Q7

(a)

Let $\widehat{\pi} = \frac{\text{no.successes}}{\text{no.observations}}$ be an estimate of the true probability p in the Binomial trials $B(n, p)$.

If we let $\widehat{\pi_i}$ be the estimates of p_i at the temperatures at 10°, at 15° and at 20°, then $\widehat{\pi}_1 = 0/30$, $\widehat{\pi}_2 = 1/30$, and $\widehat{\pi}_3 = 25/30$. By the central limit theorem,

$$\frac{\widehat{\pi}_i - p_i}{\text{sd}(\widehat{\pi}_i - p)} \approx N(0,1)$$

So we choose those p_i for which the resulting value of LHS is between $z_{0.025}$ and $z_{0.975}$

$$\{p_i : z_{0.025} \le \frac{\widehat{\pi} - p_i}{\text{sd}(\widehat{\pi} - p_i)} \le z_{0.975}\}$$

this set is the confidence interval.

If we choose an estimate of $\text{sd}(\widehat{\pi}_i) = \widehat{\pi}_i(1-\widehat{\pi}_i)/n$, then we have the Wald C.I.

$$\widehat{\pi}_i - z_{0.975}\sqrt{\widehat{\pi}_i(1-\widehat{\pi}_i)/n} \le p \le \widehat{\pi}_i - z_{0.025}\sqrt{\widehat{\pi}_i(1-\widehat{\pi}_i)/n}$$

If we choose an estimate of $\text{sd}(\widehat{\pi}_i) = p_i(1-p_i)/n$, then we have the Wilson C.I.

$$\widetilde{\pi}_i \pm z_{0.975}\sqrt{n}/(n+z_{0.975}^2)\sqrt{\widehat{\pi}(1-\widehat{\pi}_i) + z_{0.975}^2/(4n)}$$

here

$$\widetilde{\pi} = \frac{\text{no. succ.} + z_{0.975}^2/2}{\text{no. trials} + z_{0.975}^2}$$

If we increase the numbers of sucesses and failures by two in the Wald C.I., then this is the A.C. CI.

```
In [34]:
```

```python
import scipy.stats as stats
import numpy as np
z1 = stats.norm.ppf(0.025)
z2 = stats.norm.ppf(0.975)
hat_pi = np.zeros(3)
tilde_pi = np.zeros(3)
print("Wald C.I.:")
hat_pi[0] = 0 / 30
hat_pi[1] = 1 / 30
hat_pi[2] = 25 / 30

for i in range(0,3):
    se = ( hat_pi[i] * ( 1 - hat_pi[i]) / 30)**(1/2)
    a = hat_pi[i] - z2 * se
    b = hat_pi[i] - z1 * se
    print( "[{0:6.3f}, {1:6.3f}] ".format(a,b))

print("Wilkson C.I.:")
tilde_pi[0] = (0  + z2**2/2) / (30 + z2**2) # See textbook p. 12
tilde_pi[1] = (1  + z2**2/2) / (30 + z2**2)
tilde_pi[2] = (25 + z2**2/2) / (30 + z2**2)

for i in range(0,3):
    a = tilde_pi[i] - z2 *30**(1/2) / (30 + z2**2) \
        * ( hat_pi[i] * ( 1 - hat_pi[i]) + z2**2 / (4*30) )**(1/2)
    b = tilde_pi[i] + z2 *30**(1/2)/ (30 + z2**2) \
        * ( hat_pi[i] * ( 1 - hat_pi[i]) + z2**2 / (4*30) )**(1/2)
    print( "[{0:6.3f}, {1:6.3f}] ".format(a,b))

print("A.G. C.I.:")



for i in range(0,3):
    se = ( tilde_pi[i] * ( 1 - tilde_pi[i]) / (30+z2**2))**(1/2)
    a = tilde_pi[i] - z2 * se
    b = tilde_pi[i] + z2 * se
    print( "[{0:6.3f}, {1:6.3f}] ".format(a,b))
```

```
Wald C.I.:
[ 0.000,  0.000]
[-0.031,  0.098]
[ 0.700,  0.967]
Wilkson C.I.:
[ 0.000,  0.114]
[ 0.006,  0.167]
[ 0.664,  0.927]
A.G. C.I.:
[-0.021,  0.135]
[-0.008,  0.181]
[ 0.660,  0.931]
```

(b) Since we can suppose the true probability are in each C.I., and the three Wilkson confidence intervals are non-overlapping, so the true probabilities are different.

