Interactive identification using the Internet

7 April 2007

M.J. Dallwitz, T.A. Paine and E.J. Zurcher

This is an updated and expanded version of a paper that was presented at a symposium held during the XXI International Congress of Entomology in Iguassu Falls, Brazil, on 24 August 2000 (Dallwitz, Paine and Zurcher 2002).

Abstract

Computer-based interactive keys have several advantages over conventional keys: characters can be used, and their values changed, in any order; a correct identification can be made in spite of errors by the user or in the data; errors which were circumvented by the error-tolerance mechanism can be located; the user can express uncertainty by entering more than one state value, or a range of numerical values; numeric characters can be used directly, without being divided into ranges. Other features important for efficient and reliable identification include: advice on the most suitable characters to use at any stage of an identification; notes on the interpretation of characters; illustrations of characters and taxa; finding the differences and similarities between taxa; finding diagnostic descriptions. Interactive identification can be made available over the Internet in the following ways.

1. A stand-alone program.
   Example: Intkey (http://delta-intkey.com)
2. A program (Java or JavaScript) running in a Web browser.
   Example: NaviKey (http://www.navikey.net)
3. Cooperating programs running in a Web browser and server.
   Example: ActKey (http://flora.huh.harvard.edu:8080/actkey/index.jsp)
4. A program running on a Web server, and generating HTML pages.
   Example: PollyClave (http://prod.library.utoronto.ca:8090/polyclave/index.html)

Programs of type 1 must first be installed, and most are available for only one operating system (usually MS-Windows). Programs of types 1 and 2 download the data matrix at the start of a session. The user cannot proceed until the downloading is completed, but afterwards response is fast, and there is no further load on the network and server, except when subsidiary files, such as images, are required. The programs can also be used off line. In programs of types 3 and 4, the data matrix is not downloaded. Each operation requires an Internet transaction, so responses tend to be slow, and a continuing load is placed on the network and server. The programs cannot be used off line. In programs of type 4, the user interface is familiar to Web users, but may become cumbersome for some operations, particularly with large data sets. Programs of types 2–4 are potentially independent of the user's operating system and browser, but in practice there may be problems.

Introduction

Identification is the process of finding the taxon to which a specimen belongs. Several methods are available for aiding this process (e.g. Pankhurst 1991). The most important are conventional identification keys and interactive keys.

A conventional identification key is a tree with characters at the internal nodes and taxon names at the terminal nodes. Each branch corresponds to a state of the character or characters at the node from which it arises. The user starts at the root of the tree, and follows the branches corresponding to the character states exhibited by the specimen until the taxon name is reached.

Authors of conventional keys try to provide some flexibility for the user by placing alternative characters at each node, but the possibilities for doing this are limited, because the characters must have identical distributions of their states among the taxa remaining in contention at that node. An error by the user in assigning a character state to the specimen inevitably leads to a wrong identification, unless the author has allowed for the possibility of this error by placing the taxon name in the subtree corresponding to the wrongly assigned state, as well as in the subtrees corresponding to states actually exhibited by the taxon. The author's use of this mechanism must also be limited, because each possible error (taxon/character-state combination) treated in this way adds a terminal node to the tree. This increases the size of the printed key (proportional to the number of terminal nodes), and the average number of characters that must be used to obtain an identification (proportional to the logarithm of the number of terminal nodes).

After any identification, it is good practice to check its accuracy by comparing the specimen with a description or illustrations of the taxon, or with other specimens known to belong to the taxon. When a conventional key is being used, the only way to recover from a wrong identification due to an error by the user is to guess where the error was made, return to that node, and try following another branch. If the error is in the key itself (that is, an error was made by the author), recovery is not possible.

An interactive key is an interactive computer program in which the user enters attributes (character-state values) of the specimen. The program eliminates taxa whose attributes do not match those of the specimen. This process is continued until only one taxon remains. The taxon attributes are usually stored as a characters-by-taxa 'matrix'. It is also possible to store the attributes as 'rules', but this kind of program is generally less satisfactory (Dallwitz 1992).

Dallwitz, Paine and Zurcher (2000) give a comprehensive discussion of the principles of interactive keys. Dallwitz (1996) gives a list of available interactive-key programs, and contact information for them, and Dallwitz (2000) gives a detailed comparison of several of these programs.

We will use the program Intkey (Dallwitz, Paine and Zurcher 1993, 1995) to exemplify some of the features of interactive keys.

Advantages over conventional keys

A well designed interactive key has several advantages over a conventional key.

Unrestricted character use. Any characters can be used, in any order. Characters which are not available on the specimen, or whose interpretation is not clear to the user, can be avoided (provided that there is sufficient redundancy in the data).

Character deletion and changing. The values of any character can be changed at any stage of the identification, or any character deleted from the identification.

Error tolerance. A correct identification can be made in spite of errors by the user or in the data. Taxa are normally eliminated when they differ from the specimen in any way. If it is known or suspected that an error has been made, the program can be instructed to eliminate taxa only if they differ from the specimen in more than one attribute. It is immaterial where the error occurred, and whether it was made by the user or by the author of the data.

