A Genetic Dominance Simulation Program and Its Distribution Web Site for Estimates of Population Genetic Statistics

Junyuan Wu

A research paper submitted in partial fulfillment of the
requirements for the degree of Master of Science

Major Professor: Dr. Timothy Budd

Department of Computer Science
Oregon State University
Corvallis, Oregon

March 13, 2001

A Genetic Dominance Simulation Program and Its Distribution Web Site for Estimates of Population Genetic Statistics

Junyuan Wu,
Department of Computer Science
Oregon State University
Corvallis, Oregon 97331
Junyuan@cs.orst.edu

Abstract

In order to aid comparison of estimates of genetic parameters between dominant and codominant makers for population genetics society, we developed a genetic dominance simulation program to determine how the dominance and biallelism could affect the estimation of population genetic statistics. The simulation indicates that genetic diversities within populations based on allozyme allele frequencies that were transformed into biallelic dominant data were significantly lower than for nontransformed multiallelic codominant data, while population differentiation was biased upwardly in each experimental species. Microsoft Active Server Pages (ASP) is combined with Active Data Objects (ADO) to create a dynamic web site for the distribution of the simulation program. The user's information is required to be registered into a database and they can also register their published papers to be shared with genetics community.

Keywords: Genetic dominance simulation, Active Server Pages, Active Data Objects

Acknowledgement

I would like to express my deep appreciation to my major professor, Dr. Timothy Budd, for his constant encouragement, patience and great advice. Special thanks are due my minor professor Dr. Gregg Rothermel and Dr. Bella Bose for their wondeful teaching, for reading through the report, and for helpful guidance. I am also indebted to Dr. Steven Strauss and Dr. Konstantin Krutovskii for their invaluable input, time and discussion.

Table of Contents

1. Introduction.....4

2. Genetic Dominance Simulation Program.....5

2.1 Simulation Rationale.....6

2.2 Implementation of the Simulation Program.....6

2.3 Simulation Results.....9

3. Program Distribution Web Site.....12

3.1 Overview of ASP and ADO.....12

3.1.1 Activer Server Page.....12

3.1.2 Active Data Objects.....13

3.2 Implementation of the Web Site.....14

3.2.1 The Include Files.....14

3.2.2 Global.asa File.....17

3.2.3 Logging and Registration.....18

3.2.4 Web Site User Options.....26

4. Conclusions and Future Work.....32

5. References.....33

6. Appendix: Source Code.....36

6.1 Genetic Dominance Simulation Program.....36

6.2 ASP Web Pages.....56

1. Introduction

The advent of PCR-based molecular markers has led to a rapid expansion in studies describing the levels and distribution of genetic variation among populations at the DNA level. Randomly amplified polymorphic DNA (RAPD; Williams et al. 1990) and amplified fragment length polymorphism (AFLP; Vos et al. 1995) markers are now commonly used in population genetic studies (e.g., Aagaard et al. 1998; Isabel et al. 1995; Liu and Furnier 1993; Mosseler et al. 1992; Peakall et al. 1995; Szmidt et al. 1996; Travis et al. 1996; Wu et al. 1999). However, these PCR-based markers have limitations compared to allozymes, which had been the prevalent means for population studies prior to the use of PCR. At the majority of RAPD and AFLP loci the dominant allele masks the presence of the null allele in heterozygotes when assaying diploid tissues (Krutovskii et al. 1998), thus sampling variance for dominant allele frequencies is typically greater than that for codominant alleles (Lynch and Milligan 1994). The frequencies of null- and dominant alleles are inferred from the frequency of null-allele homozygotes; the precision of their estimation therefore depends on mating system assumptions and is strongly affected by the sample size. Empirical studies have also suggested that dominant markers can bias estimates of genetic diversity and differentiation among populations (e.g., Isabel et al. 1995; Szmidt et al. 1996).

Although RAPD markers have proven to be useful for population studies, and their gross patterns of diversity usually agree with that of allozymes, the levels of genetic variation, differentiation, and fine-scale genetic structures, often differ (e.g., Baruffi et al. 1995; le Corre et al. 1997; Dawson et al. 1996; Heun et al. 1994; Lannθr-Herrera et al. 1996; Latta and Mitton 1997; Liu and Furnier 1993; Peakall et al. 1995; Puterka et al. 1993). To help assess whether these differences are biological or a simple consequence of the dominance and biallelism of RAPD and AFLP markers, we developed a dominance simulation program using Visual Basic 6.0 that transforms codominant population data into a biallelic dominant data set. The program then estimates population genetic statistics with which dominant and codominant markers can be directly compared.

The simulation program is intended to be distributed over the internet so that users can freely download it. However, we want users' information to be recorded and stored into a

database before they are able to download the program. In addition, we want users to register their published papers generated from this simulation program so that other users can share updated information. The majority of dynamic pages that use databases on the web are probably created using Common Gateway Interface (CGI). However, CGI has several disadvantages. One is that it adds another level of indirection to the client-server interaction, as the server is forced to call a CGI program. Such languages as Perl and C++ tend to be among the more complex programming languages. Second, the code that CGI receives and transmits isn't easily manipulated by a lot of programming languages. Third, CGI often isn't the fastest method of accessing databases.

Active Server Pages (ASP) with the power of Active Data Objects (ADO) provides the opportunity to combine the database and the World Wide Web and has emerged as the Microsoft Solution for web databases over the past few years (Homer, 2000). ASP is actually an extension to the web server that allows server-side scripting. At the same time it also provides a compendium of objects and components, which manage interaction between the web server and the browser. ASP is a server-side scripting environment that enables you to combine HTML, server-side scripting and COM (Component Object Model) objects, to create interactive, dynamic web applications relatively easily. ADO is a general object model that provides a programming interface for universal access to data stores provided by OLE DB. ADO enables ASP (as well as many programming languages) to read and write to data stores. Thus the combination of ASP and ADO will bring together the power of databases with the universality of the web. In addition, ASP-ADO solutions sit on the server thus reducing the complexity of accommodating multiple browser types.

The rest of this report is organized as follows. Section 2 describes the dominance simulation program. Section 3 introduces the web site for distributing the simulation program. Section 4 concludes the report and discusses future directions.

2. Genetic Dominance Simulation Program

The main simulation program involves primarily with mathematical simulation and calculation. We will divide this chapter into three sections: simulation rationale, implementation of the simulation, and the simulation experimental results.

2.1 Simulation Rationale

The objective of this simulation is to determine how dominance and biallelism could affect the estimation of genetic parameters. The simulation will also allow us to determine how sample size affects the estimation of genetic parameters. Thus assuming Hardy-Weinberg equilibrium and no linkage among loci we first use codominant multi-allelic allozyme data to generate N basic populations of up to 1,000 individuals each with multi-locus genotypes that maintain the specified allele frequencies within populations. A total of S subpopulations (S_{max} = 400) of n individuals (n = 10-200) are then drawn with replacement for each of the N populations. The sampling is done in two different ways - by sampling subpopulations of size n with replacement directly from the initially generated basic population, and by resampling subpopulations of size n with replacement within the first sampled subpopulation of n individuals (bootstrap resampling). Population genetic parameters (H_S, H_T, and G_{ST}) are then calculated for each cycle of resampling in three ways. First, for a codominant data set, calculations are made considering all alleles and genotypes present in the subpopulations. Second, the same subpopulations and data are used to simulate a dominant biallelic data set by randomly selecting one allele as dominant, with the rest treated as recessive to it. The synthetic null allele frequency is then calculated from the null homozygote frequencies assuming Hardy-Weinberg equilibrium. Finally, null allele frequencies are corrected for dominance using Lynch and Milligan's (1994) equation 2a, and their asymptotically unbiased estimate of F_{ST} recommended for dominant markers is also calculated following their equation 14a.

2.2 Implementation of the simulation program

The simulation program is implemented using Microsoft Visual Basic 6.0. Its Graphic User Interface allows user to identify input and output data files, select simulation parameters including size of base populations (500-1000), size of subpopulations, and number of sampling and resampling times. Different options can be chosen for estimation of genetic parameters. Gene diversity is evaluated using H_S (within-population diversity) and H_T (total diversity within species), either unmodified (Nei 1973) or modified (Nei and Chesser 1983) for the sample size. Genetic

differentiation is evaluated via θ_w (Theta_w) (Weir and Cockerham 1984) and G_{ST} parameters that are either unmodified (Nei 1973), modified for the sample size (Nei and Chesser 1983), or modified for both the sample size and population number (Nei 1986). The screenshot for the simulation program is shown in Figure 1.

Implementation of the program is composed of the following main functions and/or procedures. They are described as follows.

I. openFile

OpenFile is used to read data into the program from an input data file that includes original allele frequency based on codominant allozyme markers.

II. theoretical_value

Theoretical_value is used to calculate theoretical values of Hs, Ht, Gst and θ_w based on original allele frequencies. It calls other two functions/procedures. The procedure *calc_theor_diversity* is to calculate Hs and Ht, while the function *calculateTheta* is to calculate θ_w. Gst is calculated based on Hs and Ht values.

III. popGenerate

PopGenerate is used to generate basic populations of up to 1,000 individuals each with multi-locus genotypes that maintain the original allele frequencies within populations.

IV. empirical_value

Empirical_value is used to calculate empirical values of Hs, Ht, Gst and θ_w based on allele frequencies in generated sets of 1000 individuals. These values can be compared with theoretical values to confirm base populations are generated correctly. Empirical_value first calls *getFrequency* procedure to obtain allele frequencies in base population based on generated genotypes from *popGenerate* procedure. *Calc_empiri_diversity* and *calculateTheta* are then called to calculate Hs, Ht, Gst and θ_w.

V. Sampling

Implementation of the sampling procedure is divided into several steps:

(1) Subpopulations of size *n* (10-200) are sampled with replacement directly from the initially generated basic population. Genotypes for all sampled individuals are generated.

(2) For a codominant data set, the *getFrequency* procedure is called to obtain multiallelic allele frequency based on genotypes generated in Step 1.

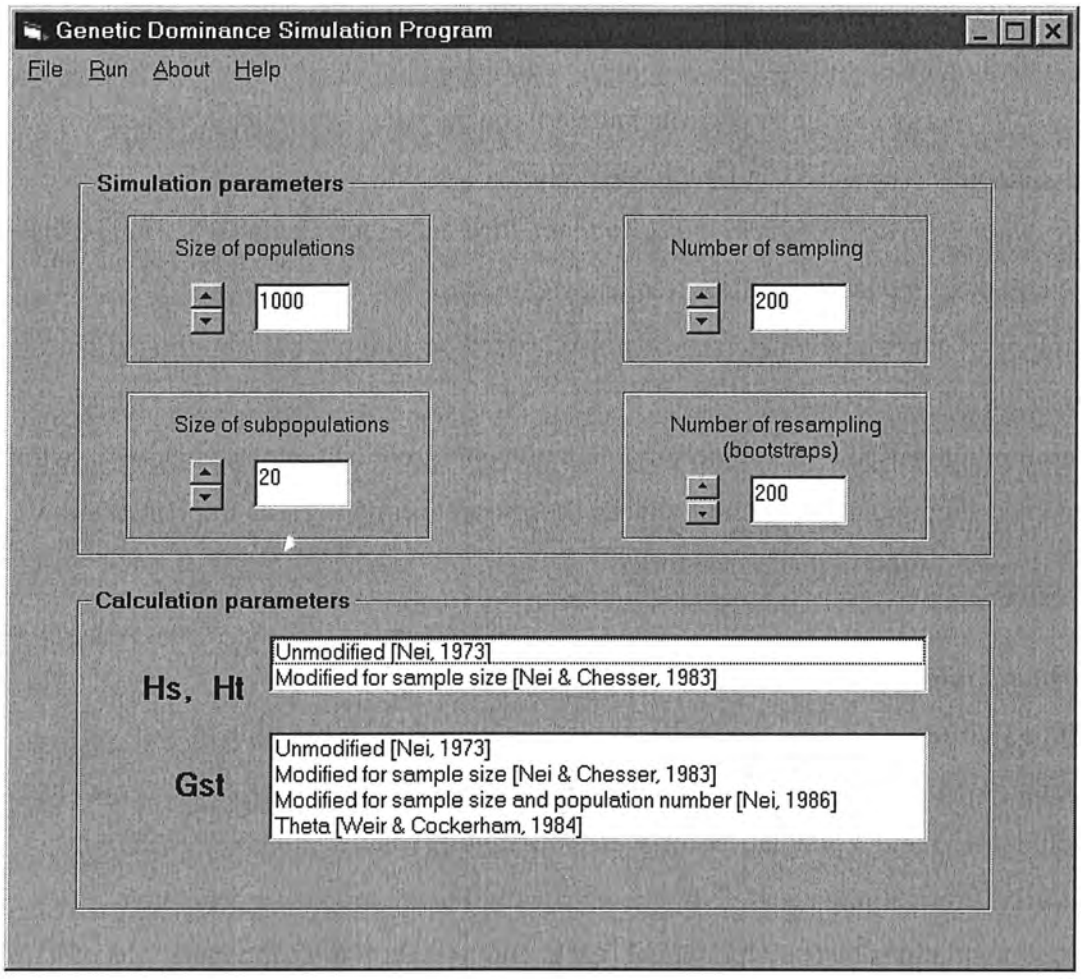


Figure 1. Graphical user interface of the genetic simulation program

(3) For simulating a dominant data set, the *get_indir_cor_freq* procedure is called to obtain diallelic allele frequency by randomly selecting one allele as dominant, with the rest treated as recessive to it. A corrected allele frequency data set is also obtained through this procedure based on Lynch and Milligan's (1994) equation 2a.
(4) *Calc_empiri_diversity* and *calculateTheta* are then called to calculate Hs, Ht, Gst and θ_w based on the three kinds of allele frequency data sets.
(5) Variance of genetic parameters is calculated based on a number of sampling circles.

VI. Resampling

Resampling is used to resample subpopulations of size *n* with replacement within the first sampled subpopulation of *n* individuals (bootstrap resampling). Its implementation process is very similar to that of sampling.

2.3 Simulation Results

Three California Closed-cone Pine species were used as an example for our simulation. We simulated dominance and biallelism in their allozyme data sets. The data sets included 4, 5 and 3 populations of *Pinus attenuata, P. muricata, and P. radiata*, respectively. From allozyme allele frequencies within populations, we generated simulated populations of 1000 individuals each, and a total of 400 subpopulations of *n* individuals were drawn with replacement from each of the populations. The program also performed 400 bootstrap resamplings using a subpopulation of size *n*. Population genetic parameters (H_S, H_T, G_{ST}, θ_w) were then calculated for each set of 400 subpopulations in the three ways described above. The results of the simulations are summarized in Figures 2 and 3. The simulations showed that diversity measurements (H_S and H_T) were likely to be underestimated by dominant biallelic markers approximately two-fold regardless of sample size. For the genetic differentiation, the estimates for the simulated dominant markers converge toward the estimates for codominant multi-allelic markers at large population sizes, but were always significantly higher. Our results demonstrate the importance of simulations to help compare and interpret the results of population studies with dominant markers.

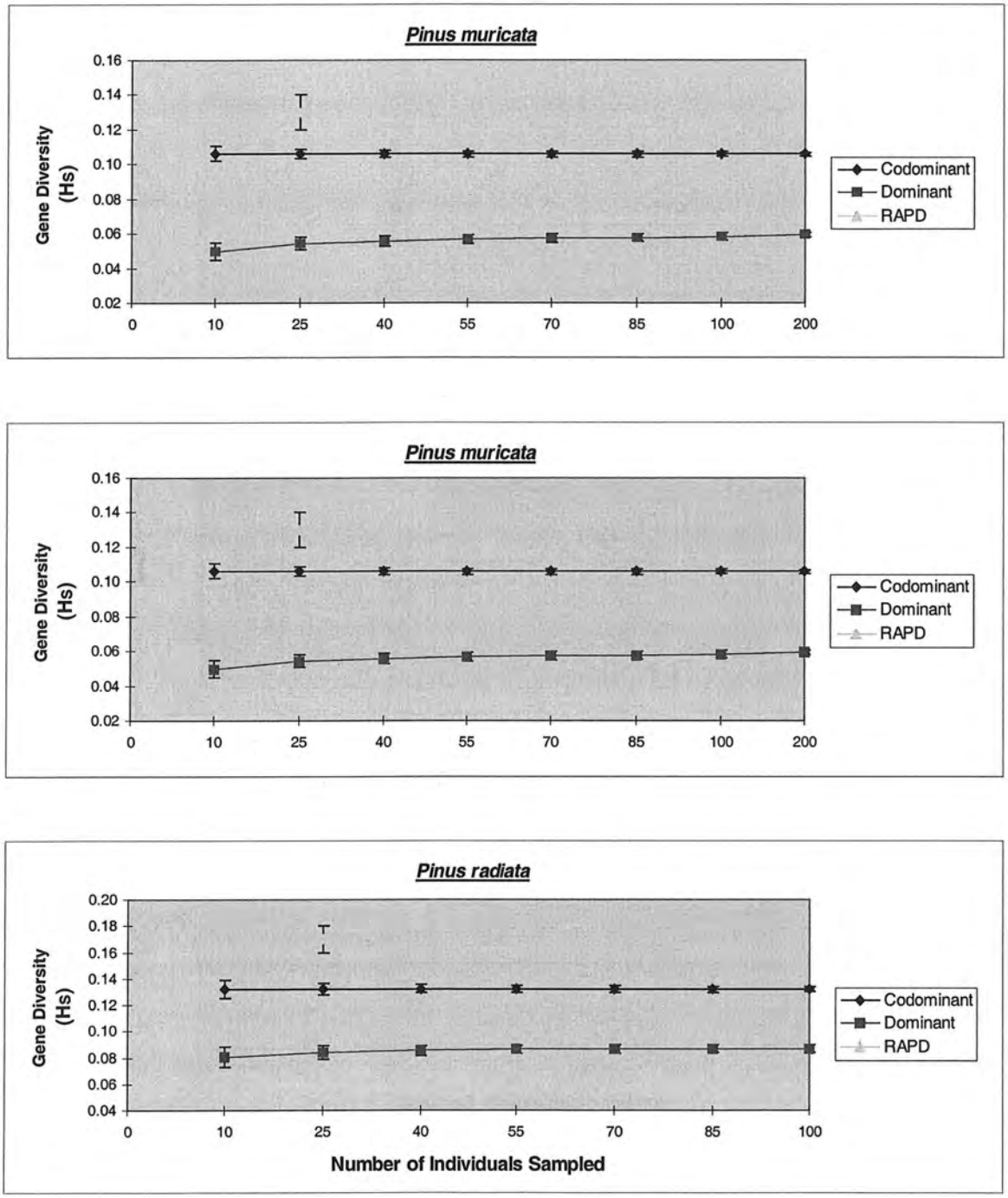


Figure 2. Genetic diversity averaged over populations of each California Closed-Cone Pine species for multiallelic allozymes and simulated dominant markers.