Q8

(a) To account the variability of each experiment.

(b)

```
In [35]:
```

```
x = [ 1, 2, 4, 1, 0, 0, 0, 12, 0, 2]
CI = np.zeros((10,2))
for i in range(0,10) :
    count = x[i]
    tilde_pi = (count  + z2**2/2) / (30 + z2**2) # See textbook p. 12
    hat_pi = count / 30
    a = tilde_pi - z2 *30**(1/2) / (30 + z2**2) \
        * ( hat_pi * ( 1 - hat_pi) + z2**2 / (4*30) )**(1/2)
    b = tilde_pi + z2 *30**(1/2)/ (30 + z2**2) \
        * ( hat_pi * ( 1 - hat_pi) + z2**2 / (4*30) )**(1/2)
    CI[i,0] = a
    CI[i,1] = b
    print( "[{0:6.3f}, {1:6.3f}] ".format(a,b))
```

```
[ 0.006,  0.167]
[ 0.018,  0.213]
[ 0.053,  0.297]
[ 0.006,  0.167]
[ 0.000,  0.114]
[ 0.000,  0.114]
[ 0.000,  0.114]
[ 0.246,  0.577]
[ 0.000,  0.114]
[ 0.018,  0.213]
```

(c) The above confidence intervals are quit different, especially the 8th one. Maybe because each box has different conditions.

(d) The boxes could be put in different position resulting in different temperature even they are in the same chamber.

(e) From part (c) and part (d), the boxes are different; you can put the data together only if all the 10 boxes have exactly the same conditions, which is not.

Q 13

The likelihood ratio C.I. is

$$\{p : -2[\log(L(p)) - \log(L(\hat{p}))] < \chi^2_{0.975}\}$$

where $\hat{p} = \frac{\text{counts}}{n} = \frac{w}{n}$ and $\log L(p) = w \log p - (n-w)\log(1-p)$ Hence

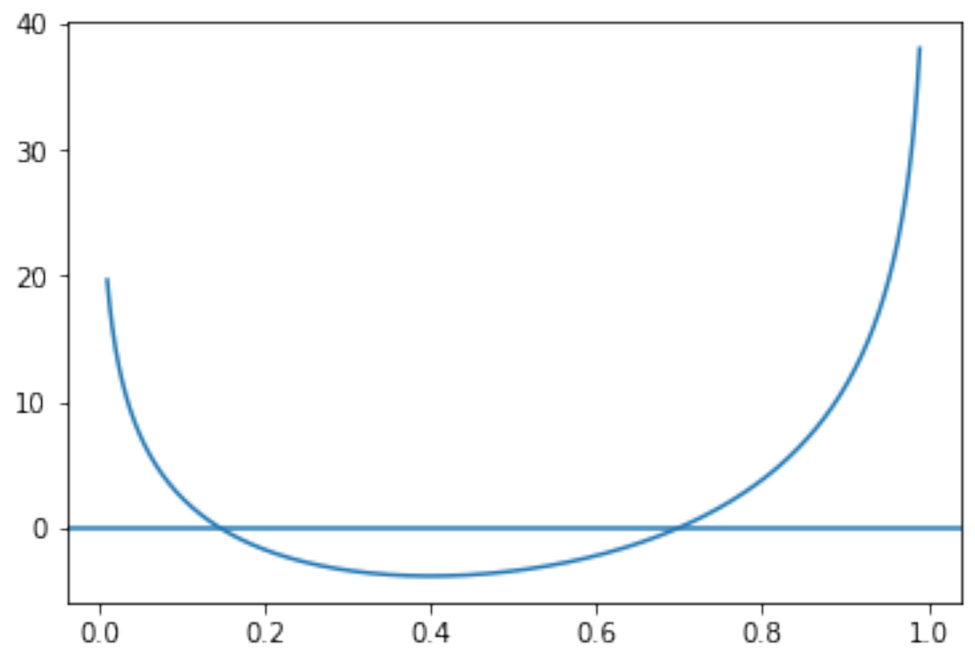
$$\{p : -2[w \log p + (n-w)\log(1-p) - \log(L(\hat{p}))] < \chi^2_{0.975}\}$$

