ANALOG: A program for estimating paleoclimate parameters using the method of modern analogs

Peter N. Schweitzer
U.S. Geological Survey
Reston, VA

December 9, 1994

U.S. Geological Survey Open-File Report 94-645

This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards (or with the North American Stratigraphic Code). Any use of trade, product, or firm names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

Contents

1 Background

Beginning in the 1970s with CLIMAP, paleoclimatologists have been trying to derive quantitative estimates of climatic parameters from the sedimentary record. In general the procedure is to observe the modern distribution of some component of surface sediment that depends on climate, find an empirical relationship between climate and the character of sediments, then extrapolate past climate by studying older sediments in the same way.

Initially the empirical relationship between climate and components of the sediment was determined using a multiple regression technique (Imbrie and Kipp, 1971). In these studies sea-floor sediments were examined to determine the percentage of various species of planktonic foraminifera present in them. Supposing that the distribution of foraminiferal assemblages depended strongly on the extremes of annual sea-surface temperature (SST), the foraminiferal assemblages (refined through use of varimax factor analysis) were regressed against the average SST during the coolest and warmest months of the year. The result was a set of transfer functions, equations that could be used to estimate cool and warm SST from the faunal composition of a sediment sample. Assuming that the ecological preference of the species had remained constant throughout the last several hundred thousand years, these transfer functions could be used to estimate SSTs during much of the late Pleistocene.

Hutson (1980) and Overpeck, Webb, and Prentice (1985) proposed an alternative approach to estimating paleoclimatic parameters. Their "method of modern analogs" revolved not around the existence of a few climatically-sensitive faunal assemblages but rather on the expectation that similar climatic regimes should foster similar faunal and floral assemblages. From a large pool of modern samples, those few are selected whose faunal compositions are most similar to a given fossil sample. Paleoclimate estimates are derived using the climatic character of only the most similar modern samples, the modern analogs of the fossil sample.

This report describes how to use the program ANALOG to carry out the method of modern analogs. It is assumed that the user has faunal census estimates of one or more fossil samples, and one or more sets of faunal data from modern samples. Furthermore, the user must understand the taxonomic categories represented in the data sets, and be able to recognize taxa that are or may be considered equivalent in the analysis.

ANALOG provides the user with flexibility in input data format, output data content, and choice of distance measure, and allows the user to determine which taxa from each modern and fossil data file are compared. Most of the memory required by the program is allocated dynamically, so that, on systems that permit program segments to grow, the program consumes only as many system resources as are needed to accomplish its task.

2 System requirements

- MS-DOS systems:
  - 80386 or compatible 32-bit CPU
  - 2 or more megabytes of extended memory
- Macintosh systems:
  - Macintosh System 7.0 or later
  - 2 or more megabytes of memory beyond what the system uses
- UNIX systems:
  - ANSI-conformant C language compiler

3 Operation

ANALOG is controlled by a user-supplied *run description* file whose name is given as the argument on the command line. The run description file, which is composed of ASCII text, indicates which data files are used as the basis of modern samples from which analogs will be selected, which data files describe the fossil samples whose paleoclimate parameters are to be estimated, the distance measure used to decide which modern samples are analogs of the fossil samples, and information about how to report the results.

3.1 Run description file

The run description file must be plain ASCII text (not a word-processor document), which can be generated using a simple text editor. If a word processor is used, select "Save As" and find the option that will ensure the output file is ASCII ("Text Only" or "DOS Text" are likely to give the correct results).

Blank lines in the run description file are ignored. Comments may be included by inserting the pound sign '#' anywhere on a line; the pound sign and all chararcters to the end of the line are considered part of the comment and are ignored by the program. Lines may begin with any number of spaces or tab characters to enhance the file's readability. Names and keywords can be in upper or lower case or a mixture, but under UNIX, file names must be given with the correct letter case, or the program won't find the files. File names should not contain braces ('{' or '}') since these indicate the grouping of items in the run description file.

The run description file consists of a sequence of keywords followed in some cases by file names or descriptive words. Items may be listed in any order, but the grouping described by braces below must be maintained. This keeps related information together. An example is shown in figure 1.

