surgiu como forma de reconhecimento da importância e influência de sua predecessora ‘S’, além de ser a letra inicial do nome de ambos os criadores.

A ideia de desenvolver uma nova linguagem surgiu em uma conversa informal, ambos, Ihaka e Gentleman, tinham a intenção de desenvolver uma ferramenta que tornaria mais simples a tarefa de ensinar aos seus alunos dos cursos introdutórios de estatística os conceitos de análise de dados e modelos gráficos. Além disso, ambos acreditavam serem capazes de desenvolver um ambiente similar ao utilizado pela linguagem S, mas que não sofresse dos problemas de memória e performance presentes nesta linguagem. Os desenvolvedores buscaram, então, combinar duas linguagens conhecidas da época: S e Scheme (SUSSMAN; JR., 1975), acreditando que da união das forças de cada uma delas surgiria uma nova linguagem com vantagens em portabilidade, eficiência computacional e gerenciamento de memória. O resultado foi uma linguagem bastante semelhante à S mas, internamente, com toda implementação e semântica derivadas de Scheme.

Já em 1995, após notícias de seu desenvolvimento espalharem-se e com sua crescente popularização, os autores Ihaka e Gentleman foram convencidos a fazer de R uma linguagem de código aberto, disponível sob os termos de licença pública geral da fundação de software livre. Ao ser questionado sobre a decisão de abrir o código de sua linguagem à toda comunidade de desenvolvedores, Mr. Ihaka disse: “Nós poderíamos ter optado pela comercialização, e teríamos vendido cinco cópias”.

Hoje, mais de 10 anos após sua criação, R é uma das mais poderosas linguagens de análise e modelagem de dados estatísticos. É utilizada por dezenas de milhares de pessoas diariamente e já recebeu elogios e aplausos dos mais renomados meios de comunicação, tais como *The New York Times* (VANCE, 2009), *Forbes* (MCNALLY, 2010), *infoWord* (WAYNER, 2010), *The Register* (MORGAN, 2012) e *Intelligent Enterprise* (STODDER, 2010), sendo diretamente utilizada pelo primeiro citado para a criação de seus infográficos. Além desta, outras gigantes corporações têm a linguagem como parte de suas ferramentas, entre elas o Google (BRODERSEN, 2014), a Ford (HINER, 2012) e o Facebook (AMIRTHA, 2014).

A explicação para o crescimento do uso de R está no aumento do valor associado aos dados e

na busca pelo desenvolvimento de métodos mais rápidos e poderosos para análise de conjuntos complexos de informações. Aliado a isso, a linguagem possui uma habilidade única de mudar, se transformar e evoluir. Graças à uma comunidade global de desenvolvedores assim que uma nova técnica de análise estatística é descoberta, a mesma já é inserida entre as bibliotecas de R, muito antes de ser incorporada em sistemas comerciais.

A difusão de novas funções é feita através da criação de “pacotes”, sendo possível a qualquer usuário desenvolver um novo pacote, com as funcionalidades que desejar, e, assim que pronto, colocar este pacote à disposição de qualquer um que deseje utilizá-lo, em um ou mais dos diversos repositórios dedicados à R presentes na Internet. Somente em um destes repositórios existem hoje mais de 6.000 pacotes.

Uma das principais vantagens da linguagem R é a possibilidade de se criar gráficos, desde os mais simples até os mais complexos e intrincados. R leva até usuários que não são analistas profissionais a possibilidade de criarem gráficos de alta qualidade de forma rápida e assertiva, tais como mapas, superfícies tridimensionais, gráficos de dispersão, histogramas e etc.

Porém, apesar de todas as vantagens da linguagem, R foi desenvolvida de forma a ser utilizada a partir de uma interface de linha de comando, onde para cada comando temos uma resposta imediata (como acontece em Scheme). Desta forma, ainda que muitos estatísticos acreditem que utilizar comandos diretamente seja mais produtivo e seguro, a curva de aprendizado necessária pode não justificar seu uso por indivíduos sem conhecimento estatístico.

Dado este cenário, no qual de um lado temos o grande poder computacional da linguagem R e de outro o fato de esta utilizar como meio de entrada uma interface de linhas de comando. A solução para que mais usuários possam utilizá-la, sem a necessidade de conhecimento e/ou treinamento em estatística e computação, consiste na criação de interfaces de usuário. Nos próximos capítulos serão apresentados os diversos aspectos da utilização de interfaces gráficas em linguagem R.

### 3 INTERFACES EM LINGUAGEM R

Para solucionar o problema da falta de uma interface de interação entre a linguagem R e os usuários que estão fora do grupo de estatísticos e programadores, e, portanto, não possuem conhecimento necessário para a utilização através de linhas de comando, existem atualmente uma série de projetos e estudos em andamento, alguns destes já bastante evoluídos. Porém, por mais que existam tais iniciativas e projetos, bibliografias e documentações sobre o assunto são bastante esparsas, em pequena quantidade e difíceis de encontrar.

Cada um destes projetos tem como objetivo disponibilizar ferramentas já utilizadas com sucesso em outra linguagem aos usuários de R. Destacam-se aqui os pacotes `RGtk2`, que fornece um link entre R e a biblioteca multiplataforma GTK+ (KRAUSE, 2007); a biblioteca de código aberto Qt (DALHEIMER, 2002), desenvolvida pela Nokia e disponível através do pacote `qtbase`; e, por fim, o pacote `TclTk`, o qual consiste na combinação de uma linguagem de scripts e um conjunto de ferramentas gráficas de interface, e está presente em R desde a versão 1.1.0 da linguagem.

O pacote `TclTk`, desenvolvido por Peter Dalgaard (DALGAARD, 2001), traz as ferramentas de criação de interfaces presentes na linguagem TK, criada por John Ousterhout como expansão a linguagem TCL de mesmo autor, e que já foi utilizada com sucesso em outras linguagens como Perl e Python. Embora ainda careça de recursos mais modernos, o pacote `TclTk` tem como grande vantagem o fato de vir previamente instalado na versão de R para Windows, deste modo evitando uma série de problemas de instalação e compatibilidade. Além disso, o pacote apresenta simplicidade de codificação, se assemelhando a simplicidade da própria linguagem R, e a portabilidade, já que pode ser usado em sistemas operacionais como Windows, Linux e OS X.

O `TclTk` ainda possui lacunas, características simples de softwares comerciais, tais como criação de instalador e inicialização através de ícone/atalho (fora do ambiente R). Para tornar possível o desenvolvimento de aplicações completas em R, através do uso do pacote `TclTk`, foi necessário o desenvolvimento de uma metodologia que é descrita na seção 3.1. Outra opção para utilização de linguagem R através de interfaces de usuário, como já descrito anteriormente neste trabalho, é o uso de uma linguagem mais poderosa, como Java, através da qual podemos criar uma interface e realizar chamadas externas para execução das funções definidas em R. Tal abordagem é descrita na sessão 3.2.

### 3.1 Criando interfaces utilizando apenas linguagem R

A criação de interfaces em R inicia-se na chamada do pacote. Isto é feito através do comando `library(tcltk)`. A simplicidade do pacote `TclTk` pode ser observada no exemplo a seguir, onde são definidas uma janela e uma *label*.

Listagem 3.1 – ‘Olá, Mundo!’ em R com interface gráfica.

```

1 library(tcltk)
2 window <- tktoplevel()
3 tktitle(window) <- "Exemplo 1"
4 label <- tklablel(window, text="Olá, Mundo!")
5 tkpack(label)

```

O código apresentado a cima, em Listagem 3.1, gera uma janela de título “Exemplo 1”, contendo a frase “Olá, Mundo!”, à qual corresponde a Figura 3.1. Através deste exemplo é possível visualizar a estrutura de comandos, que permite o acesso aos parâmetros de um objeto, no caso a janela *window*, e a modificação destes parâmetros. Pode-se, ainda, observar que os objetos são definidos de forma hierárquica e que a função `tkpack()` se encarrega do posicionamento do objeto filho na janela.

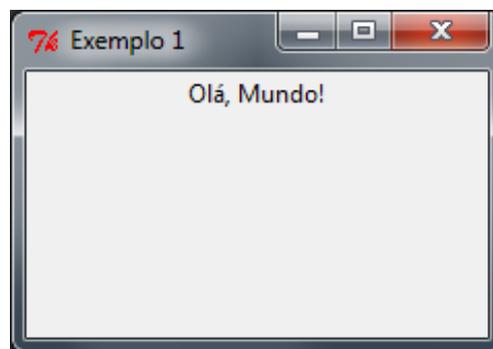


Figura 3.1 – Exemplo básico de interface em R.

A organização dos elementos presentes em uma janela é realizada através dos parâmetros `side`, `fill` e `expand` da função `tkpack()`, através destes é possível definir o posicionamento de cada elemento. Para definir grupos de elementos ou zonas na interface, utiliza-se `tkframes`, que nada mais são do que containers que também podem ser posicionados utilizando `tkpack()` e seus parâmetros. Utilizando `tkframes` é possível criar layouts complexos, porém é importante ressaltar que a ordem na qual são inseridos os elementos pode alterar toda a estrutura da interface, visto que na ausência de parâmetros de posicionamento os elementos são inseridos um abaixo do outro.

