**Exam Notes**

**Class Notes**

1. [Class 5. Introduction to Python: data structures, control flow with NOTES URL](https://nbviewer.jupyter.org/github/Parandzem-S/MGMT300-Summer2021/blob/main/Class%205.%20Introduction%20to%20Python/Class%205.%20Python%20Basics.ipynb)
2. [Class 6: Functions, While loop, Pandas, Matplotlib URL](https://github.com/Parandzem-S/MGMT300-Summer2021/blob/main/Class%206.%20Functions%2C%20While%20loop%2C%20Pandas%2C%20Matplotlib/Class%206.%20Functions%2C%20While%20loop%2C%20Pandas%2C%20Matplotlib.ipynb)
3. [Class 7. Data Visualization, Reading Data from Web, Interactive Plotting with Notes](https://nbviewer.jupyter.org/github/Parandzem-S/MGMT300-Summer2021/blob/main/Class%207.%20Data%20Visulization%2C%20Reading%20Data%20from%20Web%2C%20Interactive%20Plotting/Class%207.%20Data%20Visualization%2C%20Reading%20Data%20from%20Web%2C%20Interactive%20Plotting%20Notes.ipynb)

[URL](https://nbviewer.jupyter.org/github/Parandzem-S/MGMT300-Summer2021/blob/main/Class%207.%20Data%20Visulization%2C%20Reading%20Data%20from%20Web%2C%20Interactive%20Plotting/Class%207.%20Data%20Visualization%2C%20Reading%20Data%20from%20Web%2C%20Interactive%20Plotting%20Notes.ipynb)

1. [Class 8. Statistical Tests from Scipy and Overview of Section 2 with NOTES URL](https://github.com/Parandzem-S/MGMT300-Summer2021/blob/main/Class%208%20Statistical%20Tests%20and%20Overview/Class%208.%20Statistical%20Tests%20from%20Scipy%20and%20Overview.ipynb)
2. [Basic Charts](https://nbviewer.jupyter.org/github/Parandzem-S/Blog_metric.am/blob/master/blog1_ps_matplotlib.ipynb)
3. HW3 - <https://nbviewer.jupyter.org/github/Parandzem-S/MGMT300-Summer2021/blob/main/Homework%20Solution/Homework%203.%20Solution%20Final.ipynb>
4. HW4 - <https://nbviewer.jupyter.org/github/Parandzem-S/MGMT300-Summer2021/blob/df6401f75e5ef87e8c699beb5e29ff048dc99889/Homework%20Solution/Homework%204.%20Solution.ipynb>

**Libraries to Import**

!pip install quandl

!pip install plotly

!pip install pandas\_datareader

import pandas as pd

from pandas\_datareader import data as dt #will be used to download data from yahoo finance

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import quandl

from plotly.offline import iplot

from plotly.offline import init\_notebook\_mode

import plotly.graph\_objects as go

from scipy.stats import pearsonr

from scipy.stats import ttest\_ind

from scipy.stats import f\_oneway

from scipy.stats import levene

init\_notebook\_mode(connected=True)

**ADD DATA + SIMPLE STUFF**

wine\_df.info()

data\_car=pd.read\_excel('car\_sales.xls')

data=pd.read\_excel("Data.xlsx",sheet\_name="Data")

data\_salary=pd.read\_csv('salary.csv')

data.head(10)

data.shape

*#descriptive stat for numeric variables*

data\_imdb.describe()

For only one variable

data\_salary['race'].describe()

data\_imdb.revenue.describe()

data\_salary['race'].value\_counts()

*#correlation between numeric variables*

data\_imdb.corr()

*#extracting unique values*

wine\_df.Varietal.unique()

*#number of unique values*

wine\_df.Varietal.nunique()

*#checking missing values*

wine\_df.isnull().sum()

#grouping data by object type variables

data\_imdb.groupby(['status'])['vote\_average'].mean()

#unique values in status variable

data\_imdb['status'].unique()

Q: How many columns have **missing values**? data.isna().any() + data.isna().sum()

Q: **Drop** missing values. data.dropna(inplace=True) Check with: data.isna().any()

Q: How many numeric and how many non-numeric variables are in the data?

print('Number of numeric variables: ',len(affairs\_data.select\_dtypes(exclude='object').columns))

print('Number of object variables: ',len(affairs\_data.select\_dtypes(include='object').columns))

Q: If you have noticed column “workclass” contains lots of ‘?’ signs instead of blank values. Replace all ‘?’ with NA values. (0.5 points)

salary\_data['workclass'] = salary\_data['workclass'].replace('?', np.NaN)

salary\_data.workclass.head()