The following words are keywords, and cannot be used as the names of files:

```
basis
sample
data
transform
meta
distance
report
name
closest
verbose
```

To carry out a complete analysis, the run description file must contain at least one `basis` specification indicating the modern data base and one `sample` specification indicating the fossil samples. All other statements in the run description file are optional.

3.1.1 `verbose`

The word `verbose`, if used, causes ANALOG to print copious amounts of information to the standard output device, which is usually the display screen but may instead be redirected to a disk file. All of the input data are printed while the program is running when this option is selected. This is a good way to ensure that the program has read the input file correctly.

3.1.2 `basis` and `sample`

The input files are either *basis* files or *sample* files. Basis files are the modern samples for which environmental data are available. Sample files are the fossil samples whose environmental data are to be estimated.

The general form of the `basis` specification is

> `basis` {
> `data` *file name*: *format*
> `transform` *file name*
> `meta` *file name*: *format*
> }

The `sample` statement is identical but the keyword `sample` is used in place of `basis`.

To include a data set in the analysis, give its group (basis or sample) followed by an opening brace, and specify the following information:

1. The keyword `data`, the name of the file containing the data, and (optionally) a description of the file's format. The colon, if present, must be followed by a space or tab, and the names must not contain spaces or tabs.
2. The keyword `transform` and the name of the file containing the transformation rules (explained below).
3. The keyword `meta`, the name, and format of the file containing meta data for the samples, using the same form as the `data` statement. At present, meta data is reported only for basis samples, so you don't need to specify a meta data file for fossil samples.

Note that "name" here refers to any appropriate file specification; on UNIX or MS-DOS systems it can be either a fully-qualified directory and file name or simply the name of a file in the default directory. On Macintosh systems, the name given should be unique on the system, because the program will search all available volumes to locate it, and will use the first file found to have that name.

The group is delimited using braces so the program will know which transformation file and which meta data file go with each basis file. Only the data file specification is actually required; the other statements are optional but *will* typically be needed.

Example: To read the CLIMAP modern core-top data base as a basis, include the following text in the run description file:

```
basis {
    data climap.raw: tab
    transform climap.rule
    meta climap.meta: tab
    }
```

