Power-laws and NetworkX

Rodrigo Dorantes Gilardi
with Louis Shekhtman and Albert-László Barabási

Northeastern University 2021

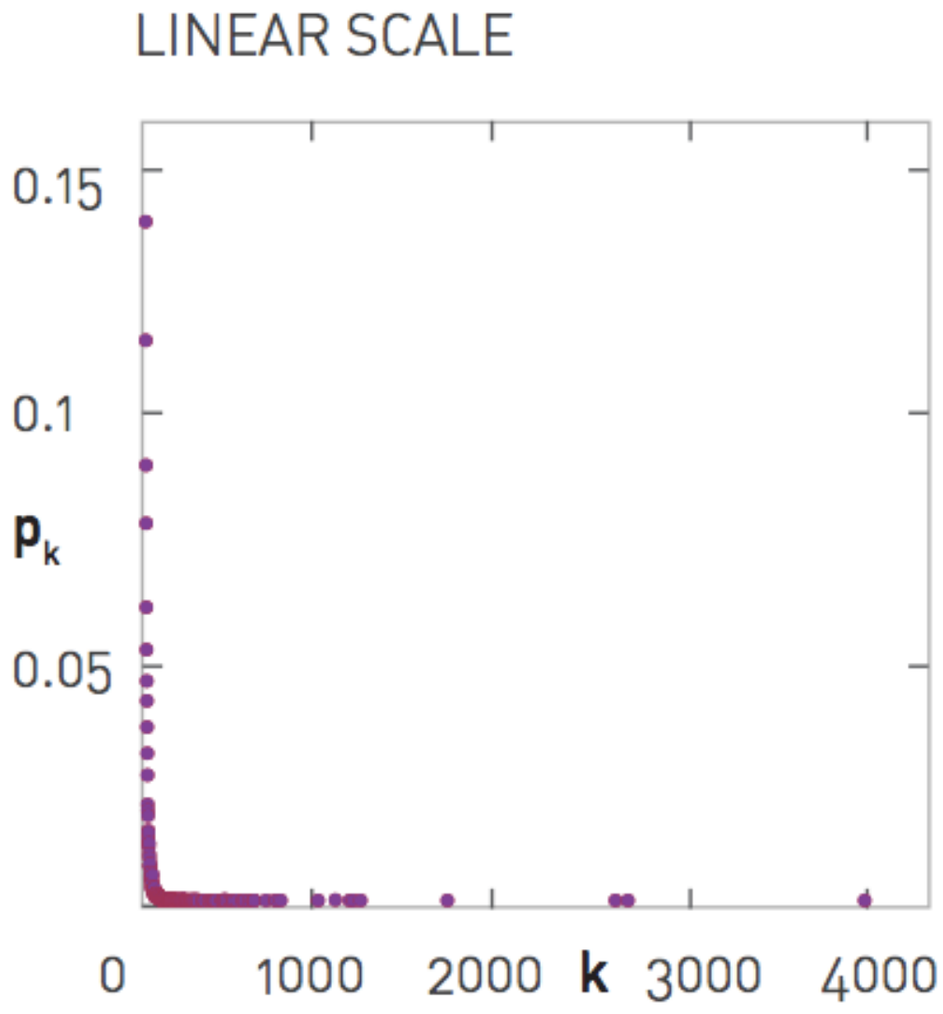
Power-law plotting

$$(k_1, k_2, \ldots, k_{N-1}, k_N)$$

1. Do a histogram

$$p_k = \frac{N_k}{N}$$

(count how many nodes have degree 1, degree 2, etc.)