Q: Dropping 2 variables in one line -

DATA.CLEAN= data.drop(["Rent\_Amount","floor"],axis=1,inplace=True)

#dropping row from dataframe

data\_clean.drop(index=0,axis=0,inplace=True)

data\_clean.head()

Q: What is the gender of a customer whose ID number is 904? How many years has the same person been married?

affairs\_data[affairs\_data.ID==904][['gender','years married']]

Q: What is the average age of people who have affair (affairs= “Yes” if a person had an affair and “No” if a person did not have an affair)?

print('Average age of people who have affair: ',affairs\_data[affairs\_data.affairs=='Yes'].age.mean())

**Pivot table** example - round(pd.pivot\_table(data,index='furniture',values='area', aggfunc=['mean','count']),2)

Separate from Discount to Gender to Female - wine\_df['Discount'][wine\_df['Gender']=='Female'].head()

print("Average age is: ",round(salary\_data.age.mean(),1))

print("Median age is: ",round(salary\_data.age.median(),1))

print("Minimum age is: ",round(salary\_data.age.min(),1))

print("Maximum age is: ",round(salary\_data.age.max(),1))

print("Standard deviation of age is: ",round(salary\_data.age.std(),1))

**Defining simple IF function**

def house\_type(data):

if data["furniture"]=="not furnished" and data["Animal\_Acceptance"]=="not acept":

return "not interested in offer"

else:

return "interested"

**Finding specific data** - print(house\_type(data[data["House\_ID"] == 344].iloc[0]))

**For loop+interactive histogram**

for i in data[["area","rooms","Floor\_Cleaned","Rent\_Amount\_Cleaned"]].columns:

fig=go.Figure(data=[go.Histogram(x=data[i],

marker\_color='#bfb2b9',

nbinsx=5)])

fig.update\_layout(

title=i,

height=500,

width=500)

fig.show()

**Use for loop to visualize distribution of all numeric variables in the dataset** -

data\_numeric = salary\_data.select\_dtypes(exclude='object')

for i in data\_numeric.columns:

plt.hist(data\_numeric[i],

color='yellow',

bins=30

)

plt.title(f"Distribution of {i}")

plt.show()

**Data from Yahoo**

from pandas\_datareader import data

FB\_df=data.DataReader(name="FB",

data\_source='yahoo',

start='2019-08-03', end='2020-08-03')

**Av and median** - print("Average: ", round(FB\_df.Close.mean(),2))

print("Median: ", round(FB\_df.Close.median(),2))

**2 line chart (2 variables)**

plt.figure(figsize=(9,8))

sns.lineplot(data=FB\_df[["Open","Close"]],

palette=['#1d63f0','#eb213c'])

plt.show()

General Notes

numbers=[1,2,25,50,60,85]

print(np.mean(numbers))

print(np.max(numbers))

print(np.min(numbers))

print(np.median(numbers))

print(np.abs(-9))

print(np.sort(numbers))

#use ? sign to look for information about any function

?np.sort

numbers=[1,2,3,4,5]

rd.shuffle(numbers) # randomly reorders the elements in a list

print(numbers)

names=["PS","NM","TA"]

rd.shuffle(names) # randomly reorders the elements in a list

print(names)

#print value counts for last five variables in the dataset by using for loop

for i in data\_imdb.columns[-5:]:

print('\n',i,'\n')

print(data\_imdb[i].value\_counts())

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

**Print on one line**

print("My favorite movie is" ,variable1,end=' ')

print("which was directed by" ,variable2,end=' ')

print("and it was premiered in" ,variable3)

**OR**

Name=input()

Director=input()

Year=input()

print(f"My favorite movie is {Name} which was directed by {Director} and it was premiered in {Year}")

**Handling exceptions (errors)**

#let's see how we can handle the exception above

try:

x=int(input("x: "))

y=int(input("y: "))

except ValueError:

print ("Invalid input. Please enter a number.")

sys.exit(1) #exit program w/ status code of 1,

#i.e. the script to exit back to either the Pythonconsole or the command prompt

result=x\*y

print(f"{x}\*{y} = {result}")

**While loop - defining a function**

#let's define a function that will search a word in specified text

def find\_word(text,word):

index=0

text=text.split()

while index<len(text):

if word==text[index]:

return index

index=index+1

return 'Nothing has been found'

LOC and ILOC

Data\_imdb.loc[[0,12,5], :] - 3 rows, all columns

Data\_imdb.loc[0:12,'budget':'original\_title']

Data\_imdb.loc[0:12,['budget','original\_title']]

In iloc column indexes should be used

In LOC right part of the range is included. In ILOC it is not

Three way of returning movie title that has the highest revenue

data\_imdb[data\_imdb['revenue']==2787965087.0]['title']

data\_imdb[data\_imdb['revenue']==2787965087.0].title

data\_imdb.loc[data\_imdb['revenue']==2787965087.0,'title']

**Notes from HW3**

i=25

if i%2==0:#The % symbol is called the Modulo Operator. It returns the remainder of dividing the left hand operand by right hand operand.