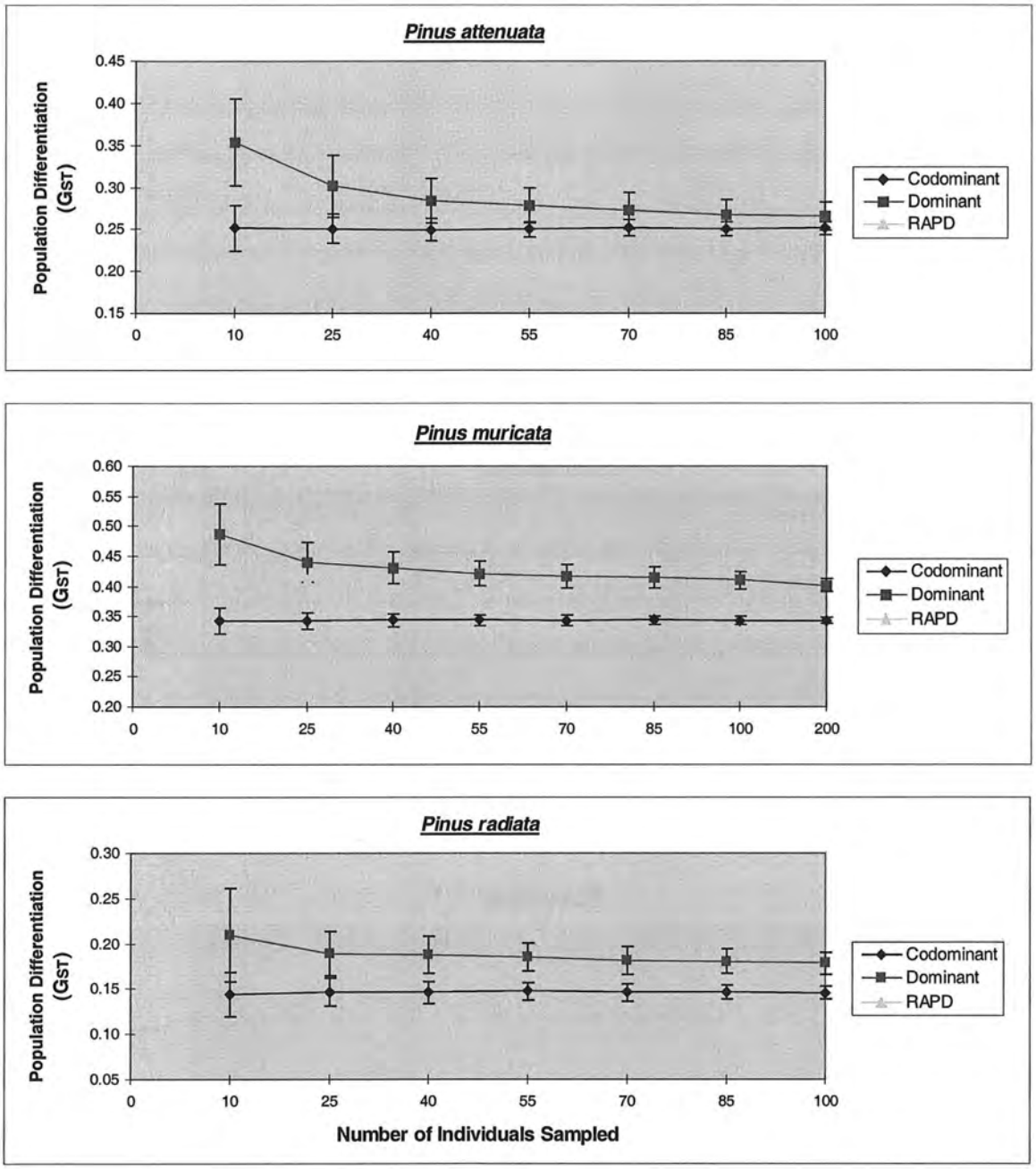


Figure 3. Genetic differentiation among populations for each California Closed-Cone Pine species for multiallelic allozymes and simulated dominant markers.

3. Program Distribution Web Site

We used the combination of ASP and ADO to develop a Genetic Dominance Simulation Web Site. There are four main functions performed on this web site. The first is to have the user login. Next the user has the option of downloading the simulation program, registering a newly published paper, and listing registered papers.

3.1 Overview of ASP and ADO

3.1.1 Active Server Page

ASP is a new server-based technology developed by Microsoft and designed to build dynamic, interactive applications for the Web or LAN-based intranet. When a request for an ASP page is sent to the web server, the web server pulls the file from its location on the server, and feeds it to the ASP Engine (ASP.DLL) on the server. The ASP engine then executes the scripting on the page and returns the dynamically-created HTML to the server, which in turn, streams it to the browser. The whole process can be illustrated in Figure 4.

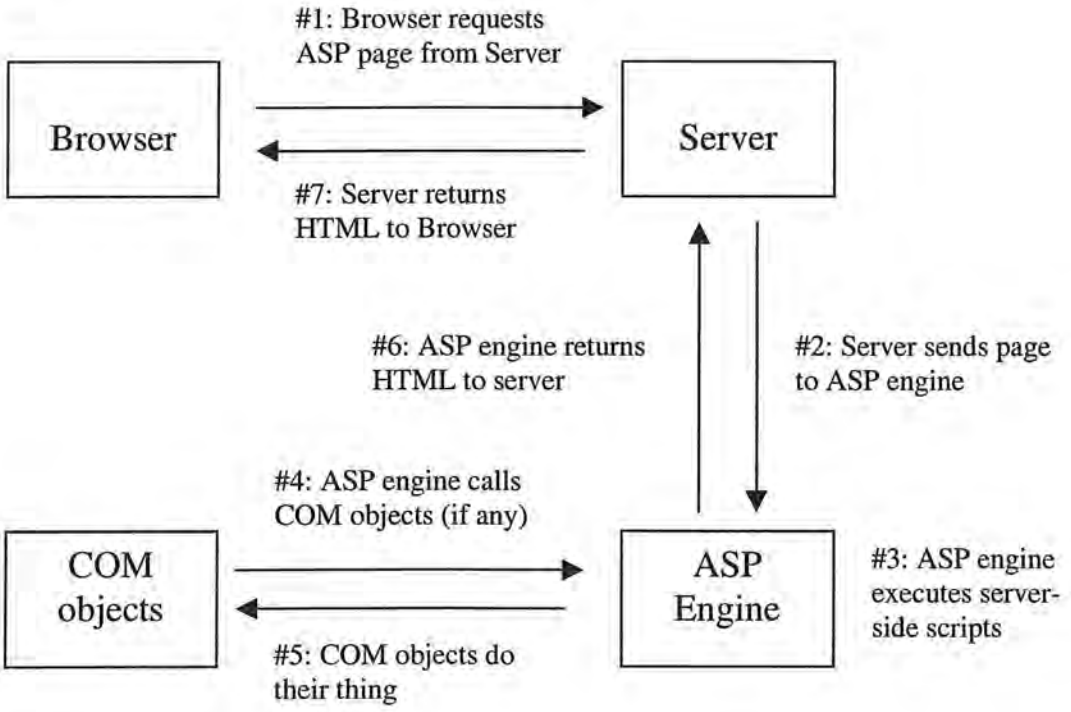


Figure 4. Flowchart of the interpretation of ASP page

As the HTML is output by the ASP Engine to the server, the server outputs the HTML in a stream to the browser. Because the page can look entirely different, depending on the results of the server-side scripting, it is called a dynamically generated page.

There are six Active Server Objects, each of which deals with a specific aspect of interactivity between the web server and the browser:

1. The *Request* object is used to deal with a request that a user might make of a site or application;
2. The *Response* object is used to deal with the server's response back to the browser;
3. The *Application* and *Session* objects are used to manage information about the application that is currently running and the unique instances of the application, which individual users run, known as sessions;
4. The ObjectContext object is used with Microsoft Transaction Server;
5. The Server object is used to provide several commonly used functions, by far the most important is its ability to create new objects or components.

Most of these objects will be utilized in our application.

3.1.2 ActiveX Data Objects

ADO is what is known as an application-level programming interface for database programming. It doesn't interact with databases directly, but instead interacts with a system-level programming interface called OLE DB. Figure 5 illustrates how an ASP script interacts with databases through an ADO interface.

ADO itself is a COM component, and therefore can be used in any COM compliant languages such as C++, Visual Basic, Java or JavaScript and VBScript. ODBC (Open Database Connectivity) provides an interface for applications to access relational databases, while OLE DB can access both relational and non-relational data sources. Eventually OLE DB should be able to replace ODBC, but for now it sits on top of ODBC and allows you to use existing ODBC drivers.

ADO has three core objects: *Connection, Command* and *RecordSet*. The *Connection* object is used to make a connection to the data store. The *Command* object is designed to run commands against the database by providing the SQL statement or Stored Procedure

along with any parameters. The *RecordSet* object is used to hold the data returned by a query. It has many properties and methods and is the most used object in ADO.

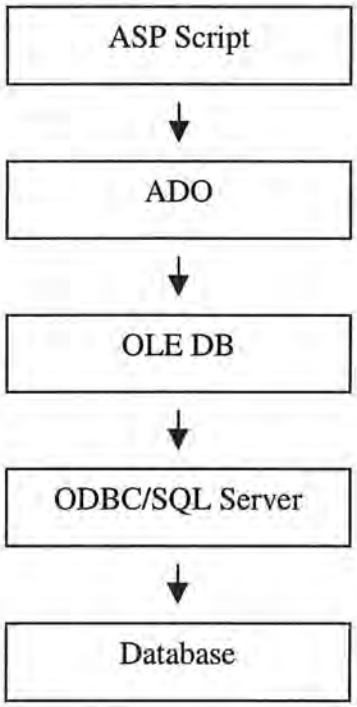


Figure 5. The interaction between ASP and a Database through ADO

3.2 Implementation of the web site

We have divided this development into three main sections, or groups of web pages. These three sections are the include files, logging on and registration, and web site user options.

3.2.1 The include files

In order to keep the code in our web pages at a minimum, we will make extensive use of include files. These contain common functionality that is used in a number of places. Seven include files are implemented. Three of them will be described in detail below. The other four include files are used for disconnection from the database, common functions, menu options and ADO constants.

I. Error handling

This include file (ErrorHandler.inc) will be included at the top of every web page that uses the database. The file contains a function called *CheckForErrors* that will be called after every call to the database. It will be used after connecting to the database, and after executing action stored procedures and simple-selects that are used in recordsets. The code for ErrorHandler.inc file is listed below.

```
<script language=vbscript runat=server>
Function CheckForErrors(objConnection)
    'Declare variables
    Dim blnDisplayErrMsg

    If objConnection.Errors.Count > 0 Then

        'Create the FileSystemObject and open the error log
        Set objFile = Server.CreateObject("Scripting.FileSystemObject")
        Set objLog = objFile.OpenTextFile( _
            Server.MapPath("ProductionErrorLog.txt"),8,True)

        'Check for an open error from VBScript
        If Err.Number > 0 Then
            Response.Write "Error opening log file<P>"
            Response.Write "Error Number: " & Err.Number & _
                ", Error Description: " & Err.Description
        End If

        'Create an error object to access the ADO errors collection
        Set objErr = Server.CreateObject("ADODB.Error")

        'Log all errors to the error log
        For Each objErr In objConnection.Errors
            If objErr.Number = 0 Then
                blnDisplayErrMsg = False
            Else
                objLog.WriteLine(objErr.Number & "|" & _
                    objErr.Description & "|" & objErr.Source & "|" & _
                    objErr.SQLState & "|" & objErr.NativeError)
                blnDisplayErrMsg = True
            End If
        Next

        'Close the log file and dereference all objects
        objLog.Close
        Set objLog = Nothing
        Set objFile = Nothing
        Set objErr = Nothing

        If blnDisplayErrMsg Then
            'Display a graceful message to the user
            Response.Write "An unforseen error has occurred and processing " & _
                "must be stopped. You can try your request again later or " & _
```

```
                "you can call our Help Desk at 888-888-1234"
            'Halt Execution
            Response.End
        End If
    End If
End Function
</script>
```

Notice that *Errors* received from ADO are handled by ADO *Errors Collection* of the *Connection* object. *Connection* and *Error* are actual objects while *Errors* is actually a collection of the *Error* objects. If errors exist, a textfile is created to log all of the errors. The file is then opened using the *OpenTextFile* method of the *FileSystemObject*. The Server.MapPath function is used to place the ProductionErrorLog.txt file in the same directory as the web pages. An ADO *Error* object is created to access the ADO *Errors collection*. All the information about ADO errors from the *Errors collection* is logged into a ProductionErrorLog.txt file using the *WriteLine* command, and we display a graceful message for the user, letting them know we had an error. Once the message is displayed, the further execution of the code will be stopped by the *End* command of ASP *Response* object.

The other error object (Err) in the above code is used to check for an open error from VBScript. It ensures that the file has indeed been opened and no error is received. Since the ASP object model doesn't support error handling, script errors are handled by the built-in error object of the script language. This VBScript *Error* object is very limited in that it only supports the "On Error Resume Next" statement. If a script error is received, it is written to the web page using the *Write* command of ASP *Response* object.

II. Authentication Checking

One of the security features used to prevent unauthorized access to our web pages is to see whether the user has logged in and has been authorized. We check to see if a *Session* variable called Authenticated has been set to a value of true. If the value has not been set to a value of true, we redirect the browser to the Default.asp web page using the *redirect* command of the *Response* object. The code for AuthenticationCheck.inc is listed as below.

```
<%
'Authentication check
If Session("Authenticated") <> True Then
    Session("ErrorMessage") = "You Have not properly logged in."
    Response.Redirect "Default.asp"
End If
%>
```

III. Connecting to the database

The database is going to be connected in the same manner most of the time. It makes sense to place this common code in an include file. The connect.inc file, shown below, sets up our script error handling with the On Error Resume Next statement, and then an ADO *Connection* object is created and the database is opened. We are using an application level variable to hold our connection string. This variable is defined in the Global.asa file, which will be discussed next.

```
<%
'Instruct VBScript to ignore the error and continue
'with the next line of code
On Error Resume Next

'Create and open the database object
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open Application("ConnectString")
%>
```

3.2.2 Global.asa File

Global.asa is an optional file in which we can declare objects and variables that have application and session level scope. This file must reside in the root directory of our web application. We use *Application_OnStart* procedure to set the ConnectString variable, which will be used by all web pages needing to connect to the database. The code in this procedure gets executed when the application is accessed for the first time and the variables that are set stay active the entire time. The code for Global.asa is listed follow.

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
Sub Session_OnStart
End Sub

Sub Session_OnEnd
End Sub
```

```
Sub Application_OnStart
    Application("ConnectString") = "DSN=simulation"    //DSN: ODBC Data Source Name
End Sub

Sub Application_OnEnd
    Application("ConnectString") = ""
End Sub
</SCRIPT>
```

3.2.3 Logging on and Registration

We want this web site to have minimum security. However, we want our users to be automatically logged in if we find a cookie on their machine. If we don't find a cookie then we will prompt the user for their login values. If the user has not registered before, a hyperlink will be provided on the login page to a registration page. Figure 6 illustrates the flow of the login process. The implementation of a few web pages shown in the illustration is discussed as follows.

I. Default.asp page

The Default.asp web page is shown in Figure 7. It has three functions. First it checks to see if a cookie exists on the user's machine. If it does, it will redirect the browser to the WelcomeBack.asp page. The second function is to display a login form if no cookie has been found. The last function is to provide a hyperlink to the Registration.asp page if the user has not registered before. Below is the code for checking for a cookie and redirecting the web page using the ASP *Response* object.

