In [1]:

```
#            Science of Decisions
#     Interactive Stroop Effect Experiment

#     Analysis Results for Udacity P1 Study

# Prepared by:  Laurie S. Reynolds
```

In [2]:

```
# Read in the data and display as a table


# Some initial imports and settings
%matplotlib inline

from IPython.display import display, HTML
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import math as math
import scipy.stats as stats

np.set_printoptions(suppress=True)

# Read in the data and display as a table
stroopData = pd.read_csv('/home/laurie/udacity/P1/stroopdata-differences.csv')

stroopData
```

Out[2]:

	Participant	Congruent	Incongruent	Difference
0	P-100	12.079	19.278	7.199
1	P-101	16.791	18.741	1.950
2	P-102	9.564	21.214	11.650
3	P-103	8.630	15.687	7.057
4	P-104	14.669	22.803	8.134
5	P-105	12.238	20.878	8.640
6	P-106	14.692	24.572	9.880
7	P-107	8.987	17.394	8.407
8	P-108	9.401	20.762	11.361
9	P-109	14.480	26.282	11.802
10	P-110	22.328	24.524	2.196
11	P-111	15.298	18.644	3.346
12	P-112	15.073	17.510	2.437
13	P-113	16.929	20.330	3.401
14	P-114	18.200	35.255	17.055
15	P-115	12.130	22.158	10.028
16	P-116	18.495	25.139	6.644
17	P-117	10.639	20.429	9.790
18	P-118	11.344	17.425	6.081
19	P-119	12.369	34.288	21.919
20	P-120	12.944	23.894	10.950
21	P-121	14.233	17.960	3.727
22	P-122	19.710	22.058	2.348
23	P-123	16.004	21.157	5.153

In [7]:

```
# Data is not displaying correctly as numbers, verify check the
data was imported as floats
stroopData.dtypes
```

Out[7]:

```
Participant     object
Congruent      float64
Incongruent    float64
Difference     float64
dtype: object
```

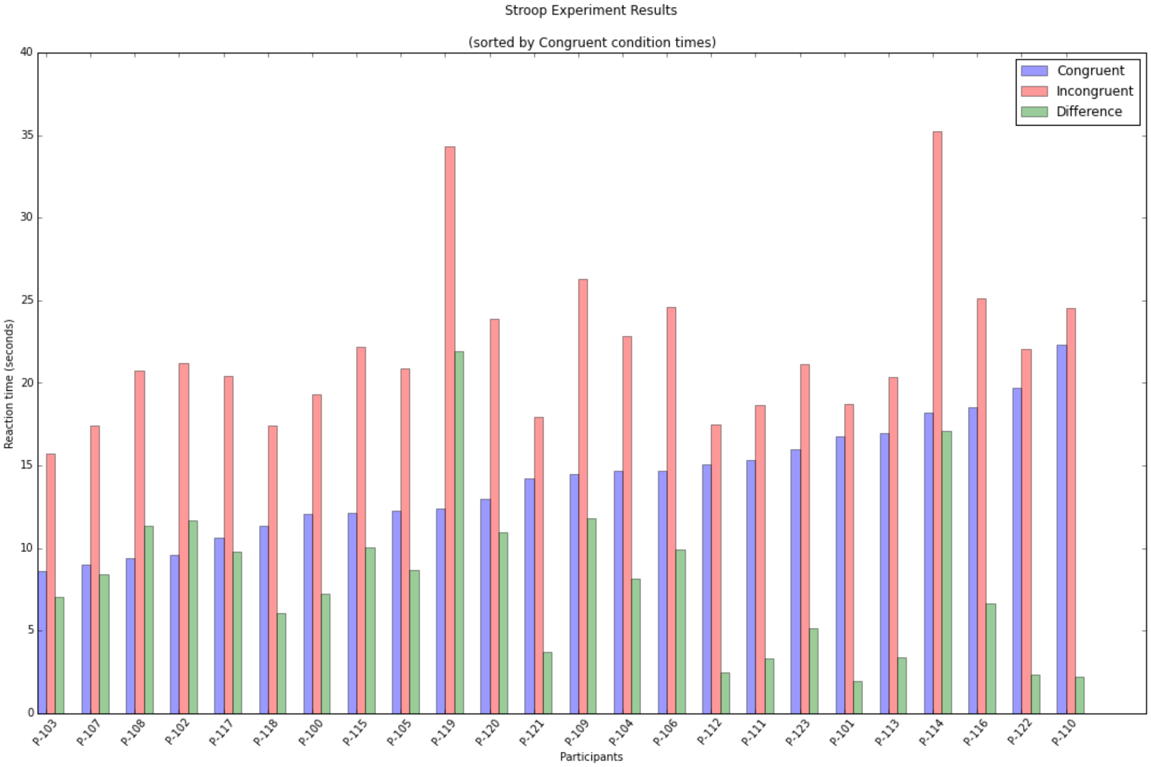
In [10]:

```
# Calculate the percentage difference

stroopData['Percent Diff'] = (stroopData['Difference'] / stroopD
ata['Congruent'] ) * 100

# display the data formatted correctly
print (stroopData.to_string (justify='right', index=False))
```

```
Participant  Congruent  Incongruent  Difference  Percent Diff
      P-100     12.079       19.278       7.199     59.599305
      P-101     16.791       18.741       1.950     11.613364
      P-102      9.564       21.214      11.650    121.810958
      P-103      8.630       15.687       7.057     81.772885
      P-104     14.669       22.803       8.134     55.450269
      P-105     12.238       20.878       8.640     70.599771
      P-106     14.692       24.572       9.880     67.247482
      P-107      8.987       17.394       8.407     93.546233
      P-108      9.401       20.762      11.361    120.848846
      P-109     14.480       26.282      11.802     81.505525
      P-110     22.328       24.524       2.196      9.835185
      P-111     15.298       18.644       3.346     21.872140
      P-112     15.073       17.510       2.437     16.167982
      P-113     16.929       20.330       3.401     20.089787
      P-114     18.200       35.255      17.055     93.708791
      P-115     12.130       22.158      10.028     82.671063
      P-116     18.495       25.139       6.644     35.923222
      P-117     10.639       20.429       9.790     92.019927
      P-118     11.344       17.425       6.081     53.605430
      P-119     12.369       34.288      21.919    177.209152
      P-120     12.944       23.894      10.950     84.595179
      P-121     14.233       17.960       3.727     26.185625
      P-122     19.710       22.058       2.348     11.912735
      P-123     16.004       21.157       5.153     32.198200
```

