

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
LABORATÓRIO DE ENGENHARIA E EXPLORAÇÃO DE PETRÓLEO

TRATAMENTO ESTATÍSTICO DE DADOS GEOQUÍMICOS

Márcio Luís Carvalho Araújo
Tamires dos Santos Soares
Thiago Rocha Gomes

MACAÉ - RJ
JANEIRO - 2014

Sumário

1	Introdução	1
1.1	Escopo do Problema	1
1.2	Objetivo	2
1.3	Organização do Documento	2
2	Especificação	3
2.1	Especificação do programa - descrição dos requisitos	3
2.1.1	Definições da interface	3
2.1.2	Entrada e saída de dados	3
2.2	Casos de uso do Programa	3
2.3	Diagrama de caso de uso geral do programa	4
2.4	Diagrama de caso de uso específico do programa	5
3	Elaboração	7
3.1	Revisão de conceitos básicos de estatística	7
3.1.1	CONCEITOS ESTATÍSTICOS BÁSICOS	7
3.1.2	REGRESSÃO LINEAR	8
3.1.3	DISTRIBUIÇÃO NORMAL	9
3.1.4	DISTRIBUIÇÃO T STUDENT	10
3.1.5	TESTE DE HIPÓTESES	12
3.1.6	TESTE DE HIPÓTESES MÉDIA	12
3.1.7	<i>OUTLIERS</i>	14
3.1.8	TESTE DO ESCORE Z MODIFICADO	15
3.1.9	TESTE DE GRUBBS	16
3.1.10	TESTE DE DIXON	20
3.1.11	TESTE DE COCHRAN	22
3.1.12	TESTE DE DOERFFEL	25
3.2	Análise de domínio	26
3.3	Identificação de pacotes – assuntos	27
3.4	Diagrama de pacotes – assuntos	27

4	AOO – Análise Orientada a Objeto	28
4.1	Diagramas de classes	28
4.1.1	Dicionário de classes	30
4.2	Dicionário de classes com atributos/métodos	31
4.3	Diagrama de sequência – eventos e mensagens	37
4.3.1	Diagramas de sequência	37
4.4	Diagrama de atividades	45
5	Projeto	52
5.1	Projeto do sistema	52
5.2	Projeto Orientado a Objeto – POO	52
6	Implementação	54
6.1	Código fonte	54
7	Teste	88
7.1	Teste 1: Descrição	88
7.2	Teste 2: Descrição	89
7.3	Teste 3 : Descrição	90
7.4	Teste 4 : Descrição	91
7.5	Teste 5 : Descrição	92
7.6	Teste 6 : Descrição	93
7.7	Teste 7: Descrição	94
7.8	Teste 8: Descrição	95
8	Documentação	97
8.1	Documentação do usuário	97
8.1.1	Como instalar o software	97
8.1.2	Como rodar o software	98
8.2	Documentação para desenvolvedor	98
8.2.1	Dependências	98
8.2.2	Como gerar a documentação usando doxygen	98

Capítulo 1

Introdução

Os métodos estatísticos estão associados ao tratamento de informações. Seu emprego tem por objetivo explorar uma certa quantidade de números e extrair dos mesmos, valiosas conclusões.

1.1 Escopo do Problema

Normalmente, em experimentos realizados no laboratório de geoquímica, observa-se que algumas amostras apresentam um valor discrepante com relação as demais. Um exemplo que pode ser dado para um melhor entendimento do problema é o das percentagens das frações se NSO¹, saturados² e aromáticos³

No laboratório tem-se n pesquisadores que utilizam a amostra 09, por exemplo, para calcular suas percentagens de saturados, aromáticos e NSO. Sabe-se que cada pesquisador pode encontrar valores diferentes destas percentagens para essa mesma amostra. Tais valores normalmente variam pouco de experimento para experimento, mas há casos em que a percentagem varia consideravelmente (Tabela 1.1).

Tabela 1.1: Percentagens de SAT, ARO e NSO para a amostra 09.

Pesquisador	% SAT	% ARO	% NSO
Fernanda	50	20	30
Hilda	47	23	30
Laercio	52	25	23
Mateus	10	70	20
Natieli	55	25	20

Como pode ser observado através da tabela 1.1, as porcentagens calculadas pelo pesquisador Mateus apresentam um valor bastante diferente das percentagens de SAT e ARO

¹Compostos NSO são compostos de petróleo que contém átomos de nitrogênio, enxofre e oxigênio em sua estrutura molecular.

²Hidrocarbonetos que possuem apenas ligações simples entre os átomos de carbono.

³Hidrocarbonetos que possuem o anel benzênico.

com relação ao que os outros pesquisadores encontraram. Acredita-se, portanto, que tais valores sejam *outliers*. A partir desta suspeita é preciso então comprová-la, o que normalmente é feito através dos testes: Teste do escore z modificado, Teste de Grubbs, Teste de Dixon, Teste de Cochran e Teste de Doerffel.

Em estatística, *outlier*, ou valor atípico, é uma observação que apresenta um grande afastamento das demais da série (que está "fora" dela), ou que é inconsistente. A existência de *outliers* implica, tipicamente, em prejuízos a interpretação dos resultados dos testes estatísticos aplicados as amostras. Em suma, um *outlier* é um fato que desvia tanto de outros fatos a ponto de gerar suspeitas de que foi gerado por um mecanismo diferente.

A determinação de *outliers*, por exemplo, pode indicar processos geoquímicos raros (como mineralizações), que pode ser usado para exploração de hidrocarbonetos como abordado por Bedregal, 2008. De um modo geral eliminam-se os *outliers* quando eles representam erros óbvios, mas frequentemente eles representam anomalias interessantes que merecem um estudo mais detalhado. Na verdade para alguns conjuntos de dados os *outliers* são a característica mais importante. A identificação de valores pertencentes a um conjunto de dados que possam ser caracterizados como *outliers*, bem como o tratamento que se deve dar a eles é tema importante no tratamento estatístico de dados. A regressão linear pode ser usada por exemplo para a determinação do gradiente térmico (variação da temperatura com a profundidade). É de grande importância para a determinação de temperaturas em subsuperfície (leia-se rochas reservatório e geradora).

1.2 Objetivo

O objetivo geral do *software* é propor uma série de tratamentos estatísticos como a regressão linear e os testes de hipóteses, bem como alguns testes que identificam valores anômalos (*outliers*). Este trabalho implementa alguns testes que identificam valores anômalos (*outliers*), que são: Teste do escore z modificado, Teste de Grubbs, Teste de Dixon, Teste de Cochran e Teste de Doerffel, além de testes de hipóteses e regressão linear que são aplicados em dados geoquímicos.

1.3 Organização do Documento

O presente documento está organizado como uma apostila, contendo a Especificação, a Elaboração onde também uma teoria básica, para melhor compreensão do programa, e o passo a passo da utilização do mesmo. , a Análise Orientada a Objeto (AOO), a Implementação, os Testes e a Documentação.

Capítulo 2

Especificação

Apresenta-se neste capítulo a especificação do sistema a ser modelado e desenvolvido.

2.1 Especificação do programa - descrição dos requisitos

A modelagem foi criada com a intenção de propor um tratamento estatístico de dados geoquímicos a partir da regressão linear, testes de hipótese e identificação de valores anômalos (*outliers*).

2.1.1 Definições da interface

Devido a sua simplicidade, o software terá interface no modo texto. Além disso, deve ser multiplataforma, ou seja, deve ser executado em GNU/LINUX, UNIX, Mac Os e Windows.

2.1.2 Entrada e saída de dados

A entrada e a saída de dados irá depender do tipo de teste selecionado. Para melhor observá-los indica-se a análise dos diagramas. A entrada de dados poderá ser feita via arquivo de disco ou via teclado.

2.2 Casos de uso do Programa

A Tabela 2.1 mostra a descrição do caso de uso do programa. Estão descritas todas as etapas realizadas no funcionamento do programa. Observe que se o usuário escolher uma opção não correspondente a nenhuma das alternativas para saída do programa (1-Selecionar o teste estatístico, 2-Fornecer os dados geoquímicos, 3-Aplicar o teste, e 4-Analisar resultados), um exemplo de cenário alternativo, a frase “Opção inválida” irá aparecer na tela, e o usuário deverá selecionar outra opção.

Tabela 2.1: Casos de uso do programa

Nome do caso de uso:	Tratamento estatístico de dados geoquímicos.
Resumo/descrição:	Escolher um método estatístico para tratar dados geoquímicos.
Etapas:	1. Selecionar o tratamento estatístico. 2. Fornecer dados geoquímicos. 3. Aplicar o tratamento. 4. Analisar resultados.
Cenário Alternativo	Um cenário alternativo envolve uma entrada errada do usuário, escolhendo um número acima de 4 (que não representa nenhuma das opções de saída do programa). Neste caso, a frase “Opção inválida” aparece na tela.

2.3 Diagrama de caso de uso geral do programa

O diagrama de caso de uso geral do programa é apresentado na Figura 2.1. Observe que o ator (o usuário) deve escolher um teste estatístico, e em seguida fornecer os dados que deseja analisar, que será a entrada do programa, contendo os informações geoquímicas, como o exemplo das percentagens de SAT, ARO e NSO. Posteriormente, o usuário deverá então selecionar a opção de aplicar o teste. De posse dos resultados, os mesmos serão analisados e interpretados.

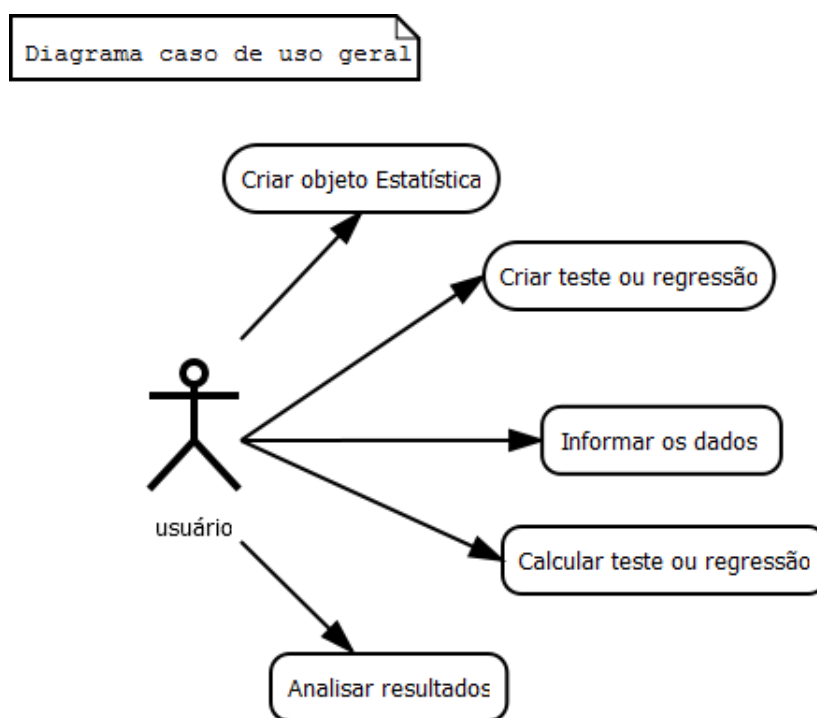


Figura 2.1: Diagrama de caso de uso – Caso de uso geral

2.4 Diagrama de caso de uso específico do programa

Apresenta-se na Figura 2.2 um diagrama de caso de uso específico do programa. Nele está detalhado o caso de uso específico que é “Testes” onde um teste deverá ser escolhido para tratar os dados geoquímicos. O usuário cria um objeto “Teste” e informa ao sistema os dados geoquímicos e o nível de significância, posteriormente é solicitada a realização do teste através do caso de uso “Realizar Teste”. Os resultados são então gerados e informados ao usuário para que o mesmo possa analisá-lo.

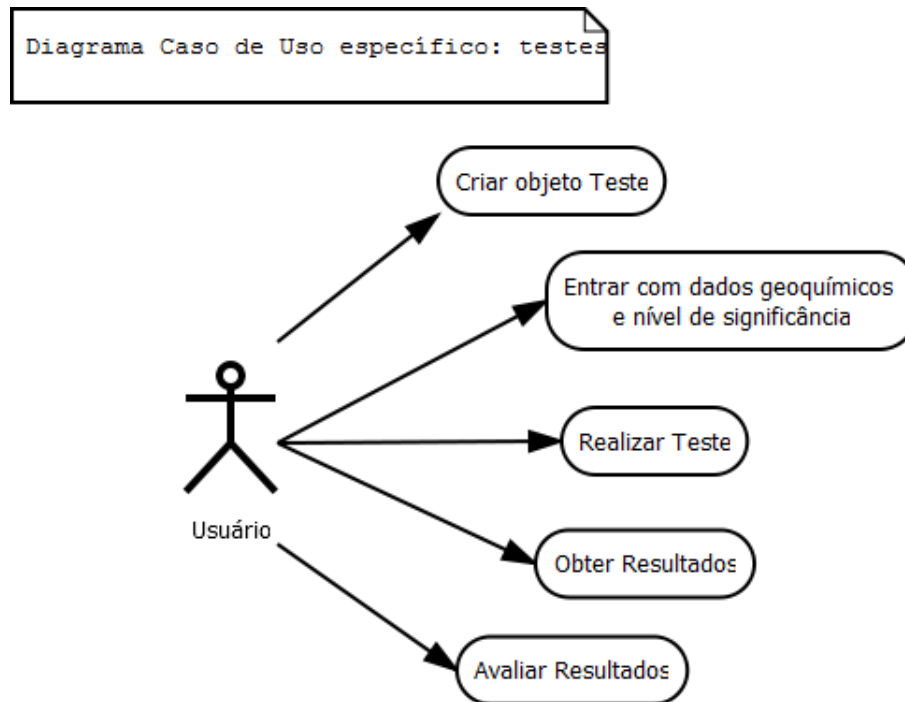


Figura 2.2: Diagrama de caso de uso específico – Detalhamento dos testes a serem utilizados

Já para o caso de uso específico “Regressão Linear”, o objeto é criado e o usuário deve fornecer os valores de x e y . O sistema então retorna o coeficiente de correlação e a regressão linear ao usuário, o que pode ser melhor visualizado através de um gráfico plotado no `gnuplot`. Com isso, o usuário pode então analisar os resultados. O diagrama de caso de uso específico “Regressão Linear” é apresentado na figura 2.3.

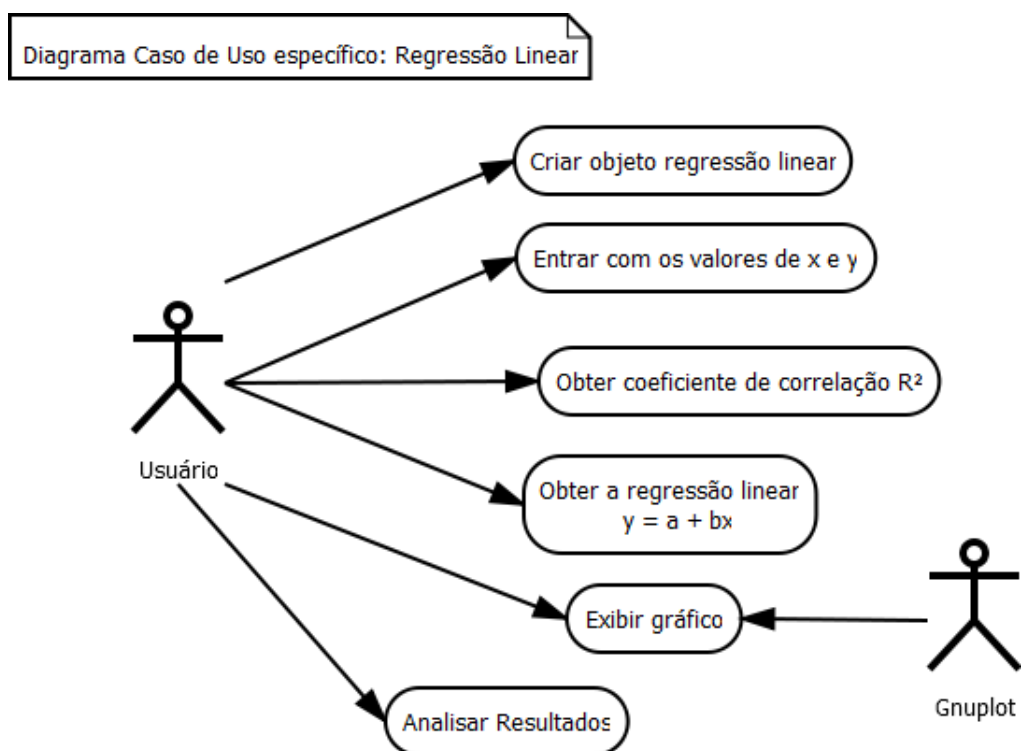


Figura 2.3: Diagrama de caso de uso específico – Detalhamento da regressão linear

Capítulo 3

Elaboração

Neste capítulo é apresentada a elaboração, que envolve o estudo de conceitos relacionados ao programa desenvolvido, a análise de domínio e a identificação de pacotes.

3.1 Revisão de conceitos básicos de estatística

Apresenta-se a seguir uma revisão de estatística que tem como base as referências [6, 10][6, 10].

3.1.1 CONCEITOS ESTATÍSTICOS BÁSICOS

- População: Qualquer coleção de indivíduos ou valores, finita ou infinita.
- Amostra: Uma parte da população, normalmente selecionada com o objetivo de fazer inferências sobre a população.
- Amostra representativa: Apresenta as características relevantes da população na mesma proporção em que elas ocorrem na própria população.
- Amostra aleatória: Amostra de N valores ou indivíduos obtida de tal forma que todos os possíveis conjuntos de N valores na população tenham a mesma chance de ser escolhidos.
- Frequência: Número de elementos num dado intervalo de interesse dividido pelo número total de elementos.
- Média amostral:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (3.1)$$

onde :

x_i = i-ésimo valor

N = Número total de valores na amostra

- Desvio:

$$d_i = x_i - \bar{x} \quad (3.2)$$

- Variância amostral:

$$S^2 = \frac{1}{N-1} \sum_{i=1}^N d_i^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2 \quad (3.3)$$

- Desvio padrão amostral

$$S = \sqrt{\frac{1}{N-1} \sum_{i=1}^N d_i^2} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (3.4)$$

3.1.2 REGRESSÃO LINEAR

Só se deve utilizar a regressão se a correlação entre as variáveis for significativa. A *correlação* mede a força ou grau de relacionamento entre duas variáveis, e a *regressão* fornece uma equação que descreve o relacionamento entre elas em termos matemáticos. O ideal seria a previsão, em função da relação existente, dos valores exatos de uma variável, mas o que se consegue é apenas prever valores médios, ou valores esperados.

O método dos mínimos quadrados é um método de ajuste de pontos a uma reta, e se baseia em que a reta resultante do ajuste seja tal que a soma dos quadrados das distâncias verticais dos pontos à reta seja mínima; esta é a reta que recebe o nome de reta dos mínimos quadrados, reta de regressão ou reta de regressão estimada, sendo os valores de a e b da equação da reta $y = a + bx$ estimados com base em dados amostrais.

O método dos mínimos quadrados minimiza a soma dos quadrados dos resíduos, ou seja, minimiza $\sum_{i=1}^n e_i^2$.

A ideia por trás do método é que, minimizando a soma do quadrado dos resíduos, encontraremos a e b que trarão a menor diferença entre a previsão de y e o y realmente observado.

Sendo assim, dado um conjunto de pares ordenados x_i, y_i , pode-se relacionar x_i e y_i utilizando-se um modelo de regressão linear bastante simples, a regressão linear. A mesma pode ser escrita da seguinte forma:

$$y_i = a + bx_i + \varepsilon_i \quad (3.5)$$

sendo a e b os coeficientes da equação linear e ε_i o erro. Note que

$$E\{\varepsilon_i\} = \sigma^2 \quad (3.6)$$

Pode-se determinar os coeficientes a e b utilizando-se as equações:

$$b = \frac{n \sum xy - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} \quad (3.7)$$

$$a = \frac{(\sum x^2)(\sum y) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2} = \frac{\sum y}{n} - b \frac{\sum x}{n} \quad (3.8)$$

Um coeficiente r^2 , conhecido como coeficiente de ajuste da regressão, pode ser determinado utilizando-se a equação abaixo:

$$r^2 = \frac{\sum \bar{y}}{\sum y^2} = b^2 \frac{\sum x^2}{\sum y^2} = \frac{(\sum xy)^2}{(\sum x^2)(\sum y^2)} \quad (3.9)$$

Veja na Figura 4.10 o diagrama de atividades da regressão linear.

3.1.3 DISTRIBUIÇÃO NORMAL

Uma distribuição estatística é uma função que descreve o comportamento de uma variável aleatória. Uma variável aleatória é uma grandeza que pode assumir qualquer valor dentro do conjunto de valores possíveis para o sistema a que ela se refere, só que cada valor desses tem uma certa probabilidade de ocorrência, governada por uma determinada distribuição de probabilidades. Se tivermos como descobrir ou estimar qual é essa distribuição, poderemos calcular a probabilidade de ocorrência de qualquer valor de interesse.

A distribuição normal é uma distribuição contínua, isto é, uma distribuição em que a variável pode assumir qualquer valor dentro de um intervalo previamente definido. Para uma variável normalmente distribuída, o intervalo é $(-\infty, +\infty)$, o que significa que ela pode assumir, pelo menos em princípio, qualquer valor real. Uma distribuição contínua da variável x é definida pela sua densidade de probabilidade $f(x)$.

$$f(x)dx = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(x-\mu)^2}{2\sigma^2} dx \quad (3.10)$$

onde:

$f(x)dx$ = Densidade de probabilidade da variável aleatória x

μ = Média populacional

σ^2 = Variância populacional

Empregaremos a notação $x \approx N(\mu, \sigma^2)$ para indicarmos que x se distribui normalmente com média μ e variância σ^2 . Para a distribuição normal padrão teremos:

$x \approx N(0, 1)$

$$f(x)dx = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{x^2}{2} \quad (3.11)$$

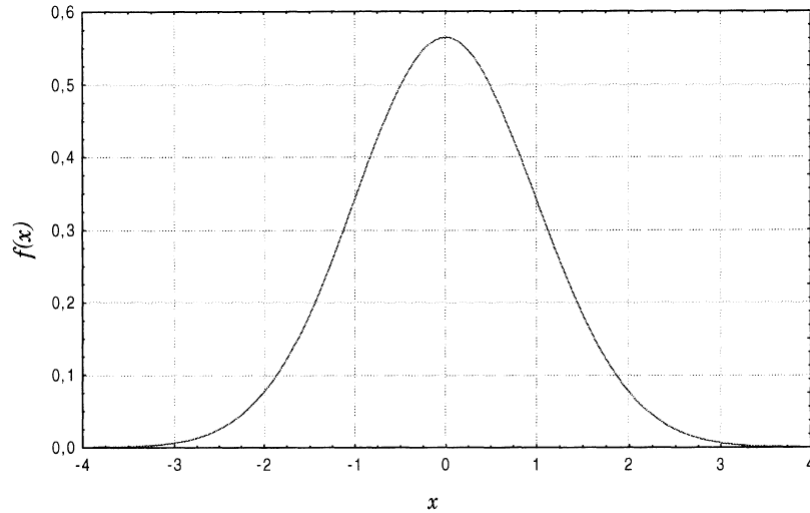


Figura 3.1: Distribuição de freqüências de uma variável aleatória $x \approx N(0, 1)$. Note que x é o afastamento em relação à média (que é zero), em número de desvios padrão

- Variável Normal Padronizada

\mathbf{x} = Variável aleatória com distribuição $N(\mu, \sigma^2)$

\mathbf{z} = Variável aleatória com distribuição $N(0, 1)$

$$z = \frac{x - \mu}{\sigma} \quad (3.12)$$

3.1.4 DISTRIBUIÇÃO T STUDENT

A distribuição t student é uma distribuição de probabilidade teórica. É simétrica, campaniforme, e semelhante à curva normal padrão, porém com caudas mais largas, ou seja, uma simulação da t de Student pode gerar valores mais extremos que uma simulação da normal. O único parâmetro que a diferencia da normal e caracteriza a sua forma é o número de graus de liberdade v . Quanto maior for esse parâmetro, mais próxima da normal ela será.

Suponha que Z tenha a distribuição normal com média 0 e variância 1, que V tenha a distribuição Chi-quadrado com v graus de liberdade, e que Z e V sejam independentes. Então:

$$t = \frac{Z}{\sqrt{\frac{V}{v}}} \quad (3.13)$$

tem a distribuição t de Student com v graus de liberdade.

A função densidade de probabilidade é:

$$f(t) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi}\Gamma(\frac{v}{2})} \left(1 + \frac{t^2}{v}\right)^{-\frac{(v+1)}{2}} \quad (3.14)$$

em que Γ^1 é a função gama. Usando-se a função beta β^2 , a função densidade de probabilidade pode ser escrita como:

$$f(t) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi}\beta(\frac{1}{2}, \frac{v}{2})} \left(1 + \frac{t^2}{v}\right)^{-\frac{(v+1)}{2}} \quad (3.15)$$

A distribuição t de Student aparece naturalmente no problema de se determinar a média de uma população (que segue a distribuição normal) a partir de uma amostra. Neste problema, não se sabe qual é a média ou o desvio padrão da população, mas ela deve ser normal.

Supondo que o tamanho da amostra n seja muito menor que o tamanho da população, temos que a amostra é dada por n variáveis aleatórias normais independentes X_1, \dots, X_n , cuja média $\bar{X}_n = (X_1 + \dots + X_n)/n$ é o melhor estimador para a média da população.

Considerando $S_n^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_n)^2$ como a variância amostral, temos o seguinte resultado:

A variável aleatória t dada por:

$$t = \frac{X_n - \mu}{S_n/\sqrt{n}} \quad (3.16)$$

ou

$$t = \sqrt{n} \frac{\bar{X}_n - \mu}{S_n} \quad (3.17)$$

segue uma distribuição t de Student com $v = n - 1$ graus de liberdade.

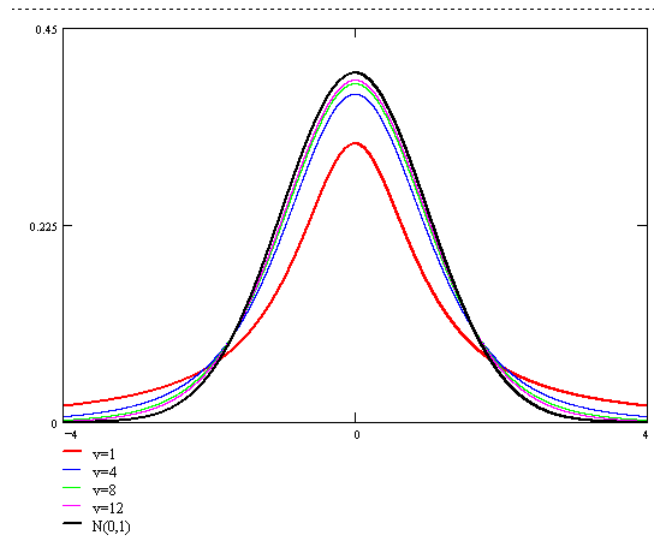


Figura 3.2: A função densidade da distribuição de Student para alguns valores de v e da distribuição normal

¹Em Matemática, a função Γ é uma função definida por $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$

²Em Matemática, a função β , também chamada de integral de Euler de primeiro tipo, é a função definida por: $\beta(x, y) = \int_0^1 t^{x-1} (1-t)^{y-1} dt$

3.1.5 TESTE DE HIPÓTESES

Em estatística, um Teste de Hipóteses³ é um método usado para verificar se os dados são compatíveis com alguma hipótese, podendo, muitas vezes, sugerir a não-validade de uma hipótese. O teste de hipóteses é um procedimento estatístico baseado na análise de uma amostra (através da teoria de probabilidades), sendo usado para avaliar parâmetros que são desconhecidos numa população, ou seja, avaliar afirmações sobre os valores de parâmetros.

