UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

Data Analysis & Pattern Recognition

Statistical inference and parameter estimation. Maximum likelihood estimation, Bayesian inference, bootstrapping

Francesc Pozo

Escola d'Enginyeria de Barcelona Est (EEBE)
Universitat Politècnica de Catalunya (UPC)

Master's Degree in Chemical Engineering
Master's Degree in Interdisciplinary and Innovative Engineering

Example

- We begin by considering a single binary random variable X where $X(\Omega) = \{0, 1\}$. For example, X might describe the outcome of flipping a coin, with $X = 1$ representing 'heads' and $X = 0$ representing 'tails'.
- We can imagine that this is a damaged coin so that the probability of landing heads is not necessarily the same as that of landing tails.
- The probability of $X = 1$ will be denoted by the parameter μ so that

$$\boxed{p(X = 1 \mid \mu) = \mu,}$$

where $0 \leq \mu \leq 1$, from which it follows that

$$\boxed{p(X = 0 \mid \mu) = 1 - \mu.}$$

Example

- The probability distribution function (pdf) over x can therefore be written in the form

$$\boxed{f(x \mid \mu) = p(X = x \mid \mu) = \mu^x (1-\mu)^{1-x}}$$

which is known as the Bernoulli distribution:

$$\boxed{X \hookrightarrow b(\mu)}$$

- It is easily verified that

$$E(X) = \mu$$
$$\text{var}(X) = \mu(1-\mu)$$

Motivating Example: MLE vs BI

Example

- Now suppose we have a data set $\mathcal{D} = \{x_1, \ldots, x_N\}$ of observed values of X.
- We can construct the likelihood function, which is a function of μ, on the assumption that the observations are drawn independently, so that

$$p(\mathcal{D} \mid \mu) = \prod_{i=1}^{N} p(X = x_i \mid \mu) = \prod_{i=1}^{N} \mu^{x_i}(1-\mu)^{1-x_i}$$

- In a frequentist setting, we can estimate a value for μ by maximizing the likelihood function. The maximum likelihood estimator is

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

which is also known as the sample mean.

Motivating Example: MLE vs BI

Example

- If we denote the number of heads within this data set by m, then we can write

$$\boxed{\mu_{\mathrm{ML}} = \frac{m}{N}}$$

"The probability of landing heads is given, in this maximum likelihood framework, by the fraction of observations of heads in the data set."

Example

- Now suppose we flip a coin, say, 3 times and happen to observe 3 heads. Then

$$N = m = 3$$

and

$$\boxed{\mu_{\mathrm{ML}} = 1}$$

- In this case, the maximum likelihood result would predict that all future observations should give heads.
- Common sense tells us that this is unreasonable, and in fact this is an extreme example of the over-fitting associated with the maximum likelihood.
- We shall see (BI) how to arrive at more sensible conclusions through the introduction of a prior distribution over μ.

Maximum Likelihood Estimators (MLE)[1]

MLE

When sampling from a population described by a pdf $f(x|\theta)$, kwoledge of θ provides knowledge of the entire population. The idea behind maximum likelihood is to select the value for θ that makes the observed data most likely under the assumed probability model.

Likelihood function

When $\mathbf{x} = \{x_1, x_2, \cdots .x_N\}$ are the observed values of a random variable X from a population with parameter θ, the likelihood function of θ for $\mathbf{x}$ is denoted by

$$\boxed{L(\theta|\mathbf{x}) = f(\mathbf{x}|\theta) = \prod_{i=1}^{N} f(x_i|\theta) = f(x_1, \theta) \cdot f(x_2|\theta) \cdots f(x_N|\theta)}$$

[1]Based on Ugarte, M.D., Militino, A.F. and Arnholt, A.T., 2015. *Probability and Statistics with R*. Chapman and Hall/CRC.

Maximum Likelihood Estimators (MLE)

Log-likelihood function

In general, the likelihood function may be difficult to manipulate, and it is usually more convenient to work with the natural logarithm of $L(\theta|\mathbf{x})$, called the log-likelihood function, since it converts products into sums.

$$\boxed{\ln\left(L(\theta|\mathbf{x})\right) = \ln\left(\prod_{i=1}^{N} f(x_i|\theta)\right) = \sum_{i=1}^{N} \ln\left(f(x_i|\theta)\right)}$$

Maximum Likelihood Estimate

Finding the value θ that maximizes the log-likelihood function is equivalent to finding the value of θ that maximizes $L(\theta|\mathbf{x})$ since the natural logarithm is a monotonically increasing function.

