Relationship between p-values and confidence intervals

Xavier Dupré

http://www.xavierdupre.fr/

7 mai 2013

Résumé
This document explains the relationship between p-value and confidence intervals. It goes on with the specific case of a binomial law. Assuming we want to determine whether or not two binomial laws are significantly different, how many observations we need to get the p-value under a given threshold.

Table des matières

1 Confidence interval ...................................................... 1
2 p-value ........................................................................... 2
3 Significant difference between samples mean .................. 3
4 Application on binomial variables ................................. 3
5 Estimate a p-value by using the distribution function ....... 4
6 Correlated variables ....................................................... 5
7 Program which produces the table in Section 4 ............... 5
8 Program which generates Figure 1 ................................. 7
9 Program which estimates the p-value .............................. 9

1 Confidence interval

The term *p-value* is very popular in the world of search engines. I usually prefer confidence interval 95%, I think it is easier to understand. Plus, because p-Value are real values, we could be tempted to compare them and it is usually wrong. On the other hand, it is more difficult to compare confidence intervals, especially if they are related to complete different variables. Their nature prevents you from doing that. However p-Values and confidence interval are similar : they tell you whether or not a metric difference is significant.

Usually, it starts from a set of identically distributed random variables \(X_i\) \(1 \leq i \leq N\). We then estimate the average \(\hat{\theta}_N = \frac{1}{N} \sum_{i=1}^{N} X_i\) and we ask the question : is \(\hat{\theta}_N\) null? In others terms, we want to know if the average is significantly different from zero. If the random variable \(X\) follows a random law which has a standard deviation \(\sigma\) we can use the central limit theorem\(^2\) which tells us :

1. Not all of them have a standard deviation. For example, if \(X\) follows a Cauchy law, \(E(X^2) \sim \int \frac{x^2}{1+x^2} dx\) which does not exist. This remark also concerns every distribution known as heavy tail distribution.
\[ \sqrt{N}\hat{\theta}_N \xrightarrow{N \to \infty} \mathcal{N}(0, \sigma) \]  
\hspace{1cm} (1)

If \( Y \sim \mathcal{N}(0, \sigma) \), then we have \( \mathbb{P}(|Y| \leq 1.96) = 0.95 \). That is why we can say:

\[ \hat{\theta}_N \text{ is not null with 95% confidence if } \sqrt{N} \frac{\hat{\theta}_N}{\sigma} > 1.96 \]  
\hspace{1cm} (2)

And the confidence interval at 95% would be:

\[ \left[ -\frac{1.96\sigma}{\sqrt{N}}, \frac{1.96\sigma}{\sqrt{N}} \right] \]  
\hspace{1cm} (3)

When \( \mathbb{E}(\hat{\theta}_N) = \theta_0 \neq 0 \), it becomes:

\[ \sqrt{N} \left[ \hat{\theta}_N - \theta_0 \right] \xrightarrow{N \to \infty} \mathcal{N}(0, \sigma) \]  
\hspace{1cm} (4)

We usually want to check if the mean is equal to a specific value using a statistical test:

\[ H_0 : \; \hat{\theta}_N = \theta_0 \]
\[ H_1 : \; \hat{\theta}_N \neq \theta_0 \]

We validate \( H_0 \) if:

\[ \hat{\theta}_N \in \left[ \theta_0 - \frac{1.96\sigma}{\sqrt{N}}, \theta_0 + \frac{1.96\sigma}{\sqrt{N}} \right] \]  
\hspace{1cm} (5)

2  \textbf{p-value}

With confidence intervals, you first choose a confidence level and then you get an interval. You then check if your value is inside or outside your interval. Inside, the gain is not significant, outside, it is.

With a p-value, we consider the problem the other way: given \( \hat{\theta}_N \), what is the probability that the difference \( |\hat{\theta}_N - \theta_0| \) is significant? Let’s consider \( Y \) following a normal law \( \mathcal{N}(0, 1) \). We are looking for:

\[ \mathbb{P} \left( |Y| > \sqrt{N} \frac{\hat{\theta}_N}{\sigma} \right) = \alpha \]  
\hspace{1cm} (6)

\( \alpha \) is the p-value.
\[ \alpha = 1 - \int_{-\beta_N}^{\beta_N} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = 2 \int_{\beta_N}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \] 

(7)

where \( \beta_N = \sqrt{N} \frac{\hat{\theta}_N}{\sigma} \)

At this point, we should not forget that we use a theorem which tells us that \( \sqrt{N} \frac{\hat{\theta}_N}{\sigma} \sim \mathcal{N}(0,1) \) when \( N \to \infty \), which means everything we said is true when \( N \) is great enough.