Um Teste de Hipóteses pode ser paramétrico ou não-paramétrico. Testes paramétricos são baseados em parâmetros da amostra, por exemplo média e desvio padrão. O uso tanto dos testes paramétricos como dos não-paramétricos está condicionado à dimensão da amostra e à respectiva distribuição da variável em estudo.

Os testes de hipóteses são sempre constituídos por duas hipóteses, a hipótese nula H_0 e a hipótese alternativa H_1 .

- Hipótese nula (H_0) : é a hipótese que traduz a ausência do efeito que se quer verificar.
- Hipótese alternativa (H_1) : é a hipótese que o investigador quer verificar.
- Nível de significância: a probabilidade de rejeitar a hipótese nula quando ela é efetivamente verdadeira (ERRO).

Finalidade: avaliar afirmações sobre os valores de parâmetros.

O valor-p é uma estatística muito utilizada para sintetizar o resultado de um teste de hipóteses. Formalmente, o valor-p é definido como a probabilidade de se obter uma estatística de teste igual ou mais extrema quanto aquela observada em uma amostra, assumindo verdadeira a hipótese nula.

3.1.6 TESTE DE HIPÓTESES MÉDIA

O teste consiste em verificar, através de uma amostra, se a média da população atende o caso em teste (conforme desejemos testar diferença, valor inferior ou valor superior a uma referência para a média), para um certo nível de significância desejado.

Inicialmente devemos calcular:

$$Z_{calc} = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}} \quad (3.18)$$

onde:

μ = média esperada da população

\bar{x} = média da amostra

σ = desvio padrão da população

n = tamanho da amostra

³A expressão teste de significância foi criada por Ronald Fisher: "*Critical tests of this kind may be called tests of significance, and when such tests are available we may discover whether a second sample is or is not significantly different from the first.*"

Em seguida consultamos na tabela da curva normal o Z correspondente a cada caso.

Finalmente verificamos se Z_{calc} se encontra na área de rejeição conforme o caso em teste.

- Caso 1 - Unilateral ou unicaudal à esquerda

$$H_0 : \mu = \mu_0$$

$$H_1 : \mu < \mu_0$$

Rejeitar se:

$$Z_{calc} < -Z_\alpha$$

- Caso 2 - Unilateral ou unicaudal à direita

$$H_0 : \mu = \mu_0$$

$$H_1 : \mu > \mu_0$$

Rejeitar se:

$$Z_{calc} > Z_\alpha$$

- Caso 3 - Bilateral

$$H_0 : \mu = \mu_0$$

$$H_1 : \mu \neq \mu_0$$

Rejeitar se:

$$Z_{calc} < -Z_{\alpha/2}$$

$$Z_{calc} > Z_{\alpha/2}$$

onde Z_α é o valor crítico tabelado para um nível de significância α e μ_0 é o valor da hipótese a ser testada.

Para o caso onde o desvio padrão da população é desconhecido, devemos utilizar a fórmula 3.19.

$$t_{calc} = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}} \quad (3.19)$$

onde:

s =desvio padrão da amostra

Em seguida, consultamos a tabela da distribuição t de Student para encontrar o t correspondente.

Finalmente verificamos se t_{calc} se encontra na área de rejeição conforme o caso em teste.

- Caso 1 - Unilateral ou unicaudal à esquerda

$$H_0 : \mu = \mu_0$$

$$H_1 : \mu < \mu_0$$

Rejeitar se:

$$t_{calc} < -t_\alpha$$

- Caso 2 - Unilateral ou unicaudal à direita

$$H_0 : \mu = \mu_0$$

$$H_1 : \mu > \mu_0$$

Rejeitar se:

$$t_{calc} > t_\alpha$$

- Caso 3 - Bilateral

$$H_0 : \mu = \mu_0$$

$$H_1 : \mu \neq \mu_0$$

Rejeitar se:

$$t_{calc} < -t_{\alpha/2}$$

$$t_{calc} > t_{\alpha/2}$$

onde t_α é o valor crítico tabelado para um nível de significância α e μ_0 é o valor da hipótese a ser testada.

Vea nas Figuras 4.16 e 4.17 os diagramas de atividades dos testes de hipótese.

3.1.7 OUTLIERS

O software também utiliza os testes estatísticos para identificar os *outliers* (anomalias) que estão relacionados a alguns dados experimentais do laboratório de geoquímica.

As observações que apresentam um grande afastamento das restantes ou são inconsistentes com elas são habitualmente designadas por *outliers*. Estas observações são também designadas por observações “anormais”, contaminantes, estranhas, extremas ou aberrantes.

A preocupação com observações *outliers* é antiga e data das primeiras tentativas de analisar um conjunto de dados. Inicialmente pensava-se que a melhor forma de lidar com este tipo de observações seria através da sua eliminação da análise. As opiniões não eram unânimes: uns defendiam a rejeição das observações “inconsistentes com as restantes”, enquanto outros afirmavam que as observações nunca deveriam ser rejeitadas simplesmente por parecerem inconsistentes com os restantes dados e que todas as observações deviam contribuir com igual peso para o resultado final.

Antes de decidir o que deverá ser feito às observações *outliers* é conveniente ter conhecimento das causas que levam ao seu aparecimento. Em muitos casos as razões da sua

existência determinam as formas como devem ser tratadas. Assim, as principais causas que levam ao aparecimento de *outliers* são:

- Erros de medição;
- Erros de execução;
- Variabilidade inerente dos elementos da população.

O estudo de outliers, independentemente da(s) sua(s) causa(s), pode ser realizado em várias fases: A fase inicial é a identificação das observações que são potencialmente aberrantes. A identificação de *outliers* consiste na detecção, com métodos subjectivos, das observações surpreendentes. A identificação é feita, geralmente, por análise gráfica ou, no caso de um número de dados ser pequeno, por observação direta dos mesmos. São assim identificadas as observações que têm possibilidades de virem a ser designadas por *outliers*.

Na segunda fase, tem-se como objetivo a eliminação da subjetividade inerente à fase anterior. Pretende-se saber se as observações identificadas como *outliers* potenciais o são, efetivamente. São efetuados testes à ou às observações “preocupantes”. Devem ser escolhidos os testes mais adequados para a situação em estudo. As observações suspeitas são testadas quanto à sua discordância. Se for aceite a hipótese de algumas observações serem *outliers*, elas podem ser designadas como discordantes. Uma observação diz-se discordante se puder considerar-se inconsistente com os restantes valores depois da aplicação de um critério estatístico objetivo. Muitas vezes o termo discordante é usado como sinónimo de *outlier*.

Na última fase é necessário decidir o que fazer com as observações discordantes. A maneira mais simples de lidar com essas observações é eliminá-las. Como já foi dito, esta abordagem, apesar de ser muito utilizada, não é aconselhável. Ela só se justifica no caso de os *outliers* serem devidos a erros cuja correção é inviável. Caso contrário, as observações consideradas como *outliers* devem ser tratadas cuidadosamente pois contêm informação relevante sobre características subjacentes aos dados e poderão ser decisivas no conhecimento da população à qual pertence a amostra em estudo. Os testes que normalmente são utilizados para a identificação de *outliers* na geoquímica serão descritos abaixo.

3.1.8 TESTE DO ESCORE Z MODIFICADO

Este teste tem sido usado de forma mais extensiva que o teste que considera como *outlier* simplesmente os valores que superam a soma da média aritmética com três desvios padrão, ou a média menos três desvios padrão, pois que tanto a média como o desvio padrão são, já, afetados pela presença do *outlier*. O teste do escore z modificado usa estimadores robustos, como a mediana, o que garante não terem sido os valores utilizados para definir um *outlier* afetados pelo mesmo.

Exemplo: Montaremos uma verificação da presença de um *outlier* com este teste

(Tabela 3.1).

Tabela 3.1: Exemplo da verificação da presença de um *outlier*

Dado Original (X_i)	$ X_i - X_m $	Z_i
3.2	0.1	-0.34
3.3	0.0	0.00
8.1	4.8	16.19
3.2	0.1	-0.34
2.9	0.4	-1.35
3.7	0.4	1.35
3.1	0.2	-0.67
3.5	0.2	0.67
3.3	0.0	0.00
9.2	5.9	19.90

1. Calcular a mediana dos dados brutos, que vale 3,3.
2. Determinar a coluna com os valores dos desvios absolutos, definida por $|X_i - X_m|$.
3. Determinar a média aritmética dos desvios absolutos (MAD), valores que constam da coluna criada no passo anterior, que vale 0,2 neste caso.
4. Calcular os valores de z modificado para cada observação, gerando a coluna três da tabela anterior; este valor é representado por $z * i$, que vale $z * i = 0,6745(X_i - X_m)/MAD$. Para a terceira observação se tem $z * i = 0,6745(8,1 - 3,3)/0,2 = 16,19$; Para a quarta observação se tem $z * i = 0,6745(3,2 - 3,3)/0,2 = -0,34$; Para a décima observação se tem $z * i = 0,6745(9,2 - 3,3)/0,2 = 19,90$;
5. Considerar como *outliers* valores de $|z * i| > 3,5$, ou seja, no caso estudado são considerados outliers os valores relativos a 16,19 e 19,90 da terceira coluna.

Veja na Figura 4.11 o diagrama de atividades para o teste Z modificado.

3.1.9 TESTE DE GRUBBS

Este teste é utilizado para dados que seguem a distribuição lognormal. O teste de Grubbs é apresentado com um exemplo prático. É um teste definido como sendo útil principalmente para testar variabilidade entre laboratórios.

Exemplo:

Tabela 3.2: Exemplificando o teste de Grubbs

Dado Original (Xi)	Ln (Xi)	Com rank
2.15	0.77	0.77
11.76	2.46	1.14
5.08	1.63	1.63
3.12	1.14	2.19
12.87	2.55	2.46
32.13	3.47	2.55
219	5.39	2.98
19.69	2.98	3.47
179	5.19	3.87
9609	9.17	4.31
327	5.79	4.62
74.2	4.31	5.19
102	4.62	5.39
47.8	3.87	5.79
8.97	2.19	9.17

1. Calcular a média e o desvio padrão dos dados já transformados em logaritmos naturais, respectivamente 3,70 e 2,17.
2. Colocar os dados logtransformados em ordem crescente, com *rank*.
3. Se houver suspeita de *outlier* para o menor valor se faz.

4.

$$\tau = \frac{Media - X_1}{S} \quad (3.20)$$

5. se houver suspeita de *outlier* para o maior valor se faz

6.

$$\tau = \frac{X_n - Media}{S} \quad (3.21)$$

7. No presente exemplo, suspeitando-se do maior valor se tem

8.

$$\tau_{15} = \frac{9,17 - 3,70}{2,17} = 2,52 \quad (3.22)$$

9. Para um $\alpha = 0,05$ se determina o τ crítico para $n = 15$, no caso 2,409.
10. Se o valor calculado for maior que o crítico se rejeita a hipótese nula e se conclui que o dado testado é um outlier; no caso presente, se rejeita a hipótese nula e o valor testado é um outlier.
11. A Tabela 3.3 contém os valores críticos para o teste de Grubbs, com α valendo 0,10, 0,05, 0,025, 0,01 e 0,005, unicaudais, ao se usar teste bicaudal se deve adotar

a mesma tabela com o dobro das probabilidades α ; esta tabela tem incrementos unitários para tamanhos de amostra entre 3 e 40 observações e incrementos de 10 unidades entre amostras com 40 a 140 observações.

Tabela 3.3: Valores críticos de Grubbs

$n \backslash \alpha$	0.10	0.05	0.025	0.01	0.005
3	1.148	1.153	1.155	1.155	1.155
4	1.425	1.463	1.481	1.492	1.496
5	1.602	1.672	1.715	1.749	1.764
6	1.729	1.822	1.887	1.944	1.973
7	1.828	1.938	2.020	2.097	2.139
8	1.909	2.032	2.126	2.221	2.274
9	1.977	2.110	2.215	2.323	2.387
10	2.036	2.176	2.290	2.410	2.482
11	2.088	2.234	2.355	2.485	2.564
12	2.134	2.285	2.412	2.550	2.636
13	2.175	2.331	2.462	2.607	2.699
14	2.213	2.371	2.507	2.659	2.755
15	2.247	2.409	2.549	2.705	2.806
16	2.279	2.443	2.585	2.747	2.852
17	2.309	2.475	2.620	2.785	2.894
18	2.335	2.504	2.651	2.821	2.932
19	2.361	2.532	2.681	2.854	2.968
20	2.385	2.557	2.709	2.884	3.001
21	2.408	2.580	2.733	2.912	3.031
22	2.429	2.603	2.758	2.939	3.060
23	2.448	2.624	2.781	2.963	3.087
24	2.467	2.644	2.802	2.987	3.112
25	2.486	2.663	2.822	3.009	3.135
26	2.502	2.681	2.841	3.029	3.157
27	2.519	2.698	2.859	3.049	3.178
28	2.534	2.714	2.876	3.068	3.199
29	2.549	2.730	2.893	3.085	3.218
30	2.563	2.745	2.908	3.103	3.236
31	2.577	2.759	2.924	3.119	3.253
32	2.591	2.773	2.938	3.135	3.270
33	2.604	2.786	2.952	3.150	3.286
34	2.616	2.799	2.965	3.164	3.301
35	2.628	2.811	2.979	3.178	3.316
36	2.639	2.823	2.991	3.191	3.330
37	2.650	2.835	3.003	3.204	3.343
38	2.661	2.846	3.014	3.216	3.356
39	2.671	2.857	3.025	3.228	3.369
40	2.682	2.866	3.036	3.240	3.381
50	2.768	2.956	3.128	3.336	3.483
60	2.837	3.025	3.199	3.411	3.560
70	2.893	3.082	3.257	3.471	3.622
80	2.940	3.130	3.305	3.521	3.673
90	2.981	3.171	3.347	3.563	3.716
100	3.017	3.207	3.383	3.600	3.754
110	3.049	3.239	3.415	3.632	3.787
120	3.078	3.267	3.444	3.662	3.817
130	3.104	3.294	3.470	3.688	3.843
140	3.129	3.318	3.493	3.712	3.867

Veja na Figura 4.13 o diagrama de atividades do teste de Grubbs.

3.1.10 TESTE DE DIXON

O teste de Dixon para valores extremos atenta para a diferença entre os valores máximo e mínimo e seus valores vizinhos, é gerada uma razão r à qual é atribuída uma certa distribuição. O teste de Dixon é usado mais comumente na detecção de pequenas quantidades de *outliers*, é recomendado quando o número de observações está entre 3 e 25; os dados são ordenados de modo crescente e uma estatística é computada para o maior ou menor valor, suspeito de ser um *outlier*. Depois de estabelecido um nível de significância, o valor é comparado com um valor da tabela, se for menor que certo valor crítico a hipótese nula não é rejeitada, ou seja, aceita-se a hipótese de não existência de *outliers*, se a hipótese nula for rejeitada (valor calculado maior que o valor crítico) se conclui que o valor testado é um *outlier*. Para testar a existência de outros *outliers* se repete o teste, mas o poder deste teste diminui à medida que o número de repetições do mesmo aumenta. Alguns autores citam que o teste de Dixon não é mais recomendado por haver melhores opções disponíveis.

Exemplo:

1. Os dados devem ser ordenados de forma crescente, sendo o menor valor o de ordem 1, e o maior valor o de ordem N
2. Chama-se Z ao valor numérico do dado de ordem N , ou seja, $Z(1)$ é o valor numérico do menor resultado e $Z(N)$ é o valor numérico do resultado de maior valor numérico, $Z(N-1)$ é o valor do penúltimo dado em ordem crescente de valor numérico; ao se proceder ao teste Q se chama QM ao valor mais elevado (suspeito de ser *outlier*) e Qm ao valor menor (suspeito de ser *outlier*).
3. Procede-se ao teste de Dixon, de acordo com três situações:
4. havendo entre 3 e 7 observações

5.

$$QM = [Z(N) - Z(N-1)] / [Z(N) - Z(1)] \quad (3.23)$$

6.

$$Qm = [Z(2) - Z(1)] / [Z(N) - Z(1)] \quad (3.24)$$

7. havendo entre 8 e 12 observações

8.

$$QM = [Z(N) - Z(N-1)] / [Z(N) - Z(2)] \quad (3.25)$$

9.

$$Qm = [Z(2) - Z(1)] / [Z(N-1) - Z(1)] \quad (3.26)$$

10. havendo entre 13 e 25 observações

11.

$$QM = [Z(N)-Z(N-2)]/[Z(N)-Z(3)] \quad (3.27)$$

12.

$$Qm = [Z(3)-Z(1)]/[Z(N-2)-Z(1)] \quad (3.28)$$

Havendo mais de 25 observações o teste não está definido, deve-se buscar outra solução. A Tabela 3.4 apresenta os valores críticos do teste de Dixon para valores de α iguais a 0,10, 0,05 e 0,01 unicaudais, para o caso bicaudal deve-se usar os mesmos valores críticos mas duplicando as probabilidades nos cabeçalhos das colunas. Esta tabela é válida ao se aplicar o teste de Dixon para conjuntos de dados que se ajustem à distribuição normal.

Tabela 3.4: Valores críticos de Dixon

n	$\alpha = 0,10$	$\alpha = 0,05$	$\alpha = 0,01$
3	0,886	0,941	0,988
4	0,679	0,765	0,889
5	0,557	0,642	0,780
6	0,482	0,560	0,698
7	0,434	0,507	0,637
8	0,479	0,554	0,683
9	0,441	0,512	0,635
10	0,409	0,477	0,597
11	0,517	0,576	0,679
12	0,490	0,546	0,642
13	0,467	0,521	0,615
14	0,492	0,546	0,641
15	0,472	0,525	0,616
16	0,454	0,507	0,595
17	0,438	0,490	0,577
18	0,424	0,475	0,561
19	0,412	0,462	0,547
20	0,401	0,450	0,535
21	0,391	0,440	0,524
22	0,382	0,430	0,514
23	0,374	0,421	0,505
24	0,367	0,413	0,497
25	0,360	0,406	0,489

Veja na Figura 4.14 o diagrama de atividades do teste de Dixon.

3.1.11 TESTE DE COCHRAN

O teste de Cochran é definido como sendo um teste para estudar variabilidade interna de um laboratório. O teste de Cochran é definido pela estatística C , que vale

$$C = S^2_{max} / \sum_{i=1}^p S_i^2 \quad (3.29)$$

onde S_{max} é o desvio padrão máximo no conjunto; a hipótese nula parte do princípio que a estatística C tem uma distribuição aproximada à de *Qui quadrado* com $(m-1)$ graus de liberdade, onde m representa o número de variáveis.

O teste de Cochran é afetado pela não normalidade dos dados, e usa uma tabela específica, a tabela de Cochran.

O teste de Cochran é uma variante do teste t (de Student, que compara conjuntos cujas variabilidades não sejam muito diferentes entre si), quando as amostras apresentam diferenças de variabilidade, verificada por um teste F.

Resumidamente o teste de Cochran exige a ordenação crescente para cada conjunto de duas repetições, aplicar a equação 3.29 e comparar o valor obtido com o valor tabelado para este teste, se o valor for menor que o tabelado não há dispersão, se for maior que o tabelado se diz haver dispersão quanto à amplitude.

Exemplo:

Tabela 3.5: Exemplificando o teste de Cochran

	A	B	C	D
Média	2.75	3.50	6.25	9.00
Variância	2.214	0.857	1.071	1.714
Desvio Padrão	1.488	0.926	1.035	1.309

O valor de C será $2,214/5,856 = 0,378$ com 4 grupos e 7 graus de liberdade, que são respectivamente k e $(n-1)$, quatro conjuntos e cada conjunto com oito valores. O valor crítico para C , considerado um α igual a 0,05, é de 0,5365. Em conclusão, se rejeita a hipótese nula de que as variâncias sejam iguais. Os valores críticos de Cochran estão nas Tabelas 3.6 e 3.7.

Tabela 3.6: Valores críticos do teste de Cochran para $\alpha = 0.05$ **$\alpha = 0,05$**

Nº grupos	2	3	4	5	6	7	8	9	10	12	15	20	24	30	40	60	120
GL																	
1	9985	9669	9065	8412	7808	7271	6798	6385	6020	5410	4709	3894	3434	2929	2370	1737	0998
2	9750	8709	7679	6838	6161	5612	5157	4775	4450	3924	3346	2705	2354	1980	1567	1131	0632
3	9392	7977	6841	5981	5321	4800	4377	4027	3733	3264	2758	2205	1907	1593	1259	0895	0495
4	9057	7457	6287	5441	4803	4307	3910	3584	3311	2880	2419	1921	1656	1377	1082	0765	0419
5	8772	7071	5895	5065	4447	3974	3595	3286	3029	2624	2195	1735	1493	1237	0968	0682	0371
6	8534	6771	5598	4783	4184	3726	3362	3067	2823	2439	2034	1602	1374	1137	0887	0623	0337
7	8332	6530	5365	4564	3980	3535	3185	2901	2666	2299	1911	1501	1286	1061	0827	0583	0312
8	8159	6333	5175	4387	3817	3384	3043	2768	2541	2187	1815	1422	1216	1002	0780	0552	0292
9	8010	6167	5017	4241	3682	3259	2926	2659	2439	2098	1736	1357	1160	0958	0745	0520	0279
10	7880	6025	4884	4118	3568	3154	2829	2568	2353	2020	1671	1303	1113	0921	0713	0497	0266
16	7341	5466	4366	3645	3135	2756	2462	2226	2032	1737	1429	1108	0942	0771	0595	0411	0218
36	6602	4748	3720	3066	2612	2278	2022	1820	1655	1403	1144	0879	0743	0604	0462	0316	0165
144	5813	4031	3093	2513	2119	1833	1616	1446	1308	1100	0889	0675	0567	0457	0347	0234	0120
∞	5000	3333	2500	2000	1667	1429	1250	1111	1000	0833	0667	0500	0417	0333	0250	0167	0083

Tabela 3.7: Valores críticos do teste de Cochran para $\alpha = 0.01$ **$\alpha = 0,01$**

Nº grupos	2	3	4	5	6	7	8	9	10	12	15	20	24	30	40	60	120
GL																	
1	9999	9933	9676	9279	8828	8376	7945	7544	7175	6528	5747	4799	4247	3632	2940	2151	1225
2	9950	9423	8643	7885	7218	6644	6152	5727	5358	4751	4069	3297	2821	2412	1915	1371	0759
3	9794	8831	7814	6957	6258	5685	5209	4810	4469	3919	3317	2654	2295	1913	1508	1069	0585
4	9586	8335	7212	6329	5635	5080	4627	4251	3934	3428	2882	2288	1970	1635	1281	0902	0489
5	9373	7933	6761	5875	5195	4659	4226	3870	3572	3099	2593	2048	1759	1454	1135	0796	0429
6	9172	7606	6410	5531	4866	4347	3932	3592	3308	2861	2386	1877	1608	1327	1033	0722	0387
7	8988	7335	6129	5259	4608	4105	3704	3378	3106	2680	2228	1748	1495	1232	0957	0668	0357
8	8823	7107	5897	5037	4401	3911	3522	3207	2945	2535	2104	1646	1406	1157	0898	0625	0334
9	8674	6912	5702	4854	4229	3751	3373	3067	2813	2419	2002	1567	1388	1100	0853	0594	0316
10	8539	6743	5536	4697	4084	3616	3248	2950	2704	2320	1918	1501	1283	1054	0816	0567	0302
16	7949	6059	4884	4094	3529	3105	2779	2514	2297	1961	1612	1248	1060	0867	0668	0461	0242
36	7067	5153	4057	3351	2858	2494	2214	1992	1811	1535	1251	0960	0810	0658	0503	0344	0178
144	6062	4230	3251	2644	2229	1929	1700	1521	1376	1157	0934	0709	0595	0480	0363	0245	0125
∞	5000	3333	2500	2000	1667	1429	1250	1111	1000	0833	0667	0500	0417	0333	0250	0167	0083

Estas tabelas contêm os valores críticos (C) do teste de Cochran para homogeneidade

de variâncias de amostras de igual tamanho. Todos os valores das duas tabelas devem ser divididos por 10.000, ou seja, elas contêm apenas a parte decimal, a parte inteira vale sempre zero. Assim, na tabela relativa a $\alpha = 0,05$, para graus de liberdade (GL) valendo 5 e dois grupos o valor tabelado vale 0,8772.

Veja o diagrama de atividades do teste de Cochran na Figura 4.15.

3.1.12 TESTE DE DOERFFEL

É um teste de simples aplicação introduzido por Doerffel em 1967 e confirmado por Dean & Dixon em 1981, e citados por Wellmer (1998). É utilizado para pequenos conjuntos de dados, e é representado basicamente por

$$Q = (X_a - X_r)/R \quad (3.30)$$

em que X_a é o valor que se suspeita seja um *outlier*, X_r é o valor adjacente (mais próximo) dele, R representa a amplitude dos dados (valor máximo – valor mínimo), e Q é o valor do teste. O valor testado será aceito se Q calculado for inferior ao valor tabelado por Doerffel (1967) e por Dean & Dixon (1981), na Tabela 3.8, reproduzida a seguir.

Tabela 3.8: Tabela de Doerffel e Dean & Dixon

n (Tamanho)	Q de Doerffel ($\alpha = 0.05$)	Q de Dean & Dixon ($\alpha = 0.05$)
3	0.97	0.94
4	0.84	0.76
5	0.73	0.64
6	0.64	0.56
7	0.59	0.51
8	0.54	0.47
9	0.51	0.44
10	0.49	0.41

Doerffel havia, anteriormente (em 1962), proposto um método para a detecção de *outliers* baseado em um diagrama, o valor é considerado aberrante por este método se superar a soma $(\mu + S * g)$, sendo que a média aritmética e o desvio padrão devem ser calculados sem o valor suspeito (pois que ele afeta estes valores), e o valor de g pode ser obtido a partir de um diagrama específico para isto, este valor representa o *threshold* de um valor aberrante; este diagrama está representado na Figura 3.3.

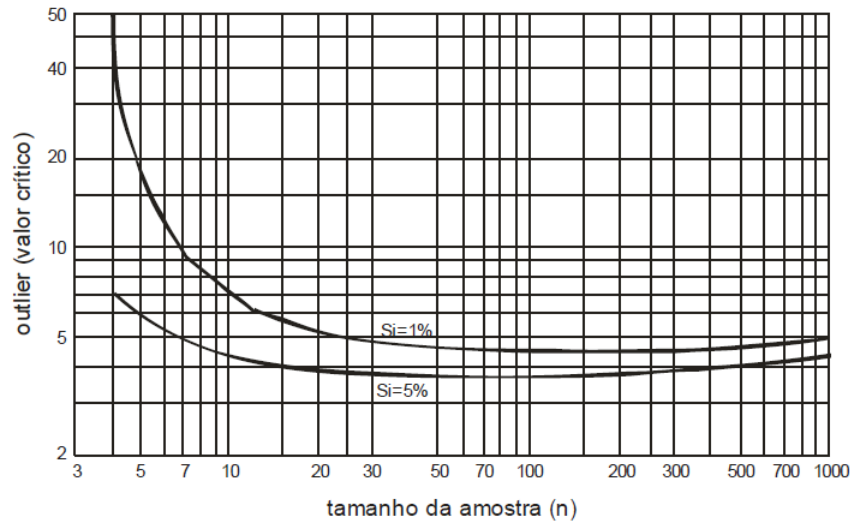


Figura 3.3: Valores críticos de Doerffel

Exemplo:

Wellmer (1998) apresenta um exemplo de aplicação com os valores expressos em WO(%) iguais a 0,8; 1,4; 0,7; 2,4; 4,6; 2,1 e 1,5; sendo o valor 4,6% suspeito de ser um *outlier*. O valor adjacente a ele vale 2,4. A aplicação do teste:

$$Q = (X_a - X_r)/R \quad (3.31)$$

$$(4,6 - 2,4)/(4,6 - 0,7) = 2,2/3,9 = 0,56 \quad (3.32)$$

Escolhendo-se o nível de significância ($Si = 5\%$) se verifica que o valor de Q é menor que o valor de Si correspondente, logo o valor 4,6 é aceitável, ou seja, não deve ser classificado como um *outlier*.

