Code Reference - American Nutritional Differences

Austin Brown

Colorado State University

MIS581

Dr. Chung

9/10/2023

#GITHUB: https://github.com/ajjbrown9/MSCapstone

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph of a graph

Description automatically generated with medium confidence

#GITHUB: https://github.com/ajjbrown9/MSCapstone

#%%

import pandas as pd

import numpy as np

import matplotlib as mplt

import matplotlib.pyplot as plt

import seaborn as sbn

import scipy.stats as stats

import os

#========================================================================================================

#Load Data

rawData = pd.read\_csv('C:/Users/Austin/Downloads/en.openfoodfacts.org.products.csv', sep='\t', nrows=1900210)

#rawData.dropna(inplace=True)

# desc = data.describe()

# head = data.head(20)

# #print(data.head(n=100))

# print(desc)

# print(head)

#========================================================================================================

#Create Working Dataframe

data = pd.DataFrame(rawData)

frame\_Working = data.filter(['code',

                            'product\_name',

                            # 'categories',# (dummy/parse needed')

                            # 'cateogries\_tags',# (dummy/parse needed')

                            # 'origins',

                            # 'categories\_en', #(dummy/parse needed

                            # 'origins\_tags',

                            # 'origins\_en',

                            # 'countries',

                            'countries\_en',

                            # 'labels', #(dummy/parse needed')

                            # 'labels\_tags',# (dummy/parse needed')

                            # 'labels\_en',# (dummy/parse needed')

                            # 'quantity',

                            # 'ingredients\_text',

                            # 'ingredients\_tags',

                            # 'ingredients\_analysis\_tags',

                            # 'allergens',

                            'additives\_n',

                            # 'additives',

                            # 'additives\_tags',

                            # 'additives\_en',

                            # 'pnns\_groups\_1',

                            # 'pnns\_groups\_2',

                            # 'food\_groups',

                            # 'food\_groups\_tags',

                            # 'food\_groups\_en',

                            'data\_quality\_errors\_tags',# (consider using rows only where this is null')

                            # 'main\_category',

                            # 'main\_category\_en',

                            'energy-kcal\_100g',

                            # 'energy\_100g',

                            # 'fat\_100g',

                            # 'saturated-fat\_100g',

                            # 'carbohydrates\_100g',

                            'sugars\_100g',

                            # 'fiber\_100g',

                            # 'proteins\_100g',

                            # 'salt\_100g',

                            'sodium\_100g',

                            # 'vitamin-a\_100g',

                            # 'vitamin-b1\_100g',

                            # 'vitamin-b2\_100g',

                            # 'vitamin-d\_100g',

                            # 'vitamin-e\_100g',

                            # 'vitamin-k\_100g',

                            # 'vitamin-c\_100g',

                            # 'vitamin-b6\_100g',

                            # 'vitamin-b9\_100g',

                            # 'vitamin-pp\_100g',

                            # 'vitamin-b12\_100g',

                            # 'folates\_100g',

                            # 'calcium\_100g',

                            # 'potassium\_100g',

                            # 'iron\_100g',

                            # 'magnesium\_100g',

                            # 'zinc\_100g',

                            # 'fruits-vegetables-nuts-estimate-from-ingredients\_100g',

                            # 'nutritional\_score-fr\_100g'

                            ], axis=1)

frame\_Working = frame\_Working[frame\_Working['data\_quality\_errors\_tags'].isnull()]

# frame\_Working = frame\_Working.loc[frame\_Working['additives\_n'].isin(range(0,100))] # Bad record

# frame\_Working = frame\_Working.loc[frame\_Working['additives\_n'].isin(np.array([0,1,2,3,4,5,6,7,8,9,10,

#                                                                               11,12,13,14,15,16,17,18,19,20,

#                                                                               21,22,23,24,25,26,27,28,29,30,

#                                                                               31,32,33,34,35,36,37,38,39,40]))] # Bad record

# frame\_Working = frame\_Working.loc[frame\_Working['code'] != 3596710517466] # Bad record

# print(frame\_Working.describe)

# frame\_Working.dropna(inplace=True)

# print(frame\_Working.describe)

#========================================================================================================