A possible MLE solution is

$$\boxed{\frac{\partial\left(\ln\left(L(\theta|\mathbf{x})\right)\right)}{\partial\theta} = 0}$$

Example

Suppose $\{x_1, x_2, \ldots, x_N\}$ are the observed values of a random variable $X \hookrightarrow N(\mu, \sigma^2)$, where σ is assumed to be known. Find the maximum likelihood estimator of μ.

The likelihood function is

$$L(\mu|\mathbf{x}) = \prod_{i=1}^{N} f(x_i|\mu) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(x_i-\mu)^2}{2\sigma^2}\right\}$$
$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left\{\sum_{i=1}^{N} \frac{-(x_i-\mu)^2}{2\sigma^2}\right\}$$

Example

The log-likelihood function is

$$\ln\left(L(\mu|\mathbf{x})\right) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i - \mu)^2$$

To find the value of μ that maximizes $\ln\left(L(\mu|\mathbf{x})\right)$, take the first-order partial derivative with respect to μ, set the answer equal to zero, and solve.

$$\frac{\partial\left(\ln\left(L(\mu|\mathbf{x})\right)\right)}{\partial\mu} = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i - \mu) = 0 \implies \boxed{\hat{\mu}_{ML} = \frac{1}{N}\sum_{i=1}^{N}x_i = \bar{x}}$$

Example

Suppose $\{x_1, x_2, \ldots, x_N\}$ are the observed values of a random variable $X \hookrightarrow N(\mu, \sigma^2)$, where μ is assumed to be known. Find the maximum likelihood estimator of σ^2.

The likelihood function is

$$L(\sigma^2|\mathbf{x}) = \prod_{i=1}^{N} f(x_i|\sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(x_i-\mu)^2}{2\sigma^2}\right\}$$

$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left\{\sum_{i=1}^{N} \frac{-(x_i-\mu)^2}{2\sigma^2}\right\}$$

Example

The log-likelihood function is

$$\ln\left(L(\sigma^2|\mathbf{x})\right) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2$$

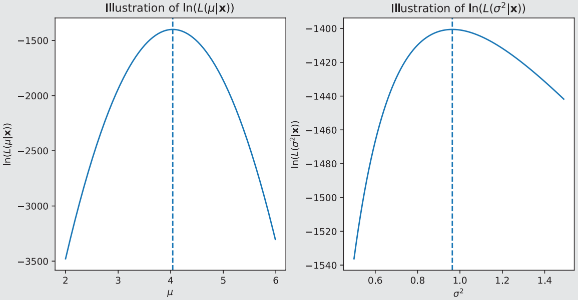
To find the value of σ^2 that maximizes $\ln\left(L(\sigma^2|\mathbf{x})\right)$, take the first-order partial derivative with respect to σ^2, set the answer equal to zero, and solve.

$$\frac{\partial\left(\ln\left(L(\sigma^2|\mathbf{x})\right)\right)}{\partial\sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{N}(x_i-\mu)^2 = 0$$

$$\Longrightarrow \boxed{\hat{\sigma}^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2 = s^2_N}$$

Example

Generate 1000 $N(4,1)$ random variables. Write log-likelihood functions for the simulated random variables and verify that the simulated maximum likelihood estimates for μ and σ^2 are reasonably close to the true parameters. Produce side-by-side graphs of $\ln(L(\mu|\mathrm{x}))$ and $\ln(L(\sigma|\mathrm{x}))$ indicating where the simulated maximum occurs in each graph.



Python code

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
N = 1000
mu = 4
sigma2 = 1
np.random.seed(1)
x = np.random.normal(mu, np.sqrt(sigma2), N)
def negloglikemu(muv):
    return N/2*np.log(2*np.pi)+N/2*np.log(sigma2)\
    +(np.sum(np.square(x))-2*muv*np.sum(x)+N*muv**2)/(2*sigma2)
def negloglike(sv):
    return N/2*np.log(2*np.pi)+N/2*np.log(sv)\
    +(np.sum(np.square(x))-2*mu*np.sum(x)+N*mu**2)/(2*sv)
rr = np.arange(2, 6, 0.01)
rr2 = np.arange(0.5, 1.5, 0.01)
f, axes = plt.subplots(1, 2, figsize=(10, 5), sharex=False)
plt.subplots_adjust(wspace=.25,hspace=0)
```

Python code

