ESE 542: Statistics for Data Science **Spring 2019**

HW Tyler Olivieri

Name: *Tyler Olivieri*

1.

Tyler Oliveri HW 5 #1

Suppose $X_1, \ldots, X_n$ are an iid random sample of size n w/ sample mean $\bar{X} = 12$ and sample variance $S^2 = 5$

a) Let $n = 5$ and suppose samples are drawn from a normal distribution w/ unknown mean μ and known variance $\sigma^2 = 9$

Let the null hypothesis be $H_0: \mu = 10$; $H_a: \mu \neq 10$

Calculate the relevant test statistic value and p-value.

Want to use z-test. The relevant test statistic is

$$Z = \frac{\bar{X} - \mu_0}{\sigma/\sqrt{n}} = \frac{12 - 10}{3/\sqrt{5}} = \frac{2\sqrt{5}}{3}$$

p-value will be smallest significance level at which the null hypothesis would be rejected.

$$p = \Pr\{\text{event more contradictory than observed} \mid H_0 \text{ is true}\}$$

$$= \Pr(Z > z \cup Z < -z \mid H_0 \text{ is true})$$

Due to symmetry under null hypothesis

$$= 2\Pr(Z > |z| \mid H_0 \text{ is true})$$

$$= 2(1 - \Pr\{Z \leq |z| \mid H_0 \text{ is true}\})$$

$$= 2(1 - \Phi(|z|))$$

$$= 2\left(1 - \Phi\left(\frac{2\sqrt{5}}{3}\right)\right) = 2(1 - \Phi(1.5)) = .13 = p\text{-value}$$

b) Using the acceptance region of this test, construct a 95% confidence interval.

let $\alpha = .05$ for 95% confidence.

$$1-\alpha = Pr\left(-z \leq \frac{\bar{X}-\mu_0}{\sigma/\sqrt{n}} \leq z\right)$$

from Normal table

$$.95 = Pr\left(-1.96 \leq \frac{\bar{X}-\mu_0}{\sigma/\sqrt{n}} \leq 1.96\right)$$

$$.95 = Pr\left(-1.96(\sigma/\sqrt{n}) - \bar{X} \leq -\mu_0 \leq 1.96(\sigma/\sqrt{n}) - \bar{X}\right)$$

$$.95 = Pr\left(1.96(\sigma/\sqrt{n}) + \bar{X} \geq \mu_0 \geq -1.96(\sigma/\sqrt{n}) + \bar{X}\right)$$

$$.95 = Pr\left(\bar{X} - 1.96(\sigma/\sqrt{n}) \leq \mu_0 \leq \bar{X} + 1.96(\sigma/\sqrt{n})\right)$$

substitute $\bar{X}$, σ, and n from known values.

$$.95 = Pr\left(12 - 1.96(3/\sqrt{5}) \leq 12 \leq 12 + 1.96(3/\sqrt{5})\right)$$

$\Rightarrow$ 95% confidence interval for μ_0

$$\left[12 - 1.96(3/\sqrt{5}),\ 12 + 1.96(3/\sqrt{5})\right]$$

$$[9.37,\ 14.63]$$

(a) Determine decision rule for $\alpha = .05$

$.05 = Pr\{\text{type I error}\}$

$= Pr\{H_0 \text{ is rejected} \mid H_0 \text{ is true}\}$

$= Pr\{z < -z_+ \cup z > z_+ \mid H_0 \text{ is true}\}$

$= 2Pr\{z < -z_+ \mid H_0 \text{ is true}\}$

$.05 = 2\Phi(-z_+)$

$\frac{.05}{2} = \Phi(-z_+)$

$\Phi^{-1}(.05/2) = -z_+$

$-\Phi^{-1}\left(\frac{.05}{2}\right) = z_+ = 1.96$

Accept H_0 if $-z_+ < z < z_+$ $\quad -1.96 < z < 1.96$

reject otherwise.

Tyler Olivieri HW3

1c) Now let $n=5$ again but suppose the samples are drawn from an unspecified distribution w/ unknown mean μ and unknown variance σ^2. Let the null hypothesis be $H_0: \mu = 10$ and the alternate hypothesis be $H_a: \mu > 10$

Calculate the relevant test statistic value and p-value.

Determine the decision rule for $\alpha = .05$.

The relevant test statistic will be t, because we do not know the distribution and n is small

$$T = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} = \frac{12 - 10}{\sqrt{5}/\sqrt{5}} = \frac{2\sqrt{5}}{\sqrt{5}} = 2$$

p-value = $\text{Prob}\{T_t > T \mid H_0 \text{ is true}\}$ one sided

$= 2[\text{Prob}\{t \leq T \mid H_0 \text{ is true}\}]$

$= 2[1 - t_{n-1}(T)] = 1 - t_4(T) = 1 - t_4(2)$

$= .06$

t_4 CDF of t-dist with 4 deg of freedom.

$\alpha = P_r \{ \text{Type 1 error} \mid H_0 \text{ is true} \}$

should be $t_{\alpha, n-1}$

$.05 = P_r (t \geq T_\alpha \mid H_0 \text{ is true})$ but I am getting $t^{-1}_{\alpha, n-1}$

$.05 = t_4 (\pm T_\alpha)$

$.05 = 1 + t_4(T_\alpha)$

$.05 - .05 = t_4(T_\alpha)$

critical value $= T_\alpha = t_4^{-1}(1 - .05) = t_4^{-1}(.95) = 2.13$

Tyler Olivieri HW 2 2)

Suppose that $Y_1, \dots, Y_n$ are an iid random sample of size n drawn from a Poisson distribution w/ unknown parameter λ.

