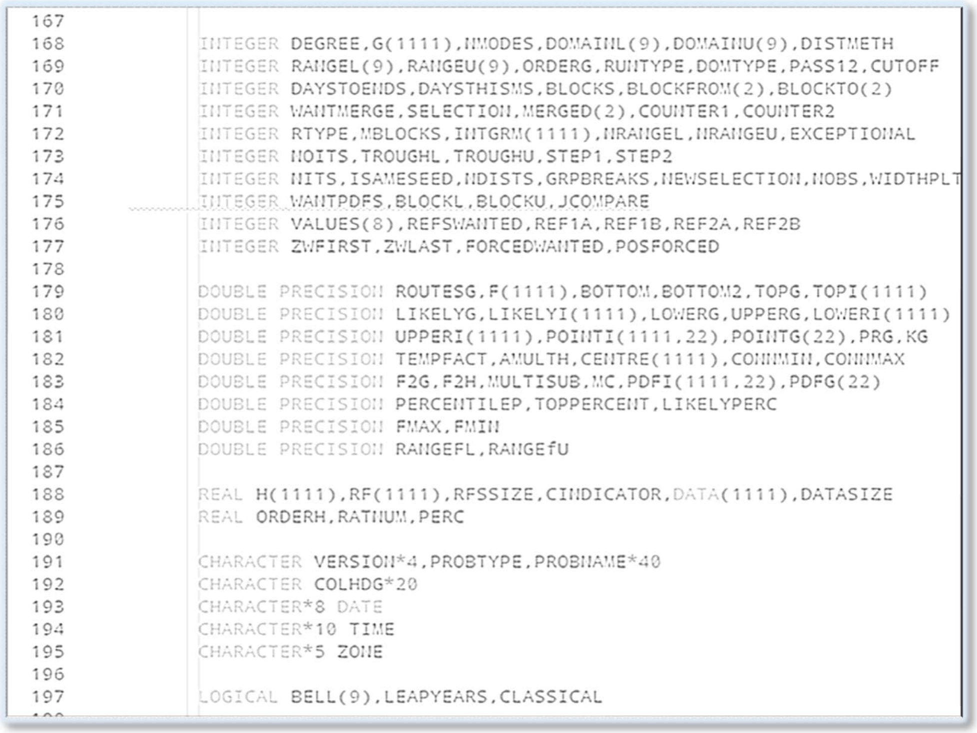
GREAT LIKELINESSES USER MANUAL
Release E02

R.Lovegrove

GREAT LIKELINESSES is a program for the generation of samples of algorithmically-defined distributions and using them to estimate mean probabilities.



```
      INTEGER DEGREE,G(1111),NMODES,DOMAINL(9),DOMAINU(9),DISTMETH
      INTEGER RANGEL(9),RANGEU(9),ORDERG,RUNTYPE,DOMTYPE,PASS12,CUTOFF
      INTEGER DAYSTOENDS,DAYSTHISMS,BLOCKS,BLOCKFROM(2),BLOCKTO(2)
      INTEGER WANTMERGE,SELECTION,MERGED(2),COUNTER1,COUNTER2
      INTEGER RTYPE,MBLOCKS,INTGRM(1111),NRANGEL,NRANGEU,EXCEPTIONAL
      INTEGER NOITS,TROUGHL,TROUGHU,STEP1,STEP2
      INTEGER NITS,ISAMESEED,NDISTS,GRPBREAKS,NEWSELECTION,NOBS,WIDTHPLT
      INTEGER WANTPDFS,BLOCKL,BLOCKU,JCOMPARE
      INTEGER VALUES(8),REFSWANTED,REF1A,REF1B,REF2A,REF2B
      INTEGER ZWFIRST,ZWLAST,FORCEDWANTED,POSFORCED

      DOUBLE PRECISION ROUTESG,F(1111),BOTTOM,BOTTOM2,TOPG,TOPI(1111)
      DOUBLE PRECISION LIKELYG,LIKELYI(1111),LOWERG,UPPERG,LOWERI(1111)
      DOUBLE PRECISION UPPERI(1111),POINTI(1111,22),POINTG(22),PRG,KG
      DOUBLE PRECISION TEMPFACT,AMULTH,CENTRE(1111),CONNMIN,CONNMAX
      DOUBLE PRECISION F2G,F2H,MULTISUB,MC,PDFI(1111,22),PDFG(22)
      DOUBLE PRECISION PERCENTILEP,TOPPERCENT,LIKELYPERC
      DOUBLE PRECISION FMAX,FMIN
      DOUBLE PRECISION RANGEFL,RANGEfU

      REAL H(1111),RF(1111),RFSSIZE,CINDICATOR,DATA(1111),DATASIZE
      REAL ORDERH,RATNUM,PERC

      CHARACTER VERSION*4,PROBTYPE,PROBNAME*40
      CHARACTER COLHDG*20
      CHARACTER*8 DATE
      CHARACTER*10 TIME
      CHARACTER*5 ZONE

      LOGICAL BELL(9),LEAPYEARS,CLASSICAL
```

The recommended citation for this manual is

Lovegrove,R.,(2020),`Great Likelinesses (release E02) User Manual', Lovegrove Mathematicals, London, November 2020
London

United Kingdom
November 2020

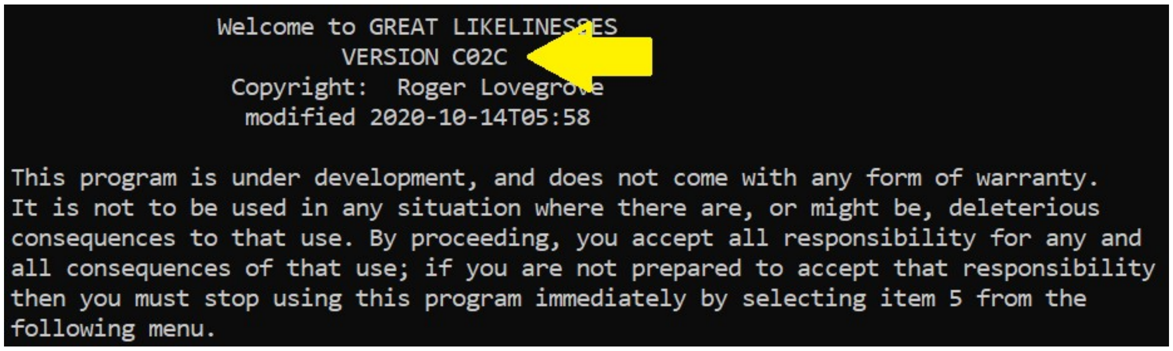
www.lovegrovemaths.co.uk

Contents

1 Introduction

GREAT LIKELINESSES (the name is a pun on that of Charles Dickens's novel *Great Expectations*) is a program for producing distributions of various types and using them, if desired, to calculate likelinesses, that is best-estimates of probabilities.

If the version number shown on the opening screen ends in a C then you are using a classical version which has been sent to someone (presumably you?) for evaluation. If it ends in an F then you have a full version.



```
               Welcome to GREAT LIKELINESSES
                        VERSION C02C
                Copyright:  Roger Lovegrove
                 modified 2020-10-14T05:58

This program is under development, and does not come with any form of warranty.
It is not to be used in any situation where there are, or might be, deleterious
consequences to that use. By proceeding, you accept all responsibility for any and
all consequences of that use; if you are not prepared to accept that responsibility
then you must stop using this program immediately by selecting item 5 from the
following menu.
```

In a classical version, the degree is limited to values which have some significance in classical probability problems. If you try specifying any other value then the program will stop immediately. The acceptable degrees are: -

2,3,4,5,6,8,13,16,26,32,36,39,52,64,128,216,256,365,512,1024

In a full version, the degree can be any whole number from 2 to 1111.

2 There is no warranty

This program is currently under development, and is not intended to be used in any situation where there are or might be deleterious consequences arising from that use.

There is no warranty of any form. For example, there is no warranty against failure of the program, or against failure of the program to produce the correct result.

By using the program, you accept full responsibility for all the consequences of that use. If you are not willing to accept that responsibility then do not use the program. The program asks for positive verification that you do accept that responsibility before it accepts any further input from you.

3 Amendments implemented in this release

Amendments implemented in this release include

- Expanded multi-modal functionality

The nature of amendments is usually such that existing versions of DETAILS.TXT and DEFAULTS.TXT will be rendered unusable. When running this version for the first time, please select item 999 from the opening menu; otherwise the program will not run properly (or at all).

4 Known Issues

4.1 General comments about over/underflow

The number-crunching involved really can push your computer to its limits. The following should be avoided if possible:-

- High degree;
- Large number of iterations;
- Overly-complicated underlying set;
- Merge block(s)
- Given histogram with large sample size;
- Required integram with large sample size.

Any one of these, if taken to extremes, but especially any combination of more than one, can cause a run-time error. If this should happen to you then you will probably firstly want to reduce the number of iterations. Alternatively, or additionally:-

- You might consider reducing the degree, for example by measuring the independent variable in larger units (eg. weekly data rather than daily).
- Ask yourself whether you really do need a complicated underlying set. It is very easy to impose far more conditions than you would ever dream of using with a traditional closed, parametric analysis.

Due to interactions between the various items, it is not possible to give precise guidance about the meanings of "High degree" etc. However, anything which increases the chances that the program will encounter a term of the form $f(i)^x$ where f(i) is very small but x is large is not a good idea.

If there is any chance that you might run into difficulties of this type then you should use the sampling files, or some other technique, to model your observational or experimental program *before* carrying it out.

4.2 Poor choice of contraction

If you choose the centre of a contraction to be a point which has a co-ordinate of 0 and then also specify a magnification of 0 then the result will be a function which does not meet the definition of a "distribution". Depending on the context of the rest of the problem, this can give a run-time error.

This is not considered to be a serious problem, since it should never arise in a genuine analysis.

5 Software relationships

5.1 Running from Windows

Even if you have DOS or a DOS-equivalent (such as FreeDOS), it is still recommended that you run the program from within Windows rather than by switching, firstly, to DOS. This is because the program can be so fast with simpler analyses that the screen buffer cannot keep up and so forces the program to slow down significantly. Windows has improved screen-buffering which largely overcomes this. Just switch to your File Manager and double click on the .exe file.

5.2 Using Spreadsheets

The .csv files have been designed to be viewed within a spreadsheet.

Open your spreadsheet by right clicking on the icon for the file you want to open and selecting `open with'. You should then be offered the choice of programs to use to open the file. Choose your favourite spreadsheet: you should then be offered a choice of options defining how the spreadsheet is to interpret the file.

5.2.1 Microsoft Excel

As the separator, choose a comma (,). As the text delimiter, choose a double quote ("). Do not choose to merge successive delimiters.

5.2.2 OpenOffice/Apache OpenOffice Calc

As the separator, choose a comma (,). As the text delimiter, choose a double quote ("). Do not choose to merge successive delimiters. For versions 3.3 and later of Calc, select `detect special numbers' (you will not be offered this option in earlier versions).

5.2.3 LibreOffice Calc
As the separator, choose a comma (,). As the text delimiter, choose a double quote ("). Do not choose to merge successive delimiters. Select `detect special numbers'.

5.2.4 Lotus 1-2-3
[Support for Lotus 1-2-3 was withdrawn by IBM (its then-owner) in September 2014. Consequently, you may find it beneficial to run it in Compatibility Mode.]

Choose `Parse as CSV'. When the spreadsheet opens, it may seem that some of the fields have been asterisked out: they have not -it's just that the default column widths are too small.

In all spreadsheets, when viewing RESULTS.CSV, you might find it helpful to widen Columns A and B.

6 Basics

6.1 Basic concepts
A coin is of degree 2; a die is of degree 6; a pack of cards is of degree 52. The degree is the number of possibilities that something (tossing a coin; rolling a die; drawing a card) may take}. It is the number of classes in a classification system.

The possibilities/classes are labelled 1,2, . . . ,N where N is *the degree*.

Given a degree, we can always define an *histogram* with that degree, e.g.

Table 1: Histogram

i	1	2	3	4	5	6
h(i)	3.0	4.2	0	0.7	1.8	0

The values in an histogram are all non-negative. If, additionally, they are all non-zero and sum to 1, then that histogram is a *distribution*:-

Table 2: Distribution

i	1	2	3	4	5	6
f(i)	0.2	0.1	0.35	0.05	0.25	0.5

If the values are all integers, then the histogram becomes an *integram*:-

Table 3: Integram

I	1	2	3	4	5	6
g(i)	3	4	0	0	1	0

The most important histogram is the integram 0 = (0,0, . . ,0)

If f is a distribution and g is an integram, both of degree N, then the probability of g given f is $\Pr(g|f)= M(g)f^g$, where M(g) is the multinomial coefficient associated with g, and $f^g=f(1)^{g(1)}. . .f(N)^{g(N)\}}$.

$$M(g) = \frac{\omega(g)!}{\prod_{i=1}^{N} g(i)!}$$

The expression for M(g) contains the product of factorials, so each g(i) has to be integer-valued, so g has to be an integram.

The best-estimate of something is its mean value. In order to have a mean value, there must be a set of values to find the mean of. This program best-estimates Pr(g|f) so there has to be a set of Pr(g|f) to find the mean of. We start with a set of distributions, called the *Underlying Set*, calculate Pr(g|f) for each f in that set and then find the mean of *those*.

The mean used is a weighted mean, where the weights depend upon available data in the form of an histogram (called the *given histogram*). The actual formula used is

$$L_P(g|h) = M(g)\frac{\int\limits_{f\in P} f^g f^h}{\int\limits_{f\in P} f^h}$$

The f^h terms started life as Pr(h|f), that is $M(h)f^h$, but the M(h) has cancelled since it appeared in both the numerator and denominator. The cancellation of the M(h) has taken with it any need for h to be integer-valued so h may be an histogram rather than specifically an integram.

In a few simple cases the integrals on the RHS can be evaluated theoretically. Usually, however, a numerical approach is needed; that is what this program does. In essence, the process is very simple: we replace the integrals by summations and the underlying set, P, by a sample of distributions selected at random from P. That sample can be surprisingly large: about a million are often needed -the program defaults to 750,000- but 100 million or more can at times be necessary. It is only recently that improvements in computer technology have made it possible for such large problems to be tackled on home computers.