```
axes[0].plot(rr, -negloglikemu(rr))
axes[0].set_xlabel('$\mu$')
axes[0].set_ylabel('$\ln(L(\mu|\mathbf{x}))$')
from scipy.optimize import fmin
import math
mumin = fmin(negloglikemu,np.array([2]))
sigma2min = fmin(negloglike,np.array([2]))
axes[0].axvline(x=mumin,linestyle='--')
axes[1].plot(rr2, -negloglike(rr2))
axes[0].set_title('Illustration of $\ln(L(\mu|\mathbf{x}))$')
axes[1].set_xlabel('$\sigma^2$')
axes[1].set_ylabel('$\ln(L(\sigma^2|\mathbf{x}))$')
axes[1].set_title('Illustration of $\ln(L(\sigma^2|\mathbf{x}))$
    ')
axes[1].axvline(x=sigma2min,linestyle='--')
plt.savefig('loglike.eps', dpi=300, bbox_inches='tight')
plt.show()
```

Example

Given the density function

$$f(x) = (\theta + 1)(1 - x)^{\theta},\ 0 \leq x \leq 1,\ \theta > 0,$$

(a) Find the maximum likelihood estimator of θ for a random sample of size N.

Example

Given the density function

$$f(x) = \theta e^{-\theta x}, \; x \geq 0, \; \theta > 0,$$

(a) Find the maximum likelihood estimator of θ for a random sample of size N.

(b) Set the seed equal to 88, and generate 1000 values from $f(x)$ when $\theta = 2$. Calculate the maximum likelihood estimate of θ from the generated values.

(c) How close is the maximum likelihood estimate in (b) to $\theta = 2$?

Posterior distribution with a sample size of one

Let us begin with a simple example in which we consider a single Gaussian random variable X. We shall suppose that the variance σ^2 is known, and we consider the task of inferring the mean μ given a set of $N = 1$ observation $\mathbf{x} = \{x_1\}$. According to Bayes' theorem:

$$p(\mu|\mathbf{x}) = \frac{p(\mathbf{x}|\mu)p(\mu)}{p(\mathbf{x})}$$

where $p(\mu|\mathbf{x})$ is the posterior probability distribution, $p(\mathbf{x}|\mu)$ is the likelihood and $p(\mu)$ is the prior probability distribution. $p(\mathbf{x})$ is the normalization constant and it can be expressed as:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mu)p(\mu)d\mu \in \mathbb{R}$$

[2]Based on Bishop, C.M., 2006. *Pattern recognition and machine learning*. Springer.

Posterior distribution with a sample size of one

Since $X \hookrightarrow N(\mu, \sigma^2)$, then

$$p(\mathbf{x}|\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(x_1-\mu)^2}{2\sigma^2}\right\}$$

If we choose a prior $p(\mu)$ given by a Gaussian

$$\mu \hookrightarrow N(\mu_0, \sigma_0^2)$$

where μ_0 and σ_0^2 are known, then

$$p(\mu) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{\frac{-(\mu-\mu_0)^2}{2\sigma_0^2}\right\}$$

Posterior distribution with a sample size of one

The posterior distribution of μ given that we have one observation $\mathbf{x} = \{x_1\}$ is

$$p(\mu|\mathbf{x}) = \frac{p(\mathbf{x}|\mu)p(\mu)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\mu)p(\mu)}{\int p(\mathbf{x}|\mu)p(\mu)d\mu} \propto p(\mathbf{x}|\mu)p(\mu)$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(x_1-\mu)^2}{2\sigma^2}\right\} \cdot \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{\frac{-(\mu-\mu_0)^2}{2\sigma_0^2}\right\}$$

$$= \underbrace{\frac{1}{\sqrt{2\pi\sigma^2\sigma_0^2}}}_{\text{constant}} \exp\left\{\frac{-x_1^2+2x_1\mu-\mu^2}{2\sigma^2} + \frac{-\mu^2+2\mu\mu_0-\mu_0^2}{2\sigma_0^2}\right\}$$

$$\propto \exp\left\{\frac{-x_1^2\sigma_0^2+2x_1\mu\sigma_0^2-\mu^2\sigma_0^2-\mu^2\sigma^2+2\mu\mu_0\sigma^2-\mu_0^2\sigma^2}{2\sigma^2\sigma_0^2}\right\}$$

Posterior distribution with a sample size of one

$$
\begin{aligned}
p(\mu|\mathrm{x}) &\propto \exp\left\{\frac{-x_1^2\sigma_0^2 + 2x_1\mu\sigma_0^2 - \mu^2\sigma_0^2 - \mu^2\sigma^2 + 2\mu\mu_0\sigma^2 - \mu_0^2\sigma^2}{2\sigma^2\sigma_0^2}\right\} \\
&= \exp\left\{\frac{-\mu^2\left(\sigma^2+\sigma_0^2\right) + 2\mu\left(\mu_0\sigma^2 + \sigma_0^2x_1\right) - \left(\mu_0^2\sigma^2 + \sigma_0^2x_1^2\right)}{2\sigma_0^2\sigma^2}\right\} \\
&= \exp\left\{\frac{-\mu^2 + 2\mu\dfrac{\mu_0\sigma^2+\sigma_0^2x_1}{\sigma^2+\sigma_0^2} - \dfrac{\mu_0^2\sigma^2+\sigma_0^2x_1^2}{\sigma^2+\sigma_0^2}}{\dfrac{2\sigma_0^2\sigma^2}{\sigma^2+\sigma_0^2}}\right\}
\end{aligned}
$$

Posterior distribution with a sample size of one

But,

$$\frac{\mu_0^2\sigma^2 + \sigma_0^2 x_1^2}{\sigma^2 + \sigma_0^2} = \left(\frac{\mu_0\sigma^2 + x_1\sigma_0^2}{\sigma^2 + \sigma_0^2}\right)^2 + \frac{\sigma^2\sigma_0^2(x - \mu_0)^2}{(\sigma^2 + \sigma_0^2)^2}$$