Plug in $w = 4$ and $n = 10$,

$$-2[4\log p + 6\log(1-p) - 4\log .4 - 6\log(1-.4)] < \chi^2_{0.975}$$

In [36]:

```
p = np.linspace(0.01,0.99,1000)
y = -2 * ( 4 * np.log(p) + 6 * np.log(1-p) - (4 * np.log(0.4) + 6 * np.log(1-0.4
)) ) - 3.84
import matplotlib.pyplot as plt
plt.plot(p,y)
plt.axhline(0)
plt.show()
```



In [37]:

```
from scipy.optimize import root
p1 = root( lambda p: -2 * ( 4 * np.log(p) + 6 * np.log(1-p) - (4 * np.log(0.4) +
6 * np.log(1-0.4)) ) - 3.84, .2).x
p2 = root( lambda p: -2 * ( 4 * np.log(p) + 6 * np.log(1-p) - (4 * np.log(0.4) +
6 * np.log(1-0.4)) ) - 3.84, .7).x
print(p1, p2)
```

```
[ 0.14567762] [ 0.6999706]
```

The two intersections are $p1 = 0.14567762, p2 = 0.6999706$, so the confidence interval:

$$[0.14567762, 0.6999706]$$

(c) For fixed w (for example $w = 4$), and n (for example $n = 10$). The Wald confidence interval is

$$\frac{w}{n} - 1.96 \times [\frac{\frac{w}{n}(1 - \frac{w}{n})}{n}]^{1/2}, \frac{w}{n} + 1.96 \times [\frac{\frac{w}{n}(1 - \frac{w}{n})}{n}]^{1/2}$$

In [40]:

```
def ab(w, n):
    return  (w/n - 1.96 * ( (w / n) *   ( 1 - w / n ) / n ) ** (1/2), w/n + 1.96
* ( (w / n) *   ( 1 - w / n ) / n ) ** (1/2))
n = 10
p = 0.5
s = 0.0
import math
for w in range(0,n+1):
    L = ab(w,n)
    YES = (L[0] <= p) & (p <= L[1])
    p_w = math.factorial(n) / ( math.factorial(w) * math.factorial( n - w)) * p*
*w * (1-p) **(n-w)
    if YES:
        s = s+ p_w
    print('w = %2d, Pr( w | p = %.1f ) = %6.4f, CI(w) = [%.4f %.4f], %r' % (w, p
, p_w, L[0], L[1], YES))
print('C(p=%.1f) = %.4f' % (p,s) )
```

```
w =  0, Pr( w | p = 0.5 ) = 0.0010, CI(w) = [0.0000 0.0000], False
w =  1, Pr( w | p = 0.5 ) = 0.0098, CI(w) = [-0.0859 0.2859], False
w =  2, Pr( w | p = 0.5 ) = 0.0439, CI(w) = [-0.0479 0.4479], False
w =  3, Pr( w | p = 0.5 ) = 0.1172, CI(w) = [0.0160 0.5840], True
w =  4, Pr( w | p = 0.5 ) = 0.2051, CI(w) = [0.0964 0.7036], True
w =  5, Pr( w | p = 0.5 ) = 0.2461, CI(w) = [0.1901 0.8099], True
w =  6, Pr( w | p = 0.5 ) = 0.2051, CI(w) = [0.2964 0.9036], True
w =  7, Pr( w | p = 0.5 ) = 0.1172, CI(w) = [0.4160 0.9840], True
w =  8, Pr( w | p = 0.5 ) = 0.0439, CI(w) = [0.5521 1.0479], False
w =  9, Pr( w | p = 0.5 ) = 0.0098, CI(w) = [0.7141 1.0859], False
w = 10, Pr( w | p = 0.5 ) = 0.0010, CI(w) = [1.0000 1.0000], False
C(p=0.5) = 0.8906
```