6.2 What the program does

The program:-

1. Finds $L_P(g|h)$, and other standard best-estimates, (means) by sampling from P.
2. Keeps track of convergence as the sampling process proceeds.
3. Produces a separate sample of distributions from P, and uses each as a generating distribution to generate simulated observations.
4. Calculates the best-estimate of the probability (ie. the likeliness) that $Pr(g|f)<x_i$ for a selection of $x_i$$ equally spaced across some interval specified by the user. Likewise for Pr(1|f), . . .,Pr(N|f)$. This is the likeliness equivalent of building up a CDF.
5. For each of the subintervals $[x_i,x_{\{i+1\}}]$ calculates the Likeliness that Pr(g|f) lies in that subinterval. This is the likeliness equivalent of building up a PDF.

It does this for an underlying set, P, chosen by the user from those listed later in this manual.

In addition, the program can carry out various manipulations on the underlying set, such as contracting it onto any centre. It can also handle merged data-blocks and filtering.

6.3 Files

Table 4: Files used by program

File name	Purpose
data.txt	User-maintained file to store input data
details.txt	Keeps details of problem, for future editing and/or use
defaults.txt	Various defaults, to personalise the program
results.csv	The results of the analysis, for viewing in a spreadsheet
debug.txt	Keeps track of the progress of the program
sampling_dis.csv	Sample of distributions
sampling_obs.csv	Observations simulated by using the distributions in sampling_dis.csv
sampling_rfs.csv	Relative frequencies for the observations in sampling_obs.csv
sampling_dis.txt	Simplified text version of sampling_dis.csv
sampling_obs.txt	Simplified text version of sampling_obs.csv
sampling_rfs.txt	Simplified text version of sampling_rvs.csv
errlog.txt	Keeps track of run-time errors
scratch1.txt scratch2.csv	Scratchpads for the program's own internal use

6.3.1 details.txt and results.csv

When you start a new problem, you type it in from the keyboard. Details are saved in the file details.txt so that you do not have to type them in again on subsequent runs. You can change some aspects of a problem by editing data.txt in a text editor.

Results are sent to the file results.csv for viewing in a spreadsheet.

details.txt will be overwritten the next time that you run a problem from the keyboard; results.csv will be overwritten the next time you run any problem. If you want to keep data.txt or results.csv beyond that then make a copy, under a different name, in the usual way.

6.3.2 Location of files

The files listed in Table 4 will be placed in the same folder that you place the .exe file.

7 Countdown

While the program is carrying out an analysis, a countdown-to-completion is sent to the screen so that you can see progress. The analysis is in two parts: a fast initial pass, during which various items are roughly estimated so as to improve the efficiency of the program, followed by a slower second pass during which likelinesses are found. During the second pass, the countdown includes details of the estimated run-time: these estimates will be thrown out if the computer is used for any other purpose while the program is running.

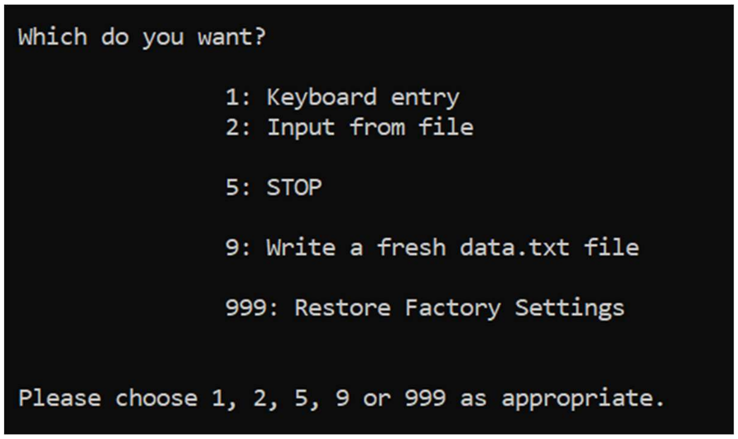
8 What you should do now

1. Form a new folder to contain the files used by this program
2. Transfer the .exe file and this manual into that folder
3. Run the .exe file twice.
   - Firstly, select item 999 from the opening menu: this will set up various defaults.
   - Secondly, select item 9 to set up a template for data.txt

You can then experiment with the program, to get a feel for what's going on.

9 Running the program

To start the program, run the .exe file by double-clicking on it in the file manager.



```
Which do you want?

               1: Keyboard entry
               2: Input from file

               5: STOP

               9: Write a fresh data.txt file

               999: Restore Factory Settings


Please choose 1, 2, 5, 9 or 999 as appropriate.
```

The opening screen gives you various options. If this is the first time that you have ever run this version then you must choose option 999 in order to set up various defaults, followed by item 9..

Otherwise, your choice will normally be between Option 1 if you are starting a completely new problem, or Option 2 if you are repeating a previous problem or running a modification of one.

To start a new problem, choose Option 1. You will then be asked a number of questions, the answers to all of which will be numerical. For most of these, you will develop standard answers which you will soon get used to giving very quickly; with practice, your fingers will type most of the answers faster than you think of them.

If you choose Option 2 then the computer will just take over and run the problem, giving you an on-screen progress report -which, for simpler problems, might flash past so quickly that you are unable to read it.

10 Stopping

The program will normally come to a stop of its own accord. There are times, though, when you might want to stop it prematurely.

You can choose `Stop' from the opening menu. This will stop the program before it has really started. Rxisting files will remain unatered.

Otherwise, simply close the window in which the program is running. However, if you do so during data-input from the keyboard then this will cause the file details.txt to be incomplete and therefore unusable on future runs.

11 Inputting data

11.1 Basics

Data can be input in several ways. These can be mixed-and-matched.

- As a typed list of observations
- As a typed sample size and a list of relative frequencies
- From a file which may contain either observations or relative frequencies (not both at the same time)
- Merge blocks

11.2 Typing a list of observations

Because the practical use of likelinesses is aimed towards smaller sample sizes, data will usually consist mostly of zeroes. To make inputting easier, you start by specifying a block about which you wish to be asked; the program defaults values outside that block to zero. Only details about that block need to be stored in details.txt (See section 17)

Negative values are not allowed. Values inside the specified block may be zero.

11.3 Typing Relative Frequencies

You will firstly be asked for a sample size then for the relative frequencies.

As when typing observations, you specify a block about which you wish to be asked, and RFs outside that block will default to 0.

The RFs cannot be negative.

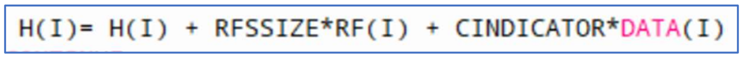
To reduce errors, and also to make things easier for you, the RFS do not need to sum to 1: the program will normalise them for you. This means that at least one of the RFs must be non-zero.

The normalised RFs are multiplied by the sample size to re-create the original data.

11.4 Reading data from a file

- The file must be called data.txt
- Use of data.txt must be switched on in defaults.txt
- The first item in data.txt must be a non-negative number, which will be called the contents-indicator.

The contents-indicator is analogous to the sample size in relative frequencies: it is used to multiply each other item in data.txt before being added into the observations.



```
H(I)= H(I) + RFSSIZE*RF(I) + CINDICATOR*DATA(I)
```

This means that

- If the contents-indicator (CINDICATOR) is 0 then the other contents of data.txt are all multiplied by 0, i.e. are in effect ignored;
- If the contents-indicator is 1 then the remaining items are added straight into H, i.e. are treated as actual observations.

Two conditions must be met before data.txt can be used:

1. The appropriate switch must be set in DEFAULTS.TXT *and*
2. The contents-indicator must be set to any positive,non-zero value.

```
(SAMPLE SIZE/CONTENTS-INDICATOR)
0
(0=Ignore this file)(1=Use the file contents as data rather than RFs)(Any other positive value=sample size)


(I=1)
0.0
(I=2)
0.0
(I=3)
0.0
(I=4)
0.0
(I=5)
0.0
```

A basic all-zero data.txt can be formed by selecting the appropriate item from the opening menu. As doing so will cause the loss of the current file, the user is twice asked to confirm that this was intended.

There is no harm in having too many items in data.txt: the program will simply read from the start of the file until it has read as much as it needs, and then ignore the rest. Having too few items will cause a run-time failure (EOF read). For this reason, the all-zero file has the maximum permissible number of items, which should normally be left in place.

It is possible to edit the labels and titles within data.txt if you wish. But this is a specialised task which you are strongly advised not to do unless you *really do* know what you are doing.

The problem is that you might form a string which the computer interprets as a number. If this should happen then the computer will from that point onwards be thrown "out of sync" and read the wrong values into the wrong variables.

You are advised to carry out a test-run after editing data.txt, to make sure that everything works correctly.

11.5 Merge Blocks

A merge block occurs in a table of data if there is a record of the total number of observations within a contiguous block of columns, but records have not been kept of the detailed figures for each column.

A common type of merge block is when data is shown as "3 or fewer", or "6 or more" or similar phrases.

Great Likelinesses can cater for three situations, as shown in the following tables.

Table 5: Merge Block at left

3 or fewer	4	5	6	7	8
9	2	0	3	3	1

Table 6: Merge Block at right

1	2	3	4	5	6-8
2	1	6	2	0	7

Table 7: Merge Blocks at each end

1-3	4	5	6-8
9	2	0	7

In practice, when data is collected in batches it sometimes happens that some batches contain a merge block but others do not. To cater for this, the program allows merge blocks to be entered independently of H1 and H2, and to overlap their non-zero entries.

The degree is the number of columns there would have been had the merging not taken place; it is 8 in each of the above tables.

The use of Merge Blocks introduces such vagueness into a problem that the resulting algorithms can be highly ill-conditioned. A significant increase in the number of iterations will be needed, but will not necessarily improve convergence to an acceptable extent; indeed, increasing the number of iterations could make convergence properties worse rather than better. For this reason, Merge Blocks should always be used with caution.

If there is a Merge Block, the program uses the information about the Merge Block to recreate the third histogram, H3. H3 is created dynamically, every time that a new distribution is selected from the underlying set and will change as that distribution changes, so it is not possible to specify it. This continually-changing nature of H3 is a significant component of the vagueness which is introduced by the use of merge blocks.

12 The required integram

The required integram, g, is specified in the same way as is H1: by giving a block about which you want to be asked, with values outside that block defaulting to zero.

The calculated Likeliness of g is shown on-screen as part of the countdown display. This means that convergence can be monitored whilst the calculations are proceeding.

13 Percentile

As part of the input routine, you are asked for a percentile for the program to estimate.

You are not asked whether or not you want a percentile, since it takes as much effort to say “No” as it does to give one. If you don't want a percentile then just reply “2 for No” and ignore the resulting 2nd percentile.

Note the form of input. If you want the 5th percentile then input 5, not 0.05 . You are not restricted to integers, so you could ask for the 2.5th percentile, but this is not usual.

Let's say that you input p, then the program calculates

$$L_P(p\%|h) = \frac{\Sigma_{f \in P} f_{p\%} f^h}{\Sigma_{f \in P} f^h}$$

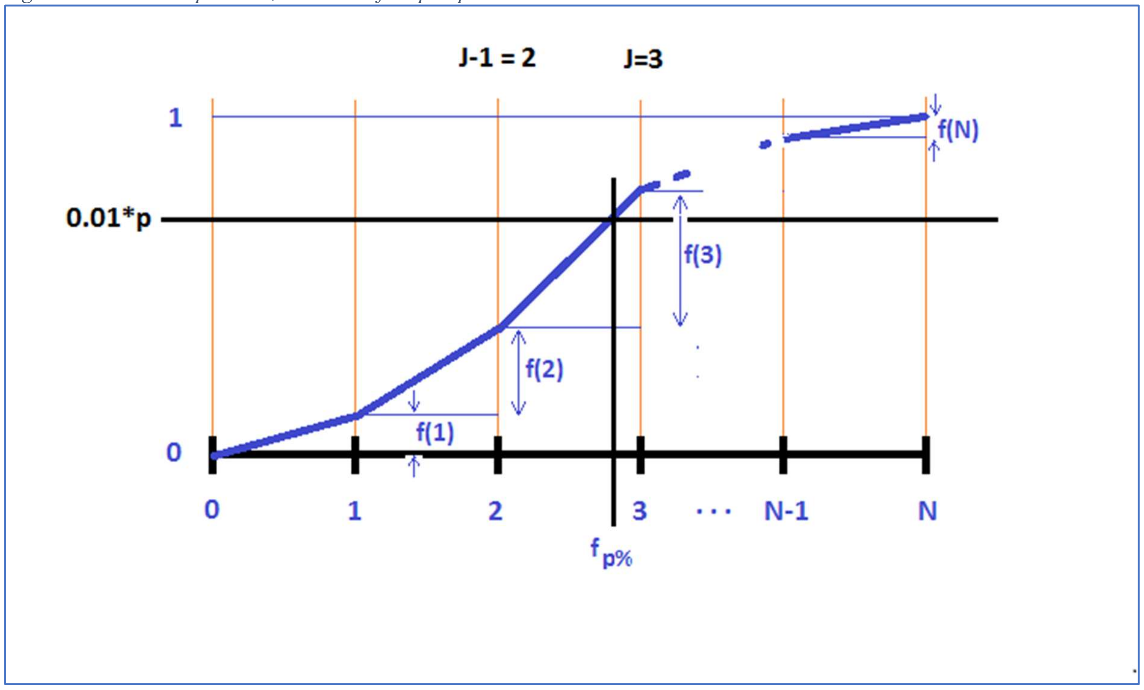
where $f_{p\%}$ is the calculated p'th percentile of f.

To find $f_{p\%}$ it is necessary to interpolate the CDF of f -so the result depends upon the method of interpolation. Also, there is no universally-agreed definition of “percentile” when the domain is finite. Consequently, percentiles on finite domains -no matter how they are calculated- should always be treated with caution.

13.1 Method of interpolation

Given a distribution f, the program firstly checks to find the value of J (J=1,...,N) for which 0.01*p lies between CDF[f(J-1)] and CDF[f(J)], where CDF[f(0)]=0. It does this by firstly checking J=1, then J=2 etc, ie by working from the left. Having found J, then program then uses linear interpolation between J-1 and J to find $f_{p\%}$.

