week3_solution

July 13, 2020

1 Week 3 - assignment solution

1.0.1 Exercise 1

- Read the dataset `metabric_clinical_and_expression_data.csv` and store its summary statistics into a new variable called `metabric_summary`.
- Just like the `.read_csv()` method allows reading data from a file, `pandas` provides a `.to_csv()` method to write `DataFrames` to files. Write your summary statistics object into a file called `metabric_summary.csv`. You can use `help(metabric.to_csv)` to get information on how to use this function.
- Use the help information to modify the previous step so that you can generate a Tab Separated Value (TSV) file instead
- Similarly, explore the method `to_excel()` to produce an excel spreadsheet containing summary statistics

[ ]:
```
# Load library
import pandas as pd

# Read metabric dataset
metabric = pd.read_csv("../data/metabric_clinical_and_expression_data.csv")

# Store summary statistics
metabric_summary = metabric.describe()
metabric_summary
```

[ ]:
```
# Write summary statistics in csv and tsv

#help(metabric.to_csv)
metabric_summary.to_csv("~/Desktop/metabric_summary.csv")
metabric_summary.to_csv("~/Desktop/metabric_summary.tsv", sep = '\t')

#metabric_summary.to_csv("~/Desktop/metabric_summary.csv", columns = ["Cohort",␣
↪"Age_at_diagnosis"])
#metabric_summary.to_csv("~/Desktop/metabric_summary.csv", header = False)
#metabric_summary.to_csv("~/Desktop/metabric_summary.csv", index = False)
```

[ ]:
```
# Write an excel spreadsheet

#help(metabric.to_excel)
metabric_summary.to_excel("~/Desktop/metabric_summary.xlsx")

#If: ModuleNotFoundError: No module named 'openpyxl'
#pip3 install openpyxl OR conda install openpyxl
```

1.0.2 Exercise 2

- Read the dataset `metabric_clinical_and_expression_data.csv` into a variable e.g. `metabric`.
- Calculate the mean tumour size of patients grouped by vital status and tumour stage
- Find the cohort of patients and tumour stage where the average expression of genes TP53 and FOXA1 is highest
- Do patients with greater tumour size live longer? How about patients with greater tumour stage? How about greater Nottingham_prognostic_index?

[ ]:
```
# Calculate the mean tumour size of patients grouped by vital status and tumour␣
 ↪stage

#import pandas as pd
#metabric = pd.read_csv("../data/metabric_clinical_and_expression_data.csv")
#help(metabric.groupby)
#metabric.groupby(['Vital_status', 'Tumour_stage']).mean()
#metabric.groupby(['Vital_status', 'Tumour_stage']).mean()['Tumour_size']
#metabric.groupby(['Vital_status', 'Tumour_stage']).size()
metabric.groupby(['Vital_status', 'Tumour_stage']).agg(['mean',␣
 ↪'size'])['Tumour_size']
```

[ ]:
```
# Find the cohort of patients and tumour stage where the average expression of␣
 ↪genes TP53 and FOXA1 is highest

metabric['TP53_FOXA1_mean'] = metabric[['TP53', 'FOXA1']].mean(axis=1)
#metabric.groupby(['Cohort', 'Tumour_stage']).agg(['mean',␣
 ↪'size'])['TP53_FOXA1_mean']
metabric.groupby(['Cohort', 'Tumour_stage']).agg(['mean',␣
 ↪'size'])['TP53_FOXA1_mean'].sort_values('mean', ascending=False).head(1)
```

[ ]:
```
# Do patients with greater tumour size live longer?
metabric_dead = metabric[metabric['Vital_status'] == 'Died of Disease']

#metabric_dead[['Tumour_size', 'Survival_time']]
metabric_dead['Tumour_size'].corr(metabric_dead['Survival_time'])
#help(metabric_dead['Tumour_size'].corr)
```

[ ]:
```
# How about patients with greater tumour stage?

#metabric_dead[['Tumour_stage', 'Survival_time']]
metabric_dead[['Tumour_stage', 'Survival_time']].groupby('Tumour_stage').
 ↪agg(['mean', 'std', 'size'])

#metabric_dead['Tumour_stage'].corr(metabric_dead['Survival_time'])
```

[ ]:
```
# How about greater Nottingham_prognostic_index?
# https://en.wikipedia.org/wiki/Nottingham_Prognostic_Index

#metabric_dead[['Nottingham_prognostic_index', 'Survival_time']]
metabric_dead['Nottingham_prognostic_index'].
 ↪corr(metabric_dead['Survival_time'])
```

1.0.3 Exercise 3 (bonus)

Review the section on missing data presented in the lecture. Consulting the user's guide section dedicated to missing data if necessary use the functionality provided by pandas to answer the following questions:

- Which variables (columns) of the metabric dataset have missing data?
- Find the patients ids who have missing tumour size and/or missing mutation count data. Which cohorts do they belong to?
- For the patients identified to have missing tumour size data for each cohort, calculate the average tumour size of the patients with tumour size data available within the same cohort to fill in the missing data

[ ]:
```
# Which variables (columns) of the metabric dataset have missing data?
metabric.info()
```

[ ]:
```
# Find the patients ids who have missing tumour size and/or missing mutation␣
 ↪count data. Which cohorts do they belong to?
#metabric[metabric['Tumour_size'].isna()]
metabric[metabric['Tumour_size'].isna()]['Cohort'].unique()

#metabric[metabric['Mutation_count'].isna()]
metabric[metabric['Mutation_count'].isna()]['Cohort'].unique()

#metabric[(metabric['Tumour_size'].isna()) & (metabric['Mutation_count'].
 ↪isna())]
```

[ ]:
```
# For the patients identified to have missing tumour size data for each cohort,␣
 ↪calculate the average tumour size of the patients with tumour size data␣
 ↪available within the same cohort to fill in the missing data
```

```
# Cohort 1
metabric_c1 = metabric[metabric['Cohort'] == 1]
metabric_c1[metabric_c1['Tumour_size'].isna()]
mean_c1 = round(metabric_c1[metabric_c1['Tumour_size'].notna()]['Tumour_size'].
 ↪mean(),1)
metabric_c1 = metabric_c1.fillna(value={'Tumour_size': mean_c1})
metabric_c1[metabric_c1['Patient_ID'].isin(["MB-0259", "MB-0284", "MB-0522"])]

# Cohort 3
#metabric_c3 = metabric[metabric['Cohort'] == 3]
#metabric_c3[metabric_c3['Tumour_size'].isna()]
#mean_c3 = round(metabric_c3[metabric_c3['Tumour_size'].notna()]['Tumour_size'].
 ↪mean(),1)
#metabric_c3 = metabric_c3.fillna(value={'Tumour_size': mean_c3})

# Cohort 5
#metabric_c5 = metabric[metabric['Cohort'] == 5]
#metabric_c5[metabric_c5['Tumour_size'].isna()]
#mean_c5 = round(metabric_c5[metabric_c5['Tumour_size'].notna()]['Tumour_size'].
 ↪mean(),1)
#metabric_c5 = metabric_c5.fillna(value={'Tumour_size': mean_c5})


### Challenge ###
# Using what you have learnt in weeks 2 (functions) and 3 (pandas),
# Write a function that takes an input dataframe and two columns e.g. "Cohort"␣
 ↪and "Tumour_size"
# and returns a new dataframe with the NaNs filled using the logic above
```