→ Always label your axes

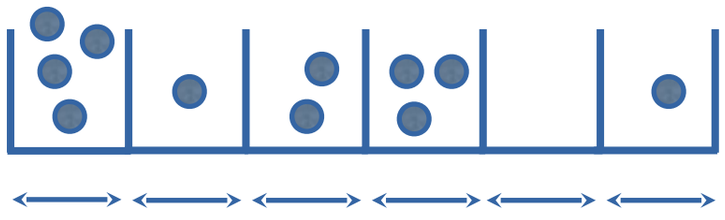
→ Use a large font size

A.-L. Barabási. (2016). *Network Science,*

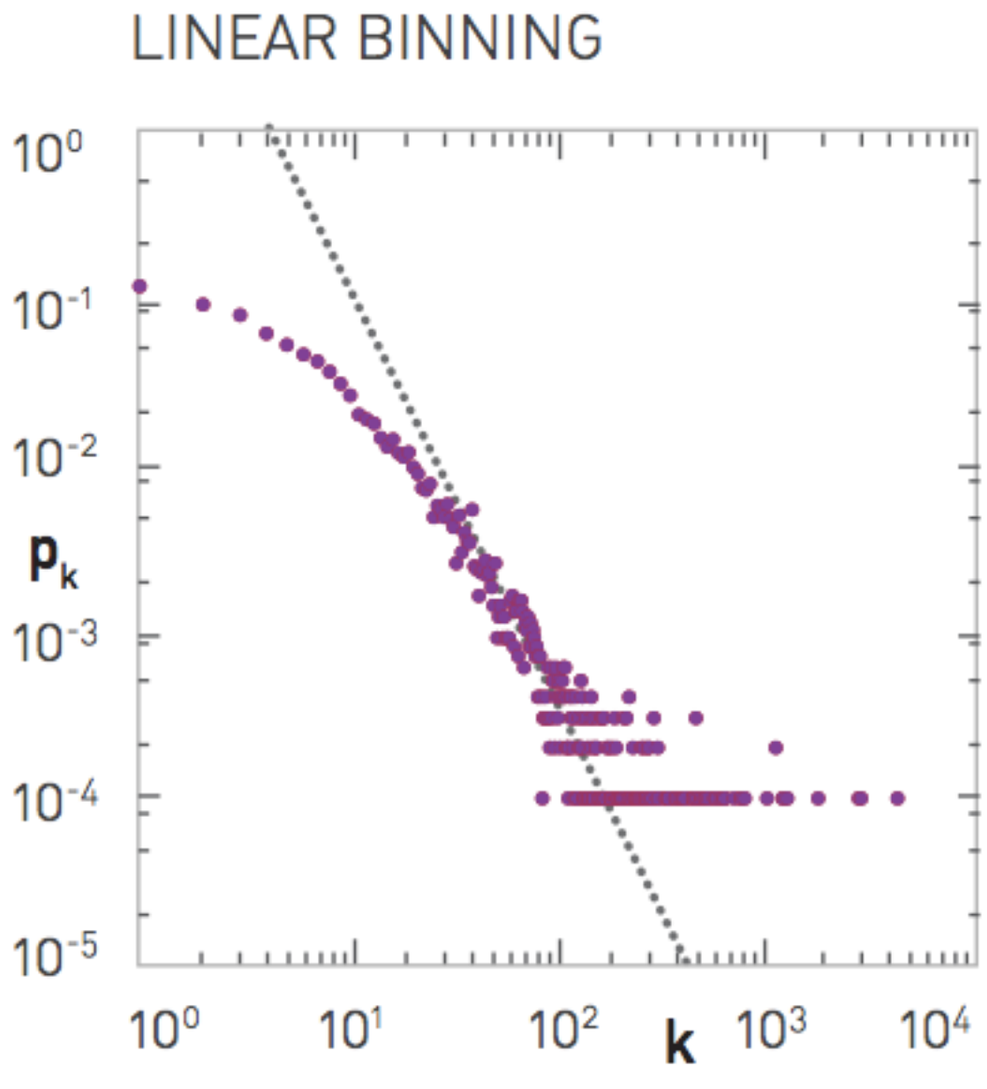
Power-law plotting

$$(k_1, k_2, \ldots, k_{N-1}, k_N)$$

2. Plot in log-log scale



Equally spaced



A.-L. Barabási. (2016). *Network Science,*

Power-law plotting

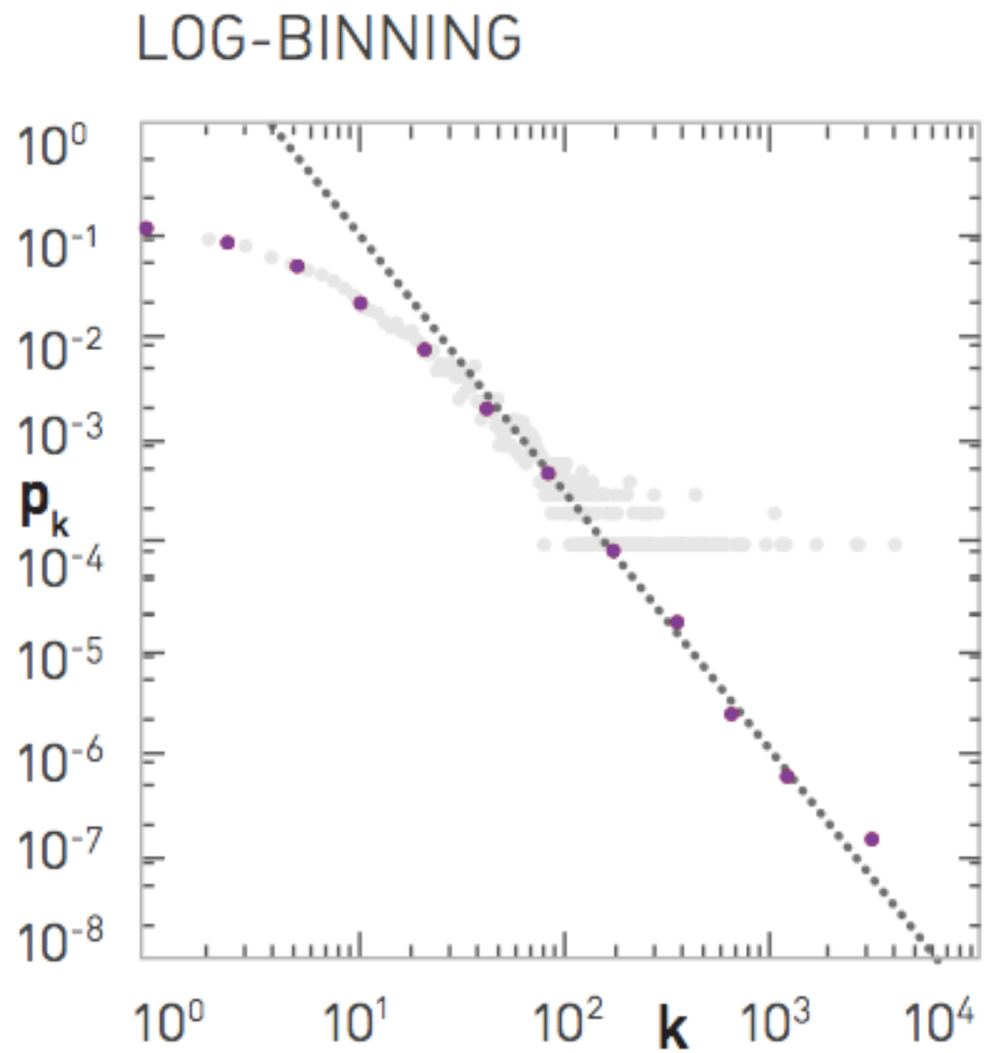
$$(k_1, k_2, \ldots, k_{N-1}, k_N)$$

3. Let’s make it “pretty”

(Not noisy, easier to understand the trend of the tail, etc.)

Possibility 1:

Logarithmic binning



A.-L. Barabási. (2016). *Network Science,*

Power-law plotting (Avoid)

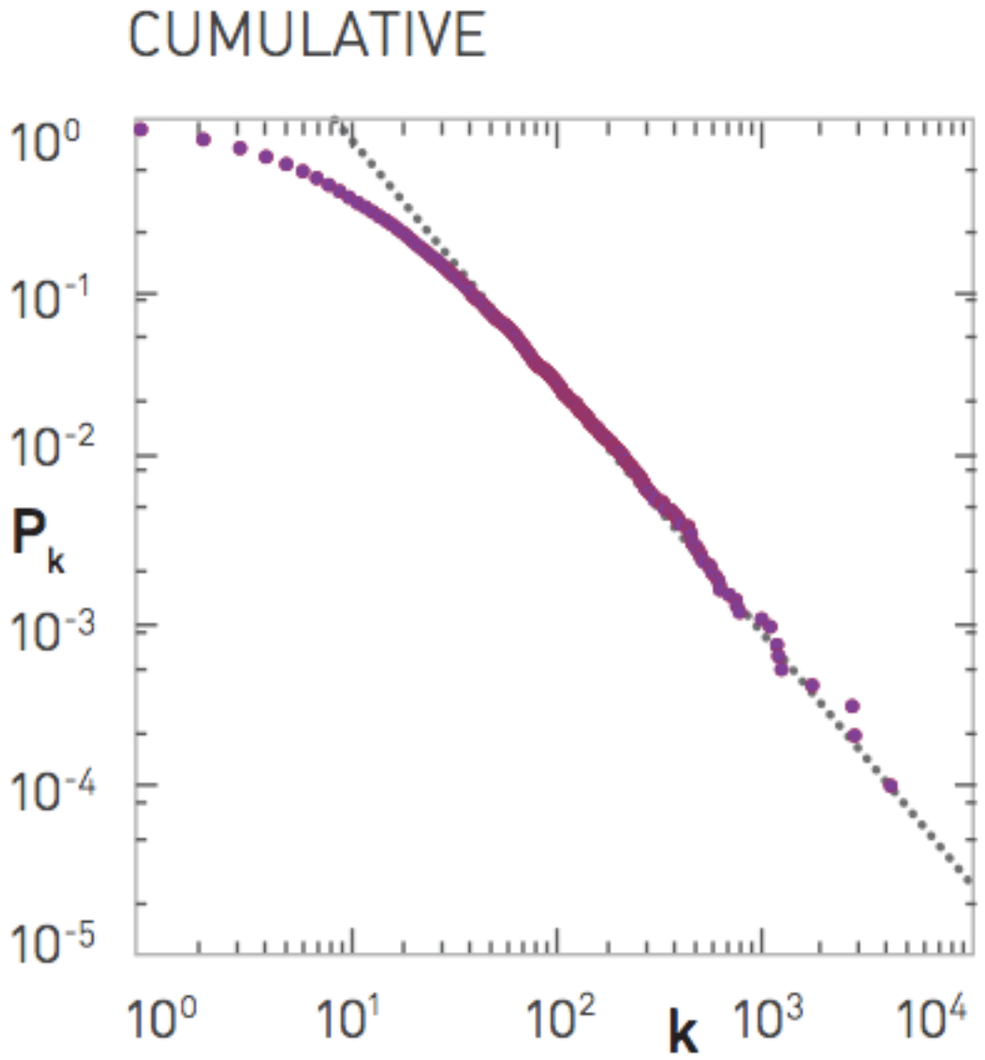
$$(k_1, k_2, \ldots, k_{N-1}, k_N)$$

3. Let's make it "pretty"

(Not noisy, easier to understand the trend of the tail, etc.)

