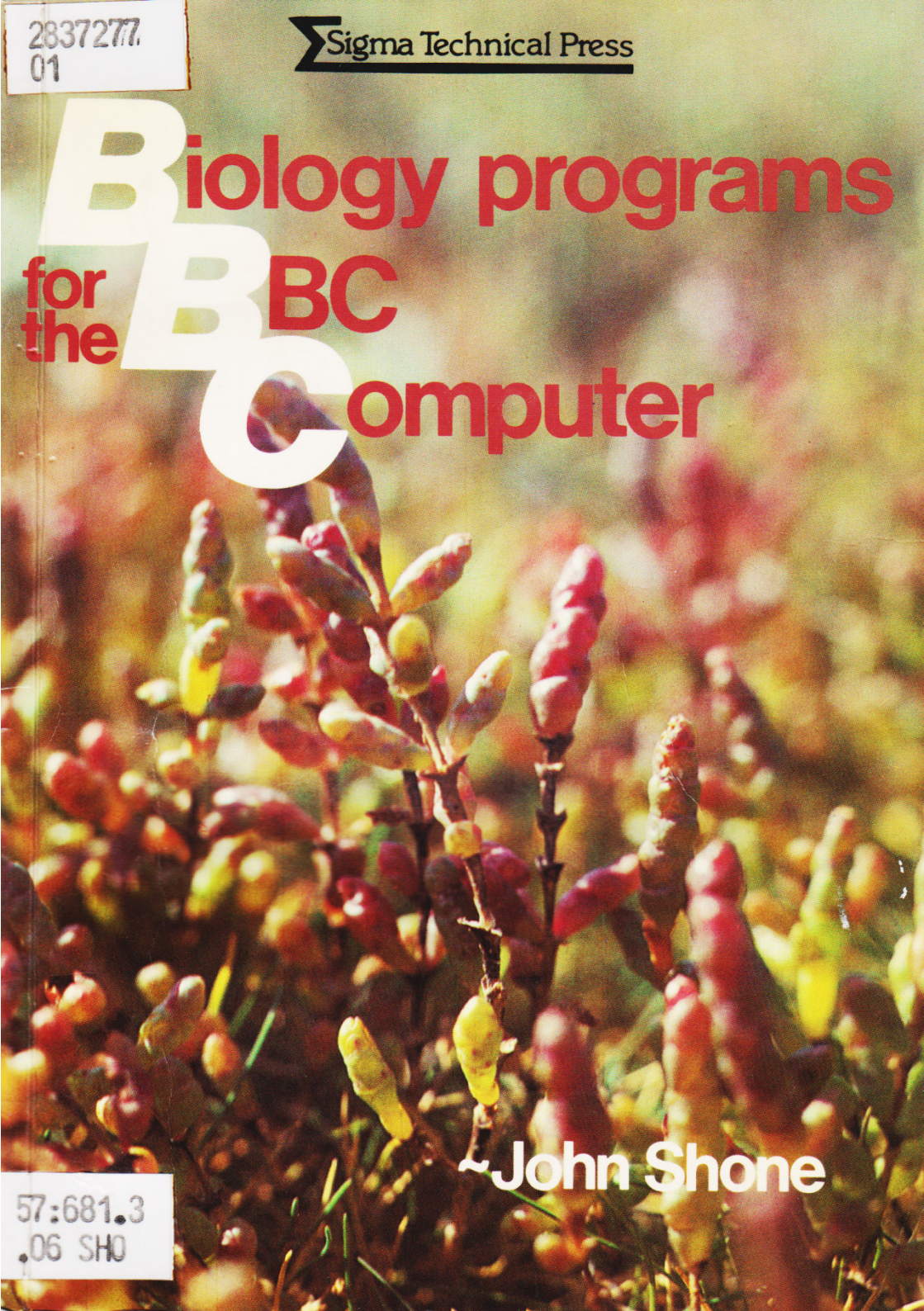


2837277.
01

 Sigma Technical Press



Biology programs for the BBC computer

~ John Shone

57:681.3
.06 SHO

Biology Programs for the BBC Computer

by

John Shone

 **Sigma Technical Press**

Copyright © 1983 by John Shone

All Rights Reserved

No part of this book may be reproduced or transmitted by any means without the prior permission of the publisher. The only exceptions are for the purposes of review, or as provided for by the Copyright (Photocopying) Act or in order to enter the programs herein onto a computer for the sole use of the purchaser of this book. This book is supplied without representation or warranty of any kind. Neither the author nor the publisher accepts any liability from the supply or use of any material contained herein.

ISBN 0 905104 53 6

Cover: a photograph of the glasswort, *Salicornia*, taken by the author.

Published by:

SIGMA TECHNICAL PRESS,
5 Alton Road,
Wilmslow,
Cheshire,
UK.

Typesetting and Production, interfaced with an Apple IIe by:

DESIGNED PUBLICATIONS LTD.
8-10 Trafford Road,
Alderley Edge,
Cheshire

Distributors:

Europe, Africa:
JOHN WILEY & SONS LIMITED,
Baffins Lane, Chichester,
West Sussex, England.

Australia, New Zealand, South-East Asia:
Jacaranda-Wiley Ltd., Jacaranda Press,
JOHN WILEY & SONS INC.,
GPO Box 859, Brisbane,
Queensland 40001, Australia.

Printed and bound in Great Britain by
J. W. Arrowsmith Ltd., Bristol

PREFACE

The aim of this little book is to present to teachers of biology, and their students, some computer programs which have proved useful in the author's teaching. I hope that they will prove useful in the reader's teaching and/or learning.

The programs presented bear, inevitably, a personal stamp. I am not a computer person, I am a biologist who has tried to integrate computers into biology teaching. The programs will probably appear rather unsophisticated in their coding to a computer professional. But, I hope that biologists will find their biology useful, or at least stimulating.

I have tried to present a range of programs covering those areas of biology which lend themselves to computer use. It seems to me that biologists can use computers in their teaching in a number of ways:

- (i) in the simulation of experiments that are too dangerous, difficult or time-consuming to be attempted in the school laboratory;
- (ii) to provide second-hand data from model systems that can be the starting point of discussion and investigation;
- (iii) to assist in repetitive statistical procedures used by biologists.

There are others, but these three are the ones upon which this book concentrates.

Software is expensive to buy. It is relatively cheap to write - except in terms of time! One aim of this book is to encourage teachers and students of biology to write their own programs. With this aim in mind, the structure of each program is outlined, alongside the full listings. A brief biological or theoretical introduction accompanies each program to assist students or non-biologists. There is a section in most program descriptions with hints and/or descriptions of each program's use.

The programs were all written in BBC BASIC for BBC Model B, OS1.2, though most will run on Model A machines.

Dedication

To my mother

CONTENTS

Glossary of Biological Terms

ix

Chapter 1: Physiology

DISSOC -	Simulation of oxygen - haemoglobin association	2
ACTPOT -	Nerve fibre simulation	12
COLOUR -	The additive mixing of light	20
SAFE -	The rhythm method of contraception	23

Chapter 2: Number Crunching

REGANAL -	Least squares regression analysis	33
CORREL -	Using correlation coefficients to assess the correlation between two variables	46
SPECDIV -	Calculating species diversity indexes	50
CHISQ -	Using the chi-squared statistic to determine the probability of observed frequencies	55
ASSOC -	Degree of association between two species in a habitat	59
AGINDEX -	Determination of aggregation indexes	67
T-TEST -	Testing for the significance of the difference between the means of two samples	73

Chapter 3: Genetics

MENDEL -	An illustration of classical genetics	80
LINK -	Autosomal genetic linkage, using various types of fowl	90
SEXL -	Sex linked characteristics	97
SICKLE -	Simulation of sickle cell anaemia genetics	103
DRIFT -	Simulation of genetic drift	111

Chapter 4: Biochemistry

FRAMES -	Helping to crack the DNA code by determining the number of bases in a single DNA codon	121
OLIGO -	A simulation to determine the primary structure of a small protein	134

Chapter 5: Ecology and Behaviour

PHYTO -	Plankton production in temperate waters	150
EPIDEM -	Simulation of the spread of diseases	162
POPEST -	Estimating population sizes	168
KLINO -	Orientation movements in animals	176

ACKNOWLEDGEMENTS

Many people have been involved directly or indirectly in the preparation of this book. Particular thanks must go to Angela Shone who painstakingly read and corrected the manuscript, and to Chris Dove who took the photographs.

I should like to thank Dr. G.A. Thompson for the epidemic model, and Peter Robinson who suggested the idea for OLIGO. I also thank many colleagues at Wyggeston and Queen Elizabeth College for their help and discussion, in particular Lynne Rigby, Cathy Lucas, Dennis Cooper, Chris Dove, Jon Scaife, Paul Jukes and Brian Fiddian.

I am grateful to the following ex-students of Wyggeston and Queen Elizabeth College who collected the data presented in the description of SPEC DIV: Philip Bird, Judith Bunten, Chris Harman, Tricia Hughes, Rekha Lad, Keerti Madhvani, Vijay Sanghani, Michael Power and Stephen Smith. Thanks to them, and other students, for being 'experimental animals' on whom most of the programs were tried.

Thanks also to the members of the Leicestershire Biology Software Group for their help and criticism.

I am grateful to the following authors and publishers for permission to use their material.

Chapter 1: COLOUR: Listing reproduced from the School Science Review by permission of The Association for Science Education.

Chapter 2: REGANAL: Data on breathing and pulse rates taken from 'Mathematics in Biology', D.C. Carter, M.S. Godsen, A. Orton, G.T. Wain and C. Wood-Robinson (1981) by permission of Thomas Nelson and Sons Ltd.

Data on metabolic rates and body mass by permission of M.K. Sands and John Murray (Publishers) Ltd., and Prentice Hall, Inc., Englewood Cliffs, N.J., the data being reproduced from Knut Schmidt-Neilson, 'Animal Physiology', 2nd Ed., 1964.

T.TEST: Data on conception rates in artificially inseminated cattle taken from R.C. Campbell, 'Statistics for biologists', Cambridge University Press.

Chapter 4:

OLIGO: Amino-acid data reproduced from:
A.L. Lehninger, 'Biochemistry', 2nd Ed., 1975 by permission of
Worth Publishers Inc.,
J. Shone, 'Determining the primary structure of an oligopeptide
- a computer simulation', J. Biol. Ed., 13, 123-126, (1979) by
permission of the Institute of Biology, and M.B.V. Roberts,
'Biology A Functional Approach Students Manual', 1974 by
permission of Thomas Nelson and Sons Ltd.

Glossary of Biological Terms

Allele:	A gene occupying a particular position on a chromosome.
Autosome:	A chromosome other than the sex chromosomes.
Bacteriophage ('Phage'):	A virus which uses bacterial cells as its hosts for reproduction.
Biosphere:	All the living and dead (as opposed to 'never alive') parts of the Earth.
Electrophoresis:	A technique used to separate molecules, often proteins, according to their charge.
Endemic:	An endemic disease is one usually associated with a particular locality. Malaria is endemic to West Africa.
Epiphyte:	A plant growing on another. Neither plant suffers any detriment from the association.
EHWN:	Extreme high water neap. The highest point on a beach reached by the tide when the annual tidal range is at its lowest.
Gene Pool:	All the genes of all individuals in a population.
Genotype:	The genetic makeup of an individual.
Giant Fibres:	Large diameter axons found in the nerve cords of many invertebrates carrying high velocity impulses. Often involved in escape behaviour.
Globular proteins:	Protein molecules folded in such a way that they become 'ball-like'. Most enzymes are globular.
Haemolytic anaemia:	A reduction in the number of red blood cells in the vascular system brought about by their destruction.

Haemophilia:	An hereditary disease in which blood clots very slowly.
Homeostasis:	A homeostatic system maintains its state in spite of environmental disturbance. For example, the body temperature of mammals and birds is regulated to remain constant at all times.
Industrial melanism:	Many organisms living in urban environments are darkly coloured.
Locus:	A particular location on a chromosome.
Meiosis:	The production of gametes by cell division. As a result of meiosis, daughter cells have half the number of chromosomes as their parental cells.
Mutagens:	Agents, such as ionising radiation or chemical compounds which cause mutation.
Mutation:	An inherited change in the structure of a gene.
Oxytocin:	A hormone produced by the pituitary gland in mammals. It causes contraction of uterine muscle and may be involved in parturition.
Parturition:	Birth.
Phenotype:	The characteristics of an organism.
Prophase:	The initial stage of cell division; it involves the duplication of chromosomes.
Quadrat:	A device for standardised sampling in ecology. Consists of a square of metal or wood.
Recombination:	A process occurring during meiotic prophase involving the interchange of genetic material between two chromosomes.
Somatic Cell:	A cell of an organism other than a gamete or potential gamete.
TMV:	Tobacco mosaic virus. A plant pathogen.
Turbellarian Platyhelminth	A freeliving flatworm. Most are freshwater.

CHAPTER 1

PHYSIOLOGY

Introduction

Students of biology in schools and colleges spend a good deal of their time investigating and discussing physiology, the integrated mechanisms of living things. As much as 60% of students' time at O level, a little less at A level and subsequently, will be spend learning physiology.

Physiology is essentially an experimental branch of biology and the teaching and learning of physiology should reflect this. However, many physiological experiments are not possible in schools and colleges. The necessary equipment may not be available, or there may not be time in the rush to prepare students for external examinations. Experiments involving human subjects are usually (though not always) inappropriate. Traditionally, then, second hand sources are used to introduce students to the techniques and results of physiological experimentation.

Computer simulation can provide a better substitute than raw data on the printed page for first-hand experimental experience. Students can investigate physiological phenomena themselves. They can define the magnitude of physiological variables and collect results from a computer model which can be used to support or refute hypotheses. With a computer, and adequate teacher guidance, they can DO physiology.

Three of the programs described in this chapter enable the simulation of physiological situations which cannot be at all easily presented experimentally in the school or college laboratory. The fourth is a demonstration of a phenomenon that students often find surprising.

DISSOC

This program involves the simulation of experiments that investigate the nature of the oxygen-haemoglobin dissociation curve.

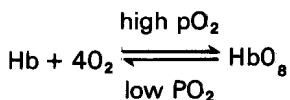
The Biological Background

The vascular system of animals probably evolved due to the need to transport respiratory gases to and from breathing surfaces, such as lungs or gills, and actively respiring tissues.

A good deal of respiratory gas, oxygen or carbon dioxide, is carried in solution in plasma, the liquid part of blood. This amount, in the case of oxygen, is roughly 0.3cm^3 per 100cm^3 plasma in mammals (1). The corresponding figure for whole blood is somewhere between 14 and 19cm^3 per 100cm^3 of blood (1). Clearly, most oxygen is carried in blood in some combined form.

Indeed, oxygen is transported from the breathing surface of mammals, and other animals, combined with a rather complex compound, haemoglobin. The compound consists basically of two parts, a protein, globin, and an iron-containing haem group. This latter is the 'business-end' of the molecule and is the site of the binding of oxygen (2).

Each haem group, in the appropriate environment can bind 4 molecules of oxygen to produce the unstable compound oxyhaemoglobin. The instability of oxyhaemoglobin means that in the appropriate circumstances, oxygen can be released and haemoglobin regenerated. Basically, in tissue where the partial pressure ($p\text{O}_2$) of oxygen is low, oxyhaemoglobin dissociates to release oxygen. In regions such as the alveoli of the lung, where the $p\text{O}_2$ is high, the association of oxygen and haemoglobin is promoted. These ideas are often summarised by the simple equation:



where Hb refers to haemoglobin, and HbO_8 to oxyhaemoglobin.

The nature of the association between oxygen and haemoglobin, and its modification by varying physical factors, is however, not simple. The relationship between the percentage saturation of haemoglobin with oxygen and $p\text{O}_2$, the oxygen-haemoglobin dissociation curve, is not linear. In fact, the curve is distinctly sigmoid, or S-shaped.

The flat part of the curve corresponds to high values of $p\text{O}_2$. Hence, quite wide variations in atmospheric $p\text{O}_2$ have minimal effects on the oxygen saturation of haemoglobin, so on the loading of haemoglobin with oxygen in the lung.

The oxygen-haemoglobin dissociation curve is much steeper at low values of pO_2 , such as are found in respiring tissues, which form an oxygen sink, continually taking up oxygen for their respiration. Low values of pO_2 promote rapid dissociation of oxyhaemoglobin by decreasing the affinity of haemoglobin for oxygen (2). At low pO_2 , small changes in pO_2 promote large changes in haemoglobin saturation.

The affinity of haemoglobin for oxygen is also modified by factors other than surrounding pO_2 . Decreasing the pH at any particular value of pO_2 promotes oxyhaemoglobin dissociation. This is known as the Bohr effect. Such pH changes are produced in exercising tissue by the rapid evolution of carbon dioxide. Hence, increased oxygen demand in active tissue is met by an increased supply. The active tissues themselves promote an improved oxygen unloading by moving the oxygen-haemoglobin dissociation curve to the right.

A similar modification in haemoglobin's oxygen affinity is produced by an increase in temperature. Once again, as with increasing pCO_2 , an increased temperature moves the dissociation curve to the right. With increasing temperature, blood unloads more oxygen into respiring tissue. Increased tissue activity, and so respiration, produces a local rise in temperature due to the less than complete efficiency of the respiratory process.

Neither the Bohr effect or the temperature effect have much significance at high pO_2 , due to the shape of the oxygen-haemoglobin dissociation curve. The nature of the dissociation curve is a beautiful example of adaptation at the biochemical level. Further information can be obtained from Maclean's excellent little book (2).

The Computer Program

Information about the program DISSOC was first published in *School Science Review* in 1982 (3).

The nature of the oxygen-haemoglobin dissociation curve has been modelled by assuming that the probability at which oxygen is bound to the haem group is increased if oxygen is already present in the molecule. This leads to the kind of non-linear kinetic model first suggested by Monod et. al. (4). This work concerned some rather special enzymes, and haemoglobin now glories in being an 'honorary enzyme'!

A detailed discussion of the kinetics of the so-called allosteric enzymes can be found in Engel (5). A simplified version of the model was used as the basis of the computer simulation DISSOC. In particular, equation 7.4 on page 85 of Engel's book forms the core of the model around which DISSOC was written. This equation occurs in a slightly modified form in line 1510 of the program listing.

Program Description

On running the program, the user is given some brief information about the nature

of the program. Then an option is given enabling the choice of the production of

1. A standard dissociation curve at $p\text{CO}_2$ 40mmHg and 38°C

or

2. Up to 3 dissociation curves calculated at varying values of $p\text{CO}_2$ and temperature, and printed in different colours on the same axes.

The vertical axis is labelled %Satn, meaning percentage saturation of haemoglobin with oxygen. The horizontal axis is $p\text{O}_2$. The curves are calculated and plotted using the procedure DISSOCURVE, lines 1310 to 1470. First of all, the axes for the graph are plotted by calling the procedure AXES, lines 1550 to 1690. Then each dissociation curve in turn is plotted. Each is preceded by a printout of the corresponding values of $p\text{CO}_2$ and temperature, lines 1340 to 1410. The X coordinate for each point of the curve is assigned by the loop starting at line 1430, and for each X coordinate a corresponding Y coordinate is calculated in the procedure CALCY, lines 1500 to 1520. The X coordinate is $p\text{O}_2$, the Y coordinate saturation of haemoglobin with oxygen. In the procedure CALCY, the value of $p\text{CO}_2$ used is that input by the user, but the value used for temperature is a linear transformation of the input value, realtemp, calculated in line 1420. This is done to increase the sigmoidicity of the resulting curve. This process cannot be justified analytically, it just makes the program work better!

Note that *FX200,1 disables the ESCAPE key, while *FX200,0 re-enables it. The positioning of ESCAPE next to 1 can be a little annoying if mistakenly pressed. *FX200,1 is a very useful alternative to ONERROR for error trapping ESCAPE. VDU23,1,0;0;0;0; turns off the flashing cursor.

More curves could be plotted by changing the values of A and the dimensions of arrays.

Using the Program

This program is designed as a package either for directed investigation, as a discussion stimulator, or for individual student use. The former is, in the absence of DISSOC, carried out using second-hand data such as that found in the Nuffield Study Guide (6). While this approach is certainly adequate, the use of semi-experimental material, over which the user has some control adds a lot to the appeal of the material.

The oxygen-haemoglobin dissociation curve is a topic that students of Advanced level Biology and Social Biology tend to find difficult at first meeting. The use of DISSOC may help.

After producing a standard dissociation curve, students can be led into suggesting that other physical variables, apart from $p\text{O}_2$ may determine the nature of the oxygen-haemoglobin association. Such hypotheses can then be tested by running the variable conditions part of the program.



Figure 1.1. A typical computer layout for student use.

DISSOC Listing

```

10REM DISSOC
20REM
30REM OXYHAEMOGLOBIN DISSOCIATION CURVE SIMUL
ATION
40REM
50*FX200,1
60REM
70REM INTRODUCTION PAGE
80REM
90MODE7
100VDU23,1,0;0;0;0;
110PROCSTART
120REM
130REM

```

```

140REM CHOOSE CONDITIONS
150REM
160MODE7
170VDU23,1,0;0;0;0;
180PROCCONDITIONS
190IF conditions=2 THEN560
200REM
210REM
220REM STANDARD CONDITIONS OF TEMPERATURE AND
pCO2
230REM
240VDU12
250PROCPRINT
260PRINT"Standard conditions"
270PROCPRINT
280PROCPRINT
290PRINT"Temperature is 38C"
300PRINT"pCO2 is 40mmHg"
310PROCPRINT
320PROCCONTINUE
330curves=1
340temp%(curves)=38
350B(curves)=40
360REM
370REM
380REM CALCULATE AND PRINT DISSOCIATION CURVE
390REM
400MODE4
410VDU23,1,0;0;0;0;
420PROCDISSOCCURVE
430REM
440REM
450REM PROGRAM CONTINUATION OPTION
460REM
470PROCANOTHERGO
480IF LEFT$(A$,1)="N"THEN PROCFINISH ELSE 160
490REM
500REM
510REM VARIABLE CONDITIONS OF TEMPERATURE AND
pCO2
520REM

```

```

530REM
540REM CHOOSE UP TO THREE CURVES
550REM
560 VDU12
570PROCPRINT
580PRINT"Variable conditions"
590PROCPRINT
600PROCCURVES
610REM
620REM
630REM INPUT VARIABLES
640REM
650VDU12
660FORI=1TOcurves
670PRINT
680PRINT
690PRINT"CURVE ";I
700PRINT
710REPEAT
720INPUT"Temperature (C)",temp
730IFtemp<1 OR temp>99 THENPRINT "Not in range
":PRINT
740UNTILtemp>0ANDtemp<100
750temp%(I)=temp
760PRINT
770REM
780REPEAT
790INPUT"pCO2      (mmHg)",B(I)
800IFB(I)<0ORB(I)>100PRINT"Out of range":PRINT
810UNTILB(I)>=0ANDB(I)<=100
820NEXTI
830REM
840REM
850REM CALCULATE AND PRINT DISSOCIATION CURVE(
S)
860REM
870MODE1
880VDU23,1,0;0;0;0;0;
890PROCDISSOCCURVE
900REM
910REM

```

```

920REM PROGRAM CONTINUATION OPTION
930REM
940PROCANOTHERGO
950IF LEFT$(A$,1)="N"THEN PROCFINISH ELSE 160
960REM
970REM
980DEFPROCPRINT
990FORI=1TO5
1000PRINT
1010NEXTI
1020ENDPROC
1030REM
1040REM
1050DEFPROCCONTINUE
1060PRINT"Press SPACE BAR to continue"
1070XX=GET
1080ENDPROC
1090REM
1100REM
1110DEFPROCCONDITIONS
1120VDU12
1130PRINT"Standard or variable conditions?"
1140PROCPRINT
1150PRINT"For standard conditions (One curve)"
1160PRINT"Type 1"
1170PRINT
1180PRINT
1190PRINT"For variable conditions (>One curve)"
1200PRINT"Type 2"
1210PROCPRINT
1220REM
1230REPEAT
1240INPUT"Standard or variable conditions",cond
itions
1250conditions=INT(ABS(conditions))
1260IF conditions>20Rconditions<1THEN PRINT:PRI
NT"Type 1 or 2":PROCPRINT
1270UNTILconditions=10Rconditions=2
1280ENDPROC
1290REM
1300REM

```

```

1310DEFPROC DISSOCURVE
1320PROCAXES
1330PROCPRINT
1340FOR A=1 TO curves
1350GCOL1,A
1360MOVE0,0
1370COLOURA
1380PRINT
1390PRINT"Curve ";A
1400PRINT"Temp=";temp%(A);"C"
1410PRINT"pCO2=";B(A);"mmHg"
1420realtemp=((1/temp%(A)^2.4))*2500)
1430FORX%=0 TO 770 STEP 5
1440PROCCALCY
1450DRAWX%,Y
1460NEXTX%,A
1470ENDPROC
1480REM
1490REM
1500DEFPROC CALCY
1510Y=800*(X%*(X%+1)^3)/(((1500*(B(A)+1)^4)/((r
ealtemp+1)^4))+((X%+1)^4))
1520ENDPROC
1530REM
1540REM
1550DEFPROC AXES
1560VDU29,400;200;
1570VDU5
1580MOVE-200,750
1590PRINT"% Satn"
1600VDU4
1610MOVE0,0
1620DRAW0,800
1630MOVE0,0
1640DRAW800,0
1650VDU5
1660MOVE750,-50
1670PRINT"pO2"
1680VDU4
1690ENDPROC
1700REM

```

```

1710REM
1720DEFPROCSTART
1730DIMtemp%(3),B(3)
1740PROCPRINT
1750PRINT"                DISSOC"
1760PROCPRINT
1770dummey=INKEY(500)
1780VDU12
1790PROCPRINT
1800PRINT"This investigation will enable"
1810PRINT"you to investigate the form of the"
1820PRINT"oxygen-haemoglobin dissociation curve
"
1830PROCPRINT
1840PROCPRINT
1850PROCCONTINUE
1860ENDPROC
1870REM
1880REM
1890DEFPROC CURVES
1900REPEAT
1910INPUT"How many curves",curves
1920IF curves >3 THENPRINT:PRINT"You may have u
p to 3 only":PRINT
1930UNTILcurves>0ANDcurves<4ANDcurves=INT(curve
s)
1940ENDPROC
1950REM
1960REM
1970DEFPROCANOTHERGO
1980VDU29,0;0;
1990MOVE10,100
2000VDU5
2010GCOLOR,3
2020INPUT"Another go",A$
2030VDU4
2040ENDPROC
2050REM
2060REM
2070DEFPROCFINISH
2080VDU12

```



```

2090COLOUR3
2100PROCPRINT
2110PROCPRINT
2120PRINT "
2130XX=GET
2140*FX200,0
2150END
2160ENDPROC

```

END "

DISSOC - Variables

conditions	value 1 for standard conditions, 2 for variable temperature and $p\text{CO}_2$ conditions
curves	number of dissociation curves plotted on same axis space
temp%(n)	temperature for curve n
B(n)	$p\text{CO}_2$ for curve n
A\$	user response to program continuation offer
I	loop control variable
XX	dummy variable of GET function
A	loop control variable for curve plotting loop
realtemp	linear transformation of temp%
X%	X coordinate for dissociation curve
Y	Y coordinate for dissociation curve
dummeY	delay in program execution

ACTPOT

Biological Background

Nerve fibres, in both vertebrate and invertebrate animals, appear to transmit information in the same kind of way. The basic unit of nervous information is the nervous impulse. This is an electrochemical event, the existence of which can be detected electronically.

Electrodes placed inside, or outside an active nerve fibre, can be used to detect small changes in the electrical potential of the membranes of nerve fibres or axons which correspond to the passing of a nervous impulse. When recorded internally, this change of membrane potential is called an action potential.

During an action potential, the membrane potential switches from its resting value of approximately -70mV to approximately $+50\text{mV}$ (7). Due to this sudden reversal of membrane potential at the site of an action potential, local currents are set up between the active region and adjacent resting regions. These local currents activate resting regions to generate their own action potentials. The existence of these local currents can be detected by electrodes external to the nerve fibre.

Because the membrane of a nerve fibre can be activated by small local electric currents, action potentials can be generated in nerve fibres using some external voltage source, called a stimulator. The generation of a stimulus pulse can be recorded using an oscilloscope, as can the action potential.

It is found that when an electrical stimulus is applied to a nerve fibre, the stimulus must be above a certain minimum strength to elicit an action potential. The nerve fibre has a threshold for its activation. Stimuli can be above or below threshold. If below, they do not initiate a nervous impulse, though they may generate some non-propagating depolarisation of the nerve cell membrane.

A strange phenomenon involving the nervous impulse is that the degree of membrane depolarisation called an action potential is independent of the magnitude of supra-threshold stimuli. The nervous impulse is not graded in relation to stimulus intensity. It is an all-or-none affair. It occurs or it doesn't; and when it does occur, it occurs completely. Information is not coded in the nervous system using the varying heights of impulses.

Action potentials have a duration of roughly one millisecond. So, in theory, the maximum frequencies of the discharge of impulses should be of the order of 1000 sec^{-1} . Such a figure is rarely reached. Most nerve fibres in mammals have impulse discharge rates of $5\text{--}100\text{ impulses sec}^{-1}$ (7).

This can be shown to be due to the fact that once a section of nerve cell membrane has been depolarised into an action potential, it becomes temporarily insensitive to further stimulation. This insensitive period is known as the refractory period of the nerve fibre. The first part of the refractory period is termed the absolute refractory period, and during this time, stimuli of any magnitude will not evoke an

action potential. During the subsequent relative refractory period, stimuli with magnitudes significantly above normal threshold strength may elicit responsiveness. The refractory period sets an upper limit on the frequencies of impulses in nerve fibres.

The Computer Program

The program ACTPOT is a very simple simulation program to demonstrate the phenomena of threshold and refractory period. The all-or-none law can also be inferred.

The VDU screen is imagined to be the screen of a double-beam oscilloscope. The top beam is connected to electrodes resting on an exposed nerve of an organism. The bottom beam shows the delivery of electrical stimuli to the nerve concerned. For the sake of simplicity, the nerve is assumed to contain a solitary nerve fibre, such that each supra-threshold stimulus evokes a single action potential or impulse.

Pressing RETURN triggers an electrical stimulus (single pulse) to the nerve. The stimulator is imagined to be connected to the oscilloscope lower beam on which occurs a stimulus mark at the time of stimulus delivery.

Both upper and lower 'beams' move across the screen giving traces of voltage (vertical axis) against time (horizontal).

When a stimulus has been delivered to the nerve, provided that it is of supra-threshold magnitude, an impulse is generated in the nerve fibre. After a time, called the latent period, the action potential appears on the upper 'beam'. The action potential is accompanied by a short beep on the computer loudspeaker. Electrophysiologists often hook the recording electrodes on the nerve to a loudspeaker via an amplifier. The latent period occurs due to the fact that the stimulating and recording electrodes are some distance apart on the nerve, and the impulse initiated by stimulation takes a little time to reach the recording electrodes. Impulse conduction velocity may be as high as 100 m sec^{-1} , which is, of course, only a fraction of the speed of electrons moving through a metal conductor. This reinforces the notion that nervous conduction is not strictly an electrical phenomenon, but an electrochemical one.

The action potential generated by a supra-threshold stimulus is modelled as a biphasic one, typical of externally recorded signals from nerve fibres. Once an action potential has been elicited in the nerve fibre, the nerve fibre becomes temporarily refractory, and immediate further stimulation causes no response.

Both 'beams' move across the screen until they come to the end when the user is asked whether he wishes to continue running the program.

Program Description

The program uses MODE 1 to produce a graphics simulation of a double-beam

CRO. The beams are labelled in lines 340-440. Text is written to the graphics cursor using VDU5, cancelled with VDU4. VDU29,400;0; moves the origin for graphics to 400,0.

The procedure **nostimulus** is then called in a REPEAT....UNTIL loop, lines 490 to 590. The loop ends when the graphics cursor has moved off the screen, at which time the value of POINT (Xcoordinate, stimtrace) is -1. During the loop, a check is made to determine whether the RETURN key has been pressed, and so whether a stimulus should be sent to the nerve fibre. This is done in line 550. If RETURN is hit, INKEY (-74) is true, and the procedure **stimulus.event** is called.

The procedure **nostimulus**, lines 770 to 910 produces the beams of the stimulus and nerve records. The nerve record is drawn first, using DRAW in line 840, and advancing Xcoordinate by steps of 2. The nerve trace is drawn in yellow, GCOL 1,2, line 810, the stimulus trace in red, GCOL,1 in line 880.

The procedure **stimulus.event** is called if RETURN is hit during the beam's traverse of the screen. This procedure occupies lines 950 to 1300. First of all, the keyboard is disabled using *FX201,1 to prevent more input from the user. A stimulus mark is then drawn in red to a height determined by the stimulus intensity previously input by the user. The procedure **nostimulus** is then called up in a REPEAT...UNTIL loop, lines 1060 to 1080 to produce the latent period of the nerve fibre.

A check is then made in line 1120, to determine whether the value of intensity, corresponding to stimulus magnitude, is below threshold. If it is, program control is sent to line 1290, the keyboard is enabled, and the procedure **stimulus.event** exited. If program control does not jump at line 1120, the current value of the computer internal clock, TIME, is compared to the variable time (see below). This is done to set the refractory period. Changing the 100 in line 1160 will modify the refractory period of the nerve fibre. If TIME is less than time+100, a stimulus has been delivered in the refractory period, and the procedure is exited as before, control passing to line 1290. On the other hand, if TIME is greater than or equal to time+100, a stimulus has been received after the end of refractory period (or is the first stimulus), and an action potential is elicited.

The action potential is drawn on the upper trace in lines 1200 to 1230. The value of TIME is recorded in the variable time, which sets the beginning of the refractory period following the action potential. VDU7 in line 1250 generates a bleep. The procedure is then exited.

*FX21,0 in line 510 flushes the input buffer of RETURNS.

*FX200,1 disables the ESCAPE key.

Using the Program

On running the program, the user is given a little information on the purpose of the program, and informed that he can deliver stimuli to the nerve fibre using RETURN.

The user is then asked to choose a value for stimulus intensity on a scale from 1 to 10. This value will remain constant during any particular run. Intensity values of 5 and above are above threshold. Choosing values below 5 will mean that the nerve fibre will not respond to stimuli generated by pressing RETURN.

Having accepted a value for intensity, the computer generates the two simulated oscilloscope beams. Pressing RETURN at this point in the program generates a stimulus mark on the lower trace. The height of this mark is dependent upon the chosen stimulus intensity. Supra-threshold stimuli cause the production of an action potential, which will be seen not to vary in size, no matter what stimulus magnitude is chosen. This demonstrates the all-or-none rule.

Further, if stimuli are presented during the refractory period, no further response is produced. This can be demonstrated by keeping the RETURN key pressed. A series of stimulus marks will result, not all of which are followed by action potentials.

At the end of a run, the user has the option of running the program again, perhaps at a new value of stimulus intensity or ending the program.

ACTPOT Listing

```
10REM ACTPOT
20REM
30REM SET UP AND INTRODUCTION
40REM
50*FX200,1
60MODE7
70VDU23,1,0;0;0;0;0;
80PRINTTAB(0,10);"This program enables you to
""investigate the properties of threshold""and
nd refractory period for a nerve fibre."
90PROCcontinue(5,20)
100CLS
110PRINTTAB(0,10);"RETURN fires a single stimu
lus to""the nerve fibre."
120PROCcontinue(5,20)
130REM
140REM
150REM INPUT INTENSITY
160REM
170REPEAT
180MODE7
```

```

190VDU23,1,0;0;0;0;0;
200PRINTTAB(10,10)
210INPUT"Intensity (1 TO 10)",I
220UNTIL I>=1 AND I<=10
230REM
240REM
250REM INITIALISE VARIABLES SET time TO ZERO
260REM
270time=0
280intensity=I*40
290nervetrace=600
300stimtrace=200
310Xcoordinate=0
320REM
330REM
340REM SET UP GRAPHICS
350REM
360MODE1
370VDU23,1;0;0;0;0;0;
380VDU29,400;0;
390VDU5
400MOVE-300,200
410PRINT"Stimulus"
420MOVE-300,600
430PRINT"Nerve"
440VDU4
450REM
460REM
470REM GENERATE PROGRAM OUTPUT
480REM
490REPEAT
500PROCnostimulus
510*FX21,0
520REM
530REM CHECK WHETHER RETURN KEY PRESSED
540REM
550IF INKEY(-74)THENPROCstimulus_event
560REM
570REM CHECK GRAPHICS CURSOR NOT OFF SCREEN
580REM
590UNTIL POINT(Xcoordinate,stimtrace)=-1

```

```

600REM
610REM  ENABLE  KEYBOARD
620REM
630*FX201,1
640REM
650PROCcontinue(1,30)
660MODE7
670VDU23,1,0;0;0;0;0;
680PRINTTAB(5,10);
690INPUT"Another run",answer$
700IF LEFT$(answer$,1)="Y" OR LEFT$(answer$,1)
="y"THEN170
710PRINTTAB(15,20);"END"
720*FX200,0
730END
740REM
750REM
760REM
770DEFPROCnostimulus
780REM
790REM  DRAW  NERVE  TRACE
800REM
810GCOL1,2
820MOVEXcoordinate,nervetrace
830Xcoordinate=Xcoordinate+2
840DRAWXcoordinate,nervetrace
850REM
860REM  DRAW  STIMULUS  TRACE
870REM
880GCOL1,1
890MOVEXcoordinate-2,stimtrace
900DRAWXcoordinate,stimtrace
910ENDPROC
920REM
930REM
940REM
950DEFPROCstimulus_event
960REM
970REM  DISABLE  KEYBOARD
980REM
990*FX201,1

```

```

1000newX=Xcoordinate
1010REM
1020REM DRAW STIMULUS MARK
1030REM
1040GCOL1,1
1050DRAWXcoordinate,stimtrace+intensity
1060REPEAT
1070PROCnostimulus
1080UNTILXcoordinate=newX+20
1090REM
1100REM CHECK WHETHER INTENSITY BELOW THRESHOLD
1110REM
1120IFintensity<200THEN1290
1130REM
1140REM SET REFRACTORY PERIOD
1150REM
1160IF TIME<time+100 THEN1290
1170REM
1180REM DRAW ACTION POTENTIAL MAKE SOUND
1190REM
1200GCOL1,2
1210MOVEXcoordinate,nervetrace
1220DRAWXcoordinate,nervetrace+200
1230DRAWXcoordinate,nervetrace-70
1240time=TIME
1250VDU7
1260REM
1270REM ENABLE KEYBOARD
1280REM
1290*FX201,0
1300ENDPROC
1310REM
1320REM
1330REM
1340DEFPROCcontinue(X,Y)
1350*FX201,0
1360PRINTTAB(X,Y);"Press space bar to continue"

1370REPEATUNTILGET=32
1380ENDPROC

```


ACTPOT - Variables

I	value of intensity of stimulus, relative units
intensity	linear transformation of I for output
time	internal clock time marking an action potential event
nervetrace	screen vertical coordinate for nerve record
stimtrace	vertical coordinate for stimulus record
Xcoordinate	horizontal coordinate for both records
newX	Xcoordinate at which stimulus delivered
X,Y	cursor coordinates for program continuation message
answer\$	user response to program continuation offer

COLOUR

This is a demonstration of the additive colour mixing of light.