#It's used to get the remainder of a division problem.

print(f'{i} is even')

else:

print(f'{i} is odd')

#first, create **list of wines**

wines= ["Thomas Fogarty 2018 Razorback Vineyard Pinot Noir",

"Migration 2018 Drum Canyon Vineyard Pinot Noir", "Zorah Voski 2018",

"Curto Marco 2017 Arborina (Barolo)","Calera 2018 Jensen Vineyard Pinot Noir", "Hin Areni Red 2014"]

for i in wines:

wine\_type=[] #create empty list of wine types

if "Pinot" in i:

wine\_type.append("Pinot Noir") #use append() function to add type of each wine to wine\_type list

elif "Areni" in i:

wine\_type.append("Areni") #use append() function to add type of each wine to wine\_type list

else:

wine\_type.append("Other") #use append() function to add type of each wine to wine\_type list

*for j in wine\_type: #use for loop to print wine types and wine names*

*print(i ,":",j)*

*\*in zip() don’t include “\_”s*

**Defining max function**

#defining list of numbers

#argument of the function is an any list of numbers

def max\_num\_in\_list(list\_num):

max\_num = list\_num[0] #take a first item of any inputted list and assign it as a max number

for a in list\_num: #then use for loop to compare each consecutive item in a list to a previous item

if a > max\_num: #if an item is higher than the previous item, then max number will become that item

max\_num = a

return max\_num

**Defining a complex function**

#defining a function

def satisfy():

try:

x=int(input("Enter number: "))

except ValueError:

print("Please enter numeric information.")

sys.exit(1)

if x==1:

y="Strongly dissatisfied"

elif x==2:

y="Dissatisfied"

elif x==3:

y='Neither dissatisfied nor satisfied'

elif x==4:

y='Satisfied'

elif x==5:

y='Very satisfied'

else:

y="Please use 1-5 range to rate your satisfaction"

return y

Q: Create a function that will print number of observations of each category of object type variables.

#arguments of function is data and column names

#df - data, i- column name

def observations\_printer(df,i):

df=df.select\_dtypes(include='object')

return df[i].value\_counts()

#applying function for one of object type variables

observations\_printer(affairs\_data,'affairs')

Q: Inside for loop use the function created above (6.j. assignment) to print the unique categories for the following variable: affairs, gender, children.

for i in ['affairs','gender','children']:

print(i)

print('\n')

print(observations\_printer(affairs\_data,i)) # observations\_printer(df,i) is a function created above. Affairs stands for data

# i stands for the column names in the ['affairs','gender','children'] list.

print('\n')

DATA TRANSFORMATION

Q: Remove all the missing values.

affairs\_data.dropna(inplace=True)

Q: Drop “ID” column.

affairs\_data.drop('ID',axis=1,inplace=True)

Q: Some variable names contain spaces (e.g. years married). Replace the space with underscore (“\_”). Use replace() function to do that. See more here.

affairs\_data.columns=affairs\_data.columns.str.replace(" ","\_")

#printing new column names

affairs\_data.columns

Q: Separate numeric and object variables. Save them as separate data under “data\_num” and “data\_obj” names for numeric and object variables respectively.

data\_num=affairs\_data.select\_dtypes(exclude='object') #numeric

data\_obj=affairs\_data.select\_dtypes(include='object') #object

Q: Create a bar chart where x-axis is affairs and y-axis is an average age. Your chart, X and Y axes should have titles. Adjust figure size to (10,6).

plt.figure(figsize=(10,6)) #adjusting figure size

plt.bar(affairs\_data.affairs.unique(), #putting unique categories of affairs on X-axis

affairs\_data.groupby('affairs').age.mean(), #grouping data by affairs, then calculation average age for Y-axis

color='g'

)

plt.show()

Q: By using for loop construct histogram for all numeric type variables in “data\_num” dataset created in Problem 7.e.

