Journal of Statistical Software

February 2015, Volume 64, Issue 2. http://www.jstatsoft.org/

Fitting Heavy Tailed Distributions: The poweRlaw Package

Colin S. Gillespie
Newcastle University

Abstract

Over the last few years, the power law distribution has been used as the data generating mechanism in many disparate fields. However, at times the techniques used to fit the power law distribution have been inappropriate. This paper describes the **poweRlaw** R package, which makes fitting power laws and other heavy-tailed distributions straightforward. This package contains R functions for fitting, comparing and visualizing heavy tailed distributions. Overall, it provides a principled approach to power law fitting.

Keywords: power laws, heavy tailed, fitting, estimation, R, Zipf, Pareto.

1. Introduction

The nineteenth century Italian economist, Vilfredo Pareto, observed that many processes do not follow the Gaussian distribution. This observation leads to the so-called 80/20 rule, that is:

> *80% of all effects results from 20% of all causes.*

This rule has been used to describe a wide variety of phenomena. For example, 20% of employees of any business are responsible for 80% of productive output or 20% of all people own 80% of all wealth. By the middle of the twentieth century, examples of these heavy tailed distributions had been used to describe the number of papers published by scientists, sizes of cities and word frequency (see Keller 2005 for references).

In a similar vein, in 1999 two ground-breaking papers were published in Science and Nature (Barabási and Réka 1999; Albert, Jeong, and Barabási 1999). In the first, the key result was that the distribution of hyper-links in the World Wide Web seemed to follow a power law distribution. Essentially, the connectivity of web pages, k, decreased with rate k^{α}. This

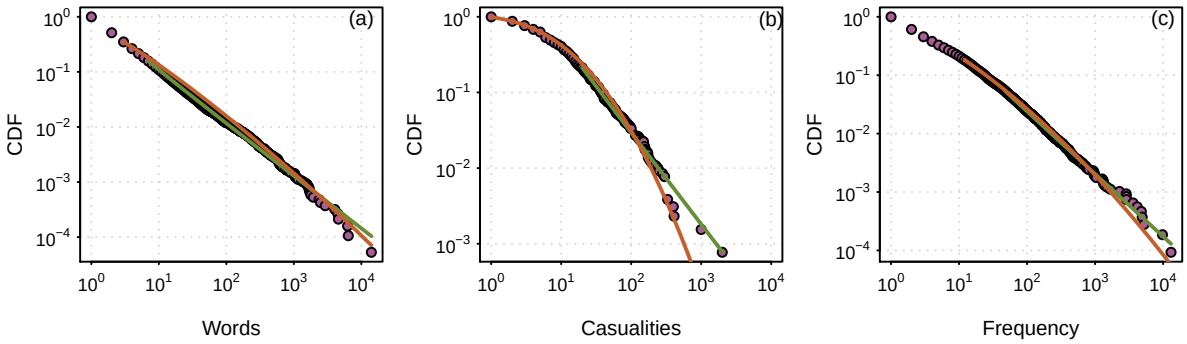

Figure 1: The cumulative distribution functions (CDFs) and their maximum likelihood power law (green) and log normal (orange) fit. (a): Unique words in the novel Moby Dick. (b): Native American Casualities in the American Indian War. (c): Word frequency in the Swiss-Prot database (version 9). Further details of the data sets are given in Section 1.

suggested a large connection variance, with a small number of large key nodes. The second paper presented a model that could generate these networks and coined the phrase *scale-free*. This phrase implicitly linked these networks to the physics of phase transitions.

Since these two landmark papers, there has been an explosion in supposed scale-free phenomena (see Clauset, Shalizi, , and Newman (2009) for a recent overview). Particular examples include

- the occurrence of unique words in the novel *Moby Dick* by Herman Melville (Newman 2005), Figure 1a;
- casualty numbers in armed conflicts (Bohorquez, Gourley, Dixon, Spagat, and Johnson 2009; Friedman 2014), Figure 1b;
- comparing manually curated databases with automatically curated biological databases (Bell, Gillespie, Swan, and Lord 2012), Figure 1c.

Recently, this apparent ubiquity of power laws in a wide range of disparate disciplines was questioned by Stumpf and Porter (2012). The authors point out that many "observed" power law relationships are highly suspect. In particular, estimating the power law exponent on a log-log plot, whilst appealing, is a very poor technique for fitting these types of models. Instead, a systematic, principled and statistical rigorous approach should be applied (see Goldstein, Morris, and Yen 2004).

In this paper we describe the R (R Core Team 2014) package **poweRlaw** (Gillespie 2015). This package enables power laws and other heavy tailed distributions to be fitted in a straightforward manner. In particular, the package provides an easy to use interface to the techniques proposed by Clauset *et al.* (2009). Functions are provided for plotting, comparing distributions and estimating parameter uncertainty.

