Pareto models

We consider the two-parameter Pareto model, written as follows:

$$\left\{ Pa(\theta_1, \theta_2); \theta_1, \theta_2 > 0 \right\}$$

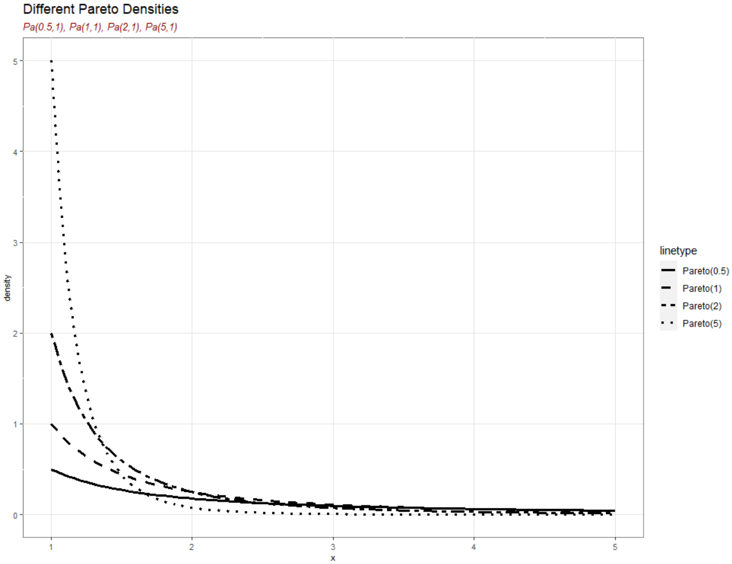
This Pareto distribution is a continuous distribution with support $x \in [\theta_2, \infty[$ and with PDF given by

$$f_{\theta_1,\theta_2}(x) = \frac{\theta_1 \theta_2^{\theta_1}}{x^{\theta_1+1}} \; \mathbb{1}_{[\theta_2,\infty[}(x)$$

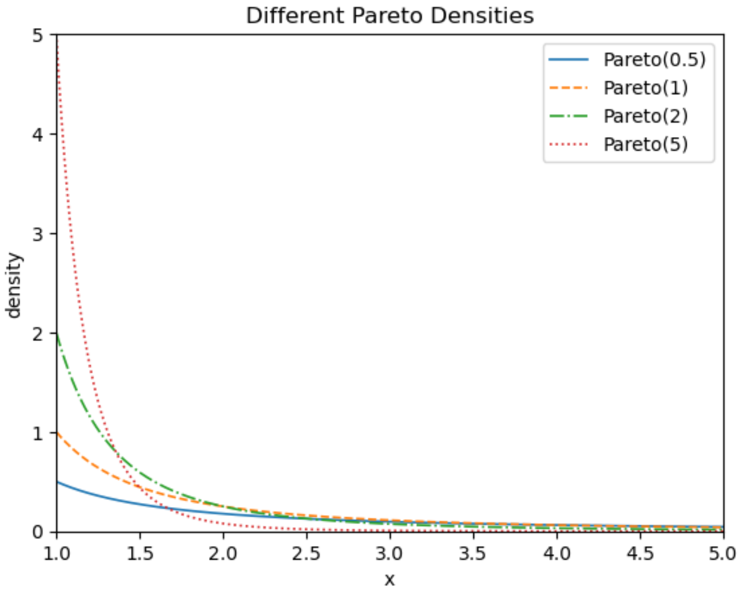
If we set $\theta_2 = 1$, then it reduces to the one-parameter Pareto distribution, which is a continuous distribution on $[1, \infty[$, with PDF given by

$$f_{\theta_1}(x) = \frac{\theta_1}{x^{\theta_1+1}} \; \mathbb{1}_{[1,\infty[}(x)$$

Visualizing the distributions in R



Visualizing the distributions in Python



Expectation of the one-parameter Pareto distribution

Let $X \sim Pa(\theta_1, \theta_2 = 1)$. Then we have that $E[X] = \frac{\theta_1}{\theta_1 - 1}$. Indeed,

$$
\begin{aligned}
E[X] &= \int_{-\infty}^{+\infty} x f_{\theta_1}(x) dx = \int_{1}^{+\infty} x \frac{\theta_1}{x^{\theta_1+1}} dx \\
&= \int_{1}^{+\infty} \frac{\theta_1}{x^{\theta_1}} dx = \theta_1 \int_{1}^{+\infty} x^{-\theta_1} dx \\
&= \theta \left[\frac{x^{-\theta_1+1}}{-\theta_1+1} \right]_1^{\infty} = \theta_1 \left(0 - \frac{1}{-\theta_1+1} \right) \\
&= \frac{-\theta_1}{-\theta_1+1} = \frac{\theta_1}{\theta_1 - 1}
\end{aligned}
$$

