

# gbutils: command line econometrics

Giulio Bottazzi

September 8, 2015

## Contents

<b>1</b>	<b>Getting Started</b>	<b>1</b>
1.1	Requirements . . . . .	4
1.2	Installation instructions . . . . .	4
<b>2</b>	<b>Documentation</b>	<b>5</b>
<b>3</b>	<b>Contributors</b>	<b>5</b>

## 1 Getting Started

This is a brief description of `gbutils` (version 5.6), a set of command line utilities for the manipulation and statistical analysis of data. These utilities read data from standard input in an ASCII format and print the result in ASCII format to standard output. See the overview for more details and join `gbutils` google group for discussion and news.

In many cases, the output of the utilities in the `gbutils` package is designed in a format suitable to be sent to other program, like the `graph` program in the `plotutils` package, for plotting. Alternatively, these utilities can be used inside an interactive `gnuplot` session, or inside a `gnuplot` script, with the help of the special datafile identifier `<` (see `Gnuplot` documentations for details).

Let us see some examples. Figure 1 reports empirical estimates of the density of 200 realizations of a exponential power random variable, obtained using both a binned distribution and a kernel estimate. This picture can be obtained in terminal with the following list of commands:

```
gbrand -c 1 -r 200 gaussian 1 > data.txt
```

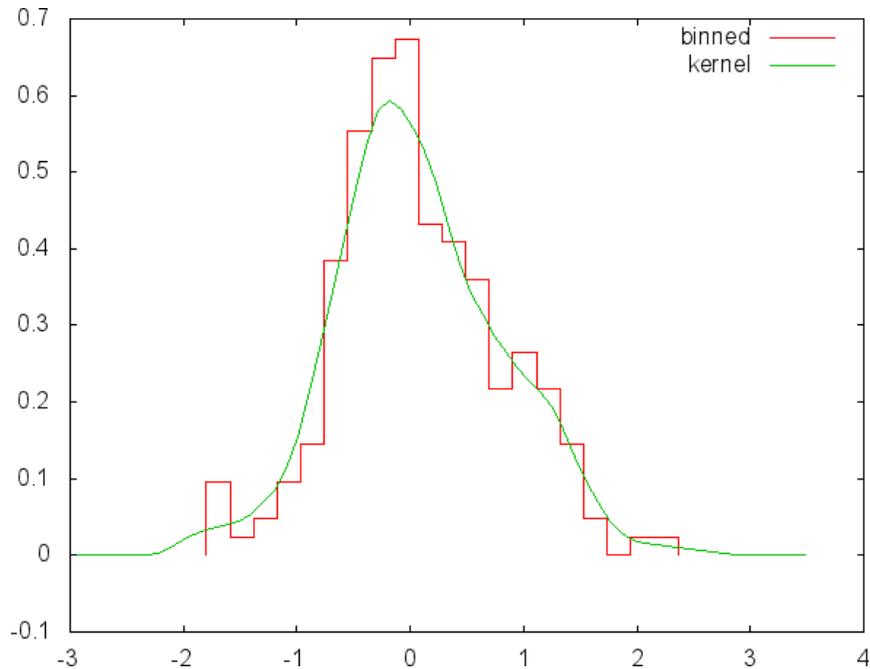


Figure 1: Empirical density estimates: binned distribution and kernel estimate

```
gbker < data.txt >kernel.txt
```

```
gbhisto -n 20 -M 2 <data.txt > histogram.txt
```

```
gbplot -T 'png enhan crop' -o example1.png plot 'w histeps title "binned",\
"kernel.txt" w l title "kernel" ' < histogram.txt
```

```
rm data.dat kernel.txt histogram.txt
```

In the first line, a sample of 200 observations is independently drawn from a Gaussian distribution with unit variance and saved in the file `data.txt`. The second line builds a kernel estimate of the density and the third an histograms. Results are saved in `kernel.txt` and `histogram.txt`, respectively. The command in the fourth line, which continues on the fifth, generates the plot and save the result in `example1.png` and the last line remove all the intermediary files.

The second example, Figure 2, is a scatter plot of couples of points together with a non-linear kernel regression.