Figure 1:Linear interpolation, as used to find p'th percenti



14 The underlying set}

14.1 Components of the underlying set

As a general rule, anything other than data (values of h(i) and g(i)) which has to be stated in order to define the problem forms part of the definition of the underlying set. Components include:-

1. fundamental type
2. degree
3. contraction
4. reflections
5. forced values
6. filters

14.2 Fundamental types of underlying set

14.2.1 Unstructured, S(N)

S(N) is the set of all distributions of degree N.

There is no relationship between the f(i) other than the requirement that they sum to 1. The fact that the domain, X_N, is ordered is irrelevant.

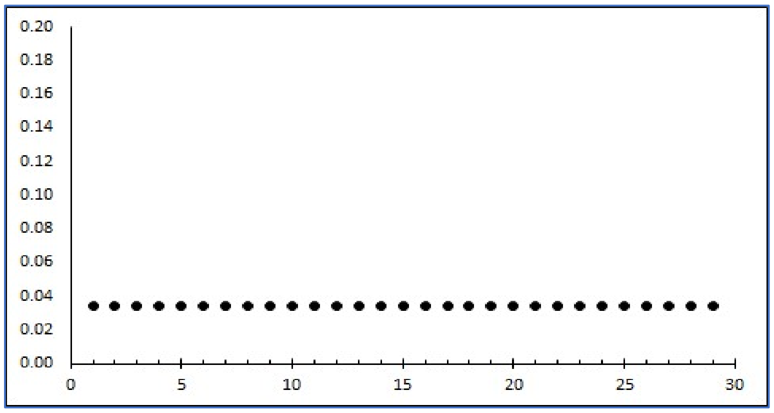


Figure 2: Likelinesses over S(29)

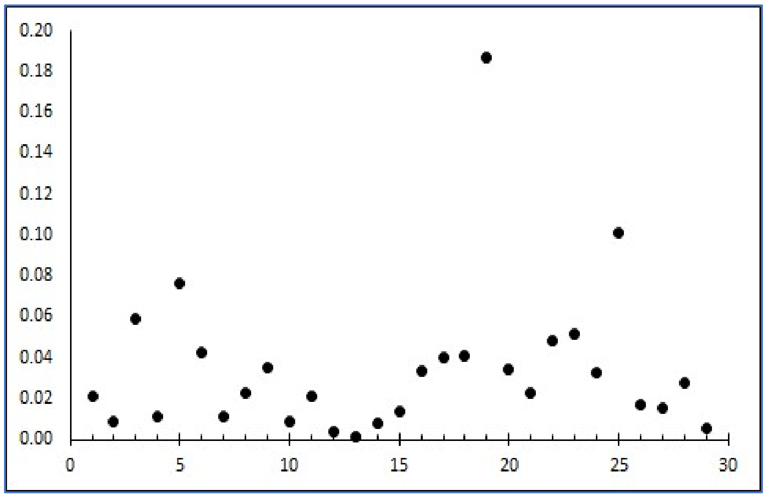


Figure 3: Individual element of S(29)

14.2.2 Ranked, R(N)

f is ranked if f(1)>f(2)>... >f(N).

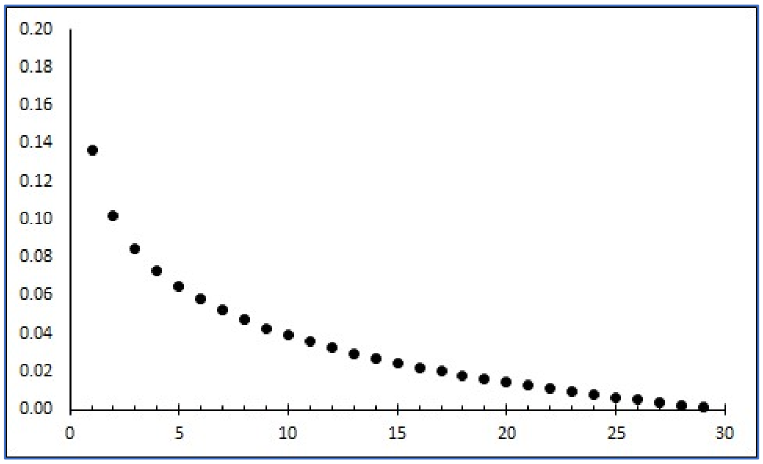


Figure 4: Likelinesses over R(29)

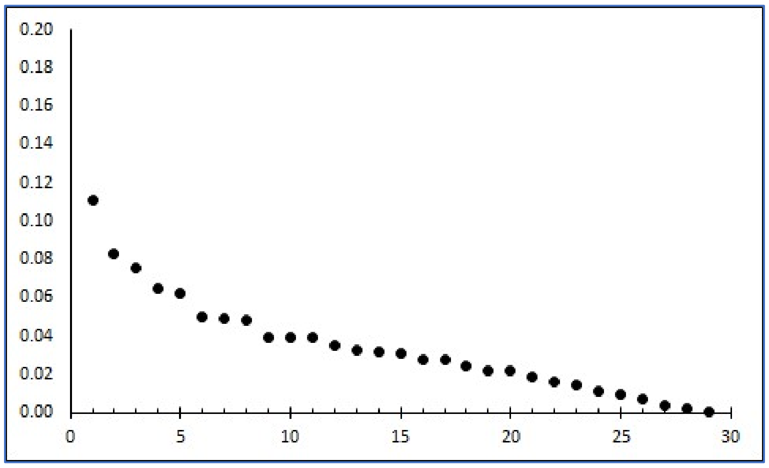


Figure 5: element of R(29)

14.2.3 Reverse-ranked, RR(N)

The mirror-images of ranked distributions, these increase to the right: f(1)<f(2). . .<f(N).

14.2.4 Step-down, SD(c,N)

c is called the "step". The function values 'up to' c are all greater than those 'above'; that is, if i≤c<k then f(i)>f(k).

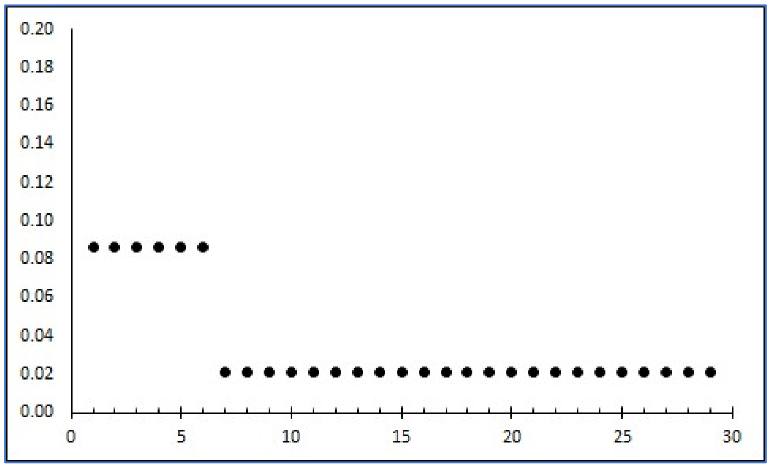


Figure 6: Likelinesses over SD(6,29)

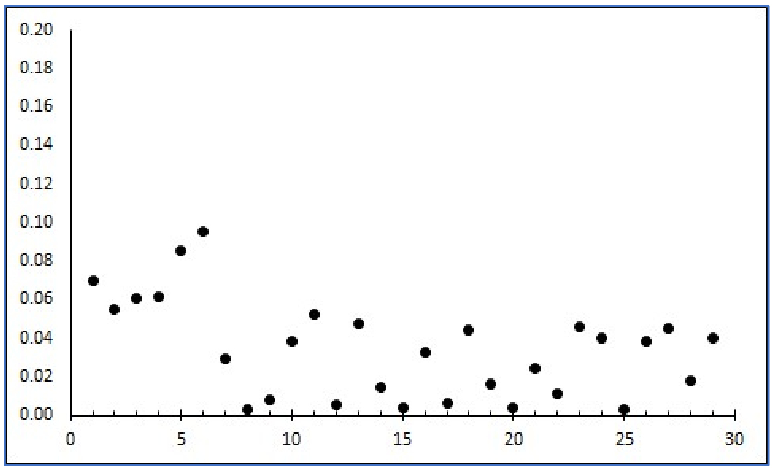


Figure 7: Individual element of SD(6,29)

14.2.5 Ranked step-down, RSD(c,N)

The same as step-down except that, in addition, the function values up to c are ranked; that is, if i<j≤c<k then f(i)>f(j)>f(k)$

c can sometimes be interpreted as a limit of discrimination for a ranked distribution.

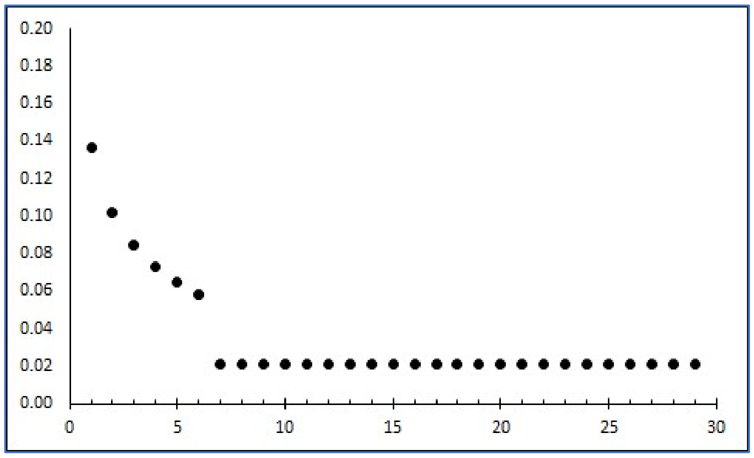


Figure 8: Likelinesses over RSD(6,29)

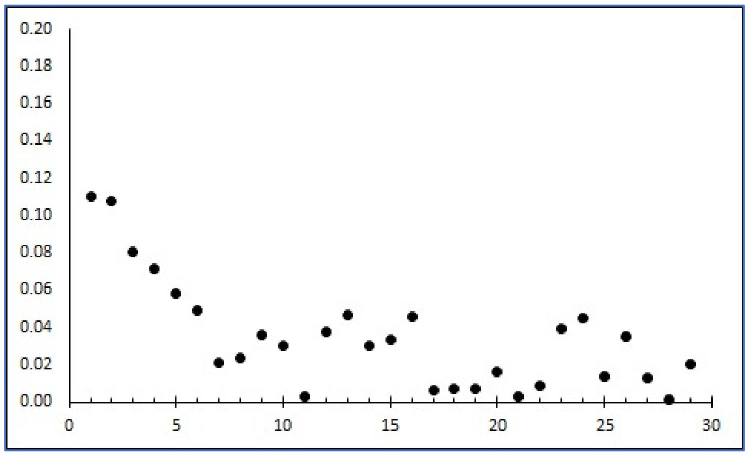


Figure 9: Individual element of RSD(6,29)

14.2.6 Tailed Step-Down, TSD(c,N)

Similar to ranked step-down except that it is the function values *above* (rather than *up to*) c which are ranked; that is, if $i \le c < j < k$ then $f(i) > f(j) > f(k)$.

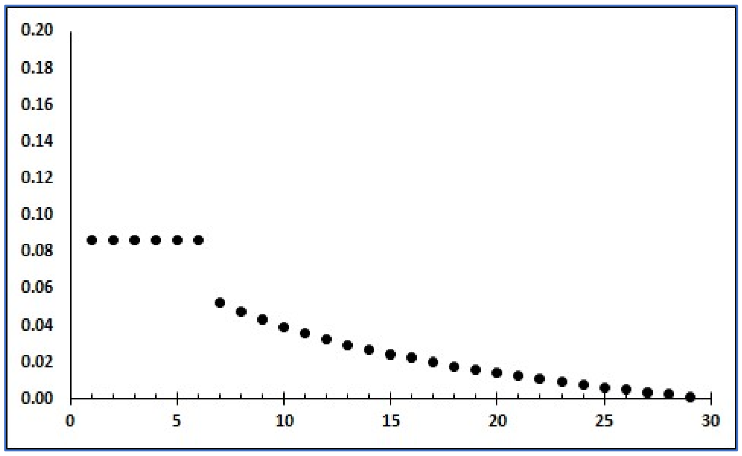


Figure 10: Likelinesses over TSD(6,29)

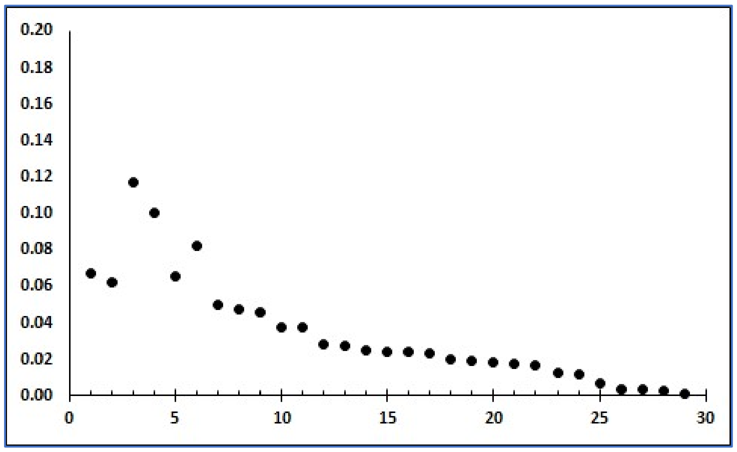


Figure 11: Individual element of TSD(6,29)

14.2.7 High/Medium/Low, HML(c,d,N)}

An HML distribution has two steps, subdividing it into 3 parts.

