Estimació dels paràmetres de la distribució Weibull

Arnau Pérez Reverte, Pau Soler Valadés

04-12-2025

Simulació, MESIO UPC-UB

1. Introducció

Aquest document és la memòria de l'entrega avaluable de la primera part de l'assignatura *Simulació* del màster MESIO, feta per Arnau Pérez Reverte i Pau Soler Valadés. El document segueix el procediment ADMEP (Aims, Data-Generation, Methods, Estimands, Performance) [TODO CITAR AIXÔ ARNAU] i contesta els punts proposats a l'enunciat de l'entrega. A diferencia d'un artícle acadèmic, en certs punts de la pràctica ens dediquem a explicar conceptes que s'haurien de donar per sabuts, però degut a la naturalesa avaluada de l'entrega, hem preferit explicar en el benentès de la màxima claredat.

Sobre l'ús d'Intel·ligència Artificial Generativa: Els continguts d'aquesta memòria han estat íntegrament escrits per humans, els seus dos autors, així com tota la seva estructura i raonaments. Tanmateix, s'ha emprat la Intel·ligència Artifical Generativa (models PerplexiyAI [TODO CITAR ARNAU] i Google Gemini 3 [TODO CITAR ARNAU]) en la recerca de fonts, explicacions sobre conceptes i pel codi de les taules que apareixen en aquest document.

2. Objectiu (Aims)

L'objectiu principal del següent estudi de simulació és l'estudi de la inferència dels paràmetres d'una distribució de Weibull mitjançant el mètode de Regressió per Rangs Medians (Median Ranks Regression o MRR). Amb aquesta finalitat, es desenvoluparà la teoria al voltant d'aquesta distribució de probabilitat, es definirà aquest procediment d'inferència en detall i s'avaluarà la qualitat del seus resultats mitjançant una sèrie de mètriques, així com es compararà amb el mètode ordinari de inferència per estimadors de màxima versemblança (Maximum Likelihood Estimation o MLE).

3. Mecanismes de Generació de Dades (Data-Generating Mechanisms)

El DGM defineix com s'utilitza el mostreig pseudoaleatori per crear dades. Aquest estudi contindrà exclusivament simulacions paramètriques de dades distribuïdes seguint una Weibull per a diferents configuracions dels paràmetres d'escala α i de forma β per tal de verificar que l'estimació s'adapta a la forma de la distribució, així com amb diferents mides de tamany mostral.

El mecanisme de generació de dades consisteix en, per a cada tamany mostral diferent n, generar una mostra $T_1, ..., T_n$ *i.i.d*, on $T_i \sim \text{Weibull}(\alpha, \beta)$ per a totes les 9 parelles de valors possibles (α, β). La taula Taula 1 mostra els valors escollits per a les simulacions.

Factor	**Tipus**	**Valor**		
		Small	***Medium***	***Large***
α (Scale)	Paràmetre de distribució	2	10	50
β (Shape)	Paràmetre de distribució	0.8	1	3
n (Mida de la mostra)	Paràmetre de mostreig	10	50	200

Taula 1: Matriu de factors a variar a l'estudi

Els valors proposats pels tres nivells de la mida de la mostra n són valors estàndards en la literatura [TODO: EL PAU TÉ ELS ARTÍCLES]. En el cas dels paràmetres α, β, primer es definiran amb detall a continuació i aquesta proposta de valors quedarà aleshores justificada.

3.1. La distribució de Weibull

Diem que una variable aleatòria contínua X segueix una distribució de Weibull de paràmetres α i β, $X \sim \text{Weibull}(\alpha, \beta)$, si té funció de densitat de probabilitat (pdf)

$$f_X(x; \alpha, \beta) = \left(\frac{\beta}{\alpha}\right)\left(\frac{x}{\alpha}\right)^{\beta-1} \exp\left(-\left(\frac{x}{\alpha}\right)^{\beta}\right) \mathbb{1}_{\{x>0\}},$$

on $\alpha > 0$ i $\beta > 0$ són els paràmetres d'escala (scale) i forma (shape) respectivament. S'observa aleshores que la seva funció de distribució de probabilitat (cdf) pren la forma

$$F_X(x) = 1 - \exp\left(-\left(\frac{x}{\alpha}\right)^{\beta}\right).$$

La distribució de Weibull, amb els seu suport als nombres reals positius $\mathbb{R}^+$, és molt útil per modelitzar el concepte de "temps de vida" en contextos com control de qualitat, epidemiologia, o el cas de l'exemple de l'enunciat, temps fins a fallada en enginyeria, entre molts altres. Els seus paràmetres es poden, a més a més, caracteritzar i interpretar de la manera següent:

- α (Scale): També conegut com vida característica, pren unitats d'acord al context del problema, és a dir, ja sigui perque estem modelitzant segons, minuts, hores, cicles, etc.
- β (Shape): En el nostre context, β és un paràmetre adimensional conegut com la proporció de fallada, i que modifica el comportament de la distrbució de Weibull de la següent forma en funció del tres casos:
  1. $\boldsymbol{\beta < 1}$**:** Proporció de fallada decreixent/"mortalitat infantil". Com major és el temps de supervivència de la unitat, menor és la probabilitat de que mori/falli en el següent instant.
  2. $\boldsymbol{\beta = 1}$**:** Proporció de fallada constant. Simplifica la distribució de Weibull a una $\text{Exp}\left(\frac{1}{\alpha}\right)$, simbolitzant que el temps de supervivència és aleatori i independent de temps transcorregut.
  3. $\boldsymbol{\beta > 1}$**:** Proporció de fallada creixent. La probabilitat de mort creix a mesura que el temps incrementa. Representa unitats/individus que empitjoren amb el pas del temps.

[TODO: POTSER POSAR UNES GRAFIQUITES SOBRE BETA?]

Com ja s'ha descrit, la distribució de Weibull s'utilitza en el contextos on les dades són temporals, i en conseqüència manté una relació molt estreta amb altres distribucions utilitzades amb aquesta mateixa finalitat. Es destaquen en particular les següents relacions:

- $\text{Weibull}(\alpha, 1) = \text{Exp}\left(\frac{1}{\alpha}\right)$
- $X \sim \text{Weibull}\left(\sqrt{2}\beta, 2\right) = \text{Rayleigh}(\beta)$

3.2. Mètode de generació de nombres aleatòries

Per tal de simular valors de la distribució de Weibull, s'ha emprat el mètode de la distribució inversa; el mètode empra la composició de dues funcions: la funció de densitat d'una distribució uniforme $U(0,1)$ juntament amb la inversa de la funció de probabilitat F de la distribució que es vulgui generar. Una variable aleatòria $U \sim U(0,1)$ té la densitat f_U següent:

$$f_U(x) = \mathbb{1}_{\{0 \leq x \leq 1\}}(x)$$

i està definida en $f_U : \mathbb{R} \mapsto [0,1]$, i una *cdf* seguint una variable aleatòria X està definida a $F_X : \mathbb{R} \mapsto [0,1]$, i la seva inversa per tant a $F_X^{-1} : [0,1] \mapsto \mathbb{R}$. Aleshores, la seva composició

$$F_X^{-1} \circ f_U \sim X$$

$$F_X^{-1} \circ f_U : \mathbb{R} \overset{f_U}{\longmapsto} [0,1] \overset{F_X^{-1}}{\longmapsto} \mathbb{R}$$

no només és possible, sinó que és una CDF de la variable X. Això és possible ja que U genera els nombres uniformement entre l'interval $[0,1]$, fent que la imatge de la composició es comporti com la distribució de la variable aleatòria que es vol generar.

En el cas de la distribució de Weibull, és fàcil veure que la funció inversa de la CDF està ben definida per $u \in [0,1]$ i té la forma:

$$F_X^{-1}(u) = \alpha(-\log(u))^{\frac{1}{\beta}}.$$

3.3. Implementació computacional

La implementació de les simulacions es realitza en el marc del llenguatge de programació Python utilitzant principalment les llibreries de computació numèrica NumPy [1] i SciPy [2] , les quals proporcionen una API d'alt nivell fàcil d'utilitzar, amb implementacions numèriques eficients testejades per la comunitat. Els objectes numèrics que s'utilitzaran són principalment del tipus `np.array`, permetent operacions vectoritzades que acceleren la realització de simulacions.

