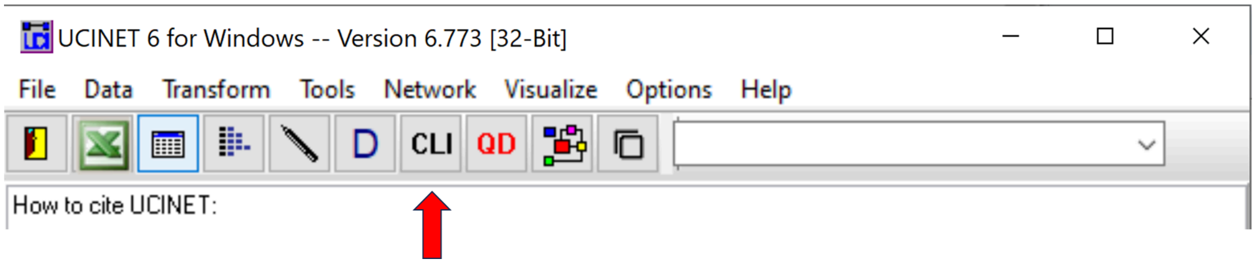
Overview of the Command Line Interface (CLI)

The command line interface (CLI) is what used to be known as the matrix algebra procedure. It provides an alternative to UCINET's menu system. At the end of this document is a summary of network analysis commands available in the CLI.

Access the CLI

To access the CLI in UCINET, go to the main menu and press the CLI button:



Types of commands

There are two kinds of commands: procedures and functions. Functions deliver a result, and this result is a dataset (almost always, a UCINET dataset). Procedures typically produce only textual output, or they produce multiple results. An example of a function is the Total command, which sums the values in a matrix and places the result in a new matrix (in this case called mysum).

```
->mysum = total(mydataset)
```

An example of a procedure is the Display command, which displays the contents of a dataset in matrix form:

```
->display mysum
```

In this manual, and in the CLI itself, we use // to embed comments.

```
        ->mysum = total(mydataset)  //calculates total of mydataset and places result in mysum
```

The -> is the prompt that CLI gives when it is waiting for a command. You don't type it, although if you did, it wouldn't hurt anything. (This is useful when cutting and pasting a set of instructions that contain the -> prompts.)

CLI commands can be nested, so that instead of typing

```
->mysum = total(mydataset)
->display mysum
```

You could instead type:

```
->dsp tot(mydataset)
```

The key difference is that the total will only be displayed as textual output, and not saved as a dataset called mysum.

Note also that most CLI commands can be abbreviated, so that Display can be written as Dsp and Total as Tot.

Parameters

Most commands require one or parameters. A parameter is pretty much anything you type after the name of the command. Almost always, the first parameter is a filename, representing the input to the command. Other parameters, if any, are typically options or additional instructions. For example, the Col command extracts a column from a matrix. So you might type

```
->gender = col(myattribs 6)   //extract col 6 from myattribs and call it gender
```

which means extract column 6 from dataset myattribs and put it in a dataset called gender. In this case there were two parameters, "myattribs" which is the name of a dataset, and "6" which is a column number.

Parameters must be separated by spaces or commas. If single parameters contains spaces or commas (such as a filename) you must enclose it in full quotes as in:

```
->gender - col("my long-named dataset" 6)
```

Much of the CLI consists of arithmetic or mathematical functions, such as totals, taking reciprocals, multiplying matrices, and so on. There are also routines for importing and exporting dataset, as well as multivariate statistical routines and network analysis routines.

Matrices

In the CLI, everything is a matrix. Even a single number is thought of as a 1-by-1 matrix. For example, when you ask CLI to output the sum of all values in a matrix called pv960, the result looks like this:

```
                1
              sum
            -----
    1 PV960 77080
```

Note the extraneous 1 on the left and the first line. These indicate the row and column numbers of the matrix.