Possibility 2:

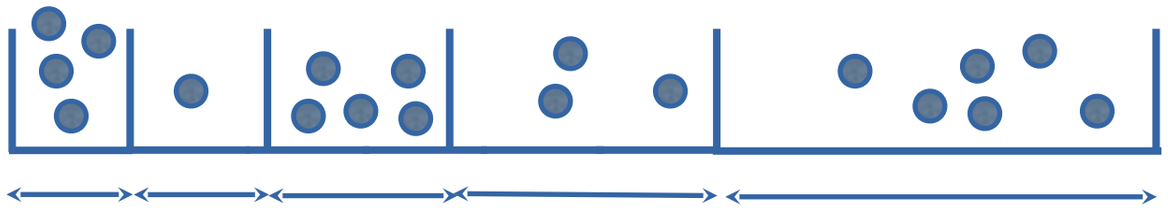
Cumulative distribution

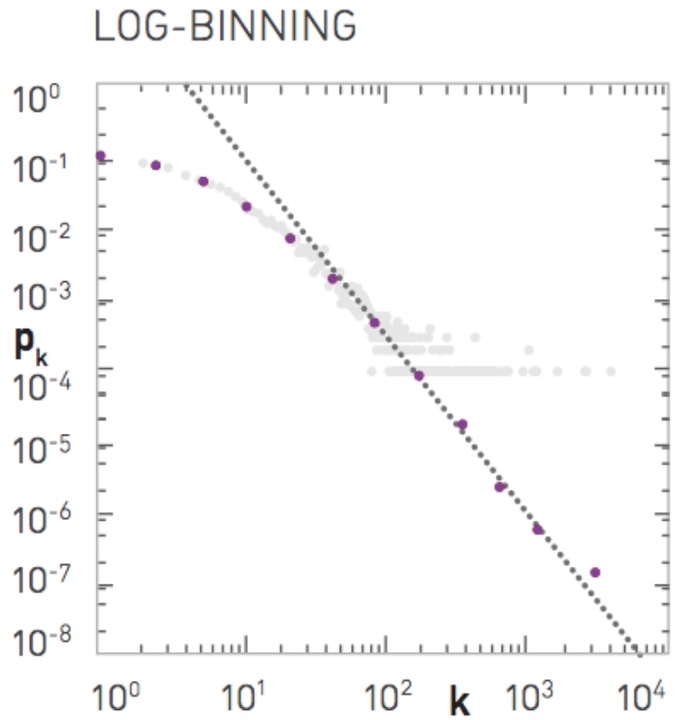


A.-L. Barabási. (2016). *Network Science,*

Power-law plotting

Logarithmic binning





Histogram where the size of the bin grows exponentially

Choose a minimum value $x_0 = k_{min}$

Create bins with edges $x_i = a x_{i-1}$

Do a histogram counting the number of points falling between these edges.
Divide by the size of the bin $x_i - x_{i-1}$

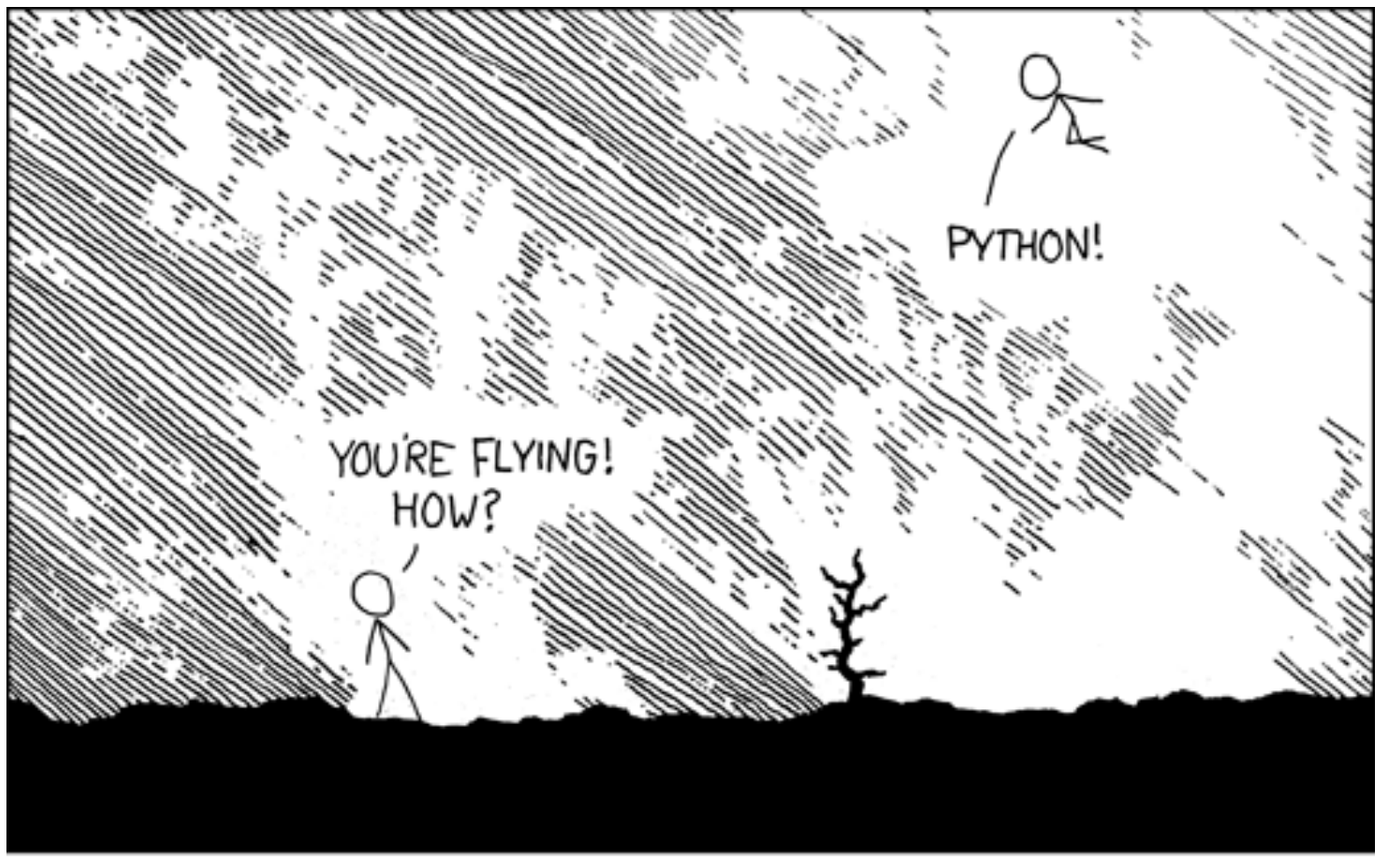
A.-L. Barabási. (2016). *Network Science,*

NetworkX: a Python module

(and related modules)

Inspired by the tutorial of Salvatore Scellato for the course “Social and Technological Network Analysis”, University of Cambridge (2011)

Why python?


PYTHON!
YOU'RE FLYING!
HOW?


I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Introduction to NetworkX - Python in one slide

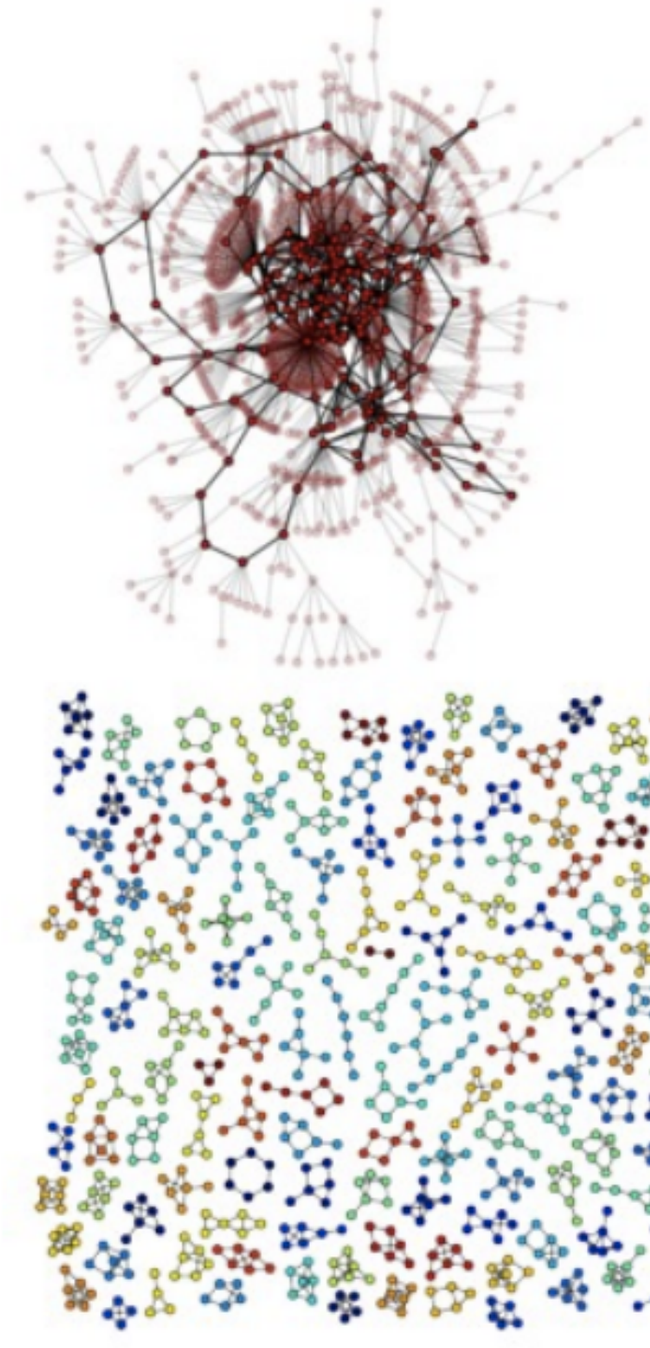
Python is an interpreted, general-purpose high-level programming language whose design philosophy emphasises code readability.

