TG Code Adaptation

Hugues VAN PETEGHEM*

April 2004

Contents

*`http://www.info.fundp.ac.be/~hvp`

1 The Pareto Distribution

1.1 Presentation

This type of distributions is known as "Pareto distributions", from the name of Wilfredo Pareto, the famous Swiss economist/sociologist. He discovered that above a certain size, cumulative income distribution, when plotted on Log-Log scales form an almost straight line. Log-Log scale is commonly used in the analysis of distribution relating the number of people's to their income level.

Pareto discovered the distribution that now bears his name while studying family income distributions in Switzerland. It was common sense knowledge that there are more families with a low income than families with a large income. What was less common knowledge was that these distributions followed a very smooth pattern.

In these terms, the two parameters of the distribution, α and β are respectively, the poverty index and the minimal income level. There is also a last parameter, m, which is the maximal income value.

Later analysis of statistical distributions have demonstrated that Pareto distributions were indeed very common in other fields such as traffic generation.

1.1.1 Mathematical overview

Probability Density Function (pdf): $f(x) = \frac{\alpha\beta^\alpha}{x^{\alpha+1}}$

Cumulative Density Function (cdf): $F(x) = 1 - \left(\frac{\beta}{x}\right)^\alpha$

Expected Value: $E(x) = \frac{\alpha\beta}{\alpha-1}$

Inverse cdf: $F^{-1}(x) = \frac{\beta}{(1-y)^{\frac{1}{\alpha}}}$

Parameters: α(exponent), β(scale) and m(maximal value)

Shape ($\alpha = 3$ & $\beta = 2$):

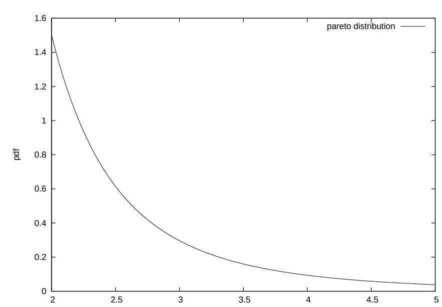

Figure 1: Pareto Distribution.

1.2 Added Code

My code is based on the GSL (GNU Scientific Library) numerical library for C programmer[1]. It is free software under the GNU General Public License. This library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1,000 functions in total. The current version is GSL-1.4. It was released on 14 August 2003. But instead of including the whole library to compute the Pareto distribution, which would be heavy for TG, I decided to include only relevant parts of the GSL code.

1.2.1 tg.y & scan.l

To support the Pareto distribution, the first step is to enable notifying this distribution in the command line of TG. This is done by adding a last distribution in "`tg.y`" (lines 192 and 746^{+}) and in the "`scan.l`" (line 121) to indicate to the Lex parser the existence of this newly added distribution.

```
0   /* tg.y */

191 (...)
192 %token (...) DIST_PARETO
193 (...)
```

```
0   /* tg.y */

745 (...)
746  | DIST_PARETO number number number   /* alpha min max */
747   {
748   char      *cp;
749   if ((cp = dist_pareto_init(&$$.tmpdist, $2.d, $3.d, $4.d)) != NULL)
750   yyerror(cp);
751   }
752   ;
753 (...)
```

```
0   /* scan.l */

120 (...)
121 "pareto"    { count(); BEGIN LEX_NORMAL; return(DIST_PARETO); }
122 (...)
```

[1] http://sources.redhat.com/gsl/

1.2.2 distribution.c & distribution.h

The second step is now to implement the body of the distribution. All the distributions are implemented into the "`distribution.c`" and exported using the "`distribution.h`" file. The work is to introduce into these two files the Pareto distribution by adding three new functions in the first file:

- `proba()`: returns a randomly generated real number between 0 and 1 (both excluded),
- `dist_pareto_gen(dist)`: returns a Pareto distributed random variable within the interval [dist->par2, dist->par3],
- `dist_pareto_init(dist, alpha, min, max)`: initializes the Pareto distributed random variable within the interval [min, max].

