Midterm 2

# -\*- coding: utf-8 -\*-  
"""  
Created on Sun Nov 13 14:00:27 2016  
  
@author: Kanak  
"""  
  
##################################################################  
#  
# Problem 1 Part 1  
#  
##################################################################  
  
  
import pandas as pd  
import numpy as np  
import math  
  
df = pd.read\_csv("D:/UCF/STA 6106 Statistical Computing/Assignments/Midterm 2/Exam2\_pb1.txt", header=None, names=["a"])  
  
def boot\_bias\_sde(x, boottimes = 10, method = 'Mean', size = None):  
 if not isinstance(x, pd.DataFrame):  
 print ("x is not a Data Frame")  
 return  
 if size == None:  
 size = x.shape[0]  
 if method.upper() == 'MEAN':  
 theta = x.mean()  
 thetab = pd.DataFrame([(x.sample(n = size, replace = True)).mean() \  
 for \_ in range(boottimes)])  
 if method.upper() == 'MEDIAN':  
 theta = x.median()  
 thetab = pd.DataFrame([(x.sample(n = size, replace = True)).median() \  
 for \_ in range(boottimes)])  
 botse = thetab.std()  
 bias = (thetab - theta).mean()  
 return {"Parameter": theta.values.flatten(),   
 "Bias": bias.values.flatten(),  
 "Standard Error": botse.values.flatten()}  
   
   
a = boot\_bias\_sde(df, boottimes = 9999, method = 'median', size = None)  
  
print (a)  
  
   
##################################################################  
#  
# Problem 1 Part 2  
#  
##################################################################  
  
def jackknife(x, method = 'Mean'):  
 if not isinstance(x, pd.DataFrame):  
 print ("x is not a Data Frame")  
 return  
 n = x.shape[0]  
 if method.upper() == 'MEAN':  
 theta = x.mean()  
 thetab = pd.DataFrame([(x.drop(i)).mean() for i in range(n)])  
 if method.upper() == 'MEDIAN':  
 theta = x.median()  
 thetab = pd.DataFrame([(x.drop(i)).median() for i in range(n)])  
   
 jkse = ((n-1)/math.sqrt(n))\*thetab.std()  
 bias = (n-1)\*(thetab.mean() - theta)  
   
  
 return {"Parameter": theta.values.flatten(),   
 "Bias": bias.values.flatten(),  
 "Standard Error": jkse.values.flatten()}  
   
jk = jackknife(df, method = "mean")  
  
print(jk)  
  
##################################################################  
#  
# Problem 1 Part 3  
#  
##################################################################  
  
  
def boot\_ci(x, alpha = 0.05, boottimes = 500, r = 100, method = 'Mean'):  
 if not isinstance(x, pd.DataFrame):  
 try:  
 x = pd.DataFrame(x)  
 except:  
 print ("x is not possible to convert as Data Frame")  
 return  
 size = x.shape[0]  
  
 def boot\_se(y, r, method, size):  
 if method.upper() == 'MEAN':  
 thetab = pd.DataFrame([(y.sample(n = size, replace = True)).mean() \  
 for \_ in range(r)])  
 if method.upper() == 'MEDIAN':  
 thetab = pd.DataFrame([(y.sample(n = size, replace = True)).median() \  
 for \_ in range(r)])  
 return thetab.std().values.flatten()  
   
 se = []  
 thetab = []  
 if method.upper() == 'MEAN':  
 theta = x.mean().values.flatten()  
 for \_ in range(boottimes):  
 y =x.sample(n = size, replace = True)  
 se.append(boot\_se(y, r, method, size))  
 thetab.append(y.mean().values.flatten())   
  
 if method.upper() == 'MEDIAN':  
 theta = x.median().values.flatten()  
 for \_ in range(boottimes):  
 y =x.sample(n = size, replace = True)  
 se.append(boot\_se(y, r, method, size))  
 thetab.append(y.median())   
   
  
 t = pd.DataFrame((thetab - np.mean(thetab))/ se)  
 thetab = pd.DataFrame(thetab)  
 sd = thetab.std().values.flatten()  
 qinterval = thetab.quantile(q = (alpha/2, 1-alpha/2))  
 tqt = t.quantile(q = (alpha/2, 1-alpha/2)).abs().values.flatten()  
 bci = theta + (tqt \* [-1, 1] \* sd)  
 bias = (thetab - theta).mean()  
 ci = {"Parameter": theta,  
 "Bootstrap CI": bci,   
 "Percentile Interval": qinterval.values.flatten(),  
 "Bias": bias.values.flatten(),  
 "Standard Error": sd}  
 return ci  
   
  
ci = boot\_ci(df, alpha = 0.05, boottimes = 5000, method = 'median')  
  
print(ci)  
  
##################################################################  
#  
# Problem 3  
#  
##################################################################  
# alternative way  
  
# -\*- coding: utf-8 -\*-  
"""  
Created on Wed Nov 9 23:58:23 2016  
  
@author: Kanak  
"""  
import numpy as np  
import pandas as pd  
x = [28, -44, 29, 30, 26, 27, 22, 23, 33, 16, 24, 40, 21, 31, 34, -2, 25, 19]  
  
def tstat(x, theta = None, sigsq = None):  
 if theta is None:  
 theta = x.mean()  
 if sigsq is None:  
 sigsq = x.var()  
   
 return (((x-theta)\*\*2)/sigsq)  
  
def baggging(x, size = None, alpha = 0.05, boottimes = 100):  
 if not isinstance(x, pd.DataFrame):  
 try:  
 x = pd.DataFrame(x)  
 except:  
 print ("x is not possible to convert as Data Frame")  
 return  
 if size == None:  
 size = x.shape[0]  
 hi = []  
 for \_ in range(boottimes):  
 y = x.sample(n = size, replace = True)  
 m = y.mean().values.flatten()  
 sd = y.var().values.flatten()  
 t = tstat(y, theta = m, sigsq = sd)  
 hi.append(t.quantile(q = 1-alpha).values.flatten())  
 h = pd.DataFrame(hi).mean()  
   
 return {"h": h.values.flatten(),  
 "Mean": x.mean().values.flatten(),  
 "Variance": x.var().values.flatten()}  
  
def predictoutlier(model, x):  
 par = model['h']  
 m = model["Mean"]  
 sd = model["Variance"]  
 t = np.array(tstat(x, m, sd))  
 if np.ndarray.min(t) < 0 or np.ndarray.max(t) > np.ndarray.min(par):  
 out = "Outlier"  
 com = ("The observation {} with t-value {} is an OUTLIER because"+  
 " given value does not fall between 0 and {}").format(x, t, par)  
 print(com)  
 else:  
 out = "Not Outlier"  
 com = ("The observation {} with t-value {} is NOT OUTLIER" +  
 " because given value falls between 0 and {}").format(x, t, par)  
 print(com)  
 return {"h": par,  
 "Mean": m,  
 "Variance": sd,  
 "t": t,   
 "Decision": out,   
 "Comment": com}   
   
   
model = baggging(x, boottimes = 9999)  
print(model)  
  
pred = predictoutlier(model, 38)  
print(pred)

## {'Standard Error': array([ 58.39764717]), 'Bias': array([ 7.9309931]), 'Parameter': array([ 331.5])}  
## {'Standard Error': array([ 50.79081714]), 'Bias': array([ 0.]), 'Parameter': array([ 437.20967742])}  
## {'Bias': array([ 8.4198]), 'Standard Error': array([ 57.71608959]), 'Bootstrap CI': array([ 206.51556028, 430.79777328]), 'Percentile Interval': array([ 224.5, 454. ]), 'Parameter': array([ 331.5])}  
## {'Variance': array([ 343.59477124]), 'h': array([ 4.01523935]), 'Mean': array([ 21.22222222])}  
## The observation 38 with t-value [ 0.81926109] is NOT OUTLIER because given value falls between 0 and [ 4.01523935]  
## {'h': array([ 4.01523935]), 'Comment': 'The observation 38 with t-value [ 0.81926109] is NOT OUTLIER because given value falls between 0 and [ 4.01523935]', 'Mean': array([ 21.22222222]), 'Variance': array([ 343.59477124]), 'Decision': 'Not Outlier', 't': array([ 0.81926109])}