Statistical Thinking in Python Part 2

4). Hypothesis test examples

a). The Vote for the Civil Rights Act 1964:

```
# Construct arrays of data: dems, reps
dems = np.array([True] * 153 + [False] * 91)
reps = np.array([True] * 136 + [False] * 35)

def frac_yea_dems(dems, reps):
    """Compute fraction of Democrat yea votes."""
    frac = np.sum(dems) / len(dems)
    return frac

# Acquire permutation samples: perm_replicates
perm_replicates = draw_perm_reps(dems, reps, frac_yea_dems, 10000)

# Compute and print p-value: p
p = np.sum(perm_replicates <= 153/244) / len(perm_replicates)
print('p-value =', p)
```

```
<script.py> output:
    p-value = 0.0002
```

b). <u>A time on website Analog</u>

```
# Compute the observed difference in mean inter-no-hitter times: nht_diff_obs
nht_diff_obs = diff_of_means(nht_dead , nht_live)

# Acquire 10,000 permutation replicates of difference in mean no-hitter time: perm_replicates
perm_replicates = draw_perm_reps(nht_dead, nht_live, diff_of_means,10000)



# Compute and print the p-value: p
p = np.sum(perm_replicates <=nht_diff_obs )/len(perm_replicates)
print('p-val =',p)
```

<script.py> output:

```
    p-val = 0.0001
```

c). <u>Hypothesis Test on Pearson Co-relation</u>

```
# Compute observed correlation: r_obs
r_obs = pearson_r(illiteracy, fertility)

# Initialize permutation replicates: perm_replicates
perm_replicates = np.empty(10000)

# Draw replicates
for i in range(10000):
    # Permute illiteracy measurments: illiteracy_permuted
    illiteracy_permuted = np.random.permutation(illiteracy)

    # Compute Pearson correlation
    perm_replicates[i] = pearson_r(illiteracy_permuted, fertility)

# Compute p-value: p
p = np.sum(perm_replicates >= r_obs)/len(perm_replicates)
print('p-val =', p)
```

<script.py> output:

```
    p-val = 0.0
```

d). Do neonicotinoid insecticides have unintended consequences?

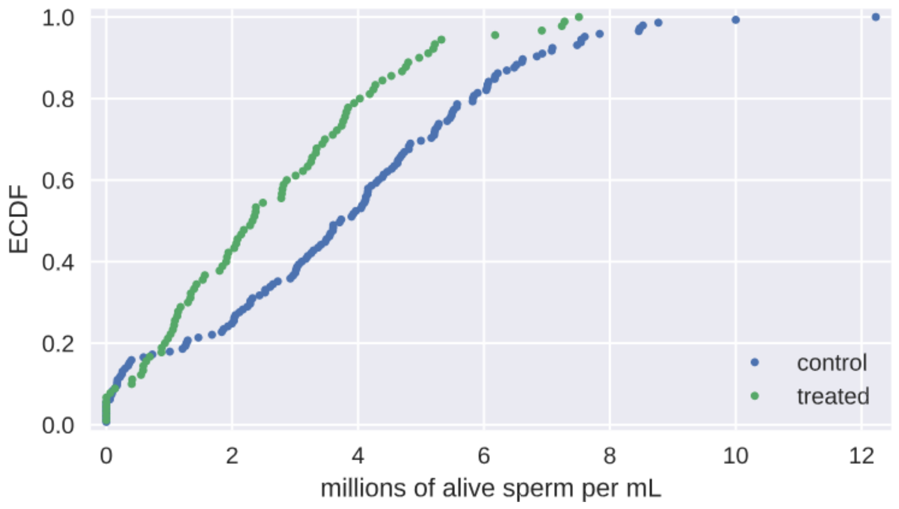
```
# Compute x,y values for ECDFs
x_control, y_control = ecdf(control)
x_treated, y_treated = ecdf(treated)

# Plot the ECDFs
plt.plot(x_control, y_control, marker='.', linestyle='none')
plt.plot(x_treated, y_treated, marker='.', linestyle='none')

# Set the margins
plt.margins(0.02)

# Add a legend
plt.legend(('control', 'treated'), loc='lower right')

# Label axes and show plot
plt.xlabel('millions of alive sperm per mL')
plt.ylabel('ECDF')
plt.show()
```



e). <u>Bootstrap Hypothesis test on bee sperm count</u>

```
# Compute the difference in mean sperm count: diff_means

diff_means = np.mean(control)- np.mean(treated)


# Compute mean of pooled data: mean_count

mean_count = np.mean(np.concatenate((control, treated)))


# Generate shifted data sets

control_shifted = control - np.mean(control) + mean_count

treated_shifted = treated - np.mean(treated) + mean_count


# Generate bootstrap replicates

bs_reps_control = draw_bs_reps(control_shifted,

                    np.mean, size=10000)

bs_reps_treated = draw_bs_reps(treated_shifted,

                    np.mean, size=10000)


# Get replicates of difference of means: bs_replicates

bs_replicates = bs_reps_control - bs_reps_treated


# Compute and print p-value: p

p = np.sum(bs_replicates >= np.mean(control) - np.mean(treated)) \

          / len(bs_replicates)

print('p-value =', p)
```

```
<script.py> output:

   p-value = 0.0
```