Using $\sum_{i=1}^{n} Y_i$ as the test statistic, find the critical value and rejection region at level α for the test.

let $T = \sum_{i=1}^{n} Y_i$

$$\begin{cases} H_0 : \lambda = \lambda_0 \\ H_a : \lambda = \lambda^* \quad \text{where } \lambda^* > \lambda_0 \end{cases}$$

let ζ be the critical value

$\alpha = Pr\{\text{Type I error occurs}\}$

$\alpha = Pr\{\text{reject } H_0 \mid H_0 \text{ is true}\}$

reject H_0 when $T > \zeta$

$\alpha = Pr\{T > \zeta \mid H_0 \text{ is true}\}$

$\alpha = 1 - Pr\{T \leq \zeta \mid H_0 \text{ is true}\}$

Since $T \sim \text{poisson}(\lambda)$, given H_0 is true $\lambda = \lambda_0$

$Pr\{T \leq \lambda \mid H_0 \text{ is true}\}$ is cdf of poisson with λ_0

However, as $n \to \infty$ $\dfrac{T - n\lambda_0}{\sqrt{\lambda_0 n}} \xrightarrow{d} N(0,1)$ where $\mathbb{E}[Y_i] = \lambda$, $Var(Y_i) = \lambda$

piazza note allows assumption of n large

thus we can say $\dfrac{T - n\lambda_0}{\sqrt{\frac{\lambda_0}{n}}}$ is approximately distributed gaussian conditioned under the null hypothesis being true.

let $z = \frac{T - n\lambda_0}{\sqrt{\frac{\lambda_0}{n}}}$

reject when $z > \ell$ where ℓ is a threshold

$$\frac{T - n\lambda_0}{\sqrt{\frac{\lambda_0}{n}}} > \ell$$

$$T > \ell\sqrt{\frac{\lambda_0}{n}} + n\lambda_0 \quad (*)$$

$\alpha = Pr\{ z > \ell \mid H_0 \text{ is true} \}$

$\alpha = 1 - Pr\{ z \leq \ell \mid H_0 \text{ is true} \}$ $z \sim N(0,1)$ given H_0 is true

$\alpha = 1 - \Phi(\ell)$

$1 - \alpha = \Phi(\ell)$

$\ell = \Phi^{-1}(1-\alpha)$

we can translate this threshold to use the statistic T instead of z using $(*)$

The rejection region is any $z > \ell$ so (ℓ, ∞)

The rejection region for T is $T > \ell\sqrt{\frac{\lambda_0}{n}} + n\lambda_0$

$$\left(\ell\sqrt{\frac{\lambda_0}{n}} + n\lambda_0, \infty\right)$$

Tyler Olivieri HW3 #3)

Consider a random sample of size $n=100$ w/ sample proportion $\hat{p} = .2$ from a population w/ a true unknown proportion p.

a) For the test $H_0: p = .25$ versus $H_a: p < .25$, calculate the relevant test statistic value and p-value. Determine the decision rule for $\alpha = .05$ and $\alpha = .01$

Test statistic: We can use the z-test here because we have large n. From CLT $\hat{p}$ calculation will be distributed approx normal.

$$\text{Variance of prop} = \frac{p(1-p)}{n}$$

$$\text{test } z = \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{n}}}$$

$$= \frac{.2 - .25}{\sqrt{\frac{.25(1-.25)}{100}}} = \frac{-.05}{.043} = -1.163$$

$\alpha = Pr\{\text{Type I error}\} = Pr\{H_0 \text{ is rejected} \mid H_0 \text{ is true}\}$

$\alpha = Pr\{z < z \mid H_0 \text{ is true}\}$ z is threshold.

This is just one-sided gaussian. $z \sim N(0,1)$ when H_0 is true.

$\alpha = \Phi(z)$

$z = \Phi^{-1}(\alpha)$

when $\alpha = .05$

$z = \Phi^{-1}(.05) = -1.65$

when $\alpha = .01$

$z = \Phi^{-1}(.01) = -2.33$

accept H_0 when $z > z$

reject H_0 when $z < z$

p value $= Pr\{z < z \mid H_0 \text{ is true}\}$

$= \Phi(-1.163)$

$= .12$

b)

For the test $H_0: p = .25$ versus $H_a: p \neq .25$

Calculate the relevant test statistic value and p-value.

Determine the decision rule for $\alpha = .05$ and $\alpha = .01$

The test statistic does not change

$$z = \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{n}}} = -1.163$$

p value $= 2 Pr\{z < -1.163 \mid H_0 \text{ is true}\}$

$= 2\Phi(-1.163) = .1213$

$\alpha = Pr\{\text{Type I error}\} = Pr\{H_0 \text{ is rejected} \mid H_0 \text{ is true}\}$

$\alpha = z(1 - \Phi(z)) \Rightarrow z = \pm\Phi^{-1}(1 - \alpha/2)$

$\alpha/2 = 1 - \Phi(z)$

$1 - \alpha/2 = \Phi(z)$

for $\alpha = .05$

$z = \Phi^{-1}(1 - .05/2), -\Phi^{-1}(1 - .05/2)$

$z = 1.96, -1.96$

accept if $-z < z < z$ reject o.w

for $\alpha = .01$

$$z = \Phi^{-1}(1 - .01/2) \quad , -\Phi^{-1}(1 - .01/2) \quad = 2.58 \; , -2.58$$