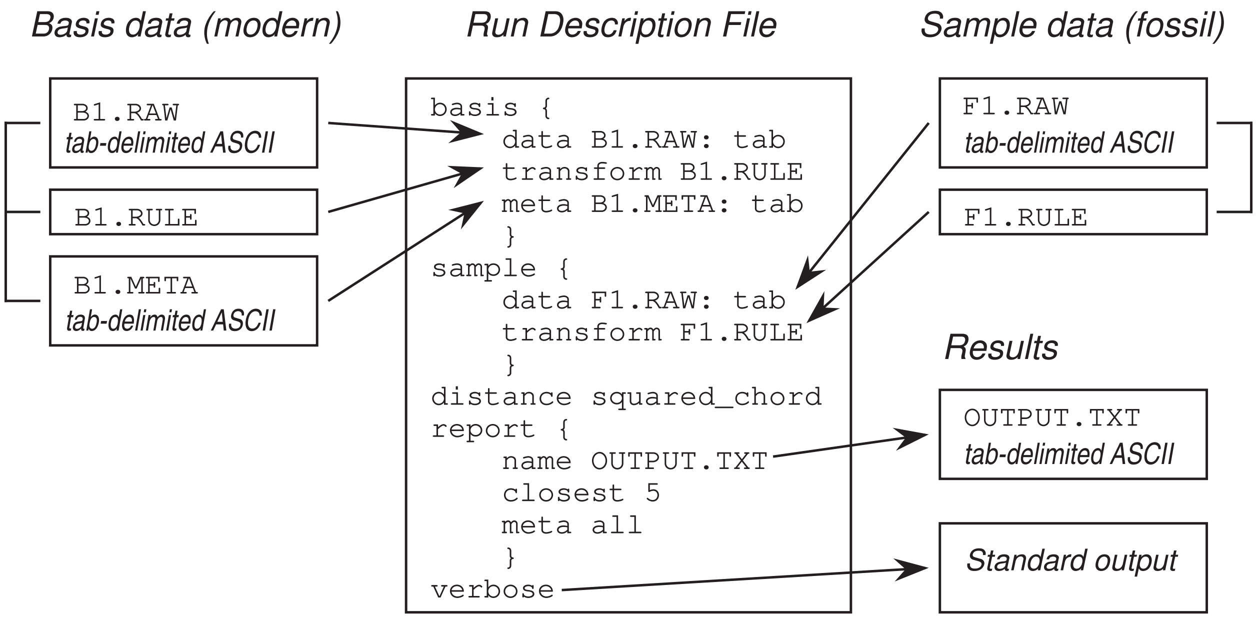

Figure 1: Information indicated in the run description file. The basis data files are shown as boxes on the left. The sample data files are shown on the right. Braces connect the raw, rule, and meta data files in the figure, but these are typically separate physical files on the disk. Results are output to the file `OUTPUT.TXT`, and the input data are echoed to the standard output device because `verbose` is specified in the run description file.

Example: To read the file `mycore.dat` as a fossil sample, include the following text in the run description file:

```
sample {
    data mycore.dat: tab
    transform mycore.rule
    }
```

3.1.3 distance

In addition to specifying the input basis and sample files, the run description file may specify a distance measure to be used to calculate the similarity between modern and fossil samples. The format is

`distance` *distance measure name*

where *distance measure name* is one of the following (the underscores must be included as shown):

```
manhattan
euclidean
squared_euclidean
canberra
squared_chord
squared_chisquared
dot_product
correlation
jaccard
```

These are the distance metrics available. The sample data values are contained in the arrays $\{s_0, s_1, \ldots, s_{n-1}\}$ and $\{t_0, t_1, \ldots, t_{n-1}\}$; the result is stored in d.

1. **Manhattan**

$$d = \sum_{i=0}^{n-1} |s_i - t_i|$$

2. **Squared Euclidean**

$$d = \sum_{i=0}^{n-1} (s_i - t_i)^2$$

3. **Euclidean**

$$d = \sqrt{\sum_{i=0}^{n-1} (s_i - t_i)^2}$$

4. **Canberra**

$$d = \sum_{i=0}^{n-1} \frac{(s_i - t_i)}{(s_i + t_i)}$$

5. **Squared Chord**
$$d = \sum_{i=0}^{n-1} (\sqrt{s_i} - \sqrt{t_i})^2$$

6. **Squared Chi-squared**
$$d = \sum_{i=0}^{n-1} \frac{(s_i - t_i)^2}{(s_i + t_i)}$$

7. **Jaccard**
$$d = 1.0 - \frac{a}{a + c}$$
where a is the number of taxa that are present in both samples and c is the number of taxa that are present in one sample and absent in the other.

8. **Dot product**
$$d = \sum_{i=0}^{n-1} s_i t_i$$

9. **Correlation**
$$d = \frac{\sum_{i=0}^{n-1} (s_i - \overline{s})(t_i - \overline{t})}{\sqrt{\sum_{i=0}^{n-1} (s_i - \overline{s})^2} \sqrt{\sum_{i=0}^{n-1} (t_i - \overline{t})^2}}$$

3.1.4 `report`

The `report` statement specifies what will be output. In this section you can specify how many of the closest basis samples to report for each fossil sample, and which meta data variables from the modern samples to include in the description of the matches. The general form of the `report` statement is

`report` {
  `name` *output file name*: *format*
  `closest` *number*
  `meta` *variable_name_1*
  `meta` *variable_name_2*
  ...
  `meta` *variable_name_n*
  }

`name` is optional (if omitted, results are directed to the standard output device); its syntax is the same as the `data` and `meta` statements in input file specifications.

`closest` should be followed by an integer; if `closest` is not indicated, the value 1 is used.

`meta` should be followed by the name of a meta data variable, in which case the statement can be repeated, or by `all` if you want all of the meta data variables added to the output for each matching modern sample. If the meta data variable's name contains any spaces, you must replace them with underscores in this statement.