In Intkey, this function is controlled by the 'Tolerance' parameter, whose value may be 0 or any positive integer. Taxa are eliminated if they differ from the specimen in more attributes than the current value of 'Tolerance'. The parameter may be set to any permitted value at any time in the identification process,

but typically it would be incremented by 1 when an identification has been made and found to be incorrect. The identification process is then continued, exactly as before. If *all* the taxa are eliminated, the program can increment ‘Tolerance’ automatically. If a single taxon remains, the program has no way of knowing whether this is the correct identification, and it is up to the user to check the identification, and, if necessary, increment ‘Tolerance’ manually.

Locating errors. The program should be able to locate user and/or data errors that were circumvented by the error-tolerance mechanism. The identification of user errors helps to improve the user’s interpretation of characters. Data errors can be reported to the author for correction in later versions.

In Intkey, errors can be located by using the ‘Differences’ command to display the differences between the specimen and the remaining taxon.

Expressing uncertainty. The user can express uncertainty by entering more than one state value, or a range of numerical values. A user who is not sure which character-state value applies to the specimen may nevertheless sometimes be confident that some state values *do not* apply. Entering all the values that may conceivably apply to the specimen eliminates those taxa that never exhibit any of those values.

Numeric characters. Numeric characters can be used directly, without being divided into ranges. In conventional keys, numeric characters such as lengths must be divided into ranges before being incorporated in the key. This usually results in loss of information. In an interactive key, the actual range of values exhibited by each taxon can be recorded in the data, and the taxon eliminated if the specimen’s value does not fall within this range.

Easy updating. The key is maintained simply by making corrections and additions to the data matrix. Updating of conventional keys is relatively difficult. Even when the key is generated by computer from a data matrix, major changes to the matrix, particularly the addition of new characters and taxa, can have a large effect on the key structure, which has to be checked and possibly re-optimized.

Important features for interactive keys

Interactive keys require other features for efficient and reliable identification. A few of the most important are described here; see Dallwitz, Paine and Zurcher (2000) for a comprehensive list.

Advice on the most suitable characters to use at any stage of an identification. The program should be able to advise the user on the most suitable characters for use at any stage of an identification. Because of the very large number of paths that may be taken through an interactive key, the ranking of the characters should be calculated directly from the data matrix for the set of taxa actually remaining at each stage of the identification. It is unsatisfactory to pre-assign rankings for a relatively small number of cases, as, for example, in a rule-based expert system.

The character-ranking algorithm used in Intkey is the same as that used in the key-generation program, Key (Dallwitz 1974). Unlike most such algorithms, it has a theoretical basis and gives sensible results for characters with three or more states, and for numeric characters. The relative weight of the separating power and the 'reliability' of the character (a subjective measure, usually supplied by the author, of the character's accuracy and/or ease of use) can be controlled by both the author and the user.

Ranking of the characters may take a considerable time in large data sets, so it is important that the computation is as efficient as possible, and that the user does not have to wait for the ranking to be completed before choosing a character.

‘Best’ algorithms should be able to handle numeric characters, as these often have high separating power. For example, the data set ‘*Festuca* of North America’ (Aiken et al. 1996) has 29 numeric characters and 67 multistate characters. When Intkey ranks these characters by their separating power, the top 17 characters are numeric. A similar tendency is shown in ‘The Families of Flowering Plants’ (Watson and Dallwitz 1992), which has 39 numeric characters and 459 multistate characters (excluding ‘characters’ used to define the classification). When the characters are ranked by separating power, 4 of the top 5, and 14 of the top 30, are numeric.

The high separating power of numeric characters is surprising to most taxonomists, as numeric characters are generally not very useful in conventional keys. There are two reasons for this. (1) Conventional keys

must use multistate characters for numeric data, and this causes a loss of separating power. (2) Numeric characters often show a large amount of overlap between taxa; in conventional keys, this results in multiple occurrences of taxa, and an increase in the *printed* length of the key. Neither of these factors applies to interactive keys.

Notes on the interpretation of characters. Extensive text to aid interpretation of characters should be conveniently available.

Illustrations of characters. Illustrations to aid interpretation of characters should be conveniently available. State selection, and changing of the selections, should be possible from the illustration screens (that is, it should not be necessary to return to a text-based screen for these operations). There should be no restrictions on the number of illustrations for each character and/or character state.

Illustrations of taxa. Taxon illustrations are useful for confirming identifications. Display of these illustrations should be flexible: there should be no limits on the number of illustrations of a taxon, the illustrations should be selectable by subject (e.g. habit, habitat, flowers, fruits, distribution map), and it should be possible to display illustrations of different taxa simultaneously.

Finding the differences and similarities between taxa. The program should be able to find the differences between members of a set of taxa, in terms of a selected set of characters. There should be no restrictions on the size of the set of taxa.

Finding diagnostic descriptions. The program should be able to find diagnostic descriptions, which distinguish a given taxon from all the other taxa. These provide a quick way of confirming the identity of a specimen. The characters should be chosen from those that have not been used in the current identification, in order to provide an independent confirmation.

Intkey has a parameter, ‘DiagLevel’, which specifies the minimum number of characters for which the diagnostic description should differ from all the other taxa. Another parameter, ‘DiagType’, distinguishes between specimen-diagnostic and taxon-diagnostic descriptions. The latter are allowed to contain characters which may sometimes be inapplicable to specimens belonging to the taxon.

