STATIST 1.3.2
User Manual

Jakson Alves de Aquino
jalvesaq@gmail.com

11th December 2005

Contents

1 Introduction **1**

2 Warnings for Windows users **1**

3 Installation from source code **2**

4 Invocation **2**

5 Menu **3**

6 Statist and Gnuplot **4**
6.1 Saving graphics 4
6.2 Box-plot 4

7 Data **5**

8 Manipulating databases **7**
8.1 Extracting columns from fixed width data files 7
8.2 Extracting a sample from a database 8
8.3 Recoding a data base 8
8.4 Selecting cases and computing new variables 10
8.5 Sorting the data base 10
8.6 Merging data files 11

9 Batch/script **12**

10 Useful tips **12**

1 Introduction

`Statist` is an easy to use, light weight statistics program. Everything is in an interactive menu: you have just to choose what you need. `Statist` is Free Software under GNU GPL and comes with absolutely no guarantee.

This manual is an incomplete and non literal translation from the original text written by Dirk Melcher, but with the addition of new material. I'm grateful to Bernhard Reiter for his suggestions of improvements to this document.

2 Warnings for Windows users

Users on GNU/Linux are much more accustomed to use console applications. One helpful feature is the command line completion, where a long file name will be completed after typing the first letters and then pressing tab. The terminal emulators, where you type in commands, can save and scroll over many lines that have come by. And, the most important, GNU/Linux is Free Software where anybody can inspect what the computer does and many people can fix bugs to make this more secure. Please, as soon as you can, try `statist` on a Free Software operating system like GNU/Linux or FreeBSD.

To create graphics with `statist` you will need a version of `gnuplot` that comes with `pgnuplot`. Under Windows, you can't send commands to `gnuplot` through `statist` , as it is possible under Linux, but you can type the commands in the `gnuplot` window.

Be careful: Don't close the `gnuplot` window. You can close only the graphic! If you close the `gnuplot` window you will have to restart `statist` to be able to create graphics again.

Some software used to manipulated data files aren't part of `statist` , but they are available for Windows. Please, search the Internet, looking for the package gnucoreutils, which is one of the GnuWin32 packages. Note, however that their installation and use might not be trivial for a Windows user. Like `statist`, they are easier to use in a Linux terminal emulator than in a DOS window.

The `statist` documentation can be found at `C:\Program Files\statist`, where there is also a sample configuration file for `statist`. You can rename it to `statistrc.txt` and edit it according to your preferences.

Unfortunately, `statist` can't produce colorized output under DOS.

3 Installation from source code

1. Open a terminal.

2. Unpack the source code, compile the program, and become root to install it. That is, type:

```
tar -xvzf statist-1.3.2.tar.gz
cd statist-1.3.2
make
```

```
# optional, if you have "check" installed
make check

# install for all users as root
su -
cd path-to/statist-1.3.2
make install
exit
```

This is the default installation that should work in most GNU/Linux distributions. If the above instructions are not enough for your case, please see the file README for details on how to install `statist` from source code.

4 Invocation

You can simply type:

```
statist data_file
```

However there are also some options that you might find useful, and, then, the invocation will be:

```
statist [ options ] data_file
```

where the available `options` are:

```
--help, -h, -? : print this help message and exit
--color        : colorized output
--silent       : don't print menu etc. (for batch/script usage)
--log          : write results to log file 'statist.log'
--nofile       : don't read a datafile when starting the program
--nobell       : no beep at errors and warnings
--thist        : histogram as text graphic, not gnuplot-graphic
--noplot       : no gnuplot-graphic
--labels <file>: read column titles and value labels from file
--xcols <conf_file> <orig_datafile> <new_data_file>
               : extract columns from a fixed width data file
--xsample <percentage> <database> <dest_file>
               : extract a sample of rows from a file
--bernhard     : special output changes from Bernhard, i.e.:
                 - table output at Miscellaneous/Std. deviation
                 - if --noplot defined no text histogram at
                       Miscellaneous/Standard deviation
--version      : show statist version and exit
```

You can also create and edit the file `~/.statistrc` and set some options there. If you have root privileges, you can also create the file `/etc/statistrc`. Options passed by the command line override the ones read from the `statistrc` file. You can find a sample `statistrc` in the documentation directory (usually `/usr/share/doc/statist`). Finally, if you choose the menu item "Preferences", you can modify some options during `statist` execution.