and, therefore

$$p(\mu|\mathbf{x}) \propto \exp\left\{\frac{-\mu^2 + 2\mu\frac{\mu_0\sigma^2 + \sigma_0^2 x_1}{\sigma^2 + \sigma_0^2} - \left(\frac{\mu_0\sigma^2 + x_1\sigma_0^2}{\sigma^2 + \sigma_0^2}\right)^2}{\frac{2\sigma_0^2\sigma^2}{\sigma^2 + \sigma_0^2}}\right\}$$

$$\times \underbrace{\exp\left\{\frac{\sigma^2\sigma_0^2(x - \mu_0)^2}{(\sigma^2 + \sigma_0^2)^2}\right\}}_{\text{constant}}$$

Posterior distribution with a sample size of one

$$p(\mu|\mathbf{x}) \propto \exp\left\{\frac{-\mu^2 + 2\mu\dfrac{\mu_0\sigma^2 + \sigma_0^2 x_1}{\sigma^2 + \sigma_0^2} - \left(\dfrac{\mu_0\sigma^2 + x_1\sigma_0^2}{\sigma^2 + \sigma_0^2}\right)^2}{\dfrac{2\sigma_0^2\sigma^2}{\sigma^2 + \sigma_0^2}}\right\}$$

Let us define

$$\sigma_1^2 = \frac{\sigma_0^2\sigma^2}{\sigma^2 + \sigma_0^2} = \frac{1}{\sigma^{-2} + \sigma_0^{-2}}$$

$$\mu_1 = \frac{\mu_0\sigma^2 + x_1\sigma_0^2}{\sigma^2 + \sigma_0^2} = \frac{\mu_0\sigma^2 + x_1\sigma_0^2}{\sigma^2 + \sigma_0^2} = \frac{1}{\sigma^{-2} + \sigma_0^{-2}}\left(\mu_0\sigma_0^{-2} + x_1\sigma^{-2}\right)$$

$$= \sigma_1^2\left(\mu_0\sigma_0^{-2} + x_1\sigma^{-2}\right)$$

Posterior distribution with a sample size of one

And hence,

$$p(\mu|\mathbf{x}) \propto \exp\left\{\frac{-(\mu-\mu_1)^2}{2\sigma_1^2}\right\},$$

from which it follows that as density, must integrate to unity,

$$p(\mu|\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left\{\frac{-(\mu-\mu_1)^2}{2\sigma_1^2}\right\}$$

The posterior distribution is given by

$$\mu|\mathbf{x} \hookrightarrow N(\mu_1, \sigma_1^2)$$

Posterior distribution with a sample of size N

We consider a single Gaussian random variable X. We shall suppose that the variance σ^2 is known, and we consider the task of inferring the mean μ given a set of N observations $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$. If we choose a prior $p(\mu)$ given by a Gaussian

$$\mu \hookrightarrow N(\mu_0, \sigma_0^2)$$

where μ_0 and σ_0^2 are known, then the posterior distribution is given by

$$\mu|\mathbf{x} \hookrightarrow N(\mu_N, \sigma_N^2)$$

$$\mu_N = \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)$$

$$\frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2}$$

Example

Consider a single Gaussian random variable X with variance $\sigma^2 = 1$. Infer the mean $\mu = \mu_N$ given the set of $N = 10$ observations

$$2.16698806, 1.52581308, 0.72238059, 2.44863382, 2.20167179,$$
$$0.44891844, 1.13245188, 0.36254031, 0.17785248, 3.27225828,$$

if we choose a prior $p(\mu)$ given by a Gaussian

$$\mu \hookrightarrow N(\mu_0 = 1, \sigma_0^2 = 1.5)$$

The bootstrap

- The bootstrap is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

[3] Based on James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. *An introduction to statistical learning* (Vol. 112, p. 18). New York: Springer.

Example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X, and will invest the remaining $1 - \alpha$ in Y.
- We wish to choose α to minimize the total risk –or variance– of our investment. In other words, we want to minimize

$$\boxed{\text{var}\left(\alpha X + (1-\alpha)Y\right)}$$

- One can show that the value that minimizes the risk is given by

$$\boxed{\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}}$$

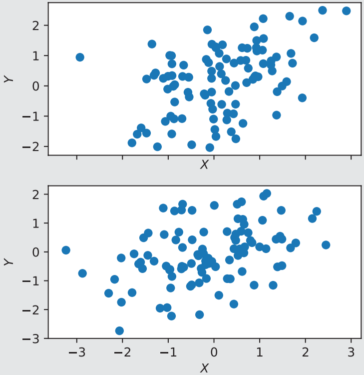
Example

- However, the values of σ_X^2, σ_Y^2 and σ_{XY} are unknown.
- We can compute estimates for these quantities, $\hat{\sigma}_X^2, \hat{\sigma}_Y^2$ and $\hat{\sigma}_{XY}$, using a data set that contains measurements for X and Y.
- We can then estimate the value of α that minimizes the variance of our investment using

$$\boxed{\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}}$$

Example

- Each panel displays 100 simulated returns for investments X and Y. From left to right and top to bottom, the resulting estimates for α are 0.704, 0.614, 0.698, and 0.486.



Python code

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
meanx = 0
meany = 0
mean = (meanx, meany)
sigmaX2 = 1
sigmaY2 = 1.25
sigmaXY = 0.5
cov = [[sigmaX2, sigmaXY], [sigmaXY, sigmaY2]]
np.random.seed(3)
x = np.random.multivariate_normal(mean, cov,size=(100,4))
f, axes = plt.subplots(2, 2, figsize=(10, 5), sharex=True)
axes[0,0].scatter(x[:,0,0], x[:,0,1])
axes[0,0].set_xlabel('$X$')
axes[0,0].set_ylabel('$Y$')
```

