```
# -*- coding: utf-8 -*-
"""DataVisualizationI.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1SymaovxJ2Otg6U8T0nZ_yGgfjhSanAbR

** Data Visualization I**
1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and
contains information
about the passengers who boarded the unfortunate Titanic ship. Use the
Seaborn library to
see if we can find any patterns in the data.
2. Write a code to check how the price of the ticket (column name: 'fare')
for each passenger
is distributed by plotting a histogram
"""



# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency
# %matplotlib inline
# %config InlineBackend.figure_format = 'retina'
plt.style.use('seaborn-ticks')

SMALL_SIZE = 13
MEDIUM_SIZE = 14
BIGGER_SIZE = 16

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=MEDIUM_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=MEDIUM_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize

titanic_data = pd.read_csv('/content/titanic-data.csv')

titanic_data.head(5)

titanic_data.info()

"""#Drop the unwanted columns"""

n_titanic_data=titanic_data.drop(['Cabin','Ticket','Name',
                                  'Fare','PassengerId'],axis=1)

n_titanic_data.head()

n_titanic_data.info()
```

```
"""We have only 714 Age values out of 891 of the entries and 2 values missing
from the Embarked Variable. We will have to decide whether to omit these or
impute them with some values when we model relationships based on Age or
Embarked."""



"""**Further Exploration - Visualizations**

We will change the keys to make them better readable and explore the initial
composition of the passengers.

Make another copy of the new dataframe

Change the embarked keys to better readable ones

And the survived keys
"""

descript = n_titanic_data.copy()

descript.loc[:,'Embarked'].replace(['C','S','Q'],
['Cherbourg','Southampton','Queenstown'],
                                    inplace=True)

descript.loc[:,'Survived'].replace([0,1],['No','Yes'],inplace=True)

# Make a function to get the composition of the variables per number of
passengers
def Groupby_OneCol_comp_plot(df, col, plt_style = 'seaborn-ticks',
color_palette = "coolwarm"):
    '''
    Group by col1, sort by size , return and plot the dataframe with a bar
and pie plot
    '''
    gr=pd.DataFrame()
    gr['{} No'.format(col)] = df.groupby(col).size()
    gr['{} Ratio'.format(col)] = np.round(gr['{}
No'.format(col)].divide(gr['{} No'.format(col)].sum())*100,0)

    print ('Total No. of {}:{}'.format(col,gr['{} No'.format(col)].sum()))

    plt.style.use(plt_style)
    sns.set_palette(sns.color_palette(color_palette))

    fig=plt.figure()
    plt.axis('off')

    fig.add_subplot(121)

    ax=gr['{} No'.format(col)].plot(kind='bar', title='{}
Counts'.format(col), figsize=(16,8), color=sns.color_palette())
    _ = plt.setp(ax.get_xticklabels(), rotation=0)
    for p in ax.patches: ax.annotate(np.round(p.get_height(),decimals=2),
                                     (p.get_x()+p.get_width()/2.,
p.get_height()),
```

```
                                    ha='center', va='center', xytext=(0,
10), textcoords='offset points')
    ax.get_yaxis().set_ticks([])
    plt.xlabel('')

    fig.add_subplot(122)
    plt.axis('off')
    gr.loc[:,'{} Ratio'.format(col)].plot(kind= 'pie',
                                    autopct='%1.1f%%',shadow=False,
                                    title='{} Ratio'.format(col),
legend=False, labels=None);
    sns.despine(top=True, right=True, left=True, bottom=False);

"""#Analysis of the Embarked variable."""

Groupby_OneCol_comp_plot(descript, 'Embarked')

"""#Correlation of Survived with Embarked ."""

def plot(table, legloc='upper right',
                                   plt_style = 'seaborn-ticks',
                                   color_palette="dark",sorter=None,
stacked=False,
                                   kind = 'bar', percentage = True,
                                custom_title=None, minimal=True,
figsize=(19,10), width=0.7 ):
    grouped = table

    #Tranform to percentages
    if percentage == True:
        grouped = np.round(grouped.divide(grouped['Total'],axis=0)*100,0)
    try:
        del grouped['Total']
    except:
        pass

    # rearrange the columns
    if sorter:
        grouped = grouped[sorter]

    plt.style.use(plt_style)
    sns.set_palette(sns.color_palette(color_palette))
    ax = grouped.plot(kind=kind,stacked=stacked, figsize=figsize,
width=width)
    _ = plt.setp(ax.get_xticklabels(), rotation=0)  # Rotate labels
    plt.legend(loc=legloc) # plot the legend normally

    #annotate the bars
    if percentage == True:
      for p in ax.patches:
ax.annotate('{}%'.format(int(np.round(p.get_height(),decimals=2))),
                                          (p.get_x()+p.get_width()/2.,
                                           p.get_height()), ha='center',
va='center',
                                         xytext=(0, 10), textcoords='offset
points')
    else:
```

```
      for p in ax.patches:
            ax.annotate(np.round(p.get_height(),decimals=2),
                                          (p.get_x()+p.get_width()/2.,
                                           p.get_height()), ha='center',
va='center',
                                         xytext=(0, 10), textcoords='offset
points')
    if minimal == True:
        ax.get_yaxis().set_ticks([])
        plt.xlabel('')
        sns.despine(top=True, right=True, left=True, bottom=False);
    else:
        pass
    # set custom title
    plt.title(custom_title)

def Groupby_TwoCol_Plot(df, col1, col2, legloc='upper right',
                                    plt_style = 'ggplot',
                                    color_palette="dark",sorter=None,
stacked=False,
                                    kind = 'bar', percentage = True,
                                custom_title=None, minimal=True,
figsize=(14,6), width=0.6):

    #Group by Placement and Representative and unstack by Placement
    grouped = df.groupby([col2,col1]).size().unstack(col2)

    #Make a totals column sort and delete after
    grouped['Total'] = grouped.sum(axis=1)
    #grouped = grouped.sort_values('Total', ascending = False)

    plot(grouped, legloc=legloc,
                                    plt_style = plt_style,
color_palette=color_palette,sorter=sorter, stacked=stacked,
                                    kind = kind , percentage = percentage,
                                  custom_title=custom_title, minimal=minimal,
figsize=figsize, width=width)

Groupby_TwoCol_Plot(descript,'Embarked', 'Survived',
color_palette=('darkred','pink'),
                    plt_style = 'seaborn-ticks', custom_title='Proportion of
Survived per Embarkation Port')

"""#Correlation of Embarked with Pclass"""

#Calculate percentages of port passengers per Class
Groupby_TwoCol_Plot(descript,'Embarked', 'Pclass',
color_palette=('cubehelix'),
                    plt_style = 'seaborn-ticks', custom_title='Proportion of
Embarked per PcClass', sorter = [1,2,3])

"""**Data Visualization II**
1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box
plot for distribution
of age with respect to each gender along with the information about whether
they survived
or not. (Column names : 'sex' and 'age')
```

2. Write observations on the inference from the above statistics

```
#Correlation of Embarked with Sex.