5 Menu

The program has a simple menu that makes it very easy to use. There is no need of remembering commands. Typing '0' you go to the next higher menu-level, or finishes the program if you already are in the *Main menu*. One tip is important: if you have chosen a menu entry by mistake, you can always cancel the process by pressing the <Return> key before entering any value or answering any question. Then, the last menu will be printed again.

If you choose a statistical procedure from the menu, you will be asked to choose the variables. Often, it's not necessary to type the entire name of a column when inputting variable names for analyzes. For example, if you have a column named

this_really_is_a_big_name

and there is no other column starting with the letter 't', you can simply type 't'. Finally, if you want to select all columns, you might simply type "all" as the name of the first column.

Actually, the whole process is self-explanatory, and you would be able to use the program even without reading this short explanation.

6 Statist and Gnuplot

Gnuplot is an interactive program that makes graphical presentations from data and functions, and `statist` creates `gnuplot` graphics for some functions. Normally, you will not have to open `gnuplot` manually. The prerequisite to use it is simply that the program is installed and in the PATH.

If you know `gnuplot` syntax, you can refine or personalize your graphics, inputting `gnuplot` commands. To do that, choose the menu option *Miscellaneous* | *Enter gnuplot commands*. You can change many things in the graphic, like line colors and types, axes labels etc... Even if you don't know `gnuplot` syntax, you can at least change the graphics title and axes labels because a list of the last commands sent to `gnuplot` will be printed in the screen. The changes will be applied to the current graphic being displayed with the `gnuplot` command "replot".

The `gnuplot` graphics can be disabled invoking the program with the option `--noplot`. This can be useful if you, for example, will work with batch processing or if your database is too big and, thus, `gnuplot` graphics are being generated too slowly.

6.1 Saving graphics

To save a graphic created by `gnuplot`, you can use a program like Gimp or KSnapShot, but there are also other ways of saving graphics. For example, open a terminal, and be sure that the terminal window and the `gnuplot` window don't overlap. Then, type:

```
import name.png
```

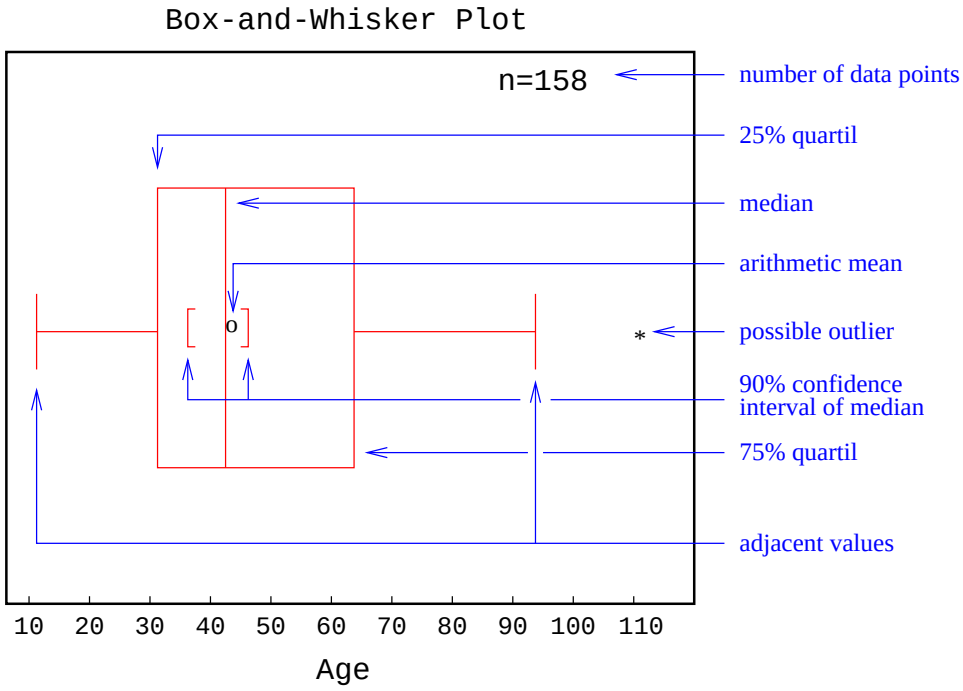
The cursor will become a cross. Then, click on the graph (not on the `gnuplot` window title bar), and the graphic will be saved as name.png. To create a graphic in a different format, choose something else instead of .png (.gif, .bmp, etc). The `import` program is part of *ImageMagik* package.