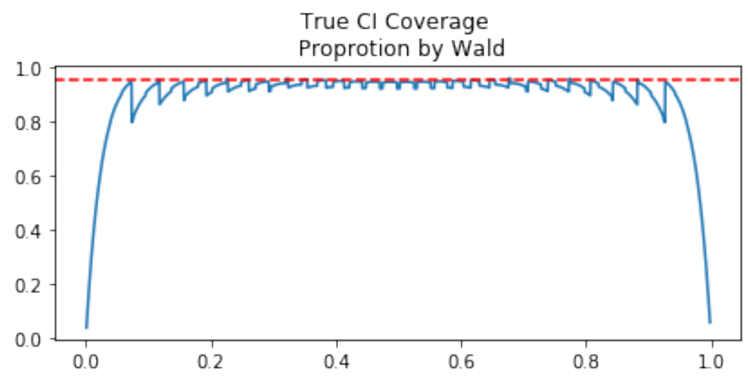
if p = 0.5, among the above 11 CIs, there are 5 containing p =0.5, i.e. they are w=3, 4, 5, 6, 7, and C(p=0.5) = Pr(3 | 0.5) + Pr(4 | 0.5) + Pr(5 | 0.5) + Pr(6 | 0.5) + Pr(7 | 0.5) = 0.1172 + 0.2051 +0.2461 + 0.2051 + 0.1172 = 0.8907

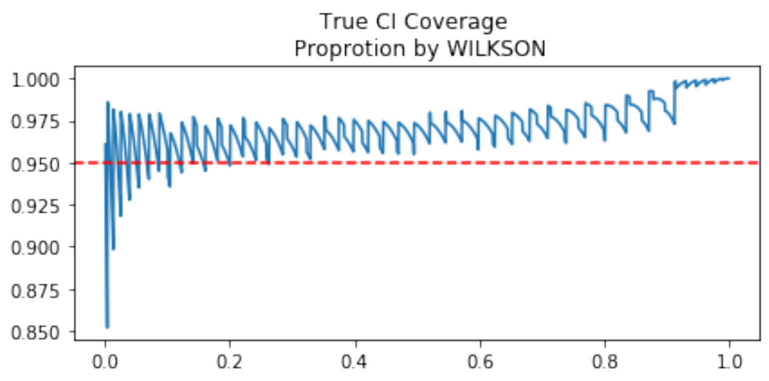
So C(p) is the long time proportion of the C.I.s which containing p. if we repeat the experiment infinitely.

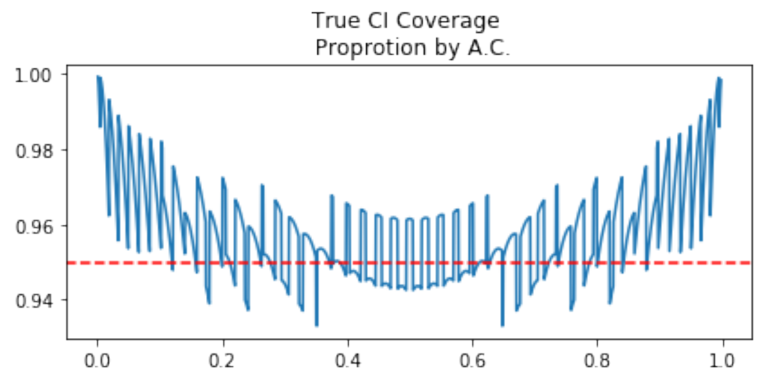
C(p) is a function of p, we can graph it.