"there should be one (and preferably only one) obvious way to do it".

- Use of **indentation** for block delimiters (!!!!)
- **Multiple programming paradigms**: primarily object oriented and imperative but also functional programming style.
- **Dynamic type** system, automatic memory management, **late binding**.
- Primitive **types**: int, float, complex, string, bool.
- Primitive **structures**: list, tuple, dictionary, set.
- Complex **features**: generators, lambda functions, list comprehension, list slicing.

Introduction to NetworkX

"Python package for the creation, manipulation and study of the structure, dynamics and functions of complex networks."



- Data structures for representing many types of networks, or graphs
- Nodes can be any (hashable) Python object, edges can contain arbitrary data
- Flexibility ideal for representing networks found in many different fields
- Easy to install on multiple platforms
- Online up-to-date documentation
- First public release in April 2005

Introduction to NetworkX - design requirements

- **Tool** to study the structure and dynamics of social, biological, and infrastructure networks
- Ease-of-use and **rapid development**
- **Open-source** tool base that can easily grow in a multidisciplinary environment with non-expert users and developers
- An easy **interface to existing code** bases written in C, C++, and FORTRAN
- To painlessly slurp in relatively large **nonstandard data sets**

When should I AVOID NetworkX to perform network analysis?

- Large-scale problems that require faster approaches (i.e. massive networks with 100M/1B edges)
- Better use of memory/threads than Python (large objects, parallel computation)

NetworkX: online resources

https://networkx.org/documentation/stable/tutorial.html

Contact

Mailing list
Issue tracker
Source

Releases

Stable (notes)

2.6.2 — July 2021
download | doc | pdf

Latest (notes)

2.7 development
github | doc | pdf

Archive

NetworkX
Network Analysis in Python

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Software for complex networks

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source 3-clause BSD license
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

Power-law plotting

Choosing the optimal binning

Example 6.6. (Choice of the optimal binning) Suppose the bottom of the lowest bin is at $k_{\min}$ and the ratio of the widths of successive bins is a. Then the n-th bin extends from $x_{n-1} = k_{\min}a^{n-1}$ to $x_n = k_{\min}a^n$, and the expected number of samples falling in this interval is:

$$\int_{x_{n-1}}^{x_n} p(k)dk = c\int_{x_{n-1}}^{x_n} k^{-\gamma}dk$$
$$= c\,\frac{a^{\gamma-1}-1}{\gamma-1}(k_{\min}a^n)^{1-\gamma}.$$

Thus, so long as $\gamma > 1$ which, as shown in Table 6.1, is true for most of the degree distribution of real networks, the number of samples per bin goes down as n increases as $a^{(1-\gamma)n}$. Consequently, the bins in the tail will have more statistical noise than those that precede them.

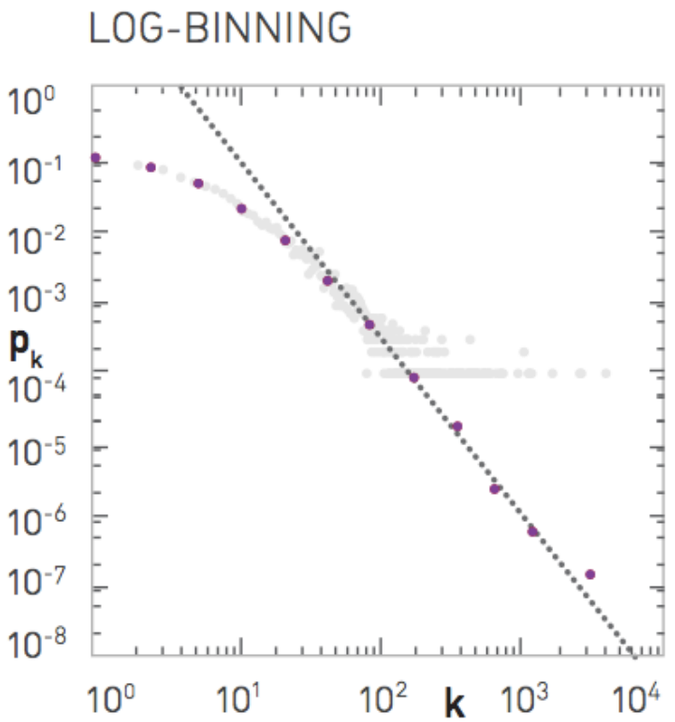
We can further reduce the fluctuations in the tails with a different binning in which the ratio of the width of successive bins is not necessarily constant. Such an *ideal binning* would indeed be obtained by imposing that the expected number of samples in each bin is *exactly* the same. Mathematically, this is equivalent to imposing that:

$$\int_{x_{n-1}}^{x_n} p(k)dk = \frac{1}{N_b}$$

where N_b is the number of bins. We thus obtain $(\frac{x_{n-1}}{k_{\min}})^{1-\gamma} - (\frac{x_n}{k_{\min}})^{1-\gamma} = \frac{1}{N_b}$ which gives:

$$x_n = k_{\min}(1 - \frac{n}{N_b})^{-\frac{1}{\gamma-1}} \qquad (6.24)$$

This choice is the one that minimizes the overall fluctuations for the same number of bins and samples. However, the problem with this choice is that the two sides of a bin are in general not integer numbers. □



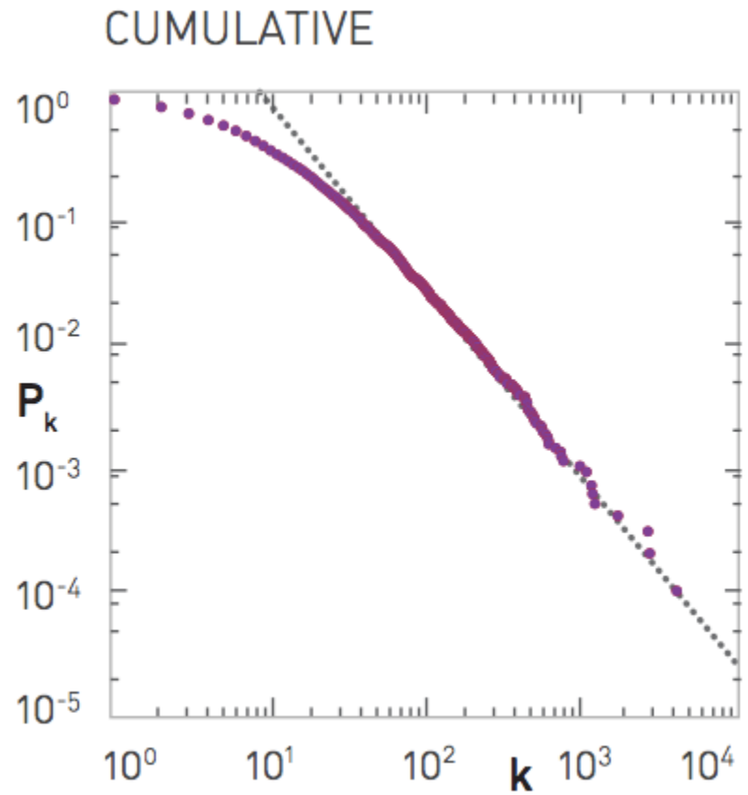
Power-law plotting

Cumulative distribution

Calculate the distribution

$$P_k = \sum_{q=k}^{\infty} p_q$$

(sum all the points with degree larger or equal k and divide by N)