Veja na Figura 4.12 o diagrama de atividades do teste de Doerffel.

3.2 Análise de domínio

- Área relacionada:

A principal área relacionada ao desenvolvimento do *software* é a Estatística, que é a ciência que utiliza-se das teorias probabilísticas para explicar a frequência da ocorrência de eventos, tanto em estudos observacionais quanto em experimento modelar a aleatoriedade e a incerteza de forma a estimar ou possibilitar a previsão de fenômenos futuros, conforme o caso.

- Sub-área relacionada:

A sub-área relacionada é à geoquímica do petróleo. Cabe a esta ciência a identificação de rochas geradoras nas bacias sedimentares, análise de maturação, biodegradação e a avali-

ação do potencial gerador das mesmas. A geoquímica pode ser definida como a aplicação de métodos químicos aos estudos de fenômenos de geologia. No caso particular da geoquímica do petróleo, trata-se da aplicação de métodos da química orgânica à geologia do petróleo. A geoquímica utiliza-se da estatística para analisar resultados de experimentos.

3.3 Identificação de pacotes – assuntos

- Geoquímica: Utilizada para fornecer os dados experimentais.
- Estatística: Utilizada para identificar as anomalias dos dados experimentais.

3.4 Diagrama de pacotes – assuntos

A Figura 3.4 representa um diagrama de pacotes simplificado, para o programa a ser desenvolvido. Nele estão contidos os pacotes (assuntos) descritos anteriormente.

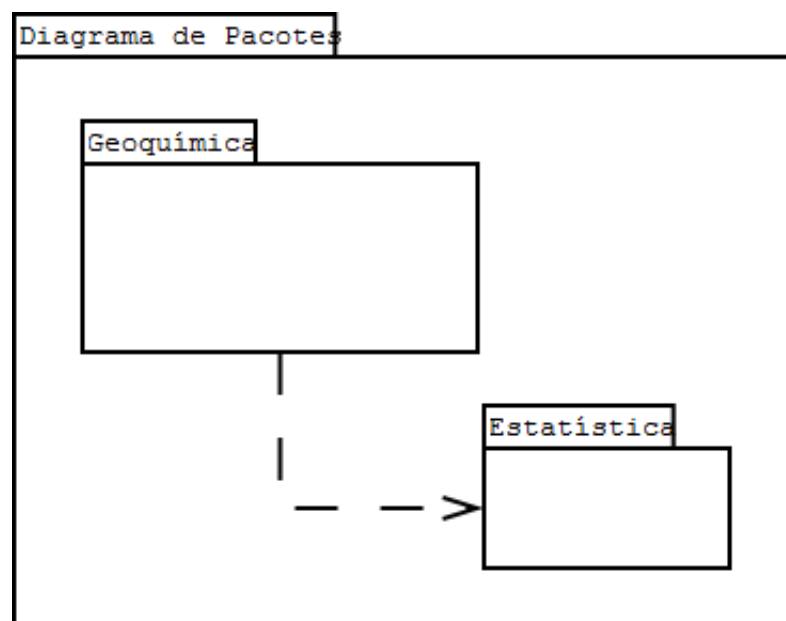


Figura 3.4: Diagrama de Pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um sistema é a AOO – Análise Orientada a Objeto. A AOO utiliza algumas regras para identificar os objetos de interesse, as relações entre os pacotes, as classes, os atributos, os métodos, as heranças, as associações, as agregações, as composições e as dependências.

O modelo de análise deve ser conciso, simplificado e deve mostrar o que deve ser feito, não se preocupando como isso será realizado.

O resultado da análise é um conjunto de diagramas que identificam os objetos e seus relacionamentos.

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

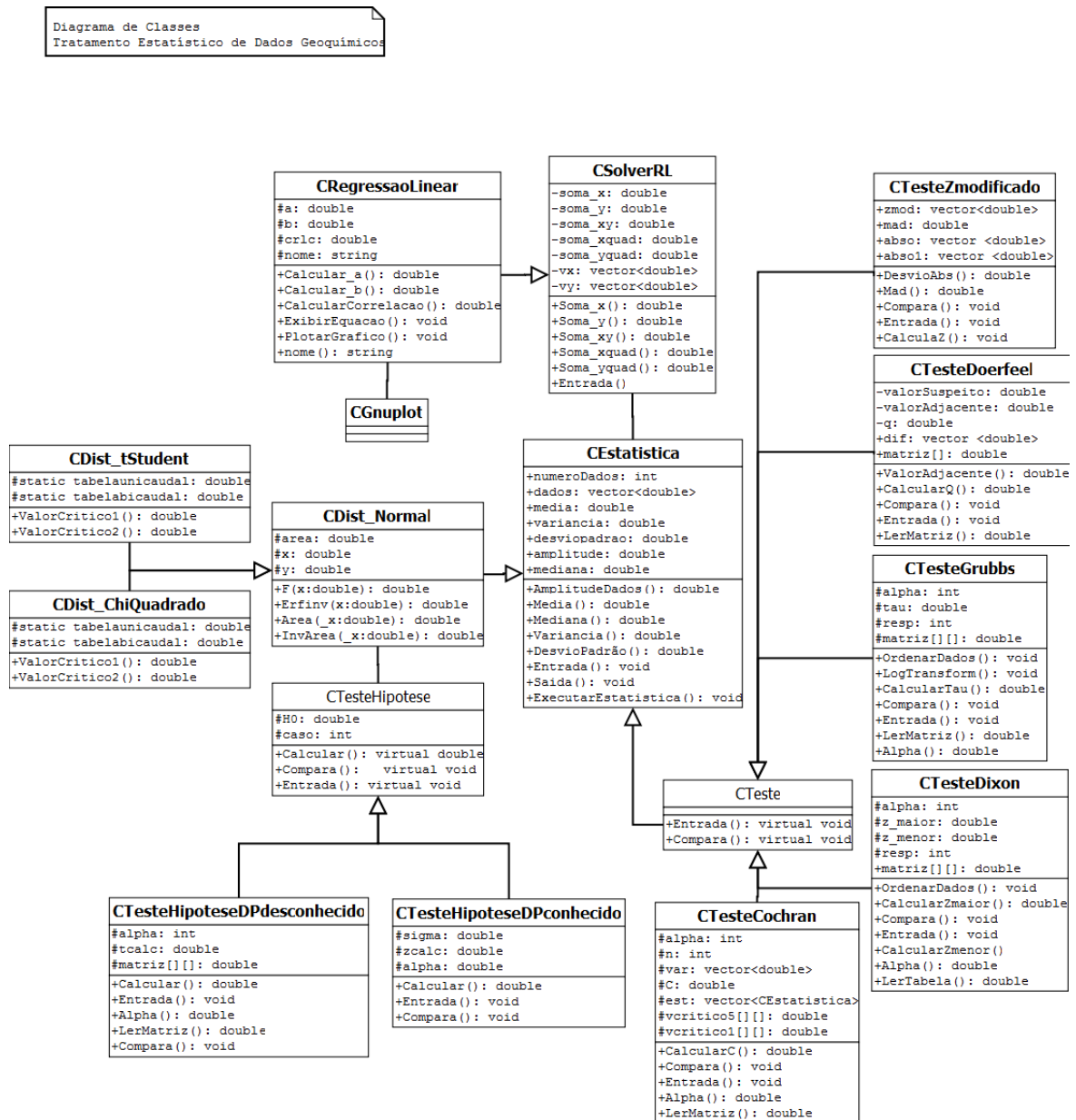


Figura 4.1: Diagrama de classes

Os métodos Get() e Set() e os métodos construtores e destrutores foram omitidos por serem padrão em linguagens de programação orientada a objeto. Entretanto, todos eles foram implementados no código do programa.

A classe CTesteHipotese é uma classe abstrata e possui o método virtual puro calcularZ(). Esta classe é uma classe base que possui como classes derivadas CTesteHipoteseDPconhecido e CTesteHipoteseDPdesconhecido. Para o teste de hipótese será implementado o polimorfismo, onde o usuário poderá escolher o teste de hipótese desejado. Nessa implementação será criado um ponteiro do tipo CTesteHipotese através do qual o programa irá executar o método calcularZ() do teste escolhido em tempo de execução.

A classe CTeste é uma classe abstrata e possui dois métodos virtuais puros Entrada() e Compara(). Esta classe é que possui como classes derivadas CTesteZmodificado, CTesteDoerffel, CTesteDixon, CTesteCochran. Para os testes para detecção de

anomalia será implementado o polimorfismo, onde o usuário poderá escolher o teste a ser realizado. Nessa implementação será criado um ponteiro para um objeto da classe base através do qual o programa irá executar os métodos `Entrada()` e `Compara()`.

4.1.1 Dicionário de classes

- Classe `CSolverRL`: Classe que possui um método que calcula o somatório de x , somatório de y , somatório de xy , somatório de x^2 , somatório de y^2 que serão utilizados pela classe `CR regressaoLinear` para o cálculo dos coeficientes a e b da equação da reta $y = a + bx$ e o coeficiente de correlação linear.
- Classe `CR regressaoLinear`: Classe representativa de uma regressão linear que descreve o relacionamento entre duas variáveis em termos matemáticos. Veja seção 3.1.2.
- Classe `CTeste`: Classe de interface que permite o uso do polimorfismo nas classes de teste de identificação de anomalias (*outliers*).
- Classe `CTesteZmodificado`: Classe representativa de um teste realizado pra identificar anomalias. Veja seção 3.1.8.
- Classe `CTesteDoerffel`: Classe representativa de um teste realizado pra identificar anomalias. Veja seção 3.1.12.
- Classe `CTesteGrubbs`: Classe representativa de um teste realizado pra identificar anomalias (muito útil para testar variabilidade entre laboratórios). Veja seção 3.1.9.
- Classe `CTesteDixon`: Classe representativa de um teste realizado pra identificar anomalias (recomendado quando o número de observações está entre 3 e 25). Veja seção 3.1.10.
- Classe `CTesteCochran`: Classe representativa de um teste realizado pra identificar anomalias (recomendado para estudar a variabilidade interna de um laboratório). Veja seção 3.1.11.
- Classe `CEstatística`: Classe representativa dos cálculos estatísticos básicos. Veja seção 3.1.1.
- Classe `CTesteHipotese`: Classe representativa de um teste cujo objetivo é verificar se os dados são compatíveis com alguma hipótese. Veja seção 3.1.5.
- Classe `CTesteHipoteseDPdesconhecido`: Classe representativa de um teste cujo objetivo é verificar se os dados são compatíveis com alguma hipótese. Veja seção 3.1.6.
- Classe `CTesteHipoteseDPconhecido`: Classe representativa de um teste cujo objetivo é verificar se os dados são compatíveis com alguma hipótese. Veja seção 3.1.6.

- Classe `CDist_Normal`: Classe representativa de distribuição normal padrão de probabilidade. Veja seção 3.1.3.
- Classe `CDist_tStudent`: Classe representativa da distribuição t de Student de probabilidade. Veja seção 3.1.4.
- Classe `CDist_ChiQuadrado`: Classe representativa da distribuição Chi-Quadrado de probabilidade.

4.2 Dicionário de classes com atributos/métodos

- Classe `CSolverRL`:

Atributos:

- double `sx` - somatório dos valores de x .
- double `sy` - somatório dos valores de y .
- double `sxy` - somatório dos valores de $x * y$.
- double `sx2` - somatório dos valores de x^2 .
- double `sy2` - somatório dos valores de y^2 .
- vector<double> `vx` - vetor de ordenada x do ponto (x, y) .
- vector<double> `vy` - vetor de ordenada y do ponto (x, y) .

Métodos:

- double `Sx()` - soma todos os valores contidos no vetor `vx` da classe `CRegressaoLinear`.
- double `Sy()` - soma todos os valores contidos no vetor `vy` da classe `CRegressaoLinear`.
- double `Sxy()` - soma dos produtos $x_i * y_j$, com $i = j$, percorrendo todos os vetores.
- double `Sx2()` - somatório do quadrado de cada elemento do vetor `vx`.
- double `Sy2()` - somatório do quadrado de cada elemento do vetor `vy`.
- void `Entrada()` - solicita os dados ao usuário.

- Classe `CRegressaoLinear`:

Atributos:

- double `a` - coeficiente linear.
- double `b` - coeficiente angular.

- double r - coeficiente de correlação.
- string equacao - armazena o nome da reta encontrada pela regressão.
- CGnuplot grafico - objeto da classe CGnuplot utilizado para acessar o programa *Gnuplot*.

Métodos:

- double A() - calcula o valor do coeficiente linear da reta.
- double B() - calcula o valor do coeficiente angular da reta.
- double Correlacao() - calcula a associação numérica entre duas variáveis.
- void ExibirEquacao() - após o cálculo, exibe a equação da reta.
- void Plotar() - plota um gráfico da reta $y = a + bx$.
- string Equacao() - retorna a equação como uma string para utilização no método Plotar().

- Classe CTesteZmodificado:

Atributos:

- vector <double> desvioAbsoluto - armazena o módulo dos desvios absolutos dos dados.
- vector <double> desvio - armazena os desvios absolutos dos dados.
- double mad - mediana dos módulos dos desvios absolutos.
- vector<double> zmod - armazena os valores da estatística calculada do teste.

Métodos:

- double DesvioAbs() - calcula os desvios absolutos e seu módulo e preenche os vetores desvioAbsoluto e desvio.
- double Mad() - calcula a mediana do módulo dos desvios absolutos.
- double Z() - calcula os valores de z modificado.
- void Compara() - compara a estatística calculada.
- void Entrada() - solicita os dados ao usuário.

- Classe CTesteDoerffel:

Atributos:

- double valorSuspeito - valor suspeito de ser uma anomalia (*outlier*).
- double valorAdjacente - valor mais próximo do *outlier*.
- double q - estatística do teste.

- static double valorCritico[] - matriz que armazena os valores críticos do teste.
- vector <double> dif - guarda os valores das diferenças entre o valor suspeito e os demais.

Métodos:

- double ValorAdjacente() - calcula o valor adjacente, mais próximo do *outlier*.
- double Q() - calcula a estatística do teste.
- void Compara() - compara o valor calculado ao da tabela para saber se será aceito ou rejeitado.
- void Entrada() - solicita os dados ao usuário.
- double ValorCritico() - retorna o valor crítico do teste.

- Classe CTesteGrubbs:

Atributos:

- int alpha - nível de significância.
- double tau - estatística do teste.
- int resp - identifica a opção do usuário por calcular o maior ou menor valor.
- static double valorCritico[] - matriz que armazena os valores críticos.

Métodos:

- double LogTransform() - método que calcula o logaritmo natural.
- double Ordenar() - métodos que ordena os dados.
- double Tau() - calcula a estatística do teste de Grubbs.
- void Compara() - compara o valor calculado ao da tabela para saber se será aceito ou rejeitado.
- void Entrada() - solicita os dados ao usuário.
- double Alpha() - método que retorna o valor da significância.
- double ValorCritico() - retorna o valor crítico tabelado.

- Classe CTesteDixon:

Atributos:

- int alpha - nível de significância.
- int resp - identifica a opção do usuário se descartará o maior ou menor elemento.
- double z_maior - maior valor de z.

- double z_menor - menor valor de z.
- static double valorCritico[] - armazena os valores críticos do teste.

Métodos:

- void Ordenar() - método que coloca os dados obtidos em ordem crescente.
- double Zmaior() - calcula a estatística do teste de Dixon para o maior elemento.
- double Zmenor() - calcula a estatística do teste de Dixon para o menor elemento.
- void Compara() - compara o valor calculado ao da tabela para saber se será aceito ou rejeitado.
- void Entrada() - solicita os dados ao usuário.
- double Alpha() - retorna o nível de significância do teste.
- double ValorCritico() - retorna o valor crítico tabelado.

- Classe CTesteCochran:

Atributos:

- int alpha - nível de significância.
- int n - quantidade de grupos de amostras.
- vector<double> var - armazena a variância de cada grupo de amostra.
- double c - estatística do teste Cochran.
- std::vector<CEstatistica> est - vetor que armazena os dados dos grupos.
- static double vcritico5[] - tabela do valor crítico para alpha igual 0.05.
- static double vcritico1[] - tabela do valor crítico para alpha igual 0.01.

Métodos:

- double C() - calcula a estatística do teste Cochran.
- void Compara() - compara o valor calculado ao da tabela para saber se será aceito ou rejeitado.
- void Entrada() - solicita os dados ao usuário.
- double Alpha() - retorna o valor do nível de significância utilizada.
- double ValorCritico() - retorna o valor crítico tabelado.

- Classe CEstatística:

Atributos:

- int numeroDados - número de dados utilizados.

- vector<double> dados - armazena os dados da amostra.
- double media - média dos dados.
- double variancia - variância dos dados.
- double desviopadrao- desvio padrão dos dados.
- double amplitude - amplitude dos dados.
- double mediana - mediana dos dados.

Métodos:

- double AmplitudeDados() - calcula e retorna a amplitude dos dados (valor máximo - valor mínimo).
- double Media() - calcula e retorna a media.
- double Mediana() - calcula e retorna a mediana.
- double Variancia() - calcula e retorna a variância.
- double DesvioPadrao() - calcula e retorna o desvio padrão.
- void Entrada() - solicita os dados ao usuário.

- Classe CTesteHipotese:

Atributos:

- double H0 - valor da hipótese nula.
- int caso - identifica o caso escolhido pelo usuário (unicaudal à esquerda, unicaudal à direita ou bicaudal).

Métodos:

- double Calcular() - calcula a estatística do teste de hipótese.
- double Entrada() - solicita dados ao usuário.
- double Compara() - compara a estatística do teste calculada com o valor tabelado.

- Classe CTesteHipoteseDPdesconhecido:

Atributos:

- int alpha - valor que identifica o nível de significância.
- double tcalc - estatística do teste.
- static double valoresTabelados[][] - armazena os valores tabelados.

Métodos:

- double Calcular() - calcula a estatística do teste de hipótese para um desvio padrão desconhecido.
- void Entrada() - solicita os dados ao usuário.
- void Compara() - método que compara a estatística calculada e o valor crítico tabelado.
- double Alpha() - método que retorna o valor da significância.
- double ValorCritico() - retorna o valor crítico do teste.

- Classe CTesteHipoteseDP conhecido:

Atributos:

- double sigma - valor do desvio padrão populacional.
- double zcalc - estatística calculada.
- double alpha - identifica o nível de significância.

Métodos:

- double Calcular() - calcula a estatística do teste de hipótese para um desvio padrão desconhecido.
- void Entrada() - solicita os dados ao usuário.
- void Compara () - Compara o valor calculado e o valor tabelado.

- Classe CDist_Normal:

Atributos:

- double area - área que representa a probabilidade acumulada.
- double x - abcissa da distribuição normal.
- double y - ordenada da distribuição normal.

Métodos:

- double F (double) - calcula a ordenada da distribuição normal.
- double Erfinv (double) - aproximação da função erro inversa por séries de Taylor.
- double Area (double) - calcula a área a esquerda do parâmetro _x utilizando a função erro.
- double InvArea (double) - o usuário entra com o valor da probabilidade e método retorna a abcissa correspondente.

- Classe CDist_tStudent:

Atributos:

- static double tabelaunicaudal[][] - tabela de valores para testes unicaudais.
- static double tabelabicaudal[][] - tabela de valores para testes bicaudais.

Métodos:

- double ValorCritico1() - retorna o valor crítico para testes unicaudais.
- double ValorCritico2() - retorna os valores críticos para testes bicaudais.

- Classe CDist_ChiQuadrado:

Atributos:

- static double tabelaunicaudal[][] - tabela de valores para testes unicaudais.
- static double tabelabicaudal[][] - tabela de valores para testes bicaudais.

Métodos:

- double ValorCritico1() - retorna o valor crítico para testes unicaudais.
- double ValorCritico2() - retorna os valores críticos para testes bicaudais.

4.3 Diagrama de sequência – eventos e mensagens

O diagrama de sequência enfatiza a troca de eventos e mensagens e sua ordem temporal. Contém informações sobre o fluxo de controle do programa. Costuma ser montado a partir de um diagrama de caso de uso e estabelece o relacionamento dos atores (usuários e sistemas externos) com alguns objetos do sistema.

4.3.1 Diagramas de sequência

O diagrama de sequência da Figura 4.2 mostra o comportamento típico de um objeto da classe CRegressaoLinear.

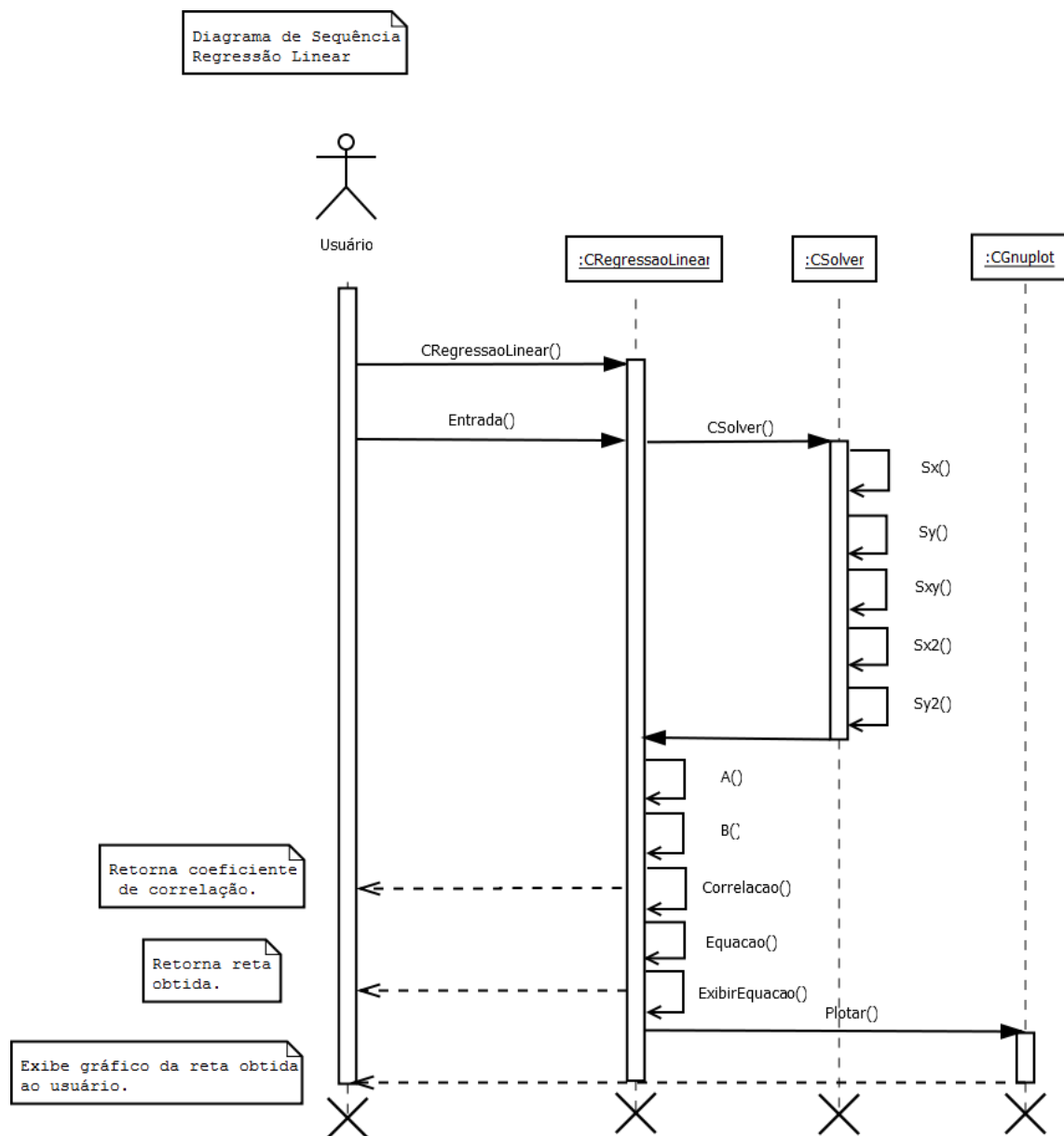


Figura 4.2: Diagrama de Sequência típico da classe CRegressaoLinear

O diagrama de sequência da Figura 4.3 mostra o comportamento típico de um objeto da classe CTesteZmodificado.

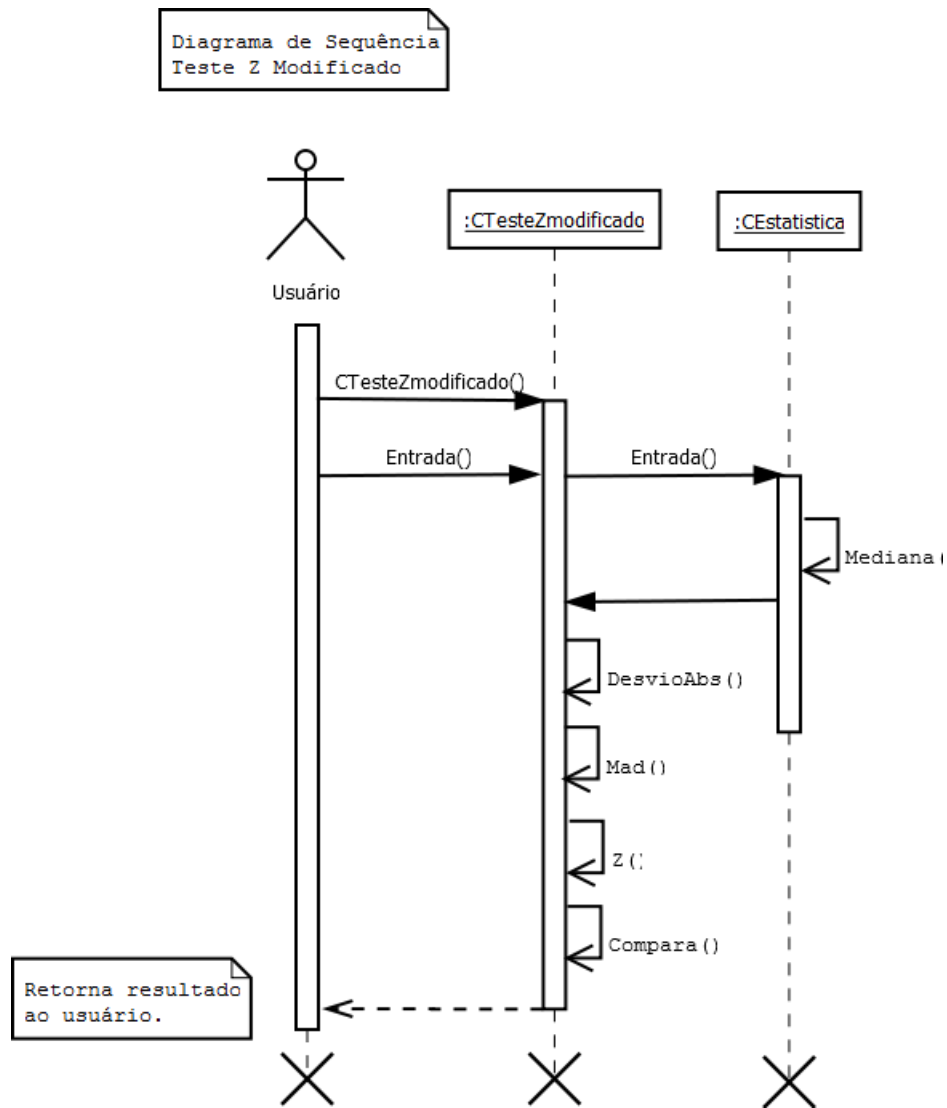


Figura 4.3: Diagrama de Sequência típico da classe CTesteZmodificado

A Figura 4.4 mostra o diagrama de sequência típico da classe CTesteDoerffel.