In [13]:

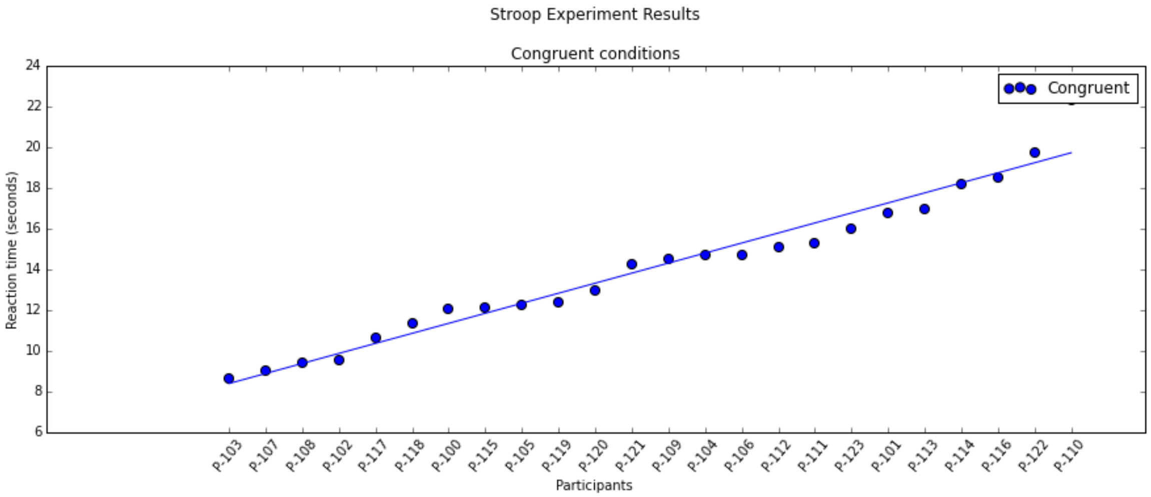
```
# Sort the data by Congruent column
#stroopSorted = stroopData.sort_values(['Congruent', 'Incongruen
t', 'Difference', 'Participant', 'Percent Diff'],  ascending=[1,
0, 0, 0,0])
n_groups = 24

stroopSorted = stroopData.sort_values(by='Congruent')
stroopSorted.index = range(0,len(stroopSorted))

fig, ax = plt.subplots(1,1,figsize=(15,10))

index = np.arange(n_groups)
bar_width = 0.2

opacity = 0.4
error_config = {'ecolor': '0.3'}

rects1 = plt.bar(index, stroopSorted['Congruent'], bar_width,
                 alpha=opacity,
                 color='b',
                 label='Congruent')

rects2 = plt.bar(index + bar_width, stroopSorted['Incongruent'],
bar_width,
                 alpha=opacity,
                 color='r',
                 label='Incongruent')

rects2 = plt.bar(index + 2*bar_width, stroopSorted['Differenc
e'], bar_width,
                 alpha=opacity,
                 color='g',
                 label='Difference')



plt.xlabel('Participants')
plt.ylabel('Reaction time (seconds)')
plt.title('Stroop Experiment Results\n\n(sorted by Congruent con
dition times)')
plt.xticks(index + bar_width, stroopSorted['Participant'])
plt.xticks(rotation=50)
plt.legend()

plt.tight_layout()
plt.show()
```


Stroop Experiment Results
(sorted by Congruent condition times)
Reaction time (seconds)
0
5
10
15
20
25
30
35
40
Congruent
Incongruent
Difference
P:103
P:107
P:108
P:102
P:117
P:118
P:100
P:115
P:105
P:119
P:120
P:121
P:109
P:104
P:106
P:112
P:111
P:123
P:101
P:113
P:114
P:116
P:122
P:110
Participants

In [14]:

```
fig, ax = plt.subplots(1,1,figsize=(15,5))

plt.scatter(index, stroopSorted['Congruent'], s=50, c='blue', la
bel='Congruent')

m, b = np.polyfit(index, stroopSorted['Congruent'], 1)

plt.plot(index, m*index + b, '-')


plt.title('Stroop Experiment Results\n\nCongruent conditions')

plt.xticks(index, stroopSorted['Participant'])
plt.xticks(rotation=50)

plt.xlabel('Participants')
plt.ylabel('Reaction time (seconds)')

plt.legend()

plt.show()
```