The Biological Background

The Young-Helmholtz theory of colour vision, first put forward in 1801 is that the retina of the eye contains receptors which are selectively stimulated by light of specific wavelengths, or narrow band of wavelengths. In particular, the cones of the eye are wavelength specific. A type of cone is selectively responsive to each of the three primary colours of red, blue and green. This is a trichromatic theory of colour vision (1).

Three different types of cone cell, containing visual pigments, iodopsins are now recognised. The perception of colour involves the additive mixing of the three primary colours, perceived by the three types of cone.

This explanation of colour vision has been modified in the recent past, not least by the elegant experiments of Land (8), but the basic idea is accepted, at least as far as O and A level students are concerned.

The Computer Program

The program COLOUR demonstrates the phenomenon of additive colour mixing. Circles of the three primary colours are generated, which in their overlapping regions demonstrate additive colour mixing.

Program Description

The program uses graphics mode 2 to generate three overlapping coloured circles. The three circles are generated in the procedure CIRCLE, lines 250 to 330. The procedure is called three times from the loop, lines 170 to 210.

For each pass of the loop, a graphics colour is chosen using 211, where 1 takes the value 0, 1 or 2. Hence 211 becomes respectively 1, 2 or 4, corresponding to mode 2 graphics colours red, green and blue. GCOL 1,211 in line 180 chooses the new colour and also ORs it with the colour already present on the screen.

For example, logical colour 1 is ORed with logical colour 2 to give colour 3. 1 is binary 001, and 2 is binary 010. 1 OR 2 is binary 011, or 3. Logical colour 3 is yellow, so red and green give yellow. Logical colour 1, red, binary 001 ORed with logical colour 4, blue, binary 100 gives binary 101, logical colour 5 or magenta. Similarly, green, binary 010, ORed with blue, binary 100 gives cyan, binary 110. Applying the OR operation to all three primary colours, 001 (red), 010 (green) and 100 (blue), gives binary 111, logical colour white.

This is summarised as follows:

red,	001	OR	green,	010	gives	yellow,	011
red,	001	OR	blue,	100	gives	magenta,	101
green,	010	OR	blue,	100	gives	cyan,	110
and							
red,	001	OR	green,	010	OR	blue,	100
							gives white, 111

More on binary logic applied to GCOL can be found in the BBC User Guide (9).

READX,Y in line 190 reads two numbers from the DATA at line 100 to be the X and Y screen coordinates for the origin of a new circle. These values are used in the procedure CIRCLE, together with R, the radius, to draw a filled circle using PLOT 85 in line 300. PLOT 85 fills a triangle specified by the two previous coordinates, and the coordinate defined in the PLOT statement itself. For a circle, these are the polar coordinates defined by the cosine and sine of I , where I is an angle in radians varying from 0 to 2π . 200 values of I are chosen, the interval between them having been determined by line 120. The variable step determined in line 120 is used as step length to increment I in the circle-filling loop in lines 290 to 320.

Using the Program

Additive colour mixing with lights often comes as a shock to students who may be more familiar with subtractive pigment mixing. COLOUR can be used as a classroom demonstration of the former phenomenon.

COLOUR - Listing

```

10REM COLOUR
20REM
30REM ADDITIVE COLOUR MIXING
40REM
50REM
60REM SET UP
70REM
80MODE2
90VDU23,1,0;0;0;0;0;
100DATA400,350,650,350,525,575
110R=250
120step=(2*PI)/200
130REM
140REM
150REM DRAW CIRCLES
160REM
170FORI=0TO2
180GCOL1,2^I

```

```

190READX,Y
200PROCCIRCLE(X,Y)
210NEXTI
220END
230REM
240REM
250DEFPROCCIRCLE(X,Y)
260LOCALI
270MOVEX,Y
280MOVEX,Y
290FORI=0TO(2*PI)STEPstep
300PLOT85,R*COS(I)+X,R*SIN(I)+Y
310MOVEX,Y
320NEXTI
330ENDPROC

```

COLOUR - Variables

R	radius of circles
step	step length for circle filling loop
I	control variable for loops
X, Y	coordinates of circle origins

SAFE

To simulate an investigation into the effectiveness of the rhythm method of contraception.

Biological Background

Fertility control is an area of active and growing research in biology. The practice of contraception has consequences, not just biological, but also sociological and political. Students of biology and social biology will usually consider the biological basis of fertility control, and will also, hopefully, consider the wider implications of the practice for themselves and society at large.

With this aim in mind, an understanding of the female menstrual cycle in man becomes crucial.

The menstrual cycle begins on the first day of menstruation, the loss of the deciduous endometrium of the uterus. This developed during the previous cycle in readiness to receive a fertilised egg. If no fertilised egg implants itself into the endometrium, menstruation occurs. On average, menstruation occurs for between three to four days. From the cessation of menstruation to about the fourteenth day of the menstrual cycle, the uterine endometrium is regenerated and becomes increasingly glandular. The endometrium may become about 6-7mm thick (1).

Menstruation normally occurs on average every 28 days. However, there is considerable variation between individuals. Regular menstruation may occur every 21 to 30 days. In many women, the onset of menstruation is rather unpredictable and irregular. Irregular menstruation is particularly common at the onset of puberty, the menarche, and approaching the menopause, the cessation of menstrual periods.

As previously implied, the role of the menstrual cycle is the regular preparation of the uterus to receive and provide appropriate physiological conditions for the growth of a fertilised egg or ovum. The release of an ovum into the female reproductive tract is called ovulation. This event appears to occur in the middle of the average 28 day menstrual cycle, during a period starting at the twelfth and ending on the fifteenth day following the onset of menstruation. However, it appears that ovulation can occur at any time between the sixth and twentieth day of the cycle (1). Certainly, the more variable the menstrual cycle, the more variable the day of ovulation.

Fertilisation, the fusion of a fertile ovum and sperm, must follow ovulation quite rapidly. Ova have a very short fertile life. Estimates of this period of viability vary from as little as eight to as much as twenty-four hours (1,10). Sperms released into the female reproductive tract from the comparative inactivity of the testis may be viable for up to twenty-four hours after ejaculation. They may retain motility for longer, but after a day on their own, become increasingly infertile (1).

Fertilisation is thus possible if intercourse takes place within twenty-four hours

prior to ovulation and up to eight hours following it. It is the time of ovulation that determines the fertile days of the menstrual cycle.

The most effective methods to prevent fertilisation following intercourse, ignoring sterilization, are those which involve the prevention of ovulation. The contraceptive pill inhibits the release of pituitary follicle-stimulating-hormone, and so inhibits ovulation. Mechanical methods of contraception either prevent ova and sperms meeting, or prevent the implantation of zygotes in the uterine endometrium. The contraceptive sheath exemplifies the former, the intra-uterine device, IUD or 'coil' the latter. All these techniques have drawbacks or side-effects, and some couples may prefer to employ none of them. The Roman Catholic Church officially sanctions none of them. People may prefer to employ the rhythm, or safe-period technique to limit their fertility.

This technique relies upon being able to predict the times in the menstrual cycle when fertility is at its highest, and avoiding intercourse at these times. But, the assessment of the day of ovulation is difficult, if not impossible. There appears to be a slight variation in body temperature around the time of ovulation. However, this change may precede, or follow ovulation by as much as three days (1).

The Computer Program

The simulation SAFE is designed to allow the assessment of the reliability of the rhythm method of contraception in fertility control.

It must be stressed that there is no clinical validity in this assessment. The program is a simulation based on very simple assumptions. The factors affecting human fertility and the use of the rhythm method of contraception are extremely complex, and no reliance should be placed on the use of the program SAFE in real life.

The program prints successive days in a year and also the day of the current menstrual cycle. Days of menstruation are indicated. Intercourse (without contraceptive) can occur on any day of the cycle, and at any time of the day. Every twenty-eight days an optional pregnancy test (urinary HCG assessment) can be taken. No other information is given, except that should pregnancy occur, the user is informed of the day upon which ovulation occurred. The program can then be exited or re-run.

Program Description

The program begins by reading into arrays a string containing all the months of the year. This string will be subsequently used to produce the date printout in the program. The string is accessed using vectors. S(n) gives the starting position of the nth month in Y\$, the string containing all the month names, L(n) contains the number of characters in the name of the nth month. D(n) contains the number of days in the nth month. Y\$, S(n), L(n) and D(n) are loaded in lines 140 to 180.

A brief introduction follows, lines 190 to 280. The user is then able to choose the

type of menstrual cycle investigated, regular or irregular, lines 310 to 410. The program then branches to the procedures REGULARCYCLE or IRREGULARCYCLE at line 470.

If procedure IRREGULARCYCLE is called, the flag PR is set to zero. PR will become 1 if successful fertilisation resulting in pregnancy occurs. DC is also set to zero. This is the length of the current cycle and set to zero when the current procedure has been called in the middle of a program run. The procedure CYCLE is then called.

If the procedure REGULARCYCLE is called in line 470, the user is asked to specify the length of the regular cycle. This is required to be between 21 and 30 days (lines 740 to 800). Line 760 ensures that LC, the chosen length of the cycle, is a positive integer. Once again, as in the procedure IRREGULARCYCLE, the pregnancy flag, PR, is set to zero, as is the day of cycle counter, DC. The procedure CYCLE is then called.

The procedure CYCLE forms the core of the program and occupies lines 860 to 1510. It consists of two nested loops with control variables J and K. The loop with control variable J determines the month of the year by calling a substring of YS using the access vectors read previously from data (line 880). The K loop runs within the J loop and determines the day of the month. The K loop runs to the value of D(J), the number of days in month J. The screen is cleared with each new value of K in line 900, and the date printed in line 910.

At 28 day intervals, when the value of Y is an integral multiple of 28 (line 1400), a branch to the pregnancy test routine in lines 1410 to 1480 is made. If the pregnancy flag is 1, the day on which ovulation occurred is printed by branching to the procedure DAYOVN at line 1550, and the program exited.

The variable DC is incremented by 1 each new day in line 1080 until it becomes greater than LC, the length of the current cycle set either in lines 750 or 960, when its value is reset to 1 (line 110). The day of the current cycle is printed in line 1130.

DC is compared with CM, the length of the period of menstruation in line 1190, and if less than or equal to CM, a message indicating menstruation is printed on the screen. CM, the days of menstruation can take values between 3 and 5, set in line 1180.

The day and hour of ovulation are set in lines 1000 to 1040.

Each day the user is given an intercourse option in lines 1210 to 1320. If the option is chosen, the time of intercourse must be specified (lines 1260 to 1320). The hour of intercourse is determined in line 1300, and compared to ovulation time in line 1370, when an assessment of whether pregnancy is true or not is made. If pregnancy is true, flag PR is set to 1.

The program ends if PR is 1 and a pregnancy test option is chosen, or the user gets to the end of the year.

Using the Program

The program can be used to make a rough assessment of the efficiency of the rhythm method of contraception. It could also be used to investigate the days of the menstrual period when peak fertilities occur. Students could investigate this by restricting 'intercourse' to certain periods of regular cycles to determine when fertility is greatest.

To assess the efficiency of the rhythm method of contraception, the nature of the cycle should be known. Students could be asked to avoid 'getting the computer pregnant', while having to choose the intercourse option a certain minimum number of times per regular cycle, or per specified number of days with the irregular cycle option.

Finally, it must be stressed that this program cannot be used out of a laboratory context. Its use then, is perhaps much more appropriate for young adults in the sixth form and beyond.

SAFE Listing

```
10REM SAFE
20REM
30REM INVESTIGATION OF RYTHM METHOD OF CONTR
ACPTION
40REM
50REM NB JUST A SIMULATION
60REM
70*FX200,1
80DATA"JanuaryFebruaryMarchAprilMayJuneJulyAu
gustSeptemberOctoberNovemberDecember",1,7,31,8,8
,28,16,5,31,21,5,30,26,3,31,29,4,30,33,4,31,37,6
,31,43,9,30,52,7,31,59,8,30,67,8,31
90REM
100REM
110REM SET UP AND START
120REM
130MODE7
140DIMS(12),L(12),D(12)
150READY$
160FORJ=1TO12
170READS(J),L(J),D(J)
180NEXTJ
```



```

190VDU12
200VDU23,1,0;0;0;0;
210FORI%=8TO9
220PRINTTAB(9,I%);CHR$&8D;"SAFE"
230NEXTI%
240REM
250REM
260REM INTRODUCTION
270REM
280XX=INKEY(500):PRINT"'''''"This program is d
esigned to enable you""to determine the safe pe
riod for ""sexual intercourse!!"
290REM
300REM
310REM CHOOSE TYPE OF CYCLE
320REM
330XX=INKEY(650)
340MODE7
350VDU23,1,0;0;0;0;0;
360PRINT"'''''"Regular or irregular menstrual
cycle?"'''''"For REGULAR  cycle Type 1""For IR
REGULAR cycle Type 2""""
370REPEAT
380INPUT"Regular or irregular cycle",C
390C=ABS(INT(C))
400IFC>2ORC<1THENPRINT:PRINT:PRINT"YOU MUST TY
PE 1 OR 2":PRINT
410UNTILC=1ORC=2
420REM
430REM
440REM
450MODE1
460VDU23,1,0;0;0;0;0;
470IFC=2THENPROCIRREGULARCYCLE ELSE PROCREGULA
RCYCLE
480REM
490REM
500REM PROGRAM CONTINUATION OPTION
510REM
520PRINT
530INPUT"Continue",X$

```

```

540IF MID$(X$,1,1)="Y"THEN 330
550MODE7
560PRINTTAB(15,10)"END"
570*FX200,0
580END
590REM
600REM
610REM
620DEFPROCIRREGULARCYCLE
630PRINT""""Irregular Cycle""""
640PR=0
650DC=0
660XX=INKEY(500)
670PROCCYCLE
680ENDPROC
690REM
700REM
710REM
720DEFPROCREGULARCYCLE
730PRINT""""Regular Cycle""""
740REPEAT
750INPUT"Length of cycle (Days)",LC
760LC=ABS(INT(LC))
770PR=0
780DC=0
790IFLC>30ORLC<21THENPRINT:PRINT"Type an INTEG
ER in the range 21-30":PRINT
800UNTILLC<31ANDLC>20ANDLC=ABS(INT(LC))
810PROCCYCLE
820ENDPROC
830REM
840REM
850REM
860DEFPROCPCYCLE
870FORJ=1TO12
880M$=MID$(Y$,S(J),L(J))
890FORK=1TOD(J)
900VDU12
910PRINT""""M$;" ";K,
920Y=Y+1:REM 28 DAYS COUNTER
930REM

```

```

940REM DETERMINE LENGTH OF CURRENT CYCLE
950REM
960IFC=2 LC=22+RND(8)
970REM
980REM DETERMINE DAY OF OVULATION
990REM
1000DO=LC-16+RND(3)
1010REM
1020REM DETERMINE HOUR OF OVULATION
1030REM
1040HO=(DO-1)*24+RND(23)
1050REM
1060REM DETERMINE DAY OF CYCLE
1070REM
1080DC=DC+1
1090REM
1100IFPR=1THEN1130
1110IF DC>LC DC=1
1120IF DC>LC THEN960
1130PRINT"'"'"Day ";DC
1140PRINT"'"'"
1150REM
1160REM DETERMINE LENGTH OF MENSTRUAL PERIOD
1170REM
1180CM=3+INT(RND(1)*3)
1190IF DC<=CM THENCOLOUR1:PRINT"Menstruating":C
OLOUR3:PRINT:PRINT
1200REM
1210REM INTERCOURSE OPTION
1220REM
1230PRINT"Intercourse (without contraceptive)?"
1240INPUT"Y or N",A$
1250IFNID$(A$,1,1)="N"THEN1400
1260REPEAT
1270PRINT
1280INPUT"Time of intercourse (1-24hrs)",T
1290T=ABS(INT(T))
1300HI=(DC-1)*24+T
1310IF T>24 OR T<1 THENPRINT:PRINT"Type an INTE
GER in the range 1-24":PRINT
1320UNTILT=ABS(INT(T))ANDT<25ANDT>0

```

```

1330REM
1340REM
1350REM ASSESS PREGNANT OR NOT
1360REM
1370IF HI>=(HO-20) AND HI<=(HO+8) PR=1
1380REM
1390REM
1400IF Y/28 <> INT(Y/28) THEN NEXTK,J
1410PRINT"*****"
1420INPUT"Pregnancy Test (Urinary HCG)",P$
1430IF MID$(P$,1,1)<>"Y" THEN NEXTK,J
1440PRINT
1450IF PR=1 THEN PRINT"Pregnancy confirmed":XX=INK
EY(500):PROC DAYOVN:ENDPROC
1460PRINT"Test negative"
1470XX=INKEY(500)
1480NEXTK,J
1490*FX200,0
1500END
1510ENDPROC
1520REM
1530REM
1540REM
1550DEFPROC DAYOVN
1560VDU12
1570PRINT"*****"Day of ovulation"
1580PRINT
1590PRINTDO
1600ENDPROC

```

SAFE - Variables

Y\$	string containing mnemonics for all twelve months
S(n)	starting position for each month's mnemonic in Y\$
L(n)	length of substring corresponding to each month in Y\$
D(n)	number of days in month n
I%	control variable of loop generating enhanced title
XX	delay in program execution

C	takes value 1 for regular menstrual cycle or 2 for irregular menstrual cycle
X\$	response to program continuation option
PR	pregnancy flag: takes value 1 if pregnancy confirmed, 0 if not
DC	day of current menstrual cycle
LC	length (days) of current cycle
J	control variable for month loop
M\$	substring of Y\$ corresponding to name of month
K	control variable for days of the month
Y	counter, used to determine time to branch to optional pregnancy test every 28 days
DO	day of cycle on which ovulation occurs
HO	hour of ovulation
CM	length of current menstrual cycle
A\$	response to intercourse option
T	time of intercourse
HI	hour of intercourse
P\$	response to pregnancy test option

References

1. Bell, G.H., Davidson, J.N. and Scarborough, H., Textbook of Physiology and Biochemistry, 6th Ed., Livingstone, Edinburgh, 1965.
2. Maclean, N., Haemoglobin, Arnold, London, 1978.
3. Shone, J., 'The oxygen-haemoglobin dissociation curve-use of a microcomputer', SSR, 64, (1982), 273.
4. Monod, J., Changeux, J.-P., and Jacob, F., Allosteric proteins and cellular control systems, J. Mol. Biol., 6, (1963), 306-329.
5. Engle, P.C., Enzyme Kinetics, Chapman & Hall, London, 1977.

6. Nuffield Advanced Science. Biological Science. Study Guide, Penguin, Harmondsworth, 1971.
7. Hodgkin, A.L., The Conduction of the Nervous Impulse, Liverpool University Press, 1967.
8. Land, E.H., 'Experiments in colour vision', Scientific American, 200, (1959), 184-199.
9. Coll, J., The BBC Microcomputer User Guide, BBC, London, 1982.
10. Farris, E.J., Human Ovulation and Fertility, Pitman, London, 1956.

CHAPTER 2

NUMBER CRUNCHING

Introduction

Biologists make great use of statistics. Very often, the statistical procedures used are very time consuming and prone to error when done on a calculator, or by hand. This may be one reason why the use of statistical procedures in school biology is much rarer than it should be.

Four of the programs described in this chapter, REGANAL, CORREL, CHISQ and T-TEST are designed to take the drudgery out of using simple statistics in biology.

The other three programs, SPECDIV, ASSOC and AGINDEX, involve the use of three statistical procedures in ecology. They form a package of routines, together with the four mentioned above, used in the analysis of the results of ecological field work.

REGANAL

This is a least-squares regression analysis program.

Theoretical Background

Biologists are often confronted with two sets of data which may be related. The relationship between the variables may be a linear one of the form:

$$y = mx + a$$

where m is the slope of the graph of y against x and a is the intercept on the y axis. Such a relationship is approximately obtained when pulse rate in beats min^{-1} is plotted against breaths per minute.

Breaths min ⁻¹	Pulse, beats min ⁻¹
16	57
16	59
19	66
20	68
20	71
23	70
24	72
26	84
27	82
28	80
28	83
30	91
34	94
36	105
41	116
44	120

In actual fact, when this data is plotted, the points are scattered. We obtain a scattergram. The above data is taken from (1).

Even though the points are scattered, there is an obvious trend in the data. The more breaths per minute (x variable), the higher the pulse rate (y variable). What is more, the points appear to lie around a straight line, the so-called best fit straight line, which has an equation of the form above. This line can be fitted to a scattergram by minimising the sum of the squares of deviations from it. This fitted line we call a regression line. The line is defined by the two parameters, slope and intercept, which can be used to make predictions. By fitting the regression line, we can predict, for example, the pulse rate at a ventilation frequency of 47 breaths min⁻¹ or 14 breaths min⁻¹, or indeed, at any frequency that we care to choose.

Consider the following data. It is derived from a theoretical model of bacterial population growth. The bacterial population begins at generation 0 with one cell. Subsequently, cells divide simultaneously and successively, each division being equally spaced in time. Such a simple model is typically used to introduce the ideas of population growth.

Generation (time)	Number of Cells in Population	Log ₁₀ Number of Cells in Population
0	1	0
1	2	0.69
2	4	1.39
3	8	2.08
4	16	2.77
5	32	3.47
.	.	.
.	.	.
10	1024	6.93

If the data is plotted linearly, number of cells against time, a typical exponential curve results. A linear regression line cannot be justifiably fitted to this data. However, if the cell numbers are transformed by taking natural logarithms, a plot of log number of cells against time is linear. The equation of the relationship is:

$$\log_e N(t) = \log_e N_0 + mt$$

where $N(t)$ is the number of cells present at time t , N_0 the number of cells present at $t=0$. In the present case, N_0 is 1, so $\log_e N_0 = 0$, and the intercept is the origin.

Thus, a logarithm transform of data may produce a linear relationship which enables extrapolation. In the above case, we could predict the number of cells present, given the conditions of the model, at any time, t , in the future.

Such a model system is obviously very unrealistic. Not only do populations not grow indefinitely, but also the numbers of cells present at any time t will not be predictable on the assumptions of such a simplistic model. Nevertheless, the scattergram of points derived from a plot of say, number of bacterial cells in a culture against time, at least for small values of t , will be scattered around a linear trend if the natural logs of numbers are used. Hence a regression analysis to determine intercept and slope becomes a useful exercise.

Logarithmic transformations of one data set in a pair of related data sets may not 'straighten out' the scattergram, such that a predictive regression line cannot be fitted to the data. In those circumstances, logarithmic transformations of both data sets may work (see (2) for example).

In this case, the equation of the best fit regression line will be:

$$\log_e y = m \log_e x + a$$

where m is the slope of the best fit straight line and a is $\log_e y$ at $x = 0$, i.e. $\log_e y(0)$.

Taking anti logs to the base e of both sides, we get

$$y = y(0)x^m$$

This type of relationship is quite common in biological systems. Amongst other things, it is the simple heterogony relationship (3).

The Computer Program

The program REGANAL allows the user to undertake a detailed regression analysis of two sets of input data to determine, if possible, the equation of a straight line to describe the relationship between the data. Where appropriate, the program allows the natural logarithmic transformation of one or both sets of data. It plots the appropriate scattergram, fits the regression line to the scattergram and prints out the numeric values of slope and intercept.

Estimates of m and a are obtained from:

$$(i) \hat{m} = \left(\sum_{i=1}^n x_i y_i - n \bar{x} \bar{y} \right) / \left(\sum_{i=1}^n x_i^2 - n \bar{x}^2 \right) \quad (ii) \hat{a} = \bar{y} - \hat{m} \bar{x}$$

where \hat{m} is the estimate of slope, n the number of pairs of data, \bar{x} the mean of the x variables, \bar{y} the y mean and \hat{a} the estimate of the intercept (4).

x_i or y_i refer to linear values of x and y, or their logarithmic transforms as appropriate.

Program Discription

The program begins by giving the user some introductory information, lines 50 to 210. The logarithms of numbers zero and less are undefined, and so logarithmic transformations of data including zeros or negative numbers will produce an error.

Data is input using procedure **input.data** called in line 240. This procedure, lines 590 to 690, loads the input data into the two-dimensional arrays, X(n,m) and Y(n,m). The value of n represents the number of input data pairs, and this data is loaded into the first rows of the arrays.

The user is given the option of transforming the data logarithmically in lines 260 to 270, calling the procedure **logtransform**. If this latter procedure is called, the second rows of the X and Y data arrays are loaded with the natural logarithms of the corresponding first rows.

Note: if any of the elements of the first rows contain zero or negative elements, an error will occur which will terminate program execution. You might like to modify the program so that it queries or rejects such data.

The user is then given the choice of a regression analysis of any of the available X and Y data. The procedure **log.or.linear** called in line 290 and running from line 820 to 970, enables the user to choose linear or log data for each variable, X and Y.

The procedure **regression** called in line 300, running from lines 1000 to 1120 calculates the slope and intercept of the best fit straight line that can be fitted to the chosen X and Y variables. This result is output in the procedure **printresult**, lines 1150 to 1240, called in line 310. Program execution is delayed in line 420, where a loop runs until the spacebar is pressed, the ASCII code for space being 32.

The procedure **find.scalefactors** determines the scalings required to fit the X and Y data to the axis space chosen for the scattergram. This is done by finding, for both sets of variables, the maximum and minimum value, and hence the range of the data. Then this range is used to find the appropriate scaling in lines 1390 and 1400.

Lines 340 to 360 set a graphics mode and call the procedure **axes**. This procedure, lines 1440 to 1580 draws the axes for the scattergram. The origin for the axes is set depending the minimum values of the X and Y variables. The axes are labelled, depending on the values of datatypeX and datatypeY set in the procedure **log.or.linear**, and using VDU5 to print text at the graphics cursor.

The procedure **plotdata** called in line 370 plots the points of the scattergram. This procedure, lines 1610 to 1680 plots the points in actual colour 9, flashing red and cyan. Logical colour 1 is changed into actual colour 9 using VDU19 at line 1640.

After a delay, the best-fit regression line is fitted by the procedure **regline** called in line 390. This procedure, lines 1710 to 1780, fits the regression line by first of all moving to the centroid of the scattergram. This point is defined by X coordinate meanX and Y coordinate meanY. The best-fit straight line passes through this point (4). A line is then drawn in yellow, logical colour 2, from this point, to a point defined by minX and the corresponding value of Y determined by the regression equation using the computed values of slope and intercept, line 1740. The graphics cursor moves back to the centroid and then draws a similar line to a point defined by maxX and its corresponding calculated Y coordinate. Finally, the line is completed, if necessary, by drawing a line to the point on the Y axis defined by the intercept (lines 1750 to 1770). The lines are drawn using GCOL1 which ORs the line with the graphics colour already present. Thus, if the line passes through a previously plotted point on the scattergram, the point becomes white.

When the plotting is complete, the user can use the stored data again to plot a different scattergram and regression line having first of all reset the variables needed to calculate intercept and slope in the procedure **reset** (lines 1810 to 1860) If this is not required, new data will be accepted by the program, or execution can be terminated (lines 450 to 560).

Using the Program

REGANAL can be used to find a mathematical relationship between two variables. By choosing the appropriate combination of linear or logarithmic scales, the user can fit a straight line to a scattergram of the data. This can then be used to write down a linear equation relating the two variables. Of course, the best-fit straight line may not be a particularly good fit. It may be that a linear regression model is not appropriate for the data under consideration. All is not lost, but the problem becomes a little less tractable. Students of A level need not worry, but more advanced courses may require a further incursion into curvilinear regression using orthogonal polynomials; reference (5) is a good starting point.

Consider the following data taken from Sand's (6) book of problems in animal physiology. It concerns the relationship between size and metabolic rate in seven mammalian species.

Species	Body mass (kg)	Metabolic rate ($\text{mm}^3\text{O}_2\text{g}^{-1}\text{hr}^{-1}$)
Mouse	0.025	1580
Rat	0.226	872
Rabbit	2.2	466
Dog	11.7	318
Man	70.0	202
Horse	700.0	106
Elephant	3800.0	67

REGANAL can be used to derive an equation to relate these two variables. The first stage is to RUN the program and input the seven pairs of data.

On the message Data pair X(1),Y(1)? we input .025,1580 return, and so on. A brief consideration of the data suggests that the two variables are not linearly related, and so logarithmic transformation seems needed.

A linear-linear data plot confirms this suspicion. Indeed, most data points in a linear-linear plot lie so close to the vertical axis that they are indistinguishable from it. It turns out that a log-log plot gives a reasonable straight line fit. The regression line for this plot has slope -0.263 (3 decimal places) and intercept 6.388 (3 decimal places). Thus the equation of the regression line is:

$$\log_e y = 6.388 - 0.263 \log_e x$$

where y is metabolic rate and x is body mass. Taking natural anti logs of both sides of this expression,

$$y = 594.666x^{-0.263}.$$

This equation relates metabolic rate in the units given to mass in kilograms for mammals.

REGANAL Listing

```

10REM REGANAL
20REM
30*FX200,1
40DIMX(2,50),Y(2,50)
50MODE7
60VDU23,1,0;0;0;0;
70REM
80FORI%=8TO9

```

```

90PRINTTAB(9,1%);CHR$&8D;"REGANAL"
100NEXTI%
110delay=INKEY(250)
120REM
130CLS
140REM
150REM INTRODUCTION
160REM
170PRINTTAB(0,10);"This program allows you to
perform""a regression analysis on data."
180delay=INKEY(500)
190CLS
200PRINTTAB(0,10);"DO NOT TRY TO LOG TRANSFORM
ZERO""OR NEGATIVE DATA."
210delay=INKEY(500)
220REM
230REM
240PROCinputdata
250CLS
260PRINTTAB(0,10);:INPUT"Do you want log trans
forms",A$
270IFLEFT$(A$,1)="Y" OR LEFT$(A$,1)="y" THEN P
ROClogtransform
280IF LEFT$(A$,1)<>"Y" AND LEFT$(A$,1)<>"y" TH
EN datatypeX=1:datatypeY=1:GOTO300
290PROClog_or_linear
300PROCregression
310PROCprintresult
320REPEATUNTILGET=32
330PROCfind_scalefactors
340MODE1
350VDU23,1,0;0;0;0;0;
360PROCaxes
370PROCplotdata
380delay=INKEY(500)
390PROCregline
400delay=INKEY(250)
410PRINTTAB(3,1);"Press spacebar to continue"
420REPEATUNTILGET=32
430CLS
440PRINTTAB(0,10);

```

```

450IF LEFT$(A$,1)<>"Y" AND LEFT$(A$,1)<>"y" TH
EN 490
460INPUT"Do you wish to use the data again",an
swer$
470IF LEFT$(answer$,1)="Y" OR LEFT$(answer$,1)
="y" THEN PROCreset:GOTO290
480PRINT:PRINT
490INPUT"Do you wish to enter more data",answe
r$
500IF LEFT$(answer$,1)<>"Y" AND LEFT$(answer$,
1)<>"y" THEN 530
510PROCreset
520GOTO50
530CLS
540PRINTTAB(10,10);"END"
550*FX200,0
560END
570REM
580REM
590DEFPROCinputdata
600LOCAL I
610CLS
620PRINTTAB(0,10);
630INPUT"How many data pairs",N
640FOR I=1TON
650CLS
660PRINTTAB(0,10);
670PRINT"Data pair X(";I;"),Y(";I;")";
680INPUTX(1,I),Y(1,I)
690NEXT I
700ENDPROC
710REM
720REM
730DEFPROClogtransform
740LOCAL I
750FOR I=1TON
760X(2,I)=LN(X(1,I))
770Y(2,I)=LN(Y(1,I))
780NEXT I
790ENDPROC
800REM

```

```

810REM
820DEFPROClog_or_linear
830REPEAT
840CLS
850PRINTTAB(0,10);
860PRINT"X variable""Type 1 for linear""Type
2 for log""
870INPUT"X variable",datatypeX
880IFdatatypeX<>1 AND datatypeX<>2 THEN 830
890UNTIL datatypeX=1 OR datatypeX=2
900REPEAT
910CLS
920PRINTTAB(0,10);
930PRINT"Y variable""Type 1 for linear""Type
2 for log""
940INPUT"Y variable",datatypeY
950IFdatatypeY<>1 AND datatypeY<>2 THEN 900
960UNTIL datatypeY=1 OR datatypeY=2
970ENDPROC
980REM
990REM
1000DEFPROCregression
1010LOCAL I
1020FOR I=1 TO N
1030sumX=sumX+X(datatypeX,I)
1040sumY=sumY+Y(datatypeY,I)
1050sumXY=sumXY+X(datatypeX,I)*Y(datatypeY,I)
1060sumX2=sumX2+X(datatypeX,I)^2
1070NEXT I
1080meanX=sumX/N
1090meanY=sumY/N
1100slope=(sumXY-(N*meanX*meanY))/(sumX2-(N*meanX^2))
1110intercept=meanY-slope*meanX
1120ENDPROC
1130REM
1140REM
1150DEFPROCprintresult
1160CLS
1170PRINTTAB(0,10);

```

```

1180IF datatypeX=1 PRINT "X linear" ELSE PRINT
"X logarithmic"
1190IF datatypeY=1 PRINT "Y linear" ELSE PRINT
"Y logarithmic"
1200PRINTTAB(0,15);
1210PRINT"Slope= ";slope
1220PRINT"Intercept= ";intercept
1230PRINTTAB(0,22);"Press spacebar to continue"
1240ENDPROC
1250REM
1260REM
1270DEFPROCfind_scalefactors
1280LOCAL I
1290minX=1E6
1300minY=1E6
1310maxX=1E-6
1320maxY=1E-6
1330FOR I=1TON
1340IF minX>X(datatypeX,I) THEN minX=X(datatype
X,I)
1350IF maxX<X(datatypeX,I) THEN maxX=X(datatype
X,I)
1360IF minY>Y(datatypeY,I) THEN minY=Y(datatype
Y,I)
1370IF maxY<Y(datatypeY,I) THEN maxY=Y(datatype
Y,I)
1380NEXT I
1390IF minX>=0 THEN scaleX=500/maxX ELSE scaleX
=500/(maxX-minX)
1400IF minY>=0 THEN scaleY=500/maxY ELSE scaleY
=500/(maxY-minY)
1410ENDPROC
1420REM
1430REM
1440DEFPROCaxes
1450IF minX<0 THEN zeroX=ABS(minX)*scaleX ELSE
zeroX=0
1460IF minY<0 THEN zeroY=ABS(minY)*scaleY ELSE
zeroY=0
1470VDU29,zeroX+50;zeroY+50;
1480MOVE0,500

```



```

1490DRAW0,-500
1500MOVE-500,0
1510DRAW500,0
1520VDU5
1530MOVE0,510
1540IF datatypeY=1 PRINT "Y" ELSE PRINT "log Y"
1550MOVE510,0
1560IF datatypeX=1 PRINT "X" ELSE PRINT "log X"
1570VDU4
1580ENDPROC
1590REM
1600REM
1610DEFPROCplotdata
1620LOCAL I
1630GCOL1,1
1640VDU19,1,9;0;
1650FORI=1TON
1660PLOT69,X(datatypeX,I)*scaleX,Y(datatypeY,I)
*scaleY
1670NEXT I
1680ENDPROC
1690REM
1700REM
1710DEFPROCregline
1720GCOL1,2
1730MOVEmeanX*scaleX,meanY*scaleY
1740DRAWminX*scaleX,(minX*slope+intercept)*scal
eY
1750MOVEmeanX*scaleX,meanY*scaleY
1760DRAWmaxX*scaleX,(maxX*slope+intercept)*scal
eY
1770DRAW0,intercept*scaleY
1780ENDPROC
1790REM
1800REM
1810DEFPROCreset
1820sumX=0
1830sumY=0
1840sumXY=0
1850sumX2=0
1860ENDPROC

```

REGANAL - Variables

I%	loop control variable for enhanced character heading
delay	program execution delay
A\$	user response to log transform option
datatypeX	linear or logarithmic values of X-1 for linear, 2 for logarithmic
datatypeY	see above
answer\$	user response to more data option AND repeat using same data option
I	loop control variable
N	number of pairs of data
X(n,m)	X data matrix
Y(n,m)	Y data matrix
sumX	sum of all X data
sumY	sum of all Y data
sumXY	sum found by adding product of corresponding X and Y values from respective matrices
sumX2	sum of squares of all X data
meanX	average of X data
meanY	average of Y data
slope	slope of best-fit least squares regression line
intercept	Y intercept of least squares regression line
minX	smallest X datum
minY	smallest Y datum
maxX	largest X datum
maxY	largest Y datum

scaleX	scalefactor for X data, used in scattergram plotting
scaleY	as above for Y data
zeroX,zeroY	screen coordinates of scattergram origin

CORREL

This program determines a product-moment correlation coefficient.

