FERNANDO AZENHA BAUTZER SANTOS

**DESENVOLVIMENTO DE PROGRAMAS
COMPUTACIONAIS VISANDO A
ESTIMATIVA DE PARÂMETROS DE
INTERESSE GENÉTICO-POPULACIONAL E
O TESTE DE HIPÓTESES GENÉTICAS**

**Dissertação apresentada ao Instituto de
Biociências da Universidade de São Paulo
para obtenção de Título de Mestre em
Ciências na área de Biologia / Genética.**

**São Paulo
2006**

ÍNDICE

	PÁGINA
INTRODUÇÃO	7
OBJETIVOS	27
RESULTADOS E CONCLUSÕES	29
TESTES EXATOS EM GENÉTICA DE POPULAÇÕES	29
TESTES EXATOS BASEADOS EM SIMULAÇÕES	37
DETERMINAÇÃO GRÁFICA DA REGIÃO DE CREDIBILIDADE A 95% USANDO O MÉTODO DE SUBAMOSTRAGENS (JACKKNIFE)	45
CONSTRUÇÃO DE INTERVALOS DE CONFIABILIDADE	47
COMPARAÇÃO DE INTERVALOS DE CONFIABILIDADE USANDO TÉCNICAS DE REAMOSTRAGEM "BOOTSTRAP" E "JACKKNIFE"	49

COMPARAÇÃO DOS RESULTADOS OBSERVADOS PELO TESTE EXATO PROPRIAMENTE DITO (SEM SIMULAÇÕES) COM OS OBTIDOS A PARTIR DE SIMULAÇÕES SEM REPOSIÇÃO	52
TESTES EXATOS POR SIMULAÇÃO SEM REPOSIÇÃO PARA O CASO GERAL DE n ALELOS	56
TESTES EXATOS PARA A ESTATÍSTICA F USADA NO ESTUDO DA ESTRUTURAÇÃO HIERÁRQUICA DAS POPULAÇÕES	58
HARDY-WEINBERG EQUILIBRIUM TESTING	63
SUMÁRIO	75
ABSTRACT	78
ANEXO	81

Orientador :

Prof. Dr. Paulo Alberto Otto

Ficha Catalográfica

Santos, FAB.

Desenvolvimento de programas computacionais visando a estimativa de parâmetros de interesse genético-populacional e o teste de hipóteses genéticas.

Dissertação de Mestrado - Instituto de Biociências da Universidade de São Paulo. Departamento de Biologia.

“

Comissão Julgadora:

“

“

“

“

“

Prof(a) .Dr(a)

“

“

“

Prof(a) .Dr(a)

“

Prof. Dr. Paulo A. Otto
Orientador

Agradecimentos

Agradeço ao Dr. Paulo Otto pela orientação sempre atenta e minuciosa.

Agradeço à Instituição financiadora FAPESP pelos recursos financeiros e reserva técnica.

Agradeço ao Dr. Sérgio Mاتيoli pela oportunidade acadêmica oferecida enquanto monitor da disciplina de sua responsabilidade (Processos Evolutivos).

Agradeço a meus familiares pela compreensão e pela ajuda oferecida durante os momentos mais difíceis da dissertação.

INTRODUÇÃO

A genética de populações conta com um arsenal impressionante de métodos estatísticos de estimação de parâmetros genético-populacionais de interesse, como é o caso de frequências alélicas, haplotípicas e genotípicas; de índices de estruturação e subdivisão populacionais (estatística F); de medidas de associação entre locos ligados ou não (coeficiente de desequilíbrio de ligação); de medidas de variabilidade intra e interpopulacional (taxas de heterozigose e índices de diversidade e identidade); de medidas de distância genética; de probabilidades de exclusão e inclusão em problemas de aplicação forense (identidade, maternidade, paternidade, filiação e troca de crianças), entre outros.

De início, muitos desses métodos tinham interesse apenas teórico, dada a sua elegância intrínseca, pois aplicavam-se a situações bastante restritas; outros tiveram e ainda têm aplicação um pouco menos restrita em setores especializados. De uma maneira geral, no entanto, esses métodos todos foram revitalizados pela recente explosão de conhecimentos na área de biologia molecular, os quais vêm permitindo o processamento de uma quantidade crescente e cada vez mais maciça de dados. Os métodos da genética de populações transformaram-se de repente na ferramenta de trabalho de um número enorme de pesquisadores, ao contrário do que ocorria antigamente, onde eram usados apenas por um punhado de iniciados. Ao lado da revitalização acima referida, ocorreu a necessidade do desenvolvimento de métodos adaptados à análise de grandes massas de dados e do desenvolvimento de técnicas de análise baseadas em reamostragens, estas últimas surgindo da necessidade de se associar aos

parâmetros estimados medidas mais exatas de confiabilidade probabilística, independentes de distribuições inferidas para a variável em estudo (como é o caso dos testes baseados na distribuição dita normal).

A IMPORTÂNCIA DOS TESTES EXATOS DE HARDY-WEINBERG

Um dos pontos-chave da teoria da evolução de Charles Darwin era a necessidade de existência de variabilidade genética para que a seleção pudesse agir. Embora um alto grau de variabilidade genética fosse a característica da maioria das populações animais, os conhecimentos de genética vigentes na época de Darwin impediram-no de vislumbrar como essa variabilidade era mantida. A solução para essa questão veio em 1908, pouco após a redescoberta e a difusão das leis de herança particulada propostas por Mendel. Nesse ano, foram publicados os trabalhos de Hardy e Weinberg, responsáveis pela criação do modelo básico da genética de populações. Estes autores mostraram de maneira irrefutável que, se as frequências de dois alelos (**A** e **a**) segregando num loco autossômico qualquer são **p** e **q** respectivamente, numa população com sistema de cruzamentos ao acaso (que equivale à combinação, também ao acaso, dos gametas produzidos pelos indivíduos da população), as probabilidades de ocorrência dos genótipos **AA**, **Aa** e **aa** são respectivamente **p^2** , **$2pq$** e **q^2** . Numa população ideal, de tamanho praticamente infinito (não sujeito, portanto, a flutuações numéricas amostrais), sem influências de eventos como mutação, seleção e migração (capazes de alterar as frequências gênicas), os genótipos tendem a se manter indefinidamente nas proporções **p^2** , **$2pq$** e **q^2** . Generalizando, esse princípio pode ser representado pela equação (Otto, comunicação pessoal):

$P(a_i a_j) = P_{ij} = (2 - \delta_{ij}) \cdot P(a_i) \cdot P(a_j) = (2 - \delta_{ij}) \cdot p_i p_j$, $i \leq j$, em que δ_{ij} (delta de Kronecker) é um operador lógico que toma os valores 1 se $i = j$ e 0 se $i \neq j$. A expressão acima reduz-se a $P(a_i a_i) = p_i^2$ se $i = j$ e a $P(a_i a_j) = 2p_i p_j$ se $i \neq j$. Esta propriedade é atingida em uma única geração em sistemas com gerações discretas e assintoticamente em sistemas com gerações contínuas.

De uma maneira mais genérica, desconsiderando-se o sistema de cruzamentos vigente na população, as frequências genotípicas podem ser colocadas sob a forma genérica:

$P(a_i a_j) = p_i \delta_{ij} F + (2 - \delta_{ij}) \cdot p_i p_j \cdot (1 - F)$, $i \geq j$, em que F ($-1 \leq F \leq 1$) é o índice de fixação da população.

Quando $F = 0$, a equação acima reduz-se ao caso pan-mítico.

Com o desenvolvimento teórico da Genética de Populações, surgiu de maneira natural a necessidade de se verificar as condições de pan-mixia. Inicialmente, isso foi conseguido através do teste não-paramétrico do qui-quadrado:

	AA	Aa	aa	total
freq. abs. obs. (o_i)	$D = o_1$	$H = o_2$	$R = o_3$	N
freq. abs. esp. (e_i)	$Np^2 = e_1$	$2Npq = e_2$	$Nq^2 = e_3$	N
com $p = (2D + H) / (2N)$, $q = (2R + H) / (2N)$ e número de graus de liberdade (g.l.) = 1.				

Nesse caso, o qui-quadrado é dado pela seguinte fórmula:

$$\chi^2 = \sum (o_i - e_i)^2 / e_i = \sum (o_i^2 / e_i) - N = D^2 / Np^2 + H^2 / 2Npq + R^2 / Nq^2 - N.$$

Para amostras pequenas, especialmente aquelas em que pelo menos um dos $e_i < 5$ costuma-se aplicar a correção de Yates (1934), que

consiste em diminuir-se 0.5 a cada frequência $o_i > e_i$ e adicionar-se 0.5 a cada frequência $o_i < e_i$.

Notou-se logo uma correspondência entre a fórmula $\chi^2 = \sum (o_i - e_i)^2 / e_i$ e a obtida da tabela de contingência

D	H/2	Np
H/2	R	Nq
Np	Nq	N

uma vez que $e_1 = Np \times Np / N = Np^2$, $e'_2 = Np \times Nq / N = Npq$, $e''_2 = e'_2$, $e_2 = e''_2 + e'_2 = 2Npq$, $e_3 = Nq \times Nq / N = Nq^2$.

É fácil demonstrar que o valor do qui-quadrado obtido desta tabela equivale exatamente à aplicação da fórmula $\chi^2 = \sum (o_i - e_i)^2 / e_i$, a qual pode ser rearranjada algebricamente como:

$$\begin{aligned}\chi^2 &= (H^2/4 - DR)^2 N / [(D + H/2)^2 (H/2 + R)^2] = \\ &= (H^2 - 4DR)^2 N / [(2D + H)^2 (H + 2R)^2].\end{aligned}$$

Aplicando-se a correção de continuidade de Yates(1934) a esta fórmula, obtemos:

$$\begin{aligned}\chi^2 &= [|H^2/4 - DR|^2 - N/2]^2 N / [(D + H/2)^2 (H/2 + R)^2] = \\ &= [|H^2 - 4DR|^2 - 2N]^2 N / [(2D + H)^2 (H + 2R)^2].\end{aligned}$$

Como os dados podem ser rearranjados conforme a tabela acima, podemos empregar outros testes aplicáveis a tabelas de contingência 2x2, como o teste G ou razão de verossimilhança (com e sem correção) e o teste exato de Fisher.

No caso do teste G, basta aplicarmos a fórmula

$$\begin{aligned}\chi^2 &\cong 2[D \log(D) + H/2 \log(H/2) + H/2 \log(H/2) + R \log(R) - (D+H/2) \\ &\log(D+H/2) - (R+H/2) \log(R+H/2) + N \log(N)] \cong \\ &\cong 2\{D \log(D) + H \log(H) + R \log(R) - H \log(2) - 2N [p \log(p) + \\ &q \log(q)] - N \log(N) \}.\end{aligned}$$

O teste G com correção de continuidade é obtido da fórmula anterior verificando-se primeiro se DR é maior ou menor que $H^2/4$. Se $DR < H^2/4$, os valores D , $H/2$ e R são substituídos por $D+0.5$, $H/2-0.5$ e $R+0.5$; caso contrário ($DR \geq H^2/4$) os valores D , $H/2$ e R são substituídos por $D-0.5$, $H/2 + 0.5$ e $R - 0.5$. Como se trata de uma tabela de contingência, os valores marginais $D + H/2$ e $R + H/2$ não se alteram.

Finalmente, ainda considerando-se a tabela de contingência

a = D	b = H/2	a + b
c = H/2	d = R	c + d
a + c	b + d	N

com marginais fixados [**(a + c)**, **(b + d)**, **(a + b)**, **(c + d)** e **N**], o equilíbrio de Hardy-Weinberg pode ser verificado através do teste exato de Fisher. Segundo esse teste, que se baseia na distribuição hipergeométrica, a probabilidade de ocorrência da tabela observada, na hipótese nula de não-associação, é

$$P(a,b,c,d) = [(a+b)! (c+d)! (a+c)! (b+d)!] / (a!b!c!d!N!).$$

O método computa as probabilidades correspondentes a todas as tabelas possíveis mantendo-se constantes os valores marginais $(a + b)$, $(c + d)$, $(a + c)$, $(b + d)$ e N . A probabilidade bicaudal do teste é obtida somando-se todos os valores de probabilidade iguais ou inferiores ao valor da probabilidade correspondente à da tabela observada.

Para o caso em que H (número observado de heterozigotos) é ímpar, os valores das caselas na diagonal secundária $(b = c = H/2)$ são substituídos por $(H+1)/2$ e $(H-1)/2$.

Foram desenvolvidos concomitantemente testes apropriados para amostras populacionais com tamanhos reduzidos. Os mais usados desses testes são os propostos por Hogben (1946) e Levene (1949), Haldane (1954) e Cannings e Edwards (1969). O teste proposto por Haldane corresponde ao teste exato que enumera todas as amostras possíveis com mesmas frequências alélicas, o qual discutiremos mais abaixo.

O raciocínio usado por Hogben, Levene e Cannings & Edwards é simples: se a amostra tem tamanho reduzido, ao tomar-se um gene qualquer da população (a_i) , diminui automaticamente a probabilidade do segundo gene sorteado ser do mesmo tipo, ou seja, a probabilidade de formação de um indivíduo homozigoto é menor que p_i^2 , o que pode ser expresso pela inequação $P(a_i a_i) < P(a_i) \times P(a_i)$. A diferença conceitual entre os testes de Hogben (1946)/ Levene (1949) e Cannings e Edwards (1969) é que o primeiro considera a formação de um genótipo $a_i a_j$ a partir de um único pool gênico, que contém os alelos a_i e a_j , enquanto o segundo considera a formação dos indivíduos a partir da combinação de gametas de dois conjuntos distintos de gametas produzidos por machos e fêmeas.

Na formulação de Hogben / Levene, os números esperados de indivíduos AA , Aa e aa são calculados segundo $(2D + H)$ $(2D + H - 1)$ /

$[2(2N - 1)]$, $(2D + H) (H + 2R) / (2N - 1)$ e $(H + 2R) (H + 2R - 1) / [2(2N - 1)]$ respectivamente. A fórmula correspondente ao teste qui-quadrado simplifica-se conforme

$$\chi^2 = 2D^2(2N-1) / [(2D + H) (2D+H-1)] + 2H^2(2N-1) / [(2D + H) (H + 2R)] + 2R^2(2N - 1) / [(2R + H) (H + 2R - 1)] - N .$$

No método proposto por Cannings e Edwards, os números esperados de indivíduos AA, Aa e aa são respectivamente $[(2D + H)^2 - H] / 4N$, $[(2D + H) (H + 2R) + H] / 2N$ e $[(H + 2R)^2 - H] / 4N$. A fórmula do teste qui-quadrado pode então ser colocada sob a forma

$$\chi^2 = 4ND^2 / [(2D + H)^2 - H] + 2NH^2 / [(2D + H) (H + 2R) + H] + 4NR^2 / [(2R + H)^2 - H] - N .$$

Outros testes aplicáveis a amostras de tamanho pequeno são citados por Elston e Forthofer (1977) e Emigh (1980). Na verdade, o teste do qui-quadrado com valores esperados condicionais detalhado por Elston e Forthofer (1977), corresponde ao modelo proposto por Hogben / Levene; e o teste de razão de verossimilhanças, também analisado por Elston e Forthofer (1977) corresponde exatamente ao teste G já por nós detalhado. Emigh (1980) reviu a maioria dos testes examinados por Elston e Forthofer (1977) e apresentou uma análise de outros métodos, como

(a) o proposto por Elston e Forthofer (1977), que considera um qui-quadrado obtido da média dos valores de dois testes, com e sem correção de continuidade;

(b) o teste T^2 condicional de Freeman e Tukey (1950), no qual os dados de frequência sofrem a transformação raiz quadrada para estabilização da variância;

(c) teste Z^2 de Mantel-Li (1974), em que o valor do teste é obtido multiplicando-se o resultado obtido com a correção de Hogben / Levene por $(2N - 3) / [2(N - 1)]$.

Emigh (1980) ainda sugere, em alguns testes analisados, correções de continuidade usando as quantidades **0.5** e **0.25**, verificando empiricamente que a última fornece resultados mais consistentes.

A literatura sobre testes exatos de Hardy-Weinberg é relativamente extensa, sendo as primeiras publicações sobre o assunto (Louis e Dempster, 1987; Guo e Thompson, 1992) ainda considerados trabalhos básicos de referência. Esses e outros trabalhos foram revistos, analisados e criticados em várias publicações mais ou menos recentes, as quais, em seu conjunto, apresentam uma lista praticamente completa de todas as pesquisas realizadas sobre o tema (Rousset e Raymond, 1995; Montoya-Delgado e cols., 2001; Chen e cols., 1999; Buckleton e cols., 2001; Wigginton e cols., 2005). Ao contrário dos demais, que se atêm a aplicações práticas, o trabalho de Montoya-Delgado e cols., apesar de aparentemente ser o mais aprofundado e o mais bem baseado matematicamente, trata do assunto sob um enfoque bayesiano, só pode ser entendido por iniciados no tema, além do fato de a argumentação básica não ter sido aceita de maneira unânime na seara da estatística matemática.

Principalmente após a popularização da rede eletrônica, surgiu um número significativo de programas que realizam testes exatos para verificação das proporções de Hardy-Weinberg, a maioria dos quais são

encontrados na rede eletrônica mundial , podendo ser carregados, instalados e usados gratuitamente nos computadores dos usuários, ou então utilizados diretamente em formulários html da própria rede nas páginas (sites) dos autores. Alguns poucos programas, como por exemplo o HWSIM (Calfell, 2006) e o HYDIAG (Rogatko & Slifker, 2006) executam apenas variações do teste exato para verificação do equilíbrio de Hardy-Weinberg. O primeiro (ainda disponível em versão para ambiente DOS) é bastante simples e de uso intuitivo, possuindo portanto as limitações inerentes a programas com essas propriedades. O segundo, que seguramente é o mais avançado sobre o assunto, usa um enfoque bayesiano não tradicional, está bem baseado matematicamente (Rogatko e cols., 2002) mas infelizmente só pode ser entendido por iniciados em estatística matemática avançada e inferência bayesiana. Com exceção desses dois programas e de vários outros com características semelhantes ao HWSIM, a maioria dos aplicativos existentes que executam testes exatos para verificação do equilíbrio de Hardy-Weinberg são programas relativamente complicados capazes de realizarem estimativas e testes de hipóteses em amostras genotípicas contendo enorme massa de dados. Os aplicativos mais importantes e mais usados dessa categoria são o TFGPA (Miller, 2006), o ARLEQUIN (Excoffier et al, 2005), o GDA (Lewis & Zaykin, 2006), o GENEPOP (Raymond & Rousset, 2006) , o GENESTRUT (Constantine, 2006) e o POPGENE (Yeh & Boyle, 2006). Esses programas todos foram analisados em profundidade num estudo comparativo realizado por Labate (2000), cujo artigo fornece não só os detalhes sobre o funcionamento desses programas como também as referências originais de trabalhos publicados (ou disponíveis na rede eletrônica mundial) pelos próprios autores dos aplicativos. O único problema prático apresentado por esses aplicativos

é que a maioria deles exige uma interface complicada de entrada de dados, através de arquivos com uma estruturação particular, extensa e pouco flexível.

Com a finalidade de compararmos os principais métodos citados acima, desenvolvemos um programa computacional (as listagens desse programa-fonte 1 dos demais programas mencionados nessa dissertação estão reunidas em anexo ao final do trabalho), que enumera todas as populações possíveis com tamanho fixo **N**, considerando os genótipos determinados por um par de alelos autossômicos. Em seguida realiza, em cada uma das $(N + 1)(N + 2)/2 - 2$ populações possíveis com **p** e **q** $\neq 0$, os testes do qui-quadrado, qui-quadrado com correção de continuidade, teste G (razão de verossimilhanças), teste G com correção de continuidade, teste de Hogben / Levene, teste de Cannings e Edwards, teste exato de Fisher, além de 3 tipos de testes exatos (discutidos em detalhe mais adiante, uma vez que constituem o objetivo desta dissertação): teste exato obtido por enumeração de todas as amostras possíveis com mesma frequência gênica (o qual equivale ao teste exato de Haldane), teste exato também condicionado à frequência gênica observada na amostra, porém obtido através de simulação sem reposição e finalmente o teste exato não condicionado à frequência gênica amostral, obtido por simulação com reposição. Em seguida, usando um diagrama convencional de dispersão de pares de valores ordenados {x,y} em gráfico cartesiano, o

programa plota os valores correspondentes a quaisquer pares de testes. Os gráficos mostrados nas figuras foram obtidos a partir de testes realizados com todas as populações (genótipos) possíveis de tamanho 40 ou 50.

A análise desses resultados apresentados nas figuras **1** a **6** mostrou que:

(1) os testes G (razão de verossimilhanças) e χ^2 com correção de continuidade tendem a fornecer valores de probabilidades maiores que os obtidos com o teste exato, aumentando o erro tipo II do teste de hipóteses;

(2) os testes de χ^2 sem correção, de Hogben / Levene, de Cannings e Edwards e o teste de verossimilhanças (teste G) tendem a fornecer valores de probabilidades menores que as obtidas com o teste exato, aumentando o erro tipo I do teste de hipóteses;

Além disto, verificamos também que os testes de Hogben / Levene e Cannings e Edwards, da mesma maneira que os testes de qui-quadrado e de razão de verossimilhança (teste G), com e sem correção de continuidade, são equivalentes, como mostram as figuras **7** a **10**. Também é confrontado o comportamento do teste de Fisher em relação ao teste exato.

Também verificamos que os testes de qui-quadrado e G (razão de verossimilhança) corrigidos fornecem quase sempre valores de P maiores que os obtidos pelos testes sem correção, (figuras **11** e **12**).

As comparações que realizamos entre os três tipos exatos do teste serão comentados mais adiante.

Realizamos também uma análise da distribuição dos erros relativos (definidos como os valores de probabilidades gerados por um

teste qualquer em relação às probabilidades do teste exato), cujos resultados são mostrados nas figuras **13** a **16**.

Verifica-se que os maiores erros relativos, no caso dos testes sem correção de continuidade (qui-quadrado e teste G), distribuem-se preferencialmente em torno das regiões críticas ou de transição correspondentes aos níveis de **0.05** e **0.95** de probabilidade. No caso do teste do qui-quadrado com correção de continuidade, uma tendência semelhante é observada, sendo que os maiores erros relativos concentram-se nos limites do domínio de **P** (**0** e **1**), estando claramente aumentados em relação aos valores dos testes sem correção de continuidade. Já no caso do teste G (razão de verossimilhanças) corrigido, os erros relativos estão aumentados e distribuem-se uniformemente ao longo do intervalo **(0,1)**.

Figura 1- Gráfico de dispersão mostrando os valores correspondentes ao teste de qui-quadrado com correção de continuidade e teste exato para todas as populações possíveis com N = 50 indivíduos.

Figura 2- Gráfico de dispersão mostrando os valores correspondentes ao teste de Hogben / Levene com correção de continuidade e teste exato para todas as populações possíveis com N = 50 indivíduos.

Figura 3- Gráfico de dispersão mostrando os valores correspondentes ao teste G com correção de continuidade e teste exato para todas as populações possíveis com $N = 50$ indivíduos.

Figura 4- Gráfico de dispersão mostrando os valores correspondentes ao teste de qui-quadrado sem correção de continuidade e teste exato para todas as populações possíveis com $N = 50$ indivíduos.

Figura 5- Gráfico de dispersão mostrando os valores correspondentes ao teste de Hogben/ Levene sem correção de continuidade e teste exato para todas as populações possíveis com $N = 50$ indivíduos.

Figura 6- Gráfico de dispersão mostrando os valores correspondentes aos testes G sem correção de continuidade e teste exato para todas as populações possíveis com $N = 50$ indivíduos.

Figura 7- Gráfico de dispersão mostrando os valores correspondentes aos testes de Cannings e Edwards e de Hogben / Levene para todas as populações possíveis com $N = 40$ indivíduos.

Figura 8- Gráfico de dispersão mostrando os valores correspondentes aos testes G com correção de continuidade e do qui-quadrado com correção de continuidade para todas as populações possíveis com $N = 40$ indivíduos.

Figura 9- Gráfico de dispersão mostrando os valores correspondentes aos testes de qui-quadrado sem correção de continuidade e G sem correção de continuidade para todas as populações possíveis com $N = 50$ indivíduos.

Figura 10- Gráfico de dispersão mostrando os valores correspondentes aos testes de Fisher e exato para todas as populações possíveis com $N = 40$ indivíduos.

Figura 11- Gráfico de dispersão mostrando os valores correspondentes aos testes de qui-quadrado com e sem correção de continuidade para todas as populações possíveis com $N = 50$ indivíduos.

Figura 12- Gráfico de dispersão mostrando os valores correspondentes aos testes G com e sem correção de continuidade para todas as populações possíveis com $N = 50$ indivíduos.

Figura 13- Erros relativos, em relação ao teste exato, dos valores de probabilidade gerados pelo teste de qui-quadrado sem correção de continuidade aplicado ao conjunto de todas as populações possíveis com $N = 50$ indivíduos.

Figura 14- Erros relativos, em relação ao teste exato, dos valores de probabilidade gerados pelo teste de qui-quadrado com correção de continuidade aplicado ao conjunto de todas as populações possíveis com $N = 50$ indivíduos.

Figura 15- Erros relativos, em relação ao teste exato, dos valores de probabilidade gerados pelo teste G (ou teste de razão de verossimilhança) sem correção de continuidade aplicado ao conjunto de todas as populações possíveis com $N = 50$ indivíduos.

Figura 16- Erros relativos, em relação ao teste exato, dos valores de probabilidade gerados pelo teste G (ou teste de razão de verossimilhança) com correção de continuidade aplicado ao conjunto de todas as populações possíveis com $N = 50$ indivíduos.

OBJETIVOS

A partir do que já existe na literatura, já revisado na introdução, propusemos a elaboração de um pacote computacional, com ênfase no problema do equilíbrio de Hardy-Weinberg, reunindo as boas qualidades apresentadas pelos softwares pesquisados (velocidade, clareza, precisão e nível técnico) a fim de simplificar o trabalho do pesquisador em genética.

Para isso, desenvolvemos no presente trabalho programas que executam testes exatos aplicáveis a situações freqüentes em genética de populações e que incluem os seguintes tópicos:

1) estimativas não enviesadas de freqüências alélicas, genotípicas e fenotípicas, com determinação dos respectivos intervalos de credibilidade com uma probabilidade $1-2\alpha$ qualquer, para o caso particular de dois alelos e para o caso generalizado de n alelos segregando num loco autossômico qualquer;

2) testes de hipóteses genéticas: equilíbrio de Hardy-Weinberg e estruturação hierárquica de populações, também para o caso particular de dois alelos e para o caso generalizado de n alelos segregando num loco autossômico qualquer;

3) comparação dos resultados obtidos com testes exatos com aproximações existentes na literatura;

4) representação gráfica, no caso de dois alelos, dos resultados de testes de $F = 0$, usando sistemas ternários isósceles (Otto e Benedetti, 1994) de coordenadas gráficas.

Para o desenvolvimento dos programas utilizamos os softwares QuickBasic (para DOS) e Visual Basic (versão 6.0 para Windows), ambos desenvolvidos pela firma Microsoft.

Ao contrário de muitos programas comerciais ou gratuitos obtidos através da rede eletrônica, os quais se acompanham às vezes de manuais com instruções complicadas, os programas que desenvolvemos são simples e de uso intuitivo, não exigindo dos usuários conhecimentos específicos de computação nem o preparo de arquivos complexos de dados.

RESULTADOS E CONCLUSÕES

TESTES "EXATOS" EM GENÉTICA DE POPULAÇÕES

Os testes chamados exatos em senso estrito restringem-se de uma maneira geral a situações bastante particulares, uma vez que consistem na comparação da probabilidade de ocorrência da amostra observada com as probabilidades de todas as amostras possíveis dentro de determinadas condições. Em genética de populações o exemplo clássico é dado pelo teste de pan-mixia no caso de dois alelos sem dominância. Dado que foram observados n_{AA} indivíduos **AA**, n_{Aa} indivíduos **Aa** e n_{aa} indivíduos **aa** num total de $n = n_{AA} + n_{Aa} + n_{aa}$ indivíduos amostrados e que as frequências gênicas da amostra são $p = P(A) = n_A / (n_A + n_a) = (2n_{AA} + n_{Aa}) / 2n$ e $q = P(a) = 1 - p$, a probabilidade de ocorrência da amostra sob hipótese de panmixia é evidentemente $P_0 = n! / (n_{AA}! n_{Aa}! n_{aa}!) \cdot (p^2)^{n_{AA}} \cdot (2pq)^{n_{Aa}} \cdot (q^2)^{n_{aa}}$. São enumeradas em seguida todas as amostras possíveis de tamanho n com mesmas frequências alélicas e calculadas as probabilidades correspondentes de ocorrência delas sob hipótese de pan-mixia. Cada probabilidade dessas (P_i) é então comparada com P_0 ; se P_i é menor ou igual a P_0 , seu valor é somado em $P = \sum P_i$, cujo valor final é a probabilidade de ocorrência da amostra observada e de todas as amostras com probabilidade menor que essa: essa é a probabilidade dita exata favorecendo a hipótese de os genótipos da amostra encontrarem-se nas proporções p^2 , $2pq$, q^2 . O programa-fonte não-compilado desenvolvido em linguagem BASIC usando o software QBasic (Microsoft Co.) mostra as etapas necessárias para se gerar essa probabilidade exata, tomando como exemplo numérico os casos $\{D = n_{AA}, H = n_{Aa}, R = n_{aa}\} = \{9, 1, 30\}$ e $\{5, 10, 15\}$ (programa 2).

D, H, R = 9,1,30

D	H	R	Prob.Cum. Dsq.			Chi-squared				P(Chi - squared)			
			Prob.			1	2	3	4	1	2	3	4
9	1	30 *	0.00	0.00	0.17	34.67	29.72	36.34	34.78	0.00	0.00	0.00	0.00
8	3	29	0.00	0.00	0.14	25.15	20.96	26.49	25.41	0.00	0.00	0.00	0.00
7	5	28	0.00	0.00	0.12	17.15	13.73	18.19	17.50	0.00	0.00	0.00	0.00
6	7	27	0.00	0.00	0.09	10.68	8.02	11.44	11.04	0.00	0.00	0.00	0.00
5	9	26	0.02	0.02	0.07	5.74	3.84	6.25	6.05	0.02	0.05	0.01	0.01
0	19	21	0.06	0.08	-.06	3.88	2.35	3.64	3.58	0.05	0.13	0.06	0.06
4	11	25	0.10	0.18	0.04	2.32	1.18	2.62	2.54	0.13	0.28	0.11	0.11
1	17	22	0.23	0.41	-.03	1.20	0.44	1.05	1.03	0.27	0.51	0.30	0.31
3	13	24	0.25	0.66	0.02	0.42	0.05	0.54	0.53	0.52	0.83	0.46	0.47
2	15	23	0.34	1.00	-.01	0.05	0.05	0.02	0.02	0.82	0.83	0.88	0.89

D, H, R = 5,10,15

D	H	R	Prob.Cum. Dsq.			Chi-squared				P(Chi - squared)			
			Prob.			1	2	3	4	1	2	3	4
10	0	20	0.00	0.00	0.22	30.00	25.67	31.31	30.00	0.00	0.00	0.00	0.00
9	2	19	0.00	0.00	0.19	21.68	18.02	22.75	21.83	0.00	0.00	0.00	0.00
8	4	18	0.00	0.00	0.16	14.70	11.72	15.56	14.95	0.00	0.00	0.00	0.00
7	6	17	0.00	0.00	0.12	9.07	6.77	9.73	9.36	0.00	0.01	0.00	0.00
0	20	10	0.01	0.01	-.11	7.50	5.42	7.06	6.86	0.01	0.02	0.01	0.01
6	8	16	0.03	0.04	0.09	4.80	3.17	5.26	5.07	0.03	0.08	0.02	0.02
1	18	11	0.06	0.10	-.08	3.68	2.27	3.36	3.26	0.06	0.13	0.07	0.07
5	10	15 *	0.11	0.22	0.06	1.88	0.92	2.16	2.08	0.17	0.34	0.14	0.15
2	16	12	0.21	0.42	-.04	1.20	0.47	1.01	0.98	0.27	0.49	0.31	0.32
4	12	14	0.26	0.68	0.02	0.30	0.02	0.41	0.40	0.58	0.89	0.52	0.53
3	14	13	0.32	1.00	-.01	0.08	0.02	0.03	0.03	0.78	0.89	0.86	0.86

Inicialmente, o programa calcula e armazena na memória do computador os logaritmos de todos os fatoriais de zero a **2N**. Como descrito anteriormente, calcula em seguida as probabilidades correspondentes a todas as populações possíveis {**D, H, R**} com **P = 2D + H** ou **Q = 1 - P = H + 2R** mantidos constantes. Além de fornecer a probabilidade [Prob.] correspondente a cada população possível (função densidade) e a probabilidade acumulada [Cum.Prob.] (função distribuição), que corresponde à probabilidade exata gerada pelo programa, o programa imprime também o valor do coeficiente de desequilíbrio [Dsq.] de Hardy-Weinberg ($D/n - p^2$), e o valor dos testes de qui-quadrado [Chi-squared] usados para testar a hipótese de pan-mixia e as probabilidades

correspondentes [P(Chi-squared)] a esses valores. Os resultados dos quatro testes empregados são mostrados nas colunas numeradas de 1 a 4. A alternativa 1 do teste é a forma tradicional sem correções, $\chi^2 = \sum[(oi-ei)^2/ei]$, que para o caso especial em que é testada a hipótese de equilíbrio reduz-se a $\chi^2 = (H^2-4DR)^2 \cdot N / [(2D+H)^2 \cdot (H+2R)^2]$ com um grau de liberdade. A alternativa 2 é a forma com a correção de Yates para continuidade, especialmente aplicável no caso de amostras pequenas. As colunas 3 e 4 mostram os valores de qui-quadrado obtidos aplicando-se as correções propostas por Hogben/Levene e Cannings e Edwards.

Nota-se que os valores do qui-quadrado correspondem apenas grosseiramente às probabilidades geradas pelo teste exato executado; de uma maneira geral, estas últimas são sempre maiores que as probabilidades correspondentes aos valores do teste de qui-quadrado. Mediante a aplicação do programa a várias situações distintas, estudos comparativos empíricos permitem verificar se existem condições que indiquem o uso de uma ou mais dessas aproximações em substituição ao teste exato mostrado acima e a outros que discutiremos nas linhas abaixo.

Obviamente, o teste exato mostrado acima é limitado pelo tamanho amostral n , que, muito grande, torna impraticável a enumeração de todas as combinações genotípicas possíveis capazes de fornecerem a mesma frequência alélica; além disso, essa enumeração é factível de uma maneira geral apenas para o caso de poucos alelos, uma vez que um número moderadamente elevado de alelos já produz um número exponencial de combinações genotípicas com mesma frequência gênica, mesmo para valores modestos do tamanho amostral n , como mostramos analiticamente nas linhas que se seguem.

O **programa 3** (apropriado para o caso de quatro alelos) enumera as combinações genotípicas possíveis para amostras de tamanho **n** fixo e frequências gênicas constantes. O exemplo numérico aplicado refere-se a uma população de tamanho **n = 8** indivíduos com frequências gênicas **P(a) = 1/16, P(b) = 3/16, P(c) = 5/16 e P(d) = 7/16**. Nesse caso, existem **39** populações possíveis.

aa	ab	ac	ad	bb	bc	bd	cc	cd	dd	a	b	c	d	npop
0	0	0	1	0	0	3	1	3	0	1	3	5	7	1
0	0	0	1	0	0	3	2	1	1	1	3	5	7	2
0	0	0	1	0	1	2	0	4	0	1	3	5	7	3
0	0	0	1	0	1	2	1	2	1	1	3	5	7	4
0	0	0	1	0	1	2	2	0	2	1	3	5	7	5
0	0	0	1	0	2	1	0	3	1	1	3	5	7	6
0	0	0	1	0	2	1	1	1	2	1	3	5	7	7
0	0	0	1	0	3	0	0	2	2	1	3	5	7	8
0	0	0	1	0	3	0	1	0	3	1	3	5	7	9
0	0	0	1	1	0	1	0	5	0	1	3	5	7	10
0	0	0	1	1	0	1	1	3	1	1	3	5	7	11
0	0	0	1	1	0	1	2	1	2	1	3	5	7	12
0	0	0	1	1	1	0	0	4	1	1	3	5	7	13
0	0	0	1	1	1	0	1	2	2	1	3	5	7	14
0	0	0	1	1	1	0	2	0	3	1	3	5	7	15
0	0	1	0	0	0	3	0	4	0	1	3	5	7	16
0	0	1	0	0	0	3	1	2	1	1	3	5	7	17
0	0	1	0	0	0	3	2	0	2	1	3	5	7	18
0	0	1	0	0	1	2	0	3	1	1	3	5	7	19
0	0	1	0	0	1	2	1	1	2	1	3	5	7	20
0	0	1	0	0	2	1	0	2	2	1	3	5	7	21
0	0	1	0	0	2	1	1	0	3	1	3	5	7	22
0	0	1	0	0	3	0	0	1	3	1	3	5	7	23
0	0	1	0	1	0	1	0	4	1	1	3	5	7	24
0	0	1	0	1	0	1	1	2	2	1	3	5	7	25
0	0	1	0	1	0	1	2	0	3	1	3	5	7	26
0	0	1	0	1	1	0	0	3	2	1	3	5	7	27
0	0	1	0	1	1	0	1	1	3	1	3	5	7	28
0	1	0	0	0	0	2	0	5	0	1	3	5	7	29
0	1	0	0	0	0	2	1	3	1	1	3	5	7	30
0	1	0	0	0	0	2	2	1	2	1	3	5	7	31
0	1	0	0	0	1	1	0	4	1	1	3	5	7	32
0	1	0	0	0	1	1	1	2	2	1	3	5	7	33
0	1	0	0	0	1	1	2	0	3	1	3	5	7	34
0	1	0	0	0	2	0	0	3	2	1	3	5	7	35
0	1	0	0	0	2	0	1	1	3	1	3	5	7	36
0	1	0	0	1	0	0	0	5	1	1	3	5	7	37
0	1	0	0	1	0	0	1	3	2	1	3	5	7	38
0	1	0	0	1	0	0	2	1	3	1	3	5	7	39

Com a finalidade de determinar-se o número máximo de populações de mesmo tamanho e mesma frequência gênica, um tópico que até agora aparentemente não recebeu a devida atenção da literatura, elaboramos os programas **(programa 3 e 4)**, respectivamente para os casos de 2 a 4 alelos e 5 e 6 alelos. Nas tabelas ali mostradas, N_1 , N_2 , etc designam os números de alelos A_1 , A_2 , etc e NPOP o número máximo de populações possíveis com mesmo tamanho $N = (N_1 + \dots + N_k) / 2$, situação que ocorre quando as frequências gênicas $q_1, q_2, \dots, q_k = N_i / (N_1 + \dots + N_k)$ são iguais: $q_1 = q_2 = \dots = q_k = N_i / (N_1 + \dots + N_k)^1$.

a) Caso de três alelos

N_1	N_2	N_3	NPOP
2	2	2	5
4	4	4	15
6	6	6	34
8	8	8	65
10	10	10	111
12	12	12	175
14	14	14	260
16	16	16	369
18	18	18	505
20	20	20	671
22	22	22	870
24	24	24	1105
26	26	26	1379
28	28	28	1695
30	30	30	2056
32	32	32	2465
34	34	34	2925
36	36	36	3439
38	38	38	4010
40	40	40	4641

¹ Utilizado o programa Visual Basic 6.0 com o processador AUTHENTICAMD AMD-K6(tm) 3D processor 28.0 MB RAM.

b) Caso de quatro alelos

N ₁	N ₂	N ₃	N ₄	NPOP	t (seg)
1	1	1	1	3	0
2	2	2	2	17	0
3	3	3	3	47	0
4	4	4	4	138	0
5	5	5	5	306	0
6	6	6	6	670	0
7	7	7	7	1270	0
8	8	8	8	2355	1
9	9	9	9	4005	1
10	10	10	10	6671	1
11	11	11	11	10493	3
12	12	12	12	16212	5
13	13	13	13	24052	7
14	14	14	14	35148	13
15	15	15	15	49836	17
16	16	16	16	69765	27
17	17	17	17	95415	36
18	18	18	18	129085	56
19	19	19	19	171435	72
20	20	20	20	225566	107

c) Caso de cinco alelos

N ₁	N ₂	N ₃	N ₄	N ₅	NPOP	t (seg)
2	2	2	2	2	73	0
4	4	4	4	4	2021	10
6	6	6	6	6	25050	309
8	8	8	8	8	187475	5387
10	10	10	10	10	1000981	58323

d) Caso de seis alelos

N ₁	N ₂	N ₃	N ₄	N ₅	N ₆	NPOP	t (seg)
1	1	1	1	1	1	15	0
2	2	2	2	2	2	388	0
3	3	3	3	3	3	4720	2
4	4	4	4	4	4	43581	27
5	5	5	5	5	5	291001	146
6	6	6	6	6	6	1594340	1241
7	7	7	7	7	7	7260840	4826
8	8	8	8	8	8	28902275	26872
9	9	9	9	9	9	101924015	84403

A partir do caso de cinco alelos, mesmo para números populacionais relativamente modestos, o número de populações possíveis com mesmas frequências gênicas e o tempo de processamento aumentam numa taxa tal que inviabilizam, sob o ponto de vista prático, a tarefa de deduzir-se uma fórmula geral para o número máximo de populações possíveis com mesmo número amostral e mesmas frequências gênicas. Fica, portanto, praticamente impossível obter-se um conjunto mínimo de pontos usados em uma aplicação de técnica matemática (quadrados mínimos ou diferenças sucessivas) a fim da dedução de uma expressão analítica que forneça o número máximo de populações possíveis com mesma frequência, em função do tamanho amostral. Por exemplo, no caso de cinco alelos, com um tamanho amostral de apenas vinte e cinco indivíduos, são necessárias cerca de 17 horas para se calcular o número de populações possíveis. No caso de seis alelos e um tamanho amostral de 27, esse tempo passa a ser cerca de 24 horas.