accept H_0 when $-z < Z < z$ $\quad -2.58 < Z < 2.58$

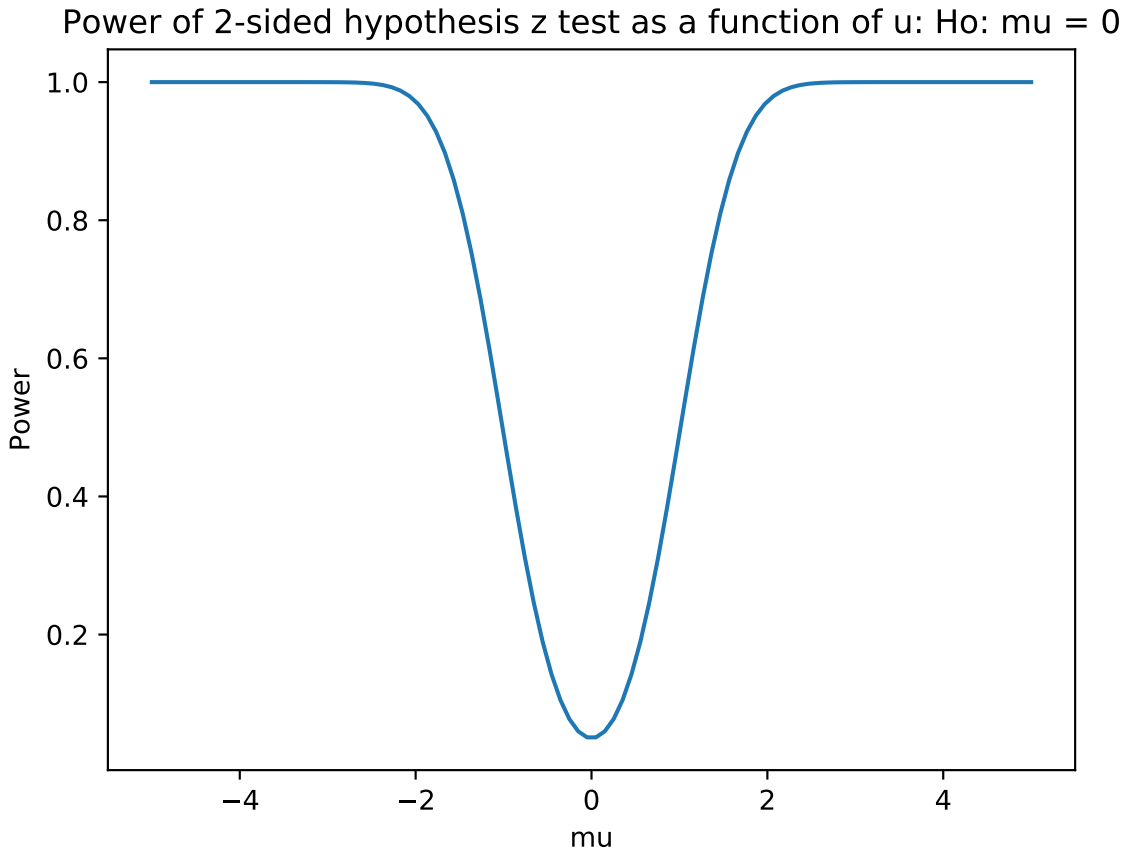
reject H_0 otherwise.