Example: The following `report` statement causes ANALOG to create a file called `analog.out` for the results, to list the ten modern samples most similar to each fossil sample, and to report for each such modern sample the meta data variables present in the CLIMAP core-top data base.

```
report {
    name analog.out: tab
    closest 10
    meta Latitude_degrees
    meta Latitude_minutes
    meta Longitude_degrees
    meta Longitude_minutes
    meta SST_cool
    meta SST_warm
    }
```

3.2 Data transformation file

In general, different data sets will contain different faunal and floral taxa, both because different taxa are prevalent in different regions and because data set providers use varying taxonomic categories, names, and abbreviations. The method of modern analogs requires that corresponding variables in different data sets be recognizable as such, otherwise it would be impossible to calculate the distance measures. The data transformation file provides a way to ensure that ANALOG recognizes corresponding variables as such. In addition, it allows the user to specify certain arithmetic operations to be performed on entire samples. An example showing the use of transformation files is depicted in figure 2.

Since the transformation file allows you to rearrange the taxa found in one file so that they are commensurate with those in another, there will generally be one transformation file for every raw data file in the analysis.

The transformation file is interpreted much like the run description file; blank lines are ignored, comments are indicated with the pound sign, spaces or tabs may precede keywords and taxon names, and keywords are not case sensitive. Certain words are recognized as keywords, and should not be used as taxon identifiers:

```
new
add
subtract
multiply
```

Raw data — *Transform file* — *Comparable data*

Data file A

ID	A	B	C	D
A1	1	2	3	4
A2	5	5	0	0
A3	4	1	2	3

Rules for A

```
ignore A
ignore B
new E {
    add A
    add B
    }
divide by sum
```

ID	C	D	E
A1	0.3	0.4	0.3
A2	0.0	0.0	1.0
A3	0.2	0.3	0.5

Data file B

ID	C	Da	Db	E
B1	3	3	1	5
B2	1	5	0	4
B3	5	1	1	3

Rules for B

```
ignore Da
ignore Db
new D {
    add Da
    add Db
    }
divide by sum
```

ID	C	D	E
B1	0.25	0.33	0.42
B2	0.1	0.5	0.4
B3	0.5	0.2	0.3

Figure 2: Use of transformation files to make data from two different input files commensurate. Data file A has the variables A,B,C, and D, which are typically the names of biological taxa. Data file B has variables C, Da, Db, and E. In order to calculate distance coefficients properly, the variables must be the same in each data set. The user determines that taxa C, D, and E are present in both data sets, and notes that taxon E is composed of both types A and B, while taxon D is composed of types Da and Db. For data file A, the counts of taxa A and B are combined to calculate the counts of taxon E. A and B themselves are then ignored since these taxa are not tabulated separately in data file B. Likewise, data file A does not break taxon D into the two parts Da and Db, while data file B does. So the rules for data file B specify the creation of taxon D from Da and Db, and Da and Db are subsequently ignored. Finally, both data sets are transformed (after the addition and deletion of variables) by conversion to proportions (`divide by sum`).

```
divide
ignore
all
by
```

The names of variables taken from the data file must be used here exactly as they appear in the data file, including punctuation and letter case, *except* that spaces embedded in the names must be replaced by underscores. This is required because the code that parses the transformation file uses spaces to separate words. By replacing spaces with underscores, the different words of the taxon identifier are kept together. The parser then converts underscores back into spaces in order to compare names from the transformation file with names in the raw data file. This also means that underscores should not appear in the names of raw data variables.

Three directives can be given in transformation files: `ignore`, `new`, and arithmetic operations.

3.2.1 `ignore`

`ignore` causes ANALOG to disregard the named variable. Its syntax is `ignore` *variable_name*.

Examples:

```
ignore Depth_(mbsf)
ignore Indeterminates
```

3.2.2 `new`

To create a new variable from one or more raw variables found in the input file or to change the name of a raw variable, use the following syntax:

`new` *new_name* {
  `add` *old_name_1*
  `add` *old_name_2*
  ...
  `add` *old_name_n*
  }

The new variable exists only while ANALOG is running. It cannot be retained because the program is not intended as a data set editing tool and is not programmed to save raw data on disk.