Para os casos de dois a quatro alelos foi possível obter um número satisfatório de valores **NPOP**. Isso nos permitiu, através de técnicas como o método dos quadrados mínimos ou o método das diferenças finitas sucessivas, obter fórmulas gerais que fornecem o número máximo de populações possíveis de mesma frequência gênica e tamanho amostral para qualquer valor de **N** (tamanho da população). Para o caso de dois alelos a função polinomial é dada simplesmente por

$$y(2) = 1 + N/2.$$

No caso de três alelos, a função tem forma

$$y(3) = 1 + 2N/3 + N^2/6 + N^3/54.$$

Para o caso de quatro alelos, existem duas funções distintas, uma para o caso do tamanho total da população ser par (y') e outra para o caso de ele ser ímpar (y''):

$$y'(4) = 1 + 19N/24 + 43N^2/144 + 9N^3/128 + 47N^4/4608 + 5N^5/6144 + N^6/36864$$

$$y''(4) = 3/8 + 181N/384 + 553N^2/2304 + 17N^3/256 + 47N^4/4608 + 5N^5/6144 + N^6/36864$$

TESTES "EXATOS" BASEADOS EM SIMULAÇÕES

Para contornar as dificuldades impostas pelo aumento do tempo de processamento e da complexidade computacional, exponencialmente crescentes com o aumento do número de alelos e o aumento do tamanho da amostra, o processo pode ser simulado em computador eletrônico, através do emprego do método Monte Carlo. Por exemplo, para determinar a probabilidade de os genótipos de uma amostra seguirem as proporções de Hardy-Weinberg, extraem-se, inicialmente, as frequências alélicas p e q a partir dos dados amostrais observados $D = n_{AA}$, $H = n_{Aa}$ e $R = n_{aa}$ segundo $p = (2D+H)/2n$ e $q = 1-p = (H+2R)/2n$. Em seguida procede-se ao sorteio, começando por gerar-se um número aleatório normalizado entre 0 e 1: se o número for menor ou igual a p , indica que um gene **A** foi obtido entre os $2n$ genes da amostra; se o número for maior que p , indica que o gene sorteado aleatoriamente foi o **a**. O processo é então repetido $2n-1$ vezes, sempre partindo-se de um número amostral de genes $2n$ e sempre comparando-se o número pseudoaleatório gerado pelo computador com o valor p . Um contador armazena o número de vezes (x) em que o gene **A** é sorteado, obtendo-se ao final do processo a quantidade de vezes que ele ocorreu; a frequência gênica do alelo **A** nessa primeira simulação é dada por $P(A) = x/2n$ e a do alelo **a** por $P(a) = 1-P(A) = 1 - x/2n$. Cada dois genes sorteados em sequência são usados para definir um genótipo. Os contadores D' , H' e R' armazenam a quantidade de vezes que os genótipos **AA**, **Aa** e **aa** foram obtidos em cada simulação de $2n$ genes e n indivíduos, completando uma população simulada. São feitas t simulações de populações (em geral, t é da ordem de 10000). Após cada simulação é calculado o valor de P_i como no teste exato e comparado com o de P_0 ; a

probabilidade exata, obtida ao final das $t = 10000$ simulações, é dada pela expressão $P = T/10000$, em que T é o número de vezes em que P_i é menor ou igual a P_0 . O **programa 6** executa exatamente isso.

NO. OF SIMULATED SAMPLES OF SIZE 300 = 10000
TOTAL ELAPSED TIME (ALL CALCULATIONS) = 1441 SEC.
DATA FROM DATAFILE TEMP1.DAT

SAMPLE P = 0.5000
SAMPLE D = 90.0000
SAMPLE H = 120.0000
SAMPLE R = 90.0000

P MEAN = 0.4999
OBS.D MEAN = 74.9785
OBS.H MEAN = 149.9842
OBS.R MEAN = 75.0373

EX. PROB. = 0.0003

É possível realizar uma comparação entre o desempenho de cada tipo de teste exato (simulação com ou sem reposição e modelo sem simulação baseado no modelo de Haldane), de acordo com as condições já mencionadas anteriormente na introdução:

Figura 17: gráfico de dispersão mostrando o comportamento do teste exato por simulações com reposição, quando comparado com o teste exato clássico (Haldane). N = 50 indivíduos.

Figura 18: gráfico de dispersão mostrando o comportamento do teste exato por simulações sem reposição, quando comparado com o teste exato clássico (Haldane). N = 50 indivíduos.

Verifica-se que o teste exato baseado em simulações sem reposição é virtualmente similar ao teste exato clássico que enumera todas as populações possíveis com mesmas frequências gênicas e tamanho amostral.

O **programa 7** realiza o teste exato para o equilíbrio de Hardy-Weinberg agora em relação a um número qualquer de alelos.

```

ALLELE( 1) =      40
GENOT( 1, 1) =     10
GENOT( 1, 2) =     20
ALLELE( 2) =      40
GENOT( 2, 2) =     10
N =          40

```

OVERALL RESULTS BASED ON 1000 SIMULATIONS OF SIZE 40 INDIVIDUALS
TOTAL ELAPSED TIME (ALL CALCULATIONS) = 33 SEC.

```

ALLELE( 1) =      40.1080
GENOTYPE( 1, 1) =    10.0840
GENOTYPE( 1, 2) =    19.9400
ALLELE( 2) =      39.8920
GENOTYPE( 2, 2) =     9.9760

```

EXACT PROB. = 0.8620

```

ALLELE( 1) =      23
GENOT( 1, 1) =      5
GENOT( 1, 2) =      6
GENOT( 1, 3) =      7
ALLELE( 2) =      31
GENOT( 2, 2) =      8
GENOT( 2, 3) =      9
ALLELE( 3) =      36
GENOT( 3, 3) =     10
N =          45

```

OVERALL RESULTS BASED ON 1000 SIMULATIONS OF SIZE 45 INDIVIDUALS
TOTAL ELAPSED TIME (ALL CALCULATIONS) = 45 SEC.

```

ALLELE( 1) =      22.9350
GENOTYPE( 1, 1) =     2.9120
GENOTYPE( 1, 2) =     8.0470
GENOTYPE( 1, 3) =     9.0640
ALLELE( 2) =      31.1070
GENOTYPE( 2, 2) =     5.2830
GENOTYPE( 2, 3) =    12.4940
ALLELE( 3) =      35.9580
GENOTYPE( 3, 3) =     7.2000

```

EXACT PROB. = 0.1110

Os programas discutidos a seguir simulam populações com probabilidades genotípicas $\{d, h, r\}$ ou $\{p^2, 2pq, q^2\}$, com a finalidade de comparar graficamente os aglomerados de pontos populacionais nas duas hipóteses.

O **programa 8** mostra um programa que simula um número qualquer de populações com probabilidades genotípicas d, h e r .

DATA FROM DATAFILE TEMP.DAT

SAMPLE P = 0.5000
SAMPLE D = 90.00
SAMPLE H = 120.00
SAMPLE R = 90.00

P MEAN = 0.4997
OBS.D MEAN = 89.85
OBS.H MEAN = 120.10
OBS.R MEAN = 90.05

As populações assim simuladas podem ser plotadas num diagrama triangular (Otto e Benedetti, 1995) através de programas simples como o **programa 9**:

O **programa 10** simula um número qualquer de populações com frequências genotípicas p^2 , $2pq$ e q^2 (em vez de d , h e r como no **programa 8**).

DATA FROM DATAFILE temp2.dat

SAMPLE P = 0.5000
SAMPLE D = 90.0000

```

SAMPLE H      = 120.0000
SAMPLE R      = 90.0000

P MEAN       = 0.4998
OBS.D MEAN   = 74.9569
OBS.H MEAN   = 149.9592
OBS.R MEAN   = 75.0839

```

O conjunto de populações assim simuladas pode ser plotado num diagrama triangular (Otto e Benedetti, 1995) através de programas simples como o GRAPHT02.BAS (**programa 9**) usado anteriormente:

Algumas linhas extras de programação permitem determinar a zona de superposição das regiões correspondentes ao intervalo de confiança a $1 - \alpha$ dos dois aglomerados de pontos, assim verificando diretamente a hipótese de pan-mixia testada. Observamos empiricamente que, se

aproximadamente pelo menos **70%** do total de pontos simulados dos aglomerados estiverem contidos na interseção das duas regiões de credibilidade **$1-\alpha$** (**$\{d, h, r\}$** e **$\{p^2, 2pq, q^2\}$**), a hipótese de pan-mixia pode ser aceita. O programa plota sob as duas hipóteses (**H_0** e **H_a**) 10000 pontos simulados em cada caso; a região exata de credibilidade a **95%** é construída, para cada hipótese, da elipse formada a partir do aglomerado de pontos resultantes da plotagem dos pontos simulados. A proporção entre os raios de cada elipse é pré-fixada, dependendo do tamanho amostral e dos valores das frequências gênicas nas duas situações e também do valor de **F** no caso saturado. A seguir, partindo-se do centróide da elipse, incrementa-se gradativamente o valor do raio principal (obtendo-se o valor do raio secundário como função da proporção pré-fixada anteriormente) até que a elipse assim construída contenha **95 %** dos pontos simulados. Em seguida, traçam-se as elipses com os raios calculados, numa inclinação que depende do valor da frequência gênica e do tamanho amostral, a partir dos centróides fixados nos pontos amostrais **$\{d, h, r\}$** e **$\{p^2, 2pq, q^2\}$** .

A listagem 11 ilustra o código do programa que realiza todas essas tarefas, além de mostrar as curvas que representam os conjuntos de pontos **$\{d = p^2 + pqF, h = 2pq(1 - F), r = q^2 + pqF\}$** , **$\{p^2, 2pq, q^2\}$** e os limites aproximados do intervalo de confiança a **95%** da parábola De Finetti, os quais contêm os conjuntos de pontos **$\{p^2 + pqF_i, 2pq(1 - F_i), q^2 + pqF_i\}$** e **$\{p^2 + pqF_s, 2pq(1 - F_s), q^2 + pqF_s\}$** , em que **$F_i$** e **$F_s$** são as soluções da equação **$F_{i,s} = \sqrt{(\chi^2_{crit} / N)}$** , com **$\chi^2_{crit} = 3.841$** e **$N$** = tamanho da amostra. Nesse caso, o eixo principal dessa elipse é simetricamente perpendicular (em relação ao eixo x de frequência gênica)

à tangente à curva no ponto amostrado (tanto no caso saturado como no caso de equilíbrio).

DETERMINAÇÃO GRÁFICA DA REGIÃO DE CREDIBILIDADE A 95% USANDO O MÉTODO DE SUBAMOSTRAGENS (JACKKNIFE)

O Método jackknife minimiza os vieses inerentes aos processos simulatórios. Nas linhas a seguir descrevemos a plotagem dos pontos das subamostragens geradas pelo método jackknife, ao invés dos pontos gerados no método bootstrap, usado no programa precedente.

O primeiro problema que surge é a respeito do tamanho das subamostragens. Um excesso de indivíduos reamostrados pode não compensar o erro inerente ao método bootstrap, ao mesmo tempo que um reduzido tamanho de reamostragem pode reforçar o erro do método bootstrap, o que não é desejável.

Verificamos que os vieses que tendem a inflar os intervalos de confiança a **95%** (ou a qualquer outro intervalo a um nível **$1 - \alpha$**) são minimizados de maneira ótima pelo método jackknife quando a relação **$\alpha = 2 * [1 - N_S / N]$** é observada, onde **N_S** é o tamanho das subamostragens e **N** o tamanho amostral. Em termos práticos, isso pode ser expresso como **$N_S \approx 97.5\% N$** para **$1 - \alpha$** equivalente a **95%**.

Para o caso de **$p=0.3$** e **$n=100$** , obtemos a seguinte figura:

A inspeção do gráfico acima mostra que a inclinação da elipse que representa a região de credibilidade no caso do bootstrap, desaparece no caso do método jackknife.

A CONSTRUÇÃO DE INTERVALOS DE CONFIABILIDADE

O processo de simulação empregado para o caso específico do equilíbrio de Hardy-Weinberg recebe o nome de bootstrap, que em sentido mais amplo é a reamostragem por sorteio ao acaso de uma amostra; fazendo-se isso um número grande de vezes (por exemplo $t = 10000$), podemos obter a distribuição esperada de todos os valores e estimar seus parâmetros de interesse e os limites de um intervalo qualquer de credibilidade (por exemplo, ordenando todas as probabilidades referentes à população sorteada e retirando os 2,5% valores menores e os 2,5% valores maiores obtemos o intervalo de credibilidade correspondente aproximadamente ao intervalo exato de confiança a 95% dos parâmetros em estudo, como frequências gênicas ou genotípicas). Esses intervalos de credibilidade já podem ser comparados entre si diretamente em substituição a testes estatísticos clássicos. Por exemplo, no caso de comparação de duas amostras com frequências alélicas p' e p'' a não superposição dos respectivos intervalos de confiança a 95% demonstra que as frequências são diferentes ao nível crítico de 5%. Este processo de construção de intervalos de confiança é ilustrado pelo programa da **programa 12**, no qual são calculados intervalos de confiança para classes genotípicas de populações contendo 2 ou mais alelos.

D,H,R = 10,15,20
 ALLELE(1) = 35
 P(1, 1) = 10
 P(1, 2) = 15
 ALLELE(2) = 55
 P(2, 2) = 20
 N = 45

GENOTYPE	P	95 % P confidence interval	
		normal	bootstrap
1/1	obs. 0.222	{0.101,0.344}	{0.111,0.356}
	exp. 0.151	{0.073,0.230}	{0.090,0.239}
1/2	obs. 0.333	{0.196,0.471}	{0.200,0.489}
	exp. 0.475	{0.431,0.520}	{0.411,0.499}
2/2	obs. 0.444	{0.299,0.590}	{0.289,0.578}
	exp. 0.373	{0.250,0.497}	{0.261,0.490}

N(1, 1) = 5
 N(1, 2) = 6
 N(1, 3) = 7
 N(2, 2) = 8
 N(2, 3) = 9
 N(3, 3) = 10
 ALLELE(1) = 23
 P(1, 1) = 5
 P(1, 2) = 6
 P(1, 3) = 7
 ALLELE(2) = 31
 P(2, 2) = 8
 P(2, 3) = 9
 ALLELE(3) = 36
 P(3, 3) = 10
 N = 45

GENOTYPE	P	95 % P confidence interval	
		normal	bootstrap
1/1	obs. 0.111	{0.019,0.203}	{0.022,0.200}
	exp. 0.065	{0.019,0.111}	{0.028,0.119}
1/2	obs. 0.133	{0.034,0.233}	{0.044,0.244}
	exp. 0.176	{0.115,0.237}	{0.118,0.237}
1/3	obs. 0.156	{0.050,0.261}	{0.067,0.267}
	exp. 0.204	{0.139,0.270}	{0.142,0.272}
2/2	obs. 0.178	{0.066,0.289}	{0.067,0.289}
	exp. 0.119	{0.051,0.186}	{0.060,0.198}
2/3	obs. 0.200	{0.083,0.317}	{0.089,0.333}
	exp. 0.276	{0.208,0.343}	{0.207,0.338}
3/3	obs. 0.222	{0.101,0.344}	{0.111,0.356}
	exp. 0.160	{0.079,0.241}	{0.090,0.250}

COMPARAÇÃO DE INTERVALOS DE CONFIABILIDADE USANDO TÉCNICAS DE REAMOSTRAGEM "BOOTSTRAP" E "JACKKNIFE"

A técnica de reamostragem por bootstrap fornece uma estimativa geral do parâmetro (obtida tomando-se a média de todos os valores do parâmetro obtidos nas t simulações) e uma distribuição do parâmetro não-enviesadas. Argumentações de natureza teórica demonstram, no entanto, que a medida de erro estatístico (variância) do parâmetro possui viés e que esse viés pode ser corrigido pela aplicação do método jackknife. Esses problemas de natureza teórica envolvendo os conceitos de simulação Monte Carlo, bootstrap e jackknife são discutidos em profundidade por Yu (2003), Lanyon (1987) e Walsh (2000).

A seguir apresentamos uma breve descrição sobre a técnica jackknife. Por exemplo, seja a amostra $n(AA) = 30$, $n(Aa) = 50$, $n(aa) = 20$, da qual deseja-se estimar a frequência gênica p e a respectiva variância. Como o número amostral é 100, é possível construir-se exatamente 100 subamostras de tamanho $n-1 = 99$, retirando-se de cada amostra, a cada vez, um indivíduo; essas subamostras estão assim distribuídas : 30 [$n(AA) = 29$, $n(Aa) = 50$, $n(aa) = 20$], 50 [$n(AA) = 30$, $n(Aa) = 49$, $n(aa) = 20$] e 20 [$n(AA) = 30$, $n(Aa) = 50$, $n(aa) = 19$]. São realizadas então 100 simulações bootstrap a partir dessas populações (ou, alternativamente, um número qualquer de simulações, partindo-se cada vez de uma população de tamanho $n-1$ indivíduos, após a retirada ao acaso, também efetuada pelo computador, de um indivíduo com genótipo **AA**, **Aa** ou **aa**). Em vez de serem usadas reamostragens de tamanho $n-1$, o tamanho pode ser qualquer $n' < n$, por exemplo $n/2$. Entretanto, o tamanho ideal para n' , como já comentado, corresponde ao tirado da equação $n' \approx (1 - \alpha/2)n$.

Cada uma das simulações realizadas com a reamostragem de n' indivíduos a partir dos n da amostra total fornece estimativas P_i e P_{ij} da frequência do alelo a_i e do genótipo $a_i a_j$. As estimativas das variâncias desses parâmetros, baseadas nas t simulações, são menos enviesadas que no caso das obtidas através do processo de bootstrap.

O código do **programa 15** define a execução do jackknife através de reamostragens da amostra de tamanho n para o caso geral de um número qualquer de alelos segregando num loco autossômico qualquer.

Os programas acima citados calculam apenas as frequências gênicas e genotípicas e os intervalos de confiança dos valores observados e esperados destas últimas; acrescentamos nas tabelas mostradas abaixo os intervalos correspondentes à aproximação normal e ao método bootstrap calculados pelo programa BOOTSTR3.BAS (**programa 12** - já mostrado na seção anterior), com a finalidade de comparar os dois conjuntos de resultados aos obtidos pelo método jackknife.

```

"
OBSERVED GENOTYPES
"
N(1-1) = 25
"
N(1-2) = 15
"
N(1-3) = 30
"
N(2-2) = 25
"
N(2-3) = 20
"
N(3-3) = 10
"
NUM. OF SIMULATIONS = 10000
SIZE OF SUB-SAMPLES = 0.975 SAMPLE SIZE

P          95% Jackknife Confidence Interval
OBSP( 1) = 0.3800
MEDP( 1) = 0.3789      {0.331,0.427}
OBSP( 2) = 0.3400
MEDP( 2) = 0.3393      {0.293,0.390}
OBSP( 3) = 0.2800
MEDP( 3) = 0.2792      {0.235,0.325}
          95 % P confidence interval
GENOTYPE      P      normal      bootstrap      97.5% S.S jackknife
-----
1/1  obs. 0.200 {0.130,0.270} {0.136,0.272} {0.144,0.256}
     exp. 0.144 {0.099,0.190} {0.102,0.194} {0.109,0.182}
1/2  obs. 0.120 {0.063,0.177} {0.064,0.176} {0.075,0.165}
     exp. 0.258 {0.218,0.299} {0.219,0.298} {0.224,0.289}
1/3  obs. 0.240 {0.165,0.315} {0.168,0.312} {0.181,0.299}
     exp. 0.213 {0.174,0.252} {0.171,0.256} {0.179,0.243}
2/2  obs. 0.200 {0.130,0.270} {0.128,0.272} {0.144,0.261}
     exp. 0.116 {0.076,0.156} {0.081,0.157} {0.086,0.152}
2/3  obs. 0.160 {0.096,0.224} {0.096,0.224} {0.112,0.213}
     exp. 0.190 {0.153,0.228} {0.154,0.233} {0.160,0.220}
3/3  obs. 0.080 {0.032,0.128} {0.032,0.136} {0.043,0.123}
     exp. 0.078 {0.047,0.110} {0.050,0.113} {0.055,0.106}
-----

```

COMPARAÇÃO DOS RESULTADOS OBSERVADOS PELO TESTE EXATO PROPRIAMENTE DITO (SEM SIMULAÇÕES) COM OS OBTIDOS A PARTIR DE SIMULAÇÕES SEM REPOSIÇÃO

O processo bootstrap acima descrito corresponde a um modelo de urna de tamanho infinito (sorteio com reposição). Modificações mais ou menos simples podem ser introduzidas nesse modelo de modo a executar um sorteio sem reposição: obviamente, este modelo admite que a frequência gênica populacional verdadeira é p , obtida na amostra $D + H + R = n$. A diferença fundamental operacional com o modelo com reposição é que, após a geração de cada número aleatório da sequência de $2n$ realizada em cada uma das t simulações, o valor de p de comparação no sorteio seguinte é calibrado de acordo com o gene (A ou a) que foi sorteado. Isso garante que todas as t reamostragens obtidas a partir da população tenham a mesma frequência gênica p e o resultado (quando baseado num número adequadamente suficiente de simulações) deve corresponder exatamente ao teste exato descrito em primeiro lugar para o caso de dois alelos.

Utilizando os dados tabulados pelo **programa 13**:

DATA FROM DATAFILE temp1.dat

SAMPLE P = 0.3333
 SAMPLE D = 5.0000
 SAMPLE H = 10.0000
 SAMPLE R = 15.0000

SIM.P MEAN = 0.3333
 SIM.D MEAN = 3.2323
 SIM.H MEAN = 13.5354
 SIM.R MEAN = 13.2323

D	H	R	f(x)	F(x)
8	4	18	0.000	0.000
7	6	17	0.004	0.004
0	20	10	0.007	0.011
6	8	16	0.029	0.040
1	18	11	0.062	0.102
5	10	15 *	0.115	0.217
2	16	12	0.209	0.426
4	12	14	0.255	0.681
3	14	13	0.319	1.000

e os correspondentes gerados para a mesma amostra {5, 10, 15} pelo programa HWEXAC01.BAS (**programa 2**), que realiza o teste exato propriamente dito, podemos comparar diretamente os resultados obtidos pelos dois métodos:

			VAL. SIMULAD.		VAL. "EXATOS"		e r r o s	
D	H	R	$f_1(x)$	$F_1(x)$	$f_2(x)$	$F_2(x)$	e.a.	e.r.
7	6	17	0.003	0.003	0.003	0.003	0.000	0.000
0	20	10	0.006	0.009	0.008	0.011	0.002	0.182
6	8	16	0.038	0.047	0.027	0.038	0.009	0.237
1	18	11	0.061	0.108	0.065	0.103	0.005	0.049
5	10	15 *	0.106	0.214	0.114	0.217	0.003	0.014
2	16	12	0.212	0.426	0.207	0.424	0.002	0.005
4	12	14	0.264	0.690	0.259	0.683	0.007	0.010
3	14	13	0.310	1.000	0.318	1.000	0.000	0.000

Na tabela acima, a primeira coluna de probabilidades corresponde aos valores da função densidade $f_1(x)$ obtidos em 1000 simulações; a segunda

coluna apresenta os valores acumulados da primeira coluna, equivalendo portanto à função distribuição $F_1(x) = \sum f_1(x)$. As duas colunas seguintes mostram as funções correspondentes $f_2(x)$ e $F_2(x) = \sum f_2(x)$ calculadas pelo teste "exato" (programa HWEXAC01.BAS - **PROGRAMA 2**); finalmente, as duas últimas colunas mostram os valores dos erros absolutos ($e.a. = |F_1(x) - F_2(x)|$) e relativos [$e.r. = e.a./F_2(x)$]. A tabela demonstra claramente que a simulação constitui uma boa aproximação dos valores exatos, mesmo usando-se um número relativamente modesto (1000) de simulações. Para melhorar a precisão do bootstrap, poder-se-ia aumentar o número de simulações, com prejuízo evidente do tempo de processamento do programa.

Apesar da vantagem (aparente) do teste exato propriamente dito fornecer as probabilidades exatas de ocorrência das diversas configurações genotípicas para o mesmo número amostral, ele realiza isso sob hipótese de a frequência da população ser exatamente a inferida da amostra, o que claramente não é verdade. Portanto, o método de simulação com reposição, que claramente corresponde ao teste exato, carrega consigo a mesma imprecisão implícita deste. De qualquer maneira, o método de simulação sem reposição equivale ao método dito "exato" e portanto pode ser usado para substituí-lo em condições de impossibilidade de aplicação do teste "exato", por exemplo no caso de amostras grandes ou de um número de alelos maior que dois, em que é praticamente impossível enumerar todas as combinações genotípicas possíveis com mesma frequência gênica. Outra observação importante tirada da comparação entre os dois métodos é que deve-se dar preferência ao método "exato" apenas quando o número amostral for muito pequeno. Com

o aumento modesto do número amostral os dois métodos tornam-se equivalentes sob o ponto de vista de tempo computacional; atingido um certo patamar, o método que utiliza as simulações torna-se mais eficiente, como mostramos abaixo de maneira empírica, comparando os tempos de computação usando os dois métodos em amostras idênticas.

TESTES EXATOS POR SIMULAÇÃO SEM REPOSIÇÃO PARA O CASO GERAL DE n ALELOS

O **programa 14** calcula, por simulação, as probabilidades exatas associadas a amostras populacionais de genótipos resultantes da combinação de um número qualquer de alelos. Na prática, devido ao tempo excessivo de processamento, o método só é factível no máximo para três alelos e um tamanho amostral modesto.

11	12	13	22	23	33	F(x)

5	1	10	11	18	2	0.0010
6	7	2	12	10	10	0.0020
7	3	4	10	18	5	0.0030
5	9	2	5	22	4	0.0040
1	15	4	3	20	4	0.0050
0	13	8	4	20	2	0.0060
0	16	5	8	9	9	0.0070
1	6	13	9	17	1 *	0.0080
6	7	2	9	16	7	0.0090
0	7	14	10	14	2	0.0100
..	..	..	..	..	..	
3	8	7	10	13	6	0.8290
2	8	9	10	13	5	0.8400
3	9	6	8	16	5	0.8560
2	9	8	8	16	4	0.8710
2	9	8	10	12	6	0.8940
2	11	6	8	14	6	0.9150
3	9	6	9	14	6	0.9260
3	8	7	9	15	5	0.9460
2	10	7	9	13	6	0.9620
2	10	7	8	15	5	0.9830
2	9	8	9	14	5	1.0000

Esse programa calcula o teste exato através de simulações de populações com mais de dois alelos, utilizando o método da cadeia de Markov, criando um vetor auxiliar numérico cujos elementos contêm exatamente a quantidade de genes presentes na amostra fornecida pelo usuário. Se existirem **x** genes do tipo 1, **y** genes do tipo 2, **z** genes do

tipo 3, etc., teremos **x** elementos 1, **y** elementos 2, **z** elementos 3, etc. ordenados de maneira aleatória no vetor numérico resultante. A partir desses elementos, são realizados n sorteios duplos que resultarão em uma nova população simulada de genótipos. A fim de garantir que o sorteio seja sem reposição, trocam-se os genes já sorteados pelos genes presentes nas últimas posições válidas (final) do vetor, diminuindo-se, ao mesmo tempo, o comprimento válido desse mesmo vetor em duas unidades a cada iteração, de maneira que na última iteração (correspondente ao n ésimo indivíduo) garante-se que todos os genes presentes no vetor auxiliar foram sorteados, gerando uma nova população com o mesmo número amostral e mesmas frequências gênicas iniciais.

Os resultados apresentados pelo **PROGRAMA 14**, foram obtidos para o teste da população com **$N = 47$, $N(11) = 1$, $N(12) = 6$, $N(13) = 13$, $N(22) = 9$, $N(23) = 17$ e $N(33) = 1$** e frequências gênicas constantes **$P(1) = 21/94$, $P(2) = 41/94$ e $P(3) = 32/94$** . A probabilidade (obtida por simulação) correspondente à população observada, assinalada por um asterisco, é **$P = 0.0080$** , a qual descarta a possibilidade de equilíbrio de Hardy-Weinberg em relação à amostra.

TESTES EXATOS PARA A ESTATÍSTICA F USADA NO ESTUDO DA ESTRUTURAÇÃO HIERÁRQUICA DAS POPULAÇÕES

Considerando-se a situação geral de k isolados diferentes, dentro de cada um dos quais as freqüências genotípicas são dadas por

$$\begin{aligned}P_i(AA) &= p_i^2 + F_i p_i q_i = F_i p_i + (1-F_i) p_i^2, \\P_i(Aa) &= 2p_i q_i (1-F_i), \\P_i(aa) &= q_i^2 + F_i p_i q_i = F_i q_i + (1-F_i) q_i^2 ;\end{aligned}$$

as freqüências genotípicas na população total são respectivamente

$$\begin{aligned}P(AA) &= \sum x_i [p_i^2 + F_i p_i q_i], \\P(Aa) &= 2 \sum x_i p_i q_i (1 - F_i), \\P(aa) &= \sum x_i [q_i^2 + F_i p_i q_i],\end{aligned}$$

em que p_i , q_i são as freqüências alélicas e F_i é o índice de fixação da i-ésima subpopulação; $x_i = N_i / \sum N_i$ é a contribuição em tamanho da i-ésima subpopulação para a população total. Na população total as freqüências alélicas são calculadas segundo

$$\begin{aligned}p &= \sum x_i p_i \\q &= 1 - p = \sum x_i q_i\end{aligned}$$

e a variância de freqüências gênicas entre as subpopulações (isolados) segundo

$$\begin{aligned}\text{var}(p) &= \sum x_i (p_i - p)^2 = \sum x_i p_i^2 - p^2 \\&= \text{var}(q) = \sum x_i (q_i - q)^2 = \sum x_i q_i^2 - q^2 .\end{aligned}$$

A correlação entre gametas tomados ao acaso dentro das subpopulações em relação aos gametas da população total, ou seja o índice de fixação gerado pela subdivisão populacional ou efeito de Wahlund (F_{ST}) é calculado segundo

$$\begin{aligned}F_{ST} = \text{var}(p)/pq &= [\sum x_i p_i^2 - (\sum x_i p_i)^2] / (\sum x_i p_i \cdot \sum x_i q_i) \\&= [\sum x_i p_i^2 - (\sum x_i p_i) (1 - \sum x_i q_i)] / (\sum x_i p_i \cdot \sum x_i q_i) \\&= [\sum x_i p_i^2 - \sum x_i p_i + \sum x_i p_i \cdot \sum x_i q_i] / (\sum x_i p_i \cdot \sum x_i q_i) \\&= [\sum x_i p_i \cdot \sum x_i q_i - \sum x_i p_i (1 - p_i)] / (\sum x_i p_i \cdot \sum x_i q_i) \\&= 1 - \sum x_i p_i q_i / (\sum x_i p_i \cdot \sum x_i q_i) = 1 - 2 \sum x_i p_i q_i / 2pq\end{aligned}$$

A correlação entre gametas que se combinam em relação aos gametas da população total (F_{IT}), ou seja o índice de fixação na população total devido a tanto a subdivisão populacional quanto à endogamia ocorrendo dentro das subpopulações é obtido diretamente de

$$F_{IT} = 1 - \sum x_i p_i (Aa) / 2pq = 1 - P(Aa) / 2pq \\ = 1 - 2 \sum x_i p_i q_i (1 - F_i) / 2pq$$

O valor de F_{IS} , o índice de fixação devido à endogamia dentro das subpopulações, é tirado de $F_{IS} = (F_{IT} - F_{ST}) / (1 - F_{ST})$, porque $F_{IT} = F_{ST} + F_{IS} - F_{IS} \cdot F_{ST}$; a última equação significa simplesmente que para um indivíduo qualquer ser heterozigoto é necessário que ele não seja homozigoto nem por causa da endogamia dentro das subpopulações nem por causa do efeito de Wahlund. De fato, de

$$(1 - F_{IT}) = (1 - F_{IS})(1 - F_{ST})$$

obtemos sucessivamente

$$1 - F_{IT} = 1 - F_{IS} - F_{ST} + F_{IS} \cdot F_{ST} , \\ F_{IT} = F_{ST} + F_{IS}(1 - F_{ST}) \\ e \\ F_{IS} = (F_{IT} - F_{ST}) / (1 - F_{ST}) .$$

Uma vez que

$$F_{IT} = 1 - P(Aa) / 2pq$$

e

$$F_{ST} = \text{var}(p) / pq ,$$

vem que

$$F_{IS} = 1 - P(Aa) / \{2[pq - \text{var}(p)]\} .$$

Porém, como

$$P(Aa) = 2 \sum x_i p_i q_i (1 - F_i)$$

e

$$pq - \text{var}(p) = pq - \sum x_i p_i^2 + p^2 = p - \sum x_i p_i^2 = \sum x_i p_i - \sum x_i p_i^2 \\ = \sum x_i p_i q_i ,$$

a equação para F_{IS} pode ser reescrita como

$$F_{IS} = 1 - \sum x_i p_i q_i (1 - F_i) / \sum x_i p_i q_i = 1 - 2 \sum x_i p_i q_i (1 - F_i) / 2 \sum x_i p_i q_i ,$$

de onde obtemos sucessivamente

$$1 - F_{IS} = \sum x_i p_i q_i (1 - F_i) / \sum x_i p_i q_i$$

e

$$F_{IS} = \sum x_i p_i q_i F_i / \sum x_i p_i q_i .$$

Os **programas 16 e 17**, além de estimarem através de métodos convencionais os parâmetros **F_{st}**, **F_{it}** e **F_{is}**, determinam através de simulações os intervalos de confiabilidade a **95%** dessas estimativas, para o caso de um número qualquer de subpopulações mas apenas um loco autossômico com dois alelos. Quando os intervalos contiverem o valor zero, conclui-se que o parâmetro não difere significativamente de zero ao nível crítico **$\alpha = 0.05$** . Para determinar esses intervalos de confiança, o programa BOOTHIE1.BAS executa 1000 simulações de cada uma das **n** subpopulações. Já o programa BOOTHIE2.BAS executa, no caso de três subpopulações, apenas 10 simulações de cada, e os diversos valores de **F_{st}**, **F_{it}** e **F_{is}** são obtidos a partir das $10 \times 10 \times 10 = 1000$ combinações possíveis entre as três subpopulações. Os resultados obtidos com os dois métodos são equivalentes. O tempo de processamento do segundo programa é, no entanto, de uma ordem de grandeza drasticamente inferior ao do primeiro.

ESTIMATES BASED ON SAMPLE NUMBERS

SUBPOP.	N(AA)	N(Aa)	N(aa)	N
1	9	12	54	75
2	5	10	10	25
3	33	14	3	50
total	47	36	67	150

p(1) = 0.2000
 F(1) = 0.5000
 p(2) = 0.4000
 F(2) = 0.1667
 p(3) = 0.8000
 F(3) = 0.1250
 p = 0.4333
 var(p) = 0.0722
 FIT = 0.5113
 FST = 0.2941
 FIS = 0.3077

AVERAGE ESTIMATES BASED ON 1000 SIMULATIONS

p(1) = 0.2002
 F(1) = 0.4886
 p(2) = 0.3955
 F(2) = 0.1457
 p(3) = 0.7991
 F(3) = 0.1129
 FIT = 0.5085

```
FST    = 0.3009
FIS    = 0.2955
```

MEDIANS AND 95% BOOTSTRAP CONFIDENCE INTERVALS

```
F( 1)  : 0.500 {0.210,0.737}
F( 2)  : 0.143 {-0.273,0.554}
F( 3)  : 0.107 {-0.190,0.415}
FIT     : 0.514 {0.369,0.638}
FST     : 0.302 {0.197,0.422}
FIS     : 0.296 {0.112,0.467}
```

ESTIMATES BASED ON SAMPLE NUMBERS

SUBPOP.	N(AA)	N(Aa)	N(aa)	N
1	9	12	54	75
2	5	10	10	25
3	33	14	3	50
total	47	36	67	150

```
p( 1)  = 0.2000
F( 1)  = 0.5000
p( 2)  = 0.4000
F( 2)  = 0.1667
p( 3)  = 0.8000
F( 3)  = 0.1250
p      = 0.4333
var(p) = 0.0722
FIT     = 0.5113
FST     = 0.2941
FIS     = 0.3077
```

AVERAGE ESTIMATES BASED ON 10 SIMULATIONS

```
p( 1)  = 0.1960
F( 1)  = 0.4972
p( 2)  = 0.3920
F( 2)  = 0.0857
p( 3)  = 0.7840
F( 3)  = 0.1253
```

AVERAGE ESTIMATES BASED ON 1000 SIMULATIONS

```
FIT     = 0.4940
FST     = 0.2909
FIS     = 0.2857
```

MEDIANS AND 95% BOOTSTRAP CONFIDENCE INTERVALS

```
FIT     : 0.493 {0.398,0.588}
FST     : 0.290 {0.198,0.398}
FIS     : 0.282 {0.176,0.390}
```

O método jackknife, já utilizado para o equilíbrio de Hardy-Weinberg, também pode ser aplicado na análise da estruturação de populações. Os intervalos de confiança a $1 - \alpha$ das frequências gênicas ficam diminuídos por este método em relação ao método bootstrap, o que já foi demonstrado anteriormente. Entretanto, para os valores de **F_{st}**, **F_{it}** e **F_{is}** os intervalos de confiança ficam aumentados comparados ao

bootstrap, pois os diversos coeficientes F são funções recíprocas dos produtos dos valores das frequências gênicas.

O comportamento desses intervalos de confiança pode ser verificado a seguir (**programa 19**):

```
OBSERVED POPULATIONS

POPULATION 1
D = 37; H = 49; R = 102

POPULATION 2
D = 167; H = 33; R = 49

POPULATION 3
D = 155; H = 12; R = 26

POPULATION 4
D = 19; H = 179; R = 66

-----
          95% CONFIDENCE INTERVALS
          P      BOOTSTRAP      97.5% SS JACKKNIFE
-----
FIT = 0.3764 { 0.3260, 0.4303} { 0.3254, 0.4306}
F(1)= 0.4070 { 0.2733, 0.5288} { 0.2748, 0.5309}
F(2)= 0.6589 { 0.5501, 0.7654} { 0.5497, 0.7657}
F(3)= 0.7713 { 0.6282, 0.8842} { 0.6282, 0.8870}
F(4)=-0.3987 {-0.4994,-0.2909} {-0.5011,-0.2939}
FST = 0.1773 { 0.1402, 0.2190} { 0.1399, 0.2189}
FIS = 0.2417 { 0.1687, 0.3109} { 0.1685, 0.3132}
-----
```

Descrevemos, em seguida, o programa computacional desenvolvido em linguagem Visual Basic for Windows, abaixo, em português, gravado em disquete anexado à presente dissertação. O texto corresponde ao encontrado no arquivo de ajuda anexado ao programa, na versão em inglês. O código consta no **programa 20**.

HARDY-WEINBERG EQUILIBRIUM TESTING

Guia rápido de uso do programa "HARDY-WEINBERG EQUILIBRIUM TESTING"

Parte 1

O que o programa faz

O programa, desenvolvido para rodar em ambiente Windows 95 ou superior, testa estatisticamente uma amostra populacional, fornecida em termos de números observados dos diferentes genótipos, com a finalidade de verificar se as frequências genotípicas em relação a um loco autossômico qualquer, distribuem-se de acordo com o princípio de Hardy-Weinberg (Hardy, 1908; Weinberg, 1908), $P(a_i a_j) = (2 - \delta_{ij}) \cdot P(a_i) \cdot P(a_j)$, em que δ_{ij} (delta de Kronecker) é um operador lógico que toma o valor 1 se $i = j$ e 0 se $i \neq j$, ou seja, a frequência esperada de homozigotos é $P(a_i a_i) = P^2(a_i)$ e a de heterozigotos é dada por $P(a_i a_j) = 2 \cdot P(a_i) \cdot P(a_j)$. Essa propriedade é válida para populações ditas pan-míticas (de pan-mixia, termo obsoleto e erudito que significa "mistura total"), ou seja, populações nas quais os cruzamentos ocorrem ao acaso (ou, alternativamente, nas quais os indivíduos resultam de combinações ao acaso entre gametas masculinos e femininos). Apesar de as populações que se apresentam em pan-mixia exibirem frequências genotípicas

marginais exatamente nas proporções de Hardy-Weinberg (condição necessária), existe um número grande de situações de dinâmica populacional não-panmítica, em que são observadas proporções marginais $P(a_i a_i) = P^2(a_i)$ e $P(a_i a_j) = 2.P(a_i).P(a_j)$. Stark (1980, 2005) e Li (1988) listaram essas situações de não-suficiência do equilíbrio de Hardy-Weinberg. O princípio, apesar de óbvio e simples, é de uma importância teórica fundamental, porque foi a partir dele que se desenvolveu o atual complexo corpo de conhecimentos da genética de populações. Sob o ponto de vista prático, sua importância foi reavivada recentemente com o explosivo desenvolvimento da biologia molecular e a introdução de um número elevado de novos polimorfismos genéticos em análises de associação alélica, aplicações forenses em casos de exclusão e inclusão de parentesco biológico etc.