Listagem 3.2 – Posicionamento de elementos com função `tkpack()`.

```

1 library(tcltk)
2 window <- tktoplevel()
3 tktitle(window) <- 'Exemplo 2'
4
5 frameTop <- tkframe(window)
6 frameBottom <- tkframe(window)
7 frameLeft <- tkframe(window)
8 frameRight <- tkframe(window)
9
10 btnTop <- tkbutton(frameTop, text = "Top")
11 btnBottom <- tkbutton(frameBottom, text = "Bottom")
12 btnLeft <- tkbutton(frameLeft, text = "Left")
13 btnRight <- tkbutton(frameRight, text = "Right")
14
15 tkpack(frameTop, side='top', fill="both", expand=TRUE)
16 tkpack(frameBottom, side='bottom', fill="both", expand=TRUE)
17 tkpack(frameLeft, side='left', fill="both", expand=TRUE)
18 tkpack(frameRight, side='right', fill="both", expand=TRUE)
19
20 tkpack(btnTop, btnBottom, btnLeft, btnRight, fill="both", expand=TRUE)

```

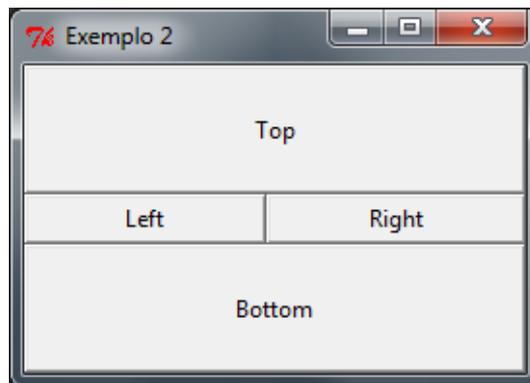


Figura 3.2 – Exemplo de interface em R com botões.

O exemplo apresentado em Listagem 3.2 demonstra o uso de `tkframes` e parâmetros de posicionamento. O parâmetro `side` especifica a posição, enquanto `fill` define os eixos nos quais o elemento se expandirá, sendo os valores possíveis `x`, `y` e `both` (ambos). Por fim, o parâmetro `expand` define se o elemento se expandirá para além de sua posição, ocupando áreas vazias vizinhas. A importância do uso de `fill` e `expand` está no fato de, sem eles, um elemento ocupar apenas o espaço necessário mínimo. No exemplo acima, caso não fossem utilizados

estes parâmetros, teria-se apenas 4 pequenos botões em cada uma das posições.

Como apresentado no exemplo anterior, botões são definidos através da primitiva `tkbutton`. No momento de criação de um botão é necessário que se especifique o `frame` no qual o mesmo será inserido e, através do parâmetro `text`, o texto que será mostrado no seu interior. Para possibilitar a interação de usuários a partir do uso de botões, utiliza-se o parâmetro `command` especificando a função a ser executada quando o botão é clicado.

### 3.1.1 Interação

O núcleo de toda interface gráfica consiste no laço de interação, é ele o responsável por manter a interface ativa pelo tempo que o usuário desejar interagir com o sistema. Em R, quando se utiliza o pacote `TclTk` este laço de interação pode ser alcançado de duas formas. A primeira delas é através do comando `tkwait.window('nomedajanela')`, juntamente com o uso do parâmetro `command` definido com as chamadas diretas das funções nos botões.

Listagem 3.3 – Controle de interface com `tkwait.window()`.

```

1 library(tcltk)
2 // Cria janela principal
3 janela <- tkoplevel(width = 500, height = 565)
4 tktitle(janela) <- "Exemplo 4"
5
6 frameA <- tkframe(janela, width = 500, height = 8)
7 // Cria botão com chamada direta da função someFunction()
8 btnA <- tkbutton(frameA, text='Botão A', width=10, command=someFunction)
9
10 // Aguarda usuário fechar janela principal
11 tkwait.window(janela)

```

Dessa forma, sempre que o usuário clicar em um botão, a função respectiva será chamada. O comando `tkwait.window()` mantém a janela informada como parâmetro ativa até que o usuário a encerre.

A segunda forma consiste na utilização de uma variável de controle juntamente com o comando `tkwait.variable(variávelcontrole)`. A função `tkwait.variable()`, diferentemente da função `tkwait.window()`, aguarda por mudanças em uma variável. Desta forma pode-se criar, após o término dos comandos de criação dos elementos gráficos da interface, um laço infinito no qual utiliza-se a função `tkwait.variable()` sobre uma variável de controle e, de acordo com o valor desta variável, são realizadas as chamadas às funções. O

uso de variável de controle implica, também, na forma como devem ser definidos os botões. Como as funções são chamadas com base no valor de uma variável, os parâmetros `command` dos botões devem ser definidos como funções de modificação desta variável, ao invés de chamadas diretas como na metodologia anterior. A segunda forma consiste na utilização de uma variável de controle juntamente com o comando `tkwait.variable(variávelcontrole)`. A função `tkwait.variable()`, diferentemente da função `tkwait.window()`, aguarda por mudanças em uma variável. Desta forma pode-se criar, após o término dos comandos de criação dos elementos gráficos da interface, um laço infinito no qual utiliza-se a função `tkwait.variable()` sobre uma variável de controle e, de acordo com o valor desta variável, são realizadas as chamadas às funções. O uso de variável de controle implica, também, na forma como devem ser definidos os botões. Como as funções são chamadas com base no valor de uma variável, os parâmetros `command` dos botões devem ser definidos como funções de modificação desta variável, ao invés de chamadas diretas como na metodologia anterior.

Listagem 3.4 – Controle de interface com `tkwait.variable()`.

```

1  library(tcltk)
2  //define variável de controle
3  control <- tclVar(0)
4  //Cria janela principal
5  janela <- tkoplevel(width = 500, height = 565)
6  tktitle(janela) <- "Exemplo5"
7  frameA <- tkframe(janela, width = 500, height = 8)
8
9  //Cria botão com chamada indireta de função através de
10 //modificação da variável de controle
11 btnA <- tkbutton(frameA, text='Botão A', width=10,
12 command=function() tclvalue(control)<-1)
13
14 //loop de execução
15 while(TRUE){
16     //Espera por modificação na variável de controle
17     tkwait.variable(control)
18     controlValue <- as.integer(tclvalue(control))
19     //Com base no valor obtido na variavel de controle,
20     //executa função respectiva
21     if (controlValue == 1){
22         someFunction()
23         control <- tclVar(0)
24     }

```

Ambas as abordagens são válidas e corretas e diferenciam-se apenas na questão de organização de código, sendo a segunda delas uma abordagem que apresenta maior centralização nas chamadas de funções da interface.

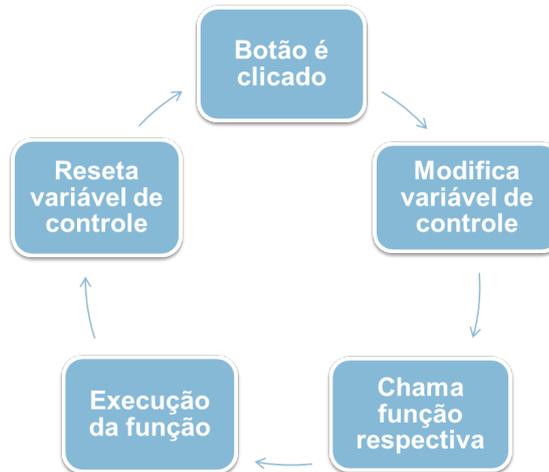


Figura 3.3 – Ciclo de abordagem utilizando espera em variável de controle.

### 3.1.2 Chamadas de Funções em R

Através da ferramenta Rscript, presente na instalação básica do ambiente de desenvolvimento da linguagem R desde as primeiras versões, pode-se executar programas e funções escritos em R fora do ambiente de desenvolvimento. Esta ferramenta, além de aumentar a flexibilidade da linguagem, é de essencial importância na execução de sistemas de usuários, já que não é desejável que um usuário tenha de acessar um sistema através de um ambiente de desenvolvimento do qual o mesmo pode não ter conhecimento. A execução de código R através da ferramenta Rscript segue o seguinte padrão:

```
Rscript [options] [-e expr [-e expr2 ...] | file] [args]
```

Exemplo:

```
C:\...\R\bin\i386\Rscript.exe exemplo.R
```

As funcionalidades de Rscript tornam possíveis até mesmo executar um programa escrito em linguagem R através de um ícone em sistemas operacionais Windows. Desta forma, aproximamos ainda mais sistemas em R de sistemas comerciais escritos em linguagens robustas.

Assim é possível observar que, através das ferramentas, funcionalidades e pacotes presentes em linguagem R, pode-se desenvolver não só interfaces em R mas todos os elementos que compõem sistemas complexos.