2. Number 4.

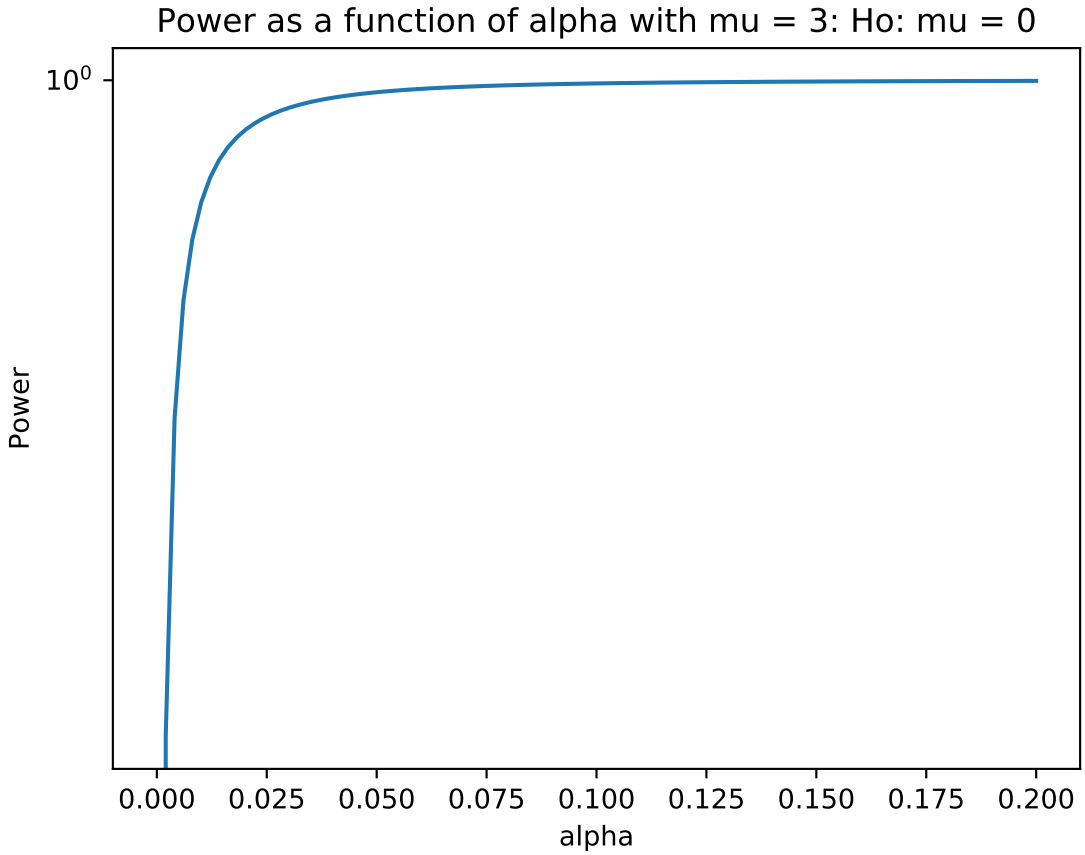
```
## Stastics for Data Science
# HW3 #4
# Tyler Olivieri

from scipy.stats import norm
import math
from matplotlib import pyplot as plt
import numpy as np

# For X1, ... , Xn iid ; n = 15 drawn N(mu, sigma^2 = 4)
##########################################################################################
# (a)
# plot the power of the test Ho: mu = 0 versus Ha: mu != 0

n = 15
var = 4
alpha = .05
mu_0 = 0
threshold = norm.ppf(1 - alpha/2)
scaling_factor = math.sqrt(var/n)
mu_vals = np.linspace(-5,5,100)
power = []

# calculate power
for mu in mu_vals:
    power.append( ( (1 - norm.cdf(threshold - ((mu - mu_0)/scaling_factor))) + norm


# plot results
plt.title("Power of 2-sided hypothesis z test as a function of u: Ho: mu = 0")
plt.xlabel("mu")
plt.ylabel("Power")
plt.plot(mu_vals,power)
plt.savefig('hw3_4_a.pdf', format='pdf')
#plt.show()
plt.clf()

##########################################################################################
# (b)
# now vary alpha and fix mu = 3
#

alpha_vals = np.linspace(0,.2,100)
mu = 3
power.clear()

# calculate power
for alpha_iter in alpha_vals:
    threshold = norm.ppf(1 - alpha_iter/2)
    power.append( (1 - norm.cdf(threshold - ((mu - mu_0)/scaling_factor))) + norm.c
```

```
# plot results
plt.title("Power as a function of alpha with mu = 3: Ho: mu = 0")
plt.xlabel("alpha")
plt.ylabel("Power")
plt.semilogy(alpha_vals,power)
plt.savefig('hw3_4_b.pdf', format='pdf')
#plt.show()
plt.clf()

##########################################################################################
# (c)
# fix alpha = .05 and plot the power of the test as n varies
#


threshold = norm.ppf(1 - alpha/2)
power.clear()

# calculate power
for n_iter in range(1,25):
    scaling_factor_n = math.sqrt(var/n_iter)
    power.append( (1 - norm.cdf(threshold - ((mu - mu_0)/scaling_factor_n))) + norm

# plot results
plt.title("Power as a function of n with mu = 3: Ho: mu = 0")
plt.xlabel("n")
plt.ylabel("Power")
plt.plot(range(1,25),power)
plt.savefig('hw3_4_c.pdf', format='pdf')
#plt.show()
plt.clf()

##########################################################################################
# (d)
# fix n = 15 = .05 and plot the power of the test as sigma^2 varies
#

n = 15
threshold = norm.ppf(1 - alpha/2)
power.clear()

var_vals = np.linspace(1,10,100)

# calculate power
for var in var_vals:
    scaling_factor = math.sqrt(var/n)
    power.append( (1 - norm.cdf(threshold - ((mu - mu_0)/scaling_factor))) + norm.c

# plot results
plt.title("Power as a function of variance with mu = 3: Ho: mu = 0")
```

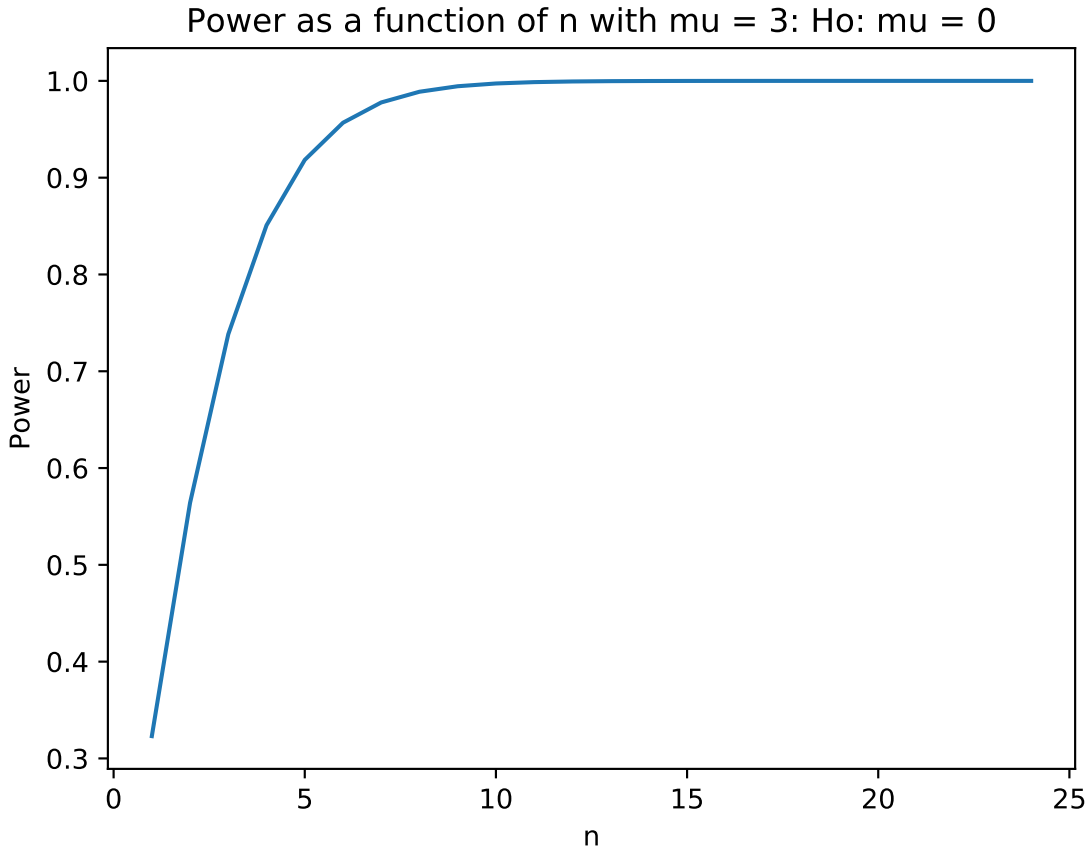
```
plt.xlabel("variance")
plt.ylabel("Power")
plt.plot(var_vals,power)
plt.savefig('hw3_4_d.pdf', format='pdf')
plt.show()
```