Interactive identification over the Internet

Interactive identification can be made available over the Internet in several ways, which differ in whether the processing is done on a Web server or the user’s machine, and in the method of loading and running the software on the user’s machine. Each method has advantages and disadvantages, particularly in the times taken for various operations. The times given below are for a 133MHz Pentium, with an Internet connection running at about 15Kbytes per second.

1. A stand-alone program

Programs of this type must be downloaded and installed before their first use. This process usually takes a few minutes, depending on the size of the program and the speed of the Internet connection. Most are available for only one operating system (usually MS-Windows). The programs download the data matrix at the start of a session. The user cannot proceed until the downloading is completed, but afterwards response is fast, and there is no further load on the network and server, except when subsidiary files, such as images, are required. The user interface can be compact and simple, and can utilize the full capabilities of the operating system. The programs can (potentially) be set up as ‘helper applications’, so that they can automatically run a specified data set by clicking on a link in a Web page. They can also be used off line.

Examples of this type of program are:
Intkey (http://delta-intkey.com)
Lucid (http://lucidcentral.org)

Intkey is free for non-commercial use and is available at the above URL by following the links ‘Programs and documentation > Intkey’. A complete, annotated example of an identification using Intkey is also available by following the links ‘Overview of the DELTA System > An Intkey example: identification’.

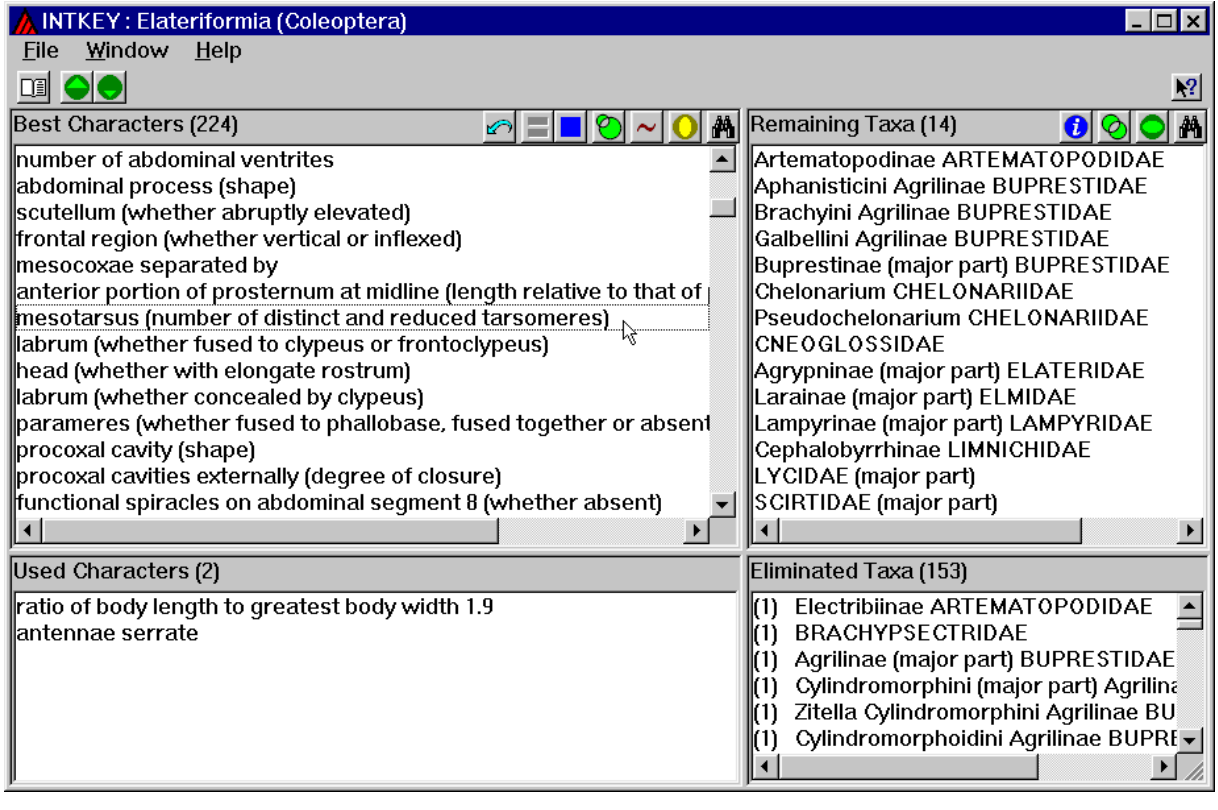
The Lucid Builder, which is required to construct keys, is commercial, but the keys themselves may be distributed without charge for non-commercial purposes. Lucid lacks many features important for efficient, accurate identification (see Dallwitz 2000).

The Intkey installation file is about 2.1MB in size. It takes about 150 seconds to download, and a further 60 seconds to install. The installed files occupy about 2.3MB (not counting the installation file, which can be deleted after installation), and are placed entirely in a separate directory — no files are added or overwritten in the Windows directories.