If $i \leq c < k \leq d < l$ then $f(i) > f(k) > f(l)$

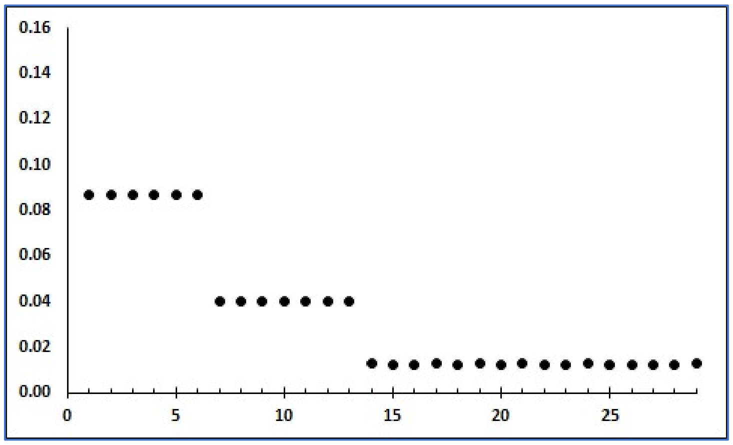


Figure 12: Likelinesses over HML(6,13,29)

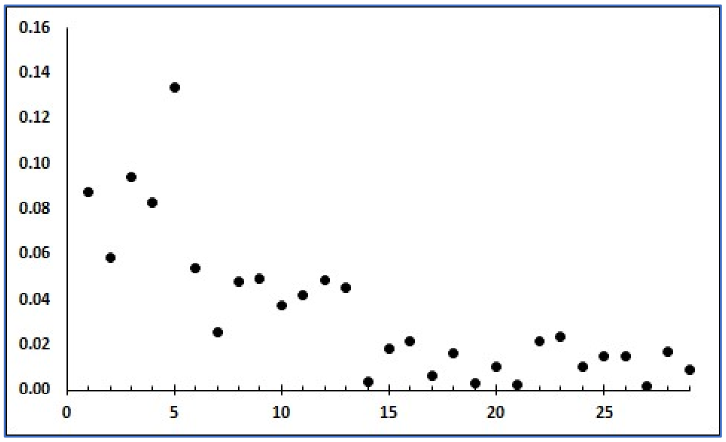


Figure 13: Individual element of HML(6,13,29)

14.2.8 Ranked High/Medium/Low, RHML(c,d,N)

A Ranked High/Medium/Low distribution is an HML distribution for which the `top step' is ranked.

If i<j≤c<k≤d<l then f(i)>f(j)>f(k)>f(l)

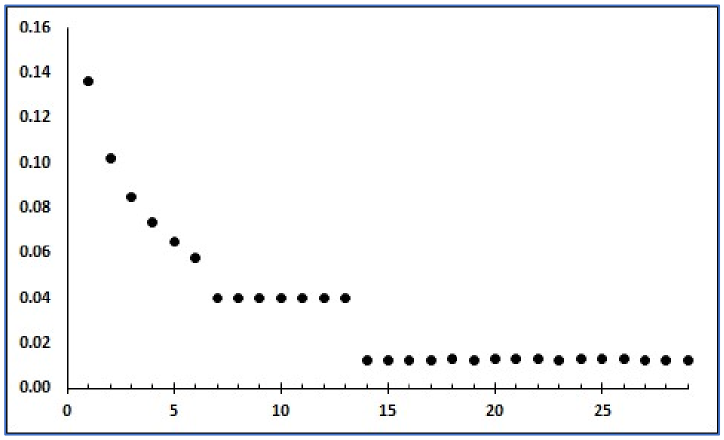


Figure 14: Likelinesses over RHML(6,13,29)

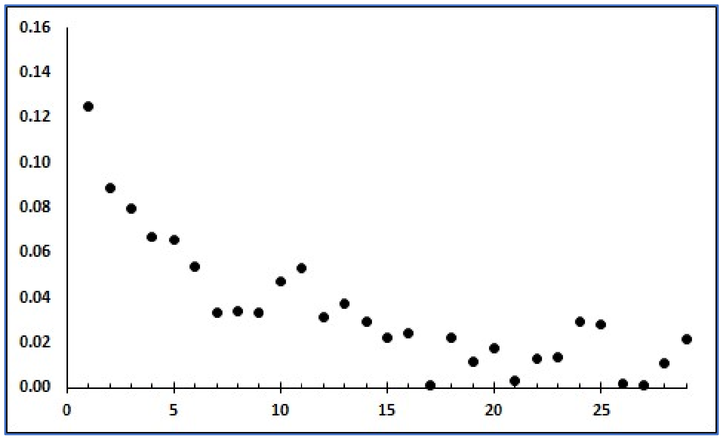


Figure 15: Individual element of RHML(6,13,29)

14.2.9 Unimodal, M(A to B,N)

Each distribution has precisely one mode. The mode does not have to be the same for every distribution used in the analysis, although it can be if so required. The program asks for a range of i which the mode may take; the smallest value is A and the largest is B. You will usually use one of two specific cases:-

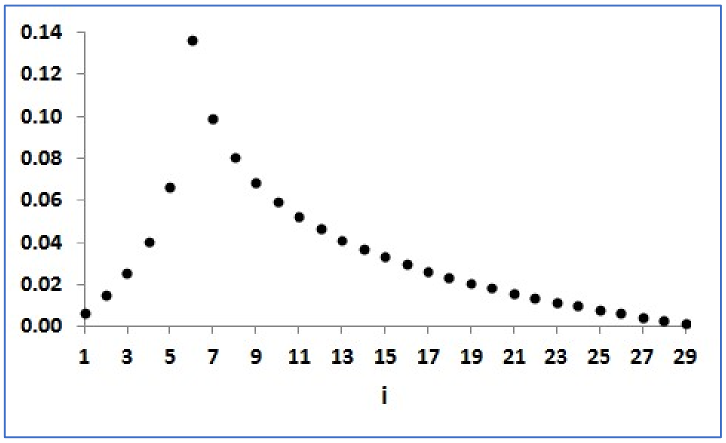


Figure 16: Likelinesses over M(6,29)

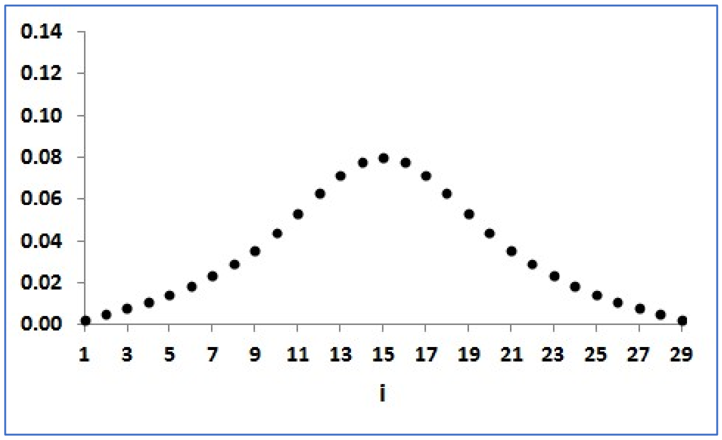


Figure 17: Likelinesses over M(29)

1. To use a specific mode, m, put A=B=m. We then write M(m,N) rather than M(m to m,N).
2. If A=1 and B=N then M(A to B,N) becomes M(1 to N,N), which is written as M(N). This is the set of all unimodal distributions of degree N.

Despite the impression given by some mathematical texts, unimodal distributions are not usually bell-shaped. You are offered the option to use only bell-shaped distributions, but you should not usually accept this offer.

There are two concepts of mode: the local and the global.

Local modes
i is a *local mode* of f if

1. i=1 and f(1)>f(2); *or*
2. i=N and f(N-1)<f(N); *or*
3. 1<i<N and f(i-1)<f(i)>f(i+1).

Global modes

i is a *global mode* of f if f(i)=max{f(j)|j∈X_N}.

This program uses local modes throughout. If you want to use unimodal distributions but with a global, rather than local, mode then use PLAT(1,A to B,N) instead.

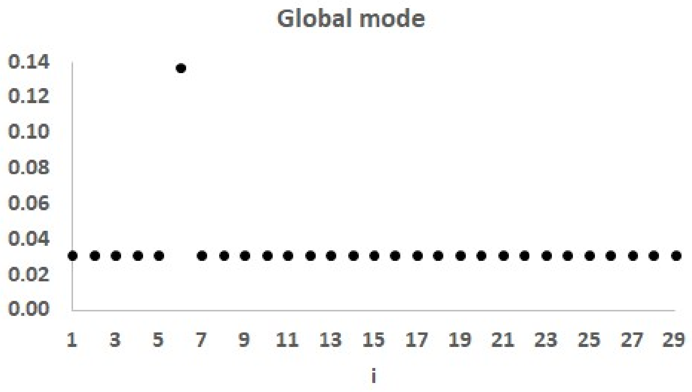


Figure 18: Likelinesses over PLAT(1,6,29)

14.2.10 Bell-shaped, B(A to B,N)

The notation used to refer to bell-shaped distributions follows that for Unimodal distributions, viz B(A to B,N), etc.

The definition used by Great Likelinesses is An unimodal distribution for which the absolute values of the first differences on either side of the mode are unimodal.

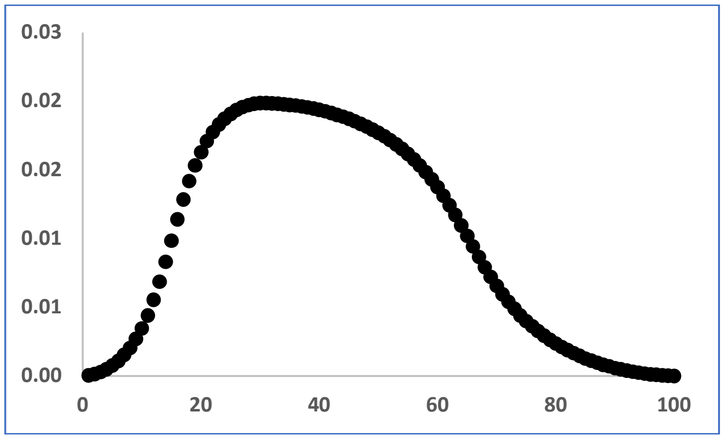


Figure 19: Likelinessses over B(30,100)

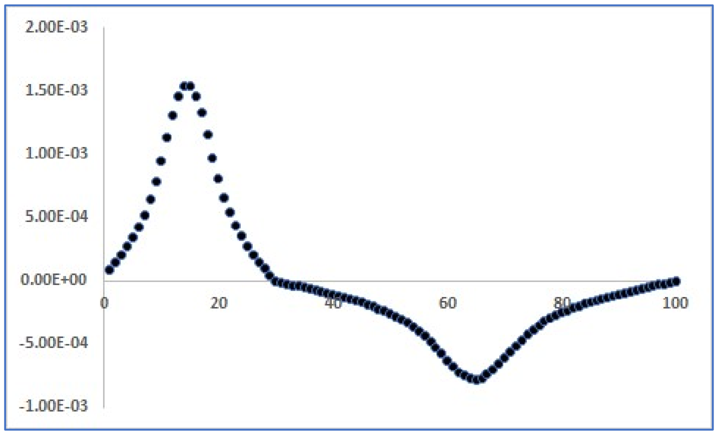


Figure 20: Forward differences of Likelinesses over B(30,100)

Figure 19 shows likelinesses over B(30,100), and Figure 20 shows their forward differences. The platykurtic shape of Figure 19 is due to the longer tail of the first differences on the right than on the left (because there is more room there).

14.2.11 U-shaped, U(A to B,N)

The logical dual of unimodal distributions, U-shaped distributions have a single `trough' rather than a single `mode'. Otherwise, there is no difference between the two.

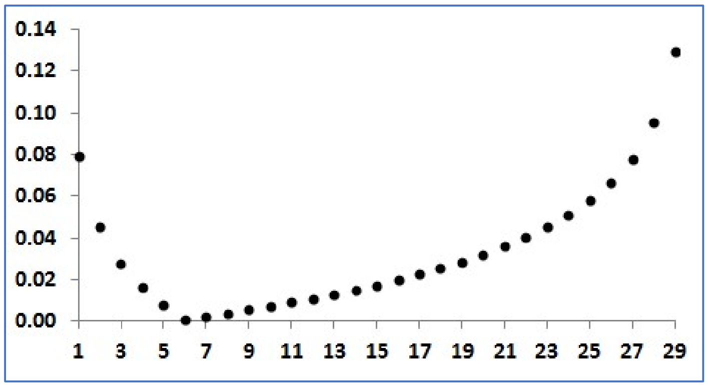


Figure 21: Likelinesses over U(6,29)

14.2.12 Multi-modal

The domain, X_N, is covered by several (maximum, 9) `mini' unimodal distributions, which may overlap. Each mini unimodal distribution has its own *essential domain*, over which that unimodal distribution takes non-zero values, and which is extended to cover the whole of X_N by using values of zero elsewhere.

During data-input, the user specifies the essential domain and the range of values that the mode make take. The combination of these two forms what is called a *piece*.

If the degree is no more than 77 (the maximum number of characters per line on some displays) then a visual *aide de memoire* will appear on the screen during data-input to help keep track of where the mini unimodal distributions are. This is switched off if the degree is more than 77.

Whenever a new multimodal distribution is needed, the program selects a mini unimodal distribution for each piece, and then uses them to produce the final distribution.

The user can use defaults.txt to choose between two ways for using the mini distributions to produce the final distribution; both methods require the specification of a set of **weights**. Those weights are randomly selected from one of five sets, specified as part of the data-input:

- S(p);
- R(p);
- SD(c,p) for some c;
- RSD(c,p)
- TSD(c,p)

where p is the number of pieces (modes).