Maximum Likelihood estimation (1/2)

To obtain a **Maximum Likelihood Estimator** (MLE) for θ from a sample of n $i.i.d.$ realizations of a Pareto distributed r.v., we first write the likelihood function, take the logarithm of this function, differenciate it w.r.t. the parameter of interest and then solve for the parameter.

$$\mathcal{L}(\theta \mid \mathbf{x}) = \prod_{i=1}^{n} f_\theta(x_i) = \prod_{i=1}^{n} \frac{\theta}{x_i^{\theta+1}} \ \mathbb{1}_{\mathbb{R}_+^*}(x_i)$$

$$= \frac{\theta^n}{\prod_{i=1}^{n} x_i^{\theta+1}} \ \prod_{i=1}^{n} \mathbb{1}_{\mathbb{R}_+^*}(x_i)$$

$$l(\theta \mid \mathbf{x}) = ln\big(\mathcal{L}(\theta \mid \mathbf{x})\big) = n\ ln(\theta) - (\theta+1)\sum_{i=1}^{n} ln(x_i) + \sum_{i=1}^{n} ln\big(\mathbb{1}_{\mathbb{R}_+^*}(x_i)\big)$$

$$\frac{\partial ln\mathcal{L}(\theta \mid \mathbf{x})}{\partial \theta} = \frac{n}{\theta} - \sum_{i=1}^{n} ln(x_i)$$

Maximum Likelihood estimation (2/2)

Setting the first order partial derivative equal to 0 and solving for θ then yields

$$\frac{n}{\theta} - \sum_{i=1}^{n} ln(x_i) = 0 \qquad \Leftrightarrow \qquad \hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} ln(x_i)}$$

So we obtain as Maximum Likelihood Estimator (MLE) the expression $\hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} ln(x_i)}$.

Method of Moments estiamation

Method of Moments (MoM): If $E[\varphi(X)] = h(\theta)$, then we have that $\hat{\theta}_{MOM} = h^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\varphi(x_i)\right)$ is a method of moments estimator for θ. So, we equate the theoretical moment with the first sample moment $E[X] = \overline{x}$ and solve for θ. We get

$$\frac{\theta}{\theta - 1} = \overline{x} \quad \Leftrightarrow \quad \theta = \theta\overline{x} - \overline{x} \quad \Leftrightarrow \quad \theta(1 - \overline{x}) = -\overline{x} \quad \Leftrightarrow \quad \theta = \frac{\overline{x}}{\overline{x} - 1}$$

In this case, our Method of Moments estimator $\hat{\theta}_{MOM} = \frac{\overline{x}}{\overline{x}-1}$ is different from the Maximum Likelihood estimator $\hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} ln(x_i)}$.

Inverse Transform method: rationale

Otherwise known as inverse CDF method. The continuous case:

Suppose that we have a continuous random variable X having Cumulative Density Function (CDF) F_X. Then, the random variable

$$U = F_X(X)$$

has a Uniform distribution $U \sim U_{[0,1]}$. So to generate a random variate x from the distribution of X, we can use the following transformation

$$F_X^{-1}(U) = x$$

where $F_X^{-1}(.)$ is the inverse CDF or quantile function.

Pareto example: introduction

Example We want to generate a sample of $10,000$ random realizations from a Pareto distribution $\mathcal{P}a(4,1)$ using the Inverse CDF method.

If $X \sim \mathcal{P}a(\theta_1, \theta_2 = 1)$ having PDF and CDF defined respectively as

$$\textbf{PDF} \qquad f_{\theta_1}(x) = \frac{\theta_1}{x^{\theta_1+1}}$$

$$\textbf{CDF} \qquad F_{\theta_1}(x) = 1 - \left(\frac{1}{x}\right)^{\theta_1}$$

Pareto example solved

Then by the Inverse CDF method, we can generate realizations of X by equating $U = F_{\theta_1}(x)$ and solving for X. We have

$$\begin{aligned} U &= 1 - \left(\frac{1}{x}\right)^4 \\ 1 - U &= \frac{1}{x^4} \\ \frac{1}{1-U} &= x^4 \\ \sqrt[4]{\frac{1}{1-U}} &= x \end{aligned}$$