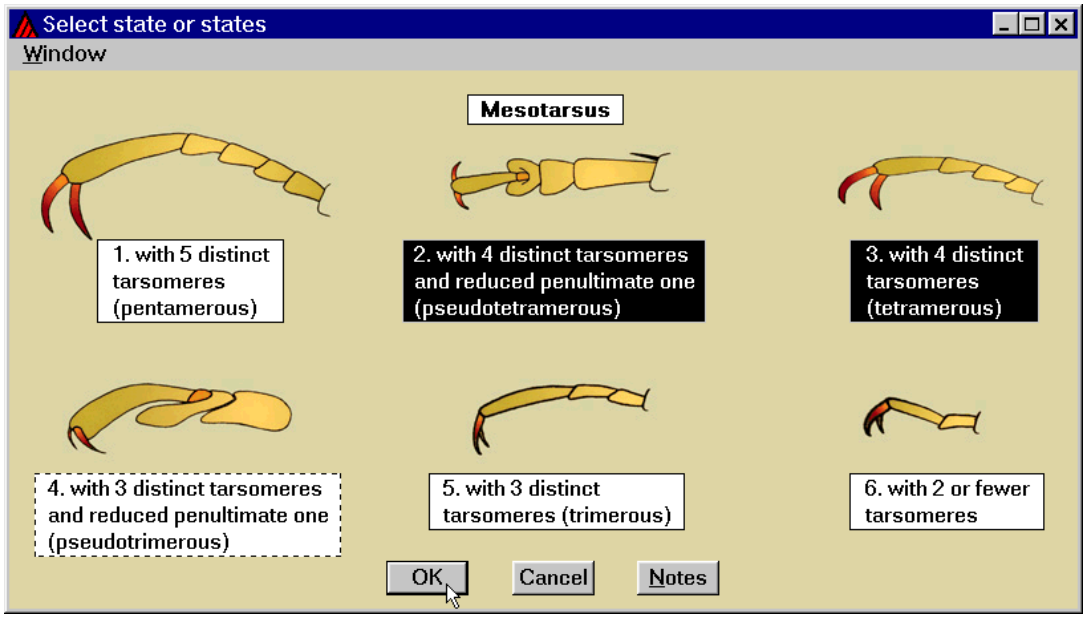
The Web site also has links to many Intkey data sets. One of these, 'The Families of Flowering Plants' (Watson and Dallwitz 1992), was used for timing tests. It contains 582 characters and 585 taxa. When running the data from the Internet, program startup and downloading of the data took 50 seconds. Thereafter the program works entirely locally, except for downloading images and descriptions when required. Simple operations such as using a character in an identification take less than 0.5 seconds. When calculating the 'Best' characters for an identification, the characters found so far are displayed after 2 seconds, and the rest after the calculation is complete. The characters are examined in descending order of character reliability, so a suitable character is almost always available within the first 2 seconds. About 320 characters were processed in this initial period.

The following sample screens were taken from 'Elateriformia of the World', also available at the above site.

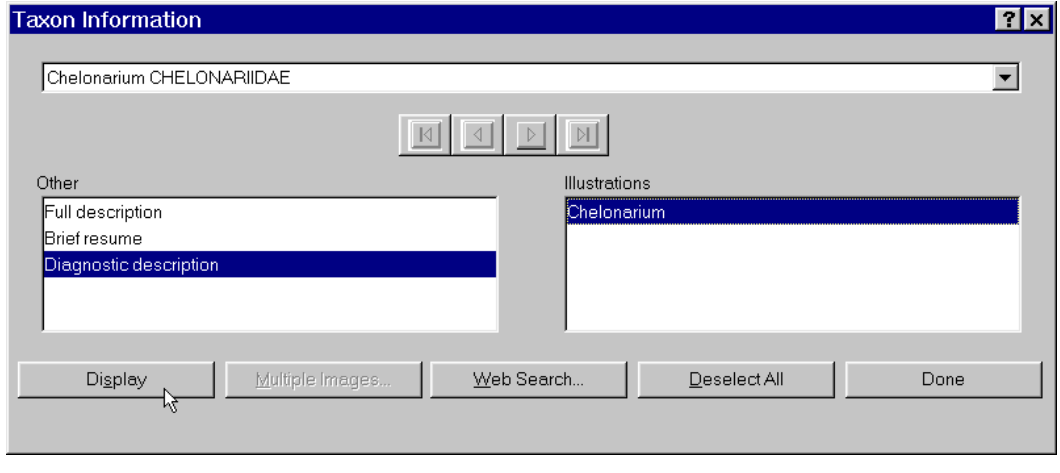
This is the main screen, part way through an identification. Two characters have been used, reducing the number of possible taxa from 167 to 14. The characters that can separate the remaining taxa have been automatically displayed in the 'Best Characters' pane, ranked as described above. One of these is about to be selected.