Examples:

```
new dupac {
    add N._dutertrei
    add Neogloboquadrina_pachyderma
    }
new Tsuga {
```

```
add Tsuga_canadensis? # note included '?'
add Tsuga_heterophylla # and '_' replaces ' '
add Tsuga_mertensiana
}
```

Note that including a taxon in a new category doesn't automatically cause that taxon to be ignored. It may be useful to explicitly `ignore` raw taxa that were used to create new categories.

It is not necessary to specify variables in any particular order. ANALOG will sort them alphabetically and will only compare taxa whose names match *exactly*. Even spaces before and after the names are considered when comparing names.

3.2.3 arithmetic operations

ANALOG interprets the `ignore` and `new` instructions and copies the raw data values (including those computed from `new` instructions) into a separate array. At that point it can carry out arithmetic operations on this new data array. The operations permitted are indicated using these keywords:

```
multiply
divide
add
subtract
```

The operation word is followed by a decimal number or one of the words

```
count
sum
ssq
mean
var
sdev
```

`count` refers to the number of valid (i.e. not missing) data values in the sample. `sum` refers to the sum of the data values for the sample. `ssq` refers to the sum of squares of the data values for the sample. `mean` refers to the arithmetic average of the data values for the sample. `var` refers to the variance of the data values for the sample. `sdev` refers to the standard deviation of the data values for the sample. The variance and standard deviation here are n-weighted, meaning that the variance is calculated as the sum of squared deviations from the mean divided by the number of values n, not $n-1$. The standard deviation is calculated as the square root of the variance.

`multiply` and `divide` may be followed by the word `by`; the `by` will be ignored. The number of arithmetic operations is not limited.

Example:

To obtain percentages from counts, use the statements

```
divide by sum
multiply by 100
```

In practice, it is not necessary to multiply by 100, as proportions are more appropriate for calculating distance measures than are percentages.

4 Input data formats

Three input file formats are currently supported: tab-delimited ASCII, the ASCII format of the North American Pollen Database, and the strict format of the CLIMAP Atlantic Ocean data set as described in the first SPECMAP data release. Other input file formats may be added, but that requires amending and recompiling the source code. The default input file format is tab-delimited ASCII; to read an input file in any other format, the format must be indicated in the run description file.

4.1 Tab-delimited ASCII

Tab-delimited ASCII indicates that the data are stored in a plain text file, one sample per line, with variable values separated by the tab character (ASCII 8). This format is typically generated when a spreadsheet program is asked to save data as "Text Only". ANALOG requires also that

- the first value on a line (i.e. the string consisting of all characters up to the first tab) contain the sample identifier, and
- the first line contain column headings, which are interpreted as taxon or meta data variable names.

Example (the symbol <*tab*> indicates the tab character):

```
Sample ID<tab>G. ruber<tab>G. sacculifer<tab>N. pachyderma
KNR110/43PC 0-2cm<tab>100<tab>50<tab>0
P1AR92B8 0-2cm<tab>0<tab>0<tab>200
```

4.2 NAPD ASCII format

The North American Pollen Database (NAPD) uses, by default, binary file formats that are proprietary to the PC-based relational data base management system PARADOX by Borland, International. In addition to PARADOX files, the data of NAPD are available in ASCII form, following some conventions. Two types of ASCII files are provided, one for modern data and a slightly different format for fossil data. ANALOG reads raw data from both varieties of these ASCII files, but can interpret meta data only in the modern file format (files with the suffix `.m50`).

4.3 SPECMAP format

The SPECMAP file format is described in file 70 of the SPECMAP data set, which can be obtained by anonymous ftp from `ftp.ngdc.noaa.gov` in the directory `paleo/specmap/specmap1`.

5 Output data formats

At present, only tab-delimited output is implemented, though other output file formats can be added. The current output format includes, for each of the closest analogs, the following information, in order:

1. Fossil sample identifier
2. Rank of the modern analog
3. Sample identifier of the modern analog
4. Distance between fossil sample and the modern analog
5. Meta data variables from the modern analog

Note that the number of lines output for each fossil sample is dictated by the `closest` statement in the `report` group given in the run description file.