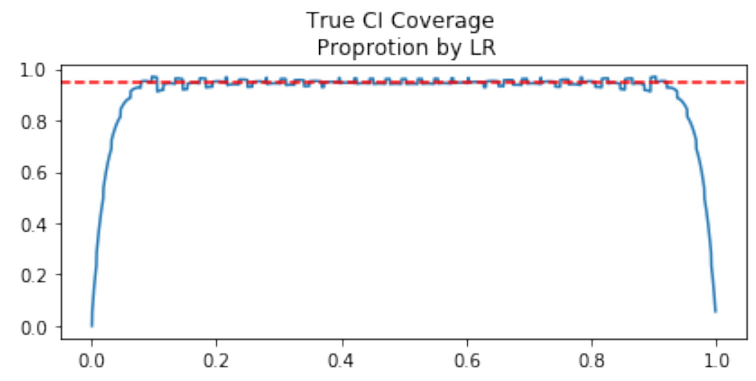
In [39]:

```
def CI_WALD(p,n=10):
    s = 0.0
    for w in range(0,n+1):
        L = (w/n - 1.96 * ( (w / n) *  ( 1 - w / n ) / n ) ** (1/2), w/n + 1.96
* ( (w / n) *  ( 1 - w / n ) / n ) ** (1/2))
        YES = (L[0] <= p) & (p <= L[1])
        p_w = math.factorial(n) / ( math.factorial(w) * math.factorial( n - w))
* p**w * (1-p) **(n-w)
        if YES:
            s = s+ p_w
    return s

def CI_LR(p, n):
    s = 0.0
    for w in range(0,n+1):
        p_w = math.factorial(n) / ( math.factorial(w) * math.factorial( n - w))
* p**w * (1-p) **(n-w)
        hat_p = w / n
        YES = -2 * ( w * np.log(p) + (n-w) * np.log(1-p) - (w * np.log(hat_p) +
(n-w) * np.log(1- hat_p)) ) < 3.84
        if YES:
            s += p_w
    return s

def CI_WILKSON(p, n):
    s = 0.
    for w in range(0,n+1):
        p_w = math.factorial(n) / ( math.factorial(w) * math.factorial( n - w))
* p**w * (1-p) **(n-w)
        z2  = 1.96

        hat_p = w / n
        tilde_p= (w  + z2**2/2) / (n + z2**2)

        a = tilde_p - z2 *n**(1/2) / (n + z2**2) \
            * ( hat_p * ( 1 - hat_p) + z2**2 / (4*n) )**(1/2)
        b = tilde_p + z2 *n**(1/2)/ (30 + z2**2) \
            * ( hat_p * ( 1 - hat_p) + z2**2 / (4*n) )**(1/2)

        YES = ( a <= p) & ( p <= b)
        if YES:
            s += p_w
    return s

def CI_AG(p, n):
    s = 0.
    for w in range(0,n+1):
        p_w = math.factorial(n) / ( math.factorial(w) * math.factorial( n - w))
* p**w * (1-p) **(n-w)
        z2  = 1.96

        hat_p = w / n
        tilde_p= (w  + z2**2/2) / (n + z2**2)
        se = ( tilde_p * ( 1 - tilde_p) / ( n + z2**2))**(1/2)
        a = tilde_p - z2 * se
        b = tilde_p + z2 * se

        YES = ( a <= p) & ( p <= b)
        if YES:
            s += p_w
    return s

t = np.arange(0.001,0.999,0.0005)
y1 = np.zeros(len(t))
y2 = np.zeros(len(t))
y3 = np.zeros(len(t))
y4 = np.zeros(len(t))
for i,p in enumerate(t):
    y1[i] = CI_WALD(p, n = 40)
    y2[i] = CI_WILKSON(p, n = 40)
    y3[i] = CI_AG(p, n = 40)
    y4[i] = CI_LR(p, n = 40)

fig, axes = plt.subplots(2,2)
ax1 = axes[0,0]
ax2 = axes[0,1]
ax3 = axes[1,0]
ax4 = axes[1,1]

ax1.plot(t,y1)
ax2.plot(t,y2)
ax3.plot(t,y3)
ax4.plot(t,y4)
fig.set_size_inches((12,6))
ax1.axhline(0.95, linestyle = 'dashed', color = 'red')
ax2.axhline(0.95, linestyle = 'dashed', color = 'red')
ax3.axhline(0.95, linestyle = 'dashed', color = 'red')
ax4.axhline(0.95, linestyle = 'dashed', color = 'red')
ax4.set_title('True CI Coverage \n Proprotion by LR')
ax1.set_title('True CI Coverage \n Proprotion by Wald')
ax2.set_title('True CI Coverage \n Proprotion by WILKSON')
ax3.set_title('True CI Coverage \n Proprotion by A.C.')
fig.tight_layout()
plt.show()
```









From the above, we see that LR is the best relatively. In general, Wilkson and LR are similar when $0.2 < \pi < 0.8$, and Wikson perferms better when π is close to the end points.