```
<%
'Check to see if a cookie exists for this user

If Len(Request.Cookies("Simulation")("UserName")) > 0 Then
    'Cookie exists

    'Authenticate the user for other web pages
    Session("Authenticated") = True

    'Redirect the browser to the welcome back page
    Response.Redirect "WelcomeBack.asp"
End If
%>
```

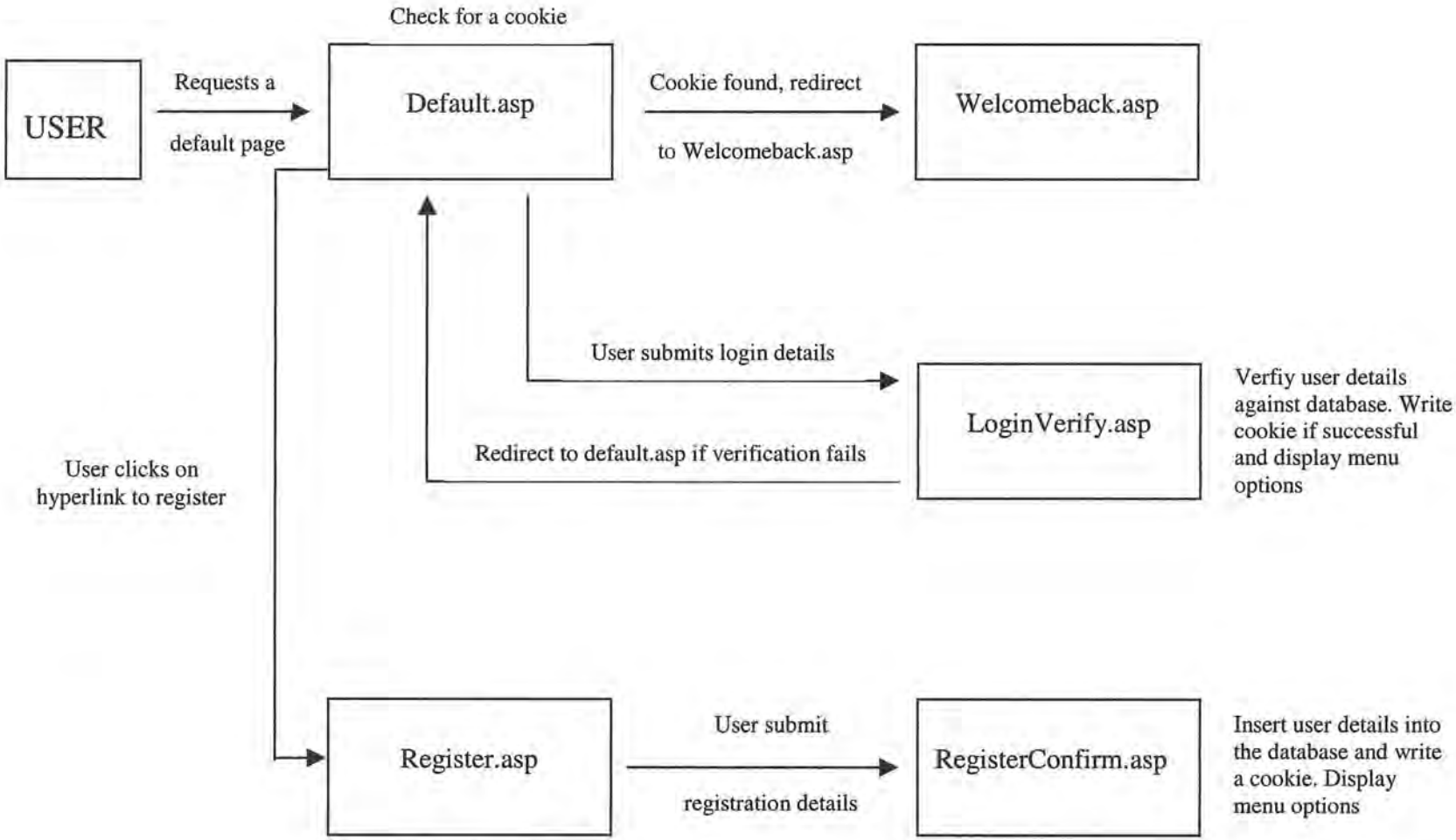


Figure 6. Flowchart of the user login process

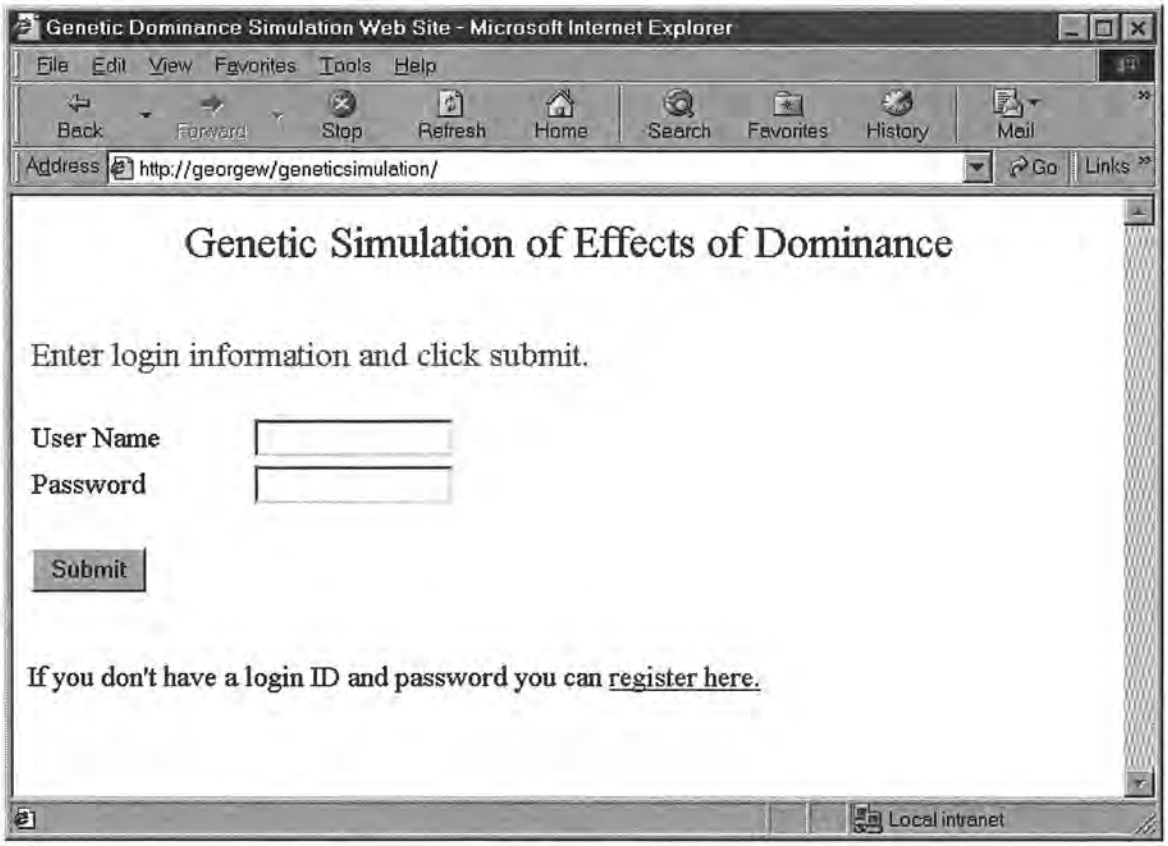


Figure 7. The Default web page

II. Register.asp Page

This page allows us to enter all of the required user information, and eventually click on a submit button which will submit all user details to be recorded in the database by the RegisterConfirm.asp page. Two steps for the Register.asp page are shown in Figure 8.

We use this Register.asp page to show how multiple steps can be processed in a single Active Server Page. In step one we have set up a hidden field on our form called FormAction. This field contains the value of the next step to process in the page. The first time we access this page, this field does not exist and the code in step one is executed. We start building a form (FrmRegister1) by specifying that the form will post the results to the Register.asp page, which is the current page. We then start building a table to input user information. Clicking the "Continue" button causes the page to be executed again, but this time the field FormAction has a value of step2 and this section of the code is executed. The user will get their information from step one, input additional information

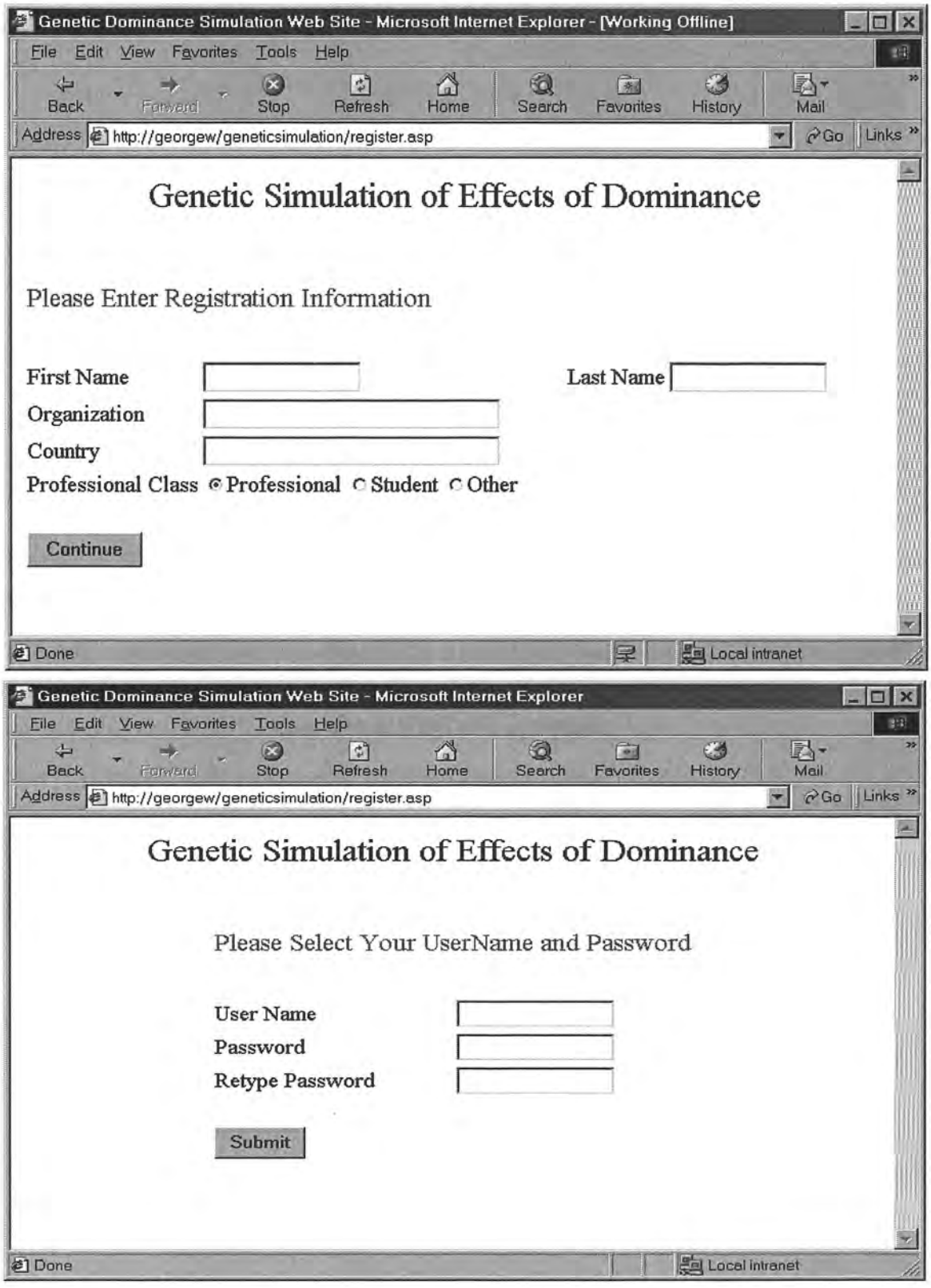


Figure 8. User registration interface

on form FrmRegister2, and then submit the data input from two sections to the RegisterConfirm.asp page. The following lists the partial code from step one and step two.

```
<%
'*************************************************************
'Step1: Display the registration form for user input
'*************************************************************
If len(Request.Form("FormAction")) = 0 then
%>
<form action=register.asp method=post name=FrmRegister1>
<Input type=hidden Name=FormAction value=step2>
<table>
      <tr >
      <td height=50 colspan=2><font size="4" color=teal>Please Enter Registration Information
            </font></td>
      </tr>
      <tr>
      <td> </td>
      </tr>
      <tr >
      <td>First Name</td>
....
<%
'*********************************************************************
'Step2: Continue to enter user information: UserName and Password
'*********************************************************************
ElseIf Request.Form("FormAction") = "step2" then
%>

<!-- #include file="Connect.inc" -->

<%
'Get the registration information entered in step1
Dim txtFirstName, txtLastName, txtOrganization, txtCountry, optClass
txtFirstName = Request.Form("txtFirstName")
txtLastName = Request.Form("txtLastName")
....

%>
<form action=registrationConfirmation.asp method=post name=FrmRegister2>
....
<table align=center>
      <tr >
      <td height=50 colspan=2><font size="4" color=teal>Please Select Your UserName and Password
            </font></td>
      </tr>
      <tr>
      <td> </td>
      </tr>
      <tr>
      <td >User Name</td>
....
%>
```

Before the user submits the form for processing, we use client-side scripts to validate all data that has been entered correctly. For example, when the user clicks the Submit button, the following script will be executed. If the user has not entered any data or entered data incorrectly, we prompt them to reenter some data using the Alert function. After all data are validated, the *Submit* method is called on the form FrmRegister2. This submits the values entered, and calls the page that is specified in the ACTION property of the form.

```
<Script Language=VbScript>
Sub btnSubmit_OnClick()
<%
    do while Not objRS.EOF
%>
    'Verify if other users have used this user name
    If frmRegister2.txtUserName.value="<%=objRS("UserName")%>" Then
        Alert "This user name has been taken"
        frmRegister2.txtUserName.focus
        Exit Sub
  end if
<%
    objRS.MoveNext
    loop
%>
    'Verify all fields that have been entered
    If Len(frmRegister2.txtUserName.value) = 0 Then
        Alert "You must enter a user name"
        frmRegister2.txtUserName.focus
        Exit Sub
    ElseIf Len(frmRegister2.txtPassword.value) = 0 Then
        Alert "You must enter a password"
        frmRegister2.txtPassword.focus
        Exit Sub
    ElseIf Len(frmRegister2.txtRetypePassword.value) = 0 Then
        Alert "You must retype your password"
        frmRegister2.txtRetypePassword.focus
        Exit Sub
    ElseIf frmRegister2.txtPassword.value <> frmRegister2.txtRetypePassword.value then
      Alert "Retyped password doesn't match"
      frmRegister2.txtPassword.value = ""
      frmRegister2.txtRetypePassword.value = ""
      frmRegister2.txtPassword.focus
      Exit Sub
    End If

    'If we get to this point all is OK, submit the form
    Call frmRegister2.submit()
End Sub
</script>
```

III. RegisterConfirm.asp Page

The RegisterConfirm.asp page reads all of the form fields from the request form which is the Register.asp page. It builds the parameters to a stored procedure and executes that stored procedure, writing all of the information to the database. Upon successful completion of this it writes a cookie to the user's machine, and gives them a list of options to execute. The page looks like the following screenshot in Figure 9.

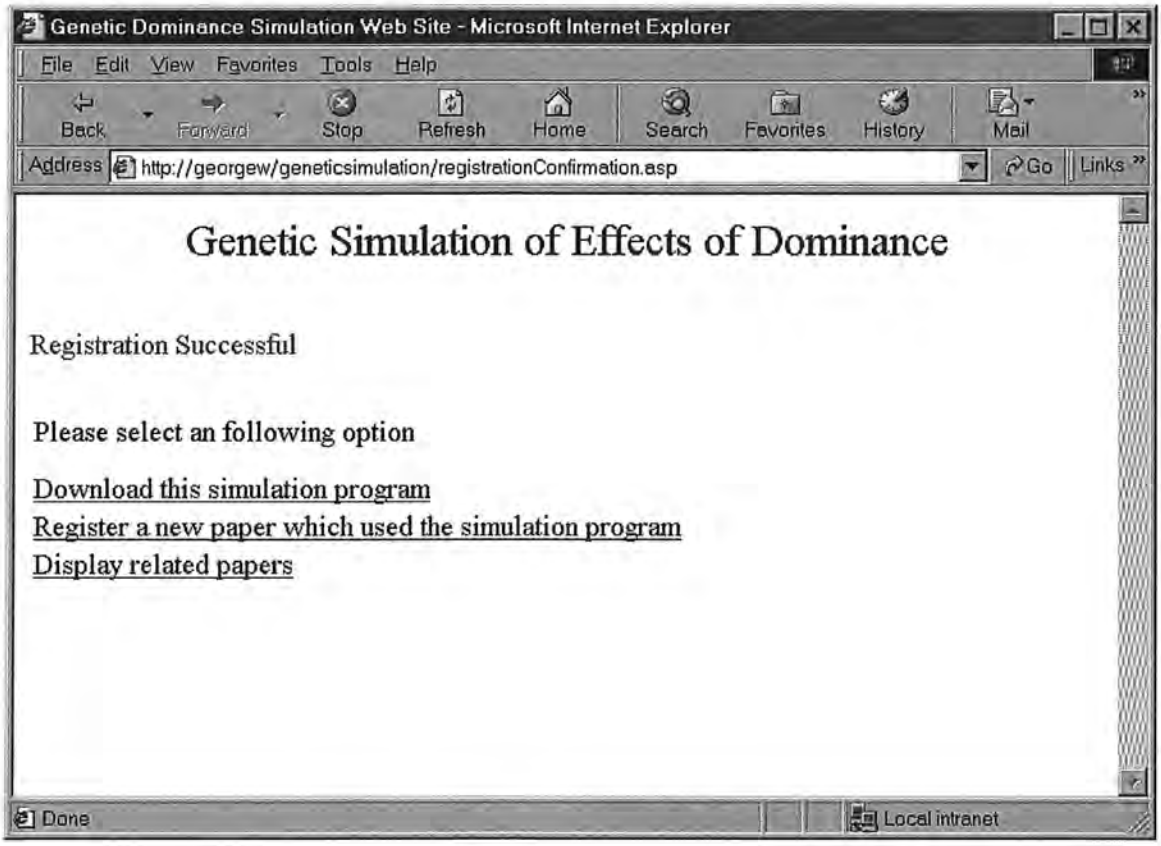


Figure 9. Screenshot for Registration confirmation page

Stored procedures have a number of benefits. They provide better maintainability since code for stored procedures resides in the database. They are highly reusable since multiple web pages and/or programs can use the same stored procedure, and they can be used for implementing business rules. Stored procedures can also greatly improve the application performance since they are considered compiled once they are created in the database, and they reduce the network traffic by the use of fewer in-line SQL statements.

Several stored procedures are used in this application for selecting records and updating databases. We use this web page to demonstrate how an Active Server Page calls stored procedures. Following is the code for the stored procedure *qInsertPerson*.

```
PARAMETERS FirstName Text, LastName Text, Organization Text, Country Text, ProfessionalClass
Long, UserName Text, Password Text;
INSERT INTO People ( FirstName, LastName, Organization, Country, ProfessionalClass, UserName,
Password )
SELECT [FirstName] AS Expr1, [LastName] AS Expr2, [Organization] AS Expr3, [Country] AS Expr4,
[ProfessionalClass] AS Expr5, [UserName] AS Expr6, [Password] AS Expr7;
```

First we define parameters specifying the parameter name and data type. Next we build the INSERT statement specifying the table and column into which we are inserting data. Finally we SELECT the parameters as expressions. In the RegisterConfirm.asp file, We then build all parameters in the SQL string, call the stored procedures and execute the SQL string using the ADO *Connection* object. The code is listed as follow.

```
<!-- #include file="Connect.inc" -->
<%
'Check for database errors
Call CheckForErrors(objConn)

'Set the parameters for the insert stored procedure
strSQL = "qInsertPerson ('" & CStr(Request.form("txtFirstName")) & _
      "','" & CStr(Request.form("txtLastName")) & _
      "','" & CStr(Request.form("txtOrganization")) & _
      "','" & CStr(Request.form("txtCountry")) & _
      "'," & CLng(Request.form("optClass")) & _
      ",'" & CStr(Request.Form("txtUserName")) & _
      "','" & CStr(Request.Form("txtPassword")) & "')"
'Execute the stored procedure to insert the person
objConn.Execute strSQL,,adCmdStoredProc

'Check for database errors
Call CheckForErrors(objConn)
%>
```

After the user record is inserted into the database, we write a cookie on the user machine using the ASP *Response* object. The cookie is then set to expire on December 31 of the current year. The user is then authenticated for access to other web pages. The code is listed as follow.

