SISTEMA DE ACOMPANHAMENTO DE DESENVOLVIMENTO ACADÊMICO

Luiz Augusto Volpi Nascimento, bolsista pela UEMS; Odival Faccenda, Orientador; Raquel Marcia Müller, Orientador; Jefersson Alex dos Santos, Autor.

Universidade estadual do Mato Grosso do Sul – UEMS.

015703@comp.uems.br; fac@uems.br; rmmuller@uems.br; jeferssonalex@gmail.com

RESUMO

Este trabalho consiste no desenvolvimento de um *software* estatístico capaz de armazenar resultados de avaliações obtidas por acadêmicos e emitir relatórios com relação às notas de uma turma em uma determinada disciplina, auxiliando professores e coordenadores no acompanhamento do desempenho de turmas e grupos de alunos. Ainda, pode auxiliar em uma auto-avaliação por parte dos professores, com relação a validade do instrumento utilizado na avaliação das turmas. O sistema foi desenvolvido na linguagem de programação Java™, da *Sun Microsystems*, com integração de banco de dados através da biblioteca SQLite. A análise estatística necessária foi realizada através do método de desenho de estudo transversal e do teste de Kolmogorov-Smirnov, que se mostrou eficiente tanto para amostras pequenas quanto para amostras grandes.

Palavras chaves: Kolmogorov-Smirnov, banco de dados, Java.

ABSTRACT

This work is the development of a statistical software capable of storing evaluation results obtained by students and issue reports with respect to notes from a class in a particular discipline, helping teachers and coordinators in monitoring the performance of teams and groups of students. Still, can assist in a self-evaluation by teachers, regarding the validity of the instrument used to evaluate the classes. The system was developed in Java ™ programming language, Sun Microsystems, integrated database through the SQLite library.

Statistical analysis was performed using the required method of cross-sectional study design and the Kolmogorov-Smirnov test, which proved effective for small samples and for large samples.

Key-words: Kolmogorov-Smirnov, DataBase, Java.

INTRODUÇÃO

A automação ou informatização de um determinado processo tem por objetivo facilitar ao máximo a sua execução, minimizando tempo e falhas (ou erros) que envolvem tal processo.

Apesar da crescente tecnologia que auxilia cada dia mais na execução das mais diversas tarefas, na área da educação os processos avaliativos ainda são realizados de forma tradicional, na forma de provas escritas, com questões pontuadas com valores de zero a dez. Quando um professor corrige as avaliações aplicadas a uma determinada turma, muitas vezes verifica que o resultado não é satisfatório, porém, nem sempre consegue identificar os motivos que originaram tal resultado.

Um *software* capaz de armazenar os resultados dos acadêmicos e emitir relatórios com relação às notas de uma determinada turma por disciplina, auxiliaria a professores e coordenadores a tomar decisões com relação ao desempenho de turmas ou de um grupo de alunos dessas turmas, bem como propiciaria uma auto avaliação em relação à validade do instrumento utilizado para medir o conhecimento da turma.

Um estudo com auxílio de técnicas estatísticas possibilitaria a criação de relatórios dessa natureza, ou seja, da mesma forma que estudos do IBGE (Instituto Brasileiro de Geografia e Estatística), por exemplo, nos possibilitam analisar o Brasil em seus mais variados aspectos, como taxa de analfabetismo e estimativas de crescimento populacional. Essa ciência em conjunto com um sistema computacional analisaria as notas de uma determinada turma e forneceria um relatório, indicando possíveis problemas durante o período em que as mesmas estão sendo ministradas e não após o fato consumado, possibilitando uma recuperação dos resultados e um melhor desempenho por parte dos alunos, resultando em um número menor de reprovações.

Esse projeto consiste da criação de um banco de dados que armazenará as notas de avaliações das disciplinas dos cursos de Ciência da Computação e Sistemas de Informação,

com o objetivo de analisar o desempenho dos acadêmicos. Tal análise envolverá métodos estatísticos que irão demonstrar se as avaliações aplicadas foram elaboradas adequadamente e se a turma em questão é homogênea ou heterogênea.

O sistema desenvolvido neste projeto será integrado com outro sistema, desenvolvido pelo projeto "Sistematização da avaliação interna dos cursos de Ciência da Computação e Sistemas de Informação", que conterá informações sobre avaliações feitas pelos alunos das disciplinas dos cursos. Dessa forma, será possível comparar o desempenho dos acadêmicos sob o ponto de vista do professor e dos alunos.