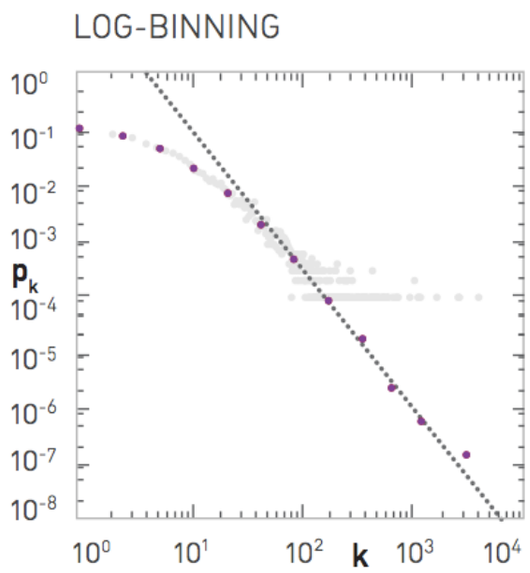
Plot the distribution in log-log scale (no binning required!)

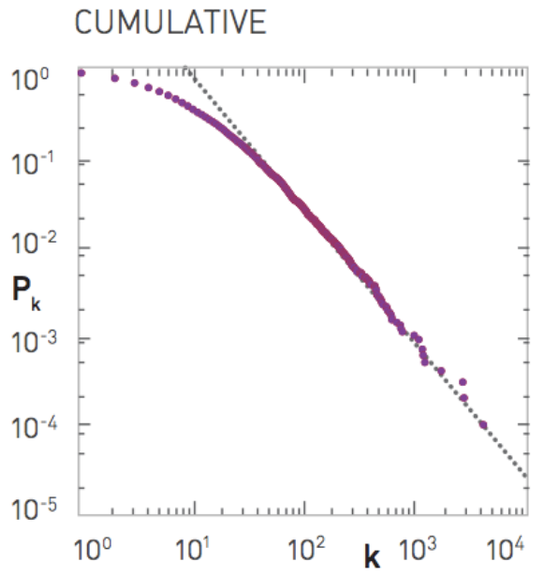
Remember that if $p_k \sim k^{-\gamma}$ then $P_k \sim k^{-\gamma+1}$

A.-L. Barabási. (2016). *Network Science,*

Power-law fitting

The wrong (but often used) way

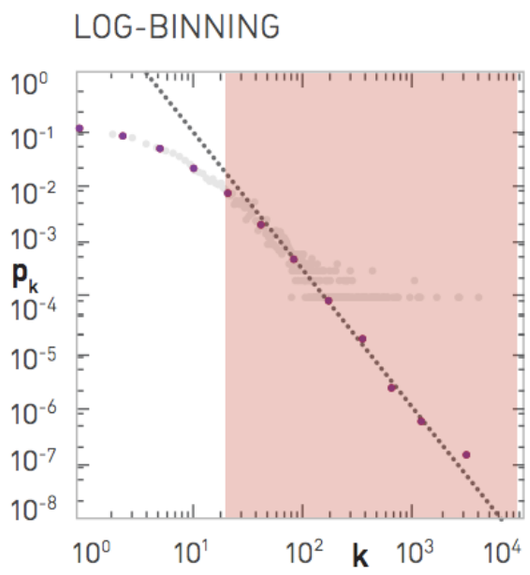


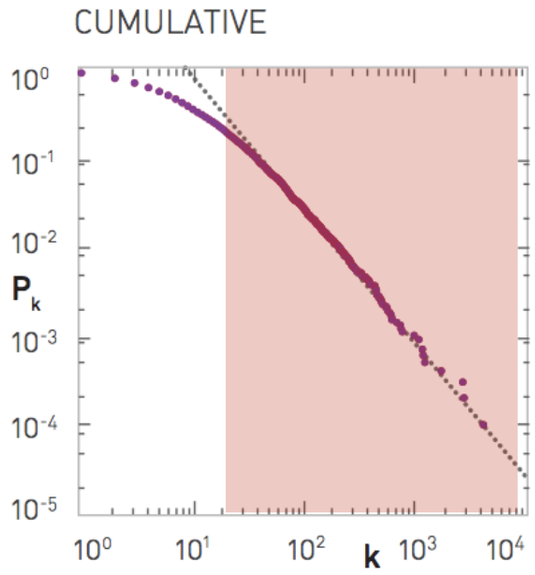


A.-L. Barabási. (2016). *Network Science,*

Power-law fitting

The wrong (but often used) way



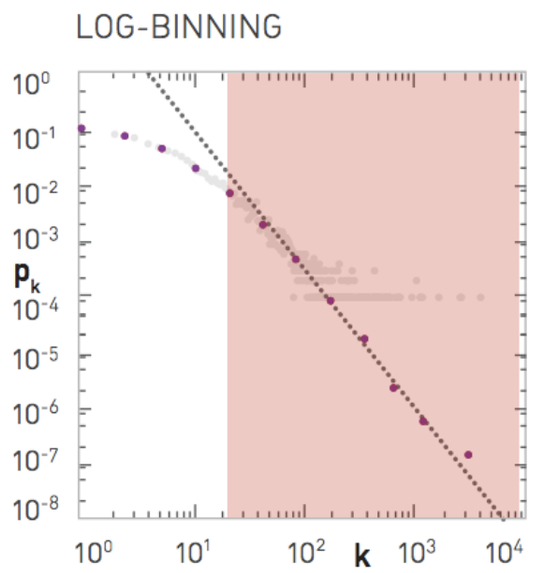


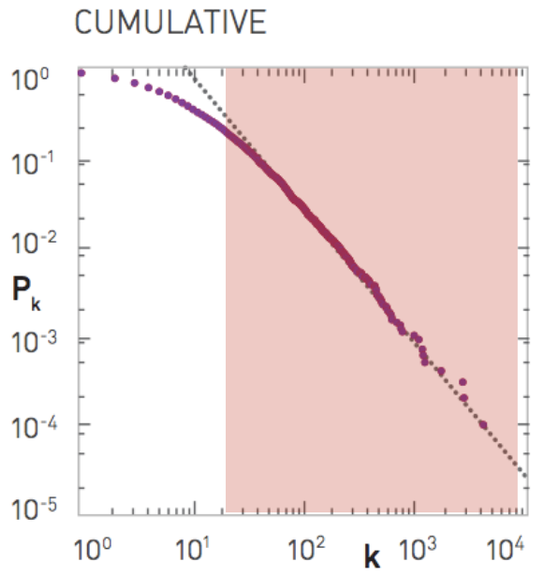
Fit with the least square method the portion of the plot that looks linear on a log-log scale

A.-L. Barabási. (2016). *Network Science,*

Power-law fitting

The wrong (but often used) way





- Fit with the least square method the portion of the plot that looks linear on a log-log scale
- Slope of the fit provides the exponent (log-binning) $\log p_k = -\gamma \log k + \text{constant}$
- Slope of the fit provides the exponent+1 (cumulative)