### 3 Significant difference between samples mean

Usually, we do not want to test if an average is null but if the difference between two averages is null. We consider two random samples having the same size, each of them described by \((X_i)\) and \((Y_i)\). All variables are independant. \((X_i)\) are distributed according the same law, we assume the same for \((Y_i)\). We expect the following difference to be null.

\[ \hat{\eta}_N = \frac{1}{N} \sum_{i=1}^{N} X_i - \frac{1}{N} \sum_{i=1}^{N} Y_i = \frac{1}{N} \left[ \sum_{i=1}^{N} X_i - Y_i \right] \]

(8)

Considering expression (8), we can applying the central limit theorem on variable \( Z = X - Y \), we get (\( \eta_0 = 0 \)) :

\[ \sqrt{N} \hat{\eta}_N \overset{N \to \infty}{\longrightarrow} \mathcal{N} \left( 0, \frac{\sqrt{\text{V}(Z)}}{N} \right) \]

(9)

If both samples do not have the same number of observations, this expression becomes :

\[ \sqrt{N} \hat{\eta}_N \overset{N_1 \to \infty, N_2 \to \infty}{\longrightarrow} \mathcal{N} \left( 0, \frac{\sqrt{\text{V}(X)}}{N_1} + \frac{\sqrt{\text{V}(Y)}}{N_2} \right) \]

(10)

### 4 Application on binomial variables

A binomial variable \( X \sim \mathcal{B}(p) \) is defined by :

\[
\begin{align*}
P(X = 0) &= 1 - p \\
P(X = 1) &= p
\end{align*}
\]

Let’s consider two series of observations \((X_i) \sim \mathcal{B}(p)\) and \((Y_i) \sim \mathcal{B}(q)\). We assume \( p \neq q \) and we want to determine how many observations we need to get a p-value below 5%. We know that
\( \mathbb{V}(X_i) = p(1-p) \) and \( \mathbb{V}(Y_i) = q(1-q) \). Table I shows the values. First column contains values for \( p \), first row contains values for \( q - p \). We also assume we have the same number \( N \) of random observations for each variable. The statistical test can be defined like following:

\[
H_0 : \ p = q = p_0 \\
H_1 : \ p \neq q
\]

If \( H_0 \) is true, then:

\[
\sqrt{N} \hat{\theta}_N \xrightarrow{N \to \infty} \mathcal{N} \left( 0, \sqrt{p_0(1-p_0)} \left( \frac{1}{N_1} + \frac{1}{N_2} \right) \right) \tag{11}
\]