Point estimation in R

```
# 4. Function that generates random one-parameter Pareto realizations
rpareto = function(n, theta1, theta2 = 1){
  data = ((1/(1-runif(n)))^{1/theta1})
  return(data)
}

# 5. Create an artificial dataset of size n = 40
set.seed(2024)
xi = rpareto(40, 4, 1)

# 6. Estimation of the parameter using MLE and MOM

# estimation using MoM and MLE
MoM.estimator = mean(xi) / (mean(xi) - 1)
MoM.estimator # [1] 4.006584

ML.estimator = 1 / mean(log(xi))
ML.estimator # [1] 3.886276
```

Point estimation in Python

```
# 3. Function to generate random one-parameter Pareto realizations
def rpareto(n, theta1, theta2=1):
    data = ((1 / (1 - np.random.rand(n))) ** (1 / theta1))
    return data

# 4. Create an artificial dataset of size n = 40
np.random.seed(2024)
xi = rpareto(40, 4, 1)

# 5. Estimation of the parameter using MLE and MOM

MoM_estimator = np.mean(xi) / (np.mean(xi) - 1)
print("MoM Estimator:", MoM_estimator)
MoM Estimator: 4.7041297591504145


MLE_estimator = 1 / np.mean(np.log(xi))
print("MLE Estimator:", MLE_estimator)
MLE Estimator: 4.554886947691036
```

Bootstrapped CI in R

```
# bootstrap
num_bootstraps = 10000
bootstrap_mom = bootstrap_mle = numeric(num_bootstraps)
set.seed(2024)
for (i in 1:num_bootstraps) {
  resample = sample(xi, replace = TRUE)
  bootstrap_mom[i] = mean(resample) / (mean(resample) - 1)
  bootstrap_mle[i] = 1 / mean(log(resample))
}

# Mean and standard error of the estimators
mean_mom = mean(bootstrap_mom); mean_mle = mean(bootstrap_mle)
standard_error_mom = sd(bootstrap_mom); standard_error_mle = sd(bootstrap_mle)
results = matrix(c(mean_mle, standard_error_mle, mean_mom, standard_error_mom),
                 ncol = 2, byrow = TRUE)
rownames(results) = c('mle', 'mom'); colnames(results) = c('mean', 'sd')
#         mean        sd
# mle 3.972500 0.5943127
# mom 4.098544 0.5458712

# asymptotic confidence intervals
CI = matrix(cbind(c(mean_mle - qnorm(1-0.05/2)*standard_error_mle,
                    mean_mle + qnorm(1-0.05/2)*standard_error_mle),
                  c(mean_mom - qnorm(1-0.05/2)*standard_error_mom,
                    mean_mom + qnorm(1-0.05/2)*standard_error_mom)),
            ncol = 2, byrow = 2)
rownames(CI) = c('mle', 'mom'); colnames(CI) = c('lower bound', 'upper bound')
#     lower bound upper bound
# mle    2.807668    5.137331
# mom    3.028656    5.168432
```

Bootstrapped CI in Python

```python
import pandas as pd
from scipy.stats import norm

# Bootstrap
num_bootstraps = 10000; bootstrap_mom = np.zeros(num_bootstraps)
bootstrap_mle = np.zeros(num_bootstraps); np.random.seed(2024)

for i in range(num_bootstraps):
    resample = np.random.choice(xi, size=len(xi), replace=True)
    bootstrap_mom[i] = np.mean(resample) / (np.mean(resample) - 1)
    bootstrap_mle[i] = 1 / np.mean(np.log(resample))

# Mean and standard error of the estimators
mean_mom = np.mean(bootstrap_mom); mean_mle = np.mean(bootstrap_mle)
standard_error_mom = np.std(bootstrap_mom); standard_error_mle = np.std(
     bootstrap_mle)
results = np.array([[mean_mle, standard_error_mle], [mean_mom,
     standard_error_mom]])
results = pd.DataFrame(results, index=['mle', 'mom'], columns=['mean', 'se'])
#              mean         sd

# Asymptotic confidence intervals
CI_mle = [mean_mle - norm.ppf(1 - 0.05/2) * standard_error_mle,
          mean_mle + norm.ppf(1 - 0.05/2) * standard_error_mle]
...
CI = pd.DataFrame([CI_mle, CI_mom], index=['mle', 'mom'], columns=['lower bound'
     , 'upper bound'])
#          lower bound  upper bound
# mle      3.388854     5.885390
# mom      3.567692     6.039047
```

References

Rizzo, M. L. (2019), Statistical Computing with R, Second Edition, Chapman Hall/CRC, ISBN 9781466553330

The R Project for Statistical Computing:
https://www.r-project.org/

Python:
https://www.python.org/

course notes