6 Error messages and warnings

Listed below are the diagnostic messages generated by ANALOG, with a brief explanation of the likely cause of the problem.

`Usage: ANALOG run_description_file_name`
You neglected to enter the name of the run description file on the command line.

`Error: could not open ...`
The name given is misspelled or the file is unavailable. If the file is the run description, which controls subsequent execution, the program does not continue. If the file is a raw or meta data file, the program tries to continue without that file's data.

`Error: unexpected end of run description file`
The run description file is incomplete or there is an opening brace that has no corresponding closing brace.

`Error: expected` *(string)*`, got` *(string)* ...
The run description or transformation file contains a syntax error. This message appears when one of the following rules is broken:

- File name must follow `data` keyword.
- Format description must follow colon after file name if colon is present.

- File name must follow `transform` keyword.
- File name must follow `meta` keyword in `basis` or `sample` groups.
- File name must follow `name` keyword in `report` group.
- Meta variable name or keyword `all` must follow `meta` keyword in `report` group.
- Opening brace must follow `basis`, `sample`, or `report` keywords.
- Taxon name must follow `ignore`, `new`, or `add` in transformation file.
- Opening brace must follow taxon name in `new` statement of transformation file.
- Name following `distance` keyword must be a recognized distance measure.
- Number must follow `closest` keyword in `report` group.
- Number or keyword `sum` must follow arithmetic operator in transformation file; keyword `by` may appear between them.

`Error: unexpected` *(string)* `...`
Extraneous text appears where a keyword or closing brace should be. Check to see if you misspelled a keyword.

`Warning: extra data at end of line ...`
`Warning: extra meta data at end of line ...`
ANALOG counts the number of variables in a tab-delimited file by looking at the column headings. If there are fewer column headings than there are columns with data in them, the program notices the extra characters at the end of the line and informs the user that they exist. It then proceeds, ignoring the extra characters.

`Warning: found meta data but no numerical data for sample` *(name)* `...`
`Warning: no meta data found for sample` *(name)* `...`
`Warning: found temperature data but no numerical data for sample` *(name)* `...`
A sample ID for meta data must match exactly the ID for the same sample in the raw data file. If they do not, the program assumes that the two records refer to different samples. If the program encounters meta data for a sample for which it has no raw data or, after having read meta data, finds samples for which no meta data could be found, these warnings are issued.

`Error: card out of order ...`
`Error: data card` *(number)* `refers to wrong core ...`
The SPECMAP data is quite specific about the order of lines referring to an individual core. Similarly, the core identifier must be the same on the data cards as on the master card.

`Error: expected NAPD ASCII header line, got "`*(string)*`" ...`
`Error: expected two numbers, got "`*(string)*`" ...`
`Warning: expected sample meta data, got "`*(string)*`" ...`

NAPD ASCII format files must abide by specific guidelines for the arrangment of data. If these guidelines are not followed, ANALOG will be unable to continue. Meta data can be read only from the modern files (those whose extension is `.m50`). An attempt to read meta data from a fossil NAPD data file (extension `.f70`) will generate the warning indicating that the meta data could not be read.

`Error: could not enlarge ...`
`Error: could not allocate ...`

An attempt to allocate memory dynamically has failed. This indicates that there is not enough memory to read, interpret, or analyze the data. On some systems it may be possible to increase the amount of memory that is available to the program.

`Error: new variable` *(name)* `already exists in data file` *(name)*
`Error: You asked to create AND ignore new variable` *(name)* `...`
`Warning: new variable` *(name)* `refers to old variable` *(name)*`, which does not occur ...`
`Warning: rule file` *(name)* `says to ignore "`*(name)*`", which isn't in` *(name)*

These messages indicate discrepancies between the rule file and the data file. Where possible, ANALOG will continue, issuing a warning. Where it cannot proceed without additional information from the user, ANALOG issues an error message and stops.

`Warning: data from` *(name)* `contains variable "`*(name)*`" not found in` *(name)*

If, after transformation, the sample data contains a taxon that is not present in the basis data, or *vice versa*, that taxon is ignored when computing the distance measure. A warning is issued because the condition may give misleading results even though the distance measures can still be calculated.