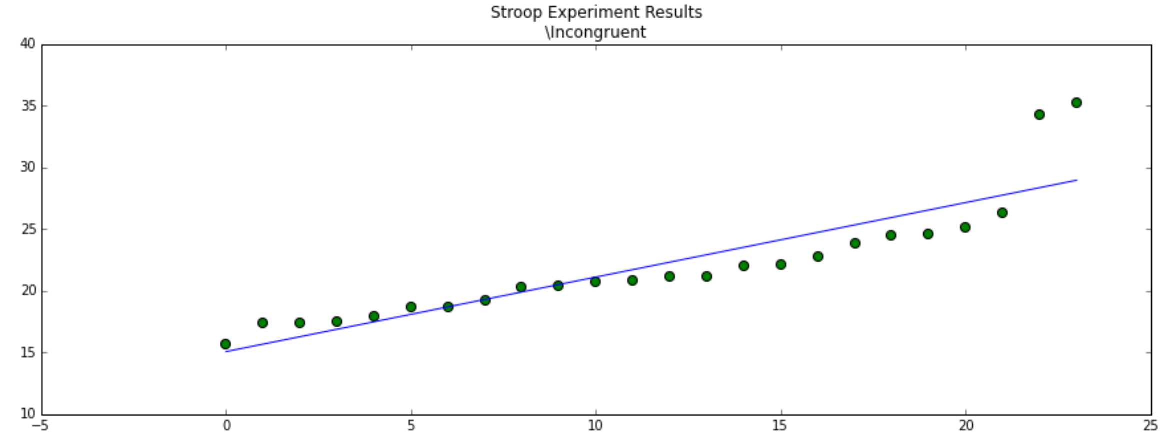
```
/usr/local/bin/anaconda2/lib/python2.7/site-packages/matplotli
b/collections.py:590: FutureWarning: elementwise comparison fail
ed; returning scalar instead, but in the future will perform ele
mentwise comparison
  if self._edgecolors == str('face'):
```



In [15]:

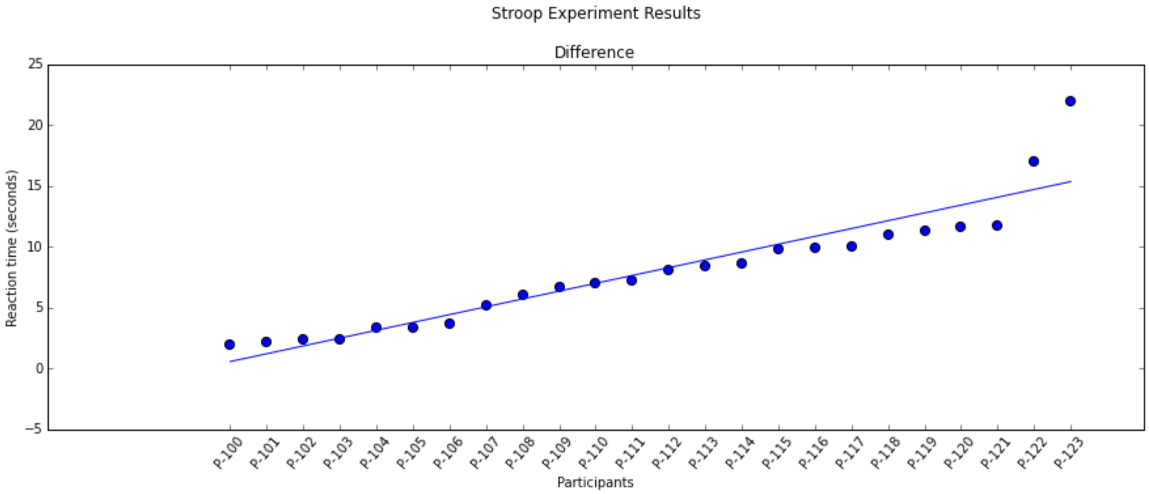
```
fig, ax = plt.subplots(1,1,figsize=(15,5))

stroopSortedIncongruent = stroopData.sort_values(['Incongruent',
'Congruent', 'Difference', 'Participant', 'Percent Diff'],  asce
nding=[1, 0, 0, 0,0])
n_groups = 24

mI, bI = np.polyfit(index, stroopSortedIncongruent['Incongruen
t'], 1)

plt.plot(index, mI*index + bI, '-')

plt.scatter(index, stroopSortedIncongruent['Incongruent'], s=50,
c='green', label='Incongruent')

plt.title('Stroop Experiment Results\n\Incongruent')



plt.xticks(index, stroopSortedDifference['Participant'])
plt.xticks(rotation=50)

plt.xlabel('Participants')
plt.ylabel('Reaction time (seconds)')

plt.show()
```

```
---------------------------------------------------------------------
-----------
NameError                                 Traceback (most recent
call last)
<ipython-input-15-91aece9ff0c7> in <module>()
     13
     14
---> 15 plt.xticks(index, stroopSortedDifference['Participant'])
     16 plt.xticks(rotation=50)
     17

NameError: name 'stroopSortedDifference' is not defined
```



In [16]:

```
fig, ax = plt.subplots(1,1,figsize=(15,5))

stroopSortedDifference = stroopData.sort_values(['Difference',
'Congruent', 'Incongruent', 'Participant', 'Percent Diff'],  asc
ending=[1, 0, 0, 0,0])
n_groups = 24

m, b = np.polyfit(index, stroopSortedDifference['Difference'],
1)

plt.plot(index, m*index + b, '-')


plt.scatter(index, stroopSortedDifference['Difference'], s=50,
c='blue')

plt.title('Stroop Experiment Results\n\nDifference')


plt.xticks(index, stroopSortedDifference['Participant'])
plt.xticks(rotation=50)

plt.xlabel('Participants')
plt.ylabel('Reaction time (seconds)')

plt.show()
```