#Plot Configurations

fig1, ax\_sug = plt.subplots()

fig2, ax\_add = plt.subplots()

fig3, ax\_sod = plt.subplots()

fig4, ax\_cal = plt.subplots()

ax\_sug.set\_xlim(0,125)

ax\_sug.set\_ylim(0,15000)

ax\_add.set\_xlim(0,20)

ax\_add.set\_ylim(0,150000)

ax\_sod.set\_xlim(0,50)

ax\_sod.set\_ylim(0,150000)

ax\_cal.set\_xlim(0,1000)

ax\_cal.set\_ylim(0,7500)

#========================================================================================================

#frame\_Additives row 838075 needs to be dropped

frame\_Additives = frame\_Working.filter(['code','product\_name','countries\_en','additives\_n'],axis=1)

frame\_Additives.dropna(inplace=True)

frame\_Additives = frame\_Additives.loc[frame\_Additives['additives\_n'].isin(range(0,1000))]

# frame\_Additives['additives\_n'] = frame\_Additives['additives\_n'].astype('int')

frame\_Additives = pd.DataFrame(frame\_Additives.assign(isAmerica= np.where(frame\_Additives['countries\_en'] == 'United States', 1, 0)))

average\_Additives = pd.DataFrame(frame\_Additives.groupby(['isAmerica']).mean(['additives\_n']))

sum\_Additives = frame\_Additives.groupby(['isAmerica']).sum(['additives\_n']).astype(float)

count\_Additives = frame\_Additives[frame\_Additives.columns[0]].count()

count\_AdditivesAmerica = frame\_Additives.groupby(['isAmerica']).count()

#Statistics

isAmerica = frame\_Additives['isAmerica'] == 1 # Set condition

#Create new column based on condition to derive new array for statitical tests H0

series\_AdditiveAmerica = frame\_Additives

series\_AdditiveAmerica = series\_AdditiveAmerica.assign(isAmericaAddSeries = (np.where(series\_AdditiveAmerica['isAmerica']==1, series\_AdditiveAmerica['additives\_n'], None)) )

series\_AdditiveAmerica.dropna(inplace=True)

series\_AdditiveAmerica = np.array(series\_AdditiveAmerica['isAmericaAddSeries'])

# series\_AdditiveAmerica = pd.to\_numeric(series\_AdditiveAmerica['additives\_n'])

# series\_AdditiveAmerica['additives\_n'] = series\_AdditiveAmerica['additives\_n'].astype('int')

# print(frame\_Additives)

#Create new column based on condition to derive new array for statitical tests H1

series\_AdditiveNonAmerica = frame\_Additives

series\_AdditiveNonAmerica = series\_AdditiveNonAmerica.assign(notAmericaAddSeries = (np.where(series\_AdditiveNonAmerica['isAmerica']==0, series\_AdditiveNonAmerica['additives\_n'], None)) )

series\_AdditiveNonAmerica.dropna(inplace=True)

series\_AdditiveNonAmerica = np.array(series\_AdditiveNonAmerica['notAmericaAddSeries'])

# series\_AdditiveNonAmerica = pd.to\_numeric(series\_AdditiveAmerica['additives\_n'])

# series\_AdditiveNonAmerica['additives\_n'] = series\_AdditiveAmerica['additives\_n'].astype('int')

# print(frame\_Additives)

# print(series\_AdditiveNonAmerica)

#OUTPUT

print('\n\n\n\n\n===================================================================')

print('Additives Describe Function')

print(frame\_Additives.describe)

print('\nAverage Additives: Non-United States')

print(series\_AdditiveNonAmerica.mean())

print('\nAverage Additives: United States')

print(series\_AdditiveAmerica.mean())

print('\nAdditive Variance: Non-United States')

print(series\_AdditiveNonAmerica.var())

print('\nAdditive Variance: United States')

print(series\_AdditiveAmerica.var())

print('\nTotal Additive Count')

print(count\_Additives)

print('\nAdditive Count: United States vs Non-United States')

print(count\_AdditivesAmerica)

#Two Sample t-test

print('\nH0: μ(Average Count U.S. Food Additives) ≤  μ(Average Count Non-U.S. Food Additives)\n')

test = stats.ttest\_ind(a=series\_AdditiveAmerica, b=series\_AdditiveNonAmerica, alternative='greater' ,equal\_var=False)

print(test)

#Additive Visualization

sbn.histplot(data=frame\_Additives, x = 'additives\_n', color = 'blue', alpha = 1, binwidth=2, ax=ax\_add, kde=True)

# #========================================================================================================

# frame\_Sugars

frame\_Sugars = frame\_Working.filter(['code','product\_name','countries\_en','sugars\_100g'],axis=1)

frame\_Sugars.dropna(inplace=True)

frame\_Sugars = frame\_Sugars.assign(isAmerica= np.where(frame\_Sugars['countries\_en'] == 'United States', 1, 0))

average\_Sugars = frame\_Sugars.groupby(['isAmerica']).mean(['sugars\_100g'])

count\_Sugars = frame\_Sugars[frame\_Sugars.columns[0]].count()

count\_SugarsAmerica = frame\_Sugars.groupby(['isAmerica']).count()

#Statistics

isAmerica = frame\_Sugars['isAmerica'] == 1 # Set condition

#Create new column based on condition to derive new array for statitical tests H0

series\_SugarsAmerica = frame\_Sugars

series\_SugarsAmerica = series\_SugarsAmerica.assign(isAmericaSugSeries = (np.where(series\_SugarsAmerica['isAmerica']==1, series\_SugarsAmerica['sugars\_100g'], None)) )

series\_SugarsAmerica.dropna(inplace=True)

series\_SugarsAmerica = np.array(series\_SugarsAmerica['isAmericaSugSeries'])

#Create new column based on condition to derive new array for statitical tests H1

series\_SugarNonAmerica = frame\_Sugars

series\_SugarNonAmerica = series\_SugarNonAmerica.assign(notAmericaSugSeries = (np.where(series\_SugarNonAmerica['isAmerica']==0, series\_SugarNonAmerica['sugars\_100g'], None)) )

series\_SugarNonAmerica.dropna(inplace=True)

series\_SugarNonAmerica = np.array(series\_SugarNonAmerica['notAmericaSugSeries'])

#OUTPUT

print('\n\n\n\n\n===================================================================')

print('Sugars Describe Function')

print(frame\_Sugars.describe)

print('\nAverage Sugars: Non-United States')

print(series\_SugarNonAmerica.mean())

print('\nAverage Sugars: United States')

print(series\_SugarsAmerica.mean())

print('\nSugars Variance: Non-United States')

print(series\_SugarNonAmerica.var())

print('\nSugars Variance: United States')

print(series\_SugarsAmerica.var())

print('\nTotal Sugars Count')

print(count\_Sugars)

print('\nSugars Count: United States vs Non-United States')

print(count\_SugarsAmerica)

#Two Sample t-test

print('\nH0: μ(Average U.S. 100g Serving Sugar Count) ≤ μ(Average Non-U.S. 100g Serving Sugar Count)\n')

test = stats.ttest\_ind(a=series\_SugarsAmerica, b=series\_SugarNonAmerica, alternative='greater' ,equal\_var=False)

print(test)

#Sugars Visualizations

sbn.histplot(data=frame\_Sugars, x = 'sugars\_100g', color = 'red', alpha = 1, binwidth=1, ax=ax\_sug, kde=True)

#========================================================================================================

# frame\_Sodium

frame\_Sodium = frame\_Working.filter(['code','product\_name','countries\_en','sodium\_100g'],axis=1)

frame\_Sodium.dropna(inplace=True)

frame\_Sodium = frame\_Sodium.assign(isAmerica= np.where(frame\_Sodium['countries\_en'] == 'United States', 1, 0))

average\_Sodium = frame\_Sodium.groupby(['isAmerica']).mean(['sodium\_100g'])

count\_Sodium = frame\_Sodium[frame\_Sodium.columns[0]].count()

count\_SodiumAmerica = frame\_Sodium.groupby(['isAmerica']).count()

#Statistics

isAmerica = frame\_Sodium['isAmerica'] == 1 # Set condition

#Create new column based on condition to derive new array for statitical tests H0

series\_SodiumAmerica = frame\_Sodium

series\_SodiumAmerica = series\_SodiumAmerica.assign(isAmericaSodSeries = (np.where(series\_SodiumAmerica['isAmerica']==1, series\_SodiumAmerica['sodium\_100g'], None)) )

series\_SodiumAmerica.dropna(inplace=True)

series\_SodiumAmerica = np.array(series\_SodiumAmerica['isAmericaSodSeries'])

#Create new column based on condition to derive new array for statitical tests H1

series\_SodiumNonAmerica = frame\_Sodium

series\_SodiumNonAmerica = series\_SodiumNonAmerica.assign(notAmericaSodSeries = (np.where(series\_SodiumNonAmerica['isAmerica']==0, series\_SodiumNonAmerica['sodium\_100g'], None)) )

series\_SodiumNonAmerica.dropna(inplace=True)

series\_SodiumNonAmerica = np.array(series\_SodiumNonAmerica['notAmericaSodSeries'])