2. Mathematical background

In this section, we introduce the discrete and continuous power law distributions. We will discuss methods for fitting these distributions. While this section just considers power law

distributions, the techniques discussed in Sections 2.2 and 2.3 are general and can be applied to any distribution.

2.1. The power law distribution

At the most basic level, there are two types of power law distributions: discrete and continuous. The continuous version has probability density function (PDF)

$$p(x) = \frac{\alpha - 1}{x_{\min}} \left(\frac{x}{x_{\min}} \right)^{-\alpha}, \tag{1}$$

where $\alpha > 1$ and $x_{\min} > 0$. While the discrete case has probability mass function (PMF)

$$\mathsf{P}(X = x) = \frac{x^{-\alpha}}{\zeta(\alpha, x_{\min})}, \tag{2}$$

where

$$\zeta(\alpha, x_{\min}) = \sum_{n=0}^{\infty} (n + x_{\min})^{-\alpha} \tag{3}$$

is the generalized zeta function (Abramowitz and Stegun 1972)[1]. When $x_{\min} = 1$, $\zeta(\alpha, 1)$ is the standard zeta function. The cumulative density functions have a relatively simple structure. For the continuous version we have

$$\mathsf{P}(X \leq x) = 1 - \left(\frac{x}{x_{\min}} \right)^{-\alpha+1}, \tag{4}$$

whilst for the discrete version we have

$$\mathsf{P}(X \leq x) = \frac{\zeta(\alpha, x)}{\zeta(\alpha, x_{\min})}. \tag{5}$$

The moments of the power law distribution are particularly interesting. For the continuous power law we have

$$\mathsf{E}[X^m] = \int_{x_{\min}}^{\infty} x^m p(x)\, dx = \frac{\alpha - 1}{\alpha - 1 - m} x_{\min}^m .$$

So when

- $1 < \alpha \leq 2$, all moments diverge, i.e., $\mathsf{E}[X] = \infty$;
- $2 < \alpha \leq 3$, all second and higher-order moments diverge, i.e., $\mathsf{E}[X^2] = \infty$;
- $3 < \alpha \leq m + 1$, all m and higher-order moments diverge, i.e., $\mathsf{E}[X^m] = \infty$.

2.2. Fitting heavy tailed distributions

[1]The **poweRlaw** package uses the zeta function from the **VGAM** package to perform this calculation (see Yee 2010).

Algorithm 1 Estimating the uncertainty in $x_{\min}$ (Clauset, Young, and Gleditsch 2007)

1: Set N equal to the number of values in the original data set.
2: **for** `i` in `1:B`:
3: Sample N values (with replacement) from the original data set.
4: Estimate $x_{\min}$ and α using the Kolmogorov-Smirnov statistic.
5: **end for**

To estimate the scaling parameter α is relatively straightforward. The maximum likelihood estimator (MLE) for the continuous power law is

$$\hat{\alpha} = 1 + n \left[\sum_{i=1}^{n} \ln \frac{x_i}{x_{\min}} \right]^{-1}, \tag{6}$$

where x_i are the observed data values and $x_i \geq x_{\min}$ (Muniruzzaman 1957). The discrete MLE of $\hat{\alpha}$ is not available, instead we use the approximation

$$\hat{\alpha} \simeq 1 + n \left[\sum_{i=1}^{n} \ln \frac{x_i}{x_{\min} - 0.5} \right]^{-1}. \tag{7}$$

The discrete MLE approximation is identical to the exact continuous MLE, except for the additional 0.5 in the denominator (see Clauset *et al.* (2009) for a complete derivation).

When calculating the MLE for α, we *condition* on a particular value of $x_{\min}$. When power laws are used in practice, it is usually argued that only the tails of the distribution follow a power law, and so $x_{\min}$ must be estimated. However as $x_{\min}$ increases, the amount of data *discarded* also increases. So it clear that some care must be taken when choosing this parameter.

The most common approach used to estimate $x_{\min}$ is from a visual inspection of the data on a log-log plot. Clearly, this is highly subjective and error prone. Instead, Clauset *et al.* (2009) recommend estimating the lower threshold using a Kolmogorov-Smirnov approach. This statistic is simply the maximum distance between the data and fitted model CDFs

$$D = \max_{x \geq x_{\min}} |S(x) - P(x)|, \tag{8}$$

where $S(x)$ and $P(x)$ are the CDFs of the data and model respectively (for $x \geq x_{\min}$). The estimate of $x_{\min}$ is the value of $x_{\min}$ that minimizes D. This approach is completely general and can be used in conjunction with other distributions.