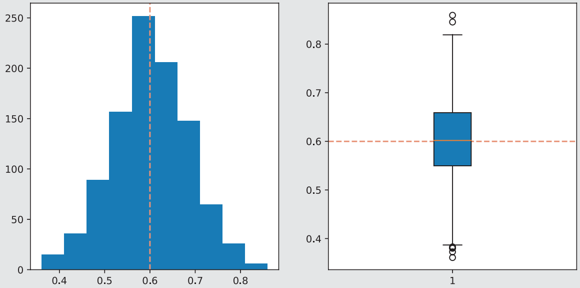
Python code

```
axes[0,1].scatter(x[:,1,0], x[:,1,1])
axes[0,1].set_xlabel('$X$')
axes[0,1].set_ylabel('$Y$')
axes[1,0].scatter(x[:,2,0], x[:,2,1])
axes[1,0].set_xlabel('$X$')
axes[1,0].set_ylabel('$Y$')
axes[1,1].scatter(x[:,3,0], x[:,3,1])
axes[1,1].set_xlabel('$X$')
axes[1,1].set_ylabel('$Y$')
plt.savefig('randomXY.eps', dpi=300, bbox_inches='tight')
plt.show()
```

Example

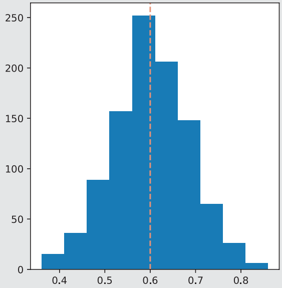
- To estimate the standard deviation of $\hat{\alpha}$, we repeat the process of simulating 100 paired observations of *X* and *Y*, and estimating α 1000 times.
- We thereby obtained 1000 estimates for α, which we can call

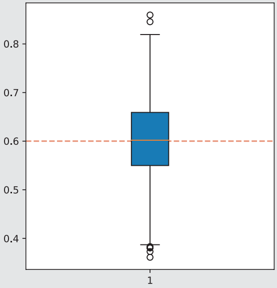
$$\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{1000}$$



Example

- For these simulations the parameters were set to $\sigma_X^2 = 1, \sigma_Y^2 = 1.25$ and $\sigma_{XY} = 0.5$, and so we know that the true value of α is 0.6. We indicated this value using a dashed vertical line on the histogram.