In [17]:

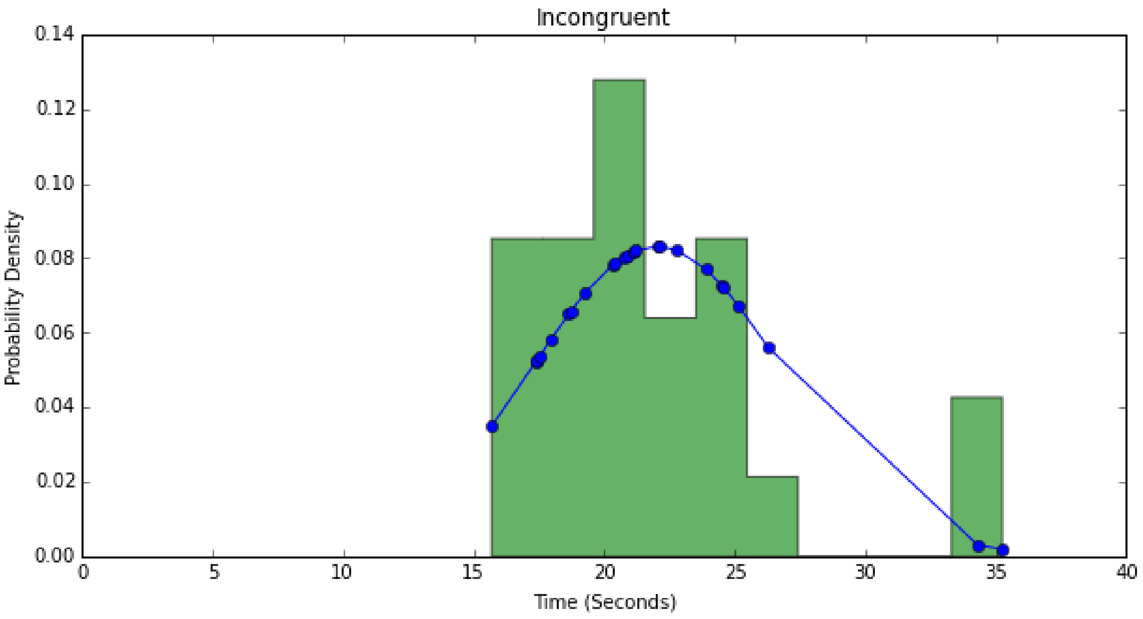
```
# Cacluatle descriptive statistics

# Assign column data to variables
participantCol = stroopData['Participant']
incongruentCol = stroopData['Incongruent']
differenceCol = stroopData['Difference']
congruentCol = stroopData['Congruent']
percentDiffCol = stroopData['Percent Diff']

descriptiveStats = pd.DataFrame( columns=['Congruent','Incongruent','Difference', 'Percent Diff'], index=['count', 'mean', 'median', 'min', 'max', 'std', 'se'])


# Calculate statistics
def calcStats(columnName, columnData) :

    descriptiveStats[columnName]['count'] = columnData.count()
    descriptiveStats[columnName]['mean'] = columnData.mean()
    descriptiveStats[columnName]['median'] = columnData.median()
    descriptiveStats[columnName]['min'] = columnData.min()
    descriptiveStats[columnName]['max'] = columnData.max()
    descriptiveStats[columnName]['std'] = stats.tstd(columnData)
    descriptiveStats[columnName]['se'] = stats.sem(columnData)


# Fill in the table
calcStats('Congruent', congruentCol)
calcStats('Incongruent', incongruentCol)
calcStats('Difference', differenceCol)
calcStats('Percent Diff', percentDiffCol)


# Print out the results
descriptiveStats
```

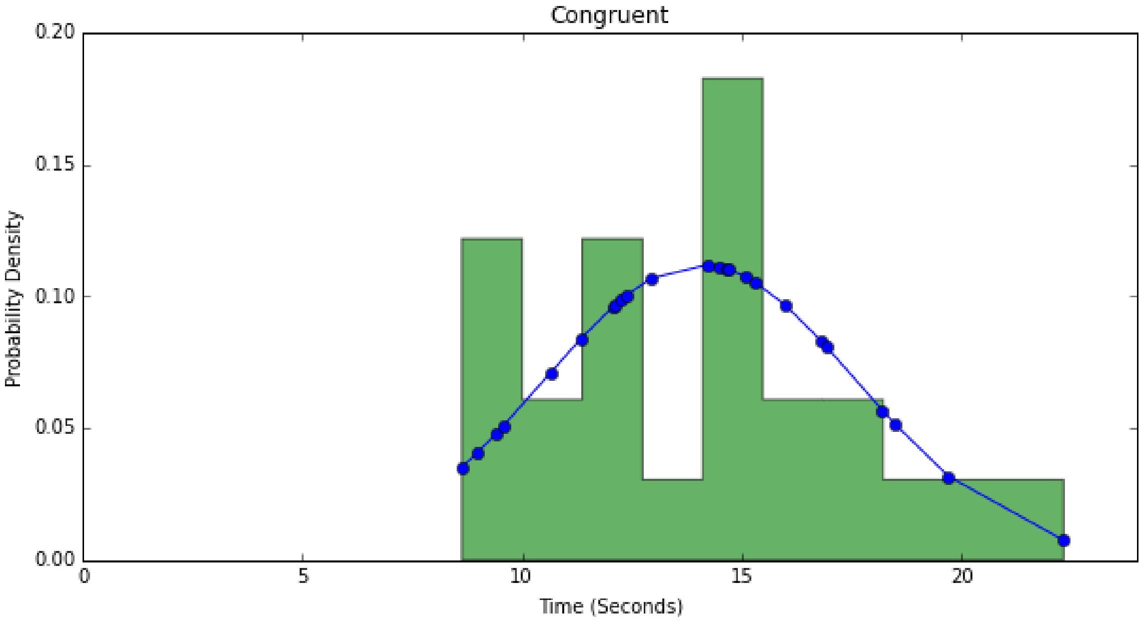
Out[17]:

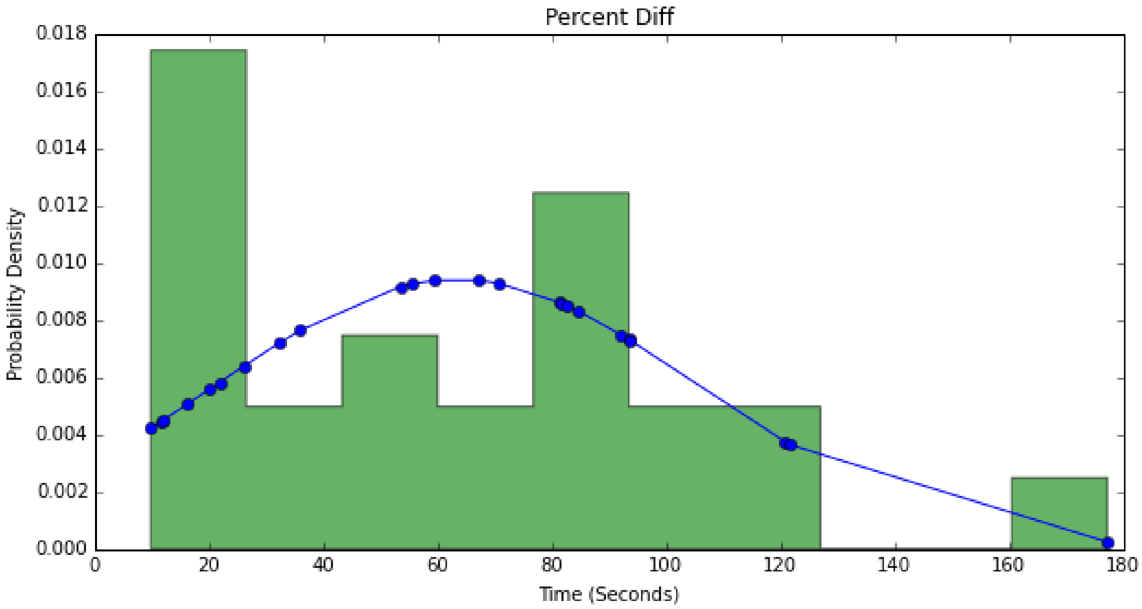
	Congruent	**Incongruent**	**Difference**	**Percent Diff**
count	24	24	24	24
mean	14.0511	22.0159	7.96479	63.4162
median	14.3565	21.0175	7.6665	63.4234
min	8.63	15.687	1.95	9.83518
max	22.328	35.255	21.919	177.209
std	3.55936	4.79706	4.86483	42.3156
se	0.726551	0.979195	0.993029	8.63764

In [65]:

```
def drawPlot(columnData, colIndex, rowIndex1, rowIndex2, title):
    sortedData = sorted(columnData)

    fit = stats.norm.pdf(sortedData, descriptiveStats[colIndex]
[rowIndex1], descriptiveStats[colIndex][rowIndex2])  #this is a
fitting indeed
    fig, ax = plt.subplots(1,1,figsize=(10,5))
    plt.plot(sortedData,fit,'-o')

    ax.set_xlim(xmin=0)

    plt.hist(sortedData,normed=True, histtype='stepfilled', alph
a=0.6)
    plt.xlabel('Time (Seconds)')
    plt.ylabel('Probability Density')
    plt.title(title)

    return {'fig':fig, 'ax':ax}



drawPlot(incongruentCol, 'Incongruent', 'mean', 'std', 'Incongru
ent')
drawPlot(congruentCol, 'Congruent', 'mean', 'std', 'Congruent')
drawPlot(percentDiffCol, 'Percent Diff', 'mean', 'std', 'Percent
Diff')
plt.show()
```

