Performance Evaluation: A Preparation for Statistics and Data Science ?

Jean-Yves Le Boudec
EPFL, Lausanne, Switzerland
https://leboudec.github.io/leboudec/
jean-yves.leboudec@epfl.ch

ABSTRACT

Lectures that use probability or statistics often appear complex to students, sometimes because the underlying stochastic models are not explicited. Writing a stochastic simulation program is a common exercise in a performance evaluation course, and we can view a simulation program as an implementation of a stochastic model. Thus, students who took such a course are trained to fully specify stochastic models, and this can help them fully understand probabilistic statements. We give examples in the context of random graphs as well as in the interpretation of residual scores in machine learning. Last we observe that Palm calculus, which is at the heart of classical queuing theory, can be used to provide important insights into sampling problems encountered in data collection tasks.

1. ON THE CORRECT USE OF PROBABILITIES

In a data science lecture, Alice, hears a statements such as

S Consider a undirected graph with N vertices where every pair of nodes is connected with probability q

(where N is a fixed parameter) and is then asked to

Q compute the distribution p of the degree (i.e. give the values of p_k for $k = 0, 1...N-1$).

What should Alice understand ? Statement S is about probabilities, and so is question Q, since it mentions a distribution. The issue with such formulations is that probabilities are used as if they were well defined physical quantities, whereas in reality, they depend on a *model*. For any statement involving probabilites to make sense, the model should be explicit (or obvious). Because Alice has previously taken a performance evaluation course, she has experience with writing simulation programs. She imagines a possible model for statement S:

M1 Consider an undirected graph that is obtained as follows. It has N vertices. For every pair of vertices, flip a biased coin that returns 1 with probability q and 0 with probability $1-q$. If a coin returns 1, draw an edge, else don't. The coin flips are independent.

But she also thinks that a different model could be:

M2 You are given an undirected graph with N vertices and e edges. Let $q = \frac{e}{\frac{N(N-1)}{2}}$, i.e. q is the proportion of pairs of nodes that are connected. Pick a first node uniformly at random among the N nodes (call it i) and pick a second node uniformly at random among the remaining $N-1$ nodes (call it j). The probability that there is an edge between i and j is q.

Both models are compatible with the statement S. The former produces a random graph; the latter produces a random pair of vertices in a given graph. The same ambiguity exists about question Q. With either model, Alice could assume that she is given a graph and picks a vertex uniformly at random among N; then she computes its degree. For every integer k, p_k is then equal to $\frac{1}{N}\times$ the number of vertices that have degree k. Note that the result depends on the specific graph that was initially given. For example, if Alice chooses model M1, the result is random, p is a random vector in the simplex and Alice concludes that she now has to compute the distribution of the random p.

She finds that this is too complicated, looks for the solution in a classical textbook [1] and obtains:

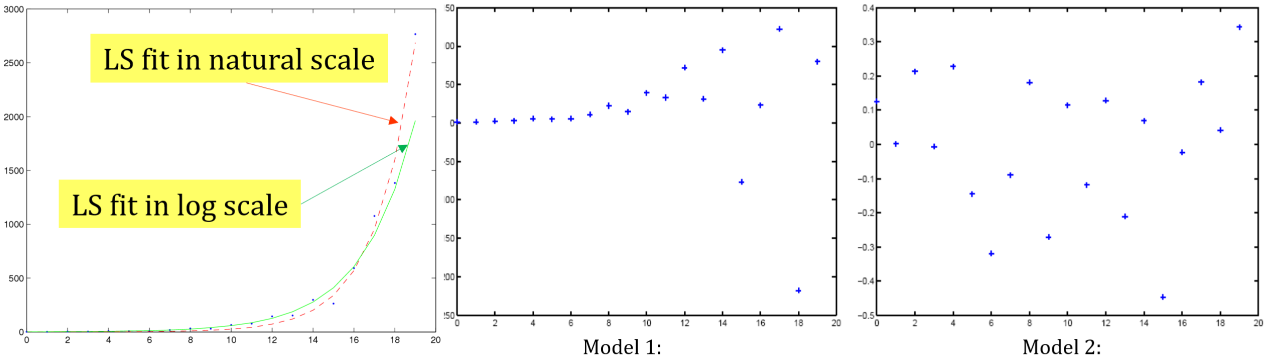
$$p_k = \begin{pmatrix} N-1 \\ k \end{pmatrix} q^k (1-q)^{N-1-k} \tag{1}$$

Alice concludes that what is expected from her is not the distribution of a distribution. She hypothesizes that the implicit model is:

M1Q Draw a graph according to M1. Draw a vertex uniformly at random. Call D its degree: it is random because both the graph and the vertex are random. Compute the distribution of the random variable D.