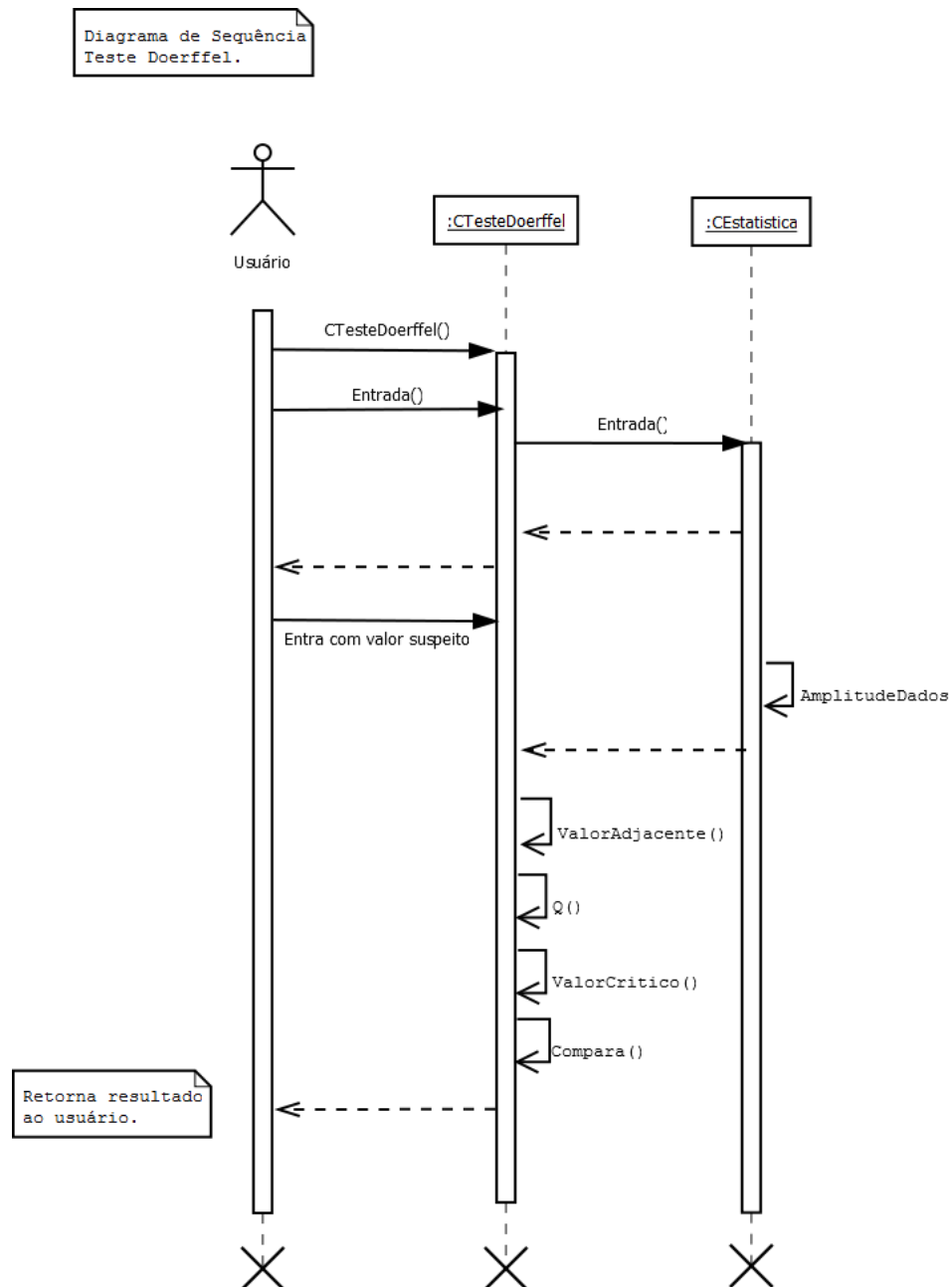


Figura 4.4: Diagrama de Sequência típico da classe CTesteDoerffel

A Figura 4.5 mostra o diagrama de sequência típico da classe CTesteGrubbs.

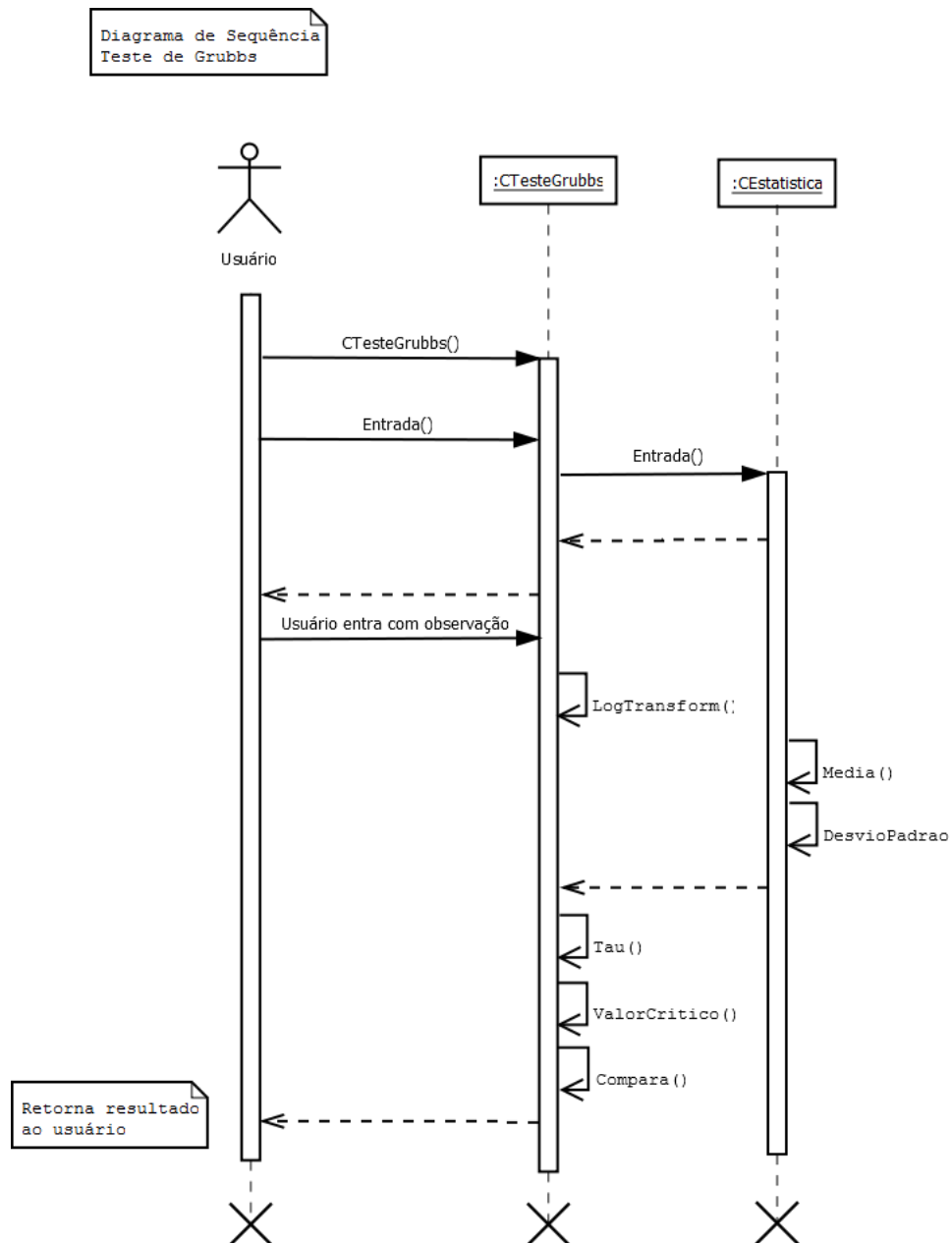


Figura 4.5: Diagrama de Sequência típico da classe CTesteGrubbs

A Figura 4.6 mostra o diagrama de sequência típico da classe CTesteDixon.

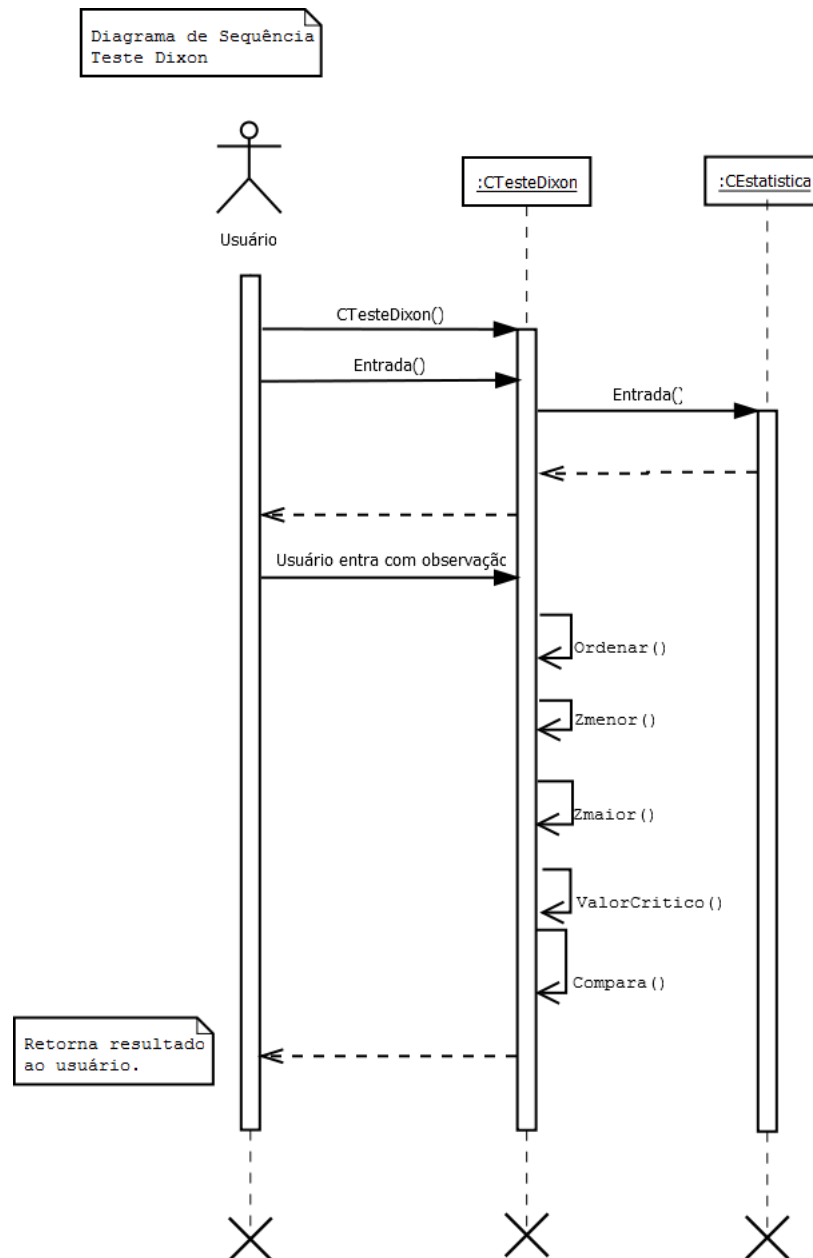


Figura 4.6: Diagrama de Sequência típico da classe CTesteDixon

A Figura 4.7 mostra o diagrama de sequência típico da classe CTesteCochran.

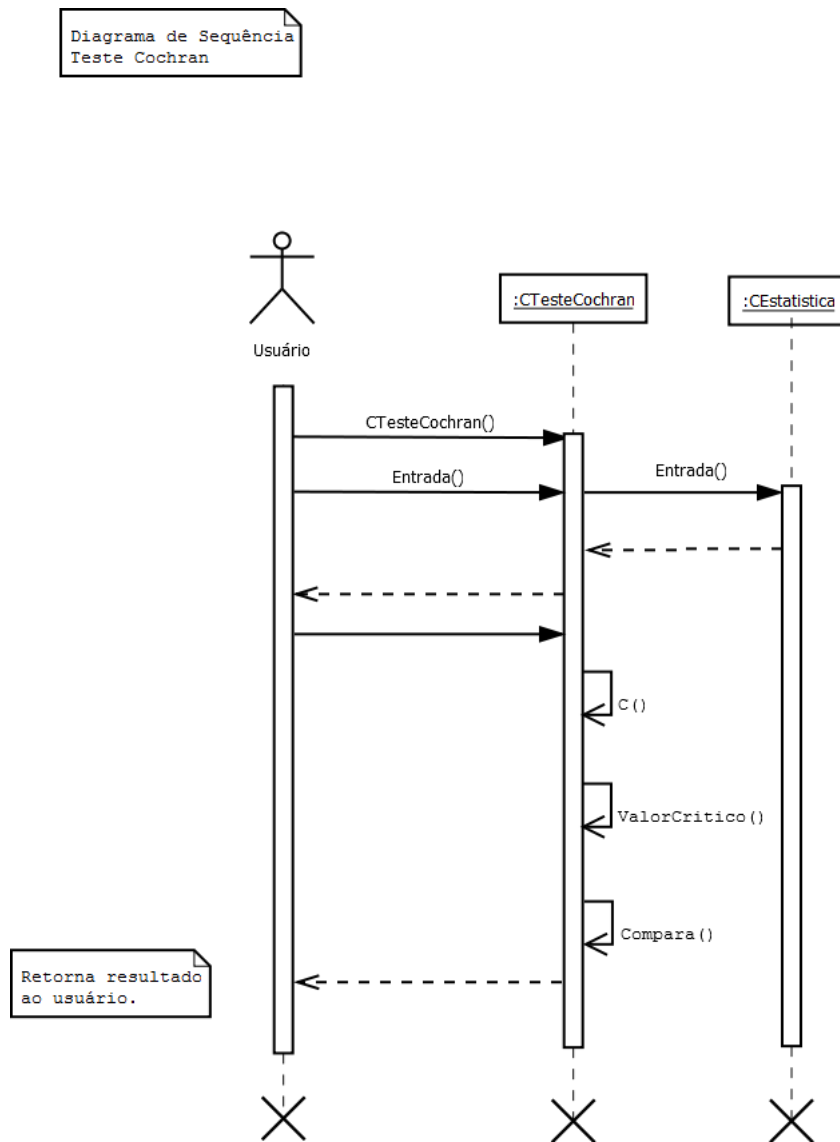


Figura 4.7: Diagrama de Sequência típico da classe CTesteCochran

A Figura 4.8 mostra o diagrama de sequência típico da classe CTesteHipoteseDPconhecido.

Diagrama de Sequência
Teste de Hipótese com Desvio Padrão Conhecido

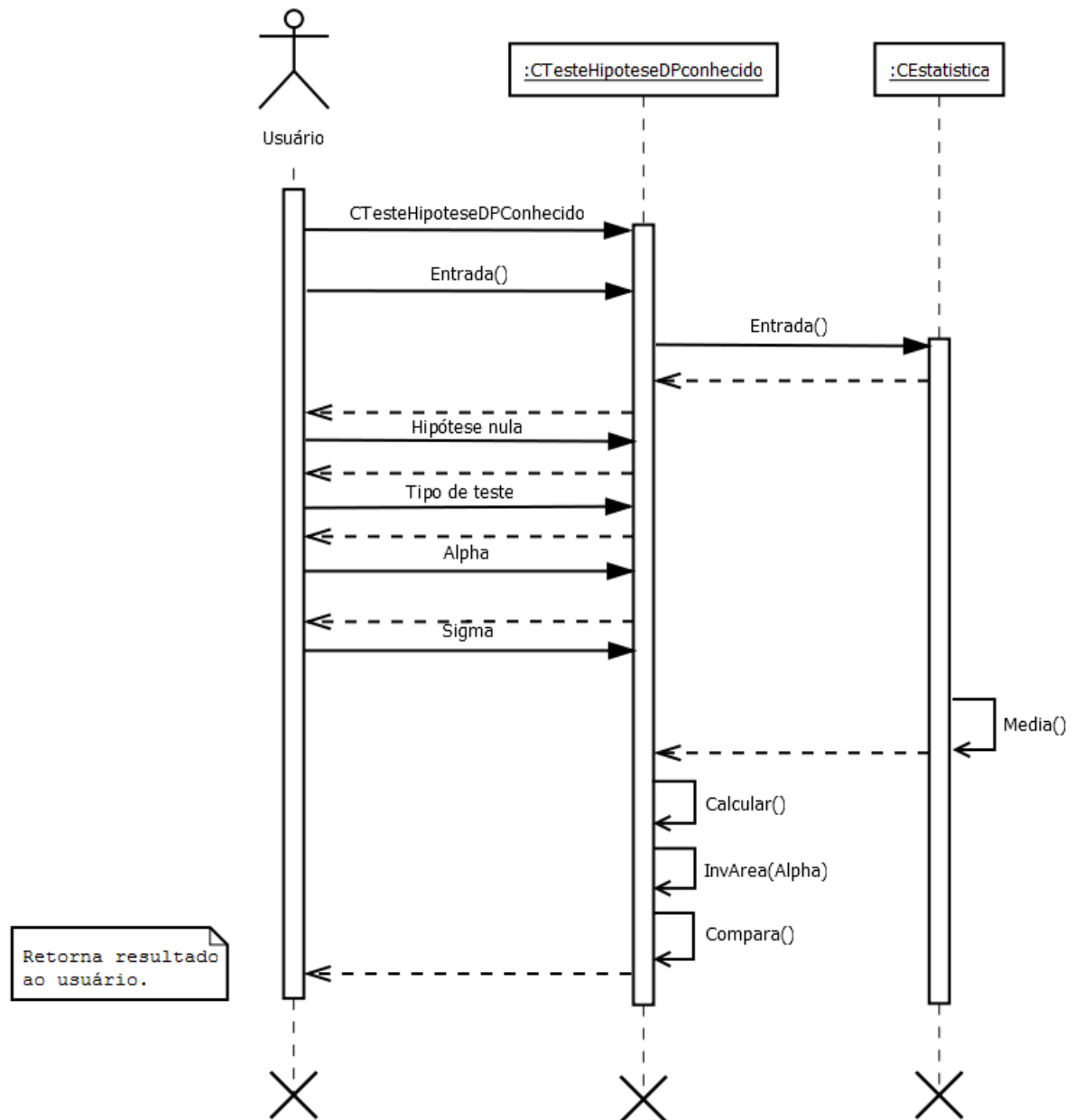


Figura 4.8: Diagrama de Sequência típico da classe CTesteHipoteseDPconhecido

A Figura 4.9 mostra o diagrama de sequência típico da classe CTesteHipoteseDPdesconhecido.

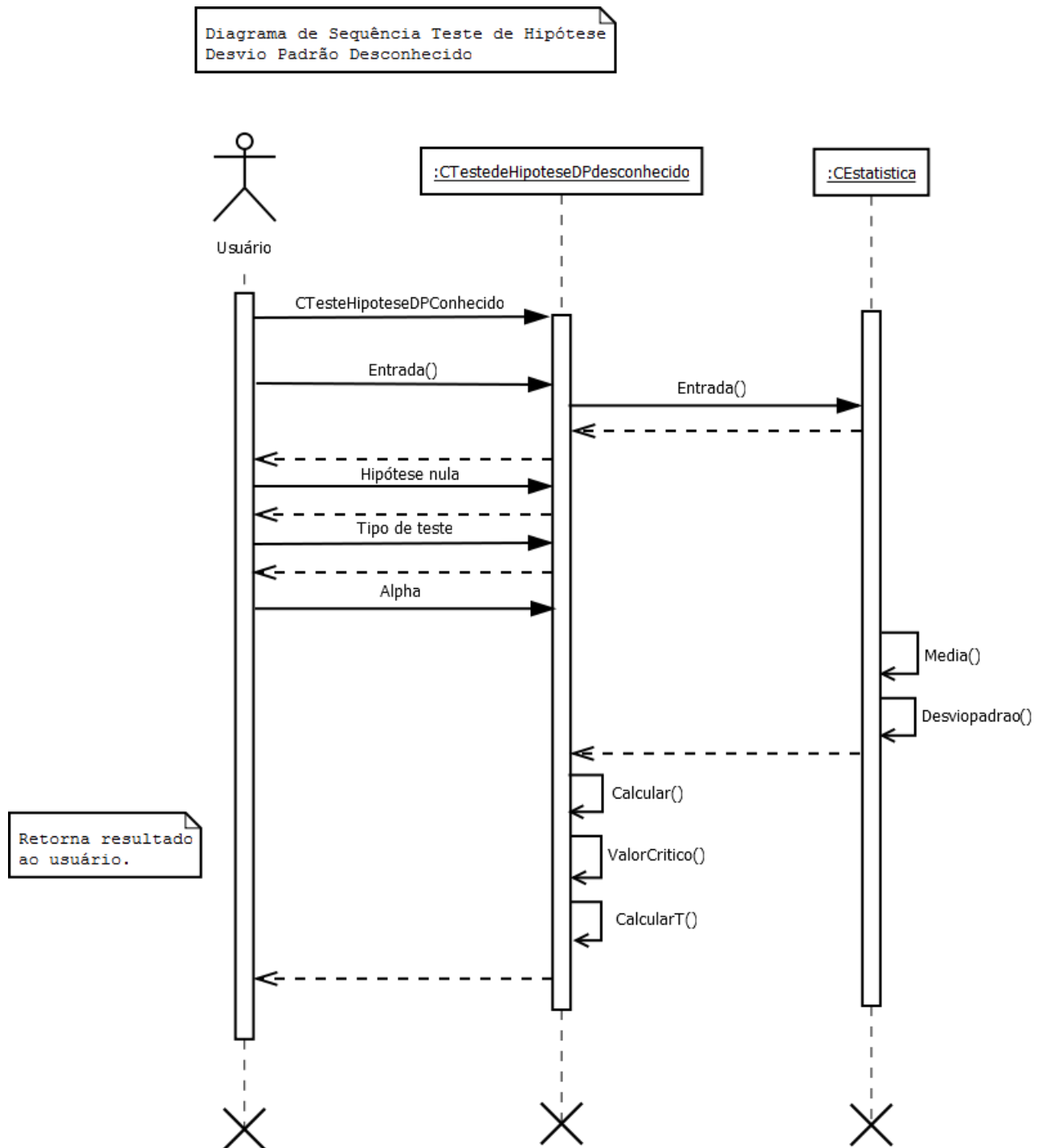


Figura 4.9: Diagrama de Sequência típico da classe CTesteHipoteseDPdesconhecido

4.4 Diagrama de atividades

Veja na Figura 4.10 o diagrama de atividades correspondente a uma atividade específica do diagrama de máquina de estado.