Theoretical Background

Correlation analysis is used to determine the extent to which two variables are associated or related (7). A correlation coefficient can be used to assess the degree of association. If the two variables are normally distributed, Pearson's r value is employed.

The range of values of r is from -1 to 1. A value of -1 or 1 implies perfect correlation between the two variables under consideration. A value of $r = 0$ implies no relationship. Usually, values of r lie in the interval $[-1, 0]$. As a 'statistical rule of thumb', if a correlation analysis is done using thirty data pairs or more, and r is equal to or greater than 0.7, or less than or equal to -0.7, the variables under consideration are significantly related.

The meaning of r values can be visualised using scattergrams. If the points in a scattergram lie on a straight line, r is 1 or -1 depending on the slope of the line. If the points are randomly distributed in the x, y axis space, no best-fit straight line is possible, and r is zero. One can imagine that as randomly distributed points arrange themselves increasingly closer to a straight line, the value of r becomes increasingly closer to 1.

The Computer Program

The value of r is calculated using the following expression:

$$r = \frac{\sum_{i=1}^n x_i y_i - \left[\left(\sum_{i=1}^n x_i \right) \left(\sum_{i=1}^n y_i \right) \right] / n}{\sqrt{\left[\sum_{i=1}^n x_i^2 - \frac{\left(\sum_{i=1}^n x_i \right)^2}{n} \right] \left[\sum_{i=1}^n y_i^2 - \frac{\left(\sum_{i=1}^n y_i \right)^2}{n} \right]}}$$

A detailed discussion of this, and other measures of correlation can be found in Haber and Runyon (7).

In the above expression, x_i and y_i are the i th data points, and n is the number of pairs of data.

Program Description

A very simple program, CORREL begins with a brief statement of the purpose of the routine in lines 50 to 120. Data is then accepted in pairs (X,Y) in lines 160 to 300. At the same time, the values of the sums used in the expression for r are accumulated. The value of r is then calculated and printed on the screen in lines 340-410.

Using the Program

The idea of correlation is important in biology and social biology. It is used quite extensively in Nuffield A level (8). This program provides a convenient way of calculating values of r , a process which otherwise would take an inordinate length of time. Data from which r values are determined could also be subjected to regression analysis using REGANAL, described in this chapter. The reader could use the heart rate/ventilation frequency data given in the account of REGANAL to determine a value for r and relate this to the output produced by the same data used in REGANAL.

CORREL Listing

```
10REM CORREL
20REM
30REM CALCULATION OF PEARSON'S  $r$  FOR NORMALLY
DISTRIBUTED VARIABLES
40REM
50*FX200,1
60MODE7
70VDU23,1,0;0;0;0;0;
80PRINTTAB(8,10);"CORREL"
90delay=INKEY(500)
100CLS
110PRINTTAB(0,10);"This program allows the cal
culation""of the correlation coefficient, ""P
earson's  $r$ ."
120delay=INKEY(500)
130REM
140REM INPUT DATA
150REM
160CLS
```

```

170VDU23,1,1;0;0;0;
180PRINTTAB(0,10)
190INPUT"How many pairs of data",datapairs
200REM
210FORI=1TODatapairs
220CLS
230PRINTTAB(0,10);"Datapair (X,Y) ";I;" ";;INP
UTX,Y
240REM
250sumX=sumX+X
260sumY=sumY+Y
270sumX2=sumX2+X^2
280sumY2=sumY2+Y^2
290sumXY=sumXY+X*Y
300NEXTI
310REM
320REM CALCULATE r
330REM
340r=(sumXY-(sumX*sumY/datapairs))/SQR((sumX2-
(sumX)^2/datapairs)*(sumY2-(sumY)^2/datapairs))
350REM
360REM PRINT RESULT
370REM
380CLS
390PRINTTAB(0,10);"Pearson's r = ";r
400*FX200,0
410PRINTTAB(10,20);"DONE"

```

CORREL - Variables

delay	delay in program execution
datapairs	number of pairs of data, X and Y
sumX	total of all X variables
sumY	total of all Y variables
sumX2	sum of squares of all X variables
sumY2	sum of squares of all Y variables

sumXY

sum of products of X's with corresponding Y's

l

loop control variable

r

calculated value of Pearson's r, product moment correlation coefficient

SPECDIV

This involves the calculation of a simple species diversity index.

Biological Background

Ecological studies in the field often generate a large amount of data which is quantitative. Biologists at all stages of their education, may be required to draw conclusions from this data which are subjectively qualitative.

For example, on sheltered rocky shores, which are dominated by fucoid algae, there is found a wide range of browsing gastropods. When compared to an exposed shore, where fucoid algae are much rarer, this range of gastropods is reduced. There appears to be a reduced gastropod species diversity. This conclusion need not be arrived at merely subjectively. Objective assessments of species diversity exist (5, 9).

One of the most widely used species diversity indices is the Shannon-Weaver Index, H' . This is a measure of the uncertainty involved in trying to guess the species of the next individual in a random sample of organisms from a habitat.

H' is given by

$$H' = - \sum_{i=1}^s p_i \log_e p_i$$

This is the form of the expression given by Poole (5); it gives a biased estimate of H' . The mean or expected value of H' , called $E(H')$ can be obtained from the series,

$$E(H') = \left[-\sum p_i \log_e p_i \right] - \left[\frac{s-1}{2N} \right] + \left[\frac{1-\sum p_i^{-1}}{12N^2} \right] + \dots$$

derived by Hutcheson (10). In these expressions, s is the number of species in the sample, p_i the proportion of the total sample consisting of the i th species and N is the total number of individuals in the sample.

The units of H' are natural bels (11), and the variance of H' is given by the series,

$$\text{Var}(H') = \left[\frac{\sum p_i \log_e^2 p_i - \left(\sum p_i \log_e p_i \right)^2}{N} \right] + \left[\frac{s-1}{2N^2} \right] + \dots$$

Usually the first two terms for $E(H')$ and the first term for $\text{var}(H')$ are sufficient. Despite this, the calculations of $E(H')$ and $\text{var}(H')$ can be rather tedious and time-consuming when done by hand, and the sample is large.

The Computer Program

The program SPECDIV enables students to derive values for $E(H')$ and $\text{var}(H')$ with a minimum of delay and minimum error. The program is very simple. It begins with a brief introduction, lines 30 to 150. This can be speeded up by pressing the spacebar during the delay time set up by INKEY statements at lines 120 and 150.

The program requires as input the number of different species in the sample, lines 190 to 220. A one-dimensional array is then set up into which is loaded the number of individuals of each species, lines 230 to 290.

Values of H' , $E(H')$ and $\text{var}(H')$ are then determined at lines 340 to 370, 420 and 460 to 510 respectively. Results of the calculations are output in lines 560 to 620 and program execution is then terminated.

Using the Program

The program can be used to calculate species diversities in biological samples. As an example, consider the following. In June 1982, the author and some sixth-form biology students visited Aberystwyth on a marine ecology field course. Some of the data collected appears below. We were interested in the gastropod species diversities at two sites, a sheltered shore and an exposed shore. The data were collected by belt transect, 1/4m quadrat samples being taken every 5m from EHWN to 80m downshore. The data from College Rocks, Aberystwyth, a sheltered rocky shore, were collected from 09.00 to 12.00 hrs on 28 June, and that from an exposed shore at Borth from 09.00 to 11.30 hrs on June 29, 1982. College Rocks is a sheltered shore, Borth semi-exposed according to the exposure scale in Lewis, pp. 288-290 (12). College Rocks is dominated by *Ascophyllum nodosum*, the shore at Borth by a midshore belt of *Balanus balandoides* and *Mytilus edulis*. Due to the absence of large fucoids at Borth, we hypothesised a lower species diversity for grazing gastropods at that location.

Species	College Rocks	Borth
<i>Patella vulgata</i>	55	100
<i>Littorina saxatillis</i>		222
<i>Littorina littorea</i>	85	679
<i>Littorina obtusata</i>	48	
<i>Gibbula umbilicalis</i>	30	33
<i>Gibbula cineraria</i>	5	
Number of species(s)	5	4
Number of individuals	223 (N_1)	1034 (N_2)
Biased estimate of H'	1.399 (H'_1)	0.942 (H'_2)
$E(H')$	1.390	0.941
$\text{var}(H')$	1.182×10^{-3}	6.23×10^{-4}

The numbers are the total numbers of individuals of each species of snail found.

The results of the calculations of species diversity indices are included in the above table. On inspection, the hypothesis that the more dense fucoids of College Rocks should support a higher gastropod species diversity than Borth, seems upheld.

The two values of $E(H')$ can be tested for significance of difference using a t-test, where t is determined as follows:

$$t = \frac{E(H'_1) - E(H'_2)}{(\text{Var}(H'_1) + \text{Var}(H'_2))^{1/2}}$$

The number of degrees of freedom is given by:

$$df = \frac{(\text{Var}(H'_1) + \text{Var}(H'_2))^2}{\text{Var}(H'_1)^2/N_1 + \text{Var}(H'_2)^2/N_2} \quad (2)$$

Using these expressions, t for the data in the table is 10.557 with 491 degrees of freedom. From a table of t values (7), it can be seen that the probability of the observed difference is very much less than .0005, and the null hypothesis that $H'_1 = H'_2$ can be rejected at this level. H' for the snails of the sheltered shore is significantly greater than that for the (semi-) exposed shore.

The reason for this is likely to be due to the reduced number of grazing niches available to snails on shores with a low algal standing crop.

SPECDIV Listing

```
10REM SPECDIV
20REM SHANNON-WEAVER DIVERSITY INDEX
30*FX200,1
40MODE7
50VDU23,1,0;0;0;0;
60REM
70REM INTRODUCTION
80REM
90FOR I%=8TO9.
100PRINTTAB(9,I%);CHR$(8D);"SPECDIV"
110NEXT I%
120delay=INKEY(500)
```

```

130CLS
140PRINTTAB(0,10);"This program allows you to
calculate""the Shannon-Weaver species diversit
y""index."
150delay=INKEY(500)
160REM
170REM INPUT DATA
180REM
190VDU23,1,1;0;0;0;
200CLS
210PRINTTAB(0,10);
220INPUT"How many different species",number_of
_species
230DIMnumber(number_of_species)
240REM
250FORI=1TONumber_of_species
260CLS
270PRINTTAB(0,10);"How many individuals of spe
cies ";I;" ";:INPUTnumber(I)
280N=N+number(I)
290NEXTI
300REM
310REM
320REM CALCULATE H'
330REM
340FORI=1TONumber_of_species
350P=number(I)/N
360H=H-P*LN(P)
370NEXTI
380REM
390REM
400REM CALCULATE E(H')
410REM
420EH=H-((number_of_species-1)/(2*N))
430REM
440REM CALCULATE VARIANCE OF H'
450REM
460FORI=1TONumber_of_species
470P=number(I)/N
480sum1=sum1+P*(LN(P))^2
490NEXTI

```

```

500REM
510varH=(sum1-H^2)/N+((number_of_species-1)/(2
*N^2))
520REM
530REM
540REM PRINT RESULTS
550REM
560CLS
570VDU23,1,0;0;0;0;0;
580PRINTTAB(0,5);"Biased estimate of H' ";H
590PRINT'"Expected value, E(H') ";EH
600PRINT'"Variance of H'          ";varH
610PRINT'"Number of species      ";number_of_sp
ecies
620PRINT'"Number of individuals ";N
630PRINTTAB(10,20);"DONE"
640*FX200,0
650END

```

SPECDIV - Variables

I%	loop control variable for enhanced characters
delay	delay in program execution
number-of-species	number of different species in sample
number(n)	number of individuals of the nth species
N	total number of all individuals in sample
P	frequency of nth species in total sample
H	biased estimate of Shannon-Weaver diversity index, H'
I	loop control variable
EH	expected value of H'
sum1	cumulative total of $P \times \log_e^2 P$
varH	variance of estimate of H'

CHISO

Calculation of the χ^2 statistic is the subject of this program.

Biological Background

One of Mendel's classic investigations into the laws of inheritance concerned pea plants. He crossed pure-breeding tall plants with pure breeding dwarf ones. The peas resulting from this cross grew into tall plants. When these F_1 plants were allowed to self-fertilise and produce seed, and the seeds planted, they produced a mixture of tall and dwarf plants. Mendel predicted that there should be 3 tall plants to every dwarf one in this F_1 generation. He had an expectation, based on theory, which could be tested by making an observation. By comparing the number of tall and dwarf plants produced in the F_1 intercross with the expected numbers, the validity of the assumptions used to make the expectation could be assessed. Mendel did not do this rigorously. He did not use the χ^2 test.

If, for the sake of argument, in the above investigation, the selfing of the F_1 produced 400 viable seeds which germinated and became plants recognisably tall or dwarf, we would expect, using Mendelian laws, 300 to be tall and 100 dwarf. Perhaps we might obtain 287 tall and 113 dwarf. The question becomes 'is the difference between what is observed and what is expected so great that the theory upon which the expectation is based can be rejected?'

To answer the question, χ^2 is calculated as follows:

$$\chi^2 = \sum_{i=1}^n (O_i - E_i)^2 / E_i$$

where n is the number of categories, 2 in this case, E_i and O_i the number of expected and observed plants respectively in each category.

For our present case, the data could be set out as follows:

	Category		Total
	Tall	Dwarf	
No. expected	300	100	400
No. observed	287	113	400

The calculation of χ^2 in this case presents no difficulty, and turns out to be 2.253 (3 decimal places). This value of χ^2 can be converted into a probability that the

observed frequencies are obtained when we expect the expected using a χ^2 table (7,8). To use such a table, we need to know the number of degrees of freedom for our calculation. This is always one less than the number of categories, so it is one in this case. The obtained value of χ^2 with one degree of freedom gives a probability between 0.25 and 0.1. This means that we will get frequencies of 287 and 113, when we expect to get 300 and 100, at least 10% of the time purely by chance. The observed frequencies are acceptably different from the expected, and can be used to support the theory upon which the expectation was based.

However, had the value of χ^2 obtained given a probability of 0.05 or less, this is taken to suggest that the observed frequencies are significantly different from those expected and this would cast doubt on the validity of the hypothesis used to generate the expectation.

The Computer Program

The program CHISQ is used to calculate χ^2 . In the example above, with only two categories, the calculation of χ^2 is simple. However, with large numbers of categories, the computations become very tedious if done by hand.

Program Description

CHISQ starts with a title page, lines 50 to 120. The user then inputs the number of categories required, then the observed and expected frequencies for each category. χ^2 is then determined. Data input and calculation are carried out in the procedure **input.data**, lines 270 to 490, called in line 160. During this procedure a check is made on the totals of observed and expected data, line 460. If these sums are disparate, the procedure **data.error** is called. This procedure, lines 600 to 680 prints a diagnostic error message, sets accumulators to zero and returns control to the procedure **input.data** from where (line 480), the procedure **input.data** is called recursively.

Finally, the results of the calculation are printed to the screen in the procedure **print.result**, lines 520 to 570, called in line 170. The program then terminates.

CHISQ Listing

```
10REM CHISQ
20REM
30REM Chi-squared test
40REM
50*FX200,1
60MODE7
```

```

70VDU23,1,0;0;0;0;
80REM
90REM Introduction
100REM
110PRINTTAB(8,10);"CHI-SQUARED TEST"
120PROCcontinue
130REM
140REM Input data and calculate chi-squared
150REM
160PROCinput_data
170PROCprint_result
180PRINTTAB(0,20)
190*FX200,0
200END
210DEFPROCcontinue
220PRINTTAB(3,20);"Press spacebar to continue"
230REPEATUNTILGET=32
240ENDPROC
250REM
260REM
270DEFPROCinput_data
280VDU23,1,1;0;0;0;
290REPEAT
300CLS
310PRINTTAB(0,10);:INPUT"How many categories"
,categories
320UNTIL categories=ABS(INT(categories))
330REM
340CLS
350FORI=1TOcategories
360REPEAT
370CLS
380PRINTTAB(0,10);"Category ";I
390PRINTTAB(0,14);:INPUT"Obs. & expected frequ
encies O,E ",O,E
400UNTIL O=ABS(O) AND E=ABS(E)
410REM
420sumO=sumO+O
430sumE=sumE+E
440chi_squared=chi_squared+(O-E)^2/E
450NEXTI

```

```

460IF sumO=sumE THEN 490
470PROCdata_error
480PROCinput_data
490ENDPROC
500REM
510REM
520DEFPROCprint_result
530CLS
540PRINTTAB(0,10);"Chi-squared is ",chi_square
d
550PRINTTAB(0,12);"with ";categories-1;" degree
e";
560IF categories-1<>1 PRINT "s of freedom." EL
SE PRINT" of freedom."
570ENDPROC
580REM
590REM
600DEFPROCdata_error
610VDU23,1,0;0;0;0;
620CLS
630PRINTTAB(3,10);"Data error-try again"
640chi_squared=0
650sumO=0
660sumE=0
670PROCcontinue
680ENDPROC

```

CHISQ - Variables

categories	number of categories
I	loop control variable
O	observed frequency
E	expected frequency
sumO	sum of observed data
sumE	sum of expected data
chi_squared	variable eventually taking the value of χ^2

ASSOC

This program calculates a measurement of the degree of association between two species in a habitat.

Biological and Theoretical Background

In any habitat, not all species present are distributed independently. Some species will show positive association. For example, symbiotic species such as beans and the bacterium *Rhizobium* will show such association, as will *Ascophyllum nodosum* with its epiphyte *Polysiphonia* on rocky shores. Conversely, species which undergo competition for the same resources, or species actively inhibited by another will show negative association.

One method which can be used to detect such associations is to employ a χ^2 test for a 2×2 contingency table (5). Strictly speaking, this technique only detects associations, it does not assess the degree of association between two species. Also, it was originally designed to be applied to discrete units, such as whole trees or ponds, which are taken as sampling units. Nevertheless, if these facts are recognised, the technique can still be applied with success to continuous habitats, such as lawns, fields, rocky shores, etc., sampled using quadrats. A discussion of tests and measures of association can be found in (5) and (9).

Basically, the method consists of recording the number of sampling units which contain species A and species B, species A and not B, B but not A and neither. This data is entered into a 2×2 contingency table.

	Species B present	Species B absent	
Species A present	a	b	a + b
Species A absent	c	d	c + d
	a + c	b + d	n

Assuming independent distribution of species A and B, then the expected number of sampling units in each class is known. For example, for class b, the expected number of sampling units is $(a + b)(b + d)/n$. Where n is the total number of sampling units. Independent distribution is the null hypothesis tested using χ^2 .

If there is a significant difference between the expected numbers of sampling units in each class and the observed numbers, then the null hypothesis of independent distribution can be rejected. If the null hypothesis is rejected, then association is implicated. The nature of the association, positive or negative can be assessed from the 2×2 contingency table. For positive association, it would be expected that $(a+d) > (b+c)$; the reverse might be the case for negative association.

χ^2 for a 2 x 2 contingency table is given by the expression:

$$\chi^2 = ((ad-bc)^2n)/((a+b)(c+d)(a+c)(b+d))$$

χ^2 can be converted into a probability value using a chi-squared table with one degree of freedom. As a rule of thumb, if the value of χ^2 is less than 3.84, then there is no support for the null hypothesis, and association is indicated.

The expression for χ^2 used above applies when n is greater than 30. If n is less than 30, an alternative expression, the Yates correction form of χ^2 is used (5).

This expression is:

$$\chi^2 = ((|ad-bc| - n/2)^2n)/((a+b)(c+d)(a+c)(b+d))$$

where $|ad-bc|$ is the modulus of $ad-bc$.

The Computer Program

The program ASSOC enables the user to input raw data from quadrats and then calculates the value of chi-squared. If necessary, Yates' correction is employed. By the use of the VDU5 statement, data is input to a table like the one above, directly.

Program Description

The program begins with a heading page, lines 80 to 130. Two pages of messages are then printed, program continuation being effected by the procedure continue, lines 140 to 190, 580 to 610.

The program then requests the user to key in the names of the two species, association between which is suspected, lines 240 to 330. Each species name must be no more than ten characters long. Species names are checked by calling the procedure **check.species.name** running from line 640 to 710. This procedure is called in lines 270 and 310. It checks the length of the input species name (converted to A\$ as the procedure's formal parameter) by using the function LEN. If the value of LEN(A\$) is more than ten, a diagnostic message is printed (line 680), and a flag, flagsp set to 1. When control returns to the main body of the program on ENDPROC, the value of flagsp is checked in a conditional branch statement (lines 290,330). If flagsp has the value 1, the program branches to take the species name again. If flagsp has the value zero, the program continues.

When species names have been entered correctly, the procedure **quadrat.data** is called in line 380. This procedure running from line 740 to 1190 takes the quantitative data needed to calculate chi-squared from the user. First of all, a graphics mode is called in line 370, prior to entering the procedure (mode calls are not allowed within procedures). Line 780 moves the graphics origin and lines 790 to 900 draw up the outlines of the data table on the screen. The table headings are written in lines 940 to 1030 using VDU5 to write text to the graphics cursor.

Species A information is written in red, logical colour 1, species B information in yellow, logical colour 2. The main table heading is written in white, logical colour 3. Lines 1050 to 1090 print screen messages; first of all assigning the input species names to A or B in the appropriate colour, and then informing the user that input should be a number in each cell of the table. Data is accepted from the user and input to the appropriate cell of the data table in lines 1130 to 1180, once again using VDU5.

Once the data has been loaded, it is validated by calling the procedure **validate** from line 420. This procedure, lines 1220 to 1260, checks that the denominator of the expression for chi-squared is not zero. If it is, the value of flagv is set to 1 (line 1240). This procedure also calculates n, the number of quadrat samples in line 1250. Program control then returns to line 430.

Text mode 7 is entered at line 430. If flagv has been set to 1 in the procedure **validate**, a diagnostic message is printed (line 460) and the program terminated by calling the procedure **end** (lines 1290 to 1330). If flagv is zero, then a message to the effect that the data is valid is printed and procedure **continue** called (lines 470 and 480). Finally, the procedure **calculation** is called in line 520. The procedure **calculation**, lines 1360 to 1450 first of all determines the value of chi-squared. The expression used depends upon the value of n (line 1370). The screen is then cleared and the results output to the user, lines 1380 to 1440. If n is less than 30, the user is informed that a Yates correction has been used (line 1410).

Using the Program

Species B is Rhizobium		
No of quadrats		
Species B present	?34	?9
Species B absent	?12	?
Input appropriate data-no. of quadrats		

Figure 2.1. Data Table for Input of Quadrant Numbers

Associations between species in continuous habitats can be revealed by first of all taking standard quadrat samples at random in the habitat and then scoring the quadrats as belonging to one of the four cells of the 2 x 2 contingency table. The numbers of quadrats in each cell can then be used as input to the program ASSOC, from which a probability of association can be assessed. Association analysis has formed the basis of a good deal of project work for A level students, particularly on residential field courses. A package of programs including ASSOC and SPECDIV and AGINDEX, all described in this chapter, is as essential an ecological tool as a quadrat.

ASSOC Listing

```
10REM ASSOC
20REM
30REM ASSOCIATION INDEX USING CHI-SQUARED
40REM
50REM TITLE AND INTRODUCTION
60REM
70*FX200,1
80MODE7
90VDU23,1,0;0;0;0;0;
100FORI%=8TO9
110PRINTTAB(9,I%);CHR$&8D;"ASSOC"
120NEXTI%
130delay=INKEY(500)
140CLS
150PRINTTAB(0,10);"This program enables you to
determine""the degree of association between"
""two species."
160PROCcontinue
170CLS
180PRINTTAB(0,10);"The program uses chi-square
d for""a 2 X 2 contingency table."
190PROCcontinue
200REM
210REM
220REM INPUT NAMES OF SPECIES
230REM
240CLS
```

```

250VDU23,1,1;0;0;0;
260PRINTTAB(0,10);
270INPUT"Name of the first species",species1$
280PROCcheck_species_name(species1$)
290IF flagsp=1 THEN 270
300PRINT
310INPUT"Name of second species",species2$
320PROCcheck_species_name(species2$)
330IF flagsp=1 THEN 310
340REM
350REM INPUT RAW QUADRAT DATA
360REM
370MODE1
380PROCquadrat_data
390REM
400REM CHECK VALIDITY OF DATA
410REM
420PROCvalidate
430MODE7
440VDU23,1,0;0;0;0;0;
450PRINTTAB(0,10);
460IF flagv=1 PRINT"Data invalid-can't continue"
e":PROCend
470PRINTTAB(13,10);"Data OK"
480PROCcontinue
490REM
500REM CALCULATE CHI-SQUARED AND PRINT RESULT
510REM
520PROCcalculation
530PRINTTAB(18,24);"END"
540*FX200,0
550END
560REM
570REM
580DEFPROCcontinue
590PRINTTAB(5,20);"Press spacebar to continue"
600REPEATUNTILGET=32
610ENDPROC
620REM
630REM
640DEFPROCcheck_species_name(A$)

```

```

650flagsp=0
660IF LEN(A$)<=10THEN 710
670PRINT
680PRINT"Species name too long-try again"
690PRINT
700flagsp=1
710ENDPROC
720REM
730REM
740DEFPROCquadrat_data
750REM
760REM DRAW TABLE FOR DATA
770REM
780VDU29,200;0;
790MOVE-200,100
800DRAW900,100
810DRAW900,700
820DRAW-200,700
830MOVE200,100
840DRAW200,900
850MOVE550,900
860DRAW550,100
870MOVE900,700
880DRAW900,900
890MOVE-200,400
900DRAW900,400
910REM
920REM LABEL TABLE
930REM
940VDU5
950GCOLOR,1
960MOVE230,780:PRINT"Species A":MOVE230,750:PR
INT"present"
970MOVE580,780:PRINT"Species A":MOVE580,750:PR
INT"absent"
980GCOLOR,2
990MOVE-150,250:PRINT"Species B":MOVE-150,220:
PRINT"absent"
1000MOVE-150,550:PRINT"Species B":MOVE-150,520:
PRINT"present"
1010GCOLOR,3

```

```

1020MOVE-150,780:PRINT"No of":MOVE-150,750:PRIN
T"quadrats"
1030VDU4
1040PRINT
1050COLOUR1:PRINT"Species A is ";species1$
1060PRINT
1070COLOUR2:PRINT"Species B is ";species2$
1080COLOUR3
1090PRINTTAB(0,30);"Input appropriate data-no.
of quadrats"
1100REM
1110REM INPUT DATA
1120REM
1130VDU5
1140MOVE300,550:INPUTa
1150MOVE650,550:INPUTb
1160MOVE300,250:INPUTc
1170MOVE650,250:INPUTd
1180VDU4
1190ENDPROC
1200REM
1210REM
1220DEFPROCvalidate
1230flagv=0
1240IF a+b=0 OR c+d=0 OR a+c=0 OR b+d=0 THEN f1
agv=1
1250n=a+b+c+d
1260ENDPROC
1270REM
1280REM
1290DEFPROCend
1300PRINTTAB(5,20);"Program terminated"
1310*FX200,0
1320END
1330ENDPROC
1340REM
1350REM
1360DEFPROCcalculation
1370IF n<30 THEN chi_squared=((ABS(a*d-b*c)-n/2
)^2*n)/(((a+b)*(c+d)*(a+c)*(b+d)) ELSE chi_square
d=(((a*d)-(b*c))^2)*n)/(((a+b)*(c+d)*(a+c)*(b+d)
)

```

```

1380CLS
1390PRINTTAB(0,10);"Chi-squared = ";chi_squared
1400PRINT
1410IF n<30 THEN PRINT "Yates correction employ
ed":PRINT
1420PRINT"One degree of freedom"
1430PRINT
1440PRINT"n = ";n
1450ENDPROC

```

ASSOC - Variables

l%	control variable of enhanced character loop
delay	delay in program execution
species1\$	generic name of first species
species2\$	generic name of second species
flagsp	takes the value 1 if logical length of species name greater than 10
flagv	takes the value 1 if input data not suitable for calculation of χ^2
A\$	formal parameter for procedure check species name: actual parameter is species1\$ or species2\$
a	number of quadrats containing species 1 and species 2
b	number of quadrats containing species 1 and not species 2
c	number of quadrats containing species 2 and not species 1
d	number of quadrats containing neither species 1 or 2
n	total number of quadrats
chi_squared	calculated value of χ^2

AGINDEX

This program allows you to determine aggregation indices.

Biological and Theoretical Background

Techniques used to estimate the degree of aggregation of species, usually plants, are discussed fully by Poole (5). One of these, the use of Morisita's Index, applies to populations consisting of isolated groups of individuals, as may be the case for algae on rocky shores or marram grass on sand dunes. Morisita's Index is relatively independent of quadrat size.

The second measure of aggregation described in this section is that proposed by Clark and Evans (13). It is a nearest neighbour analysis which can be used when the population density is known. Individuals are chosen at random, and the distance between the chosen individual and its nearest neighbour of the same species measured. The observed results are then tested against the null hypothesis that the distribution of the organisms is random.

Morisita's Index, I_{δ} is calculated from:

$$I_{\delta} = ((\sum_{i=1}^N n_i(n_i-1))N)/(n(n-1))$$

where N is the number of samples, n_i the number of individuals in the i th sample, and n the total of all individuals in all samples.

If the index has a value of 1, the individuals are dispersed at random. If the index is less than 1, the population has a regular pattern. If the index is more than 1, the individuals are aggregated.

The calculated statistic for the Clark and Evans method is R , where R is the ratio of the average distance between chosen individuals and their nearest neighbours, and the expected mean if dispersion was random.

The average measured distance is given by $\bar{r} = \sum r/N$, where r is the nearest neighbour distance and N the number of measurements. The expected mean, assuming random dispersion, is $E(r) = 1/(2\sqrt{p})$, where p is the population density measured in the same units as r .

If the dispersion of individuals is random, $\bar{r}/E(r)$ will be 1. The significance of deviation from $E(r)$ is found by calculating a standardised normal variate, z , where

$$z = (\bar{r} - E(r))/SE(r)$$

and

$$SE(r) = .26136/(\sqrt{Np})$$

As a rule of thumb, if z is -1.96 or less, then the population is aggregated, r being significantly different from $E(r)$.

The Computer Program

The program AGINDEX allows the user to determine aggregation indices by either or both of these methods, by keying the appropriate data into the computer.

The program then prints either Morisita's Index, or the results of the nearest neighbour analysis, and then offers the option of using the other method.

Program Description

After a brief statement of the program's use, the user is asked to choose either Morisita's or nearest neighbour method, lines 20 to 250. In line 260, the procedure `morisita` or the procedure `nearest.neighbour` is called depending on the value of the variable `method`.

The procedure `morisita`, lines 530 to 810, first of all takes data input from the user (lines 590 to 720) and then calculates the value of I_δ (line 740). Results are printed to the screen in lines 760 to 800.

The procedure `nearest.neighbour`, lines 840 to 1270, works in analogous fashion, taking input in lines 900 to 1020, carrying out calculations in lines 1110 to 1150, and printing results to the screen in lines 1170 to 1250.

On exiting either procedure, control is returned to line 300, where the user is given the option of using the other method or terminating the program.

AGINDEX Listing

```
10REM AGINDEX
20REM
30REM Determination of aggregation indices by
Morisita's method
40REM & by nearest neighbour analysis.
50REM
60*FX200,1
70MODE7
80VDU23,1,0;0;0;0;
90REM Introduction
100REM
```

```

110PRINTTAB(6,10);"Aggregation analysis"
120PROCdelay(200)
130PRINTTAB(0,15);"You can choose one of two m
ethods"
140PROCdelay(200)
150PROCcontinue
160REM
170REM Choose method of analysis
180REM
190REPEAT
200CLS
210VDU23,1,1;0;0;0;
220PRINTTAB(0,10);"Which method?"
230PRINTTAB(0,12);"Type 1 for Morisita's metho
d""Type 2 for nearest neighbour analysis"
240PRINTTAB(0,17);:INPUT"Which method",method
250UNTIL method=1 OR method=2
260IF method=1 PROCmorisita ELSE PROCnearest_n
eighbour
270REM
280REM Other method option
290REM
300REPEAT
310CLS
320PRINTTAB(0,10);:INPUT"Do you want to try th
e other method",answer$
330UNTIL LEFT$(answer$,1)="Y" OR LEFT$(answer$
,1)="y" OR LEFT$(answer$,1)="N" OR LEFT$(answer$
,1)="n"
340IFLEFT$(answer$,1)="N" OR LEFT$(answer$,1)=
"n" THEN 380
350REM
360IF method=1 PROCnearest_neighbour ELSE PROC
morisita
370GOTO300
380*FX200,0
390CLS
400PRINTTAB(10,10);"END"
410END
420DEFPROCcontinue
430PRINTTAB(3,20);"Press spacebar to continue"

```

```

440REPEATUNTILGET=32
450ENDPROC
460REM
470REM
480DEFPROCdelay(cenitsecs)
490delay=INKEY(cenitsecs)
500ENDPROC
510REM
520REM
530DEFPROCmorisita
540CLS
550PRINTTAB(8,10);"Morisita's Index"
560VDU23,1,0;0;0;0;0;
570PROCcontinue
580VDU23,1,1;0;0;0;0;
590REPEAT
600CLS
610PRINTTAB(0,10);:INPUT"How many samples (N)"
,Nsamples
620UNTIL Nsamples=ABS(INT(Nsamples))
630REM
640FORI=1TONsamples
650REPEAT
660CLS
670PRINTTAB(0,10);"How many individuals in sam
ple ";I;
680INPUTn
690UNTIL n=ABS(INT(n))
700totn=totn+n
710sumn=sumn+n*(n-1)
720NEXTI
730REM
740MIndex=sumn*Nsamples/(totn*(totn-1))
750REM
760CLS
770PRINTTAB(0,10);"Morisita's Index = ";MIndex
780PRINTTAB(0,13);Nsamples;" samples"
790PRINTTAB(0,14);totn;" individuals"
800PROCcontinue
810ENDPROC
820REM

```

```

830REM
840DEFPROCnearest_neighbour
850CLS
860VDU23,1,0;0;0;0;
870PRINTTAB(3,10);"Nearest neighbour analysis"
880PROCcontinue
890VDU23,1,1;0;0;0;
900REPEAT
910CLS
920PRINTTAB(0,10);:INPUT"How many measurements
",measurements
930UNTIL measurements=ABS(INT(measurements))
940REM
950FORI=1Tomeasurements
960REPEAT
970CLS
980PRINTTAB(0,10);"Measurement ";I;
990INPUTr
1000UNTILr=ABS(r)
1010sumr=sumr+r
1020NEXTI
1030REM
1040meanr=sumr/measurements
1050REM
1060REPEAT
1070CLS
1080PRINTTAB(0,10);:INPUT"Population density ",
popdens
1090UNTIL popdens=ABS(popdens)
1100REM
1110Er=1/(2*SQR(popdens))
1120REM
1130R=meanr/Er
1140SEr=.26136/(SQR(measurements*popdens))
1150z=(meanr-Er)/SEr
1160REM
1170VDU23,1,0;0;0;0;
1180CLS
1190PRINTTAB(0,5);"Nearest neighbour analysis"
1200PRINTTAB(0,10);"R = ",R
1210PRINTTAB(0,12);"z = ";z

```

```

1220PRINTTAB(0,14);"Mean r = ",meanr
1230PRINTTAB(0,15);"E(r) = ",Er
1240PRINTTAB(0,16);"SE(r) = ",SEr
1250PRINTTAB(0,17);measurements;" measurements"
1260PROCcontinue
1270ENDPROC

```

AGINDEX-variables:

method	takes the value 1 or 2 depending on chosen method; 1 for Morisita's method, 2 for nearest neighbour analysis
answer\$	user response to other method option
delay	delay in program execution
Nsamples	number of samples
n	number of individuals in a particular sample
totn	number of individuals in all samples
sumn	sum of values of $n(n-1)$ for all samples
l	loop control variable
MIndex	Morisita's index of aggregation
measurements	number of measurements
r	distance from an organism to its nearest neighbour of the same species
sumr	sum of r's
meanr	mean value for r
popdens	population density
Er	expected value of meanr
R	ratio of meanr to Er
SEr	standard error
z	standardised normal variate

T-TEST

This is a test of the significance of the difference between the means of two samples.

Biological and Theoretical Background

It is often necessary in the course of biological investigations, to assess whether the difference between the means of two samples is significant, or whether it has arisen due to the operation of chance factors.

To do this, two statistics are needed for each sample or data set. These statistics are the mean and standard deviation for each sample.

Mean, \bar{x} , is given by $\sum_{i=1}^n x_i / n$ where x_i is the i th data item and n is the number of data items.

If a data set represents a complete population, then standard deviation, σ , is given by $\sqrt{\sum x_i^2 / n - \bar{x}^2}$. If a data set is a small sample of a total population, then the standard deviation of the population is estimated using s , where $s = \sqrt{(\sum (x_i - \bar{x})^2) / (n - 1)}$.

These statistics can be used in two ways to assess the degree of significance of the difference between two means, a crude way and a quite sophisticated way.