Un dels focus d'aquest treball i especialment de l'assignatura és la garantia de reproducibilitat d'un estudi de simulació, és a dir, desenvolupar el codi de la simulació de tal forma que qualsevol usuari interessat en reproduïr (de forma exacta) els fets presentats en aquesta memòria ho pugui fer sense cap inconvenient. Amb aquest fi, el codi annex a la memòria s'estructura fonamentalment en diferents blocs de codi.

En primer lloc, el mòdul `src/dgm.py` conté la classe `WeibullDistribution` la qual inicialitzada amb els paràmetres `alpha` i `beta` adequats encapsula una Weibull(α, β) de la qual es pot generar una mostra de mida n mitjançant el mètode `sample(n: int)`, que implementa el mètode descrit a l'apartat anterior. La generació de nombres aleatoris ben definida la proporciona NumPy amb `np.random.default_rng(seed)`, que per darrera implementa l'algoritme de Mersenne-Twister proporcionada una llavor (`seed`). S'observa que, creat un objecte generador de nombres aleatoris de NumPy `rng`, la classe `WeibullDistribution` permet ser inicialitzada amb aquest generador especificament, de forma que si es canvia de paràmetres durant l'estudi de simulació, el fet de compartir el generador garanteix el control total sobre la seqüència de nombres que es van generant a cada punt del programa.

4. Mètodes

Aquest apartat descriu teòricament el mètodes avaluats en aquest estudi, que són estimació dels paràmtres per MLE, per MRR i l'aproximació de Betrand de l'MRR.

4.1. Estimadors de Màxima Versemblança (MLE)

L'estimació mitjançant MLE és el mètode tradicional per l'estimació de paràmetres. Consiteix a trobar l'estimador de màxima versemblança, que no és més que

$$\hat{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\theta\in\Theta} L(\theta|\boldsymbol{x})$$

On $L(\theta|\boldsymbol{x})$ és la funció de versemblança associada a la mostra $\boldsymbol{x}$. Pel cas de la Weibull, aquesta pren la forma següent:

$$L(\alpha,\beta|\boldsymbol{x}) = \prod_{i=1}^{n} f(x_i|\alpha,\beta) = \prod_{i=1}^{n}\left[\frac{\alpha}{\beta}\left(\frac{x_i}{\beta}\right)^{\alpha-1} e^{-\left(\frac{x_i}{\beta}\right)^{\alpha}}\right]$$

Com que ln és una funció monòtona creixent, és equivalent minitmizar $\ell(\alpha,\beta) = \ln L(\alpha,\beta)$ sent els càlculs molt més senzills:

$$\begin{aligned}\ell(\alpha,\beta) &= \sum_{i=1}^{n}\left[\ln\alpha - \ln\beta + (\alpha-1)(\ln x_i - \ln\beta) - \left(\frac{x_i}{\beta}\right)^{\alpha}\right] \\ &= n\ln\alpha - n\ln\beta + (\alpha-1)\sum_{i=1}^{n}\ln x_i - n(\alpha-1)\ln\beta - \sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{\alpha} \\ &= n\ln\alpha - n\alpha\ln\beta + (\alpha-1)\sum_{i=1}^{n}\ln x_i - \beta^{-\alpha}\sum_{i=1}^{n} x_i^{\alpha}\end{aligned}$$

Ara hem d'obtenir les derivades $\frac{\partial\ell}{\partial\beta}$ $\frac{\partial\ell}{\partial\alpha}$ per a trobar el màxim de la funció. Aquest càlcul és feixuc i no contribueix als resultats que es volen ensenyar, així que es deixa a l'Annex 1. Els valors de les expressions són:

$$\frac{\partial\ell}{\partial\alpha} = \frac{1}{\hat{\alpha}} + \frac{1}{n}\sum_{i=1}^{n}\ln x_i - \frac{\sum_{i=1}^{n} x_i^{\hat{\alpha}}\ln x_i}{\sum_{i=1}^{n} x_i^{\hat{\alpha}}} = 0$$

$$\frac{\partial\ell}{\partial\beta} = \hat{\beta} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^{\hat{\alpha}}\right)^{\frac{1}{\hat{\alpha}}}$$

Per tant, trobant el mínim del paràmetre α trobem el de β. El paràmetre s'acostuma a trobar computacionalment amb mètode de Newton-Rhapson. En el cas del nostre codi, emprem la llibreria `scipy` per a obtenir els estimadors mitjançant MLE.