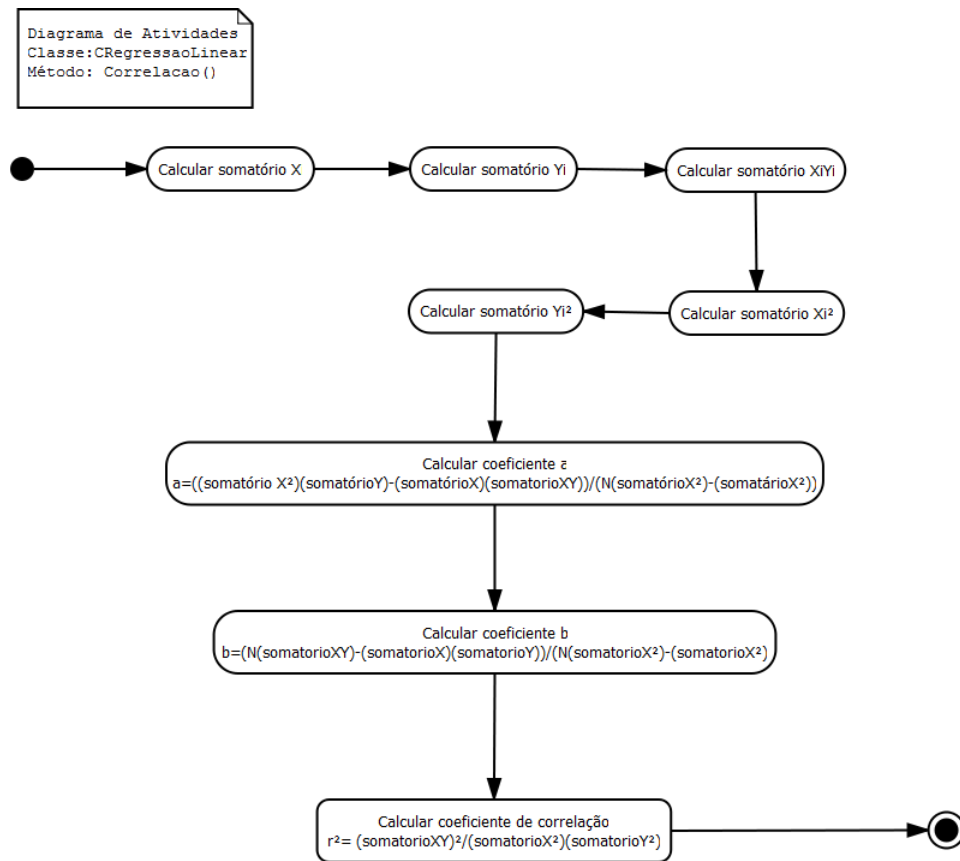


Figura 4.10: Diagrama de Atividades da classe CRegressãoLinear::Correlacao()

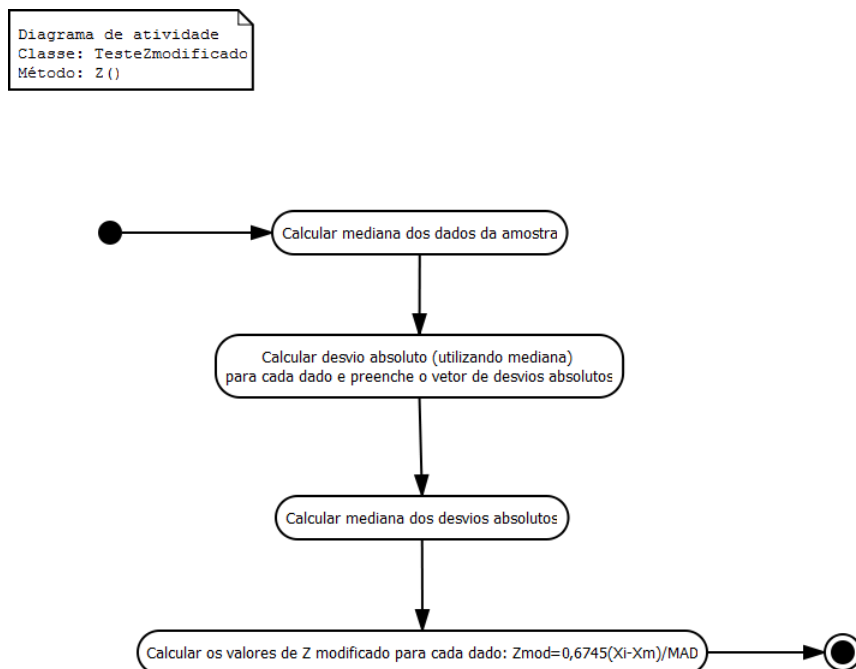
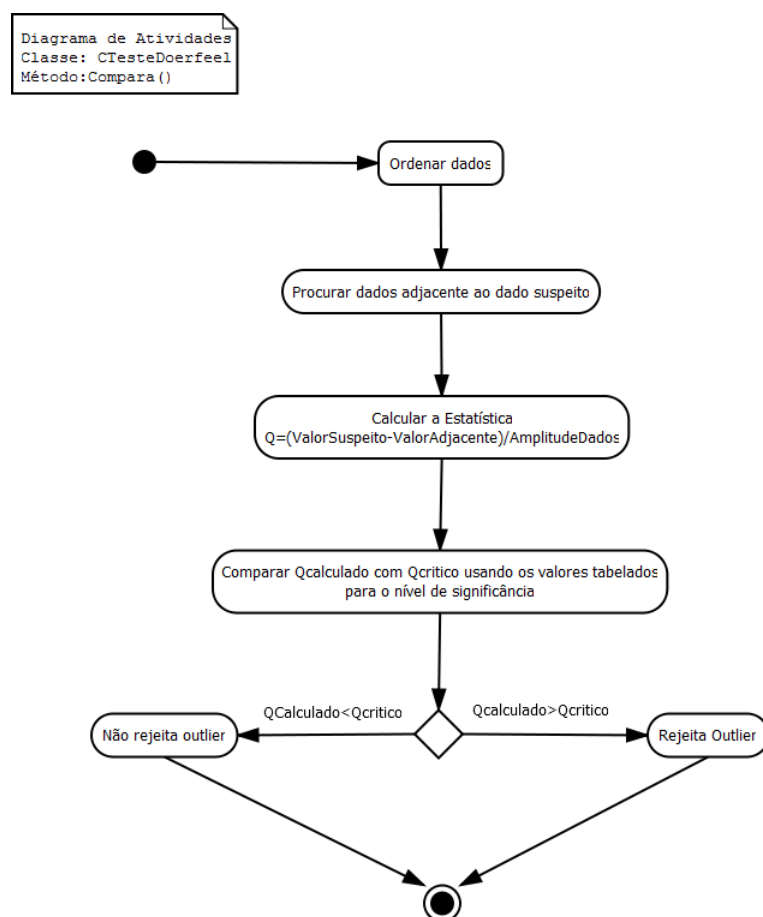
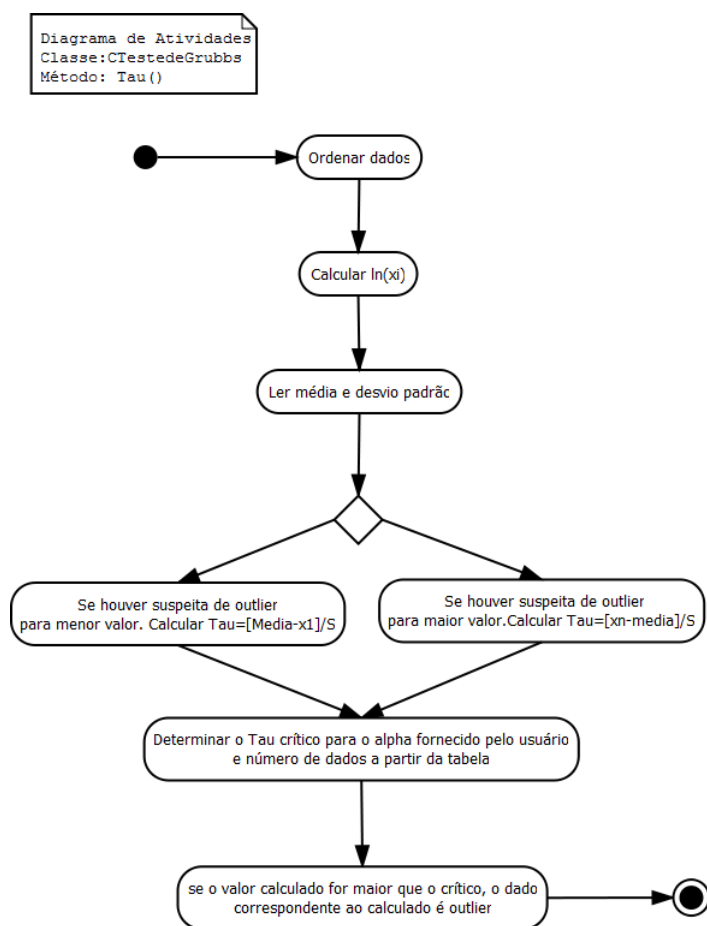
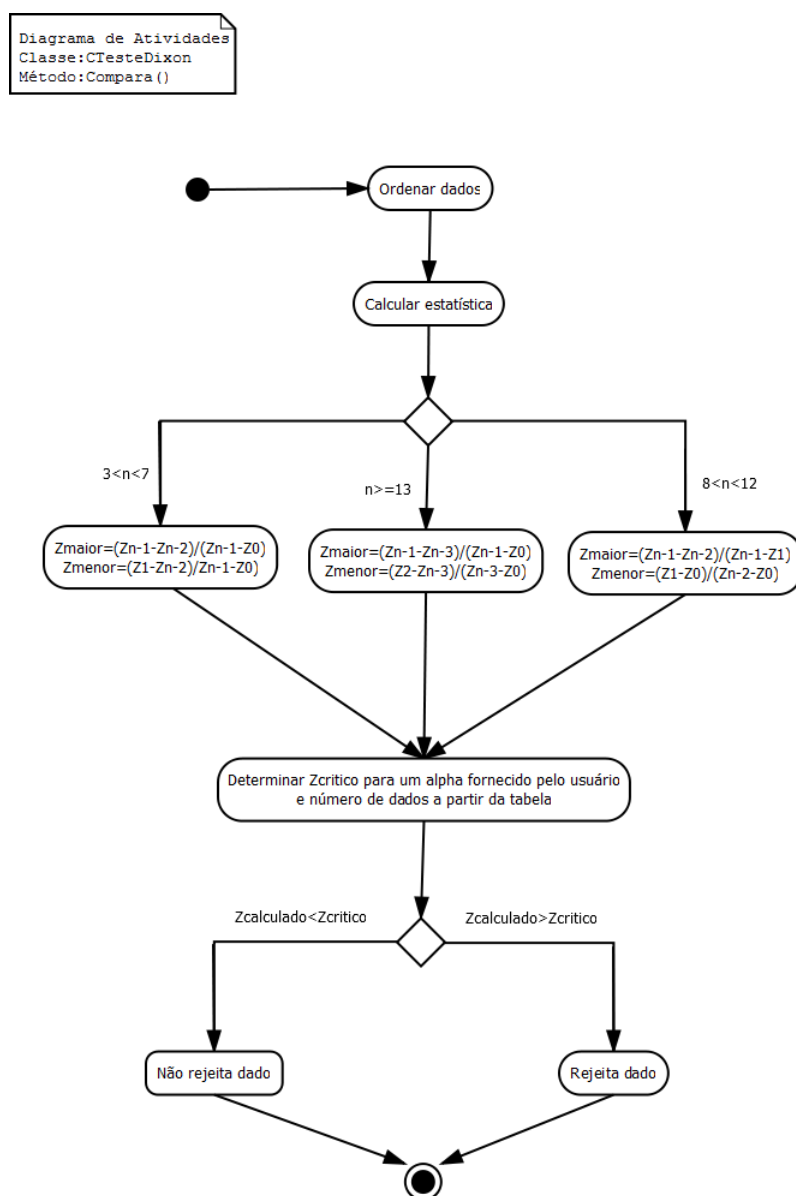


Figura 4.11: Diagrama de Atividades da classe CTesteZmodificado::Z()

Figura 4.12: Diagrama de Atividades da classe `CTesteDoerfeel::Compara()`

Figura 4.13: Diagrama de Atividades da classe `CTesteGrubbs::Tau()`

Figura 4.14: Diagrama de Atividades da classe `CTesteDixon::Compara()`

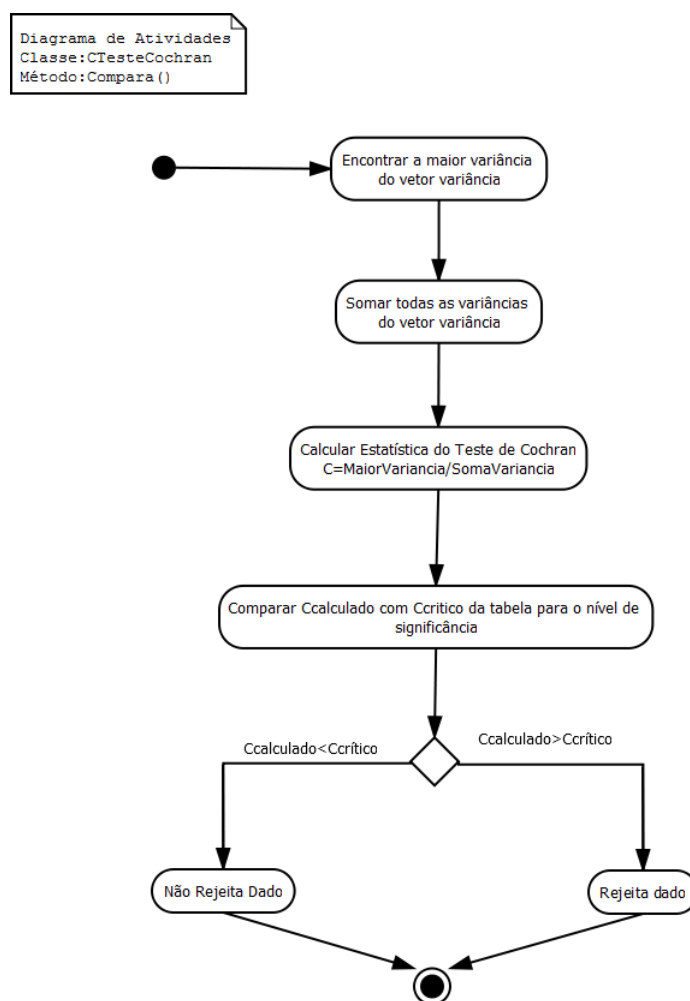


Figura 4.15: Diagrama de Atividades da classe CTesteCochran::Compara()

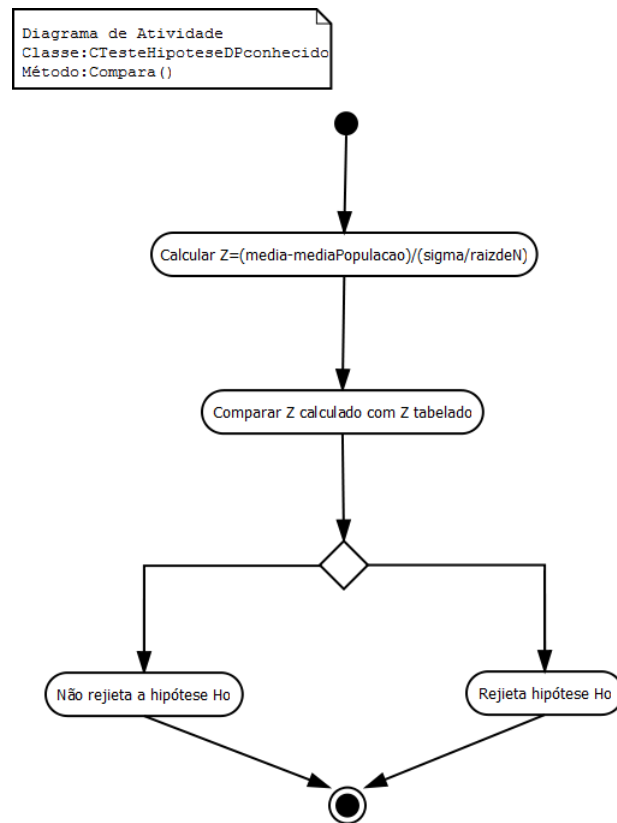


Figura 4.16: Diagrama de Atividades da classe CTesteHipoteseDPconhecido::Compara()

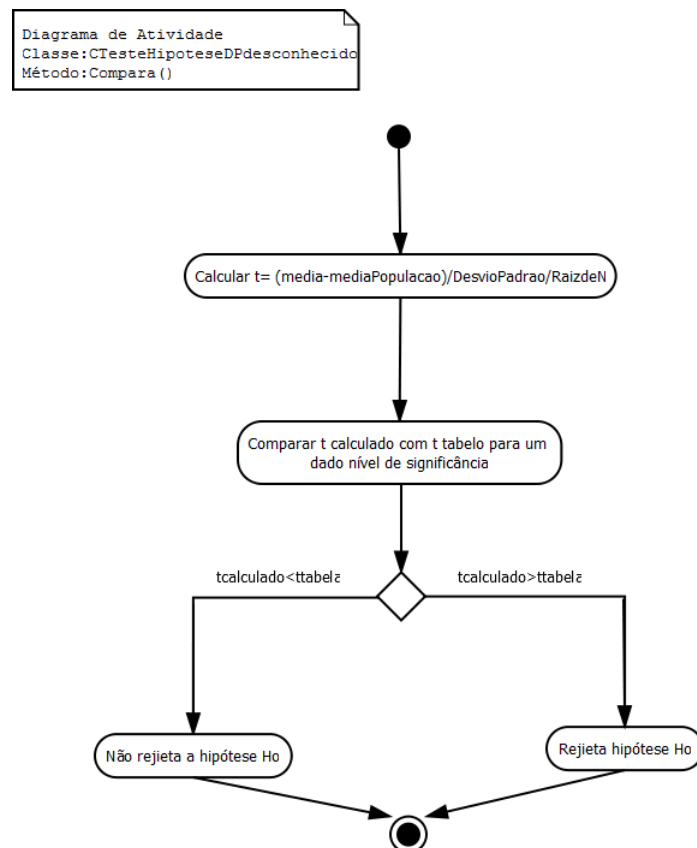


Figura 4.17: Diagrama de Atividades da classe CTesteHipoteseDPdesconhecido::Compara()

Capítulo 5

Projeto

5.1 Projeto do sistema

Depois de análise orientada a objeto desenvolve-se o projeto do sistema, qual envolve etapas como a definição dos protocolos, de interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas do controle, a seleção das plataformas dos sistemas, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto. Devem-se definir padrões de documentação, padrões para o nome das classes, padrões de retorno e de parâmetros, características da interface do usuário e características de desempenho.

5.2 Projeto Orientado a Objeto – POO

O projeto orientado a objeto é a etapa posterior ao projeto do sistema. Baseia-se na análise, mas considera as decisões do projeto do sistema. Acrescenta a análise envolvida e as características da plataforma escolhida. Passa pelo maior detalhamento do funcionamento do programa, acrescentando atributos e métodos que envolvem a solução de problemas específicos não identificados durante a análise. Envolve a otimização da estrutura de dados e dos algoritmos, a minimização do tempo de execução, de memória e de custos.

- Recursos: O programa utiliza o HD, o processador e o teclado do computador e o software livre `Gnuplot` para gerar gráficos.
- Plataformas: O programa é multiplataforma, funcionando tanto no Windows quanto no GNU/ Linux. Foram utilizadas bibliotecas padrão como `iostream`, `cmath`, `vector`, `string`, entre outras.
- Controle: Caso o usuário entre com algum dado errado será enviada uma mensagem de erro.

- Ambiente de desenvolvimento integrado: O programa foi compilado no GNU/Linux, utilizando o compilador g++, software livre de simples utilização.

Capítulo 6

Implementação

Neste capítulo serão apresentados os arquivos de cabeçalho e de implementação de todas as classes do programa.

6.1 Código fonte

Apresenta-se na listagem 6.1 o arquivo com código da classe CEstatistica.h

Listing 6.1: Arquivo de cabeçalho da classe CEstatistica.

```
#ifndef CTESTEDIXON_H
#define CTESTEDIXON_H
#include "CTeste.h"
#include <algorithm>
#include <utility>

class CTesteDixon : public CTeste
{
protected:
    int alpha; // Nível de significancia, assume os valores de 0, 1 ou 2.
    double z_maior; //Estatistica calculada caso o suspeito de ser outlier
        seja o maior numero
    double z_menor; // Estatistica calculada caso o suspeito de ser outlier
        seja o menor numero
    int resp; //Serve apenas para indicar se vai descartar o menor ou o
        maior elemento
    double matriz[23][3] = { //0.1    0.05    0.01    <-- Niveis de
        significancia
                                {0.886, 0.941, 0.988},
                                {0.679, 0.765, 0.889},
                                {0.557, 0.642, 0.780},
                                {0.482, 0.560, 0.698},
                                {0.434, 0.507, 0.637},
                                {0.479, 0.554, 0.683},
                                {0.441, 0.512, 0.635},
```

```

        {0.409, 0.477, 0.597},
        {0.517, 0.576, 0.679},
        {0.490, 0.546, 0.642},
        {0.467, 0.521, 0.615},
        {0.492, 0.546, 0.641},
        {0.472, 0.525, 0.616},
        {0.454, 0.507, 0.595},
        {0.438, 0.490, 0.577},
        {0.424, 0.475, 0.561},
        {0.412, 0.462, 0.547},
        {0.401, 0.450, 0.535},
        {0.391, 0.440, 0.524},
        {0.382, 0.430, 0.514},
        {0.374, 0.421, 0.505},
        {0.367, 0.413, 0.497},
        {0.360, 0.406, 0.489}
    };

public:
    void OrdenarDados(); //Metodo que ordena os dados do vetor
    double CalcularZmaior(); //Metodo que calcula a estatistica do teste
        caso o suspeito de ser um outlier seja o maior valor
    double CalcularZmenor(); //Metodo que calcula a estatistica do teste
        caso o suspeito de ser um outlier seja o menor valor
    double LerTabela(); // Metodo que retorna o valor critico do teste
    void Entrada(); //Metodo de entrada, pede dados iniciais
    double Alpha(); //Metodo que retorna a significancia --> 0.10, 0.05 e
        0.01
    void Compara(); //Metodo que compara a estatistica calcula e o valor
        critico tabelado
};

#endif

```

Apresenta-se na listagem 6.2 o arquivo com código da classe CEstadistica.cpp

Listing 6.2: Arquivo de implementação da classe CEstadistica.

```

#ifndef CTESTEDIXON_H
#define CTESTEDIXON_H
#include "CTeste.h"
#include <algorithm>
#include <utility>

class CTesteDixon : public CTeste
{
protected:
    int alpha; // Nivel de significancia, assume os valores de 0, 1 ou 2.
    double z_maior; //Estatistica calculada caso o suspeito de ser outlier
        seja o maior numero

```

```

double z_menor; // Estatística calculada caso o suspeito de ser outlier
                 seja o menor numero
int resp; // Serve apenas para indicar se vai descartar o menor ou o
           maior elemento
double matriz[23][3] = { // 0.1    0.05    0.01  <- Níveis de
                        significancia
                        {0.886, 0.941, 0.988},
                        {0.679, 0.765, 0.889},
                        {0.557, 0.642, 0.780},
                        {0.482, 0.560, 0.698},
                        {0.434, 0.507, 0.637},
                        {0.479, 0.554, 0.683},
                        {0.441, 0.512, 0.635},
                        {0.409, 0.477, 0.597},
                        {0.517, 0.576, 0.679},
                        {0.490, 0.546, 0.642},
                        {0.467, 0.521, 0.615},
                        {0.492, 0.546, 0.641},
                        {0.472, 0.525, 0.616},
                        {0.454, 0.507, 0.595},
                        {0.438, 0.490, 0.577},
                        {0.424, 0.475, 0.561},
                        {0.412, 0.462, 0.547},
                        {0.401, 0.450, 0.535},
                        {0.391, 0.440, 0.524},
                        {0.382, 0.430, 0.514},
                        {0.374, 0.421, 0.505},
                        {0.367, 0.413, 0.497},
                        {0.360, 0.406, 0.489}
                        };

public:
    void OrdenarDados(); // Metodo que ordena os dados do vetor
    double CalcularZmaior(); // Metodo que calcula a estatística do teste
                             caso o suspeito de ser um outlier seja o maior valor
    double CalcularZmenor(); // Metodo que calcula a estatística do teste
                             caso o suspeito de ser um outlier seja o menor valor
    double LerTabela(); // Metodo que retorna o valor crítico do teste
    void Entrada(); // Metodo de entrada, pede dados iniciais
    double Alpha(); // Metodo que retorna a significancia --> 0.10, 0.05 e
                    0.01
    void Compara(); // Metodo que compara a estatística calcula e o valor
                    crítico tabelado
};

#endif

```

Apresenta-se na listagem 6.3 o arquivo com código da classe CSolverRL.h

Listing 6.3: Arquivo de cabeçalho da classe CSolverRL.

```

#ifndef CSOLVERRL_H
#define CSOLVERRL_H
#include <iostream>
#include <vector>

class CSolverRL
{
protected:
    std::vector <double> vx;
    std::vector <double> vy;
    double soma_x;
    double soma_y;
    double soma_xy;
    double soma_xquad;
    double soma_yquad;
public:
    void Entrada();
    double Soma_x();
    double Soma_y();
    double Soma_xy ();
    double Soma_xquad();
    double Soma_yquad ();
};

#endif

```

Apresenta-se na listagem 6.4 o arquivo com código da classe CSolverRL.cpp

Listing 6.4: Arquivo de implementação da classe CSolverRL.

```

#include <iostream>
#include <vector>
#include "CSolverRL.h"

using namespace std;

void CSolverRL::Entrada()
{
    double auxiliar1 ;
    double auxiliar2;
    do{
        cout << "Entre com os pares de dados, primeiro a variavel independente: \n" << endl ;
        cin >> auxiliar1 ; cin.get ();
        if(cin.good())
            vx.push_back (auxiliar1);
    }
}

```

```

cout << "Agora entre com o variavel dependente: (ctrl+ d encerra a
    entrada de dados)" << endl;
cin >> auxiliar2; cin.get();
if(cin.good())
vy.push_back (auxiliar2);
} while (cin.good());
cin.clear();
if (vx.size() != vy.size())
{
    cout << "O tamanho dos vetores devem ser iguais." << endl;
    cout << "Entre com os dados novamente." << endl;
    vx.clear(); vy.clear();
    this->Entrada();
}
}

double CSolverRL::Soma_x()
{
    soma_x = 0.0;
    for (int i = 0; i < vx.size() ; i++)
    {
        soma_x += vx[i];
    }
    return soma_x;
}

double CSolverRL::Soma_y()
{
    soma_y = 0.0;
    for (int i = 0; i < vy.size() ; i++)
    {
        soma_y += vy[i];
    }
    return soma_y;
}

double CSolverRL::Soma_xy ()
{
    soma_xy = 0.0;
    for (int i = 0; i < vx.size() ; i++)
    {
        soma_xy += vx[i] * vy[i];
    }
    return soma_xy;
}

double CSolverRL::Soma_xquad()
{

```

```

    soma_xquad = 0.0;
    for (int i = 0; i < vx.size() ; i++)
    {
        soma_xquad += vx[i] * vx[i];
    }
    return soma_xquad;
}

double CSolverRL::Soma_yquad ()
{
    soma_yquad = 0.0;
    for (int i = 0; i < vy.size() ; i++)
    {
        soma_yquad += vy[i] * vy[i];
    }
    return soma_yquad;
}

```

Apresenta-se na listagem 6.5 o arquivo com código da classe CRegressaoLinear.h

Listing 6.5: Arquivo de cabeçalho da classe CRegressaoLinear.

```

#ifndef CREGRESSAOLINEAR_H
#define CREGRESSAOLINEAR_H
#include <iostream>
#include <vector>
#include <string>
#include "CSolverRL.h"

class CRegressaoLinear : public CSolverRL
{
protected:
    double a;
    double b;
    double crlc;
    std::string nome;
public:
    double Calcular_a();
    double Calcular_b();
    std::string Nome();
    double CalcularCorrelacao();
    void ExibirEquacao();
    void PlotarGrafico();
    void Entrada();
};

#endif

```

Apresenta-se na listagem 6.6 o arquivo com código da classe CRegressaoLinear.cpp

Listing 6.6: Arquivo de implementação da classe CRegressaoLinear.

```

#include <sstream>
#include <cmath>
#include "CRegressaoLinear.h"
#include "CGnuplot.h"
#include "pstream.h"

using namespace std;

std::string CRegressaoLinear::Nome()
{
    ostringstream nome1;
    nome1 << Calcular_a() << " * x + " << Calcular_b() << " ";
    nome = nome1.str();
    return nome;
}

double CRegressaoLinear::Calcular_a()
{
    a = 0.0;
    a = (vx.size()*Soma_xy() - Soma_x()*Soma_y()) / (vx.size()*Soma_xquad() - Soma_x()*Soma_x());
    return a;
}

double CRegressaoLinear::Calcular_b()
{
    b = 0.0;
    b = (Soma_xquad()*Soma_y() - Soma_xy()*Soma_x()) / (vx.size()*Soma_xquad() - Soma_x()*Soma_x());
    return b;
}

double CRegressaoLinear::CalcularCorrelacao()
{
    crlc = 0.0;
    crlc = (Soma_xy()*Soma_xy()) / (Soma_xquad()*Soma_yquad());
    cout << "O coeficiente do ajuste da regressao e: " << crlc << endl;
}

void CRegressaoLinear::ExibirEquacao()
{
    cout << "A equacao obtida com a regressao linear e: " << Calcular_a() <<
        " * x + " << Calcular_b() << endl;
}

```

```

}

void CRegressaoLinear::PlotarGrafico()
{
    CGnuplot *graf = new CGnuplot ("lines");
    graf->set_xrange(0,5);
    graf->set_yrange(0,10);
    graf->plot_equation(Nome() , Nome());
}

void CRegressaoLinear::Entrada()
{
    CSolverRL::Entrada();
}

```

Apresenta-se na listagem 6.7 o arquivo com código da classe CTeste.h

Listing 6.7: Arquivo de cabeçalho da classe CTeste.

```

#ifndef CTESTE_H
#define CTESTE_H

#include "CEstatistica.h"

class CTeste : public CEstatistica
{
public:
    virtual void Entrada() = 0;
    virtual void Compara() = 0;
};

#endif

```

Apresenta-se na listagem 6.8 o arquivo com código da classe CTeste.cpp

Listing 6.8: Arquivo de implementação da classe CTeste.

```

#include "CTeste.h"

```

Apresenta-se na listagem 6.9 o arquivo com código da classe CTesteCochran.h

Listing 6.9: Arquivo de cabeçalho da classe CTesteCochran.

```

#ifndef CTESTECOCHRAN_H
#define CTESTECOCHRAN_H

#include "CTeste.h"
#include <vector>
#include <numeric>
#include <algorithm>