### 3.2 Criando interfaces utilizando Java com chamadas externas à linguagem R

A criação de um sistema composto por uma interface gráfica desenvolvida em Java através da qual são realizadas chamadas a código em linguagem R, segue, inicialmente, os mesmos passos comumente empregados na criação de qualquer interface gráfica nesta linguagem. A biblioteca gráfica mais utilizada em Java, atualmente, é a `Swing` (LOY et al., 2002) e está disponível desde a versão 1.2 da linguagem. `Swing` é considerada a biblioteca oficial para criação de componentes de interface gráfica, mas ainda existem algumas outras bibliotecas de terceiros, sendo a `SWT` (NORTHOVER; WILSON, 2004) a mais famosa. Desenvolvida pela IBM, a biblioteca `SWT` é conhecida por ter sido utilizada na criação do ambiente de desenvolvimento integrado `Eclipse`.

Não é objetivo deste trabalho explicar a utilização de bibliotecas gráficas em linguagem Java, sendo este um assunto já bastante abordado e para o qual existem bibliografias em grande quantidade, tais como (SCHILDT, 2006) e (ZUKOWSKI, 2005), além da já citada (LOY et al., 2002). Vale ressaltar apenas que `Swing` é uma biblioteca gráfica bastante simples no que diz respeito aos conceitos necessários para utilizá-la, tendo sua complexidade residente em sua grande gama de possibilidades.

Após a criação da interface gráfica, deve-se incluir a chamada ao código R. Isto deve ser feito através da utilização da ferramenta `Rscript`. Esta chamada pode ser incluída em botões, eventos ou em qualquer outro ponto no qual funcionalidades presentes em código R devam ser utilizadas. Em linguagem Java para que se possa utilizar uma ferramenta externa através de linhas de comando do sistema operacional, como é o caso de `Rscript`, deve-se utilizar o comando `Runtime.getRuntime().exec(comando)`, onde ‘comando’ corresponde a chamada da ferramenta `Rscript` juntamente com a localização do código R e de seus parâmetros. A classe `Runtime` possibilita a toda aplicação Java interagir com o sistema operacional.

Listagem 3.5 – Chamada de ferramenta `Rscript` através de Java utilizando classe `Runtime`.

```

1 String comando = "cmd start cmd.exe /K \"cd "+ R_folder + "&& Rscript.exe "
2   "C:\MyScriptLocation\myscript.R " + args[0..n] + "\"";
3
4 Runtime.getRuntime().exec(comando);

```

Para melhor entendimento, pode-se dividir o comando apresentado em três segmentos: O primeiro segmento, constituído pelo trecho `cmd start cmd.exe`, é responsável por iniciar e manter a execução do prompt de comandos enquanto ambos os comandos separados pelo conector '&&' são executados; o segundo segmento, cujo trecho de código é `cd C:\R_folder`, é responsável por localizar a ferramenta Rscript; já o terceiro segmento, formado pelo código `&& Rscript.exe C:\MyScriptLocation\myscript.R`, é responsável por executar a ferramenta Rscript sobre o arquivo que contém o código em linguagem R, passando ainda os parâmetros desejados.

O exemplo apresentado acima demonstra o funcionamento da chamada de Rscript em sistemas Windows, porém, nos demais sistemas operacionais (Linux e Mac OS) a execução segue os mesmos preceitos. É necessário iniciar a execução dos comandos, posicionar a execução do sistema operacional no diretório que contém a ferramenta Rscript, e, por fim, executar a ferramenta com a localização do arquivo em R e de seus parâmetros.

### **3.3 Considerações Finais**

Dadas estas duas abordagens de utilização do poder computacional da linguagem R, através de interface criada na mesma linguagem ou em linguagem Java. No próximo capítulo deste trabalho é apresentado um estudo de caso através do qual é possível, através da aplicação prática de ambas as abordagens, obter dados para posterior análise quantitativa e qualitativa. O objetivo deste estudo é o de desvendar as vantagens e desvantagens que uma abordagem pode ter sobre a outra, se há superioridade de alguma ou se são ambas equivalentes.

## 4 ESTUDO DE CASO

Para observar as duas possíveis abordagens de uso de linguagem R através de interfaces gráficas de usuário, apresentadas neste trabalho, desenvolveu-se um estudo de caso. Neste estudo de caso ambas as abordagens são utilizadas para a implementação de um sistema real. Foi escolhido, para este estudo, o sistema de diagnóstico de linearidade, o qual é utilizado para a verificação de modelos estatísticos sobre dados de experimentos laboratoriais.

### 4.1 Requisitos do sistema de Diagnóstico de Linearidade

É comum em ambientes laboratoriais, o desenvolvimento de novas metodologias, e sobre tais metodologias, frequentemente são utilizados modelos de regressão. Estes modelos são utilizados para verificar o comportamento dos dados quanto alguns pressupostos para validação da metodologia, sendo um destes pressupostos o de adequação a linearidade (NETER et al., 1996). A análise de regressão é uma modelagem matemática que busca verificar a relação entre duas ou mais variáveis e avaliar como determinadas variáveis estão influenciando as demais. Através da coleta de dados e do uso de métodos estatísticos é possível fazer suposições sobre o modelo, prever valores para a variável resposta e avaliar seu comportamento.

Quando um modelo de regressão é considerado para uma aplicação, no caso deste estudo, o modelo de regressão linear simples, usualmente não é possível inferir se tal modelo é apropriado ou não para a aplicação. Para tal inferência, necessita-se de uma série de diagnósticos do modelo em questão, através de métodos gráficos e testes deseja-se possibilitar a avaliação da adequação do modelo linear, e conseqüentemente o uso de técnicas corretivas que podem ser úteis quando os dados não estão em conformidade com o modelo de regressão.

Para o estudo do modelo de regressão linear, considera-se dois fatores importantes, a variável preditora e a variável resposta. Neste estudo a variável preditora será a razão de concentração (X) da variável que será analisada e que terá conseqüentemente uma razão de resposta (Y) relacionada a concentração. O diagnóstico da variável resposta diretamente não é muito usual na análise de regressão pois os valores das observações sobre a variável resposta são uma função do nível da variável preditora. Portanto, o diagnóstico da variável resposta é realizado indiretamente através da análise dos resíduos.

Neste estudo os seguintes teste estatísticos são utilizados: *Brown-Forsythe Test*, *Breusch-Pagan Test* e *F Test for Lack of Fit* (NETER et al., 1996). Deseja-se ainda que os usuários possam carregar uma base com os dados de seus experimentos, escolher entre quatro possíveis modelos de regressão (linear, quadrático, linear com pesos e quadrático com pesos) e que pos-

sam definir pontos de resposta e/ou concentrações que desejam excluir dos testes. É, também necessário que os usuários possam selecionar suas colunas de concentração e resposta dentre as presentes no arquivo de dados selecionado. Por fim, um relatório em formato HTML deve mostrar aos usuários os resultados de todos os testes bem como os gráficos gerados. Um exemplo deste relatório é apresentado no Apêndice A.

Assim, os requisitos correspondentes aos elementos de interface do sistema e suas respectivas funcionalidades são:

- Botão para seleção do arquivo de entrada, juntamente com área de visualização do nome e caminho do arquivo, devendo ser possível apenas a seleção de arquivos dos tipos “.xls” e “.xlsx”.
- Seleção do tipo de regressão a ser utilizado, devendo ser apresentadas todas as opções de análise de regressão mencionadas, sendo elas: Linear, Quadrática, Linear com pesos e Quadrática com pesos.
- Seletores para planilha e colunas de concentração e resposta. Deve ser possível aos usuários selecionar a planilha dentro do arquivo Microsoft Excel que contém os dados a serem utilizados, bem como a coluna com os dados que devem ser utilizados como concentração e a coluna que deve ser utilizada como resposta.
- Lista com todos os pontos de concentração presentes no arquivo de dados, possibilitando aos usuários a seleção de pontos que não desejam utilizar na análise, de forma a existir a possibilidade de remoção de possíveis valores discrepantes.
- Lista com os níveis de concentração presentes no arquivo de dados, onde “nível de concentração” representa um conjunto de pontos com valores semelhantes de concentração.
- Botão para realização da análise.

Desta forma, uma ferramenta desenvolvida deve ser eficaz, robusta e de fácil manuseio, para auxiliar nas avaliações de linearidade dos laboratórios em geral. Além de trazer novos recursos para a avaliação de resultados de linearidade como a diversidade de testes e de gráficos, a ferramenta dá aos laboratórios um ganho de tempo, pois todas as informações são geradas instantaneamente e sem a necessidade de envolver setores estatísticos em toda análise.

## **4.2 Implementação utilizando apenas linguagem R**

Para a implementação do estudo de caso apenas em linguagem R, utilizou-se o pacote TclTk com abordagem de espera em variável de controle. Esta abordagem, como já des-

crito no capítulo 3.1, utiliza a função `tkwait.variable(controle)` onde “controle” é uma variável cujo valor pode ser modificado através de botões da interface gráfica.