And the final step is to modify the "`distribution.h`" (line 43) file to export the new `dist_pareto_init(dist, alpha, min, max)` function:

```
/* distribution.c */

double
proba ()
  {
  double res = (1.0*random()/(RAND_MAX+1.0));
  while (res==0.0 || res == 1.0) res = (1.0*random()/(RAND_MAX+1.0));
  return(res);
  }
```

```
/* distribution.c */

double
dist_pareto_gen(dist)

  distribution *dist;
  {
  double x = proba();
  double z = pow (x, -1/(dist->par1));
  double res = (dist->par2)*z;
  while (res>dist->par3)
    {
    x = proba();
    z = pow (x, -1/(dist->par1));
    res = (dist->par2)*z;
    }
  return(res);
  }
```

```
/* distribution.c */

char *
dist_pareto_init(dist, alpha, min, max)

  distribution *dist;
  double       alpha;
  double       min;
  double       max;
  {

  dist->generate = dist_pareto_gen;
  dist->par1 = alpha;
  dist->par2 = min;
  dist->par3 = max;
  return (NULL);
  }
```

```
/* distribution.h */

(...)
extern char                  *dist_pareto_init();
```

1.3 Usage

Once all the modifications mentioned above are completed, it is possible to use the Pareto distribution just as the exponential one for example. But the Pareto distribution requires three (and only three) parameters. In order: exponent (α), scale (β) and the maximum value.

The Pareto distribution might be, as all the other distributions, used in both **arrival** or **length** argument:

```
# client1.tg
on 0:15 tcp 10.0.14.2.81
at 2 setup
at 0.247940 arrival exponential 0.0083 length pareto 1.1 81.5 66666
packet 12
wait 22.284628
(...)
```

```
# client2.tg
on 0:15 tcp 10.0.14.2.81
at 2 setup
at 13.596063 arrival pareto 2 3 5 length 634
packet 82
wait 11.670262
(...)
```

1.4 Proof

This is only a graphical proof (the mathematical proof will follow soon). On the figure 2, the dotted line represents the Pareto pdf function with parameters: $\alpha = 1.1$, $beta = 81.5$ and $m = 66666$ (see section 1.1.1 for more details).

The histogram is a graphical representation of a set of 10,000 random generated numbers, based on the Pareto distribution (with the same parameters) implemented into TG.

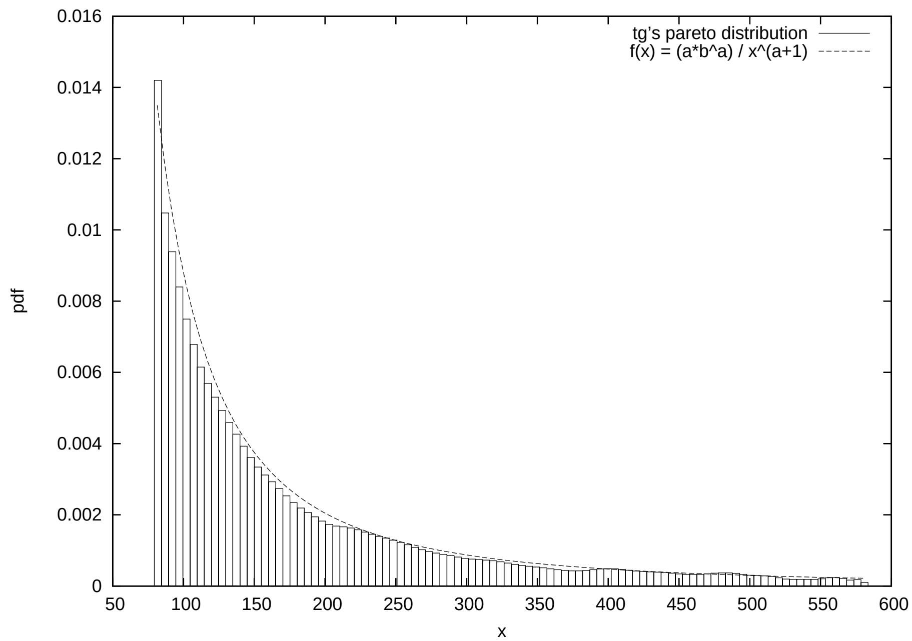

Figure 2: TG's Pareto distribution.

As we can see, the histogram sticks to the line. This means the implementation of the Pareto distribution is acceptable (until a mathematical proof is made).

2 The Exponential Distribution

2.1 Changed code

To fit as much as possible the GSL code, I lightly changed the exponential distribution in the "`distribution.c`" file. The `dist_exp_gen(dist)` function has been transformed to make it interact with the previously presented `proba()` function (see section 1.2.2 for more details):

```
/* distribution.c (old)*/
double
dist_exp_gen(dist)

  distribution   *dist;
  {

  double  value;

  do
    {
    value = -dist->par1 * log(((double)(random() + 1))/
        (double)((unsigned)MAX_RANDOM + 1));
    } while ((value < dist->par2) || (value > dist->par3));

  return(value);
  }
```

```
/* distribution.c (new)*/
double
dist_exp_gen(dist)

  distribution    *dist;
  {
  double u = proba();
  double res = -dist->par1*log(u);
  while (res>dist->par3 || res<dist->par2)
    {
    u = proba();
    res = -dist->par1*log(u);
    }
  return(res);
  }
```

2.2 Usage

The usage of the exponential distribution didn't changed at all. It can still be used in both **arrival** or **length** argument, with one (mean) or three (mean, min, max) arguments. Please, see TG's documentation[2] for more details.

2.3 Proof

As for the Pareto distribution, the exponential distribution proof is only graphical. On the figure 3, the dotted line represents the exponential pdf function centered on 12.

The histogram is a graphical representation of a set of 100,000 random generated numbers, based on the exponential distribution (with the same parameter) implemented into TG.

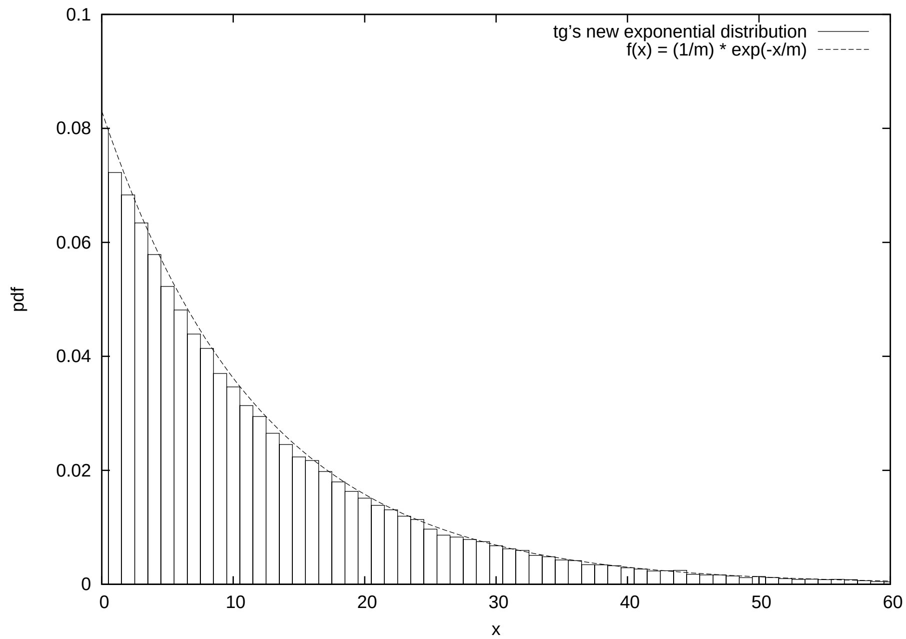

Figure 3: TG's new exponential distribution.

We can take the same conclusion as the one from section 1.4: the implementation of the exponential distribution is acceptable (until a mathematical proof is made).

[2]`http://www.postel.org/tg/tg.htm`