```



```

class CTesteCochran : public CTeste
{
protected:
    double C; //Estatistica calculada do teste de Cochran, valor a ser
               comparado com o tabelado.
    int n; // Numero de grupos de dados
    int alpha; //Valor da significancia
    std::vector <double> var; //Vetor que guarda a variancia dos grupos
    std::vector <CEstatistica> est; //Vetor que guarda os dados de cada
                                   grupo
    double vcritico5[10][9] = //Tabela do valor critico para alpha =
                               0.05
    {
    {0.9985, 0.9669, 0.9065, 0.8412, 0.7808, 0.7271, 0.6798, 0.6385,
     0.6020},
    {0.9750, 0.8709, 0.7679, 0.6838, 0.6161, 0.5612, 0.5157, 0.4775,
     0.4450},
    {0.9392, 0.7977, 0.6841, 0.5981, 0.5321, 0.4800, 0.4377, 0.4027,
     0.3733},
    {0.9057, 0.7457, 0.6287, 0.5441, 0.4803, 0.4307, 0.3910, 0.3584,
     0.3311},
    {0.8772, 0.7071, 0.5895, 0.5065, 0.4447, 0.3974, 0.3595, 0.3286,
     0.3029},
    {0.8534, 0.6771, 0.5598, 0.4783, 0.4184, 0.3726, 0.3362, 0.3067,
     0.2823},
    {0.8332, 0.6530, 0.5365, 0.4564, 0.3980, 0.3535, 0.3185, 0.2901,
     0.2666},
    {0.8159, 0.6333, 0.5175, 0.4387, 0.3817, 0.3384, 0.3043, 0.2768,
     0.2541},
    {0.8010, 0.6167, 0.5017, 0.4241, 0.3682, 0.3259, 0.2926, 0.2659,
     0.2439},
    {0.7880, 0.6025, 0.4884, 0.4118, 0.3568, 0.3154, 0.2829, 0.2568, 0.2353}
    };
    double vcritico1 [10][9] = //Tabela do valor critico para alpha =
                               0.01
    {
    {0.9999, 0.9933, 0.9676, 0.9279, 0.8828, 0.8376, 0.7945, 0.7544,
     0.7175},
    {0.9950, 0.9423, 0.8643, 0.7885, 0.7218, 0.6644, 0.6152, 0.5727,
     0.5358},
    {0.9794, 0.8831, 0.7814, 0.6957, 0.6258, 0.5685, 0.5209, 0.4810,
     0.4469},
    {0.9586, 0.8335, 0.7212, 0.6329, 0.5635, 0.5080, 0.4627, 0.4251,
     0.3934},
    {0.9373, 0.7933, 0.6761, 0.5875, 0.5195, 0.4659, 0.4226, 0.3870,
     0.3572},
    }

```

```
{0.9172, 0.7606, 0.6410, 0.5531, 0.4866, 0.4347, 0.3932, 0.3592,
  0.3308},
{0.8988, 0.7335, 0.6129, 0.5259, 0.4608, 0.4105, 0.3704, 0.3378,
  0.3106},
{0.8823, 0.7107, 0.5897, 0.5037, 0.4401, 0.3911, 0.3522, 0.3207,
  0.2945},
{0.8674, 0.6912, 0.5702, 0.4854, 0.4229, 0.3751, 0.3373, 0.3067,
  0.2813},
{0.8539, 0.6743, 0.5536, 0.4697, 0.4084, 0.3616, 0.3248, 0.2950, 0.2704}
};
```

```
public:
    double CalculaC(); //Metodo que calcula a estatistica do teste
    void Entrada(); //Metodo de entrada, pede os dados iniciais
    double LerMatriz(); //Metodo que retorna o valor critico tabelado
    void Compara(); //Metodo que compara o valor calculado e o valor
        critico tabelado
    double Alpha(); //Metodo que retorna o valor da variancia utilizada
};

#endif
```

Apresenta-se na listagem 6.10 o arquivo com código da classe CTesteCochran.cpp

Listing 6.10: Arquivo de implementação da classe CTesteCochran.

```
#include "CTesteCochran.h"

using namespace std;

double CTesteCochran::CalculaC()
{
    C = *max_element(var.begin (),var.end ()) / accumulate (var.begin(),
        var.end(),0.);
    return C;
}

double CTesteCochran::LerMatriz()
{
    if (alpha == 0)
        return vcritico5[est[0].dados.size()-2][n-2];
    if (alpha == 1)
        return vcritico1[est[0].dados.size()-2][n-2];
}
```

```

void CTesteCochran::Compara()
{
    double numero = *max_element(var.begin(), var.end());
    vector<double>::iterator it = find(var.begin(), var.end(), numero);
    if (CalculaC() > LerMatriz())
    {
        cout << "0_grupo" << (it - var.begin()+1) << "apresenta variancia_
            nao_homogenea_com_los_demais_grupos" << endl;
        cout << "0_valor_de_C_calculado_e" << CalculaC() << endl;
        cout << "0_valor_critico_tabelado_para_o_nivel_de_significancia=" <<
            << Alpha() << "e" << LerMatriz() << endl;
    }
    else
    {
        cout << "0_grupo" << (it - var.begin()+1) << "apresenta variancia_
            homogenea_com_los_demais_grupos" << endl;
        cout << "0_valor_de_C_calculado_e" << CalculaC() << endl;
        cout << "0_valor_critico_tabelado_para_o_nivel_de_significancia=" <<
            << Alpha() << "e" << LerMatriz() << endl; }
    }
}

void CTesteCochran::Entrada()
{
    cout << "\tTeste de Cochran\n" << "\tAte 10 grupos de dados\n" << "\t
        tAte 11 dados por grupo\n" << endl;
    cout << "Entre com o numero de grupos:";
    cin >> n; cin.get();
    for (int i = 0; i < n; i++)
    {
        CEstatistica p1;
        cout << "Entre com os dados do grupo" << i+1 << endl;
        p1.CEstatistica::Entrada();
        cout << "A variancia do grupo e" << p1.Variancia() << endl;
        var.push_back(p1.Variancia());
        est.push_back(p1);
    }
    for (int i = 1; i < n; i++)
    {
        if (est[i-1].dados.size() != est[i].dados.size())
        { cerr << "0 numero de dados de cada grupo precisa ser igual\n";
            exit (EXIT_FAILURE); }
    }
    cout << "Agora, entre com o valor da significancia\n"
        << "Digite (0+ENTER) para alpha=0.05:" << endl
        << "Digite (1+ENTER) para alpha=0.01:";
    cin >> alpha; cin.get();
}

```

```
double CTesteCochran::Alpha()
{
    double aux;
    if (alpha == 0)
        aux = 0.05;
    if (alpha == 1)
        aux = 0.01;
    return aux;
}
```

Apresenta-se na listagem 6.11 o arquivo com código da classe CTesteDoerffel.h

Listing 6.11: Arquivo de cabeçalho da classe CTesteDoerffel.

```
#ifndef CTesteDoerffel_h
#define CTesteDoerffel_h

#include "CTest.h"
#include <algorithm>
#include <numeric>
#include <iostream>

class CTesteDoerffel : public CTeste
{
protected:
    double valorSuspeito; // Valor suspeito de ser um outlier
    double valorAdjacente; // Valor adjacente ao dado suspeito de ser um
        outlier
    std::vector <double> dif; // Necessario para calcular o valor
        adjacente, guarda os valores das diferencas entre o valor suspeito
        e os outros
    double q; //Estatistica calculada
    double matriz [8] = {0.97, 0.84, 0.73, 0.64, 0.59, 0.54, 0.51, 0.49};
        // Vetor com os valores criticos
public:
    double ValorAdjacente(); // Metodo que identifica o valor adjacente ao
        dado suspeito
    double CalcularQ(); // Metodo que calcula a estatistica utilizada no
        teste de Doerffel
    void Compara(); //Metodo que compara o valor calculado e o valor
        critico
    double LerMatriz(); //Metodo que retorna o valor critico
    void Entrada(); //Metodo de entrada, pede os dados iniciais.

};
#endif
```

Apresenta-se na listagem 6.12 o arquivo com código da classe CTesteDoerffel.cpp

Listing 6.12: Arquivo de implementação da classe CTesteDoerffel.

```
#include "CTesteDoerffel.h"

double CTesteDoerffel::CalcularQ()
{
    int n = dados.size();
    q = (valorSuspeito - ValorAdjacente()) / Amplitudedados();
    return q;
}

void CTesteDoerffel::Compara()
{
    if (CalcularQ() > LerMatriz())
    {std::cout << "0_dado_testado_e_considerado_um_outlier" << std::endl;
      std::cout << "0_valor_do_teste_calculado_e_" << CalcularQ() << std::endl;
      std::cout << "0_valor_critico_tabelado_para_alpha=0.05_e_n=" <<
        dados.size() << "e_" << LerMatriz() << std::endl; }
    else
    {std::cout << "0_dado_testado_nao_e_considerado_um_outlier" << std::endl;
      std::cout << "0_valor_do_teste_calculado_e_" << CalcularQ() << std::endl;
      std::cout << "0_valor_critico_tabelado_para_alpha=0.05_e_n=" <<
        dados.size() << "e_" << LerMatriz() << std::endl; }
}

double CTesteDoerffel::LerMatriz()
{
    return matriz[dados.size()-3];
}

void CTesteDoerffel::Entrada()
{
    std::cout << "\n\tTeste de Doerffel\n" << std::endl;
    CEstadistica::Entrada();
    std::cout << "Agora, entre com o valor suspeito de ser um outlier: ";
    std::cin >> valorSuspeito; std::cin.get();
}

double CTesteDoerffel::ValorAdjacente()
{
    for (int i = 0; i < dados.size(); i++)
    {
        if (std::abs(valorSuspeito - dados[i]) != 0)
        {dif.push_back(std::abs(valorSuspeito - dados[i])); }
    }
}
```

```

    }
    std::vector<double>::iterator it = min_element(dif.begin(), dif.end()
    );
    valorAdjacente = dados[(it - dif.begin())];
    return valorAdjacente;
}

```

Apresenta-se na listagem 6.13 o arquivo com código da classe CTesteGrubbs.h

Listing 6.13: Arquivo de cabeçalho da classe CTesteGrubbs.

```

#ifndef CTesteGrubbs_h
#define CTesteGrubbs_h

#include <algorithm>
#include <iostream>
#include <cmath>
#include "CTest.h"

class CTesteGrubbs : public CTeste
{
protected:

    int alpha;
    double tau;
    int resp;
    double matriz[18][5] = {
        {1.148, 1.153, 1.155, 1.155, 1.555},
        {1.425, 1.463, 1.481, 1.492, 1.496},
        {1.602, 1.672, 1.715, 1.749, 1.764},
        {1.729, 1.822, 1.887, 1.944, 1.973},
        {1.828, 1.938, 2.020, 2.097, 2.139},
        {1.909, 2.032, 2.126, 2.221, 2.274},
        {1.977, 2.110, 2.215, 2.323, 2.387},
        {2.036, 2.176, 2.290, 2.410, 2.482},
        {2.088, 2.234, 2.355, 2.485, 2.564},
        {2.134, 2.285, 2.412, 2.550, 2.636},
        {2.175, 2.331, 2.462, 2.607, 2.699},
        {2.213, 2.371, 2.507, 2.659, 2.755},
        {2.247, 2.409, 2.549, 2.705, 2.806},
        {2.279, 2.443, 2.585, 2.747, 2.852},
        {2.309, 2.475, 2.620, 2.785, 2.894},
        {2.335, 2.504, 2.651, 2.821, 2.932},
        {2.361, 2.532, 2.681, 2.854, 2.968},
        {2.385, 2.557, 2.709, 2.884, 3.001}
    };

public:

```

```

void OrdenarDados(); //Metodo que ordena os dados
void LogTransform(); //Metodo que transforma vetor de dados no logaritmo
    natural do vetor de dados
double CalcularTau(); //Metodo que calcula e retorna a estatistica do
    teste
void Compara(); //Metodo que compara o valor calculado e o valor critico
    tabelado
double LerMatriz(); //Metodo que retorna o valor critico tabelado
void Entrada(); //Metodo de entrada.
double Alpha(); //Metodo que retorna o valor da significancia

};
#endif

```

Apresenta-se na listagem 6.14 o arquivo com código da classe CTesteGrubbs.cpp

Listing 6.14: Arquivo de implementação da classe CTesteGrubbs.

```

#include "CTesteGrubbs.h"

void CTesteGrubbs::OrdenarDados()
{
    std::sort(dados.begin(), dados.end());
}

void CTesteGrubbs::LogTransform()
{
    for (int i = 0; i < dados.size(); i++)
        {dados[i] = std::log(dados[i]);}
}

double CTesteGrubbs::CalcularTau()
{
    double x = Media();
    double y = Desviopadrazo();
    int n = dados.size()-1;
    OrdenarDados();
    if (resp == 2)
        {tau = (x - dados[0]) / y;}
    if (resp = 1)
        {tau = (dados[n] - x) / y;}
    return tau;
}

void CTesteGrubbs::Entrada()
{
    std::cout << "\n\tTeste de Grubbs\n" << std::endl;
    CEstatistica::Entrada();
}

```

```

std::cout << "Agora, entre com a significancia: "
    << "\nDigite (0+ENTER) para alpha=0.10: "
    << "\nDigite (1+ENTER) para alpha=0.05: "
    << "\nDigite (2+ENTER) para alpha=0.025: "
    << "\nDigite (3+ENTER) para alpha=0.01: "
    << "\nDigite (4+ENTER) para alpha=0.005: ";
std::cin >> alpha; std::cin.get();
std::cout << "Deseja testar o maior valor de observacao (digite 1+ENTER)"
    << "\nDeseja testar o menor valor de observacao (digite 2+ENTER): ";
std::cin >> resp; std::cin.get();
}

double CTesteGrubbs::Alpha()
{
    double aux;
    if (alpha == 0)
        aux = 0.10;
    if (alpha == 1)
        aux = 0.05;
    if (alpha == 2)
        aux = 0.025;
    if (alpha == 3)
        aux = 0.01;
    if (alpha == 4)
        aux = 0.005;
    return aux;
}

double CTesteGrubbs::LerMatriz()
{
    return matriz [dados.size()-3][alpha];
}

void CTesteGrubbs::Compara()
{
    LogTransform();
    if (CalcularTau() > LerMatriz())
    {std::cout << "O valor testado com o nivel de significancia= " <<
        Alpha() << " e considerado outlier." << std::endl;
        std::cout << "O valor do teste calculado e " << CalcularTau() << std::endl;
        std::cout << "O valor critico tabelado e " << LerMatriz() << std::endl;}
    else
    {std::cout << "O valor testado com o nivel de significancia= " <<
        Alpha() << " nao e considerado outlier." << std::endl;
}

```



```

        std::cout << "0_valor_do_teste_calculado_e_" << CalcularTau() << std
        ::endl;
        std::cout << "0_valor_critico_tabelado_e_" << LerMatriz() << std::
        endl; }
    }

```

Apresenta-se na listagem 6.15 o arquivo com código da classe CTesteZmodificado.h

Listing 6.15: Arquivo de cabeçalho da classe CTesteZmodificado.

```

#ifndef CTesteZmodificado_h
#define CTesteZmodificado_h
#include "CTeste.h"
#include <vector>

class CTesteZmodificado : public CTeste
{
public:
    std::vector <double> abso; // Vetor desvio absoluto |Xi - Xm|
    std::vector <double> abso1; // Vetor desvio (Xi - Xm)
    double mad; // Mediana de |Xi - Xm|
    std::vector <double> zmod; // Vetor que armazena Zmod
    double Mad(); // Retorna a mediana de |Xi - Xm| --> mad
    void Compara(); // Compara a estatística calculada
    void CalculaZ(); // Metodo que calcula o vetor de Zmod
    void DesvioAbs(); // Metodo que preenche os vetores abso e abso1
    void Entrada(); // Metodo de entrada
};

#endif

```

Apresenta-se na listagem 6.16 o arquivo com código da classe CTesteZmodificado.cpp

Listing 6.16: Arquivo de implementação da classe CTesteZmodificado.

```

#include <iostream>
#include <algorithm>
#include "CTesteZmodificado.h"

void CTesteZmodificado::CalculaZ()
{
    double x4 = Mad();
    for (int i = 0; i < dados.size(); i++)
    { zmod.push_back((0.6745*abso1[i]) / x4); }
}

void CTesteZmodificado::DesvioAbs()
{

```

```

    double x5 = Mediana();
    for (int i = 0; i < dados.size(); i++)
    {
        double dabs = dados[i] - x5;
        abso1.push_back(dabs);
        abso.push_back(std::abs(dabs));
    }
}

double CTesteZmodificado::Mad()
{
    std::sort(abso.begin(), abso.end());
    int n = abso.size()-1;
    if (abso.size()%2 == 0)
        {mad = (abso[n/2] + abso[(n/2)+1])/2;}
    else
        {mad = abso[(n+1)/2];}
    return mad;
}

void CTesteZmodificado::Compara()
{
    DesvioAbs();
    CalculaZ();
    for (int i = 0; i < dados.size(); i++)
    {
        if (zmod[i] > 3)
            {std::cout << "Considera-se como outlier o seguinte dado: " <<
                dados[i] << ", que tem como Zmod: " << zmod[i] << std::endl;}
    }
}

void CTesteZmodificado::Entrada()
{
    std::cout << "\n\tTeste_Z_modificado\n" << std::endl;
    CEstatistica::Entrada();
}

```

Apresenta-se na listagem 6.17 o arquivo com código da classe CTesteDixon.h

Listing 6.17: Arquivo de cabeçalho da classe CTesteDixon.

```

#ifndef CTESTEDIXON_H
#define CTESTEDIXON_H
#include "CTeste.h"
#include <algorithm>
#include <utility>

class CTesteDixon : public CTeste
{

```

```

protected:
    int alpha; // Nível de significancia, assume os valores de 0, 1 ou 2.
    double z_maior; //Estatística calculada caso o suspeito de ser outlier
        seja o maior numero
    double z_menor; // Estatística calculada caso o suspeito de ser outlier
        seja o menor numero
    int resp; //Serve apenas para indicar se vai descartar o menor ou o
        maior elemento
    double matriz[23][3] = { //0.1    0.05    0.01    <-- Niveis de
        significancia
                                {0.886, 0.941, 0.988},
                                {0.679, 0.765, 0.889},
                                {0.557, 0.642, 0.780},
                                {0.482, 0.560, 0.698},
                                {0.434, 0.507, 0.637},
                                {0.479, 0.554, 0.683},
                                {0.441, 0.512, 0.635},
                                {0.409, 0.477, 0.597},
                                {0.517, 0.576, 0.679},
                                {0.490, 0.546, 0.642},
                                {0.467, 0.521, 0.615},
                                {0.492, 0.546, 0.641},
                                {0.472, 0.525, 0.616},
                                {0.454, 0.507, 0.595},
                                {0.438, 0.490, 0.577},
                                {0.424, 0.475, 0.561},
                                {0.412, 0.462, 0.547},
                                {0.401, 0.450, 0.535},
                                {0.391, 0.440, 0.524},
                                {0.382, 0.430, 0.514},
                                {0.374, 0.421, 0.505},
                                {0.367, 0.413, 0.497},
                                {0.360, 0.406, 0.489}
                                };

public:
    void OrdenarDados(); //Metodo que ordena os dados do vetor
    double CalcularZmaior(); //Metodo que calcula a estatística do teste
        caso o suspeito de ser um outlier seja o maior valor
    double CalcularZmenor(); //Metodo que calcula a estatística do teste
        caso o suspeito de ser um outlier seja o menor valor
    double LerTabela(); // Metodo que retorna o valor critico do teste
    void Entrada(); //Metodo de entrada, pede dados iniciais
    double Alpha(); //Metodo que retorna a significancia --> 0.10, 0.05 e
        0.01
    void Compara(); //Metodo que compara a estatística calcula e o valor
        critico tabelado
};

```

```
#endif
```

Apresenta-se na listagem 6.18 o arquivo com código da classe CTesteDixon.cpp

Listing 6.18: Arquivo de implementação da classe CTesteDixon.

```
#include "CTesteDixon.h"

void CTesteDixon::OrdenarDados()
{
    std::sort(dados.begin(), dados.end());
}

double CTesteDixon::CalcularZmaior()
{
    OrdenarDados();
    int n = dados.size() - 1;
    if (n+1 >= 3 and n+1 <= 7)
    {z_maior = (dados[n] - dados[n-1])/(dados[n] - dados[0]);}
    else if (n+1 >= 8 and n+1 <= 12)
    {z_maior = (dados[n] - dados[n-1])/(dados[n] - dados[1]);}
    else if (n+1 <= 13 and n+1 <= 25)
    {z_maior = (dados[n] - dados[n-2])/(dados[n] - dados[2]);}
    else
    {std::cerr << "O teste de Dixon nao esta definido para mais de 25 observacoes." << std::endl;}

    return z_maior;
}

double CTesteDixon::CalcularZmenor()
{
    OrdenarDados();
    int n = dados.size() - 1;
    if (n+1 >= 3 and n+1 <= 7)
    {z_menor = (dados[1] - dados[0])/(dados[n] - dados[0]);}
    else if (n+1 >= 8 and n+1 <= 12)
    {z_menor = (dados[1] - dados[0])/(dados[n-1] - dados[0]);}
    else if (n+1 <= 13 and n+1 <= 25)
    {z_menor = (dados[2] - dados[0])/(dados[n-2] - dados[0]);}
    else
    {std::cerr << "O teste de Dixon nao esta definido para mais de 25 observacoes." << std::endl;}

    return z_menor;
}

double CTesteDixon::LerTabela()
{
    return matriz [dados.size()-3][alpha];
}
```

```

}

double CTesteDixon::Alpha()
{
    double aux2;
    if (alpha == 0)
        aux2 = 0.10;
    if (alpha == 1)
        aux2 = 0.05;
    if (alpha == 2)
        aux2 = 0.01;

    return aux2;
}

void CTesteDixon::Compara()
{
    if (resp == 1)
    {
        if (CalcularZmaior() > LerTabela())
        {
            std::cout << "0_maior_valor_testado_com_um_nivel_de_significancia_
                de_" << Alpha() << "e_considerado_um_outlier" << std::endl;
            std::cout << "0_valor_calculado_e_" << CalcularZmaior() << std::
                endl;
            std::cout << "0_valor_critico_tabelado_para_esse_valor_de_
                significancia_e_" << LerTabela() << std::endl;
            std::cout << "0_maior_valor_foi_apagado:" << dados[dados.size()-1]
                << std::endl;
            dados.pop_back();}
        else
        { std::cout << "0_maior_valor_testado_com_um_nivel_de_significancia_
                de_" << Alpha() << "nao_e_considerado_um_outlier" << std::endl;
            std::cout << "0_valor_calculado_e_" << CalcularZmaior() << std::
                endl;
            std::cout << "0_valor_critico_tabelado_para_esse_valor_de_
                significancia_e_" << LerTabela() << std::endl;
            std::cout << "0_maior_valor_foi_mantido:" << dados[dados.size()-1]
                << std::endl; }
    }
    if (resp == 2)
    {
        if (CalcularZmenor() > LerTabela())
        {
            std::cout << "0_menor_valor_testado_com_um_nivel_de_significancia_
                de_" << Alpha() << "e_considerado_um_outlier" << std::endl;
            std::cout << "0_valor_calculado_e_" << CalcularZmenor() << std::
                endl;

```

```

        std::cout << "0_valor_critico_tabelado_para_esse_valor_de_
            significancia_e_" << LerTabela() << std::endl;
        std::cout << "0_menor_valor_foi_apagado:" << dados[0] << std::endl
            ;
        dados.erase (dados.begin());}
    else
    {std::cout << "0_menor_valor_testado_com_um_nivel_de_significancia_de_
        _" << Alpha() << "_nao_e_considerado_um_outlier" << std::endl;
        std::cout << "0_valor_calculado_e_" << CalcularZmenor() << std::endl
            ;
        std::cout << "0_valor_critico_tabelado_para_esse_valor_de_
            significancia_e_" << LerTabela() << std::endl;
        std::cout << "0_menor_valor_foi_mantido:" << dados[0] << std::endl
            ;}
    }

    std::cout << "Gostaria_de_realizar_um_novo_teste?(s_ou_n):_";
    char auxiliar; std::cin >> auxiliar; std::cin.get();
    if (auxiliar == 's')
    {std::cout << "Deseja_testar_o_maior_valor_de_observacao_(digite_1_+_
        ENTER)_"
            << "\nDeseja_testar_o_menor_valor_de_observacao_(digite_2_+_
                _ENTER):_";
        std::cin >> resp; std::cin.get();
        Compara();}
}

void CTesteDixon::Entrada()
{
    std::cout << "\n\tTeste_de_Dixon\n" << std::endl;
    std::cout << "Entre_com_o_nivel_de_significancia_" << std::endl;
    std::cout << "Digite_(0_+_ENTER)_para_alpha=_0.10_\n"
        << "Digite_(1_+_ENTER)_para_alpha=_0.05_\n"
        << "Digite_(2_+_ENTER)_para_alpha=_0.01_: ";
    std::cin >> alpha; std::cin.get();
    CEstatistica::Entrada();
    std::cout << "Deseja_testar_o_maior_valor_de_observacao_(digite_1_+_
        ENTER)_"
            << "\nDeseja_testar_o_menor_valor_de_observacao_(digite_2_+_
                _ENTER):_";
    std::cin >> resp; std::cin.get();
}

```

Apresenta-se na listagem 6.19 o arquivo com código da classe CDist_Normal.h

Listing 6.19: Arquivo de cabeçalho da classe CDistNormal.

```

#ifndef CDISTNORMAL_H
#define CDISTNORMAL_H

#include <iostream>

```

```
#include "CEstatistica.h"

class CDistNormal : public CEstatistica
{
protected:
    double area; // Area que representa a probabilidade acumulada
    double x; // Abcissa da distribuicao normal
    double y; // Ordenada da distribuicao normal
public:
    CDistNormal():area(0.0), x(0.0), y(0.0){}; //Construtor

    double f(double _x); //Calcula y para distribuicao normal

    double erfinv(double x); // Aproximacao da funcao erro inversa por
        series de Taylor

    double Area(double _x); // Calcula a area a esquerda do parametro _x
        utilizando a funcao erro

    double InvArea(double _x); //O usuario entra com o valor da area e o
        metodo retorna a abcissa correspondente
};

#endif
```

Apresenta-se na listagem 6.20 o arquivo com código da classe CDist_Normal.cpp

Listing 6.20: Arquivo de implementação da classe CDistNormal.

```
#include <iostream>
#include <cmath>

#include "CDistNormal.h"

double CDistNormal::f(double _x)
{
    x = _x;
    y = 1./sqrt(2.*M_PI*Variancia()) * pow(M_E, -((x-Media()*(x - Media()
        ))/2.*Variancia()));
    return y;
}

double CDistNormal::Area(double _x)
{
    x = _x;
    area = 0.5*(1 - erf ((-x) / sqrt(2)));
    return area;
}
```

```
double CDistNormal::erfinv(double x) // Aproximacao da inversa da funcao
    erro
{
    double m;
    m = 0.5*sqrt(M_PI)* ( x + M_PI*x*x*x/12 + 7*M_PI*M_PI*pow(x,5)/480 +
        127*pow(M_PI,3)*pow(x,7)/40320 + 4369*pow(M_PI,4)*pow(x,9)/5806080
        +
        34807*pow(M_PI,5)*pow(x,11)/182476800 );
    return m;
}

double CDistNormal::InvArea(double _x)
{
    area = _x;
    x = -sqrt(2)*(erfinv (1-(2*area))));
    return x;
}
```