The crude way is often used at A level due to its simplicity and ease of use. It involves calculating an estimate of the standard error of the difference between the two means, $s_{\bar{x}_1 - \bar{x}_2}$, where \bar{x}_1 is the mean of the first data set, and \bar{x}_2 the mean of the second. In (8) and (14), formulae are given to calculate this statistic using the population standard deviation, σ , for each data set. Standard error is:

$$\text{Standard error (SE)} = \sqrt{\sigma_1^2 / n_1 + \sigma_2^2 / n_2}$$

where σ_1^2 is the square of the standard deviation of the first data set (the variance), σ_2^2 is the variance of the second data set, n_1 and n_2 are the number of items in each data set.

As a rule of thumb, if the actual difference between the two means, $\bar{x}_1 - \bar{x}_2$, is greater than or equal to twice the standard error as calculated above, then the difference between the means is significant, and has not arisen by chance.

The sophisticated way also involves the calculation of the standard error of the difference between the two means. In this case, an estimate of standard error, $s_{\bar{x}_1 - \bar{x}_2}$ is used.

$$s_{\bar{x}_1 - \bar{x}_2} = \sqrt{(s_1^2 / (n_1 - 1)) + (s_2^2 / (n_2 - 1))}$$

where s_1 is an estimate of the standard deviation of the first data set, and s_2 that of the second.

The statistic t is then calculated to assess the degree to which the observed difference between the means is different to the expected difference. Usually, this latter null hypothesis, is that the expected difference between the two means is zero. Hence t is given by:

$$t = (\bar{x}_1 - \bar{x}_2) / (s \sqrt{\frac{1}{n_1} + \frac{1}{n_2}})$$

with $n_1 + n_2 - 1$ degrees of freedom.

This value of t can then be compared to a table of t giving critical values for significance (7).

An example of the use of both procedures is given below.

The Computer Program

The program T-TEST allows the user to key in to the computer raw data in two sets, the difference between the means of which is then tested by calculating both t and the standard error of difference.

Output from the program shows the mean and standard deviations of each data set, together with the calculated value of t using the null hypothesis that no difference exists between the means. The number of degrees of freedom for t is output allowing the use of a table of t . Finally, the standard error of the difference between the means is printed together with the actual difference between the two means.

Program Description

The program begins with a brief introduction (lines 70 to 100). Data is then keyed into the computer using the procedure **input.data** (lines 340 to 510) called in line 140 for the first data set and in line 160 for the second. The procedure uses the formal parameter $A\$$, which takes the value "First" and then "Second" during the two calls made to it.

The procedure first of all requests the user to key in the number of items of data in the data set. This is checked in line 410, and only a positive integer will be accepted. Items are then requested one at a time in lines 440 to 500. In lines 480 to 490 accumulators for the sum of all data items, sum , and the sum of the squares of all data items, $sum\$$ are employed. These variables are translated as belonging to data sets 1 or 2 when **ENDPROC** returns program control to lines 150 or 170 respectively.

The procedure **calculate.t** is then called from line 210. This procedure occupying lines 540 to 610 calculates the means for each data set, line 550, the standard

error estimate, line 560, and t, line 570. Calculations in lines 580 to 600 produce a crude estimate of standard error of difference.

Finally, the results are printed by the procedure **print.results** running from lines 640 to 770, called in line 230. Program execution then terminates.

Using the Program

Consider the following data concerning the conception rates in cows artificially inseminated using semen from two bulls.

Bull 1	74.2	62.1	57.7	71.7	62.0	76.1	70.6	68.3	68.4
	79.8	71.1	70.9	65.5	61.2	60.8	73.9	51.9	63.7
Bull 2	49.6	49.2	53.2	56.5	69.1	54.2	80.7	62.7	71.5
	67.5	64.6	75.4	79.6	59.8	68.8	60.2		

This data is taken from Campbell (15).

T-TEST can be used to decide whether the conception rates in the two data sets are significantly different.

When the above data is entered into the program, the following results are obtained:

Data Set (Bull) 1 Mean = 67.217, standard deviation = 6.931 n = 18

Data Set (Bull) 2 Mean = 63.913, standard deviation = 9.680 n = 16

t is calculated to be 1.119 with 32 degrees of freedom, standard error of difference is 2.920.

All results are quoted to three decimal places.

From a t table in (7), the probability of the observed difference in the means occurring on the null hypothesis of no difference is greater than .2 on a two-tailed test and greater than .1 for a one-tailed test. This indicates that the observed difference in the means is not significantly different from zero.

This conclusion is supported by the standard error test, where the standard error of difference between the means, 2.92 is more than half the actual difference in the means, 3.304.

T-TEST Listing

```
10REM T-TEST
20REM
30REM Test of significance of difference between means of small samples
40REM
50REM Introduction
60REM
70MODE7
80VDU23,1,0;0;0;0;
90PRINTTAB(0,10);"Test of significance of difference"" between means of two samples."
100PROCcontinue
110REM
120REM Input data
130REM
140PROCinput_data("First")
150sum1=sum:sumS1=sumS:n1=items
160PROCinput_data("Second")
170sum2=sum:sumS2=sumS:n2=items
180REM
190REM Calculate_t
200REM
210PROCcalculate_t
220REM
230PROCprint_results
240*FX200,0
250END
260REM
270REM
280DEFPROCcontinue
290PRINTTAB(3,20);"Press spacebar to continue"
300REPEATUNTILGET=32
310ENDPROC
320REM
330REM
340DEFPROCinput_data(A$)
350sum=0:sumS=0
360REPEAT
```

```

370CLS
380PRINTTAB(8,10);A$;" data set"
390delay=INKEY(100)
400PRINTTAB(8,15);:INPUT"How many items",items
410UNTIL items=ABS(INT(items))
420PROCcontinue
430REM
440FORI=1TOitems
450CLS
460PRINTTAB(8,15);A$;" data set"
470PRINTTAB(10,17);"Item ";I;:INPUTitem
480sum=sum+item
490sumS=sumS+item^2
500NEXTI
510ENDPROC
520REM
530REM
540DEFPROCcalculate_t
550mean1=sum1/n1:mean2=sum2/n2
560s=SQR(((sumS1-(sum1^2/n1)+sumS2-(sum2^2/n2)
)/(n1+n2-2))*(1/n1+1/n2))
570t=(mean1-mean2)/s
580sd1=SQR(sumS1/n1-mean1^2)
590sd2=SQR(sumS2/n2-mean2^2)
600SE=SQR(sd1^2/n1+sd2^2/n2)
610ENDPROC
620REM
630REM
640DEFPROCprint_results
650CLS
660PRINTTAB(0,4);"Data Set 1"
670PRINTTAB(0,6);"Mean is ";mean1
680PRINTTAB(0,7);"Standard deviation is ";sd1
690PRINTTAB(0,8);"n is ";n1
700PRINTTAB(0,11);"Data Set 2"
710PRINTTAB(0,14);"Mean is ";mean2
720PRINTTAB(0,15);"Standard deviation is ";sd2
730PRINTTAB(0,16);"n is ";n2
740PRINTTAB(0,18);"t is ";t;"with ";n1+n2-2;"
degrees of freedom."
750 PRINTTAB(0,21);"SE of diff btwn means is "

```

```

;SE
  760 PRINT"Actual difference is ";ABS(mean1-me
n2)
  770ENDPROC

```

T-TEST - Variables

sum1	sum of all items in data set 1
sum 2	sum of all items in data set 2
sumS1	sum of squares of all items in data set 1
sumS2	sum of squares of all items in data set 2
n1	number of items in data set 1
n2	number of items in data set 2
sum	sum of data items
sumS	sum of squares of data items
item	value of input data item
items	number of data items in a data set
mean1	mean for first data set
mean2	mean for second data set
s	sample estimate of standard error of difference between means
t	calculated value of the t statistic
sd1	standard deviation of first data set
sd2	standard deviation of second data set
SE	standard error of difference between means

References

1. Carter, D.C., Godsen, M.S., Orton, A., Wain, G.T. and Wood-Robinson, C., *Mathematics in Biology*, Nelson, Walton-on-Thames, 1981.
2. Alexander, R. McN., *Size and Shape*, Arnold, London, 1971.
3. Huxley, J.S., *Problems of Relative Growth*, Methuen, London, 1932.
4. Mosteller, F., Rourke, R.E.K., and Thomas, G.B., *Probability with Statistical Applications*, Addison-Wesley, London, 2nd Ed., 1970.
5. Poole, R.W., *An Introduction to Quantitative Ecology*, McGraw-Hill, Tokyo, 1974.
6. Sands, M.K., *Problems in Animal Physiology*, John Murray, London, 1975.
7. Haber, A., and Runyon, R.P., *General Statistics*, Addison-Wesley, London, 2nd Ed., 1973.
8. Nuffield Advanced Science. *Biological Science. Study Guide*, Penguin, Harmondsworth, 1970.
9. Krebs, C.J., *Ecology - The Experimental Analysis of Distribution and Abundance*, Harper and Row, New York, 1978.
10. Hutcheson, K., 'A Test for Comparative Diversities based on The Shannon Formula', *J. Theor. Biol.*, 29, (1970), 151-154.
11. Good, I.J., *Probability and the Weighing of Evidence*, Griffin, London 1950.
12. Lewis, J.R., *The Ecology of Rocky Shores*, English Universities Press, London, 1964.
13. Clark, P.J., and Evans, F.C., 'Distance to Nearest Neighbour as a Measure of Spatial Relationships in Populations', *Ecology*, 35, (1954), 445-453.
14. Roberts, M.B.V., *Biology: A Functional Approach. Students Manual*, Nelson, Walton-on-Thames, 1974.
15. Campbell, R.C., *Statistics for Biologists*, Cambridge University Press, Cambridge, 2nd Ed., 1974.

CHAPTER 3

GENETICS

Introduction

In many ways, the study of genetics lies at the heart of modern biology. If there is a unifying concept in the life sciences, it may be the idea of evolution by natural selection, an idea described quantitatively in the language of genetics.

Until quite recently, genetics meant simply the study of inheritance; and that meant Mendel's laws and their exceptions. This is classical genetics.

The first three programs described in this chapter fall under this heading.

MENDEL is a simple simulation of some of Mendel's breeding experiments with peas. LINK simulates the genetics of a dihybrid genetic situation involving two pairs of alleles occupying loci on the same chromosome. SEXL simulates experiments involving a sex-linked phenotype in *Drosophila*.

Increasingly, genetics is being used to generate an understanding of the processes that have produced the staggering variety and range of forms of living things, of evolution. This use of genetics is often called population genetics. The fourth and fifth programs in this chapter are simulations in this area.

SICKLE allows the user to investigate a simple model in which selection brings about the evolution of a model population. DRIFT demonstrates that in small populations, evolution can occur in the absence of selection.

MENDEL — a program to illustrate classical genetics

This program is a simulation of experiments to demonstrate and investigate Mendel's Laws.

Biological Background

Mendel's laws are the cornerstone of genetics. Students of O and A level biological sciences are expected to gain a thorough understanding of these laws. Future study in genetics is really impossible without an easy familiarity with them.

Mendel's laws are the rules of simple inheritance. The first law concerns the particulate nature of inheritance, and states that for each feature of an organism (phenotype), each somatic cell carries two particles or alleles.

Further, the law states that each gamete produced by an organism has only one of these alleles. The nature of the two alleles in somatic cells, is referred to as the cells' genotype. It is often the case that the two alleles are not identical, and one may be dominant to the other.

One of Mendel's experiments involved crossing together two sorts of pea plants, one with very tall stems and one with short or dwarf stems. He found that if the parent plants used in this cross came from pure strains of tall or dwarf, that the progeny or offspring produced, the so-called F1 generation were all tall stemmed. Further, he found that if these F1 tall plants were allowed to self-fertilise, to intercross, then the progeny produced consisted of 75% tall plants and 25% dwarf ones.

The information for dwarf had 'skipped' a generation, but must have been present in the F1 tall plants; it had been masked by tall information. Tall is dominant, dwarf recessive.

This result can be explained if it is assumed that each parent cell, except gametes contain two alleles. Let T be the tall allele, and t be the recessive allele. Then, the tall parent has genotype TT, and the dwarf parent has genotype tt. If we assume that the gametes produced by these parents each contain one of the pair of alleles in somatic cells, then each tall gamete has genotype T, and each dwarf genotype t. Thus each fertilisation of a tall gamete by a dwarf one (or vice versa) gives rise to a zygote and hence a new plant of genotype Tt.

The phenotype of this plant is tall. Now, if one of these plants is allowed to self-fertilise, or is crossed to a plant of the same genotype, then half of the gametes produced by the F1 hybrid have the genotype T and half the genotype t. Assuming that fertilisation occurs at random, and that gametes containing the allele T are as likely to be fertilised, or effect fertilisation as those gametes containing t, then the probabilities of the production of zygotes containing at least one T allele, or no T alleles are .75 and .25 respectively. A similar argument applies to the simultaneous and independent tossing of two coins. The probability of a result consisting of a least one head is .75. That of two tails is .25.

Mendel's first law allows a prediction of the outcome when an F1 hybrid is crossed to an individual with the genotype tt. (An individual with both alleles the same is homozygous for that pair of alleles, else it is heterozygous.) When a heterozygote, Tt is crossed with a homozygous recessive, tt in this case, Mendel's first law predicts that the progeny should consist of 50% tall and 50% dwarf plants. The probabilities of occurrence of zygotes containing at least one T or no Ts are respectively .5 and .5. These probabilities are obtained, as above, by multiplying together the relative frequencies of gametes of different genotypes produced by the two parents. In the present case, the F1 plant produces two gamete types, T and t, with the same relative frequency of 1/2. The homozygous recessive parent

produces only one type of gamete, that containing the allele *t*; the relative frequency of this gamete is 1. This latter cross, called a testcross, gives rise to approximately equal numbers of tall heterozygotes and homozygous recessives.

The second of Mendel's two laws concerns the outcome of crosses between individuals which differ from each other in two phenotypes. The second law is a law of dihybrid genetics; the crosses described previously were monohybrid. Mendel's second law depends upon the validity of Mendel's first law, which is always assumed to apply, at least in the first instance.

Another of Mendel's original investigations with pea plants concerned the cross between two types of plant from pure-breeding strains. One of these types produced round and yellow peas, the other produced wrinkled and green ones. The F₁ progeny plants all produced round, yellow peas. Round is dominant to wrinkled and yellow is dominant to green. If the genotypes of the round, yellow and wrinkled, green parents were respectively RRY_Y and rryy, then that of the F₁ would be RrYy, a dihybrid. The second law concerns the progeny produced by this type of individual when intercrossed or testcrossed to the rryy, or double homozygote.

Assuming Mendel's first law, and that the separation of members of the two pairs of alleles into gametes occurs independently, then the relative frequencies of the four possible gamete genotypes, RY, Ry, rY and ry are all $\frac{1}{4}$.

This is really a statement of Mendel's second law, the Law of Independent Assortment. If the law is true for a particular dihybrid, then it is possible to predict the composition of intercross and testcross progenies.

Assuming random fertilisation and Mendel's second law, the relative frequencies of the four possible phenotypes produced by intercross turn out to be:

round, yellow	9/16
round, green	3/16
wrinkled, yellow	3/16
wrinkled, green	1/16

Once again, these figures are obtained by multiplying the relative frequencies of each gamete genotype produced by the two parents and taking sums for each phenotype in the progeny.

Similarly, Mendel's second law predicts that the ratio of these four phenotypes produced by testcross should be 1: 1: 1: 1 – the same ratio as that of different gamete genotypes produced by the F₁.

Whilst Mendel's first law is usually true, the second law seems to be the exception rather than the rule. The assortment of pairs of alleles into gametes cannot occur independently if the pairs of alleles lie on the same chromosomes. It is only when the two pairs of alleles under consideration are on different chromosomes that Mendel's second law predictions work.

This happens to be the case for the round/wrinkled and yellow/green alleles.

A more detailed account of the genetic background can be found in Srb et al (1). A fascinating account of Mendel's life and work can be found in Bronowski (2).

The computer program

The program MENDEL was written to guide a class through the argument leading to Mendel's laws. It has also been used for individual investigation.

The program generates data in the form of numbers of progeny of different phenotypes produced by crosses set up by the program itself or by the user. It can be used to generate data enabling the design and testing of hypotheses in simple monohybrid or dihybrid genetics.

The program contains a probabilistic model which generates appropriate progeny at random. Hence repetitions of the same cross rarely give rise to the same progeny, as is of course the case in nature.

Program description

MENDEL, as with the other two programs in this classical genetics section, is built around the procedure **determine.progeny**, which occupies lines 310 to 500. It uses the formal parameters a\$, b\$, c\$, d\$ and x1, x2 and x3.

The strings are the names of the different phenotypes which the current genetic situation is likely to generate. The numeric variables are the probabilities of occurrence of the different phenotypes respectively, the probability of occurrence of the phenotype d\$ being $(1-x_1-x_2-x_3)$.

The procedure begins by printing as column headings the names of potential phenotypes. In the case of mono-hybrid genetics, both a\$ and d\$ are just one space, " ", which appears as a blank on the screen. During any one call of the procedure, 100 progeny are generated.

Lines 400 to 450 are occupied by the progeny generator. Line 410 generates a random number between 0 and 0.999999, which is used to increment by 1, the appropriate variable, totaln in line 420 using the formal parameters x1, x2 and x3. n in totaln is 1,2,3 or 4. For example, if $x_1 = .25$, $x_2 = .5$ and $x_3 = .75$ and R takes the value .9963, then in line 420, the variable total 4 will be incremented by 1. If in the same call, R took the value .497, then as $R < x_2$, the variable total2 is incremented by 1. The values of the total accumulators are then printed under the appropriate phenotypes in line 440. If a \$ is " "; as is the situation in the mono-hybrid case, only the values of total3 are printed to the screen.

Lines 450 to 485 allow the user to generate more data else the procedure terminates or generates a diagnostic error message. The user can produce as many phenotypes as he likes by answering Y or y to the prompt "More?".

Mendel begins by calling the procedure **introductory.remarks**, lines 530 to 610 from line 80. As usual, *FX200, 1 disables the ESCAPE key and VDU23, 1,0;0;0;0; turns off the flashing cursor.

The user can then choose monohybrid or dihybrid genetics by calling the procedure **mono.or.di** from line 120. This procedure, occupying lines 700 to 770 contains the line 820 **VDU23, 1,1;0;0;**, which turns the flashing cursor on. On leaving this procedure, the variable **gen.type** has the value 1 or 2, depending on which type of crosses, mono- or dihybrid respectively have been chosen. This variable is then used in the conditional branch at line 130 to call the procedure **mono** or the procedure **di**. These procedures are very similar; **mono** occupies lines 800 to 940, **di** lines 1210 to 1350.

Both procedures begin by printing a heading and then the nature and result of the first cross. The first cross is one involving two pure-breeding parents. The user is then asked to choose the type of cross he wishes to investigate by calling the procedure **type.of.cross** lines 970 to 1070. This procedure uses the formal parameter **hom.rec.parent\$**, the value of which is passed to the procedure from the procedure **mono.or.di**. In the case of **di**, 'WRINKLED GREEN' is passed into **hom.rec.parent\$**. The value of this variable is used in the printout from line 1060, which prints the nature of the type of cross chosen by the user. Control then returns to the procedure **mono** or the procedure **di**. From either procedure, the procedure **determine.progeny** described above, is called. When control returns, the user is given the option of choosing the type of cross other than just carried out by calling the procedure **other.cross**, lines 1100 to 1180. If the other cross is not chosen, control passes to line 170, where the user is given the option of continuing the run or ending the program.

Using the program

The program MENDEL can be used to introduce students of O or A level, or indeed CSE or other levels, to the fundamentals of genetics. Because the program generates realistic genetic data, the program can be used to supplement real investigations, and its results treated as 'first hand'.

The monohybrid section of the program demonstrates, with the guidance of the teacher, the particulate nature of inheritance, and can be used to introduce the idea of dominance. By setting up a testcross and discussing the result, a class can be led to suggest models for inheritance. They can design an hypothesis which can be tested by running the intercross option.

This sequence can be followed for the dihybrid case, and the assumptions of the model checked by running appropriate parts of the program.

The program generates data which can be used to design hypotheses which can be tested, supported or refuted by experiment, i.e. by running other parts of the program. MENDEL can thus help students to do real science, to experience and use scientific method (3). Of course, this is done better in the laboratory, using organisms. However, the use of real organisms may be difficult; a single cross using *Drosophila* may take 2-3 weeks, each of Mendel's crosses took a year. The computer enables students to collect cross data from as many experiments as a lesson permits, and follow a complete experimental procedure in the same time.

MENDEL - Listing

```
10REM MENDEL
20REM
30REM Introduction
40REM
50*FX200,1
60MODE7
70VDU23,1,0;0;0;0;
80PROCintroductory_remarks
90REM
100REM Choose mono- or dihybrid genetics
110REM
120PROCmono_or_di
130IFgen_type=1 PROCmono ELSE PROCdi
140REM
150REM Program continuation offer
160REM
170REPEAT
180VDU23,1,1;0;0;0;
190CLS
200PRINTTAB(0,10);:INPUT"Do you wish to c
ontinue ",answer$
210f1$=LEFT$(answer$,1)
220UNTIL f1$="Y" OR f1$="y" OR f1$="N" OR
f1$="n"
230IF f1$="Y" OR f1$="y" THEN 120
240CLS
250VDU23,1,0;0;0;0;
260*FX200,0
270PRINTTAB(10,10);"END"
280END
290REM
300REM
310DEFPROCdetermine_progeny(a$,b$,c$,d$,x
1,x2,x3)
320CLS
330VDU23,1,0;0;0;0;
340PRINTTAB(0,5);a$,b$,c$,d$
350total1=0
```

```

360total12=0
370total13=0
380total14=0
390PRINT
400FOR K%=1 TO 100
410R=RND(1)
420IF R<x1 THEN total1=total1+1 ELSE IF R
<x2 THEN total2=total2+1 ELSE IF R<x3 THEN
total3=total3+1 ELSE total4=total4+1
430NEXT K%
440IF a$=" "PRINT " ";total2;total3 ELSE
PRINT ;total1;total2;total3;total4
450REPEAT
460INPUT "More", answer$
470IF LEFT$(answer$,1)<>"Y" AND LEFT$(ans
wer$,1)<>"y" AND LEFT$(answer$,1)<>"N" AND
LEFT$(answer$,1)<>"n" PRINT "Type Y or N"
480UNTIL LEFT$(answer$,1)="Y" OR LEFT$(an
swer$,1)="y" OR LEFT$(answer$,1)="N" OR LEF
T$(answer$,1)="n"
485IF LEFT$(answer$,1)="Y" OR LEFT$(answe
r$,1)="y" THEN 350
490PRINT
500ENDPROC
510REM
520REM
530DEFPROC introductory_remarks
540FOR I%=8 TO 9
550PRINT TAB(13,I%);CHR$(8D); "MENDEL "
560NEXT I%
570delay=INKEY(200)
580CLS
590PRINT TAB(0,5); "This program enables yo
u to""simulate Mendel's breeding""exper
iments with peas."
600PROC continue
610ENDPROC
620REM
630REM
640DEFPROC continue
650PRINT TAB(3,23); "Press spacebar to cont
inue"

```

```

660REPEATUNTILGET=32
670ENDPROC
680REM
690REM
700DEFPROCmono_or_di
710REPEAT
720CLS
730VDU23,1,1;0;0;0;
740PRINTTAB(0,10);"Mono- or dihybrid gene
tics?" "" "For monohybrid Type 1" "" "For dih
ybrid Type 2" "" ""
750INPUT"Type of genetics",gen_type
760UNTIL gen_type=1 OR gen_type=2
770ENDPROC
780REM
790REM
800DEFPROCmono
810CLS
820VDU23,1,0;0;0;0;
830PRINTTAB(5,5);"MONOHYBRID CROSSES"
840delay=INKEY(100)
850PRINTTAB(0,10);"The first cross is:" ""
' "Pure TALL x Pure DWARF"
860delay=INKEY(100)
870PRINTTAB(0,17);"The F1 generation is "
;:delay=INKEY(100):PRINT"ALL TALL"
880PROCcontinue
890PROCtype_of_cross("DWARF")
900IF cross_type=1 PROCdetermine_progeny(
" ", "TALL", "DWARF", " ", 0, .75, 1) ELSE PROCde
termine_progeny(" ", "TALL", "DWARF", " ", 0, .5
, 1)
910PROCcontinue
920PROCother_cross
930IF f1$="Y" OR f1$="y" THEN 890
940ENDPROC
950REM
960REM
970DEFPROCtype_of_cross(hom_rec_parent$)
980REPEAT
990CLS

```

```

1000VDU23,1,1;0;0;0;
1010PRINTTAB(0,10);"Which type of cross?"
''"For intercross Type 1"''"For testcross
Type 2"''''''
1020INPUT"Which ",cross_type
1030UNTIL cross_type=1 OR cross_type=2
1040VDU23,1,0;0;0;0;
1050CLS
1060IF cross_type=1 PRINTTAB(5,5);"Intercr
oss: F1 x F1" ELSE PRINTTAB(5,5);"Testcross: F1 x ";hom_rec_parent$
1070ENDPROC
1080REM
1090REM
1100DEFPROCother_cross
1110REPEAT
1120CLS
1130VDU23,1,1;0;0;0;
1140PRINTTAB(5,10);
1150INPUT"Other cross",answer$
1160f1$=LEFT$(answer$,1)
1170UNTIL f1$="Y" OR f1$="y" OR f1$="N" OR
f1$="n"
1180ENDPROC
1190REM
1200REM
1210DEFPROCdi
1220CLS
1230VDU23,1,0;0;0;0;
1240PRINTTAB(7,5);"DIHYBRID CROSSES"
1250delay=INKEY(100)
1260PRINTTAB(0,10);"The first cross is:"
' "Pure ROUND,YELLOW x Pure WRINKLED,GREEN"
1270delay=INKEY(100)
1280PRINTTAB(0,17);"The F1 generation is "
;:delay=INKEY(100):PRINT"ALL ROUND,YELLOW"
1290PROCcontinue
1300PROCtype_of_cross("WRINKLED,GREEN")
1310IF cross_type=1 PROCdetermine_progeny(
"RND,YEL","RND,GRN","WRNK,YEL","WRNK,GRN",.
5625,.75,.9375) ELSE PROCdetermine_progeny(

```

```

"RND, YEL ", "RND, GRN", "WRNK, YEL ", "WRNK, GRN", .
25, .5, .75)
  1320PROCcontinue
  1330PROCother_cross
  1340IF f1$="Y" OR f1$="y" THEN 1300
  1350ENDPROC

```

MENDEL-variables

gen.type	type of genetics to be investigated, 1 for monohybrid, 2 for dihybrid
answer\$	user response to program prompts
f1\$	first letter of answers
a\$,b\$,c\$,d\$	formal string variables of procedure determineprogeny; values depend on value of genotype
x1,x2,x3	formal numeric variables of procedure determine progeny: represent probabilities of occurrence of corresponding phenotypes; values depend on genotype and cross.type
total1, total2, total3 total4	number of progeny of each of four possible phenotypes produced by one cross of any type
K%, l%	loop control variable
hom-rec.parent\$	formal variable of procedure type-of-cross: has value 'DWARF' if gen-type 1, 'WRINKLED, GREEN' if gen.type 2
cross.type	type of cross chosen: 1 for intercross, 2 for testcross
delay	delay in program execution
R	random number in range 0 to 0.999999

LINK

A simulation of experiments to investigate the phenomenon of autosomal linkage.

Biological background

The phenomenon of autosomal linkage is central in classical genetic theory.

It allows a conceptual connection to be made between the behaviour of alleles during their segregation into gametes as predicted by Mendel's first law, and that of chromosomes, which carry the alleles.

This connection was first made by Morgan (4), and a grasp of the argument leading to it is required of all students of A level and beyond.

The phenomenon of linkage concerns 2 pairs of alleles. The allelic pairs are said to be linked if each has a member on the same chromosome. The implication of this is that during the segregation of alleles into gametes during meiosis, the pairs of alleles do not segregate independently; they do not follow Mendel's second law.

Mendel's second law, when it applies, suggests that given a dihybrid, AaBb, gametes produced by the dihybrid can be grouped into four equally likely classes, AB, Ab, aB, and ab. Put another way, the four gamete genotypes occur in a ratio of 1 : 1 : 1 : 1. This occurs as Mendel's first law predicts that each gamete produced by the dihybrid has probability .5 of containing an A allele and so probability .5 of containing an a allele. The same argument applies to the alleles B and b, and if the segregation of the two allelic pairs occurs independently, then the probability of any gamete containing a and b, say, is $1/2 \times 1/2 = 1/4$. This applies equally to the other gamete genotypes, AB, Ab and aB.

In linkage then, the probabilities of occurrence of the four sorts of gamete are not all .25. Two gamete genotypes are more likely to occur than the other two. The more likely gametes are parental ones which segregate together, on the same chromosome segment. The less likely gametes are recombinants produced by crossing-over, or recombination of genetic material during meiotic prophase.

The computer program

The program LINK enables the simulation of crosses involving two contrasting phenotypes of the feathers of fowl. These are frizzle and normal, and coloured and white feathers. This situation was first described by Hutt (1). The program generates data representing the progenies of crosses set up by the user or the computer. The computer sets up the parental cross and the user can set up an intercross or testcross of the F1 animals.

Program description

The program is built around the procedure **determine.progeny** occupying lines

290 to 490 and called from line 910. The procedure is described in the program description section for MENDEL.

LINK begins by calling the procedure **start**. This occupies lines 520 to 680 and is called in line 80. A title page, lines 530 to 560 is followed by a brief explanatory note, lines 580 to 610. Then a parental cross between double homozygotes is set up and the F1 phenotypes printed, lines 620 to 670. Control then returns to line 120, from which the procedure **testcross.or.intercross** is called.

This procedure, occupying lines 780 to 920, allows the user to choose to cross the F1 animals, the dihybrids, to either other F1s or the homozygous recessive phenotype, lines 790 to 840. A message is then printed in lines 860 to 890 depending on the value of the variable **cross.type**, input at line 830. This variable has the value 1 if intercross has been chosen, 2 if testcross is preferred. The variable **cross.type** is used to call the procedure **determine.progeny** from line 910. The value of **cross.type** determines the values passed into the formal parameters x1, x2 and x3 of **determine.progeny**. These values are the probabilities of occurrence of the three phenotypes, frizzle, white; frizzle, coloured, and normal, white, taken from the account of Hutt's work in Srb et.al. (1). The probability of occurrence of normal, coloured individuals in cross progeny is $(1-x_1-x_2-x_3)$.

On exiting from **testcross.or.intercross** control passes to line 160. At line 200, the user has the option of continuing the run, and choosing the other cross or exiting from the program.

Using the program

LINK can be used to take a class through a quantitative argument to introduce the phenomenon of autosomal linkage, and deviation from Mendel's second law.

On running the program, the computer sets up the first cross which is between pure breeding frizzle, white and pure breeding normal, coloured fowl. The F1 progeny are all frizzle and white. Hence, the mutant frizzle is dominant to normal and white is dominant to coloured.

The program then gives the user the option of setting up an intercross or testcross of the F1 animals. Perhaps the best choice at this stage is testcross, which reveals the frequencies of occurrence of the various gamete genotypes produced by the F1 fowl. A typical progeny from a sample run of the testcross option, using 4 repetitions (inputting 'Y' or 'y' on the prompt More?) is as follows:

194 frizzle,white: 57 frizzle,coloured: 38 normal,white: 211 normal, coloured.

The parental combinations of frizzle and white and normal and coloured outnumber the recombinants. Certainly, the results obtained show a marked deviation from those expected on the basis of Mendel's second law. If Mendel's second law was true for these two pairs of alleles, then the ratio of progeny produced by the cross should be 1 : 1 : 1 : 1, expected numbers of 125 : 125 : 125 :

125 respectively. On the basis of this expectation, chi-squared turns out to be 194.8 with three degrees of freedom, giving a probability of very much less than .01. Hence there is a significant deviation from the results expected on the basis of Mendel's second law.

On inspection of the testcross result, it could be suggested that the phenotype segregation has produced a ratio of 4: 1: 1: 4. Using this hypothesis to produce expected numbers of progeny, chi-squared has a value of 4.645 with three degrees of freedom, corresponding to a probability of between .1 and .2. Hence, there is no significant deviation from expectation based on a 4: 1: 1: 4 segregation of F1 gamete genotypes. It is thus possible to write down the frequencies of occurrence of the four types of gamete produced by the F1, as follows:

$F_i = 0.4$; $F_i = 0.1$; $f_i = 0.1$; $f_i = 0.4$

where I is the white allele, i the coloured allele, F frizzle and f normal.

If these frequencies are inserted into a Punnett square, the phenotype frequencies expected from an F1 intercross can be predicted, as long as it is possible to assume random fertilisation. The frequencies of each of the sixteen possible zygotes are found by multiplying the appropriate gamete frequencies.

F1	F1	F_i .4	F_i .1	f_i .1	f_i .4
F_i .4		FF_{ii} .16	FF_{ii} .04	Ff_{ii} .04	Ff_{ii} .16
F_i .1		FF_{ii} .04	FF_{ii} .01	Ff_{ii} .01	Ff_{ii} .04
f_i .1		Ff_{ii} .04	Ff_{ii} .01	ff_{ii} .01	ff_{ii} .04
f_i .4		Ff_{ii} .16	Ff_{ii} .04	ff_{ii} .04	ff_{ii} .16

Punnett square used to predict frequencies of genotypes produced by F1 intercross

The expected F2 phenotype frequencies turn out to be:

frizzle, white	frizzle, coloured	normal, white	normal, coloured
$F-I-$	$F-ii$	$ffI-$	$ffii$
0.66	0.09	0.09	0.16

Thus, given 500 intercross progeny, $.66 \times 500 = 330$ should be frizzle, white. A similar calculation can be done for the other phenotypes to produce expected phenotype numbers:

frizzle, white	330
frizzle, coloured	45
normal, white	45
normal, coloured	80

This expectation, this prediction from the hypothetical F1 gamete segregation, can be checked by running the intercross part of LINK. A typical run is summarised below; five repetitions of the intercross gave 500 progeny:

frizzle, white	326
frizzle, coloured	49
normal, white	39
normal, coloured	86

Using these results as observed data, and the previous figures as expected data, chi-squared is 1.654 with three degrees of freedom, $.5 < \text{prob} < .7$. Thus, there is no significant difference between the results expected and those observed. The ratio of gamete genotypes produced by the F1 is 4: 1: 1: 4, and not 1: 1: 1: 1, hence autosomal linkage, deviation from Mendel's second law, is demonstrated.

LINK - Listing

```

10REM LINK
20REM
30REM Introduction and start
40REM
50*FX200,1
60MODE7
70VDU23,1,0;0;0;0;0;
80PROCstart
90REM
100REM Choose test-cross or intercross of
F1 animals
110REM
120PROCtestcross_or_intercross
130REM

```

```

140REM Continuation option
150REM
160PROCcontinue
170VDU23,1,1;0;0;0;
180CLS
190PRINTTAB(5,10);
200INPUT"Continue ",answer$
210f1$=LEFT$(answer$,1)
220IF f1$="Y" OR f1$="y" THEN 120 ELSE IF
f1$="N" OR f1$="n" THEN 230 ELSE PRINTTAB(
5);"Type Y or N":GOTO200
230CLS
240PRINTTAB(10,10);"END"
250*FX200,0
260END
270REM
280REM
290DEFPROCdetermine_progeny(a$,b$,c$,d$,x
1,x2,x3)
300CLS
310VDU23,1,0;0;0;0;
320PRINTTAB(0,5);a$,b$,c$,d$
330total1=0
340total2=0
350total3=0
360total4=0
370PRINT
380FORK%=1TO100
390R=RND(1)
400IF R<x1 THEN total1=total1+1 ELSE IF R
<x2 THEN total2=total2+1 ELSE IF R<x3 THEN
total3=total3+1 ELSE total4=total4+1
410NEXTK%
420PRINT;total1,;total2,;total3,;total4
430REPEAT
440VDU23,1,1;0;0;0;
450INPUT"More",answer$
460IF LEFT$(answer$,1)<>"Y" AND LEFT$(ans
wer$,1)<>"y" AND LEFT$(answer$,1)<>"n" AND
LEFT$(answer$,1)<>"N" PRINT"Type Y or N"
470UNTIL LEFT$(answer$,1)="Y" OR LEFT$(an

```

```

swer$,1)="y" OR LEFT$(answer$,1)="N" OR LEFT$(answer$,1)="n"
  475 IF LEFT$(answer$,1)="Y" OR LEFT$(answer$,1)="y" THEN 330
  480PRINT
  490ENDPROC
  500REM
  510REM
  520DEFPROCstart
  530CLS
  540FORI%=8TO9
  550PRINTTAB(13,I%);CHR$&8D;"LINK"
  560NEXTI%
  570
  580delay=INKEY(200)
  590CLS
  600PRINTTAB(0,5);"This program enables you to investigate""the genetics of a dihybrid cross""in fowl."
  610PROCcontinue
  620CLS
  630PRINTTAB(0,5);"The first cross is: ""FRIZZLE, WHITE x Pure NORMAL, COLOURED""(Both pure-breeding)"
  640
  650delay=INKEY(300)
  660PRINTTAB(0,15);"The result of this cross is: ""ALL F1 progeny are FRIZZLE, WHITE"
  670PROCcontinue
  680ENDPROC
  690REM
  700REM
  710DEFPROCcontinue
  720VDU23,1,0;0;0;0;
  730PRINTTAB(3,23);"Press spacebar to continue"
  740REPEATUNTILGET=32
  750ENDPROC
  760REM
  770REM

```

```

780DEFPROCtestcross_or_intercross
790REPEAT
800CLS
810PRINTTAB(0,5);"Testcross or intercross
of F1?"'"'"For testcross      Type 1'"'"For
intercross      Type 2'"'"'"
820VDU23,1,1;0;0;0;
830INPUT"Which cross",cross_type
840UNTIL cross_type=1 OR cross_type=2
850
860VDU23,1,0;0;0;0;
870CLS
880IF cross_type=1 THEN PRINTTAB(0,5);"Te
stcross:  F1 x NORMAL, COLOURED" ELSE PRINT
TAB(5,5);"Intercross:  F1 x F1"
890PROCcontinue
900
910IF cross_type=1 THEN PROCdetermine_pro
geny("FRZ,WHT","FRZ,COL","NOR,WHT","NOR,COL
",.4,.5,.6) ELSE PROCdetermine_progeny("FRZ
,WHT","FRZ,COL","NOR,WHT","NOR,COL",.66,.75
,.84)
920ENDPROC

```

LINK-variables

answer\$	user response to program prompts
f1\$	first letter of answer\$
a\$, b\$, c\$, d\$, x1, x2, x3	formal parameters of procedure determine.progeny
total1, total2, total3, total4	number of progeny of each of four possible phenotypes
K%, I%	loop control variables
R	random number between 0 and 0.999999
delay	delay in program execution
cross.type	type of cross, 1 for intercross, 2 for testcross

SEXL

A simulation of investigations into the genetics of the sex-linked phenotype, white eye in *Drosophila*.