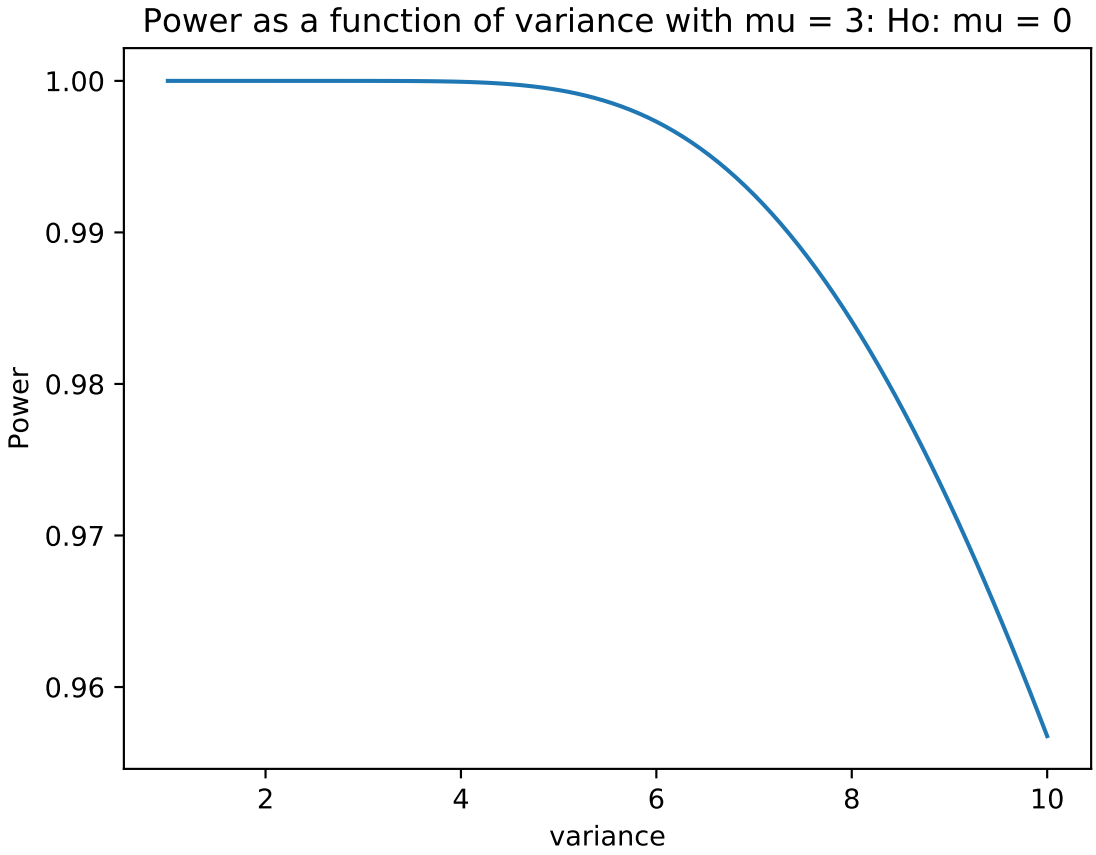
(a)


Power as a function of alpha with mu = 3: Ho: mu = 0
10^0
Power
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
alpha

(b)


Power as a function of n with mu = 3: Ho: mu = 0
Power
n
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25

(c)



(d)

(e) compare and interpret results

The results make sense. The power of the test is defined as the probability that the test rejects the null hypothesis when the alternative hypothesis is true.

For the first part where the power is observed as a function of μ. The power is at its lowest when $\mu = \mu_0$. This makes sense because the test should not reject the null hypothesis, if the null hypothesis is true, it would be an error, so the probability that the null hypothesis is rejected is low.

Conversely, the power grows as the true μ of the distribution deviates from the null hypothesis, as there is a higher probability that alternative hypothesis is true and that the test should reject H_o, thus the power of the test increases. The observed test statistics are more likely to be in the rejection region if the true μ is much different from the null hypothesis.

As alpha grows, the probability that the null hypothesis will be rejected also grows, as the rejection region will grow with alpha. Since the null hypothesis is incorrect, the test should be rejected. Thus, the power of the test increases when we allow more rejection of the null hypothesis.

Similarly, as n grows, there is more chance that the observation will be closer to $\mu = 3$, which is in the rejection region. Thus, the probability that the null hypothesis will be rejected grows, and the power of the test grows because it is correct to reject the null hypothesis.

However, when the variance grows, the data will be more spread out, and the observed μ will be more erratic. The test statistic will get closer to 0, as it is scaled inversely by the variance. Since this is a two sided test, the closer the test statistic is to 0, the more likely that null hypothesis is true. Thus

the probability that the null hypothesis is incorrectly accepted increases and the power of the test decreases.