Apresenta-se na listagem 6.21 o arquivo com código da classe CTesteHipotese.h

Listing 6.21: Arquivo de cabeçalho da classe CTesteHipotese.

```
#ifndef CTesteHipotese_h
#define CTesteHipotese_h

#include <iostream>
#include "CDistNormal.h"

class CTesteHipotese : public CDistNormal
{
protected:
double h0; //Hipotese nula
int caso; // caso --> unicaudal a esquerda, unicaudal a direita ou
    bicaudal
public:
virtual void Compara() = 0;
virtual void Entrada() = 0;
virtual double Calcular() = 0;
};

#endif
```

Apresenta-se na listagem 6.22 o arquivo com código da classe CTesteHipotese.cpp

Listing 6.22: Arquivo de implementação da classe CTesteHipotese.

```
#include "CTesteHipotese.h"
```

Apresenta-se na listagem 6.23 o arquivo com código da classe CTesteHipoteseDPconhecido.h

Listing 6.23: Arquivo de cabeçalho da classe CTesteHipoteseDPconhecido.

```
#ifndef CTesteHipoteseDPconhecido_h
#define CTesteHipoteseDPconhecido_h

#include <iostream>
#include <cmath>
#include "CTesteHipotese.h"

class CTesteHipoteseDPconhecido : public CTesteHipotese
{
protected:
double sigma; // Desvio padrao conhecido da populacao
double zcalc; // Zcalc - valor calculado
double alpha; // Identifica o nivel de significancia
public:
double Calcular(); //Calcula Zcalc
void Compara(); //Compara Zcalc e Z_alpha ou Z_alpha/2
void Entrada(); // Metodo de Entrada
};

#endif
```

Apresenta-se na listagem 6.24 o arquivo com código da classe CTesteHipoteseDPconhecido.cpp

Listing 6.24: Arquivo de implementação da classe CTesteHipoteseDPconhecido.

```
#include "CTesteHipoteseDPconhecido.h"

double CTesteHipoteseDPconhecido::Calcular()
{
    zcalc = (Media() - h0) / (sigma / sqrt(dados.size()));
    return zcalc;
}

void CTesteHipoteseDPconhecido::Entrada()
{
    std::cout << "\n\tTeste de hipoteses com desvio padrao conhecido\n" <<
        std::endl;
    CEstatistica::Entrada();
    std::cout << "\nA media amostral e" << Media() << std::endl;
    std::cout << "Agora, enuncie a hipotese nula:";
    std::cin >> h0; std::cin.get();
    std::cout << "Entre com o desvio padrao da populacao:";
    std::cin >> sigma; std::cin.get();
    std::cout << "Entre com o caso desejado:\n"
```

```

        << "Unicaudal_a esquerda(digite_1+_ENTER):_\n"
        << "Unicaudal_a direita(digite_2+_ENTER):_\n"
        << "Bilateral(digite_3+_ENTER):_";
std::cin >> caso; std::cin.get();
std::cout << "Entre_com_o_nivel_de_significancia:_";
std::cin >> alpha; std::cin.get();
}

void CTesteHipoteseDPconhecido::Compara()
{
    if (caso == 1)
    {
        std::cout << "\n\tTeste_unicaudal_a esquerda_realizado\n" << std::endl;
        if (Calcular() < InvArea(alpha) )
        {std::cout << "A_hipotese_nula_Ho_mi=_ " << h0 << "_deve_ser_rejeitada.\n";
        std::cout << "Zcalc=_ " << Calcular() << std::endl;
        std::cout << "Valor_tabelado_de_Z_para_alpha=_ " << alpha << ",_e_"
            << InvArea(alpha) << std::endl; }
        else
        {std::cout << "A_hipotese_nula_Ho_mi=_ " << h0 << "_nao_deve_ser_rejeitada.\n";
        std::cout << "Zcalc=_ " << Calcular() << std::endl;
        std::cout << "Valor_tabelado_de_Z_para_alpha=_ " << alpha << ",_e_"
            << InvArea(alpha) << std::endl; }
    }
    if (caso == 2)
    {
        std::cout << "\n\tTeste_unicaudal_a direita_realizado\n" << std::endl;
        if (Calcular() > InvArea(1 - alpha))
        {std::cout << "A_hipotese_nula_Ho_mi=_ " << h0 << "_deve_ser_rejeitada.\n";
        std::cout << "Zcalc=_ " << Calcular() << std::endl;
        std::cout << "Valor_tabelado_de_Z_para_(1-_alpha)_=_ " << 1-alpha
            << ",_e_" << InvArea(1 - alpha) << std::endl; }
        else
        {std::cout << "A_hipotese_nula_Ho_mi=_ " << h0 << "_nao_deve_ser_rejeitada.\n";
        std::cout << "Zcalc=_ " << Calcular() << std::endl;
        std::cout << "Valor_tabelado_de_Z_para_(1-alpha)_=_ " << 1-alpha
            << ",_e_" << InvArea(1 - alpha) << std::endl; }
    }
    if (caso == 3)
    {
        std::cout << "\n\tTeste_bicaudal_realizado\n" << std::endl;

```

```

    if (Calcular() > InvArea(1 - (alpha/2)) or Calcular() < InvArea(
        alpha/2))
    {std::cout << "A hipotese nula Ho:  $\mu = \mu_0$ " << h0 << " deve ser rejeitada.\n";
    std::cout << "Zcalc=" << Calcular() << std::endl;
    std::cout << "Valor tabelado de Z para  $(1 - \alpha/2)$ " << 1 - (alpha/2) << ", e" << InvArea(1 - (alpha/2)) << std::endl;
    std::cout << "Valor tabelado de Z para  $\alpha/2$ " << alpha/2 << ", e" << InvArea(alpha/2) << std::endl;
    }
    else
    {std::cout << "A hipotese nula Ho:  $\mu = \mu_0$ " << h0 << " nao deve ser rejeitada.\n";
    std::cout << "Zcalc=" << Calcular() << std::endl;
    std::cout << "Valor tabelado de Z para  $(1 - \alpha/2)$ " << 1 - (alpha/2) << ", e" << InvArea(1 - (alpha/2)) << std::endl;
    std::cout << "Valor tabelado de Z para  $\alpha/2$ " << alpha/2 << ", e" << InvArea(alpha/2) << std::endl;}
}
}

```

Apresenta-se na listagem 6.25 o arquivo com código da classe CTesteHipoteseDPdesconhecido.h

Listing 6.25: Arquivo de cabeçalho da classe CTesteHipoteseDPdesconhecido.

```

#ifndef CTesteHipoteseDPdesconhecido_h
#define CTesteHipoteseDPdesconhecido_h

#include <iostream>
#include <cmath>

#include "CTesteHipotese.h"

class CTesteHipoteseDPdesconhecido: public CTesteHipotese
{
protected:
    int alpha; // Valor para identificar o nivel de significancia
    double tcalc; // Estatistica calculada
    double matriz [30][7]= { // 0.50    0.25    0.10    0.05    0.025    0.01
        0.005    <- valores de significancia (Tabela bicaudal)
        {1.00000, 2.4142, 6.3138, 12.706, 25.542, 63.657, 127.32},
        {0.81650, 1.6036, 2.9200, 4.3127, 6.2053, 9.9248, 14.089},
        {0.76489, 1.4226, 2.3534, 3.1825, 4.1765, 5.8409, 7.4533},
    };
}

```

```

{0.74070, 1.3444, 2.1318, 2.7764, 3.4954, 4.6041,
 5.5976},
{0.72669, 1.3009, 2.0150, 2.5706, 3.1634, 4.0321,
 4.7733},
{0.71756, 1.2733, 1.9432, 2.4469, 2.9687, 3.7074,
 4.3168},
{0.71114, 1.2543, 1.8946, 2.3646, 2.8412, 3.4995,
 4.0293},
{0.70639, 1.2403, 1.8595, 2.3060, 2.7515, 3.3554,
 3.8325},
{0.70272, 1.2297, 1.8331, 2.2622, 2.6850, 3.2498,
 3.6897},
{0.69981, 1.2213, 1.8125, 2.2281, 2.6338, 3.1693,
 3.5814},
{0.69745, 1.2145, 1.7959, 2.2010, 2.5931, 3.1058,
 3.4966},
{0.69548, 1.2089, 1.7823, 2.1788, 2.5600, 3.9545,
 3.4284},
{0.69384, 1.2041, 1.7709, 2.1604, 2.5326, 3.0123,
 3.3725},
{0.69200, 1.2001, 1.7613, 2.1448, 2.5096, 2.9768,
 3.3257},
{0.69120, 1.1967, 1.7530, 2.1315, 2.4899, 2.9467,
 3.2860},
{0.69013, 1.1937, 1.7459, 2.1199, 2.4729, 2.9208,
 3.2520},
{0.68919, 1.1910, 1.7396, 2.1098, 2.4581, 2.8982,
 3.2225},
{0.68837, 1.1887, 1.7341, 2.1009, 2.4450, 2.8784,
 3.1966},
{0.68763, 1.1866, 1.7291, 2.0930, 2.4334, 2.8609,
 3.1737},
{0.68696, 1.1848, 1.7247, 2.0860, 2.4231, 2.8453,
 3.1534},
{0.68635, 1.1831, 1.7207, 2.0796, 2.4138, 2.8314,
 3.1352},
{0.68580, 1.1816, 1.7171, 2.0739, 2.4055, 2.8188,
 3.1188},
{0.68531, 1.1802, 1.7139, 2.0687, 2.3979, 2.8073,
 3.1040},
{0.68485, 1.1789, 1.7109, 2.0639, 2.3910, 2.7969,
 3.0905},
{0.68443, 1.1777, 1.7081, 2.0595, 2.3846, 2.7874,
 3.0782},
{0.68405, 1.1766, 1.7056, 2.0555, 2.3788, 2.7787,
 3.0669},
{0.68370, 1.1757, 1.7033, 2.0518, 2.3734, 2.7707,
 3.0565},

```

```

        {0.68335, 1.1748, 1.7011, 2.0484, 2.3685, 2.7633,
          3.0469},
        {0.68304, 1.1739, 1.6991, 2.0452, 2.3638, 2.7564,
          3.0380},
        {0.68276, 1.1731, 1.6973, 2.0423, 2.3596, 2.7500,
          3.0298}
    };

public:
    double Calcular(); // Metodo que calcula a estatistica do teste
    double Alpha(); // Metodo que retorna o valor da significancia
    void Entrada(); // Metodo de entrada
    double LerMatriz(int); // Metodo que retorna o valor critico do teste
    void Compara(); // Metodo que compara a estatistica do teste e o valor
        critico
};

#endif

```

Apresenta-se na listagem 6.26 o arquivo com código da classe CTesteHipoteseDPdesconhecido.cpp

Listing 6.26: Arquivo de implementação da classe CTesteHipoteseDPdesconhecido.

```

#include "CTesteHipoteseDPdesconhecido.h"

double CTesteHipoteseDPdesconhecido::Calcular()

{
    tcalc = (Media() - h0) / (Desviopadiao()/sqrt(dados.size()));
    return tcalc;
}

void CTesteHipoteseDPdesconhecido::Entrada()
{
    std::cout << "\n\tTeste de hipoteses com desvio padiao desconhecido\n"
        << std::endl;
    CEstatistica::Entrada();
    std::cout << "A media amostral e " << Media() << std::endl;
    std::cout << "O desvio padiao da amostra e " << Desviopadiao() << std
        ::endl;
    std::cout << "Agora, enuncie a hipotese nula:";
    std::cin >> h0; std::cin.get();
    std::cout << "Entre com o caso desejado:\n"
        << "Unilateral ou unicaudal a esquerda (digite 1):\n"
        << "Unilateral ou unicaudal a direita (digite 2):\n"
        << "Bilateral (digite 3):";
    std::cin >> caso; std::cin.get();
    std::cout << "Entre com o nivel de significancia\n"
        << "Digite 0 + ENTER para alpha = 0.50\n"

```

```

        << "Digite 1 + ENTER para alpha = 0.25\n"
        << "Digite 2 + ENTER para alpha = 0.10\n"
        << "Digite 3 + ENTER para alpha = 0.05\n"
        << "Digite 4 + ENTER para alpha = 0.025\n"
        << "Digite 5 + ENTER para alpha = 0.01\n"
        << "Digite 6 + ENTER para alpha = 0.005: ";
    std::cin >> alpha; std::cin.get();
}

double CTesteHipoteseDPdesconhecido::Alpha()
{
    double x2;
    if (alpha == 0)
        x2 = 0.50;
    if (alpha == 1)
        x2 = 0.25;
    if (alpha == 2)
        x2 = 0.10;
    if (alpha == 3)
        x2 = 0.05;
    if (alpha == 4)
        x2 = 0.025;
    if (alpha == 5)
        x2 = 0.01;
    if (alpha == 6)
        x2 = 0.005;
    return x2;
}

double CTesteHipoteseDPdesconhecido::LerMatriz(int x3)
{
    return matriz [dados.size()-2][x3];
}

void CTesteHipoteseDPdesconhecido::Compara()
{
    if (caso == 1)
    {
        std::cout << "\n\tTeste unicaudal a esquerda realizado\n" << std::endl;
        if (Calcular() < -LerMatriz(alpha-1))
        {std::cout << "A hipotese nula Ho: mi = " << h0 << " deve ser rejeitada.\n";
        std::cout << "tcalc = " << Calcular() << std::endl;
        std::cout << "Valor tabelado de t para phi = " << dados.size()-1 << " e alpha = " << Alpha() << ", e " << LerMatriz(alpha-1) << std::endl; }
        else

```

```

        {std::cout << "A hipotese nula  $H_0: \mu = \mu_0$ " << h0 << " não deve ser rejeitada.\n";
        std::cout << "tcalc=" << Calcular() << std::endl;
        std::cout << "Valor tabelado de  $t$  para  $\phi = \nu$ " << dados.size()-1 <<
            << " e  $\alpha = \alpha$ " << Alpha() << ", e" << LerMatriz(alpha-1) <<
            std::endl;}
    }
    if (caso == 2)
    {
        std::cout << "\n\tTeste unicaudal a direita realizado\n" << std::endl;
        if (Calcular() > LerMatriz(alpha-1))
        {std::cout << "A hipotese nula  $H_0: \mu = \mu_0$ " << h0 << " deve ser rejeitada.\n";
        std::cout << "tcalc=" << Calcular() << std::endl;
        std::cout << "Valor tabelado de  $t$  para  $\phi = \nu$ " << dados.size()-1 <<
            << " e  $\alpha = \alpha$ " << Alpha() << ", e" << LerMatriz(alpha-1) << std::endl;}
        else
        {std::cout << "A hipotese nula  $H_0: \mu = \mu_0$ " << h0 << " não deve ser rejeitada.\n";
        std::cout << "tcalc=" << Calcular() << std::endl;
        std::cout << "Valor tabelado de  $t$  para  $\phi = \nu$ " << dados.size()-1 <<
            << " e  $\alpha = \alpha$ " << Alpha() << ", e" << LerMatriz(alpha-1) <<
            std::endl;}
    }
    if (caso == 3)
    {
        std::cout << "\n\tTeste bicaudal realizado\n" << std::endl;
        if (Calcular() > LerMatriz(alpha) or Calcular() < -LerMatriz(alpha))
        {std::cout << "A hipotese nula  $H_0: \mu = \mu_0$ " << h0 << " deve ser rejeitada.\n";
        std::cout << "tcalc=" << Calcular() << std::endl;
        std::cout << "Valor tabelado de  $t$  para  $\phi = \nu$ " << dados.size()-1 <<
            << " e  $\alpha = \alpha$ " << Alpha() << ", e" << LerMatriz(alpha) << std::endl;
        }
        else
        {std::cout << "A hipotese nula  $H_0: \mu = \mu_0$ " << h0 << " não deve ser rejeitada.\n";
        std::cout << "tcalc=" << Calcular() << std::endl;
        std::cout << "Valor tabelado de  $t$  para  $\phi = \nu$ " << dados.size()-1 <<
            << " e  $\alpha = \alpha$ " << Alpha() << ", e" << LerMatriz(alpha) <<
            std::endl;
        }
    }
}
}

```

Apresenta-se na listagem 6.27 o arquivo com código da classe main.cpp

Listing 6.27: main.cpp.

```
#include <iostream>
#include "CRegressaoLinear.h"
#include "CGnuplot.h"
#include "CTesteCochran.h"
#include "CTesteZmodificado.h"
#include "CTesteDixon.h"
#include "CTesteGrubbs.h"
#include "CTesteDoerffel.h"
#include "CTesteHipoteseDPconhecido.h"
#include "CTesteHipoteseDPdesconhecido.h"

int main()
{
    bool repeat = false;
    std::cout << "\t\nEsse programa realiza testes de hipotese e testes para
        identificacao de Outliers\n" << std::endl;
    std::cout << "0 que gostaria de fazer?\n"
        << "Para realizar testes de hipotese, digite 0+ENTER:\n"
        << "Para realizar testes de identificacao de Outliers, digite 1+ENTER:";

    int resp1; std::cin >> resp1; std::cin.get();
    if (resp1 == 0)
    {
        do
        {
            std::cout << "0 teste de hipoteses pode ser realizado de duas
                formas:\n"
                << "Quando o desvio padrao da populacao e conhecido ou
                quando o desvio padrao e desconhecido.\n"
                << "Para realizar o teste de hipoteses quando o desvio
                padrao e desconhecido, digite 0+ENTER:\n"
                << "Para realizar o teste de hipoteses quando o desvio
                padrao populacional e conhecido, digite 1+ENTER:"
                << "\n";

            int resp2; std::cin >> resp2; std::cin.get();
            CTesteHipotese* teste;

            switch (resp2){
            case 0:
                teste = new CTesteHipoteseDPdesconhecido;
                teste->Entrada();
                teste->Compara();
                repeat = false;
            }
```



```

        break;
    case 1:
        teste = new CTesteHipoteseDPconhecido;
        teste->Entrada();
        teste->Compara();
        repeat = false;
        break;
    default:
        std::cerr << "Entrada_invalida." << std::endl;
        repeat = true;
        break;}
}while (repeat == true);

}

else if (resp1 == 1)
{

do
{
    std::cout << "0_teste_de_identificacao_de_Outliers_pode_ser_realizado_por_varios_metodos_diferentes.\n"
        << "Para_usar_o_teste_de_Dixon, digite 0 + \n"
        << "ENTER: \n"
        << "Para_usar_o_teste_de_Cochran, digite 1 + \n"
        << "ENTER: \n"
        << "Para_usar_o_teste_de_Grubbs, digite 2 + \n"
        << "ENTER: \n"
        << "Para_usar_o_teste_Z_modificado, digite 3 + \n"
        << "ENTER: \n"
        << "Para_usar_o_teste_de_Doerffel, digite 4 + \n"
        << "ENTER: ";
    int resp3; std::cin >> resp3; std::cin.get();
    CTeste* obj;

    switch (resp3){
    case 0:
        obj = new CTesteDixon;
        obj->Entrada();
        obj->Compara();
        repeat = false;
        break;
    case 1:
        std::cout << "\n\t0_teste_de_Cochran_e_realizado_para_testar_a_homogeneidade_da_variancia_de_varios_grupos_de_dados.\n" << std::endl;
        obj = new CTesteCochran;
        obj->Entrada();

```

```
        obj->Compara();
        repeat = false;
        break;
    case 2:
        obj = new CTesteGrubbs;
        obj->Entrada();
        obj->Compara();
        repeat = false;
        break;
    case 3:
        obj = new CTesteZmodificado;
        obj->Entrada();
        obj->Compara();
        repeat = false;
        break;
    case 4:
        obj = new CTesteDoerffel;
        obj->Entrada();
        obj->Compara();
        repeat = false;
        break;
    default:
        std::cerr << "Entrada_invalida." << std::endl;
        repeat = true;
        break;}
}while (repeat == true);

}
else
{
    std::cerr << "Entrada_invalida." << std::endl;
    main();
}

std::cout << "Deseja_sair_do_programa?s_ou_n:" ;
char resp4; std::cin >> resp4; std::cin.get();
if (resp4 == 's')
return 0;
else
    main();
}
```

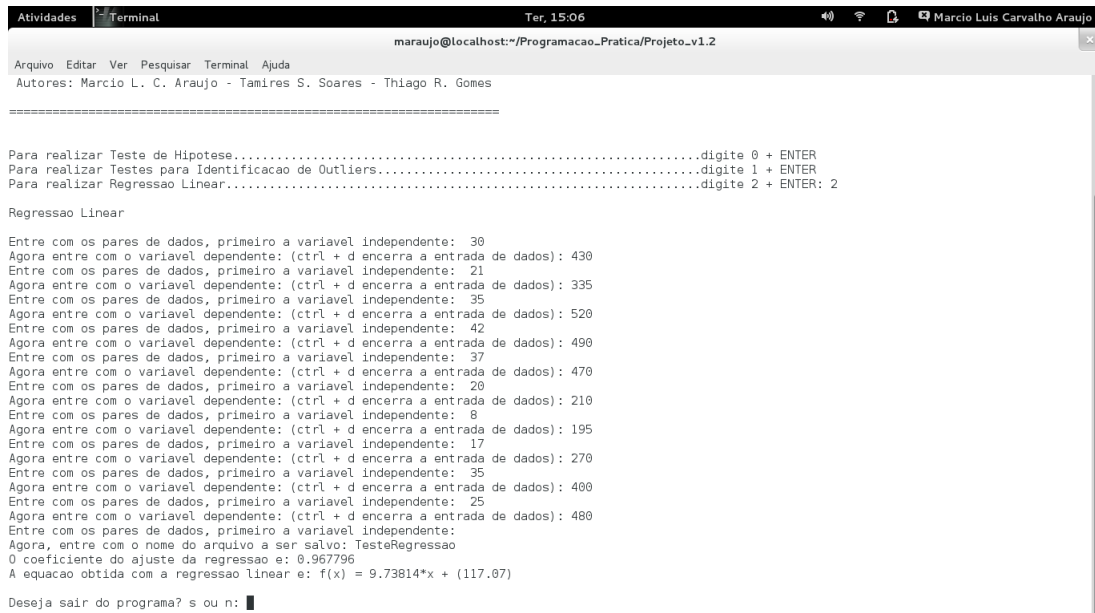
Capítulo 7

Teste

Neste capítulo iremos apresentar a execução dos programas.

7.1 Teste 1: Descrição

As Figuras 7.1 e 7.2 mostram a execução classe `CRegressaoLinear`, onde o usuário escolhe a opção de realizar a regressão linear e em seguida entra com os pares de dados (variável independente e variável dependente). Com o comando `ctrl + d` no GNU/Linux ou `ctrl + z` no Windows, o usuário encerra a entrada de dados e obtém o coeficiente do ajuste da regressão, a equação da reta e o gráfico da reta com os pontos inseridos inicialmente. Exemplo extraído de [1].



```
Atividades Terminal Ter, 15:06 maraujo@localhost:~/Programacao.Pratica/Projeto_v1.2
Arquivo Editar Ver Pesquisar Terminal Ajuda
Autores: Marcio L. C. Araujo - Tamires S. Soares - Thiago R. Gomes

=====

Para realizar Teste de Hipotese.....digite 0 + ENTER
Para realizar Testes para Identificacao de Outliers.....digite 1 + ENTER
Para realizar Regressao Linear.....digite 2 + ENTER: 2

Regressao Linear

Entre com os pares de dados, primeiro a variavel independente: 30
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 430
Entre com os pares de dados, primeiro a variavel independente: 21
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 335
Entre com os pares de dados, primeiro a variavel independente: 35
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 520
Entre com os pares de dados, primeiro a variavel independente: 42
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 490
Entre com os pares de dados, primeiro a variavel independente: 37
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 470
Entre com os pares de dados, primeiro a variavel independente: 20
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 210
Entre com os pares de dados, primeiro a variavel independente: 8
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 195
Entre com os pares de dados, primeiro a variavel independente: 17
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 270
Entre com os pares de dados, primeiro a variavel independente: 35
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 400
Entre com os pares de dados, primeiro a variavel independente: 25
Agora entre com o variavel dependente: (ctrl + d encerra a entrada de dados): 480
Entre com os pares de dados, primeiro a variavel independente:
Agora, entre com o nome do arquivo a ser salvo: TesteRegressao
0 coeficiente do ajuste da regressao e: 0.967796
A equacao obtida com a regressao linear e: f(x) = 9.73814*x + (117.07)

Deseja sair do programa? s ou n: █
```

Figura 7.1: Tela do programa mostrando a execução da classe `CRegressaoLinear`

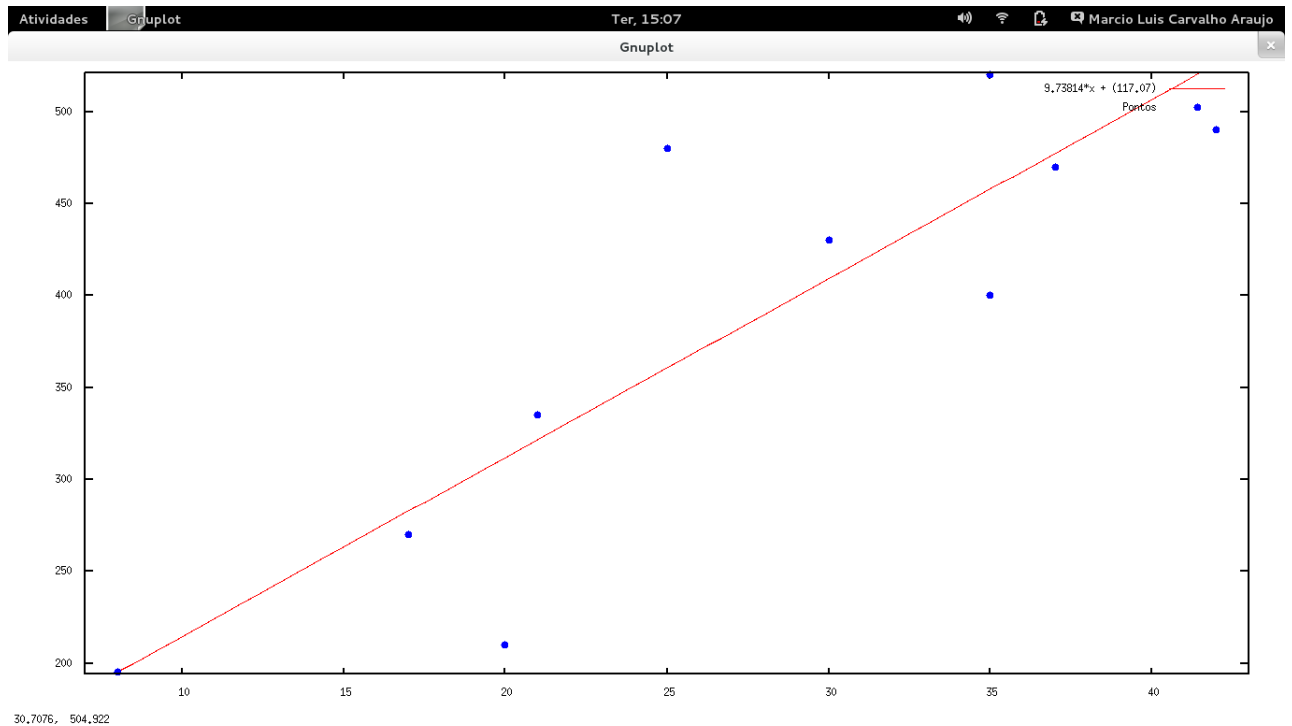


Figura 7.2: Tela do programa mostrando o gráfico obtido pela classe CRegressaoLinear

Apresenta-se na listagem 7.1 o arquivo gerado pela classe CRegressaoLinear.