2.3. Parameter uncertainty

For a particular value of $x_{\min}$, the standard error of the MLE $\hat{\alpha}$ can be calculated analytically. However, to account for the additional uncertainty of $x_{\min}$ it is necessary to use a bootstrap procedure (Efron and Tibshirani 1993). Essentially, we sample with replacement from the original data set and then re-infer the parameters at each step (see Algorithm 1). The bootstrapping algorithm can be applied to any distribution and can run in parallel.

2.4. Alternative distributions

The techniques discussed in the preceding sections provide flexible methods for estimating

Algorithm 2 Testing the power law hypothesis (Clauset *et al.* 2009)

1: Calculate point estimates for $x_{\min}$ and the scaling parameter α.
2: Calculate the Kolmogorov-Smirnov statistic, KS_d, for the original data set.
3: Set n_1 equal to the number of values below $x_{\min}$.
4: Set $n_2 = n - n_1$ and $P = 0$.
5: **for** `i` in `1:B`:
6: Simulate n_1 values from a uniform distribution: $U(1, x_{\min})$ and n_2 values from a power law distribution (with parameter α).
7: Calculate the associated Kolmogorov-Smirnov statistic, KS_{sim}.
8: If $KS_d > KS_{sim}$, then $P = P + 1$.
9: **end for**
10: $P = P/B$.

distribution parameters and the lower cut-off, $x_{\min}$. In this section, we discuss methods for testing whether the underlying distribution could plausibly have a power law form.

Since it is possible to fit a power law distribution to *any* data set, it is appropriate to test whether the observed data actually follows a power law. A standard goodness-of-fit test is to use bootstrapping to generate a p value to quantify the plausibility of the hypothesis. If the p value is large, then any difference between the empirical data and the model can be explained with statistical fluctuations. If $p \simeq 0$, then the model does not provide a plausible fit to the data and another distribution may be more appropriate. When testing against the power law distribution the hypotheses are:

H_0: data is generated from a power law distribution;

H_1: data is not generated from a power law distribution.

The bootstrapping procedure is detailed in Algorithm 2. Essentially, we perform a hypothesis test by generating multiple data sets (with parameters $x_{\min}$ and α) and then "re-inferring" the model parameters. However, this technique does have computational issues. In particular, when the scaling parameter $\alpha \leq 2$, the first moment (i.e., $\mathsf{E}[X]$) is infinite and so extremely large values frequently occur. Since generating random numbers for the discrete power law distributions involves partitioning the cumulative density this may make this approach unsuitable.

An alternative approach to assessing the power law model is a direct comparison with another model. A standard technique is to use Vuong's test, which is a likelihood ratio test for model selection using the Kullback-Leibler criterion. The test statistic, R, is the ratio of the log likelihoods of the data between the two competing models. The sign of R indicates which model is *better*. Since the value of R is subject to error, we use the method proposed by Vuong (1989). See Appendix C in Clauset *et al.* (2009) for further details.

3. Example: word frequency in Moby Dick

This example investigates the frequency of occurrence of unique words in the novel Moby Dick by Herman Melville (Clauset *et al.* 2009; Newman 2005). The data can be downloaded from `http://tuvalu.santafe.edu/~aaronc/powerlaws/data.htm` or directly loaded from the **poweRlaw** package

```
R> data("moby", package = "poweRlaw")
```

This data set contains the frequency of 18855 words. The most commonly occurring word occurred 14086 times.

3.1. Fitting a discrete power law

To fit a discrete power law, we first create a discrete power law object using the `displ` constructor[2].

```
R> library("poweRlaw")
R> pl_m <- displ$new(moby)
```

The object `pl_m` is an S4 reference object. Initially the lower cut-off, $x_{\min}$, is set to the smallest x value and the scaling parameter, α, is set to `NULL`.

```
R> pl_m$getXmin()

[1] 1

R> pl_m$getPars()

NULL
```

The object also has standard setters:

```
R> pl_m$setXmin(5)
R> pl_m$setPars(2)
```

For a given $x_{\min}$ value, we can estimate the corresponding α value using its MLE.

```
R> estimate_pars(pl_m)

$pars
[1] 1.926

$value
[1] 14873

$counts
function gradient
       5        5

$convergence
[1] 0
```

[2] `displ`: **dis**crete **p**ower **l**aw.

```
$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

attr(,"class")
[1] "estimate_pars"
```

Alternatively, we can estimate the exponent using a parameter scan

```
R> estimate_pars(pl_m, pars = seq(1.5, 2.5, 0.01))
```

To estimate the lower bound $x_{\min}$, we use the Kolmogorov-Smirnov approach described in Section 2.2

```
R> (est_pl <- estimate_xmin(pl_m))

$KS
[1] 0.008253

$xmin
[1] 7

$pars
[1] 1.953

attr(,"class")
[1] "estimate_xmin"
```

For the Moby Dick data set, the minimum is achieved when $x_{\min} = 7$ and $D(7) = 0.00825$. Similar to the `estimate_pars` functions we can limit the search space using the `xmin` and `pars` arguments.