3. Number 5.

```
## Numpy Practice

import numpy as np
from urllib.request import urlopen
import math

url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

# Below are the names corresponding to each column, e.g. 2nd column has values of sepa
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')

# TODO 1: Import data from the above url into a numpy ndarray named 'iris'.
tuple_list = []

f = urlopen(url)
data = f.read().decode("utf-8")
for line in data.splitlines():
    split_line = line.split(',')
    tuple_list.append(tuple(split_line))

iris = np.array(tuple_list)

# remove last erroneous line
iris = iris[:-1]

# TODO 2: Print out the data. Observe the different kinds of species of irises.
print(iris)

# Now let's separate the data into 2 numpy arrays.
# The first array (iris_1d) will contain the species name (5th column).
# The second array (irid_2d) will contain the other 4 columns. Convert this into a 'dt
# TODO 3: Create iris_1d and iris_2d.
iris_1d = np.zeros(len(iris), dtype=object)
iris_2d = np.zeros((len(iris), 4), dtype=float)

for i in range(0, len(iris)):
    el0, el1, el2, el3, el4 = iris[i]
    iris_1d[i] = el4
    iris_2d[i,0] = el0
    iris_2d[i,1] = el1
    iris_2d[i,2] = el2
    iris_2d[i,3] = el3

### Wrangling

# Now that you have imported data, you must check to see if the data is fit for analys
# First, let's modify the data a bit to put in some NaN(not a number) values in it
# TODO 4: Randomly convert 20 entries in 'iris_2d' into NaNs.
```

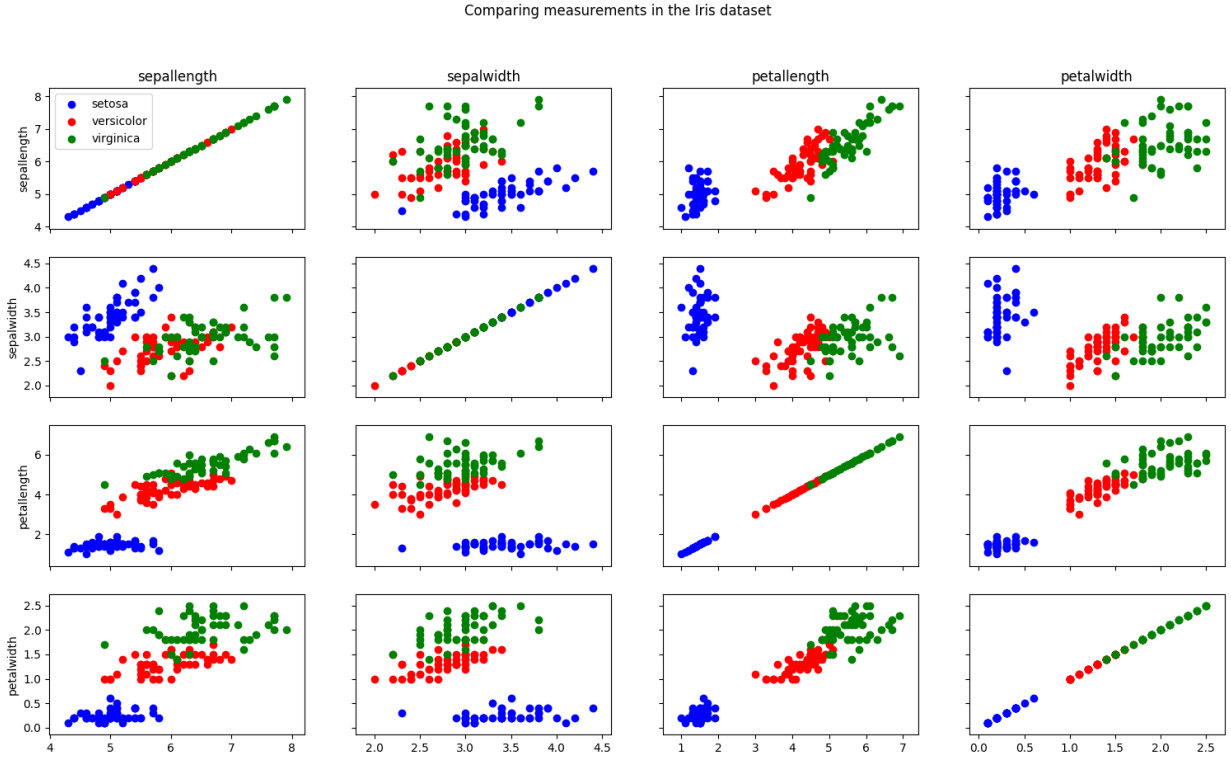
```
for i in range(1, 20):
    # randomly select row index
    r_idx = np.random.uniform(0,len(iris_2d)-1)
    r_idx = round(r_idx)

    # randomly select col index
    c_idx = np.random.uniform(0,3)
    c_idx = round(c_idx)

    iris_2d[r_idx,c_idx] = np.nan

# There are several ways to deal with corrupted data (NaNs).
# One way to do this is to replace NaNs with the average of the other values in that c
# TODO 5: Replace the NaN entries in the data with the average values of each column.

# take average of each column
col_mean = np.nanmean(iris_2d, axis=0)

# find where the nans are
idxs = np.where(np.isnan(iris_2d))

# replace nan with mean
iris_2d[idxs] = np.take(col_mean, idxs[1])

# TODO 6: Write a function to check if there are NaN values in the data. You may use b
print(np.isnan(iris_2d).any())

# ### Filtering

# TODO 7: Filter the rows of 'iris_2d' that has petallength (3rd column) > 1.5 and sep
filtered_iris = []
for i in range(0,len(iris_2d)):
    if iris_2d[i,0] < 5 and iris_2d[i,2] > 1.5:
        # not sure what to do with the filtered rows  ... I stored them in a list
        filtered_iris.append(iris_2d[i,:])

print(filtered_iris)

# Create a deep copy of 'iris_2d'.
iris_2d_c = iris_2d.copy()

# TODO 8: Convert the column 'petallength' (3rd col) to a string according to the follo
# - <3 --> 'small'
# - 3-5 --> 'medium'
# - >=5 --> 'large'


# I was not sure if the strings should replace the entries in the numpy array or to cr
# loop over 3rd column
string_list = []
```

```
for i in range(0, len(iris_2d_c)):
    if iris_2d[i,2] < 3:
        string_list.append("small")
    elif iris_2d[i,2] >= 3 and iris_2d[i,2] < 5:
        string_list.append("medium")
    else:
        string_list.append("large")

print(string_list)