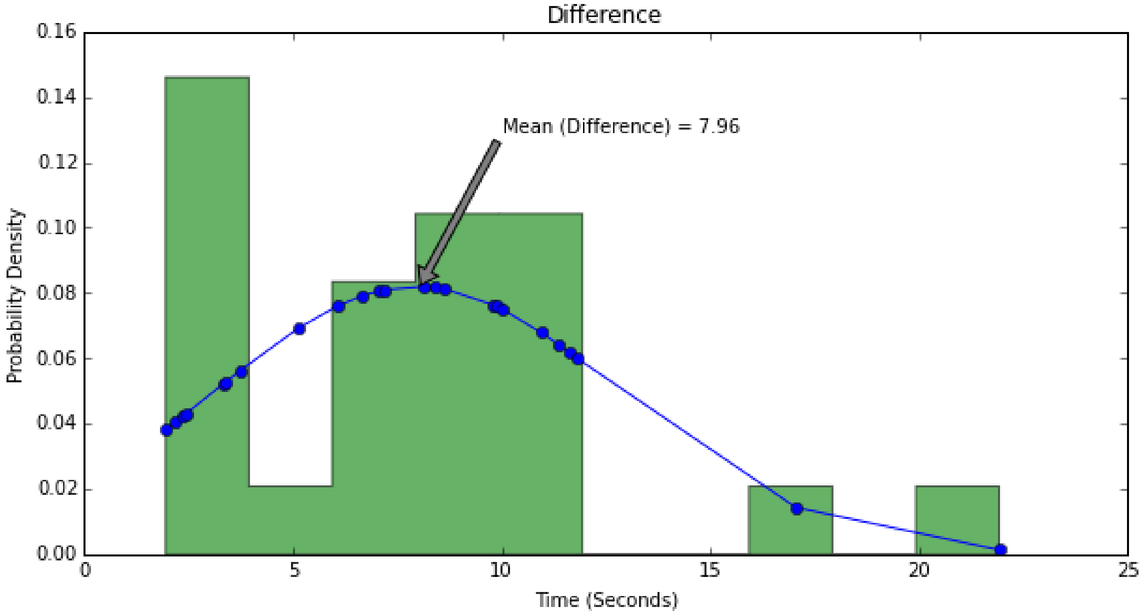

Incongruent
Probability Density
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
30
35
40
Time (Seconds)


Congruent
Probability Density
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
Time (Seconds)



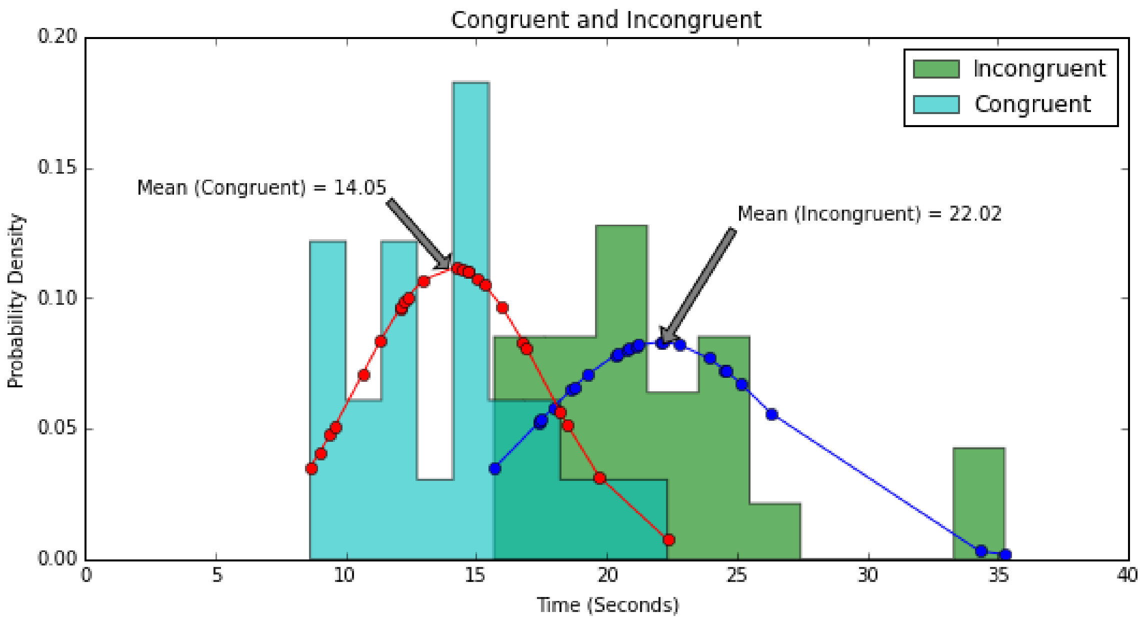
In [70]:

```
result = drawPlot(differenceCol, 'Difference', 'mean', 'std', 'D
ifference')
result['ax'].annotate('Mean (Difference) = 7.96', xy=(7.96, 0.0
8), xytext=(10,0.13),
            arrowprops=dict(facecolor='grey', shrink=0.05))
plt.show()
```



In [19]:

```
 fig, ax = plt.subplots(1,1,figsize=(10,5))

def drawPlot(columnData, colIndex, rowIndex1, rowIndex2, labelName):
    sortedData = sorted(columnData)

    fit = stats.norm.pdf(sortedData, descriptiveStats[colIndex][rowIndex1], descriptiveStats[colIndex][rowIndex2])  #this is a fitting indeed

    plt.plot(sortedData,fit,'-o')

    ax.set_xlim(xmin=0)

    plt.hist(sortedData,normed=True, histtype='stepfilled', alpha=0.6, label=labelName)
    plt.xlabel('Time (Seconds)')
    plt.ylabel('Probability Density')

ax.annotate('Mean (Congruent) = 14.05', xy=(14.051, 0.11), xytext=(2.0,0.14),
            arrowprops=dict(facecolor='grey', shrink=0.05))

ax.annotate('Mean (Incongruent) = 22.02', xy=(22.0159, 0.08), xytext=(25.0,0.13),
            arrowprops=dict(facecolor='grey', shrink=0.05))
plt.title('Congruent and Incongruent')
drawPlot(incongruentCol, 'Incongruent', 'mean', 'std', 'Incongruent')
drawPlot(congruentCol, 'Congruent', 'mean', 'std', 'Congruent')

plt.legend()
plt.show()
```



