Exploring the Stroop Effect utilizing IPython

Yigal Weinstein

November 2, 2015

Introduction

So the stage is set with the following description for the Stroop experiment:

> In a Stroop task, participants are presented with a list of words, with each word displayed in a color of ink. The participant's task is to say out loud the color of the ink in which the word is printed. The task has two conditions: a congruent words condition, and an incongruent words condition. In the congruent words condition, the words being displayed are color words whose names match the colors in which they are printed: for example RED , BLUE . In the incongruent words condition, the words displayed are color words whose names do not match the colors in which they are printed: for example PURPLE , ORANGE . In each case, we measure the time it takes to name the ink colors in equally-sized lists. Each participant will go through and record a time from each condition. [**Udacity-P1-Stroop**]

1 What is our independent variable? What is our dependent variable?

The independent variable is a binary condition that of either congruency or incongruency between the color of a word, and the symantic meaning of the word. As example the word 'RED' is a congruent example where the word 'red' has the color of its symantic meaning, i.e. the color it represents, and 'BLUE' is an incongruent case where the word 'blue' which has the symantic meaning of the color blue is instead the color green. Frome here on out the terms congruent color selection, CCS, and incongruent color selection, ICS, will be used to denote the two conditions the dependent variable may take that of a word representing a color being colored with or with a different color respectively.

The dependent variable is the time, in seconds, it takes to read through an equal sized lists of CCS or ICS words.

2 What is an appropriate set of hypotheses for this task? What kind of statistical test do you expect to perform? Justify your choices.

The null hypothesis, H_0 is the statement that the ICS task will take less or an equal amount of time to the CCS task. The alternative hypothesis, H_A is the statement that the mean population time to read an ICS list will be greater than an equally sized CCS list. The null and alternative hypothesis choice stems from a need to test quantatively the validity and extent of an already well respected phenomenon the Stroop effect. The Stroop effect being defined to mean that cognitive incongruences will lead to an increase in the reaction time of a task.

It is true that this single tailed objective may not provide insight if indeed the reaction time for ICS word lists take, on the average, less time to complete, however this is a test regarding the scope and veracity of the Stroop effect which has been shown in many previous quantative tests to have general validity. If indeed the ICS task takes less time it is enough that H_0 should be kept to cause doubt regarding the validity of either the test or the Stroop effect. That is due to the large body of work already done to test the effect if H_0 is kept there is either something wrong with the experiment or the Stroop effect has either more limitations on generalizability than expected or there are indeed real issues with it's veracity. So to sum things up via a set of equations:

$$H_0 : \ \mu_{ICS} \leq \mu_{CCS} \tag{1}$$

$$H_A : \ \mu_{ICS} > \mu_{CCS} \tag{2}$$

As the two sets of data are correlated, the same subjects are used to take both the ICS and CCS tests, the proper test to use is a dependent t-test for paired samples. Given the hypothesis devised above the single tail test will be used.

3 Report some descriptive statistics regarding this dataset. Include at least one measure of central tendency and at least one measure of variability.

Below I've printed the mean, sample standard deviation, minimum values, 25th, 50th, 75th percentiles, maximums, and the median for both the CCS and ICS samples, all values are in seconds.

	Congruent	Incongruent
count	24.000000	24.000000
mean	14.051125	22.015917
std	3.559358	4.797057
min	8.630000	15.687000
25%	11.895250	18.716750
50%	14.356500	21.017500
75%	16.200750	24.051500
max	22.328000	35.255000

To measure the central tendency consider the mean, and the median - which in this case is simply the 50th percentile. Notice the fact that for both CCS and ICS the median and mean are nearly identical, for the CCS case it's less than one third of a second difference, and for the ICS case the difference between these two measure of central tendency is only slightly greater than a second. This points to both distribution being evenly distributed around the mean. As we can see for nearly all percentiles, minimum and maximum, and mean values the CCS times are roughly two thirds of the value of the ICS times, and all times for every category are significantly longer for the ICS sample relative to the CCS.