### Statisticsl

# You may use scipy.stats package as you desire, although you can get through all the
import scipy.stats as st

# Please re-import the iris dataset (the same way as TODO 1)
# TODO 9: Compute the means of the sepalwidth of the different species of irises and r
tuple_list = []

f = urlopen(url)
data = f.read().decode("utf-8")
for line in data.splitlines():
    split_line = line.split(',')
    tuple_list.append(tuple(split_line))

iris = np.array(tuple_list)

# remove last erroneous line
iris = iris[:-1]
iris_1d = np.zeros(len(iris), dtype=object)
iris_2d = np.zeros((len(iris), 4), dtype=float)

for i in range(0, len(iris)):
    el0, el1, el2, el3, el4 = iris[i]
    iris_1d[i] = el4
    iris_2d[i,0] = el0
    iris_2d[i,1] = el1
    iris_2d[i,2] = el2
    iris_2d[i,3] = el3

# find idices of corresponding species
species = np.unique(iris_1d)
means = np.zeros(len(species))
for i in range(0,len(species)):
    idxs = np.where(iris_1d[:] == species[i])
    means[i] = np.mean(iris_2d[idxs,1])

# reverse sort or highest to lowest means
ordered_means = np.sort(means)[::-1]

# Hypothesis testing
```

```
# TODO 10: Compute the p-value for the hypothesis test:
# H_0 = mean(sepallength of Iris-setosa) < 5.0
# H_a = mean(sepallength of Iris-setosa) >= 5.0
# and determine whether to accept or reject H_0 with significance level alpha = 0.05
Ho_mu = 5
alpha = .05

# first compute x_bar or sample mean of sepallength of Iris-setosa, which is column 0
idxs = np.where(iris_1d[:] == "Iris-setosa")
n = len(idxs[0])
x_bar = np.mean(iris_2d[idxs,0])
var = np.var(iris_2d[idxs,0])

# use t test because we don't know population std dev, and student t distribtuion conv
t = (x_bar - Ho_mu)/(math.sqrt(var/n))

# calculate p-value
p_value = 1 - st.t.cdf(t,n-1)
print(p_value)

# determine threshold at significance level alpha = 0.05
threshold = st.t.ppf(1-alpha, n-1)

if t > threshold:
    print("reject Ho")
else:
    print("accept Ho")

### Plotting

from matplotlib import pyplot as plt

# TODO 11: Construct a plot that enables us to compare the 4 different measurements (C
# You are free to choose whichever plot that will fit this goal. The grading criteria
# 1) accurateness of the plots
# 2) choice of an appropriate type of plot
# 3) labels and scaling of the axis

# 16 subplots where the ith row axis and jth column correspond to a measurement
# extract indices
setosa_idx = np.where(iris_1d[:] == "Iris-setosa")
versicolor_idx = np.where(iris_1d[:] == "Iris-versicolor")
virginica_idx = np.where(iris_1d[:] == "Iris-virginica")

# create subplot
fig, axes = plt.subplots(4,4, sharex = 'col', sharey = 'row')
for i in range(0,4):
    for j in range(0,4):
        axes[i,j].scatter(iris_2d[setosa_idx, j], iris_2d[setosa_idx, i], c = 'b', lab
        axes[i,j].scatter(iris_2d[versicolor_idx, j], iris_2d[versicolor_idx, i], c =
        axes[i,j].scatter(iris_2d[virginica_idx, j], iris_2d[virginica_idx, i], c = 'g
```

```
# label appropriately
axes[0,0].legend()
fig.suptitle('Comparing measurements in the Iris dataset')
axes[0,0].set_title('sepallength')
axes[0,0].set_ylabel('sepallength')
axes[0,1].set_title('sepalwidth')
axes[1,0].set_ylabel('sepalwidth')
axes[0,2].set_title('petallength')
axes[2,0].set_ylabel('petallength')
axes[0,3].set_title('petalwidth')
axes[3,0].set_ylabel('petalwidth')
#plt.savefig('hw3_5.pdf', format='pdf')
plt.show()
```