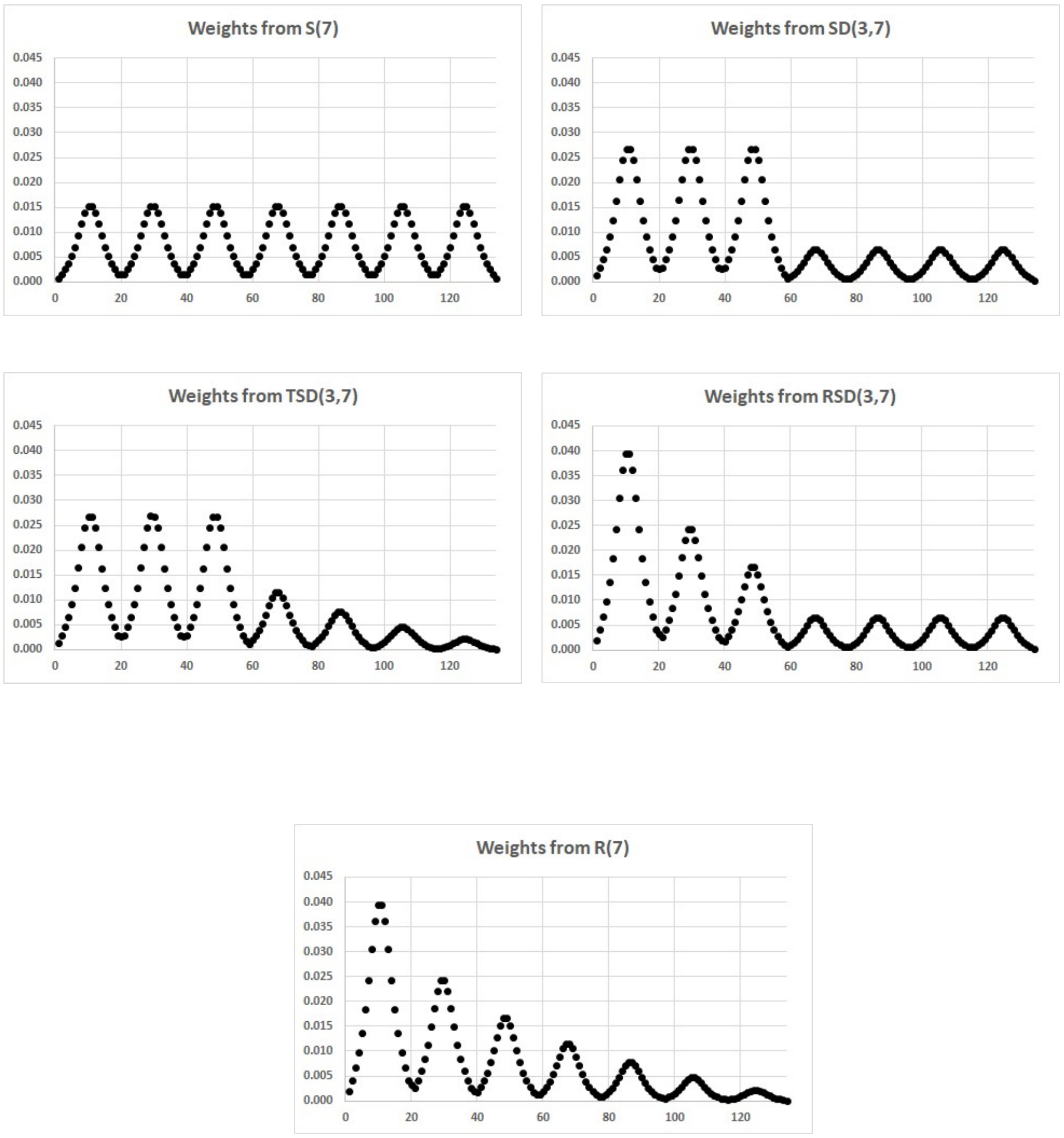
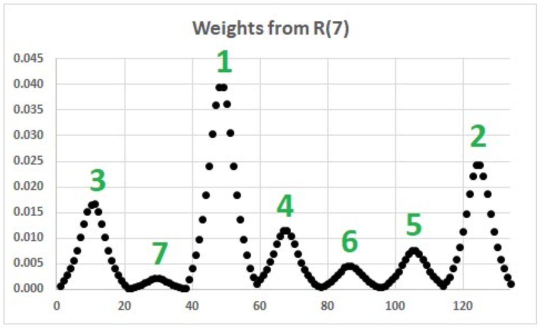


Figure 22: Example Likelinesses using different types of weights

The program applies weights to the pieces in the order in which their details are typed into the computer, which need not be left-to-right; this gives flexibility when defining the problem. For example, using the same pieces as used to produce the above figures, but entering their details in the order shown by the numbering gives the following with weights taken from R(7).



When a new multimodal distribution is being constructed, the program selects a single point from whichever of these sets the user had specified and then uses its co-ordinates as the weights.

There are two ways to use those weights:

- as coefficients in a linear combination of pieces;
- as probabilities of selection of unimodal distributions

14.2.12.1 Weights used as coefficients in linear combinations

Having selected the mini unimodal distributions, the program forms a linear-combination by using the weights as the coefficients.

14.2.12.2 Weights used as probabilities of selection

If:-

1. Each essential domain extends across the whole of X_N, *and*
2. You have chosen the option in defaults.txt to make this happen

then the weights, once chosen, will not be used as coefficients in a linear combination but, rather, as probabilities-of-selection. Every time a new distribution is needed, weights will be chosen and then one of the pieces will be selected to be used as the required distribution, the weights being the probabilities-of-selection of the pieces.

If the program is using linear combinations, then all distributions will come from the same multimodal population, but if it is using probabilities-of-selection then some will come from Piece 1, some from Piece 2, etc so there will be a distinct population for each piece.

- When weights are used as probabilities-of-selection, each distribution is unimodal. The multimodal effect arises as the result of selections from different populations

- When weights are used as coefficients, each distribution is multimodal

Consider the following example:-

Degree= 59
Multimodal with 3 pieces

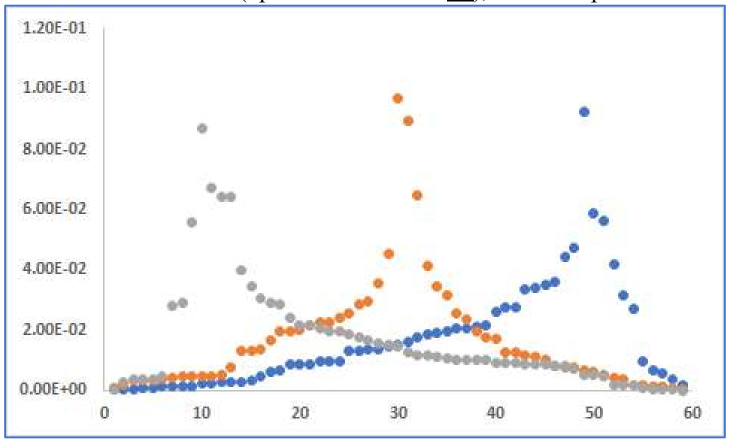
The modes are of equal dominance, so weights are to be taken from S(3)

For Piece 1 the essential domain goes from 1 to 59
The Mode is allowed to vary from 10 to 10

For Piece 2 the essential domain goes from 1 to 59
The Mode is allowed to vary from 30 to 30

For Piece 3 the essential domain goes from 1 to 59
The Mode is allowed to vary from 49 to 49

Using weights as probabilities -of-selection (option 1 in defaults.txt), three sample distributions were as shown



in

- Figure

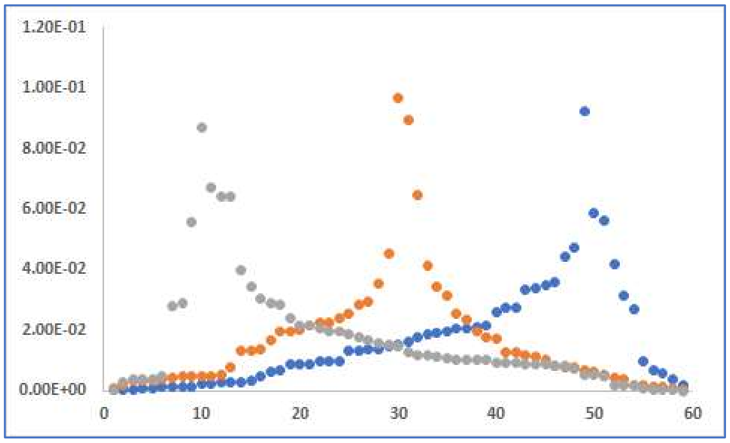


Figure 22: Weights used as probabilities-of-selection

- Using the weights as coefficients for linear combinations (option 2 in defaults.txt), a sample distribution was as shown in Figure 23

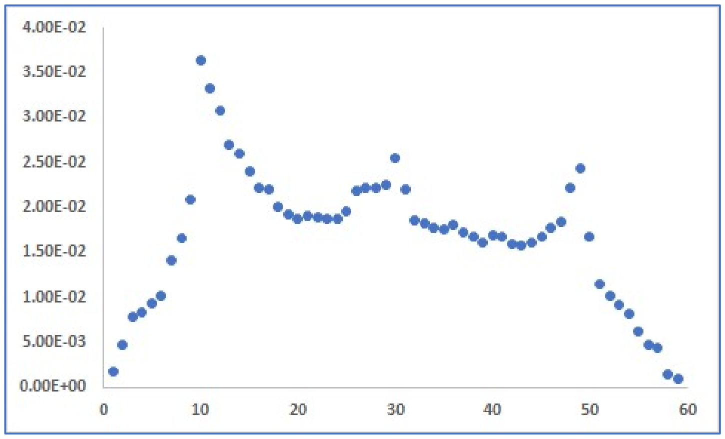


Figure 23: Weights used as coefficients

The choice of how to use weights affects the shape and distribution of individual distributions, but not their mean value (ie. the likeliness).

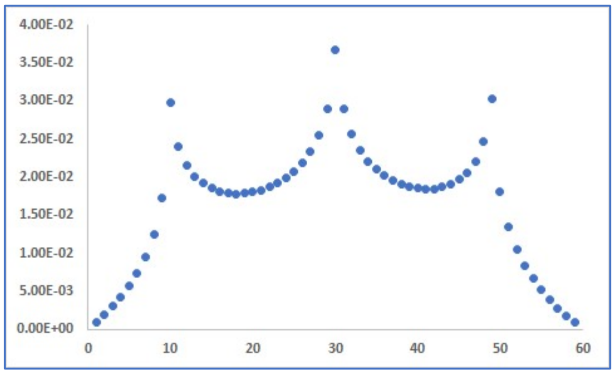


Figure 23: Likelinesses for each choice

Example

Specification. 7 modes, consisting of three zones:

- An <u>initial zone</u> of two pieces, with the second peak higher than the first
- A <u>transition zone</u>, of the next two pieces, in which values die away
- A <u>final steady-state zone</u> consisting of three pieces.

We model this by using 7 pieces with RSD weights with a step at 4, but with details of the left-most two pieces entered in right-to-left order.

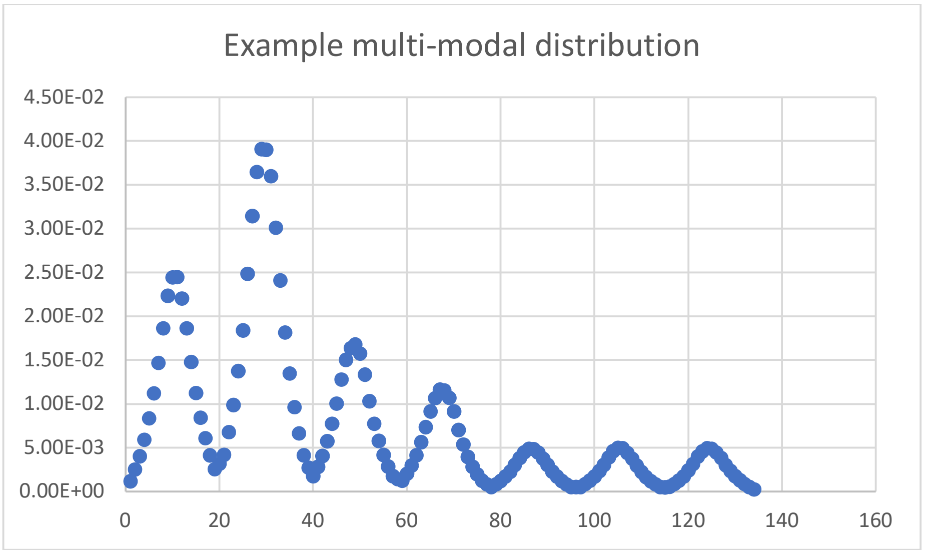


Figure 24: Example multi-modal distribution

14.2.13 12.2.13 Plateau, PLAT(w,A to B,N)

These are related to step-down distributions: take a step-down distribution and cycle it to the right.

Plateau distributions need two numbers: the width and the start of the plateau. If the plateau has width w and starts at b then it finishes at c=b+(w-1)

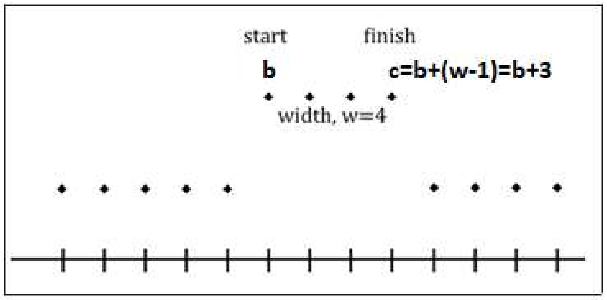


Figure 25: Terminologu for Plateau distributions

The user specifies the width and a range of values A to B for the start, where B+(w-1)≤N.

The set of all plateau distributions of degree N with width w which start in the range A to B is denoted by PLAT(w,A to B,N). If B=A then we write PLAT(w,A,N) rather than PLAT(w,A to A,N).

There are two special cases:-

1. PLAT(w,1,N) is SD(w,N);
2. PLAT(1,A to B,N) is equivalent to M(A to B,N) but using the global, rather than local, concept of mode.

14.2.14 Sequential systems, SS(N)

There is nothing special about those distributions produced as the outputs from sequential systems: every distribution can be considered as having been produced in that way.

What is special is the totality of such distributions rather than the individual distributions *per se*; in particular, the fact that the distributions are not spread uniformly over S(N). Figure Figure 25: Distribution of sequential systems shows 500 distributions in SS(3).

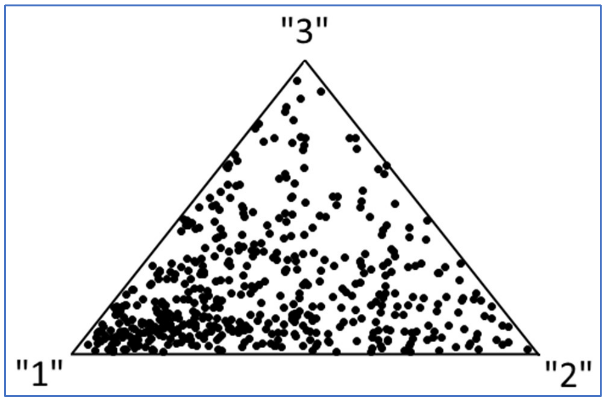


Figure 25: Distribution of sequential systems

The algorithm:-

1. Selects r∈R(N)
2. Divides throughout by r(1) to give the function F(i)= r(i)/r(1). (This function is ranked with F(1)=1)
3. Forms the forward differences f(i)= F(i)-F(i+1), with f(N)=F(N).