`Warning: could not create output file` *(name)*`; using stdout instead.`

The named file could not be created. Instead, results will be directed to the standard output device.

7 Programming notes

ANALOG was developed on a Data General AViiON 6220 server running DG/UX 5.4R2.01, a version of UNIX. The program is written using Standard C. It has also been successfully compiled and run under MS-DOS using GNU C 2.5.7 and the GO32 DOS-extender, and with Watcom C/386 9.5b and DOS4GW 1.95, the DOS-extender from Rational Systems Inc, and on the Macintosh using THINK C v. 6.0. It is possible to compile the program so that it will run under Macintosh System 6.x; replace the reference to `Mac_fopen` with the standard C `fopen` in the file `input.c`, and do not include the input file `macfiles.c` in the project. Running under System 6, the data files must reside in the same folder as the program.

ANALOG was developed on a Data General AViiON 6220 server running DG/UX 5.4R2.01, a version of UNIX. The program is written using Standard C. It has also been successfully compiled and run under MS-DOS using GNU C 2.5.7 and the GO32 DOS-extender, and with Watcom C/386

9.5b and DOS4GW 1.95, the DOS-extender from Rational Systems Inc, and on the Macintosh using THINK C v. 6.0.

It is possible to compile the Macintosh version so that it will run under Macintosh System 6. As distributed, System 7 is required because the program makes a call to PBCatSearch, a toolbox routine that locates a file by name on a Macintosh volume. If you replace the reference to `Mac_fopen` with the standard C `fopen` in the file `input.c`, and do not include the input file `macfiles.c` in the project, the program will run under System 6, but the data files must then reside in the same folder as the program.

An ancillary program called `transpos` is available for MS-DOS and UNIX systems. This program transposes the rows and columns of a tab- delimited text file. A typical use might be to get a list of column headings from a tab-delimited text file. Cut the first line from the tab- delimited text file, then run the program `transpos`, redirecting output to a disk file. The output will contain a list of column headings, one per line. `transpos` and its documentation can be obtained using the URL `http://geochange.er.usgs.gov/pub/tools/transpose/README.html`.

7.1 Additional file formats

Additional input file formats may be implemented. The code to read the new format must be supplied by the user, compiled, and linked with the rest of the source code. To enable ANALOG to recognize the new file format, it is necessary to add an array element to the static array `read_function` in file `input.c`. The first member of the array element is a character string that will be used to identify the format; this string will be given as the format in the run description file. The second member is a pointer to the function that reads raw data from this type of file. The third member is a pointer to the function that reads meta data from this type of file.

The actions taken by the new read functions may be complex. Briefly, the read function is given a pointer to a data base structure, whose members include the information specified in the run description file. If `p` is the pointer passed to the read function, then raw data are read from `p->raw.filespec`; the variable names are stored in `p->raw.name_buffer`; pointers to the names are stored in `p->raw.name`, and the number of raw variables is stored in `p->raw.count`. Raw data for each sample are stored in the arrays `p->sample[n].raw`. The meta data must be read likewise from `p->meta.filespec`, the number of variables in `p->meta.count`, the names in `p->meta.name_buffer`, and pointers to the names in `p->meta.name`. Meta data values are stored as strings in the array `p->sample[n].meta_buffer`, and pointers to those strings are stored in `p->sample[n].meta`. See the code in `input.c` for details.

Adding another output format is essentially the same, but is less complex, because the output data are simpler in form. See the code in `results.c` for clues.

8 References

- Hutson, W.H., 1980, The Agulhas current during the Late Pleistocene: Analysis of modern faunal analogs: *Science*, v. 207, p. 64-66.
- Imbrie, J., and Kipp, N., 1971, A new micropaleontological method for quantitative paleoclimatology: Application to a late Pleistocene Caribbean core: *in* Turekian, K.K. (ed.), *The Late Cenozoic Glacial Ages*, Yale University Press, pp. 71-181.
- Overpeck, J.T., Webb, T. III, and Prentice, I.C., 1985, Quantitative interpretation of fossil pollen spectra: Dissimilarity coefficients and the method of modern analogs for pollen data: *Quaternary Research*, v.23, p. 87-108.
- North American Pollen Database, Illinois State Museum, Springfield, Illinois, USA, and National Geophysical Data Center, Boulder, Colorado, USA.