To set the power law object to these optimal values, we just use the `xmin` setter.

```
R> pl_m$setXmin(est_pl)
```

To allow the user to explore different distributions and model fits, all distribution objects have generic plot methods. For example,

```
R> plot(pl_m)
```

creates a log-log plot of the data, while the `lines` function

```
R> lines(pl_m, col = 2)
```

adds the fitted distribution (to get Figure 1a). When calling the `plot` and `lines` function, the data plotted is invisibly returned, i.e.,

```
R> dd <- plot(pl_m)
R> head(dd, 3)
```

```
  x      y
1 1 1.0000
2 2 0.5141
3 3 0.3505
```

This makes it straightforward to create graphics using other R packages.

To fit other distributions, we follow a similar procedure. For example, to fit the discrete log normal distribution, we begin by creating a '`dislnorm`' object and estimating the parameters[3].

```
R> ln_m <- dislnorm$new(moby)
R> est_ln <- estimate_xmin(ln_m)
```

Then we update the object

```
R> ln_m$setXmin(est_ln)
```

and add the corresponding line to the plot

```
R> lines(ln_m, col = 3)
```

giving Figure 1a.

Figure 1 gives example data sets, with associated power law and log normal fits. Plotting the data in this manner has two clear benefits. First, it highlights how much data is being discarded when fitting $x_{\min}$. Second, it provides an easy comparison with other distributions.

3.2. Parameter uncertainty

To get a handle on the uncertainty in the parameter estimates, we use a bootstrapping procedure, via the `bootstrap` function. This procedure can be applied to **any** distribution object. Furthermore, the bootstrap procedure can utilize multiple CPU cores to speed up inference using the base package **parallel**. To generate five thousand bootstrap samples, using four cores, we use the following command.

```
R> bs <- bootstrap(pl_m, no_of_sims = 5000, threads = 4, seed = 1)
```

By default the `bootstrap` function will use the MLE to infer the parameter values and check all values of $x_{\min}$. When the $x_{\min}$ search space is large, then it is recommend that it is truncated. For example

```
R> bootstrap(pl_m, xmins = seq(2, 20, 2))
```

will only calculate the Kolmogorov-Smirnov statistics at values of $x_{\min}$ equal to

$$2, 4, 6, \ldots, 20 \, .$$

A similar argument exists for the parameters.

The bootstrap function returns a '`bs_xmin`' object. This object is a list that consists of three components:

[3] `dislnorm`: **dis**crete log-**norm**al.