An indication of the relative spread can be observed by looking at the sample standard deviations. the ICS case is slightly more spread out than the CCS case $4.797s$ to $3.559s$ respectively, i.e slightly over a second more for the ICS sample. To gain a better descriptive understanding of the spread we can also look at the range, i.e. $R = x_{max} - x_{min}$ of the two samples:

$$R_{CCS} = x_{CCS,max} - x_{CCS,min} = 22.328000 - 8.630000 = 13.698$$
$$R_{ICS} = x_{ICS,max} - x_{ICS,min} = 35.255000 - 15.687000 = 19.568$$

From both the sample standard deviation and the range we can see that the spread of the ICS sample is greater both in terms of the range, i.e. outliers, and of the sample standard deviation:

4 Provide one or two visualizations that show the distribution of the sample data. Write one or two sentences noting what you observe about the plot or plots.

```
In [17]: import numpy as np
         import matplotlib.pyplot as plt

         %pylab inline
         pylab.rcParams['figure.figsize'] = (8.0, 7.0)

         fig, ax = plt.subplots()

         index = np.arange(stroopdata.shape[0])
         bar_width = 0.36
```

```
opacity = 0.6

rects1 = plt.bar(index, stroopdata['Congruent'],
                 bar_width,
                 alpha=opacity,
                 color='k',
                 label='CCS Sample')

rects2 = plt.bar(index + bar_width, stroopdata['Incongruent'],
                 bar_width,
                 alpha=opacity,
                 color='b',
                 label='ICS Sample')

# add some text for labels, title and axes ticks
plt.rc('text', usetex=True)
plt.rc('font', family='sans')
plt.xlabel(r'\textbf{Subject Number}')
plt.ylabel(r'\textit{Time} (s)', fontsize=16)
plt.title(r'Scores for CCS and ICS',fontsize=16, color='gray')
plt.xticks(index + bar_width, range(1,25))
plt.legend()

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

```
Populating the interactive namespace from numpy and matplotlib
```

In [12]:
```
"""
Demo of TeX rendering.

You can use TeX to render all of your matplotlib text if the rc
parameter text.usetex is set.  This works currently on the agg and ps
backends, and requires that you have tex and the other dependencies
described at http://matplotlib.org/users/usetex.html
properly installed on your system.  The first time you run a script
you will see a lot of output from tex and associated tools.  The next
time, the run may be silent, as a lot of the information is cached in
~/.tex.cache

"""
import numpy as np
import matplotlib.pyplot as plt
```

```
# Example data
t = np.arange(0.0, 1.0 + 0.01, 0.01)
s = np.cos(4 * np.pi * t) + 2

plt.rc('text', usetex=True)
plt.rc('font', family='sans')
plt.plot(t, s)

plt.xlabel(r'\textbf{time} (s)')
plt.ylabel(r'\textit{voltage} (mV)',fontsize=16)
plt.title(r"\TeX\ is Number "
          r"$\displaystyle\sum_{n=1}^\infty\frac{-e^{i\pi}}{2^n}$!",
          fontsize=16, color='gray')
# Make room for the ridiculously large title.
plt.subplots_adjust(top=0.8)

plt.savefig('tex_demo')
plt.show()
```

The bar graph above shows graphically that each subject took more time on the ICS word list than on the CCS word list.

5 Now, perform the statistical test and report your results. What is your confidence level and your critical statistic value? Do you reject the null hypothesis or fail to reject it? Come to a conclusion in terms of the experiment task. Did the results match up with your expectations?

6 What do you think is responsible for the effects observed? Can you think of an alternative or similar task that would result in a similar effect? Some research about the problem will be helpful for thinking about these two questions!

Refrences:

- PANDAS:
- http://pandas.pydata.org/pandas-docs/stable/io.html#io-read-csv-table # link for reading in CSV file
- Udacity:
- https://docs.google.com/document/d/1-OkpZLjG_kX9J6LIQ5IltsqMzVWjh36QpnP2RYpVdPU/pub # Udacity official document on general instructions for further projects as well as specifically for the Stroop Effect, P1.

- https://docs.google.com/document/d/1bqyi1Fm5truesLhmbAq16Zl-Ajj9bnNIU_68P60nDQg/pub # Udacity official documentation outlining what is expected to satisfactorily complete a project
- matplotlib
- http://matplotlib.org/examples/api/barchart_demo.html # The bar graph example was used as a basis for the bar graph in this report.
- http://matplotlib.org/api/colors_api.html # understanding colors in matplotlib
- http://matplotlib.org/users/usetex.html # using LaTeX in matplotlib graphs

In [19]:
```
%%bash
ipython3 nbconvert --template udacity.tplx  --to latex "Stroop Effect.ipynb" 2>&1 > build.log
pdflatex "Stroop Effect" >> build.log
bibtex "Stroop Effect" >> build.log
pdflatex "Stroop Effect" >> build.log
```