O programa, que é de uso intuitivo, executa testes conhecidos na literatura como "exatos", lançando mão de simulações executadas por programas de computação eletrônica. As simulações realizadas dessa maneira, lançam mão de números praticamente aleatórios ("números pseudo-aleatórios") gerados por programas; os métodos que usam essa estratégia são conhecidos genericamente pelo nome de "método Monte Carlo". Os testes exatos realizados pelo programa são realizados em dois moldes, equivalendo aos modelos de sorteio dos elementos de urnas de tamanho infinito (no caso da simulação com reposição) e finito (no caso de simulação sem reposição). Na simulação sem reposição, permite-se variação aleatória das frequências alélicas da amostragem testada, o que não ocorre no caso da amostragem sem reposição, na qual as frequências alélicas permanecem constantes. Por causa disso, esse último processo de

simulação assemelha-se ao que ocorre numa cadeia de processos estocásticos (cadeia de Markov). Apesar de o modelo de sorteio aleatório com reposição corresponder exatamente ao que a literatura estatística denomina de testes genéticos exatos (Fisher, 1935; Haldane, 1954), o modelo de sorteio aleatório com reposição é mais realístico, uma vez que permite variações aleatórias da frequência gênica estimada a partir da amostra que está sendo testada.

Além das simulações acima referidas, o programa realiza reamostragens com subamostras da amostra testada, apenas com a finalidade de obter intervalos de confiabilidade das frequências alélicas e genotípicas médias obtidas por simulação. Esse processo, que recebe o nome de "jackknife" (em contraste com reamostragens de mesmo tamanho descritas anteriormente e que são conhecidas pelo nome genético de "bootstrap"), permite obter limites de confiabilidade a um intervalo probabilístico qualquer (geralmente 95%), com menos viéses do que aqueles obtidos na amostra original (sem subamostragem).

O programa realiza, também, a critério do usuário, testes estatísticos tradicionais, como o do qui-quadrado, teste G ou de razão logarítmica de verossimilhanças e testes especialmente aplicáveis para amostras de tamanho reduzido, como é o caso dos métodos desenvolvidos por Hogben/Levene e Cannings e Edwards (Hogben, 1946; Levene, 1949; Cannings e Edwards, 1968) e do teste exato de Fisher. Para o caso especial de dois alelos apenas, são apresentadas também correções de continuidade (correção de Yates) para os testes de qui-quadrado, G e de Hogben/Levene.

Parte 2

Funcionamento do programa

A interface gráfica da tela inicial contém dois botões de comando. Clicando-se no botão intitulado **ABOUT**, aparecerá uma caixa de texto contendo informações e créditos sobre o programa.

O acionamento do botão **RUN PROGRAM** abre uma interface gráfica contendo uma caixa de texto, na qual o usuário deverá digitar o número total de alelos contidos em sua amostra populacional. O programa aceita apenas números inteiros entre 2 e 20. Após a digitação correta do número de alelos, o usuário deverá clicar o botão OK. A interface contém ainda mensagem explicativa e uma lista de testes estatísticos tradicionais que poderão, facultativamente, também ser executados. Precedendo os nomes que identificam os testes, existem "check boxes" (as quais aparecerão já habilitadas) para que o usuário possa escolher (com um click do mouse) quais dentre eles sejam executados ou não pelo programa.

Clicando-se o botão OK, aparecerá uma nova interface gráfica com a matriz de entrada de dados genotípicos (números observados de cada genótipo). Cada célula da matriz é identificada pelos alelos i e j correspondentes aos genótipos ordenados ij , com $i \leq j$, dispostos ortogonal e marginalmente à matriz e identificados por algarismos arábicos ($i, j = 1, 2, \dots, 20$). A matriz, que é sempre triangular superior, ou seja, contém valores não nulos válidos apenas acima da diagonal principal, que é composta sempre por $n(n+1)/2$ células, em que n é o número de alelos da amostra. Só serão aceitos números inteiros positivos [$n(i, j) \geq 0$] e o tamanho amostral máximo permitido é de 100000 indivíduos. Após a inserção correta da amostragem observada pelo

usuário, clica-se o botão OK, aparecendo em seguida ainda uma mensagem de confirmação a respeito destes dados antes do programa mostrar a próxima interface contendo os resultados.

Hardy - Weinberg Equilibrium Testing

INPUT MATRIX OF OBSERVED ABSOLUTE GENOTYPE FREQUENCIES

1	2	
1	44	90
2		54

CONFIRM DATA INPUT?

☒ YES ☐ NO

Nessa tela aparecerão campos contendo informações a respeito da evolução dos processos de simulação computacional: o número de simulações realizados até aquele instante, o tempo decorrido desde o início das simulações e uma barra gráfica indicando o percentual relativo de simulações realizadas até o momento; após o término das simulações, aparecerá um campo informando o número de ordenações de dados necessárias com a finalidade de se obterem os valores dos intervalos de confiabilidade.

Após o término dos processos de simulação, aparecerá uma caixa de texto contendo uma barra de rolamento vertical, descrevendo:

- a) os dados amostrais observados (objetos dos testes) sob a forma de frequências alélicas e genotípicas absolutas e relativas;
- b) as estimativas médias das frequências alélicas obtidas por simulação bootstrap com e sem reposição, acompanhadas dos respectivos intervalos de confiabilidade a 95%;
- c) as estimativas médias das frequências alélicas obtidas por simulação bootstrap com e sem reposição e por simulação jackknife por meio de subamostragens com tamanho correspondente a 97.5% do da amostra testada, acompanhadas dos respectivos intervalos de confiabilidade a 95%;
- d) os valores de probabilidade de todos os testes executados;
- e) a indicação do local do computador onde esse texto se encontra armazenado sob a forma de arquivo.

O conteúdo típico de um desses arquivos correspondente aos dados das páginas anteriores é mostrado abaixo:

OBSERVED DATA

GENOTYPE	OBS. NO.	FREQ.

1-1	44	0.234
1-2	90	0.479
2-2	54	0.287

TOTAL	188	1.000

ALLELE	OBS.NO.	FREQ.
1	178	0.473
2	198	0.527
TOTAL	376	1.000

COMPUTER-SIMULATED DATA

NUMBER OF SIMULATIONS = 1000

AVERAGE ALLELE FREQUENCIES WITH CORRESPONDING 95% CONFIDENCE INTERVALS
(BOOTSTRAP SIMULATIONS WITH REPLACEMENT AND 97.5% SUB-SAMPLING
JACKKNIFE)

ALLELE	WITH REPLACEMENT		97.5% SS JACKKNIFE	
	AV.FR.	95% C.I.	AV.FR.	95% C.I.
1	0.474	0.426-0.524	0.473	0.464-0.481
2	0.526	0.476-0.574	0.527	0.519-0.536

AVERAGE GENOTYPE FREQUENCIES WITH CORRESPONDING 95% CONFIDENCE INTERVALS
(BOOTSTRAP SIMULATIONS WITH AND WITHOUT REPLACEMENT AND 97.5% SUB-
SAMPLING JACKKNIFE METHOD)

GENOT.	WITH REPLACEMENT		W/OUT REPLACEMENT		97.5% SS JACKKNIFE	
	AV.FR.	95% C.I.	AV.FR.	95% C.I.	AV.FR.	95% C.I.
1-1	0.224	0.186-0.261	0.225	0.170-0.287	0.224	0.184-0.259
1-2	0.499	0.426-0.574	0.498	0.426-0.564	0.498	0.416-0.573
2-2	0.277	0.239-0.314	0.277	0.218-0.340	0.278	0.232-0.314

PROBABILITY VALUES

TEST	P

BOOTSTRAP WITHOUT REPLACEMENT	0.650
BOOTSTRAP WITH REPLACEMENT	0.574
CHI-SQUARED TEST	0.585
CHI-SQ. TEST WITH CORRECTION	0.689
HOGBEN/LEVENE METHOD	0.560
CORRECTED HOGBEN/LEVENE METHOD	0.663
G (LOG-LIKELIHOOD) TEST	0.585
CORRECTED G (LOG-LIKELIH.) TEST	0.689
CANNINGS & EDWARDS METHOD	0.561
FISHER EXACT TEST	0.661

[this text is recorded on archive c:\temp\prob01.txt]

Após o cálculo desses valores, o programa mostra um botão chamado **SHOW GRAPHS**, que, quando clicado, abre uma tela mostrando o comportamento assintótico dos valores de probabilidade para os casos com e sem reposição, em função do número acumulado de simulações.

Para o caso de dois alelos, também pode ser visualizado um gráfico, representado sob a forma de um diagrama trilinear (Otto e Benedetti, 1995), contendo:

a) a parábola representativa do conjunto de pontos nas proporções de Hardy-Weinberg $\{d = p^2, h = 2pq, r = q^2\}$;

b) as curvas que representam os limites inferior e superior a um nível aproximado de confiabilidade a 95% dessa parábola; essas curvas contêm os conjuntos de pontos $\{d' = p^2 + F_s pq, h' = 2pq(1 - F_s), r' = q^2 + F_s pq\}$ e $\{d'' = p^2 + F_i pq, h'' = 2pq(1 - F_i), r'' = q^2 + F_i pq\}$, em que F_i e F_s (com $F_s = -F_i$) são as soluções da equação

$$\chi^2_{cr} = N \cdot F^2 \text{ ou } F_{s,i} = \pm \sqrt{(\chi^2_{cr}/N)},$$

em que $\chi^2_{cr} = 3.841$ e N é o tamanho da amostra sendo testada;

c) o conjunto de pontos populacionais simulados;

- d) o ponto populacional {**d**, **h**, **r**} sob teste;
- e) a elipse que representa o intervalo aproximado de confiabilidade a 95%, sob hipótese de pan-mixia, para uma população com tamanho amostral e frequências gênicas iguais aos da amostra testada.

Um botão **END PROGRAM**, presente nas interfaces contendo a caixa de texto com os resultados e os gráficos, termina o programa e fecha as interfaces gráficas.

SUMÁRIO

Desenvolvemos no presente trabalho uma série de programas computacionais visando a estimativa de parâmetros genético-populacionais de interesse e o teste de hipóteses genéticas comumente empregadas em genética de populações, em especial a análise do equilíbrio de Hardy-Weinberg para um número qualquer de alelos autossômicos e apenas secundariamente a análise da estruturação populacional (F_{st} , F_{it} , F_{is}) para o caso de um único loco com dois alelos e um número qualquer de subpopulações.

A principal contribuição do trabalho é um programa compilado (executável) desenvolvido em linguagem Visual Basic (Microsoft, Inc.) para interface gráfica Windows 95 ou superior que analisa dados amostrais colhidos de uma população diplóide em relação a um loco autossômico no qual segrega um número qualquer de alelos. O programa verifica se as proporções genotípicas marginais de uma população estão de acordo com as esperadas segundo o modelo pan-mítico de Hardy-Weinberg $\{P(a_ia_i) = p_i^2, P(a_ia_j) = 2p_ip_j\}$ mediante a aplicação de testes estatísticos clássicos (testes do qui-quadrado e da razão de verossimilhanças, com ou sem correção de continuidade, e teste exato de Fisher) e de modificações dos mesmos apropriadas para amostras de tamanho reduzido (testes de Hogben/Levene e Cannings & Edwards). Também executa o teste exato proposto originalmente por Haldane, baseado na ordenação de todas as amostras possíveis com as mesmas frequências gênicas da amostra.

Finalmente, o programa calcula a probabilidade exata em relação à hipótese de equilíbrio de Hardy-Weinberg mediante a aplicação de

sorteios de números pseudo-aleatórios gerados por computador eletrônico (método Monte Carlo) por meio de reamostragens sem e com reposição (ou seja fixando-se ou não as frequências gênicas da amostra original). A primeira técnica corresponde virtualmente ao teste exato de Haldane, com a vantagem de poder ser aplicada tanto a amostras dialélicas numerosas quanto a amostras polialélicas de qualquer tamanho, situações que limitam drasticamente a aplicação do teste de Haldane na prática devido ao número exponencialmente grande de populações capazes de serem formadas com mesmas frequências alélicas. A segunda técnica, que é a preferida universalmente em programas como o aqui apresentado, é a mais realista, uma vez que permite a ocorrência de flutuações aleatórias nas frequências alélicas das populações simuladas em relação à amostra original. O programa fornece ainda os intervalos exatos de credibilidade a 95% das frequências alélicas e genotípicas, por meio de reamostragens empregando as técnicas de bootstrap e de jackknife.

Para o caso de dois alelos o programa mostra, num sistema de coordenadas triangulares:

o ponto amostral $P \{d, h, r\}$;

a parábola de De Finetti representativa dos conjuntos de pontos populacionais $\{p^2, 2pq, q^2\}$;

as curvas que representam os limites aproximados de confiabilidade a 95% da parábola de De Finetti, cada uma delas representando os conjuntos de pontos populacionais $\{p^2 + pqF, 2pq(1-F), q^2 + pqF\}$, com $F = F_{sup} = -\sqrt{(\chi^2/N)}$ ou $F = F_{inf} = +\sqrt{(\chi^2/N)}$ e $\chi^2 = 3.841$;

o conjunto de pontos populacionais obtidos por meio de simulações com reamostragens jackknife e a elipse correspondente que contém 95% deles,

a qual, portanto, representa os limites não-enviesados de confiabilidade a 95% dos pares ordenados $\{h, p\}$ de pontos populacionais P simulados.

Ainda para o caso de dois alelos, o programa permite a visualização de um gráfico que mostra o comportamento da convergência para o valor da probabilidade exata em função do número acumulado de simulações.

Outra contribuição do presente trabalho que consideramos importante sob o ponto de vista prático é um detalhado estudo empírico que compara criticamente os resultados obtidos mediante a aplicação de todos os métodos citados anteriormente.

ABSTRACT

In the present dissertation we developed a series of computer programs aiming at the estimation of parameters of genetic-population interest and the test of genetic hypotheses commonly used in population genetics, with emphasis on the analysis of Hardy-Weinberg equilibrium in the generalized case of any number of alleles segregating in a given autosomal locus. Other programs were developed in order to cope with the analysis of hierarchical population structure (F_{st} , F_{it} , F_{is}) for the case of a single autosomal locus with two alleles and any number of subpopulations.

The main contribution of this dissertation is a compiled (executable) program developed by means of the Visual Basic (Microsoft, Inc.) software which analyzes sample data drawn from a diploid population in relation to an autosomal locus with any number of segregating alleles. The program verifies whether the marginal population genotypic proportions follow the so-called Hardy-Weinberg proportions $\{P(aiai) = p_i^2, P(aiaj) = 2p_i p_j\}$ through the application of classic statistical tests (such as the chi-squared and log-likelihood (G) tests with and without continuity correction and Fisher's exact test) and of modifications suitable for small-sized samples (as the tests proposed by Hogben/Levene and Cannings & Edwards). It performs also the exact originally proposed by Haldane, which is based on the ordination of all possible samples with the same size and same allelic frequencies of the sample being analyzed.

Eventually, the program calculates the exact probability in relation to the hypothesis of Hardy-Weinberg equilibrium by means of the application of computer-generated pseudorandom numbers (Monte Carlo

method) through resampling without and with reposition (that is, fixing or not the allelic frequencies of the original sample). The first approach corresponds virtually to Haldane's exact test, with the advantage that it can be applied to large diallelic samples as well as to polyallelic samples of any size, situations that restrict greatly the practical use of Haldane's exact test due to the exponentially large number of possible populations with the same size and gene frequency. The second technique, which turns out to be universally preferred in programs like the one presented here, is the more realistic one, since it allows the occurrence of random fluctuations in gene frequency in the simulated samples in relation to the original one. The program gives also the exact 95% credibility intervals of allelic and genotypic frequencies, using the bootstrap and jackknife resampling techniques.

For the two-allele case the program shows, in a trilinear diagram:

- a) the sample population point $P \{d, h, r\}$;
- b) De Finetti's parabola, that represents the set of population points in Hardy-Weinberg proportions $\{p^2, 2pq, q^2\}$;
- c) the curves that represent the approximate upper and lower limits of De Finetti's parabola, each of them representing the set of population points $\{p^2+pqF, 2pq(1-F), q^2+pqf\}$, with F given by $F = F_{up} = -\sqrt{(\chi^2/N)}$ or $F = F_{lo} = +\sqrt{(\chi^2/N)}$ and $\chi^2 = 3.841$;
- d) the set of population points obtained by means of jackknife resampling simulations and the corresponding ellipse which contains 95% of these points, and, therefore, represents the 95% credibility unbiased limits of ordinated pairs of values $\{p, h\}$ of population points P simulated.

For the two-allele case, the program enables the visualization of a graph showing the convergence behavior to the exact probability value as function of the accumulated number of simulations.

Another important practical contribution of the present dissertation is a detailed empirical study that compares critically the results obtained by means of all methods mentioned above.

ANEXO

PROGRAMA 1 (VISUAL BASIC 6.0)

Public NAL, N, chisq, p99

```
Private Sub Command1_Click()
Dim F(20000), AL(2), cont(3), VM(500)
FileName1$ = "dato50.doc"
Command1.Visible = False
Open FileName1$ For Output As #1
N = 50: LIM = 2000
```

```
Print #1, "D    H    R    PEcrep    PEsrep    Pchi    Pchicor    Pli    Plicor    PG    PGCOR" &
vbCrLf
```

```
For d = 0 To N
  For h = 0 To N - d
    r = N - d - h
    If d <> N And r <> N Then
For i = 1 To 2
  cont(i) = 0
Next i
p = (2 * d + h)
```

```
For i = 1 To 2 * N
  VM(i) = 0
Next i
```

```
For i = 1 To p
  VM(i) = 1
Next i
```

```
For i = 1 To 2 * N
  F(i) = F(i - 1) + Log(i)
Next i
Prob = 0
TI = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TI = TI + Val(Mid$(Time$, 7, 2))
F1 = F(N)
F2 = F(2 * N)
F3 = F(p) + F(2 * N - p): Prob = Prob + F3
F4 = F(d) + F(h) + F(r): Prob = Prob - F4
Prob = Prob + F1 - F2 + h * Log(2)
p = p / (2 * N)
NS = 0
While NS < LIM
  NS = NS + 1
simd = 0: simh = 0: simr = 0
simd1 = 0: simh1 = 0: simr1 = 0
  For k = 1 To N
    A = Rnd: B = Rnd
    c = Int(A * (2 * N - 2 * (k - 1) - 1) + 1): al1 = VM(c)
    aux = VM(c): VM(c) = VM(2 * N - 2 * (k - 1)): VM(2 * N - 2 * (k - 1))
= aux
    e = Int(B * (2 * N - 2 * (k - 1) - 2) + 1): al2 = VM(e)
    If A <= p And B <= p Then
      simd = simd + 1
    Else
      If A > p And B > p Then simr = simr + 1 Else simh = simh + 1
    End If
    If al1 <> 0 And al2 <> 0 Then
      simd1 = simd1 + 1
```

```

Else
  If al1 = 0 And al2 = 0 Then
    simr1 = simr1 + 1
  Else
    simh1 = simh1 + 1
  End If
End If
aux = VM(e): VM(e) = VM(2 * N - 2 * (k - 1) - 1): VM(2 * N - 2 * (k - 1) - 1) = aux
Next k
simp1 = (2 * simd + simh)
simp2 = (2 * simd1 + simh1)

For i = 1 To 2
  If i = 1 Then
    simp = simp1: simda = simd: simha = simh: simra = simr
  Else
    simp = simp2: simda = simd1: simha = simh1: simra = simr1
  End If
  PR = 0
  F3 = F(simp) + F(2 * N - simp): PR = PR + F3
  F4 = F(simda) + F(simha) + F(simra): PR = PR - F4
  PR = PR + F1 - F2 + simha * Log(2)
  If PR <= Prob Then cont(i) = cont(i) + 1
Next i
Wend
chisq = d ^ 2 / (N * p ^ 2) + h ^ 2 / (2 * N * p * (1 - p)) + r ^ 2 / (N * (1 - p) ^ 2) -
N
A = CHISQUAREDIST()
pchi = p99
chisq = (Abs(4 * d * r - h ^ 2) - 2 * N) ^ 2
chisq = chisq / ((2 * d + h) * (h + 2 * r)) ^ 2 * N
A = CHISQUAREDIST()
pchicor = p99
li = 0: licor = 0: pli = 0: plicor = 0
If (2 * d + h - 1) And (2 * r + h - 1) <> 0 Then
  li = (N - 1) * ((2 * d + h) * (2 * r + h) - (2 * N - 1) * h) ^ 2
  li = li / ((2 * d + h) * (2 * d + h - 1) * (2 * r + h) * (2 * r + h - 1))
  chisq = li
  A = CHISQUAREDIST()
  pli = p99
  licor = (N - 1) * (Abs((2 * d + h) * (2 * r + h) - (2 * N - 1) * h) - 4 * N * 0.5) ^ 2
  licor = licor / ((2 * d + h) * (2 * d + h - 1) * (2 * r + h) * (2 * r + h - 1))
  chisq = licor
  A = CHISQUAREDIST()
  plicor = p99
End If

g = 0: g1 = 0
If d <> 0 Then g = d * Log(d)
If h <> 0 Then g = g + h * Log(h)
If r <> 0 Then g = g + r * Log(r)
g = g - h * Log(2) - 2 * N * (p * Log(p) + (1 - p) * Log((1 - p))) - N * Log(N)
g = 2 * g
A = Exp(g): pg = 2 / (1 + A)
If d * r < h ^ 2 / 4 Then
  dd = d + 0.5: hh = h / 2 - 0.5: rr = r + 0.5
Else
  dd = d - 0.5: hh = h / 2 + 0.5: rr = r - 0.5
End If
If dd <> 0 Then g1 = dd * Log(dd)
If hh <> 0 Then g1 = g1 + 2 * hh * Log(hh)
If rr <> 0 Then g1 = g1 + rr * Log(rr)
g1 = g1 - 2 * N * (p * Log(p) + (1 - p) * Log((1 - p))) - N * Log(N)
g1 = 2 * g1
A = Exp(g1): pg1 = 2 / (1 + A)
DS = Format(d, "00"): HS = Format(h, "00"): RS = Format(r, "00")
p1 = Format(cont(1) / LIM, "0.0000")
p8 = Format(cont(2) / LIM, "0.0000")
p2 = Format(pchi, "0.0000")
p3 = Format(pchicor, "0.0000")

```

```

p4 = Format(pg, "0.0000")
p5 = Format(pg1, "0.0000")
p6 = Format(pli, "0.0000")
p7 = Format(plicor, "0.0000")
Print #1, DS, HS, RS, p1, p8, p2, p3, p6, p7, p4, p5, vbCrLf
Text1.Text = DS & " " & h & " " & r
Form1.Refresh
End If
Next h, d
Close #1
End Sub

Sub GRID1_KEYPRESS(A As Integer)
MSHFLEXGRIDEDIT GRID1, Text2, A
End Sub
Sub GRID1_DBLCLICK()
MSHFLEXGRIDEDIT GRID1, Text2, 32
End Sub
Sub MSHFLEXGRIDEDIT(MSHFLEXGRID As Control, _
EDT As Control, A As Integer)

If MSHFLEXGRID.Col < MSHFLEXGRID.Row Then Exit Sub

If MSHFLEXGRID.Row = NAL + 1 Or MSHFLEXGRID.Col = NAL + 1 _
Then Exit Sub

Select Case A
Case 0 To 32
EDT = MSHFLEXGRID
EDT.SelStart = 1000

Case Else
EDT = Chr(A)
EDT.SelStart = 1
End Select

EDT.Move MSHFLEXGRID.Left + MSHFLEXGRID.CellLeft, _
MSHFLEXGRID.Top + MSHFLEXGRID.CellTop, _
MSHFLEXGRID.CellWidth - 8, _
MSHFLEXGRID.CellHeight - 8
EDT.Visible = True

EDT.SetFocus
End Sub
Sub GRID1_GOTFOCUS()
If Text2.Visible = False Then Exit Sub
GRID1 = Text2
Text2.Visible = False
End Sub
Sub GRID1_LEAVECELL()
If Text2.Visible = False Then Exit Sub
GRID1 = Text2
Text2.Visible = False
End Sub

Function fgi(r, s)
fgi = s + GRID1.Cols * r
End Function

Function CHISQUAREDIST()
R99 = 1: V99 = 1: W99 = chisq
K99 = W99 ^ (Int((V99 + 1) / 2)) * Exp(-W99 / 2) / R99
If Int(V99 / 2) = V99 / 2 Then GoTo Label1
If W99 > 0 Then J99 = Sqr(2 / W99 / 3.141592653599): GoTo Label2
Label1: J99 = 1
Label2: L99 = 1: M99 = 1
CHILoop: V99 = V99 + 2: M99 = M99 * W99 / V99: If M99 < 0.0000001 Then p99 = 1 - J99 * K99
* L99: GoTo end1
L99 = L99 + M99: GoTo CHILoop
end1:

```

End Function

PROGRAMA 2 (QBASIC 1.1)

```

REM PROGRAM FILENAME HWEXAC01.BAS
CLS : DEFDBL A-Z
INPUT "D, H, R = ", D, H, R
N = D + H + R: P = 2 * D + H: Q = H + 2 * R
DIM F(2 * N), D3(4), H3(4), R3(4), P99(4)
FOR I = 1 TO 2 * N: F(I) = F(I - 1) + LOG(I): NEXT I
IF P > Q THEN MAXPQ = P ELSE MAXPQ = Q
SIZE = N / 2 + 1: DIM D1(SIZE), H1(SIZE), R1(SIZE), PROB1(SIZE)
IF INT(H / 2) = H / 2 THEN HMIN = 0 ELSE HMIN = 1
IF D > R THEN HMAX = Q ELSE HMAX = P
I = 0: FOR H9 = HMIN TO HMAX STEP 2: I = I + 1
H1 = H9: D1 = (P - H1) / 2: R1 = (Q - H1) / 2
PROB1(I) = F(N) - F(2 * N) + F(P) - F(D1) + F(Q) - F(R1)
PROB1(I) = PROB1(I) + H1 * LOG(2) - F(H1)
PROB1(I) = 100000 * INT(100000000000# * EXP(PROB1(I))) + H1
NEXT H9: NEWSIZE = I: GOSUB QUICKSORT
PRINT "-----"
PRINT "  D    H    R    Prob.Cum. Dsq.      Chi-squared      P(Chi - squared)"
PRINT "          Prob.          1      2      3      4      1      2      3      4"
PRINT "-----"
FOR I = 1 TO NEWSIZE
PROB2 = PROB1(I) / 10 ^ 15: PROBT = PROBT + PROB2
H2 = INT(PROB1(I) - INT(PROB1(I) / 10 ^ 5) * 10 ^ 5)
D2 = (P - H2) / 2: R2 = (Q - H2) / 2
X = D2 + H2 / 2: IF X = 0 THEN X = .00000000001#
Y = H2 / 2 + R2: IF Y = 0 THEN Y = .00000000001#
P1 = X / N: Q1 = 1 - P: SEPQ = SQR((P * Q) / (2 * N))
C(1) = (H2 ^ 2 - 4 * D2 * R2) ^ 2 * N / (P ^ 2 * Q ^ 2)
C(2) = (ABS(4 * D2 * R2 - H2 ^ 2) - 2 * N) ^ 2
C(2) = C(2) / ((2 * D2 + H2) * (H2 + 2 * R2)) ^ 2 * N
D3(3) = X * (2 * X - 1) / (2 * N - 1)
H3(3) = 4 * X * Y / (2 * N - 1)
R3(3) = Y * (2 * Y - 1) / (2 * N - 1)
D3(4) = X ^ 2 / N - H2 / (4 * N)
H3(4) = 2 * X * Y / N + H2 / (2 * N)
R3(4) = Y ^ 2 / N - H2 / (4 * N)
FOR J = 3 TO 4
C(J) = D2 ^ 2 / D3(J) + H2 ^ 2 / H3(J) + R2 ^ 2 / R3(J) - N
NEXT J
DISEQ = D2 / N - (P / (2 * N)) ^ 2
FOR J = 1 TO 4
CHISQ = C(J)
IF C(J) = 0 THEN P99(J) = 1: GOTO NEXTJ
GOSUB CHISQUAREDIST: P99(J) = 1 - J99 * K99 * L99
NEXTJ: NEXT J
PRINT USING "####"; D2; H2; R2;
IF H2 = H THEN PRINT " *"; ELSE PRINT " ";
PRINT USING " ###"; PROB2;
PRINT USING " ###"; PROBT;
PRINT USING " ###"; DISEQ;
FOR J = 1 TO 4
PRINT USING "####.###"; C(J);
NEXT J
FOR J = 1 TO 4
PRINT USING " ###"; P99(J);
NEXT J
PRINT
NEXT I
PRINT "-----"
END
QUICKSORT:
D99 = 10: K = 1: L = 1: R99 = NEWSIZE
560 REM EXCHANGE SORT IF BLOCKSIZE <= D99
IF R99 - L + 1 > D99 THEN 590
GOSUB 900: GOTO 780
590 I = L: J = R99: W = RND(1) * (R99 - L) + .5
W = INT(W + L): F = PROB1(W)
REM PARTITION PHASE

```

```

620 IF PROB1(I) >= F THEN 640
I = I + 1: GOTO 620
640 IF PROB1(J) <= F THEN 660
J = J - 1: GOTO 640
660 IF I > J THEN 680
W = PROB1(I): PROB1(I) = PROB1(J): PROB1(J) = W: I = I + 1
J = J - 1
680 IF I <= J THEN 620
REM BOOKKEEPING PHASE
IF J - L >= R99 - I THEN 740
IF I >= R99 THEN 730
H99(K) = I: K = K + 1: H99(K) = R99: K = K + 1
730 R99 = J: GOTO 770
740 IF L >= J THEN 760
H99(K) = L: K = K + 1: H99(K) = J: K = K + 1
760 L = I
770 IF L < R99 THEN 590
780 IF K <= 1 THEN 800
K = K - 1: R99 = H99(K): K = K - 1: L = H99(K): GOTO 560
800 RETURN
END
900 REM EXCHANGE SORT
FOR I = L TO R99 - 1: K1 = I
FOR J = I + 1 TO R99: IF PROB1(J) >= PROB1(K1) THEN 940
K1 = J
940 NEXT J
IF K1 = I THEN 970
W = PROB1(I): PROB1(I) = PROB1(K1): PROB1(K1) = W
970 NEXT I
RETURN
CHISQUAREDIST:
R99 = 1: V99 = 1: W99 = CHISQ
K99 = W99 ^ (INT((V99 + 1) / 2)) * EXP(-W99 / 2) / R99
IF INT(V99 / 2) = V99 / 2 THEN GOTO LABEL1
J99 = SQR(2 / W99 / 3.141592653599#): GOTO LABEL2
LABEL1: J99 = 1
LABEL2: L99 = 1: M99 = 1
CHILOOP: V99 = V99 + 2: M99 = M99 * W99 / V99: IF M99 < .0000001 THEN RETURN
L99 = L99 + M99: GOTO CHILOOP
RETURN

```

LISTAGEM 3 (VISUAL BASIC 6.0):

```

REM program filename samegen4.bas
REM the program generates all possible genotype combinations with
REM the same allele frequencies in the four-allele case
CLS : DEFINT A-D, N
INPUT "a,b,c,d = "; a, b, c, d
timewas = VAL(MID$(TIME$, 1, 2)) * 3600 + VAL(MID$(TIME$, 4, 2)) * 60
timewas = timewas + VAL(MID$(TIME$, 7, 2))
n = (a + b + c + d) / 2
PRINT "aa ab ac ad bb bc bd cc cd dd  a   b   c   d   npop"
PRINT "-----"
FOR aa = 0 TO n
FOR ab = 0 TO n: IF aa + ab > n THEN GOTO nextdd
FOR ac = 0 TO n: IF aa + ab + ac > n THEN GOTO nextdd
FOR ad = 0 TO n: IF aa + ab + ac + ad > n THEN GOTO nextdd
FOR bb = 0 TO n: IF aa + ab + ac + ad + bb > n THEN GOTO nextdd
FOR bc = 0 TO n: IF aa + ab + ac + ad + bb + bc > n THEN GOTO nextdd
FOR bd = 0 TO n: IF aa + ab + ac + ad + bb + bc + bd > n THEN GOTO nextdd
FOR cc = 0 TO n: IF aa + ab + ac + ad + bb + bc + bd + cc > n THEN GOTO nextdd
FOR cd = 0 TO n: IF aa + ab + ac + ad + bb + bc + bd + cc + cd > n THEN GOTO nextdd
FOR dd = 0 TO n: IF aa + ab + ac + ad + bb + bc + bd + cc + cd + dd > n THEN GOTO nextdd
IF 2 * aa + ab + ac + ad = a AND ab + 2 * bb + bc + bd = b AND ac + bc + 2 * cc + cd = c
AND ad + bd + cd + 2 * dd = d THEN
    np = np + 1
    PRINT USING "###"; aa; ab; ac; ad; bb; bc; bd; cc; cd; dd;
    PRINT USING "####"; a; b; c; d;
    PRINT USING "#####"; np
    DO: LOOP WHILE INKEY$ <> " "
END IF
IF aa = a / 2 AND bb = b / 2 AND cc = c / 2 AND dd = d / 2 THEN GOTO endnow
nextdd: NEXT dd, cd, cc, bd, bc, bb, ad, ac, ab, aa
endnow:
PRINT "-----"
timeis = VAL(MID$(TIME$, 1, 2)) * 3600
timeis = timeis + VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2))
PRINT USING "timeis = #####.## seconds"; timeis
PRINT USING "timewas = #####.## seconds"; timewas
PRINT USING "#####.## seconds"; timeis - timewas

```

LISTAGEM 4 E 5

```

REM PROGRAM FILENAME POSSPOP1.BAS (QBASIC 1.1)
REM maximum number of possible populations with same gene frequencies
REM and with the same size, for the cases of 2, 3, and 4 alleles
CLS
INPUT "FILE NAME FOR STORING DATA (DRIVE:FILENAME.EXT) = "; FILENAME$
OPEN FILENAME$ FOR OUTPUT AS #1
PRINT #1, " N                NPOP"
PRINT #1, "-----"
FOR N1 = 1 TO 10
N2 = N1: N3 = N2: N4 = N3
N = (N1 + N2 + N3 + N4) / 2
IF INT(N) = N THEN
DD = 0
FOR I = 0 TO N1 / 2: FOR J = 0 TO N2 / 2
FOR K = 0 TO N3 / 2: FOR M = 0 TO N4 / 2
LOCATE 15, 15: PRINT USING " ## ## ## ## ## #####"; N1; I; J; K; M; DD
FOR T = 0 TO N1 - 2 * I: FOR U = 0 TO N1 - 2 * I - T
R = N1 - 2 * I - T - U
FOR W = 0 TO N2 - 2 * J - T
G = N2 - 2 * J - T - W
IF N3 - 2 * K - U - W >= 0 THEN
S = N3 - 2 * K - U - W
NAC = I + J + K + M + T + U + R + W + G + S
IF 2 * M + R + G + S = N4 AND N = NAC THEN DD = DD + 1
END IF
NEXT W, U, T, M, K, J, I
PRINT #1, N1, N2, N3, N4, N5, N6 DD
END IF
NEXT N1
PRINT #1, "-----"
CLOSE #1

REM PROGRAM FILENAME POSSPOP2.BAS (VISUAL BASIC 6.0)
REM maximum number of possible populations with same gene frequencies
REM and with the same size, for the cases of 5 and 6 alleles
CLS
NMAX = 12
NAL = 5
Open "result08.doc" For Output As #1
For N1 = 12 To NMAX
N2 = N1: N3 = N1: N4 = N1: N5 = N1
TI = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TI = TI + Val(Mid$(Time$, 7, 2))
N = (N1 + N2 + N3 + N4 + N5 + N6) / 2
If N = Int(N) Then
DD = 0
For I = 0 To N1 / 2: For J = 0 To N2 / 2: For K = 0 To N3 / 2
For M = 0 To N4 / 2: For H = 0 To N5 / 2: For Q = 0 To N6 / 2
For T = 0 To N1 - 2 * I: For U = 0 To N1 - 2 * I - T
For R = 0 To N1 - 2 * I - T - U: For RR = 0 To N1 - 2 * I - T - U - R
RRR = N1 - 2 * I - T - U - R - RR
For W = 0 To N2 - 2 * J - T: For G = 0 To N2 - 2 * J - T - W
For GG = 0 To N2 - 2 * J - T - W - G
GGG = N2 - 2 * J - T - W - G - GG
For PP = 0 To N3 - 2 * K - U - W: For PPP = 0 To N3 - 2 * K - U - W - PP
PPPP = N3 - 2 * K - U - W - PP - PPP
For S = 0 To N4 - 2 * M - R - G - PP
SS = N4 - 2 * M - R - G - PP - S
If N5 - 2 * H - RR - GG - PPP - S >= 0 Then
ZZ = N5 - 2 * H - RR - GG - PPP - S
NAC = I + J + K + M + H + Q + T + U + R
NAC = NAC + RR + RRR + W + G + GG + GGG
NAC = NAC + PP + PPP + PPPP + S + SS + ZZ
If (2 * Q + RRR + GGG + PPPP + SS + ZZ = N6 And N = NAC) Then DD= DD +1
End If
Next S, PPP, PP, GG, G, W, RR, R, U, T, Q, H, M, K, J, I

```



```

TF = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TF = TF + Val(Mid$(Time$, 7, 2))
TT = TF - TI
CLMAX(N) = DD
Temp(N) = TT
If N <> 0 Then Print #1, N1,N2,N3,N4,N5,N6,DD
End If
Next N1
Close #1

```

LISTAGEM 6 (QBASIC 1.1):

```

REM PROGRAM FILENAME H_W_EXT7.BAS
DEFDBL A-Z: CLS
INPUT "D, H, R = "; D, H, R: N = D + H + R
INPUT "NUMBER OF SIMULATIONS = "; T
CHDIR "C:\TEMP": FILES "*.DAT"
INPUT "FILE NAME FOR STORING DATA (DRIVE:FILENAME.EXT) = "; FILENAME$
CLS : OPEN FILENAME$ FOR OUTPUT AS #1
PRINT #1, FILENAME$
P = 2 * D + H: Q = H + 2 * R: P0 = P / (2 * N): P1 = P: Q1 = Q
PRINT #1, D, H, R, N, P0, T
TIMEWA$ = TIME$
TIMEWAS = VAL(MID$(TIMEWA$, 1, 2)) * 3600 + VAL(MID$(TIMEWA$, 4, 2)) * 60
TIMEWAS = TIMEWAS + VAL(MID$(TIMEWA$, 7, 2))
D1 = D: H1 = H: R1 = R
K = N: GOSUB FACTORIAL: NFAC = FACT
K = 2 * N: GOSUB FACTORIAL: N2FAC = FACT
CONSTFAC = NFAC - N2FAC: D1 = D: H1 = H: R1 = R: GOSUB FACTPQDHR
PROB2 = EXP(CONSTFAC + PFAC + QFAC - DFAC - HFAC - RFAC + H1 * LOG(2))
FOR I = 1 TO T
    R1 = 0: D1 = 0: H1 = 0
    FOR INDIV = 1 TO N
        A = RND(1): B = RND(1)
        IF A < P0 AND B < P0 THEN D1 = D1 + 1: GOTO NEXTINDIV
        IF A >= P0 AND B >= P0 THEN R1 = R1 + 1: GOTO NEXTINDIV
        H1 = H1 + 1
    NEXTINDIV: NEXT INDIV
    P1 = 2 * D1 + H1: Q1 = H1 + 2 * R1
    D1TOT = D1TOT + D1: H1TOT = H1TOT + H1: R1TOT = R1TOT + R1
    PTOT = PTOT + P1 / (2 * N)
    GOSUB FACTPQDHR
    PROB1 = EXP(CONSTFAC + PFAC + QFAC - DFAC - HFAC - RFAC + H1 * LOG(2))
    IF PROB1 <= PROB2 THEN PROBT = PROBT + 1
    PRINT #1, P1, H1
    LOCATE 6: PRINT USING "PERFORMING SIMULATION NO. ##### "; I;
    PRINT USING "OUT OF #####"; T
NEXT I
CLOSE #1
TIMENO$ = TIME$
TIMENOW = VAL(MID$(TIMENO$, 1, 2)) * 3600 + VAL(MID$(TIMENO$, 4, 2)) * 60
TIMENOW = TIMENOW + VAL(MID$(TIMENO$, 7, 2))
CLS : PRINT "NO. OF SIMULATED SAMPLES OF SIZE "; N; " = "; T
PRINT "TOTAL ELAPSED TIME (ALL CALCULATIONS) = "; TIMENOW - TIMEWAS; " SEC."
PRINT "DATA FROM DATAFILE "; FILENAME$: PRINT
PRINT USING "SAMPLE P      =      #.#####"; P0
PRINT USING "SAMPLE D      = #####.#####"; D
PRINT USING "SAMPLE H      = #####.#####"; H
PRINT USING "SAMPLE R      = #####.#####"; R: PRINT
PRINT USING "P MEAN        =      #.#####"; PTOT / T
PRINT USING "OBS.D MEAN    = #####.#####"; D1TOT / T
PRINT USING "OBS.H MEAN    = #####.#####"; H1TOT / T
PRINT USING "OBS.R MEAN    = #####.#####"; R1TOT / T: PRINT
PRINT USING "EX. PROB.     =      #.#####"; PROBT / T
DO: LOOP WHILE INKEY$ <> " "
END
FACTPQDHR: K = P1: GOSUB FACTORIAL: PFAC = FACT
          K = Q1: GOSUB FACTORIAL: QFAC = FACT
          K = D1: GOSUB FACTORIAL: DFAC = FACT
          K = H1: GOSUB FACTORIAL: HFAC = FACT
          K = R1: GOSUB FACTORIAL: RFAC = FACT: RETURN
FACTORIAL: FACT = 0: FOR J = 1 TO K: FACT = FACT + LOG(J): NEXT J: RETURN