Listagem 4.1 – Estudo de caso - controle de interface em R

```

1 while(TRUE){
2   tkwait.variable(controle)
3   controle_value <- as.integer(tclvalue(controle))
4   if (controle_value == 1){
5     LinearDiagnostic()
6     controle <- tclVar(0)
7   }
8 }

```

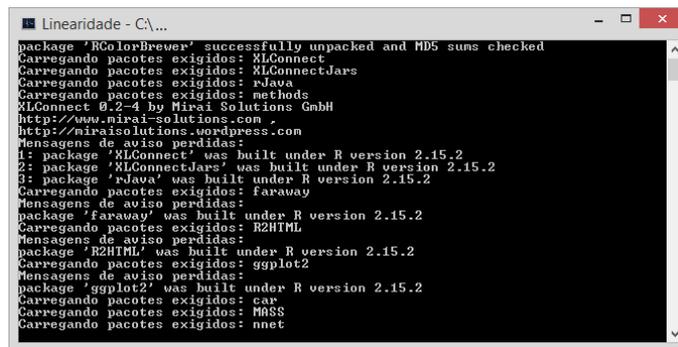


Figura 4.1 – Interface criada em R.

A função `LinearDiagnostic()`, através de chamadas a funções auxiliares, realiza todas as análises, elabora e apresenta o relatório final da aplicação, sendo assim a função principal do sistema criado em R.

Além da interface gráfica, apresentada na Figura 4.1, um *prompt* de comandos, Figura 4.2, acompanha a execução do sistema. Ele é o responsável por toda a computação em linguagem R, desde o carregamento dos pacotes até a criação e manutenção do componentes da interface.

Este *prompt* não pode ser encerrado, sob pena de encerrar a execução de toda a aplicação.



```

Linearidade - C:\...
package 'RColorBrewer' successfully unpacked and MD5 sums checked
Carregando pacotes exigidos: XLConnect
Carregando pacotes exigidos: XLConnectJars
Carregando pacotes exigidos: rJava
Carregando pacotes exigidos: methods
XLConnect 0.2-4 by Mirai Solutions GmbH
http://www.mirai-solutions.com
http://miraisolutions.wordpress.com
Mensagens de aviso perdidas:
1: package 'XLConnect' was built under R version 2.15.2
2: package 'XLConnectJars' was built under R version 2.15.2
3: package 'rJava' was built under R version 2.15.2
Carregando pacotes exigidos: faraway
Mensagens de aviso perdidas:
package 'faraway' was built under R version 2.15.2
Carregando pacotes exigidos: R2HTML
Mensagens de aviso perdidas:
package 'R2HTML' was built under R version 2.15.2
Carregando pacotes exigidos: ggplot2
Mensagens de aviso perdidas:
package 'ggplot2' was built under R version 2.15.2
Carregando pacotes exigidos: car
Carregando pacotes exigidos: MASS
Carregando pacotes exigidos: nnet

```

Figura 4.2 – Prompt de comandos que acompanha a execução da interface em R.

Para o carregamento da base de dados do usuário utilizou-se o pacote XLConnect, o qual possibilita a leitura de um arquivo em formato Microsoft Excel (.xls/.xlsx) selecionado pelo usuário através do botão ‘Procurar da interface. Já para a elaboração do relatório final, utilizou-se o pacote R2HTML, o qual possibilita a criação de um relatório em formato HTML. Assim, os usuários têm como resposta do sistema uma página web local, com gráficos, tabelas e todo o conteúdo necessário.

É possível ainda, através da interface gráfica gerada, definir o modelo a ser utilizado e selecionar as concentrações e pontos que os usuários não desejam incluir na elaboração do modelo. Sempre que uma concentração é selecionada, todos os pontos referentes a esta concentração são também selecionados. Para a escolha da planilha, coluna de concentração e coluna de resposta são utilizados menus do tipo “dropdown”, os quais apresentam aos usuários todas as possibilidades existentes dentro do arquivo selecionado.

As funções auxiliares responsáveis por todos os testes estatísticos, geração de gráficos e criação de relatório foram agrupadas em um arquivo separado, o qual é incluído no programa principal a partir da primitiva `source("funcoes.r")`. A primitiva `source()` funciona de forma a carregar todo o código presente no arquivo especificado dentro do arquivo chamador, dessa forma possibilita-se uma maior organização do código fonte.

Assim, a implementação da interface em linguagem R consiste basicamente de um módulo inicial, no qual são importadas as bibliotecas e inicializadas as variáveis globais. Seguido de um módulo de definição da interface através do uso de uma estrutura hierárquica de frames, criados a partir da função `tkframes()`, e no qual também está inserida a função de carregamento do arquivo de dados. Em um terceiro módulo, tem-se o conjunto de funções responsáveis pela seleção da planilha e colunas de concentração e resposta. Além da criação dos eventos de chamada destas funções, os quais correspondem à troca de valores nos seletores. A função

principal desta abordagem encontra-se em um quarto módulo e é responsável pela captura dos dados da interface e tratamento dos mesmos. Por fim, um módulo contendo o laço principal do sistema, através do uso da função `tkwait.variable()`, mantém a interface ativa.

### 4.3 Implementação utilizando interface em linguagem Java

Seguindo a abordagem de criação de sistema com interface gráfica desenvolvida em linguagem Java com chamadas externas a funcionalidades desenvolvidas em linguagem R, para a elaboração da aplicação descrita neste estudo de caso seguiu-se uma série de passos. Inicialmente uma interface gráfica foi desenvolvida com a utilização da biblioteca `Swing`. Porém, devido ao fato de em linguagem Java não existir uma forma direta de ler arquivos em formato Microsoft Excel, foi necessária a criação de uma classe para fornecer as ferramentas necessárias a este fim, possibilitando aos usuários o carregamento de seus arquivos de dados. À esta classe foi dado o nome `ExcelAdapter` e seus métodos `getSheets()`, `getColumns()` e `readColumn()` possibilitam a leitura e apresentação dos dados presentes nos arquivos carregados pelos usuários, desde que estes possuam os formatos corretos. O conjunto de ferramentas da biblioteca `Apache POI` mantida pela Apache Software Foundation foi utilizado como base para a criação desta classe. Por fim, fez-se necessária a criação de duas outras classes, `OSEnvironment` e `REnvironment`, responsáveis respectivamente por obter informações sobre o sistema operacional sobre o qual a aplicação está sendo executada e executar as chamadas à linguagem R. Na Figura 4.3 está representada a interface gráfica desenvolvida em Java e na Figura 4.4 pode-se observar a estrutura de classes criadas nesta implementação.

A classe `OSEnvironment` possui como principal componente o método `getOS()`, o qual, através do uso da primitiva `System.getProperty("os.name")`, permite descobrir o tipo de sistema operacional no qual a aplicação está inserida. Esta informação é de grande importância, pois para cada sistema operacional tem-se uma forma diferente de se executar a ferramenta `Rscript`. Embora a sintaxe do uso de `Rscript` seja a mesma, o que pode ser executado em um sistema Microsoft Windows através do comando `cmd start cmd.exe` em um sistema Linux seria executado através de um comando do tipo `/bin/bash`.

O objetivo da classe `REnvironment` é semelhante ao da classe `OSEnvironment`, porém direcionado a informações da instalação local do ambiente R. Através dos métodos desta classe torna-se possível a localização e a execução da ferramenta `Rscript`. Para a localização optou-se por uma abordagem na qual os locais mais recorrentes são verificados, outra opção seria inferir que o local da ferramenta `Rscript` esteja previamente definido como parte da variável de sistema

“PATH”. A execução segue o modelo descrito anteriormente, no qual através de comandos do sistema operacional localiza-se a ferramenta e a executa incluindo o arquivo de códigos e seus parâmetros.

Listagem 4.2 – Método runRonWindows da classe REnvironment

```

1 public void runRonWindows(String script_resource , String arguments [])
2 throws IOException , InterruptedException {
3     //Obtem arquivo de script R
4     String scriptAbsolutePath = findRonWindows() + script_resource ;
5
6     //Obtem caminho do arquivo
7     String scriptLocation = scriptAbsolutePath.substring(0 ,
8     scriptAbsolutePath.lastIndexOf(File.separator));
9
10    //Monta comando
11    String command = "cmd start cmd.exe /K \"cd "+ findRonWindows() +
12    " && Rscript.exe " + scriptAbsolutePath + " " + scriptLocation ;
13
14    //insere argumentos no comando
15    for(String arg : arguments){
16        command = command + " " + arg ;
17    }
18    command = command + " \";
19
20    //Executa script R
21    Runtime.getRuntime().exec(command);
22 }

```

O código apresentado em Listagem 4.2 demonstra a execução da ferramenta Rscript em um sistema Microsoft Windows. É possível observar que o primeiro parâmetro passado ao Rscript consiste na localização do arquivo de códigos, isso é feito para possibilitar ao R maior facilidade de localização durante sua execução. O método `findRonWindows()` (linhas 4 e 11 da Listagem 4.2), pertence a mesma classe e tem como responsabilidade a localização da instalação do ambiente R em sistemas Microsoft Windows.

Descritas ambas as abordagens para a criação da aplicação definida neste estudo de caso, no próximo capítulo são apresentadas as análises quantitativa e qualitativa dos resultados obtidos. Através do uso de métricas de software e das características observadas são discutidas as vantagens e desvantagens de cada modelo de desenvolvimento.