In [20]:

```
# Calculate the t statistic

tstatistic, pvalue = stats.ttest_rel(incongruentCol,congruentCol)

# TTest returns two tailed pvalue - divide in half for a one tailed test
pvalue = pvalue /2

print "T Statistic = %f" % tstatistic
print "P-Value = %g" % pvalue
```

```
T Statistic = 8.020707
P-Value = 2.0515e-08
```

In [21]:

```
# Cohen's D

cohensD = (descriptiveStats['Incongruent']['mean'] - descriptiveStats['Congruent']['mean']) / descriptiveStats['Difference']['std']

print "Cohen's D: %f " % (cohensD)
```

```
Cohen's D: 1.637220
```

In [22]:

```
# Confidence Interval

import math as math

marginOfError = descriptiveStats['Difference']['std']/math.sqrt(descriptiveStats['Difference']['count']) * 1.714

ciLow = descriptiveStats['Difference']['mean'] - marginOfError
ciHigh = descriptiveStats['Difference']['mean'] + marginOfError

print "Margin of Error: %f" % marginOfError
print "Confidence interval: (%f, infinity)" % (ciLow)
```

```
Margin of Error: 1.702051
Confidence interval: (6.262741, infinity)
```

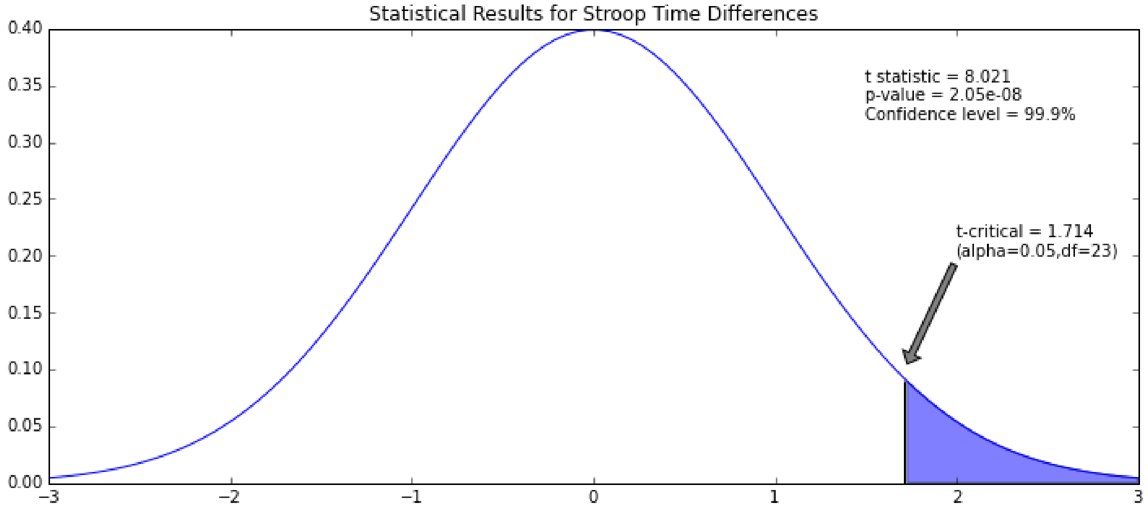
In [24]:

```
# Calculate r squared

tStatisticSquared = tstatistic * tstatistic

rSquared = tStatisticSquared / (tStatisticSquared + 23)

print "r squared: %f" % rSquared
```

```
r squared: 0.736636
```

In [25]:

```
# Display the t-critical and t statistic

fig, ax = plt.subplots(1,1,figsize=(12,5))

mean = 0
variance = 1
sigma = math.sqrt(variance)
x = np.linspace(-3,3,100)
y = mlab.normpdf(x,mean,sigma)

plt.vlines(1.714, 0, 0.09)

section = np.arange(1.714, 3, 1/20.)
plt.fill_between(x[78:100],y[78:100], facecolor='blue', alph
a=0.5)
ax.annotate('t statistic = 8.021\np-value = 2.05e-08\nConfidence
level = 99.9%', xy=(1.5, 0.1), xytext=(1.5,0.32))
ax.annotate('t-critical = 1.714\n(alpha=0.05,df=23)', xy=(1.714,
0.1), xytext=(2.0,0.2),
            arrowprops=dict(facecolor='grey', shrink=0.05),)

plt.title('Statistical Results for Stroop Time Differences')
plt.plot(x,y)

plt.show()
```



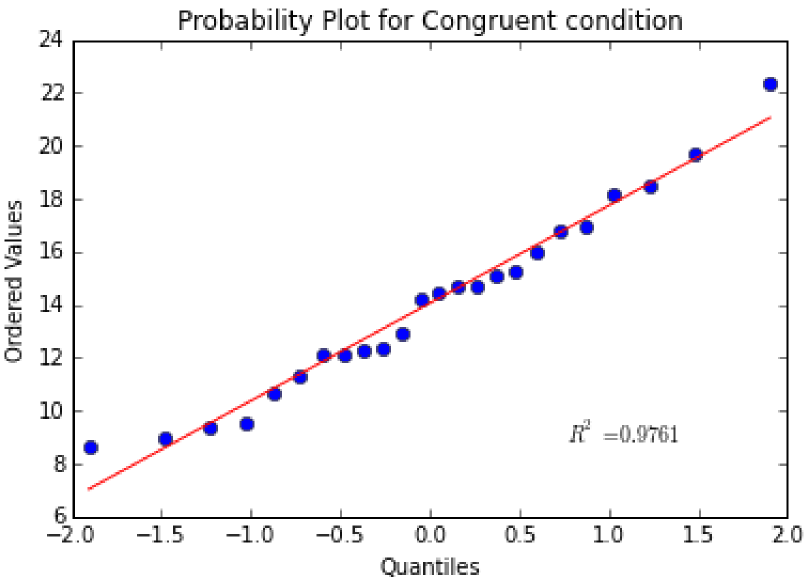
In [72]:

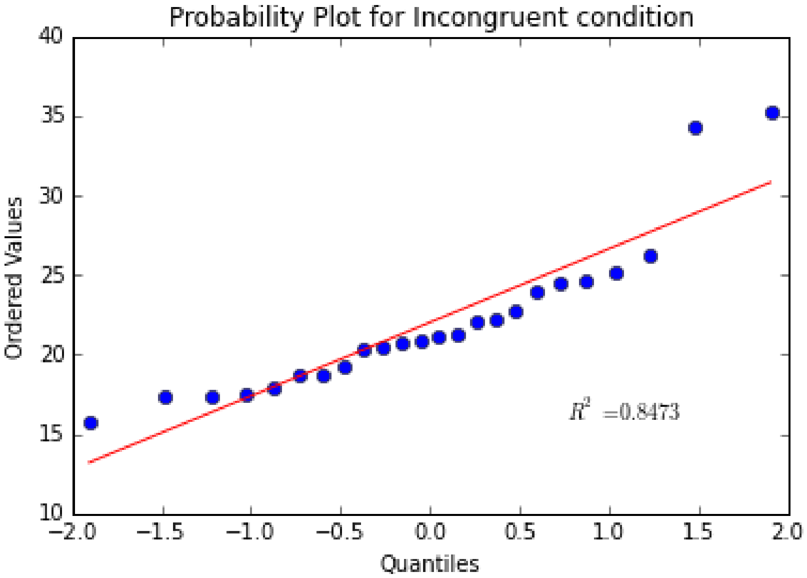
```
# Run the Shaprio-Wilk test

congruentShapiro, congruentP, congruentVals = stats.shapiro(congruentCol, reta=True)

inconShapiro, inconP, inconVals = stats.shapiro(incongruentCol, reta=True)

diffShapiro, diffP, diffVals = stats.shapiro(differenceCol, reta=True)

print "Congruent Shapiro-wilk:   %f, p-value = %f" % (congruentShapiro, congruentP)
print "Incongruent Shapiro-wilk: %f, p-value = %f" % (inconShapiro, inconP)
print "Difference Shapiro-wilk:  %f, p-value = %f" % (diffShapiro, diffP)
```

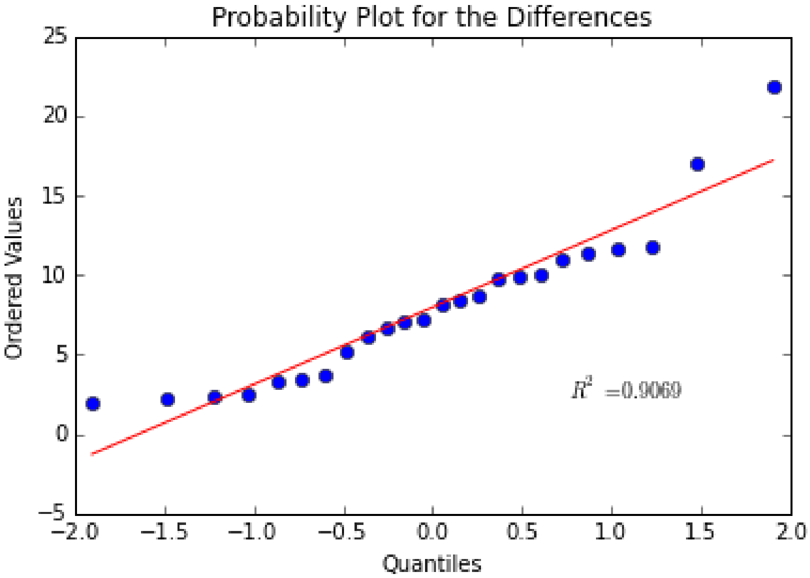
```
Congruent Shapiro-wilk:   0.970923, p-value = 0.689803
Incongruent Shapiro-wilk: 0.853947, p-value = 0.002590
Difference Shapiro-wilk:  0.910420, p-value = 0.036017
```

In [73]:

```
# Draw the QQ plots
stats.probplot(congruentCol, dist="norm", plot=plt)
plt.title("Probability Plot for Congruent condition")
plt.show()

stats.probplot(incongruentCol, dist="norm", plot=plt)
plt.title("Probability Plot for Incongruent condition")
plt.show()

stats.probplot(differenceCol, dist="norm", plot=plt)
plt.title("Probability Plot for the Differences")
plt.show()
```


Probability Plot for Congruent condition
Ordered Values
Quantiles
24
22
20
18
16
14
12
10
8
6
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
R^2 =0.9761


Probability Plot for Incongruent condition
Ordered Values
Quantiles
40
35
30
25
20
15
10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
R^2 =0.8473


Probability Plot for the Differences
Ordered Values
Quantiles
25
20
15
10
5
0
−5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
R^2 =0.9069

```
In [ ]:
```