This process defines a bijection (but not a *linear* bijection) from R(N) to S(N). The point $L_{R(N)}$ maps to Zipf's Law.

The underlying set, SS(N), is the set of distributions produced by the program.

14.2.15 Random values

This underlying set comes with a health warning.

The use of random values is the naïve way to construct a random distribution: for each i, select a random value between 0 and 1, then normalise the result.

This is ***wrong.*** Using random values in this way does not produce an uniform covering of S(N).

It is doubtful whether you will have a need for this underlying set. If you want to select random distributions then use the set S(N)

14.2.16 Reverse-ranked with unimodal slope

(As the name suggests)

14.2.17 Ranked with ranked slope

(As the name suggests)

14.2.18 Ranked with unimodal slope

(As the name suggests)

14.3 Modifications of the underlying set

14.3.1 Contractions

A contraction is a mapping of the form f↦q+α(f-q). α is the *magnitude*, and q the *centre*, of the contraction.

A magnitude of 0 replaces every distribution by the centre: this gives a singleton underlying set.
A magnitude of 1 maps every distribution to itself, and so has no effect.

If you do want to have a contraction then you need to specify its centre together with upper and lower limits on the magnitude.

- When giving the centre, you can just type its co-ordinates. If the degree is large, however, this will involve a lot of typing and so be at high risk of including a typing mistake. The program therefore also gives you the choice of several standard cases.
- If you specify limits on the magnitude within 0.0001 of one-another then they will be given equal values.
- If you say that you do not want a contraction then details.txt will include the specification of a nominal contraction which has a magnitude of 1 and therefore has no effect. This is to make it easy for you to change the magnitude by editing details.txt. Strictly speaking, this means there will always be a contraction.

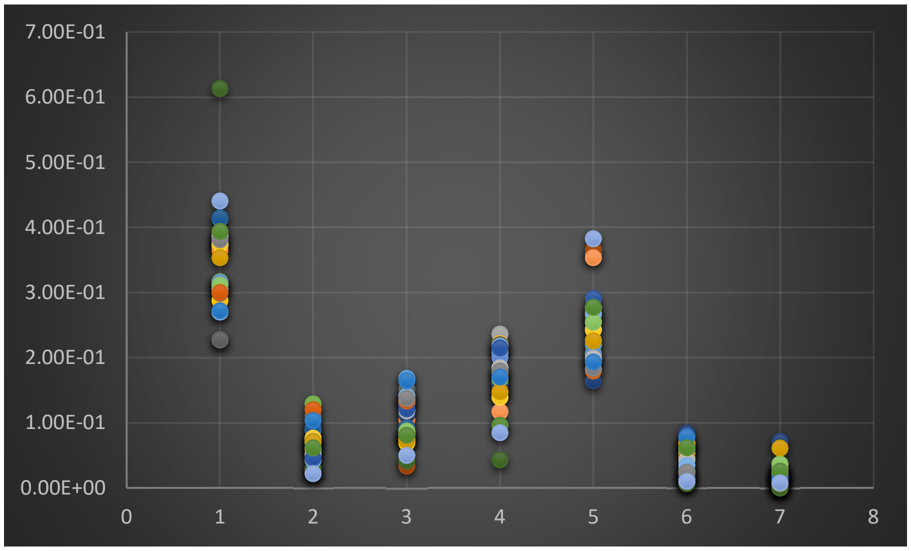
Whenever the program needs a contraction it will make a random choice of magnitude intermediate between the limits that you specified. If the limits are equal then that value will always be used.

- If the magnitude selected by the program is less than 0.0001 then the distribution being operated upon will be replaced by the centre of the contraction; this is equivalent to using a magnitude of 0.
- If the magnitude is greater than 0.9999 then the distribution will not be altered; this is equivalent to using a magnitude of 1.

(These bullet points are interpreted literally and so do cover negative magnitudes and magnitudes greater than 1.)

14.3.2 Reflections

The reflection of f from A to B takes the distribution f and reverses the order of f(A), ..., f(B). The reflection from 2 to 5 of (1,2,3,4,5,6,7) is (1,5,4,3,2,6,7). The reflection from A to A does nothing, ie is the identity mapping.



Great Likelinesses allows you to specify up to two reflections: one to be carried out before any contraction, and the other after. You are given the choice of four options:

- Choice 0: No reflection
- Choice 1: Use just a reflection before any contraction
- Choice 2: Use just a reflection after any contraction
- Choice 3: Use both reflections

Great Likelinesses always uses both reflections, to make it easy for you to modify an analysis by altering the stored details rather than having to type them in the correct format and position *ab initio*. If you say that you do not want either of them then that reflection is defaulted to the reflection from 1 to 1. details.txt always contains choice 3, followed by the specification of two reflections, defaulted if necessary.

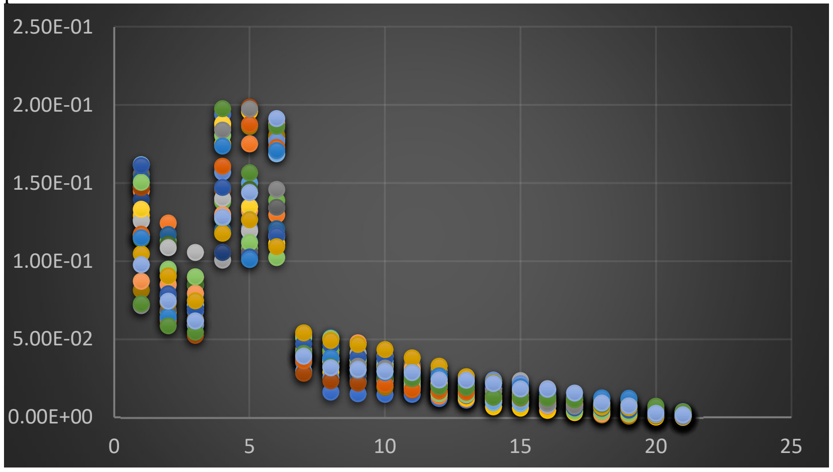
Provided you are not using a contraction, you can use the two reflections to interchange (ie. swap) f(A) and f(B) without affecting the intermediate values.

- If B=A+1 or B=A+2 then carry out the reflection from A to B
- If B>A+2 then carry out (in either order) the reflections from A to B and from A+1 to B-1.

14.3.3 Forced Values

You may force upper and lower limits on f(i) for a block of i. For example, you may require f(i) to be between 0.1 and 0.2 for i= 4,5,6.

The computer will firstly choose a distribution meeting your other specifications, and then it will overwrite the values inside the block by values chosen at random within the specified limits. That is, it retains the underlying set outside the block but destroys it inside the block, replacing it by random values. Making the substitution inside the block will, however, throw the overall sum off 1, so it needs to be brought back again by re-scaling. Since the values inside the block are at their specified values, they cannot be altered so all rescaling must take place outside the block.



When reading the following list, it must be appreciated that in FORTRAN it is not meaningful to ask if two REAL (or DOUBLE PRECISION) variables are equal: it is necessary, instead, to set a non-zero tolerance and to ask if they are within that tolerance of one-another. Generally speaking, *Great Likelinesses* uses a tolerance of 0.0001 for this purpose.

- Although the same limits will be used for all i in the block, each f(i) will be independently selected at random between those limits.
- The block must not extend over the whole of the domain, X_N. There must be at least one i not in the block. If this were not the case then there would be nothing that could be adjusted to bring the sum back to 1.
- If you place limits on n values of f(i) then the upper limit must be no more than 1/n, otherwise the sum of the values inside the block could exceed 1.
- If the limits are within 0.0001 of one-another then it will be assumed that you want them to be equal, so each will be replaced by their mean.

- If the upper limit is 0.9999 or more then it will be set to 1.0000
- If the upper limit is 1/n-0.0001 or more then the lower limit must differ by more than 0.0001

The final choice you have to make is where in the sequence of modifications the imposition of forced values is to take place. The other modifications are carried out in the order: reflection, contraction, reflection. It does not automatically follow that you will want the forcing of values to take place at the end of this sequence (although you normally will), so you need to say what you want. There are four possibilities: before the first reflection, before the contraction, before the second reflection, at the very end.

15 Filtering

Once a distribution has been produced, it can be subjected to an accept/reject process according to criteria which you may set.

- If accepted, it is included in the analysis
- If rejected, it is discarded and another distribution is generated in its place (and also subjected to the filtering).

Currently, two types of filtering have been enabled.

15.1 Type 1 filtering

You specify three things:

(i) A block of i which you want to be considered by the filtering process. For example, you might want to look at i=5, …, 10

(ii) A range of probabilities, eg 0.04-0.14
- 0 and/or 1 are allowed

(iii) Upper and lower limits on how many of the f(i)s within the block lie in that range
- The two limits may be the same
- 0 may be used as a limit

The program then looks at the distribution and counts how many of the f(i)s within the block do fall within the range given in (ii). If that number is (inclusively) within the limits given in (iii) then the distribution is accepted, otherwise it is rejected.

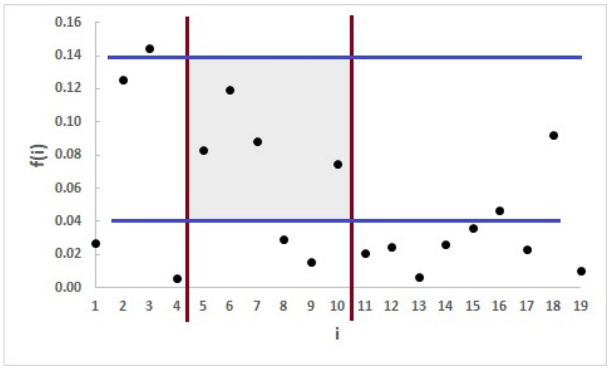


Figure 26: Type 1 filter

Figure 26 shows 4 f(i)s in the block i=5, …,10 and within the range of probabilities from 0.04 to 0.14. So

- With limits on the number of f(i)s of 1-3, this distribution would be rejected ;
- With limits of 3-5, it would be accepted
- With limits of 4-4, it would be accepted.

Example: To select those distributions for which f(4),f(5) and f(6) are all less than 0.1

i. Choose the block i=4,5,6
ii. Choose the range 0.0-0.1
iii. Choose the limits 3 to 3 (ie. require precisely 3 -that is, all of them)

Example: To select those distributions for which none of f(4),f(5) and f(6) is between 0.05 and 0.10

i. Choose the block i=4,5,6
ii. Choose the range 0.05-0.10
iii. Choose the limits 0 to 0

It is possible to specify an impossible filter, which no distribution could pass. For example, with a distribution of degree 5, requiring each f(i) to be greater than 0.25 would be impossible. Such an impossible filter would throw the program into a never-ending loop since it would be continually discarding one distribution after another in its hunt for an acceptable distribution. Even if a filter were possible, meeting its requirements might be so difficult that excessive run times would be encountered. Either situation would be easily identifiable once the program has started running; the program could then be stopped manually.

15.2 Type 2 filtering

You specify three things:

(i) A block of i, called the object block, which you want to be considered by the filtering process. For example, you might want to look at i= 3,4,5,6
(ii) Another block of i, called the comparison block, to form the basis of a comparison, for example i=14,15,16,17,18
(iii) Upper and lower limits on how many of the f(i)s within the object block are to be greater than (or equal to) the largest f(i) within the comparison block). In Figure 27, for example, there are 3.
  - The two limits may be the same
  - 0 may be used as a limit

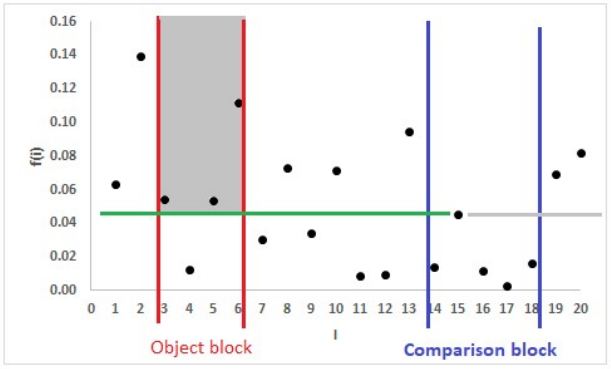


Figure 27 Type 2 filter

- The Object and Comparison blocks do not have to be distinct. They may overlap or even be the same.
- Interchanging the roles of the two blocks can give solutions to additional requirements. For example, in Figure 27 it is not possible to specify that all of the f(i)s in {3,4,5,6} should be less than all those in {14,…, 18}, but swap the roles of Object and Comparison blocks and it becomes possible.

16 Frequency distributions and CDFs
The expected distribution of a probability can at times be useful. This will normally be concentrated in a fairly small region around the likeliness, so it would be inefficient to look at the whole of [0,1]; instead, we use a smaller interval to cover the range of interest, and then partition that interval into 20 cells and calculate how often the probability could be expected to fall into each cell.

The difficulty is that such a range of interest cannot be determined until the frequency distribution has been constructed, but the frequency distribution cannot be constructed until the range of interest has been chosen.

To get round this, the program carries out a quick first-pass through the iterations during which it collects enough information to enable it to make a reasonable estimate of the interval: which it then stores in details.txt. The intention is not to `get the interval right', but, rather, to be able to present you with enough information to make a better choice to suit your own needs, modifying details.txt accordingly. The program actually has very little to go on, so sometimes it does get things very wrong: but this is usually easily corrected by you.

When running a problem for the first time (i.e. from the keyboard), you will not see any mention of the distributions of the probabilities: the program will just work away automatically producing its ranges of interest, which it then stores in details.txt for you to modify on later runs if you wish.

17 details.txt
You may edit details.txt with any simple text editor: use of a wordprocessor is not recommended since it might insert invisible codes.

You would edit details.txt if, for example, you wanted to change the values of some data without having to retype the whole of the problem-specification.