```

LISTAGEM 7 (QBASIC 1.1):

```

REM PROGRAM FILENAME HW_EX11A.BAS
REM 'EXACT' HARDY-WEINBERG TEST FOR ANY NUMBER OF AUTOSOMAL ALLELES
CLS : DEFDBL A-Z: INPUT "NUMBER OF ALLELES = "; K
DIM P(K), SP(K), GENOTYPE(K, K), TGENOTYP(K, K), ALLELE(K), TALLEL(K)
INPUT "NUMBER OF SIMULATIONS = "; T
FOR I = 1 TO K: FOR J = I TO K
PRINT USING "N(###"; I; : PRINT ", "; : PRINT USING "###) = "; J;
INPUT GENOTYPE(I, J): N = N + GENOTYPE(I, J)
ALLELE(I) = ALLELE(I) + GENOTYPE(I, J)
ALLELE(J) = ALLELE(J) + GENOTYPE(I, J): NEXT J, I: CLS
FOR I = 1 TO K: P(I) = ALLELE(I) / (2 * N): NEXT I
FOR I = 1 TO K: PRINT USING "ALLELE(###) = "; I;
PRINT USING " #####"; ALLELE(I)
FOR J = I TO K: PRINT USING "GENOT(###"; I; : PRINT ", ";
PRINT USING "###) = "; J; : PRINT USING "#####"; GENOTYPE(I, J)
'DO: LOOP WHILE INKEY$ <> " "
NEXT J, I
PRINT USING "N = #####"; N
DO: LOOP WHILE INKEY$ <> " "
CLS
TIMEWA$ = TIME$
TIMEWAS = VAL(MID$(TIMEWA$, 1, 2)) * 3600 + VAL(MID$(TIMEWA$, 4, 2)) * 60
TIMEWAS = TIMEWAS + VAL(MID$(TIMEWA$, 7, 2))
K9 = N: GOSUB FACTORIAL: NFAC = FACT
K9 = 2 * N: GOSUB FACTORIAL: N2FAC = FACT
CONSTFAC = NFAC - N2FAC
GOSUB EXACTPROBCALC: PROB2 = EXPROB
FOR I = 1 TO K: SSP = SSP + P(I): SP(I) = SP(I) + SSP: NEXT I
FOR I = 1 TO T
GOSUB CLEARALLVAR
FOR INDIV = 1 TO N
A = RND(1): B = RND(1)
FOR I88 = 1 TO K
IF A > SP(I88 - 1) AND A <= SP(I88) THEN
AI88 = I88: ALLELE(I88) = ALLELE(I88) + 1
TALLEL(I88) = TALLEL(I88) + 1
END IF
IF B > SP(I88 - 1) AND B <= SP(I88) THEN
BI88 = I88: ALLELE(I88) = ALLELE(I88) + 1
TALLEL(I88) = TALLEL(I88) + 1
END IF
NEXT I88
IF AI88 > BI88 THEN SWAP AI88, BI88
GENOTYPE(AI88, BI88) = GENOTYPE(AI88, BI88) + 1
TGENOTYP(AI88, BI88) = TGENOTYP(AI88, BI88) + 1
NEXT INDIV
GOSUB EXACTPROBCALC: PROB1 = EXPROB
IF PROB1 <= PROB2 THEN PROBT = PROBT + 1
LOCATE 10: PRINT "SIMULATION NO. "; I
NEXT I
'DO: LOOP WHILE INKEY$ <> " "
CLS
TIMENO$ = TIME$
TIMENOW = VAL(MID$(TIMENO$, 1, 2)) * 3600 + VAL(MID$(TIMENO$, 4, 2)) * 60
TIMENOW = TIMENOW + VAL(MID$(TIMENO$, 7, 2))
PRINT "OVERALL RESULTS BASED ON "; T; " SIMULATIONS OF SIZE "; N; " INDIVIDUALS"
PRINT "TOTAL ELAPSED TIME (ALL CALCULATIONS) = "; TIMENOW - TIMEWAS; " SEC."
FOR I77 = 1 TO K
PRINT USING "ALLELE(###) = "; I77; : PRINT USING "#####.#####"; TALLEL(I77) / T
FOR J77 = I77 TO K
PRINT USING "GENOTYPE(###"; I77; : PRINT ", ";
PRINT USING "###) = "; J77; : PRINT USING "#####.#####"; TGENOTYP(I77, J77) / T
'DO: LOOP WHILE INKEY$ <> " "
NEXT J77, I77
PRINT : PRINT USING "EXACT PROB. = #####"; PROBT / T
END
CLEARALLVAR:

```

```

HETLOG = 0: ALLELFAC = 0: GENOTFAC = 0
FOR I66 = 1 TO K
  ALLELE(I66) = 0
  FOR J66 = I66 TO K: GENOTYPE(I66, J66) = 0
NEXT J66, I66
RETURN
EXACTPROBCALC:
GOSUB FACTALLEL: GOSUB FACTGENOT
FOR I9 = 1 TO K - 1: FOR J9 = I9 + 1 TO K
  IF I9 <> J9 THEN HETLOG = HETLOG + GENOTYPE(I9, J9) * LOG(2)
NEXT J9, I9
EXPROB = EXP(CONSTFAC + ALLELFAC - GENOTFAC + HETLOG)
RETURN

FACTALLEL:
FOR I9 = 1 TO K
  K9 = ALLELE(I9): GOSUB FACTORIAL: ALLELFAC = ALLELFAC + FACT
NEXT I9
RETURN

FACTGENOT:
FOR I9 = 1 TO K: FOR J9 = I9 TO K
  K9 = GENOTYPE(I9, J9): GOSUB FACTORIAL: GENOTFAC = GENOTFAC + FACT
NEXT J9, I9
RETURN

FACTORIAL:
FACT = 0
FOR J = 1 TO K9
  FACT = FACT + LOG(J)
NEXT J
RETURN

```

LISTAGEM 8 (QBASIC 1.1):

```

REM PROGRAM FILENAME H_W_EX10.BAS
REM GENERATES T RANDOM SAMPLES OF SIZE N = D + H + R
REM WITH GENOTYPE PROBABILITIES d = D/N, h = H/N, r = R/N
DEFDBL A-Z: CLS
INPUT "D, H, R = "; D, H, R: N = D + H + R
INPUT "NUMBER OF SIMULATIONS = "; T
CHDIR "C:\TEMP": FILES "*.DAT"
INPUT "FILE NAME FOR STORING DATA (DRIVE:FILENAME.EXT) = "; FILENAME$
OPEN FILENAME$ FOR OUTPUT AS #1
PRINT #1, FILENAME$
P = 2 * D + H: Q = H + 2 * R: P0 = P / (2 * N): P1 = P: Q1 = Q
PRINT #1, D, H, R, N, P0, T
P(1) = D / N: P(2) = H / N: P(3) = R / N
FOR I = 1 TO 3: SSP = SSP + P(I): SP(I) = SP(I) + SSP: NEXT I
CLS
FOR I = 1 TO T
  D1 = 0: H1 = 0: R1 = 0
  FOR INDIV = 1 TO N
    A = RND(1)
    IF A <= SP(1) THEN D1 = D1 + 1
    IF A > SP(1) AND A <= SP(2) THEN H1 = H1 + 1
    IF A > SP(2) AND A <= SP(3) THEN R1 = R1 + 1
  NEXT INDIV
  P1 = 2 * D1 + H1: Q1 = H1 + 2 * R1
  D1TOT = D1TOT + D1: H1TOT = H1TOT + H1: R1TOT = R1TOT + R1
  PTOT = PTOT + P1
  PRINT #1, P1, H1
LOCATE 6: PRINT USING "PERFORMING SIMULATION NO. ##### "; I;
  PRINT USING "OUT OF #####"; T
NEXT I
CLOSE #1
PRINT : PRINT "DATA FROM DATAFILE "; FILENAME$: PRINT
PRINT USING "SAMPLE P      =      #.#####"; P0
PRINT USING "SAMPLE D      = #####.#####"; D
PRINT USING "SAMPLE H      = #####.#####"; H
PRINT USING "SAMPLE R      = #####.#####"; R: PRINT
PRINT USING "P MEAN        =      #.#####"; PTOT / (2 * T * N)
PRINT USING "OBS.D MEAN    = #####.#####"; D1TOT / T
PRINT USING "OBS.H MEAN    = #####.#####"; H1TOT / T
PRINT USING "OBS.R MEAN    = #####.#####"; R1TOT / T: PRINT

```

LISTAGEM 9 (QBASIC 1.1):

```
REM PROGRAM FILENAME GRAPHT02.BAS
REM ROUTINE FOR DISPLAYING ISOSCELES TRIANGLE COORDINATES
REM EQUALLY SPACED TICKS ON X AND Y AXES
REM PLOTTING OF HW POPULATIONS GENERATED BY EXACT TEST
DEFDBL A-Z: CLS
CHDIR "C:\TEMP": FILES "*.DAT"
INPUT "FILE NAME (DRIVE:FILENAME.EXT):", FILENAME$
OPEN FILENAME$ FOR INPUT AS #1
INPUT #1, A$
INPUT #1, D99, H99, R99, N99, P99, K99
XMIN = 0: XMAX = 1
YMIN = 0: YMAX = 1
SCREEN 12: REM VGA GRAPHICS MODE RESOLUTION 480 x 640
REM SMALL TRIANGLE WITH EXPLANATIONS
LINE (440, 200)-(520, 40)
LINE -(600, 200)
LINE (440, 200)-(467, 200)
LINE (484, 200)-(538, 200)
LINE (556, 200)-(600, 200)
LINE (506, 136)-(506, 155)
LINE (506, 179)-(506, 200)
LINE (472, 136)-(482, 136)
LINE (499, 136)-(530, 136)
LINE (547, 136)-(568, 136)
CIRCLE (506, 136), 3
LOCATE 9, 62: PRINT "d"
LOCATE 9, 68: PRINT "r"
LOCATE 11, 64: PRINT "h"
LOCATE 13, 60: PRINT "P"
LOCATE 13, 69: PRINT "q"
REM GRAPH AXES
LINE (130, 40)-(130, 400)
LINE (140, 410)-(500, 410)
REM SIDES OF ISOSCELES TRIANGLE
LINE (140, 400)-(320, 40)
LINE -(500, 400)
LINE -(140, 400)
REM Y-AXIS TICKS
FOR I = 40 TO 400 STEP 18
    LINE (126, I)-(130, I)
NEXT I
REM X-AXIS TICKS
FOR I = 140 TO 500 STEP 18
    LINE (I, 410)-(I, 414)
NEXT I
FOR X = XMIN TO XMAX STEP 1 / 400
    Y = 2 * X * (1 - X)
    GOSUB PLOTXY
    IF X = XMIN THEN PSET (XSCREEN, YSCREEN) ELSE LINE -(XSCREEN, YSCREEN)
NEXT X
X = XMAX
Y = 2 * X * (1 - X)
GOSUB PLOTXY
LINE -(XSCREEN, YSCREEN)
FOR I99 = 1 TO K99
    INPUT #1, X: X = X / (2 * N99)
    INPUT #1, Y: Y = Y / N99
    GOSUB PLOTXY
    CIRCLE (XSCREEN, YSCREEN), 1
NEXT I99
CLOSE #1
LOCATE 2, (81 - LEN(A$)) / 2: PRINT A$
LOCATE 3, 9: PRINT USING "####.##"; YMAX
'LOCATE 14, 15: PRINT "h"
```

```

LOCATE 26, 9: PRINT USING "####.##"; YMIN
LOCATE 27, 15: PRINT USING "###.##"; XMIN
LOCATE 27, 60: PRINT USING "###.##"; XMAX
'LOCATE 27, 40: PRINT "p"
STAYHERE:
    REM PRESS THE SPACE BAR TO RETURN TO QBASIC
    IF INKEY$ <> " " THEN GOTO STAYHERE
    SCREEN 0, 0, 0
END
PLOTXY:
    XSCREEN = 140 + 360 * (X - XMIN) / (XMAX - XMIN)
    YSCREEN = 400 - 360 * (Y - YMIN) / (YMAX - YMIN)
RETURN

```

LISTAGEM 10 (QBASIC 1.1):

```

REM PROGRAM FILENAME H_W_EX13.BAS
REM 'EXACT' HARDY-WEINBERG TEST (TWO AUTOSOMAL ALLELES)
REM GENERATES T RANDOM SAMPLES OF SIZE N = D + H + R
REM WITH GENOTYPE PROBABILITIES d = p^2, h = 2pq, r = q^2
REM p = (2D+H)/2, q = 1-p
DEFDBL A-Z: CLS
INPUT "D, H, R = "; D, H, R: N = D + H + R
INPUT "NUMBER OF SIMULATIONS = "; T
CHDIR "C:\TEMP": FILES "*.DAT"
INPUT "FILE NAME FOR STORING DATA (DRIVE:FILENAME.EXT) = "; FILENAME$
OPEN FILENAME$ FOR OUTPUT AS #1
PRINT #1, FILENAME$
P = 2 * D + H: Q = H + 2 * R: P0 = P / (2 * N)
PRINT #1, D, H, R, N, P0, T
CLS
FOR I = 1 TO T
    INDIV = 0: R1 = 0: D1 = 0: H1 = 0
    DO WHILE INDIV < N
        A = RND(1)
        B = RND(1)
        IF A < P0 AND B < P0 THEN
            GENOTYPE = 2
        ELSEIF A >= P0 AND B >= P0 THEN
            GENOTYPE = 0
        ELSE
            GENOTYPE = 1
        END IF
        SELECT CASE GENOTYPE
            CASE 0
                INDIV = INDIV + 1
                R1 = R1 + 1
            CASE 2
                INDIV = INDIV + 1
                D1 = D1 + 1
            CASE 1
                INDIV = INDIV + 1
                H1 = H1 + 1
        END SELECT
    LOOP
    P1 = 2 * D1 + H1: Q1 = H1 + 2 * R1
    D1TOT = D1TOT + D1: H1TOT = H1TOT + H1: R1TOT = R1TOT + R1
    PTOT = PTOT + P1
    PRINT #1, P1, H1
LOCATE 6: PRINT USING "PERFORMING SIMULATION NO. ##### "; I;
    PRINT USING "OUT OF #####"; T
NEXT I
CLOSE #1
PRINT : PRINT "DATA FROM DATAFILE "; FILENAME$: PRINT
PRINT USING "SAMPLE P      =      #.#####"; P0
PRINT USING "SAMPLE D      = #####.#####"; D
PRINT USING "SAMPLE H      = #####.#####"; H
PRINT USING "SAMPLE R      = #####.#####"; R: PRINT
PRINT USING "P MEAN        =      #.#####"; PTOT / (2 * T * N)
PRINT USING "OBS.D MEAN    = #####.#####"; D1TOT / T
PRINT USING "OBS.H MEAN    = #####.#####"; H1TOT / T
PRINT USING "OBS.R MEAN    = #####.#####"; R1TOT / T: PRINT

```


LISTAGEM 11 (VISUAL BASIC 6.0):

```

Public N, P, NS, R88, L88, I, XSCREEN, YSCREEN, F3, F4, F1, F2
Public XMAX, YMAX, XMIN, YMIN, X, Y, F5, F6, F7, PR, NA, HET
Dim ORD(50000, 11), H88(25), F(1000), cy(3), rd(3, 3), ct(3), factor(3), ang(3)
Private Sub Command1_Click()
Command1.Visible = False
Form1.Refresh
Dim YS1(2)

D = Val(Text1.Text)
H = Val(Text2.Text)
R = Val(Text3.Text)
N = D + H + R
Dim SORT(2), LIM1(7), LIMS(7), DIF(50000, 4)
P = (2 * D + H) / (2 * N)
PD = D / N: PH = H / N: PR = R / N
indf = 1 - PH / (2 * P * (1 - P))

SORT(1) = PD: SORT(2) = PD + PH
NA = 2 * D + H: HET = H

If D = 0 Then A11 = 1
If H = 0 Then A22 = 1
If R = 0 Then A33 = 1

For I = 1 To 2 * N
F(I) = F(I - 1) + Log(I)
Next I

F1 = F(N)
F2 = F(2 * N)
b = Prob()
probl = PR
LIM = 10000: X1 = 1000: Y1 = 4000
'Print #1, D & vbCrLf & H & vbCrLf & R
PO = Format(P, "##0.0000")
'Print #1, N & vbCrLf & PO & vbCrLf & LIM & vbCrLf

'Input #1, A$
'Input #1, D99, H99, R99, N99, P99, K99
XMIN = 0: XMAX = 1
YMIN = 0: YMAX = 1
Rem VGA GRAPHICS MODE RESOLUTION 480 x 640
Rem SMALL TRIANGLE WITH EXPLANATION

Picture1.Line (640 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(720 * 7.5 * 6 / 5, 40 * 4.5 / 0.5)
Picture1.Line -(800 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (640 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(667 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (684 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(738 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (756 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(800 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (706 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(706 * 7.5 * 6 / 5, 155 * 4.5 / 0.5)
Picture1.Line (706 * 7.5 * 6 / 5, 179 * 4.5 / 0.5)-(706 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (672 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(682 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)
Picture1.Line (699 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(730 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)
Picture1.Line (747 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(768 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)
Picture1.Circle (706 * 7.5 * 6 / 5, 136 * 4.5 / 0.5), 30
Picture1.CurrentX = 682 * 7.5 * 6 / 5
Picture1.CurrentY = 124 * 4.5 / 0.5
Picture1.Print "d"
Picture1.CurrentX = 735 * 7.5 * 6 / 5
Picture1.CurrentY = 124 * 4.5 / 0.5
Picture1.Print "r"
Picture1.CurrentX = 700 * 7.5 * 6 / 5
Picture1.CurrentY = 155 * 4.5 / 0.5
Picture1.Print "h"
Picture1.CurrentX = 670 * 7.5 * 6 / 5
Picture1.CurrentY = 187 * 4.5 / 0.5
Picture1.Print "p"

```

```

Picture1.CurrentX = 740 * 7.5 * 6 / 5
Picture1.CurrentY = 187 * 4.5 / 0.5
Picture1.Print "q"
Picture1.CurrentX = 90 * 7.5
Picture1.CurrentY = 470 * 4.5 / 0.5
Picture1.Print "0.0"
Picture1.CurrentX = 90 * 7.5
Picture1.CurrentY = 40 * 4.5 / 0.5
Picture1.Print "1.0"
Picture1.CurrentX = 90 * 7.5 * 5 / 3 * 8.5 / 9.8
Picture1.CurrentY = 500 * 4.5 / 0.5
Picture1.Print "0.0"
Picture1.CurrentX = 450 * 7.5 * 5 / 3 * 8.5 / 9.8
Picture1.CurrentY = 500 * 4.5 / 0.5
Picture1.Print "1.0"
Rem GRAPH AXES
Picture1.Line (130 * 7.5, 40 * 4.5 / 0.414)-(130 * 7.5, 400 * 4.5 / 0.414)
Picture1.Line (100 * 7.5 * 5 / 3 * 8.5 / 9.8, 410 * 4.5 / 0.414)-(460 * 7.5 * 5 / 3 * 8.5 / 9.8, 410 * 4.5 / 0.414)
Rem SIDES OF ISOSCELES TRIANGLE
Picture1.Line (100 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)-(280 * 7.5 * 5 / 3 * 8.5 / 9.8, 40 * 4.5 / 0.414)
Picture1.Line -(460 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)
Picture1.Line -(100 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)
Rem Y-AXIS TICKS
For I = 40 To 400 Step 18
    Picture1.Line (126 * 7.5, I * 4.5 / 0.414)-(130 * 7.5, I * 4.5 / 0.414)
Next I
Rem X-AXIS TICKS
For I = 100 To 460 Step 18
    Picture1.Line (I * 7.5 * 5 / 3 * 8.5 / 9.8, 410 * 4.5 / 0.414)-(I * 7.5 * 5 / 3 * 8.5 / 9.8, 414 * 4.5 / 0.414)
Next I

TI = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TI = TI + Val(Mid$(Time$, 7, 2))

Randomize Timer
NA = 2 * D + H

While NS < LIM
    NS = NS + 1
    SIMD = 0: SIMR = 0: SIMH = 0
    S_D = 0: S_R = 0: S_H = 0
    For K = 1 To N
        c = Rnd: W = Rnd: e = Rnd
        If e <= P And W <= P Then
            S_D = S_D + 1
        Else
            If e > P And W > P Then
                S_R = S_R + 1
            Else
                S_H = S_H + 1
            End If
        End If
        If c <= SORT(1) Then
            SIMD = SIMD + 1
        Else
            If c <= SORT(2) Then
                SIMH = SIMH + 1
            Else
                SIMR = SIMR + 1
            End If
        End If
    Next K
    ORD(NS, 1) = (2 * SIMD + SIMH) / (2 * N)
    ORD(NS, 3) = (2 * S_D + S_H) / (2 * N)
    If D - A11 * SIMD <> 0 Then ORD(NS, 2) = (SIMD - D) ^ 2 / Abs(D - A11 * SIMD)

```

```

If H - A22 * SIMH <> 0 Then ORD(NS, 2) = ORD(NS, 2) + (SIMH - H) ^ 2 / Abs(H - A22 * SIMH)
If R - A33 * SIMR <> 0 Then ORD(NS, 2) = ORD(NS, 2) + (SIMR - R) ^ 2 / Abs(R - A33 * SIMR)
ORD(NS, 5) = SIMH
ORD(NS, 6) = (S_D - N * P ^ 2) ^ 2 / (N * P ^ 2)
ORD(NS, 6) = ORD(NS, 6) + (S_H - 2 * N * P * (1 - P)) ^ 2 / (2 * N * P * (1 - P))
ORD(NS, 6) = ORD(NS, 6) + (S_R - N * (1 - P) ^ 2) ^ 2 / (N * (1 - P) ^ 2)
ORD(NS, 7) = S_H
ORD(NS, 8) = ORD(NS, 2)
ORD(NS, 9) = ORD(NS, 6)
If D - a1 * S_D <> 0 Then ORD(NS, 10) = (S_D - D) ^ 2 / Abs(D - A11 * S_D)
If H - a2 * S_H <> 0 Then ORD(NS, 10) = ORD(NS, 10) + (S_H - H) ^ 2 / Abs(H - A22 * S_H)
If R - A3 * S_R <> 0 Then ORD(NS, 10) = ORD(NS, 10) + (S_R - R) ^ 2 / Abs(R - A33 * S_R)
ORD(NS, 11) = (SIMD - N * P ^ 2) ^ 2 / (N * P ^ 2)
ORD(NS, 11) = ORD(NS, 11) + (SIMH - 2 * N * P * (1 - P)) ^ 2 / (2 * N * P * (1 - P))
ORD(NS, 11) = ORD(NS, 11) + (SIMR - N * (1 - P) ^ 2) ^ 2 / (N * (1 - P) ^ 2)
ORD(NS, 4) = SIMD
HET = S_H: NA = 2 * S_D + S_H
a = Prob()
prob2 = PR
If prob2 <= prob1 Then cont11 = cont11 + 1

TOTS_D = TOTS_D + S_D
TOTS_H = TOTS_H + S_H
TOTS_R = TOTS_R + S_R

TOTSIMD = TOTSIMD + SIMD
TOTSIMH = TOTSIMH + SIMH
TOTSIMR = TOTSIMR + SIMR
NA = 2 * SIMD + SIMH

a = NS * 1335 / LIM
Picture2.Line (a1, 0)-(a, 375), , B
a1 = a
Picture3.Cls
PER = NS / LIM * 100
AU = Format(PER, "00")
Picture3.Print AU
Wend

I = 2: a = SORTVALUES()
max1 = ORD(Int(NS * 0.95), 2)
I = 6: a = SORTVALUES()
max2 = ORD(Int(NS * 0.95), 6)

ang(1) = -Atn((2 - 4 * P) * (1 - indf))
ang(2) = -Atn(2 - 4 * P)

factor(2) = 1.7
cy(1) = H / N: cy(2) = 2 * P * (1 - P)
If N >= 150 Then
    If cy(1) > 0.1 Then
        factor(1) = 1.5 - 0.5 * N / 300
    Else
        factor(1) = 0.5
        If P < 0.15 Or P > 0.85 Then ang(1) = -ang(1)
    End If
Else
    If N >= 50 Then
        If cy(1) > 0.1 And cy(1) <= 0.5 Then
            factor(1) = factor(2)
        Else
            If cy(1) <= 0.1 Then
                factor(1) = 0.5
                If P < 0.15 Or P > 0.85 Then ang(1) = -ang(1)
            End If
            If cy(1) > 0.5 Then factor(1) = 2
        End If
    Else
        If cy(1) < 0.5 And cy(1) > 0.1 Then

```

```

        factor(1) = factor(2)
    Else
        If cy(1) <= 0.1 Then
            factor(1) = 0.67
            If P < 0.15 Or P > 0.85 Then ang(1) = -ang(1)
        End If
        If cy(1) > 0.5 Then factor(1) = 2
    End If
End If
End If
End If

For j = 1 To 2
    Print pi: rr1 = 0: rr2 = 0: pi = 0
    While pi < 0.95
        ct(j) = 0
        rr1 = rr1 + 0.0005
        rr2 = factor(j) * rr1
        For I = 1 To NS
            If (ORD(I, 1 + (j - 1) * 2) - P) <> 0 Then
                ang1 = Atn((ORD(I, 5 + (j - 1) * 2) / N - cy(j)) / (ORD(I, 1 + (j - 1) * 2) - P))
            Else
                If (ORD(I, 5 + (j - 1) * 2) / N - cy(j)) >= 0 Then
                    ang1 = 3.1415 / 2
                Else
                    ang1 = 3.1415 + 3.1415 / 2
                End If
            End If
            Y = rr2 * Cos(ang1) * Sin(ang(j)) + rr1 * Sin(ang1) * Cos(ang(j))
            X = rr2 * Cos(ang1) * Cos(ang(j)) - rr1 * Sin(ang1) * Sin(ang(j))
            b = (ORD(I, 5 + (j - 1) * 2) / N - cy(j)) ^ 2 + (ORD(I, 1 + (j - 1) * 2) - P) ^ 2
            If X ^ 2 + Y ^ 2 >= b Then ct(j) = ct(j) + 1
        Next I
        pi = ct(j) / NS
    Wend
    rd(j, 1) = rr1: rd(j, 2) = rr2
Next j

For I = 1 To NS
    a1 = Format(ORD(I, 1), "0.0000")
    a2 = Format(ORD(I, 5) / N, "0.0000")
    Print #1, a1, a2
    'If ORD(I, 8) <= max1 Then Print #2, a1, a2
    If ORD(I, 8) >= max1 - 0.5 And ORD(I, 8) <= max1 + 0.5 Then
        Print #2, a1, a2
    End If
Next I
Print #1, -1, -1: Print #2, -1, -1
For I = 1 To NS
    a1 = Format(ORD(I, 3), "0.0000")
    a2 = Format(ORD(I, 7) / N, "0.0000")
    Print #1, a1, a2
    'If ORD(I, 9) <= max2 Then Print #2, a1, a2
    If ORD(I, 9) >= max2 - 0.5 And ORD(I, 9) <= max2 + 0.5 Then
        Print #2, a1, a2
    End If
Next I

Picture1.ForeColor = &HC000C0
If indf < 0 Then
    b = -indf / (1 - indf): e = 1 / (1 - indf)
Else
    b = 0: e = 1
End If
For X = b To e Step 1 / 400
    Y = 2 * X * (1 - X) * (1 - indf)
    a = PLOTXY()
    If X = b Then Picture1.PSet (XSCREEN, YSCREEN) Else Picture1.Line -(XSCREEN, YSCREEN)
Next X

```

```

X = e
Y = 2 * X * (1 - X) * (1 - indf)
a = PLOTXY()
Picture1.Line -(XSCREEN, YSCREEN)

For I = 1 To NS
    If ORD(I, 8) > max1 Then
        Picture1.ForeColor = &HFF0000
    Else
        If ORD(I, 11) <= max2 Then
            CINTER = CINTER + 1
            a1 = Format(ORD(I, 1), "0.0000")
            a2 = Format(ORD(I, 5) / N, "0.0000")
            Print #3, a1, a2
        End If
        Picture1.ForeColor = &H80C0FF
    End If
    X = ORD(I, 1): Y = ORD(I, 5) / N
    a = PLOTXY()
    If X <> 0 And X <> 1 Then
        Z = Rnd
        If Z <= 1 / 5 Then
            XSCREEN = XSCREEN + 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
            YSCREEN = YSCREEN + 360 * Rnd * 1 / N * (4.5 / 0.414)
            If YSCREEN > 4347.8260869565 Then YSCREEN = 4347.8260869565 - 3.4
            If XSCREEN > 4987.2448979591 Then XSCREEN = 4987.2448979591 - 3.4
            If XSCREEN > 3035.7142857142 Then
                If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.00511701337 * XSCREEN - 5652.181 - 3.4
            End If
        Else
            If Z <= 2 / 5 Then
                XSCREEN = XSCREEN - 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
                YSCREEN = YSCREEN + 360 * Rnd * 1 / N * (4.5 / 0.414)
                If XSCREEN < 1084.1836734693 Then
                    XSCREEN = 1084.1836734693 - 3.4
                Else
                    If XSCREEN <= 3035.7142857142 Then
                        If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 * XSCREEN + 6521.737 - 3.4
                    Else
                        If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.005117 * XSCREEN - 5652.181 - 3.4
                    End If
                End If
            Else
                If Z <= 3 / 5 Then
                    XSCREEN = XSCREEN - 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
                    YSCREEN = YSCREEN - 360 * Rnd * 1 / N * (4.5 / 0.414)
                    If YSCREEN < 434.7826086956 Then YSCREEN = 434.7826086956 - 3.4
                    If XSCREEN < 1084.1836734693 Then
                        XSCREEN = 1084.1836734693 - 3.4
                    Else
                        If XSCREEN <= 3035.7142857142 Then
                            If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 * XSCREEN + 6521.737 - 3.4
                        Else
                            If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.005117 * XSCREEN - 5652.181 - 3.4
                        End If
                    End If
                End If
            Else
                If Z <= 4 / 5 Then
                    XSCREEN = XSCREEN + 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
                    YSCREEN = YSCREEN - 360 * Rnd * 1 / N * (4.5 / 0.414)
                    If YSCREEN < 434.7826086956 Then YSCREEN = 434.7826086956 - 3.4
                    If XSCREEN > 4987.2448979591 Then
                        XSCREEN = 4987.2448979591 - 3.4
                    Else
                        If XSCREEN > 3035.7142857142 Then

```

```

        If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.00511701337 *
XSCREEN - 5652.181 - 3.4
    Else
        If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 *
XSCREEN + 6521.737 - 3.4
    End If
    End If
    End If
    End If
    End If
    End If
    End If
    End If
    End If
    End If
    Picture1.Circle (XSCREEN, YSCREEN), 10
Next I
Print #3, -1, -1
For I = 1 To NS
    If ORD(I, 9) > max2 Then
        Picture1.ForeColor = &H80FF80
    Else
        If ORD(I, 10) <= max1 Then
            CINTER = CINTER + 1
            a1 = Format(ORD(I, 3), "0.0000")
            a2 = Format(ORD(I, 7) / N, "0.0000")
            Print #3, a1, a2
        End If
        Picture1.ForeColor = &H808080
    End If
    X = ORD(I, 3): Y = ORD(I, 7) / N
    a = PLOTXY()
    If X <> 0 And X <> 1 Then
        Z = Rnd
        If Z <= 1 / 5 Then
            XSCREEN = XSCREEN + 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
            YSCREEN = YSCREEN + 360 * Rnd * 1 / N * (4.5 / 0.414)
            If YSCREEN > 4347.8260869565 Then YSCREEN = 4347.8260869565 - 3.4
            If XSCREEN > 4987.2448979591 Then XSCREEN = 4987.2448979591 - 3.4
            If XSCREEN > 3035.7142857142 Then
                If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.00511701337 * XSCREEN
- 5652.181 - 3.4
            End If
        Else
            If Z <= 2 / 5 Then
                XSCREEN = XSCREEN - 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
                YSCREEN = YSCREEN + 360 * Rnd * 1 / N * (4.5 / 0.414)
                If XSCREEN < 1084.1836734693 Then
                    XSCREEN = 1084.1836734693 - 3.4
                Else
                    If XSCREEN <= 3035.7142857142 Then
                        If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 *
XSCREEN + 6521.737 - 3.4
                    Else
                        If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.005117 * XSCREEN -
5652.181 - 3.4
                    End If
                    If YSCREEN > 4347.8260869565 Then YSCREEN = 4347.8260869565 - 3.4
                End If
            Else
                If Z <= 3 / 5 Then
                    XSCREEN = XSCREEN - 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
                    YSCREEN = YSCREEN - 360 * Rnd * 1 / N * (4.5 / 0.414)
                    If YSCREEN < 434.7826086956 Then YSCREEN = 434.7826086956 - 3.4
                    If XSCREEN < 1084.1836734693 Then
                        XSCREEN = 1084.1836734693 - 3.4
                    Else
                        If XSCREEN <= 3035.7142857142 Then
                            If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 *
XSCREEN + 6521.737 - 3.4
                        Else
                            If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.005117 * XSCREEN -
5652.181 - 3.4
                        End If
                    End If
                End If
            End If
        End If
    End If

```

```

End If
Else
If Z <= 4 / 5 Then
XSCREEN = XSCREEN + 360 * Rnd * 1 / (2 * N) * 7.5 * 5 / 3 * 8.5 / 9.8
YSCREEN = YSCREEN - 360 * Rnd * 1 / N * (4.5 / 0.414)
If YSCREEN < 434.7826086956 Then YSCREEN = 434.7826086956 - 3.4
If XSCREEN > 4987.2448979591 Then
XSCREEN = 4987.2448979591 - 3.4
Else
If XSCREEN > 3035.7142857142 Then
If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.00511701337 *
XSCREEN - 5652.181 - 3.4
Else
If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 *
XSCREEN + 6521.737 - 3.4
End If
End If
End If
End If
End If
End If
End If
Picture1.Circle (XSCREEN, YSCREEN), 10
Next I

Picture1.ForeColor = &H80000012
For X = XMIN To XMAX Step 1 / 400
Y = 2 * X * (1 - X)
a = PLOTXY()
If X = XMIN Then Picture1.PSet (XSCREEN, YSCREEN) Else Picture1.Line -(XSCREEN, YSCREEN)
Next X
X = XMAX
Y = 2 * X * (1 - X)
a = PLOTXY()

Picture1.Line -(XSCREEN, YSCREEN)

F2 = Sqr(3.841 / N)
b2 = F2 / (1 + F2): e2 = 1 / (1 + F2)

Picture1.ForeColor = &HC000C0
If indf < 0 Then
b = -indf / (1 - indf): e = 1 / (1 - indf)
Else
b = 0: e = 1
End If
For X = b To e Step 1 / 400
Y = 2 * X * (1 - X) * (1 - indf)
a = PLOTXY()
If X = b Then Picture1.PSet (XSCREEN, YSCREEN) Else Picture1.Line -(XSCREEN, YSCREEN)
Next X
X = e
Y = 2 * X * (1 - X) * (1 - indf)
a = PLOTXY()
Picture1.Line -(XSCREEN, YSCREEN)

Picture1.ForeColor = &H80FFFF
For j = -1 To 1 Step 2
If j = 1 Then b2 = 0: e2 = 1
For X = b2 To e2 Step 1 / 400
Y = 2 * X * (1 - X) * (1 - 1 * j * F2)
a = PLOTXY()
If X = b2 Then Picture1.PSet (XSCREEN, YSCREEN) Else Picture1.Line -(XSCREEN, YSCREEN)
Next X
X = e2
Y = 2 * X * (1 - X) * (1 - 1 * j * F2)
a = PLOTXY()
Picture1.Line -(XSCREEN, YSCREEN)
Next j

Picture1.ForeColor = &H0&

```

```

For j = 1 To 2
  For I = -rd(j, 1) To rd(j, 1) Step 1 / 50000
    YA = rd(j, 2) / rd(j, 1) * Sqr(rd(j, 1) ^ 2 - I ^ 2)
    Y = cy(j) + I * Sin(ang(j)) + YA * Cos(ang(j))
    X = P + I * Cos(ang(j)) - YA * Sin(ang(j))
    a = PLOTXY()
    If I <> -rd(j, 1) Then
      If X >= 0 And X <= 1 And Y >= 0 And Y <= 1 Then
        If YSCREEN >= 2.00511701337 * XSCREEN - 5652.181 Then
          If YSCREEN >= -2.00511495846 * XSCREEN + 6521.737 Then Picture1.Line (xant1, yant1)-
(XSCREEN, YSCREEN)
          End If
        End If
      End If
      xant1 = XSCREEN: yant1 = YSCREEN
      Y = cy(j) + I * Sin(ang(j)) - YA * Cos(ang(j))
      X = P + I * Cos(ang(j)) + YA * Sin(ang(j))
      a = PLOTXY()
      If I <> -rd(j, 1) Then
        If X >= 0 And X <= 1 And Y >= 0 And Y <= 1 Then
          If YSCREEN >= 2.00511701337 * XSCREEN - 5652.181 Then
            If YSCREEN >= -2.00511495846 * XSCREEN + 6521.737 Then Picture1.Line (xant2, yant2)-
(XSCREEN, YSCREEN)
            End If
          End If
        End If
      End If
      xant2 = XSCREEN: yant2 = YSCREEN
    End If
  Next I, j

TF = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TF = TF + Val(Mid$(Time$, 7, 2)):
Text4.Text = "TEMPO GASTO EM TODAS AS SIMULACOES COM POP. DE TAM. " & Str$(N) & " : " &
TF - TI & " SEG."
Text4.Text = Text4.Text & vbCrLf
Text4.Text = Text4.Text & " D OBS=" & D & vbCrLf
Text4.Text = Text4.Text & " H OBS=" & H & vbCrLf
Text4.Text = Text4.Text & " R OBS=" & R & vbCrLf & vbCrLf
DESP = Format(N * P ^ 2, "##0.0000")
Text4.Text = Text4.Text & " D ESP=" & DESP & vbCrLf
HESP = Format(2 * N * P * (1 - P), "##0.0000")
Text4.Text = Text4.Text & " H ESP=" & HESP & vbCrLf
RESP = Format(N * (1 - P) ^ 2, "##0.0000")
Text4.Text = Text4.Text & " R ESP=" & RESP & vbCrLf & vbCrLf
Text4.Text = Text4.Text & " D MEDIO=" & TOTSIMD / NS & vbCrLf
Text4.Text = Text4.Text & " H MEDIO=" & TOTSIMH / NS & vbCrLf
Text4.Text = Text4.Text & " R MEDIO=" & TOTSIMR / NS & vbCrLf & vbCrLf
PE = Format(cont11 / NS, "##0.0000")
Text4.Text = Text4.Text & " P.E=" & PE & vbCrLf & vbCrLf
For I = 1 To 7
  LIM1(I) = Format(ORD(b + 1, I), "##0.0000")
  LIM5(I) = Format(ORD(NS - b, I), "##0.0000")
Next I
Text4.Text = Text4.Text & "I.C. DE PGEN={" & LIM1(1)
Text4.Text = Text4.Text & " , " & LIM5(1) & "}" & vbCrLf
Text4.Text = Text4.Text & "I.C. DE F={" & LIM1(2)
Text4.Text = Text4.Text & " , " & LIM5(2) & "}" & vbCrLf
Text4.Text = Text4.Text & "I.C. DE PBOOT={" & LIM1(3)
Text4.Text = Text4.Text & " , " & LIM5(3) & "}" & vbCrLf
Text4.Text = Text4.Text & "I.C. DE DGEN={" & LIM1(4)
Text4.Text = Text4.Text & " , " & LIM5(4) & "}" & vbCrLf
Text4.Text = Text4.Text & "I.C. DE HGEN={" & LIM1(5)
Text4.Text = Text4.Text & " , " & LIM5(5) & "}" & vbCrLf
Text4.Text = Text4.Text & "I.C. DE HBOOT={" & LIM1(7)
Text4.Text = Text4.Text & " , " & LIM5(7) & "}" & vbCrLf & vbCrLf
Text4.Text = Text4.Text & "I.C. DE RGEN={" & LIM1(6)
Text4.Text = Text4.Text & " , " & LIM5(6) & "}" & vbCrLf
QCRIT1 = Format(ORD(NS - b, 8), "##0.0000")
QCRIT2 = Format(ORD(NS - b, 9), "##0.0000")
Text4.Text = Text4.Text & "QCRIT1=" & QCRIT1 & vbCrLf
Text4.Text = Text4.Text & "QCRIT2=" & QCRIT2 & vbCrLf
Text4.Text = Text4.Text & "INDICE=" & S & vbCrLf

