コンピュータリテラシ発展 ～Pythonを学ぶ～

第10回：表計算とデータ分析をやってみよう

shimizu@info.shonan-it.ac.jp

今回の授業内容

今回の授業内容

- 前回の課題解説
- データの分析
- データの可視化
- 課題

前回の課題解説

前回の課題解説

- 前回の課題の解答例を示します
- 解答例について質問があればご連絡ください

解答例

https://colab.research.google.com/drive/14DnmmedLqBkYnQXj4QWiYCMertJYKhqy?usp=sharing

データの分析

データの準備

- これから行う分析のためにデータを準備します
- Moodleにある「**data_analysis.zip**」をダウンロードして解凍してください
- Google Driveの作業場所に「**data_analysis**」フォルダごとアップロードします
- 「**data_analysis**」の中身は次のとおりです
  - 「customer.csv」
  - 「item.csv」
  - 「transaction_1.csv」
  - 「transaction_2.csv」

データの読み込み

目標：商品の購買データと顧客データを組み合わせ，購買分析を行います

- pandasを使ってデータを読み込み，顧客ID上位5名の顧客情報を表示させます
- `read_csv()` 関数：CSVファイルをpandasにデータを読み込ませます
- 読み込んだデータはDataFrameオブジェクトとして扱うことができます
- DataFrameオブジェクトの `head()` メソッドでデータの先頭5行を表示できます

```
import pandas as pd  # pandasモジュールをインポート

# CSVファイルのパスを指定
path = '/content/drive/MyDrive/???/data_analysis/customer.csv'

# 指定されたCSVファイルを読み込み，データフレームに格納
customer = pd.read_csv(path)

# データフレームの最初の5行を表示
customer.head()
```

データの読み込み

- データセットの読み込み関数
- pandasでは各種データセットを読み込む関数が用意されています

データ形式	関数	解説
[illegible]きタを読み[illegible]む		データを読み込む

データの読み込み

- データの基本情報を確認します
- pandasには，データの基本情報を簡単に表示するオプションも提供されています
- DataFrameオブジェクトのオプションを以下の表にまとめます

オプション	解説
`shape`	行数と列数を表示する（例:15行6列の場合，(15,6)と出力）
`columns`	列名を表示する
`dtypes`	列名と，列のデータ型を表示

データの結合

- 以下の2つの結合方法について解説します
  - データを縦方向に結合する処理
  - 特定の列をキーにして横方向にデータを結合する処理

データの結合

データを縦方向に結合

- `concat()` 関数を使ってデータを縦に連結します
  - **transaction_1.csv**と**transaction_2.csv**は同じ種別のデータを扱っています
  - データ分析をする際に1つのデータとなっている方が扱いやすい
  - この2つのデータを縦に連結します

データの結合

- `concat()` 関数でデータを縦に連結する
  - `ignore_index` を `True` に指定することで，連結前のデータがもつインデックスラベルを無視し，新しくインデックスラベルを作成します
  - `ignore_index` を `True` に指定しない場合（省略），元データのインデックスラベルをそのまま使用します

```
path = '/content/drive/MyDrive/???/data_analysis/'

# 指定されたCSVファイルを読み込み、データフレームに格納
transaction_1 = pd.read_csv(path + 'transaction_1.csv')
transaction_2 = pd.read_csv(path + 'transaction_2.csv')

# 2つのデータフレームを結合し、新しいデータフレームに格納
# ignore_index=True は、結合後のデータフレームのインデックスを再設定することを示す
transaction = pd.concat([transaction_1, transaction_2], ignore_index=True)

transaction
```

データの結合

特定の列をキーにして横方向にデータを結合

- `merge()` 関数で特定のキーをもとにデータを横に結合する
  - transactionデータは顧客IDがある
  - customerデータは顧客IDに紐づく顧客情報（顧客名，性別，年齢）がある
  - 顧客IDをキーとして結合処理を行う

データの結合

- `merge()` 関数で特定のキーをもとにデータを横に結合します
  - 引数 `on` に特定キー（顧客ID）を指定します
  - `concat()` 関数とデータの指定方法は異なるので注意

```
# transactionデータフレームとcustomerデータフレームを '顧客ID' 列でマージ
# '顧客ID' 列の値が一致する行を結合して、新しいデータフレームを作成
sales_data = pd.merge(transaction, customer, on='顧客ID')

# マージされたデータフレームの内容を表示
sales_data
```

データの集計

- 合計を求める
  - データの集計作業を行う
  - 顧客IDごとの合計購買金額と合計購入数を求める
  - `merge` した

データの集計

- 合計を求める
  - データの集計作業を行う
  - 顧客IDごとの合計購買金額と合計購入数を求める

```
# sales_data データフレームを '顧客ID' 列でグループ化し、
# '購買金額' と '購入数' 列の合計を計算
sales_per_customer = sales_data.groupby('顧客ID')[['購買金額', '購入数']].sum()

# 顧客ごとの合計購買金額と合計購入数を含むデータフレームの内容を表示
sales_per_customer
```