The program determined to accept the null hypothesis.



4. Number 6.

   (a)
```
## Stastics for Data Science
# HW3 #6
# Tyler Olivieri

from scipy.stats import norm
import math
from matplotlib import pyplot as plt
import numpy as np
import csv
```

```
import os
import scipy.stats as st

#########################################################################
# (a) for experiment i, state your null and alternative hypothesis
#
# Since we will accept the change if the experiment increases performance,
# we don't care about a two-sided test
# We will be confident the experiment increases performance if the null hypothesis
# rejected
# Ho: average number of minutes in experiment i = 49.75
# Ha: average number of minutes in experiment i > 49.75
#
# p-value = Pr{observation contradicts null hypothesis more} = Pr{Z >= z | Ho true}
# where t is t test (current obeservation)
# p-value = 1 - Pr{Z <= z | Ho true} = 1 - std_gaussian_cdf(z)
# we can use the z-test/ standard gaussian because we have large n for each experin
#

#########################################################################
# (b) Find the experiments that have a p-value less than alpha = .05
#
alpha = .05
time_mean = 49.75
action_mean = 31.3
dir_str = "/home/snazyman/stat_ds/data_science/hw3/experiments"
directory = os.fsencode(dir_str)
null_rej_time = []

# loop over experiments directory
for file in os.listdir(directory):
    filename = os.fsdecode(file)

    # open experiment*.csv file
    if filename.endswith(".csv"):
        with open(dir_str + '/' + filename) as csvfile:
            data_time_list = []
            data_reader = csv.reader(csvfile)

            # read in the data from the experiment*.csv
            for row in data_reader:
                data_time_list.append(row[1])

            # remove first entry - it is not data but descriptor
            del data_time_list[0]

            data_time_array = np.array(data_time_list,dtype='float')

            # compute z-score
            # ddof=1 uses 1/n-1 instead of 1/n in variance calculation
            z_time = (np.mean(data_time_array) - time_mean) / math.sqrt(np.var(data
```

```
            # compute p value
            p_value_time = 1 - st.norm.cdf(z_time)

            # compare to alpha
            # save the experiments that reject Ho
            if p_value_time < alpha:
                null_rej_time.append(filename)

# compute the probability that you find a significant result due to chance under ar
# Pr{finding at least one signifcant result} = 1 - Pr{finding no signicant result}
# Pr{finding no signifcant result} = Pr{first test gives no significant result and
# Pr{finding at least one signifcant result} = 1 - (1 - alpha)^n
num_hypothesis = 70
p_sig_chance = 1 - (1-alpha)**num_hypothesis

print(null_rej_time)
print(p_sig_chance)

# This poses a problem because the probability of finding a significant result is n

##############################################################################
# (c)
# Apply the bonferroni correction, set cut-off significance to alpha/m
# find significant experiments

bonferroni_alpha = .05/num_hypothesis
bonferroni_null_rej_time = []

# loop over experiments directory
for file in os.listdir(directory):
    filename = os.fsdecode(file)

    # open experiment*.csv file
    if filename.endswith(".csv"):
        with open(dir_str + '/' + filename) as csvfile:
            data_time_list = []
            data_reader = csv.reader(csvfile)

            # read in the data from the experiment*.csv
            for row in data_reader:
                data_time_list.append(row[1])

            # remove first entry - it is not data but descriptor
            del data_time_list[0]

            data_time_array = np.array(data_time_list,dtype='float')

            # compute z-score
            # ddof=1 uses 1/n-1 instead of 1/n in variance calculation
            z_time = (np.mean(data_time_array) - time_mean) / math.sqrt(np.var(data
```

```
            # compute p value
            p_value_time = 1 - st.norm.cdf(z_time)

            # compare to alpha
            # save the experiments that reject Ho
            if p_value_time < bonferroni_alpha:
                bonferroni_null_rej_time.append(filename)

p_sig_chance = 1 - (1-bonferroni_alpha)**num_hypothesis
print(bonferroni_null_rej_time)
print(p_sig_chance)

################################################################################
# (d)
# Implement Holm-Bonferroni procedure
# find significant experiments

hb_null_rej_time = []

exp_idx = 0
p_value_array = np.zeros(num_hypothesis)
exp_array = []

# loop over experiments directory
for file in os.listdir(directory):
    filename = os.fsdecode(file)

    # open experiment*.csv file
    if filename.endswith(".csv"):
        with open(dir_str + '/' + filename) as csvfile:
            data_time_list = []

            data_reader = csv.reader(csvfile)

            # read in the data from the experiment*.csv
            for row in data_reader:
                data_time_list.append(row[1])


            # remove first entry - it is not data but descriptor
            del data_time_list[0]

            data_time_array = np.array(data_time_list,dtype='float')

            # compute z-score
            # ddof=1 uses 1/n-1 instead of 1/n in variance calculation
            z_time = (np.mean(data_time_array) - time_mean) / math.sqrt(np.var(data

            # compute p value
            p_value_time = 1 - st.norm.cdf(z_time)
```

```
                p_value_array[exp_idx] = p_value_time
                exp_array.append(filename)

                exp_idx = exp_idx + 1

    # order p values
    p_value_ordered_idx = p_value_array.argsort()

    # apply Holm-Bonferroni procedure to check if we should reject Ho[k]
    k = 0
    while (p_value_array[p_value_ordered_idx[k]] <= (alpha/(num_hypothesis+1-k))):

        # reject null hypothesis k
        hb_null_rej_time.append(exp_array[p_value_ordered_idx[k]])

        k = k + 1

        # if k = num_hypothesis we rejected all null hypothesis and can break out of lo
        # removes accessing (p_value_array[p_value_ordered_idx[num_hypothesis]] which
        if k == num_hypothesis:
            break

    print(hb_null_rej_time)
```

(b) The following list had experiments with p-value less than α in (b):

For the time hypothesis: ['experiment49.csv', 'experiment44.csv', 'experiment24.csv', 'experiment16.csv', 'experiment1.csv', 'experiment69.csv', 'experiment5.csv', 'experiment36.csv', 'experiment61.csv', 'experiment47.csv', 'experiment27.csv', 'experiment6.csv', 'experiment15.csv', 'experiment48.csv', 'experiment40.csv']

The probability of finding a significant result due to chance was 0.972416309563225

(c) After applying the bonferroni correction, the following lists had experiments with p-value less than the corrected α

For the time hypothesis: ['experiment49.csv', 'experiment24.csv', 'experiment16.csv', 'experiment1.csv', 'experiment69.csv', 'experiment5.csv', 'experiment61.csv', 'experiment27.csv', 'experiment6.csv', 'experiment48.csv']

Now the probability of finding a significant result with the correction due to chance is 0.04878756968022091, which is slightly below α. This is good!

(d) Finally, applying the Holm-Bonferroni procedure gave

For the time hypothesis: ['experiment49.csv', 'experiment5.csv', 'experiment24.csv', 'experiment48.csv', 'experiment1.csv', 'experiment69.csv', 'experiment27.csv', 'experiment16.csv', 'experiment61.csv', 'experiment6.csv']

This rejected the same experiments as the bonferonni, but has it ordered with respect to p-value