OBJETIVOS

O objetivo geral do projeto é o desenvolvimento de um software que organize, analise e interprete estatisticamente os resultados acadêmicos dos cursos de Ciência da Computação e Sistemas de Informação.

Os objetivos específicos do projeto são:

- Estudar o banco de dados já desenvolvido para os alunos oriundos de cotas, verificando a viabilidade de sua aplicação e a validade dos códigos já desenvolvidos para o âmbito dos cursos;
- Desenvolver um módulo para lançamento das notas pelos professores dos cursos;
- Calcular a distribuição de freqüência dos resultados de provas das disciplinas dos cursos e apresentar as medidas estatísticas com interpretação dos mesmos;
- Verificar o grau de heterogeneidade entre os alunos nas diferentes disciplinas em cada curso;
- Criar um módulo de geração de relatórios automatizado, para subsidiar as coordenações dos referidos cursos no acompanhamento didático e pedagógico, bem como oferecer ao professor subsídio para testar se o instrumento utilizado para avaliar a turma está aferido;
- Integrar o software com o sistema que contém a avaliação da disciplina por parte dos alunos;
- Aplicar um caso de teste nos cursos para avaliação dos resultados esperados.

MÉTODOS

O desenvolvimento do projeto será baseado no banco de dados desenvolvido junto com o Software Estatístico de Análise de Desempenho Acadêmico (Santos, 2005), que é capaz de ler informações de um banco de dados e resumi-las em tabelas e gráficos.

O software a ser desenvolvido tem por finalidade a geração de relatórios interpretativos, com informações extraídas de um banco de dados, baseando-se no que se chama de Controle Estatístico de Processos Avaliativos. Ter dados armazenados em um banco é uma forma de garantir a integridade e a organização dos mesmos e, mais que isso, disponibilizar esses dados na forma de relatórios interpretativos é uma forma de utilizar esses dados da maneira mais eficiente possível. Um teste de hipótese consiste em saber se os resultados de uma amostra contrariam ou não uma determinada afirmação feita sobre a população (Bussab, 1986). O propósito de um software com Controle Estatístico de Processos Avaliativos é testar as hipóteses de que os dados se adéquam a alguns padrões podendo, através disso, tirar conclusões.

As etapas de desenvolvimento envolvem os seguintes passos:

- Projeto do banco de dados. Nessa etapa será feito um estudo do banco de dados existente, realizando-se adequações para as necessidades atuais. O SGBD (Sistema Gerenciador de Banco de Dados) utilizado nas bases de dados será o SQLite. Serão utilizados os recursos computacionais dos laboratórios do curso de Ciência da Computação.
- A interface do sistema (API - *Application Programming Interface*, interface de programação de aplicativos) será desenvolvida na linguagem de programação Java™, da Sun Microsystems. A linguagem é orientada a objetos e fornece vários módulos para trabalho com interface gráfica, redes e banco de dados (JDBC), além de ser gratuita.
- A análise estatística será feita através do método de desenho de estudo transversal. Será utilizada uma amostra por conveniência. Todos os alunos identificados nas séries dos cursos serão avaliados para inclusão na amostra. Para o tratamento das variáveis contínuas, será calculada a diferença de médias ponderadas (modelo de efeito randômico), com intervalo de confiança de 95% correspondente às notas nas diferentes disciplinas das séries e cursos. Os dados serão submetidos à Análise de Variância com nível de significância α = 5%

para verificar se existem diferenças significativas no desempenho acadêmico entre os alunos investigados. Nos ensaios que apresentarem resultados significativos, será utilizado o teste de Tukey, para verificar quais os segmentos estudados apresentam diferenças significativas comparados entre si. Quando necessário, os dados originais serão transformados para bases logarítmicas para sua melhor distribuição ou em escalas que apresentam propriedades similares. A existência de heterogeneidade estatística nos estudos foi planejada para ser avaliada por um teste de heterogeneidade, Qui-Quadrado ou Kolmogorov.

FUNDAMENTAÇÃO TEÓRICA

DESENHO DE ESTUDO

Um desenho de estudo é um plano de trabalho de investigação com o propósito de solucionar uma questão ciêntifica.

Este conceito envolve a identificação do tipo de abordagem de uma metodologia que é utilizada para responder uma questão específica, implicando assim, a identificação de algumas características do estudo (a amostra e a população estudada, a unidade de análise, etc). Depois de definidas as características básicas do estudo, criam-se uma série de padrões terminológicos que definem estas caracteŕisticas e que constituem tipos ou desenhos de estudo.