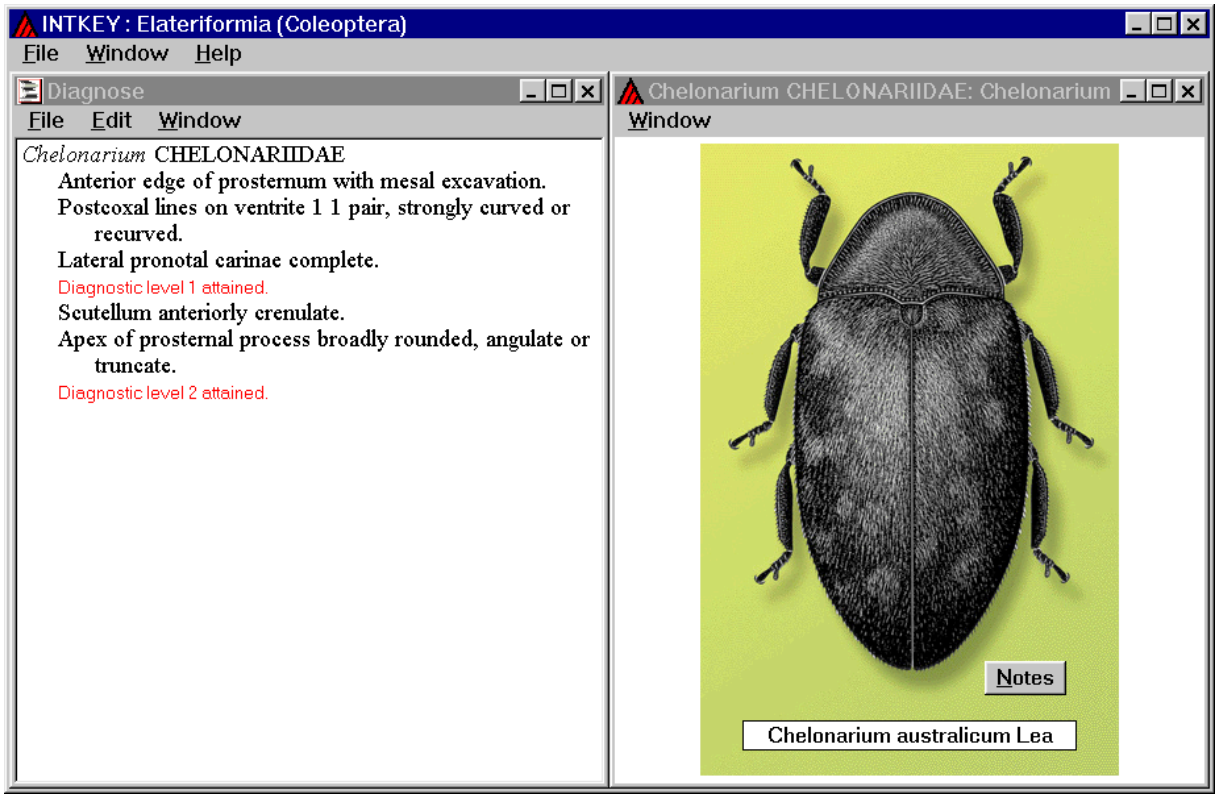
The following screen is then automatically displayed. Pressing the 'Notes' button would display notes on the interpretation of the character. States 2 and 3 have been selected, because it is difficult to distinguish between them in the specimen.



After using this character, a single taxon remains. Pressing the 'Information' button gives access to descriptions and illustrations, as shown in the following screen. In this example, a diagnostic description and the single image of the taxon have been selected to be displayed. Alternatively, the 'Web Search' button can be used to search for the selected taxon using a nominated general-purpose search engine (e.g. Google) or taxonomic database (e.g. ITIS).



The following screen shows the requested information. The diagnostic description contains only characters not used in the identification, and separates the taxon in at least 2 respects from every other taxon in the database.



2. A program (Java or JavaScript) running in a Web browser

Programs of this type do not have to be installed before use — they are downloaded and run automatically by the Web browser. However, it may be necessary to download and install supporting software, and this may be considerably larger than an identification program of type 1. For example, the installation file for the Sun Java 2 Runtime Environment, Version 1.42 for MS-Windows, is 14.6MB in size. Downloading and starting the identification program itself may take a significant time, depending on the size of the program, the speed of the Internet connection, whether the program is cached from a previous use, and the speed of the user's computer. Java and JavaScript programs should be independent of the user's operating system and browser, but in practice there may be compatibility problems. The programs download the data matrix at the start of a session, and the user cannot proceed until the downloading is completed. There is no further load on the network and server, except when subsidiary files, such as images, are required. Response times may be slow owing to inefficient computation in the browser. The user interface can be compact and simple, but design may be somewhat restricted by the limitations of the programming language and by compatibility considerations. The programs can also be used off line.