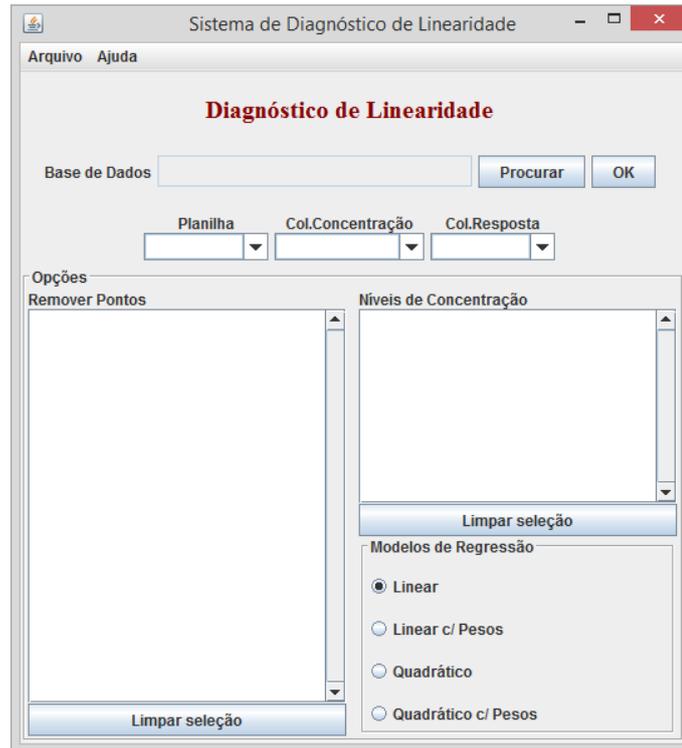


Figura 4.3 – Interface desenvolvida em Java.

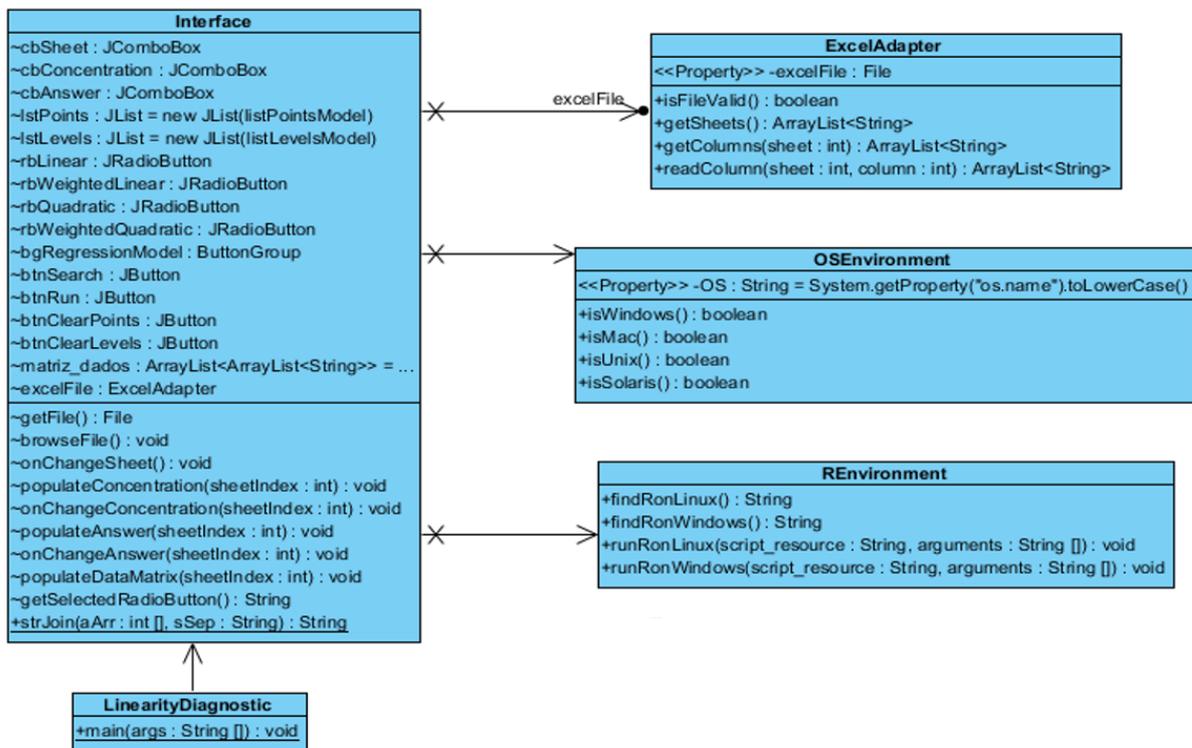


Figura 4.4 – Diagrama de classes da aplicação desenvolvida em Java.

## 5 DISCUSSÃO

Neste capítulo são apresentadas as análises realizadas sobre as implementações descritas no capítulo anterior. Uma análise quantitativa mostra os resultados numéricos obtidos através da aplicação de métricas sobre cada uma das implementações, além de medidas de uso de processamento e memória. A seguir, uma análise qualitativa apresenta as percepções e características obtidas a partir da implementação de cada abordagem.

### 5.1 Análise Quantitativa

Para a realização da análise quantitativa entre as abordagens apresentadas neste trabalho, foram selecionadas métricas de software (SOMMERVILLE, 2010) que podem ser calculadas tanto sobre o paradigma de orientação a objetos quanto sobre programação estruturada. Devido ao fato de atualmente não existirem ferramentas de cálculo de métricas sobre código escrito em linguagem R, fez-se uso da semelhança entre esta linguagem e a linguagem de programação C para que, após uma tradução de código, a extração das métricas fosse possível. Além disso, procurou-se comparar o processamento em percentual de CPU utilizada por cada implementação, bem como a quantidade de memória necessária. Por fim, foi realizada uma comparação estatística entre os tempos de execução de cada abordagem, considerando o tempo desde o apertar do botão *OK* da interface até a confecção do relatório.

A tradução de código R em código C foi realizada seguindo os três passos descritos no artigo (VOULGAROPOULOU; SPANOS; ANGELIS, 2012). Inicialmente todas as linhas de código devem ser acompanhadas por ‘;’, o que não é necessário em linguagem R mas torna-se obrigatório em linguagem C. O segundo passo consiste na modificação dos caracteres marcadores de comentários, o caractere ‘#’ utilizado em R deve ser substituído pelos caracteres ‘//’ para correta marcação em C. Por fim, declarações de funções que em R podem ser realizadas em apenas uma linha devem ser modificadas para um mínimo de duas linhas, isto se faz necessário pelo fato de o software utilizado para a obtenção das métricas não reconhecer funções declaradas em apenas uma linha de código.

Para a obtenção das métricas em código C traduzido, foi utilizada a ferramenta Source-Monitor (Campwood Software LLC, 2015). Já para a extração de métricas da abordagem desenvolvida em linguagem Java, foi utilizado o plugin `metrics2`<sup>1</sup> para o ambiente de desenvolvimento integrado Eclipse. As seguintes métricas foram empregadas.

---

<sup>1</sup><http://metrics2.sourceforge.net/>

1. **Complexidade**, a qual é calculada através da aplicação dos conceitos desenvolvidos por Thomas J. McCabe em 1976 (MCCABE, 1976), e mede a quantidade de caminhos de execução independentes a partir do código fonte. Esta métrica é conhecida como complexidade ciclomática de McCabe e pode ser aplicada a métodos, funções ou classes. Um número alto de complexidade é ruim, visto que um código complexo torna difícil a manutenção prejudica o desempenho e facilita o surgimento de falhas.
2. **Números totais de funções, métodos e classes.**
3. **Linhas de código efetivas**, ou seja, o número total de linhas de código menos as linhas deixadas em branco, os comentários e os limitantes de blocos (e.g. ‘{’, ‘}’ ). Este número está diretamente ligado à quantidade de trabalho necessário ao desenvolvimento e, conseqüentemente ao custo do mesmo.
4. **Fan-in e Fan-out** são, respectivamente, o número de funções ou métodos que chamam uma determinada função ou método ‘X’ e o número de funções ou métodos chamados por ‘X’.
5. **Número de pacotes ou bibliotecas externas utilizadas.**

A tabela 5.1 apresenta os resultados obtidos em cada uma das abordagens de implementação. É importante informar que o arquivo de funções em linguagem R não foi incluído na obtenção das métricas, visto que o mesmo é utilizado por ambas as abordagens logo não se faz relevante à comparação. De forma semelhante as classes da API do Java também não forma incluídas.

Métricas/Abordagem	Java	R
Complexidade média	2,78	4,17
Número de métodos	26	-
Número de classes	6	-
Número de funções	-	6
Linhas de código	617	395
Bibliotecas/Pacotes	13	7(22)
Fan-in médio	0,83	1
Fan-in máximo	1	1
Fan-out médio	2	1,33
Fan-out máximo	8	8

Tabela 5.1 – Métricas de software.

Na linha referente ao número de pacotes ou bibliotecas externas utilizadas, o número apresentado para a abordagem em R é representado pelo número de pacotes diretamente utilizados seguido pelo número total de pacotes, o qual inclui todas as dependências que devem ser insta-

ladas.