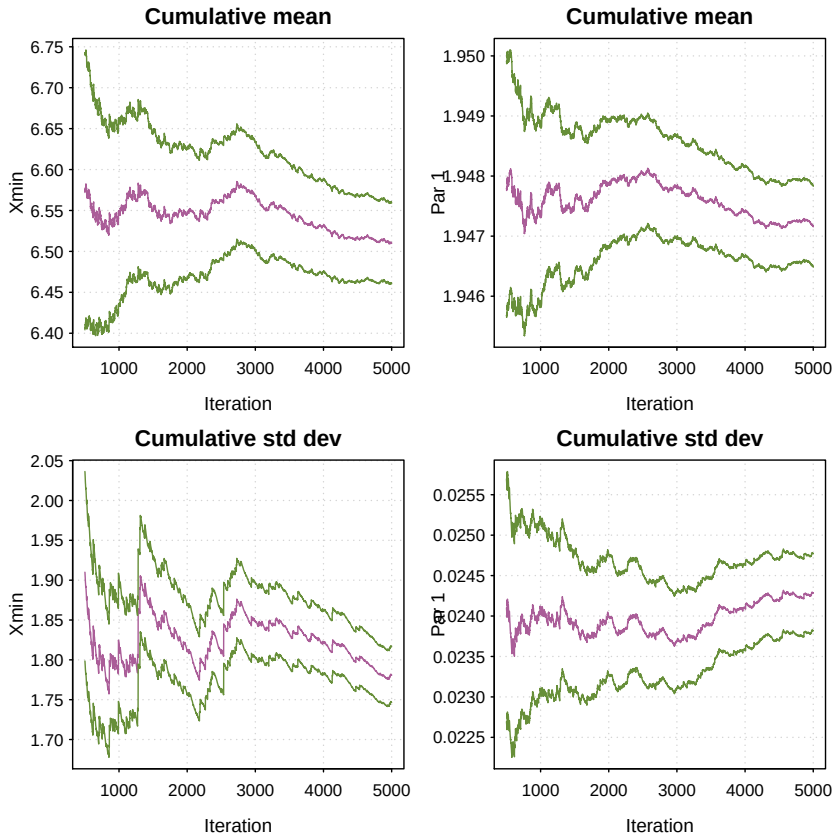

Figure 2: Results from the standard bootstrap procedure (for the power law model) using the Moby Dick data set: `bootstrap(pl_m)`. The top row shows the sequential mean estimate of parameters $x_{\min}$ and α. The bottom row shows the sequential estimate of standard deviation for each parameter. The dashed-green lines give approximate 95% confidence intervals. After 5000 iterations, the standard deviation of $x_{\min}$ and α is estimated to be 1.8 and 0.02 respectively.

1. `gof`: the goodness-of-fit statistic obtained from the Kolmogorov-Smirnov test. This value should correspond to the value obtained from the `estimate_xmin` function;
2. `bootstraps`: a data frame containing the results from the bootstrap procedure;
3. `sim_time`: the average simulation time, in seconds, for a single bootstrap.

The bootstrap results can be explored in a variety of ways. First we can estimate the standard deviation of the parameter uncertainty, i.e.,

```
R> sd(bs$bootstraps[, 2])

[1] 1.781

R> sd(bs$bootstraps[, 3])

[1] 0.02429
```

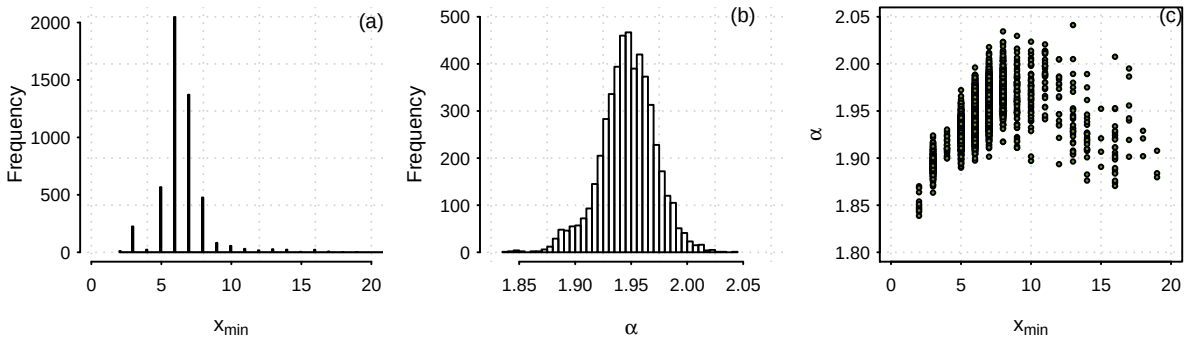

Figure 3: Characterizing uncertainty in parameter values using five thousand bootstraps. (a) Histogram of $x_{\min}$ (standard deviation 1.8). (b) Histogram of α (standard deviation 0.02). (c) Scatter-plot of the $x_{\min}$ against α.

Alternatively, we can visualize the results using the `plot` method

```
R> plot(bs, trim = 0.1)
```

to obtain Figure 2. The top row of graphics in Figure 2 give a sequential 95% confidence interval for the mean estimate of the parameters. The bottom row of graphics give a 95% confidence interval for the standard deviation of the parameters. The parameter `trim` in the `plot` function controls the percentage of samples displayed. When `trim = 0`, all iterations are displayed. When `trim = 0.1`, we only display the final 90% of data.

We can also construct histograms of the parameters

```
R> hist(bs$bootstraps[, 2], breaks = "fd")
R> hist(bs$bootstraps[, 3], breaks = "fd")
```

to get Figures 3a & b. A joint scatter plot is useful in highlighting the strong dependency that often exists between the scaling parameter α and $x_{\min}$

```
R> plot(bs$bootstraps[, 2], bs$bootstraps[, 3])
```

and yields Figure 3c.

A similar bootstrap analysis can be obtained for the log normal distribution

```
R> bootstrap(ln_m)
```

In this case we would obtain uncertainty estimates for both of the log normal parameters.

3.3. Comparison to other distributions

The main thrust of Stumpf and Porter (2012) is that many of the systems that are characterized as having a power law distribution, could equally come from another heavy tailed distribution. The **poweRlaw** package provides two methods for testing the power law hypotheses.