Biological background

Sex determination in higher animals usually involves the so-called sex chromosomes. In mammals, and in *Drosophila*, these are called X and Y chromosomes. The X chromosome is usually larger than the Y chromosome. It is the presence of the Y chromosome which determines maleness (5). Males have one X and one Y chromosome, XY; males are heterogametic. Females have two X chromosomes, XX; females are homogametic.

X chromosomes appear to behave as normal non-sex chromosomes, autosomes. They carry alleles which can determine phenotype. The phenotypes controlled by alleles on sex-chromosomes, sometimes called sex-linked alleles, are not necessarily to do with sex. A mutant allele on the X chromosome of *Drosophila* can produce the white eye phenotype. Wild-type animals have bright red eyes. The strange thing about Y chromosomes is that they appear to carry no genetic information available to the cell; they have no alleles.

Hence, whilst females can be homozygous or heterozygous for a pair of sex-linked alleles, as is the case with autosomes, the terms homo- and heterozygous cannot be applied to males. They only have one allele for each sex-linked phenotype, not two.

Thus sex-linked characters show a deviation from Mendel's first law.

It is a feature of sex-linked characters that they often show themselves more often in males than in females. The inherited conditions haemophilia and colour-blindness in man are determined by sex-linked recessive alleles.

If these alleles are present on the X chromosome of a human male, then he will show these conditions. However, a female may have the recessive genes on one of her X chromosomes, but will not show the corresponding phenotype if a dominant normal allele is found at the same locus on her other X chromosome. Such heterozygous females are said to be carriers of sex-linked conditions.

White eye in *Drosophila* is a sex-linked condition, the white allele being recessive to the normal allele. A cross between a heterozygous female and a phenotypically normal male gives rise to a progeny in which males have probability .5 of having white eyes. In the same progeny, no females will have white eyes, but they have probability .5 of being carriers.

The computer program

The program SEXL was written to enable students to investigate the genetics of white eye in *Drosophila* (6).

Program description

As in the other two programs in this classical genetics section, SEXL is written around the procedure **determine-progeny**. In SEXL the procedure occupies lines 610 to 800.

A brief introduction, procedure **introduction-and-start**, lines 830 to 970 enables the user to set up the first cross by assigning the value 1 or 2 to the variable **first-cross**. This variable is then used to print a title page for the cross, line 140, and to call the procedure **determine-progeny**, line 150, assigning appropriate values to the formal parameters of that procedure.

A full description of the working of the procedure **determine-progeny** can be found under MENDEL in this chapter.

On exiting **determine-progeny**, control passes to line 200, and thence to line 220, from where the user can choose the phenotype of a male to cross to an F1 carrier female. The variable **F1-cross** thus assigned is used to pass the appropriate variables to the formal parameters of the procedure **determine-progeny** in line 320.

The user can then choose to cross an F1 female to the other male, lines 380 to 480. If this offer is rejected in line 400, control passes to line 500 from line 430. The user is then offered the option of another run of the program, or the program can be terminated.

Using the program

A class of students can be introduced to the unexpected nature of sex linkage by setting up the appropriate crosses, and led to make testable hypotheses based on data obtained from the program.

They might be asked to suggest reasons why the results of a cross between a phenotypically normal male and an F1 female gives rise to white eyed females. Ideas can be tested by crossing the F1 female to a white eyed male.

SEXL Listing

```
10REM SEXL
20REM
30REM Introduction and start
40REM
50*FX200,1
```



```

60MODE7
70VDU23,1,0;0;0;0;
80PROCintroduction_and_start
90REM
100REM Determine progeny of first cross
110REM
120CLS
130VDU23,1,0;0;0;0;
140IFfirst_cross=1 THEN PRINTTAB(0,5);"Homozygous RED FEMALE x WHITE MALE" ELSE PRINTTAB(0,5);"Homozygous WHITE FEMALE x RED MALE"
150PROCcontinue(3,20)
160IF first_cross=1 PROCdetermine_progeny("Mal,wht","Mal,red","Fem,wht","Fem,red",0,.5,.5) ELSE PROCdetermine_progeny("Mal,wht","Mal,red","Fem,wht","Fem,red",.5,.5,.5)
170REM
180REM Choose male for cross of F1 female
s
190REM
200VDU23,1,0;0;0;0;0;
210PROCcontinue(5,23)
220REPEAT
230CLS
240PRINTTAB(0,5);"F1 FEMALE x RED MALE (Type 1)"" WHITE MALE (Type 2)""
250VDU23,1,1;0;0;0;0;
260INPUT"Which cross",F1_cross
270UNTIL F1_cross=1 OR F1_cross=2
280CLS
290VDU23,1,0;0;0;0;0;
300IF F1_cross=1 PRINTTAB(3,5);"F1 FEMALE x any RED MALE" ELSE PRINTTAB(3,5);"F1 FEMALE x any WHITE MALE"
310PROCcontinue(3,20)
320IF F1_cross=1 PROCdetermine_progeny("Mal,wht","Mal,red","Fem,wht","Fem,red",.25,.5,.5) ELSE PROCdetermine_progeny("Mal,wht","Mal,red","Fem,wht","Fem,red",.25,.5,.75)

```

```

330VDU23,1,0;0;0;0;
340PROCcontinue(5,23)
350REM
360REM Choose other male for cross to F1
female
370REM
380REPEAT
390CLS
400PRINTTAB(0,5);:INPUT"F1 FEMALE x other
MALE",answer$
410UNTIL LEFT$(answer$,1)="Y" OR LEFT$(an
swer$,1)="y" OR LEFT$(answer$,1)="N" OR LEF
T$(answer$,1)="n"
420CLS
430IF LEFT$(answer$,1)="N" OR LEFT$(answe
r$,1)="n" THEN 500
440IF F1_cross=2 PRINTTAB(3,5);"F1 FEMALE
x any RED MALE" ELSE PRINTTAB(3,5);"F1 FEM
ALE x any WHITE MALE"
450PROCcontinue(3,20)
460IFF1_cross=2 PROCdetermine_progeny("Ma
l,wht","Mal,red","Fem,wht","Fem,red",.25,.5
,.5) ELSE PROCdetermine_progeny("Mal,wht","
Mal,red","Fem,wht","Fem,red",.25,.5,.75)
470IF F1_cross=1 THEN F1_cross=2 ELSE F1_
cross=1
480GOTO330
490REM
500REM Other run option
510REM
520CLS
530PRINTTAB(5,10);:INPUT"Another run",ans
wer$
540IF LEFT$(answer$,1)="Y" OR LEFT$(answe
r$,1)="y" THEN 120
550CLS
560PRINTTAB(10,10);"END"
570*FX200,0
580END
590REM
600REM

```

```

610DEFPROCdetermine_progeny(a$,b$,c$,d$,x
1,x2,x3)
620CLS
630VDU23,1,0;0;0;0;
640PRINTTAB(0,5);a$,b$,c$,d$
650total1=0
660total2=0
670total3=0
680total4=0
690PRINT
700FORK%=1TO100
710R=RND(1)
720IF R<x1 THEN total1=total1+1 ELSE IF R
<x2 THEN total2=total2+1 ELSE IF R<x3 THEN
total3=total3+1 ELSE total4=total4+1
730NEXTK%
740PRINT;total1,;total2,;total3,;total4
750REPEAT
760INPUT"More",answer$
770IF LEFT$(answer$,1)<>"Y" AND LEFT$(ans
wer$,1)<>"y" AND LEFT$(answer$,1)<>"N" AND
LEFT$(answer$,1)<>"n" PRINT"Type Y or N"
780UNTIL LEFT$(answer$,1)="Y" OR LEFT$(an
swer$,1)="y" OR LEFT$(answer$,1)="N" OR LEF
T$(answer$,1)="n"
785IF LEFT$(answer$,1)="Y" OR LEFT$(answe
r$,1)="y" THEN 650
790PRINT
800ENDPROC
810REM
820REM
830DEFPROCintroduction_and_start
840FORI%=8TO9
850PRINTTAB(12,I%);CHR$&8D;"SEXL"
860NEXTI%
870delay=INKEY(200)
880CLS
890PRINTTAB(0,5);"This program enables yo
u to""investigate the genetics of""whit
e eye in Drosophila."
900PROCcontinue(0,20)

```

```

910REPEAT
920CLS
930VDU23,1,1;0;0;0;
940PRINTTAB(0,5);"The first cross is: ""
Pure RED FEMALE X WHITE MALE (Type 1)""OR
""Pure WHITE FEMALE x RED MALE (Type 2)""
""
950INPUT"First cross",first_cross
960UNTIL first_cross=1 OR first_cross=2
970ENDPROC
980REM
990REM
1000DEFPROCcontinue(a,b)
1010PRINTTAB(a,b);
1020PRINT"Press space bar to continue"
1030REPEATUNTILGET=32
1040ENDPROC

```

SEXl variables:

first_cross	first cross set up by user.
F1_cross	F1 cross; 1 for red male, 2 for white male x F1 female
answer\$	user response to program prompts
a\$, b\$, c\$, d\$, x1, x2, x3, total1, total2, total3, total4	see MENDEL
K%, l%, R	as for MENDEL

SICKLE

This simulation permits investigations into the population genetics of sickle-cell anaemia.

Biological background

Sickle cell anaemia is an inherited disease of man. In extreme cases, the condition causes haemolytic anaemia to an extent that usually leads to early death. In some sufferers, the anaemia may be absent, although symptoms occur. The name sickle-cell anaemia comes from the unusual appearance of the red-blood cells of sufferers.

The genetics of the condition is quite simple. It involves a single locus, occupied by the gene HbA or its allele HbS. HbS is the mutant allele, and if present in the homozygous state leads to the the extreme form of the disease. Normal individuals are homozygous for the HbA allele. Heterozygotes, HbAHbS, show mild symptoms, sometimes referred to as sickle-cell trait (1).

The presence of the sickle allele, HbS confers selective disadvantage upon its possessors. Homozygous HbS individuals will have reduced fitness compared to individuals containing the HbA allele. HbSHbS individuals will have much less chance of surviving to reproductive age than heterozygotes or HbA homozygotes. Thus, in a randomly breeding population, the frequency of the HbS allele should decrease with time due to selection.

Consider a population or even better, a gene pool, in which the initial relative frequency of the deleterious HbS allele is q . Then the initial relative frequency of the HbA allele is $(1-q) = p$. Let $p = q$ initially.

Then, assuming random mating, with each generation of breeding, the relative frequency of HbS will decline, and that of HbA increase due to selection, until $p \gg q$; q may never quite reach zero, due to heterozygotes, which may not be exposed to selection. Still, a directional change in the composition of the gene pool will occur due to selection, the result being a population of well adapted organisms. The population can be said to have evolved.

The selective disadvantage conferred by the HbS allele would be expected to result in an extremely low relative frequency for it in all human populations. Whilst this is true for most human populations living away from the tropics, it is not true for those populations living in regions where falciparum malaria is endemic. The reason for this apparent anomaly is that the sickle cell allele confers malarial resistance on its possessors. Such malarial resistance can be of little use to HbSHbS individuals, who will still succumb to sickle-cell anaemia, but its presence in heterozygotes means that their fitness is greater than that of homozygous HbA individuals. Heterozygotes have selective advantage compared to both homozygotes.

This selective situation will lead to the maintenance of quite a high, relatively invariable relative frequency of the HbS allele in geographical regions of endemic

malaria. If q is stable, then so is p . Out of this genotypic stability arises phenotypic stability, the proportions of the three phenotypes remaining relatively stable from one generation to the next. This phenomenon is referred to as stable polymorphism, and arises due to stabilising selection (7,8).

It can be shown that at equilibrium, the value of p and q are as follows:

$$p = s/(s + t)$$

$$q = t/(s + t) \quad (1,8)$$

where t and s represent selective coefficients used to measure the decrease in fitness suffered by HbAHbA and HbSHbS individuals respectively (9).

The following table summarises the situation.

Genotype	Phenotype	Frequency	Fitness
HbAHbA	normal	p^2	$1 - t$
HbAHbS	mild sickle	$2pq$	1
HbSHbS	sickle	q^2	$1 - s$

In the table, fitnesses are quoted for a region of endemic malaria. Where malaria is not endemic, the fitness of HbAHbA would be greater, approximately 1. The values of t and s have a range from 0 to 1. Values of 1 mean that the genotype is lethal to all intents and purposes. This is probably the case for the HbS homozygotes.

Stable polymorphisms are very common in biological systems; industrial melanism in the peppered moth, *Biston betularia*, and banding in the snail *Cepea nemoralis* are phenomena maintained, at least in part, by stabilising selection. Both are commonly quoted in O and A level courses (10, 11).

The computer program

The program SICKLE was written to enable students to investigate the effects of disruptive or directional selection, and stabilising selection, on a theoretical human gene pool. Students are able to define the initial composition of the gene pool by specifying the relative frequencies of HbS and HbA alleles. They can then specify selective coefficients for each genotype. The program then prints out the values of p ($F(\text{HbA})$) and q ($F(\text{HbS})$) both before and after selection.

Program description

The program begins with a title page, line 110 to 170. This is followed by a brief statement of the purpose of the program, lines 180 to 230. The user is then requested to specify the initial frequency of the HbS allele, variable HbS. The value of this is checked to lie between 0 and 1 (lines 280 to 340). If the user's value for HbS is valid, the program calculates and prints the corresponding value of p , the frequency of the HbA allele in the gene pool, variable name HbA, line 350 to 360.

In lines 420 to 480, the user is asked to specify selective coefficients against each of the three genotypes. These are the variables selDOMHOM, selHET and selRECHOM for each of HbAHbA, HbAHbS and HbSHbS respectively.

The values of these variables are required to lie between 0 and 1 inclusive.

Lines 540 to 580 print headings for the results table. Mode 3 is used for neatness. Note that Mode 3 is not available on A machines.

The calculation of the frequencies of the two alleles HbA and HbS is done in the procedure **calcnewfrequencies**, running from line 1110 to 1170. The procedure is called twice for each generation of breeding, once before and once after selection, lines 680 and 740 respectively. The procedure calculates the values of allele frequency to 5 places of decimals, see line 1160 for example.

Running from line 630 to line 780 is a FOR....NEXT loop which uses the control variable K. The value of this variable is that of the current generation. Its value is set to 1 in line 120, and is then incremented by 9 for each run of the loop. If more generations of selection are chosen by the user in lines 800 to 830, the current value of K is increased by 10 and the loop begun again. In this way, results of 10 generations of selection are printed at any one time on the screen, and the user can choose as many generations of selection as he wishes.

Using the program

The program can be used to demonstrate that selection of different types can produce changes in the relative frequencies of alleles in gene pools.

Directional selection leads to a unidirectional change in gene frequencies, stabilising selection results in invariable gene frequencies, the equilibrium point being reached from whatever starting values of allelic frequency are chosen.

Consider the following runs of SICKLE. In the first run, the user may choose a value of the frequency of HbS, $F(\text{HbS})$, to be .5.

The population under consideration lives in Aberdeen (!) so when selective coefficients are chosen they might be 0 for HbAHbA, 0 for HbAHbS and 1 for HbSHbS. Directional, or disruptive selection has been chosen. The program then generates the frequencies of HbA and HbS for 10 generations, giving figures before and after selection. By generation 10, after selection, the value of p, i.e. $F(\text{HbA})$ rises to 0.91667 from its initial value of 0.5, whilst the value of q has fallen to .08333, printed by the computer in exponential notation as $8.333\text{E-}2$, 8.333×10^{-2} . After 100 generations of selection, $F(\text{HbA})$ has risen to 0.99019, $F(\text{HbS})$ has fallen to 0.00981 ($9.81\text{E-}3$), and after 200 generations, the corresponding figures are 0.99506 and 0.00494. The rate of decline of q, $F(\text{HbS})$ is decreasing with time.

There are fewer recessive homozygotes as time passes, and the HbS gene is hiding in heterozygotes.

In a second run, the user may use the same initial gene frequencies, 0.5 for HbS and 0.5 for HbA, but may choose to locate the gene pool in Sukuta in the Gambia, where malaria is endemic. He could choose a selective coefficient of .3 for the HbA homozygotes, 0 for heterozygotes as they have malarial resistance conferred by the HbS allele, and total lethality, selective coefficient 1 for recessive homozygotes. After 10 generations of selection, the relative frequencies of the HbA and HbS alleles are 0.76567 and 0.23433 respectively. After 26 generations, the frequencies of the two alleles have become stable at 0.76922 and 0.23078. Stabilising selection has produced a stable polymorphism. Another run using stabilising selection, using the same selective coefficients could start with an initial frequency of HbS of 0.1. After 10 generations of selection, the frequency of the HbS allele has risen(!) to 0.22255, and continues to rise to an equilibrium value 2×10^{-5} greater than that in the previous run. The equilibrium gene pool frequencies are independent of initial gene frequency.

The theoretical equilibrium values for p and q, using the expressions quoted above, are 0.76923 and 0.23077 respectively. So the model used in SICKLE gives a good approximation to reality.

The program can thus be used to show the effects of selection on the composition of a gene pool and to demonstrate that selective change is environmentally dependent.

SICKLE Listing

```

10REM SICKLE
20REM
30REM SICKLE CELL ANAEMIA POPULATION GENETICS
SIMULATION
40REM
50REM TURN OFF ESCAPE KEY
60*FX200,1
70REM
80REM
90REM SETUP AND INTRODUCTION
100REM
110MODE7
120K=1
130VDU23,1,0;0;0;0;0;
140FOR I%=8TO9
150 PRINTTAB(17,I%);CHR$&8D;"SICKLE"
160NEXT
170delay=INKEY(250)
180VDU12

```



```

190PRINTTAB(0,15);"This program enables you to
investigate"
200PRINT"the population genetics of sickle-"
210PRINT"cell anaemia."
220delay=INKEY(300)
230PROCcontinue
240REM
250REM
260REM INPUT STARTING DATA
270REM
280REPEAT
290VDU12
300PRINTTAB(5,10);:INPUT"Initial frequency of
HbS allele",HbS
310IF HbS>=0 AND HbS<=1 THEN 340
320PRINTTAB(8,20);"Invalid input-try again"
330delay=INKEY(250)
340UNTIL HbS>=0 AND HbS<=1
350PRINTTAB(5,12);"Initial frequency of HbA is
";1-HbS
360HbA=1-HbS
370PROCcontinue
380REM
390REM
400REM CHOOSE SELECTIVE COEFFICIENTS
410REM
420VDU12
430PRINTTAB(5,10);:INPUT"Selection against HbA
HbA",selDOMHOM
440IFselDOMHOM<0 OR selDOMHOM>1 THEN PRINTTAB(
8,20);"Invalid input-try again":delay=INKEY(250)
:PRINTTAB(0,20);"
":PRINTTAB(29,10);"":GOTO430
450PRINTTAB(5,12);:INPUT"Selection against HbA
HbS",selHET
460IFselHET<0 OR selHET>1 THEN PRINTTAB(8,20);
"Invalid input-try again":delay=INKEY(250):PRINT
TAB(0,20);"":PRIN
TTAB(29,12);"":GOTO450
470PRINTTAB(5,14);:INPUT"Selection against HbS
HbS",selRECHOM

```

```

480IFselRECHOM<0 OR selRECHOM>1 THEN PRINTTAB(
8,20);"Invalid input-try again":delay=INKEY(250)
:PRINTTAB(0,20);"
":PRINTTAB(29,14);"                                ":GOTO470
490PROCcontinue
500REM
510REM
520REM SET UP TABLE FOR RESULTS PRINTOUT
530REM
540MODE3
550VDU23,1,0;0;0;0;0;
560PRINTTAB(10,5);" ", "Before", " ", "After"
570PRINTTAB(10,6);" ", "selection", " ", "selecti
on"
580PRINTTAB(10,8);"Gen", "F(HbA)", "F(HbS)", "F(H
bA)", "F(HbS)"
590REM
600REM
610REM CALCULATE NEW ALLELE FREQUENCIES
620REM
630FORJ=K TO K+9
640counter=counter+1
650H1=HbA^2
660H2=2*HbA*HbS
670H3=HbS^2
680PROCcalcnewfrequencies
690W=HbA
700X=HbS
710H1=H1*(1-selDOMHOM)
720H2=H2*(1-selHET)
730H3=H3*(1-selRECHOM)
740PROCcalcnewfrequencies
750Y=HbA
760Z=HbS
770PRINTTAB(10,9+counter);J,;W,;X,;Y,;Z
780NEXT
790counter=1
800INPUT"More",A$
810IF LEFT$(A$,1)="N" OR LEFT$(A$,1)="n"THEN88
0
820delay=INKEY(100)

```

```

825K=K+10
830GOTO540
840REM
850REM
860REM PROGRAM CONTINUATION OPTION
870REM
880MODE7
890VDU23,1,0;0;0;0;
900PRINTTAB(5,10);:INPUT"Another run",A$
910K=1
920IF LEFT$(A$,1)="Y" OR LEFT$(A$,1)="y" THEN
280
930REM
940REM
950REM END OF PROGRAM SEQUENCE
960REM
970VDU12
980PRINTTAB(10,10);"END"
990*FX200,0
1000END
1010REM
1020REM
1030REM
1040DEFPROCcontinue
1050PRINTTAB(6,23);"Type space bar to continue"

1060REPEATUNTILGET=32
1070ENDPROC
1080REM
1090REM
1100REM
1110DEFPROCcalcnewfrequencies
1120HbA=((H1*2)+H2)/((H1+H2+H3)*2)
1130HbS=((H3*2)+H2)/((H1+H2+H3)*2)
1140 HbA=HbA+.000005
1141HbS=HbS+.000005
1150HbA=(INT(HbA*100000))/100000
1160HbS=(INT(HbS*100000))/100000
1170ENDPROC

```

SICKLE-variables:

I%	control variable of loop generating enhanced title
K	counter for generations of selection
delay	delay in program execution
HbS	frequency of HbS allele
HbA	frequency of dominant HbA allele
selDOMHOM	fraction of normal individuals not surviving to reproduce
selHET	fraction of heterozygous individuals not surviving to reproduce
selRECHOM	fraction of homozygous recessive individuals not surviving to reproduce
counter	counter used in printout formatting
H1	HbAHbA genotype frequency
H2	HbAHbS genotype frequency
H3	HbSHbS genotype frequency
W	frequency of HbA allele before selection
X	frequency of HbS allele before selection
Y	frequency of HbA allele after selection
J	control variable of generation loop
A	user response to program continuation offer

DRIFT

This is stochastic simulation of genetic drift.

Biological background

The relative frequencies of alleles in gene pools change due to the operation of selection. This phenomenon can be demonstrated using the program SICKLE. One of the assumptions upon which the mathematical treatment of such changes is based is that the size of the population under consideration is infinite, or certainly very large. If this is not the case, then changes in the relative frequencies of alleles may occur through sampling error (12). These changes in gene frequency are random and may lead to the loss of an allele from a small population, with the corresponding fixation of other alleles. This phenomenon is called genetic drift. It is of great evolutionary significance in small populations, such as may occur on small islands, or in residual populations such as that of the whooping crane, which in 1970 had a UK population of 57(12). Studies of human populations have revealed that the phenomenon of drift has a substantial role in human evolution, see Cavalli-Sforza(13).

Consider a population of N individuals which form a randomly breeding isolated group. Further consider any two allele Mendelian system. Then if all the individuals of the group contribute gametes to the populations' gamete pool, $2N$ gametes are used to produce each generations' N individuals.

Consider a gene a and its allele A . Let the frequency of a in the gene pool be q , and that of A be $1 - q = p$. We are interested in changes in q , designated Δq . The mean of Δq will always be zero as long as mating is truly random; however, the variance of Δq is given by that of the binomial (14),

$$\text{var}(\Delta q) = pq/(2N)$$

So, as N becomes large, $\text{var}(\Delta q)$ becomes small. When N is greater than 100,000 the potential for gene loss through drift is negligible. However, if N is less than 100, $\text{var}(\Delta q)$ is relatively large, and random fluctuations in q , genetic drift, are large.

It has been suggested that the time to fixation for any allele is roughly $4N$ generations (12).

The computer program

The program DRIFT is a simulation that allows the user to set up an initial population of between 10 and 100 individuals. The population is stable with time and subject to no selection, mutation or migration. The program generates data from which it plots a graph of the variation of the relative frequency, q of an allele with time. A Monte-Carlo simulation is employed, gametes being chosen from the gene pool purely at random to produce the next generation.

Program description

The program begins with a title page, procedure **header** called from line 80 and occupying lines 820 to 910. Lines 830 to 850 set the background colour to cyan; line 870 sets the text colour to red. Line 90 clears the screen, setting teletext mode, and at line 100, the procedure **start-values** is called.

This procedure, occupying lines 940 to 1060, first of all allows the user to choose the population size for the population under consideration. The value of the variable **pop**, used to store this value, must be between 10 and 100 inclusive (lines 900,1000). If the value input is not in this range, a diagnostic message is printed in line 990, and after a short delay, the screen is cleared by passing control back to line 960, and the input is requested again. Lines 1010 to 1050 request the user to input the initial frequency of the recessive allele, **q**, error-trapped to be within the range 0 to 1 exclusive. On **ENDPROC** in line 1060, control passes to line 110, from where the procedure **set-up** is called.

This procedure, running from line 1090 to 1160 uses the values of **pop** and **q** to assign the value 1 or 2 to the elements of the array **allele** (). This array is dimensioned to 200, corresponding to a maximum value of **pop** of 100. Its elements represent the gametes used to generate the next generation of **pop** individuals. An a gamete is represented as an array element of value 1, an A gamete as a value of 2.

Lines 160 to 500 draw the axes for the results graph, and also draw in a horizontal line for a value of **q** of 1, and a red line for a value of .5.

The axes are labelled using **VDU5** to print text at the graphics cursor and the variable **X**, increased by steps of 40 each 40 generations. The graphics cursor is moved to coordinates representing the initial value of **q** at generation zero in line 540, and the current graphics colour set to yellow in line 550. The graph will thus appear in yellow, the axes being white.

The **REPEAT...UNTIL** loop at lines 580 to 660 generates the data used to plot the graph. First, the counter **K** is incremented by 1. This variable is used to stop the loop if its value reaches 40 (line 660). Next, the procedure **mate** is called from line 600. This procedure, occupying lines 1300 to 1390 begins by setting all the elements of the array **flag** () to zero. It then chooses **pop** values at random and sets the corresponding value of the array to 1. Line 1360 ensures that no element is chosen twice. Control then returns to line 610 from where the procedure **translate-flags** is called.

This procedure, occupying lines 1420 to 1490 determines the new value of **q** in the new generation. The loop starting at line 1440 looks through the array **flag** (), and if an element is found with a value of 1, **q** is incremented by 1 if the corresponding value of **allele** () is also 1. In this way, a new value for **q** is generated, which, when passed to line 650 is used to produce the next section of the graph. Before this is done, the colour of the next plotting point is checked in line 640. If the colour is white, the variable **flag** is set to 1, and this value is used to stop the **REPEAT...UNTIL** loop at line 660. This flag means that one of the alleles A or a has

reached fixation, and this information is conveyed to the user as a message in line 750. The program is then terminated. If the REPEAT...UNTIL loop is terminated by a value of K of 40, the user is given the option of continuing the run. If the user chooses continuation by responding with C RETURN at line 730, control passes to line 160, axes are redrawn and labelled and more values of q generated and plotted.

Using the program

Students of A level biology are rarely convinced by a quantitative argument that genetic drift is a real phenomenon. However, they are more easily convinced if the mathematics follows a visual demonstration made possible with runs of DRIFT.

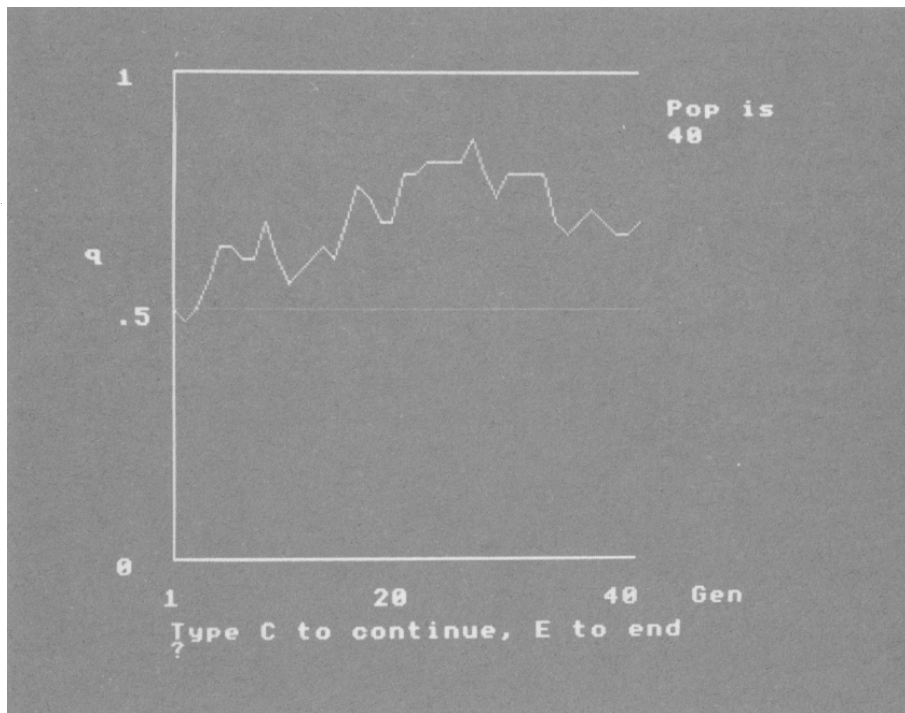


Figure 3.1 Output from a run of Drift.

In a series of runs of DRIFT, an initial population of 15 with q initially 0.5 reached fixation at generations varying from 8 to 71, with a mean of 30 (8 repetitions). A population of 100 individuals, q initially 0.5 reached fixation at generation numbers 98 to 383 with a mean of 252 (4 repetitions).

It is also apparent from the curves that the generation to generation variation of q is much greater with a population size of 15 than one of 100.

DRIFT Listing

```
10REM DRIFT
20REM
30REM Introduction and start
40REM
50DIMallele(200),flag(200)
60*FX200,1
70MODE1
80PROCheader
90MODE7
100PROCstart_values
110PROCset_up
120X=-39
130REM
140REM Set up graph and axes
150REM
160MODE1
170X=X+40
180VDU23,1,0;0;0;0;0;
190VDU29,200;200;
200MOVE0,800
210DRAW0,0
220DRAW800,0
230MOVE0,400
240GCOLOR,1
250DRAW800,400
260GCOLOR,3
270MOVE0,800
280DRAW800,800
290VDU5
300MOVE-100,800
310PRINT"1"
320MOVE-100,0
330PRINT"0"
340MOVE-100,400
350PRINT".5"
360MOVE-150,500
370PRINT"q"
```



```

380MOVE-20,-50
390PRINT;X
400MOVE350,-50
410PRINT;X+19
420MOVE750,-50
430PRINT;X+39
440MOVE900,-50
450PRINT"Gen"
460MOVE850,750
470PRINT"Pop is"
480MOVE850,700
490PRINT;pop
500VDU4
510REM
520REM Plot graph
530REM
540MOVE0,q*800
550GCOLOR,2
560flag=0
570K=0
580REPEAT
590K=K+1
600PROCmate
610PROCtranslate_flags
615 IF q=1 flag=1:GOTO 650
620PROCset_up
640IF POINT(K*20,q*800)=3 flag=1
650DRAWK*20,q*800
660UNTIL K=40 OR flag=1
670IF flag=1 THEN 750
680REM
690REM Continuation option
700REM
710PRINTTAB(6,29);
720PRINT"Type C to continue, E to end"
730PRINTTAB(6,30);:INPUTresponse$
740IF response$="C" THEN 160 ELSE IF response$
="E" THEN 760 ELSE PRINT"You must respond with C
or E":CLS:GOTO710
750PRINTTAB(6,29);"Fixation in generation ";K+
X

```

```

760*FX200,0
770delay=INKEY(200)
780PRINTTAB(17,30);"END"
790END
800REM
810REM
820DEFPROCheader
830COLOUR130
840VDU19,2,6,0,0,0,0
850CLS
860VDU23,1,0;0;0;0;0;
870COLOUR1
880PRINTTAB(5,10);"GENETIC DRIFT SIMULATOR"
890PRINTTAB(4,20);"Press spacebar to continue"
900REPEATUNTILGET=32
910ENDPROC
920REM
930REM
940DEFPROCstart_values
950REPEAT
960CLS
970VDU23,1,0;0;0;0;0;
980PRINTTAB(5,10);:INPUT"Population size",pop
990IF pop<>INT(pop) OR pop<10 OR pop>100 THEN
PRINTTAB(5,20);"Out of range-try again":delay=IN
KEY(200)
1000UNTIL pop=INT(pop) AND pop>=10 AND pop<=100
1010REPEAT
1020CLS
1030PRINTTAB(5,10);:INPUT"Frequency of a allele
",q
1040IF q<=0 OR q>=1 THEN PRINTTAB(5,20);"Out of
range-try again":delay=INKEY(200)
1050UNTIL q>0 AND q<1
1060ENDPROC
1070REM
1080REM
1090DEFPROCset_up
1100FORI%=1 TO INT(q*pop*2+.5)
1110allele(I%)=1
1120NEXTI%

```

```

1130 FORI%=q*pop*2+1 TO pop*2
1140 allele(I%)=2
1150NEXTI%
1160ENDPROC
1170REM
1180REM
1300DEFPROCmate
1310 FORI%=1TOpop*2
1320flag(I%)=0
1330NEXTI%
1340FORJ%=1TOpop
1350gamete=RND(pop*2)
1360IF flag(gamete)=1 THEN 1350
1370flag(gamete)=1
1380NEXTJ%
1390ENDPROC
1400REM
1410REM
1420DEFPROCtranslate_flags
1430q=0
1440FORI%=1TOpop*2
1450IF flag(I%)<>1 THEN 1470
1460IF allele(I%)=1 q=q+1
1470NEXTI%
1480q=q/pop:p=1-q
1490ENDPROC

```

DRIFT-variables:

X	counter used to print numbers to generation scale of axes of graph
q	value of relative frequency of a allele
flag	set to one if allele reached fixation
K	counter used to terminate REPEAT...UNTIL loop when value is 40: graph is printed in segments 40 generations long
response\$	user response to continuation option

delay	delay in program execution
pop	population size
allele(n)	array of elements representing gamete pool
flag(n)	array used to choose gametes from gamete pool for random mating
I%, J%	loop control variables
gamete	random integer in the range 0 to pop x 2, used to choose gametes at random from gamete pool

References

1. Srb, A. M., Owen, R. D., and Edgar, R. S., General genetics, Freeman, San Francisco, 2nd Ed., 1965.
2. Bronowski, J., The ascent of man, BBC, London, 1973.
3. Popper, K., The logic of scientific discovery, Hutchinson, London, 3rd. Ed., 1972.
4. Morgan, T. H., Evolution and genetics, Princeton University Press, Princeton, 1925.
5. Levine, R. P., Genetics, Holt, Rinehart and Winston, New York, 1962.
6. Shone, J., 'Genetics and the minicomputer', SSR, 62, (1980), 83-85.
7. Sheppard, P.M., Natural selection and heredity, Hutchinson, London, 3rd. Ed., 1967.
8. Lewontin, R. C., The genetic basis of evolutionary change, Columbia University Press, New York, 1974.
9. Cook, L. M., Coefficients of natural selection, Hutchinson, London 1971.
10. Nuffield Advanced Science. Biological Science. Study Guide., Penguin, Harmondsworth, 1970.
11. Nuffield Biology. Text V, The perpetuation of life, Penguin, Harmondsworth, 1967.

12. Wilson, E. O., *Sociobiology*, Harvard University Press, Cambridge, Massachusetts, 1975.
13. Cavalli-Sforza, L. L., 'Genetic drift in an Italian population', *Scientific American*, August 1969.
14. Mosteller, F., Rourke, R. E. K., and Thomas, G. B., *Probability with statistical applications*, Addison-Wesley, London, 2nd. Ed. 1970.

CHAPTER 4

BIOCHEMISTRY

Introduction

Many experiments in biochemistry are not available to students in schools and colleges. The problems tend to be such things as the availability of time or apparatus, or the complexity of the proposed investigations. These problems vanish if the experiments are translated into a simple computer model.

There is a wide range of experimentation in biochemistry that lends itself to the computer simulation approach, including biosynthesis, metabolic pathways research, enzyme kinetics. In this chapter, I have chosen two simulations to illustrate 'key-board' biochemistry.

One of them is a simulation providing data and the potential for student designed experimentation to enable the sequencing of a small oligopeptide stored in the computer. This program, OLIGO, allows students to conduct different series of experiments simultaneously, and so can be used on a minimum number of keyboards whilst still allowing a whole class access to the computer model.

The other program is a simulation of an aspect of molecular biology, an area where class based investigation of a practical nature is severely restricted.

FRAMES

This is a simulation based on the experiments of Crick et. al. (1) to determine the number of bases in a single DNA codon.

Biological background

The cracking of the genetic code has been one of the triumphs of modern biology. The genes contained in the nucleus of a cell are used by the cell as a source of information. They contain a message. The result of decoding the message, and acting upon the new message is the phenotype. A phenotype can be thought of as being in some sense isomorphic to genotype (2).

What a cell is, and what it does, depends upon the proteins that it contains. In particular, it depends upon the enzymes that the cell manufactures. The enzymes control the operation of the cell's biochemistry; they synthesise the compounds that constitute the chemical machinery of the cell. The enzymes produced by cells depend upon the genotype of the cell.

Genes contain the information needed by a cell to manufacture the appropriate enzymes. This is the one gene-one enzyme hypothesis, or better the one gene-one polypeptide hypothesis (3). The phenotype is, in one sense, the range of proteins made by a cell. The message contained in the genes is revealed through proteins (4). It is better to think of genes and proteins being isomorphic; one maps to the other.

Proteins consist at a primary level of a single chain of amino-acids. There are twenty or so naturally occurring amino-acids found in proteins.

Different proteins have different sequences of amino-acids. One of the smallest proteins which occur naturally, ribonuclease, has 124 amino-acids in total. In a protein, each amino-acid is called a residue. Many proteins are coiled into complex shapes. This is particularly true of the enzymes such as ribonuclease.

The shape of an enzyme determines what it can do biochemically. The shape is more important than the amino-acid sequence. However, the shape depends upon the amino-acid sequence (2,4). Thus, if a gene contains the necessary information needed to string together the right number of amino-acids in the right order, it also contains the information the sequence needs to arrange itself into the appropriate shape. The shape is implicit in the amino-acid sequence, just as the amino-acid sequence must be implicit in the gene message.

The question becomes: "What is the rule governing the isomorphism between proteins and genes?". How is the information required to assemble the sequence of residues of a protein coded into the 'aperiodic crystal' called a gene (5)?

Like proteins, genes are chemically polymers. They consist of a linear arrangement of similar subunits. Genes are parts of chromosomes, and the functional piece of a chromosome is a single molecule of DNA, deoxyribonucleic acid. Genes are composed of segments of DNA (6) and chromosomes are linear combinations of genes. The monomers of which the polymer DNA is made are called nucleotides. All nucleotides consist of a combination of three small molecules, a sugar, phosphoric acid and a base. The sugar in DNA is always 2-deoxy-D-ribose, a five carbon sugar, giving DNA its name. The only variation in nucleotides comes from the base. This is usually a purine, adenine or guanine, or a pyrimidine, cytosine or thymine. Nucleotides are often referred to by the first letter of the base that they contain as A, G, C or T. A remarkable property of these nucleotides is that the amounts of A and T are roughly the same in any DNA sample; the same is true of C and G (7). This, along with other, more telling evidence, led Watson and Crick (8) to propose that a DNA molecule consists of two polymers of nucleotides which spiral around each other in a double-helix. In higher cells only one of these polymers at a time carries information used by the cell to manufacture proteins with the

corresponding residue sequence.

A typical DNA sequence might be:

AGAATAAGGTAATGATGAGGAAGCGTTAAACACAAAAATAGAAGACGCACCCGTCTAGGG

The isomorphic amino-acid sequence in a protein would be:

Ser-Tyr-Ser-Ile-Thr-Thr-Pro-Ser-Gln-Phe-Val-Phe-Leu-Ser-Ser-Ala-Tyr-Ala-Asp-Pro

These are in fact, the first twenty residues in wild-strain TMV protein (9).

The rule that maps the protein to the DNA may not be obvious. Finding the rule involves breaking the genetic code.

It transpires that the DNA message above can be subdivided into pieces of information called codons. A single codon codes for the occurrence of a particular residue in a protein. The sequence of codons maps to the sequence of residues. Each codon is three nucleotides long. In the above sequences, AGA maps to the residue Ser, for serine; ATA maps to Tyr for tyrosine. The next codon, AGG, like the first, codes for the residue serine.

Serine has more than one codon. This is true for a number of amino-acids. The genetic code is said to be redundant.

There are 4^3 possible condons, so that sixty four codons code for twenty amino acids.

The evidence for the triplet nature of codons comes from Crick et. al. (1).

They showed that an integral multiple of three point deletion mutants in T4 bacteriophage caused little change in the ability of the 'phage to infest *Escherichia coli* cells. The principle is this. Consider a message,

CAT

Where the letters are read in groups of three, as in the genetic code. If this were a DNA message, it would imply an amino-acid sequence of poly-valine, the codon for valine being CAT. Now consider what happens if one letter is deleted from the message at random. For example, consider the message with the fourth letter deleted. The resultant message is split up into letter triplets for easier reading.

CAT.ATC.ATC.ATC.ATC.ATC., etc.

The message has changed. After the point of deletion, each codon is now ATC. In DNA "language" ATC 'means' 'end of sequence'!

What if two letters are randomly deleted from the first message? Consider the following in which letters 3 and 4 are deleted. Once again, the resultant message is printed in three letter groups.

CAA.TCA.TCA.TCA.TCA.TCA., etc.

Compared to the original message, this is nonsense. If a cell attempted to manufacture a protein using the information of the mutated message, the function of the protein would probably be totally different from that of the original one.

What if three letters are deleted at random from the original message? In the following message, letters 3, 4 and 11 have been removed.

CAA.TCA.TCT.CAT.CAT.CAT.CAT.,etc.

After the position of the last mutant, the original message returns. As long as the garbled sequence is relatively short, this last mutant, were it a DNA sequence, could code for a protein not too different from that produced using the original, unmutated DNA message, able to carry out its biological function.

Crick et. al. used proflavin to induce deletion mutants in T4 'phage. The deletion of a base from a DNA message is called a frame-shift mutation.

Chromosomes are very large intranuclear objects and ribosomes are rather large cytoplasmic objects. The DNA message containing the information needed to manufacture proteins, the protein 'blueprint', is found in chromosomes; the protein manufacturing sites are the cytoplasmic ribosomes. The synthesis of protein requires the expenditure of a good deal of energy.

It would require more if DNA had to be moved to ribosomes to enable its blueprint to be read, or if ribosomes had to be moved to chromosomes. What actually happens is that small sections of the DNA message are complementarily copied as messenger RNA (m-RNA) in the nucleus under the control of the enzyme RNA-polymerase (3). This process is called transcription. RNA is a nucleotide polymer. However, unlike DNA, molecules of RNA are single stranded. They also contain nucleotides built using D-ribose and the bases adenine, cytosine, guanine and uracil. Uracil replaces thymine, and binds complementarily to adenine.

Transcription of the DNA sequence near the beginning of this section would give rise, using the complementarity rule, to the following m-RNA message:

UCUUAUCCAUAUACUACUCCUUCGCAUUUGUGUUUUUAUCUUCUGCGUGGGCAGAUCC

Hence the first DNA codon, AGA is isomorphic to the first m-RNA codon, UCU due to complementary association of C with G, and U with A. The genetic code, as written in DNA codons, can also be expressed in RNA codons, complementary and isomorphic to the DNA ones. Indeed, ribosomes translate the message for protein manufacture using m-RNA.

Mnemonics for the naturally occurring amino-acids and the RNA version of the genetic code are given below.

Amino-acid mnemonics

Ala	alanine	Arg	arginine
Asn	asparagine	Asp	aspartate
Cys	cysteine	Gln	glutamine
Glu	glutamate	Gly	glycine
His	histidine	Ile	isoleucine
Leu	leucine	Lys	lysine
Met	methionine	Phe	phenylalanine
Pro	proline	Ser	serine
Thr	threonine	Trp	tryptophan
Tyr	tyrosine	Val	valine

Genetic code for RNA codons

First base	Second base			Third base	
	A	C	G	U	
A	Lys	Thr	Arg	Ile	A
	Asn	Thr	Ser	Ile	C
	Lys	Thr	Arg	Met	G
C	Asn	Thr	Ser	Ile	U
	Gln	Pro	Arg	Leu	A
	His	Pro	Arg	Leu	C
	Gln	Pro	Arg	Leu	G
	His	Pro	Arg	Leu	U
G	Glu	Ala	Gly	Val	A
	Asp	Ala	Gly	Val	C
	Glu	Ala	Gly	Val	G
	Asp	Ala	Gly	Val	U
	END	Ser	END	Leu	A
U	Tyr	Ser	Cys	Phe	C
	END	Ser	Trp	Leu	G
	Tyr	Ser	Cys	Phe	U

Codons UAA, UGA and UAG are termination codons marking spaces between separate sections of the corresponding DNA message.

The computer program

The program FRAMES is designed to enable the user to simulate an experiment based on that of Crick et. al. (1) to determine the number of nucleotides in a DNA codon. The computer stores an RNA message which, when translated into an amino-acid sequence, gives rise to an oligopeptide of twenty amino-acids corresponding to the first twenty residues of wild-strain TMV protein (9).

The user can choose to apply mutagens to the DNA sequence isomorphic to the stored RNA sequence. The mutagens, modelling the effects of proflavin, cause point deletion frame-shift mutants of the DNA. One to nine point deletions can be chosen. The corresponding RNA sequence is then translated into an oligopeptide, the residue sequence of which is printed to the screen. This procedure is applied ten times, producing ten mutant sequences which can be compared to the original, unmutated sequence.

Program description

The program begins by loading amino-acid mnemonics into a four by four by four array. This array, a cube of side 4, has sixty-four elements loaded in lines 170 to 220 from data occupying lines 90 to 120. Loop control variable I represents the first base, J the second and K the third, storing the mnemonics in the genetic code cube according to the genetic code table given above. The genetic code cube is the array `cube$(i,j,k)`. As with all arrays of dimension more than one, the cube occupies a lot of memory in the computer; so much so that the program will not run on model A machine.

Lines 270 and 280 assign values to `parentPROTEIN$` and `parentRNA$`. The value of `parentRNA$` is the m-RNA sequence which is isomorphic to the sequence of residues represented by `parentPROTEIN$`.

Lines 330 to 480 are a title page and introduction. Lines 350 to 370 print the program header in enhanced characters. The screen is then cleared after a delay and a brief introduction printed in lines 440 to 460. A lot of information is printed to the screen. Screenfuls of information can be very intimidating to students, so the delay in line 450 paces the presentation of the information. More delays could be introduced, or the lengths of delays changed by varying the value of the argument of the `INKEY` statement.

On choosing to continue program execution in line 840, the user is presented with more information, lines 530 to 590. The value of `parentRNA$`, the original m-RNA sequence is printed to the screen, line 550. This is followed by the corresponding oligopeptide residue sequence, produced by calling the procedure **translate** using `parentRNA$` as its actual parameter.

Mode O is chosen for the printout to allow the complete sequence of the parent protein to be printed to the screen on the same line.

On choosing to continue program execution in line 590, the user is asked to choose the number of point deletion mutants to be made in the parental DNA (line 640 to 700). In fact, the deletions will be made in the original m-RNA string variable.

In lines 750 to 840, the parental protein is again printed to the screen using `parentRNA$` as the actual parameter for the procedure **translate** (lines 770, 780). This is then followed by ten mutant oligopeptides. These are produced by

calling the procedure **mutantsequence** from line 810. This procedure has two formal parameters, the actual values of which are **parentRNA\$** and **mutations%**.

The latter variable is the number of point deletion mutants chosen by the user. Ten calls are made to this procedure from the **FOR....NEXT** loop running from line 800 to 820. Printout from this part of the program is again in mode O for compactness.

This means that a monitor is really needed, the definition of an ordinary television screen being too poor to allow mode O characters to be seen easily.

On choosing to continue the program at line 840, the user is presented with a menu of options. The user can choose to repeat the program with the same number of point deletion mutants or he can choose a different number of mutations. The program can be terminated or restarted (lines 890 to 950).

If program termination is chosen, the end program option of lines 1000 to 1040 is entered. *FX200, 0 re-enables **ESCAPE**, and the program terminates in line 1040.

The procedure **translate** occupies lines 1080 to 1210. It has formal parameter **source\$**. To translate **source\$** into a residue sequence, **source\$** must be chopped into codons. **source\$** will be a string of bases; codons are three-base segments of **source\$**. The codons are read from **source\$** in line 1110. They are produced by taking substrings of **source\$** starting at every fourth character in **source\$**. Each substring has a length of 3 characters and begins at position **position** in **source\$**. The variable **position** is the control variable of the loop running from line 1100 to 1190. Line 1100 ensures that the values of **position** are 1, 4, 7... to a value such that no substring of **source\$** can have a logical length other than 3. The keyword **MOD** returns the value of the remainder left after integer division. For example, 7 MOD 2 is 1, as 7 divided by 2 gives remainder 1. The use of **MOD** in line 1100 means that characters can be read from **source\$** only up to an integer multiple of three. Using the value of **position**, the codons are taken from **source\$** as successive value of the variable **codon\$** in line 1110.

The values of **codon\$** are used to access one of the elements of the genetic code cube, **code\$ (i,j,k)**, corresponding to the appropriate amino-acid or termination sign of the relevant codon. To do this, the **FOR....NEXT** loop in lines 1120 to 1150 turns each character, or base in **codon\$** into a coordinate used to read a value from the cube. Each base is taken in turn from **codon\$** in line 1130 as a single character long substring. The value of this character is then transformed into an integer in the range 1 to 4 in line 1140 and loaded into an element of the array **base(n)**, where **n** is an integer 1 to 3. For example, take the case that **codon\$** has the value **ACA**. Then in one complete run of loop **l**, the first value of **base\$** will be **A**. The ASCII code for **A** is 65. In line 1140, this is ORed with 16, hexadecimal 10 (&10) to give 81. On subtracting 79 and dividing by 2, we obtain a value of 1.

The variable **base(1)** then takes this value. The next character in **codon\$** is '**C**', ASCII value 67. When ORed with &10, this gives 83. Taking away 79 and dividing by 2 gives 2. Hence the value of **base(2)** becomes 2. In a similar way, the value of

base(3) is given the value 1 corresponding to the third base of codon\$, A. The values of the three elements of base(n) are then used as coordinates to access the value Thr from the cube. Thr has coordinates 1,2,1 corresponding to codon ACA. ORing is a process of comparison involving the binary representations of two numbers. In the present example, we have seen that ASC('A') OR 810 is 81. The binary equivalent of 65, the ASCII code for 'A' is 1000001. That of 810 (denary 16) is 0010000. 65 OR 810 is thus 1010001, denary 81. In the case of U, the ASCII code is 85, binary 1010101, so ASC('U') OR 810 is the same, 1010101, 85.

In line 1160, the variable aminoacid\$ is loaded by accessing the genetic code using coordinates base(1), base(2), base(3). Lines 1170 and 1180 determine appropriate output formats. Line 1180 starts a new oligopeptide if one of the mutant codons gives rise to a sequence terminator, UAA, UGA or UAG.

The procedure **mutantsequence** occupies lines 1250 to 1340. It uses two formal parameters, X\$ and m. X\$ corresponds to a sequence of characters representing an RNA message, m is the number of mutations to be made in the message. In line 1270, X\$ is copied to Y\$ for subsequent processing. The FOR...NEXT loop of lines 1280 to 1310 removes m characters from Y\$. Line 1290 chooses a random position in Y\$ depending upon its length. Line 1300 reduces the length of Y\$ by one character by removing the character at position X determined in the previous line. This is done by adding the characters to the left of position X, and those to the right of position X using LEFT\$ and RIGHT\$ respectively. The result is used to assign a new value to Y\$. The procedure translate is then called in line 1320, using the final value of Y\$ as actual parameter.

Using the program

Reading the paper in Nature by Crick et. al. (1) describing their investigation into the number of nucleotides in a DNA codon, one gets a great sense of the excitement that must have been around in biology at that time. The very secrets of life itself seemed about to be revealed. Being involved in the process of finding some fundamental rule or process in science must indeed be extremely exciting. Students of biology in school and colleges are usually deprived of that excitement. It is difficult, if not impossible to do any experiments in an attempt to reveal the rules of the genetic code. I modestly suggest that the use of the computer simulation FRAMES is a start in doing just that. By the application of a series of mutation procedures, students can begin to see a pattern in the resultant mutant oligopeptides; sequences produced using integral multiples of three deletions return later amino-acids in the sequence of their unmutated parental equivalents. This is not the case for other numbers of deletions.

This conclusion could be reached by students through guided discussion with the teacher or by their own investigation of the computer model.

The procedure translate could also form the kernel of other programs in this area of biology. Students could simulate the investigations of Nirenberg et. al. (10), which

gave the information need to express the isomorphism between codons and amino-acids that we refer to as the genetic code. I hope that readers will take up the challenge to write the appropriate program.

FRAMES Listing

```

10REM FRAMES
20REM
30REM Turn off ESCAPE
40*FX200,1
50REM
60REM
70REM Amino-acid mnemonics in cube order
80REM
90DATA Lys,Asn,Asn,Lys,Thr,Thr,Thr,Thr,Ile,Ile
,Ile,Met,Arg,Ser,Ser,Arg
100DATA Glu,His,His,Glu,Pro,Pro,Pro,Pro,Leu,Leu
,Leu,Leu,Arg,Arg,Arg,Arg
110DATA END,Tyr,Tyr,END,Ser,Ser,Ser,Ser,Leu,Phe
,Phe,Leu,END,Cys,Cys,Trp
120DATA Glu,Asp,Asp,Glu,Ala,Ala,Ala,Ala,Val,Val
,Val,Val,Gly,Gly,Gly,Gly
130REM
140REM
150REM Load genetic code cube
160REM
170DIM code$(4,4,4),base(3)
180FOR I=1 TO 4
190FOR J=1 TO 4
200FOR K=1 TO 4
210 READ code$(I,J,K)
220 NEXT K,J,I
230REM
240REM
250REM Set up parental protein and m-RNA
260REM
270 parentPROTEIN$="Ser-Tyr-Ser-Ile-Thr-Thr-Pr
o-Ser-Gln-Phe-Val-Phe-Leu-Ser-Ser-Ala-Try-Ala-As
p-Pro":REM First 20 amino-acids wild-strain TMV

```

```

protein (Dayhoff, 1969)-See Haggis, p.265
  280parentRNA$="UCUUAUCCAUUACUACUCCUUCGCAUUUG
UGUUUUUAUCUUCUGCGUGGGGAGAUCCC"
  290REM
  300REM
  310REM Set up and begin
  320REM
  330MODE7
  340VDU23,1,0;0;0;0;
  350FORI%=8TO9
  360PRINTTAB(9,I%);CHR$&8D;"FRAMES"
  370NEXTI%
  380REM
  390REM
  400REM Introduction
  410REM
  420delay=INKEY(300)
  430CLS
  440PRINTTAB(0,5);"This program is designed to
enable you""to investigate the number of bases
in a DNA codon."
  450delay=INKEY(300):PRINTTAB(0,10)"The compute
r contains a DNA sequence""which will generate
a protein sequence."
  460PRINT"By applying mutagens you can cause""
"point deletion mutants in the parental""DNA.""
'"The mutant DNA will be translated""into new a
mino-acid sequences."
  470PRINTTAB(3,20)"Press SPACE BAR to continue"
  480REPEATUNTILGET=32
  490REM
  500REM
  510REM Introduction continued
  520REM
  530MODE0
  540VDU23,1,0;0;0;0;0;
  550PRINTTAB(0,5)"Parent m-RNA transcribed from
DNA is:'''parentRNA$
  560PRINT""""Corresponding to parental protei
n:'''
  570PROCtranslate(parentRNA$)

```

```

580PRINTTAB(15,20)"Press SPACE BAR to continue
"
590REPEATUNTILGET=32
600REM
610REM
620REM Choose mutation procedure
630REM
640MODE0
650VDU23,1,0;0;0;0;0;
660PRINTTAB(8,5)"How many deletions do you wis
h to make in the parental DNA?"'"SPC(21)"Type a
number between 1 and 9"
670REPEAT
680PRINTTAB(27,15);:INPUT"How many deletions",
mutations%
690IF mutations%<1 OR mutations%>9 PRINTTAB(46
,15)"
"
700UNTIL mutations%>0 AND mutations%<10
710REM
720REM
730REM Translate mutant m-RNA and print mutant
proteins
740REM
750MODE0
760VDU23,1,0;0;0;0;0;
770PRINTTAB(0,2)"Parent protein is:"'"
780PROCtranslate(parentRNA$)
790PRINTTAB(0,7)"Mutant proteins with ";mutati
ons%;" deletions are:"'"
800FORK=1TO10
810PROCmutantsequence(parentRNA$,mutations%)
820NEXTK
830PRINTTAB(15,30)"Press SPACE BAR to continue
"
840REPEATUNTILGET=32
850REM
860REM
870REM Menu of options
880REM
890REPEAT
900MODE7

```



```

910VDU23,1,0;0;0;0;
920PRINTTAB(0,5)"Options""1 Repeat with same
number of mutants""2 Repeat with new number of
deletions""3 Start again""4 End the program""
""
930INPUT"Option",option%
940UNTILoption%>0 AND option%<5
950ON option% GOTO750,640,330,1000
960REM
970REM
980REM End program option
990REM
1000CLS
1010PRINTTAB(17,15)"END"
1020REPEATUNTILGET
1030*FX200,0
1040END
1050REM
1060REM
1070REM
1080DEFPROCtranslate(source$)
1090LOCALI
1100FOR position=1TO(LEN(source$)-LEN(source$)
MOD 3) STEP3
1110codon$=MID$(source$,position,3)
1120FORI=1TO3
1130base$=MID$(codon$,I,1)
1140base(I)=((ASC(base$) OR &10)-79)/2
1150NEXTI
1160aminoacid$=code$(base(1),base(2),base(3))
1170IF position<>1 aminoacid$="-"+aminoacid$
1180IF RIGHT$(aminoacid$,3)="END" PRINT ELSE PR
INTaminoacid$;
1190NEXTposition
1200PRINT
1210ENDPROC
1220REM
1230REM
1240REM
1250DEFPROCmutantsequence(X$,m)
1260LOCALI,X

```

```

1270Y$=X$
1280FORI=1TOM
1290X=RND(LEN(Y$)-2)
1300Y$=LEFT$(Y$,X)+RIGHT$(Y$,LEN(Y$)-X-1)
1310NEXTI
1320PROCtranslate(Y$)
1330PRINT
1340ENDPROC

```

FRAMES-variables

code\$(I,J,K)	amino-acid corresponding to (first base, second base, third base) m-RNA codon
base(n)	base at positions 1 to 3 in a codon
parentRNA\$	original TMV m-RNA sequence corresponding to original protein
parentPROTEIN\$	original TMV non-mutant protein
mutations%	number of point frame shift mutants induced in transcribed m-RNA sequence
options%	options for program continuation
source\$	m-RNA sequence to be translated into protein
position	access vector to choose every third base in source\$
codon\$	three consecutive characters in source\$ representing one m-RNA codon
base\$	character representing one base in a chosen codon
amino-acid\$	amino-acid accessed from code\$(I,J,K) corresponding to chosen m-RNA codon
m	number of mutations induced, equals mutation\$
X\$	m-RNA sequence to be translated into protein
Y\$	X\$ less point deletion mutants
X	position in X\$ where point deletion mutant effected.
I%	loop control variable for enhanced title printout

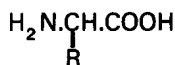
OLIGO

A simulation of biochemical procedures to determine the primary structure of oligopeptides, small proteins.

Biological background

Proteins are the basic sub-units of which all living things are composed. As enzymes they also control the operation of the collection of biochemical reactions that we call life.

Proteins are polymers of amino-acids. There are twenty or so naturally occurring amino-acids, but they all have the same basic design:



where R is variable. The nature of R is different in each amino-acid. If R is CH_3 -, the amino-acid is alanine; if R is H-, the amino-acid is glycine. Note that the amino-acid has an NH_2 - end and a -COOH end.

If alanine and glycine are condensed together under the action of an appropriate enzyme, a dipeptide results. In this case, the dipeptide would be:



This dipeptide retains an -NH_2 and a -COOH end. The other amino- and acid groups contribute to the formation of the -CO-NH- , or peptide linkage.

If further amino-acids are condensed onto this dipeptide, a growing polymer results-an oligopeptide or protein. The resultant molecule still possesses an -NH_2 and a -COOH end. The sequence of amino-acids or residues, in an oligopeptide or protein is referred to as the primary structure of the oligopeptide or protein. Remarkably, this primary structure determines higher levels of structure, such as the folding of the chain of amino-acids into complex globular shapes as occur in enzymes (2, 3, 4, 6). Obviously, the determination of the primary structure of protein molecules is an important step in understanding them, and their functions, further.

The first large protein of which the primary structure was known was insulin, the pancreatic hormone involved in the homeostatic control of blood sugar concentration, a deficiency of which causes the symptoms of diabetes.

This primary structure was determined by Sanger and his co-workers in the early 1950's (11). Their techniques form the basis of the program described in this section. These days, oligopeptide sequencing is largely automated using a technique known as Edman degradation (3).

Sanger and his co-workers used a variety of techniques to determine the amino-

acid sequence of proteins. The complete protein molecule can be subjected to hydrolysis using concentrated acids over a long period. This results in a mixture of the constituent amino-acids of the protein. The amino-acids present in the resultant mixture can be identified by one- or two-dimensional paper chromatography or electrophoresis. Estimates of the concentrations of different amino-acids in the mixture can also be made.

Less drastic acid hydrolysis yields a mixture of fragments of the original oligopeptide which can be isolated using a variety of techniques (3, 6).

Sequencing of these fragments resulting from partial acid hydrolysis can reveal regions of overlap which can be fitted together to reveal the original complete amino-acid residue sequence. It tends to be the case that partial acid hydrolysis produces a mixture of different fragments, the points of hydrolysis of the original oligopeptide being distributed approximately randomly along its length. The characterisation of the fragments can be assisted by their relative molecular mass determinations, see (3) for details.

Enzymes can also be used to fragment large proteins or oligopeptides. Other chemical reagents such as cyanogen bromide also degrade oligopeptides. The enzymes used for enzymic hydrolysis of the original oligopeptide or subsequent fragments are pepsin, trypsin and chymotrypsin. These enzymes hydrolyse peptide linkages involving particular amino-acids. For example, chymotrypsin is specific for those linkages involving tyrosine, tryptophan and phenylalanine. Trypsin hydrolyses sequences containing lysine or arginine. Enzymic hydrolysis can give rise to mixtures of fragments which can be subjected to further analysis.

The fourth technique used in 'classic' oligopeptide sequencing involves the identification of the amino-acids at the NH_2 and $-\text{COOH}$ ends of the molecule. In the former case, the original oligopeptide or subsequent fragments, can be treated with 1-fluoro-2,4-dinitrobenzene (FDNB). This attaches to the $-\text{NH}_2$ group of that terminal amino-acid, the nature of which can be determined by acid hydrolysis and chromatography. The use of FDNB for $-\text{NH}_2$ end characterisation is rare nowadays, 1-dimethyl-aminonaphthalene-5-sulphonyl chloride (dansyl chloride) being preferred (3).

The identification of the $-\text{COOH}$ terminal amino-acid of a sequence is carried out using lithium borohydride. This reagent reduces the $-\text{COOH}$ terminal amino-acid to an α -amino alcohol. On acid hydrolysis, this compound can be isolated from the other free amino-acids in the resultant mixture, and can be identified by chromatography.

By applying appropriate combinations of these procedures, together with determinations of the relative molecular masses of the original oligopeptide and subsequent fragments, the amino-acids sequence of any protein or oligopeptide can be determined.

The computer program

The program OLIGO is a simulation which allows the user to apply the techniques

outlined above to an oligopeptide, the primary structure of which is stored in the program. Any of the four analytical procedures of enzymic hydrolysis, complete acid hydrolysis and $-NH_2$ and $-COOH$ end characterisation can be performed on the the original oligopeptide or subsequent fragments.

Information about up to 100 fragments can be stored by the program during a run.

OLIGO was first described by the author (12) in 1979, when the program was named JSHPOLY.

Program description

Unlike all the other programs in this book, OLIGO is not written in structured BASIC. It was originally conceived from an idea by Peter Robinson and written for a main-frame computer. Its original (1979) form has been adapted for microcomputers, first the RML 380Z and now the BBC micro. Unstructured BASIC is difficult to follow from a long listing, but I hope that the reader will persevere. Intrepid readers might like to structure the coding for themselves! Whether this would make the program any shorter is another matter.

The program makes a lot of use of subroutines. These are rather similar to procedures used in structured BASIC, but their definition is by a starting line only, not a DEFPROC or anything similar, and they are exited via a RETURN, which sends program control to the line following the one in which the subroutine was called by a GOSUB command.

The program begins by printing a title page and giving a brief statement of its function, lines 70 to 220. In line 140, the stored parent oligopeptide is read from a DATA statement in line 2770 into P\$. The amino-acids are coded as single letters as tabulated below.

In lines 260 to 300, further data is read, in particular the relative molecular masses and R_f values of each of the oligopeptide's constituent amino-acids. Names of enzymes are also loaded into variables together with key words for printouts and a series of characters used in the enzyme hydrolysis parts of the program, E\$.

Each fragment produced from the original oligopeptide is located within P\$ using access vectors. These give the starting character and logical length of each fragment. Each fragment is numbered as it is produced. Starting position vectors are stored in the array S(n), logical lengths in the array L(n). The original oligopeptide is numbered fragment 1. In the listing given, the parent oligopeptide is GIVEECCA, so S(1) contains the value 1 and L(1) 8, the oligopeptide having 8 residues. It is, in fact, the small pituitary hormone oxytocin, involved in uterine contraction initiation during parturition. The access vectors for this oligopeptide are loaded in lines 150 and 290. F is a counter storing the number of fragments produced so far.

More information is printed from lines 310 to 400. At line 380, a subroutine starting at line 2440 is called. This subroutine occupies lines 2440 to 2490 and

determines the relative molecular mass of fragment Q. It also prints out this value, variable W (+ 18 for one water molecule), accessing the appropriate parts of the array M(n) which stores the relative molecular masses of ordered residues, using access vectors S(Q) and L(Q). When this subroutine is first called, Q has a value of 1 set in line 290, and so it determines the relative molecular mass of the original oligopeptide.

On RETURN in line 2490, control passes to line 1670 via the unconditional branch statement at line 410. Starting in line 1670 is a routine enabling the user to choose one of the four analytical procedures: enzymic hydrolysis, N- and C-terminal analysis, complete or partial hydrolysis. The routine ends in line 1780, where control passes to that part of the program determined by the value of A, the users valid response to the program continuation offer in line 1740.

If enzymic hydrolysis is chosen, control passes to line 450. First of all, the user is given the choice of enzyme to be used, lines 470 to 570.

Depending on the choice made, control then passes to line 1820, 1830, 1840 and 1850 for pepsin, trypsin, chymotrypsin or cyanogen bromide respectively. Each of these lines has the same basic structure. First of all a heading is printed by calling a subroutine at line 2730. Then a series of nested subroutines called via line 2620 fragment the protein or chosen fragment. The resultant fragments are given access vectors in lines 2340 to 2360. The fragments are then sorted into ascending logical lengths using a shuttle sort in lines 2090 to 2270. The relative molecular mass of each fragment is then determined by calling the subroutine starting at line 2440 from the loop running from line 2280 to 2320 and also printed to the screen. Control then passes to line 1470 where the user is offered the choice of continuing the program. If the user inputs P when continuation is offered, the program prints the residue sequence of the parent oligopeptide. If the user chooses to continue the program, control passes to line 1670, procedure choice.

A similar route through the program is followed by choosing partial acid hydrolysis. The partial acid hydrolysis routine starts at line 620. The main difference between this procedure and enzymic hydrolysis is that a random element is introduced. In lines 690 to 890, the chosen fragment is hydrolysed at random, and access vectors assigned to the resultant fragments. The resultant fragments are then sorted by length in lines 2080 to 2270 and the results printed to the screen. This part of the program could generate no or many new fragments. To allow for this, and to ensure that information about all new fragments passes to the user via the screen, paging mode is engaged using VDU14 at line 1680.

If the complete hydrolysis procedure is chosen by the user, control passes to line 960. The Rf values of the residues of the chosen fragment are accessed from the array where they are stored, R(n), and loaded into the working array C(m), where m is the number of characters in the chosen fragment. The elements of this latter array are then sorted into numerical order, lines 1070 to 1190. A table of Rf values and 'concentration' is then printed from lines 1210 to 1310. 'Concentration' refers to the number of occurrences of an amino-acid of a particular Rf value in the chosen fragment. Control then passes to line 1470, program continuation.

Finally, if N- and C- terminal analysis is chosen, control passes to line 1360. In lines 1420 and 1430, the R_f values of the two terminal amino-acids are printed from the array R(n) using the access vectors of the chosen fragment. Control then passes to line 1470.

If the user chooses to continue the run of the program by inputting 'Y' in response to the offer in line 1480, control passes to line 1550 where the user is asked to choose the number of the fragment he wishes to investigate.

Having taken a valid input, control passes to line 1670 where a procedure is chosen.

Using the program

Students can determine the primary structure of a small oligopeptide such as oxytocin working in small groups quite easily during a typical double lesson.

Because the program stores the fragments produced by any procedure, as well as printing them to the screen, a number of pairs of students can use the program simultaneously. Upto 100 fragments can be stored by the program.

Students need some data to be able to run the program. This is given in the data table below. They also need some basic protein chemistry, in particular some experience in amino-acid chromatography.

During a program run, a large number of fragments may be generated. Paging mode may be engaged as a result. During a printout of a long list of fragments, very likely when partial hydrolysis is chosen, the printout may stop and the shift lock red light come on. Printout can be resumed to termination by pressing the shift key. At the program continuation point, typing P return will produce as output, the original oligopeptide.

Amino acid	Mnemonic	R _f value	Rel mol mass	Letter code
alanine	ala	0.38	89	A
arginine	arg	0.2	174	R
aspartate	asp	0.24	133	D
cysteine	cys	0.08	121	C
glutamate	glu	0.3	147	E
glycine	gly	0.26	75	G
isoleucine	ile	0.72	131	I

histidine	his	0.21	155	H
leucine	leu	0.73	131	L
methionine	met	0.55	149	M
phenylalanine	phe	0.68	165	F
proline	pro	0.43	115	P
serine	ser	0.27	105	S
threonine	thr	0.35	119	T
tryptophan	trp	0.5	204	W
tyrosine	tyr	0.45	181	Y
valine	val	0.6	117	V

Amino-acid data used in OLIGO

(Rf values are quoted for butan-1-ol:water:ethanoic acid. Relative molecular masses are stored in DATA statements in the program as the values quoted minus 18, the rel mol mass of water. The data in the table is collected together from (3), (12) and (13)).

The oligopeptide stored in the version of OLIGO listed below is oxytocin, GIVEECA. From the above table, its primary structure is thus gly-ile-val-glu-glu-cys-cys-ala.

Other oligopeptides of any length up to 21 residues can be used for class analysis by changing the DATA statement at line 2770. The structure of this data is: oligopeptide in code form, rel mol mass of first amino-acid, Rf value of first amino-acid, rel mol mass of second amino-acid, Rf value of second amino-acid, Rf value of second amino-acid, etc.....'FSYLDEKRM'.