for i in data\_num.columns:

print(i)

plt.hist(data\_num[i])

plt.title(i)

plt.show()

CLASS 3

1. plt.figure(figsize=(8,8)) *#changing the size of figure*

**plt.subplot(2,1,1)** \* *#first plot when plotting to figures side by side, or under each other*

plt.scatter(data\_numeric.horsepow, *#X-axis values*

data\_numeric.price *#Y-axis values*

)

plt.title("Price VS Horsepower") *#title of the first chart*

plt.subplot(2,1,2) *#second plot*

plt.scatter(data\_numeric.horsepow, *#X-axis values*

data\_numeric.sales *#Y-axis values*

)

plt.title("Sales VS Horsepower") *#title of second chart*

plt.show()

* Part means: (x,y,z)

x=number of rows or columns

y=change to see the charts next to each other

z=the rank of the chart in the output

WITHOUT subplot python automatically puts all charts together in one.

1. Box plots are good for finding **outliers** in the data.
2. Important

*#calculating mean heights by position*

mean\_height=my\_data.Height.groupby(my\_data.Pos).mean() *#returns the mean value by each position*

posistion=my\_data.Pos.unique() *#returns array with names of positions*

plt.figure(figsize=(9,7),facecolor="#f9f5f2") *#making plot bigger*

*#draw plot*

plt.plot(posistion,

mean\_height,

color="#ff8019",

marker='o')

*#giving titles*

plt.title("WNBA Players Mean Height by Position")

plt.ylabel("Height (cm)")

plt.xlabel("Positions")

plt.show()

BOX PLOT

sns.boxplot(salary\_data.income,

salary\_data.age,

orient='v').set(title='Age vs Income')

plt.show()

Plot for object variables

sns.countplot(x = 'Geder', hue = 'income', data = salary\_data, palette = 'magma')

plt.title('Gender vs Income')

plt.show()

Simple scatter plot

plt.scatter(salary\_data.age,

salary\_data.hours\_per\_week

)

plt.xlabel('age')

plt.ylabel('hours\_per\_week')

plt.title("Age VS Hours per week")

plt.show()

**NOTEs**

round(pd.pivot\_table(data,index='furniture',values='area', aggfunc=['mean','count']),2)

def house\_type(data):

if data["furniture"]=="not furnished" and data["Animal\_Acceptance"]=="not acept":

return "not interested in offer"

else:

return "interested"

print(house\_type(data[data["House\_ID"] == 344].iloc[0]))

For loop interactive histogram

for i in data[["area","rooms","Floor\_Cleaned","Rent\_Amount\_Cleaned"]].columns:

fig=go.Figure(data=[go.Histogram(x=data[i],

marker\_color='#bfb2b9',

nbinsx=5)])

fig.update\_layout(

title=i,

height=500,

width=500)

fig.show()

**Q: import from yahoo**

from pandas\_datareader import data

FB\_df=data.DataReader(name="FB",

data\_source='yahoo',

start='2019-08-03', end='2020-08-03')

Q: average and median close prices?

print("Average: ", round(FB\_df.Close.mean(),2))

print("Median: ", round(FB\_df.Close.median(),2))

**One chart two lines**

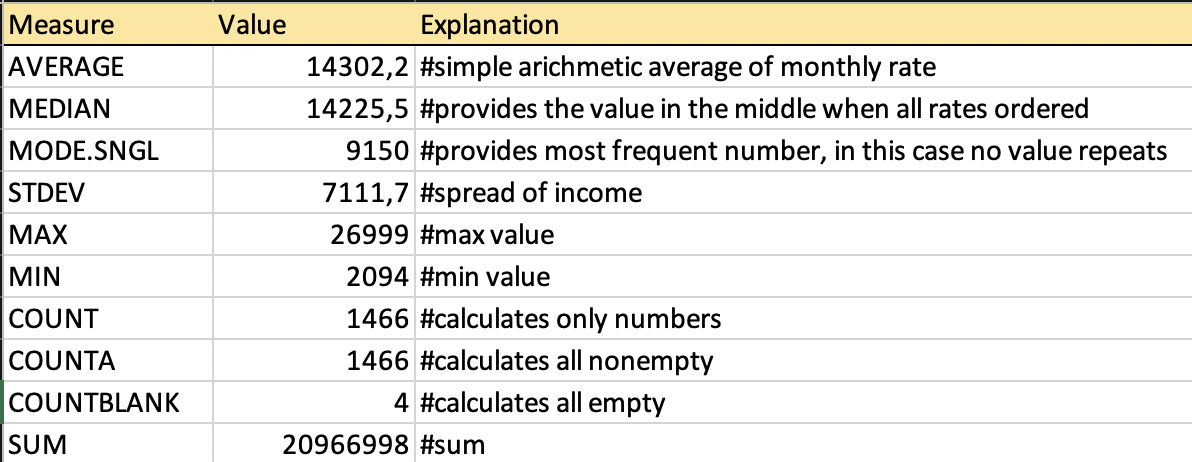
plt.figure(figsize=(9,8))

sns.lineplot(data=FB\_df[["Open","Close"]],

palette=['#1d63f0','#eb213c'])

plt.show()