É possível observar, através da tabela 5.1, que a abordagem R possui uma maior complexidade média, embora possua um número consideravelmente menor de linhas de código, o qual em percentual corresponde a um número 35,98% menor do que o apresentado pela abordagem em linguagem Java. Pode-se atribuir isso ao fato de a abordagem em Java utilizar o paradigma de orientação a objetos, o qual leva a uma maior organização do sistema ao custo da criação de um número maior de componentes. O número alto de métodos em Java também tem relação com a necessidade de criação de componentes capazes de trazer à linguagem Java funcionalidades que seriam acessadas naturalmente em linguagem R, como o caso da leitura de arquivos em formato Microsoft Excel. Além disso, observa-se uma grande semelhança no que diz respeito às métricas de ‘Fan-in’ e ‘Fan-out’.

Para a obtenção dos números de processamento e memória utilizados por cada implementação, foi utilizada a ferramenta de monitoramento de recursos do sistema operacional Microsoft Windows. Aqui não se buscou precisão absoluta nos dados, apenas uma forma de comparação entre os processos.

Abordagem	Threads	% CPU Médio
R	13	1,26
Java	21	0,13

Tabela 5.2 – Medidas de processamento.

Abordagem	Confirmar(KB)	Conjunto de Trabalho(KB)	Compartilhável(KB)	Privada(KB)
R	60.616	52.068	12.432	39.636
Java	145.712	49.920	14.952	34.968

Tabela 5.3 – Medidas de memória utilizada.

As medidas de memória utilizadas na tabela 5.3 correspondem à:

1. Confirmar: Corresponde à quantidade de memória que o sistema operacional reservou para o processo.
2. Conjunto de trabalho: Quantidade de memória atualmente em uso pelo processo.
3. Compartilhável: Parte da memória que está sendo usada pelo processo e compartilhada com outros processos.
4. Privada: Quantidade de memória em uso apenas pelo processo em questão.

Os números apresentados nas tabelas 5.2 e 5.3 mostram uma semelhança grande entre as abordagens. Quando se trata de processamento, a abordagem em Java apresenta um uso menor

de processamento, com a utilização de mais Threads, isso pode ser atribuído ao uso da máquina virtual Java, a qual em comparação com o ambiente no qual são executados os códigos R, apresenta um estado muito mais sofisticado atualmente. No quesito memória utilizada, os resultados são ainda mais semelhantes e fica difícil eleger qualquer das abordagens como superior.

Para a comparação estatística entre os tempos de execução de ambas as abordagens, realizou-se, inicialmente, o teste de normalidade para o conjunto de dados de ambas as plataformas. O teste de *Shapiro-Wilk* (CONOVER, 1999) rejeita a hipótese de normalidade nos dois casos com  $p\text{-valor} < 0.001$ . Na tabela 5.4 tem-se a descrição dos dados de execução obtidos, enquanto na tabela 5.5 é possível observar os  $p\text{-valores}$  gerados.

Abordagem	N	Minimo	Máximo	Média	Desvio Padrão	CV (%)
R	30	1.409001	1.745826	1.504269	0.080944	5.380946
Java	30	4.725772	9.778457	5.805931	1.131856	19.49482

Tabela 5.4 – Descrição dos dados.

Abordagem	p-valor Shapiro-Wilk
R	0.000008761
Java	0.000006043

Tabela 5.5 –  $p\text{-valores}$  do teste de *Shapiro-Wilk*.

Assim, como não foi evidenciada normalidade dos dados, utilizou-se o teste *U* de *Wilcoxon-Mann-Whitney* (CONOVER, 1999) para comparar o tempo de execução das duas abordagens. Este teste é adequado para verificar se duas amostras independentes foram extraídas de uma mesma população, trata-se de uma alternativa não paramétrica para comparar duas amostras quando a suposição de normalidade não é obtida. O teste rejeita a hipótese nula ( $p\text{-valor} < 0.001$ ) de que as distribuições das amostras foram selecionadas da mesma população.

Com isso, conclui-se que existem diferenças significativas nos tempos de execução das abordagens, sendo a abordagem com linguagem R a que apresenta menores valores. Na figura 5.1 pode-se observar o gráfico gerado com os resultados obtidos para cada abordagem. O nível de significância utilizado nos testes é de 5%.

## 5.2 Análise Qualitativa

A primeira percepção que se tem ao comparar as abordagens é a diferença visual entre a interface criada em linguagem Java e a interface criada em linguagem R, o que pode influenciar em uma escolha, considerando a preferência dos desenvolvedores. Em Java tem-se um número

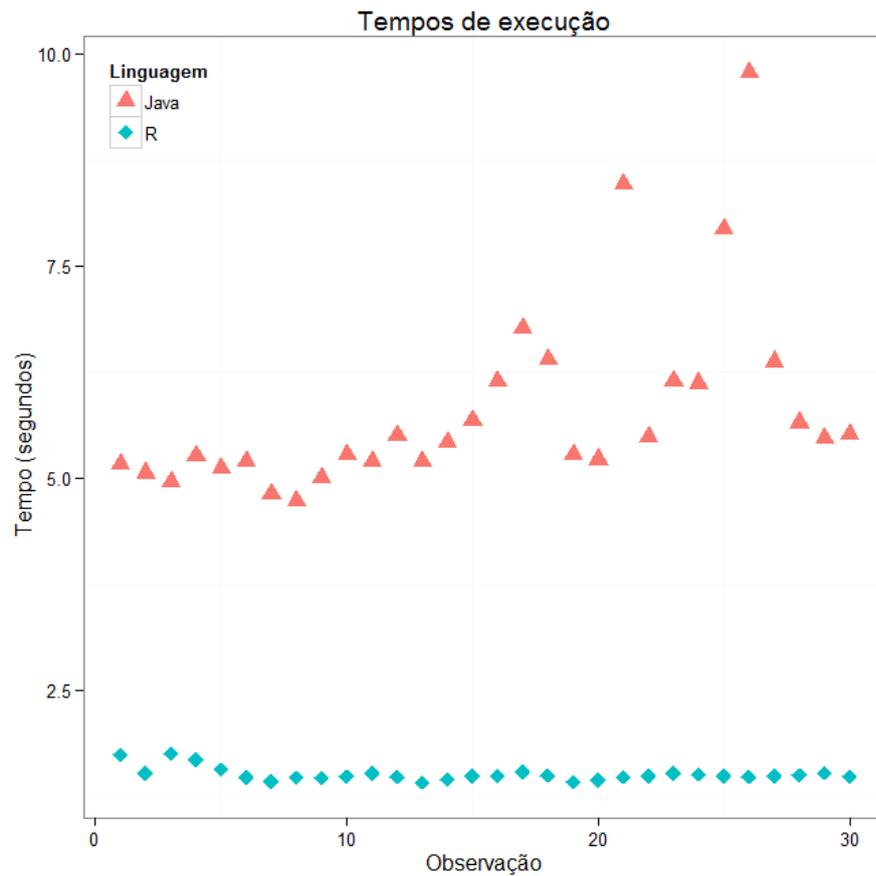


Figura 5.1 – Gráfico de diferenças entre tempos de execução.

muito maior de componentes gráficos e uma maior variedade de opções de organização destes componentes. A biblioteca gráfica utilizada em Java vem sendo utilizada já a bastante tempo e apresenta um estado da arte robusto e cheio de opções. Enquanto na linguagem R, para a organização dos componentes de uma interface, tem-se apenas quatro opções de posicionamento (“Top”, “Bottom”, “Left”, “Right”), o que leva a necessidade de se utilizar frames sobre frames para organizações mais complexas. Em Java, com o uso de *Swing*, tem-se um leque de opções de gerenciadores de *layout*, tais como `BorderLayout`, `BoxLayout`, `GridBagLayout`, entre outros, através dos quais é possível organizar os componentes com base em zonas, colunas ou tabelas.

Ainda sobre o ponto de vista visual, a linguagem R tem ainda uma desvantagem no fato de fazer necessária a execução de um prompt de comandos durante a apresentação da interface. Isto, em um ambiente voltado ao usuário final, passa a ser de uma relevância bastante grande.

Porém, ainda que a implementação em linguagem Java traga uma série de vantagens no âmbito gráfico, esta acaba por trazer também uma série de problemas que devem ser enfrentados. Entre tais problemas, tem-se como a principal dificuldade gerada a leitura de arquivos em for-

mato Microsoft Excel (.xls/.xlsx), para a qual é necessária a criação de uma classe especializada juntamente com o uso de bibliotecas externas. O uso de bibliotecas externas já apresenta em si uma dificuldade, pois estudá-las e entendê-las para um uso correto e eficiente, o que acaba por adicionar complexidade ao projeto. Ainda como dificuldade, a implementação Java traz a necessidade de se conhecer o ambiente no qual a aplicação está sendo executada, isto é, o sistema operacional utilizado. Isto é necessário pois para cada sistema operacional tem-se uma forma diferente de executar a ferramenta Rscript. A execução da ferramenta Rscript também é dificultada nesta implementação, pois é necessário localizá-la, ainda que ambas as abordagens partam do pressuposto da utilização de um instalador capaz de posicionar corretamente os arquivos necessários, tais como pacotes e scripts em R, em locais conhecidos. Os últimos dois problemas da abordagem Java tornam necessária a criação de duas outras classes, o que, novamente, agrega complexidade ao projeto.