Python code

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
meanx = 0
meany = 0
mean = (meanx, meany)
sigmaX2 = 1
sigmaY2 = 1.25
sigmaXY = 0.5
cov = [[sigmaX2, sigmaXY], [sigmaXY, sigmaY2]]
np.random.seed(3)
x = np.random.multivariate_normal(mean, cov,size=(100,1000))
alpha_list = list()
for k in range(0, 1000):
    sigmaY2hat = np.var(x[:,k,1],ddof=0)
    sigmaX2hat = np.var(x[:,k,0],ddof=0)
    sigmaXYhat = np.cov([x[:,k,0],x[:,k,1]],ddof=0)[0,1]
    alphahat =  (sigmaY2hat-sigmaXYhat)/(sigmaY2hat+sigmaX2hat
    -2*sigmaXYhat)
    alpha_list.append(alphahat)
```

Python code

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
f, axes = plt.subplots(1, 2, figsize=(10, 5), sharex=False)
axes[0].hist(alpha_list)
axes[0].axvline(x=0.6,linestyle='--',color='darksalmon')
axes[1].boxplot(alpha_list,patch_artist=True)
axes[1].axhline(y=0.6,linestyle='--',color='darksalmon')
plt.savefig('histobox.eps', dpi=300, bbox_inches='tight')
plt.show()
```

Example

- The mean over all 1000 estimates for α is

$$\bar{\alpha} = \frac{1}{1000} \sum_{i=1}^{1000} \hat{\alpha}_i = 0.6030,$$

  very close to $\alpha = 0.6$.

- The standard deviation of the estimates is

$$\sqrt{\frac{1}{1000-1} \sum_{i=1}^{1000} (\hat{\alpha}_i - \bar{\alpha})^2} = 0.084$$

- This gives us a very good idea of the accuracy of $\hat{\alpha}$. Roughly speaking, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

Python code

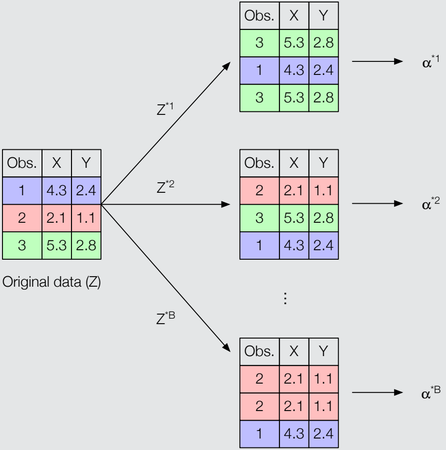
```
>>np.mean(alpha_list)
  0.6030401995913561
>>np.std(alpha_list,ddof=1)
  0.08399535702038463
```

The Bootstrap: Back to the Real World!

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set with replacement.
- Each of these “bootstrap data sets” is created by sampling with replacement, and is the same size as our original dataset. As a result some observations may appear more than once and some not at all.

Example

- A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations.



The bootstrap

- Consider an original data set Z with n observations.
- We randomly select n observations (with replacement) from the data set in order to produce a bootstrap data set, $Z^{\star 1}$.
- We can use $Z^{\star 1}$ to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^{\star 1}$.
- This procedure is repeated B times in order to produce B different bootstrap data sets

$$Z^{\star 1}, Z^{\star 2}, \ldots, Z^{\star B},$$

and B corresponding α estimates

$$\hat{\alpha}^{\star 1}, \hat{\alpha}^{\star 2}, \ldots, \hat{\alpha}^{\star B}.$$

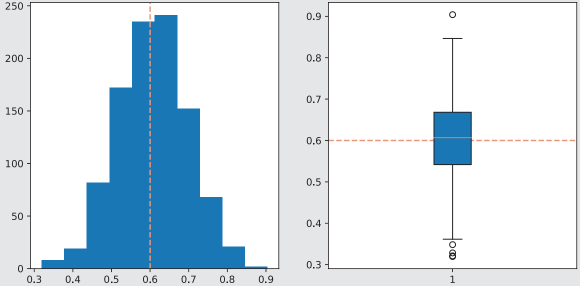
The bootstrap

- We can compute the standard deviation of these bootstrap estimates —aka standard error— using the formula

$$SE(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}\left(\hat{\alpha}^{\star i} - \frac{1}{B}\sum_{j=1}^{B}\hat{\alpha}^{\star j}\right)^2}$$

Example

- *Left*: A histogram of the estimates of α obtained from 1000 bootstrap samples from a single data set. *Right*: The estimates of α displayed in the left panel are shown as a boxplot. In each panel, the dark salmon dashed line indicates the true value of α.