A.-L. Barabási. (2016). *Network Science,*

Power-law fitting

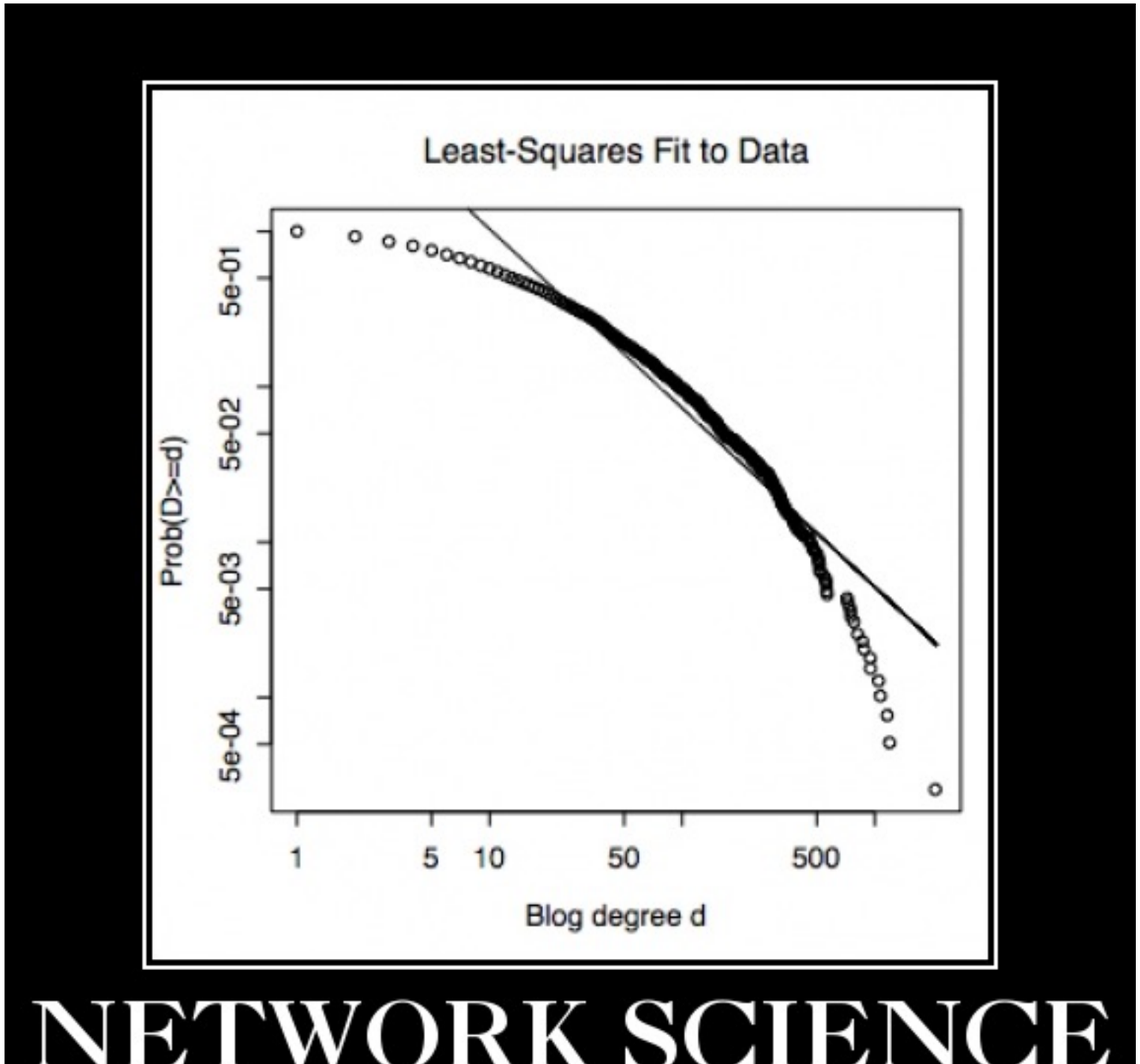
The wrong (but often used) way

Why is this so wrong?

- Probability distributions need to be normalized (probabilities must sum to 100%). Haphazardly fitting a line to data will almost never yield a valid distribution!
- Many, many functions appear linear when plotted in a log-log scale
- Discrepancies in the tail matter more than than elsewhere in any putative fit
- “Argument from eyeballing” is not sound science

Power-law fitting

How NOT to do it



Power-law fitting

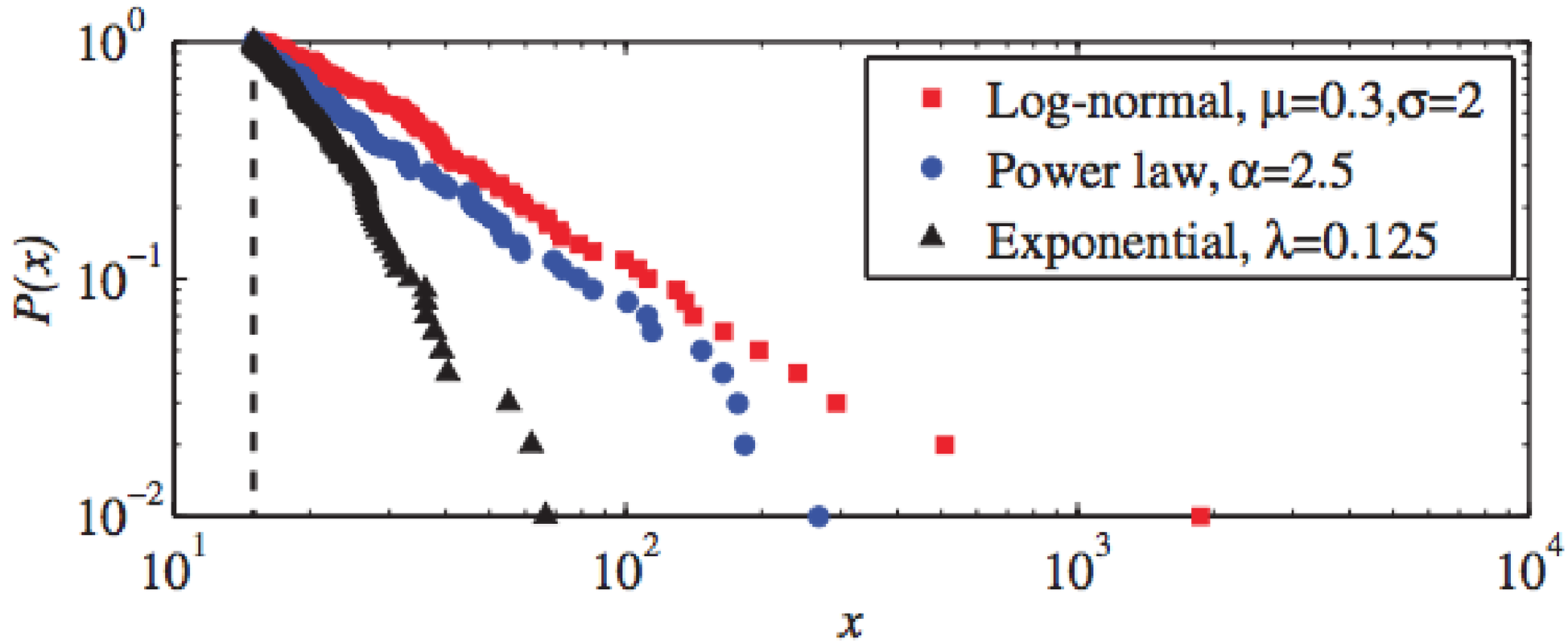
Problems with linear regression approach