ESTUDOS TRANSVERSAIS

Assim como o estudo de corte, é utilizado para designar um conjunto de indivíduos com caracteristicas em comum para serem observados, no estando, todas as suas medições são feitas em um único momento, diferente do estudo de corte onde os indivíduos são observados por um periodo de tempo.

A sua maior vantagem sobre os estudos de corte tem relação com a prontidão com que se pode tirar conclusões e a não existência de um período de sequimento. Desta forma, os estudos transversais são mas rápidos, baratos, fáceis e não são sensíveis a problemas a problemas como perda de seguimento entre outros, caracteristicos de estudos longitudinais.

TESTE DE KOLMOGOROV – SMIRNOV

O teste de Kolmogorov-Smirnov, assim como qui-quadradro, é ultilizado para determinar se uma determinada distribuição difere-se significativamente de uma distribuição teórica(D'HAINAUT,1997). Portanto, na maioria das vezes , encontra-se três tipos de situação:

- Pergunta-se se a distribuição observada é significativamente diferente de uma distribuição uniforme.
- Faz uma comparação da distribuição observada em uma amostra com uma distribuição determinada (a distribuição normal por exemplo).
- Compara-se a distribuição observada com uma distribuição conhecida de uma população finita supondo ser de onde extraída a amostra.

O teste de Kolmogorov – Smirnov é bem aplicado a amostras pequenas e, em muitos casos, melhor que o teste Qui-Quadrado que tem a mesma aplicação mas não esta particularmente submetido as mesmas restrições deste teste (grandes amostras e freqüências teóricas maiores que 5).

PRINCÍPIO DO TESTE

O teste de Kolmogorov–Smirnov é baseado na comparação das proporções acumuladas das distribuições observada e teórica, segundo mostra a Equação 1. Se a distribuição observada for semelhante a distribuição teórica, não deverá haver diferenças entre as duas a não ser diferenças devidas ás flutuações da amostragem. A distribuição da amostragem desta diferença é a probabilidade da veracidade da hipótese nula que pode estar ligada à maior das diferenças encontradas.

Na Equação 1, designa-se como D o valor absoluto da maior diferença entre a proporção acumulada teórica e a acumulada observada para cada uma das classes da distribuição.

$$D = | P_{acum} - P'_{acum} |Max, \qquad (1)$$

sendo “P acum” a proporção acumulada observada e “P' acum” a proporção teórica.

Para aplicar o teste deve-se construir uma coluna da distribuição observada, começando pelas categorias mais baixas e anotar as proporções acumuladas de cada categoria. Construir também uma coluna com as mesmas categorias, anotando as proporções acumuladas teóricas de cada categoria. Efetuar a diferença absoluta entre as proporções acumuladas observadas e teórica de cada categoria e examinar se as diferenças encontradas e designar por D a maior destas diferenças.

RESULTADOS

A CLASSE KolmogorovSmirnovTesting.java

A classe *KolmogorovSmirnovTesting.java* realiza o teste de Kolmogorov–Smirnov e retorna o valor absoluto calculado "D" e o valor absoluto critico corrigido[1] "Dc" para se aceitar ou rejeitar a hipótese nula. Kolmogorov e Smirnov mostraram que para amostras com pelo menos 35 dados a hipótese nula deve ser rejeitada se D maior ou igual a Dc sendo:

- Dc a divisão de 1,22 pela raiz de N ao nível de 0,1;
- Dc a divisão de -1,36 pela raiz de N ao nível de 0,05;
- Dc a divisão de 1,63 pela raiz de N ao nível de 0,01;

sendo N o tamanho da amostra.

Já para calcular-se a proporção acumulada teórica é preciso calcular cada valor da amostra da função de densidade de probabilidade, sendo essa a integração da função de densidade da distribuição normal:

A Equação 2 mostra a função de densidade da distribuição normal:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)} \qquad (2)$$

onde μ é a média e σ é o desvio padrão de X.

1 Em algumas situações, como em amostras pequenas, o teste de Shapiro-Wilk apresentava resultados mais eficientes. Para corrigir essa deficiência, adequou-se a constante do teste de Kolmogorov-Smirnov para 0.8886 dividido por raiz de (N + 1.5), de forma que, nestes casos, o teste de Kolmogorov-Smirnov atingisse a mesma eficiência que o teste de Shapiro-Wilk (para grandes amostras este teste já era mais eficiente).