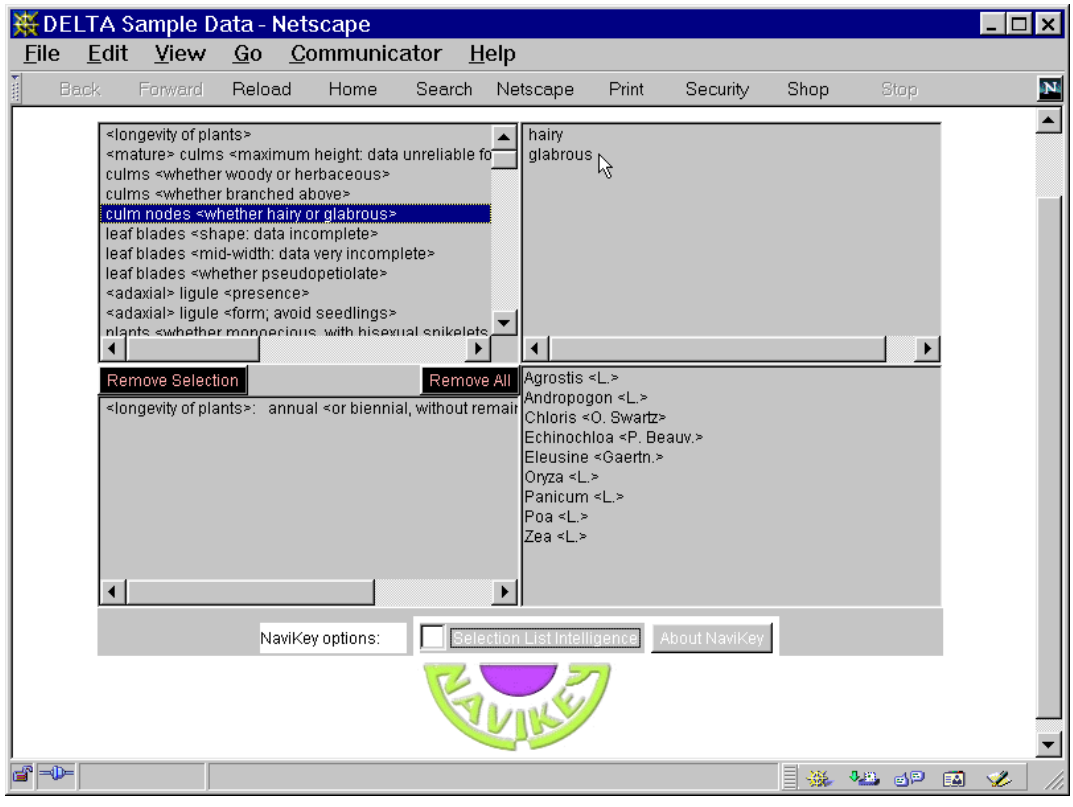
Examples of this type of program are:
Lucid Version 3 (http://lucidcentral.org)
NaviKey (http://www.navikey.net)

NaviKey is free for non-commercial use. It uses Java applets, and loads the data from DELTA files (Dallwitz 1980; Dallwitz, Paine and Zurcher 1993) when the program is started. Tests were carried out using Version 2.2 (July 1999).

Working from a local hard disk, loading the applets and the 'Flowering Plants' data (582 characters and 585 taxa) took 105 seconds. Working from the Internet, the same operations took 230 seconds. Other operations take the same time whether running locally or from the Internet. The program has a feature,

'selection list intelligence', which removes redundant characters from the list. With this feature off, using a character took 10 seconds; with it on, it took 65 seconds.

The following NaviKey screen uses the sample data supplied with the DELTA programs (Dallwitz, Paine and Zurcher 1993). An identification is in progress. The state 'annual' of the character 'longevity of plants' has been selected, leaving 9 taxa remaining from the original 14. The character 'culm nodes' has been selected, and its state 'glabrous' is about to be selected.



3. Cooperating programs running in a Web browser and server

Programs of this type do not have to be installed before use. The 'client' — the program running in the Web browser — is downloaded and run automatically by the browser. Downloading and starting the program may take a significant time, depending on the size of the program, the speed of the Internet connection, whether the program is cached from a previous use, and the speed of the user's computer. Java and JavaScript programs should be independent of the user's operating system and browser, but in practice there may be compatibility problems. The division of work between the server and client programs could be done in various ways, with the extremes approaching type 2 (data downloaded at the start, most of the work done by the client) and type 4 (no data downloaded, most of the work done by the server). The most useful division would probably be to:

- download the character descriptions and taxon names at the start (because these are typically displayed repeatedly during a session)
- carry out the data-matrix computations on the server (e.g. 'best' characters, taxa possessing a given attribute)
- use character, state, and taxon numbers to exchange information between the server and the client (e.g. the user's selections, and the results of the server's computations)

There is a continuing load on the network and server. The load on the network may be small compared with programs of type 4, because the information can be exchanged in a compact form. The load on the server may be comparatively large, because of the amount of computation required (e.g. for 'best' characters, differences, diagnostic descriptions). The response time is the time taken for a small Web transaction, plus the computation time on the server, plus the time taken for the client to interpret and display the results. The user interface can be compact and simple, but design may be somewhat restricted

by the limitations of the programming language and by compatibility considerations. The programs cannot be used off line.

Examples of this type of program are:
ActKey (http://flora.huh.harvard.edu:8080/actkey/index.jsp)
NaviKey (client-server version – Internet Archive:
http://www.herbaria.harvard.edu/computerlab/web_keys/navikey/)

Both are free for non-commercial use. No tests have been carried out on these programs.