Outro ponto bastante relevante é a diferença significativa nos tempos de execução, com a abordagem R sendo mais rápida e apresentando um conjunto de valores que variam pouco, enquanto a implementação com Java apresenta valores mais altos e uma maior variabilidade nos dados. Esta diferença de tempos pode ser atribuída à necessidade de se inicializar o ambiente R sempre que é necessário realizar análises estatísticas na abordagem com linguagem Java. O que não acontece quando utiliza-se apenas linguagem R, pois neste caso o ambiente já está em execução desde o início da aplicação.

A implementação em linguagem R tem como grande vantagem a facilidade de leitura de arquivos e a manipulação destes. Em R graças a estrutura de dados “data.frame” é possível criar uma matriz com diferentes tipos de dados em cada coluna, isto é, é possível em R ter-se uma matriz na qual a primeira coluna seja do tipo “String”, a segunda do tipo “Integer” e uma terceira coluna de qualquer outro tipo de dado. A facilidade com a qual isto é acessado torna R uma linguagem poderosa no tratamento de dados.

Um ponto que também deve ser observado é a necessidade da presença de uma instalação da máquina virtual Java para a execução da implementação em linguagem Java. Embora atualmente segundo o site “java.com”, 97% dos computadores empresariais e 89% de todos os computadores dos Estados Unidos possuem esta instalação de Java, há ainda a necessidade de a versão do usuário ter de ser compatível com a versão na qual a aplicação foi desenvolvida.

Com bases nas análises quantitativa e qualitativa apresentadas, no próximo capítulo é apresentada a conclusão deste trabalho, a qual é gerada a partir das percepções de valores e características obtidos de ambas as abordagens através do estudo de caso definido no capítulo 4 e tem base nos conceitos apresentados nos capítulos 2 e 3.

<b>Abordagem R</b>	<b>Abordagem Java</b>
Pontos Positivos	Pontos Positivos
(+) Leitura de arquivos facilitada (+) Menor número de linhas de código (+) Maior velocidade de execução	(+) Organização de componentes gráficos mais evoluída
Pontos Negativos	Pontos Negativos
(-) Necessária execução de prompt (-) Menos formas de organização de componentes gráficos (-) Código mais complexo	(-) Necessário uso de bibliotecas externas e criação de classe para leitura de arquivos (-) Necessário conhecer sistema operacional (-) Necessário localizar Rscript

Tabela 5.6 – vantagens e desvantagens de cada abordagem.

### 5.3 Limitações do Trabalho

Algumas limitações deste trabalho devem ser observadas. Entre elas o fato de apenas um programador estar envolvido no desenvolvimento das implementações aqui apresentadas. Além disso, os teste apresentados foram realizados apenas sobre o sistema operacional Microsoft Windows. Há, também, a dificuldade de se comparar duas linguagens de paradigmas diferentes.

### 5.4 Considerações Finais

A partir das análises apresentadas neste capítulo e com base nos resultados obtidos em cada uma delas, tem-se, no próximo capítulo, a conclusão deste trabalho, na qual está presente e consolidado todo o conhecimento gerado no decorrer do mesmo.

## 6 CONCLUSÃO

Após os conceitos apresentados e observando os resultados das análises quantitativa e qualitativa é possível concluir que quando se trata da criação de interfaces gráficas de usuário a linguagem Java possui significativa superioridade à linguagem R. Porém, esta superioridade tem como custo um acréscimo relevante de carga de trabalho, tornando assim a abordagem de criação de interfaces através de R uma melhor opção quando se trata de projetos de tamanho menor.

Os diversos componentes gráficos e a diversidades de gerenciadores de *layout* tornam a linguagem Java a opção ideal para a criação de interfaces gráficas de usuário complexas presentes em grandes projetos. Esta vantagem que a linguagem possui vem a justificar o acréscimo de trabalho gerado pela necessidade de inclusão de classes e métodos auxiliares de tratamento que não seriam necessárias em uma abordagem que utiliza apenas linguagem R. Entretanto, em projetos pequenos, com interfaces mais simples o acréscimo de complexidade não se justifica, já que este tipo de interface pode de forma aceitável ser realizado utilizando apenas as ferramentas presentes na linguagem R.

O estudo de caso apresentado mostrou-se um ótimo exemplo para comparação das abordagens, pois o mesmo encontra-se no limiar entre uma estrutura de interface simples e uma estrutura complexa. Através das implementações fica evidente que através da linguagem Java obtém-se um visual mais atraente, ao passo que a complexidade exigida para tal também se torna evidente. Assim, em casos como este, fica ao desenvolvedor ou projetista a decisão entre um melhor visual ao preço de uma maior carga de trabalho que por sua vez leva a um tempo de desenvolvimento maior e conseqüente aumento do custo do projeto, ou um visual mais simples acompanhado de um tempo menor de projeto, com um código menor mas com maior complexidade. Nos demais casos, quando se trata de um projeto com interfaces complexas e elaboradas sugere-se o uso de linguagem Java, enquanto projetos com interfaces simples levam a uma implementação com o uso de R somente.

Este trabalho abre a possibilidade para trabalhos futuros, entre eles o desenvolvimento de um estudo comparativo de interfaces em R e Java com a presença de testes com usuários, o que tende a gerar uma comparação mais rica entre as abordagens e facilitar a escolha dos programadores.

## REFERÊNCIAS

- AMIRTHA, T. **WHY THE R PROGRAMMING LANGUAGE IS GOOD FOR BUSINESS**. 2014. Available from Internet: <<http://www.fastcompany.com/3030063/why-the-r-programming-language-is-good-for-business>>.
- BECKER, R. A.; CHAMBERS, J. M. **S: An Interactive Environment for Data Analysis and Graphics**. [S.l.]: Chapman and Hall/CRC, 1984. (His Competencies for Teaching; V. 3).
- BRODERSEN, K. H. **CausalImpact: A new open-source package for estimating causal effects in time series**. 2014. Available from Internet: <<http://google-opensource.blogspot.de/2014/09/causalimpact-new-open-source-package.html>>.
- Campwood Software LLC. **SourceMonitor (TM) Version 3.5**. 2015. <<http://www.campwoodsw.com/sourcemonitor.html>>.
- CONOVER, W. J. **Practical Nonparametric Statistics**. 3. ed. [S.l.]: Wiley, 1999.
- DALGAARD, P. The r-tcl/tk interface. **Proceedings of the 2nd International Workshop on Distributed Statistical Computing**, 2001.
- DALHEIMER, M. K. **Programming with Qt**. 2. ed. [S.l.]: O'Reilly Media, 2002.
- DELWICHE, L.; SLAUGHTER, S. **The little SAS book**. 5. ed. [S.l.]: SAS Institute, 2012.
- FIELD, A. **Discovering Statistics using IBM SPSS Statistics**. 4. ed. [S.l.]: SAGE publications Ltd, 2013.
- HINER, J. **Ford's Big Data chief sees massive possibilities, but the tools need work**. 2012. Available from Internet: <<http://www.zdnet.com/article/fords-big-data-chief-sees-massive-possibilities-but-the-tools-need-work/>>.
- IHAKA, R.; GENTLEMAN, R. R. A language for data analysis and graphics. **Journal of Computational and Graphical Statistics, Volume 5, Number 3. Pages 299-314**, 1996.
- KRAUSE, A. **Foundations of GTK+ Development**. [S.l.]: Apress, 2007. (Expert's Voice in Open Source).
- LOY, M. et al. **Java Swing**. 2. ed. [S.l.]: O'Reilly Media, 2002.
- MCCABE, T. J. A complexity measure. **IEEE Trans. Softw. Eng.**, IEEE Press, Piscataway, NJ, USA, v. 2, n. 4, p. 308–320, jul. 1976. ISSN 0098-5589. Available from Internet: <<http://dx.doi.org/10.1109/TSE.1976.233837>>.
- MCNALLY, S. **Names You Need to Know in 2011: R Data Analysis Software**. 2010. Forbes, October 11, 2010. Available from Internet: <<http://www.forbes.com/sites/smcnally/2010/11/10/names-you-need-to-know-in-2011-r-data-analysis-software/>>.
- MORGAN, T. P. **R is ready for big data**. 2012. The Register, June 03, 2012. Available from Internet: <[http://www.theregister.co.uk/2012/06/03/big\\_data\\_r\\_statistical\\_analysis/](http://www.theregister.co.uk/2012/06/03/big_data_r_statistical_analysis/)>.
- NETER, J. et al. **Applied Linear Statistical Models**. 4. ed. [S.l.]: McGraw-Hill/Irwin, 1996. (Irwin Series in Statistics).

NORTHOVER, S.; WILSON, M. **SWT: The Standard Widget Toolkit**. 1. ed. [S.l.]: Addison-Wesley Professional, 2004.

SCHILDT, H. **Swing: A Beginner's Guide**. 1. ed. [S.l.]: McGraw-Hill Education, 2006.

SOMMERVILLE, I. **Software Engineering**. 9. ed. [S.l.]: Pearson, 2010.

STODDER, D. **Expert Analysis: You Can Predict That R Will Succeed**. 2010. InformationWeek, March 03, 2010. Available from Internet: <<http://www.informationweek.com/software/information-management/expert-analysis-you-can-predict-that-r-will-succeed/d/d-id/1087294?>>

SUSSMAN, G. J.; JR., G. L. S. Scheme: An interpreter for extended lambda calculus. **MIT AI Lab. AI Lab Memo AIM-349. December 1975**, 1975.

VANCE, A. **Data Analysts Are Mesmerized by the Power of Program R**. 2009. New York Times, January 7, 2009, page B6. Available from Internet: <[http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?pagewanted=1&\\_r=1](http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?pagewanted=1&_r=1)>.

VOULGAROPOULOU, S.; SPANOS, G.; ANGELIS, L. Analyzing measurements of the r statistical open source software. In: **Proceedings of the 2012 35th Annual IEEE Software Engineering Workshop**. Washington, DC, USA: IEEE Computer Society, 2012. (SEW '12), p. 1–10. ISBN 978-0-7695-4947-7. Available from Internet: <<http://dx.doi.org/10.1109/SEW.2012.7>>.

WAYNER, P. **7 programming languages on the rise**. 2010. InfoWorld, October 25, 2010. Available from Internet: <<http://www.infoworld.com/article/2624163/techology-business/7-programming-languages-on-the-rise.html>>.

WU, X. et al. Data mining with big data. **Knowledge and Data Engineering, IEEE Transactions**, v. 26, p. 97–107, 2013.

ZUKOWSKI, J. **The Definitive guide to Java Swing**. [S.l.]: Apress, 2005.

## APÊNDICE A SISTEMA DE DIAGNÓSTICO DE LINEARIDADE

Neste apêndice é apresentado um exemplo de relatório gerado pela aplicação *Sistema de Diagnóstico de Linearidade*.

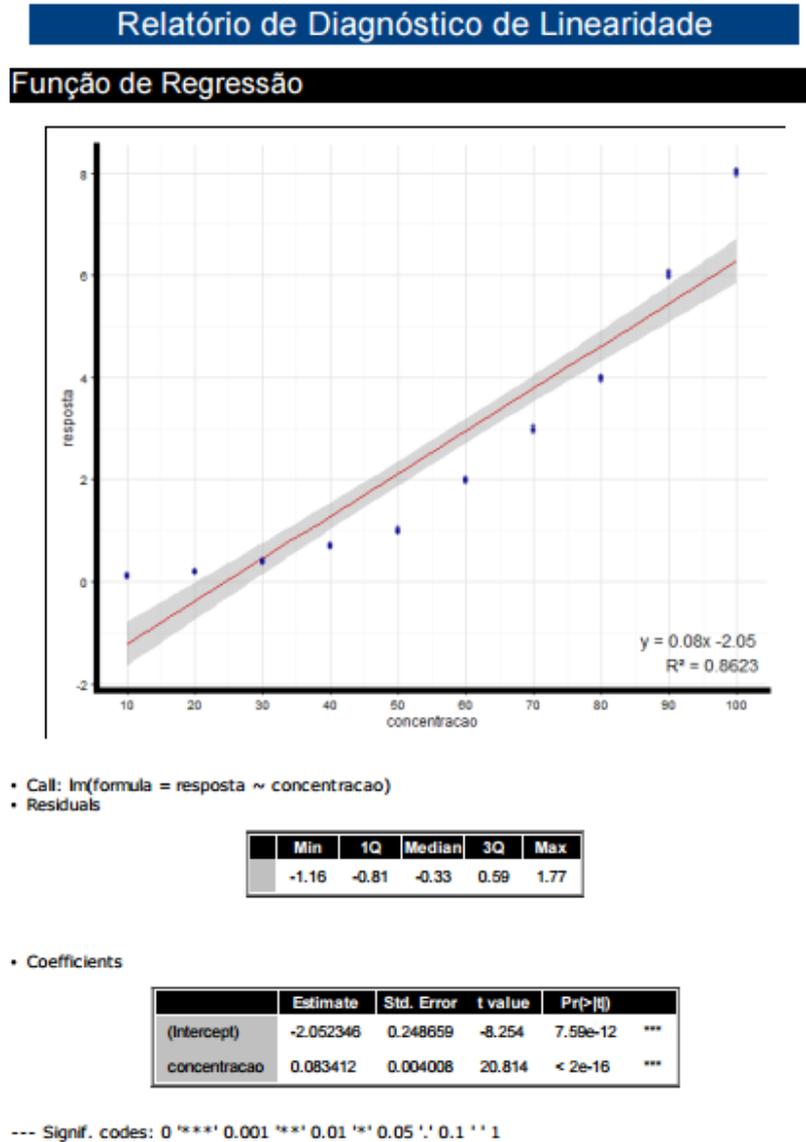


Figura A.1 – Relatório - Sistema de Diagnóstico de Linearidade

- Residuals standard error: 0.9631 on 68 degrees of freedom
- Multiple R-Squared: **0.8643**
- Adjusted R-Squared: **0.8623**
- F-statistics: **433.22** on 1 and 68 DF. P-value: **0**.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
t\$X	1	401.80	401.80	433.22	< 2.2e-16 ***
Residuals	68	63.07	0.93		

--- Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Teste de Ajuste da Função de Regressão

### F-test for Lack of Fit

- $F^* = 7635.85$   
 $F = 2.1$

- p-valor = 0

[+ Informações](#)

## Testes de Homocedasticidade

### Brown-Forsythe

- $t^* = |-0.52|$   
 $t = 2$

- p-valor = 0.6064

### Breusch-Pagan

- $X^{2*} = |2.35|$   
 $X^2 = 3.84$

- p-valor = 0.1257

Figura A.2 – Relatório - Sistema de Diagnóstico de Linearidade

### Teste de Ajuste da Função de Regressão

#### F-test for Lack of Fit

- $F^* = 7635.85$   
 $F = 2.1$

- p-valor = 0

[+ Informações](#)

### Testes de Homocedasticidade

#### Brown-Forsythe

- $t^* = |-0.52|$   
 $t = 2$

- p-valor = 0.6064

#### Breusch-Pagan

- $\chi^{2*} = [2.35]$   
 $\chi^2 = 3.84$

- p-valor = 0.1257

[+ Informações](#)

### Teste de 'outliers'

Não foram detectados outliers

Figura A.3 – Relatório - Sistema de Diagnóstico de Linearidade

## Informações

### Testes

#### F Test for Lack of Fit

O F Test for Lack of Fit, tem como objetivo verificar se a função de regressão está adequadamente associada aos dados observados. A hipótese a ser testada é, se para um determinado valor esperado de Y, a equação da reta de regressão obtida poderá fornecer um valor de X significativamente robusto, associado ao valor esperado.

A estatística calculada pelo teste é  $F^*$ , sua interpretação é dada pela relação o valor da distribuição de F tabelado

Se  $F^* \leq F$ , ou p-value  $> \alpha$ : a função de regressão obtida se ajusta ao modelo.

Se  $F^* > F$ , ou p-value  $\leq \alpha$  a função de regressão obtida não se ajusta ao modelo.

Foi utilizado para esta análise um nível de significância  $\alpha = 0.05$  e  $(c - 2, n - c)$  graus de liberdade.

[voltar](#)

#### Brown-Forsythe Test

O teste de Brown-Forsythe é primeiro dos dois testes apresentados para testar se a variância é constante, este teste consiste em separar os dados em dois grupos, e baseado no teste t para duas amostras, comparar se a média dos desvios absolutos de um grupo difere significativamente da média dos desvios do outro grupo.

A estatística calculada pelo teste é  $t^*$ , sua interpretação é dada pela relação com o valor da distribuição de t tabelado

Se  $t^* \leq t$ , ou p-value  $> \alpha$ : conclui-se que a variância do erro é constante ao longo de X. (homocedasticidade)

Se  $t^* > t$ , ou p-value  $\leq \alpha$ : conclui-se que a variância do erro NÃO é constante ao longo de X.(heterocedasticidade)

Foi utilizado para esta análise um nível de significância  $\alpha = 0.05$  e  $(n-2)$  graus de liberdade.

#### Breusch-Pagan Test

O Breusch-Pagan test é um segundo teste para verificar se a variância é constante, este teste pressupõe que a variância constante esta diretamente associada ao segundo termo da equação de regressão, portanto a hipótese de variância constante a ser testada é quando este termo é igual a 0.

A estatística calculada pelo teste é  $X^{2*}$ , e sua interpretação se da pela relação com o valor da distribuição de  $X^2$  tabelado

Se  $X^{2*} \leq X^2$ , ou p-value  $> \alpha$ : conclui-se que a variância do erro é constante ao longo de X.(homocedasticidade)

Se  $X^{2*} > X^2$ , ou p-value  $\leq \alpha$ : conclui-se que a variância do erro NÃO é constante ao longo de X. (heterocedasticidade)

Foi utilizado para esta análise um nível de significância  $\alpha = 0.05$  e 1 grau de liberdade.

[voltar](#)

#### Outliers

Retorna o p-valor do teste de Bonferroni para os valores considerados extremos nas observações (outliers).

[voltar](#)

Figura A.4 – Relatório - Sistema de Diagnóstico de Linearidade