OLIGO Listing

```
>*.
  10REM OLIGO
  20REM
  30DIMC(21),L(100),M(21),R(21),S(100),F$(21),P
  $(21),BE$(16),SE$(23)
  40REM
```



```

50REM Header and introduction
60REM
70MODE7
80VDU23,1,0;0;0;0;
90PRINT:PRINT:PRINT"                                OLIGO"
100GOSUB1970
110GOSUB2710
120PRINT:PRINT:PRINT
130XX=INKEY(200)
140READP$
150L(1)=LEN(P$)
160VDU12
170PRINT:PRINT:PRINT
180PRINT"This program is designed to enable yo
u""to determine the amino-acid sequence""of an
oligopeptide.""The oligopeptide has ";L(1);"
amino-acids"
190GOSUB2710
200GOSUB2710
210PRINTTAB(5,20)"Type any key to continue"
220XX=GET
230REM
240REM Load arrays and read names
250REM
260FORI=1TOL(1)
270READM(I),R(I)
280NEXTI
290F=1:J=1:S(F)=1:Q=F
300READE$,PE$,TE$,CE$,BE$,H$
310VDU12
320PRINT:PRINT:PRINT
330REM
340REM First investigation
350REM
360PRINT"Investigation of complete oligopeptid
e"
370PRINT:PRINT"(Fragment 1 for subsequent acce
ss)"
380GOSUB2440
390GOSUB1970
400GOSUB2710

```

```

410GOTO1670
420REM
430REM Enzyme hydrolysis
440REM
450VDU12
460PRINT:PRINT:PRINT
470PRINT"Which enzyme?"
480PRINT
490PRINT"For ";PE$;" Type 1"
500PRINT"  "For ";TE$;" Type 2"
510PRINT"  "For ";CE$;" Type 3"
520PRINT"  "For ";BE$;" Type 4":PRINT
530REPEAT
540INPUT"Which enzyme",A
550A=ABS(INT(A))
560IFA<10RA>4THENPRINT:PRINT"Type an integer b
etween 1 and 4"
570UNTILA>0ANDA<5
580ON A GOTO1820,1830,1840,1850
590REM
600REM Partial acid hydrolysis
610REM
620VDU12
630PRINT:PRINT
640PRINT"Partial Acid Hydrolysis":PRINT
650GOSUB2590
660PRINT:PRINT"Fragments produced:"
670P=F:X=0
680IFL(J)<4THENGOSUB2710:PRINT"No new fragment
s produced":GOTO1470
690FORK=S(J)TOS(J)+L(J)-1
700V=INT(RND(1)*L(J))+1
710IFRND(1)>.3OR(K+V)>S(J)+L(J)THEN890
720F$=MID$(P$,K,V)
730IFLEN(F$)>=L(J)THEN890
740GOSUB2400
750IFK<>S(J)THEN800
760F$=MID$(P$,K+V,L(J)-L(F))
770GOSUB2400
780S(F)=K+V
790GOTO890

```

```

800IF K+V=S(J)+L(J) THEN F$=MID$(P$,S(J),L(J)-
L(F)):GOSUB2400:S(F)=S(J):GOTO890
810F$=MID$(P$,S(J),K-S(J))
820GOSUB2400
830S(F)=K+V
840GOTO890
850F$=MID$(P$,K+V,L(J)-L(F)-L(F-1))
860GOSUB2400
870S(F)=K+V
880GOTO890
890NEXTK
900IF P=F THENGOSUB2710:PRINT"No new fragments
produced":GOTO1470
910GOSUB2080
920GOTO1470
930REM
940REM Complete acid hydrolysis
950REM
960VDU12
970PRINT:PRINT
980PRINT"Comlete Acid Hydrolysis"
990GOSUB2590
1000Q=J
1010GOSUB2440
1020T=1
1030FORK=S(J)TOS(J)+L(J)-1
1040C(T)=R(K)
1050T=T+1
1060NEXTK
1070FORT=1TOL(J)-1
1080IFC(T)<=C(T+1)THEN1190
1090X=C(T)
1100C(T)=C(T+1)
1110C(T+1)=X
1120Y=T-1
1130IFY<10RC(Y)<=C(Y+1)THEN1190
1140Z=C(Y)
1150C(Y)=C(Y+1)
1160C(Y+1)=Z
1170Y=Y-1
1180GOTO1130

```

```

1190NEXTT
1200PRINT
1210PRINT "Rf value", "Concentration"
1220B=0
1230FORQ=1TOL(J)
1240T=1
1250IF C(Q)=B THEN1310
1260PRINTC(Q);
1270B=C(Q)
1280IF (Q+T) > L(J) OR C(Q) < > C(Q+T) THENPRINTT:GOTO13
10
1290T=T+1
1300GOTO1280
1310NEXTQ
1320GOTO1470
1330REM
1340REM N and C Terminal analysis
1350REM
1360VDU12
1370PRINT:PRINT:PRINT
1380PRINT "Terminal analysis"
1390GOSUB2590:PRINT
1400PRINT:Q=J
1410GOSUB2440
1420PRINT:PRINT "Rf of FDNB-bound amino-acid=";R
(S(J)+L(J)-1)
1430PRINT:PRINT "Rf value of C-terminal amino-ac
id=";R(S(J))
1440REM
1450REM Continuation option
1460REM
1470GOSUB2710
1480INPUT "Do you wish to continue",A$
1490IFLEFT$(A$,1)="Y" THEN1550
1500IFLEFT$(A$,1)="N" THENGOSUB2710:PRINT "End of
program":END
1510IF A$="P" THENVDU12:PRINT:PRINT:PRINT:PRINT "A
mino-acid sequence is ";P$:END
1520GOSUB2710
1530PRINT "Type Y or N"
1540GOTO1470

```

```

1550REPEAT
1560GOSUB1970
1570VDU12:VDU15
1580PRINT:PRINT:PRINT:PRINT
1590INPUT"Which fragment",J
1600J=INT(ABS(J))
1610GOSUB2710
1620IF J<10RJ>F THENPRINT"No such fragment. The
re are ";F;" only"
1630UNTILJ>0ANDJ<=F
1640REM
1650REM Choice of procedure
1660REM
1670PRINT"Which procedure?"
1680VDU14
1690PRINT:PRINT"For enzymic hydrolysis Type 1"
1700PRINT"For N- and C- terminal analysis Type
2"
1710PRINT"For complete hydrolysis Type 3"
1720PRINT"For partial hydrolysis Type 4":PRINT
1730REPEAT
1740INPUT"Which procedure",A
1750A=ABS(INT(A))
1760IFA<10RA>4THENPRINT:PRINT"Type an integer b
etween 1 and 4"
1770UNTILA>0ANDA<5
1780ON A GOTO450,1360,960,620
1790REM
1800REM Enzyme choice
1810REM
1820SE$=PE$:GOSUB2730:AA=1:CC=6:GOSUB2620:GOTO1
470
1830SE$=TE$:GOSUB2730:AA=7:CC=2:GOSUB2620:GOTO1
470
1840SE$=CE$:GOSUB2730:AA=1:CC=3:GOSUB2620:GOTO1
470
1850SE$=BE$:GOSUB2730:AA=9:CC=1:GOSUB2620:GOTO1
470
1860REM
1870REM Fragments print out
1880REM

```

```

1890PRINT:PRINT"Fragments produced:"
1900X=0:V=0:P=F
1910RETURN
1920REM
1930F$=MID$(P$,S(J)+X,V-X)
1940GOSUB2340
1950RETURN
1960REM
1970FORDD=1TO1500
1980NEXT
1990RETURN
2000REM
2010REM
2020IF X<>0 THEN2040
2030PRINT:PRINT:PRINT"No new fragments produced
":RETURN
2040IF X=L(J) THEN2080
2050F$=MID$(P$,S(J)+X,L(J)-X)
2060GOSUB2340
2070REM
2080IF F-P=1 THEN2280
2090FORI=P+1 TO F-1
2100IFL(I)<=L(I+1)THEN2270
2110X1=L(I)
2120X2=S(I)
2130L(I)=L(I+1)
2140S(I)=S(I+1)
2150L(I+1)=X1
2160S(I+1)=X2
2170Y=I-1
2180IF Y<(P+1) OR L(Y)<=L(Y+1) THEN 2270
2190Z1=L(Y)
2200Z2=S(Y)
2210L(Y)=L(Y+1)
2220S(Y)=S(Y+1)
2230L(Y+1)=Z1
2240S(Y+1)=Z2
2250Y=Y-1
2260GOTO2180
2270NEXTI
2280FORY=P+1TOF

```

```

2290PRINT:PRINT"Fragment ";Y
2300Q=Y
2310GOSUB2440
2320NEXTY
2330REM
2340F=F+1
2350S(F)=S(J)+X
2360L(F)=LEN(F$)
2370X=X+L(F)
2380RETURN
2390REM
2400F=F+1
2410L(F)=LEN(F$)
2420RETURN
2430REM
2440W=0
2450FORI=S(Q)TOS(Q)+L(Q)-1
2460W=W+M(I)
2470NEXTI
2480PRINT:PRINT"Rel mol mass of frag ";Q;"=";W+
18
2490RETURN
2500REM
2510FORK=S(J)TOS(J)+L(J)-2
2520V=V+1
2530FORBB=1TOLEN(SS$)
2540IFMID$(P$,K,1)=MID$(SS$,BB,1)THENGOSUB1930:
GOTO2560
2550NEXTBB
2560NEXTK
2570RETURN
2580REM
2590PRINT:PRINT" of fragment ";J
2600RETURN
2610REM
2620GOSUB2590
2630GOSUB1890
2640IFL(J)=1THENGOSUB2020:RETURN
2650REM
2660SS$=MID$(E$,AA,CC)
2670GOSUB2510

```

```

2680GOSUB2020
2690RETURN
2700REM
2710PRINT:PRINT:PRINT:PRINT:RETURN
2720REM
2730VDU12
2740SE$=SE$+H$
2750PRINTSE$
2760RETURN
2770DATA"GIVEECCA",57,.26,113,.72,99,.6,129,.3,
129,.3,103,.08,103,.08,71,.38,"FSYLDEKRM"
2780DATA"Pepsin","Trypsin","Chymotrypsin","Cyan
ogen bromide"," hydrolysis"

```

OLIGO-variables:

C(n)	working array to process Rf values
L(n)	logical length access vectors
M(n)	relative molecular masses of residues of complete oligopeptide
R(n)	Rf values of residues of original oligopeptide
S(n)	starting position access vectors
F\$	new fragment
P\$	original oligopeptide
PE\$, BE\$,	enzyme names
SE\$, CE\$	
XX	delay in program execution
I, K, T, Q, BB, Y, DD	loop control variables
F	number of fragments produced so far
J	number of current fragment

E\$	string containing codes for amino-acids 'searched for' by enzymes for fragment hydrolysis
H\$	'hydrolysis'
A	value for chosen procedure or enzyme
P	temporary store of value of F
V	length of randomly produced fragments in partial hydrolysis
X, Y, Z	working variables used in shuttle sort of Rf values
B, T	working variables used in complete hydrolysis procedure
X1, X2	swap variables used in fragment shuttle sort
Z1, Z2	
W	relative molecular mass of fragment
AA, CC	access vectors used to find substring of E\$ containing 'searched-for' amino-acids for hydrolysing enzyme

References

1. Crick, F. H. C., Barnett, L., Brenner, S. and Watts-Tobin, R., J., 'General nature of the genetic code for proteins', *Nature*, 192, (1961), 1227-1232.
2. Hofstadter, D. R., Godel, Escher, Bach: An eternal golden braid, Penguin Harmondsworth, 1980.
3. Lehninger, A. L., Biochemistry, Worth, New York, 2nd Ed., 1975.
4. Monod, J., Chance and necessity, Collins, Glasgow, 1972.
5. Schrödinger, E., What is life?, Cambridge University Press, Cambridge 1967.
6. Mahler, H. R., and Cordes, E. H., Biological chemistry, Harper & Row New York, 1966.
7. Chargaff, E., 'Preface to a grammar of biology: A hundred years of nucleic acid research', *Science*, 172, (1971), 637-642.
8. Watson, J. D., The double helix, Penguin, Harmondsworth, 1970.

9. Haggis, G. H., Introduction to molecular biology, Longman, London, 2nd Ed., 1974.
10. Nirenberg, M. W., and Leder, P., 'RNA codewords and protein synthesis', Science, 145, (1964), 1399-1407.

Nirenberg, M. W., and Matthaei, J. H., 'The dependence of cell-free protein synthesis in *E. coli* upon naturally occurring or synthetic polyribnucleotides, Proc. Nat. Acad. Sci., 47, (1961), 1588-1602.
11. Thompson, E. O. P., 'The insulin molecule', Scientific American, 192, (1955), 36-41.
12. Shone, J., 'Determining the primary structure of an oligopeptide-a computer simulation', J. Biol. Ed., 13, (1979), 123-126.
13. Roberts, M. B. V., Biology: a functional approach. Students manual, Nelson, Walton-on-Thames, 1974.

CHAPTER 5

ECOLOGY and BEHAVIOUR

Introduction

Ecology is the study of the inter-relationships of organisms and their biotic and abiotic environments. As with behaviour, which is the study of the reactions of living things to their environments, the study of ecology presents a variety of difficulties. Perhaps the greatest problem is the multi-variate nature of the phenomena that make up ecology and behaviour.

To a certain extent, the computer modelling of such phenomena can help to show the multivariate nature of ecology and behaviour, and indicate the need for care in the interpretation of real and 'model' data. The first program in this chapter, PHYTO, has been written to stress this multivariable problem, and to demonstrate a technique for coping with it. The last program, KLINO, shows that great care may be needed when interpreting data produced by models when compared with real experimental biology.

Academic ecology is becoming increasingly quantitative. Especially in the field of population dynamics. Differential equations are used to model population phenomena, such models being used to make predictions about, amongst other things, the future behaviour of natural populations. The program EPIDEM is a simple introduction to the computer assisted modelling of a population phenomenon using differential equations.

The fourth program described in this final chapter enables the further investigation of a simple technique used to estimate the size of an animal population.

PHYTO

This program is a simulation of phytoplankton production in temperate waters.

Biological background

The mass of biological material present in a particular place at one instant in time is called the standing crop. The actively photosynthetic plants of open waters in the sea and in lakes are collectively referred to as phytoplankton. If the standing crops, or biomasses per unit area of water surface, are determined for phytoplankton

monthly for a twelve month period, the figures often show two separate peaks of biomass. One of these peaks occurs in the spring, the other in the autumn. When conditions appear to be most favourable for photosynthesis, in the summer, biomasses of phytoplankton populations are low. This is, at first glance, a paradox since photosynthesis, the manufacture of organic materials from simple inorganic raw materials, is a generator of biomass.

The production of new biological tissue, new biomass certainly depends upon photosynthesis. This, in its turn, requires chlorophyll, the green pigment typical of higher plants, which acts as a solar energy collector, this energy being locked into large organic molecules during the photosynthetic process.

The synthesis of chlorophyll necessitates a continued supply of magnesium, used to coordinate the nitrogen atoms of the four central pyrrole rings of the molecule (1).

In addition, the synthesis of new plant cells necessitates a continued supply of nitrogen in nitrate form.

This nitrogen is used by the plant in the manufacture of proteins and other nitrogenous compounds. A supply of phosphate ions is also a pre-requisite for continued phytoplankton production, the phosphorus being needed in the synthesis of such compounds as adenosine triphosphate and nucleic acids.

Other minerals, taken up as ions in solution from the surrounding waters, are needed for other metabolic activities, cofactors in biochemical reactions, raw materials in biosynthesis and so on. Continued phytoplanktonic population growth is thus intimately dependent upon an adequate supply of dissolved minerals in the surrounding fluid medium. Should these essential materials become absent, phytoplankton production would decrease, and due to the continued respiration of existing cells and the activities of herbivorous animals (zooplankton), far from increasing, phytoplankton biomass per unit area of water surface would decrease (2).

Of all the elements mentioned, probably the most important limiter of production is phosphorus. The available forms of phosphorus, the phosphates, tend to be rather rare. A good deal of phosphorus in the biosphere is locked up into shells and bone, the mineral component of which is necessarily rather insoluble! Hence, the recycling of phosphorus through ecosystems is rather a slow affair. Phosphorus is usually in poor supply.

What of the two peaked phytoplankton curve? Does the availability of minerals such as phosphorus become even less than usual during the warm summer months in open waters? The answer appears to be 'yes'.

During the warmer months of the year, the upper layers of the sea or a lake themselves become heated. This upper region of water is called an epilimnion. Below this warm, well-lit zone is a layer of water which is much colder, increasingly so with depth, and much darker. This layer is referred to as the hypolimnion. Between these two zones, one highly productive (or potentially so), one rather unproductive, is a barrier, a thermocline. This discontinuity in the vertical zonation

of open waters effectively prevents any mixing of waters from the two layers, the epilimnion above and hypolimnion below (3,4).

The recycling of minerals essential for the continued production of new phytoplankton biomass depends upon the decomposition of material that is derived from other living things. This decomposition is brought about by micro-organisms, many of which are to be found in the lower water depths, in the hypolimnion.

During the summer, when a thermocline is present, the nutrients released by decomposition below the thermocline become unavailable to the epilimnion.

However, during the autumn, when the epilimnion begins to cool and weather becomes stormier, the thermocline breaks down and circulation of water from lower depths becomes possible. This, in turn brings nutrients into the upper water layers, removing the resource deficiency which caused the summertime decline in production.

During the winter months, declining temperatures and short days cause a decline in production. As the temperature of the water rises during the spring, and longer days enable more light energy to be absorbed by chlorophyll, production rises.

Thus, a number of influences contribute to the production of new biomass in the epilimnion of open waters. Riley (5) has designed a mathematical model enabling the prediction of phytoplankton biomass. A simplified version of this model is used as the basis for the program PHYTO.

Riley's model accounts for phytoplankton production as the net outcome of three influences:

- (i) photosynthesis, P
- (ii) phytoplankton respiration, R
- (iii) grazing by herbivorous zooplankton, G

The model also takes account of the nutrient depletion factor due to the summer thermocline. Details of the model can be found in (2) or (5).

Basically, photosynthesis, P increases production, which is decreased by the other two factors,

$$\text{production} = P - R - G$$

P is dependent upon the intensity of sunlight at the sea surface, the maximum depth at which photosynthesis can take place, and the extinction coefficient of water. It is also dependent upon a nutrient factor, N, which has low values in the summer but higher values in the spring and autumn:

$$P = (A I / z) (1 - \exp(-kz)) (N)$$

where A is a constant, I the solar radiation at the water surface, z the maximum depth at which photosynthesis can occur, and k the extinction coefficient.

Respiration, R is given by:

$$R = R_0 \exp(rt)$$

where R_0 is respiration at 0°C , r a constant and T is temperature. Grazing is approximated using gZ , where g is a constant and Z is the biomass of zooplankton.

The computer program

The program PHYTO uses a simplified version of Riley's model and stored data to generate curves of phytoplankton biomass for thirty six successive months.

Biological phenomena are influenced by many variables, and often deterministic models such as Riley's give a distorted view of biological reality. Other factors, acting apparently at random, modify the behaviour predicted from theory. To assist students to grasp this idea, each figure for monthly biomass is subjected to multiplication by a factor drawn from a table of random normal deviates. The resulting pattern of change of biomass with time for any one year then more or less approximates the pattern predicted by the model. A randomising factor adds 'noise' to the system. This noise can be removed by averaging the data for a number of years, and the program allows this to be done. This idea of 'signal averaging' is very important in biology, where an underlying signal is often masked by random noise (6).

Program description

The program begins by loading a series of arrays with data, lines 90 to 310. First of all, data concerning the ecological variables (such as grazing factor and solar radiation) is loaded into the appropriate array. Each element of an array, say the temperature array, represents the mean value of that ecological variable for a particular month. The data used to load the arrays, lines 1700 to 1810 is taken from a variety of sources, or subjectively assessed.

Modification of this data will obviously change the output from the program.

The first letters of the names of months are loaded in lines 230 to 250 from data in line 1850. Lines 290 to 310 load random normal deviates (mean 0, variance 1) from data at lines 1890 to 1980 into the array randomizer. This array is used to generate 'noise' in program output. The use of random normal deviates in simulation can produce some interesting effects. Normal deviates can be generated from an appropriate function, or taken from published tables such as those in (7).

The procedure **introduction** occupying lines 1510 to 1660 and called from line 370 gives a brief statement of the function of the program. Some of this introduction is printed in colour to highlight significant words using CHR\$.

For example, CHR\$(81) causes text following it to be printed in red. CHR\$(87) returns text colour to white.

On ENDPROC control passes to line 390, where a graphics mode is called. The VDU28 call at line 400 reserves the top portion of the screen for text in a text window. VDU23;8202;0;0;0 at line 410 turns off the flashing cursor.

The procedure **axes** called from line 430 occupies lines 610 to 790. It uses the formal parameter **t\$** used to print the vertical axis label. The procedure sets up the horizontal axis for time, labelled with the names of months, and the vertical axis for biomass.

The procedure **calculation-and-plot** occupying lines 820 to 1080 is the kernel of the program. It consists of two FOR...NEXT loops nested within a third. The outer loop uses the control variable **year** and is used to generate three years worth of data for output. The loop running from line 910 to 970 is used to calculate the phytoplankton biomass for each month in a twelve month period. Biomass for each month is determined in line 930. The values of the constants used in the expressions described above are those given by Krebs (2). Noise is added to the calculated value of biomass by multiplying the result by **zdeviate**, a randomly chosen normal deviate from the array **randomizer**, line 920. During a complete run of this loop, maximum and minimum values of biomass are found. These are used to calculate a scaling for subsequent data plotting, lines 980 to 990.

The figures for biomass in relative units are plotted in the predrawn axis space using the loop occupying lines 1040 to 1060. The three curves for the three years are drawn in different colours, set by **GCOL0,year** in line 880.

On **ENDPROC**, control passes to line 460 from where the procedure continuation is called. This procedure, occupying lines 1110 to 1230 prints information for the user in the text window. The user is offered three options for program continuation or termination. Three more curves can be produced, the program can be terminated or a mean curve can be calculated. This mean curve is produced by calculating the average relative biomass for each month produced so far by the program. The procedure **meancurve** is called from line 520 and occupies lines 1260 to 1480. The mean curve is drawn in magenta, by changing logical colour 1 to actual colour 5 in line 1330. The procedure has a similar construction to the plot part of **calculation-and-plot**, each element of the twelve element array **mean** taking a value determined by accumulators, **total(n)**, the total relative biomass for each month, and **seasons**, the total number of years for which data has been generated. These accumulators are incremented during the procedure **calculation-and-plot**, lines 870 and 960. The values for monthly means are also scaled to give relative data for the actual plot.

Using the program

PHYTO can be used in class to introduce the ideas of varying phytoplankton production over a twelve month period and signal averaging. A theoretical exercise supplementary to the former use can be found in Sands (8).

Due to the simplicity of the calculation part of PHYTO, it is not inconceivable that students could be encouraged to develop their own mathematical models of phytoplankton production under teacher guidance. Line 930 in the program could then be modified to run any model so designed. The ecological variable data could also be varied, and probably improved by further research. Data could be entered

for non-temperate waters in which temporary thermoclines are absent, and the effect on phytoplankton production thus investigated. The order of data in lines 1700 to 1810 is that in which it is read in lines 160 to 200.

PHYTO - Listing

```
10REM PHYTO
20
30REM PHYTOPLANKTON BIOMASS MODEL
40REM
50*FX200,1
60
70REM SET ARRAYS OF MONTHLY VARIABLES
80REM
90DIMgrazing(12),radiation(12),max_depth(12),
temperature(12)
100DIMmonth$(12),nutrient_factor(12),randomize
r(100),biomass(12)
110DIMtotal(12),mean(12)
120
130REM LOAD ARRAYS
140REM
150FORmonth=1TO12
160READgrazing(month)
170READradiation(month)
180READmax_depth(month)
190READtemperature(month)
200READnutrient_factor(month)
210NEXTmonth
220REM
230FORI=0TO11
240READmonth$(I)
250NEXTI
260
270REM READ RANDOM NORMAL DEVIATES INTO ARRY
280REM
290FORI=1TO100
300READrandomizer(I)
310NEXTI
320
```



```

330REM GO
340REM
350MODE7
360VDU23,1,0;0;0;0;0;
370PROCintroduction
380
390MODE1
400VDU28,10,10,35,0
410VDU23;8202;0;0;0
420
430PROCaxes("rel.")
440PROCcalculation_and_plot
450
460PROCcontinuation
470CLS
480CLG
490
500ON option GOTO 430,520,550
510
520PROCmean_curve
530GOTO460
540
550*FX200,0
560MODE7
570PRINTTAB(10,15);"END OF PROGRAM"
580END
590
600
610DEFPROCaxes(t$)
620VDU19,1,1,0,0,0
630GCOLOR,3
640VDU29,200;100;
650VDU23,1,0;0;0;0;0;
660MOVE0,500
670DRAW0,0
680DRAW1000,0
690VDU5
700MOVE-150,300
710PRINTt$
720MOVE-150,250
730PRINT"mass"

```

```

740FORI=0TO11
750MOVEI*90,-50
760PRINTmonth$(I)
770NEXTI
780VDU4
790ENDPROC
800
810
820DEFPROCcalculation_and_plot
830COLOUR2
840PRINTTAB(5,3);"Calculating"
850COLOUR3
860FORyear=1TO3
870seasons=seasons+1
880GCOL0,year
890minP=5000
900maxP=0
910FORmonth=1TO12
920zdeviate=randomizer(RND(100))
930biomass(month)=(25*(radiation(month)/max_de
pth(month))*(1-EXP(-.1*max_depth(month)))*(nutri
ent_factor(month)/.55)-(.0175*EXP(.069*temperatu
re(month)))-grazing(month))*(zdeviate*.26+1)
940IF maxP<biomass(month) maxP=biomass(month)
950IF minP>biomass(month) minP=biomass(month)
960total(month)=total(month)+biomass(month)
970NEXTmonth
980range=maxP-minP
990scale_factor=500/range
1000REM
1010REM DRAW CURVES
1020REM
1030CLS
1040FORI=1TO12
1050IF I=1 MOVE (I-1)*90,(biomass(I)-minP)*scal
e_factor ELSE DRAW(I-1)*90,(biomass(I)-minP)*sca
le_factor
1060NEXTI
1070NEXTyear
1080ENDPROC
1090

```

```

1100
1110DEFPROCcontinuation
1120REPEAT
1130CLS
1140COLOUR2
1150PRINT"Options: "" ""
1160COLOUR3
1170PRINT"Type 1 for 3 more years""Type 2 for
mean curve""Type 3 to end program""
1180COLOUR2
1190PRINT"Option";
1200COLOUR3
1210INPUToption
1220UNTILoption=1 OR option=2 OR option=3
1230ENDPROC
1240
1250
1260DEFPROCmean_curve
1270PROCaxes("mean")
1280CLS
1290COLOUR2
1300PRINTTAB(5,1);seasons;" years' data"
1310COLOUR3
1320GCOL0,1
1330VDU19,1,5,0,0,0
1340min=5000
1350max=0
1360FORI=1TO12
1370mean(I)=total(I)/seasons
1380IF mean(I)>max max=mean(I)
1390IF min>mean(I) min=mean(I)
1400NEXTI
1410range=max-min
1420scale_factor=500/range
1430FORI=1TO12
1440IF I=1 MOVE(I-1)*90,(mean(I)-min)*scale_fac
tor ELSE DRAW(I-1)*90,(mean(I)-min)*scale_factor
1450NEXTI
1460PRINTTAB(0,6);"Press spacebar to carry on"
1470REPEATUNTILGET=32
1480ENDPROC

```

```

1490
1500
1510DEFPROCintroduction
1520FORI%=8TO9
1530PRINTTAB(12,I%);CHR$(&8D);CHR$(&82);"PHYTO"
1540NEXTI%
1550delay=INKEY(400)
1560CLS
1570PRINTTAB(3,5);"This program draws curves of
" ""
1580PRINT" ";CHR$(&84);"monthly";CHR$(&82);"phy
toplankton";CHR$(&84);"biomass";CHR$(&87);
1590PRINTTAB(5,9);"for each of";CHR$(&81);"thre
e";CHR$(&87);"years"
1600PRINTTAB(3,20);CHR$(&86);"Press spacebar to
continue";CHR$(&87);
1610REPEATUNTILGET=32
1620CLS
1630PRINTTAB(5,5)"You can also choose to see""
"a curve of";CHR$(&85);"MEAN";CHR$(&84);"monthly
standing crop";CHR$(&87);
1640PRINTTAB(4,20);CHR$(&86);"Press spacebar to
continue";CHR$(&87);
1650REPEATUNTILGET=32
1660ENDPROC
1670
1680
1690REM ECOLOGICAL VARIABLE DATA
1700DATA .6,57,30,5,.55
1710DATA .75,91,35,6,.55
1720DATA1.5,126,40,6.5,.55
1730DATA 6.75,160,50,8,.58
1740DATA 2.25,195,60,9,.58
1750DATA 1.8,229,60,10,.4
1760DATA 1.8,210,60,10,.4
1770DATA 1.8,191,50,10,.4
1780DATA 2.25,172,40,9,.5
1790DATA 1.8,134,40,7,.55
1800DATA .75,95,35,6,.55
1810DATA .6,57,30,5,.55
1820

```

```

1830
1840REM MONTH NAMES
1850DATA J,F,M,A,M,J,J,A,S,O,N,D
1860
1870
1880REM RANDOM NORMAL DEVIATES
1890DATA -1.11,0.99,-6E-2,0.41,0.78,0.15,-0.34,
0.34,-0.13,-1.79
1900DATA -0.71,-3E-2,-0.16,-0.2,1.3,-0.66,1.17,
0.87,-1.22,0.24
1910DATA 0.34,-0.58,0.35,-0.41,1.59,1.58,-0.34,
0.17,-0.44,0.11
1920DATA 0.15,0.1,2.05,1.61,1.24,-0.55,0.91,-0.
89,-1.39,2.11
1930DATA -0.89,6E-2,-0.21,0.27,1.29,0.83,-0.52,
-0.53,0.19,-2.03
1940DATA 0.88,-1.38,2.53,1.13,-0.49,0.42,2.55,-
1.02,-0.77,0.31
1950DATA 1.37,0.76,1.36,-0.77,-0.87,0.29,-1.79,
-0.53,2.04,-1.2
1960DATA 2.15,1.36,-0.17,-0.58,1.86,0.59,0.37,0
.39,0.11,-0.22
1970DATA -1.38,1.5,0.44,-0.39,-0.53,-0.32,-0.38
,-0.31,-0.76,0.69
1980DATA 0.57,1.04,1.9,1.26,0.27,-0.3,-7E-2,1.4
9,-1.22,-0.86

```

PHYTO-variables:

grazing(n)	array of monthly grazing factors
radiation(n)	array of monthly incident solar radiation
max.depth(n)	maximum depth at which photosynthesis can occur for each month
temperature(n)	mean monthly temperature
month\$(n)	string array containing first letters of month names
nutrient.factor(n)	monthly nutrient factor
randomizer(m)	array storing random normal deviates

biomass(n)	calculated phytoplankton biomass for each month
total(n)	total biomass for particular month over seasons years
mean(n)	mean biomass for particular month over seasons years
year	takes value 1, 2 or 3
minP, min	minimum values of biomass(n) and mean(n)
maxP, max	maximum values of biomass(n) and mean(n)
month	takes values 1 to 12 step 1
zdeviate	random normal deviate chosen from array randomizer(n)
range, scale_factor	used to calculate scalings for curve plotting
l, l%	loop control variables
option	user response to program continuation offer
delay	delay in program execution

EPIDEM

A simple simulation of the spread of diseases.

Biological and theoretical background

Infectious diseases exert a major controlling influence over the growth of natural populations. Epidemics may be caused by one or more infectives, individuals incubating the disease and mingling in a population of uninfected susceptibles.

In a simple model of an epidemic, let S be the number of susceptibles. The rate of increase of susceptibles will be birth rate minus rate of infection.

It can be assumed that those infected subsequently become immune. This can be expressed as a differential equation:

$$\frac{dS}{dt} = \text{birth rate} - (pSI)$$

where t is time, p the probability of infection and I the number of infectives.

Another differential equation can be written down for the rate of increase of infectives:

$$\frac{dI}{dt} = (pSI) - kI$$

in which kI is the rate of conversion of infectives into obvious 'patients' and k is a constant.

Values for the constants in the above equations can be estimated if the model is applied to a human population of say 10000 people, in which most individuals live to about 75, so birth rate is approximately the same as death rate, about 125 per annum, 5 per fortnight. The incubation period is such that k is about 2.5. A reasonable value for p could be about $1/400$.

Hence the two differential equations become:

$$\frac{dS}{dt} = 5 - (SI)/400 \quad (1)$$

$$\frac{dI}{dt} = (SI)/400 - 2.5I \quad (2)$$

(the unit of time is the fortnight).

Using time slices of 1 fortnight, these differential equations can be turned into recurrence relations, a form eminently useful for writing computer programs around differential equations. Hence,

$$S_{t+1} = S_t + 5 - S_t I_t / 400 \quad (3)$$

$$I_{t+1} = I_t + S_t I_t / 400 - 2.5 I_t \quad (4)$$

These recurrence relations are used as the core of the procedure CALC, lines 710 and 720, where susceptibles is S_{t+1} and infectives I_{t+1} .

The time course of many epidemics is such that the number of cases shows an oscillation (9). Such oscillations are predicted by this simple mathematical model.

The computer program

The two recurrence relations (3) and (4) above can be used as the basis of a simple computer program to predict the pattern of variation of the number of cases of infectious disease in time. EPIDEM is such a program. EPIDEM produces, as output, a graph of number of cases (infectives) against time.

Program description

The version of EPIDEM herein presented is a slightly modified version of a program written by G. A. Thompson (10).

The program is very simple. The core of the program is the procedure CALC, lines 700 to 730. Lines 710 to 720 contain the recurrence relations derived above. The procedure is called repeatedly from a FOR...NEXT loop running from line 520 to 570. The values of the control variable of this loop, T, represent fortnightly intervals from 2 to 100. The procedure CALC is called from line 530, and the number of infectives plotted against time in line 560, the values of T and infectives having been scaled arbitrarily. PLOT5 is equivalent to DRAW. The point corresponding to the first value of T, T=1, is determined by calling CALC from line 480, the graphics cursor then being moved to the appropriate coordinates in line 510.

The axes for the plot are drawn and labelled in lines 260 to 430. Mode 0 is chosen for the plot. Other modes will do, but the graphics origin set at line 270 will have to be changed, as will the coordinates used to print the axis labels.

The user inputs two items of data in lines 200 and 210, the starting values for number of susceptibles and number of infectives respectively. These values are input in teletext mode 7.

Lines 620 to 630 offer the user the option of continuing the program and inputting more starting data by pressing the spacebar. Pressing any other key stops the program.

It must be pointed out that this program is not error trapped. I have not intended to present a completely 'student-proof' program; I have tried to show how a simple mathematical model can form the basis of a computer simulation. The reader might like to 'complete' the program for himself.

The reader must also be warned that certain combinations of input data will produce some very strange results, in part due to the scaling of infectives chosen in line 560. So, be warned!

Using the program

This program has not been written for class use. It is an illustration of the process of turning a piece of theoretical biology into its visual representation made possible by the use of a microcomputer.

Nevertheless, it produces some interesting results. Consider an infected population in a steady state:

$$\frac{dS}{dt} = \frac{dI}{dt} = 0$$

Hence, from equations (1) and (2), $S=1000$, $I=2$.

Using these two values for input as susceptibles and infectives respectively gives the expected, but rather uninteresting output from the computer. Choosing non-equilibrium values, say 1020 and 1 respectively gives a much more interesting result!

EPIDEM - Listing

```
10REM EPIDEM
20
30REM EPIDEMIC MODEL-NB NOT ERROR TRAPPED
40
50*FX200,1
60REM
70REM TITLE SEQUENCE
80REM
90MODE7
100VDU23,1,0;0;0;0;
110FORI%=8TO9
120PRINTTAB(9,I%);CHR$(8D);"EPIDEM"
130NEXT
140delay=INKEY(500)
150CLS
160REM
170REM
180REM INPUT INITIAL DATA
190REM
200INPUTTAB(10,10)"Susceptibles",s
210INPUTTAB(10,15)"Infectives",i
220REM
230REM
240REM CALL GRAPHICS MODE & DRAW AXES
250REM
260MODE0
270VDU29,200;100;
280MOVE0,1000
290DRAW0,0
300DRAW1000,0
310REM
320REM
330REM SET PRINT TO GRAPHICS CURSOR MODE
340VDU5
350MOVE-200,500
360REM
```

```

370REM
380REM PRINT AXIS TITLES
390REM
400PRINT"infectives"
410MOVE500,-50
420PRINT"Time"
430VDU4
440REM
450REM
460REM CALCULATE INFECTIVES AND PLOT ON AXIS SPACE
470REM
480PROCCALC(s,i)
490s=susceptibles
500i=infectives
510MOVE10,infectives*200
520FORT=2T0100
530PROCCALC(s,i)
540s=susceptibles
550i=infectives
560PLOT5,T*10,infectives*200
570NEXT
580REM
590REM
600REM PROGRAM CONTINUATION OPTION
610REM
620PRINTTAB(0,30)"Press SPACE BAR to continue"
630IFGET=32RUN ELSE MODE7
640PRINTTAB(15,10)"END"
650*FX200,0
660END
670REM
680REM
690REM
700DEFPROCCALC(s,i)
710susceptibles=s+5-(s*i)/400
720infectives=i+(s*i)/400-2.5*i
730ENDPROC

```

EPIDEM-variables

I%	control variable of loop generating enhanced title
delay	delay in program execution
susceptibles	number of individuals in the population who are potential sufferers of disease
infectives	number of sufferers of disease
T	control variable of plotting loop
s, i	initial values of susceptibles and infectives; formal parameters of procedure CALC

POPEST

A program to estimate the sample proportion giving minimal practical error in calculating the size of a motile population using the Lincoln index method.

Biological and theoretical background

A simple way to estimate the size of a population of motile animals is to use a mark-recapture method. The simplest of these methods is the Lincoln index (9).

A random sample of the population whose size is to be estimated is taken. These individuals are marked in some way such that the mark can be recognised later by the experimenter. The captured and marked animals are then released into their natural habitat. After a time, a second random sample is taken from the same population, and the number of individuals in the second sample which had been previously marked is recorded.

Let the size of the original random sample be n_1 and the size of the second random sample be n_2 . Further let the number of marked individuals in the second sample be n_3 . If \hat{N} is an estimate of the size of the population of true size N , then

$$\hat{N} = (n_1 n_2) / n_3$$

The variance of \hat{N} , $\text{var}(\hat{N})$ is given approximately by:

$$\text{var}(\hat{N}) = (n_1^2 n_2 (n_2 - n_3)) / n_3^3 \quad (9)$$

Poole (9) states that an effective estimate of N is possible only if $n_1 n_2 > 4N$.

The use of the Lincoln index can be justified only when the following assumptions are made.

1. When released back into the population, the animals of the first sample redistribute themselves at random.
2. The mark on the released animals must in no way increase the animals' probability of being caught and eaten by predators, or in any other way affect its normal way of life.
3. The generation time of the species concerned must be long in relation to the interval between the first and second samples.
4. The population must be discreet, not subject to emigration or immigration during the course of the estimate.
5. Having been caught once, the probability of subsequent recapture must be the same as that of the probability of a first capture.