The first method uses the bootstrapping technique described in Algorithm 2. This is accessed using a similar interface as the standard bootstrap function.

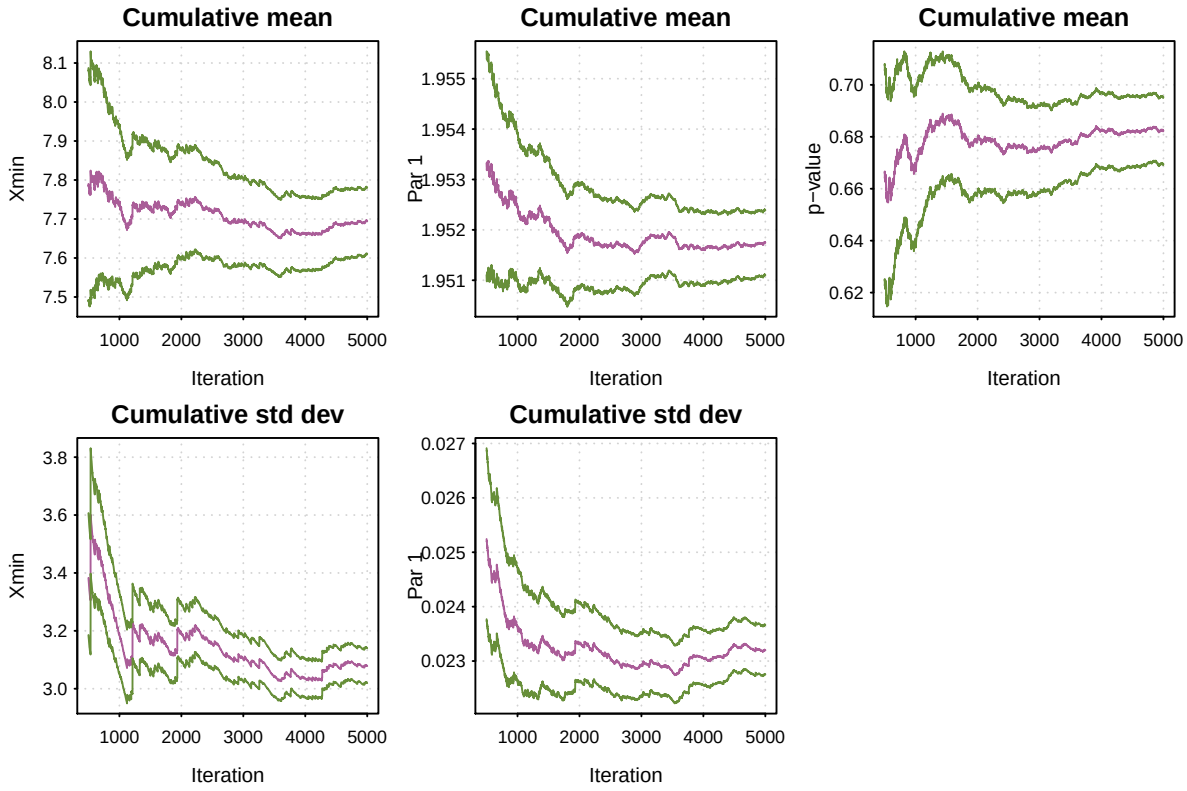

Figure 4: Results from the bootstrap procedure for determining the plausibility of the power law hypothesis for the Moby Dick data set: `bootstrap_p(m_pl)`. The top row shows the sequential mean estimate of parameters $x_{\min}$, α and the p value. The bottom row shows the sequential estimate of standard deviation for each parameter. The dashed-green lines give approximate 95% confidence intervals.

```
R> bs_p <- bootstrap_p(pl_m, no_of_sims = 5000, threads = 4, seed = 1)
```

Again this function can be run in parallel (using the `threads` argument) and has the option to restrict the $x_{\min}$ search space. The output from the `bootstrap_p` function has very similar structure to the `bootstrap` function. However, this function does return one additional element – the p value for the hypothesis test:

H_0: data is generated from a power law distribution;

H_1: data is not generated from a power law distribution.

In this particular example, we estimate $p = 0.681$, i.e., the underlying distribution for generating the Moby Dick data set could be a power law distribution. Again, the output can be easily visualized with

```
R> plot(bs_p)
```

to obtain Figure 4. Notice that Figure 4 has an additional plot for the p value. This enables the user to assess the accuracy of the estimated p value.