Python code

```
import numpy as np
meanx = 0
meany = 0
mean = (meanx, meany)
sigmaX2 = 1
sigmaY2 = 1.25
sigmaXY = 0.5
cov = [[sigmaX2, sigmaXY], [sigmaXY, sigmaY2]]
np.random.seed(3)
x = np.random.multivariate_normal(mean, cov,size=(100,1000))
bootM = np.zeros((100,2,1000))
alpha_list2 = list()
```

The Bootstrap

Python code

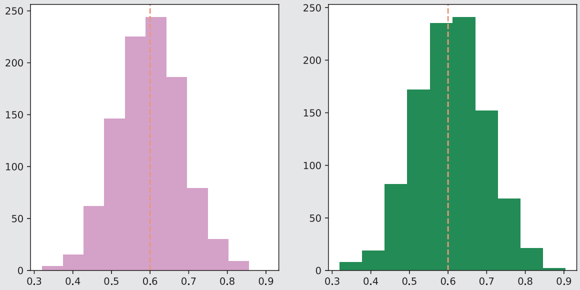
```
for i in range(0,1000):
    nprc = np.random.choice(100,100) # array with 100 random integers between 0 and 99
    for k in range(0,100):
        bootM[k,:,i]=x[nprc[k],0,:] #first bootstrap sample
    sigmaY2hat = np.var(bootM[:,1,i],ddof=0)
    sigmaX2hat = np.var(bootM[:,0,i],ddof=0)
    sigmaXYhat = np.cov([bootM[:,0,i],bootM[:,1,i]],ddof=0)[0,1]
    alphahat =  (sigmaY2hat-sigmaXYhat)/(sigmaY2hat+sigmaX2hat-2*sigmaXYhat)
    alpha_list2.append(alphahat) # 1000 estimates of alpha
```

Python code

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
f, axes = plt.subplots(1, 2, figsize=(10, 5), sharex=False)
axes[0].hist(alpha_list2)
axes[0].axvline(x=0.6,linestyle='--',color='darksalmon')
axes[1].boxplot(alpha_list2,patch_artist=True)
axes[1].axhline(y=0.6,linestyle='--',color='darksalmon')
plt.savefig('histobox2.eps', dpi=300, bbox_inches='tight')
plt.show()
```

Example

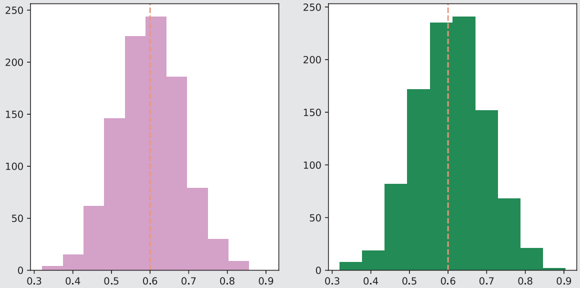
- *Left*: A histogram of the estimates of α obtained by generating 1000 simulated data sets from the true population. *Right*: A histogram of the estimates of α obtained from 1000 bootstrap samples from a single data set. In each panel, the dark salmon dashed line indicates the true value of α.
- Note that both histograms look very similar!



The Bootstrap

Example

- The standard deviation of these bootstrap estimates is 0.090, very close to the estimate of 0.084 obtained using 1000 simulated data sets.



Python code