Sua integração seria dada pela Equação 3:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-(u-\mu)}{2\sigma^2}} du \qquad (3)$$

A distribuição normal padrão assumi μ sendo uma constante no valor 0 e σ assumindo o valor 1. Logo esta integração ficaria na forma da Equação 4:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-u^2}{2}} du \qquad (4)$$

No programa, este cálculo da função de densidade de probabilidade é feito em cima da função densidade da distribuição normal padrão e sua integração é feita através da integração numérica de 1/3 de Simpsons implementada em uma função na classe *Integration.java* e chamada através da função *getProbability(double a, double b)* a qual pertence a classe abstrata *TheoricalDistribution.java* e é herdada pela classe *NormalDistribution.java*. A função *getProbability()* é mostrada na seção Resultados.

A função *getProbability(a,b)* se resume a:

```
public double getProbability(double a, double b) {
      try {
        return Integration.simpsonsOneThird(a, b, genFunct);
      } catch (UndefinedImageException uie) {
      return 0;
      }
   }
```

Sendo 'a' o valor inicial da integração, 'b' o valor final e genFunct a função a ser integrada.

A CLASSE ReportFactory.java

A função da *ReportFactory.java* é gerar um pequeno relatório a ser impresso na janela interna do programa, com a conclusão calculada pelo programa com relação à eficiência da prova, a heterogeneidade da turma, a simetria das notas e a normalidade.

Com relação a eficiência da prova, é realizado o teste de Kolmogorov-Smirnov e dependendo dos seguintes casos é concluído que:

- Caso o desvio padrão seja igual a 0, conclui-se que a prova foi desestimulante e totalmente mal elaborada.
- Caso o valor absoluto calculado menos o valor absoluto crítico seja maior que 0,14, conclui-se que a prova discriminou mal os alunos.
- Caso o valor absoluto calculado menos o valor absoluto critico seja maior que ou igual a 0, conclui-se que a prova não é normal, mas discriminou razoavelmente os alunos.
- Caso contrário, a prova foi normal e permitiu detectar bem os alunos.

Já para tirar uma conclusão da heterogeneidade da turma é utilizado o resultado do calculo do teste de Fisher para calcular a curtose, e caso:

- O desvio padrão for zero, conclui-se que a turma é totalmente homogênea.
- A curtose seja leptocúrtica, toma-se como conclusão que a turma é menos heterogênea que o normal.
- A curtose seja platicúrtica, conclui-se que em relação a heterogeneidade, a turma pode ser considerada normal.
- Caso contrario a turma é considerada mais heterogênea que o normal.

Para tirar a conclusão sobre a normalidade do teste o programa realiza o teste de Kolmogorov-Smirnov com os valores da amostra dependendo do resultado, o programa conclui que:

- A distribuição é constante, caso o desvio padrão seja igual a 0.
- A distribuição não é normal, caso o o valor absoluto critico encontrado no teste seja menor que o valor absoluto calculado.
- A distribuição é normal, caso o valor absoluto critico seja maior ou igual ao valor absoluto calculado.

Já para ter uma conclusão sobre a simetria do teste, é utilizado o Teste de Fisher que tem como filosofia testar a diferença de dois grupos (G1 e G2) em relação a uma variável qualquer que só admita dois valores como resposta: sim ou não, positivo ou negativo, etc. Neste caso, utilizamos apenas o valor do grupo 1(G1) para tirar uma das seguintes conclusões:

- A distribuição das notas é constante. Prova foi totalmente incapaz de discriminar os alunos, caso o desvio padrão for igual a zero.
- A distribuição das notas apresenta uma forte assimétrica positiva. Prova com viés no sentido de apresentar mais itens difíceis que fáceis, caso o índice do grupo 1 seja maior que 1.
- A distribuição das notas apresenta uma assimétrica positiva, moderada. Isto é, a prova se caracterizou por apresentar mais itens difíceis que fáceis, caso o indice do grupo 1 seja maior que 0,45.
- A distribuição das notas é simétrica, revelando um equilíbrio na escolha dos itens da prova, caso o índice do grupo 1 seja maior que -0,45.
- A distribuição das notas apresenta uma assimétrica negativa, moderada. Isto é, a prova apresentou mais itens fáceis que difíceis, caso o índice do grupo 1 seja maior que -1.
- Caso contrario, conclui-se que a distribuição das notas apresenta uma forte assimétrica negativa. Prova com viés no sentido de apresentar mais itens fáceis que difíceis.

A CLASSE SaveReportAction.java

A classe *SaveReportAction.java* tem a função de realizar a ação de salvar a analise em um arquivo.