```
<%
'Save the user information to a cookie
Response.Cookies("Simulation")("UserName") = Request.Form("txtUserName")
Response.Cookies("Simulation")("Password") = Request.Form("txtPassword")
```

```
'Set the expiration date of the cookie to the last day of the current year
Response.Cookies("Simulation").Expires = "December 31, " & Year(Now)

'Authenticate the user for other web pages
Session("Authenticated") = True
%>
```

3.2.4 Web Site User Options

The Options.asp page contains the options that are available for the user. They can click on any of the hyperlinks to jump to another page in the web site. There are currently three options: download the simulation program, register a paper and display the related papers. The screenshot is shown in Figure 10.

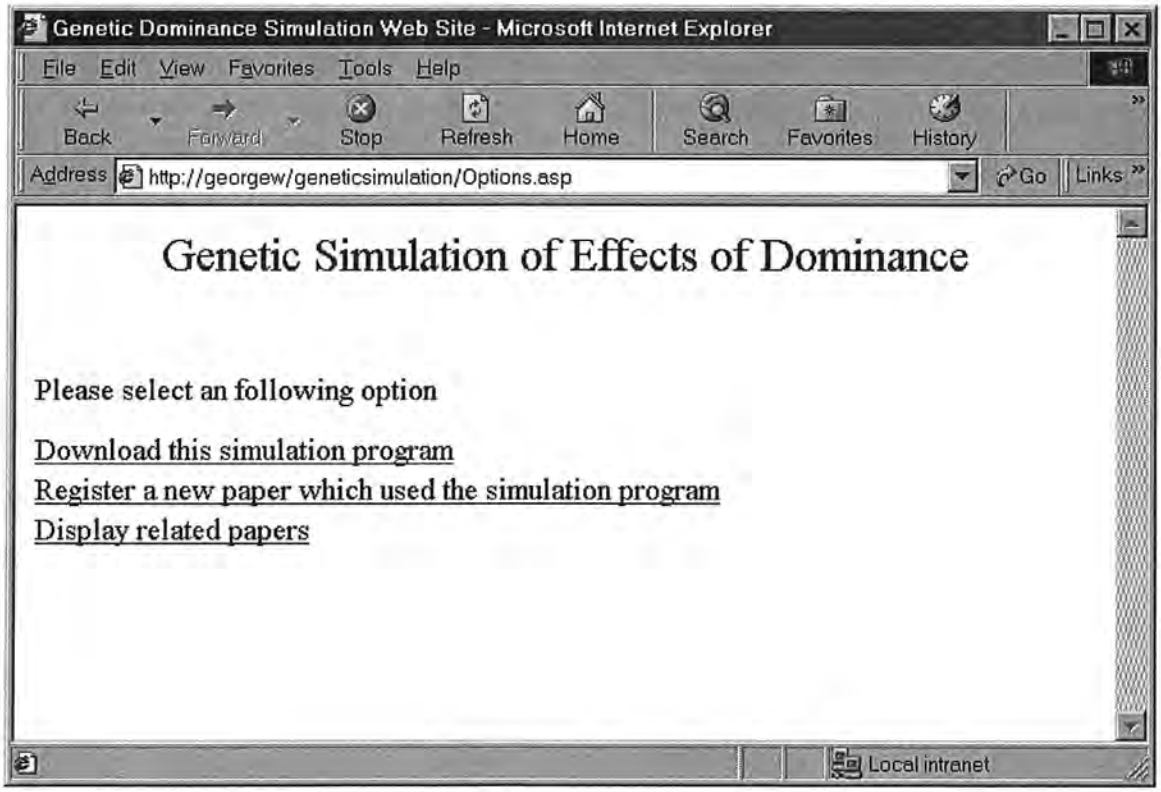


Figure 10. User options interface

The option for downloading a program is simply linked to a download.html file to download an executable simulation program. We will discuss the web pages of other two options here.

I. RegisterNewPaper.asp Page

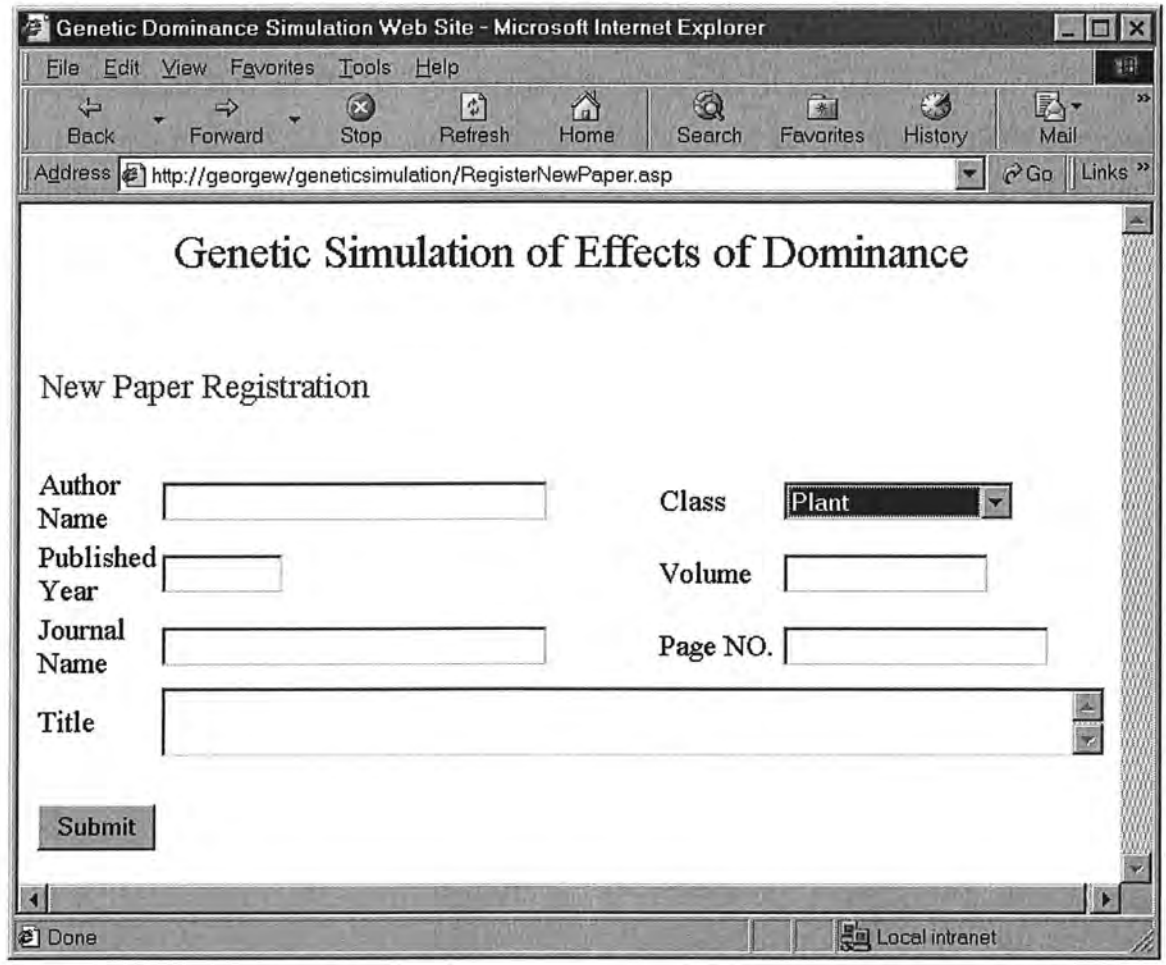
This web page allows a user to register their published paper utilizing the simulation program on the web site. This page also includes two steps. It contains a form that the user can fill in with their paper information. When they click on the submit button, the form will post the data back to the RegisterNewPaper.asp page, which will then be recorded into the database for completion of the registeration. This is shown in the following two screenshots in Figure 11.

The common code for this page includes four include files at the top of the page. The first include file adds ADO constants and the second file checks to see if the user has been authenticated and redirects the browser to the Default.asp page if they have not. The third include file contains some common functions that can be shared with other pages., and the last include file contains error handling functions. These include files were put into the top of web page as follows.

```
<!-- #include file="adovbs.inc" -->
<!-- #include file="AuthenticationCheck.inc" -->
<!-- #include file="CommonFunctions.inc" -->
<!-- #include file="ProductionErrorHandler.inc" -->

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>
....
```

This page calls two stored procedures. The first is *qAllPaperClasses* which selects all of the different paper classes from the Paper table in the database. An ADO *Recordset* is created and filled with the data selected from the stored procedure. The code for opening a recordset through the stored procedure is listed as follow.

```
<%
'Check for database errors
Call CheckForErrors(objConn)

'Create the recordset object and open the recordset
Set objRS = Server.CreateObject("ADODB.Recordset")
strSQL = "qAllPaperClasses"
objRS.Open strSQL, objConn, adOpenForwardOnly, , adCmdStoredProc

'Check for database errors
Call CheckForErrors(objConn)
%>
```



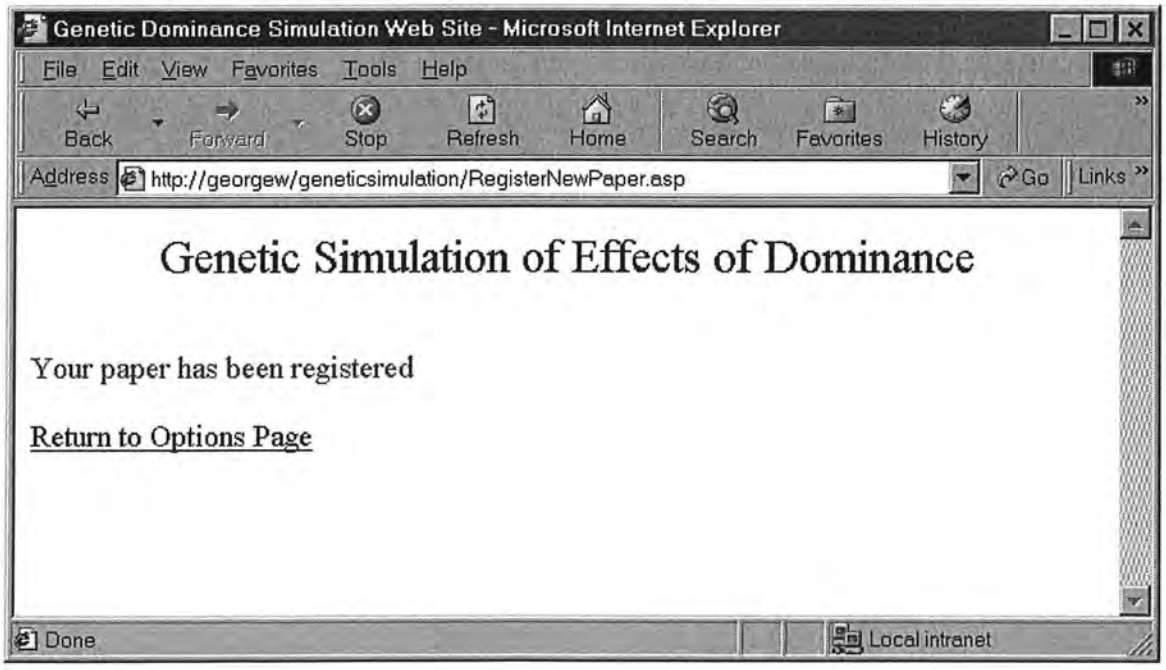


Figure 11. Screenshots for paper registeration

This recordset (objRS) is then used to load data into a paper class combox by a special Window_OnLoad event coded as client-side script. This is the first event to fire when the page loads on the user's browser. An Option element is created and assigned with text and value from the recordset. The option is then added to the combo box. After executing this event, we will have a combo box with a drop down list of all the available paper class names by looping through the recordset. The code for Window_OnLoad event is listed as follows.

```
<script language=vbscript>
Sub Window_OnLoad()
<%
    Do While Not objRS.EOF
%>
        Set objOption = document.createElement("OPTION")
        objOption.text = "<%=objRS("ClassName")%>"
        objOption.value = "<%=objRS("ClassName")%>"
        document.all.cboClass.add objOption
<%
        objRS.MoveNext
    Loop
%>
    <!-- #include file="Disconnect.inc" -->
    Set objOption = Nothing
End Sub
```

The second stored procedure is *qInsertPaper* which is to insert the new paper details into the Paper table. Following is the code for setting the SQL string with the name of the stored procedure to execute and pass it the appropriate parameters. Once the SQL string has been built, it is executed through the ADO *Connect* object to insert the new paper to the Paper table. Since all string parameters must be enclosed in single quote marks, the common function ConvertString is used to replace all single quote marks with two consecutive single quote marks.

```
<%
    'Check for database errors
    Call CheckForErrors(objConn)

    'Run the paper title through the string conversion routine
    'just in case there are any single quotes
    strTitle = ConvertString(Request.Form("txtTitle"))

    'Set the parameters for the insert stored procedure
    strSQL = "qInsertPaper ('" & CStr(Request.Form(txtAuthorName) )& _
        "','" & CStr(Request.Form("cboClass")) & _
        "'," & CLng(Request.Form("txtPublished")) & _
```

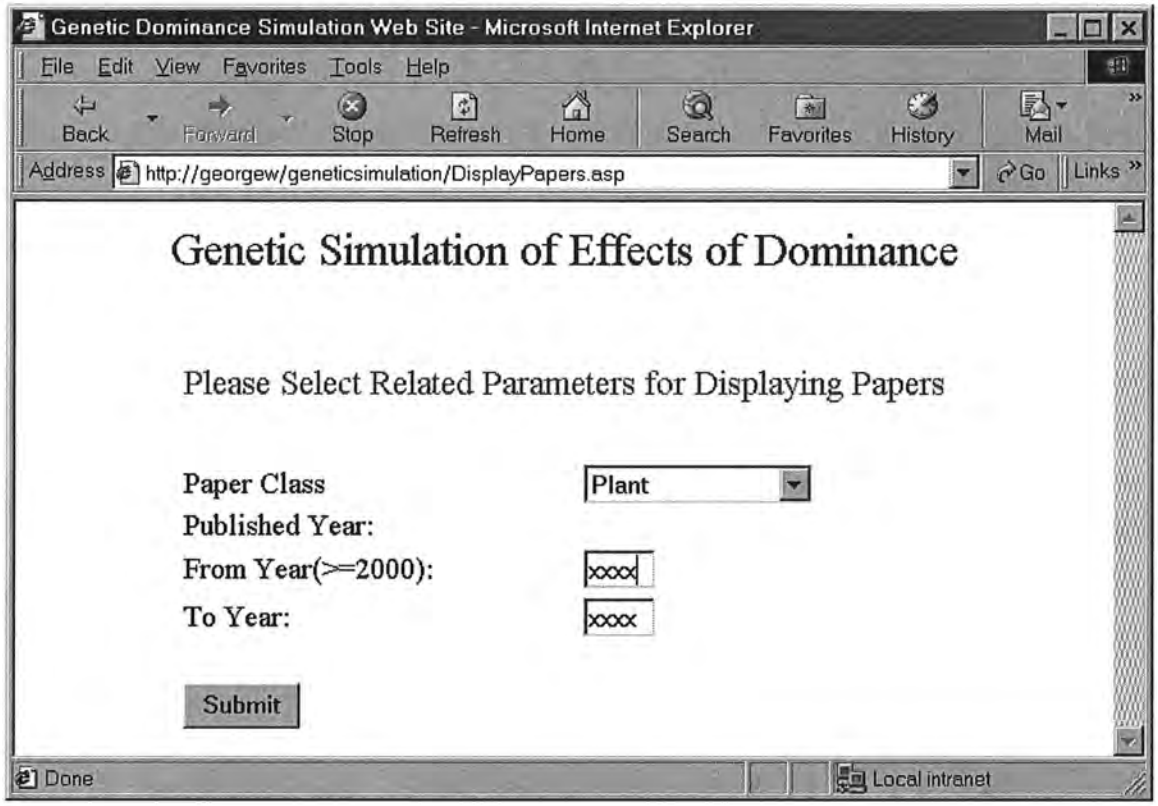
```
        ","" & CStr(Request.Form("txtVolume")) & _
        ""," & CStr(Request.Form("txtJournalName")) & _
        ""," & CStr(Request.Form("txtPageNo")) & _
        ""," & CStr(strTitle) & "")"
    'Insert the new boat
    objConn.Execute strSQL,,adCmdStoredProc

    'Check for database errors
    Call CheckForErrors(objConn)
%>
```

II. DisplayPapers.asp Page

This is the last web page in our web site and this page selects all papers from the database based on selection parameters, and displays the data in a table formatted with multiple table headers and rows. Thus this page is also divided into two steps. The first step is to present a form for the user to input selection parameters for related published papers. The second step is then to display the paper information by searching through the database. Two screenshots are shown in Figure 12.

In this web page, the stored procedure *qAllPaperClasses* is again used to populate a paper class combo box for the user to select the type of paper they need. The other stored procedure called in this procedure is *qDisplayPapers* which is to select all papers based on the user's criteria. Its code is listed as follows.

```
PARAMETERS Class Text, FromYear IEEEDouble, ToYear IEEEDouble;
SELECT paper.AuthorName, paper.paperClass, paper.publishedYear, paper.JournalVolume,
paper.JournalName, paper.PageNo, paper.PaperTitle
FROM paper
WHERE (((paper.paperClass)=[Class]) and ((paper.publishedYear)>=[FromYear]) and
((paper.publishedYear)<=[ToYear]))
ORDER BY paper.publishedYear, paper.AuthorName;
```



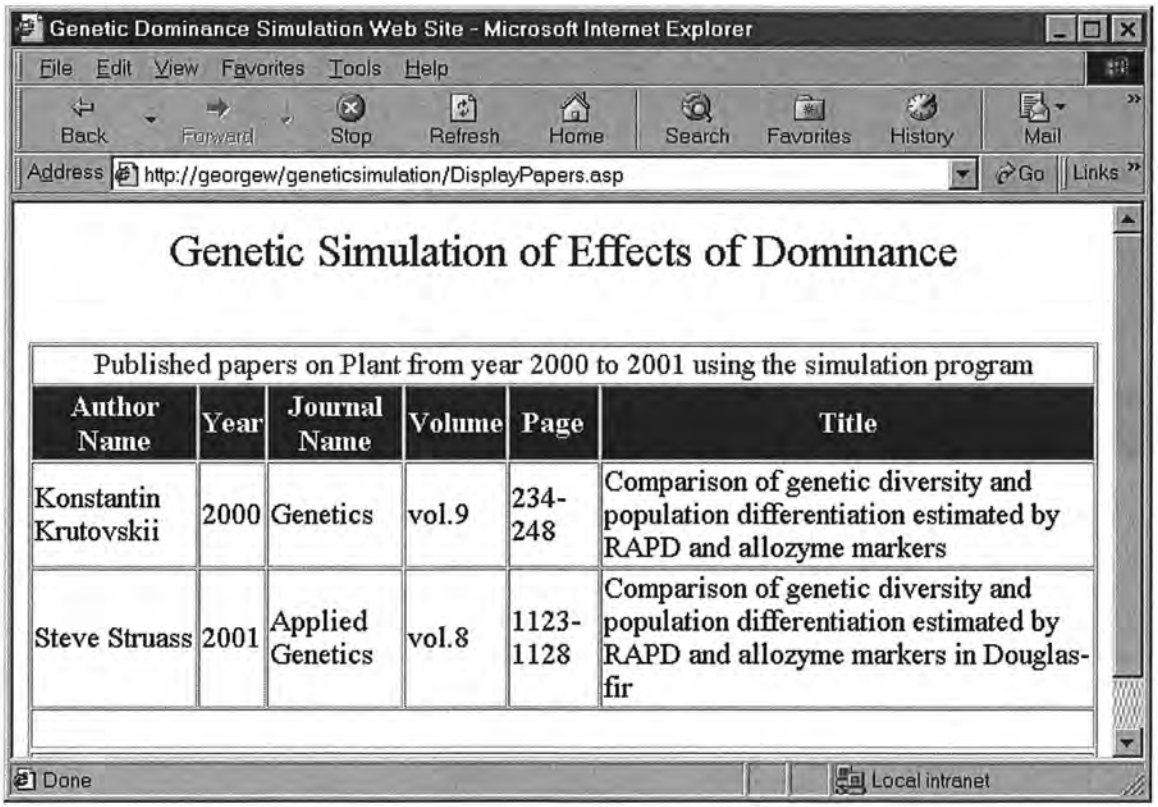


Figure 12. Screenshots for paper display

In this page we used an ADO *Command* object to pass value to the above stored procedure. The code is listed as follow. We first create an ADO *Command* object, then set its ActiveConnection property to the *Connection* object that we have created. Using the stored procedure we created earlier, we set the commandText property of the Command object to call the stored procedure and pass all of the parameters. Finally, a *Recordset* object is created to hold the records that will be returned when we execute the *Command* object. All of the records held in the recordset will be displayed in our web page.