4. A program running on a Web server, and generating HTML pages

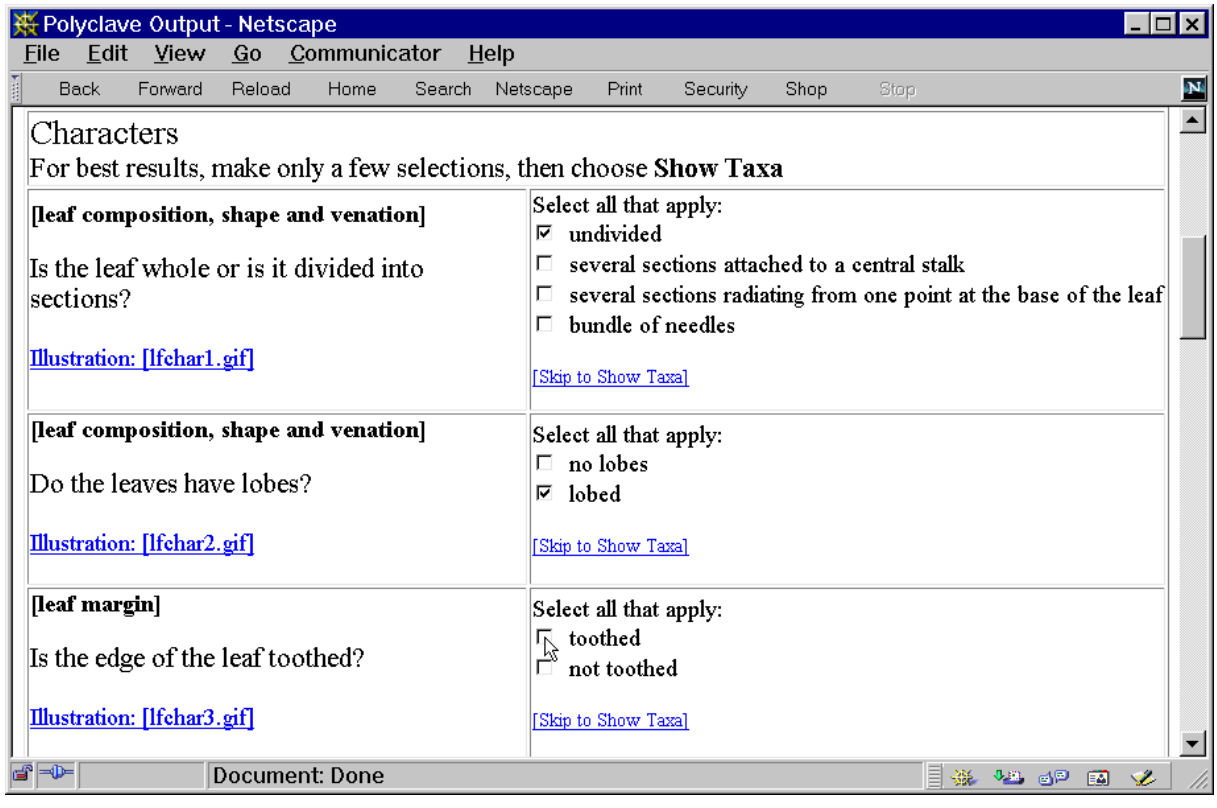
Programs of this type do not have to be installed before use, as they reside entirely in the server. The Web browser handles only standard HTML pages generated by the server. Typically, many of the HTML pages contain the whole character list or a substantial part of it, which may have to be downloaded afresh after a transaction. There is therefore a continuing heavy load on the network and server. Response times may be slow because of the amount of information downloaded at each transaction, and because of slow computation in the server if it is also carrying out tasks for other users. The user interface tends to be cumbersome, because of the limitations of HTML. The programs cannot be used off line.

Examples of this type of program are:
20q (http://www.discoverlife.org/nh/id/)
3I (http://ctap.inhs.uiuc.edu/dmitriev/3i.asp)
DAP (Delta Access Perl) (http://www.axel-findling.de/programs/dap/)
PollyClave (http://prod.library.utoronto.ca:8090/polyclave/index.html)

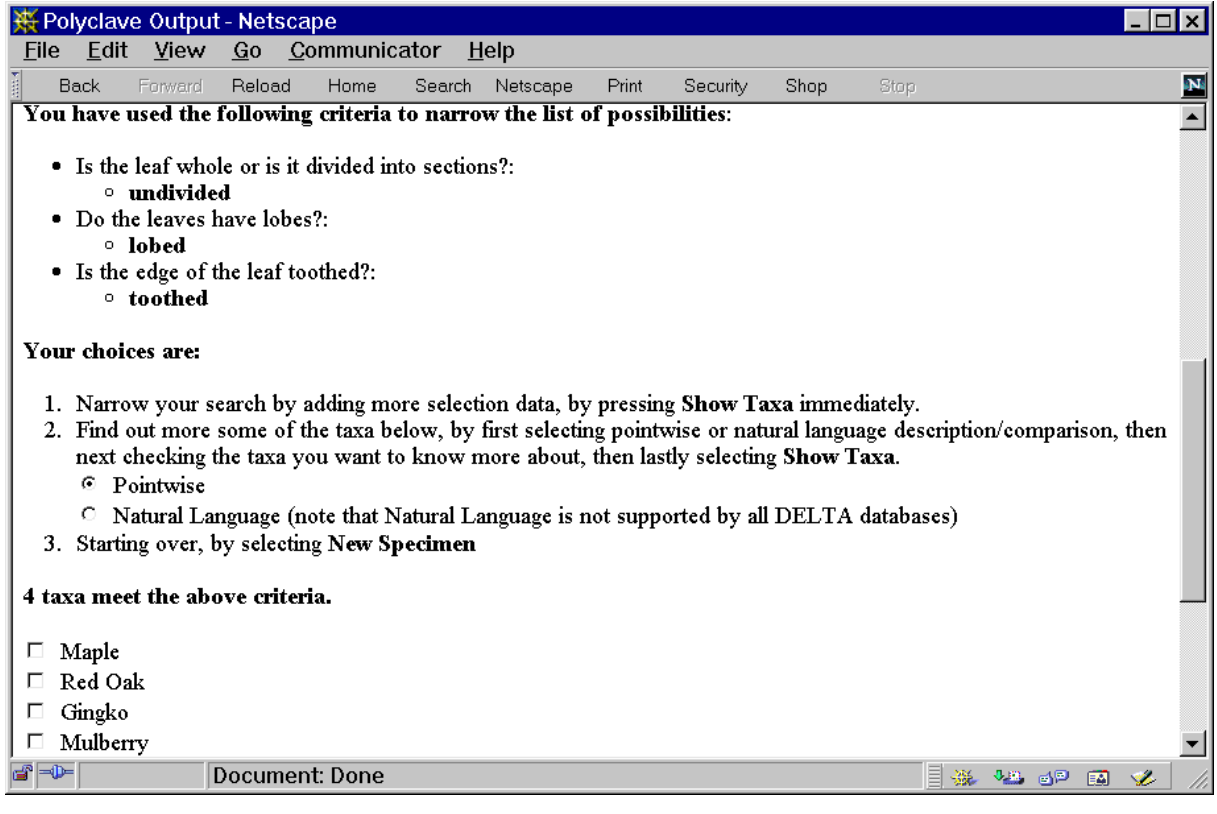
All are free for non-commercial use. There are examples of keys using DAP at http://www.lias.net/index.html, and examples of keys using the other programs at the above sites.