Esta classe implementa a interface *actionlinester*, que é a interface associada ao ato de ação (no caso o clique do mouse) que é invocada no método *createmenubar()* da classe *SEADFrame*. Quando a ação de salvar é solicitada através da barra de menu do aplicativo a classe *SaveReportAction* chama o método *saveFile()* que abre uma janela para que seja escolhido o diretório onde o arquivo será salvo. Nesta janela é escolhido o nome e o diretório do arquivo que é enviado e após isto o método chama a função *save()* que está localizada a classe *SEADintFrame* ,que é a classe que cria a janela interna do aplicativo onde fica a analise. O método *save()* por sua vez salva no arquivo um objeto do tipo AbsFreqDistribution que é a classe que representa uma distribuição estatística de freqüências absolutas.

Conexão com o Banco de Dados

Nesta etapa do projeto foi criado um banco de dados onde pode-se cadastrar cursos, matérias e provas às quais o sistema pode analisá-las e da mesma for com a qual é feita com as notas das provas armazenadas em arquivos .sample.

Este banco de dados utiliza três tabelas para armazenar os dados sobre o curso, matéria e prova. A tabela "curso" é representada pela tupla curso (nome, área) onde os atributos representam as colunas da tabela que representam os nomes dos cursos cadastrados e as areas de cada curso. Outra tabela existente é a que armazena os dados das matérias de cada curso e é representada pela tupla matéria (nome,curso,professor,carga horária) onde os atributos representam as colunas que armazenam o nome da matéria, o curso a qual ela pertence, o professor que rege a matéria e a sua carga horária respectivamente. A tabela que armazena os dados das provas e representada pela tupla prova (curso, matéria, idProva, data, nota) onde os atributos da tupla representam respectivamente o curso que a matéria pertence, a matéria sobre a qual foi realizada a prova, a identificação da prova, o ano em que ela foi dada e a nota tirada pelo aluno. Com relação a tabela prova, foi adicionada a coluna curso para que não haja erros na busca de notas de uma matéria que é lecionada em cursos diferentes.

SQLite

A biblioteca SQLite foi escolhida para ser o motor do banco de dados por ser uma biblioteca compacta, com todas as funcionalidades ativas , com seu tamanho podendo ser inferior a 300 KB.

SQLite é uma biblioteca C que implementa um banco de dados SQL embutido. Programas que usam a biblioteca SQLite pode ter acesso a bancos de dados SQL sem executar um processo RDBMS separado.

SQLite não é uma biblioteca de cliente usada para conectar com um grande servidor de banco de dados. A biblioteca SQLite lê e escreve diretamente no arquivo do banco de dados em disco.

A CLASSE DBConnection

A classe DBConnection foi criada para realizar a conexão entre o banco de dados e o sistema podendo assim inserir cursos, matérias e notas de provas no banco de dados além de poder realizar análises das notas das provas cadastradas neste mesmo Banco de dados.

Para isto esta classe possui um construtor que verifica se o banco de dados existe e as funções *addCourse(), getCourse(), addDiscipline(), getDiscipline(), addAssessment(), getAssessment(), getData() e getId()* implementadas.

A função *addCourse* tem como parâmetros duas cadeias de caracteres, uma com o nome do curso e a segunda com o nome da área onde o curso atua, verifica se o curso existe no banco e caso não exista insere na tabela do banco que representa os cursos. A função *getCourse* busca no banco de dados os cursos já cadastrados e retorna um vetor de strings com os nomes dos cursos. Já a função *addDiscipline* recebe como parâmetro o curso a qual ela pertence, o nome da matéria, o nome do professor regente e a carga horária da mesma, a função busca no banco se a matéria já existe e ,caso não exista, insere ela no banco. A função *getDiscipline* busca no banco as matérias cadastradas e armazena em um vetor de strings. A função *addAssessment* recebe como parâmetro o nome do curso, da matéria , um identificador da prova, o ano que a prova foi dada e um vetor com as notas da prova e depois e verificar se a prova existe ou não no banco, cadastra as notas caso não exista. A função *getAssessment* recebe como parâmetro o nome do curso, da matéria que a prova foi dada, o identificador da prova e ano, busca através destes parâmetros no banco e retorna um vetor com as notas em ponto flutuante. A função *getData* recebe como parâmetro o nome da matéria, busca todas as datas das provas desta matéria e as armazenas em um vetor. A função *getId* recebe como parâmetro o nome do curso, da matéria e o ano em que a prova foi dada e retorna os identificadores de todas as provas desta matéria que foram aplicadas no ano escolhido.