Later, Alice looks up the derivation of Equation (1) in [1] and obtains:

> *"In a random network the probability that node i has exactly k links is the product of three terms:*
>
> - *The probability that k of its links are present, or* q^k.
> - *The probability that the remaining* $(N-1-k)$ *links are missing, or* $(1-q)^{N-1-k}$
> - *The number of ways we can select k links from* $N-1$ *potential links a node can have, or* $\begin{pmatrix} N-1 \\ k \end{pmatrix}$
>
> *Consequently the degree distribution of a random network follows the binomial distribution".*



Model 1:
$Y_{t_i} = ae^{\alpha t_i} + \epsilon_i, \epsilon_i \sim iid\ N\left(0, \sigma^2\right)$

Model 2:
$\log Y_{t_i} = \log a + \alpha t_i + \epsilon_i, \epsilon_i \sim iid\ N\left(0, \sigma^2\right)$

Figure 1: Virus Expansion data, least square fit in log and natural scales (left); the corresponding residuals (center and left) show that the natural scale is not compatible with the underlying model. From [2].

Alice finds this “proof” magical and tries to make sense of it. She finds that it implicitly assumes a different model, which she formulates as:

M1Q′ Consider one vertex of interest in a set of N. For each of the other $N-1$ vertices, flip a biased coin that returns 1 with probability q and 0 with probability $1-q$. If a coin returns 1, draw an edge, else don’t. The coin flips are independent. Count the number of edges and call it D', the degree of the vertex of interest.

Equation (1) is justified in [1] using model M1Q′. It is perhaps obvious to the author of [1] that the outputs of M1Q and M1Q′ have the same distribution, but Alice is not convinced (she was trained to use the Coq proof assistant !), therefore, she imagines the following proof.

Alice’s proof Let $X_{i,j}$ be the decision taken by M1 for the link between vertices i, j, with $1 \leq i < j \leq N$. The random variables $X_{i,j}$ are independent and $\mathbb{P}(X_{i,j} = 1) = q$, $\mathbb{P}(X_{i,j} = 0) = 1-q$. Let I be the vertex chosen by M1Q, so that I is independent of all $X_{i,j}$ and $\mathbb{P}(I = i) = \frac{1}{N}$ for $i = 1, 2...N$. The output of M1Q is $D = \sum_{1\leq i<I} X_{i,I} + \sum_{I<i\leq N} X_{I,i}$.

Let Y_i be the decision taken by M1Q′ for the link between vertices 1 and i. The random variables Y_i are independent and $\mathbb{P}(Y_i = 1) = q$, $\mathbb{P}(Y_i = 0) = 1-q$. The output of M1Q′ is $D' = \sum_{2\leq i\leq N} Y_i$.

The conditional distribution of D given that $I = i_0$ is the distribution of $\sum_{1\leq i<i_0} X_{i,i_0} + \sum_{i_0<i\leq N} X_{i_0,i}$. It is equal to the distribution of D', because $X_{i,j}$ and $Y_{i'}$ have same distribution and are independent for all $i < j$ and all i', i.e. $\mathbb{P}(D = k|I = i_0) = \mathbb{P}(D' = k)$. Furthermore, it is the same for all i_0. By the formula of total probabilities:

$$\begin{aligned}
\mathbb{P}(D = k) &= \sum_{1\leq i_0\leq N} \mathbb{P}(D = k|I = i_0)\mathbb{P}(I = i_0) \\
&= \mathbb{P}(D' = k) \sum_{1\leq i_0\leq N} \mathbb{P}(I = i_0) \\
&= \mathbb{P}(D' = k)
\end{aligned}$$

i.e. D and D' have same distribution.

Alice now takes comfort in the belief that the model implicitly used in her data science class is M1Q, i.e. we pick a random graph and a random vertex. Furthermore, she has the satisfaction of understanding the proof of Equation (1). She continues reading [1] and, a few lines later, finds the statement:

> *“In a given realization of a random network some nodes gain numerous links, while others acquire only a few or no links (Image 3.3). These differences are captured by the degree distribution, p_k, which is the probability that a randomly chosen node has degree k”.*

Oh ! Alice is now puzzled and lost her comfort. The graph is now fixed and p_k is interpreted as the probability that a random node has degree k ! The author of [1] did not explicit his model and confuses students by simultaneously using different models. He should take a performance evaluation course !

In summary: if you mention probabilities (or distributions, or averages), make sure that you can, as a thought experiment, imagine a simulation program that would produce a sample of what you are computing probabilities for. If you don’t do this, the absence of an explicit model can make the computation of probabilities appear magical or doubtful; worse, it can lead to inconsistencies.

2. CLASSICAL STATISTICS AND SIMULATORS

The requirement to formulate a possible simulation program in order to give a meaning to statements involving probabilities naturally extends to statistics. A common formulation in classical statistics is that the data at hand can be explained by a stochastic model that depends on a fixed but unknown parameter θ. In the language of performance evaluation: the data at hand was produced by a simulator, the code of which is known but depends on some unknown parameter θ.

Consider for example the computer-virus expansion data in Fig. 1, where Y_t is the number of infected hosts at time t and the problem is to predict the number of future infected hosts, by training the prediction system over past data. A

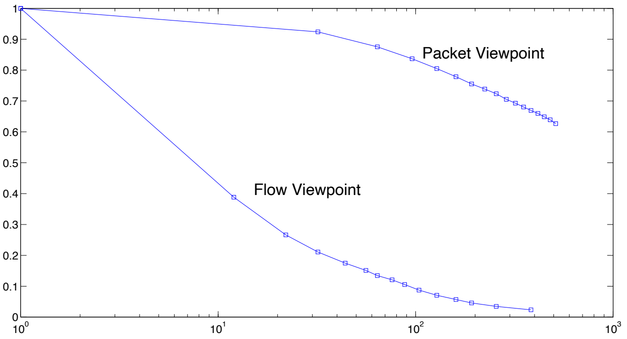


Figure 2: Proportion of flows that have a size larger than the x value, in bytes [5]. Figure from [2].

common machine learning answer to this question consists in minimizing the score equal for example to $\sum_t (e_t(\theta)^2)$ where $e_t(\theta)$ is the error term when the parameter is θ and t is in the training set. This least square score is very common, but is sometimes replaced by variants that use weights, or that replace the euclidian distance by absolute deviation (i.e. the score is $\sum_t |e_t(\theta)|$). Also, the scale in which the error terms e_t are measured (e.g. log versus natural) may be subject to discussion.

Viewing the data as produced by a simulator provides an interesting angle of attack to address such issues. For example, here we could assume that the data is produced by the simulator $Y_t = f(t, \theta) + \varepsilon_t$ where the sequence ε_t is identically distributed gaussian noise, f is a known function and θ is the parameter to be learnt. It is well known that this leads to least squares, i.e. to the score function $\sum_t (e_t(\theta))^2$ with $e_t(\theta) = Y_t - f(t, \theta)$. An alternative simulator is $\log(Y_t) = \log(f(t, \theta)) + \varepsilon_t$, with the same assumption on ε_t; this leads to the score $\sum_t (e_t(\theta)^2)$ with $e_t(\theta) = \log(Y_t) - \log(f(t, \theta))$. Yet a different simulator is obtained if we assume that ε_t is identically distributed Laplace noise instead of gaussian; this leads to minimum absolute deviation instead of least-square [2]. Deciding which simulator is a better model of the data is a simpler task than deciding which scores provides better learning, as it can be analyzed by comparing the residuals to the distribution assumed by the simulator (see Fig. 1 for an example).

3. PALM, PASTA AND THE IMPORTANCE OF THE VIEWPOINT

According to massive data collected by the local railway company, less than 5% of all trains arrive late. The local consumer association uses a crowdsourcing app and finds that close to 30% of consumers complain about late trains. Who says the truth ? Assuming that both data collections are true and have the same criteria for late trains, these two numbers might not be incompatible, as they depend on different *sampling methods*.

In queuing theory, it is frequent to compare distributions seen by an arriving customer and seen at an arbitrary point in time. This *change of viewpoint* underlies the derivation of the Pollaczek-Khinchine formula for the M/G/1 queue and is present in the celebrated Little's law $N = \lambda R$, where N (average numbers in system) is a time-average, R (average response time) is seen by an arriving customer and λ is the rate of arrival or departure of customers. Such results are instances of Palm calculus, which makes links between performance metrics obtained with different viewpoints. PASTA (Poisson Arrivals See Time Averages, more precisely: Poisson arrivals independent of system state see time averages) is an example where both viewpoints coincide and can be used to obtain the time average viewpoint by Poisson sampling [3]. The classical presentation of Palm calculus uses the formalism of abstract probability spaces [4], but a more gentle presentation that uses the language of simulators is also possible [2].