1. “Eyeballing” is not sound science

Power-law fitting

Problems with linear regression approach

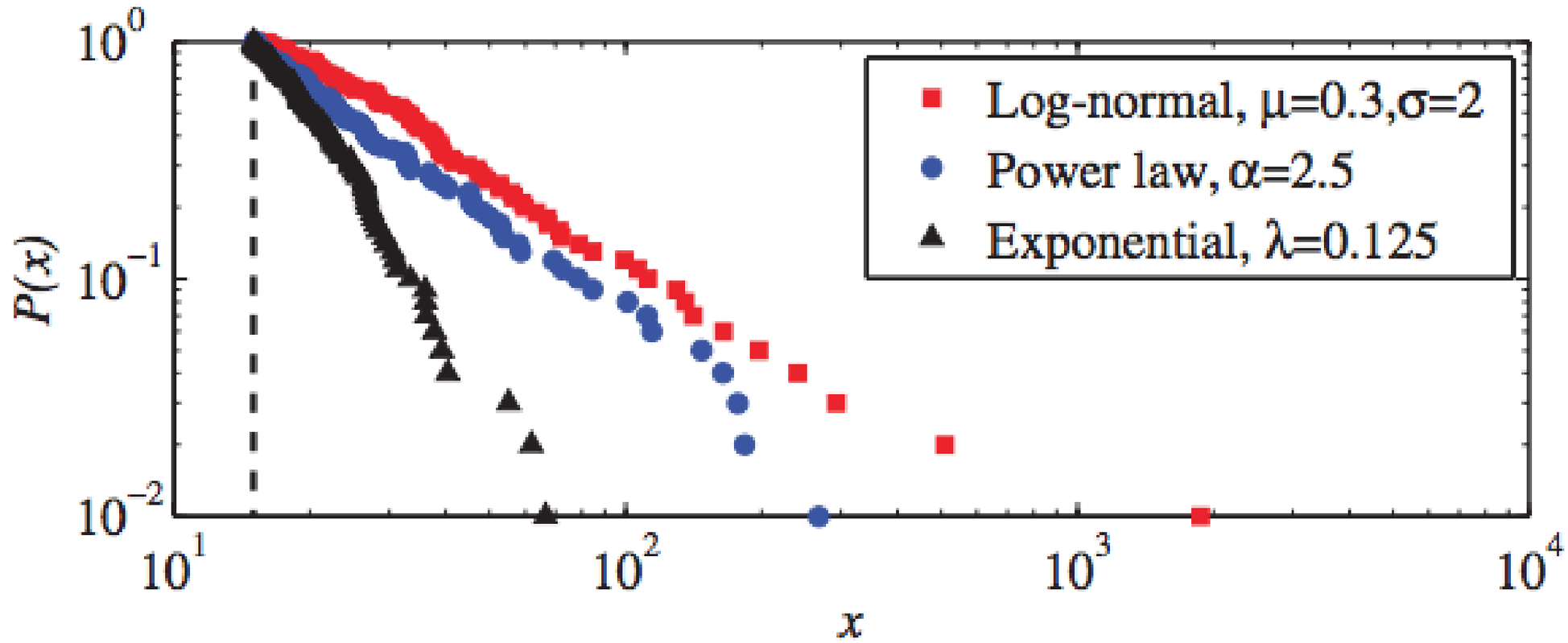
1. “Eyeballing” is not sound science
2. Many things look linear in log-log scale


Log-normal, μ=0.3,σ=2
Power law, α=2.5
Exponential, λ=0.125
P(x)
x
10^0
10^-1
10^-2
10^1
10^2
10^3
10^4

Power-law fitting

Problems with linear regression approach

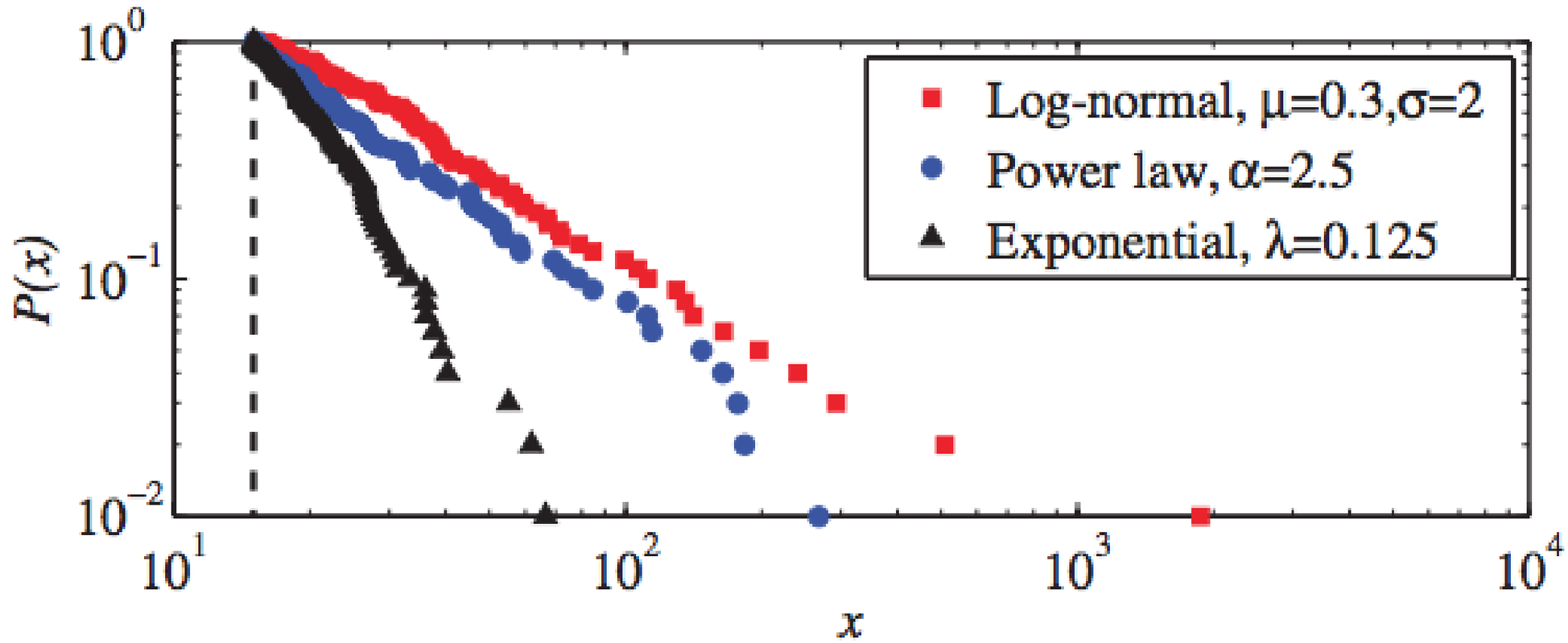
1. “Eyeballing” is not sound science
2. Many thing look linear in log-log scale
3. “Body” and “tail” should not be given equal weight


Log-normal, μ=0.3,σ=2
Power law, α=2.5
Exponential, λ=0.125
P(x)
x
10^0
10^-1
10^-2
10^1
10^2
10^3
10^4

Power-law fitting

Problems with linear regression approach

1. “Eyeballing” is not sound science
2. Many thing look linear in log-log scale
3. “Body” and “tail” should not be given equal weight
4. The result is almost surely not a probability distribution


Log-normal, μ=0.3,σ=2
Power law, α=2.5
Exponential, λ=0.125
P(x)
x
10^0
10^-1
10^-2
10^1
10^2
10^3
10^4

Power-law fitting

The right way (based on maximum- likelihood)

Maximum-likelihood estimation (**MLE**) is a method of estimating the parameters of a statistical model

Make a hypothesis for a model

$$p(k) = Ck^{-\gamma} \qquad C = (\gamma - 1) K_{min}^{\gamma - 1}$$

Choose the values that maximize the likelihood function.
For a power law this takes the form”

$$\mathcal{L}(k|\gamma) = \prod_{i=1}^{N} \frac{\gamma - 1}{K_{min}} \left(\frac{k_i}{K_{min}} \right)^{-\gamma}$$

Estimator of the exponent (derivative of the above w.r.t. gamma = 0)

$$\gamma = 1 + N \left[\sum_{i=1}^{N} \ln \frac{k_i}{K_{min} - \frac{1}{2}} \right]^{-1} .$$

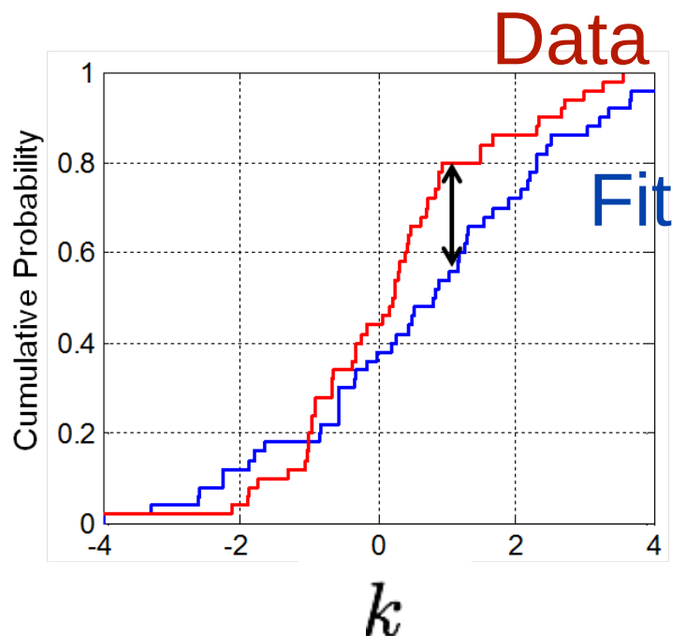
A.-L. Barabási. (2016). *Network Science,*