"""

#Calculate percentages of port passengers per Sex
Groupby_TwoCol_Plot(descript,'Embarked', 'Sex',
color_palette=('blue','green'),
                    plt_style = 'seaborn-ticks', custom_title='Proportion of
Sex per PcClass',
                   legloc='upper left')

"""#Analysis of the Sex variable"""

Groupby_OneCol_comp_plot(descript, 'Sex', color_palette = ('yellow','green')
)

#Calculate percentages of Pclass per Sex
Groupby_TwoCol_Plot(descript,'Pclass', 'Sex',
color_palette=('yellow','purple'),
                    plt_style = 'seaborn-ticks', custom_title='Proportion of
Sex per PcClass',
                   legloc='upper left')

"""#Correlation of Sex with Survived"""

Groupby_TwoCol_Plot(descript,'Survived', 'Sex',
color_palette=('blue','purple'),
                    plt_style = 'seaborn-ticks', custom_title='Proportion of
Sex per Survived',
                   legloc='upper left')

"""#Analysis of the Age variable"""

#Make a dataframe for non missing 'Age'values
not_missing = n_titanic_data[(n_titanic_data['Age'].notnull())]

#And replace the survived keys
not_missing.loc[:,'Survived'].replace([0,1],['No','Yes'],inplace=True)

print ('No. of Passengers with not missing Age
Values:{}'.format(len(not_missing)))

ax=plt.figure()
plt.suptitle('Passenger Age Distribution')
ax.add_subplot(121)
sns.distplot(not_missing['Age'],bins=11)
ax.add_subplot(122)
sns.violinplot(not_missing['Age']);

# Get summary descriptive statistics
v= pd.DataFrame(not_missing['Age'].describe())

#Change the index labels and round the values reported
v.index = ['Population Size', 'Mean', 'Std. Deviation', 'Min', '25% Qt',
'Median',\
               '75% Qt', 'Max']
```

```
v = v.round(decimals=3)
v

"""We observe that the percentage of children below 10 that survived was
significantly higher and almost nobody over 70 year's old survived. We would
like to examine if this was by luck or by some other underlying reason (like
the 'Women and Children first' rule)."""

#Make a datframe with the sample populations
age = pd.DataFrame()
age['all'] = not_missing['Age']
not_survived = age['Not-survived'] =
not_missing['Age'][not_missing['Survived']=='No']
survived = age['Survived'] =
not_missing['Age'][not_missing['Survived']=='Yes']

#Get the summary statistics
var = age.describe()

#Change the index labels and round the values reported
var.index = ['Sample Size', 'Mean', 'Std. Deviation', 'Min', '25% Qt',
'Median',\
               '75% Qt', 'Max']
var = var.round(decimals=3)

var.loc[:,['Not-survived','Survived']]

"""#Survived- Age Statistical Chi-SquaredTest

We will test the following hypotheses:

H0 : The Null Hypothesis, that there is no relationship between the Survived
and Age variables (independent)  →Oi≠Ei


H1 : The Alternative Hypothesis, that there is a relationship between the
Survived and Age variables (dependent)  →Oi=Ei
"""

#Create age-groups
age_labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69',
              '70-80']
age_group_values = pd.cut(not_missing.Age, range(0,81,10),
                                    right=False, labels=age_labels)
not_missing.loc[:,'age-groups'] = age_group_values

#Set the value for the one 80-year old outside the bins
#chi-squared is notvalid for no of observations below 5
not_missing.loc[not_missing['Age']>=80, 'age-groups'] = '70-80'

#Make an observed-table for chi-squared test
obs_table = pd.crosstab([not_missing['Survived']],[not_missing['age-
groups']])

obs_table

#Compute Chi-square statistic
```

```
chi2, p, dof, expected = chi2_contingency(obs_table)

#report results
print('chi2:{}\ndof:{}\np:{}'.format(chi2,dof,p))

"""For a=.05 and 7 degrees of freedom, p is smaller than 0.05 and we
therefore reject the Null-Hypothesis and accept that Survived and Age are
dependent variables and that there is indeed a relationship between age and
survivabilit"""
```