**Excel Notes**

****

**Useful Functions**

**FIND is case sensitive, SEARCH is not**

=COUNTIF

=IFERROR(FIND("Science";D2);"")

=RIGHT([@BusinessTravel];LEN([@BusinessTravel])-SEARCH("\_";[@BusinessTravel]))

=CONCATENATE([@EducationField];" ";[@JobRole])

=[@EducationField]&" "&[@JobRole]

=AND([@MonthlyRate]>10000;[@MonthlyRate]<20000)

=IF(AND([@EnvironmentSatisfaction]>3;[@JobSatisfaction]>3);"satisfied";IF(OR([@EnvironmentSatisfaction]="";[@JobSatisfaction]="");"";"dissatisfied"))

Wine data

=VLOOKUP([@[Offer '#]],OfferInformation!$C$2:$G$33,5,FALSE)

VLOOKUP

* If we’re bringing data to “X” sheet, lookup value should be in “X” sheet
* Values should be unique in the sheet from where you’re bringing the data
* ID should be on the **left** of the column values that you’re trying to bring

=INDEX(OfferInformation!$A:$A,MATCH([@[Offer '#]],OfferInformation!$C:$C,0))

=INDEX('0) Raw Data 1'!$F$2:$F$1471;MATCH([@EmployeeID];'0) Raw Data 1'!$G$2:$G$1471))

- the last value here before exact match is the variable (in a different sheet, that has unique values)

* The first argument of INDEX is what we want to bring (e.g Campaign)
* The first argument of MATCH (lookup value) should be unique in the sheet from where we’re getting the data. (it can have duplicates in sheet where you’re righting the formula)

=VLOOKUP([@[Offer '#]],OfferInformation!$C$2:$F$33,4,FALSE)

IF + VLOOKUP

=IF(VLOOKUP([@EmployeeID];'0) Raw Data 2'!$A$2:$D$1471;4;0)="";"";VLOOKUP([@EmployeeID];'0) Raw Data 2'!$A$2:$D$1471;4;0))

COMPLEX INDEX AND MATCH

=IFNA(INDEX(Table1[Score\_Value];MATCH(1;(Table1[Student\_ID]=[@[Student\_ID]])\*(Table1[Score\_Type]<>"")\*(Table1[Score\_Type]="math score")\*(Table1[Score\_Value]<>"");0));"")

NOTEs

=VALUE(RIGHT([@[Rent\_Amount]];LEN([@[Rent\_Amount]])-2))

=IF(INDEX([floor];MATCH([@[House\_ID]];[House\_ID];0))="-";"";INDEX([floor];MATCH([@[House\_ID]];[House\_ID];0)))

=INDEX(Table2[animal];MATCH([@[House\_ID]];Table2[House\_ID];0))

**Python**

**Libraries needed for data analyzation in Python**

#Reading libs

import pandas as pd

from pandas\_datareader import data as dt #will be used to download data from yahoo finance

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import quandl

from plotly.offline import iplot

from plotly.offline import init\_notebook\_mode

import plotly.graph\_objects as go

from scipy.stats import pearsonr

from scipy.stats import ttest\_ind #Independent Sample T-test

from scipy.stats import f\_oneway #One-Way ANOVA

init\_notebook\_mode(connected=True)

**Creating a Pie Chart**

col=["#003f5c","#a05195"]

plt.figure(figsize=(12,8),facecolor="#f2f4f9")

plt.pie(data\_salary.Geder.value\_counts(),

labels=data\_salary.income.unique(),

colors=col)

plt.title("Relationship between Income and Gender")

plt.legend()

plt.show()

8c

plt.figure(figsize=(20,15))

pd.crosstab(data\_salary.income,data\_salary.race).plot(kind="bar",figsize=(15,6),color=['#FFC300','#581845'])

plt.title('Relationship between Income and Race')

plt.xticks(rotation = 0)

plt.legend(["Female", "Male"])

plt.show()