INTERAGINDO COM O USUÁRIO

Para que se possa inserir os dados no banco de dados e fazer análises das provas cadastradas no mesmo é preciso que o usuário tenha como entrar com os dados ou então selecionar a prova desejada para analise, para isso foram criadas classes para que haja esta interação com o usuário. As classes são *insertCourseFrame, insertDisciplineFrame, insertAssessmentFrame e newDBAnalisysFrame*.

Um objeto da classe *insertCourseFrame* representa uma janela interna do programa contendo dois campos para a inserção de textos onde em um deve-se informar com o nome do curso e no outro a área de atuação do curso. Este objeto e criado pelo método c*reateCourseFrame* da classe *SEADFrame* que adiciona esta janela a área de trabalho do programa.

Uma janela usada para a inserção de uma matéria no banco e representada pelo objeto da classe *insertDisciplineFrame* que, para evitar que o usuário tente cadastrar uma matéria em um curso inexistente, utiliza um campo de múltipla escolha onde o usuário deve escolher a qual curso a matéria a ser cadastrada pertence. O objeto desta classe é criado pela função *createDisciplineFrame* da classe *SEADFrame* e é inserido na área de trabalho principal do programa.

Na tela criada pela classe *insertAssessmentFrame,* o usuário encontra um campo de múltipla escolha para não cadastrar a prova em um curso inexistente e, também, outro campo de múltipla escolha para escolher uma das matérias do curso selecionado. Além disto, o usuário encontra outros três campos: um para o identificador, um para a data da prova e outro para que se entre com as notas, separadas por vírgula. No caso das notas, quando o usuário confirmar o cadastro o sistema vai reconhecer as notas como uma cadeia de caracteres, dividí-la e armazenar em um vetor convertendo cada nota para um valor do tipo ponto flutuante. O objeto desta classe é criado pelo método *createAssessmentFrame* da classe *SEADFrame* que adiciona a janela a are a de trabalho principal do programa.

Para que sejam inseridas as informações no banco de dados, os objetos destas três classes se comunicam através de mensagens com um objeto da classe *DBConnection* para que ele execute as funções necessárias.

DISCUSSÃO

O teste de Kolmogorov-Smirnov que se mostrou extremamente eficiente tanto para amostras pequenas quanto para amostras grandes (maiores que 50) e agora é utilizado em todas as amostras. Além disto, o relatório impresso agora também tem uma análise mais bem colocada para o entendimento do usuário que utilize o aplicativo.

Agora também é possível cadastrar as notas das provas em um banco de dados e também que sejam feitas análises de notas cadastradas em um banco de dados que atua em conjunto ao sistema.

Apesar da modificação do teste Qui-Quadrado para o de Kolmogorov, a análise não teve um aumento na taxa de erro e continuo tão eficiente quanto quando não se utilizava o teste de qui-quadrado para pequenas amostras.

Com a integração do banco de dados o sistema se tornou mais robusto não precisando mais que as notas sejam salvas em um arquivo especifico do programa (.sample).

Apesar de utilizável o banco de dados pode ser aprimorado para que apenas um banco seja utilizado em toda uma instituição, independente do numero de cursos, com maior eficiência ou então o programa pode ser adaptado a um banco já existente na instituição.

AGRADECIMENTOS

Agradeço à Universidade Estadual do Mato Grosso do Sul pela ajuda pelo apoio financeiro. Ao acadêmico Jefersson Santos, que nunca negou ajuda e me auxiliou muito no entendimento dos códigos-fonte do sistema. Ao professor Nielsen Cassiano Simões, que sempre se mostrou disposto a me ajudar mesmo tendo que gastar algumas horas a mais no laboratório. Ao professor Nilton Cezar de Paula, que me auxiliou e muito a entender como funciona um banco de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

D'HAINAUT, LOUIS. **Conceitos e métodos da Estatística – Volume I**. Fundação Calouste Gulbenkian,1997.

Sites de Internet

Site oficial do SQLite. **Informações sobre o SQLite.** Disponível em: http://sqlite.org/about.html (último acesso: 18/08/2009)

Site do Serviço de Bioestatística e Informática Médica Faculdade de Medicina da Universidade do Porto. **Desenhos de Estudo**. Disponível em: http://stat2.med.up.pt/cursop/main.php3?capitulo=desenhos_estudo&numero=1&titulo=Desenhos+de+estudo (último acesso 06/05/2010)