9 Technical contact

The technical contact is the author of this relase of ANALOG and its documentation:

Peter N. Schweitzer
Mail Stop 955, National Center
U.S. Geological Survey
Reston, VA 22092

Tel: (703) 648-6533
FAX: (703) 648-6647
email: `pschweit"er@usgs.gov`

ANALOG may be obtained electronically through anonymous `ftp` from `geochange.er.usgs.gov` in the directory `pub/tools/analog`.

10 Appendix 1: Annotated list of included digital files

10.1 Source code

`Makefile` *Make* description file for UNIX.

`analog.h` Data structures and constants used in the program.

`analog.c` Main program source code.

`distance.c` Distance measure calculation.

`error.c` Error and warning output. Modify this file if you want more sophisticated handling of error and warning messages.

`input.c` Interpretation of the input data. Modify this file to read files in formats that are not already supported.

`macfiles.c` Macintosh code to locate input files by name anywhere on the system.

`parse.c` Interpretation of the run description file.

`results.c` Output of results for each sample.

`rules.c` Interpretation of the data transformation files.

10.2 Executable code

`ANALOG.EXE` MS-DOS executable code, requires `DOS4GW.EXE`

`DOS4GW.EXE` Rational Systems, Inc. DOS-extender. This enables the 32-bit MS-DOS program `ANALOG.EXE` to execute.

`analog.hqx` Macintosh executable file, encoded using *BinHex 4.0*.

10.3 Data bases

10.3.1 CLIMAP

This data base has been included with ANALOG both because I expect it to be widely used and because in its original format it cannot properly be read by ANALOG. I have modified it so that it could be read by ANALOG, changing it to tab-delimited ASCII text format. These are the digital files which pertain to the CLIMAP core top data base:

`world.fauna` The core top data base received from Brown University, corrected.

`changes.fauna` Changes made to `world.fauna` to fix typos in core identifiers.

`world.sst` The meta data file received from Brown University, corrected.

`changes.sst` Changes made to `world.sst` to fix typos in core identifiers.

`cc_meta.c` Program to convert `world.sst` to be readable by ANALOG as tab-delimited ASCII.

`cc_raw.c` Program to convert `world.fauna` to be readable by ANALOG as tab-delimited ASCII.

`climap.meta` CLIMAP core top data base meta data file in tab-delimited ASCII format. Includes core locations and SST estimates taken from published atlases.

`climap.raw` CLIMAP core top data base raw data file in tab-delimited ASCII format.

`climap.rule` List of taxa from `climap.raw`; modify this file to create an appropriate transformation file for `climap.raw`.

`run_description` Example run description file for reading in the CLIMAP core top data base. As distributed, this file lacks a `sample` statement, so no analysis is actually carried out. But it can be modified to analyze a set of fossil samples by adding one or more `sample` statements.

10.3.2 NAPD

`modern.m50` The modern samples of the North American Pollen Database.

`taxa.m50` A list of taxon names from `modern.m50`; modify this list to create an appropriate transformation file.

`run_description` A sample run description containing commands to read `modern.m50`.

10.3.3 SPECMAP

`specmap.070` Text file describing the format of `specmap.071` and `specmap.072`.

`specmap.071` Modern Atlantic core-top data base.

`specmap.072` Fossil samples. These lack meta data.

`test_specmap` A sample run description containing commands to read `specmap.071` and `specmap.072`.

10.4 Documentation

`analog.tex` Source code for this document, in LaTeX.

`analog.ps` PostScript version of this document.

`analog.html` Hypertext version of this document, in HTML.

`io.gif` Figure 1, in Graphic Interchange Format (GIF).

`io.pict` Figure 1, in Macintosh PICT format.

`io.eps` Figure 1, in encapsulated PostScript.

`rule.gif` Figure 2, in Graphic Interchange Format (GIF).

`rule.pict` Figure 2, in Macintosh PICT format.

`rule.eps` Figure 2, in encapsulated PostScript.