データの集計

- 合計を求める - データの集計作業を行う
  - `groupby()` メソッドでグループ化された結果を `GroupBy` オブジェクトと呼ぶ
  - `GroupBy` オブジェクトの他のメソッドを以下にまとめる

メソッド名	解説
`count()`	グループ化されたデータの個数を表示
`mean()`	グループ化されたデータの平均値を表示
`sum()`	グループ化されたデータの総和を表示
`describe()`	グループ化されたデータの統計情報を表示（例：最大値,標準偏差）

データの集計

- 平均を求める
  - データの集計作業を行う
  - `mean()` メソッドを使って「購買日」ごとの平均売上を求める

```
# sales_data データフレームを '購買日' 列でグループ化し、
# '購買金額' 列の平均を計算
sales_per_day = sales_data.groupby('購買日').購買金額.mean()

# 日ごとの平均購買金額を含むデータフレームの内容を表示
sales_per_day
```

データの可視化

データを可視化するためのライブラリ

- 集計データを使ってグラフを作成します
- 使用するライブラリ
  - Matplotlib
    - Pythonでグラフを描くための基本的なライブラリ
    - グラフの基本的な設定を行う
  - seaborn
    - Matplotlibをベースにしたより複雑な可視化を簡単に行うことができるライブラリ

データを可視化するためのライブラリ

- 集計データを使ってグラフを作成します
- **Matplotlib**も**seaborn**もColabにはインストール済みです
- インポートする必要があります

```
# matplotlibモジュールのpyplotサブモジュールをインポート（グラフ描画のためのツール）
from matplotlib import pyplot as plt

# seabornモジュールをインポート（データの可視化を行うための高レベルなインターフェース）
import seaborn as sns
```

データを可視化するためのライブラリ

- 集計データを使ってグラフを作成します
- 日本語フォントの使用について
  - Matplotlibは日本語に対応していないので日本語を表示するためには日本語フォントを指定する必要があります
  - japanize_matplotlibライブラリをインストールしてインポートします

```
# pipコマンドを使って japanize_matplotlib パッケージをインストール
!pip install japanize_matplotlib

# japanize_matplotlib パッケージをインポート
import japanize_matplotlib
```

データを可視化するためのライブラリ

- 集計データを使ってグラフを作成する
- **sales_per_customer**データを棒グラフで表示します
- `barplot()` 関数を使用します
- 引数「**data**」に対象のDataFrameオブジェクトを指定する

```
# 顧客ごとの合計購買金額を棒グラフで表示
# x軸に顧客ID、y軸に購買金額を指定
ax = sns.barplot(x=sales_per_customer.index, y='購買金額', data=sales_per_customer)
```

データを可視化するためのライブラリ

- 集計データを使ってグラフを作成する
- **sales_per_day**データを折れ線グラフで表示する
- `lineplot()` 関数を使用する
- 引数「**data**」に対象のDataFrameオブジェクトを指定する

```
# 日ごとの平均購買金額を折れ線グラフで表示
# data引数にsales_per_dayデータフレームを指定
ax = sns.lineplot(data=sales_per_day)
```

データを可視化するためのライブラリ

- グラフのサイズを調整する
  - x軸のデータ表示が重なってしまい見づらいのでグラフを横に広くします
  - **Matplotlib**の設定で修正可能
  - `figure()` 関数の `figsize` 引数を指定する

```
# グラフのサイズを指定（幅18、高さ4）
plt.figure(figsize=(18, 4))

# 日ごとの平均購買金額を折れ線グラフで表示
# data引数にsales_per_dayデータフレームを指定
ax = sns.lineplot(data=sales_per_day)
```

データを可視化するためのライブラリ

- グラフの表示を設定
  - `set_title()` メソッド：グラフタイトルを指定
  - `set()` メソッド：引数 `xlabel`，`ylabel` を指定してx軸，y軸のラベルを作成

```
# グラフのサイズを指定（幅18、高さ4）
plt.figure(figsize=(18, 4))

# 日ごとの平均購買金額を折れ線グラフで表示
# data引数にsales_per_dayデータフレームを指定
ax = sns.lineplot(data=sales_per_day)

# グラフのタイトルを設定
ax.set_title('購買日ごとの平均売上')

# x軸とy軸のラベルを設定
ax.set(xlabel='購買日', ylabel='平均売上（円）')
```

課題

課題

- Moodleにある「SCfCL_10_prac.ipynb」ファイルをダウンロードしてColabにアップロードしてください
- 課題が完了したら「File」>「Download」>「Download .ipynb」で「.ipynb」形式でダウンロードしてください
- ダウンロードした **.ipynbファイル** をMoodleに提出してください

提出期限は 12月 5 日(木) 20時まで です