The second method is to directly compare two distributions using a likelihood ratio test. For this test, both distributions must use the same $x_{\min}$ value. For example, to compare the power law model to the log normal, we first the set threshold to be the same as the power law model.

```
R> ln_m <- dislnorm$new(moby)
R> ln_m$setXmin(7)
```

Next we estimate the parameters (conditional on $x_{\min} = 7$)

```
R> est <- estimate_pars(ln_m)
```

and update the model

```
R> ln_m$setPars(est)
```

Then we can use Vuong's method to compare models.

```
R> comp <- compare_distributions(pl_m, ln_m)
```

The object `comp` object contains Vuong's test statistic, p value and the ratio of the log likelihoods. For this particular comparison, we have $p = 0.682$ which relates to the hypotheses

H_0: both distributions are equally far from the true distribution;
H_1: one of the test distributions is closer to the true distribution.

Hence, we cannot reject H_0 and it is not possible to determine which is the best fitting model.

4. Package overview

In the previous example we created a '`displ`' object

```
R> pl_m <- displ$new(moby)
```

to represent the discrete power law distribution. This particular object has class '`displ`' and also inherits the '`discrete_distribution`' class. Other available distributions are given in Table 1.

The classes given in Table 1 are S4 reference classes[4]. Each '`distribution`' object has four fields:

- `dat`: the data set;
- `xmin`: the lower cut-off $x_{\min}$;
- `pars`: a vector of parameter values;
- `internal`: a list of values used in different numerical procedures. This will differ between distribution objects. In general, the user will not interact with the `internal` field.

[4]See `?setRefClass` for further details on reference classes.

Distribution	Class name	# of parameters
Discrete power law	'`displ`'	1
Discrete log normal	'`dislnorm`'	2
Discrete exponential	'`disexp`'	1
Poisson	'`dispois`'	1
Continuous power law	'`conpl`'	1
Continuous log normal	'`conlnorm`'	2
Exponential	'`conexp`'	1

Table 1: Available distributions in the **poweRlaw** package. Each class also inherits either the '`discrete_distribution`' or '`ctn_distribution`' class.

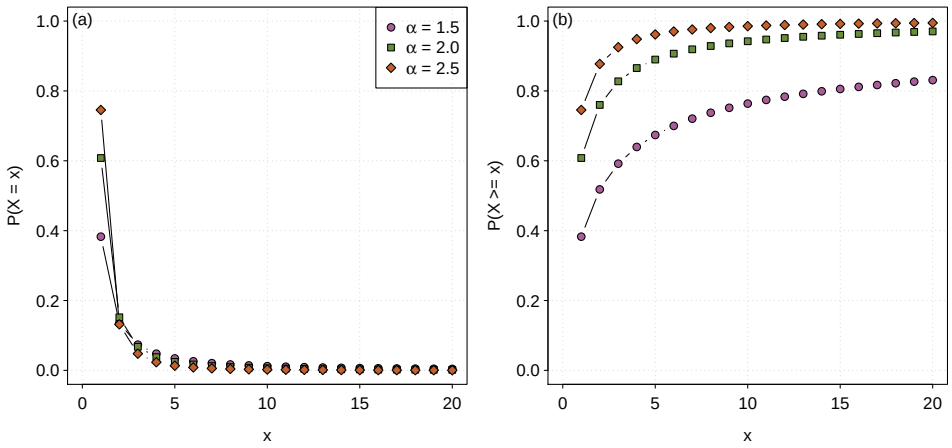

Figure 5: The (a) probability mass function and (b) probability distribution function for the discrete power law, where $x_{\min} = 1$ and α as indicated.

Using this particular object orientated framework has three distinct benefits.

1. After fitting a single distribution, fitting all other distributions follows an almost identical route.

2. It is straightforward to add new distributions to the package.

3. The `internal` field allows efficient caching of data structures when updating the `xmin` and `pars` fields. In particular, when the data is first loaded, efficient vector operations can be carried out and used as a look-up table, i.e., taking log's of the data.

Distribution objects have a number of methods available (see Table 2). All `dist_*` methods depend on the *type* of distribution. For example, to plot the probability mass function of the discrete power law distribution, we first create a discrete power law object

```
R> m <- displ$new()
R> m$setXmin(1)
```

then use the `dist_pdf` function to obtain the probabilities for particular parameter values

Method name	Description
`dist_cdf`	Cumulative density/mass function (CDF)
`dist_pdf`	Probability density/mass function (PDF)
`dist_rand`	Random number generator
`dist_data_cdf`	Data CDF
`dist_ll`	Log likelihood
`estimate_xmin`	Point estimates of the cut-off point and parameter values
`estimate_pars`	Point estimates of the parameters (conditional on the current $x_{\min}$ value)
`bootstrap`	Bootstrap procedure (uncertainty in $x_{\min}$)
`bootstrap_p`	Bootstrap procedure to test whether we have a power law

Table 2: A list of methods available for '`distribution`' objects. These methods do not change the object states.

```
R> x <- 1:20
R> m$setPars(1.5)
R> plot(x, dist_pdf(m, x), type = "b")
R> m$setPars(2.0)
R> lines(x, dist_pdf(m, x), type = "b")
R> m$setPars(2.5)
R> lines(x, dist_pdf(m, x), type = "b")
```

This gives Figure 5a. Likewise, to obtain the CDF we use the `dist_cdf` function, i.e.,

```
R> plot(x, dist_cdf(m, x), type = "b")
```

to obtain Figure 5b.