Most datasets consist of a single matrix, typically an adjacency matrix representing ties among a set of nodes. But datasets can also contain multiple matrices, although they must have the exact same number of rows of columns. In fact, such datasets are probably better thought of as 3-dimensional matrices, with rows, columns and levels. If you think of them that way, there is no question that the various matrices in a dataset must have the same rows and columns: the matrices are just 2-dimensional slices of a 3-dimensional object.

If you display a 3D matrix in CLI, it will display each level of the matrix one after the other. For example, the Padgett dataset contains two matrices, one called Padgm and the other called Padgb. Displaying Padgett gets you this:

```
->dsp padgett

Matrix: PADGM


                 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
                AC AL BA BI CA GI GU LA ME PA PE PU RI SA ST TO
                CI BI RB SC ST NO AD MB DI ZZ RU CC DO LV RO RN
                AI ZZ AD HE EL RI AG ER CI  I ZZ  I LF IA ZZ AB
                UO  I OR RI LA    NI TE        I     I TI  I UO
                 L     I     N        S                       N
                -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
   1 ACCIAIUOL   0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
   2   ALBIZZI   0  0  0  0  0  1  1  0  1  0  0  0  0  0  0  0
   3 BARBADORI   0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0
   4  BISCHERI   0  0  0  0  0  0  1  0  0  0  1  0  0  0  1  0
   5 CASTELLAN   0  0  1  0  0  0  0  0  0  0  1  0  0  0  1  0
   6    GINORI   0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   7  GUADAGNI   0  1  0  1  0  0  0  1  0  0  0  0  0  0  0  1
   8 LAMBERTES   0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
   9    MEDICI   1  1  1  0  0  0  0  0  0  0  0  0  1  1  0  1
  10     PAZZI   0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
  11   PERUZZI   0  0  0  1  1  0  0  0  0  0  0  0  0  0  1  0
  12     PUCCI   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  13   RIDOLFI   0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  1
  14  SALVIATI   0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0
  15   STROZZI   0  0  0  1  1  0  0  0  0  0  1  0  1  0  0  0
  16 TORNABUON   0  0  0  0  0  0  1  0  1  0  0  0  1  0  0  0

Matrix: PADGB
```

```
              1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
             AC AL BA BI CA GI GU LA ME PA PE PU RI SA ST TO
             CI BI RB SC ST NO AD MB DI ZZ RU CC DO LV RO RN
             AI ZZ AD HE EL RI AG ER CI  I ZZ  I LF IA ZZ AB
             UO  I OR RI LA    NI TE        I     I TI  I UO
              L     I     N        S                       N
             -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 1 ACCIAIUOL  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 2   ALBIZZI  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 3 BARBADORI  0  0  0  0  1  1  0  0  1  0  1  0  0  0  0  0
 4  BISCHERI  0  0  0  0  0  0  1  1  0  0  1  0  0  0  0  0
 5 CASTELLAN  0  0  1  0  0  0  0  1  0  0  1  0  0  0  0  0
 6    GINORI  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0
 7  GUADAGNI  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0
 8 LAMBERTES  0  0  0  1  1  0  1  0  0  0  1  0  0  0  0  0
 9    MEDICI  0  0  1  0  0  1  0  0  0  1  0  0  0  1  0  1
10     PAZZI  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
11   PERUZZI  0  0  1  1  1  0  0  1  0  0  0  0  0  0  0  0
12     PUCCI  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13   RIDOLFI  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
14  SALVIATI  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
15   STROZZI  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
16 TORNABUON  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
```

Focusing on datasets as 3-dimensional matrices makes CLI functions a lot more understandable. However, it is also convenient to think of datasets as containers of multiple matrices, and often a CLI procedure will be applied separately to each matrix in a dataset. Typically, more mathematical commands treat all three dimensions as equally important and can slice the data across any dimension. In contrast, more analytical routines typically treat multidimensional datasets as collections of 2-dimensional matrices, and apply their analyses separately for each matrix.

- Master list of all CLI functions
- Master list of all CLI procedures