#OUTPUT

print('\n\n\n\n\n===================================================================')

print('Sodium Describe Function')

print(frame\_Sodium.describe)

print('\nAverage Sodium: Non-United States')

print(series\_SodiumNonAmerica.mean())

print('\nAverage Sodium: United States')

print(series\_SodiumAmerica.mean())

print('\nSodium Variance: Non-United States')

print(series\_SodiumNonAmerica.var())

print('\nSodium Variance: United States')

print(series\_SodiumAmerica.var())

print('\nTotal Sodium Count')

print(count\_Sodium)

print('\nSodium Count: United States vs Non-United States')

print(count\_SodiumAmerica)

#Two Sample t-test

print('\nH0: μ(Average U.S. 100g Serving Sodium Count) ≤ μ(Average Non-U.S. 100g Serving Sodium Count)\n')

test = stats.ttest\_ind(a=series\_SodiumAmerica, b=series\_SodiumNonAmerica, alternative='greater' ,equal\_var=False)

print(test)

#Sodium Visualizations

sbn.histplot(data=frame\_Sodium, x = 'sodium\_100g', color = 'green', alpha = 1, binwidth=1, ax=ax\_sod, kde=True)

#========================================================================================================

# frame\_Calories

frame\_Calories = frame\_Working.filter(['code','product\_name','countries\_en','energy-kcal\_100g'],axis=1)

frame\_Calories.dropna(inplace=True)

frame\_Calories = frame\_Calories.assign(isAmerica= np.where(frame\_Calories['countries\_en'] == 'United States', 1, 0))

average\_Calories = frame\_Calories.groupby(['isAmerica']).mean(['energy-kcal\_100g'])

count\_Calories = frame\_Calories[frame\_Calories.columns[0]].count()

count\_CaloriesAmerica = frame\_Calories.groupby(['isAmerica']).count()

#Statistics

isAmerica = frame\_Calories['isAmerica'] == 1 # Set condition

#Create new column based on condition to derive new array for statitical tests H0

series\_CaloriesAmerica = frame\_Calories

series\_CaloriesAmerica = series\_CaloriesAmerica.assign(isAmericaCalSeries = (np.where(series\_CaloriesAmerica['isAmerica']==1, series\_CaloriesAmerica['energy-kcal\_100g'], None)) )

series\_CaloriesAmerica.dropna(inplace=True)

series\_CaloriesAmerica = np.array(series\_CaloriesAmerica['isAmericaCalSeries'])

#Create new column based on condition to derive new array for statitical tests H1

series\_CaloriesNonAmerica = frame\_Calories

series\_CaloriesNonAmerica = series\_CaloriesNonAmerica.assign(notAmericaCalSeries = (np.where(series\_CaloriesNonAmerica['isAmerica']==0, series\_CaloriesNonAmerica['energy-kcal\_100g'], None)) )

series\_CaloriesNonAmerica.dropna(inplace=True)

series\_CaloriesNonAmerica = np.array(series\_CaloriesNonAmerica['notAmericaCalSeries'])

#OUTPUT

print('\n\n\n\n\n===================================================================')

print('Calories Describe Function')

print(frame\_Calories.describe)

print('\nAverage Calories: Non-United States')

print(series\_CaloriesNonAmerica.mean())

print('\nAverage Calories: United States')

print(series\_CaloriesAmerica.mean())

print('\nCalories Variance: Non-United States')

print(series\_CaloriesNonAmerica.var())

print('\nCalories Variance: United States')

print(series\_CaloriesAmerica.var())

print('\nTotal Calories Count')

print(count\_Calories)

print('\nCalories Count: United States vs Non-United States')

print(count\_CaloriesAmerica)

#Two Sample t-test

print('\nH0: μ(Average U.S. 100g Serving Caloric Count) ≤ μ(Average Non-U.S. 100g Serving Caloric Count)\n')

test = stats.ttest\_ind(a=series\_CaloriesAmerica, b=series\_CaloriesNonAmerica, alternative='greater' ,equal\_var=False)

print(test)

#Calories Visualizations

sbn.histplot(data=frame\_Calories, x = 'energy-kcal\_100g', color = 'purple', alpha = 1, binwidth=1, ax=ax\_cal, kde=True)