4.2. Regressió per Rangs a la Mediana (MRR)

El mètode de Median Ranks Regression (MRR) per a l'estimació dels paràmetres consisteix en tres passos:

1. Donada la mostra d'observacions de temps de fallada $T_1, ..., T_n$, primer s'ordena la mostra en ordre creixent, obtenint així la mostra $\hat{T}_1, ..., \hat{T}_n$ on
$$\hat{T}_i := T_{(i)}.$$
Es guarda també la permutació $\sigma : \{1, .., n\} \longrightarrow \{1, .., n\}$ tal que $\sigma(T_1, ..., T_n) = \left(\hat{T}_1, ..., \hat{T}_n\right)$.
2. Es calcula el rang medià MR_j de cada fallada $\hat{T}_j$ de la mostra mitjançant l'equació
$$0.50 = \sum_{k=j}^{N}\binom{N}{k}\mathrm{MR}_j^k\left(1-\mathrm{MR}_j\right)^{N-k}$$
i s'utilitza com a estimació de la no-fiabilitat: $Q\left(\hat{T}_j\right) \approx \mathrm{MR}_j$.

   Aquesta equació es pot resoldre en primera instància utilitzant algorismes de cerca d'arrels com el mètode de Newton. No obstant, es pot veure a [3, p. 15] que el costat de la

dreta coincideix amb el valor $I_{\mathrm{MR}_j}(j, n-j+1)$, que es correspon amb l'anomenada funció beta incompleta regularitzada, definida com

$$I_x(a,b) = \frac{B(x;a,b)}{B(a,b)},$$

on B és la funció Beta incompleta

$$B(x;a,b) = \int_0^x t^{a-1}(1-t)^{b-1}dt$$

i posem $B(a,b) \coloneqq B(1;a,b)$. Per tant,

$$0.50 = I_{\mathrm{MR}_j}(j, n-j+1)$$

i el problema es simplifica a treballar amb la inversa (sobre x) de $I_x(a,b)$, la qual es troba implementada en paquets de programari estadístic [2].

És fàcil observar a més a més que aquest càlcul del MR_j depèn només de n i de j, i no del valor observat de $\hat{T}_j$, el qual s'incorporarà més endavant. És per això que una molt bona aproximació al valor real de l'MR_j es pot obtenir mitjançant

$$\mathrm{MR}_j \approx \frac{j-0.3}{n+0.4}.$$

3. Es transformen les estimacions de no-fiabilitat mitjançant l'aplicació

$$Q \mapsto \log(-\log(1-Q)).$$

Aquesta funció està ben definida, ja que $Q \in (0,1)$ i $-\log(1-Q) > 0$. S'observa que la funció transforma la *cdf* de la Weibull en una recta. En efecte,

$$Q \coloneqq F_X(x;\alpha,\beta) = 1 - \exp\left(-\left(\frac{x}{\alpha}\right)^{\beta}\right)$$

$$Q - 1 = -\exp\left(-\left(\frac{x}{\alpha}\right)^{\beta}\right)$$

$$-\log(1-Q) = \left(\frac{x}{\alpha}\right)^{\beta}$$

$$\log(-\log(1-Q)) = \beta\log(x) - \beta\log(\alpha).$$

4. Amb la mostra de parells de temps de fallada i les seves estimacions de no-fiabilitat $\left(T_{\sigma^{-1}(j)}, Q_j\right)$, s'utilitza Mínims Quadrats per ajustar una recta i extreure els coeficients resultants m i b. Llavors, la inferència sobre els paràmetres del nostre model de Weibull es pot aconseguir mitjançant

$$m \equiv \beta$$

$$b \equiv -\beta\log(\alpha) \Rightarrow \alpha = \exp\left(-\frac{b}{\beta}\right).$$

4.3. Implementació computacional

Els diferents mètodes d'estimació descrits prèviament s'implementen com a funcions al mòdul `src/inference.py`.