Another way is to directly use the `gnuplot` drivers, e.g. after a plot enter a few more `gnuplot` commands.

```
set output "output.eps"
set terminal postscript color eps
replot
.
```

6.2 Box-plot

You probably will have no problem interpreting `statist` graphics. The only one that might need some explanation is the *Box-and-Whisker Plot*. The picture below shows the meaning of each piece of this graphic:



7 Data

`Statist` reads data from simple ASCII files (text files). If the program is not invoked with an ASCII file name, it will immediately asks for the the name of a data-file. Without data-file, there is nothing to do, unless you declare the option `--nofile` while invoking the program in order to use the keyboard to input data manually (choose from the menu: *Data management* | *Read column from terminal*). However, only rarely it is reasonable to do this.

A data-file consists of one or several columns of data. The columns of numbers must be separated from each other by either tab character or empty spaces. Missing values must be indicated by the capital letter 'M'. Below is an example of data-file:

```
#Example data-file for statist
  1  3  5  6
  7  8  9 10
 11 12 13 14
 15  M 16  M
```

As you can infer from the above example, commentaries begin with the symbol '#' and are ignored. Empty-lines are also ignored. When `statist` reads the data file, to each column is assigned one variable. The first column will be column 'a', the second will be 'b', etc. However, it will be easier to understand a data file with many variables if its columns have more meaningful names. To do that, begin a line with the "#%". Like commentary lines, the line must begin with one '#', but this symbol must be followed by one '%'. Then, the names will be assigned to the variables:

```
#%kow kaw ec50
0.34 4.56 0.23
1.23 5.45 6.76
6.78 1.34 9.60
```

The number of variable names declared must be exactly the same as the number of columns. Only letters, digits, and '_' are allowed to be used in names, and letters with accents may cause problems. If you use the option `--labels labels_file` `statist` will use the value labels and the column titles present in `labels_file`. When running some graphics and analyzes, `statist` will replace column names and variable values with their labels. A `labels_file` is a list of column names plus their labels followed by a list of values with their labels. Information for different columns are separated by a blank line, as in the example:

```
stat Do you like statistics?
0 No
1 Yes
2 No answer

color What's your favorite color?
```

```
0 Red
1 Green
2 Blue
3 Other
```

In the above example, the datafile has a column named "stat" and other named "color". The values of the variable "stat" are always "0", "1", or "2". You can use the same file with labels for different datafiles. There is no problem if some columns remain without labels, or if some labels don't find their column in the database. Thus, if you have a database with hundreds of columns and want to work with various subsets that share some columns, you can write one single labels file. If you choose in the menu the option "Read another file", the labels will be applied to the appended columns. Note: large value labels will need too much space and the table of "Compare means" can no longer fit in the screen; if you have large labels, you will be able do run "Compare means" with only very few columns at the same time.

Each column of the database is saved as a temporary binary file in `/tmp`, where all values are stored as double precision floating point numbers (real numbers). These files are erased when you quit `statist` . The missing values are stored as the smallest possible number, that is: $-1.7976931348623157 \times 10^{308}$. You have to be sure that this number isn't in your data file as a valid number, because it would not be treated as a very small number; it would be interpreted as a missing value.

Before each analysis, `statist` reads the selected columns from temporary files into ram, and, if necessary, either deletes the rows that have at least one missing value or simply deletes missing values. However, the deletions occur only in a copy of the temporary files that is created in the computer memory. The temporary files remain intact until you quit the program. For example the menu option *Regressions and correlations* | *Multiple linear correlation* will delete all rows that have missing values in any one of the chosen columns. You should do this analysis if each row in your database represents a single case, what is very common in social sciences. The menu option *Tests* | *t-test for comparison of two means of two samples* will delete every missing value, but a missing value in a column will not cause the entire row to be deleted. You should use this analysis if, for example, the columns in your database represent different series of similar experiments, and you would like to compare the two sets of results.

If you want to work only with subsets of your database, you can write columns into a text file (ASCII file), choosing the menu option *Data Management* | *Export columns as ASCII-data*. You can also read data from several files simultaneously (*Data Management* | *Read another file*). When you *Read another file*, new columns are added to the database, and if a column name in the new file is already in use in the current database, the symbol "_" will be appended to it.

Another possibility is to join columns (*Data manipulation* | *Join columns*). In this case, the selected columns will be concatenated in a bigger one.