### Table 1

<table>
<thead>
<tr>
<th>( \hat{d} )</th>
<th>-0.200</th>
<th>-0.100</th>
<th>-0.020</th>
<th>-0.010</th>
<th>-0.002</th>
<th>-0.001</th>
<th>0.001</th>
<th>0.002</th>
<th>0.010</th>
<th>0.020</th>
<th>0.100</th>
<th>0.200</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>91</td>
<td>3650</td>
<td>91235</td>
<td>364939</td>
<td>91235</td>
<td>3650</td>
<td>914</td>
<td>37</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>79</td>
<td>1729</td>
<td>6915</td>
<td>172866</td>
<td>691463</td>
<td>172866</td>
<td>6915</td>
<td>1729</td>
<td>79</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15</td>
<td>98</td>
<td>2449</td>
<td>9796</td>
<td>244893</td>
<td>979572</td>
<td>244893</td>
<td>9796</td>
<td>2449</td>
<td>98</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td>31</td>
<td>123</td>
<td>3074</td>
<td>12293</td>
<td>307317</td>
<td>12293</td>
<td>3074</td>
<td>123</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>37</td>
<td>145</td>
<td>3602</td>
<td>14406</td>
<td>360137</td>
<td>1440548</td>
<td>360137</td>
<td>14406</td>
<td>3602</td>
<td>145</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td>41</td>
<td>162</td>
<td>4034</td>
<td>16135</td>
<td>403354</td>
<td>1613413</td>
<td>403354</td>
<td>16135</td>
<td>4034</td>
<td>162</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>0.35</td>
<td>44</td>
<td>175</td>
<td>4370</td>
<td>17479</td>
<td>436966</td>
<td>1747864</td>
<td>436966</td>
<td>17479</td>
<td>4370</td>
<td>175</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>0.40</td>
<td>47</td>
<td>185</td>
<td>4610</td>
<td>18449</td>
<td>460976</td>
<td>1843901</td>
<td>460976</td>
<td>18449</td>
<td>4610</td>
<td>185</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>0.45</td>
<td>48</td>
<td>191</td>
<td>4754</td>
<td>19016</td>
<td>475381</td>
<td>1901523</td>
<td>475381</td>
<td>19016</td>
<td>4754</td>
<td>191</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>49</td>
<td>193</td>
<td>4802</td>
<td>19208</td>
<td>480183</td>
<td>1920760</td>
<td>480183</td>
<td>19208</td>
<td>4802</td>
<td>193</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>0.55</td>
<td>48</td>
<td>191</td>
<td>4754</td>
<td>19016</td>
<td>475381</td>
<td>1901523</td>
<td>475381</td>
<td>19016</td>
<td>4754</td>
<td>191</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>0.60</td>
<td>47</td>
<td>185</td>
<td>4610</td>
<td>18449</td>
<td>460976</td>
<td>1843901</td>
<td>460976</td>
<td>18449</td>
<td>4610</td>
<td>185</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>0.65</td>
<td>44</td>
<td>175</td>
<td>4370</td>
<td>17479</td>
<td>436966</td>
<td>1747864</td>
<td>436966</td>
<td>17479</td>
<td>4370</td>
<td>175</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>0.70</td>
<td>41</td>
<td>162</td>
<td>4034</td>
<td>16135</td>
<td>403354</td>
<td>1613413</td>
<td>403354</td>
<td>16135</td>
<td>4034</td>
<td>162</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>37</td>
<td>145</td>
<td>3602</td>
<td>14406</td>
<td>360137</td>
<td>1440548</td>
<td>360137</td>
<td>14406</td>
<td>3602</td>
<td>145</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>0.80</td>
<td>31</td>
<td>123</td>
<td>3074</td>
<td>12293</td>
<td>307317</td>
<td>12293</td>
<td>3074</td>
<td>123</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.85</td>
<td>25</td>
<td>98</td>
<td>2449</td>
<td>9796</td>
<td>244893</td>
<td>979572</td>
<td>244893</td>
<td>9796</td>
<td>2449</td>
<td>98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.90</td>
<td>18</td>
<td>70</td>
<td>1729</td>
<td>6915</td>
<td>172866</td>
<td>691463</td>
<td>172866</td>
<td>6915</td>
<td>1729</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.95</td>
<td>10</td>
<td>37</td>
<td>913</td>
<td>3650</td>
<td>91235</td>
<td>364939</td>
<td>91235</td>
<td>3650</td>
<td>913</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 Estimate a p-value by using the distribution function

Expression [7] gives a way to estimate the p-value. Computing the integral is not always possible but there is a way to do it using Monte Carlo method. Let’s assume \( X \sim \mathcal{N}(0,1) \). We denote \( f_X \) as the density function of \( X \). We also consider an interval \( I = [-a,a] \). Then we have \( f(a) = f(-a) \) and:

\[
\mathbb{P}(X \in I) = \mathbb{P}(|X| \leq a) = \mathbb{P}(f(X) \geq f(a)) \tag{12}
\]

This is true because \( f \) is decreasing for \( x > 0 \). The p-value \( \alpha \) for a estimator \( \beta \) using Monte Carlo method is:

\[
\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(f(X_i) < f(\beta)) \rightarrow \alpha \tag{13}
\]

Assuming every \( (X_i)_i \) follows a normal law \( \mathcal{N}(0,1) \).
6 Correlated variables

Let’s assume we now have a vector a correlated variables $X = (X_1,...X_d)$ drawn following a law $\mathcal{N}(\theta_0, \Sigma)$.

The central limit theorem is still valid:

$$\sqrt{N}\hat{\theta}_N \xrightarrow{N \to \infty} \mathcal{N}(\theta_0, \Sigma)$$

(14)

We know estimators for the average and the covariance matrix defined as follows:

$$\hat{\theta}_N = \frac{1}{n} \sum_{i=1}^{N} X_i$$

(15)

$$\hat{\Sigma}_N = \frac{1}{n} \sum_{i=1}^{N} (X_i - \hat{\theta}_N)(X_i - \hat{\theta}_N)'$$

(16)

We usually want to check if:

$$H_0 : \hat{\theta}_N = \theta_0$$

$$H_1 : \hat{\theta}_N \neq \theta_0$$

If $\Lambda$ is diagonal matrix of $\Sigma$ (diagonal matrix with eigen values). All eigenvalues are real and positive, we then define:

$$\Sigma = PA\Lambda P'$$

(17)

We consider $Z_i = (X_i - \hat{\theta}_N)\Sigma^{-\frac{1}{2}}$. We then have $\mathbb{E}(Z_i) = 0$ and $\mathbb{V}(Z_i) = I_2$ where $I_2$ is the identity matrix. We could now consider each dimension of $Z_i$ independently as illustrated in Figure 1: it shows the difference on an example if we consider the correlation of two variables correlated such as $

\begin{pmatrix}
0.1 & 0.05 \\
0.05 & 0.2
\end{pmatrix}

$. But that would not be the best way to do it. The confidence interval for a couple of indenpendant gaussian $(N_1, N_2)$ variables is an ellipse. Two independent normal $N_1^2 + N_2^2$ with a null mean and standard deviation equal to one follows a $\chi^2$ law. Based on that, we can deduce a boundary for the confidence zone at 95%. Figure 2 shows this zone for a non-correlated couple and a correlated couple $(\Sigma = \begin{pmatrix} 0.1 & 0.05 \\ 0.05 & 0.2 \end{pmatrix})$.

Correction

7 Program which produces the table in Section 4

```python
from scipy.stats import norm
```
**Figure 1:** We assume we observe two Bernoulli variables correlated. Red points represents the area for which we would accept hypothesis H0 in case both variables are independent. Blue area represents the same but with the correlation. See Section 8.

**Figure 2:** We assume we observe two Bernoulli variables correlated. Red points represents the area for which we would accept hypothesis H0 in case both variables are independent. Blue area represents the same but with the correlation. See Section 8.

```python
def pvalue(p, q, N):
    theta = abs(p-q)
    var = p*(1-p)
    bn = (2*N)**0.5 * theta / var**0.5
    ret = (1 - norm.cdf(bn))*2
    return ret

def pvalue_N(p, q, alpha):
    theta = abs(p-q)
    var = p*(1-p)
    rev = abs(norm.ppf(alpha/2))
    N = 2 * (rev * var**0.5 / theta)**2
    return int(N+1)

def alphatable(ps, dps, alpha):
    values = []
    for p in ps:
        row=[]
        for dp in dps:
            q = p+dp
            r = pvalue_N(p,q, alpha) if 1 >= q >= 0 else -1
            row.append(r)
        values.append(row)
    return values
```
values.append (row)
return values

def pprint(ps,dps,table, format, fileFormat = "latex") :
    text = []
    if fileFormat == "latex" :
        cc = "r" + ("r" * len(dps))
        text.append("\begin{tabular}{" + cc + "}")
        row = ["" ] + [ r"%1.3f" % _ for _ in dps ]
        text.append("%".join(row) + r"\\hline")
        for p,line in zip(ps,table) :
            row = [r"%1.2f" % p] + \
            [ format % _ if _ != -1 else "" for _ in line ]
            text.append("&".join(row) + r"\\hline")
        text.append("\end{tabular}")
        return "\n".join(text)
    else :
        # TSV
        row = ["" ] + [ r"%1.3f" % _ for _ in dps ]
        text.append("\t".join(row) )
        for p,line in zip(ps,table) :
            row = ["%1.2f" % p] + \
            [ format % _ if _ != -1 else "" for _ in line ]
            text.append("\t".join(row) )
        return "\n".join(text)

if __name__ == "__main__" :
    print ("norm.ppf(0.025)",norm.ppf (0.025)) # -1.9599639845400545
    ps = [0.001, 0.002] + [ 0.05*i for i in range (1,20) ]
    dps = [ -0.2, -0.1, -0.02, -0.01, -0.002, -0.001,
           0.2, 0.1, 0.02, 0.01, 0.002, 0.001, ]
    dps.sort()
    t = alphatable (ps, dps, 0.05)
    print (pprint (ps, dps, t, "%d", "latex"))
    print (pprint (ps, dps, t, "%d", "html"))