1. **Median Ranks Regression (MRR)**: En el cas del MRR, la funció `median_ranks_regression` pren una mostra de simulació arbitrària i implementa el procediment explicat anteriorment. En particular, el paràmetre `method` permet escollir entre `"beta"` i `"bernard"`, per calcular els rangs medians utilitzan la forma explícita de la funció beta incompleta regularitzada (disponible per `from scipy.special import betaincinv`) o bé l'aproximació de Bernard. En ambdós casos s'utilitzen operacions matricials de NumPy,

de forma que el procés és vectoritzat i per tant més ràpid. Finalment, els Mínims Quadrats es computen mitjançant `linregress` de `scipy.stats`.

2. **Maximum Likelihood Estimation (MLE)**: **TODO**

5. Estimand

Volem estimar els valors d'α, β de la weibull empiricament, donades les dades que hem generat per simulació.

Per a cada combinació dels paràmetres (n, α, β) generarem $m = 1000$ repetitions independents. Per a cadascuna d'aquestes, calcularem els estimadors d'α, β per a els tres mètodes que estem avaluant: MLE ($\hat{\alpha}_{\text{ML}}, \hat{\beta}_{\text{ML}}$), MRR ($\hat{\alpha}_{\text{MRR}}, \hat{\beta}_{\text{MRR}}$) i l'aproximació de Bertrand de MRR ($\hat{\alpha}_{\text{B}}, \hat{\beta}_{\text{B}}$).

Un cop trobats tots els estimands, utilitzarem les mesures de l'apartat següent per inferir-ne la qualitat.

6. Avaluació del Rendiment

Per tal d'avaluar el mètode de Regressió per Rangs de Mediana (MRR), realitzarem una comparativa directa amb l'Estimació de Màxima Versemblança (MLE). Aquesta avaluació es durà a terme mitjançant una simulació de Monte Carlo amb m iteracions.

En aquest context, denotem θ com el valor real del paràmetre i $\hat{\theta}_i$ com el valor estimat en la i-èssima iteració. Les mètriques de rendiment seleccionades per a l'anàlisi es presenten a la taula següent:

Mètrica	**Fórmula**	**Descripció**
Biaix (Bias)	$\frac{1}{m}\sum_{i=1}^{m}\left(\hat{\theta}_i - \theta\right)$	Indica la distància mitjana entre l'esperança de l'estimador i el valor real.
Error Estàndard Empíric (ESE)	$\sqrt{\frac{1}{m-1}\sum_{i=1}^{m}\left(\hat{\theta}_i - \bar{\hat{\theta}}\right)^2}$	Mesura la variabilitat (desviació estàndard) de les estimacions al voltant de la seva mitjana.
Error Estàndard de Monte Carlo (MCSE)	$\frac{\text{ESE}}{\sqrt{m}}$	Quantifica la incertesa de l'estimació deguda al nombre finit de simulacions (m).
Error Quadràtic Mitjà (MSE)	$\frac{1}{m}\sum_{i=1}^{m}\left(\hat{\theta}_i - \theta\right)^2$	Mesura global que combina la variància i el biaix ($\text{MSE} \approx \text{ESE}^2 + \text{Biaix}^2$).

Taula 1: Resum de les mètriques de rendiment utilitzades per comparar els mètodes.

7. Results

Resultats: he agrupat en csv per alpha beta. Cadascun conté 3 files, una per a cada n. proposo escriure directament aquestes taules a l'apartat de resultats, partint-ho per MLE, MRR i Betrand

1. files: n
2. Columnes: Biax, MCSE MSE de MRR i després de MLE
3. Hi haurà 9 taules, una per cada valor diferent de alpha beta.