8 Manipulating databases

8.1 Extracting columns from fixed width data files

To extract columns from a fixed width data file, and save them in a `statist` data file, type:

```
statist --xcols config_file original_datafile new_datafile
```

The content of a "config_file" is simply a list of variable names and their position in the fixed width data file, as in the example below:

```
born 1-4
sex 8-8
income 11-15
```

With the above config_file, `statist` would read the following database:

```
1971 522   2365
19609991  32658
19455632
19674131  32684
```

And output:

```
#%born sex     income
1971    2       2365
1960    1       32658
1945    2       M
1967    1       32684
```

8.2 Extracting a sample from a database

If you will work with a very big database that you still don't know very well, you may find it useful to begin the exploration of the database using a sample of it, which would be faster than using the entire database. After discovering what analyzes are more relevant for your research, you could re-run these analyzes with the original database.

To extract a percentage of the database rows, invoke `statist` in the following way:

```
statist --xsample percentage database dest_file
```

where "percentage" must be a integer number between 1 and 99. The new database, "dest_file" will be created with *approximately* the requested percentage or rows extracted from "data_base".

8.3 Recoding a data base

For some kinds of data manipulation we will need some programs that are not part of `statist`, but are available in most GNU/Linux distributions (and are also installable under DOS/-Windows). For small data files, with few variables, you can use your preferred text editor or spreadsheet program. However, if your file is too big, or has too many variables, it might be more convenient to use the tools described here and in the following sections.

Sometimes, we need to recode some values in a database. Suppose, for example, that in a given data file, the value "999" means missing value for the variable age, and in some analyzes we want "age classes" and not "age". We still want to use the variable "age" in other analyzes, and, thus, we need to recode "age" into a different variable. To create the new data base with the recoded variables we could use `awk`, an external program. Suppose that the column "age" was the second one:

```
awk '{if(/#/) {print $0 "\t" "AGE1"}
        else {
          if(NF == 0) {print $0}
            else {
              if ($2 <= 20){age1 = 1} else
                if ($2 > 20 && $2 <= 50){age1 = 2} else
                if ($2 > 51 && $2 < 999){age1 = 3} else
                {age1 = "M"}
              {print $0 "\t" age1}
            }
       }
     }' datafile.dat > newfile.dat
```

The expression inside the quotes are `awk` commands. With this command, `awk` would read the following data file:

```
#%sex age
2     23
1     88
2     10
2     36
3     999
1     55
```

And output:

```
#%sex  age     AGE1
0       23      2
1       88      3
0       10      1
```

```
0       36      2
M       999     M
1       55      3
```

At first, the `awk` command might looks like complex, but let me explain it:

`$`: The symbol '`$`' means "field", that is, a column of a `statist` data file.

`$0`: has a special meaning: the *entire line*.

`if(/#/) {print $0 "\t" "AGE1"}`: If the line has the symbol '#', print the entire line plus a tab character plus the string "AGE1". This line contains our column names (unless you inserted commentaries in the data file).

`if(NF == 0) {print $0}`: If the number of fields is zero, simply print the entire line.

`if ($2 > 20 && $2 <= 50){age1 = 2}`: If the second field has a value higher than 20 and lower or equal to 50, the value of the variable "age1" will be 2.

`print $0 "\t" age1`: Print the entire line plus a tab character plus the value of the variable `age1`.

We also use `awk` to select cases and compute new variables. So, please refer to its manual or info page for more details on its usage (in a terminal, type `info awk`). Frequently, our `awk` commands will begin testing whether the line contains the column names and whether it is a empty line.

8.4 Selecting cases and computing new variables

We can use `awk` to accomplish two other tasks: (1) create a new data base by selecting only some cases from a existing data file, and (2) compute a new variable using the values of some existing variables. Here we show only two examples of `awk` usage.

Suppose that the second column of a data file has the variable "sex", coded '0' for males and '1' for females, and that we want to include only females in some analyzes. Typing the following command in a terminal would create the new data file we need:

```
awk '{if(/#/ || $2 > 0) {print $0}
}' data_file.dat > new_data_file.dat
```

We are telling `awk` that if either it finds the symbol '#' in a line (because it probably contains our column names, or a commentary), or the second field of a line has a number bigger than 0 it have to output the entire line ("`||`" means "or"). Finally we are also telling to the shell program that we want the output redirected from the screen to the file new_data_file.dat.