8  Program which generates Figure[1]

import numpy, matplotlib, random, pylab, math

def matrix_square_root(sigma) :
    eigen, vect = numpy.linalg.eig(sigma)
    dim = len(sigma)
    res = numpy.identity(dim)
    for i in range(0,dim) :
        res[i,i] = eigen[i]**0.5
    return vect * res * vect.transpose()

def chi2_level (alpha = 0.95) :
    N = 1000
    x = [ random.gauss(0,1) for _ in range(0,N) ]
    y = [ random.gauss(0,1) for _ in range(0,N) ]
    r = map ( lambda c : (c[0]**2+c[1]**2)**0.5, zip(x,y))
    r = list(r)
    r.sort()
    res = r [ int (alpha * N) ]
return res

def square_figure(mat, a):
    x = []
    y = []
    for i in range(0,100):
        x.append(a * mat[0][0]**0.5)
        y.append((random.random()-0.5) * a * mat[1][1]**0.5*2)
        x.append(-a * mat[0][0]**0.5)
        y.append((random.random()-0.5) * a * mat[1][1]**0.5*2)
        y.append(a * mat[1][1]**0.5)
        x.append((random.random()-0.5) * a * mat[0][0]**0.5*2)
        y.append(-a * mat[1][1]**0.5)
        x.append((random.random()-0.5) * a * mat[0][0]**0.5*2)
    pylab.plot(x,y, 'ro')

    x = []
    y = []
    for i in range(0,100):
        x.append(a)
        y.append((random.random()-0.5) * a*2)
        x.append(-a)
        y.append((random.random()-0.5) * a*2)
        y.append(a)
        x.append((random.random()-0.5) * a*2)
        y.append((-a)
        x.append((random.random()-0.5) * a*2)
    xs,ys = [],[]
    for a,b in zip(x,y):
        ar = numpy.matrix([[a], [b]]).transpose()
        we = ar * root
        xs.append(we[0,0])
        ys.append(we[0,1])
    pylab.plot(xs,ys, 'bo')
    pylab.show()

def circle_figure(mat, a):
    x = []
    y = []
    for i in range(0,200):
        z = random.random() * math.pi * 2
        i = a * mat[0][0]**0.5 * math.cos(z)
        j = a * mat[0][0]**0.5 * math.sin(z)
        x.append(i)
        y.append(j)
    pylab.plot(x,y, 'ro')

    x = []
    y = []
    for i in range(0,200):
        z = random.random() * math.pi * 2
        i = a * math.cos(z)
        j = a * math.sin(z)
        x.append(i)

y.append ( j )

xs,ys = [],[]
for a,b in zip (x,y):
    ar = numpy.matrix( [ [a], [b] ] ).transpose()
    we = ar * root
    xs.append ( we [0,0] )
    ys.append ( we [0,1] )

pylab.plot(xs,ys, 'bo')
pylab.show() 

if __name__ == '__main__' :
    level = chi2_level ()
    mat = [ [0.1, 0.05], [0.05, 0.2] ]
    npmat = numpy.matrix(mat)
    root = matrix_square_root (npmat)

    square_figure (mat, 1.96)
    circle_figure (mat, level)

9 Program which estimates the p-value

import random, math

def densite_gauss (mu, sigma, x) :
    e = -(x - mu)**2 / (sigma**2 * 2)
    d = 1. / ((2*math.pi)**0.5 * sigma)
    return d * math.exp (e)

def simulation_vector (N, mu, sigma) :
    return [ random.gauss(mu,sigma) for n in range(N) ]

def ratio (vector, x, fdensite) :
    under = 0
    above = 0
    fx = fdensite(x)
    for u in vector :
        f = fdensite (u)
        if f >= fx : above += 1
        else : under += 1
    return float(above) / float (above + under)

if __name__ == "__main__" :
    x   = 1.96
    N   = 10000
    mu  = 0
    sigma = 1

    v   = simulation_vector (N, mu, sigma)
    g   = ratio (v, x, lambda y : densite_gauss (mu, sigma, y) )
    print (g)