```
>>np.std(alpha_list2,ddof=1)
  0.08970390965071548
```

Estimating the Accuracy of a Linear Regression Model: Labwork

Simple linear regression

- Simple linear regression is a very straightforward approach for predicting a quantitative response Y on the basis of a single predictor variable X. It assumes that there is approximately a linear relationship between X and Y. Mathematically, we can write this linear relationship as

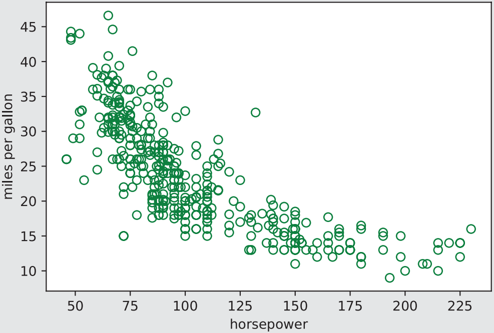
$$\boxed{Y \approx \beta_0 + \beta_1 X.}$$

Example

For example, X may represent `horsepower` and Y may represent `mpg` (miles per gallon) (with respect to the `Auto` data set). Then we can regress `mpg` onto `horsepower` by fitting the model

$$\boxed{\texttt{mpg} \approx \beta_0 + \beta_1 \times \texttt{horsepower}.}$$

Example



The Auto data set. For a number of cars, `mpg` and `horsepower` are shown. There is a pronounced relationship between `mpg` and `horsepower`.

Estimating the Accuracy of a Linear Regression Model: Labwork

Example

- The bootstrap approach can be used to assess the variability of the coefficient estimates and predictions from a statistical learning method.
- We will use the bootstrap approach to assess the variability of the estimates for β_0 and β_1, the intercept and slope terms for the linear regression model that uses `horsepower` to predict `mpg` in the `Auto` data set.
- We first import the following Python packages:

```
>>import numpy as np
>>import csv
>>import pandas as pd
>>from sklearn.linear_model import LinearRegression
>>from sklearn.preprocessing import PolynomialFeatures
```

Example

- We then load the `Auto` data set and remove the missing values.

```
>>filename = "Auto.csv"
>>df = pd.read_csv(filename)
>>moddf = df.dropna()
>>v = moddf.values
```

- The following Python code can be used to compute the intercept and slope estimates for the linear regression model:

```
>>xtrain = v[:,3].reshape((-1,1))
>>ytrain = v[:,0]
>>xtrain_ = PolynomialFeatures(degree=1, include_bias=False).
    fit_transform(xtrain)
>>model = LinearRegression().fit(xtrain_, ytrain)
>>beta0 = model.intercept_
>>beta1 = model.coef_[0]
```

Estimating the Accuracy of a Linear Regression Model: Labwork

Example

- Set this seed so that we all have the exact same result.

```
>>np.random.seed(3)
```

- Create $B = 1000$ bootstrap data sets of $n = N = 392$ observations.
- Create a list **beta0** with the $B = 1000$ corresponding β_0 estimates:

$$\hat{\beta}_0^{\star 1}, \hat{\beta}_0^{\star 2}, \ldots, \hat{\beta}_0^{\star 1000}$$

- Create a list **beta1** with the $B = 1000$ corresponding β_1 estimates:

$$\hat{\beta}_1^{\star 1}, \hat{\beta}_1^{\star 2}, \ldots, \hat{\beta}_1^{\star 1000}$$

Estimating the Accuracy of a Linear Regression Model: Labwork

Example

- Compute the standard deviation —standard error— $SE(\hat{\beta}_0)$ and $SE(\hat{\beta}_1)$ of these bootstrap estimates. You can use `np.std` with `ddof=1`. In your Python code, denote these two values as `std_beta01` and `std_beta11`, respectively.
- Compare the intercept β_0 with the linear regression in the full set of 392 observations with $\frac{1}{1000}\sum_{i=1}^{1000}\hat{\beta}_0^{\star i}$. Compare the slope β_1 too.
- Compare the slope β_1 with the linear regression in the full set of 392 observations with $\frac{1}{1000}\sum_{i=1}^{1000}\hat{\beta}_1^{\star i}$.

Example

- Import the following Python packages and plot a histogram of the estimates of β_0 and β_1, respectively, from the $B = 1000$ bootstrap samples.

```
>>%matplotlib inline
>>import numpy as np
>>import matplotlib.pyplot as plt
```

- **Repeat** this labwork to compute the bootstrap standard error estimates and the standard linear regression estimates that result from fitting a quadratic model

$$\boxed{\texttt{mpg} \approx \beta_0 + \beta_1 \times \texttt{horsepower} + \beta_2 \times \texttt{horsepower}^2}$$

to the data. In your Python code, denote the standard errors as `std_beta02`, `std_beta12` and `std_beta22`, respectively.

Example

- Send the Jupyter Notebook to `francesc.pozo@upc.edu`. Add comments to the code to make it easier to understand.
- You can work in pairs or in threes.
- **Deadline**: January 7th, 2019.