There are two types of item in details.txt: structural and non-structural. Structural items affect the layout of the remainder of details.txt; non-structural items do not. You are strongly advised not to edit structural items because of the knock-on effects for the rest of the file (which are usually not as easy to predict as might be thought).

Each item is preceded by a brief description. Descriptions of non-structural items are in CAPITALS and are enclosed in square brackets [~].

To help you find your way around, items in a block of similar items are usually preceded by an indication of where you are, eg `h(11)'. These are not descriptions so the lower case and the round brackets should not be taken as indicating a structural item: the description is at the start of the block.

If you make a syntactical mistake whilst typing the details of a new problem then the program can, and will, ask you to re-enter the information. If you make a syntactical mistake when editing details.txt, however, then the program cannot ask you to re-enter the information, because the program will not be running. The first you will know of the mistake is when you subsequently try running the program and a run-time error occurs; details of this will be sent to the screen and to the file errlog.txt. Behaviour is similar to that of a compiler: the error might not be picked up immediately and the reported form of the error might not be the actual form.

The contents of details.txt are in a standard layout chosen to make subsequent editing as easy as possible, and so are not simply a repetition of your typing. The basic idea is that it is easier -and less error-prone- to alter an existing value than it is to insert an omitted value, so everything is specifically given and nothing is implied. Examples are:-

When specifying an histogram, you give a block about which you wish to be asked, and the program defaults values outside that block to zero. Regardless of which block you specify, the block stored in data.txt always runs from 1 to N and the defaulted zero values are all specifically given.

Regardless of whether or not you say that you want to use a contraction, the program always gives you one, albeit one which has no effect because it has a magnitude of 1. So the answer to the question `Do you want to use a contraction?' is always stored as `1' for `YES', followed by the centre and magnitude of a contraction. If you say that you do want a contraction then the stored details will be as specified by you. If you say that you do not want a contraction then the magnitude will be 1 (and you don't have to worry about where the centre is, because the contraction will have no effect).

When specifying the centre of a contraction, you are given the choice between various standard cases and specifying all the co-ordinates yourself. Regardless of how you reply, details.txt contains the answer 'specify them myself', followed by all the co-ordinates.

The two things you will most often want to alter are (a) the number of iterations and (b) the subintervals used for the calculation of distributions. For convenience, these have been placed together, and are preceded by a line of asterisks across the screen, terminating with the words FREQUENT CHANGES HERE.

18 results.csv

18.1 Table 1: Input Data

This Table summarises the data as input by you.

If you are not using any Merge blocks, then each will be shown as `not used'. For any Merge block that you are using, the column `size' will show the number of observations you have specified for that block, and the extent of the block will be shown by the fields that have been `asterisked out'.

If you are not using a contraction then the row `contraction' will show a contraction of size 1 and centre $"1"$. If you are using a contraction, then its size and centre will be as specified by you.

18.2 Table 2: Random selection of 25 distributions

For convenience, either for your own interest or for use when writing a presentation, this section shows a random selection of 25 distributions. Also shown is Pr(g|f).

18.3 Table 3: Convergence of Likelinesses

This table shows convergence of the calculated likelinesses as the iterations proceed. Plotted points are concentrated towards the beginning and end of the curve.

The very beginning of the iterative process is not shown since values there are highly erratic and would throw out your spreadsheet's automatic scaling of the vertical axis. This does mean that the calculations will usually get to be very close to the final value before the tabulation has even started. Consequently, the automatic scaling will make variations seem more extreme than they really are: pay particular attention to the vertical scale; the whole of Figure 28, for example, is contained within an interval of approximately 0.001.

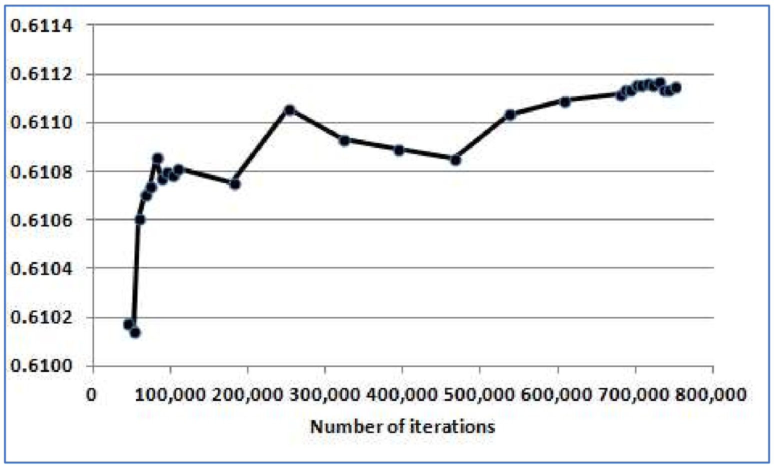


Figure 28: Convergence of Likelinesses

18.4 Table 4: Likelinesses

This will be the main table of interest: it gives the likelinesses of the standard integrams (the required integram, g, and each of the integrams "I".

It also shows the Multinomial Consistency, which is an indication of how well the likelinesses obey the Multinomial Theorem. This will usually be just for the purposes of reporting, since the program itself makes use of the Multinomial Theorem unnecessary.

18.5 Tables 5&6: Frequency and Cumulative distributions

The program partitions each range of interest (see Section 0) into 20 subintervals and finds the likeliness that Pr(g|f) or Pr(i|f), as appropriate, is in each of those subintervals. The results are output as Table 5, which consists of a number of small tables, one for each of the standard integrams. The centres of the cells have been included to make graph-plotting easier.

The cumulative sums of the likelinesses in each of the sub-tables of Table 5 are then formed, to give the best-estimated CDF for Pr(g|f) and for each Pr(i|f). These are output as Table 6.

19 The Sampling files

19.1 Introduction

Every time the program is run, it forms six files which are called the *sampling files*. These come in two sets of three: three in CSV format and three in TXT format. Their contents are given in Table 8

Table 8: Contents of sampling files

File	Contents
sampling_dis.csv, sampling_dis.txt	Sample of distributions\\
sampling_obs.csv, sampling_obs.txt	Observations simulated by using the distributions in sampling_dis as generating distributions.
sampling_rfs.csv, sampling_rfs.txt	Relative frequencies for the observations in sampling_obs

In the CSV files, the data is divided into groups and group-level data is produced. In the TXT files, there is no grouping.

The three in CSV format are intended to be used in a spreadsheet; the three in TXT format are intended to be read as data by some other program, possibly a future version of this one.

How many distributions there are, how many simulated observations there are per distribution and how many distributions there are per group, are controlled by altering the appropriate values in defaults.txt. The distributions used to construct the sampling files are totally distinct from those used to find likelinesses, so your choices here will not affect the main analyses.

Great Likelinesses does not place an upper limit on the number of distributions which may be included in the sampling files. However, the user's software might: an older spreadsheet, for example, might have a maximum of about 65,000 rows.

Many users will not have any need for the sampling files, but for those who do they can be a valuable part of the program.

19.2 CSV files

Each CSV file is split into groups, the size of which is set in defaults.txt.

At the top of the file, there is an initial header area, giving basic information about how the groups have been set up.

- Specifying a group size greater than the number of distributions means that the program will never reach the end of a group and so not get round to producing group-level data: this is equivalent to turning grouping off.
- A group size of 0 is defaulted to the number of distributions and so prints group data at, and only at, the very end of the files.

Think of each row as representing the results of a test, where each test consists of a number of observations. Each group represents the results of an experiment, where each experiment consists of a number of tests. The whole file consists of a number of repetitions of an experiment, and shows the variation which might be encountered.

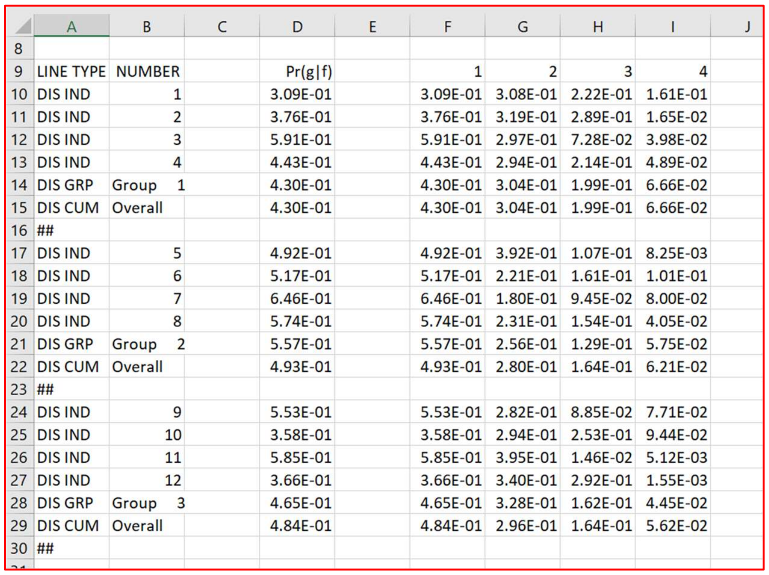
Each line starts with a status field, which says what that line is all about. By sorting on this, a file can be split into its various components of individual results, group sums/averages and grand total.

The first three characters of the status field correspond to the name of the file (DIS,OBS,RFS) and say what the line contains (distributions, observations, relative frequencies). The next three characters (IND,GRP,CUM) give the level of the data (individual, group, cumulative-so-far).

Details of Group and Cumulative data are given in Table 9.

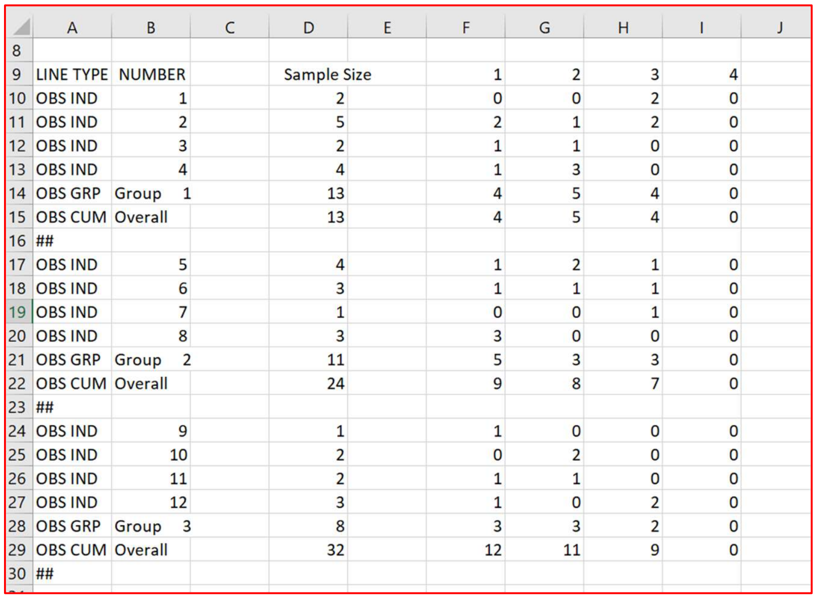
Table 9: Group & Cumulative data in sampling files

File	Group data	Cumulative data
sampling_dis.csv (Figure 29)	Mean of the f(i)'s for that group	Mean of all the f(i)'s to date
sampling_obs.csv (Figure 30)	Total observations for that group	Total of all observations to date
sampling_rfs.csv (Figure 31)	RFs for the group	RFs for all observations to date



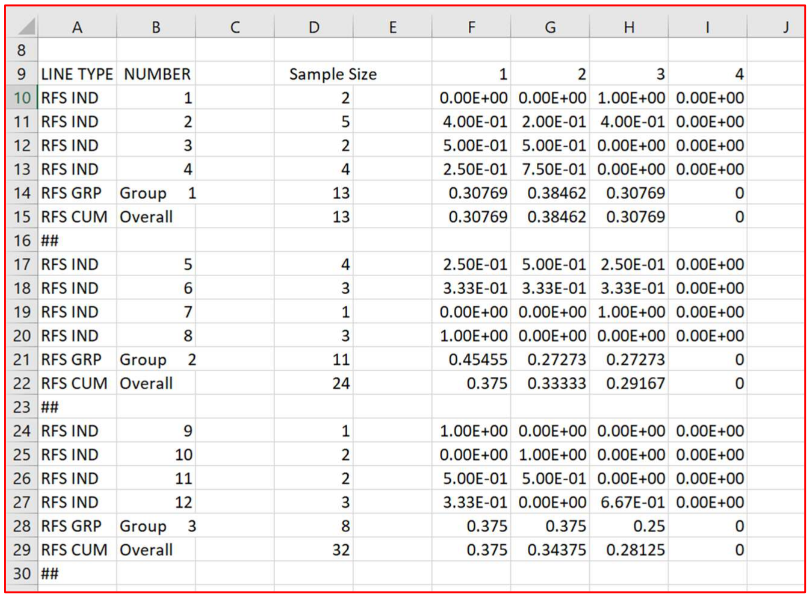
	A	B	C	D	E	F	G	H	I	J
8										
9	LINE TYPE	NUMBER		Pr(g|f)		1	2	3	4	
10	DIS IND	1		3.09E-01		3.09E-01	3.08E-01	2.22E-01	1.61E-01	
11	DIS IND	2		3.76E-01		3.76E-01	3.19E-01	2.89E-01	1.65E-02	
12	DIS IND	3		5.91E-01		5.91E-01	2.97E-01	7.28E-02	3.98E-02	
13	DIS IND	4		4.43E-01		4.43E-01	2.94E-01	2.14E-01	4.89E-02	
14	DIS GRP	Group 1		4.30E-01		4.30E-01	3.04E-01	1.99E-01	6.66E-02	
15	DIS CUM	Overall		4.30E-01		4.30E-01	3.04E-01	1.99E-01	6.66E-02	
16	##									
17	DIS IND	5		4.92E-01		4.92E-01	3.92E-01	1.07E-01	8.25E-03	
18	DIS IND	6		5.17E-01		5.17E-01	2.21E-01	1.61E-01	1.01E-01	
19	DIS IND	7		6.46E-01		6.46E-01	1.80E-01	9.45E-02	8.00E-02	
20	DIS IND	8		5.74E-01		5.74E-01	2.31E-01	1.54E-01	4.05E-02	
21	DIS GRP	Group 2		5.57E-01		5.57E-01	2.56E-01	1.29E-01	5.75E-02	
22	DIS CUM	Overall		4.93E-01		4.93E-01	2.80E-01	1.64E-01	6.21E-02	
23	##									
24	DIS IND	9		5.53E-01		5.53E-01	2.82E-01	8.85E-02	7.71E-02	
25	DIS IND	10		3.58E-01		3.58E-01	2.94E-01	2.53E-01	9.44E-02	
26	DIS IND	11		5.85E-01		5.85E-01	3.95E-01	1.46E-02	5.12E-03	
27	DIS IND	12		3.66E-01		3.66E-01	3.40E-01	2.92E-01	1.55E-03	
28	DIS GRP	Group 3		4.65E-01		4.65E-01	3.28E-01	1.62E-01	4.45E-02	
29	DIS CUM	Overall		4.84E-01		4.84E-01	2.96E-01	1.64E-01	5.62E-02	
30	##									