Only small PollyClave data sets are available on the Internet. The response times for these are typical of small Web transactions — about 2–4 seconds. With a data matrix of about 500 characters and 500 taxa, I estimate that loading the character list would take about 20 seconds. This operation may be required at each cycle of the identification, but the browser's 'back' button can be used in some circumstances (even that would take 8 seconds). In each cycle, states can be selected from 1 or more characters, though using several characters without the guidance of 'Best' will increase the chance of errors. After states have been selected, it would take about 7 seconds for the program to respond with the list of remaining taxa.

The following PollyClave screen shows the start of an identification. The state 'undivided' of the character 'Is the leaf whole or is it divided into sections?' and the state 'lobed' of the character 'Do the leaves have lobes?' have been selected. The state 'toothed' of the characters 'Is the edge of the leaf toothed?' is about to be selected. The user then moves to the bottom of the page (not visible in this screen) by means of the scroll bar or the link 'Skip to Show Taxa', and presses the button 'Show Taxa Matching Selections'.



The following screen is displayed. It shows that 4 taxa remain. The user can then return to the previous screen by using the browser's 'Back' button, or obtain the best characters to separate the remaining taxa by pressing the 'Rank Characters' button at the bottom of the page (not visible on this screen).



References

Aiken, S.G., Dallwitz, M.J., McJannet, C.L., and Consaul, L.L. 1996 onwards. *Festuca* of North America: descriptions, illustrations, identification and information retrieval. http://www.mun.ca/biology/delta/arcticf/

Dallwitz, M.J. 1974. A flexible computer program for generating identification keys. *Syst. Zool.* 23: 50–7.

Dallwitz, M.J., 1980. A general system for coding taxonomic descriptions. *Taxon* 29: 41–46.

Dallwitz, M.J., 1992. A comparison of matrix-based taxonomic identification systems with rule-based systems. In 'Proceedings of IFAC workshop on expert systems in agriculture', pp. 215–218. (Ed. F. L. Xiong.) (International Academic Publishers, Beijing.) Also available at http://delta-intkey.com

Dallwitz, M.J., 1993. DELTA and INTKEY. In 'Advances in computer methods for systematic biology: artificial intelligence, databases, computer vision', pp. 287–296. (Ed. R. Fortuner;) (The Johns Hopkins University Press, Baltimore, Maryland.)

Dallwitz, M.J., 1996 onwards. Programs for interactive identification and information retrieval. http://delta-intkey.com

Dallwitz, M.J., 2000 onwards. A comparison of interactive identification programs. http://delta-intkey.com

Dallwitz, M.J., Paine, T.A., Zurcher, E.J., 1993 onwards. User's guide to the DELTA system: a general system for processing taxonomic descriptions. http://delta-intkey.com

Dallwitz, M.J., Paine, T.A., and Zurcher, E.J. 1995 onwards. User's guide to Intkey: a program for interactive identification and information retrieval. 1st edition. http://delta-intkey.com

Dallwitz, M.J., Paine, T.A., Zurcher, E.J., 2000 onwards. Principles of interactive keys. http://delta-intkey.com

Dallwitz, M.J., Paine, T.A., and Zurcher, E.J. 2002. Interactive identification using the Internet. In 'Towards a global biological information infrastructure — challenges, opportunities, synergies, and the role of entomology', pp. 23–33. European Environment Agency Technical Report 70. (Eds H. Saarenmaa and E. S. Nielsen.) (EEA, Copenhagen.) http://reports.eea.europa.eu/technical_report_2001_70/en/

Pankhurst, R.J., 1991. Practical taxonomic computing. 202pp. (Cambridge University Press, Cambridge)

Watson, L., and Dallwitz, M.J. 1992 onwards. The families of flowering plants: descriptions, illustrations, identification, and information retrieval. http://delta-intkey.com