```
<%
'Check for database errors
Call CheckForErrors(objConn)

Set objCmd = server.CreateObject("ADODB.command")
Set objCmd.ActiveConnection=objConn
objCmd.CommandText="{Call qDisplayPapers ('" &strPaperClass & _
"'," &intFromYear & "," &intToYear &")}"
Set objRS = objCmd.execute

'Check for database errors
Call CheckForErrors(objConn)
%>
```

4. Conclusions and Future Work

The simulation program we implemented can quantitatively determine how the dominance and biallelism could affect the estimation of population genetic statistics. It will thus allow us to make a more accurate comparison of estimation of genetic parameters between dominant and codominant markers. Our results demonstrate the importance of simulations to help compare and interpret the results of population studies with dominant markers. Thus the simulation program is currently highly needed by the genetics society. The dynamic web site generated by Active Server Pages in combination with Active Data Objects (ADO) provides the users with an opportunity to register their information and papers into the database, search for other related papers from the database, and download the simulation program.

Although the simulation program and the web site meet our initial requirements, they can be improved in some areas in the future. Currently our backend database is Microsoft Access. Its maximum number of concurrent users is 255, and in many cases you would only be able to manage much smaller numbers. If the web site becomes popular in the

future, the database can be moved to a SQL server system that will have better performance than the Access database. All script files should be reusable with minimal modifications. In addition, users may require more functionality and features from the simulation program, advanced versions can be developed accordingly and distributed through the web site.

5. References

Aagaard JE, Krutovskii KV, and Strauss SH, 1998. RAPDs and allozymes exhibit similar levels of diversity and differentiation among populations and races of Douglas-fir. Heredity 81:69-78.

Homer Alex, 2000. Alex Homer's Professional ASP 3.0 Web Techniques. Wrox Press Inc. Chicago, USA.

Baruffi L, Damiani G, Guglielmino CR, Bandi C, Malacrida AR, and Gasperi G, 1995. Polymorphism within and between populations of *Ceratitis capitata*: comparison between RAPD and multilocus enzyme electrophoresis data. Heredity 74:425-437.

le Corre V, Dumolin-LapПgue S, and Kremer A, 1997. Genetic variation at allozyme and RAPD loci in sessile oak *Quercus petraea* (Matt.) Liebl.: the role of history and geography. Mol Ecol 6:519-529.

Dawson IK, Simons AJ, Waugh R, and Powell W, 1996. Diversity and genetic differentiation among subpopulations *Gliricidia sepium* revealed by PCR-based assays. Heredity 74:10-18.

Heun M, Murphy JP, and Phillips TD, 1994. A comparison of RAPD and isozyme analyses for determining the genetic relationships among *Avena sterilis* L. accessions. Theor Appl Genet 87:689-696.

Isabel N, Beaulieu J, and Bousquet J, 1995. Complete congruence between gene diversity estimates derived from genotypic data at enzyme and random amplified polymorphic DNA loci in black spruce. Proc Natl Acad Sci USA 92:6369-6373.

Krutovskii KV, Vollmer SS, Sorensen FC, Adams WT, Knapp SJ, and Strauss SH, 1998. RAPD genome maps of Douglas-fir. J. Heredity 89:197-205.

Lannϑr-Herrera C, Gustafsson M, and Bryngelsson T, 1996. Diversity of wild *Brassica oleraceae* as estimated by isozyme and RAPD analysis. Genetic Resources and Crop Evolution 43:13-23.

Latta RG and Mitton JB, 1997. A comparison of population differentiation across four classes of gene marker in limber pine (*Pinus flexilis* James). Genetics 146:1153-1163.

Lewis PO, 1994. GeneStat-PC 3.3. Department of Statistics, North Carolina State University, Raleigh, North Carolina.

Li P and Adams WT, 1989. Range-wide patterns of allozyme variation in Douglas-fir (*Pseudotsuga menziesii*). Can J For Res 19:149-161.

Liu Z and Furnier GR, 1993. Comparison of allozyme, RFLP, and RAPD markers for revealing genetic variation within and between trembling aspen and bigtooth aspen. Theor Appl Genet 87:97-105.

Lynch M and Milligan BG, 1994. Analysis of population genetic structure with RAPD markers. Mol Ecol 3:91-99.

Mosseler A, Egger KN, and Hughes GA, 1992. Low levels of genetic diversity in red pine confirmed by random amplified polymorphic DNA markers. Can J For Res 22:1332-1337.

Nei M, 1973. Analysis of gene diversity in subdivided populations. Proc Natl Acad Sci USA 70: 3321-3323.

Nei M, 1986. Definition and estimation of fixation indices. Evolution 40:643-645.

Nei M and Chesser RK, 1983. Estimation of fixation indices and gene diversities. Ann Hum Genet 47:253-259.

Peakall R, Smouse PE, and Huff DR, 1995. Evolutionary implications of allozyme and RAPD variation in diploid populations of dioecious buffalograss *BuchloN dactyloides*. Mol Ecol 4:135-147.

Puterka GJ, Black IV WC, Steiner WM, and Burton RL, 1993 Genetic variation and phylogenetic relationships among worldwide collections of the Russian wheat aphid, *Diuraphis noxia* (Mordvilko), inferred from allozyme and RAPD-PCR markers. Heredity 70:604-618.

Szmidt AE, Wang X, and Lu M, 1996. Empirical assessment of allozyme and RAPD variation in *Pinus sylvestris* (L.) using haploid tissue analysis. Heredity 76:412-420.

Travis SE, Maschinski J, and Keim P, 1996. An analysis of genetic variation in *Astragalus cremnophylax* var. *cremnophylax*, a critically endangered plant, using AFLP markers. Mol Ecol 5:735-745.

Vos P, Hogers R, Bleeker M, Reijans M, van de Lee T, Hornes M, Frijters A, Pot J, Peleman J, Kuiper M, and Zabeau M, 1995. AFLP: a new technique for DNA fingerprinting. Nucl Acids Res 23:4407-4414.

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

Williams JG, Kubelik AR, Livak KJ, Rafalski JA, and Tingey SV, 1990. DNA polymorphisms amplified by arbitrary primers are useful as genetic markers. Nucl Acids Res 18:6531-6535.

Wu J, Krutovskii KV, and Strauss SH, 1999. Nuclear DNA diversity, population differentiation and phylogenetic relationships in the California closed-cone pines based on RAPD and allozyme markers. Genome 42: 893-908.

6. Appendix: Source code

6.1. Genetic dominance simulation program

```
'variable names must be explicitly declared before use.
Option Explicit

'first element in array starts at index 1
Option Base 1

'inputFile is an input file which imports original allele frequency
'outputFile is an output file which outputs the calculated results
Public inputFileName As String, outputFileName As String

'basePopSize: size of generated basic populations
'subPopSize: the number of individuals sampled from the population
'Nsam: the number of times the generated base population is sampled _
'Nresam: the number of times the sampled subpopulations is resampled _
      by bootstrap resampling

Public basePopSize As Integer, subPopSize As Integer, _
    Nsam As Integer, Nresam As Integer

'Npop: number of populations; Nloc: Number of locus _
Nalle: number of alleles; MaxNalle: maximum number of alleles among all loci
Public Npop As Integer, Nloc As Integer
Dim Nalle() As Integer, MaxNalle As Integer

'Options to calculate genetic diversity and differention
Dim diversityOption As Integer, differenOption As Integer

'Allele frequency
Dim AlleFreq() As Double

'two alleles at each locus for individuals in generated base populations
Dim allele1() As Integer, allele2() As Integer

'allele frequencies based on generated base populations
Dim baseAlleFreq() As Double

'Allele1 and allele2 for the individuals in the sampled populations
Dim samAllele1() As Integer, samAllele2() As Integer
'Allele1 and allele2 for the individuals in the bootstrap-resampled populations
Dim resamAllele1() As Integer, resamAllele2() As Integer

'nth population number, e.g. popNo(1), popNo(2)
'Dim popNo() As Integer

'used for random number generator
Public ranInt1 As Integer, ranInt2 As Integer, ranInt3 As Integer

'used for storing scrollbar values
Dim x1 As Integer, x2 As Integer, x3 As Integer, x4 As Integer

Private Sub Form_Load()
TxtPopSize.Text = 1000
TxtSubPopSize.Text = 20
TxtNoSam.Text = 200
TxtNoResam.Text = 200
x1 = x2 = x3 = x4 = 0
'Open "c:\simulation\options1.prg" For Output As #5
```

```
End Sub

Private Sub mnuAbout_Click()
Dim Aboutstr As String, crlf As String
'Chr$(13):carriage return; Chr$(10):newline
crlf = Chr$(13) + Chr$(10)
Aboutstr = "Population genetic dominance simulation program" & crlf _
& "Designed and programmed by: Junyuan Wu junyuan@cs.orst.edu" & crlf
MsgBox Aboutstr, vbOKOnly, "About the program"
End Sub

Private Sub mnuHelp_Click()
Dim HelpStr As String, crlf As String
'Chr$(13):carriage return; Chr$(10):newline
crlf = Chr$(13) + Chr$(10)
HelpStr = "How to use the system:" & crlf _
& "1. Select an input datafile under file menu" & crlf _
& "2. Select an output datafile under file menu" & crlf _
& "3. Run calculation under run menu" & crlf _
& "4. Wait until the message 'calculation finished' shows up" & crlf _
& crlf _
& "The structure of input datafile (see the format of sample.dat):" & crlf _
& "First line: the number of populations" & crlf _
& "Second line: the number of loci" & crlf _
& "Third line: the number of alleles for each locus" & crlf _
& "Follwoing lines: allele frequencies in each locus" & crlf
MsgBox HelpStr, vbOKOnly, "Help"
End Sub

Private Sub SimuFrame_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub TxtPopSize_Change()
Dim intpress As Integer
If CInt(TxtPopSize.Text) > 1000 Or CInt(TxtPopSize.Text) < 500 Then
   intpress = MsgBox("value should be between 500 and 1000", vbOKOnly, "population size")
End If
End Sub

Private Sub TxtSubPopSize_Change()
Dim intpress As Integer
If CInt(TxtSubPopSize.Text) > 200 Or CInt(TxtSubPopSize.Text) < 10 Then
   intpress = MsgBox("value should be between 10 and 200", vbOKOnly, "Subpopulation size")
End If
End Sub

Private Sub TxtNoSam_Change()
Dim intpress As Integer
If CInt(TxtNoSam.Text) > 400 Or CInt(TxtNoSam.Text) < 100 Then
   intpress = MsgBox("value should be between 100 and 400", vbOKOnly, "Number of Sampling")
End If
End Sub

Private Sub TxtNoResam_Change()
Dim intpress As Integer
If CInt(TxtNoResam.Text) > 400 Or CInt(TxtNoResam.Text) < 100 Then
   intpress = MsgBox("value should be between 100 and 400", vbOKOnly, "Number of Resampling")
End If
End Sub

Private Sub VScrollPopSize_Change()
```

```
If x1 < VScrollPopSize.Value Then
   TxtPopSize.Text = TxtPopSize.Text + 1
Else
   TxtPopSize.Text = TxtPopSize.Text - 1
End If

x1 = VScrollPopSize.Value
End Sub
Private Sub VScrollSubPop_change()
If x2 < VScrollSubPop.Value Then
   TxtSubPopSize.Text = TxtSubPopSize.Text + 1
Else
   TxtSubPopSize.Text = TxtSubPopSize.Text - 1
End If

x2 = VScrollSubPop.Value
End Sub
Private Sub VScrollNsam_change()
If x3 < VScrollNsam.Value Then
   TxtNoSam.Text = TxtNoSam.Text + 1
Else
   TxtNoSam.Text = TxtNoSam.Text - 1
End If

x3 = VScrollNsam.Value
End Sub
Private Sub VScrollNresam_change()
If x4 < VScrollNresam.Value Then
   TxtNoResam.Text = TxtNoResam.Text + 1
Else
   TxtNoResam.Text = TxtNoResam.Text - 1
End If

x4 = VScrollNresam.Value
End Sub
Private Sub mnuFileExit_Click()
Unload Me
End
End Sub

Private Sub mnuFileInput_Click()
'Dim inputFileName As String
cdbDialogOpen.DialogTitle = "Choose an input file"
cdbDialogOpen.Filter = "All Files(*.*)"
'cdbDialog.filename = "*.dat"
cdbDialogOpen.ShowOpen
inputFileName = cdbDialogOpen.InitDir + cdbDialogOpen.FileName
'Write #5, inputFileName

End Sub

Private Sub mnuFileOut_Click()
'Dim outputFileName As String
CdbDialogSave.DialogTitle = "Choose an output file"
CdbDialogSave.Filter = "All Files(*.*)"
'cdbDialog.FileName = "*.out"
CdbDialogSave.ShowSave
outputFileName = CdbDialogSave.InitDir + CdbDialogSave.FileName
'Write #5, outputFileName

End Sub
```

```
Private Sub mnuRunCalculate_Click()
'The number of items in the list box
Dim nCnt As Integer, intpress As Integer
For nCnt = 0 To DiversityList.ListCount - 1
   If DiversityList.Selected(nCnt) Then
      diversityOption = nCnt + 1
   Else
      diversityOption = 1 'default
   End If
Next

For nCnt = 0 To DifferentiationList.ListCount - 1
   If DifferentiationList.Selected(nCnt) Then
      differenOption = nCnt + 1
      nCnt = DifferentiationList.ListCount
   Else
      differenOption = 1 'default
   End If
Next

basePopSize = CInt(TxtPopSize.Text)
subPopSize = CInt(TxtSubPopSize.Text)
Nsam = CInt(TxtNoSam.Text)
Nresam = CInt(TxtNoResam.Text)

Call openFile

'Calculate theoretical values of Hs, Ht, Gst and theta
Call theoretical_value

'Generate populations with 1000 individuals each based on original _
allele frequency
Call popGenerate

'Calculate empirical values of Hs, Ht, Gst and theta based on allele frequencies in _
generated sets of 1000 individuals
Call empirical_value

Call sampling

Call Resampling

Close #2

intpress = MsgBox("The calculation is completed", vbOKOnly, "Finished running")

End Sub

Private Sub openFile()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim intpress As Integer

'Reading input data from inputFile
Do While inputFileName = ""
   intpress = MsgBox("You need to select an input data file", vbOKOnly, "Select input data file")
   Call mnuFileInput_Click
Loop

Open inputFileName For Input As #1
'Open inputFile For Input As #1
```

```
Input #1, Npop, Nloc

'declare dynamic array Nalle()
ReDim Nalle(Nloc)

For i = 1 To Nloc
   Input #1, Nalle(i)
Next i

'get the largest number of alleles among all loci
MaxNalle = GetMaxNalle(Nalle, Nloc)

'declare dynamic array AlleFreq()
ReDim AlleFreq(Npop, Nloc, MaxNalle)

'read allele frequency from input dada file
For i = 1 To Nloc
   For j = 1 To Nalle(i)
      For k = 1 To Npop
        Input #1, AlleFreq(k, i, j)
      Next k
   Next j
Next i

Close #1

Do While outputFileName = ""
   intpress = MsgBox("You need to select an output data file", vbOKOnly, "Select output data file")
   Call mnuFileOut_Click
Loop

Open outputFileName For Output As #2
Print #2, "Genetic parameters are calculated based on follwing fomulas:"
If diversityOption = 1 Then
   Print #2, "Hs & Ht - unmodified(Nei 1973)"
Else
   Print #2, "Hs & Ht - modified for sample size(Nei & Chesser 1983)"
End If
Select Case differenOption:
   Case 1: Print #2, "Gst - unmodified(Nei, 1973)"
   Case 2: Print #2, "Gst - modified for sample size(Nei & Chesser 1983)"
   Case 3: Print #2, "Gst - modified for sample size and population number(Nei 1986)"
   Case 4: Print #2, "Gst - is actually Theta(Weir & Cockerham 1984)"
End Select
Print #2, "Fst - calculated only for simulated dominant biallelic data sets(Lynch & "
Print #2, Tab(7); "Milligan 1994)"
Print #2,
Print #2, "Last letter denotes for genetic parameters (Hs, Ht, Gst):"
Print #2, "c - based on corrected allele frequencies from Lynch & Milligan(1994)"
Print #2, "d - based on simulated codominant multiallelic data sets"
Print #2, "e - based on generated base population data sets"
Print #2, "i - based on simulated dominant biallelic data sets"
Print #2,
End Sub

Function GetMaxNalle(A() As Integer, B As Integer) As Integer
Dim i As Integer
Dim C As Integer
C = 0
For i = 1 To B
   If C < A(i) Then
      C = A(i)
```

```
    End If
  Next i
  GetMaxNalle = C
  End Function
  Private Sub theoretical_value()
  'Hs:within-population genetic diversity; _
   Ht:total genetic diversity; Gst:population differentiation
  Dim Hs As Double, Ht As Double, Gst As Double
  Dim outString As Variant

  Call calc_theor_diversity(AlleFreq, Nalle, Hs, Ht)

  If differenOption <= 2 Then
    Gst = 1 - Hs / Ht
  Else
    If differenOption = 3 Then
      Gst = Npop * (Ht - Hs) / (Npop * Ht - Hs)
    Else
      Gst = calculateTheta(AlleFreq, Nalle, basePopSize)
    End If
  End If
  Print #2, "Generated base population size: "; basePopSize
  Print #2, "Theoretical values based on initial allele frequencies from input data file"
  Print #2, Spc(1); "Hs"; Tab(11); "Ht"; Tab(19); "Gst"
  Print #2, Format(Hs, ".0000"); Tab(10); Format(Ht, ".0000"); Tab(18); Format(Gst, ".0000")
  Print #2,
  End Sub
  'Calculate theoretical genetic diversity
  Private Sub calc_theor_diversity(freq() As Double, nAllele() As Integer, _
                        Hs As Double, Ht As Double)
  Dim i As Integer
  Dim j As Integer
  Dim k As Integer
  Dim Hs1 As Double, sum1 As Double
  Dim Ht1 As Double, sum2 As Double
  Dim Pk1 As Double, Pk2 As Double

  Hs = 0
  Ht = 0
  For i = 1 To Nloc
    sum1 = 0
    sum2 = 0
    For j = 1 To nAllele(i)
      Pk1 = 0
      Pk2 = 0
      For k = 1 To Npop
        Pk1 = Pk1 + freq(k, i, j) * freq(k, i, j)
        Pk2 = Pk2 + freq(k, i, j)
      Next k
      Pk1 = Pk1 / Npop
      Pk2 = Pk2 / Npop
      sum1 = sum1 + Pk1
      sum2 = sum2 + Pk2 * Pk2
    Next j
    Hs1 = 1 - sum1
    Ht1 = 1 - sum2
    Hs = Hs + Hs1
    Ht = Ht + Ht1
  Next i
  Hs = Hs / Nloc
  Ht = Ht / Nloc
```

```
End Sub
'Calculate Theta value
Function calculateTheta(A() As Double, B() As Integer, nSize As Integer) As Double
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim m As Integer
'average allele frequency"
ReDim qa(Nloc, MaxNalle)
Dim sum1 As Double, sum2 As Double
Dim s As Double, H As Double
Dim h_s As Double, a_li As Double, b_li As Double, c_li As Double

For i = 1 To Nloc
  For j = 1 To B(i)
    qa(i, j) = 0
    For k = 1 To Npop
      qa(i, j) = qa(i, j) + A(k, i, j)
    Next k
    qa(i, j) = qa(i, j) / Npop
  Next j
Next i

sum1 = 0
sum2 = 0
For i = 1 To Nloc
  For j = 1 To B(i)
    s = 0
    H = 0
    For k = 1 To Npop
      s = s + (A(k, i, j) - qa(i, j)) ^ 2
      h_s = 0
      For m = 1 To B(i)
        If m <> j Then h_s = h_s + A(k, i, m)
      Next m
      H = H + A(k, i, j) * h_s
    Next k
    s = s / (Npop - 1)
    H = 2 * H / Npop
    If Nalle(i) > 2 Then H = H / (Nalle(i) - 1)
    c_li = 0.5 * H
    a_li = s - 1 / (nSize - 1) * (qa(i, j) * (1 - qa(i, j)) _
        - (Npop - 1) * s / Npop - 0.25 * H)
    b_li = nSize / (nSize - 1) * (qa(i, j) * (1 - qa(i, j)) _
        - (Npop - 1) * s / Npop - 0.25 * (2 * nSize - 1) / nSize * H)
    sum1 = sum1 + a_li
    sum2 = sum2 + a_li + b_li + c_li
  Next j
Next i
calculateTheta = sum1 / sum2

End Function

Private Sub popGenerate()
'for each population, generate a number of individuals(basePopSize)each with _
multi-locus genotypes that maintain the original _
allele frequencies within populations

ReDim allele1(Npop, basePopSize, Nloc)
ReDim allele2(Npop, basePopSize, Nloc)
Dim i As Integer
Dim j As Integer
```

```
Dim k As Integer, n As Integer
Dim sum As Double, ranValue As Double

'initialize the random-number generator with a seed based on _
the system timer (without argument)
Randomize

For i = 1 To Npop
   For j = 1 To basePopSize
      For k = 1 To Nloc
         'generation of first allele in each locus _
          for all individuals
         ranValue = Rnd
         'rnd returns a random value between 0 and 1
         sum = 0
         For n = 1 To Nalle(k)
            sum = sum + AlleFreq(i, k, n)
            If ranValue < sum Then GoTo LabelI
         Next n
LabelI:    allele1(i, j, k) = n 'get the first allele number

         'generation of second allele in each locus _
         for all individuals
         ranValue = Rnd
         sum = 0
         For n = 1 To Nalle(k)
            sum = sum + AlleFreq(i, k, n)
            If ranValue < sum Then GoTo labelII
         Next n
labelII:   allele2(i, j, k) = n 'get the second allele number
      Next k
   Next j
Next i


End Sub
Private Sub empirical_value()

Dim Hse As Double, Hte As Double, Gste As Double

'Hs1:diversity unmodified ; Hs2:diveristy unbiased for sample size
'Ht1:diversity unmodified ; Ht2:diveristy unbiased for sample size
Dim Hs1 As Double, Hs2 As Double, Ht1 As Double, Ht2 As Double

ReDim baseAlleFreq(Npop, Nloc, MaxNalle)
Call getFrequency(allele1, basePopSize, allele2, baseAlleFreq)
Call calc_empiri_diversity(baseAlleFreq, Nalle, basePopSize, Hs1, Hs2, Ht1, Ht2)
If diversityOption = 1 Then
   Hse = Hs1
   Hte = Ht1
Else
   Hse = Hs2
   Hte = Ht2
End If

Select Case differenOption
   Case 1: Gste = 1 - Hs1 / Ht1
   Case 2: Gste = 1 - Hs2 / Ht2
   Case 3: Gste = Npop * (Ht2 - Hs2) / (Npop * Ht2 - Hs2)
   Case 4: Gste = calculateTheta(baseAlleFreq, Nalle, basePopSize)
End Select
```

```
Print #2, "Empirical values based on allele frequencies from generated populations of 1000 indivudals"
Print #2, Spc(1); "Hse"; Tab(11); "Hte"; Tab(19); "Gste"
Print #2, Format(Hse, ".0000"); Tab(10); Format(Hte, ".0000"); Tab(18); Format(Gste, ".0000")
Print #2,
End Sub

Private Sub calc_empiri_diversity(freq() As Double, nAllele() As Integer, nPopSize As Integer, _
        Hse1 As Double, Hse2 As Double, Hte1 As Double, Hte2 As Double)

Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim Hs1 As Double, Hs2 As Double, Pk1 As Double, sum1 As Double
Dim Ht1 As Double, Ht2 As Double, Pk2 As Double, sum2 As Double
Hse1 = 0
Hse2 = 0
Hte1 = 0
Hte2 = 0
For i = 1 To Nloc
  sum1 = 0
  sum2 = 0
  For j = 1 To nAllele(i)
    Pk1 = 0
    Pk2 = 0
    For k = 1 To Npop
      Pk1 = Pk1 + freq(k, i, j) * freq(k, i, j)
      Pk2 = Pk2 + freq(k, i, j)
    Next k
    Pk1 = Pk1 / Npop
    Pk2 = Pk2 / Npop
    sum1 = sum1 + Pk1
    sum2 = sum2 + Pk2 * Pk2
  Next j
  Hs1 = 1 - sum1
  Ht1 = 1 - sum2
  Hs2 = (1 - sum1) * 2 * nPopSize / (2 * nPopSize - 1)
  Ht2 = 1 - sum2 + Hs2 / (2 * nPopSize * Npop)
  Hse1 = Hse1 + Hs1
  Hse2 = Hse2 + Hs2
  Hte1 = Hte1 + Ht1
  Hte2 = Hte2 + Ht2
Next i
Hse1 = Hse1 / Nloc
Hse2 = Hse2 / Nloc
Hte1 = Hte1 / Nloc
Hte2 = Hte2 / Nloc

End Sub
Private Sub getFrequency(A() As Integer, m As Integer, B() As _
Integer, C() As Double)
'Calculate allele frequencies based on generated individuals(basePopSize), _
they should be very similar to those from original data input file _
c() return values to calling procedure

Dim i As Integer
Dim j As Integer
Dim k As Integer, n As Integer

For i = 1 To Npop
  For j = 1 To Nloc
    For k = 1 To Nalle(j)
      C(i, j, k) = 0
```

```
        For n = 1 To m
          If A(i, n, j) = k Then C(i, j, k) = _
            C(i, j, k) + 1
          If B(i, n, j) = k Then C(i, j, k) = _
            C(i, j, k) + 1
        Next n
        C(i, j, k) = C(i, j, k) / m / 2
      Next k
    Next j
Next i

End Sub
'Sample subpopulations of size subPopSize with replacement from generated basic population
Private Sub sampling()

Dim i As Integer, j As Integer
Dim k As Integer, n As Integer, s As Integer
Dim ranValue As Double

ReDim samAllele1(Npop, basePopSize, Nloc), samAllele2(Npop, basePopSize, Nloc)

'initialize the random-number generator
'Randomize ranInt2
Randomize

'Parameters calculated based on codominant allele frequencies
Dim Hsd() As Double, Htd() As Double, Gstd() As Double
ReDim Hsd(Nsam), Htd(Nsam), Gstd(Nsam)

'Parameters calculated based on dominant and null allele frequencies
Dim Hsi() As Double, Hti() As Double, Gsti() As Double
ReDim Hsi(Nsam), Hti(Nsam), Gsti(Nsam)

'Parameters calculated based on corrected allele frequencies _
based on Lynch & Milligan's formula
Dim Hsc() As Double, Htc() As Double, Gstc() As Double
ReDim Hsc(Nsam), Htc(Nsam), Gstc(Nsam)

'Fst value based on Lynch & Milligan's formula
Dim Fst() As Double
ReDim Fst(Nsam)

'Avarage value of those parameters
Dim AvgHsd As Double, AvgHtd As Double, AvgGstd As Double
Dim AvgHsi As Double, AvgHti As Double, AvgGsti As Double
Dim AvgHsc As Double, AvgHtc As Double, AvgGstc As Double
Dim AvgFst As Double

'Standard Deviation of those parameters
Dim StdHsd As Double, StdHtd As Double, StdGstd As Double
Dim StdHsi As Double, StdHti As Double, StdGsti As Double
Dim StdHsc As Double, StdHtc As Double, StdGstc As Double
Dim StdFst As Double

'the number of sampling
For i = 1 To Nsam
  For j = 1 To Npop
    For k = 1 To subPopSize
      ranValue = Rnd

      'n will be between 1 and basePopSize
      n = Int(ranValue * basePopSize) + 1
```

```
        For s = 1 To Nloc
          samAllele1(j, k, s) = allele1(j, n, s)
          samAllele2(j, k, s) = allele2(j, n, s)
        Next s
      Next k
    Next j
    Call calculate_sampling(i, Hsd, Htd, Gstd, Hsi, Hti, Gsti, Hsc, _
                    Htc, Gstc, Fst)
Next i

'Get parameters' average values over number of samplings
AvgHsd = average(Hsd, Nsam)
AvgHtd = average(Htd, Nsam)
AvgGstd = average(Gstd, Nsam)
AvgHsi = average(Hsi, Nsam)
AvgHti = average(Hti, Nsam)
AvgGsti = average(Gsti, Nsam)
AvgHsc = average(Hsc, Nsam)
AvgHtc = average(Htc, Nsam)
AvgGstc = average(Gstc, Nsam)
AvgFst = average(Fst, Nsam)

StdHsd = Stdev(Hsd, Nsam)
StdHtd = Stdev(Htd, Nsam)
StdGstd = Stdev(Gstd, Nsam)
StdHsi = Stdev(Hsi, Nsam)
StdHti = Stdev(Hti, Nsam)
StdGsti = Stdev(Gsti, Nsam)
StdHsc = Stdev(Hsc, Nsam)
StdHtc = Stdev(Htc, Nsam)
StdGstc = Stdev(Gstc, Nsam)
StdFst = Stdev(Fst, Nsam)

Print #2, "Number of sampling: "; Nsam; " cycles"
Print #2, "Sample size for each population: "; subPopSize
Print #2, Tab(2); "Hsd"; Tab(9); "Htd"; Tab(16); "Gstd"; Tab(23); "Hsi"; _
      Tab(30); "Hti"; Tab(37); "Gsti"; Tab(44); "Hsc"; Tab(51); "Htc" _
      ; Tab(58); "Gstc"; Tab(65); "Fst"
For i = 1 To Nsam
  Print #2, Format(Hsd(i), ".0000"); Tab(8); Format(Htd(i), ".0000"); _
        Tab(15); Format(Gstd(i), ".0000"); Tab(22); Format(Hsi(i), ".0000"); _
        Tab(29); Format(Hti(i), ".0000"); Tab(36); Format(Gsti(i), ".0000"); _
        Tab(43); Format(Hsc(i), ".0000"); Tab(50); Format(Htc(i), ".0000"); _
        Tab(57); Format(Gstc(i), ".0000"); Tab(64); Format(Fst(i), ".0000")
Next i
For i = 1 To 70
  Print #2, "-";
Next i
Print #2, "-"
Print #2, "Average Values:"
Print #2, Format(AvgHsd, ".0000"); Tab(8); Format(AvgHtd, ".0000"); _
      Tab(15); Format(AvgGstd, ".0000"); Tab(22); Format(AvgHsi, ".0000"); _
      Tab(29); Format(AvgHti, ".0000"); Tab(36); Format(AvgGsti, ".0000"); _
      Tab(43); Format(AvgHsc, ".0000"); Tab(50); Format(AvgHtc, ".0000"); _
      Tab(57); Format(AvgGstc, ".0000"); Tab(64); Format(AvgFst, ".0000")
Print #2, "Standard deviation:"
Print #2, Format(StdHsd, ".0000"); Tab(8); Format(StdHtd, ".0000"); _
      Tab(15); Format(StdGstd, ".0000"); Tab(22); Format(StdHsi, ".0000"); _
      Tab(29); Format(StdHti, ".0000"); Tab(36); Format(StdGsti, ".0000"); _
      Tab(43); Format(StdHsc, ".0000"); Tab(50); Format(StdHtc, ".0000"); _
      Tab(57); Format(StdGstc, ".0000"); Tab(64); Format(StdFst, ".0000")
```

```
For i = 1 To 70
   Print #2, "-";
Next i
Print #2, "-"
Print #2,

End Sub
'After sampling, calculating Hs, Ht, and Gst values directly and indirectly
Private Sub calculate_sampling(A As Integer, Hsd() As Double, Htd() As Double, _
      Gstd() As Double, Hsi() As Double, Hti() As Double, Gsti() As Double, _
      Hscor() As Double, Htcor() As Double, Gstcor() As Double, Fst() As Double)

Dim i As Integer
Dim samAlleFreq() As Double
ReDim samAlleFreq(Npop, Nloc, MaxNalle)

'declare allele frequencies for dominant and null alleles
Dim indir_freq() As Double
ReDim indir_freq(Npop, Nloc, 2)

'declare corrected allele frequencies for dominant and null alleles _
 based on Lynch and Milligan's(1994) equation 2a
Dim cor_freq() As Double
ReDim cor_freq(Npop, Nloc, 2)

'number of alleles in each locus for simulated data set
Dim simu_nalle() As Integer
ReDim simu_nalle(Nloc)

Dim Hs1 As Double, Hs2 As Double, Ht1 As Double, Ht2 As Double

'For simulated data, each locus has two alleles (one dominant, one null)
For i = 1 To Nloc
   simu_nalle(i) = 2
Next i

Call getFrequency(samAllele1, subPopSize, samAllele2, samAlleFreq)
Call calc_empiri_diversity(samAlleFreq, Nalle, subPopSize, Hs1, Hs2, Ht1, Ht2)

If diversityOption = 1 Then
   Hsd(A) = Hs1
   Htd(A) = Ht1
Else
   Hsd(A) = Hs2
   Htd(A) = Ht2
End If

Select Case differenOption
   Case 1: Gstd(A) = 1 - Hs1 / Ht1
   Case 2: Gstd(A) = 1 - Hs2 / Ht2
   Case 3: Gstd(A) = Npop * (Ht2 - Hs2) / (Npop * Ht2 - Hs2)
   Case 4: Gstd(A) = calculateTheta(samAlleFreq, Nalle, subPopSize)
End Select

Call get_indir_cor_freq(samAllele1, samAllele2, indir_freq, cor_freq)

Call calc_empiri_diversity(indir_freq, simu_nalle, subPopSize, Hs1, Hs2, Ht1, Ht2)
If diversityOption = 1 Then
   Hsi(A) = Hs1
   Hti(A) = Ht1
Else
   Hsi(A) = Hs2
```

```
    Hti(A) = Ht2
End If

Select Case differenOption
    Case 1: Gsti(A) = 1 - Hs1 / Ht1
    Case 2: Gsti(A) = 1 - Hs2 / Ht2
    Case 3: Gsti(A) = Npop * (Ht2 - Hs2) / (Npop * Ht2 - Hs2)
    Case 4: Gsti(A) = calculateTheta(indir_freq, simu_nalle, subPopSize)
End Select

Call calc_empiri_diversity(cor_freq, simu_nalle, subPopSize, Hs1, Hs2, Ht1, Ht2)
If diversityOption = 1 Then
    Hscor(A) = Hs1
    Htcor(A) = Ht1
Else
    Hscor(A) = Hs2
    Htcor(A) = Ht2
End If

Select Case differenOption
    Case 1: Gstcor(A) = 1 - Hs1 / Ht1
    Case 2: Gstcor(A) = 1 - Hs2 / Ht2
    Case 3: Gstcor(A) = Npop * (Ht2 - Hs2) / (Npop * Ht2 - Hs2)
    Case 4: Gstcor(A) = calculateTheta(cor_freq, simu_nalle, subPopSize)
End Select

Fst(A) = Lynch_Fst(indir_freq, cor_freq)
End Sub
'Resample subpopulations with bootstrap resampling
Private Sub Resampling()

Dim i As Integer, j As Integer
Dim k As Integer, n As Integer, s As Integer
Dim ranValue As Double

'Parameters calculated based on codominant allele frequencies
Dim Hsd() As Double, Htd() As Double, Gstd() As Double
ReDim Hsd(Nresam), Htd(Nresam), Gstd(Nresam)

'Parameters calculated based on dominant and null allele frequencies
Dim Hsi() As Double, Hti() As Double, Gsti() As Double
ReDim Hsi(Nresam), Hti(Nresam), Gsti(Nresam)

'Parameters calculated based on corrected allele frequencies _
based on Lynch & Milligan's formula
Dim Hsc() As Double, Htc() As Double, Gstc() As Double
ReDim Hsc(Nresam), Htc(Nresam), Gstc(Nresam)

'Fst value based on Lynch & Milligan's formula
Dim Fst() As Double
ReDim Fst(Nresam)


ReDim resamAllele1(Npop, basePopSize, Nloc), resamAllele2(Npop, basePopSize, Nloc)

'initialize the random-number generator
Randomize ranInt2

'Avarage value of those parameters
Dim AvgHsd As Double, AvgHtd As Double, AvgGstd As Double
Dim AvgHsi As Double, AvgHti As Double, AvgGsti As Double
Dim AvgHsc As Double, AvgHtc As Double, AvgGstc As Double
```

```
Dim AvgFst As Double

'Standard Deviation of those parameters
Dim StdHsd As Double, StdHtd As Double, StdGstd As Double
Dim StdHsi As Double, StdHti As Double, StdGsti As Double
Dim StdHsc As Double, StdHtc As Double, StdGstc As Double
Dim StdFst As Double

For i = 1 To Nresam
  For j = 1 To Npop
    For k = 1 To subPopSize
      ranValue = Rnd
      n = Int(ranValue * subPopSize) + 1
      For s = 1 To Nloc
        resamAllele1(j, k, s) = samAllele1(j, n, s)
        resamAllele2(j, k, s) = samAllele2(j, n, s)
      Next s
    Next k
  Next j
  Call calculate_resampling(i, Hsd, Htd, Gstd, Hsi, Hti, Gsti, Hsc, _
                  Htc, Gstc, Fst)
Next i

'Get parameters' average values over number of samplings
AvgHsd = average(Hsd, Nresam)
AvgHtd = average(Htd, Nresam)
AvgGstd = average(Gstd, Nresam)
AvgHsi = average(Hsi, Nresam)
AvgHti = average(Hti, Nresam)
AvgGsti = average(Gsti, Nresam)
AvgHsc = average(Hsc, Nresam)
AvgHtc = average(Htc, Nresam)
AvgGstc = average(Gstc, Nresam)
AvgFst = average(Fst, Nresam)

StdHsd = Stdev(Hsd, Nresam)
StdHtd = Stdev(Htd, Nresam)
StdGstd = Stdev(Gstd, Nresam)
StdHsi = Stdev(Hsi, Nresam)
StdHti = Stdev(Hti, Nresam)
StdGsti = Stdev(Gsti, Nresam)
StdHsc = Stdev(Hsc, Nresam)
StdHtc = Stdev(Htc, Nresam)
StdGstc = Stdev(Gstc, Nresam)
StdFst = Stdev(Fst, Nresam)

Print #2, "Number of bootstrap resampling: "; Nresam; " cycles"
Print #2, "Sample size for each population: "; subPopSize
Print #2, Tab(2); "Hsd"; Tab(9); "Htd"; Tab(16); "Gstd"; Tab(23); "Hsi"; _
      Tab(30); "Hti"; Tab(37); "Gsti"; Tab(44); "Hsc"; Tab(51); "Htc" _
      ; Tab(58); "Gstc"; Tab(65); "Fst"
For i = 1 To Nsam
  Print #2, Format(Hsd(i), ".0000"); Tab(8); Format(Htd(i), ".0000"); _
        Tab(15); Format(Gstd(i), ".0000"); Tab(22); Format(Hsi(i), ".0000"); _
        Tab(29); Format(Hti(i), ".0000"); Tab(36); Format(Gsti(i), ".0000"); _
        Tab(43); Format(Hsc(i), ".0000"); Tab(50); Format(Htc(i), ".0000"); _
        Tab(57); Format(Gstc(i), ".0000"); Tab(64); Format(Fst(i), ".0000")
Next i
For i = 1 To 70
  Print #2, "-";
Next i
Print #2, "-"
```

```
Print #2, "Average Values:"
Print #2, Format(AvgHsd, ".0000"); Tab(8); Format(AvgHtd, ".0000"); _
        Tab(15); Format(AvgGstd, ".0000"); Tab(22); Format(AvgHsi, ".0000"); _
        Tab(29); Format(AvgHti, ".0000"); Tab(36); Format(AvgGsti, ".0000"); _
        Tab(43); Format(AvgHsc, ".0000"); Tab(50); Format(AvgHtc, ".0000"); _
        Tab(57); Format(AvgGstc, ".0000"); Tab(64); Format(AvgFst, ".0000")
Print #2, "Standard deviation:"
Print #2, Format(StdHsd, ".0000"); Tab(8); Format(StdHtd, ".0000"); _
        Tab(15); Format(StdGstd, ".0000"); Tab(22); Format(StdHsi, ".0000"); _
        Tab(29); Format(StdHti, ".0000"); Tab(36); Format(StdGsti, ".0000"); _
        Tab(43); Format(StdHsc, ".0000"); Tab(50); Format(StdHtc, ".0000"); _
        Tab(57); Format(StdGstc, ".0000"); Tab(64); Format(StdFst, ".0000")
For i = 1 To 70
   Print #2, "-";
Next i
Print #2, "-"
End Sub
'After resampling, calculating Hs, Ht, and Gst values directly and indirectly
Private Sub calculate_resampling(A As Integer, Hsd() As Double, Htd() As Double, _
      Gstd() As Double, Hsi() As Double, Hti() As Double, Gsti() As Double, _
      Hscor() As Double, Htcor() As Double, Gstcor() As Double, Fst() As Double)

Dim resamAlleFreq() As Double
ReDim resamAlleFreq(Npop, Nloc, MaxNalle)
'declare allele frequencies for dominant and null alleles
Dim indir_freq() As Double
ReDim indir_freq(Npop, Nloc, 2)

'declare corrected allele frequencies for dominant and null alleles _
 based on Lynch and Milligan's(1994) equation 2a
Dim cor_freq() As Double
ReDim cor_freq(Npop, Nloc, 2)

'number of alleles in each locus for simulated data set
Dim simu_nalle() As Integer
ReDim simu_nalle(Nloc)

Dim Hs1 As Double, Hs2 As Double, Ht1 As Double, Ht2 As Double
Dim i As Integer

'For simulated data, each locus has two alleles (one dominant, one null)
For i = 1 To Nloc
   simu_nalle(i) = 2
Next i

Call getFrequency(resamAllele1, subPopSize, resamAllele2, resamAlleFreq)
Call calc_empiri_diversity(resamAlleFreq, Nalle, subPopSize, Hs1, Hs2, Ht1, Ht2)

If diversityOption = 1 Then
   Hsd(A) = Hs1
   Htd(A) = Ht1
Else
   Hsd(A) = Hs2
   Htd(A) = Ht2
End If

Select Case differenOption
   Case 1: Gstd(A) = 1 - Hs1 / Ht1
   Case 2: Gstd(A) = 1 - Hs2 / Ht2
   Case 3: Gstd(A) = Npop * (Ht2 - Hs2) / (Npop * Ht2 - Hs2)
   Case 4: Gstd(A) = calculateTheta(resamAlleFreq, Nalle, subPopSize)
End Select
```

```
Call get_indir_cor_freq(resamAllele1, resamAllele2, indir_freq, cor_freq)

Call calc_empiri_diversity(indir_freq, simu_nalle, subPopSize, Hs1, Hs2, Ht1, Ht2)
If diversityOption = 1 Then
   Hsi(A) = Hs1
   Hti(A) = Ht1
Else
   Hsi(A) = Hs2
   Hti(A) = Ht2
End If

Select Case differenOption
   Case 1: Gsti(A) = 1 - Hs1 / Ht1
   Case 2: Gsti(A) = 1 - Hs2 / Ht2
   Case 3: Gsti(A) = Npop * (Ht2 - Hs2) / (Npop * Ht2 - Hs2)
   Case 4: Gsti(A) = calculateTheta(indir_freq, simu_nalle, subPopSize)
End Select

Call calc_empiri_diversity(cor_freq, simu_nalle, subPopSize, Hs1, Hs2, Ht1, Ht2)
If diversityOption = 1 Then
   Hscor(A) = Hs1
   Htcor(A) = Ht1
Else
   Hscor(A) = Hs2
   Htcor(A) = Ht2
End If

Select Case differenOption
   Case 1: Gstcor(A) = 1 - Hs1 / Ht1
   Case 2: Gstcor(A) = 1 - Hs2 / Ht2
   Case 3: Gstcor(A) = Npop * (Ht2 - Hs2) / (Npop * Ht2 - Hs2)
   Case 4: Gstcor(A) = calculateTheta(cor_freq, simu_nalle, subPopSize)
End Select

Fst(A) = Lynch_Fst(indir_freq, cor_freq)
End Sub

Private Sub get_indir_cor_freq(A() As Integer, B() As Integer, _
                indir_freq() As Double, cor_freq() As Double)

'Calculate allele frequency either based on simulated dominant _
 biallelic data set by randomly selecting one allele as dominant, with the _
 rest treated as recessive to it, or based on Lynch & Milligan's formula

Dim i As Integer, j As Integer, k As Integer, s As Integer

'random allele in each locus
Dim ranAlle() As Integer
ReDim ranAlle(Nloc)
Dim count As Integer

'Pick up a random allele as dominant allele for each locus
Randomize
For i = 1 To Nloc
   ranAlle(i) = Int(Rnd * Nalle(i)) + 1
Next i

For i = 1 To Npop
   For j = 1 To Nloc
      count = 0
```

```
      For k = 1 To subPopSize
        If ((A(i, k, j) <> ranAlle(j)) And (B(i, k, j) <> ranAlle(j))) Then
        count = count + 1
        End If
      Next k
      indir_freq(i, j, 1) = Sqr(count / subPopSize)
      indir_freq(i, j, 2) = 1 - indir_freq(i, j, 1)
      cor_freq(i, j, 1) = 8 * subPopSize * (indir_freq(i, j, 1) ^ 3) _
        / (indir_freq(i, j, 1) ^ 2 * (8 * subPopSize + 1) - 1)
      cor_freq(i, j, 2) = 1 - cor_freq(i, j, 1)
    Next j
  Next i

  End Sub

  Function Lynch_Fst(indir_freq() As Double, cor_freq() As Double) As Double

  Dim i As Integer, j As Integer, k As Integer, l As Integer, s As Integer

  'The proportion of sampled individuals(subPopSize) with null homozygotes
  Dim X() As Double
  ReDim X(Npop, Nloc)

  'Gene diversity(Hj) within population(j)
  'Between-population diversity(H)
  'Mean witn-population divesity(Hwithin)
  'Mean between-population gene diversity(Hbetween)
  'Total gene diversity(Htotal)
  'Variance of Hwithin(varHw)
  'Variance of Hbetween(varHb)
  Dim Hj() As Double, H() As Double
  Dim Hwithin As Double, Hbetween As Double, Htotal As Double
  ReDim Hj(Npop), H(Npop, Npop)

  'Vb represents the variance among diversity measures involving _
  nonoverlapping population pairs (e.g. H12, H34, .....)
  'Cb represents the covariance among diversity measures involving _
  overlapping population pairs (e.g. [H12, H23], [H12, H14], .....)
  Dim Vb As Double, Cb As Double

  'Number of nonoverlapping and overlapping diveristy pairs
  Dim Nnonoverlap As Integer, Noverlap As Integer

  Dim sum As Double, sum0 As Double, sum1 As Double
  Dim sum2 As Double, sum3 As Double, sum4 As Double

  'Sampling covariance of the within- and between-population estimates _
  of gene diversity
  Dim varHw As Double, varHb As Double, covHbHw As Double

  'Get null homozygote frequency
  For j = 1 To Npop
    For k = 1 To Nloc
      X(j, k) = indir_freq(j, k, 1) ^ 2
    Next k
  Next j

  'Calculate heterozygosity between populations j and k
  For j = 1 To Npop
    For k = 1 To Npop
      H(j, k) = 0
      For l = 1 To Nloc
```

```
        H(j, k) = H(j, k) + (cor_freq(j, l, 1) - cor_freq(k, l, 1)) ^ 2
        If (X(k, l) > 0) Then H(j, k) = H(j, k) - (2 - X(j, l) - X(k, l)) / (4 * subPopSize)
      Next l
      H(j, k) = H(j, k) / Nloc
    Next k
Next j

'Calculate gene diversity for each population(Hj)
For j = 1 To Npop
  Hj(j) = 0
  For k = 1 To Nloc
    Hj(j) = Hj(j) + 2 * cor_freq(j, k, 1) * cor_freq(j, k, 2)
    If (X(j, k) > 0) Then Hj(j) = Hj(j) + (1 - X(j, k)) / (2 * subPopSize)
  Next k
  Hj(j) = Hj(j) / Nloc
Next j

'Calculate Hwithin
Hwithin = 0
For j = 1 To Npop
  Hwithin = Hwithin + Hj(j)
Next j
Hwithin = Hwithin / Npop

'Calculate Hbetween (averaged over all distinct pairs of populations)
Hbetween = 0
For k = 1 To Npop
  For j = k + 1 To Npop
    Hbetween = Hbetween + H(j, k)
  Next j
Next k
Hbetween = 2 * Hbetween / (Npop * (Npop - 1))

'Calculate Htotal
Htotal = Hbetween + Hwithin

'Calculate varHw
varHw = 0
For j = 1 To Npop
  varHw = varHw + (Hj(j) - Hwithin) ^ 2
Next j
varHw = varHw / Npop / (Npop - 1)

'Calculate varHb
varHb = 0
'when more than three populations are used, nonoverlapping population _
pairs exist (e.g. H12, H34). Vb is calculated.
If Npop >= 4 Then
  sum1 = 0
  sum0 = 0
  If (Npop / 2 * 2 = Npop) Then
    For j = 1 To (Npop - 1) Step 2
      sum1 = sum1 + H(j, j + 1) ^ 2
      sum0 = sum0 + H(j, j + 1)
    Next
    Nnonoverlap = Npop / 2
  Else
    For j = 1 To (Npop - 2) Step 2
      sum1 = sum1 + H(j, j + 1) ^ 2
      sum0 = sum0 + H(j, j + 1)
    Next j
    sum2 = 0
```

```
      For j = 2 To (Npop - 1) Step 2
         sum2 = sum2 + H(j, j + 1) ^ 2
      Next j
      sum1 = (sum1 + sum2) / 2
      Nnonoverlap = Npop / 2
   End If
   Vb = (1 / Nnonoverlap) * (sum1 - sum0 ^ 2 / Nnonoverlap)

   'Calculate Cb
   sum2 = 0
   sum3 = 0
   sum4 = 0
   Noverlap = 0

   For j = 1 To Npop - 2
      For k = j + 1 To Npop - 1
         sum3 = sum3 + H(j, k)
         For l = k + 1 To Npop
            sum2 = sum2 + H(j, k) * H(j, l)
            Noverlap = Noverlap + 1
         Next l
      Next k
   Next j

   For j = 1 To Npop - 2
      For l = j + 2 To Npop
         sum4 = sum4 + H(j, l)
      Next l
   Next j
   Cb = 1 / Noverlap * (sum2 - sum3 * sum4 / Noverlap)
   varHb = 2 / (Npop * (Npop - 1)) * (Vb + 2 * (Npop - 2) * Cb)
End If

'calculate covHbHw
covHbHw = 0
If Npop >= 3 Then
   sum = 0
   For j = 1 To Npop
      sum1 = 0
      For k = 1 To Npop
      If k <> j Then
         sum1 = sum + H(j, k)
      End If
      Next k
      sum = sum + Hj(j) * sum1
   Next j
   covHbHw = 1 / Npop * (1 / (Npop * (Npop - 1)) * sum - Hwithin * Hbetween)
End If
Lynch_Fst = Hbetween / Htotal * (1 + (Hbetween * varHw - Hwithin * varHb _
   + (Hbetween - Hwithin) * covHbHw) / Hbetween / Htotal ^ 2) ^ (-1)

End Function


Function average(A() As Double, n As Integer) As Double
Dim k As Integer
Dim sum As Double

sum = 0
For k = 1 To n
   sum = sum + A(k)
Next k
```

```
average = sum / n

End Function

Function Stdev(A() As Double, n As Integer) As Double
Dim k As Integer
Dim sum As Double, avg As Double

sum = 0
avg = average(A, n)
For k = 1 To n
   sum = sum + (A(k) - avg) ^ 2
Next k
sum = sum / (n - 1)
Stdev = Sqr(sum)

End Function
```

6.2. ASP web pages

(1) Include files:

//ProductionErrorHandler.inc:

```
<script language=vbscript runat=server>
Function CheckForErrors(objConnection)
    'Declare variables
    Dim blnDisplayErrMsg

    If objConnection.Errors.Count > 0 Then

        'Create the FileSystemObject and open the error log
        Set objFile = Server.CreateObject("Scripting.FileSystemObject")
        Set objLog = objFile.OpenTextFile( _
            Server.MapPath("ProductionErrorLog.txt"),8,True)

        'Check for an open error from VBScript
        If Err.Number > 0 Then
            Response.Write "Error opening log file<P>"
            Response.Write "Error Number: " & Err.Number & _
                ", Error Description: " & Err.Description
        End If

        'Create an error object to access the ADO errors collection
        Set objErr = Server.CreateObject("ADODB.Error")

        'Log all errors to the error log
        For Each objErr In objConnection.Errors
            If objErr.Number = 0 Then
                blnDisplayErrMsg = False
            Else
                objLog.WriteLine(objErr.Number & "|" & _
                    objErr.Description & "|" & objErr.Source & "|" & _
                    objErr.SQLState & "|" & objErr.NativeError)
                blnDisplayErrMsg = True
            End If
        Next

        'Close the log file and dereference all objects
        objLog.Close
        Set objLog = Nothing
        Set objFile = Nothing
        Set objErr = Nothing

        If blnDisplayErrMsg Then
            'Display a graceful message to the user
            Response.Write "An unforseen error has occurred and processing " & _
                "must be stopped. You can try your request again later or " & _
                "you can call our Help Desk at 888-888-1234"
            'Halt Execution
            Response.End
        End If
    End If
End Function
</script>
```

//AuthenticationCheck.inc:

```
<%
'Authentication check
If Session("Authenticated") <> True Then
        Session("ErrorMessage") = "You Have not properly logged in."
        Response.Redirect "Default.asp"
End If
%>
```

//Connect.inc:

```
<%
'Instruct VBScript to ignore the error and continue
'with the next line of code
On Error Resume Next

'Create and open the database object
Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open Application("ConnectString")
%>
```

//CommonFunctions.inc:

```
<script language=vbscript runat=server>
Function ConvertString(strInput)
        Dim intPos
  intPos = 1
  Do
    intPos = InStr(intPos, strInput, "'", vbTextCompare)
    If intPos > 0 Then
      strInput = Left(strInput, intPos) + Right(strInput, Len(strInput) - (intPos - 1))
      intPos = intPos + 2
    End If
  Loop While intPos > 0
  ConvertString = strInput
End Function
</script>
```

//Disconnect.inc:

```
<%
'Close and dereference database objects
If IsObject(objRS) Then
        objRS.Close
        Set objRS = Nothing
End IF
objConn.Close
Set objConn = Nothing
%>
```

//MenuOptions.inc:

```
<table>
        <tr>
        <td height=50><font color=navy>Please select an following option</font></td>
        </tr>
        <tr>
        <td><a href="Download.html"
                onmouseover="window.status='Download this simulation program'"
```

```
            onmouseout="window.status="">
            Download this simulation program</font></td>
        </tr>
        <tr>
        <td><a href="RegisterNewPaper.asp"
            onmouseover="window.status='Register a new paper which used the simulation program '"
            onmouseout="window.status="">
            Register a new paper which used the simulation program</font></td>
        </tr>
        <tr>
        <td><a href="DisplayPapers.asp"
            onmouseover="window.status='Display related papers '"
            onmouseout="window.status="">
            Display related papers</font></td>
        </tr>
</table>
```

(2) Login and registration:

//Default.asp:

```
<%
'Check to see if a cookie exists for this user

If Len(Request.Cookies("Simulation")("UserName")) > 0 Then
        'Cookie exists

        'Authenticate the user for other web pages
        Session("Authenticated") = True

        'Redirect the browser to the welcome back page
        Response.Redirect "WelcomeBack.asp"
End If
%>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
        <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>
<br><br>

<%
'Check for an error message which indicates the previous
'login attempt failed
If Len(Session("ErrorMessage")) > 0 Then
        Response.Write "<font color=red>" & _
                Session("ErrorMessage") & "</font><br>"
End If
%>

<!--Display login form-->
<form action=loginverification.asp method=post name=frmDefault>
<table>
        <tr>
        <td colspan=2><font size="4" color=teal>Enter login information and click submit.</td>
```

```
        </tr>
        <tr>
        <td>  </td>
        </tr>
        <tr>
        <td>User Name</td>
        <td><input type=text name=txtUserName size=15></td>
        </tr>
        <tr>
        <td>Password</td>
        <td><input type=text name=txtPassword size=15></td>
        </tr>
        <tr>
        <td>  </td>
        </tr>
        <tr>
        <td><input type=submit name=btnSubmit value=Submit></td>
        </tr>
</table>
</form>

<br>
If you don't have a login ID and password you can
<a href="register.asp"
        onmouseover="window.status='Register on the Genetic Dominance Simulation Web Site'"
        onmouseout="window.status=''">register here.</a>
</BODY>
</HTML>
```

//WelcomeBack.asp:

```
<!-- #include file="AuthenticationCheck.inc" -->
<!-- #include file="adovbs.inc" -->
<!-- #include file="ProductionErrorHandler.inc" -->
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
        <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>

<!-- #include file="Connect.inc" -->

<br><br>
        <font color=teal>Welcome Back
<%
        'check for database errors
        Dim strUserName
        strUserName=Request.Cookies("Simulation")("UserName")
        call CheckForErrors(objConn)

        set objCmd=Server.CreateObject("ADODB.Command")
        set objCmd.ActiveConnection=objConn
        objCmd.CommandText="{CALL qparmUserName ('" & cStr(strUserName) & "')}"
        set objRS=Server.CreateObject("ADODB.Recordset")
        set objRS=objCmd.Execute
```

```
        call CheckForErrors(objConn)
%>
        <%=objRS("FirstName")%> 
        <%=objRS("LastName")%></font>
<br><br>

<!-- #include file="DisConnect.inc" -->
<!-- #include file="MenuOptions.inc" -->

</BODY>
</HTML>
```

//Register.asp:

```
<!-- #include file="adovbs.inc" -->
<!-- #include file="ProductionErrorHandler.inc" -->
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
        <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>
<br><br>
<%
'************************************************************
'Step1: Display the registration form for user input
'************************************************************
If len(Request.Form("FormAction")) = 0 then
%>
<form action=register.asp method=post name=FrmRegister1>
<Input type=hidden Name=FormAction value=step2>
<table>
        <tr >
        <td height=50 colspan=2><font size="4" color=teal>Please Enter Registration Information
                </font></td>
        </tr>
        <tr>
        <td> </td>
        </tr>
        <tr >
        <td>First Name</td>
        <td><input type=text name=txtFirstName size=15></
        td>
        <td width=50></td>
        <td>Last Name</td>
        <td><input type=text name=txtLastName size=15></td>
        </tr>
        <tr>
        <td>Organization</td>
        <td><input type=text name=txtOrganization size=30></td>
        </tr>
        <tr>
        <td>Country</td>
        <td><input type=text name=txtCountry size=30></td>
        </tr>
        <tr>
```