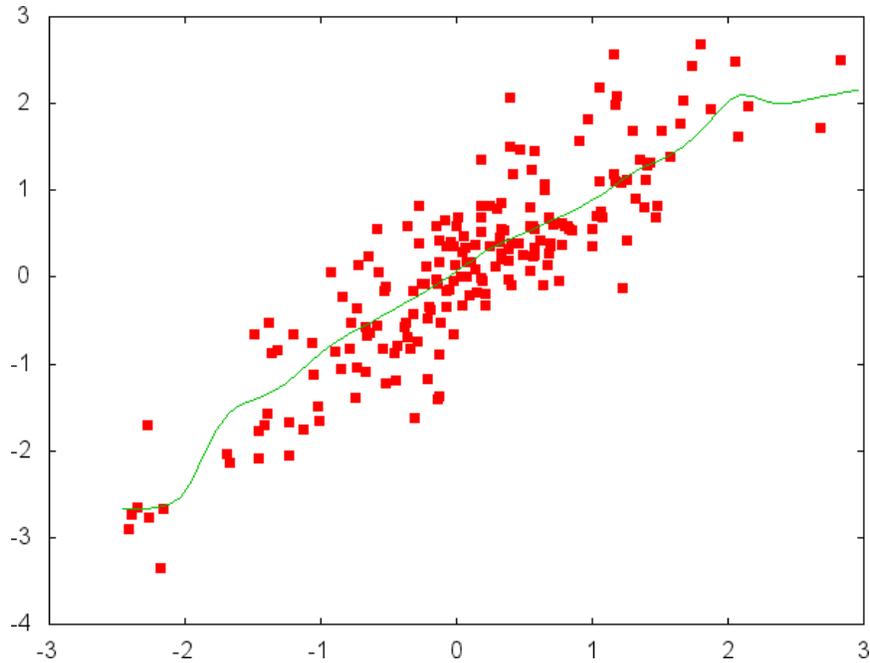


Figure 2: Scatter plot and non-linear regression

```

gbrand -c 2 -r 200 gaussian 1 | gbfun 'x1' 'x1+.5*x2' > data.txt

gbkreg < data.txt > kernel_reg.txt

gbplot -T 'png enhan crop' -o example2.png plot 'w p pt 5,\
"kernel_reg.txt" w l ' < data.txt

rm kernel_reg.txt data.txt
  
```

The data generated in the first line and saved in `data.txt` are independent couples of correlated random variables. A kernel regression is performed in the second line. The third (and fourth) lines produce the plot.

What these commands do is to generate random samples, somehow manipulate them, perform some model estimation and finally plot the result. All this in one line of code.

Both pictures are generated using `gbplot`, which is an interface to the `gnuplot` graphic program. To reproduce these pictures the latter must be installed in your system. Grasping all the details of the lines above is probably complicated. The baseline message, whoever, should be clear: even

complicated analysis can be split in a sequence of very simple steps.

The programs in `gbutils` are completely written in C. Some of them can make use of external libraries (see below).

These programs have been tested (and repeatedly used for many years) on different Linux distributions and should compile and run, in principle, on any Unix platform. Other operating systems, most notably Windows, have been scarcely tested.

These programs have been originally written for personal use. Even if today they are used by several persons, please consider that I maintain and develop them in my spare time. They are distributed under the GPL license in the hope they could be of help to other people, but without any implied warranty.

Please report bugs to <mailto:gbutils@googlegroups.com>

## 1.1 Requirements

To be installed from source, the package requires a C compiler and the standard C library. Unix-like environment in which GNU auto-tools can work is required for automatic installation.

Many programs in `gbutils` do not require ANY external library apart the standard ones. There are however exceptions. Several programs use the GNU Scientific Library (GSL) (version  $\geq 1.6$ ) and other the GNU matheval library (version  $\geq 1.0.1$ ).

See the "Programs summary table" for the list of dependencies of the different utilities.

At install time, if a library is not found, the programs requiring it are omitted from the list of programs to be installed.

If the `zlib` library is found on the system, all the utilities are compiled with the capability of reading gz-compressed input.

For more information about GSL, including installation procedures on various platforms, check [here](#). GNU matheval library is hosted [here](#) and `zlib` home page is [here](#).

Notice that, in principle, using programs linked with different libraries can produce slightly different outputs.

## 1.2 Installation instructions

On Linux, the `gbutils` package can be either installed using a `.deb` package or from source. The first method is recommended. You can download the binaries executable for your system from the [Debian](#) or the [Ubuntu](#)

repositories. Alternatively, the latest source code is available from the `cafed` repository. Check the `README` file in the distributed package for instruction about the installation from source.

In `Windows`, the `gbutils` package is installed in the `Cygwin` environment. Follow the instructions in the `cygwin` installation page.

## 2 Documentation

For a brief survey of the package features see the overview. The `gbget` utility is the Swiss knife of the package, the main tool to perform data manipulation. Its special role commands a specific `gbget` tutorial.

A brief summary of options and modes of operation can be obtained from all the programs using the command line option `-h` or `--help`. All the programs are accompanied by automatically generated manual pages which can be inspected with the `man` Unix utility.

## 3 Contributors

Cees Diks, Federico Tamagni, Angelo Secchi and Davide Pirino provided helpful suggestions and coding.