8. Annex 1

Derivada respecte β:

$$\frac{\partial \ell}{\partial \beta} = -\frac{n\alpha}{\beta} - \left(\sum_{i=1}^{n} x_i^{\alpha}\right)\left(\frac{\partial}{\partial \beta}\beta^{-\alpha}\right)$$

$$\frac{\partial \ell}{\partial \beta} = -\frac{n\alpha}{\beta} - \left(\sum_{i=1}^{n} x_i^{\alpha}\right)(-\alpha\beta^{-\alpha-1})$$

$$-\frac{n\alpha}{\beta} + \alpha\beta^{-\alpha-1}\sum_{i=1}^{n} x_i^{\alpha} = 0$$

Multipliquem per $\frac{\beta}{\alpha}$ ($\alpha, \beta \neq 0$):

$$-n + \beta^{-\alpha}\sum_{i=1}^{n} x_i^{\alpha} = 0$$

$$\beta^{\alpha} = \frac{1}{n}\sum_{i=1}^{n} x_i^{\alpha}$$

Aïllant β, obtenim $\hat{\beta}$ com una funció respecte $\hat{\alpha}$:

$$\hat{\beta} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^{\hat{\alpha}}\right)^{\frac{1}{\hat{\alpha}}}$$

Derivada parcial respecte α:

$$\frac{\partial \ell}{\partial \alpha} = \frac{n}{\alpha} - n\ln\beta + \sum_{i=1}^{n}\ln x_i - \sum_{i=1}^{n}\frac{\partial}{\partial \alpha}\left(\frac{x_i}{\beta}\right)^{\alpha}$$

Si en adonem que $\frac{\partial}{\partial \alpha} z^{\alpha} = z^{\alpha}\ln z$, podem posar $z_i = \frac{x_i}{\beta}$:

$$\frac{\partial \ell}{\partial \alpha} = \frac{n}{\alpha} - n\ln\beta + \sum_{i=1}^{n}\ln x_i - \sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{\alpha}\ln\left(\frac{x_i}{\beta}\right) = 0$$

Substituïnt $\ln\left(\frac{x_i}{\beta}\right) = \ln x_i - \ln\beta$:

$$\frac{n}{\alpha} - n\ln\beta + \sum_{i=1}^{n}\ln x_i - \sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{\alpha}(\ln x_i - \ln\beta) = 0$$

Usant l'expressió de β, sabem que $\sum\left(\frac{x_i}{\beta}\right)^{\alpha} = n$. I ho expandim a l'últim terme:

$$\frac{n}{\alpha} - n\ln\beta + \sum_{i=1}^{n}\ln x_i - \sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{\alpha}\ln x_i + \ln\beta\underbrace{\sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{\alpha}}_{=n} = 0$$

$$\frac{n}{\alpha} + \sum_{i=1}^{n}\ln x_i - \sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{\alpha}\ln x_i = 0$$

Substituïm $\beta^{\alpha} = \frac{1}{n}\sum x_i^{\alpha}$ a l'equació i dividim per n

$$\frac{n}{\alpha} + \sum_{i=1}^{n}\ln x_i - \frac{\sum_{i=1}^{n} x_i^{\alpha}\ln x_i}{\frac{1}{n}\sum_{i=1}^{n} x_i^{\alpha}} = 0$$

$$\frac{1}{\hat{\alpha}} + \frac{1}{n}\sum_{i=1}^{n}\ln x_i - \frac{\sum_{i=1}^{n} x_i^{\hat{\alpha}}\ln x_i}{\sum_{i=1}^{n} x_i^{\hat{\alpha}}} = 0$$

Obtenint l'equació que hem de minimitzar.

9. Annex

Posar totes les classes a lo copia i enganxa. Més enllà, fer ènfasi al main, i fer un esforç en que imprimieixi els mateixos results exactes que es presenten al report. (EG la taula, s'ha de trobar com fer-ho)

Bibliografia

[1] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sept. 2020, doi: 10.1038/s41586-020-2649-2.

[2] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[3] J. Dutka, "The incomplete beta function – a historical profile," *Archive for History of Exact Sciences*, vol. 24, no. 1, pp. 11–29, 1981, doi: 10.1007/BF00327713.