The fact that it is usually very difficult to make all of these assumptions for a real population means that in practice the Lincoln index technique is rarely used to estimate population sizes. A full discussion of this and other methods of population size estimation can be found in Poole (9).

Nevertheless, students of Advanced level biology may well be expected to calculate estimates of population size using the Lincoln index. The computation of a Lincoln index, involving simple multiplication and division operations on three numbers is too trivial even for biologists to require a computer program for its execution!

It is a more useful exercise to consider how the accuracy of estimates of population size vary with the sizes of the two random samples, n_1 and n_2 . We could ask what size of sample in relation to population size, N , gives minimal error in \hat{N} ? Obviously, in the limiting case, if $n_1=n_2=N$, then error must be zero. Error in this case could be expressed as a relative error, $(|\hat{N}-N|)/N$.

In the theoretical case, we could let n_1 and n_2 always be equal, and by varying n_1/N , called the sample proportion, for fixed N , it would be possible to assess a minimum value for n_1/N giving minimum practical error in estimating N . This task is a very appropriate one for a computer, and is the purpose of the program POPEST. Pooles' assertion, mentioned above, that for 'effectiveness', $n_1 n_2 > 4N$ can be tested.

The computer program

POPEST was written to assess the minimum sample proportion giving maximum reliability for practical purposes when using the Lincoln index technique to estimate the size of a motile population.

Repeated estimates of the size of a population are made by simulation. The computer 'contains a population' of a constant 500 individuals. A random sample of these are taken and 'marked'. A second sample, the same size as the first is then taken, and a Lincoln index population estimate made. The relative error in the estimate is then calculated and plotted against sample proportion, sample size/500. The sample size is increased from 10 to 500 in steps of 10.

Program description

A run of POPEST is terminated by pressing the ESCAPE key. If ESCAPE is pressed, control passes to the procedure **end**, lines 910 to 990. ESCAPE is detected as being hit in line 60.

On running the program, the array **animal** is dimensioned to 500 to represent the 500 individuals of the population whose size is to be estimated. A brief introduction and program description is printed to the screen by calling the procedure **introduction** from line 110. This procedure, occupying lines 1020 to

1300, begins by printing a title page in double height blue characters, lines 1030 to 1050. After a delay, three pages of information are presented, the second and third of which are printed to the screen on pressing the spacebar (ASCII code 32). Significant words and phrases in the presented information are printed in colour using CHR\$.

On ENDPROC, control is passed to line 130. A graphics mode is called in line 130 and a text-window of 8 lines set up in line 150 using VDU28. The procedure **draw.axes** is then called from line 160.

The procedure **draw.axes** occupies lines 410 to 580. The graphics origin is moved to coordinates 279,200 in line 430, and graphics colour red chosen in line 420. The axes for the output graph are drawn in this colour at lines 440 and 450. The axes are labelled in the program segment running from line 460 to 570. VDU5 allows text to be printed at the graphics cursor, line 460, this being cancelled by VDU4 at line 570. Axis labels and error values for the vertical axis are drawn in cyan, logical colour 2 being changed to actual colour 6, cyan, in line 480 using VDU19. On ENDPROC control passes to line 180.

The FOR...NEXT loop running from line 180 to 360 is the kernel of the program. The values of the loop control variable, J are the successive values of the sample size, $n_1 = n_2$, used to calculate \hat{N} . With each new value of J, the procedure **calc.lincoln.index** is called in line 190. This procedure uses the actual parameter J. The procedure **calc.lincoln.index** occupies lines 610 to 670 and uses the formal parameter sample-size into which the current value of J is passed. The loop control variable I is made LOCAL in line 620. In lines 640 to 660, the values of the elements of the array animal are all set to zero, which means 'uncaptured, unmarked'. The variable n3 is zeroed in line 670 for each new value of J. In lines 690 to 740, J animals are randomly chosen, choice being indicated by setting the value of the appropriate array variable to 1. The REPEAT...UNTIL loop occupying lines 700 to 720 ensures that no array element is picked and marked twice. The loop running from line 760 to line 810 represents the second random sample from the population. A random integer is chosen in line 770 and this value assigned to the variable second-capture.

The conditional branch statement at line 780 is used to check that the array element animal (second.sample) has not been previously chosen. If the value of the chosen element is 1, then it was also chosen in the first sample, and so the accumulator of recaptured, marked animals, n3, is incremented by 1. To mark the fact that the animal represented by the chosen array element has been captured, the value of the element is set to 2.

A Lincoln index estimate of population size is then determined in line 820. If no elements of the array animal were chosen in the first and second samples, the value of n3 is zero. Under these circumstances, the value of **line.index** is arbitrarily set to zero. Control passes on ENDPROC to line 200.

At line 200, the procedure **calculate.error** is called. This procedure occupies lines 860 to 880 and calculates the relative error in \hat{N} . Control passes to line 210 on ENDPROC.

Lines 230 to 340 print information about the last calculation in the text window set up at line 150. Values for N , \hat{N} , sample-size n_1 , sample proportion n_1/N and error are printed. The text window is cleared for each new value of J by CLS at line 210. The other parts of the screen are not affected by this command. The point corresponding to the current value of J , sample-size and error is plotted on the axis space at line 350. PLOT69 generates a point.

Using the program

Students of Advanced level biology, and other courses, will sooner or later come across the Lincoln index estimator of population size.

The determination of a Lincoln index using a model population, of peas say, or mealworms, is a simple exercise. POPEST can extend this work in approaching the question raised above, viz. 'what sample proportion gives minimal error in \hat{N} ?'. The obvious answer is 1, you count the population!

What we want to know is the proportion of the population needing to be sampled to give an efficient estimate of its size. Poole's criterion, if n_1 and n_2 are the same, and N is 500, implies that n_1 should be greater than 45. In this case, sample proportion, n_1/N is .09. The reader can run the program to decide whether this is true. The program is terminated by hitting the ESCAPE key. After a delay, the END message appears in the text window, then the whole screen is cleared.

POPEST Listing

```

10REM POPEST
20
30REM Estimation of sample proportion giving
minimal error
40REM in calculation of Lincoln index
50
60ONERRORPROCend
70DIManimal(500)
80
90MODE7
100VDU23,1,0;0;0;0;0;
110PROCintroduction
120
130MODE1
140VDU23,1,0;0;0;0;0;
150VDU28,0,8,39,0

```



```

160PROCdraw_axes
170
180FORJ=10TO500STEP10
190PROCcalc_lincoln_index(J)
200PROCcalculate_error
210CLS
220VDU23,1,0;0;0;0;
230COLOUR2
240PRINTTAB(10,1);"Type ESCAPE to finish"
250PRINTTAB(5,4);"N=500"
260COLOUR3
270PRINTTAB(14,4);"n1=";J
280PRINTTAB(25,4);"n1/N=";J/500
290COLOUR2
300PRINTTAB(5,6);"Estimate=";INT(lincoln_index

310COLOUR3
320PRINTTAB(25,6);"Error=";
330COLOUR1
340PRINT;INT(error*1000)/1000
350PLOT69,J*2,error*500
360NEXTJ
370
380END
390
400
410DEFPROCdraw_axes
420GCOL0,1
430VDU29,279;200;
440MOVE0,500
450DRAW0,0:DRAW900,0
460VDU5
470GCOL0,2
480VDU19,2,6,0,0,0,0
490MOVE-200,300
500PRINT"Error"
510MOVE750,-75
520PRINT"n1/N"
530MOVE-50,500
540PRINT"1"
550MOVE-50,0

```

```

560PRINT "0"
570VDU4
580ENDPROC
590
600
610DEFPROCcalc_lincoln_index(sample_size)
620LOCAL I
630IF INKEY(-35) PROCend
640FOR I=1 TO 500
650animal(I)=0
660NEXT
670n3=0
680
690FOR I=1 TO sample_size
700REPEAT
710marked_animal=RND(500)
720UNTIL animal(marked_animal)=0
730animal(marked_animal)=1
740NEXT I
750
760FOR K=1 TO sample_size
770second_capture=RND(500)
780IF animal(second_capture)=2 THEN 770
790IF animal(second_capture)=1 n3=n3+1
800animal(second_capture)=2
810NEXT K
820IF n3<>0 lincoln_index=(sample_size)^2/n3 EL
SE lincoln_index=0
830ENDPROC
840
850
860DEFPROCcalculate_error
870error=(ABS(lincoln_index-500))/500
880ENDPROC
890
900
910DEFPROCend
920COLOUR3
930delay=INKEY(500)
940CLS
950PRINT TAB(15,3); "END"

```

```

960delay=INKEY(250)
970CLG
980END
990ENDPROC
1000
1010
1020DEFPROCintroduction
1030FORI%=8TO9
1040PRINTTAB(12,I%);CHR$(&8D);CHR$(&84);"POPEST
"
1050NEXTI%
1060delay=INKEY(400)
1070
1080CLS
1090PRINTTAB(1,5);"This program is designed to
enable"" you to investigate the effect of""
1100PRINT"      ";CHR$(&82);"sample size";CHR$(&8
7);"on";CHR$(&85);"estimates""
1110PRINT"      "CHR$(&87);"of";CHR$(&83);"pop
ulation size";CHR$(&87);" ""
1120PRINT" using the";CHR$(&86);"LINCOLN INDEX"
;CHR$(&87);"technique"
1130delay=INKEY(300)
1140PRINTTAB(4,20);"Press spacebar to continue"
1150REPEATUNTILGET=32
1160CLS
1170
1180PRINTTAB(2,5);"A population of";CHR$(&86);"
500 individuals";CHR$(&87);"""" is sample
d repeatedly"
1190delay=INKEY(300)
1200PRINTTAB(0,10);"With each repetition a";CHR
$(&86);"LINCOLN INDEX";CHR$(&87);""""estimate o
f population size is made""using an";CHR$(&81)
;"increasing";CHR$(&83)"sample proportion"
1210delay=INKEY(300)
1220PRINTTAB(3,20);CHR$(&87);"Press spacebar to
continue"
1230REPEATUNTILGET=32
1240CLS
1250

```

```

1260PRINTTAB(0,12);"The";CHR$(&81);"error";CHR$
(&87);"inherent in the estimate is""then plott
ed against";CHR$(&83);"sample proportion";CHR$(&
87);
1270delay=INKEY(300)
1280PRINTTAB(6,20);"Press spacebar to begin"
1290REPEATUNTILGET=32
1300ENDPROC

```

POPEST-Variables

animal(n)	array, the elements of which represent individuals in the sampled population
J, K, I, I%	loop control variables
lincoln_index	Lincoln index estimate of population size
error	relative error in Lincoln index
sample.size	formal parameter of procedure calc.lincoln_index
marked_animal	random integer 1 to 500 used to choose random element of animal(n)
second.capture	random integer in range 1 to 500, see above
n3	number of array elements chosen in first and second samples
delay	delay in program execution

KLINO

A model of klinokinesis

Biological background

Many invertebrate and vertebrate animals show orientation movements called taxes or kinesis. These are changes in patterns of movement brought about by changes in the external environments of organisms, when such external changes are perceived through the sensory systems of the organisms concerned. Some of these induced movements have a direction which relates in a predictable way to the direction of the stimulus. For example, *Calliphora* larvae consistently move away from a source of unidirectional light in a direction directly opposite to that of the stimulus. This is a taxis, in particular a negative phototaxis.

Sometimes, organisms respond to environmental changes perceived as stimuli by responses unrelated to the direction of the stimulus. These sorts of responses are called kinesis.

For example, if woodlice are subjected to a choice chamber, in which one half is moist, and the other half dry, it is possible to show that the speed of movement of the organisms is significantly higher in the dry side than in the moist. The humidity of the surroundings affects the speed of movement of the woodlice. This is an orthokinesis (11). The effect of this orthokinesis in woodlice is that they tend to aggregate in moist situations. This is of adaptive significance in that being crustaceans of the Order Isopoda, sub-order Oniscoidea (12), the absence of a waxy cuticle makes them prone to rapid dessication in dry habitats. They tend to be restricted therefore to moist, dark situations such as are found in leaf litter. The increase in speed which they show in dry situations means that their probability of reaching a more favourable situation is increased. Their low speed in moist situations means that their chance of leaving such a situation is decreased.

The survival value of such instinctive behaviour is difficult to dispute.

Klinokinesis refers to a change in an animals frequency of turning during locomotion (11, 13.) As with orthokinesis, it seems likely that klinokinetic behaviour would be of survival value.

Consider a hypothetical invertebrate at rest in its habitat. It is an organism that shows photo-orthokinesis, coming to rest in the dark, but moving rapidly when illuminated. Imagine that the organism is suddenly illuminated.

What is its best klinokinetic strategy? To move rapidly with few turns, that is in straight lines, to find a 'hide' or to set off on a path on which there are lots of turns? The question is: 'What is the best klinokinetic strategy to locate randomly dispersed hides when suddenly disturbed?' The disturbance could be caused by a predator.

The wrong choice of strategy could mean it takes a long time to find a hide

increasing the probability of being caught and eaten. The program KLINO investigates this hypothetical situation.

Disturbed insects such as cockroaches run off in straight paths as fast as they can (14). Indeed, the giant fibres of their nerve-cords, designed for rapid impulse conduction, mediate the forward movement of the antennae prior to setting off—probably to give early warning of barriers in the way of these ‘blind’ runs!

Wigglesworth (15) describes the rapid turning movements of lice which tend to keep them in favourable conditions.

The computer program

KLINO was written to investigate the hypothetical situation outlined above; to enable students to predict the best strategy for escape to be used by an animal suddenly exposed to unfavourable stimuli in a habitat where there are randomly dispersed ‘hides’.

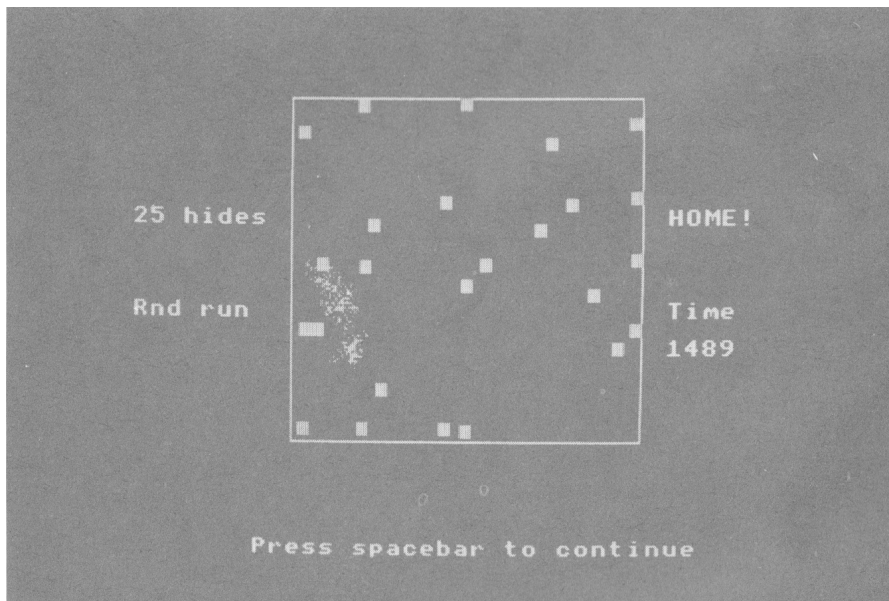


Figure 5.1 A random run with 25 hides

Two alternative strategies are available to the animal, which is assumed to show photo-orthokinesis. Either it can move in straight lines, linear movement, or it can show rapid turns, in the extreme case, random movement.

In each case, the speed of the animal is the same.

From ten to fifty randomly distributed hides can be set up in a rectangular habitat on the screen. A random starting position for the organism is then chosen. The user can then choose one or other of the above-mentioned kinetic strategies. The time taken for the hypothetical organism to reach one of the hides is then measured in centiseconds and printed to the screen. The user is then given a menu of options to choose from for continuing the run. Amongst other things, he can repeat a similar run with the same hides or choose the alternative strategy; or he can choose a different arrangement of hides or a different starting place. By choosing appropriate numbers of hides, different starting positions and strategies, the user can investigate the best escape strategy in this model.

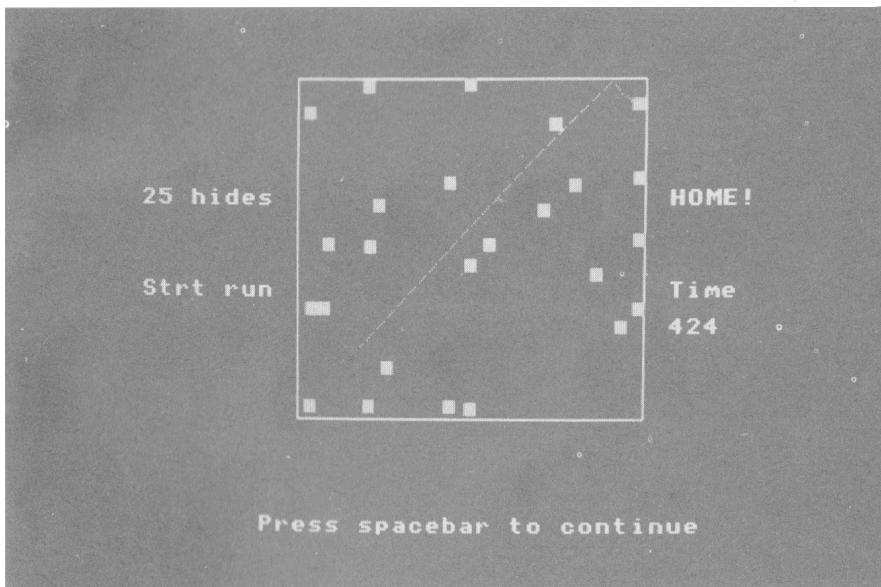


Figure 5.2 A straight Run from the Same Start Position as Figure 5.1

Program description

KLINO begins by printing a header page. This is printed from the procedure **header** occupying lines 1110 to 1160 and called from line 80. This procedure prints the program title in magenta double height characters. After a delay, the screen is cleared and an introduction page written to the screen from the procedure **introduction** called from line 100.

This procedure occupying lines 1190 to 1310 first of all prints to the screen a statement of the purpose of the program. Key phrases are highlighted by printed them in colour using CHR\$. On pressing the spacebar, the user is presented with a

second page of introductory information, the delivery of which is paced using INKEY (lines 1230 to 1290). On ENDPROC at line 1310, control passes to line 120.

At line 120, graphics mode 1 is called, followed in line 130 by the procedure **hides**. This procedure, occupying lines 700 to 940 allows the user to set up between ten and fifty random hides of the same size in the habitat of the hypothetical organism. It starts by asking the user to input the number of hides required. The input statement at line 740 occurs within a REPEAT...UNTIL loop, lines 710 to 750, which prevents program continuation until a valid input has been received from the user (line 750).

On the acceptance of a valid input for the number of hides required, the procedure **habitat** is called from line 780. This procedure occupies lines 390 to 490 and draws the boundaries of the habitat in white, logical colour 3, the graphics origin having been moved to coordinates 300,300 by the VDU29 at line 430.

On ENDPROC at line 490, control passes back to the procedure **hides** at line 790, where logical colour 1, red is chosen. The VDU19 at line 800 then converts logical colour 1 into actual colour 5, magenta, the colour in which the hides will be printed to the screen. The FOR...NEXT loop occupying lines 810 to 910 then chooses random starting coordinates for the bottom left hand corner of each hide (lines 820 and 830). The arguments of the RND statements are chosen such that the hides will not overlap the edge of the habitat set in lines 440 to 480. These starting coordinates are stored as elements of the arrays `hide-startX(n)` and `hide-startY(n)` for subsequent use. Each hide is drawn to the screen in lines 860 to 900 using PLOT85 which fills in a triangle between three specified coordinates in the current foreground colour, actual colour 5 in this case. Each hide is a 20 x 20 square. When the hides are all printed to the screen, the user is asked if he is satisfied with their distribution in line 920. If the first letter of his response is N or n, indicating dissatisfaction, the chosen number of hides are relocated by passing control back to 780 otherwise ENDPROC at line 940 sends program control back to line 150.

At line 160, the procedure **start.position** is called. This procedure occupying lines 520 to 670 sets up the position where the hypothetical animal is initially at rest prior to disturbance. The procedure begins by calling the procedure **habitat**. The procedure **print.hides** is then called from line 550.

This procedure, occupying lines 970 to 1080 uses the values of the elements of the arrays `hide-startX(n)` and `hide-startY(n)` to draw the chosen number of hides in their original positions in the habitat, as before in actual colour magenta, logical colour red using PLOT85. ENDPROC sends program control back to the procedure **start-position** at line 560.

The procedure **start.position** continues by printing a message from line 560 and then choosing X and Y coordinates for the organisms' start position in lines 570 and 580. If the values of X and Y thus chosen overlap a hide, the choice is made again. This is made possible by using POINT(X,Y) at line 590.

This returns a value depending on the logical colour of the pixel specified by the X and Y coordinates. Although the actual colour of the hides is magenta, colour 5, their logical colour is 1. So, if the value of POINT(X,Y) is 1, new values of X and Y are determined. A pixel chosen in this way is indicated by a flashing red-cyan dot, set up by changing logical colour 2 to actual colour 9, flashing red-cyan, in lines 600 and 610. PLOT69 at line 620 plots this point on the screen. Lines 540 to 630 are enclosed within a REPEAT...UNTIL loop enabling the user to choose what he considers to be an appropriate starting position. When the user is satisfied with the starting position, its X and Y coordinates are stored as Xstart and Ystart in lines 650 and 660 for later recall. On ENDPROC program control passes to line 180.

At line 180 teletext mode 7 is entered and the procedure **type-of-run** is then called from line 190. This procedure occupying lines 1400 to 1460 allows the user to choose the straight run or random run strategy. On ENDPROC control passes to line 210 where graphics mode 1 is reentered.

The procedure **run** is then called from line 220. This procedure occupies lines 1490 to 1780. It begins by drawing the habitat boundaries and then the chosen hides by calling the procedures **habitat** and **printhides** respectively.

The number of hides and the type of run are then printed to the screen from lines 1520 and 1530. The starting position chosen previously is plotted in yellow, lines 1540 and 1550. The user is then asked to continue, line 1560, the message being erased in line 1570 when the spacebar is pressed to start the animals' run.

At the start of the run, the pseudo-variable **TIME** is set to zero, line 1600.

The actual run is generated in the REPEAT...UNTIL loop of lines 1630 to 1720. It stops when the coordinates reached have logical colour 1 (lines 1700, 1720). The initial values of X and Y, the starting position coordinates, are incremented by the amounts corresponding to the current values of the variables **incX** and **incY**. These latter variables have values determined by the type of run, straight or random, and the values of random numbers generated during the run (lines 1640 to 1670). A continuing run is drawn to the screen in line 1710. Lines 1610 and 1620 determine the initial direction of a straight run.

When a run ends in line 1720, a message to that effect is printed to the screen, line 1730. The current value of **TIME** in centiseconds is also printed in line 1760. On ENDPROC at line 1780, the program control passes to line 240, program continuation and then line 260.

At line 270, the procedure **menu**, occupying lines 1810 to 1840 is called.

This procedure offers various (six) alternatives for continuing or terminating program execution. The users' choice is acted upon by a program branch effected by the multiple branch statement at line 280. For example, if the other type of run option, option 4 is chosen, the program branches to line 290, and the value of the variable **type-run** is changed.

The program continues until option 6, program termination is chosen.

Using the program

KLINO was written to enable students to assess the relative merits of two escape strategies. The scenario can be introduced by the teacher by guided discussion, and an hypothesis suggested. For the sake of argument, let us suggest that the straight run strategy involving minimum number of turns per unit time seems to be the best strategy.

The program can then be used to test this hypothesis. In one such test of the hypothesis, 80 separate runs were timed, 40 for each of straight and random strategies. Eight runs of each type were carried out at each of 10, 20, 30, 40 and 50 hides in the habitat. For each number of hides, 4 different start positions were chosen and two random and two straight runs timed from each start position.

The results obtained produced a mean time to reach a hide of 1724 centiseconds for random run, with a biased estimate of standard deviation of 2623.8 centiseconds for this data. The corresponding figures for the straight runs were a mean of 518.4 centiseconds to find a hide, standard deviation 586.1 centiseconds. Applying a t-test to the figures gives a value of t of 2.8005 with 79 degrees of freedom, giving a one-tailed test probability of less than .005. Hence the results of the two types of run are significantly different at this level. In particular, the mean for the random run is significantly higher than that of the straight run. The straight run strategy therefore appears to be best.

The use of a t-test to assess the significance of the difference between the two means could be criticised in this case as the biased estimates of the standard deviations of the two data sets appear widely disparate. A t-test is only valid under conditions of homogeneity of variance.

If the pairs of data for each strategy at each starting point are added, the resultant data for each starting position can be analysed using a Wilcoxon matched pairs signed-rank test, a non-parametric procedure (16). If this test is employed, the difference between the pairs of data for random and straight runs is significant at the .005 level. Once again, straight running turns out to be the best strategy.

How does the prediction of the model compare with the behaviour of actual organisms? One of the classical experiments in the literature about klinokinesis is that is that of Ulliyott (17) on the turbellarian platyhelminth *Dendrocoelum lacteum*. A detailed analysis of this work can be found in (11).

In one of Ulliyott's investigations, the animals were left in the dark for about 28 minutes and then they were illuminated from above. Their rate of change of direction was then measured. This set up is very similar to that of KLINO. KLINO predicts that the animals should show a decrease in their rate of turning, or run straight. In fact, *Dendrocoelum* is reported to show an increase in its rate of turning; its movements become increasingly erratic.

What has gone wrong? Nothing. If Ulliyott's results are correct, and they can be checked by repetition, they can be explained in adaptive terms. From the planarians' point of view, sudden illumination could mean that it has moved out of a favourable habitat, perhaps into view. Turning round is the 'obvious' thing to do!

Why run off in a straight line when there is a hide close by? The animals were dark-adapted, used to being in the dark; the sudden illumination was interpreted by the nervous system as wandering into the light.

Is the model in KLINO in some sense wrong? No, it is not. Both situations, model and reality are in a sense valid. Their interpretation makes them so.

It makes sense for a dark-adapted animal to behave as *Dendrocoelum* does given certain circumstances. It would also make sense for an animal to behave as predicted by KLINO—at least, disturbed cockroaches and other escaping organisms seem to think so!

Indeed Ulliyott's results are supported by the model if the start position is next to a hide. Then, random runs reach home significantly faster than straight runs.

It seems to me that the use of this program offers a valuable lesson when thinking about animal behaviour. What is an appropriate strategy for movement depends on a variety of things such as the local circumstances, or what the organism has been doing before. Perhaps in the sense of oversimplification the computer model is 'wrong'. Behaviour, like all biology is multi-variate. The most sophisticated computer model can only approximate reality.

Computer models cannot exactly mirror biological systems, but I believe that they can help us think about them.

KLINO Listing

```
10REM KLINO
20
30DIMhide_startX(50),hide_startY(50)
40*FX200,1
50MODE7
60VDU23,1,0;0;0;0;0;
70
80PROCheader
90CLS
100PROCintroduction
110
120MODE1
130PROC hides
140
150MODE1
160PROCstart_position
170
180MODE7
```

```

190PROCtype_of_run
200
210MODE1
220PROCrun
230
240PROCcontinue(6,28)
250
260MODE7
270PROCmenu
280NOPTION GOTO 210,150,120,290,50,320
290IFtype_run=1 type_run=2 ELSE type_run=1
300GOTO210
310
320MODE7
330VDU23,1,0;0;0;0;
340*FX200,0
350PRINTTAB(13,13);"END"
360END
370
380
390DEFPROChabitat
400CLS
410GCOL0,3
420VDU23,1,0;0;0;0;
430VDU29,300;300;
440MOVE0,0
450DRAW0,600
460DRAW650,600
470DRAW650,0
480DRAW0,0
490ENDPROC
500
510
520DEFPROCstart_position
530REPEAT
540PROChabitat
550PROCprint_hides
560PRINTTAB(9,1);"Choose start position"
570X=RND(620)+20
580Y=RND(570)+20
590IF POINT(X,Y)=1 THEN 570

```

```

600GCOL0,2
610VDU19,2,9,0,0,0,0
620PLOT69,X,Y
630PRINTTAB(14,30)::INPUT"Satisfied",answer$
640UNTIL LEFT$(answer$,1)="Y" OR LEFT$(answer$,
,1)="y"
650Xstart=X
660Ystart=Y
670ENDPROC
680
690
700DEFPROC hides
710REPEAT
720CLS
730VDU23,1,1;0;0;0;0;
740PRINTTAB(5,10)::INPUT"How many hides (10-50
)",hides
750UNTIL hides=INT(hides) AND hides>=10 AND hid
es<=50
760VDU23,1,0;0;0;0;0;
770PROCcontinue(4,20)
780PROC habitat
790GCOL0,1
800VDU19,1,5,0,0,0,0
810FOR I%=1 TO hides
820X=10+RND(619)
830Y=10+RND(570)
840hide_startX(I%)=X
850hide_startY(I%)=Y
860MOVEX,Y
870MOVEX+20,Y
880PLOT85,X+20,Y+20
890MOVEX,Y+20
900PLOT85,X,Y
910NEXT I%
920PRINTTAB(13,30)::INPUT"Satisfied",answer$
930IF LEFT$(answer$,1)="N" OR LEFT$(answer$,1)
="n" THEN 780
940ENDPROC
950
960

```

```

970DEFPROCprint_hides
980PROCchabitat
990GCOLOR,1
1000VDU19,1,5,0,0,0,0
1010FORI%=1TOhides
1020MOVEhide_startX(I%),hide_startY(I%)
1030MOVEhide_startX(I%)+20,hide_startY(I%)
1040PLOT85,hide_startX(I%)+20,hide_startY(I%)+2
0
1050MOVEhide_startX(I%),hide_startY(I%)+20
1060PLOT85,hide_startX(I%),hide_startY(I%)
1070NEXTI%
1080ENDPROC
1090
1100
1110DEFPROCheader
1120FORI%=8TO9
1130PRINTTAB(13,I%);CHR$(133);CHR$&8D;"KLINO"
1140NEXTI%
1150delay=INKEY(300)
1160ENDPROC
1170
1180
1190DEFPROCintroduction
1200PRINTTAB(0,7);"This program is designed to
help you""investigate the";CHR$(130);"best str
ategy";CHR$(135);"for""escape for a small inve
rtebrate ""CHR$(133);"suddenly exposed to light
.";CHR$(135)
1210delay=INKEY(200)
1220PROCcontinue(3,20)
1230CLS
1240PRINTTAB(0,2);"You will be able to set up";
CHR$(129);"randomly""distributed";CHR$(130);"h
ides";CHR$(135);"in which the animal""can esca
pe predation."
1250delay=INKEY(300)
1260PRINTTAB(0,10);"You can choose";CHR$(129);"
ONE";CHR$(135);"of";CHR$(129);"TWO";CHR$(135);"e
scape""strategies: ""1 Rapid";CHR$(133);"ran
dom";CHR$(135);"movement""

```

```

1270delay=INKEY(100)
1280PRINT"OR""2  Rapid";CHR$(131);"linear";CHR
R$(135);"movement"
1290delay=INKEY(300)
1300PROCcontinue(3,22)
1310ENDPROC
1320
1330
1340DEFPROCcontinue(a,b)
1350PRINTTAB(a,b);CHR$(134);"Press spacebar to
continue";CHR$(135)
1360REPEATUNTILGET=32
1370ENDPROC
1380
1390
1400DEFPROctype_of_run
1410VDU23,1,1;0;0;0;
1420REPEAT
1425CLS
1430PRINTTAB(0,10);"Which type of run?"'"Type
1   for";CHR$(131);"STRAIGHT";CHR$(135);"RUN"'
"Type 2   for";CHR$(133);"RANDOM";CHR$(135);"RUN
"'"'"
1440INPUT"Which type of run ",type_run
1450UNTIL type_run=1 OR type_run=2
1460ENDPROC
1470
1480
1490DEFPROCrun
1500PROCchabitat
1510PROCprint_hides
1520PRINTTAB(0,10);hides;" hides"
1530PRINTTAB(0,15);:IF type_run=1 PRINT"Strt ru
n" ELSE PRINT"Rnd run"
1540GCOLOR,2
1550PLOT69,Xstart,Ystart
1560PROCcontinue(6,28)
1570PRINTTAB(5,28);"
"
1580X=Xstart
1590Y=Ystart

```

```

1600TIME=0
1610 IF RND(1)>.5 incX=5 ELSE incX=-5
1620 IF RND(1)>.5 incY=5 ELSE incY=-5
1630REPEAT
1640IF type_run=2 AND X>640 incX=-RND(10) ELSE
IF type_run=2 AND X<30 incX=RND(10) ELSE IF type
_run=2 AND RND(1)>.5 incX=RND(10) ELSE IF type_r
un=2 incX=-RND(10)
1650IF type_run=2 AND Y<30 incY=RND(10) ELSE IF
type_run=2 AND Y>590 incY=-RND(10) ELSE IF type
_run=2 AND RND(1)>.5 incY=RND(10) ELSE IF type_r
un=2 incY=-RND(10)
1660IF type_run=1 AND X>640 OR X<20 incX=-incX
1670IF type_run=1 AND Y>590 OR Y<20 incY=-incY
1680X=X+incX
1690Y=Y+incY
1700IF POINT(X,Y)=1 THEN 1720
1710PLOT69,X,Y
1720UNTIL POINT(X,Y)=1
1730PRINTTAB(31,10);"HOME!"
1750PRINTTAB(31,15);"Time"
1760PRINTTAB(31,17);TIME
1770delay=INKEY(200)
1780ENDPROC
1790
1800
1810DEFPROCmenu
1820PRINTTAB(0,5);CHR$(133);"Options";CHR$(135)
;"'"1 Repeat with same hides and start'"2 Rep
eat with different start'"3 Repeat with differ
ent hides and start'"4 Repeat with other type
of run'"5 Start again'"6 End the program'"
1830INPUT" Option ",option
1840ENDPROC

```


KLINO-variables:

hide.startX(n), hide.startY(n)	array storing starting positions of hides
option	user input in response to menu
X, Y	random coordinates
answer\$	user response to 'Satisfied?'
Xstart, Ystart	starting coordinates of animals' run
hides	number of hides in habitat
l%	loop control variable
delay	delay in program execution
a, b	line number, character position: formal variables of procedure continue
type-run	1 for straight run, 2 for random run
incX, incY	variables used to change values of X and Y coordinates

References

1. Lehninger, A. L., Biochemistry, Worth, New York, 2nd. Ed., 1975.
2. Krebs, C. J., Ecology: The experimental analysis of distribution and abundance, Harper & Row, New York, 2nd. Ed., 1978.
3. Cushing, D. H., Science and the fisheries, Arnold, London 1977.
4. Yonge, C. M., The sea shore, Collins, London, 1949.
5. Riley, G. A., 'Factors controlling phytoplankton populations of Georges Bank', J. Mar. Res., 6, (1946), 104-113.
6. Kay, R. H., Experimental biology: Measurement and analysis, Chapman & Hall, London, 1964.
7. Mosteller, F., Rourke, R. E. K., and Thomas, G. B., Probability with statistical applications, Addison-Wesley, London, 2nd. Ed., 1970.
8. Sands, M. K., Problems in ecology: Teachers Edition, Mills & Boon, London, 1978.

9. Poole, R. W., An introduction to quantitative ecology, McGraw-Hill, Tokyo, 1974.
10. Thompson, G. A., personal communication.
11. Fraenkel, G. S., and Gunn, D. L., The orientation of animals, Oxford University Press, London, 1940.
12. Green, J., A biology of crustacea, Witherby, London, 1961.
13. Hinde, R. A., Animal behaviour, McGraw-Hill, Tokyo, 2nd. Ed., 1970.
14. Roeder, K. D., Nerve cells and insect behaviour, Harvard University Press, Cambridge, Massachussets, rev. ed., 1967.
15. Wigglesworth, V.B., 'The sensory physiology of the human louse *Pediculus humanus corporis* de Beer (Anoplura)', Parasitology, 33, (1941), 67-109.
16. Haber, A., and Runyon, R. P., General statistics, Addison-Wesley, London, 2nd. E., 1973.
17. Ulliyott, P., 'The behaviour of *Dendrocoelum lacteum* I and II', J. Exp. Biol., 13, (1936), 253-278.

BIOLOGY — on a computer?

Yes - because a computer can be used to simulate what happens in real life - at a fraction of the cost and in a much shorter time than "real" biological experiments. And, when you **do** perform live experiments, you really need a computer to process the results.

The computer chosen by John Shone, an experienced teacher of biology, is the BBC computer. This is now the most popular computer in Britain's schools. There are numerous books on how to program it, on learning BASIC and other machine-specific functions but very few that actually help in a particular subject.

This book is an exception; it tackles biology and examines all those aspects of the subject that lend themselves to computation. As such, it will be invaluable to teachers, students AND to parents with BBC Computers wishing to provide valuable reinforcement and homework for their children in this enjoyable subject. The book includes sufficient subject description of every biological topic covered, full listings of programs actually tested in schools, and suggestions of how to use the programs.

Contents include:

Number crunching – deviation; standard error test; association; aggregation, species diversity indexes

Genetics – Mendel's Laws; population genetics; evolution model; genetic drift

Ecology and Behaviour – photosynthesis; epidemics; instinctive behaviour

Physiology – oxygen and haemoglobin; respiration; nerve fibre modelling; colour vision; rhythm method of contraception.

Enjoy this and other fine books, from:

Sigma Technical Press

5 Alton Road
Wilmslow
Cheshire
SK9 5DY

Price £6.95

ISBN 0 905104 53 6