Figure 29: Example sampling_dis.csv



	A	B	C	D	E	F	G	H	I	J
8										
9	LINE TYPE	NUMBER		Sample Size		1	2	3	4	
10	OBS IND	1		2		0	0	2	0	
11	OBS IND	2		5		2	1	2	0	
12	OBS IND	3		2		1	1	0	0	
13	OBS IND	4		4		1	3	0	0	
14	OBS GRP	Group 1		13		4	5	4	0	
15	OBS CUM	Overall		13		4	5	4	0	
16	##									
17	OBS IND	5		4		1	2	1	0	
18	OBS IND	6		3		1	1	1	0	
19	OBS IND	7		1		0	0	1	0	
20	OBS IND	8		3		3	0	0	0	
21	OBS GRP	Group 2		11		5	3	3	0	
22	OBS CUM	Overall		24		9	8	7	0	
23	##									
24	OBS IND	9		1		1	0	0	0	
25	OBS IND	10		2		0	2	0	0	
26	OBS IND	11		2		1	1	0	0	
27	OBS IND	12		3		1	0	2	0	
28	OBS GRP	Group 3		8		3	3	2	0	
29	OBS CUM	Overall		32		12	11	9	0	
30	##									

Figure 30: Example sampling_obs.csv

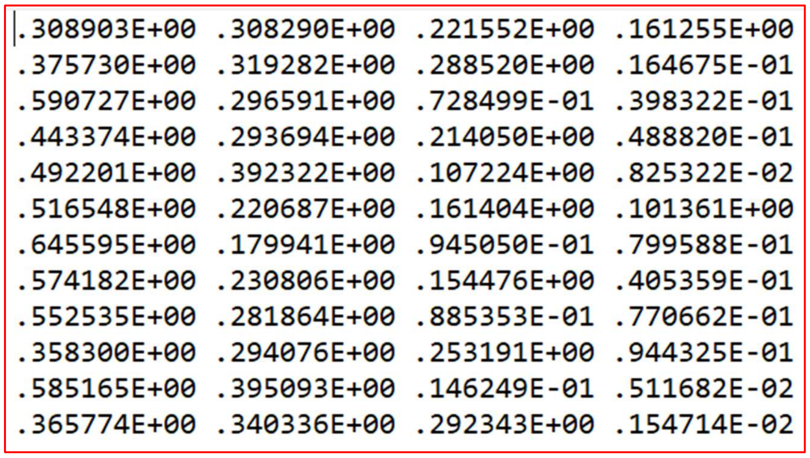


	A	B	C	D	E	F	G	H	I	J
8										
9	LINE TYPE	NUMBER		Sample Size		1	2	3	4	
10	RFS IND	1		2		0.00E+00	0.00E+00	1.00E+00	0.00E+00	
11	RFS IND	2		5		4.00E-01	2.00E-01	4.00E-01	0.00E+00	
12	RFS IND	3		2		5.00E-01	5.00E-01	0.00E+00	0.00E+00	
13	RFS IND	4		4		2.50E-01	7.50E-01	0.00E+00	0.00E+00	
14	RFS GRP	Group 1		13		0.30769	0.38462	0.30769	0	
15	RFS CUM	Overall		13		0.30769	0.38462	0.30769	0	
16	##									
17	RFS IND	5		4		2.50E-01	5.00E-01	2.50E-01	0.00E+00	
18	RFS IND	6		3		3.33E-01	3.33E-01	3.33E-01	0.00E+00	
19	RFS IND	7		1		0.00E+00	0.00E+00	1.00E+00	0.00E+00	
20	RFS IND	8		3		1.00E+00	0.00E+00	0.00E+00	0.00E+00	
21	RFS GRP	Group 2		11		0.45455	0.27273	0.27273	0	
22	RFS CUM	Overall		24		0.375	0.33333	0.29167	0	
23	##									
24	RFS IND	9		1		1.00E+00	0.00E+00	0.00E+00	0.00E+00	
25	RFS IND	10		2		0.00E+00	1.00E+00	0.00E+00	0.00E+00	
26	RFS IND	11		2		5.00E-01	5.00E-01	0.00E+00	0.00E+00	
27	RFS IND	12		3		3.33E-01	0.00E+00	6.67E-01	0.00E+00	
28	RFS GRP	Group 3		8		0.375	0.375	0.25	0	
29	RFS CUM	Overall		32		0.375	0.34375	0.28125	0	
30	##									

Figure 31: Example sampling_rfs.csv

19.3 TXT files

The TXT files are so simple (Figure 32) that they do not include even column headings.



```
.308903E+00 .308290E+00 .221552E+00 .161255E+00
.375730E+00 .319282E+00 .288520E+00 .164675E-01
.590727E+00 .296591E+00 .728499E-01 .398322E-01
.443374E+00 .293694E+00 .214050E+00 .488820E-01
.492201E+00 .392322E+00 .107224E+00 .825322E-02
.516548E+00 .220687E+00 .161404E+00 .101361E+00
.645595E+00 .179941E+00 .945050E-01 .799588E-01
.574182E+00 .230806E+00 .154476E+00 .405359E-01
.552535E+00 .281864E+00 .885353E-01 .770662E-01
.358300E+00 .294076E+00 .253191E+00 .944325E-01
.585165E+00 .395093E+00 .146249E-01 .511682E-02
.365774E+00 .340336E+00 .292343E+00 .154714E-02
```

Figure 32: Example sampling_dis.txt

In the TXT files, observations are in I8 format, distributions and RFs are in E11.6, all with one space as a separator.

20 Defaults.txt

20.1 Introduction

defaults.txt is intended to contain the answers to questions which most people either would not be interested in, or would not usually want to alter. These questions could rapidly become annoying if asked every time the program was run.

Because defaults.txt is much simpler than data.txt, error-reporting is minimal: either something works or it produces a run-time error. If the latter then the cause of the problem is easily spotted: usually either a text string has not been enclosed in quotes or a non-integer numerical value has been used.

Some errors -usually involving nonsensical integer values- are easily detectable as errors; the program will use standard values if one of these is detected.

Some items could (at least in theory) take unlimited integer values. For these, there is no `nonsensical integer value' which could be specified as part of the program. The user, however, could voluntarily place limits on these and so -to give protection against gross typing etc errors- is given the ability to give maximum acceptable values.

Each item also has a Factory Setting, which is hardwired into the program. All items can be reset to their factory settings by selecting item 999 from the opening menu.

20.2 NUMBER OF ITERATIONS TO BE USED

Each iteration corresponds to one distribution selected at random from the underlying set. Specify the number here.

If you rarely have an interest in anything apart from basic likelinesses then it should be possible to reduce the default number of iterations to substantially fewer than the Factory Setting: 100,000 or fewer will often be good enough. However, if the program runs fast enough for you then you should ask yourself why you are risking your precision by reducing the number.

On the other hand, if you are usually interested in PDFs then you might find that an increase to substantially more than the Factory Setting would be convenient.

If the number of iterations is set as ≤0 then the program will not carry out any iterations, but will still produce the sampling files. In addition, if input is from the keyboard then you will not be asked anything about the given data, defaults being written to details.txt; if input is from the file then it will be as normal.

In practice, your choice for the number of iterations will be either 0 (if you are interested only in producing the sampling files) and 750,000 (if you want to carry out an analysis): some people might sometimes want to specify more than that.

Factory Setting: 750000

20.3 SEEDING THE RANDOM NUMBER GENERATOR

There are two options:-

1. The program chooses the same seed every time it is run. This is always the same, but you have no control over its value.
2. The program chooses a different seed every time it is run.

If defaults.txt contains any integer other than 1 then the program uses option 2.

Factory Setting: 2

20.4 WHETHER PDFs AND CDFs ARE WANTED

Whether pdfs and cdfs of the various probabilities are wanted

1: Yes, they are wanted
2: No, they are not wanted

Other values are replaced by 2.

The calculation of pdfs and cdfs can consume a significant proportion of the program's efforts and so should be switched off unless actually needed.

Factory Setting: 2

20.5 WHETHER data is to be read from DATA.TXT

Whether data is to be read from data.txt

1: Yes, read from data.txt
2: No, do not read from data.txt

Factory Setting: 2

20.6 THE MAXIMUM NUMBER OF DISTRIBUTIONS WHICH MAY BE SPECIFIED IN THE NEXT ITEM

The next item asks for the number of distributions to be used in the sampling files. There is no natural upper limit to this, which makes that item particularly vulnerable to gross typing errors.

To give some protection, the user may specify an upper limit to the number of distributions which may be specified in the next item. If a number larger than that upper limit is entered, it will be reset to that upper limit.

If the maximum entered here is 0 or negative then no upper limit is imposed.

Factory Setting: 10000

20.7 THE NUMBER OF DISTRIBUTIONS WANTED IN THE SAMPLING FILES

Each time the program is run, a number of distributions meeting the problem-definition is sent to the sampling files. This item specifies how many there should be.

If 0 or fewer distributions is chosen then the program sends 100 distributions to the sampling files.

Factory Setting: 100

20.8 BROKEN INTO GROUPS OF

The CSV sampling files are broken into groups by the insertion of group-level data and a blank line after every n'th distribution. Insert the size of the groups (ie. the value of n) here.

A group size of 0 or less forces a single group consisting of all the distributions. A group size greater than the number of distributions switches grouping off.

Factory Setting: 0

20.9 HOW FREQUENTLY A NEW DISTRIBUTION IS TO BE CHOSEN

There are three options:-

1. Make every distribution a new distribution,
2. Choose a new distribution only at the start of each group -so that all the distributions within a group are the same.
3. Choose a new distribution only at the start of the sampling files -so that all the distributions throughout the sampling files are the same.

The analysis of experiments often assumes that there is only one generating distribution. To model this, set all the distributions within each group to be the same (Option 2). Then each group models the results from a single experiment, and the different groups model the possible variation in experimental results.

Factory Setting: 1

20.10 THE MAXIMUM NUMBER OF OBSERVATIONS GENERATED PER DISTRIBUTION WHICH MAY BE SPECIFIED IN THE NEXT ITEM

The next item asks for the number of observations to be generated per distribution in the sampling files. There is no natural upper limit to this, but the user may voluntarily impose one here.

If the code used in the next item is 0 or -1 then no upper limit is imposed.

If the code used is negative (other than -1) then the negative of the same limit is applied.

Factory Setting: 50

20.11 CODE GIVING THE NUMBER OF OBSERVATIONS GENERATED PER DISTRIBUTION

Each distribution sent to sampling_dis.csv is used as a generating distribution to simulate at least one observation. You specify the actual number of observations here by giving an integer, n, which has the effect given by Table 10.

Table 10: Choices for the number of observations generated per distribution

n	Number of observations simulated per distribution
1,2,…	n
0	ω(g)
-1	Chosen at random from 1,…,ω(g)
-2,-3,…	Chosen at random from 1,…,-n

If n is negative, the random choice of the number of observations is made every time a distribution is sent to file (ie. it is not a `once-and-for-all' decision).

Factory Setting: 0

20.12 WHETHER THE WEIGHTS IN A MULTIMODAL DISTRIBUTION ARE TO BE USED AS PROBABILITIES-OF-SELECTION RATHER THAN COEFFICIENTS

There are two options:-

1. Option 1: YES. Use them as probabilities-of-selection.
2. Option 2: NO. Use them as the coefficients in a linear combination.

By selecting Option 1, you will be forcing the weights to be used as probabilities-of-selection ***if*** the conditions are appropriate.

Factory Setting: 1

21 Odds and Ends

(An unordered list of things to remember and things which do not easily fit in elsewhere.)

- If you have been looking at any file but have forgotten to close it down before running the program again then you will receive a run-time error or be thrown back into Windows. Close the file and -if your system offers you the choice- choose **R**etry. If your system does not offer you this choice then you may need to restart your computer.

- For basic problems [no merging, contractions or Relative Frequencies; no given data; required integram ="1"; 750,000 iterations.], run times using a 64-bit laptop were as given in Table 11

Table 11: Relative run-times for simple cases

Underlying set	**Degree, N**						
	2	**5**	**10**	**25**	**50**	**75**	**100**
B(N)	1	2	3	9	23	42	68
S(N)	2	3	5	13	32	60	115
R(N)	2	3	5	13	33	61	117
M(N)	2	3	7	18	47	89	172
U(N)	2	3	7	18	49	90	174

- To investigate the effects of the sample size of the given data, take advantage of the fact that input relative frequencies are normalised before use, so they do not actually have to be relative frequencies provided they are not negative. Do not give any data as the input histogram but give it, instead, as input relative frequencies; varying the sample size then does just that.
- In defaults.txt, if you choose to have a single group, by eg. selecting a group size of 0, and also choose to have a new distribution only at the start of a group then every entry in the sampling files will use the same distribution. However, that distribution will be selected at random and you will not have any say in its choice. To have just a single distribution, \textbf{\textit{specified by you}}, throughout the whole of the sampling files, when running the program, specify a contraction of magnitude zero, centred on the required distribution.
- Don't be worried by the fact that the times allowed for in the countdown information are measured in days. At several points in the coding, it was necessary to take a view about how many iterations the program was being designed to cater for; it was decided to base the design on needing 1 billion. For that many iterations, the most long-winded analysis I could find took two days. But that was under extreme circumstances using a much slower computer than even the slowest modern laptop. You will not come anywhere near needing that: a typical use will rarely need more than 3 or 4 minutes