```



```

Text4.Text = Text4.Text & "r1=" & rd(1, 1) & vbCrLf
Text4.Text = Text4.Text & "r1=" & rd(1, 2) & vbCrLf
Text4.Text = Text4.Text & "r1=" & rd(2, 1) & vbCrLf
Text4.Text = Text4.Text & "r1=" & rd(2, 2) & vbCrLf
End Sub

Function SORTVALUES()
D88 = 10: K88 = 1: L88 = 1: R88 = NS
560 Rem EXCHANGE SORT IF BLOCKSIZE <= D88
    If R88 - L88 + 1 > D88 Then GoTo 590
    a = EXCHANGE_SORT: GoTo 780
590 I88 = L88: J88 = R88: W88 = Rnd(1) * (R88 - L88) + 0.5
    W88 = Int(W88 + L88): F88 = ORD(W88, I)
    Rem PARTITION PHASE
620 If ORD(I88, I) >= F88 Then GoTo 640
    I88 = I88 + 1: GoTo 620
640 If ORD(J88, I) <= F88 Then GoTo 660
    J88 = J88 - 1: GoTo 640
660 If I88 > J88 Then GoTo 680
    If I <> 4 Then
        W88 = ORD(I88, I): ORD(I88, I) = ORD(J88, I): ORD(J88, I) = W88: I88 = I88 + 1: J88
        = J88 - 1
    Else
        W88 = ORD(I88, I): ORD(I88, I) = ORD(J88, I): ORD(J88, I) = W88
        W88 = ORD(I88, I + 1): ORD(I88, I + 1) = ORD(J88, I + 1): ORD(J88, I + 1) = W88: I88
        = I88 + 1: J88 = J88 - 1
    End If
680 If I88 <= J88 Then GoTo 620
    Rem BOOKKEEPING PHASE
    If J88 - L88 >= R88 - I88 Then GoTo 740
    If I88 >= R88 Then GoTo 730
    H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GoTo 770
740 If L88 >= J88 Then GoTo 760
    H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 If L88 < R88 Then GoTo 590
780 If K88 <= 1 Then GoTo 800
    K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GoTo 560
800
    End Function
Function EXCHANGE_SORT()
    For I88 = L88 To R88 - 1: K1 = I88
        For J88 = I88 + 1 To R88: If ORD(J88, I) >= ORD(K1, I) Then GoTo 940
        K1 = J88
940 Next J88
    If K1 = I88 Then GoTo 970
    If I <> 4 Then
        W88 = ORD(I88, I): ORD(I88, I) = ORD(K1, I): ORD(K1, I) = W88
    Else
        W88 = ORD(I88, I): ORD(I88, I) = ORD(K1, I): ORD(K1, I) = W88
        W88 = ORD(I88, I + 1): ORD(I88, I + 1) = ORD(K1, I + 1): ORD(K1, I + 1) = W88
    End If
970 Next I88
End Function

Function PLOTXY():
    XSCREEN = (100 + 360 * (X - XMIN) / (XMAX - XMIN)) * 7.5 * 5 / 3 * 8.5 / 9.8
    YSCREEN = (400 - 360 * (Y - YMIN) / (YMAX - YMIN)) * (4.5 / 0.414)
End Function

Function Prob()
F3 = F((NA - HET) / 2)
F4 = F(NA)
F5 = F(N - (NA + HET) / 2)
F6 = F(2 * N - NA)
F7 = F(HET)
PR = F1 - F2 - F3 + F4 - F5 + F6 - F7 + HET * Log(2)

```

End Function

LISTAGEM 12 (QBASIC 1.1):

```

REM PROGRAM FILENAME BOOTSTR3.BAS
REM BOOTSTRAP 95% CONFIDENCE INTERVALS FOR GENOTYPE FREQUENCIES
REM BASED ON 1000 SIMULATED POINTS FOR EACH GENOTYPE
CLS : DEFDBL A-Z: INPUT "NUMBER OF ALLELES = "; K: T = 1000
DIM P(T), GENOTYPE(K, K), ALLELE(K), H88(25)
FOR I = 1 TO K: FOR J = I TO K
PRINT USING "N(###"; I; : PRINT ", "; : PRINT USING "###) = "; J;
INPUT GENOTYPE(I, J): N = N + GENOTYPE(I, J)
ALLELE(I) = ALLELE(I) + GENOTYPE(I, J)
ALLELE(J) = ALLELE(J) + GENOTYPE(I, J)
NEXT J, I: CLS
FOR I = 1 TO K: PRINT USING "ALLELE(###) = "; I;
PRINT USING " #####"; ALLELE(I)
Q(I) = ALLELE(I) / (2 * N)
FOR J = I TO K: PRINT USING "P(###"; I; : PRINT ", ";
PRINT USING "###) = "; J; : PRINT USING "#####"; GENOTYPE(I, J)
DO: LOOP WHILE INKEY$ <> " "
NEXT J, I
PRINT USING "N = #####"; N: PRINT
DO: LOOP WHILE INKEY$ <> " "
PRINT "          95 % P confidence interval"
PRINT "GENOTYPE      P          normal      bootstrap"
PRINT "-----"
FOR I = 1 TO K: FOR J = I TO K
P = GENOTYPE(I, J) / N
PRINT USING " #"; I; : PRINT "/"; : PRINT USING "# "; J;
PRINT USING " obs. #.### "; P;
PRINT USING "{#.###"; P - 1.96 * SQR(P * (1 - P) / N); : PRINT ", ";
PRINT USING "{#.###"; P + 1.96 * SQR(P * (1 - P) / N);
RANDOMIZE TIMER: FOR I99 = 1 TO T: P(I99) = 0
FOR INDIV = 1 TO N
A = RND: IF A <= P THEN P(I99) = P(I99) + 1 / N
NEXT INDIV
NEXT I99
GOSUB SORTVALUES
PRINT USING "{#.###"; P(26); : PRINT ", ";
PRINT USING "{#.###"; P(975)
IF I = J THEN
EXGE = Q(I) * Q(J)
SEEX = SQR(4 * Q(I) ^ 3 * (1 - Q(I)) / (2 * N))
ELSE
EXGE = 2 * Q(I) * Q(J)
SEEX = SQR(4 * Q(I) * Q(J) * (Q(I) + Q(J) - 4 * Q(I) * Q(J)))
SEEX = SEEX / SQR(2 * N)
END IF
PRINT USING "      exp. #.### "; EXGE;
PRINT USING "{#.###"; EXGE - 1.96 * SEEX; : PRINT ", ";
PRINT USING "{#.###"; EXGE + 1.96 * SEEX;
FOR I99 = 1 TO T: Q I = 0: QJ = 0
FOR INDIV = 1 TO N
B = RND
IF B <= Q(I) THEN QI = QI + 1
IF B > Q(I) AND B <= Q(I) + Q(J) THEN QJ = QJ + 1
C = RND
IF C <= Q(I) THEN QI = QI + 1
IF C > Q(I) AND C <= Q(I) + Q(J) THEN QJ = QJ + 1
NEXT INDIV
QI = QI / (2 * N): QJ = QJ / (2 * N)
IF I = J THEN P(I99) = QI ^ 2 ELSE P(I99) = 2 * QI * QJ
NEXT I99
GOSUB SORTVALUES
PRINT USING "{#.###"; P(26); : PRINT ", ";
PRINT USING "{#.###"; P(975)

```

```

'DO: LOOP WHILE INKEY$ <> " "
NEXT J, I
PRINT "-----"
END
SORTVALUES:
D88 = 10: K88 = 1: L88 = 1: R88 = T
560 REM EXCHANGE SORT IF BLOCKSIZE <= D88
    IF R88 - L88 + 1 > D88 THEN 590
    GOSUB 900: GOTO 780
590 I88 = L88: J88 = R88: W88 = RND(1) * (R88 - L88) + .5
    W88 = INT(W88 + L88): F88 = P(W88)
    REM PARTITION PHASE
620 IF P(I88) >= F88 THEN 640
    I88 = I88 + 1: GOTO 620
640 IF P(J88) <= F88 THEN 660
    J88 = J88 - 1: GOTO 640
660 IF I88 > J88 THEN 680
    W88 = P(I88): P(I88) = P(J88): P(J88) = W88: I88 = I88 + 1
    J88 = J88 - 1
680 IF I88 <= J88 THEN 620
    REM BOOKKEEPING PHASE
    IF J88 - L88 >= R88 - I88 THEN 740
    IF I88 >= R88 THEN 730
    H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GOTO 770
740 IF L88 >= J88 THEN 760
    H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 IF L88 < R88 THEN 590
780 IF K88 <= 1 THEN 800
    K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88)
    GOTO 560
800 RETURN

900 REM EXCHANGE SORT
    FOR I88 = L88 TO R88 - 1: K1 = I88
    FOR J88 = I88 + 1 TO R88: IF P(J88) >= P(K1) THEN 940
    K1 = J88
940 NEXT J88
    IF K1 = I88 THEN 970
    W88 = P(I88): P(I88) = P(K1): P(K1) = W88
970 NEXT I88
RETURN

```

LISTAGEM 13 (QBASIC 1.1):

```

REM PROGRAM FILENAME H_WEX13A.BAS
REM 'EXACT' HARDY-WEINBERG TEST (TWO AUTOSOMAL ALLELES)
REM GENERATES T RANDOM SAMPLES OF SIZE N = D + H + R
REM WITH GENOTYPE PROBABILITIES d = p^2, h = 2pq, r = q^2
REM p = (2D+H)/2, q = 1-p
REM WITHOUT REPLACEMENT
DEFDBL A-Z: CLS
INPUT "D, H, R = "; D, H, R: N = D + H + R: P = 2 * D + H: DIM NH(P + 2), PROB1(P + 2),
AUX(P + 2)
INPUT "NUMBER OF SIMULATIONS = "; T: CHDIR "C:\TEMP": FILES "*.DAT"
INPUT "FILE NAME FOR STORING DATA (DRIVE:FILENAME.EXT) = "; FILENAME$
OPEN FILENAME$ FOR OUTPUT AS #1: PRINT #1, FILENAME$
PRINT #1, D, H, R, P, N, T
CLS
FOR I = 1 TO T
    INDIV = 0: R1 = 0: D1 = 0: H1 = 0: N1 = N: P1 = P
    DO WHILE INDIV < N
        A = RND(1): B = RND(1): GENOT = 3
        IF A < P1 / (2 * N1) AND B < (P1 - 1) / (2 * N1 - 1) THEN GENOT = 2
        IF A >= P1 / (2 * N1) AND B >= P1 / (2 * N1 - 1) THEN GENOT = 0
        IF GENOT = 3 THEN GENOT = 1
        N1 = N1 - 1: P1 = P1 - GENOT
        SELECT CASE GENOT
            CASE 0: R1 = R1 + 1
            CASE 2: D1 = D1 + 1
            CASE 1: H1 = H1 + 1
        END SELECT
        INDIV = INDIV + 1
    LOOP
    D1TOT = D1TOT + D1: H1TOT = H1TOT + H1: R1TOT = R1TOT + R1
    PTOT = PTOT + 2 * D1 + H1: NH(H1) = NH(H1) + 1
LOCATE 6: PRINT USING "PERFORMING SIMULATION NO. ##### "; I;
    PRINT USING "OUT OF #####"; T
NEXT I:
PRINT : PRINT "DATA FROM DATAFILE "; FILENAME$: PRINT
PRINT USING "SAMPLE P      =      #.#####"; P / (2 * N)
PRINT USING "SAMPLE D      = #####.#####"; D
PRINT USING "SAMPLE H      = #####.#####"; H
PRINT USING "SAMPLE R      = #####.#####"; R: PRINT
PRINT USING "SIM.P MEAN    =      #.#####"; PTOT / (2 * T * N)
PRINT USING "SIM.D MEAN    = #####.#####"; D1TOT / T
PRINT USING "SIM.H MEAN    = #####.#####"; H1TOT / T
PRINT USING "SIM.R MEAN    = #####.#####"; R1TOT / T: PRINT
FOR I = 0 TO P + 1
    IF NH(I) <> 0 THEN
        PROB1(I) = NH(I): AUX(I) = I
    END IF
NEXT I
GOSUB QUICKSORT
PRINT #1, "      D      H      R      f(x)      F(x)"
PRINT #1, "-----"
FOR H1 = 0 TO P + 1
    IF PROB1(H1) <> 0 THEN
        AC = AC + PROB1(H1) / T
        PRINT #1, USING "##### "; (P - AUX(H1)) / 2; AUX(H1); N - (P + AUX(H1)) / 2;
        IF AUX(H1) = H THEN PRINT #1, "*"; ELSE PRINT #1, " ";
        PRINT #1, USING " #.###"; PROB1(H1) / T;
        PRINT #1, USING " #.###"; AC
    END IF
NEXT H1
PRINT #1, "-----"
CLOSE #1
END
QUICKSORT:

```

```

D99 = 10: K = 1: L = 1: R99 = P + 1
560 REM EXCHANGE SORT IF BLOCKSIZE <= D99
    IF R99 - L + 1 > D99 THEN 590
        GOSUB 900: GOTO 780
590 I = L: J = R99: W = RND(1) * (R99 - L) + .5
    W = INT(W + L): F = PROB1(W)
    REM PARTITION PHASE
620 IF PROB1(I) >= F THEN 640
    I = I + 1: GOTO 620
640 IF PROB1(J) <= F THEN 660
    J = J - 1: GOTO 640
660 IF I > J THEN 680
    W = PROB1(I): PROB1(I) = PROB1(J): PROB1(J) = W
    W = AUX(I): AUX(I) = AUX(J): AUX(J) = W: I = I + 1: J = J - 1
680 IF I <= J THEN 620
    REM BOOKKEEPING PHASE
    IF J - L >= R99 - I THEN 740
        IF I >= R99 THEN 730
            H99(K) = I: K = K + 1: H99(K) = R99: K = K + 1
730 R99 = J: GOTO 770
740 IF L >= J THEN 760
    H99(K) = L: K = K + 1: H99(K) = J: K = K + 1
760 L = I
770 IF L < R99 THEN 590
780 IF K <= 1 THEN 800
    K = K - 1: R99 = H99(K): K = K - 1: L = H99(K): GOTO 560
800 RETURN
900 REM EXCHANGE SORT
    FOR I = L TO R99 - 1: K1 = I
        FOR J = I + 1 TO R99: IF PROB1(J) >= PROB1(K1) THEN 940
            K1 = J
940 NEXT J
    IF K1 = I THEN 970
        W = PROB1(I): PROB1(I) = PROB1(K1): PROB1(K1) = W
        W = AUX(I): AUX(I) = AUX(K1): AUX(K1) = W
970 NEXT I
RETURN

```

LISTAGEM 14 (QBASIC 1.1):

```
RFM PROGRAM FILENAME EXTESTM.BAS
REM THIS PROGRAM CALCULATES THE EXACT TEST
REM IN THE MULTIALLELE CASE USING
REM NS SIMULATIONS USING THE MARKOV CHAIN METHOD

CLS
DIM SIM(10, 10), NU(10)
INPUT "N. AL.="; NAL
FOR I = 1 TO NAL
  FOR J = 1 TO NAL
    PRINT "N(" + STR$(I) + STR$(J) + ")";
    INPUT OBS(I, J)
    N = N + OBS(I, J)
  NEXT J, I
INPUT "FILE NAME FOR STORING DATA (DRIVE:FILENAME.EXT) = "; FILENAME$
OPEN FILENAME$ FOR OUTPUT AS #1

LIM = 500
DIM F(2 * N), MOLD(2 * N), ORD2(LIM, NAL * (NAL + 1) / 2), ORD(LIM)
DIM CONT(LIM), H88(LIM)

DEF FNEXSORT
  FOR I88 = L88 TO R88 - 1: K1 = I88
    FOR J88 = I88 + 1 TO R88: IF CONT(J88) >= CONT(K1) THEN 940
    K1 = J88
  940 NEXT J88
  IF K1 = I88 THEN 970
  W88 = ORD(I88): ORD(I88) = ORD(K1): ORD(K1) = W88
  FOR J = 1 TO NAL * (NAL + 1) / 2
    W88 = ORD2(I88, J): ORD2(I88, J) = ORD2(K1, J): ORD2(K1, J) = W88
  NEXT J
  W88 = CONT(I88): CONT(I88) = CONT(K1): CONT(K1) = W88
  970 NEXT I88

END DEF

DEF FNSORTVALUES
D88 = 10: K88 = 1: L88 = 1: R88 = CLASS
560 REM EXCHANGE SORT IF BLOCKSIZE <= D88
  IF R88 - L88 + 1 > D88 THEN 590
  A = FNEXSORT: GOTO 780
590 I88 = L88: J88 = R88: W88 = RND(1) * (R88 - L88) + .5
  W88 = INT(W88 + L88): F88 = CONT(W88)
  REM PARTITION PHASE
620 IF CONT(I88) >= F88 THEN 640
  I88 = I88 + 1: GOTO 620
640 IF CONT(J88) <= F88 THEN 660
  J88 = J88 - 1: GOTO 640
660 IF I88 > J88 THEN 680
  W88 = ORD(I88): ORD(I88) = ORD(J88): ORD(J88) = W88
  FOR J = 1 TO NAL * (NAL + 1) / 2
    W88 = ORD2(I88, J): ORD2(I88, J) = ORD2(J88, J): ORD2(J88, J) = W88
  NEXT J
  W88 = CONT(I88): CONT(I88) = CONT(J88): CONT(J88) = W88: I88 = I88 + 1: J88 = J88 - 1
680 IF I88 <= J88 THEN 620
  REM BOOKKEEPING PHASE
  IF J88 - L88 >= R88 - I88 THEN 740
  IF I88 >= R88 THEN 730
  H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GOTO 770
740 IF L88 >= J88 THEN 760
  H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 IF L88 < R88 THEN 590
```

```

780 IF K88 <= 1 THEN 800
    K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GOTO 560
800
END DEF

FOR I = 1 TO NAL
    FOR J = I TO NAL
        NU(I) = NU(I) + OBS(I, J)
        NU(J) = NU(J) + OBS(I, J)
    NEXT J
NEXT I
K = 0
FOR I = 1 TO NAL
    FOR J = 1 TO NU(I)
        K = K + 1
        MOLD(K) = I
    NEXT J
NEXT I

FOR I = 1 TO NAL
    NU(I) = NU(I) / (2 * N)
NEXT I

FOR I = 1 TO 2 * N
    F(I) = F(I - 1) + LOG(I)
NEXT I

TI = VAL(MID$(TIME$, 1, 2)) * 3600 + VAL(MID$(TIME$, 4, 2)) * 60
TI = TI + VAL(MID$(TIME$, 7, 2))
F1 = F(N)
F2 = F(2 * N)

DEF FNPBROB
H = 0: PR = 0
FOR Q = 1 TO NAL
    F3 = F(NU(Q) * 2 * N): PR = PR + F3
    FOR J = Q TO NAL
        F4 = F(SIM(Q, J)): PR = PR - F4
        IF Q <> J THEN H = H + SIM(Q, J):
    NEXT J
NEXT Q
PR = PR + F1 - F2 + H * LOG(2)
END DEF

WHILE NS < LIM
    NS = NS + 1
    LOCATE 20, 20
    PRINT "SIMULATION: "; NS
    FOR I = 1 TO NAL
        FOR J = I TO NAL
            SIM(I, J) = 0
        NEXT J, I

    FOR K = 2 * N - 1 TO 1 STEP -2
        C = INT(K * RND) + 1: AL1 = MOLD(C)
        SWAP MOLD(C), MOLD(K + 1)
        E = INT((K - 1) * RND) + 1: AL2 = MOLD(E)
        SWAP MOLD(E), MOLD(K)
        IF AL2 < AL1 THEN SWAP AL1, AL2
        SIM(AL1, AL2) = SIM(AL1, AL2) + 1
    NEXT K
    A = FNPBROB: Y = 0
    FOR I = 1 TO CLASS
        IF PR < ORD(I) THEN
            CONT(I) = CONT(I) + 1
        ELSE
            IF PR = ORD(I) THEN
                CONT(I) = CONT(I) + 1: Y = 1
            END IF
        END IF
    NEXT I

```



```

IF Y = 0 THEN
  CLASS = CLASS + 1
  K = 0
  FOR I = 1 TO NAL
    FOR J = I TO NAL
      K = K + 1
      ORD2(CLASS, K) = SIM(I, J)
    NEXT J, I
  K = 0
  FOR I = 1 TO CLASS - 1
    IF PR < ORD(I) THEN
      CONT(CLASS) = CONT(I - 1) + 1: I = CLASS: K = 1
    END IF
  NEXT I
  IF K = 0 THEN CONT(CLASS) = CONT(CLASS - 1) + 1
  ORD(CLASS) = PR
END IF
A = FNSORTVALUES
WEND

TF = VAL(MID$(TIME$, 1, 2)) * 3600 + VAL(MID$(TIME$, 4, 2)) * 60
TF = TF + VAL(MID$(TIME$, 7, 2)): PRINT #1,
PRINT #1, "TIME USED TO DO ALL CALCULATIONS OF A SAMPLE WITH " + STR$(N); " IND. AND " +
STR$(NS) + " SIM.="; TF - TI; " SEC."
FOR J = 1 TO NAL
  FOR K = J TO NAL
    PRINT #1, STR$(J) + STR$(K); : PRINT #1, " ";
  NEXT K, J
PRINT #1, "      P.E."
FOR I = 1 TO CLASS
  K = 0: M = 0
  FOR J = 1 TO NAL
    FOR L = J TO NAL
      M = M + 1
      IF ORD2(I, M) = OBS(J, L) THEN K = K + M
      PRINT #1, USING " ### "; ORD2(I, M);
    NEXT L, J
    IF K = (M * (M + 1)) / 2 THEN
      PRINT #1, " * ";
    ELSE
      PRINT #1, "   ";
    END IF
  PRINT #1, USING "#.####"; CONT(I) / LIM
NEXT I
CLOSE #1

```

```

LISTAGEM 15 (VISUAL BASIC 6.0):
Public NAL, SIM, L88, R88
Dim PROB1(10000)
Private Sub Command1_Click()
NAL = Val(Text1.Text)
Text2.Visible = True
Text1.Visible = False
Label1.Visible = False
Command1.Visible = False
Command2.Visible = True
GRID1.Visible = True
GRID1.Cols = NAL + 2
GRID1.Rows = NAL + 2

For I = 1 To NAL
    GRID1.FontBold = True
    GRID1.TextArray(fgi(I, 0)) = I
    GRID1.TextArray(fgi(0, I)) = I
Next I

    GRID1.TextArray(fgi(I, 0)) = "TOT"
    GRID1.TextArray(fgi(0, I)) = "TOT"
    GRID1.AllowBigSelection = False
    Text2 = ""

For I = 0 To NAL + 1
    GRID1.ColWidth(I) = GRID1.ColWidth(I) / 2
Next I
End Sub
Private Sub Command2_Click()
Dim AUX(2, 50), TOTM(50), MP(20)
Dim SIMP(10000, 20), MULT(10000, 40), GEN(40, 10000)
Dim SIMT(20, 20), TOTGEN(50), SIMU(20, 20), NMAX(20, 20)
Dim NMN(20, 20), MULMIN(50), MULMAX(50), PMIN(20), PMAX(20)
Dim AL(2), B(2), N(20, 20), P(20), PP(20), MPP(20), PPMAX(20), PPMIN(20)
Dim AA(50), NNMAX(20, 20), NNMIN(20, 20), BB(50), MMAX(20), MMIN(20)
Dim CC(20, 20), PMMIN(20, 20), PMMAX(20, 20)
GRID1.Visible = False
Command2.Visible = False
Label2.Visible = True
Text3.Visible = True
Picture1.Visible = True
Picture2.Visible = True
Text2.Visible = False
Form1.Refresh
Rem MULTIALLELLE JACKNIFE-1
SIM = 10000: Rem NUMERO DE SIMULACES

For I = 1 To NAL
    For J = I To NAL
        N(I, J) = Val(GRID1.TextArray(fgi(I, J)))
        NT = NT + N(I, J)
    Next J, I

For I = 1 To NAL
    For J = 1 To NAL
        P(I) = P(I) + N(I, J) + N(J, I)
    Next J, I
For J = 1 To NAL - 1
    AUX(1, J) = AUX(1, J - 1) + P(J) / (2 * NT)
Next J

For I = 1 To NAL
    For J = I To NAL
        S = S + 1
        AUX(2, S) = AUX(2, S - 1) + N(I, J) / NT
    Next J, I

```

```

SAMPLE = 3 * NT / 2
For I = 1 To SIM
For K = 1 To NAL
  For J = K To NAL
    SIMU(K, J) = 0
  Next J, K
For K = 1 To SAMPLE
  C = Rnd
  For V = 1 To S
    If C <= AUX(2, V) Then
      GEN(V, I) = GEN(V, I) + 1
      V = S + 1
    End If
  Next V
  For V = 1 To 2
    AL(V) = 0
    B(V) = Rnd
    For G = 1 To NAL - 1
      If B(V) <= AUX(1, G) Then
        AL(V) = G: G = NAL + 1
      End If
    Next G
    If AL(V) = 0 Then AL(V) = NAL
  Next V
  If AL(1) > AL(2) Then
    MM = AL(1): AL(1) = AL(2): AL(2) = MM
  End If
  SIMU(AL(1), AL(2)) = SIMU(AL(1), AL(2)) + 1
Next K
W = 0
For V = 1 To NAL
  For G = V To NAL
    W = W + 1
    SIMT(V, G) = SIMT(V, G) + SIMU(V, G)
    TOTGEN(W) = TOTGEN(W) + GEN(W, I)
  Next G
  For J = 1 To NAL
    SIMP(I, J) = SIMP(I, J) + SIMU(V, J) + SIMU(J, V)
  Next J
Next V
A = I * 1395 / SIM
Picture1.Line (A1, 0)-(A, 375), , B
A1 = A
Picture2.Cls
PER = I / SIM * 100
AU = Format(PER, "00")
Picture2.Print AU

Next I

S = 0
For Y = 1 To NAL
  For U = Y To NAL
    S = S + 1
    For I = 1 To SIM
      PROB1(I) = GEN(S, I)
    Next I
    X = SORTVALUES
    NMIN(Y, U) = PROB1(Int(0.025 * SIM + 1))
    NMAX(Y, U) = PROB1(Int(0.975 * SIM))
  Next U, Y

For X = 1 To NAL
  For I = 1 To SIM
    PROB1(I) = SIMP(I, X)
  Next I
  Y = SORTVALUES
  PMIN(X) = PROB1(Int(0.025 * SIM + 1))
  PMAX(X) = PROB1(Int(0.975 * SIM))
Next X

```

```

S = 0
For X = 1 To NAL
  For Y = X + 1 To NAL
    S = S + 1
    For I = 1 To SIM
      MULT(I, S) = SIMP(I, Y) * SIMP(I, X)
      TOTM(S) = TOTM(S) + MULT(I, S)
    Next I
  Next Y, X

For T = 1 To S
  For I = 1 To SIM
    PROB1(I) = MULT(I, T)
  Next I
  X = SORTVALUES
  MULMIN(T) = PROB1(Int(0.025 * SIM + 1))
  MULMAX(T) = PROB1(Int(0.975 * SIM))
Next T
W = 0

For I = 1 To NAL
  For J = 1 To NAL
    MP(I) = MP(I) + SIMT(I, J) + SIMT(J, I)
  Next J, I
  For J = 1 To NAL
    MP(J) = MP(J) / (2 * SAMPLE * SIM)
  Next J

Text3.Text = " OBSERVED GENOTYPES" & vbCrLf

For I = 1 To NAL
  For J = I To NAL
    Text3.Text = Text3.Text & " N(" & I & "-" & J & ") = " & N(I, J) & vbCrLf
  Next J, I
  Text3.Text = Text3.Text & vbCrLf
  Text3.Text = Text3.Text & " NUM. OF SIMULATIONS = " & SIM & vbCrLf
  Text3.Text = Text3.Text & "          P          95% Jackknife Confidence Interval " &
vbCrLf
  For I = 1 To NAL
    PP(I) = Format(P(I) / (2 * NT), "0.0000")
    Text3.Text = Text3.Text & " OBSP(" & Str$(I) & ") = " & PP(I) & vbCrLf
    MPP(I) = Format(MP(I), "0.0000")
    PPMAX(I) = Format(PMAX(I) / (2 * SAMPLE), "0.0000")
    PPMIN(I) = Format(PMIN(I) / (2 * SAMPLE), "0.0000")
    Text3.Text = Text3.Text & " MEDP(" & Str$(I) & ") = " & MPP(I) & " { " & PPMIN(I) &
" , " & PPMAX(I) & " }" & vbCrLf
  Next I

S = 0: W = 0
For I = 1 To NAL
  For J = I To NAL
    Text3.Text = Text3.Text & " " & Str$(I) & " / " & Str$(J) & vbCrLf
    S = S + 1
    AA(S) = Format(TOTGEN(S) / (SAMPLE * SIM), "0.0000")
    NNMIN(I, J) = Format(NMIN(I, J) / SAMPLE, "0.0000")
    NNMAX(I, J) = Format(NMAX(I, J) / SAMPLE, "0.0000")
    Text3.Text = Text3.Text & "      obs. " & AA(S) & " { " & NNMIN(I, J) & " , " &
NNMAX(I, J) & " }" & vbCrLf
    If I <> J Then
      W = W + 1
      BB(W) = Format(2 * TOTM(W) / (SIM * (2 * SAMPLE) ^ 2), "0.0000")
      MMIN(W) = Format(2 * MULMIN(W) / (2 * SAMPLE) ^ 2, "0.0000")
      MMAX(W) = Format(2 * MULMAX(W) / (2 * SAMPLE) ^ 2, "0.0000")
      Text3.Text = Text3.Text & "      exp. " & BB(W) & " { " & MMIN(W) & " , " &
MMAX(W) & " }" & vbCrLf
    Else
      CC(I, J) = Format(MP(I) * MP(J), "0.0000")
      PPMIN(I, J) = Format(PMIN(I) * PMIN(J) / (2 * SAMPLE) ^ 2, "0.0000")
      PPMAX(I, J) = Format(PMAX(I) * PMAX(J) / (2 * SAMPLE) ^ 2, "0.0000")

```

```

        Text3.Text = Text3.Text & "      exp. " & CC(I, J) & "      { " & PMMIN(I, J) & " , " &
PMMAX(I, J) & " }" & vbCrLf
    End If
    Next J, I

End Sub

Sub GRID1_KEYPRESS(A As Integer)
MSHFLEXGRIDEDIT GRID1, Text2, A
End Sub
Sub GRID1_DBLCLICK()
MSHFLEXGRIDEDIT GRID1, Text2, 32
End Sub

Sub MSHFLEXGRIDEDIT(MSHFLEXGRID As Control, _
EDT As Control, A As Integer)

If MSHFLEXGRID.Col < MSHFLEXGRID.Row Then Exit Sub

If MSHFLEXGRID.Row = NAL + 1 Or MSHFLEXGRID.Col = NAL + 1 _
Then Exit Sub

Select Case A
Case 0 To 32
EDT = MSHFLEXGRID
EDT.SelStart = 1000

Case Else
EDT = Chr(A)
EDT.SelStart = 1
End Select

EDT.Move MSHFLEXGRID.Left + MSHFLEXGRID.CellLeft, _
MSHFLEXGRID.Top + MSHFLEXGRID.CellTop, _
MSHFLEXGRID.CellWidth - 8, _
MSHFLEXGRID.CellHeight - 8
EDT.Visible = True

EDT.SetFocus
End Sub
Sub GRID1_GOTFOCUS()
If Text2.Visible = False Then Exit Sub
GRID1 = Text2
Text2.Visible = False
End Sub
Sub GRID1_LEAVECELL()
If Text2.Visible = False Then Exit Sub
GRID1 = Text2
Text2.Visible = False
End Sub

Function fgi(R, S)
fgi = S + GRID1.Cols * R
End Function

Function SORTVALUES()
Dim H88(5000)
D88 = 10: K88 = 1: L88 = 1: R88 = SIM
560 Rem EXCHANGE SORT IF BLOCKSIZE <= D88
    If R88 - L88 + 1 > D88 Then GoTo 590
    A = EXCHANGESORT: GoTo 780
590 I88 = L88: J88 = R88: W88 = Rnd(1) * (R88 - L88) + 0.5
    W88 = Int(W88 + L88): F88 = PROB1(W88)
    Rem PARTITION PHASE
620 If PROB1(I88) >= F88 Then GoTo 640
    I88 = I88 + 1: GoTo 620
640 If PROB1(J88) <= F88 Then GoTo 660
    J88 = J88 - 1: GoTo 640
660 If I88 > J88 Then GoTo 680

```

```

      W88 = PROB1(I88): PROB1(I88) = PROB1(J88): PROB1(J88) = W88: I88 = I88 + 1: J88 = J88
- 1
680 If I88 <= J88 Then GoTo 620
      Rem BOOKKEEPING PHASE
      If J88 - L88 >= R88 - I88 Then GoTo 740
      If I88 >= R88 Then GoTo 730
      H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GoTo 770
740 If L88 >= J88 Then GoTo 760
      H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 If L88 < R88 Then GoTo 590
780 If K88 <= 1 Then GoTo 800
      K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GoTo 560
800
End Function

Function EXCHANGESORT()
      For I88 = L88 To R88 - 1: K1 = I88
      For J88 = I88 + 1 To R88: If PROB1(J88) >= PROB1(K1) Then GoTo 940
      K1 = J88
940 Next J88
      If K1 = I88 Then GoTo 970
      W88 = PROB1(I88): PROB1(I88) = PROB1(K1): PROB1(K1) = W88
970 Next I88

End Function

```

LISTAGENS 16 E 17:

```

REM PROGRAM FILENAME BOOTHIE1.BAS (QBASIC 1.1)
CLS : DEFDBL A-Z
INPUT "Number of subpopulations = "; K
T = 1000: DIM P9(T)
DIM D(K), H(K), R(K), N(K), P(K), X(K), FILENAME$(K + 3), H88(25)
NAME$(1) = "FIT" : NAME$(2) = "FST" : NAME$(3) = "FIS" : "
DATA 09,12,54,05,10,10,33,14,03
CHDIR "c:\temp\": FILES "*.txt"
INPUT "text file name (ex.: c:\temp\results0.txt) = "; FILENAME$
OPEN FILENAME$ FOR OUTPUT AS #99
'OPEN "c:\temp\results1.txt" FOR OUTPUT AS #99
TIMEWA$ = TIME$
TIMEWAS = VAL(MID$(TIMEWA$, 1, 2)) * 3600 + VAL(MID$(TIMEWA$, 4, 2)) * 60
TIMEWAS = TIMEWAS + VAL(MID$(TIMEWA$, 7, 2))
PRINT #99, "ESTIMATES BASED ON SAMPLE NUMBERS": PRINT #99,
PRINT #99, "SUBPOP. N(AA) N(Aa) N(aa) N"
PRINT #99, "-----"
FOR I = 1 TO K
'PRINT USING "SUBPOP. ##: "; I;
'INPUT "N(AA),N(Aa),N(aa) = "; D(I), H(I), R(I)
READ D(I), H(I), R(I)
N(I) = D(I) + H(I) + R(I)
N = N + N(I): D = D + D(I): H = H + H(I): R = R + R(I)
PRINT #99, USING "#### "; I; D(I); H(I); R(I); N(I)
NEXT I: CLS
PRINT #99, "-----"
PRINT #99, "total "; : PRINT #99, USING "#### "; D; H; R; N: PRINT #99,
FOR I = 1 TO K
P(I) = (2 * D(I) + H(I)) / (2 * N(I))
F(I) = 1 - (H(I) / N(I)) / (2 * P(I) * (1 - P(I)))
PRINT #99, USING "p(##) = "; I; : PRINT #99, USING "#.####"; P(I)
PRINT #99, USING "f(##) = "; I; : PRINT #99, USING "#.####"; F(I)
X(I) = N(I) / N: P = P + X(I) * P(I)
VP = VP + X(I) * P(I) * P(I): NEXT I
VP = VP - P * P: FIT = 1 - (H / N) / (2 * P * (1 - P))
FST = VP / (P * (1 - P)): FIS = (FIT - FST) / (1 - FST)
PRINT #99, "p = "; : PRINT #99, USING "#.####"; P
PRINT #99, "var(p) = "; : PRINT #99, USING "#.####"; VP
PRINT #99, "FIT = "; : PRINT #99, USING "#.####"; FIT
PRINT #99, "FST = "; : PRINT #99, USING "#.####"; FST
PRINT #99, "FIS = "; : PRINT #99, USING "#.####"; FIS: PRINT #99,
PRINT #99, USING "AVERAGE ESTIMATES BASED ON #### SIMULATIONS": T: PRINT #99,
FOR I = 1 TO K
FILENAME$(I) = "c:\temp\POPULAT" + MID$(STR$(I), 2, 1) + ".DAT"
NEXT I
FILENAME$(K + 1) = "c:\temp\DATA_Fit.DAT"
FILENAME$(K + 2) = "c:\temp\DATA_Fst.DAT"
FILENAME$(K + 3) = "c:\temp\DATA_Fis.DAT"
FOR I9 = 1 TO K
OPEN FILENAME$(I9) FOR OUTPUT AS #I9
SSP = 0: FOR I = 1 TO 3: SP(I) = 0: NEXT I
P(1) = D(I9) / N(I9): P(2) = H(I9) / N(I9): P(3) = R(I9) / N(I9)
FOR I = 1 TO 3
SSP = SSP + P(I): SP(I) = SP(I) + SSP
NEXT I
FOR I = 1 TO T
LOCATE 10: PRINT USING "SUBPOPULATION ##: "; I9;
PRINT USING "SIMULATION ####"; I
D1 = 0: H1 = 0: R1 = 0
FOR INDIV = 1 TO N(I9)
A = RND(1)
IF A <= SP(1) THEN D1 = D1 + 1
IF A > SP(1) AND A <= SP(2) THEN H1 = H1 + 1

```

```

      IF A > SP(2) AND A <= SP(3) THEN R1 = R1 + 1
NEXT INDIV
P1 = 2 * D1 + H1: Q1 = H1 + 2 * R1
F1 = 1 - H1 / (P1 * Q1 / (P1 + Q1))
PRINT #I9, D1, H1, R1, P1
SUMF1 = SUMF1 + F1: SUMP1 = SUMP1 + P1 / (2 * N(I9))
NEXT I
CLOSE #I9
PRINT #99, USING "P(##) = "; I9; : PRINT #99, USING "#.####"; SUMP1 / T
PRINT #99, USING "F(##) = "; I9; : PRINT #99, USING "#.####"; SUMF1 / T
SUMF1 = 0: SUMP1 = 0
NEXT I9
FOR I = 1 TO K: OPEN FILENAME$(I) FOR INPUT AS #I: NEXT I
FOR I = K + 1 TO K + 3: OPEN FILENAME$(I) FOR OUTPUT AS #I: NEXT I
FOR I = 1 TO T
  FOR J = 1 TO K
    INPUT #J, D, H, R, P
    LOCATE 10: PRINT USING "F-STATISTICS  ##: "; J;
      PRINT USING "CALCULAT.  ####"; I
    HT = HT + H: P = P / (2 * N(J))
    PT = PT + X(J) * P: VARP = VARP + X(J) * P * P
  NEXT J
  VARP = VARP - PT * PT: FIT = 1 - (HT / N) / (2 * PT * (1 - PT))
  FST = VARP / (PT * (1 - PT)): FIS = (FIT - FST) / (1 - FST)
  HT = 0: PT = 0: VARP = 0
  PRINT #K + 1, FIT: PRINT #K + 2, FST: PRINT #K + 3, FIS
  SUMFIT = SUMFIT + FIT: SUMFST = SUMFST + FST: SUMFIS = SUMFIS + FIS
NEXT I
PRINT #99, "FIT      = "; : PRINT #99, USING "#.####"; SUMFIT / T
PRINT #99, "FST      = "; : PRINT #99, USING "#.####"; SUMFST / T
PRINT #99, "FIS      = "; : PRINT #99, USING "#.####"; SUMFIS / T: PRINT #99,
PRINT #99, "MEDIAN AND 95% BOOTSTRAP CONFIDENCE INTERVALS": PRINT #99,
FOR I = 1 TO K + 3: CLOSE #I: NEXT I
FOR I = 1 TO K + 3
  OPEN FILENAME$(I) FOR INPUT AS #I
  FOR J = 1 TO T
    LOCATE 10: PRINT USING "F DATA SET  ##: "; I;
      PRINT USING "SORTING  ####"; J
    IF I > K THEN INPUT #I, F ELSE INPUT #I, D, H, R, P
    IF I <= K THEN F = 1 - H / (P * (1 - P / (2 * N(I))))
    P9(J) = F
  NEXT J
  CLOSE #I
  GOSUB SORTVALUES
  IF I <= K THEN
    PRINT #99, USING "F(##)  : "; I;
  ELSE
    PRINT #99, NAME$(I - 3);
  END IF
  PRINT #99, USING "#.### "; (P9(500) + P9(501)) / 2;
  PRINT #99, USING "{#.####"; P9(26); : PRINT #99, ", ";
  PRINT #99, USING "#.####"; P9(975)
NEXT I
CLOSE #99
TIMEI$ = TIME$
TIMEIS = VAL(MID$(TIMEI$, 1, 2)) * 3600 + VAL(MID$(TIMEI$, 4, 2)) * 60
TIMEIS = TIMEIS + VAL(MID$(TIMEI$, 7, 2))
LOCATE 15: PRINT USING "TOTAL PROCESSING TIME = ##### SECONDS"; TIMEIS - TIMEWAS
END
SORTVALUES:
D88 = 10: K88 = 1: L88 = 1: R88 = T
560 REM EXCHANGE SORT IF BLOCKSIZE <= D88
  IF R88 - L88 + 1 > D88 THEN 590
  GOSUB 900: GOTO 780
590 I88 = L88: J88 = R88: W88 = RND(1) * (R88 - L88) + .5
  W88 = INT(W88 + L88): F88 = P9(W88)
  REM PARTITION PHASE
620 IF P9(I88) >= F88 THEN 640
  I88 = I88 + 1: GOTO 620
640 IF P9(J88) <= F88 THEN 660
  J88 = J88 - 1: GOTO 640