The other methods, `estimate_*` and `bootstrap_*`, work with general distribution objects (although internally they use `dist_*` methods). See the associated help files for further details.

5. Conclusion

In recent years an over-enthusiastic fitting of power laws to a wide variety of systems has resulted in the inevitable (and needed) call for caution. Stumpf and Porter (2012) correctly highlight that many supposed power law relationships are at best dubious and some obviously false. These problems in determining the underlying distribution of these mechanisms can (in some part) be attributed to the lack of available and easy to use software packages for fitting heavy tailed distributions. The **poweRlaw** package aims to solve this problem. By providing an easy to use and consistent interface, researchers can now fit, and more importantly, compare a variety of truncated distributions fitted to their data set.

Acknowledgements

The author gratefully acknowledges Aaron Clauset and Jeff Friedman for their constructive comments on the manuscript and discussions on the implementation of the methodology.

References

Abramowitz M, Stegun IA (1972). *Handbook of Mathematical Function with Formulas, Graphs, and Mathematical Tables*, volume 55 of *Applied Mathematics Series*. 10th edition. National Bureau of Standards.

Albert R, Jeong H, Barabási AL (1999). "Internet: Diameter of the World-Wide Web." *Nature*, **401**(6749), 130–131.

Barabási AL, Réka A (1999). "Emergence of Scaling in Random Networks." *Science*, **286**(5439), 509–512.

Bell MJ, Gillespie CS, Swan D, Lord P (2012). "An Approach to Describing and Analysing Bulk Biological Annotation Quality: A Case Study Using UniProtKB." *Bioinformatics*, **28**(18), i562–i568.

Bohorquez JC, Gourley S, Dixon AR, Spagat M, Johnson NF (2009). "Common Ecology Quantifies Human Insurgency." *Nature*, **462**(7275), 911–914.

Clauset A, Shalizi CR, , Newman MEJ (2009). "Power-Law Distributions in Empirical Data." *SIAM Review*, **51**(4), 661–703.

Clauset A, Young M, Gleditsch KS (2007). "On the frequency of severe terrorist events." *Journal of Conflict Resolution*, **51**(1), 58–87.

Efron B, Tibshirani R (1993). *An Introduction to the Bootstrap*, volume 57. Chapman & Hall/CRC.

Friedman JA (2014). "Using Power Laws to Estimate Conflict Size." *Journal of Conflict Resolution*. doi:10.1177/0022002714530430. Forthcoming.

Gillespie CS (2015). ***poweRlaw**: Analysis of Heavy Tailed Distributions*. R package version 0.30.0, URL http://CRAN.R-project.org/package=poweRlaw.

Goldstein ML, Morris SA, Yen GG (2004). "Problems with Fitting to the Power-Law Distribution." *The European Physical Journal B*, **41**(2), 255–258.

Keller EF (2005). "Revisiting 'Scale-Free' Networks." *BioEssays: News and Reviews in Molecular, Cellular and Developmental Biology*, **27**(10), 1060–1068.

Muniruzzaman ANM (1957). "On Measures of Location and Dispersion and Test of Hypothesis on a Pareto Distribution." *Calcutta Statistical Association Bulletin*, **7**(27), 115–126.

Newman MEJ (2005). "Power Laws, Pareto Distributions and Zipf's Law." *Contemporary Physics*, **46**(5), 323–351.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Stumpf MPH, Porter MA (2012). "Mathematics. Critical Truths about Power Laws." *Science*, **335**(6069), 665–666.

Vuong QH (1989). "Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses." *Econometrica*, **57**(2), 307–333.

Yee TW (2010). "The **VGAM** Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34. URL http://www.jstatsoft.org/v32/i10/.

Affiliation:

Colin S. Gillespie
School of Mathematics & Statistics
Newcastle University
Newcastle upon Tyne
NE1 7RU, United Kingdom
E-mail: colin.gillespie@newcastle.ac.uk
URL: http://www.mas.ncl.ac.uk/~ncsg3/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 64, Issue 2 *Submitted:* 2013-06-10
February 2015 *Accepted:* 2014-05-28