Power-law fitting

The right way (based on maximum- likelihood)

How to find K_{min}

Calculate the cumulative distribution of data and fitted distribution

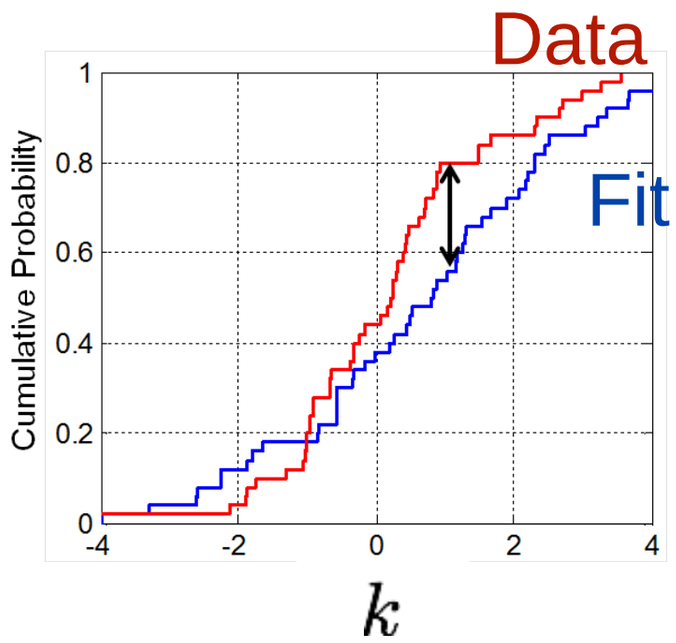


Power-law fitting

The right way (based on maximum- likelihood)

How to find K_{min}

Calculate the cumulative distribution of data and fitted distribution



Calculate the maximum distance between the two cumulative distributions

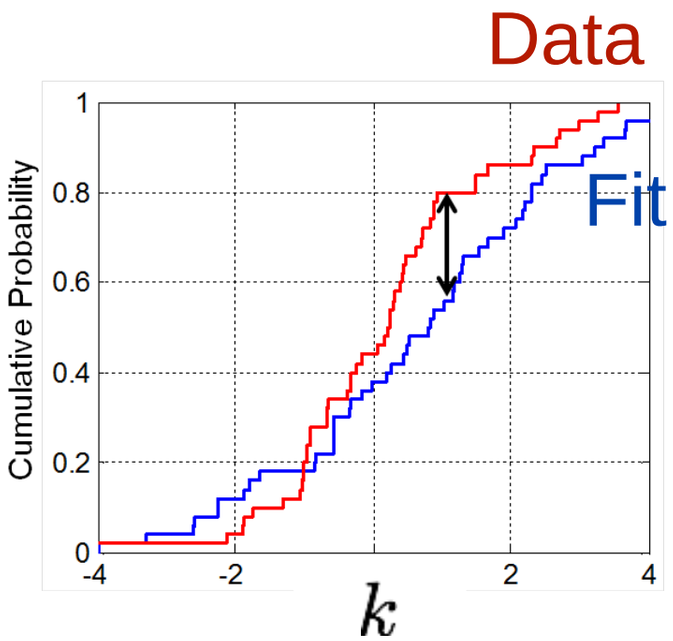
$$D = \max_{k \geq K_{min}} |S_k - P_k|$$

Power-law fitting

The right way (based on maximum- likelihood)

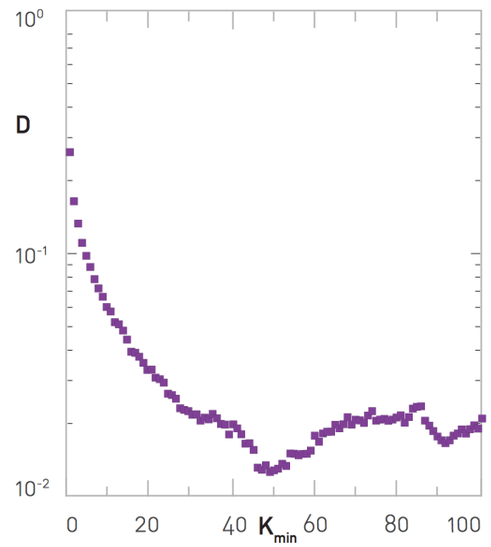
How to find K_{min}

Calculate the cumulative distribution of data and fitted distribution



Calculate the maximum distance between the two cumulative distributions

$$D = \max_{k \geq K_{min}} |S_k - P_k|$$



Take the K_{min} that minimizes D

There is code here: http://tuvalu.santafe.edu/~aaronc/powerlaws/

POWER-LAW DISTRIBUTIONS IN EMPIRICAL DATA

AARON CLAUSET*, COSMA ROHILLA SHALIZI†, AND M. E. J. NEWMAN‡

Abstract. Power-law distributions occur in many situations of scientific interest and have significant consequences for our understanding of natural and man-made phenomena. Unfortunately, the detection and characterization of power laws is complicated by the large fluctuations that occur in the tail of the distribution—the part of the distribution representing large but rare events—and by the difficulty of identifying the range over which power-law behavior holds. Commonly used methods for analyzing power-law data, such as least-squares fitting, can produce substantially inaccurate estimates of parameters for power-law distributions, and even in cases where such methods return accurate answers they are still unsatisfactory because they give no indication of whether the data obey a power law at all. Here we present a principled statistical framework for discerning and quantifying power-law behavior in empirical data. Our approach combines maximum-likelihood fitting methods with goodness-of-fit tests based on the Kolmogorov-Smirnov statistic and likelihood ratios. We evaluate the effectiveness of the approach with tests on synthetic data and give critical comparisons to previous approaches. We also apply the proposed methods to twenty-four real-world data sets from a range of different disciplines, each of which has been conjectured to follow a power-law distribution. In some cases we find these conjectures to be consistent with the data while in others the power law is ruled out.

Scale-Free Networks Well Done

Ivan Voitalov,[1,2] Pim van der Hoorn,[1,2] Remco van der Hofstad,[3] and Dmitri Krioukov[1,2,4,5]

[1] *Department of Physics, Northeastern University, Boston, Massachusetts 02115, USA*
[2] *Network Science Institute, Northeastern University, Boston, Massachusetts 02115, USA*
[3] *Department of Mathematics and Computer Science, Eindhoven University of Technology, Postbus 513, 5600 MB Eindhoven, Netherlands*
[4] *Department of Mathematics, Northeastern University, Boston, Massachusetts 02115, USA*
[5] *Department of Electrical & Computer Engineering, Northeastern University, Boston, Massachusetts 02115, USA*

We bring rigor to the vibrant activity of detecting power laws in empirical degree distributions in real-world networks. We first provide a rigorous definition of power-law distributions, equivalent to the definition of regularly varying distributions that are widely used in statistics and other fields. This definition allows the distribution to deviate from a pure power law arbitrarily but without affecting the power-law tail exponent. We then identify three estimators of these exponents that are proven to be statistically consistent—that is, converging to the true value of the exponent for any regularly varying distribution—and that satisfy some additional niceness requirements. In contrast to estimators that are currently popular in network science, the estimators considered here are based on fundamental results in extreme value theory, and so are the proofs of their consistency. Finally, we apply these estimators to a representative collection of synthetic and real-world data. According to their estimates, real-world scale-free networks are definitely not as rare as one would conclude based on the popular but unrealistic assumption that real-world data comes from power laws of pristine purity, void of noise and deviations.