```
        <td>Professional Class</td>
        <td colspan=4>
            <input type=radio name=optClass value=1 checked>Professional
            <input type=radio name=optClass value=2>Student
            <input type=radio name=optClass value=3>Other
        </td>
        </tr>
        <tr>
        <td>  </td>
        </tr>
        <tr>
        <td><input type=button name=btnContinue value=Continue></td>
        </tr>
</table>
</form>

<Script Language=VbScript>
Sub btnContinue_OnClick()
        'Verify all fields have been entered
        If Len(frmRegister1.txtFirstName.value) = 0 Then
            Alert "You must enter a first name"
            frmRegister1.txtFirstName.focus
            Exit Sub
        ElseIf Len(frmRegister1.txtLastName.value) = 0 Then
            Alert "You must enter a last name"
            frmRegister1.txtLastName.focus
            Exit Sub
        ElseIf Len(frmRegister1.txtOrganization.value) = 0 Then
            Alert "You must enter an organization name"
            frmRegister1.txtOrganization.focus
            Exit Sub
        ElseIf Len(frmRegister1.txtCountry.value) = 0 Then
            Alert "You must enter a country name"
            frmRegister1.txtCountry.focus
            Exit Sub
        End If

        'If we get to this point all is OK, submit the form
        Call frmRegister1.submit()
End Sub
</script>
<%
'*********************************************************************
'Step2: Continue to enter user information: UserName and Password
'*********************************************************************
ElseIf Request.Form("FormAction") = "step2" then
%>

<!-- #include file="Connect.inc" -->

<%
'Get the registration information entered in step1
Dim txtFirstName, txtLastName, txtOrganization, txtCountry, optClass
txtFirstName = Request.Form("txtFirstName")
txtLastName = Request.Form("txtLastName")
txtOrganization = Request.Form("txtOrganization")
txtCountry = Request.Form("txtCountry")
optClass = Request.Form("optClass")

'Check for database errors
Call CheckForErrors(objConn)
```

```
'Create the recordset object, set the SQL string and open the recordset
Set objRS = server.CreateObject("ADODB.recordset")

strSQL = "qAllUserName"
objRS.Open strSQL, objConn, adOpenForwardOnly, , adCmdStoredProc

'Check for database errors
Call CheckForErrors(objConn)

%>
<form action=registrationConfirmation.asp method=post name=FrmRegister2>
<Input type=hidden Name=txtFirstName value='<%=txtFirstName%>'>
<Input type=hidden Name=txtLastName value='<%=txtLastName%>'>
<Input type=hidden Name=txtOrganization value='<%=txtOrganization%>'>
<Input type=hidden Name=txtCountry value='<%=txtCountry%>'>
<Input type=hidden Name=optClass value=<%=optClass%>>
<table align=center>
      <tr >
      <td height=50 colspan=2><font size="4" color=teal>Please Select Your UserName and Password
            </font></td>
      </tr>
      <tr>
      <td> </td>
      </tr>
      <tr>
      <td >User Name</td>
      <td><input type=text name=txtUserName size=15></
      td>
      </tr>
      <tr >
      <td >Password</td>
      <td><input type=password name=txtPassword size=15></td>
      </tr>
      <tr>
      <td>Retype Password</td>
      <td><input type=password name=txtRetypePassword size=15></td>
      </tr>
      <tr>
      <td>  </td>
      </tr>
      <tr>
      <td><input type=button name=btnSubmit value=Submit></td>
      </tr>
</table>
</form>

<Script Language=VbScript>
Sub btnSubmit_OnClick()
<%
      do while Not objRS.EOF
%>
      'Verify if other users have used this user name
      If frmRegister2.txtUserName.value="<%=objRS("UserName")%>" Then
            Alert "This user name has been taken"
            frmRegister2.txtUserName.focus
            Exit Sub
   end if
<%
      objRS.MoveNext
      loop
%>
   'Verify all fields that have been entered
```

```
        If Len(frmRegister2.txtUserName.value) = 0 Then
            Alert "You must enter a user name"
            frmRegister2.txtUserName.focus
            Exit Sub
        ElseIf Len(frmRegister2.txtPassword.value) = 0 Then
            Alert "You must enter a password"
            frmRegister2.txtPassword.focus
            Exit Sub
        ElseIf Len(frmRegister2.txtRetypePassword.value) = 0 Then
            Alert "You must retype your password"
            frmRegister2.txtRetypePassword.focus
            Exit Sub
        ElseIf frmRegister2.txtPassword.value <> frmRegister2.txtRetypePassword.value then
          Alert "Retyped password doesn't match"
          frmRegister2.txtPassword.value = ""
          frmRegister2.txtRetypePassword.value = ""
          frmRegister2.txtPassword.focus
            Exit Sub
        End If

        'If we get to this point all is OK, submit the form
        Call frmRegister2.submit()
End Sub
</script>

<%
end if
%>

</BODY>
</HTML>
```

//RegisterConfirm.asp:

```
<!-- #include file="adovbs.inc" -->
<!-- #include file="ProductionErrorHandler.inc" -->
<!-- #include file="Connect.inc" -->

<%
'Check for database errors

Call CheckForErrors(objConn)

'Set the parameters for the insert stored procedure
strSQL = "qInsertPerson ('" & CStr(Request.form("txtFirstName")) & _
        "','" & CStr(Request.form("txtLastName")) & _
        "','" & CStr(Request.form("txtOrganization")) & _
        "','" & CStr(Request.form("txtCountry")) & _
        "'," & CLng(Request.form("optClass")) & _
        ",'" & CStr(Request.Form("txtUserName")) & _
        "','" & CStr(Request.Form("txtPassword")) & "')"
'Execute the stored procedure to insert the person
objConn.Execute strSQL,,adCmdStoredProc

'Check for database errors
Call CheckForErrors(objConn)
%>

<!-- #include file="DisConnect.inc" -->

<%
'Save the user information to a cookie
```

```
Response.Cookies("Simulation")("UserName") = Request.Form("txtUserName")
Response.Cookies("Simulation")("Password") = Request.Form("txtPassword")

'Set the expiration date of the cookie to the last day of the current year
Response.Cookies("Simulation").Expires = "December 31, " & Year(Now)
'Response.Write Response.Cookies("Simulation")("UserName")

'Authenticate the user for other web pages
Session("Authenticated") = True
%>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
      <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>
<br><br>
      <font color=teal>Registration Successful
      </font>
<br><br>

<!-- #include file="MenuOptions.inc" -->

</BODY>
</HTML>
```

//LoginVerify.asp:

```
<!-- #include file="adovbs.inc" -->
<!-- #include file="ProductionErrorHandler.inc" -->
<!-- #include file="Connect.inc" -->

<%
'Check for database errors
Call CheckForErrors(objConn)

'Verify user information in the database

'Create the recordset object, set the SQL string and parameters
'and open the recordset
Set objRS = Server.CreateObject("ADODB.Recordset")
strSQL = "qparmVerifyLogin '" & CStr(Request.Form("txtUserName")) & _
      "','" & CStr(Request.Form("txtPassword")) & "'"
objRS.Open strSQL, objConn, adOpenForwardOnly, , adCmdStoredProc

'Check for database errors
Call CheckForErrors(objConn)

'Check for empty recordset which indicates user information
'was not found
If objRS.EOF or objRS.BOF Then
      Session("ErrorMessage") = "No record found - Please ensure all information was entered correctly"
      Response.Redirect "default.asp"
Else
      Session("ErrorMessage") = Empty
End If
```

```
%>

<!-- #include file="Disconnect.inc" -->

<%
'Save the user information to a cookie
Response.Cookies("Simulation")("UserName") = Request.Form("txtUserName")
Response.Cookies("Simulation")("Password") = Request.Form("txtPassword")

'Set the expiration date of the cookie to the last day of the
'current year
Response.Cookies("Simulation").Expires = "December 31, " & Year(Now)

'Authenticate the user for other web pages
Session("Authenticated") = True
%>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
      <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>
<br><br>

<!-- #include file="MenuOptions.inc" -->

</BODY>
</HTML>
```

(3) User options:

//Options.asp:

```
<!-- #include file="AuthenticationCheck.inc" -->
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
      <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>
<br><br>

<!-- #include file="MenuOptions.inc" -->

</BODY>
</HTML>
```

//RegisterNewPaper.asp:

```
<!-- #include file="adovbs.inc" -->
<!-- #include file="AuthenticationCheck.inc" -->
<!-- #include file="CommonFunctions.inc" -->
<!-- #include file="ProductionErrorHandler.inc" -->

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
      <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>
<br><br>

<%
'**********************************************************************
'* Step 1: Display the New paper form for user input
'**********************************************************************
If Len(Request.Form("FormAction")) = 0 Then
%>

      <form action=RegisterNewPaper.asp method=post name=frmNewPaper>
      <input type=hidden name=FormAction value=Step2>
      <table>
            <tr>
            <td height=50 colspan=2><font size="4" color=teal>New Paper Registration
                  </font></td>
            </tr>
            <tr>
            <td> </td>
            </tr>
            <tr>
            <td>Author Name</td>
            <td><input type=text name=txtAuthorName size=30></td>
            <td width=50></td>
            <td>Class</td>
            <td><select name=cboClass></select></td>
            </tr>
            <tr>
            <td>Published Year</td>
            <td><input type=text name=txtPublished size=8></td>
            <td width=50></td>
            <td>Volume</td>
            <td><input type=text name=txtVolume size=15></td>
            </tr>
            <tr>
            <td>Journal Name</td>
            <td><input type=text name=txtJournalName size=30></td>
            <td width=50></td>
            <td>Page NO.</td>
            <td><input type=text name=txtPageNo size=20></td>
            </tr>
            <tr>
            <td>Title</td>
            <td colspan=4><textarea name=txtTitle cols="60" wrap></textarea></td>
            </tr>
```

```
            <tr>
            <td>  </td>
            </tr>
            <tr>
            <td><input type=button name=btnSubmit value=Submit></td>
            </tr>
        </table>
        </form>

<!-- #include file="Connect.inc" -->

<%
'Check for database errors
Call CheckForErrors(objConn)

'Create the recordset object and open the recordset
Set objRS = Server.CreateObject("ADODB.Recordset")
strSQL = "qAllPaperClasses"
objRS.Open strSQL, objConn, adOpenForwardOnly, , adCmdStoredProc

'Check for database errors
Call CheckForErrors(objConn)
%>

<script language=vbscript>
Sub Window_OnLoad()
<%
    Do While Not objRS.EOF
%>
            Set objOption = document.createElement("OPTION")
            objOption.text = "<%=objRS("ClassName")%>"
            objOption.value = "<%=objRS("ClassName")%>"
            document.all.cboClass.add objOption
<%
            objRS.MoveNext
    Loop
%>
    <!-- #include file="Disconnect.inc" -->

    Set objOption = Nothing
End Sub

Sub btnSubmit_OnClick()
    'Verify required fields are complete
    If Len(frmNewPaper.txtAuthorName.value) = 0 Then
        Alert "You must enter an author name"
        frmNewPaper.txtAuthorName.focus
        Exit Sub
    ElseIf frmNewPaper.cboClass.selectedIndex = -1 Then
        Alert "You must select your paper class"
        frmNewPaper.cboClass.focus
        Exit Sub
    ElseIf Len(frmNewPaper.txtPublished.value) = 0 Then
        Alert "You must enter when your paper was published"
        frmNewPaper.txtPublished.focus
        Exit Sub
    ElseIf Len(frmNewPaper.txtVolume.value) = 0 Then
        Alert "You must enter the journal volume"
        frmNewPaper.txtVolume.focus
        Exit Sub
    ElseIf Len(frmNewPaper.txtJournalName.value) = 0 Then
        Alert "You must enter the journal name"
```

```
            frmNewPaper.txtJournalName.focus
        ElseIf Len(frmNewPaper.txtPageNo.value) = 0 Then
            Alert "You must enter the page number"
            frmNewPaper.txtPageNo.focus
        ElseIf Len(frmNewPaper.txttitle.value) = 0 Then
            Alert "You must enter the journal title"
            frmNewPaper.txtTitle.focus

        End If

        'If we get to this point all is OK, submit the form
        Call frmNewPaper.submit()
End Sub
</script>

<%
'*********************************************************************
'* Step 2: Process the new boat form the user has submitted
'*********************************************************************
ElseIf Request.Form("FormAction") = "Step2" Then
%>

        <!-- #include file="Connect.inc" -->

<%
        'Check for database errors
        Call CheckForErrors(objConn)

        'Run the author name through the string conversion routine
        'just in case there are any single quotes
        strTitle = ConvertString(Request.Form("txtTitle"))

        'Set the parameters for the insert stored procedure
        strSQL = "qInsertPaper ('" & CStr(Request.Form("txtAuthorName")) & _
                "','" & CStr(Request.Form("cboClass")) & _
                "'," & CLng(Request.Form("txtPublished")) & _
                ",'" & CStr(Request.Form("txtVolume")) & _
                "','" & CStr(Request.Form("txtJournalName")) & _
                "','" & CStr(Request.Form("txtPageNo")) & _
                "','" & CStr(strTitle) & "')"
        'Insert the new boat
        objConn.Execute strSQL,,adCmdStoredProc

        'Check for database errors
        Call CheckForErrors(objConn)
%>

        <!-- #include file="DisConnect.inc" -->


        <!--Display registration message-->
        <font color=teal>Your paper has been registered</font>
        <br><br>
        <a href="Options.asp"
                onmouseover="window.status='Return to Options Page'"
                onmouseout="window.status=''">Return to Options Page</a>
<%
End If
%>
</BODY>
</HTML>
```

//DisplayPapers.asp

```
<!-- #include file="adovbs.inc" -->
<!-- #include file="Connect.inc" -->
<!-- #include file="ProductionErrorHandler.inc" -->

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE>Genetic Dominance Simulation Web Site</TITLE>
</HEAD>
<BODY>

<!--Display the page data-->
<div align=center>
      <big><big><font color=navy>Genetic Simulation of Effects of Dominance</font></big></big>
</div>
<br><br>
<%
'*************************************************************
'Step1: Display the form for user to input paper parameters*
'*************************************************************
If len(Request.Form("FormAction")) = 0 then
%>
<form action=DisplayPapers.asp method=post name=FrmDisplay>
<Input type=hidden Name=FormAction value=step2>
<table align=center>
      <tr >
      <td height=50 colspan=2><font size="4" color=teal>Please Select Related Parameters for Displaying Papers
            </font></td>
      </tr>
      <tr>
      <td> </td>
      </tr>
      <tr>
      <td>Paper Class</td>
      <td><select name=cboClass></select></td>
      </tr>
      <td >Published Year:</td>
      </tr>
      <tr>
      <td>From Year(>=2000):</td>
      <td><input type=text name=txtFromYear size=4 value=xxxx></
      td>
      </tr>
      <tr >
      <td >To Year:</td>
      <td><input type=text name=txtToYear size=4 value=xxxx></td>
      </tr>
      <tr>
      <td>  </td>
      </tr>
      <tr>
      <td><input type=button name=btnSubmit value=Submit></td>
      </tr>
</table>
</form>


<%
'Check for database errors
Call CheckForErrors(objConn)
```

```
'Create the recordset object and open the recordset
Set objRS = Server.CreateObject("ADODB.Recordset")
strSQL = "qAllPaperClasses"
objRS.Open strSQL, objConn, adOpenForwardOnly, , adCmdStoredProc

'Check for database errors
Call CheckForErrors(objConn)
%>

<Script Language=VbScript>
Sub Window_OnLoad()
<%
    Do While Not objRS.EOF
%>
        Set objOption = document.createElement("OPTION")
        objOption.text = "<%=objRS("ClassName")%>"
        objOption.value = "<%=objRS("ClassName")%>"
        document.all.cboClass.add objOption
<%
        objRS.MoveNext
    Loop
%>
    <!-- #include file="Disconnect.inc" -->

    Set objOption = Nothing
End Sub

Sub btnSubmit_OnClick()
    'Verify required fields are complete

    If frmDisplay.cboClass.selectedIndex = -1 Then
        Alert "You must select your paper class"
        frmDisplay.cboClass.focus
        Exit Sub
    ElseIf Len(frmDisplay.txtFromYear.value) = 0 Then
        Alert "You must enter a starting year for displaying papers"
        frmDisplay.txtYearFrom.focus
        Exit Sub
    ElseIf Len(frmDisplay.txtToYear.value) = 0 Then
        Alert "You must enter an end year for displaying papers"
        frmDisplay.txtToYear.focus
        Exit Sub
    End If

    'If we get to this point all is OK, submit the form
    Call frmDisplay.submit()
End Sub

</script>
<%
'*************************************************************
'Step2: Display the paper information**********************
'*************************************************************
ElseIf Request.Form("FormAction") ="step2"  then
    Dim strPaperClass, intFromYear, intToYear
    strPaperClass=CStr(Request.Form("cboClass"))
    intFromYear=cLng(Request.Form("txtFromYear"))
    intToYear=cLng(Request.Form("txtToYear"))

    'Check for database errors
    Call CheckForErrors(objConn)
```

```
        Set objCmd = server.CreateObject("ADODB.command")
        Set objCmd.ActiveConnection=objConn
        objCmd.CommandText="{Call qDisplayPapers ('" &strPaperClass & _
        "'," &intFromYear & "," &intToYear &")}"
        Set objRS = objCmd.execute

        'Check for database errors
        Call CheckForErrors(objConn)
%>

<%
        if objRS.EOF or objRS.BOF then
                Response.Write "No paper record is available during this period"
%>
                <a href="Options.asp"
                onmouseover="window.status='Return to Options Page'"
                onmouseout="window.status=''">Return to Options Page</a
<%
        else
%>
        <!--Build the table title row-->
        <table border=1 cellspacing=1>
        <tr>
        <td colspan=6 align=center><font color=teal>Published papers on <%=strPaperClass%>
        from year <%=intFromYear%> to <%=intToYear%> using the simulation program</td>
        </tr>
        <tr>
        <th bgcolor=navy><font color=white>Author Name</font></th>
        <th bgcolor=navy><font color=white>Year</font></th>
        <th bgcolor=navy><font color=white>Journal Name</font></th>
        <th bgcolor=navy><font color=white>Volume</font></th>
        <th bgcolor=navy><font color=white>Page</font></th>
        <th bgcolor=navy><font color=white>Title</font></th>
        </tr>

<%
        'Loop through the recordset building the table
        Do While Not objRS.EOF
%>
        <!--Build a row of data in the table-->
        <tr>
        <td><%=objRS("AuthorName")%></td>
        <td><%=objRS("PublishedYear")%></td>
        <td><%=objRS("JournalName")%></td>
        <td><%=objRS("JournalVolume")%></td>
        <td><%=objRS("PageNo")%></td>
        <td><%=objRS("PaperTitle")%></td>
        </tr>
<%
        objRS.MoveNext
Loop
%>
        <!--Build the last row in the table with a hyper link to the options page-->
        <tr>
        <td colspan=6>  </td>
        <tr>
        <tr>
        <td colspan=6><a href="Options.asp"
                onmouseover="window.status='Return to Options Page'"
                onmouseout="window.status=''">Return to Options Page</a></td>
        </tr>
```

```
        </table>

<!-- #include file="Disconnect.inc" -->


<%
        end if
end if
%>

</BODY>
</HTML>
```