```



```

660 IF I88 > J88 THEN 680
    W88 = P9(I88): P9(I88) = P9(J88): P9(J88) = W88: I88 = I88 + 1: J88 = J88 - 1
680 IF I88 <= J88 THEN 620
    REM BOOKKEEPING PHASE
    IF J88 - L88 >= R88 - I88 THEN 740
    IF I88 >= R88 THEN 730
    H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GOTO 770
740 IF L88 >= J88 THEN 760
    H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 IF L88 < R88 THEN 590
780 IF K88 <= 1 THEN 800
    K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GOTO 560
800 RETURN
900 REM EXCHANGE SORT
    FOR I88 = L88 TO R88 - 1: K1 = I88
    FOR J88 = I88 + 1 TO R88: IF P9(J88) >= P9(K1) THEN 940
    K1 = J88
940 NEXT J88
    IF K1 = I88 THEN 970
    W88 = P9(I88): P9(I88) = P9(K1): P9(K1) = W88
970 NEXT I88
RETURN

```

```

REM PROGRAM FILENAME BOOTHIE2.BAS (QBASIC 1.1)
CLS : DEFDBL A-Z
K = 3
DIM D(K), H(K), R(K), N(K), P(K), X(K), H88(25), P9(1000)
NAME$(1) = "FIT      ": NAME$(2) = "FST      ": NAME$(3) = "FIS      ": 
DATA 09,12,54,05,10,10,33,14,03
OPEN "c:\temp\results2.txt" FOR OUTPUT AS #99
TIMEWA$ = TIME$
TIMEWAS = VAL(MID$(TIMEWA$, 1, 2)) * 3600 + VAL(MID$(TIMEWA$, 4, 2)) * 60
TIMEWAS = TIMEWAS + VAL(MID$(TIMEWA$, 7, 2))
PRINT #99, "ESTIMATES BASED ON SAMPLE NUMBERS": PRINT #99,
PRINT #99, "SUBPOP.   N(AA)   N(Aa)   N(aa)   N"
PRINT #99, "-----"
FOR I = 1 TO K
'PRINT USING "SUBPOP. ##: "; I;
'INPUT "N(AA),N(Aa),N(aa) = "; D(I), H(I), R(I)
READ D(I), H(I), R(I)
N(I) = D(I) + H(I) + R(I)
N = N + N(I): D = D + D(I): H = H + H(I): R = R + R(I)
PRINT #99, USING "####      "; I; D(I); H(I); R(I); N(I)
NEXT I: CLS
PRINT #99, "-----"
PRINT #99, " total      "; : PRINT #99, USING "####      "; D; H; R; N: PRINT #99,
FOR I = 1 TO K
P(I) = (2 * D(I) + H(I)) / (2 * N(I))
F(I) = 1 - (H(I) / N(I)) / (2 * P(I) * (1 - P(I)))
PRINT #99, USING "p(##) = "; I; : PRINT #99, USING "#.#####"; P(I)
PRINT #99, USING "F(##) = "; I; : PRINT #99, USING "#.#####"; F(I)
X(I) = N(I) / N: P = P + X(I) * P(I)
VP = VP + X(I) * P(I) * P(I): NEXT I
VP = VP - P * P: FIT = 1 - (H / N) / (2 * P * (1 - P))
FST = VP / (P * (1 - P)): FIS = (FIT - FST) / (1 - FST)
PRINT #99, "p      = "; : PRINT #99, USING "#.#####"; P
PRINT #99, "var(p) = "; : PRINT #99, USING "#.#####"; VP
PRINT #99, "FIT = "; : PRINT #99, USING "#.#####"; FIT
PRINT #99, "FST = "; : PRINT #99, USING "#.#####"; FST
PRINT #99, "FIS = "; : PRINT #99, USING "#.#####"; FIS: PRINT #99,
PRINT #99, "AVERAGE ESTIMATES BASED ON 10 SIMULATIONS"
FILENAME$(1) = "c:\temp\DAT1_Fit.DAT"
FILENAME$(2) = "c:\temp\DAT1_Fst.DAT"
FILENAME$(3) = "c:\temp\DAT1_Fis.DAT"
FOR I9 = 1 TO K
SSP = 0: FOR I = 1 TO 3: SP(I) = 0: NEXT I
P(1) = D(I9) / N(I9): P(2) = H(I9) / N(I9): P(3) = R(I9) / N(I9)
FOR I = 1 TO 3
SSP = SSP + P(I): SP(I) = SP(I) + SSP
NEXT I
FOR I = 1 TO 10
LOCATE 10: PRINT USING "SUBPOPULATION ##: "; I9;
PRINT USING "SIMULATION #####"; I
D1 = 0: H1 = 0: R1 = 0
FOR INDIV = 1 TO N(I9)
A = RND(1)
IF A <= SP(1) THEN D1 = D1 + 1
IF A > SP(1) AND A <= SP(2) THEN H1 = H1 + 1
IF A > SP(2) AND A <= SP(3) THEN R1 = R1 + 1
NEXT INDIV
P1 = 2 * D1 + H1: Q1 = H1 + 2 * R1
F1 = 1 - H1 / (P1 * Q1 / (P1 + Q1))
H2(I9, I) = H1: P2(I9, I) = P1
SUMF1 = SUMF1 + F1: SUMP1 = SUMP1 + P1 / (2 * N(I9))
NEXT I
PRINT #99, USING "p(##) = "; I9; : PRINT #99, USING "#.#####"; SUMP1 / 10
PRINT #99, USING "F(##) = "; I9; : PRINT #99, USING "#.#####"; SUMF1 / 10
SUMF1 = 0: SUMP1 = 0
NEXT I9: PRINT #99,
I4 = 1
FOR I = 1 TO 3: OPEN FILENAME$(I) FOR OUTPUT AS #I: NEXT I
FOR I1 = 1 TO 10: FOR I2 = 1 TO 10: FOR I3 = 1 TO 10
LOCATE 10: PRINT USING "F-STATISTICS: CALCULAT. #####"; I4

```

```

HT = H2(1, I1) + H2(2, I2) + H2(3, I3)
P11 = P2(1, I1) / (2 * N(1))
P12 = P2(2, I2) / (2 * N(2))
P13 = P2(3, I3) / (2 * N(3))
PT = X(1) * P11 + X(2) * P12 + X(3) * P13
VARP = X(1) * P11 * P11 + X(2) * P12 * P12 + X(3) * P13 * P13 - PT * PT
FIT = 1 - (HT / N) / (2 * PT * (1 - PT))
FST = VARP / (PT * (1 - PT)): FIS = (FIT - FST) / (1 - FST)
I4 = I4 + 1
PRINT #1, FIT: PRINT #2, FST: PRINT #3, FIS
SUMFIT = SUMFIT + FIT: SUMFST = SUMFST + FST: SUMFIS = SUMFIS + FIS
NEXT I3, I2, I1
FOR I = 1 TO 3: CLOSE #I: NEXT I
PRINT #99, "AVERAGE ESTIMATES BASED ON 1000 SIMULATIONS"
PRINT #99, "FIT      = "; : PRINT #99, USING "#.#####"; SUMFIT / 1000
PRINT #99, "FST      = "; : PRINT #99, USING "#.#####"; SUMFST / 1000
PRINT #99, "FIS      = "; : PRINT #99, USING "#.#####"; SUMFIS / 1000: PRINT #99,
PRINT #99, "MEDIAN AND 95% BOOTSTRAP CONFIDENCE INTERVALS"
FOR I = 1 TO 3
OPEN FILENAME$(I) FOR INPUT AS #I
FOR J = 1 TO 1000
LOCATE 10: PRINT USING "F DATA SET      ##: "; I;
      PRINT USING "SORTING      #####"; J
INPUT #I, F
P9(J) = F
NEXT J
CLOSE #I
GOSUB SORTVALUES
PRINT #99, NAME$(I);
PRINT #99, USING "#.### "; (P9(500) + P9(501)) / 2;
PRINT #99, USING "{#.#####"; P9(26); : PRINT #99, ", ";
PRINT #99, USING "#.#####"; P9(975)
NEXT I
CLOSE #99
TIMEIS = TIME$
TIMEIS = VAL(MID$(TIMEIS, 1, 2)) * 3600 + VAL(MID$(TIMEIS, 4, 2)) * 60
TIMEIS = TIMEIS + VAL(MID$(TIMEIS, 7, 2))
LOCATE 15: PRINT USING "TOTAL PROCESSING TIME = ##### SECONDS"; TIMEIS - TIMEWAS
END
SORTVALUES:
D88 = 10: K88 = 1: L88 = 1: R88 = 1000
560 REM EXCHANGE SORT IF BLOCKSIZE <= D88
      IF R88 - L88 + 1 > D88 THEN 590
      GOSUB 900: GOTO 780
590 I88 = L88: J88 = R88: W88 = RND(1) * (R88 - L88) + .5
      W88 = INT(W88 + L88): F88 = P9(W88)
      REM PARTITION PHASE
620 IF P9(I88) >= F88 THEN 640
      I88 = I88 + 1: GOTO 620
640 IF P9(J88) <= F88 THEN 660
      J88 = J88 - 1: GOTO 640
660 IF I88 > J88 THEN 680
      W88 = P9(I88): P9(I88) = P9(J88): P9(J88) = W88: I88 = I88 + 1: J88 = J88 - 1
680 IF I88 <= J88 THEN 620
      REM BOOKKEEPING PHASE
      IF J88 - L88 >= R88 - I88 THEN 740
      IF I88 >= R88 THEN 730
      H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GOTO 770
740 IF L88 >= J88 THEN 760
      H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 IF L88 < R88 THEN 590
780 IF K88 <= 1 THEN 800
      K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GOTO 560
800 RETURN

```

```
900 REM EXCHANGE SORT
    FOR I88 = L88 TO R88 - 1: K1 = I88
    FOR J88 = I88 + 1 TO R88: IF P9(J88) >= P9(K1) THEN 940
    K1 = J88
940 NEXT J88
    IF K1 = I88 THEN 970
    W88 = P9(I88): P9(I88) = P9(K1): P9(K1) = W88
970 NEXT I88
RETURN
```

LISTAGEM 18:

```

REM PROGRAM FILENAME BOOTHIE3.BAS
CLS : DEFDBL A-Z
K = 4
DIM D(K, 5), H(K, 5), R(K, 5), N(K, 5), P(K, 5), X(K, 5), H88(25), P9(500), RDOM(K), PS(K)
DIM FIT(K), FIS(K), FST(K), FITS(500, K), FISS(500, K), FSTS(500, K), HTN(10)
DIM SHOMOS(K), NG(K), SP(K), PP(K), S(K, K, K), SFITT(500), SFIST(500), SFSTT(500)
OPEN "c:\temp\result2.txt" FOR OUTPUT AS #99
FILENAME$(1) = "c:\temp\DAT1_Fit.DAT"
FILENAME$(2) = "c:\temp\DAT1_Fst.DAT"
FILENAME$(3) = "c:\temp\DAT1_Fis.DAT"

INPUT "NUM. OF LOCUS = ", NL
INPUT "NUM. OF SUBPOPULATIONS = ", NP

DEF FNFPARCIAL
FOR J = 1 TO NL
  NHARM(J) = 0: H0(J) = 0: AC(J) = 0: AC3(J) = 0
  FOR K = 1 TO 2
    AC1(J, K) = 0
  NEXT K, J

FOR J = 1 TO NL
  FOR L = 1 TO NP
    NHARM(J) = NHARM(J) + 1 / N(J, L)
  NEXT L
  NHARM(J) = NP / NHARM(J)
NEXT J

FOR J = 1 TO NL
  FOR L = 1 TO NP
    PIJ = (2 * DF(J, L) + HF(J, L)) / (2 * N(J, L))
    AC1(J, 1) = AC1(J, 1) + PIJ / NP
    AC1(J, 2) = AC1(J, 2) + (1 - PIJ) / NP
    AC(J) = AC(J) + (PIJ ^ 2 + (1 - PIJ) ^ 2) / NP
    AC2(J, L) = PIJ ^ 2 + (1 - PIJ) ^ 2
    IF N(J, L) <> 0 THEN H0(J) = H0(J) + (RF(J, L) + DF(J, L)) / (NP * N(J, L))
  NEXT L, J

FOR J = 1 TO NL
  FOR K = 1 TO 2
    AC3(J) = AC3(J) + AC1(J, K) ^ 2
  NEXT K, J

FOR L = 1 TO NL
  H0(L) = 1 - H0(L)
NEXT L

FOR L = 1 TO NL
  HS(L) = NHARM(L) / (NHARM(L) - 1) * (1 - AC(L) - H0(L) / (2 * NHARM(L)))
  IF HS(L) <> 0 THEN FFIS(L) = 1 - H0(L) / HS(L)
  HTN(L) = 1 - AC3(L) + HS(L) / NHARM(L) - H0(L) / (2 * NHARM(L) * NP)
  IF HTN(L) <> 0 THEN FFIT(L) = 1 - H0(L) / HTN(L): FFST(L) = 1 - HS(L) / HTN(L)
  FOR J = 1 TO NP
    IF AC2(L, J) <> 1 THEN FF(L, J) = 1 - (1 - (DF(L, J) + RF(L, J)) / N(L, J)) / (1 - AC2(L, J))
  NEXT J
NEXT L
END DEF

FOR J = 1 TO NL
  PRINT
  FOR I = 1 TO NP

```

```

PRINT USING "ALLELLE ## - SUBPOP. ##: "; J; I;
INPUT "N(AA), N(Aa), N(aa) = "; D(J, I), H(J, I), R(J, I)
N(J, I) = D(J, I) + H(J, I) + R(J, I)
NT(J) = NT(J) + N(J, I): DT(J) = DT(J) + D(J, I): HT(J) = HT(J) + H(J, I)
RT(J) = RT(J) + R(J, I)
NEXT I
NEXT J

DEF FNFTOTAL

FOR M = 1 TO NL
FOR J = 1 TO NP
MSP(M, J) = 0: MSI(M, J) = 0: MSG(M, J) = 0
NC(M, J) = 0
NEXT J, M
S1 = 0: S2 = 0: S3 = 0

FOR K = 1 TO NL
FOR M = 1 TO 2
FOR J = 1 TO NP
MSP(K, M) = MSP(K, M) + N(K, J) * (PPP(K, M) - PM(M)) ^ 2
MSI(K, M) = MSI(K, M) + N(K, J) * (PPP(K, M) + DI(K, J, M) / N(K, J) - 2 * PPP(K, M)^2)
MSG(K, M) = MSG(K, M) + N(K, J) * (PPP(K, M) - DI(K, J, M) / N(K, J))
NC(K, M) = NC(K, M) + N(K, J) ^ 2
NEXT J
MSP(K, M) = 2 * MSP(K, M) / (NP - 1)
MSI(K, M) = MSI(K, M) / (NT(K) - NP)
MSG(K, M) = MSG(K, M) / NT(K)
NC(K, M) = NT(K) - NC(K, M) / NT(K) / (NP - 1)
S3 = S3 + 2 * NC(K, M) * MSG(K, M): S1 = S1 + MSP(K, M) - MSI(K, M)
S2 = S2 + MSP(K, M) + (NC(K, M) - 1) * MSI(K, M) + NC(K, M) * MSG(K, M)
NEXT M, K
FITT = 1 - S3 / S2: FSTT = S1 / S2: FIST = (FITT - FSTT) / (1 - FSTT)
END DEF

FOR J = 1 TO NL
FOR M = 1 TO NP
DI(J, M, 1) = D(J, M): DI(J, M, 2) = R(J, M)
NEXT M
PPP(J, 1) = (2 * DT(J) + HT(J)) / (2 * NT(J))
PPP(J, 2) = 1 - PPP(J, 1)
PM(1) = PM(1) + PPP(J, 1) / NL
PM(2) = PM(2) + PPP(J, 2) / NL
NEXT J
A = FNFTOTAL
FSTI = FSTT: FISI = FIST: FITI = FITT

PRINT #99, : PRINT #99,
PRINT #99, "ESTIMATES BASED ON SAMPLE NUMBERS": PRINT #99,
PRINT #99, "LOCUS SUBPOP. N(AA) N(Aa) N(aa) N"
PRINT #99, "-----": CLS

FOR J = 1 TO NL
PRINT #99,
FOR I = 1 TO NP
PRINT #99, USING "#### "; J; I; D(J, I); H(J, I); R(J, I); N(J, I)
NEXT I
PRINT #99, "-----"
PRINT #99, " total "; : PRINT #99, USING "#### "; DT(J); HT(J); RT(J);
NT(J): PRINT #99,
NEXT J

FOR I = 1 TO NL
FOR J = 1 TO NP
DF(I, J) = D(I, J): HF(I, J) = H(I, J): RF(I, J) = R(I, J)
NEXT J, I

FOR J = 1 TO NL
FOR I = 1 TO NP
NG(J) = NG(J) + 2 * D(J, I) + H(J, I)

```

```

    SHOMO(J) = SHOMO(J) + D(J, I) + R(J, I)
NEXT I
NG(J) = NG(J) / (2 * NT(J)): NGF(J) = NG(J)
SHOMO(J) = SHOMO(J) / NT(J): FSHOMO(J) = FSHOMO(J)
NEXT J

A = FNFPARCIAL

FOR J = 1 TO NL
    FIT(J) = FFIT(J): FST(J) = FFST(J): FIS(J) = FFIS(J)
    FOR L = 1 TO NP
        F(J, L) = FF(J, L)
    NEXT L, J

FOR J = 1 TO NL
    FOR I = 1 TO NP
        P(J, I) = (2 * D(J, I) + H(J, I)) / (2 * N(J, I))
        X(J, I) = N(J, I) / NT(J): PU(J) = PU(J) + X(J, I) * P(J, I)
        PRINT #99, "P(" + STR$(J) + "," + STR$(I) + ") = "; : PRINT #99, USING "#.####"; P(J, I)
        PRINT #99, "F(" + STR$(J) + "," + STR$(I) + ") = "; : PRINT #99, USING "#.####"; F(J, I)
    NEXT I

PRINT #99, "P(" + STR$(J) + ") = "; : PRINT #99, USING "#.####"; PU(J)
PRINT #99, "FIT(" + STR$(J) + ") = "; : PRINT #99, USING "#.####"; FIT(J)
PRINT #99, "FST(" + STR$(J) + ") = "; : PRINT #99, USING "#.####"; FST(J)
PRINT #99, "FIS(" + STR$(J) + ") = "; : PRINT #99, USING "#.####"; FIS(J): PRINT #99,
NEXT J

PRINT #99, USING "TOTAL FIS = #.####"; FIST
PRINT #99, USING "TOTAL FST = #.####"; FSTT
PRINT #99, USING "TOTAL FIT = #.####"; FITT

TIMEWA$ = TIME$
TIMEWAS = VAL(MID$(TIMEWA$, 1, 2)) * 3600 + VAL(MID$(TIMEWA$, 4, 2)) * 60
TIMEWAS = TIMEWAS + VAL(MID$(TIMEWA$, 7, 2))

FOR NSIM = 50 TO 450 STEP 100
    SUMFITT = 0: SUMFIST = 0: SUMFSTT = 0

    PRINT #99, : PRINT #99,
    PRINT #99, "AVERAGE ESTIMATES BASED ON" + STR$(NSIM) + " SIMULATIONS"
    DFT = 0
    FOR I = 1 TO NSIM
        FOR J = 1 TO NL
            FOR K = 1 TO NP
                SPP(J, K) = 0: D1(J, K) = 0: H1(J, K) = 0: R1(J, K) = 0
            NEXT K, J
        PTOT = 0

        FOR J = 1 TO NL
            FOR I9 = 1 TO NP
                SSP = 0: FOR K = 1 TO 3: SP(K) = 0: NEXT K
                PP(1) = D(J, I9) / N(J, I9): PP(2) = H(J, I9) / N(J, I9): PP(3) = R(J, I9) / N(J, I9)
                FOR K = 1 TO 3
                    SSP = SSP + PP(K): SP(K) = SP(K) + SSP
                NEXT K

                LOCATE 10: PRINT USING "SUBPOPULATION ##: "; I9;
                PRINT USING "SIMULATION ####"; I
                FOR INDIV = 1 TO N(J, I9)
                    A = RND(1)
                    IF A <= SP(1) THEN D1(J, I9) = D1(J, I9) + 1
                    IF A > SP(1) AND A <= SP(2) THEN H1(J, I9) = H1(J, I9) + 1
                    IF A > SP(2) AND A <= SP(3) THEN R1(J, I9) = R1(J, I9) + 1
                NEXT INDIV
                P1 = 2 * D1(J, I9) + H1(J, I9): Q1 = H1(J, I9) + 2 * R1(J, I9)
                SUMP1(J, I9) = SUMP1(J, I9) + P1 / (2 * N(J, I9))
                SPP(J, 1) = SPP(J, 1) + P1 / (2 * N(J, I9)) / NP
            NEXT I9, J

```

```

FOR J = 1 TO NL
  SPP(J, 2) = 1 - SPP(J, 1)
NEXT J

FOR J = 1 TO 2
  PM(J) = 0
NEXT J

FOR J = 1 TO NL
  FOR M = 1 TO 2
    DI(J, M, 1) = D1(J, M): DI(J, M, 2) = R1(J, M)
    PPP(J, M) = SPP(J, M)
    PM(M) = PM(M) + SPP(J, M) / NL
  NEXT M, J

A = FNFTOTAL
SFITT(I) = FITT: SFIST(I) = FIST: SFSTT(I) = FSTT
SUMFIST = SUMFIST + SFIST(I): SUMFSTT = SUMFSTT + SFSTT(I): SUMFITT = SUMFITT + SFITT(I)

FOR J = 1 TO NL
  AC1S(J) = 0: NGS(J) = 0: SHOMOS(J) = 0: HOS(J) = 0
NEXT J

FOR J = 1 TO NL
  FOR K = 1 TO NP
    DF(J, K) = D1(J, K): HF(J, K) = H1(J, K): RF(J, K) = R1(J, K)
    NGS(J) = NGS(J) + 2 * D1(J, K) + H1(J, K)
    SHOMOS(J) = SHOMOS(J) + D1(J, K) + R1(J, K)
  NEXT K
  NGS(J) = NGS(J) / (2 * NT(J)): NGF(J) = NGS(J)
  SHOMOS(J) = SHOMOS(J) / NT(J): FSHOMO(J) = SHOMOS(J)
NEXT J

A = FNFPARCIAL

FOR J = 1 TO NL
  FITS(I, J) = FFIT(J): FSTS(I, J) = FFST(J): FISS(I, J) = FFIS(J)
  FOR L = 1 TO NP
    FS(J, L) = FF(J, L)
    SUMF(J, L) = SUMF(J, L) + FS(J, L)
  NEXT L, J
NEXT I

FOR J = 1 TO NL
  FOR I9 = 1 TO NP
    PRINT #99, "p(" + STR$(J) + "," + STR$(I9) + ") = "; : PRINT #99, USING "#.####";
    SUMP1(J, I9) / NSIM;
    PRINT #99, USING " DIF.= #.####"; ABS(P(J, I9) - SUMP1(J, I9) / NSIM)
    DFT = DFT + ABS(P(J, I9) - SUMP1(J, I9) / NSIM)
    PRINT #99, "F(" + STR$(J) + "," + STR$(I9) + ") = ";
    PRINT #99, USING "#.####"; SUMF(J, I9) / NSIM;
    PRINT #99, USING " DIF.= #.####"; ABS(F(J, I9) - SUMF(J, I9) / NSIM)
    DFT = DFT + ABS(F(J, I9) - SUMF(J, I9) / NSIM)
    SUMF(J, I9) = 0: SUMP1(J, I9) = 0
  NEXT I9:
  I4 = 1
  NEXT J
  PRINT #99, USING "TOTAL FIT =#.####"; SUMFITT / NSIM;
  PRINT #99, USING " DIF.= #.####"; ABS(FITT - SUMFITT / NSIM)
  DFT = DFT + ABS(FITT - SUMFITT / NSIM)
  PRINT #99, USING "TOTAL FIS =#.####"; SUMFIST / NSIM;
  PRINT #99, USING " DIF.= #.####"; ABS(FIST - SUMFIST / NSIM)
  DFT = DFT + ABS(FIST - SUMFIST / NSIM)
  PRINT #99, USING "TOTAL FST =#.####"; SUMFSTT / NSIM;
  PRINT #99, USING " DIF.= #.####"; ABS(FSTI - SUMFSTT / NSIM)
  DFT = DFT + ABS(FSTI - SUMFSTT / NSIM)

PRINT #99, USING "DIF.TOT= #.#### "; DFT: PRINT #99,
IF NSIM / 2 = NSIM \ 2 THEN

```



```

M = 1
ELSE
M = 0
END IF
NAME$(1) = "FIT": NAME$(2) = "FIS": NAME$(3) = "FST"

FOR J = 1 TO NL
FOR K = 1 TO NSIM
P9(K) = FITS(K, J)
NEXT K
GOSUB SORTVALUES
S(J, 1, 1) = (P9(NSIM / 2) + P9(NSIM / 2 + 1)) / 2
S(J, 1, 2) = P9(INT(.975 * NSIM))
S(J, 1, 3) = P9(INT(1 + .025 * NSIM))
NEXT J

FOR J = 1 TO NL
FOR K = 1 TO NSIM
P9(K) = FISS(K, J)
NEXT K
GOSUB SORTVALUES
S(J, 2, 1) = (P9(NSIM / 2) + P9(NSIM / 2 + 1)) / 2
S(J, 2, 2) = P9(INT(.975 * NSIM))
S(J, 2, 3) = P9(INT(1 + .025 * NSIM))
NEXT J

FOR J = 1 TO NL
FOR K = 1 TO NSIM
P9(K) = FSTS(K, J)
NEXT K
GOSUB SORTVALUES
S(J, 3, 1) = (P9(NSIM / 2) + P9(NSIM / 2 + 1)) / 2
S(J, 3, 2) = P9(INT(.975 * NSIM))
S(J, 3, 3) = P9(INT(1 + .025 * NSIM))
NEXT J

FOR K = 1 TO 3
FOR J = 1 TO NL
PRINT #99, NAME$(K); : PRINT #99, "(" + STR$(J) + ") = ";
PRINT #99, USING "#.### "; S(J, K, 1);
PRINT #99, USING "{#.###"; S(J, K, 3); : PRINT #99, ",";
PRINT #99, USING "#.###"; S(J, K, 2)
NEXT J, K

FOR I = 1 TO NSIM
P9(I) = SFITT(I)
NEXT I
GOSUB SORTVALUES
PRINT #99, "TOTAL FIT =";
IF M = 1 THEN
PRINT #99, USING "#.### "; (P9(NSIM / 2) + P9(NSIM / 2 + 1)) / 2;
ELSE
PRINT #99, USING "#.### "; P9(INT(NSIM / 2 + 1));
END IF
PRINT #99, USING "{#.###"; P9(INT(1 + .025 * NSIM)); : PRINT #99, ",";
PRINT #99, USING "#.###"; P9(INT(.975 * NSIM))

FOR I = 1 TO NSIM
P9(I) = SFIST(I)
NEXT I
GOSUB SORTVALUES
PRINT #99, "TOTAL FIS =";
IF M = 1 THEN
PRINT #99, USING "#.### "; (P9(NSIM / 2) + P9(NSIM / 2 + 1)) / 2;
ELSE
PRINT #99, USING "#.### "; P9(INT(NSIM / 2 + 1));
END IF
PRINT #99, USING "{#.###"; P9(INT(1 + .025 * NSIM)); : PRINT #99, ",";
PRINT #99, USING "#.###"; P9(INT(.975 * NSIM))

FOR I = 1 TO NSIM

```

```

    P9(I) = SFSTT(I)
NEXT I
GOSUB SORTVALUES
PRINT #99, "TOTAL FST =";
IF M = 1 THEN
    PRINT #99, USING "#.### "; (P9(NSIM / 2) + P9(NSIM / 2 + 1)) / 2;
ELSE
    PRINT #99, USING "#.### "; P9(INT(NSIM / 2 + 1));
END IF
PRINT #99, USING "{#.###"; P9(INT(1 + .025 * NSIM)); : PRINT #99, ", ";
PRINT #99, USING "#.###"; P9(INT(.975 * NSIM))

NEXT NSIM
CLOSE #99
TIMEIS = TIME$
TIMEIS = VAL(MID$(TIMEIS, 1, 2)) * 3600 + VAL(MID$(TIMEIS, 4, 2)) * 60
TIMEIS = TIMEIS + VAL(MID$(TIMEIS, 7, 2))
LOCATE 15: PRINT USING "TOTAL PROCESSING TIME = ##### SECONDS"; TIMEIS - TIMEWAS
END
SORTVALUES:
D88 = 10: K88 = 1: L88 = 1: R88 = NSIM
560 REM EXCHANGE SORT IF BLOCKSIZE <= D88
    IF R88 - L88 + 1 > D88 THEN 590
    GOSUB 900: GOTO 780
590 I88 = L88: J88 = R88: W88 = RND(1) * (R88 - L88) + .5
    W88 = INT(W88 + L88): F88 = P9(W88)
    REM PARTITION PHASE
620 IF P9(I88) >= F88 THEN 640
    I88 = I88 + 1: GOTO 620
640 IF P9(J88) <= F88 THEN 660
    J88 = J88 - 1: GOTO 640
660 IF I88 > J88 THEN 680
    W88 = P9(I88): P9(I88) = P9(J88): P9(J88) = W88: I88 = I88 + 1: J88 = J88 - 1
680 IF I88 <= J88 THEN 620
    REM BOOKKEEPING PHASE
    IF J88 - L88 >= R88 - I88 THEN 740
    IF I88 >= R88 THEN 730
    H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GOTO 770
740 IF L88 >= J88 THEN 760
    H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 IF L88 < R88 THEN 590
780 IF K88 <= 1 THEN 800
    K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GOTO 560
800 RETURN
900 REM EXCHANGE SORT
    FOR I88 = L88 TO R88 - 1: K1 = I88
    FOR J88 = I88 + 1 TO R88: IF P9(J88) >= P9(K1) THEN 940
    K1 = J88
940 NEXT J88
    IF K1 = I88 THEN 970
    W88 = P9(I88): P9(I88) = P9(K1): P9(K1) = W88
970 NEXT I88
RETURN

```

LISTAGEM 19 (VISUAL BASIC 6.0)

REM program f-jack.vbp

```
Public NP, I, LIM, L88, R88
Dim ORD(1000, 50), H88(1000)
Private Sub Command1_Click()
Frame1.Visible = False
Text1.Visible = False
Text2.Visible = True
Command2.Visible = True
Command1.Visible = False
GRID1.Visible = True
NP = Val(Text1.Text)
```

```
GRID1.Cols = 5
GRID1.Rows = NP + 2
```

```
For I = 1 To NP
    GRID1.FontBold = True
    GRID1.TextArray(fgi(I, 0)) = "POP " & I
Next I
    GRID1.TextArray(fgi(0, 1)) = "D"
    GRID1.TextArray(fgi(0, 2)) = "H"
    GRID1.TextArray(fgi(0, 3)) = "R"
```

```
GRID1.TextArray(fgi(0, 4)) = "TOT"
GRID1.TextArray(fgi(I, 0)) = "TOT"
```

```
Text2 = " "
```

```
GRID1.AllowBigSelection = False
```

```
End Sub
```

```
Private Sub Command2_Click()
Dim SIMPT(1000), SIMFIT(1000), SIMF(1000, 10), SIMFST(1000), FMIN(10), FMAX(10)
Dim SIMFIS(1000), SUMF(1000), D(10), H(10), R(10), TOT(3), N(10), P(10), jack(50)
Dim EXPF(10), AL(10, 2), SIMH(10), SIMR(10), SIMD(10), SIMP(10), MF(10), EXPF1(10)
Dim SHj(10), SRj(10), SDj(10), SPj(10)
Dim SPTj(1000), SIMFITj(1000), SIMFj(1000, 10), SIMFSTj(1000)
Dim SIMFISj(1000), SUMFj(1000)
Command2.Visible = False
```

```
For I = 1 To NP
    D(I) = Val(GRID1.TextArray(fgi(I, 1)))
    H(I) = Val(GRID1.TextArray(fgi(I, 2)))
    R(I) = Val(GRID1.TextArray(fgi(I, 3)))
```

```

TOT(1) = TOT(1) + D(I) : TOT(2) = TOT(2) + H(I) : TOT(3) = TOT(3) + R(I)
N(I) = D(I) + H(I) + R(I)
P(I) = (2 * D(I) + H(I)) / (2 * N(I))
NT = NT + N(I)
Next I

For I = 1 To NP
    jack(I) = Int(0.975 * N(I))
Next I

For j = 1 To NP
    NTJ = NTJ + jack(j)
Next j

PT = (2 * TOT(1) + TOT(2)) / (2 * NT)
For I = 1 To 3
    GRID1.TextArray(fgi(NP + 1, I)) = TOT(I)
Next I

GRID1.TextArray(fgi(I - 1, 4)) = NT

For I = 1 To NP
    GRID1.TextArray(fgi(I, 4)) = N(I)
Next I

GRID1.Refresh

Picture2.Visible = True
Picture3.Visible = True
Picture4.Visible = True
Label4.Visible = True
Label5.Visible = True
Label6.Visible = True
Label6.Refresh
Label4.Refresh
Label5.Refresh

LIM = 1000
TI = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TI = TI + Val(Mid$(Time$, 7, 2))

EXPFIT = 1 - TOT(2) / NT * 1 / (2 * PT * (1 - PT))
For I = 1 To NP
    EXPF(I) = 1 - H(I) / N(I) * 1 / (2 * P(I) * (1 - P(I)))
    EXPXIPI = EXPXIPI + N(I) / NT * P(I) ^ 2
Next I

EXPFAST = (EXPXIPI - PT ^ 2) / (PT * (1 - PT))
EXPFAST = (EXPFIT - EXPFAST) / (1 - EXPFAST)

```

```

For I = 1 To NP
  For j = 1 To 2
    AL(I, j) = D(I) + (j - 1) * H(I): AL(I, j) = AL(I, j) / N(I)
  Next j, I

TA = TI
For I = 1 To LIM
  TF = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
  TF = TF + Val(Mid$(Time$, 7, 2))

If TF - TA > 0.01 Then
  Picture4.Cls
  Picture4.Print I
  TA = TF
End If
TT = Format(TF - TI, "00")
Picture2.Cls
Picture2.Print TT
XX1 = I * 2175 / LIM
Picture3.Line (0, 0)-(XX1, 435), , B

For j = 1 To NP
  SIMD(j) = 0: SIMH(j) = 0: SIMR(j) = 0
  SDj(j) = 0: SHj(j) = 0: SRj(j) = 0
Next j

For j = 1 To NP
  For l = 1 To N(j)
    A = Rnd
    If l <= jack(j) Then
      For k = 1 To 2
        If A <= AL(j, k) Then
          If k = 1 Then SDj(j) = SDj(j) + 1 Else SHj(j) = SHj(j) + 1
          k = 4
        End If
      Next k
      If k = 3 Then SRj(j) = SRj(j) + 1
    End If
    For k = 1 To 2
      If A <= AL(j, k) Then
        If k = 1 Then SIMD(j) = SIMD(j) + 1 Else SIMH(j) = SIMH(j) + 1
        k = 4
      End If
    Next k
    If k = 3 Then SIMR(j) = SIMR(j) + 1
  Next l
Next j

SIMHT = 0: SIMDT = 0: SDTj = 0: SHTj = 0
For k = 1 To NP
  SIMHT = SIMHT + SIMH(k)
  SIMDT = SIMDT + SIMD(k)

```

```

    SIMP(k) = (2 * SIMD(k) + SIMH(k)) / (2 * N(k))
    SHTj = SHTj + SHj(k)
    SDTj = SDTj + SDj(k)
    SPj(k) = (2 * SDj(k) + SHj(k)) / (2 * jack(k))
Next k

SIMPT(I) = (2 * SIMDT + SIMHT) / (2 * NT)
SIMFIT(I) = 1 - SIMHT / NT * 1 / (2 * SIMPT(I) * (1 - SIMPT(I)))
SPTj(I) = (2 * SDTj + SHTj) / (2 * NTJ)
SIMFITj(I) = 1 - SHTj / NTJ * 1 / (2 * SPTj(I) * (1 - SPTj(I)))
ORD(I, 1) = SIMFIT(I)
SIMXIPI = 0: SIMXIPIj = 0
For k = 1 To NP
    If N(k) <> 0 And SIMP(k) <> 1 And SIMP(k) <> 0 Then
        SIMF(I, k) = 1 - SIMH(k) / N(k) * 1 / (2 * SIMP(k) * (1 - SIMP(k)))
    Else
        SIMF(I, k) = 0
    End If

    If jack(k) <> 0 And SPj(k) <> 1 And SPj(k) <> 0 Then
        SIMFj(I, k) = 1 - SHj(k) / jack(k) * 1 / (2 * SPj(k) * (1 - SPj(k)))
    Else
        SIMFj(I, k) = 0
    End If

    ORD(I, 1 + k) = SIMF(I, k)
    SIMXIPI = SIMXIPI + N(k) / NT * SIMP(k) ^ 2
    SIMXIPIj = SIMXIPIj + jack(k) / NTJ * SPj(k) ^ 2
Next k
SIMFST(I) = (SIMXIPI - SIMPT(I) ^ 2) / (SIMPT(I) * (1 - SIMPT(I)))
SIMFSTj(I) = (SIMXIPIj - SPTj(I) ^ 2) / (SPTj(I) * (1 - SPTj(I)))
ORD(I, 1 + k) = SIMFST(I)
SIMFISj(I) = (SIMFITj(I) - SIMFSTj(I)) / (1 - SIMFSTj(I))
SIMFIS(I) = (SIMFIT(I) - SIMFST(I)) / (1 - SIMFST(I))

ORD(I, 2 + k) = SIMFIS(I)
ORD(I, 3 + k) = SIMFITj(I)
For j = 1 To NP
    ORD(I, 3 + k + j) = SIMFj(I, j)
Next j
ORD(I, 3 + k + j) = SIMFSTj(I)
ORD(I, 4 + k + j) = SIMFISj(I)
Next I

For I = 1 To LIM
    SUMFIT = SUMFIT + SIMFIT(I)
    SUMFITj = SUMFITj + SIMFITj(I)
    For j = 1 To NP
        SUMF(j) = SUMF(j) + SIMF(I, j)
        SUMFj(j) = SUMFj(j) + SIMFj(I, j)
    Next j
    SUMFST = SUMFST + SIMFST(I)
    SUMFIS = SUMFIS + SIMFIS(I)