Now, suppose that you want to calculate an index using three variables from your data base, and that the index would be the sum of columns 1 and 2 divided by the value of the third column:

```
awk '{if(/#/) {print $0 "\tidx"} else
{{idx = ($1 + $2) / $3}
{print $0 "\t" idx}}}' datafile.dat > newfile.dat
```

Warning: `statist` always uses dot as decimal separator while working with data files. But if the decimal separator in your language is a comma, `awk` will use it in the outputs. To avoid this, type the following command in the terminal before using `awk`:

```
export LC_ALL=en
```

With the above command, the language, numbers, etc will be set to English. Note that programs started in this terminal will also run in English. To reset the terminal you have to "export LC_ALL=xx" again, using your language code instead of "xx" (or close the terminal and open another).

8.5 Sorting the data base

We can use some other programs if we want to sort the rows of the entire database using one more columns as keys. Suppose, for example, that we want to sort our database using the 12th column as key. The following commands would do the job:

```
head -n 1 datafile.dat > columnnames
sort -g -k 12,12 datafile.dat > sorted
cat columnnames sorted > sorted_datafile.dat
```

With the above commands we have sorted our file in three steps: (1) We created the file "columnnames" containing the first line of "datafile.dat". (2) We created the file "sorted", a sorted version of our database. However, in this file the 12th column name was treated as number and its line sorted. It might no longer be the first line of the file. In this case, to create a sorted database with the original names, we use the third command. (3) We concatenated the files "columnnames" and "sorted" to create "sorted_datafile.dat". Please, see manual pages of `head`, `sort`, and `cat` for details on how to use them.

8.6 Merging data files

To merge data files using a variable as key, we use another external program: `join`. Suppose that you have a data file containing information about people, and that some people actually are married with each other. You want to know the mean age difference between husbands and wives. You can't run analyzes to compare people in deferment rows, only variables in different columns. However, your data base has a variable that might be used as key: *house*. People who has the same value for the variable "house" and that are married, actually are married with each other. You should follow some steps to achieve your goal: (1) Use `awk` to create two different data files, one only with married men and other only with married woman. (2) Use `join` to merge the two data files in a new one. If the house variable is the first column in both data files, you should simply type:

```
join -e "" women.dat men.dat > couples.dat
```

The above command would get the two following files:

```
#%house income age              #%house income age
123     4215   23               123     3256   27
124     3251   35               125     4126   25
126     0      20               126     4261   22
127     1241   45               128     3426   60
```

And would output:

```
#%house income age income age
123 4215 23 3256 27
126 0 20 4261 22
```

There is no problem with the duplicate occurrence of “income” and “work”, because `statist` will append ‘_’ to the second one. If you have to merge files using more than one column as key, you can use `awk` to create a single key column that concatenates the characters of all keys. For example, if your key variables are the columns 2 and 3:

```
awk ’{if(/#/) {print "#%key" "\t" $0} else {
       if(NF == 0) {print $0} else {
          {print $2$3 "\t" $0}
        }
       }
      }’ people.dat > people_with_key.dat
```

9 Batch/script

If you have to repeat many times the same analysis, you would became bored of starting `statist` , and, again and again, choosing the same options from the menu. If this is your case, you can use the batch mode. You have to invoke `statist` with the option `--silent`, and give to it a file containing what you would have to type if `statist` was running in the normal mode. The only difference is that while in silent mode `statist` doesn’t print the message "Please, continue with <RETURN>", and, thus, you don’t have to include these <RETURN> keys. For example, if you want to run a correlation between variables “a” and “b” in a data file called “day365.dat” you could create a file named, for example, “autopilot” with the following content:

```
2
1
a
b
0
0
```

The next step would be to invoke `statist` with the following command:

```
statist --silent --noplot day365.dat < autopilot
```

The result will be printed in the screen. However, if you prefer the results saved in a file called, say, report365, type:

```
statist --silent --noplot day365.dat < autopilot > report365
```

10 Useful tips

- Please, report any problem that you find (program bugs, documentation faults, grammar mistakes, etc...) to: statist-list@itevation.de. You are also invited to make suggestions and ask for new features.
- When you see a question like “Do something? (y/N),” the upper case “N” means that if you type any letter other than “y”, and even if you simply press <Enter>, it will be assumed that your answer is “No”.
- You can get the last version of `statist` on its website:

  http://wald.intevation.org/projects/statist/