```
/home/yigal/.local/lib/python3.4/site-packages/IPython/nbconvert.py:13: ShimWarning: The ‘IPython.nbconv
  "You should import from ipython_nbconvert instead.", ShimWarning)
[NbConvertApp] Converting notebook Stroop Effect.ipynb to latex
[NbConvertApp] Support files will be in Stroop Effect_files/
[NbConvertApp] Making directory Stroop Effect_files
[NbConvertApp] Making directory Stroop Effect_files
[NbConvertApp] Writing 32871 bytes to Stroop Effect.tex
```

In [ ]:
```
%ls
```

In [ ]:
```
#from notebook.nbextensions import install_nbextension
#install_nbextension('https://goo.gl/5TK96v', user=True, destination="vim_binding.js")
# Or if you prefre a full URL
#install_nbextension('https://rawgithub.com/lambdalisue/jupyter-vim-binding/master/nbextensions/
#Jupyter.utils.load_extensions('vim_binding')
```

In [8]:
```
%%javascript
require(['base/js/utils'],
function(utils) {
   utils.load_extensions('calico-spell-check', 'calico-document-tools', 'calico-cell-tools');
});
```

```
<IPython.core.display.Javascript object>
```

Examples of citations: (Pérez and Granger, 2007) or (Papa and Markov, 2007) (Udacity, 2015) (, 1999).

In [24]:
```
!gksu ipython3 install-nbextension https://bitbucket.org/ipre/calico/downloads/calico-document
```

```
(gksu:11779): GConf-CRITICAL **: gconf_value_free: assertion 'value != NULL' failed
Traceback (most recent call last):
  File "/usr/local/bin/ipython3", line 11, in <module>
    sys.exit(start_ipython())
  File "/usr/local/lib/python3.4/dist-packages/IPython/__init__.py", line 118, in start_ipython
    return launch_new_instance(argv=argv, **kwargs)
  File "/usr/local/lib/python3.4/dist-packages/traitlets/config/application.py", line 591, in launch_ins
    app.initialize(argv)
  File "<decorator-gen-111>", line 2, in initialize
  File "/usr/local/lib/python3.4/dist-packages/traitlets/config/application.py", line 75, in catch_conf
```

```
    return method(app, *args, **kwargs)
  File "/usr/local/lib/python3.4/dist-packages/IPython/terminal/ipapp.py", line 305, in initialize
    super(TerminalIPythonApp, self).initialize(argv)
  File "<decorator-gen-7>", line 2, in initialize
  File "/usr/local/lib/python3.4/dist-packages/traitlets/config/application.py", line 75, in catch_conf
    return method(app, *args, **kwargs)
  File "/usr/local/lib/python3.4/dist-packages/IPython/core/application.py", line 386, in initialize
    self.parse_command_line(argv)
  File "/usr/local/lib/python3.4/dist-packages/IPython/terminal/ipapp.py", line 300, in parse_command_l
    return super(TerminalIPythonApp, self).parse_command_line(argv)
  File "<decorator-gen-4>", line 2, in parse_command_line
  File "/usr/local/lib/python3.4/dist-packages/traitlets/config/application.py", line 75, in catch_conf
    return method(app, *args, **kwargs)
  File "/usr/local/lib/python3.4/dist-packages/traitlets/config/application.py", line 487, in parse_comm
    return self.initialize_subcommand(subc, subargv)
  File "<decorator-gen-3>", line 2, in initialize_subcommand
  File "/usr/local/lib/python3.4/dist-packages/traitlets/config/application.py", line 75, in catch_conf
    return method(app, *args, **kwargs)
  File "/usr/local/lib/python3.4/dist-packages/traitlets/config/application.py", line 418, in initializ
    subapp = import_item(subapp)
  File "/usr/local/lib/python3.4/dist-packages/ipython_genutils/importstring.py", line 31, in import_it
    module = __import__(package, fromlist=[obj])
ImportError: No module named 'notebook'
```

```
In [20]: IPython.display.YouTubeVideo("YbM8rrj-Bms")
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-20-8f1b1a3b6f91> in <module>()
----> 1 IPython.display.YouTubeVideo("YbM8rrj-Bms")


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