ADVANCED STATISTICS

#confidence interval st.t.interval(alpha,length/dof,loc,scale) where: alpha: probability that an RV will be drawn from the returned range length: length of the data set loc: location parameter scale: scale parameter

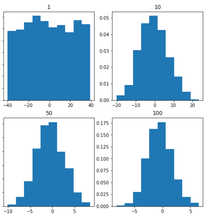
In [2]:
```
#confidence interval
import numpy as np
import scipy.stats as st
data=[1,1,2,3,4,5,3,5,3,6,3,7,9,3],4,5,1,5,29,9]
st.t.interval(alpha=0.90,df=len(data)-1,loc=np.mean(data),scale=st.sem(data))
```

Out[2]:
```
(3.581288133621515, 10.218711866378488)
```

In [6]:
```
#confidence Interval at normal distribution
import numpy as np
import scipy.stats as st
#data=[1,1,2,3,4,5,3,5,3,6,3,7,9,33],4,5,1,5,29,9]
#st.t.interval(alpha=0.90,df=len(data)-1,loc=np.mean(data),scale=st.sem(data))
data=np.random.randint(5,10,100)
st.norm.interval(alpha=0.90,loc=np.mean(data),scale=st.sem(data))
```

Out[6]:
```
(6.739204827651076, 7.200795172348924)
```

In [11]:
```
#center limit theorem
import numpy as np
import matplotlib.pyplot as pt
num=[1,10,50,100]
means=[]
for j in num:
    np.random.seed(1)
    x=[np.mean(np.random.randint(-40,40,j)) for i in range(1000)]
    means.append(x)
    k=0
fig,axrpt.subplots(2,2,figsize=(8,8))
for i in range(0,2):
    for j in range(0,2):
        ax[i,j].hist(means[k],10,density=True)
        ax[i,j].set_title(label=num[k])
        k+=1
pt.show()
```



In [25]:
```
#standard error
from scipy.stats import sem
#import scipy.stats as sem
data=[3,2,5,4,29,49,299,393,20,302,49,50,2994,204,20,492,103,299,402,29]
print(sem(data))
```

```
146.68603958971208
```

In [27]:
```
import numpy as np
data=np.arange(20,40,2)
print(data)
np.std(data,ddof=1)/np.sqrt(np.size(data))
```

```
[20 22 24 26 28 30 32 34 36 38]
1.9148542155126762
```

Out[27]:

In [28]:
```
#p-value
#one tailed test
import scipy.stats as st
st.t.sf(abs(-.47),df=12)
```

Out[28]:
```
0.3233906798700002
```

In [30]:
```
#right_tailed test
import scipy.stats as st
st.t.sf(abs(1.87),df=24)
```

Out[30]:
```
0.03686532838323424
```

In [31]:
```
#two tailed test
import scipy.stats as st
st.t.sf(abs(1.36),df=319)*2
```

Out[31]:
```
0.18304931466593782
```

In [ ]:

In [34]:
```
#one sample mean
from statsmodels.stats.weightstats import ztest
pre=[30,31,34,40,36,35,34,30,28,29]
ztest(pre,value=40)
```

Out[34]:
```
(-6.167179251545867, 6.951893640280303e-10)
```

In [25]:
```
#two sample mean
#import statsmodels.stats.weightstats as ztest
from statsmodels.stats.weightstats import ztest
pre=[30,31,34,40,36,35,34,30,28,29]
post=[30,31,32,38,32,31,32,29,28,30]
ztest(pre,post,value=0)
```

Out[25]:
```
(0.9580156390269369, 0.3380548725619153)
```

In [35]:
```
#one proportional z_test using numpy
import math as mt
p=0.66
p0=0.80
n=100
a=(p-p0)
b=p0*(1-p0)/n
c=a/mt.sqrt(b)
print(z)
```

```
1.4999999999999984
```

In [41]:
```
from statsmodels.stats.proportion import proportions_ztest
proportions_ztest(counts=60,nobs=100,value=0.66)
```

Out[41]:
```
(-0.8164965809277268, 0.41421617824252466)
```

In [49]:
```
#two proportional z_test
import numpy as np
from statsmodels.stats.proportion import proportions_ztest
a=np.array([812,1021])
b=np.array([812*28,1021*190])
z_test,p_value=proportions_ztest(count=a,nobs=b,alternative="two-sided")
print("Z_test=",round(z_test,4),"P_value=",p_value)
```

```
Z_test= -4.2237 P_value= 2.4033301426853568e-05
```

In [3]:
```
#Z_TEST
import math
import numpy as np
from numpy.random import randn
from statsmodels.stats.weightstats import ztest
sample=20
mean=50
alpha=0.05
standard=8.5
data=standard*randn(20)+sample
print('mean=%.2f std=%.2f' % (np.mean(data),np.std(data)))
print("Z_test=",mean,"p_value=",standard)
```

```
mean=-20.09 std=7.69
Z_test= 50 p_value= 8.5
```

In [4]:
```
import math
import numpy as np
from numpy.random import randn
from statsmodels.stats.weightstats import ztest
sample=20
mean=50
alpha=0.05
standard=8.5
data=sample-mean/standard
print('mean=%.2f std=%.2f' % (np.mean(data),np.std(data)))
print("Z_test=",mean,"p_value=",standard)
```

```
mean=-14.12 std=0.00
Z_test= 50 p_value= 8.5
```