```

```

SUMFSTj = SUMFSTj + SIMFSTj(I)
SUMFISj = SUMFISj + SIMFISj(I)
Next I

Picture4.Cls
Picture4.Print LIM

For I = 1 To 4 + k + j
    A = SORTVALUES
Next I

B = Int(LIM * 0.025)

FITMIN = Format(ORD(B + 1, 1), "#0.0000")
FITMAX = Format(ORD(LIM - B, 1), "#0.0000")
Text3.Text = Text3.Text & "FIT=(" & FITMIN
Text3.Text = Text3.Text & "," & FITMAX & ")" & vbCrLf
For I = 1 To NP
    FMIN(I) = Format(ORD(B + 1, 1 + I), "#0.0000")
    FMAX(I) = Format(ORD(LIM - B, 1 + I), "#0.0000")
    Text3.Text = Text3.Text & "F(" & I & ")=" & FMIN(I)
    Text3.Text = Text3.Text & "," & FMAX(I) & ")" & vbCrLf
Next I
FSTMIN = Format(ORD(B + 1, 1 + I), "#0.0000")
FSTMAX = Format(ORD(LIM - B, 1 + I), "#0.0000")
Text3.Text = Text3.Text & "FST=(" & FSTMIN
Text3.Text = Text3.Text & "," & FSTMAX & ")" & vbCrLf
FISMIN = Format(ORD(B + 1, 2 + I), "#0.0000")
FISMAX = Format(ORD(LIM - B, 2 + I), "#0.0000")
Text3.Text = Text3.Text & "FIS=(" & FISMIN
Text3.Text = Text3.Text & "," & FISMAX & ")" & vbCrLf & vbCrLf

MFIT = Format(SUMFIT / LIM, "#0.0000")
Text3.Text = Text3.Text & "MFIT=" & MFIT & vbCrLf
EXPFIT1 = Format(EXPFIT, "#0.0000")
Text3.Text = Text3.Text & "EXPFIT=" & EXPFIT1 & vbCrLf & vbCrLf
For I = 1 To NP
    MF(I) = Format(SUMF(I) / LIM, "#0.0000")
    Text3.Text = Text3.Text & "MF(" & I & ")=" & MF(I) & vbCrLf
    EXPF1(I) = Format(EXPF(I), "#0.0000")
    Text3.Text = Text3.Text & "EXPF(" & I & ")=" & EXPF1(I) & vbCrLf & vbCrLf
Next I
MFST = Format(SUMFST / LIM, "#0.0000")
Text3.Text = Text3.Text & "MFST=" & MFST & vbCrLf
EXPFST1 = Format(EXPFST, "#0.0000")
Text3.Text = Text3.Text & "EXPFST=" & EXPFST1 & vbCrLf & vbCrLf
MFIS = Format(SUMFIS / LIM, "#0.0000")
Text3.Text = Text3.Text & "MFIS=" & MFIS & vbCrLf
EXPFIS1 = Format(EXPFIS, "#0.0000")
Text3.Text = Text3.Text & "EXPFIS=" & EXPFIS1 & vbCrLf & vbCrLf

Text3.Text = Text3.Text & vbCrLf & vbCrLf

FITMIN = Format(ORD(B + 1, 3 + k), "#0.0000")

```

```

FITMAX = Format(ORD(LIM - B, 3 + k), "#0.0000")
Text3.Text = Text3.Text & "FIT=(" & FITMIN
Text3.Text = Text3.Text & "," & FITMAX & ")" & vbCrLf
For I = 1 To NP
    FMIN(I) = Format(ORD(B + 1, 3 + k + I), "#0.0000")
    FMAX(I) = Format(ORD(LIM - B, 3 + k + I), "#0.0000")
    Text3.Text = Text3.Text & "F(" & I & ")=(" & FMIN(I)
    Text3.Text = Text3.Text & "," & FMAX(I) & ")" & vbCrLf
Next I
FSTMIN = Format(ORD(B + 1, 3 + k + I), "#0.0000")
FSTMAX = Format(ORD(LIM - B, 3 + k + I), "#0.0000")
Text3.Text = Text3.Text & "FST=(" & FSTMIN
Text3.Text = Text3.Text & "," & FSTMAX & ")" & vbCrLf
FISMIN = Format(ORD(B + 1, 4 + k + I), "#0.0000")
FISMAX = Format(ORD(LIM - B, 4 + k + I), "#0.0000")
Text3.Text = Text3.Text & "FIS=(" & FISMIN
Text3.Text = Text3.Text & "," & FISMAX & ")" & vbCrLf & vbCrLf

TF = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TF = TF + Val(Mid$(Time$, 7, 2))

MFIT = Format(SUMFITj / LIM, "#0.0000")
Text3.Text = Text3.Text & "MFIT=" & MFIT & vbCrLf
EXPFIT1 = Format(EXPFIT, "#0.0000")
Text3.Text = Text3.Text & "EXPFIT=" & EXPFIT1 & vbCrLf & vbCrLf
For I = 1 To NP
    MF(I) = Format(SUMFj(I) / LIM, "#0.0000")
    Text3.Text = Text3.Text & "MF(" & I & ")=" & MF(I) & vbCrLf
    EXPF1(I) = Format(EXPF(I), "#0.0000")
    Text3.Text = Text3.Text & "EXPF(" & I & ")=" & EXPF1(I) & vbCrLf & vbCrLf
Next I
MFST = Format(SUMFSTj / LIM, "#0.0000")
Text3.Text = Text3.Text & "MFST=" & MFST & vbCrLf
EXPFST1 = Format(EXPFST, "#0.0000")
Text3.Text = Text3.Text & "EXPFST=" & EXPFST1 & vbCrLf & vbCrLf
MFIS = Format(SUMFISj / LIM, "#0.0000")
Text3.Text = Text3.Text & "MFIS=" & MFIS & vbCrLf
EXPFIS1 = Format(EXPFIS, "#0.0000")
Text3.Text = Text3.Text & "EXPFIS=" & EXPFIS1 & vbCrLf & vbCrLf
Picture2.Cls
Picture2.Print TF - TI

End Sub

Function SORTVALUES()
D88 = 10: K88 = 1: L88 = 1: R88 = LIM
560 Rem EXCHANGE SORT IF BLOCKSIZE <= D88
    If R88 - L88 + 1 > D88 Then GoTo 590
    A = EXCHANGESORT: GoTo 780
590 I88 = L88: J88 = R88: W88 = Rnd(1) * (R88 - L88) + 0.5
    W88 = Int(W88 + L88): F88 = ORD(W88, I)
    Rem PARTITION PHASE
620 If ORD(I88, I) >= F88 Then GoTo 640
    I88 = I88 + 1: GoTo 620
640 If ORD(J88, I) <= F88 Then GoTo 660

```



```

        J88 = J88 - 1: GoTo 640
660 If I88 > J88 Then GoTo 680
        W88 = ORD(I88, I): ORD(I88, I) = ORD(J88, I): ORD(J88, I) = W88: I88 = I88 + 1: J88 =
J88 - 1
680 If I88 <= J88 Then GoTo 620
        Rem BOOKKEEPING PHASE
        If J88 - L88 >= R88 - I88 Then GoTo 740
        If I88 >= R88 Then GoTo 730
        H88(K88) = I88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GoTo 770
740 If L88 >= J88 Then GoTo 760
        H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = I88
770 If L88 < R88 Then GoTo 590
780 If K88 <= 1 Then GoTo 800
        K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GoTo 560
800
End Function

Function EXCHANGESORT()
    For I88 = L88 To R88 - 1: K1 = I88
        For J88 = I88 + 1 To R88: If ORD(J88, I) >= ORD(K1, I) Then GoTo 940
        K1 = J88
940 Next J88
        If K1 = I88 Then GoTo 970
        W88 = ORD(I88, I): ORD(I88, I) = ORD(K1, I): ORD(K1, I) = W88
970 Next I88

End Function

Sub GRID1_KEYPRESS(A As Integer)
MSHFLEXGRIDEDIT GRID1, Text2, A
End Sub

Sub GRID1_DBLCLICK()
MSHFLEXGRIDEDIT GRID1, Text2, 32
End Sub

Sub MSHFLEXGRIDEDIT(MSHFLEXGRID As Control, _
EDT As Control, A As Integer)

If MSHFLEXGRID.Row = NP + 1 Or MSHFLEXGRID.Col = 4 _
Then Exit Sub

Select Case A
Case 0 To 32
    EDT = MSHFLEXGRID
    EDT.SelStart = 1000

Case Else
    EDT = Chr(A)
    EDT.SelStart = 1
End Select

EDT.Move MSHFLEXGRID.Left + MSHFLEXGRID.CellLeft, _
    MSHFLEXGRID.Top + MSHFLEXGRID.CellTop, _

```

```

MSHFLEXGRID.CellWidth - 8, _
MSHFLEXGRID.CellHeight - 8
EDT.Visible = True

EDT.SetFocus
End Sub
Sub GRID1_GOTFOCUS()
If Text2.Visible = False Then Exit Sub
GRID1 = Text2
Text2.Visible = False
End Sub
Sub GRID1_LEAVECELL()
If Text2.Visible = False Then Exit Sub
GRID1 = Text2
Text2.Visible = False
End Sub

Function fgi(R, s)
fgi = s + GRID1.Cols * R
End Function

```

LISTAGEM 20 (VISUAL BASIC 6.0)

REM Hardy - Weinberg Equilibrium Testing
REM CODE

MODULE 1

Global ww, nal, cte2, cte3, cte4, cte5
Global N(21, 21), cte10, pe(3, 101), esp, jack(1001, 21, 21), Pjack(1001, 21)
Global qq, lim, totn, P(21), XC, YC

FORM 1

```
Private Sub Command1_Click()
Command1.Visible = False
Form1.Hide
Form2.Show
End Sub
```

```
Private Sub Command2_Click()
MsgBox " Copyright 2006 by Fernando A. Bautzer Santos & Paulo A. Otto. " & nl _
& vbCrLf & nl _
& vbCrLf & nl _
& " Departamento de Genética e Biologia Evolutiva " & nl _
& " Universidade de São Paulo. " & nl _
& vbCrLf & nl _
& vbCrLf & nl _
& " POBox 11461 " & nl _
& " 05422-970; São Paulo SP, Brazil. " & nl _
& vbCrLf & nl _
& vbCrLf & nl _
& " fernandoabs@ig.com.br " & nl _
& " otto@usp.br. ", , "Hardy - Weinberg Equilibrium Testing"
End Sub
```

FORM 2

```
Private Sub Command4_Click()
Label4.Visible = False: Text21.Locked = False: Text21.ForeColor = &H80000012
Command4.Visible = False: Command2.Visible = True: Form2.Refresh
End Sub
Private Sub Command2_Click()
a = Val(Text21.Text): Command2.Visible = False: Label4.Visible = False
Form2.Refresh
B$ = " .0123456789"
For i = 1 To Len(Text21.Text)
For j = 1 To 12
If Mid$(Text21.Text, i, 1) = Mid$(B$, j, 1) Then
num = 1: If Mid$(Text21.Text, i, 1) <> Mid$(B$, 1, 1) Then X = 1
If X = 1 Then If Mid$(Text21.Text, i, 1) = Mid$(B$, 1, 1) Then Y = 1
If Y = 1 Then If Mid$(Text21.Text, i, 1) <> Mid$(B$, 1, 1) Then w = 1
End If
Next j
If num = 0 Then Exit For
Next i
If a < 2 Or Int(a) <> a Or num = 0 Or w = 1 Then
Label4.Visible = True: Text21.ForeColor = &HFF&
Form2.Refresh: Command4.Visible = True: Text21.Locked = True
Else
If a > 20 Then
Label4.Caption = " MAXIMUM NUMBER OF ALLELES PERMITTED IS 20.PLEASE REINSERT NUMBER OF ALLELES."
```

```

Label4.Visible = True: Text21.ForeColor = &HFF&: Text21.Locked = True
Form2.Refresh: Command4.Visible = True
Else
    nal = a: Text21.Enabled = False
    If nal = 2 Then Check6.Enabled = True: Check6.Value = 1
    Check3.Enabled = True: Check4.Enabled = True: Check5.Enabled = True: Check13.Enabled =
True
    Check3.Value = 1: Check4.Value = 1: Check5.Value = 1: Check13.Value = 1
    Command1.Visible = True
    Form2.Refresh
    End If
End If
End Sub

```

```

Private Sub Command1_Click()
Command1.Visible = False
Label2.Visible = False
Label4.Visible = False
Form2.Refresh
cte2 = Check3.Value: cte3 = Check4.Value: cte4 = Check5.Value: cte5 = Check6.Value: cte10
= Check13.Value
Form2.Hide
Form4.Show
End Sub

```

FORM 3

```

Public K, prob, x4, x3, x2, x22, x25, x5, x6, x7, x8, x9, x20, x21, v99, p99, chisq, l88,
R88, F1, F2, x24
Dim nf(21, 21), PF(21), F(200001)
Dim Prob1(10000), ord(211, 3), prob2(211, 4)

```

```

Private Sub Command1_Click()
End
End Sub

```

```

Private Sub Command2_Click()
Picture1.Visible = False: Picture2.Visible = False
Picture5.Visible = False: Frame1.Visible = False
Label2.Visible = False: Label5.Visible = False
Command2.Visible = False
Form3.Hide
Form5.Show
End Sub

```

```

Private Sub Command5_Click()
Command5.Visible = False
Label6.Visible = False
ww = 0: End
End Sub

```

Private Sub Form_Load()

```

Form3.Show: Form3.Refresh
Dim totp(21), AL(3), AL2(3), AUX2(5), totpjack(21), apr(21, 21), vmax(401, 3)
Dim TOT(21, 21), TOT2(21, 21), RESUL(3, 21, 21), totjack(21, 21), vmax1(21), apr1(21)
Dim RESUL2(3, 21, 21), sp1(3, 21), jack1(3, 21), resjack(3, 21, 21)

```

```

TI = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TI = TI + Val(Mid$(Time$, 7, 2))

```

```

For i = 1 To nal
    For j = 1 To nal
        totn = totn + N(i, j)
    
```

```

    P(i) = P(i) + N(i, j)
    P(j) = P(j) + N(i, j)
Next j
Next i

For i = 1 To nal
    If P(i) = 0 Then
        For j = 1 To nal
            P(j) = 0
        Next j
        totn = 0: Label6.Visible = True: Command5.Visible = True
        Form3.Refresh: Exit Sub
    End If
Next i

v99 = nal * (nal + 1) / 2 - nal
Open App.Path & "\results.txt" For Output As #1

If totn > 100000 Then
    Label6.Caption = " SAMPLE SIZE EXCEDES MAXIMUM PERMITTED SIZE. "
    Label6.Visible = True: Command5.Visible = True
    Form3.Refresh: Exit Sub
End If

If totn > 10000 Then
    lim = Int(60 - (totn - 10000) / 90000 * 40)
Else
    If totn > 5000 Then
        lim = Int(500 - (totn - 5000) / 5000 * 450)
    Else
        If totn > 500 Then
            lim = Int(1000 - (totn - 500) / 4500 * 500)
        Else
            lim = 1000
        End If
    End If
End If

ReDim temp(2 * totn + 1)
ReDim sim(lim + 1, 21, 21), sim2(lim + 1, 21, 21)
ReDim p3(1001, 21)

If lim >= 500 Then
    esp = 10
Else
    If lim >= 100 Then
        esp = 5
    Else
        esp = 1
    End If
End If

For i = 1 To 2 * totn
    F(i) = F(i - 1) + Log(i)
Next i

ac = P(1): i = 1
For j = 1 To 2 * totn
    If j > ac Then
        i = i + 1
        ac = ac + P(i)
    End If
    temp(j) = i
Next j

For i = 1 To nal: PF(i) = P(i): Next i
For i = 1 To nal: For j = i To nal: nf(i, j) = N(i, j): Next j, i
F1 = F(totn): F2 = F(2 * totn)
a = probcalc(): iprob = prob

```

```

Picture1.Visible = True: Picture2.Visible = True
Picture5.Visible = True: Frame1.Visible = True
Label2.Visible = True: Label5.Visible = True:
Form3.Refresh

While NS < lim
TF = Val(Mid$(Time$, 1, 2)) * 3600 + Val(Mid$(Time$, 4, 2)) * 60
TF = TF + Val(Mid$(Time$, 7, 2))
NS = NS + 1
If TF - TA > 0.01 Then
    Picture1.Cls
    Picture1.Print NS
    TA = TF
End If
TT = Format(TF - TI, "00")
Picture2.Cls
Picture2.CurrentX = 0
Picture2.Print TT
XX1 = NS * 1115 / lim
Picture4.Line (0, 0)-(XX1, 435), , BF
PER = NS / lim * 100
FP = Format(PER, "00")
Picture6.Cls
Picture6.CurrentX = 0
Picture6.Print FP

For K = 2 * totn - 1 To 1 Step -2
For i = 1 To 2
    ac = P(1) / (2 * totn): a = Rnd
    For j = 1 To nal
        If a < ac Then
            AL(i) = j: j = nal + 1
        Else
            ac = ac + P(j + 1) / (2 * totn)
        End If
    Next j
Next i
If AL(2) < AL(1) Then
    TAUX = AL(1): AL(1) = AL(2): AL(2) = TAUX
End If
sim2(NS, AL(1), AL(2)) = sim2(NS, AL(1), AL(2)) + 1
C = Int(K * Rnd) + 1
AL2(1) = temp(C):
TAUX = temp(C): temp(C) = temp(K + 1): temp(K + 1) = TAUX
e = Int((K - 1) * Rnd) + 1: AL2(2) = temp(e)
TAUX = temp(e): temp(e) = temp(K): temp(K) = TAUX
If AL2(2) < AL2(1) Then
    TAUX = AL2(1): AL2(1) = AL2(2): AL2(2) = TAUX
End If
If K - 2 >= Int(0.05 * totn) Then jack(NS, AL2(1), AL2(2)) = jack(NS, AL2(1), AL2(2)) +
1
    sim(NS, AL2(1), AL2(2)) = sim(NS, AL2(1), AL2(2)) + 1
Next K

For i = 1 To nal
For j = i To nal
    TOT(i, j) = TOT(i, j) + sim(NS, i, j)
    TOT2(i, j) = TOT2(i, j) + sim2(NS, i, j)
Next j, i

For i = 1 To nal
For j = i To nal
    Pjack(NS, i) = Pjack(NS, i) + jack(NS, i, j)
    Pjack(NS, j) = Pjack(NS, j) + jack(NS, i, j)
    p3(NS, i) = p3(NS, i) + sim2(NS, i, j)
    p3(NS, j) = p3(NS, j) + sim2(NS, i, j)
    totjack(i, j) = totjack(i, j) + jack(NS, i, j)
Next j

```

```

    totpjack(i) = totpjack(i) + Pjack(NS, i)
    totp(i) = totp(i) + p3(NS, i)
Next i

For i = 1 To nal: PF(i) = P(i): Next i
For i = 1 To nal: For j = i To nal: nf(i, j) = sim(NS, i, j): Next j, i
a = probcalc(): PR = prob

If PR <= iprob Then cont = cont + 1

For i = 1 To nal: PF(i) = p3(NS, i): Next i
For i = 1 To nal: For j = i To nal: nf(i, j) = sim2(NS, i, j): Next j, i
a = probcalc(): PR = prob

If PR <= iprob Then cont2 = cont2 + 1

If NS / esp = NS \ esp Then
    qq = qq + 1: pe(1, qq) = cont / NS: pe(2, qq) = cont2 / NS
End If
Wend
Picture1.Cls
Picture1.Print lim

If lim / esp <> lim \ esp Then
    qq = qq + 1
    pe(1, qq) = cont / lim
    pe(2, qq) = cont2 / lim
End If

If nal <> 2 And cte2 = 1 Then
    chisq = 0
    For i = 1 To nal
        For j = i To nal
            If i <> j Then factor = P(i) * P(j) / (2 * totn) Else factor = P(i) * P(j) / (4 *
totn)
            chisq = chisq + N(i, j) ^ 2 / factor
        Next j, i
    Next i
    chisq = chisq - totn
    If Abs(chisq) < 908 Then
        a = CHISQUAREDIST()
        pchi = p99
    Else
        pchi = 0
    End If
End If

If nal <> 2 And cte3 = 1 Then
    For i = 1 To nal
        For j = i To nal
            If i <> j Then SUMH = SUMH + N(i, j) / totn
            If N(i, j) <> 0 Then G = G + N(i, j) / totn * Log(N(i, j) / totn)
        Next j
    Next i
    G = G - P(i) / totn * Log(P(i) / (2 * totn))
    Next i
    G = G - SUMH * Log(2)
    G = 2 * G: chisq = Abs(totn * G)
    If chisq < 908 Then a = CHISQUAREDIST(): pg = p99 Else pg = 0
End If

If nal <> 2 And cte4 = 1 Then
    For i = 1 To nal
        For j = i To nal
            If i = j Then
                ei = 0.5 * P(i) * (P(i) - 1) / (2 * totn - 1)
            Else
                ei = P(i) * P(j) / (2 * totn - 1)
            End If
            If ei <> 0 Then hog = hog + (N(i, j) - ei) ^ 2 / ei
        Next j, i
    Next i

```

```

    If hog < 908 Then chisq = hog: a = CHISQUAREDIST(): phog = p99 Else phog = 0
End If

If nal <> 2 And cte10 = 1 Then

    For i = 1 To nal
        For j = i + 1 To nal
            sumhce = sumhce + N(i, j)
        Next j, i

    For i = 1 To nal
        For j = i To nal
            If i = j Then
                If P(i) ^ 2 - sumhce <> 0 Then ce = ce + N(i, j) ^ 2 / ((P(i) ^ 2 - sumhce) / (4 * totn))
            Else
                ce = ce + N(i, j) ^ 2 / ((P(i) * P(j) + N(i, j)) / (2 * totn))
            End If
        Next j, i

    If ce - totn < 908 Then chisq = ce - totn: a = CHISQUAREDIST(): pce = p99 Else pce = 0
End If

    If nal = 2 And cte2 = 1 Then
        chisq = N(1, 1) ^ 2 / (P(1) ^ 2 / (4 * totn)) + N(1, 2) ^ 2 / ((P(1) * (2 * totn - P(1))) / (2 * totn)) + N(2, 2) ^ 2 / ((2 * totn - P(1)) ^ 2 / (4 * totn)) - totn
        a = CHISQUAREDIST()
        pchi = p99
        chisq = (Abs(4 * N(1, 1) * N(2, 2) - N(1, 2) ^ 2) - 2 * totn) ^ 2
        chisq = chisq / ((2 * N(1, 1) + N(1, 2)) * (N(1, 2) + 2 * N(2, 2))) ^ 2 * totn
        a = CHISQUAREDIST()
        pchicor = p99
    End If

    If nal = 2 And cte4 = 1 Then
        hog = 0: hogcor = 0: phogcor = 0
        If (2 * N(1, 1) + N(1, 2) - 1) And (2 * N(2, 2) + N(1, 2) - 1) <> 0 Then
            hog = (totn - 1) * ((2 * N(2, 2) + N(1, 2)) * (2 * N(1, 1) + N(1, 2)) - (2 * totn - 1) * N(1, 2)) ^ 2
            hog = hog / ((2 * N(1, 1) + N(1, 2)) * (2 * N(1, 1) + N(1, 2) - 1) * (2 * N(2, 2) + N(1, 2)) * (2 * N(2, 2) + N(1, 2) - 1))
            chisq = hog
            a = CHISQUAREDIST()
            phog = p99
            hogcor = (totn - 1) * (Abs((2 * N(1, 1) + N(1, 2)) * (2 * N(2, 2) + N(1, 2)) - (2 * totn - 1) * N(1, 2)) - 4 * totn * 0.5) ^ 2
            hogcor = hogcor / ((2 * N(1, 1) + N(1, 2)) * (2 * N(1, 1) + N(1, 2) - 1) * (2 * N(2, 2) + N(1, 2)) * (2 * N(2, 2) + N(1, 2) - 1))
            chisq = hogcor
            a = CHISQUAREDIST()
            phogcor = p99
        End If
    End If

    If nal = 2 And cte3 = 1 Then
        If N(1, 1) <> 0 Then G = N(1, 1) / totn * Log(N(1, 1) / totn)
        If N(1, 2) <> 0 Then G = G + N(1, 2) / totn * Log(N(1, 2) / totn)
        If N(2, 2) <> 0 Then G = G + N(2, 2) / totn * Log(N(2, 2) / totn)
        G = G - N(1, 2) / totn * Log(2) - P(1) / totn * Log(P(1) / (2 * totn)) - P(2) / totn * Log(P(2) / (2 * totn))
        G = 2 * G: chisq = Abs(totn * G)
        a = CHISQUAREDIST()
        pg = p99
        If N(1, 1) * N(2, 2) < N(1, 2) ^ 2 / 4 Then
            DD = (N(1, 1) + 0.5) / totn: hh = (N(1, 2) / 2 - 0.5) / totn: rr = (N(2, 2) + 0.5) / totn
        Else
            DD = (N(1, 1) - 0.5) / totn: hh = (N(1, 2) / 2 + 0.5) / totn: rr = (N(2, 2) - 0.5) / totn
        End If
        If DD > 0 Then g1 = DD * Log(DD)
    End If

```



```

If hh > 0 Then g1 = g1 + 2 * hh * Log(hh)
If rr > 0 Then g1 = g1 + rr * Log(rr)
g1 = g1 - P(1) / totn * Log(P(1) / (2 * totn)) - P(2) / totn * Log(P(2) / (2 * totn))
g1 = 2 * g1: chisq = Abs(totn * g1)
a = CHISQUAREDIST()
pg1 = p99
End If

If nal = 2 And cte5 = 1 Then
If N(1, 1) = 0 And N(1, 2) = 1 Or N(2, 2) = 0 And N(1, 2) = 1 Then
s$ = "-"
Else
a = N(1, 1): D = N(2, 2)
If Int(N(1, 2) / 2) = N(1, 2) / 2 Then
B = N(1, 2) / 2: C = N(1, 2) / 2
Else
If N(1, 2) > 1 Then B = (N(1, 2) + 1) / 2: C = (N(1, 2) - 1) / 2 Else B = 1: C = 0
End If
ac = a + C: bd = B + D: ab = a + B: cd = C + D
For i = 1 To 3: AUX2(i) = 0: Next i
If ab > cd Then aux = ab: ab = cd: cd = aux: aux = a: a = C: C = aux: aux = B: B = D: D
= aux
If ac > bd Then aux = ac: ac = bd: bd = aux: aux = a: a = B: B = aux: aux = C: D = C: C
= aux
If ab > ac Then aux = ab: ab = ac: ac = aux: aux = bd: cd = bd: bd = aux: aux = B: B =
D: D = aux
If a > B Then aux = a: a = B: B = aux: aux = C: C = D: D = aux: aux = bd: bd = ac: ac =
aux
X1 = a: x2 = ab: x3 = ac: x4 = totn: x0 = X1 - x3 * x2 / x4
If x3 = x4 - x3 Then AUX2(3) = 1
If x3 <= x4 - x3 Then
AUX2(1) = 1: x20 = x4 - x3: x21 = x4: x22 = x2
w = a1(): w = a2()
If X1 = 0 Then w = a2()
End If
x20 = x3: x21 = x4: x22 = x2: w = a1(): w = a3()
x25 = x2 - X1: w = a4(): w = a3()
x3 = x9: x9 = x24: x25 = x2: X1 = 0: x2 = 0
If x24 <= x3 Then X1 = x24
While x25 <> 0
x5 = x5 - 1: x6 = x6 - 1: x7 = x7 + 1: x8 = x8 + 1
x9 = x9 * x5 * x6 / (x7 * x8):
If x9 <= x3 Then If AUX2(1) = 1 Then x2 = x2 + x9 Else X1 = X1 + x9
x25 = x25 - 1
Wend
pfish = X1 + x2
End If
End If

If nal = 2 And cte10 = 1 Then
de = (P(1) ^ 2 - N(1, 2)) / (4 * totn)
he = (P(1) * P(2) + N(1, 2)) / (2 * totn)
re = (P(2) ^ 2 - N(1, 2)) / (4 * totn)
If de <> 0 Then ce = N(1, 1) ^ 2 / de
If he <> 0 Then ce = ce + N(1, 2) ^ 2 / he
If re <> 0 Then ce = ce + N(2, 2) ^ 2 / re
ce = ce - totn
chisq = ce: a = CHISQUAREDIST()
pce = p99
End If

Picture5.Visible = True
DD = 2 * nal + 3 * (nal * (nal + 1)) / 2
For j = 1 To nal
For K = j To nal
For i = 1 To lim
Probl(i) = sim(i, j, K)
Next i
cc = cc + 1
a = SORTVALUES()
AA = 0

```

```

If Int(0.025 * lim) <> 0.025 * lim Then AA = 1
RESUL(1, j, K) = Prob1(Int(0.025 * lim) + AA): RESUL(2, j, K) = Prob1(1 + Int(0.975 *
lim))

Picture5.Cls
Picture5.Print "ORDERING DATA CLUSTER NUMBER " & cc & " OF " & DD

For i = 1 To lim
    Prob1(i) = sim2(i, j, K)
Next i
cc = cc + 1
a = SORTVALUES()
RESUL2(1, j, K) = Prob1(Int(0.025 * lim) + AA): RESUL2(2, j, K) = Prob1(1 + Int(0.975 *
lim))

Picture5.Cls
Picture5.Print "ORDERING DATA CLUSTER NUMBER " & cc & " OF " & DD

For i = 1 To lim
    Prob1(i) = jack(i, j, K)
Next i
cc = cc + 1
a = SORTVALUES()
resjack(1, j, K) = Prob1(Int(0.025 * lim) + AA): resjack(2, j, K) = Prob1(1 + Int(0.975
* lim))
Picture5.Cls
Picture5.Print "ORDERING DATA CLUSTER NUMBER " & cc & " OF " & DD

Next K, j

For i = 1 To nal
    For j = 1 To lim
        Prob1(j) = p3(j, i)
    Next j
    cc = cc + 1
    a = SORTVALUES()
    spl(1, i) = Prob1(Int(0.025 * lim) + AA): spl(2, i) = Prob1(1 + Int(0.975 * lim))
    Picture5.Cls
    Picture5.Print "ORDERING DATA CLUSTER NUMBER " & cc & " OF " & DD

For j = 1 To lim
    Prob1(j) = Pjack(j, i)
Next j
cc = cc + 1
a = SORTVALUES()
jack1(1, i) = Prob1(Int(0.025 * lim) + AA): jack1(2, i) = Prob1(1 + Int(0.975 * lim))
Picture5.Cls
Picture5.Print "ORDERING DATA CLUSTER NUMBER " & cc & " OF " & DD

Next i
Max = 0: l = 0: K = 0: max1 = 0: l1 = 0
For i = 1 To nal
    For j = i To nal
        K = K + 1
        If N(i, j) = Max Then
            l = l + 1: vmax(1, 1) = i: vmax(1, 2) = j
        Else
            If N(i, j) > Max Then
                l = 1: vmax(1, 1) = i: vmax(1, 2) = j: Max = N(i, j)
            End If
        End If
        cc = Format(N(i, j) / totn, "0.000")
        apr(i, j) = Val(cc)
        sapr = sapr + apr(i, j)
    Next j
    If P(i) = max1 Then l1 = l1 + 1: vmax1(l1) = i
    If P(i) > max1 Then l1 = 1: vmax1(l1) = i: max1 = P(i)
    cc1 = Format(P(i) / (2 * totn), "0.000")
    apr1(i) = Val(cc1)

```

```

sapr1 = sapr1 + apr1(i)
Next i

If Abs(sapr - 1) >= Abs(0.0001) Then dif = 1000 * sapr - 1000
If Abs(sapr1 - 1) >= Abs(0.0001) Then dif1 = 1000 * sapr1 - 1000
a = Int(dif) : B = Int(dif1)

If Abs(dif - a) > 0.5 Then
If dif > 0 Then
dif = a + 1
Else
dif = a - 1
End If
Else
dif = a
End If

If Abs(dif1 - B) > 0.5 Then
If dif1 > 0 Then
dif1 = B + 1
Else
dif1 = B - 1
End If
Else
dif1 = B
End If
jj = 0
If Abs(dif) / 1 > 0.3 * Max Then
K = 0: jj = 1
For i = 1 To nal
For j = i To nal
K = K + 1: prob2(K, 1) = apr(i, j): prob2(K, 2) = i: prob2(K, 3) = j
Next j, i
a = ord1()

End If
If jj = 0 Then

la = 1
For i = 1 To Abs(dif)
If la = 0 Then la = 1
apr(vmax(la, 1), vmax(la, 2)) = apr(vmax(la, 1), vmax(la, 2)) - dif / Abs(dif) * 0.001
la = la - 1
Next i

Else
la = 1

For i = 1 To Abs(dif)
apr(ord(i, 1), ord(i, 2)) = apr(ord(i, 1), ord(i, 2)) - dif / Abs(dif) * 0.001
Next i

End If

la = 11
For i = 1 To Abs(dif1)
If la = 0 Then la = 11
apr1(vmax1(la)) = apr1(vmax1(la)) - dif1 / Abs(dif1) * 0.001
la = la - 1
Next i

Print #1, "OBSERVED DATA ": Print #1,
Text3.Text = " OBSERVED DATA " & vbCrLf & vbCrLf
Print #1, "GENOTYPE OBS. NO. FREQ."
Text3.Text = Text3.Text & " GENOTYPE OBS. NO. FREQ." & vbCrLf
Print #1, "-----"
Text3.Text = Text3.Text & " -----" & vbCrLf
For i = 1 To nal
For j = i To nal

```

```

Print #1, i & "-" & j;
Text3.Text = Text3.Text & " " & i & "-" & j
K1 = Len(N(i, j)): K2 = Len(i): K3 = Len(j)
For l = 1 To 11 - K2 - K3
  If l <> 1 Then Print #1, " ";
  Text3.Text = Text3.Text & " ": Next l
Print #1, N(i, j);: Text3.Text = Text3.Text & N(i, j)
cc = Format(apr(i, j), "0.000")
For l = 1 To 12 - K1
  If l <> 1 Then Print #1, " ";
  Text3.Text = Text3.Text & " ": Next l
Print #1, cc: Text3.Text = Text3.Text & cc & vbCrLf
Next j, i

K = Len(totn)
Print #1, "-----"
Text3.Text = Text3.Text & " -----" & vbCrLf
Print #1, "TOTAL" & totn;
Text3.Text = Text3.Text & " TOTAL" & totn
For i = 1 To 12 - K: Print #1, " ";
Text3.Text = Text3.Text & " ": Next i
Print #1, "1.000": Print #1, : Text3.Text = Text3.Text & "1.000" & vbCrLf & vbCrLf

Print #1, "ALLELE" & OBS.NO. & "FREQ."
Text3.Text = Text3.Text & " ALLELE" & OBS.NO. & "FREQ." & vbCrLf
Print #1, "-----"
Text3.Text = Text3.Text & " -----" & vbCrLf
For i = 1 To nal
  KK = Len(i): KK1 = Len(P(i))
  Print #1, "" & i;: Text3.Text = Text3.Text & " " & i
  For j = 1 To 12 - KK
    If j <> 1 Then Print #1, " ";
    Text3.Text = Text3.Text & " ": Next j
  Print #1, P(i);: Text3.Text = Text3.Text & P(i)
  cc = Format(apr1(i), "0.000")
  For j = 1 To 12 - KK1
    If j <> 1 Then Print #1, " ";
    Text3.Text = Text3.Text & " ": Next j
  Print #1, cc: Text3.Text = Text3.Text & cc & vbCrLf
Next i
Print #1, "-----"
Text3.Text = Text3.Text & " -----" & vbCrLf
KK = Len(2 * totn)
Print #1, "TOTAL" & 2 * totn;
Text3.Text = Text3.Text & " TOTAL" & 2 * totn
For i = 1 To 12 - KK
  Print #1, " ";
  Text3.Text = Text3.Text & " ": Next i
Print #1, "1.000": Print #1, : Text3.Text = Text3.Text & "1.000" & vbCrLf & vbCrLf

Print #1, "COMPUTER-SIMULATED DATA"
Text3.Text = Text3.Text & " COMPUTER-SIMULATED DATA" & vbCrLf

Print #1, "NUMBER OF SIMULATIONS = " & lim & vbCrLf
Text3.Text = Text3.Text & " NUMBER OF SIMULATIONS = " & lim & vbCrLf & vbCrLf

Print #1, "AVERAGE ALLELE FREQUENCIES WITH CORRESPONDING 95% CONFIDENCE INTERVALS "
Print #1, "(BOOTSTRAP SIMULATIONS WITH REPLACEMENT AND 97.5% SUB-SAMPLING JACKKNIFE)":
Print #1,
  Text3.Text = Text3.Text & " AVERAGE ALLELE FREQUENCIES WITH CORRESPONDING 95% CONFIDENCE INTERVALS " & vbCrLf
  Text3.Text = Text3.Text & " (BOOTSTRAP SIMULATIONS WITH REPLACEMENT AND 97.5% SUB-SAMPLING JACKKNIFE)" & vbCrLf & vbCrLf

Print #1, " WITH REPLACEMENT 97.5% SS JACKKNIFE"
Text3.Text = Text3.Text & " WITH REPLACEMENT 97.5% SS JACKKNIFE" & vbCrLf
Print #1, "ALLELE" & AV.FR. & 95% C.I. & AV.FR. & 95% C.I.
Text3.Text = Text3.Text & " ALLELE" & AV.FR. & 95% C.I. & AV.FR. & 95% C.I. & vbCrLf
Print #1, "-----"
Text3.Text = Text3.Text & " -----" & vbCrLf

```

```

For i = 1 To nal
    a = Format(spl(1, i) / (2 * totn), "0.000"): B = Format(spl(2, i) / (2 * totn), "0.000")
    C = Format(totp(i) / (2 * totn * lim), "0.000"): D = Format(totpjack(i) / ((2 * (totn - 1 - Int(0.025 * totn))) * lim), "0.000")
    If i < 10 Then
        Text3.Text = Text3.Text & " " & i & " " & C & " " & a & "-" & B
        Print #1, " " & i & " " & C & " " & a & "-" & B;
    Else
        Text3.Text = Text3.Text & " " & i & " " & C & " " & a & "-" & B
        Print #1, " " & i & " " & C & " " & a & "-" & B;
    End If
    a = Format(jack1(1, i) / (2 * (totn - 1 - Int(0.025 * totn))), "0.000"): B = Format(jack1(2, i) / (2 * (totn - 1 - Int(0.025 * totn))), "0.000")
    Print #1, " " & D & " " & a & "-" & B
    Text3.Text = Text3.Text & " " & D & " " & a & "-" & B & vbCrLf
Next i
Print #1, "-----": Print #1,
Text3.Text = Text3.Text & "-----" & vbCrLf
& vbCrLf

Print #1, "AVERAGE GENOTYPE FREQUENCIES WITH CORRESPONDING 95% CONFIDENCE INTERVALS"
Print #1, "(BOOTSTRAP SIMULATIONS WITH AND WITHOUT REPLACEMENT AND 97.5% SUB-": Print #1,
Print #1, "SAMPLING JACKKNIFE METHOD)": Print #1,
Text3.Text = Text3.Text & " AVERAGE GENOTYPE FREQUENCIES WITH CORRESPONDING 95% CONFIDENCE INTERVALS " & vbCrLf
Text3.Text = Text3.Text & " (BOOTSTRAP SIMULATIONS WITH AND WITHOUT REPLACEMENT AND 97.5% SUB-" & vbCrLf & " SAMPLING JACKKNIFE METHOD) " & vbCrLf & vbCrLf

Print #1, " WITH REPLACEMENT W/OUT REPLACEMENT 97.5% SS JACKKNIFE"
Text3.Text = Text3.Text & " WITH REPLACEMENT W/OUT REPLACEMENT 97.5% SS JACKKNIFE" & vbCrLf

Print #1, "GENOT. AV.FR. 95% C.I. AV.FR. 95% C.I. AV.FR. 95% C.I."
Text3.Text = Text3.Text & " GENOT. AV.FR. 95% C.I. AV.FR. 95% C.I. AV.FR. 95% C.I." & vbCrLf
Print #1, "-----"
Text3.Text = Text3.Text & "-----"
-----" & vbCrLf

For i = 1 To nal
    For j = i To nal
        Print #1, i & "-" & j;
        K = Len(i): K1 = Len(j)
        Text3.Text = Text3.Text & " " & i & "-" & j
        For l = 1 To 11 - K - K1: Print #1, " "; Text3.Text = Text3.Text & " ": Next l
        a = Format(RESUL(1, i, j) / totn, "0.000"): B = Format(RESUL2(1, i, j) / totn, "0.000")
        C = Format(RESUL(2, i, j) / totn, "0.000"): D = Format(RESUL2(2, i, j) / totn, "0.000")
        e = Format(TOT(i, j) / (totn * lim), "0.000"): ff = Format(TOT2(i, j) / (totn * lim), "0.000")
        G = Format(totjack(i, j) / ((totn - 1 - Int(0.025 * totn)) * lim), "0.000")
        Print #1, e & " " & a & "-" & C;
        If i < 10 And j < 10 Then
            Text3.Text = Text3.Text & e & " " & a & "-" & C
        Else
            If i < 10 And j >= 10 Or i >= 10 And j < 10 Then
                Text3.Text = Text3.Text & e & " " & a & "-" & C
            Else
                Text3.Text = Text3.Text & e & " " & a & "-" & C
            End If
        End If
        Print #1, " " & ff & " " & B & "-" & D;
        Text3.Text = Text3.Text & " " & ff & " " & B & "-" & D
        a = Format(resjack(1, i, j) / (totn - 1 - Int((0.025 * totn) / 2)), "0.000"): B = Format(resjack(2, i, j) / (totn - 1 - Int((0.025 * totn) / 2)), "0.000")
        Print #1, " " & G & " " & a & "-" & B
        Text3.Text = Text3.Text & " " & G & " " & a & "-" & B & vbCrLf
    Next j, i
    Print #1, "-----": Print #1,
    Print #1,

```

```

Text3.Text = Text3.Text & " -----
-----" & vbCrLf & vbCrLf

Print #1, "PROBABILITY VALUES": Print #1,
Text3.Text = Text3.Text & " PROBABILITY VALUES" & vbCrLf & vbCrLf

Print #1, "TEST                                P"
Text3.Text = Text3.Text & " TEST                                P" & vbCrLf
Print #1, "-----"
Text3.Text = Text3.Text & " -----" & vbCrLf

Text3.Visible = True
a = Format(cont / lim, "0.000"): B = Format(cont2 / lim, "0.000")
Text3.Text = Text3.Text & " BOOTSTRAP WITHOUT REPLACEMENT " & a & vbCrLf
Text3.Text = Text3.Text & " BOOTSTRAP WITH REPLACEMENT " & B & vbCrLf
Print #1, "BOOTSTRAP WITHOUT REPLACEMENT " & a
Print #1, "BOOTSTRAP WITH REPLACEMENT " & B

If cte2 = 1 Then
a = Format(pchi, "0.000")
Form3.Refresh
Print #1, "CHI-SQUARED TEST " & a
Text3.Text = Text3.Text & " CHI-SQUARED TEST " & a & vbCrLf
If nal = 2 Then
a = Format(pchicor, "0.000")
Print #1, "CHI-SQ. TEST WITH CORRECTION " & a
Text3.Text = Text3.Text & " CHI-SQ. TEST WITH CORRECTION " & a & vbCrLf
End If
End If

If cte4 = 1 Then
a = Format(phog, "0.000")
If nal = 2 Then B = Format(phogcor, "0.000")
Print #1, "HOGBEN/LEVENE METHOD " & a
Text3.Text = Text3.Text & " HOGBEN/LEVENE METHOD " & a & vbCrLf
If nal = 2 Then
Print #1, "CORRECTED HOGBEN/LEVENE METHOD " & B
Text3.Text = Text3.Text & " CORRECTED HOGBEN/LEVENE METHOD " & B & vbCrLf
End If
End If

If cte3 = 1 Then
a = Format(pg, "0.000")
If nal = 2 Then B = Format(pgl, "0.000")
Print #1, "G (LOG-LIKELIHOOD) TEST " & a
Text3.Text = Text3.Text & " G (LOG-LIKELIHOOD) TEST " & a & vbCrLf
If nal = 2 Then
Print #1, "CORRECTED G (LOG-LIKELIH.) TEST " & B
Text3.Text = Text3.Text & " CORRECTED G (LOG-LIKELIH.) TEST " & B & vbCrLf
End If
End If

If cte10 = 1 Then
a = Format(pce, "0.000")
Print #1, "CANNINGS & EDWARDS METHOD " & a
Text3.Text = Text3.Text & " CANNINGS & EDWARDS METHOD " & a & vbCrLf
End If

If cte5 = 1 Then
If s$ <> "-" Then
a = Format(pfish, "0.000")
Print #1, "FISHER EXACT TEST " & a
Text3.Text = Text3.Text & " FISHER EXACT TEST " & a & vbCrLf
Else
Print #1, "FISHER EXACT TEST -"
Text3.Text = Text3.Text & " FISHER EXACT TEST -" & vbCrLf
End If
End If

Print #1, "-----"

```

```

Text3.Text = Text3.Text & " -----" & vbCrLf & vbCrLf

Text3.Text = Text3.Text & " [THIS TEXT IS RECORDED ON FILE RESULTS.TXT]"
XC = totjack(1) / ((2 * (totn - 1 - Int(0.025 * totn))) * lim)
YC = totjack(1, 2) / ((totn - 1 - Int(0.025 * totn)) * lim)
Text3.Locked = True: Command1.Visible = True: Command2.Visible = True
Form3.Refresh
Close #1

End Sub

Function probcalc()
prob = 0: h = 0
For i = 1 To nal
    F3 = F(PF(i)): prob = prob + F3
    For j = i To nal
        F4 = F(nf(i, j)): prob = prob - F4
        If i <> j Then h = h + nf(i, j)
    Next j
Next i
prob = prob + F1 - F2 + h * Log(2)
End Function

Function SORTVALUES()
Dim H88(5000)
D88 = 10: K88 = 1: L88 = 1: R88 = lim
560 Rem EXCHANGE SORT IF BLOCKSIZE <= D88
    If R88 - L88 + 1 > D88 Then GoTo 590
    a = EXCHANGESORT(): GoTo 780
590 i88 = L88: J88 = R88: W88 = Rnd(1) * (R88 - L88) + 0.5
    W88 = Int(W88 + L88): F88 = Prob1(W88)
    Rem PARTITION PHASE
620 If Prob1(i88) >= F88 Then GoTo 640
    i88 = i88 + 1: GoTo 620
640 If Prob1(J88) <= F88 Then GoTo 660
    J88 = J88 - 1: GoTo 640
660 If i88 > J88 Then GoTo 680
    W88 = Prob1(i88): Prob1(i88) = Prob1(J88): Prob1(J88) = W88: i88 = i88 + 1: J88 = J88
- 1
680 If i88 <= J88 Then GoTo 620
    Rem BOOKKEEPING PHASE
    If J88 - L88 >= R88 - i88 Then GoTo 740
    If i88 >= R88 Then GoTo 730
    H88(K88) = i88: K88 = K88 + 1: H88(K88) = R88: K88 = K88 + 1
730 R88 = J88: GoTo 770
740 If L88 >= J88 Then GoTo 760
    H88(K88) = L88: K88 = K88 + 1: H88(K88) = J88: K88 = K88 + 1
760 L88 = i88
770 If L88 < R88 Then GoTo 590
780 If K88 <= 1 Then GoTo 800
    K88 = K88 - 1: R88 = H88(K88): K88 = K88 - 1: L88 = H88(K88): GoTo 560
800
End Function

Function EXCHANGESORT()
    For i88 = L88 To R88 - 1: K1 = i88
    For J88 = i88 + 1 To R88: If Prob1(J88) >= Prob1(K1) Then GoTo 940
    K1 = J88
940 Next J88
    If K1 = i88 Then GoTo 970
    W88 = Prob1(i88): Prob1(i88) = Prob1(K1): Prob1(K1) = W88
970 Next i88

End Function

Function CHISQUAREDIST()
r99 = 0: w99 = chisq: v99a = v99
For i = v99a To 2 Step -2: r99 = r99 + Log(i): Next i
If w99 <> 0 Then
k99 = Log(w99) * (Int((v99a + 1) / 2)) - w99 / 2 - r99

```

```

If Int(v99a / 2) = v99a / 2 Then GoTo Label11
If w99 <> 0 Then j99 = 0.5 * (Log(2 / w99 / 3.141592653599)): GoTo Label22
Label11: j99 = 0
Label22: l99 = 1: m99 = 0
CHILOOP: v99a = v99a + 2: m99 = m99 + Log(w99) - Log(v99a): If m99 < Log(0.0000001) Then
p99 = 1 - Exp(k99 + j99) * l99: GoTo endl
l99 = l99 + Exp(m99): GoTo CHILOOP
endl:
Else
p99 = 1
End If
End Function

Function a1()
x9 = 1: x25 = x22
While x25 <> 0: x9 = x9 * (x20 + 1 - x25) / (x21 + 1 - x25): x25 = x25 - 1: Wend
x24 = x9
End Function

Function a2()
x5 = x2 + 1: x6 = x3 + 1: x7 = 0: x8 = x4 - x3 - x2
End Function

Function a3()
x5 = x2 + 1: x6 = x4 - x3 + 1: x7 = 0: x8 = x3 - x2
End Function

Function a4()
While x25 <> 0
x5 = x5 - 1: x6 = x6 - 1: x7 = x7 + 1: x8 = x8 + 1
x9 = x9 * x5 * x6 / (x7 * x8): x25 = x25 - 1
Wend
End Function

Function ord1()
px = K
For i = 1 To K
Max = 0
For j = 1 To px
If prob2(j, 1) = Max Then ord(i, 1) = prob2(j, 2): ord(i, 2) = prob2(j, 3): aux = j
If prob2(j, 1) > Max Then Max = prob2(j, 1): ord(i, 1) = prob2(j, 2): ord(i, 2) =
prob2(j, 3): aux = j
Next j
AUX2 = prob2(px, 1): prob2(px, 1) = prob2(aux, 1): prob2(aux, 1) = AUX2
AUX2 = prob2(px, 2): prob2(px, 2) = prob2(aux, 2): prob2(aux, 2) = AUX2
AUX2 = prob2(px, 3): prob2(px, 3) = prob2(aux, 3): prob2(aux, 3) = AUX2
px = px - 1
Next i
End Function

FORM 4

Dim ref(401), s

Private Sub Command3_Click()
sumr = 0: Command3.Visible = False
For i = 1 To nal
K = 0
For j = i To nal
If ref(20 + sumr + K) <> 1 Then
TEXT2(20 + sumr + K).Enabled = True
Else
TEXT2(20 + sumr + K).Locked = False
TEXT2(20 + sumr + K).ForeColor = &H0&
ref(20 + sumr + K) = 0
End If
K = K + 1
Next j
sumr = sumr + 20 - i + 1
Next i

```



```

Label2.Visible = False
Command1.Visible = True: Form4.Refresh
End Sub

Private Sub Form_Load()

For i = 0 To 249: TEXT2(i).FontSize = 8: Next i
For i = 0 To 19: TEXT2(i) = i + 1: TEXT2(i).Enabled = False: Next i
a = 1
For i = 230 To 249: TEXT2(i) = a: a = a + 1: TEXT2(i).Enabled = False: Next i

For i = 1 To nal
TEXT2(i - 1).Visible = True: TEXT2(229 + i).Visible = True
K = 0
For j = i To nal
TEXT2(20 + sumr + K).Visible = True
K = K + 1
Next j
sumr = sumr + 20 - i + 1
Next i
Form4.Refresh
End Sub

Private Sub Command1_Click()
sumr = 0: Command1.Visible = False: wr = 0
Label2.Visible = False
B$ = ".0123456789"
For i = 1 To nal
a = 0: X = 0: Y = 0: w = 0: K = 0
For m = i To nal
If TEXT2(20 + sumr + K).Text = "" Then
num = 1
Else
For j = 1 To Len(TEXT2(20 + sumr + K).Text)
num = 0
C$ = Mid$(TEXT2(20 + sumr + K).Text, j, 1)
If C$ = Mid$(B$, 2, 1) Then a = a + 1
For l = 1 To 12
If C$ = Mid$(B$, 1, 1) Then
num = 1: If C$ <> Mid$(B$, 1, 1) Then X = 1
If X = 1 Then If C$ = Mid$(B$, 1, 1) Then Y = 1
If Y = 1 Then If C$ <> Mid$(B$, 1, 1) Then w = 1
End If
Next l
If num = 0 Then Exit For
Next j
End If
If Val(TEXT2(20 + sumr + K)) < 0 Or Int(Val(TEXT2(20 + sumr + K))) <> Val(TEXT2(20 +
sumr + K)) Or num = 0 Or a > 1 Or w = 1 Then
wr = 1: TEXT2(20 + sumr + K).ForeColor = &HFF&: ref(20 + sumr + K) = 1
End If
K = K + 1
Next m
sumr = sumr + 20 - i + 1
Next i

sumr = 0
For l = 1 To nal
K = 0
For j = 1 To nal
If ref(20 + sumr + K) <> 1 Then
TEXT2(20 + sumr + K).Enabled = False
Else
TEXT2(20 + sumr + K).Locked = True
End If
K = K + 1
Next j
sumr = sumr + 20 - 1 + 1
Next l

If wr = 1 Then

```

```

    Label2.Visible = True: Command3.Visible = True: Exit Sub
End If

If ww = 0 Then
    Label1.Caption = " CONFIRM DATA INPUT? ": Option1.Visible = True: Option2.Visible = True
End If

If s <> 1 Then Command2.Visible = True: Label1.Visible = True: Form4.Refresh
If s = 1 Then
    sumr = 0
    For l = 1 To nal
        K = 0
        For j = 1 To nal
            TEXT2(20 + sumr + K).Enabled = True
            K = K + 1
        Next j
        sumr = sumr + 20 - l + 1
    Next l
    sumr = 0
    For i = 1 To nal
        K = 0
        For j = i To nal
            N(i, j) = Val(TEXT2(20 + sumr + K).Text)
            K = K + 1
        Next j
        sumr = sumr + 20 - i + 1
    Next i
    Command1.Visible = True
    Form4.Hide
    Form3.Show
End If
End Sub

Private Sub Command2_Click()
    sumr = 0
    For l = 1 To nal
        K = 0
        For j = 1 To nal
            TEXT2(20 + sumr + K).Enabled = True
            K = K + 1
        Next j
        sumr = sumr + 20 - l + 1
    Next l
    Label1.Visible = False: Option1.Visible = False: Option2.Visible = False
    If Option2.Value = True Then
        ww = 1: Option2.Value = False: Command2.Visible = False: Command1.Visible = True: s = 1:
        Form4.Refresh
        Exit Sub
    Else
        sumr = 0
        For i = 1 To nal
            K = 0
            For j = i To nal
                N(i, j) = Val(TEXT2(20 + sumr + K).Text)
                K = K + 1
            Next j
            sumr = sumr + 20 - i + 1
        Next i
    End If
    sumr = 0
    For l = 1 To nal
        K = 0
        For j = 1 To nal
            TEXT2(20 + sumr + K).Text = ""
            K = K + 1
        Next j
        sumr = sumr + 20 - l + 1
    Next l
    Option1.Value = False: Option2.Value = False
    Command2.Visible = False: Command1.Visible = True
    Form4.Hide

```

```
Form3.Show
End Sub
```

FORM 5

```
Private Sub Command1_Click()
Form5.Hide
Form3.Show
End Sub
```

```
Private Sub Command2_Click()
Form5.Hide
Form6.Show
End Sub
```

```
Private Sub Command3_Click()
End
End Sub
```

```
Private Sub Form_Load()
Form5.Show
Picture1.CurrentX = 3000: Picture1.CurrentY = 5200
Picture1.Print "NUMBER OF SIMULATIONS"
Picture1.Line (500, 5000)-(500, 500)
Picture1.Line (500, 5000)-(8000, 5000)
Picture1.Line (8000, 5000)-(8000, 5150)
Picture1.Line (450, 500)-(500, 500)
Picture1.CurrentX = 300: Picture1.CurrentY = 400
Picture1.Print "1"
Picture1.CurrentX = 300: Picture1.CurrentY = 4900
Picture1.Print "0"
Picture1.Line (450, 5000)-(500, 5000)
Picture1.Line (2500, 100)-(3000, 100)
Picture1.CurrentX = 3100: Picture1.CurrentY = 50
Picture1.FontSize = 7
Picture1.Print "PROBABILITY OF EXACT TEST BASED ON SIMULATIONS WITH REPLACEMENT"
Picture1.Line (2500, 300)-(3000, 300), 1000
Picture1.CurrentX = 3100: Picture1.CurrentY = 250
Picture1.Print "PROBABILITY OF EXACT TEST BASED ON SIMULATIONS WITHOUT REPLACEMENT"
Picture1.FontSize = 8
Picture1.CurrentX = 7800: Picture1.CurrentY = 5100
Picture1.Print lim
Picture1.CurrentX = 200: Picture1.CurrentY = 1000
Picture1.Print "P"
Picture1.CurrentX = 200: Picture1.CurrentY = 1300: Picture1.Print "R"
Picture1.CurrentX = 200: Picture1.CurrentY = 1600: Picture1.Print "O"
Picture1.CurrentX = 200: Picture1.CurrentY = 1900: Picture1.Print "B"
Picture1.CurrentX = 200: Picture1.CurrentY = 2200: Picture1.Print "A"
Picture1.CurrentX = 200: Picture1.CurrentY = 2500: Picture1.Print "B"
Picture1.CurrentX = 200: Picture1.CurrentY = 2800: Picture1.Print "I"
Picture1.CurrentX = 200: Picture1.CurrentY = 3100: Picture1.Print "L"
Picture1.CurrentX = 200: Picture1.CurrentY = 3400: Picture1.Print "I"
Picture1.CurrentX = 200: Picture1.CurrentY = 3700: Picture1.Print "T"
Picture1.CurrentX = 200: Picture1.CurrentY = 4000: Picture1.Print "Y"
X = 500: Y1 = 5000: Y2 = Y1
If lim / esp <> lim \ esp Then div = qq - 1 Else div = qq
For i = 1 To div
y11 = 5000 - pe(1, i) * 4500: xi = 500 + i / div * 7500
yi2 = 5000 - pe(2, i) * 4500
Picture1.Line (X, Y1)-(xi, y11)
Picture1.Line (X, Y2)-(xi, yi2), 1000
X = xi
Y1 = y11: Y2 = yi2
Next i
If div = qq - 1 Then
Picture1.Line (X, Y1)-(8000, y11)
Picture1.Line (X, Y2)-(8000, yi2), 1000
End If
Command3.Visible = True
If nal = 2 Then Command2.Visible = True
Form5.Refresh
```

```

End Sub

FORM 6

Public XSCREEN, YSCREEN
Public XMAX, YMAX, XMIN, YMIN, X, Y
Dim rd(3)

Private Sub Command1_Click()
Form6.Hide
Form3.Show
End Sub

Private Sub Command2_Click()
End
End Sub

Private Sub Form_Load()
Form6.Show
Form6.Refresh
Dim YS1(2)

XMIN = 0: XMAX = 1
YMIN = 0: YMAX = 1
Picture1.Visible = False: Form6.Refresh

Rem VGA GRAPHICS MODE RESOLUTION 480 x 640
Rem SMALL TRIANGLE WITH EXPLANATION
Picture1.Line (370 * 7.5 * 5 / 3 * 8.5 / 9.8, 20 * 4.5 / 0.5)-(820 * 7.5 * 6 / 5, 20 * 4.5 / 0.5)
Picture1.Line (370 * 7.5 * 5 / 3 * 8.5 / 9.8, 20 * 4.5 / 0.5)-(370 * 7.5 * 5 / 3 * 8.5 / 9.8, 220 * 4.5 / 0.5)
Picture1.Line (820 * 7.5 * 6 / 5, 220 * 4.5 / 0.5)-(820 * 7.5 * 6 / 5, 20 * 4.5 / 0.5)
Picture1.Line (820 * 7.5 * 6 / 5, 220 * 4.5 / 0.5)-(370 * 7.5 * 5 / 3 * 8.5 / 9.8, 220 * 4.5 / 0.5)
Picture1.Line (640 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(720 * 7.5 * 6 / 5, 40 * 4.5 / 0.5)
Picture1.Line -(800 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (640 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(667 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (684 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(738 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (756 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)-(800 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (706 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(706 * 7.5 * 6 / 5, 155 * 4.5 / 0.5)
Picture1.Line (706 * 7.5 * 6 / 5, 179 * 4.5 / 0.5)-(706 * 7.5 * 6 / 5, 200 * 4.5 / 0.5)
Picture1.Line (672 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(682 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)
Picture1.Line (699 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(730 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)
Picture1.Line (747 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)-(768 * 7.5 * 6 / 5, 136 * 4.5 / 0.5)
Picture1.Circle (706 * 7.5 * 6 / 5, 136 * 4.5 / 0.5), 30
Picture1.CurrentX = 682 * 7.5 * 6 / 5
Picture1.CurrentY = 124 * 4.5 / 0.5
Picture1.Print "d"
Picture1.CurrentX = 735 * 7.5 * 6 / 5
Picture1.CurrentY = 124 * 4.5 / 0.5
Picture1.Print "r"
Picture1.CurrentX = 700 * 7.5 * 6 / 5
Picture1.CurrentY = 155 * 4.5 / 0.5
Picture1.Print "h"
Picture1.CurrentX = 670 * 7.5 * 6 / 5
Picture1.CurrentY = 187 * 4.5 / 0.5
Picture1.Print "p"
Picture1.CurrentX = 740 * 7.5 * 6 / 5
Picture1.CurrentY = 187 * 4.5 / 0.5
Picture1.Print "q"
Picture1.CurrentX = 80 * 7.5
Picture1.CurrentY = 470 * 4.5 / 0.5
Picture1.Print "0.0"
Picture1.CurrentX = 80 * 7.5
Picture1.CurrentY = 40 * 4.5 / 0.5
Picture1.Print "1.0"
Picture1.CurrentX = 90 * 7.5 * 5 / 3 * 8.5 / 9.8

```

```

Picture1.CurrentY = 500 * 4.5 / 0.5
Picture1.Print "0.0"
Picture1.CurrentX = 450 * 7.5 * 5 / 3 * 8.5 / 9.8
Picture1.CurrentY = 500 * 4.5 / 0.5
Picture1.Print "1.0"
Rem GRAPH AXES
Picture1.Line (130 * 7.5, 40 * 4.5 / 0.414)-(130 * 7.5, 400 * 4.5 / 0.414)
Picture1.Line (100 * 7.5 * 5 / 3 * 8.5 / 9.8, 410 * 4.5 / 0.414)-(460 * 7.5 * 5 / 3 * 8.5 / 9.8, 410 * 4.5 / 0.414)
Rem SIDES OF ISOSCELES TRIANGLE
Picture1.Line (100 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)-(280 * 7.5 * 5 / 3 * 8.5 / 9.8, 40 * 4.5 / 0.414)
Picture1.Line -(460 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)
Picture1.Line -(100 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)
Rem Y-AXIS TICKS
For i = 40 To 400 Step 18
    Picture1.Line (126 * 7.5, i * 4.5 / 0.414)-(130 * 7.5, i * 4.5 / 0.414)
Next i
Rem X-AXIS TICKS
For i = 100 To 460 Step 18
    Picture1.Line (i * 7.5 * 5 / 3 * 8.5 / 9.8, 410 * 4.5 / 0.414)-(i * 7.5 * 5 / 3 * 8.5 / 9.8, 414 * 4.5 / 0.414)
Next i

Picture1.FillColor = &HFFFFFF

factor = 4
Picture2.Visible = True
While pi < 0.95
    ct = 0
    rr1 = rr1 + 0.00005
    rr2 = factor * rr1
    For i = 1 To lim
        If (Pjack(i, 1) / (2 * (totn - 1 - Int(0.025 * totn))) - XC) <> 0 Then
            angi = Atn((Abs(jack(i, 1, 2) / (totn - 1 - Int(0.025 * totn)) - YC)) / Abs((Pjack(i, 1) / (2 * (totn - 1 - Int(0.025 * totn))) - XC)))
        Else
            angi = 3.1415 / 2
        End If
        Y = rr2 * Sin(angi)
        X = rr1 * Cos(angi)
        B = (jack(i, 1, 2) / (totn - 1 - Int(0.025 * totn)) - YC) ^ 2 + (Pjack(i, 1) / (2 * (totn - 1 - Int(0.025 * totn))) - XC) ^ 2
        If X ^ 2 + Y ^ 2 >= B Then ct = ct + 1
    Next i
    pi = ct / lim
    If AA = 3 Then AA = 0
    AA = AA + 1: Picture2.Cls
    If AA = 1 Then
        Picture2.Print "PLEASE WAIT. ESPECIALLY IN THE CASE OF VERY SMALL SAMPLES," & vbCrLf &
        "THE PROCESSING MAY TAKE SEVERAL MINUTES."
    Else
        If AA = 2 Then
            Picture2.Print "PLEASE WAIT. ESPECIALLY IN THE CASE OF VERY SMALL SAMPLES," & vbCrLf &
            "THE PROCESSING MAY TAKE SEVERAL MINUTES.."
        Else
            Picture2.Print "PLEASE WAIT. ESPECIALLY IN THE CASE OF VERY SMALL SAMPLES," & vbCrLf &
            "THE PROCESSING MAY TAKE SEVERAL MINUTES..."
        End If
    End If
Wend
rd(1) = rr1: rd(2) = rr2
Picture2.Visible = False
Form6.Refresh

Picture1.ForeColor = &H80FF80
For i = 1 To lim
    X = Pjack(i, 1) / (2 * (totn - 1 - Int(0.025 * totn))): Y = jack(i, 1, 2) / (totn - 1 - Int(0.025 * totn))
    a = PLOTXY()

```

```

If X <> 0 And X <> 1 Then
Z = Rnd
If Z <= 1 / 5 Then
XSCREEN = XSCREEN + 360 * Rnd * 1 / (2 * totn) * 7.5 * 5 / 3 * 8.5 / 9.8
YSCREEN = YSCREEN + 360 * Rnd * 1 / totn * (4.5 / 0.414)
If YSCREEN > 4347.8260869565 Then YSCREEN = 4347.8260869565 - 3.4
If XSCREEN > 4987.2448979591 Then XSCREEN = 4987.2448979591 - 3.4
If XSCREEN > 3035.7142857142 Then
If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.00511701337 * XSCREEN
- 5652.181 - 3.4
End If
Else
If Z <= 2 / 5 Then
XSCREEN = XSCREEN - 360 * Rnd * 1 / (2 * totn) * 7.5 * 5 / 3 * 8.5 / 9.8
YSCREEN = YSCREEN + 360 * Rnd * 1 / totn * (4.5 / 0.414)
If XSCREEN < 1084.1836734693 Then
XSCREEN = 1084.1836734693 - 3.4
Else
If XSCREEN <= 3035.7142857142 Then
If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 *
XSCREEN + 6521.737 - 3.4
Else
If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.005117 * XSCREEN -
5652.181 - 3.4
End If
If YSCREEN > 4347.8260869565 Then YSCREEN = 4347.8260869565 - 3.4
End If
Else
If Z <= 3 / 5 Then
XSCREEN = XSCREEN - 360 * Rnd * 1 / (2 * totn) * 7.5 * 5 / 3 * 8.5 / 9.8
YSCREEN = YSCREEN - 360 * Rnd * 1 / totn * (4.5 / 0.414)
If YSCREEN < 434.7826086956 Then YSCREEN = 434.7826086956 - 3.4
If XSCREEN < 1084.1836734693 Then
XSCREEN = 1084.1836734693 - 3.4
Else
If XSCREEN <= 3035.7142857142 Then
If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 *
XSCREEN + 6521.737 - 3.4
Else
If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.005117 * XSCREEN -
5652.181 - 3.4
End If
End If
Else
If Z <= 4 / 5 Then
XSCREEN = XSCREEN + 360 * Rnd * 1 / (2 * totn) * 7.5 * 5 / 3 * 8.5 / 9.8
YSCREEN = YSCREEN - 360 * Rnd * 1 / totn * (4.5 / 0.414)
If YSCREEN < 434.7826086956 Then YSCREEN = 434.7826086956 - 3.4
If XSCREEN > 4987.2448979591 Then
XSCREEN = 4987.2448979591 - 3.4
Else
If XSCREEN > 3035.7142857142 Then
If YSCREEN < 2.00511701337 * XSCREEN - 5652.181 Then YSCREEN = 2.00511701337 *
XSCREEN - 5652.181 - 3.4
Else
If YSCREEN < -2.00511495846 * XSCREEN + 6521.737 Then YSCREEN = -2.00511495846 *
XSCREEN + 6521.737 - 3.4
End If
End If
End If
End If
End If
End If
End If
Picture1.Circle (XSCREEN, YSCREEN), 10
Next i

Picture1.ForeColor = &H80000012
For X = XMIN To XMAX Step 1 / 400
Y = 2 * X * (1 - X)

```

```

    a = PLOTXY()
    If X = XMIN Then Picture1.PSet (XSCREEN, YSCREEN) Else Picture1.Line -(XSCREEN, YSCREEN)
Next X
X = XMAX
Y = 2 * X * (1 - X)
a = PLOTXY()

Picture1.Line -(XSCREEN, YSCREEN)

F2 = Sqr(3.841 / totn)
b2 = F2 / (1 + F2): e2 = 1 / (1 + F2)

Picture1.ForeColor = &HFF&
For j = -1 To 1 Step 2
    If j = 1 Then b2 = 0: e2 = 1
    For X = b2 To e2 Step 1 / 400
        Y = 2 * X * (1 - X) * (1 - 1 * j * F2)
        a = PLOTXY()
        If X = b2 Then Picture1.PSet (XSCREEN, YSCREEN) Else Picture1.Line -(XSCREEN, YSCREEN)
    Next X
    X = e2
    Y = 2 * X * (1 - X) * (1 - 1 * j * F2)
    a = PLOTXY()
    Picture1.Line -(XSCREEN, YSCREEN)
Next j

Picture1.ForeColor = &HFF0000
For i = -rd(1) To rd(1) Step 1 / 500000
    YA = rd(2) / rd(1) * Sqr(rd(1) ^ 2 - i ^ 2)
    Y = YC + YA
    X = XC + i
    a = PLOTXY()
    If i <> -rd(1) Then
        If X >= 0 And X <= 1 And Y >= 0 And Y <= 1 Then
            If YSCREEN >= 2.00511701337 * XSCREEN - 5652.181 Then
                If YSCREEN >= -2.00511495846 * XSCREEN + 6521.737 Then Picture1.Line (xant1, yant1)-
(XSCREEN, YSCREEN)
                End If
            End If
        End If
        xant1 = XSCREEN: yant1 = YSCREEN
        Y = YC - YA
        X = XC + i
        a = PLOTXY()
        If i <> -rd(1) Then
            If X >= 0 And X <= 1 And Y >= 0 And Y <= 1 Then
                If YSCREEN >= 2.00511701337 * XSCREEN - 5652.181 Then
                    If YSCREEN >= -2.00511495846 * XSCREEN + 6521.737 Then Picture1.Line (xant2, yant2)-
(XSCREEN, YSCREEN)
                    End If
                End If
            End If
        End If
        xant2 = XSCREEN: yant2 = YSCREEN
    Next i
X = P(1) / (2 * totn): Y = N(1, 2) / totn: a = PLOTXY()
Picture1.FillColor = &HFF&
Picture1.Circle (XSCREEN, YSCREEN), 30, 2000
Picture1.FillColor = &H80000009
Picture1.ForeColor = &H0&
Command1.Visible = True: Command2.Visible = True
Picture1.CurrentX = 400 * 7.5 * 5 / 3 * 8.5 / 9.8: Picture1.CurrentY = 50 * 4.5 / 0.5
Picture1.FontSize = 7
Picture1.Print "d = N(1,1) / n"
Picture1.CurrentX = 400 * 7.5 * 5 / 3 * 8.5 / 9.8: Picture1.CurrentY = 80 * 4.5 / 0.5
Picture1.Print "h = N(1, 2) / n":
Picture1.CurrentX = 400 * 7.5 * 5 / 3 * 8.5 / 9.8: Picture1.CurrentY = 110 * 4.5 / 0.5
Picture1.Print "r = N(2, 2) / n"
Picture1.CurrentX = 400 * 7.5 * 5 / 3 * 8.5 / 9.8: Picture1.CurrentY = 140 * 4.5 / 0.5
Picture1.Print "p = d + h / 2"
Picture1.CurrentX = 400 * 7.5 * 5 / 3 * 8.5 / 9.8: Picture1.CurrentY = 170 * 4.5 / 0.5

```

```

Picture1.Print "q = r + h / 2"
Picture1.ForeColor = &HFF0000
Picture1.FontSize = 8
Picture1.Circle (4500, 2500), 150, , , 2
Picture1.ForeColor = &H80000012
Picture1.CurrentX = 4700: Picture1.CurrentY = 2400
Picture1.Print "95% H-W CONFIDENCE REGION"
Picture1.Line (4400, 2900)-(4700, 2900)
Picture1.CurrentX = 4800: Picture1.CurrentY = 2800
Picture1.Print "HARDY-WEINBERG"
Picture1.CurrentX = 4800: Picture1.CurrentY = 3000
Picture1.Print "EQUILIBRIUM CURVE"
Picture1.Line (4600, 3400)-(4850, 3400), 1000
Picture1.CurrentX = 4900: Picture1.CurrentY = 3300
Picture1.Print "APPROX. 95% CONF. LIMITS"
Picture1.CurrentX = 4900: Picture1.CurrentY = 3500
Picture1.Print "OF H-W CURVE "
Picture1.FillColor = &HFF&
Picture1.Circle (4850, 3900), 30, 2000
Picture1.CurrentX = 5000: Picture1.CurrentY = 3800
Picture1.Print "SAMPLE POINT"
Picture1.Visible = True
Picture1.Line (100 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)-(280 * 7.5 * 5 / 3 * 8.5 / 9.8, 40 * 4.5 / 0.414)
Picture1.Line -(460 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)
Picture1.Line -(100 * 7.5 * 5 / 3 * 8.5 / 9.8, 400 * 4.5 / 0.414)
Form6.Refresh
End Sub

```

```

Function PLOTXY():
    XSCREEN = (100 + 360 * (X - XMIN) / (XMAX - XMIN)) * 7.5 * 5 / 3 * 8.5 / 9.8
    YSCREEN = (400 - 360 * (Y - YMIN) / (YMAX - YMIN)) * (4.5 / 0.414)
End Function

```

```

Private Sub Picture3_Click()
End Sub

```


REFERÊNCIAS BIBLIOGRÁFICAS

Barton NH, Slatkin M. 1986. A quasi-equilibrium theory of the distribution of rare alleles in a subdivided population. *Heredity* 56: 409-415.

Black WC, Krafur ES. 1985. A fortran program for the calculation of two linkage disequilibrium coefficients. *Theor Appl Genet* 70: 491-496.

Buckleton J, Triggs CM, Curran JM. Detection of deviations from genetic equilibrium- a comment on population data on the thirteen CODIS core loci, Budowle, et al. 1999. *J Forensic Sci* 46(1):198-200.

Calfell F. 2006. HWSIM. Disponível em URL: <http://info.med.yale.edu/genetics/kkidd/programs.html>, krunch.med.yale.edu/hwsim/hwsim.txt.

Cannings C, Edwards AWF. 1969. Expected genotypic frequencies in a small sample: deviation from Hardy-Weinberg equilibrium. *Amer J Hum Genet* 21: 245-247.

Chapco W. 1976. An Exact Test of the Hardy-Weinberg Law. *Biometrics* 32: 183-189.

Chen JJ et al. Hardy-Weinberg testing for HLA class II (DRB1, DQA1, DQB1, and DPB1) loci in 26 human ethnic groups. 1999. *Tissue Antigens* 54: 533-542.

Cockerham C C. 1973. Analyses of gene frequencies. *Genetics*, 74: 679-700.

Constantine CC et al. 2006. GENESTRUT. Disponível Na URL-
<http://numbat.murdoch.edu.au/vetschl/imgad/GenStrut.htm>.

Dempster A P, Laird, N M, Rubin, D B. 1977. Maximum likelihood from
incomplete data via the EM algorithm. J Royal Stat Soc B 39: 1-38.

Elston R C, Forthofer R. 1977. Testing for Hardy-Weinberg Equilibrium
in Small Samples. Biometrics 33: 536-542.

Emigh T H. 1980. A Comparison of Tests for Hardy-Weinberg
Equilibrium. Biometrics 36: 627-642.

Excoffier L, Laval G, Schneider S. 2005. Arlequin (version 3.0): an
integrated software package for population genetics data analysis.
Evolutionary Bioinformatics Online 1: 47-50.

Garnier-Gere P, Dillmann C. 1992. A computer program for testing
pairwise linkage disequilibria in subdivided populations. J Heredity
83:239.

Goudet J. 1995. Fstat version 1.2: a computer program to calculate
F-statistics. J. Hered. 86: 485-486.

Goudet, J., Raymond, M., De Meeüs, T., Rousset, F. 1996. Testing
differentiation in diploid populations. Genetics 144: 933-1940

Guo SW, Thompson EA, 1992. Performing the exact test of Hardy-
Weinberg proportions for multiple alleles. Biometrics 48: 361-372.

Hardy GH. 1908. Mendelian proportions in a mixed population. Science
28: 49-50.

Haldane J B S. 1954. An exact test for randomness of mating. J Genet
52: 631-635

Hochberg Y, 1988. A sharper Bonferroni procedure for multiple tests
of significance. Biometrika 75: 800-802.

Hogben L. 1946. An introduction to mathematical genetics. Norton, New York.

Labate JA. 2000. Software for population genetic analyses of molecular marker data. Crop science 40: 1521-1528.

Lanyon SM. 1987. Jackknifing and bootstrapping: important "new" statistical techniques for ornithologists. Auk 104.

Levene H. 1949. On a matching problem arising in genetics. Ann Math Statist 20: 91-94.

Lewis PO, Zaykin D. 2006. Genetic data analysis: computer program for the analysis of allelic data. Disponível na URL - <http://hydrodictyon.eeb.uconn.edu/people/plewis/software.php>.

Li CC. 1988. Pseudo-random mating populations. In celebration of the 80th anniversary of the Hardy-Weinberg law. Genetics 119: 731-737.

Louis EJ, Dempster ER. 1987. An exact test for Hardy-Weinberg and multiple alleles. Biometrics 43:805-811.

Mantel, N. 1967. The detection of disease clustering and a generalized regression approach. Cancer Res 27:209-220.

Marsaglia G, Zaman A, Tsang WW. 1990. Toward a universal random number generator. Stat Prob Letters 8:35-39.

Mehta CR, Patel NR. 1983. A network algorithm for performing Fisher's exact test in $r \times c$ contingency tables. J Am Stat Assoc 78: 427-434.

Michalakis Y, Excoffier L. 1996. A generic estimation of population subdivision using distances between alleles with special interest to microsatellite loci. Genetics 142: 1061-1064.

Miller MP. 2006. Tools for population genetic analysis (TFGPA). Disponível na URL - <http://www.marksgeneticsoftware.net>.

Montoya-Delgado L E, Irony T Z, Pereira C de B, Whittle M R. 2001. An Unconditional Exact Test for the Hardy-Weinberg Equilibrium Law: Sample-Space Ordering Using the Bayes Factor. *Genetics* 158: 875-883.

Ohta T. 1982. Linkage disequilibrium due to random genetic drift in finite subdivided populations. *Proc Nat Acad Sci USA* 79: 1940-1944.

Otto P A, Benedetti L A. 1995. A Note on the Graphical Representation of Genotype Frequencies in Population Genetics. *The Journal of Heredity* 86,2: 163-164.

Pereira C A de B, Rogatko A. 1984. The Hardy-Weinberg equilibrium under a Bayesian perspective. *Braz J Genet* 4: 689-707.

Raymond M, Rousset F. 1995. An exact test for population differentiation. *Evolution*, 49 : 1280-1283.

Robertson A, Hill WG. 1984. Deviations from Hardy-Weinberg proportions : sampling variances and use in estimation of inbreeding coefficients. *Genetics* 107 : 713-718.

Raymond M , Rousset F. 1995. GENEPOP (version 1.2): population genetics software for exact tests and ecumenicism. *J. Heredity*, 86:248-249. Disponível na URL: <http://wbiomed.curtin.edu.au/genepop/>.

Rogatko A, Slifker MJ, Babb JS. 2002. Hardy-Weinberg Equilibrium Diagnostics. *Theoretical Population Biology*, 62: 251-257.

Rogatko A, Slifker M J. HYDIAG. 2006. Disponível na URL: www.fccc.edu/users/rogatko/hwdiag.uide.html.

Rousset F. 1996. Equilibrium values of measure of population subdivision for stepwise mutation processes. *Genetics* 142: 1357-1362.

Rousset F. 1997. Genetic differentiation and estimation of gene flow from F-statistics under isolation by distance. *Genetics* 145: 1219-1228.

Rousset F, Raymond M. 1995. Testing heterozygote excess and deficiency. *Genetics* 140: 1413-1419.

Slatkin M. 1985. Rare alleles as indicators of gene flow. *Evolution* 39: 53-65.

Slatkin M. 1993. Isolation by distance in equilibrium and non-equilibrium populations. *Evolution* 47:264-279.

Stark AE. 1980. Inbreeding systems: classification by a canonical form. *J Math Biol* 10:305.

Stark AE. 2005. The Hardy-Weinberg principle. *Genet. Mol. Biol.* Vol 23 n° 3 São Paulo July/Sept 2005.

Swofford D L, Selander RB. 1981. Biosys-1: a FORTRAN program for the comprehensive analysis for electrophoretic data in population genetics and systematics. *J Heredity* 72: 281-283.

Walsh B. 2000. Resampling methods: randomization tests, jackknife and bootstrap estimators. Lecture notes for EEB 596z.

Weinberg W. 1908. Über den Nachweis der Vererbung beim Menschen. *Jahreshefte des Vereins für vaterländische Naturkunde in Württemberg* 64: 368-382.

Weir BS, Cockerham CC. 1984. Estimating F-statistics for the analysis of population structure. *Evolution* 38: 1358-1370.

Weir BS, 1990a. Intraspecific differentiation. In: *Molecular systematic* (Hillis DM and Moritz C, eds). Sinauer Associates, Sunderland, pp. 373-410.

Weir BS, 1990b. *Genetic data analysis*. Sinauer Publ., Sunderland.

Wigginton J E, Cutler D J, Abecasis G R. 2005. A Note on Exact Tests of Hardy-Weinberg Equilibrium. *Am J Hum Genet* 76:887-893.

Yates F. 1934. Contingency table involving small numbers and the χ^2 test. Journal of the Royal Statistical Society (Supplement) 1: 217-235.

Yeh FC, Boyle TJB. 2006. Population genetic analysis of codominant markers and quantitative traits. Disponível na URL: <http://www.ualberta.ca/~fyeh/faq.htm>.

Yu CH. 2003. Resampling methods: concepts, applications and justification. Practical Assessment, Research & Evaluation 8(19).