Listing 7.1: Arquivo gerado pela classe CRegressaoLinear.

```
Pontos
(30, 430)
(21, 335)
(35, 520)
(42, 490)
(37, 470)
(20, 210)
(8, 195)
(17, 270)
(35, 400)
(25, 480)
0 coeficiente do ajuste da regressao e: 0.967796
A equacao obtida com a regressao linear e: f(x) = 9.73814*x + (117.07)
```

7.2 Teste 2: Descrição

A Figura 7.3 mostra a sequência de execução da classe CTesteCochran onde, o usuário entra a com a opção 1 para realizar os testes para identificação de *outliers* e depois escolhe a opção 1 para realizar o teste de Cochran. O usuário entra com o número de grupos e depois insere os dados e termina a inserção dos mesmos com o comando *ctrl + d* no GNU/Linux ou *ctrl + z* no Windows. O usuário escolhe o valor do nível de significância, o programa então calcula a estatística do teste e a compara com o valor crítico tabelado

específico para o teste. Em seguida, o programa apresenta a consideração final de que algum grupo apresenta ou não, a variância não homogênea com os demais. Exemplo extraído de [2].

```

Atividades Terminal Ter, 15:12 maraujo@localhost:~/Programacao_Pratica/Projeto_v1.2
Arquivo Editar Ver Pesquisar Terminal Ajuda
Entre com os dados (ctrl + d para encerrar): 50.0071
Entre com os dados (ctrl + d para encerrar): 50.0072
Entre com os dados (ctrl + d para encerrar):
A variancia do grupo e 3e-09
Entre com os dados do grupo 2
Entre com os dados (ctrl + d para encerrar): 50.007
Entre com os dados (ctrl + d para encerrar): 50.0076
Entre com os dados (ctrl + d para encerrar): 50.0075
Entre com os dados (ctrl + d para encerrar): 50.0071
Entre com os dados (ctrl + d para encerrar): 50.0078
Entre com os dados (ctrl + d para encerrar):
A variancia do grupo e 1.15e-07
Entre com os dados do grupo 3
Entre com os dados (ctrl + d para encerrar): 50.0072
Entre com os dados (ctrl + d para encerrar): 50.0074
Entre com os dados (ctrl + d para encerrar): 50.0073
Entre com os dados (ctrl + d para encerrar): 50.0072
Entre com os dados (ctrl + d para encerrar):
A variancia do grupo e 8e-09
Entre com os dados do grupo 4
Entre com os dados (ctrl + d para encerrar): 50.0073
Entre com os dados (ctrl + d para encerrar): 50.0074
Entre com os dados (ctrl + d para encerrar): 50.0073
Entre com os dados (ctrl + d para encerrar): 50.0072
Entre com os dados (ctrl + d para encerrar):
A variancia do grupo e 7e-09
Agora, entre com o valor da significancia
Para alpha = 0.05.....digite (0 + ENTER) :
Para alpha = 0.01.....digite (1 + ENTER) : 0
Agora, entre com o nome do arquivo a ser salvo: TesteCochran
0 grupo 2 apresenta variancia nao homogenea com os demais grupos
0 valor de C calculado e 0.864662
0 valor critico tabelado para o nivel de significancia = 0.05 e 0.6287
Deseja sair do programa? s ou n: 

```

Figura 7.3: Tela mostrando a execução da classe CTesteCochran

Apresenta-se na listagem 7.2 o arquivo gerado pela classe CTesteCochran.

Listing 7.2: Arquivo gerado pela classe CTesteCochran.

```

0 grupo 2 apresenta variancia nao homogenea com os demais grupos
0 valor de C calculado e 0.864662
0 valor critico tabelado para o nivel de significancia = 0.05 e 0.6287

```

7.3 Teste 3 : Descrição

A Figura 7.4 mostra a sequência de execução do teste Z modificado, onde o usuário escolhe a opção 1 para realizar os testes de identificação de *outliers* e depois a opção 3 para realizar o teste Z modificado. O usuário insere os dados e termina sua inserção com o comando *ctrl+d* no GNU/Linux ou *ctrl+z* no Windows. O programa executa estatística do teste e compara para valores maiores que 3, considerando os valores com magnitude maior que 3 como *outlier*. Exemplo extraído de [6].

```

Atividades Terminal Ter, 15:14
maraujo@localhost:~/Programacao_Pratica/Projeto_v1.2

Arquivo Editar Ver Pesquisar Terminal Ajuda
Teste de Hipotese | Identificacao de Outliers | Regressao Linear
Autores: Marcio L. C. Araujo - Tamires S. Soares - Thiago R. Gomes

=====

Para realizar Teste de Hipotese.....digite 0 + ENTER
Para realizar Testes para Identificacao de Outliers.....digite 1 + ENTER
Para realizar Regressao Linear.....digite 2 + ENTER: 1
0 teste de identificacao de Outliers pode ser realizado por varios metodos diferentes.

Teste de Dixon.....digite 0 + ENTER:
Teste de Cochran.....digite 1 + ENTER:
Teste de Grubbs.....digite 2 + ENTER:
Teste Z Modificado.....digite 3 + ENTER:
Teste de Doerffel.....digite 4 + ENTER: 3

Teste Z modificado

Entre com os dados (ctrl + d para encerrar): 3.2
Entre com os dados (ctrl + d para encerrar): 3.3
Entre com os dados (ctrl + d para encerrar): 8.1
Entre com os dados (ctrl + d para encerrar): 3.2
Entre com os dados (ctrl + d para encerrar): 2.9
Entre com os dados (ctrl + d para encerrar): 3.7
Entre com os dados (ctrl + d para encerrar): 3.1
Entre com os dados (ctrl + d para encerrar): 3.5
Entre com os dados (ctrl + d para encerrar): 3.3
Entre com os dados (ctrl + d para encerrar): 9.2
Entre com os dados (ctrl + d para encerrar):
Agora, entre com o nome do arquivo a ser salvo: TesteZModificado
Considera-se como outlier o seguinte dado: 8.1, que tem como Zmod: 16.188
Considera-se como outlier o seguinte dado: 9.2, que tem como Zmod: 19.8978

Deseja sair do programa? s ou n: n

```

Figura 7.4: Tela mostrando a execução da classe CTesteZmodificado

Apresenta-se na listagem 7.3 o arquivo gerado pela classe CTesteZmodificado.

Listing 7.3: Arquivo gerado pela classe CTesteZmodificado.

```

Considera-se como outlier o seguinte dado: 8.1, que tem como Zmod:
16.188
Considera-se como outlier o seguinte dado: 9.2, que tem como Zmod:
19.8978

```

7.4 Teste 4 : Descrição

A Figura 7.5 mostra a sequência de execução do teste de hipóteses com desvio padrão desconhecido, onde o usuário escolhe a opção zero para execução do teste de hipótese e depois a opção 1 para realizar o teste de hipóteses com desvio padrão conhecido. O usuário insere os dados e termina inserção com o comando *ctrl + d* no GNU/Linux ou *ctrl + z* no Windows. Depois o usuário escolhe o caso do teste. O usuário entra com a hipótese nula e o programa executa a estatística do teste e compara com o valor tabelado para considerar ou não a rejeição da hipótese nula. Exemplo extraído de [3].

```

Atividades Terminal Ter, 15:17
maraujo@localhost:~/Programacao.Pratica/Projeto_v1.2

Arquivo Editar Ver Pesquisar Terminal Ajuda
Entre com os dados (ctrl + d para encerrar): 22
Entre com os dados (ctrl + d para encerrar): 22
Entre com os dados (ctrl + d para encerrar): 23
Entre com os dados (ctrl + d para encerrar): 22
Entre com os dados (ctrl + d para encerrar): 20
Entre com os dados (ctrl + d para encerrar): 24
Entre com os dados (ctrl + d para encerrar):

Media Amostral: 21.8

Desvio Padrao da Amostra: 1.39925

Agora entre com a Hipotese Nula (Ho): 20

Selecione a Regiao Critica desejada:
a) Unilateral ou unicaudal a esquerda.....(digite 0 + ENTER):
b) Unilateral ou unicaudal a direita.....(digite 1 + ENTER):
c) Bilateral.....(digite 2 + ENTER): 1

Selecione o Nivel de Significancia:
Para alpha = 0.25.....(digite 1 + ENTER)
Para alpha = 0.10.....(digite 2 + ENTER)
Para alpha = 0.05.....(digite 3 + ENTER)
Para alpha = 0.025.....(digite 4 + ENTER)
Para alpha = 0.01.....(digite 5 + ENTER)
Para alpha = 0.005.....(digite 6 + ENTER) : 3
Agora, entre com o nome do arquivo a ser salvo: TesteHipoteseDPdesconhecido

Teste de Hipotese Unicaudal a Direita realizado com sucesso!

tcalc = 5.75298

Valor tabelado de t para phi = 19 e alpha = 0.05: 1.7291

A Hipotese Nula (Ho)  $\mu_0 = 20$  deve ser rejeitada!

Deseja sair do programa? s ou n: 

```

Figura 7.5: tela mostrando a execução da classe CTesteHipoteseDPdesconhecido

Apresenta-se na listagem 7.4 o arquivo gerado pela classe CTesteHipoteseDPdesconhecido.

Listing 7.4: Arquivo gerado pela classe CTesteHipoteseDPdesconhecido.

Teste de Hipotese Unicaudal a Direita realizado com sucesso!

tcalc = 5.75298

Valor tabelado de t para phi = 19 e alpha = 0.05: 1.7291

7.5 Teste 5 : Descrição

A Figura 7.6 mostra a sequência de execução de um teste de hipóteses com o desvio padrão da população conhecido. O usuário inicia escolhendo a opção zero para execução do teste de hipótese e depois a opção 1 para realizar o teste de hipóteses com desvio padrão conhecido. O usuário insere os dados e termina inserção com o comando *ctrl + d* no GNU/Linux o *ctrl + z* no Windows. Depois, o usuário entra com o desvio padrão e escolhe o tipo do teste. O usuário entra com a hipótese nula e o programa executa a estatística do teste e compara com o valor tabelado para rejeição ou não da hipótese nula. Exemplo extraído de [4].

```

Atividades Terminal Ter, 15:20
maraujo@localhost:~/Programacao_Pratica/Projeto_v1.2

Arquivo Editar Ver Pesquisar Terminal Ajuda

Entre com os dados (ctrl + d para encerrar): 76.2
Entre com os dados (ctrl + d para encerrar): 78.3
Entre com os dados (ctrl + d para encerrar): 76.4
Entre com os dados (ctrl + d para encerrar): 74.7
Entre com os dados (ctrl + d para encerrar): 72.6
Entre com os dados (ctrl + d para encerrar): 78.4
Entre com os dados (ctrl + d para encerrar): 75.7
Entre com os dados (ctrl + d para encerrar): 70.2
Entre com os dados (ctrl + d para encerrar): 73.3
Entre com os dados (ctrl + d para encerrar): 74.2
Entre com os dados (ctrl + d para encerrar):

Media Amostral: 75

Agora entre com a Hipotese Nula (Ho): 72

Selecione a Regiao Critica desejada:
a) Unilateral ou unicaudal a esquerda.....(digite 0 + ENTER):
b) Unilateral ou unicaudal a direita.....(digite 1 + ENTER):
c) Bilateral.....(digite 2 + ENTER): 2

Entre com o Nivel de Significancia: 0.05
Entre com o Desvio Padrao da Populacao: 2
Agora, entre com o nome do arquivo a ser salvo: TesteHipoteseDPconhecido

Teste Bicaudal Realizado com sucesso!

Zcalc = 4.74342

Valor tabelado de Z para (1 - alpha/2) = 0.975: 1.79654

Valor tabelado de Z para alpha/2 = 0.025: -1.79654

A Hipotese Nula (Ho) m1 = 72 deve ser rejeitada!

Deseja sair do programa? s ou n: 

```

Figura 7.6: Tela mostrando a execução da Classe CTesteHipoteseDPconhecido

Apresenta-se na listagem 7.5 o arquivo gerado pela classe CTesteHipoteseDPconhecido.

Listing 7.5: Arquivo gerado pela classe CTesteHipoteseDPconhecido.

```
Teste Bicaudal Realizado com sucesso!
```

```
Zcalc = 4.74342
```

```
Valor tabelado de Z para (1 - alpha/2) = 0.975: 1.79654
```

```
Valor tabelado de Z para alpha/2 = 0.025: -1.79654
```

7.6 Teste 6 : Descrição

A Figura 7.7 mostra a sequência de execução do teste de Grubbs. Inicialmente, o usuário escolhe a opção 1 para realizar teste de identificação de *outlier* e depois a opção 2 para execução do Teste de Grubbs. O usuário entra com os dados e termina a inserção com *ctrl+d* no GNU/Linux ou *ctrl+z* no Windows. Depois escolhe o nível de significância e escolhe se testará o menor ou maior valor. O programa executa a estatística do teste e compara com um valor crítico tabelado e informa se o mesmo é ou não considerado um valor *outlier*. Exemplo extraído de [6].


```

Atividades - Terminal
maraujo@localhost:~/Programacao_Pratica/Projeto_v1.2

Arquivo Editar Ver Pesquisar Terminal Ajuda
Teste Z Modificado.....digite 3 + ENTER:
Teste de Doerffel.....digite 4 + ENTER: 2

Teste de Grubbs

0 teste de Grubbs esta definido para um grupo de 3 ate 20 dados.
Entre com os dados (ctrl + d para encerrar): 2.15
Entre com os dados (ctrl + d para encerrar): 11.76
Entre com os dados (ctrl + d para encerrar): 5.08
Entre com os dados (ctrl + d para encerrar): 3.12
Entre com os dados (ctrl + d para encerrar): 12.87
Entre com os dados (ctrl + d para encerrar): 32.13
Entre com os dados (ctrl + d para encerrar): 219
Entre com os dados (ctrl + d para encerrar): 19.69
Entre com os dados (ctrl + d para encerrar): 179
Entre com os dados (ctrl + d para encerrar): 9609
Entre com os dados (ctrl + d para encerrar): 327
Entre com os dados (ctrl + d para encerrar): 74.2
Entre com os dados (ctrl + d para encerrar): 102
Entre com os dados (ctrl + d para encerrar): 47.8
Entre com os dados (ctrl + d para encerrar): 8.97
Entre com os dados (ctrl + d para encerrar):
Agora, entre com a significancia:
Para alpha = 0.10.....digite (0 + ENTER) :
Para alpha = 0.05.....digite (1 + ENTER) :
Para alpha = 0.025.....digite (2 + ENTER) :
Para alpha = 0.01.....digite (3 + ENTER) :
Para alpha = 0.005.....digite (4 + ENTER) : 1

Para testar o menor valor de observacao.....(digite 0 + ENTER)
Para testar o maior valor de observacao.....(digite 1 + ENTER): 1
Agora, entre com o nome do arquivo a ser salvo: TesteGrubbs
0 valor testado com o nivel de significancia = 0.05 e considerado outlier.
0 valor do teste calculado e 2.5231
0 valor critico tabelado e 2.409

Deseja sair do programa? s ou n: 

```

Figura 7.7: Tela do Programa mostrando a execução do Teste de Grubbs

Apresenta-se na listagem 7.6 o arquivo gerado pela classe CTesteGrubbs.

Listing 7.6: Arquivo gerado pela classe CTesteGrubbs.

```

0 valor testado com o nivel de significancia = 0.05 e considerado
  outlier.
0 valor do teste calculado e 2.5231
0 valor critico tabelado e 2.409

```

7.7 Teste 7: Descrição

A Figura 7.8 mostra a execução da classe CTesteDoerffel. O usuário escolhe a opção 1 para realizar testes *outliers* e depois escolhe a opção 4 para realizar o Teste de Doerffel. O usuário insere os dados e termina a sua inserção com o comando *ctrl + d* no GNU/Linux ou *ctrl + z* no Windows. Depois o usuário entra com um valor suspeito, já inserido pelo mesmo nos dados, e o programa executa a estatística do teste e compara com o valor tabelado para concluir se o valor é considerado um *outlier* ou não. Exemplo extraído de [6].

```

Atividades Terminal Ter, 15:26 Marcio Luis Carvalho Araujo
maraujo@localhost:~/Programacao_Pratica/Projeto_v1.2

Arquivo Editar Ver Pesquisar Terminal Ajuda
Teste de Hipotese | Identificacao de Outliers | Regressao Linear
Autores: Marcio L. C. Araujo - Tamires S. Soares - Thiago R. Gomes

=====

Para realizar Teste de Hipotese.....digite 0 + ENTER
Para realizar Testes para Identificacao de Outliers.....digite 1 + ENTER
Para realizar Regressao Linear.....digite 2 + ENTER: 1
0 teste de identificacao de Outliers pode ser realizado por varios metodos diferentes.

Teste de Dixon.....digite 0 + ENTER:
Teste de Cochran.....digite 1 + ENTER:
Teste de Grubbs.....digite 2 + ENTER:
Teste Z Modificado.....digite 3 + ENTER:
Teste de Doerffel.....digite 4 + ENTER: 4

Teste de Doerffel

0 teste de Doerffel esta definido para um grupo de 3 ate 10 dados.
Entre com os dados (ctrl + d para encerrar): 0.8
Entre com os dados (ctrl + d para encerrar): 1.4
Entre com os dados (ctrl + d para encerrar): 0.7
Entre com os dados (ctrl + d para encerrar): 2.4
Entre com os dados (ctrl + d para encerrar): 4.6
Entre com os dados (ctrl + d para encerrar): 2.1
Entre com os dados (ctrl + d para encerrar): 1.5
Entre com os dados (ctrl + d para encerrar):
Agora, entre com o valor suspeito de ser um outlier: 4.6
Entrada realizada com sucesso.
Agora, entre com o nome do arquivo a ser salvo: TesteDoerffel
0 dado testado nao e considerado um outlier
0 valor do teste calculado e 0.564103
0 valor critico tabelado para alpha = 0.05 e n = 7 e 0.59

Deseja sair do programa? s ou n: 

```

Figura 7.8: Tela mostrando a execução da classe CTesteDoerffel

Apresenta-se na listagem 7.7 o arquivo gerado pela classe CTesteGrubbs.

Listing 7.7: Arquivo gerado pela classe CTesteDoerffel.

```

0 dado testado nao e considerado um outlier
0 valor do teste calculado e 0.564103
0 valor critico tabelado para alpha = 0.05 e n = 7 e 0.59

```

7.8 Teste 8: Descrição

A figura 7.9 mostra a execução da classe CTesteDixon. Inicialmente, o usuário escolhe a opção um para realizar testes de identificação de *outliers* e depois seleciona a opção zero para executar o teste de Dixon. O primeiro passo do cálculo é a escolha do nível de significância e depois a inserção do grupo de dados. O usuário encerra a entrada de dados com o comando *ctrl+d* no GNU/Linux ou *ctrl+z* no Windows. Depois escolhe se testará o maior ou o menor valor e assim é calculada a estatística do teste, comparada com o valor tabelado e concluído se é ou não considerado um valor *outlier*. Exemplo extraído de [5].

```

Atividades Terminal Ter, 15:28 maraujo@localhost:~/Programacao_Pratica/Projeto_v1.2
Arquivo Editar Ver Pesquisar Terminal Ajuda

Teste de Dixon.....digite 0 + ENTER:
Teste de Cochran.....digite 1 + ENTER:
Teste de Grubbs.....digite 2 + ENTER:
Teste Z Modificado.....digite 3 + ENTER:
Teste de Doerffel.....digite 4 + ENTER: 0

Teste de Dixon

Entre com o nivel de significancia
Para alpha = 0.10.....digite (0 + ENTER)
Para alpha = 0.05.....digite (1 + ENTER)
Para alpha = 0.01.....digite (2 + ENTER) :1
Entre com os dados (ctrl + d para encerrar): 0.700
Entre com os dados (ctrl + d para encerrar): 0.680
Entre com os dados (ctrl + d para encerrar): 0.667
Entre com os dados (ctrl + d para encerrar): 0.660
Entre com os dados (ctrl + d para encerrar): 0.690
Entre com os dados (ctrl + d para encerrar): 0.733
Entre com os dados (ctrl + d para encerrar): 0.703
Entre com os dados (ctrl + d para encerrar): 0.677
Entre com os dados (ctrl + d para encerrar):
Para testar o menor valor de observacao.....(digite 0 + ENTER)
Para testar o maior valor de observacao.....(digite 1 + ENTER): 0
Agora, entre com o nome do arquivo a ser salvo: TesteDixon
0 menor valor testado com um nivel de significancia de 0.05 nao e considerado um outlier
0 valor calculado e 0.145833
0 valor critico tabelado para esse valor de significancia e 0.554
0 menor valor foi mantido: 0.66
Gostaria de realizar um novo teste? (s ou n): s
Para testar o menor valor de observacao.....(digite 0 + ENTER)
Para testar o maior valor de observacao.....(digite 1 + ENTER): 1
0 maior valor testado com um nivel de significancia de 0.05 nao e considerado um outlier
0 valor calculado e 0.378788
0 valor critico tabelado para esse valor de significancia e 0.554
0 maior valor foi mantido: 0.733
Gostaria de realizar um novo teste? (s ou n): 

```

Figura 7.9: Tela do Programa mostrando a execuão da classe CTesteDixon

Apresenta-se na listagem 7.8 o arquivo gerado pela classe CTesteDixon.

Listing 7.8: Arquivo gerado pela classe CTesteDixon.

```

0 menor valor testado com um nivel de significancia de 0.05 nao e
    considerado um outlier
0 valor calculado e 0.145833
0 valor critico tabelado para esse valor de significancia e 0.554
0 menor valor foi mantido: 0.66

0 maior valor testado com um nivel de significancia de 0.05 nao e
    considerado um outlier
0 valor calculado e 0.378788
0 valor critico tabelado para esse valor de significancia e 0.554
0 maior valor foi mantido: 0.733

```

Capítulo 8

Documentação

A presente documentação refere-se ao uso do programa de Tratamento Estatístico de Dados Geoquímicos. Esta documentação tem o formato de uma apostila que explica passo a passo como usar o programa.

8.1 Documentação do usuário

Descreve-se aqui o manual do usuário, um guia que explica, passo a passo a forma de instalação e uso do software desenvolvido.

8.1.1 Como instalar o software

Para instalar o software execute o seguinte passo a passo:

No GNU/Linux:

- Crie uma pasta em um diretório;
- Salve todos os arquivos do programa (*.h e *.cpp) nesta pasta;
- Salve também os arquivos de interface com o **gnuplot**: **CGnuplot.h** e **CGnuplot.cpp**;
- Instale o **gnuplot** através do terminal utilizando os seguintes comandos:
 - No Fedora: `sudo yum install gnuplot`;
 - No Ubuntu: `sudo apt-get install gnuplot`.

No Windows:

- Crie uma pasta em um diretório;
 - Salve todos os arquivos do programa (*.h e *.cpp) nesta pasta;
- Salve também os arquivos de interface com o **gnuplot**: **CGnuplot.h** e **CGnuplot.cpp**;
- Instale o **gnuplot** disponível em <http://www.gnuplot.info/download.html>.

8.1.2 Como rodar o software

Através do terminal, entre na pasta onde estão os arquivos do software. Para rodá-lo no GNU/Linux, utilize os seguintes comandos nesta sequência:

```
g++ -std=c++11 *.cpp
./a.out
```

É importante lembrar que para compilar e rodar o programa é necessário que o compilador g++ da GNU esteja instalado.

Para rodar o software no Windows utilize, por exemplo, o Microsoft Visual C++. O Dev-C++, por não ser mais atualizado, não suporta algumas das inovações do C++11 que fazem parte do software.

Veja no Capítulo ?? - Teste, exemplos de uso do programa.

8.2 Documentação para desenvolvedor

Apresenta-se nesta seção a documentação para o desenvolvedor, isto é, informações para usuários que queiram modificar, aperfeiçoar ou ampliar este programa.

8.2.1 Dependências

Para compilar o programa é necessário atender as seguintes dependências:

- Instalar o compilador g++ da GNU disponível em <http://gcc.gnu.org>. Para instalar no GNU/Linux use o comando `sudo yum install gcc` (no Fedora) ou `sudo apt-get install gcc` (no Ubuntu);
- Biblioteca CGnuplot; os arquivos para acesso a biblioteca CGnuplot devem estar no diretório com os códigos do programa;
- O programa gnuplot, disponível no endereço <http://www.gnuplot.info/download.html>, deve estar instalado. É possível que haja necessidade de setar o caminho para execução do gnuplot.
- É necessário também que se tenha instalado algum editor de texto para a visualização dos arquivos de disco gerados pelo software.

8.2.2 Como gerar a documentação usando doxygen

A documentação do código do software deve ser feita usando o padrão JAVADOC, conforme apresentada no Capítulo - Documentação, do livro texto da disciplina de Programação Orientada a Objeto em C++. Depois de documentar o código, use o programa doxygen para gerar a documentação. O programa doxygen lê os arquivos com os códigos (*.h e *.cpp) e gera uma documentação muito útil e de fácil navegação no formato html.

- Veja informações sobre uso do formato JAVADOC em:
 - <http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html>
- Veja informações sobre o programa `doxygen` em
 - <http://www.stack.nl/~dimitri/doxygen/>

Passos para gerar a documentação usando o `doxygen`.

- Documente o código usando o formato JAVADOC. Um bom exemplo de código documentado é apresentado nos arquivos da biblioteca CGnuplot, abra os arquivos `CGnuplot.h` e `CGnuplot.cpp` no editor de texto e veja como o código foi documentado.
- Abra um terminal.
- Vá para o diretório onde está o código.

```
cd /caminho/para/seu/codigo
```

- Peça para o `doxygen` gerar o arquivo de definições (este arquivo que diz para o `doxygen` como deve ser a documentação).

```
doxygen -g
```

- Peça para o `doxygen` gerar a documentação.

```
doxygen
```

- Verifique a documentação gerada abrindo o arquivo `html/index.html`.

```
firefox html/index.html
```

ou

```
chrome html/index.html
```

O programa `doxywizard` é uma interface gráfica que permite a configuração do arquivo de definições (gerado pelo comando `doxygen -g`).

No GNU/Linux o `doxywizard` pode ser instalado com o comando `sudo yum install doxygen-doxywizard` (no Fedora) ou `sudo apt-get install doxygen-doxywizard` (no Ubuntu).

Apresenta-se a seguir algumas imagens com as telas das saídas geradas pelo programa `doxygen`.

Referências Bibliográficas

- [1] 88
- [2] 90
- [3] 91
- [4] 92
- [5] 95
- [6] J.L.S. Andriotti. *Interpretação de Dados de Prospecção Geoquímica com o Auxílio de Estatística*. CPRM, Serviço Geológico do Brasil., 2010. 7, 90, 93, 94
- [7] D. C.Hoaglin B. Iglewicz. How to detect and handle outliers. *ASQC Quality Press*, 16, 1993.
- [8] M.A.P. Bedregal. Análise estatística e geoestatística de dados geoquímicos de superfície aplicada a exploração de hidrocarbonetos. Master's thesis, UFRJ, 2008.
- [9] V. Barnett; T. Lewis. *Outliers in Statiscal Data*. John Wiley & Sons, 1994.
- [10] J.S. Fonseca; G.A. Martins. *Curso de Estatística*. Atlas., 2006. 7
- [11] R. E. Shiffler. Maximum z scores and outliers. *The American Statistics.*, 42:79–80, 1988.

Índice Remissivo

Análise orientada a objeto, 28

AOO, 28

assuntos, 27

Casos de uso, 3

Diagrama de pacotes, 27

Diagrama de pacotes (assuntos), 27

Diagrama de sequência, 37

Elaboração, 7

Eventos, 37

Implementação, 54

módulos, 27

Mensagens, 37

Projeto do sistema, 52