Palm calculus can be applied to a large variety of settings, well beyond queuing theory. Consider for example the measurements of internet flow sizes reported in [5]: $\approx$ 10% of flows have a size larger than 200 bytes, whereas $\approx$ 85% of packets are in such flows. This is a typical example of difference in sampling method: both curves in Figure 2 give the distributions of flow sizes, the top curve samples per packet whereas the bottom one samples per flow. Palm's inversion formula reconciles the two viewpoints and gives $f_P(s) = \eta s f_F(s)$ where $f_F()$ [resp. $f_P()$] is the probability density function of flow sizes per flow [resp. per packet] and η^{-1} is the average flow size [2].

Even simpler, some Palm calculus formulas can be derived empirically, by assuming the data is obtained from a simulation, and by comparing the codes that compute different metrics. Consider for example Little's formula and assume you run a discrete-event simulation in order to evaluate N, the average number of customers in the system of interest; it can be defined as $\frac{1}{T} \int_0^T C(t) dt$ where $C(t)$ is the number of customers present at simulated time t and T is the finish time of the simulation. One way to estimate N is to maintain a counter `backlogCtr` that is initially 0 and accumulates the value of $\int_0^{t \text{now}} C(t) dt$; whenever an event (arrival, departure) occurs, this counter is incremented by $C^- \Delta t$ where C^- is the value of the counter C just before the event occurs and Δt is the time elapsed since the previous occurrence of an event. At the end of the simulation, an estimate of N is $\bar{N} = \frac{1}{T}$`backlogCtr`. Similarly, the average response time R is estimated by $\bar{R} = \frac{1}{M} \sum_{n=1}^{M} R_m$ where R_m is the response time of the mth customer and M is the total number of customer served in the simulation. The value of the sum in this formula can be obtained by maintaining a counter `respTimeCtr` that is initially 0 and is incremented at every event. The increment is the integral of all increments in response time of all customers present in the system between now and the previous event; there are C^- such customers and their response time increment is Δt, i.e. `respTimeCtr` is incremented by $C^-\Delta t$, same as `backlogCtr`. Thus these two counters are equal throughout the simulation. At the end of the simulation, R is estimated by $\bar{R} = \frac{1}{T}\texttt{respTimeCtr} = \frac{1}{T}\texttt{backlogCtr}$. It follows that $\bar{N} = \frac{M}{T}\bar{R}$. Observe that the rate of arrival can be estimated by $\bar{\lambda} = \frac{M}{T}$ and we obtain Little's formula $\bar{N} = \bar{\lambda}\bar{R}$.

The same method can be applied to the conflicting measures of late trains. Imagine a simulation with N train arrival events and let $D_n = 1$ if the nth arrival event is late, $D_n = 0$ if it is on time. The railway company's estimate is $\bar{D} = \frac{1}{N}\sum_{n=1}^{N} D_n$. Let P_n be the number of passengers leaving the train at the nth arrival event. The consumer app estimates $D^* = \frac{\sum_{n=1}^{N} P_n D_n}{\sum_{n=1}^{N} P_n}$. The average number of passengers per train arrival event is $\bar{P} = \frac{1}{N}\sum_{n=1}^{N} P_n$ and the average number of passengers per late train arrival event is $\bar{P}_{\text{late}} = \frac{\sum_{n=1}^{N} P_n D_n}{\sum_{n=1}^{N} D_n}$. Thus $D^* = \bar{D}\frac{\bar{P}_{\text{late}}}{\bar{P}}$. If, in average, there are 6 times more passengers in late trains than in all trains, then the two estimations are compatible !

4. CONCLUSION

Probability and statistics are branches of science that may appear complex to students, as they involve an in-depth understanding of what probability really means. After following a performance evaluation course, many students find that they are well equipped to fully grasp the meaning of probability and statistics as used in data science courses. This is perhaps because performance evaluation courses do exercise the theory of probability in powerful ways, but certainly also because viewing data as the output of a simulator can provide useful insights !

5. REFERENCES

[1] Albert-László Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987):20120375, 2013.

[2] Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems.* EPFL Press, Lausanne, Switzerland, also available online at https://leboudec.github.io/perfeval/, 2010.

[3] Nicolas Hohn and Darryl Veitch. Inverting sampled traffic. *IEEE/ACM Transactions on Networking*, 14(1):68–80, 2006.

[4] Pierre Brémaud. *Point Process Calculus in Time and Space: An Introduction with Applications*, volume 98. Springer Nature, 2020.

[5] A. Shaikh, J. Rexford, and K.G. Shin. Load-sensitive routing of long-lived IP flows. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 215–226. ACM New York, NY, USA, 1999.