In [7]:
```
#one sample test
import scipy.stats as stats
import pandas as pd
Age_Google_India=[20,43,29,33,27,45,37,33,29,22,23,41,32]
z_statistic,p_value=stats.ttest_1samp(a=Age_Google_India,popmean=24)
print("Z_test=",z_statistic,"P_value=",round(p_value,4))
```

```
Z_test= 3.5415385751139955 P_value= 0.0041
```

In [58]:
```
import scipy.stats as stats
import pandas as pd
data=[20,19,30,49,30,20,30,40,39,30,49,49,39,47,30,37,28,50]
t_statistic,p_value=stats.ttest_1samp(data,popmean=4)
print('T_test=',round(t_statistic,2),'P_value=',round(p_value,2))
```

```
T_test= 8.22 P_value= 0.0
```

In [8]:
```
#two sample mean
import numpy as np
import scipy.stats as stats
data1=np.array([14,15,15,16,13,8,14,17,16,1,19,20,21,15,158,16,16,13,14,12])
data2=np.array([28,39,29,40,10,4,20,10,30,9,30,23,23,48,399,30,20,29,29,39])
#print(data1)
#print(data2)
print()
s_statistic,p_value=stats.ttest_ind(a=data1,b=data2,equal_var=True)
print('T_test=',round(s_statistic,3),'p_value=',round(p_value,3))
```

```
T_test= -1.13 p_value= 0.266
```

In [9]:
```
#propositional sample mean
import scipy.stats as stats
pre=[30,31,34,40,36,35,34,30,28,29]
post=[30,31,32,38,32,31,32,29,28,30]
z_sta,p_value=stats.ttest_rel(pre,post)
print('z_test=',round(z_sta,3),'p_value=',round(p_value,3))
```

```
z_test= 2.585 p_value= 0.029
```

In [16]:
```
#import math
#import numpy as np
#from numpy.random import randn
from statsmodels.stats.weightstats import ztest as ztest
pre=[30,31,34,40,36,35,34,30,28,29]
post=[30,31,32,38,32,31,32,29,28,30]
ztest(pre,post,value=0)
```

Out[16]:
```
(0.9580156390269369, 0.3380548725619153)
```

In [51]:
```
#t-test
g1=np.array([14,15,15,16,13,8,14,17,16,14,19,20,21,15,15,16,16,13,14,12])
g2=np.array([15,17,14,17,14,8,8,12,19,17,22,24,16,13,16,13,18,19,15,13])
```

In [57]:
```
#one sample t-test
import scipy.stats as stats
t_test,p_value=stats.ttest_1samp(a=g1,popmean=19)
print("T_test=",round(t_test,4),"P_value=",p_value)
```

```
T_test= -6.037 P_value= 8.302460899806943e-06
```

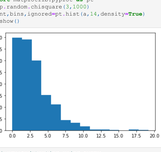
In [60]:
```
#two sample t_test
import scipy.stats as stats
t_test,p_value=stats.ttest_ind(a=g1,b=g2,equal_var=True)
print('T_test=',round(t_test,3),'P_value=',round(p_value,3))
```

```
T_test= -0.828 P_value= 0.413
```

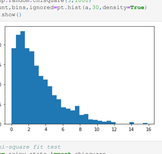
In [61]:
```
#paired t_test
import scipy.stats as stats
t_test,p_value=stats.ttest_rel(g1,g2)
print('T_test=',round(t_test,3),'P_value=',round(p_value,3))
```

```
T_test= -1.548 P_value= 0.138
```

In [62]:
```
#chi-squared testing using numpy
import numpy as np
import matplotlib.pyplot as pt
a=np.random.chisquare(1,1000)
count,bins,ignored=pt.hist(a,14,density=True)
pt.show()
```



In [63]:
```
#chi-squared testing using numpy
import numpy as np
import matplotlib.pyplot as pt
a=np.random.chisquare(1,1000)
count,bins,ignored=pt.hist(a,30,density=True)
pt.show()
```



In [67]:
```
#chi-square fit test
from scipy.stats import chisquare
chi1_test,p_value=chisquare([16,18,16,14,12,12])
print('chi_test=',round(chi1_test,2),'p_value=',round(p_value,3))
```

```
chi_test= 2.0 p_value= 0.849
```

In [74]:
```
#chi_square test of independence
data=[[120,90,60],[110,95,45]]
import scipy.stats as stats
stats.chi2_contingency(data)
```

Out[74]:
```
(0.8640353390889618,
 0.6493978887389376,
 2,
 array([[115. ,  92.5,  42.5],
        [115. ,  92.5,  42.5]]))
```

In [76]:
```
from scipy.stats import f_oneway
p1=[34,29,49,30,20,48,20,39]
p2=[30,49,30,49,30,84,28,93]
p3=[48,83,28,58,68,90,62,28]
f_value,p_value=f_oneway(p1,p2,p3)
print('F_value=',round(f_value,3),'P_value=',round(p_value,4))
```

```
F_value= 3.488 P_value= 0.0502
```

In [1]:
```
#two way anova
import numpy as np
import pandas as pd
df=pd.DataFrame({'Water':np.repeat(['daily','weekly'],15),
                 'Sun':np.tile(np.repeat(['low','med','high'],15),2),
                 'Weight': [6,6,6,5,6,5,5,6,4,5,6,6,7,8,7,3,4,4,4,5,4,4,4,4,4,5,6,6,7,8])
```

```
df[:10]
```

```
  Input In [1]
    'Weight': [6,6,6,5,6,5,5,6,4,5,6,6,7,8,7,3,4,4,4,5,4,4,4,4,4,5,6,6,7,8])}
                                                                             ^
SyntaxError: invalid syntax
```

In [ ]:

In [ ]:

In [ ]: