

第11回

データ処理・分析・可視化の総合実践 (Pandas)

出席認証コード: 7394

本日のロードマップ

 最終目標：インタラクティブなビザ発行数分析ダッシュボード

Part	内容	成果物
Part 1	Pandas基礎	データ読み込み・観察・基本フィルタ
Part 2	データ分析	集計・ランキング・複数条件
Part 3	Streamlit連携	インタラクティブUI
Part 4	データ可視化	美しいグラフ・ダッシュボード
Part 5	統合	完成版アプリ

完成版デモ

今日の90分で、これを作ります！

イントロダクション

今日完成させるアプリ

- 年選択 → 国別ランキング表示
- 複数国選択 → 時系列グラフ比較
- 様々なグラフタイプ → 棒グラフ、線グラフ、円グラフ
- 美しいUI → サイドバー、タブ機能

使用するデータ

日本政府ビザ発行統計データ

- 年別・国別のビザ発行数
- 実際の公開データを使用（実践的！）

Part 1: Pandas基礎

データをプログラムで扱えるようになるう

1-1. 環境準備とデータ読み込み

ライブラリのインポート

```
import pandas as pd
import streamlit as st
import plotly.express as px
```

CSVファイルの読み込み

```
# 'visa_number_in_japan.csv'をアップロードしておく
df = pd.read_csv('visa_number_in_japan.csv')
```

ポイント: `df` は「DataFrame」の略。表形式データを扱う基本形です。

1-2. データの観察

データの中身を覗いてみる

```
# 最初の5行を表示
df.head()

# データの基本情報（行数、列数、データ型）
df.info()

# データのサイズ（行数、列数）
df.shape

# 列名の確認
df.columns
```

🔍 **確認ポイント:** 何年分のデータ？どんな国が含まれている？

1-3. 基本的なフィルタリング

条件を指定してデータを絞り込む

```
# 中国のデータだけを取り出す
china_data = df[df['Country'] == 'China']
china_data.head()

# 2017年のデータだけを取り出す
year_2017 = df[df['Year'] == 2017]
year_2017.head()

# 発行数が10,000以上のデータ
large_count = df[df['Number of issued_numerical'] >= 10000]
```

💡 **ポイント:** `df[条件式]` で必要なデータだけを抽出できる

Part 2: データ分析

集計・ランキング・複雑な条件

2-1. 複数条件フィルタリング

AND条件（両方の条件を満たす）

```
# 2017年かつ中国のデータ  
china_2017 = df[(df['Year'] == 2017) & (df['Country'] == 'China')]
```

OR条件（いずれかの条件を満たす）

```
# 中国または韓国のデータ  
china_or_korea = df[(df['Country'] == 'China') | (df['Country'] == 'Korea')]
```

isin()メソッド（複数の値のいずれかに一致）

```
# 指定した国のデータをまとめて取得  
countries_list = ['China', 'Korea', 'United States']  
selected_data = df[df['Country'].isin(countries_list)]
```

2-2. データ集計 (groupby)

国別の合計発行数

```
country_totals = df.groupby('Country')['Number of issued_numerical'].sum()  
print(country_totals.head())
```

年別の合計発行数

```
yearly_totals = df.groupby('Year')['Number of issued_numerical'].sum()  
print(yearly_totals)
```

🔑 キーポイント: `groupby()` はExcelのピボットテーブルのような機能

2-3. 演習：自分でランキングを作ろう

チャレンジ問題

以下の3つのランキングを作ってみましょう！

問題1: 2015年の国別TOP5ランキング

```
# ヒント：年でフィルタ → 国別集計 → 上位5位  
ranking_2015 = # ここを書いてみよう！
```

問題2: 各国の平均発行数ランキング

```
# ヒント：sum()ではなくmean()を使う  
country_average = # ここを書いてみよう！
```

問題3: ビザ発行数が最も多い年を持つ国TOP3

```
# ヒント：max()を使って各国の最大発行数を求める
```

解答例

```
# 問題1の解答
ranking_2015 = df[df['Year'] == 2015].groupby('Country')['Number of issued_numerical'].sum().sort_values(ascending=False).head(5)

# 問題2の解答
country_average = df.groupby('Country')['Number of issued_numerical'].mean().sort_values(ascending=False)

# 問題3の解答
country_max = df.groupby('Country')['Number of issued_numerical'].max().sort_values(ascending=False)
```

 **発見:** データを様々な角度から集計すると、新しい洞察が得られます！

Part 3: Streamlit連携

インタラクティブなWebアプリを作ろう

3-1. 基本的なUI要素

セレクトボックス（ドロップダウン）

```
countries = df['Country'].unique() # 国のリストを取得
selected_country = st.selectbox('国を選択', countries)


# 選択された国のデータを表示
country_data = df[df['Country'] == selected_country]
st.dataframe(country_data)
```

スライダー（数値選択）

```
selected_year = st.slider('年を選択',
                           min_value=int(df['Year'].min()),
                           max_value=int(df['Year'].max()),
                           value=2017)
```

マルチセレクト（複数選択）

```
selected_countries = st.multiselect(  
    '複数国を選択',  
    countries,  
    default=['China', 'Korea'] # デフォルト選択  
)  
  
# 選択された国々のデータを表示  
multi_data = df[df['Country'].isin(selected_countries)]  
st.dataframe(multi_data)
```

 インタラクティブさが鍵: ユーザーが操作できるアプリに！

3-2. インタラクティブなデータ表示

年選択 → その年のランキング表示

```
# スライダーで年を選択
selected_year = st.slider('年を選択', 2006, 2017, 2017)

# その年のデータを抽出
year_data = df[df['Year'] == selected_year]

# 国別集計とランキング
ranking = year_data.groupby('Country')['Number of issued_numerical'].sum().sort_values(ascending=False).head(10)

# 結果を表示
st.subheader(f'{selected_year}年 国別TOP10')
st.dataframe(ranking)
```

3-3. パフォーマンス最適化

st.cache_dataでデータ読み込みを高速化

```
@st.cache_data
def load_data():
    """データを読み込み、キャッシュに保存"""
    return pd.read_csv('visa_number_in_japan.csv')

# キャッシュされたデータを使用
df = load_data()
```

⚡ 効果: 2回目以降のアクセスが高速になる！

セッション状態の活用

```
# ページリロード時にも選択状態を保持
if 'selected_countries' not in st.session_state:
    st.session_state.selected_countries = ['China', 'Korea']
```

Part 4: データ可視化

データを美しく、分かりやすく表現

4-1. Streamlitの基本グラフ

年別推移の折れ線グラフ

```
yearly_totals = df.groupby('Year')['Number of issued_numerical'].sum()  
st.line_chart(yearly_totals)
```

国別ランキングの棒グラフ

```
top10_2023 = df[df['Year'] == 2023].groupby('Country')['Number of issued_numerical'].sum().sort_values(ascending=False).head(10)  
st.bar_chart(top10_2023)
```

👍 **メリット:** シンプルで高速、コードが短い

👎 **デメリット:** カスタマイズが限定的

4-2. Plotly Expressによる高度なグラフ

インタラクティブな折れ線グラフ

```
import plotly.express as px

# データの準備
line_data = df.groupby(['Year', 'Country'])['Number of issued_numerical'].sum().reset_index()

# 上位5カ国だけをグラフ化
top5_countries = df.groupby('Country')['Number of issued_numerical'].sum().sort_values(ascending=False).head(5).index
line_data_top5 = line_data[line_data['Country'].isin(top5_countries)]

# インタラクティブな折れ線グラフ
fig_line = px.line(line_data_top5,
                   x='Year',
                   y='Number of issued_numerical',
                   color='Country',
                   title='上位5カ国の年次推移')
st.plotly_chart(fig_line)
```

美しい棒グラフ

```
# 2017年TOP10データの準備
top10_data = df[df['Year'] == 2017].groupby('Country')['Number of issued_numerical'].sum().sort_values(ascending=False).head(10).reset_index()

# カスタマイズした棒グラフ
fig_bar = px.bar(top10_data,
                  x='Country',
                  y='Number of issued_numerical',
                  title='2017年 国別ビザ発行数 TOP10',
                  color='Number of issued_numerical',
                  color_continuous_scale='Blues')

# 軸ラベルの回転
fig_bar.update_layout(xaxis_tickangle=-45)
st.plotly_chart(fig_bar)
```

円グラフで構成比を表示

```
# 円グラフ用データ
pie_data = df[df['Year'] == 2017].groupby('Country')['Number of issued_numerical'].sum().head(10).reset_index()

fig_pie = px.pie(pie_data,
                 values='Number of issued_numerical',
                 names='Country',
                 title='2017年 国別構成比 (TOP10)')
st.plotly_chart(fig_pie)
```

✨ **Plotlyの強み:** ズーム、パン、ホバー情報など、インタラクティブ機能が豊富

4-3. 複数国の時系列比較

ユーザー選択による動的グラフ

```
# マルチセレクトで国を選択
selected_countries = st.multiselect('比較したい国を選択',
                                     countries,
                                     default=['China', 'Korea', 'United States'])

if selected_countries: # 選択された国がある場合のみ
    # 選択された国のデータを抽出
    filtered_data = df[df['Country'].isin(selected_countries)]
    comparison_data = filtered_data.groupby(['Year', 'Country'])['Number of issued_numerical'].sum().reset_index()

    # 比較グラフを作成
    fig_comparison = px.line(comparison_data,
                             x='Year',
                             y='Number of issued_numerical',
                             color='Country',
                             title='選択国の年次推移比較')

    st.plotly_chart(fig_comparison)
```


Part 5: 総合ダッシュボード作成

 演習：完成版ダッシュボードを作ろう！




演習の目標

以下の機能を持つインタラクティブなダッシュボードを作成してください！

完成イメージ

- サイドバーで年・国・グラフタイプを選択
- タブ機能で「ランキング」「時系列比較」「構成比」を切り替え
- 動的なグラフがユーザーの選択に応じて更新される

必要な機能一覧

1.  サイドバーによる操作パネル
2.  タブによる機能分け
3.  選択に応じた動的グラフ表示

実装手順

ステップ1: サイドバーによる操作パネルを作成

```
# サイドバーに設定を配置
st.sidebar.header('⚙️ 設定パネル')
selected_year = st.sidebar.slider('年を選択', 2006, 2017, 2017)
selected_countries = st.sidebar.multiselect('国を選択', countries, default=['China', 'Korea'])
graph_type = st.sidebar.selectbox('グラフタイプ', ['折れ線', '棒グラフ', '円グラフ'])
```

ステップ2: タブによる機能分けを実装

```
# メインエリアをタブで分割
tab1, tab2, tab3 = st.tabs(['📊 ランキング', '📈 時系列比較', '🍰 構成比'])

with tab1:
    st.subheader(f'{selected_year}年 国別ランキング')
    # ここにランキング表示のコードを書いてください

with tab2:
    st.subheader('複数国の推移比較')
    # ここに時系列比較のコードを書いてください

with tab3:
    st.subheader('国別構成比')
    # ここに円グラフのコードを書いてください
```

実装のヒント

タブ1「ランキング」に実装すること

- サイドバーで選択した年のデータでランキング作成
- `graph_type` の選択に応じてグラフを切り替え
- ランキングテーブルも表示

タブ2「時系列比較」に実装すること

- サイドバーで選択した国々の年次推移を比較
- Plotlyの折れ線グラフを使用
- 国が選択されていない場合の対応

タブ3「構成比」に実装すること

- サイドバーで選択した年の構成比を円グラフで表示
- TOP10のデータを使用

チャレンジ目標

以下の全ての要件を満たすダッシュボードを完成させよう！

- ✓ サイドバーで年・国・グラフタイプが選択できる
- ✓ タブで機能が分かれている
- ✓ 選択に応じてグラフが動的に変更される
- ✓ エラーが発生しない（例：国が未選択の場合）
- ✓ 美しく見やすいレイアウト



あなたも今日、データサイエンティストの第一歩を踏み出しました

まとめと振り返り

今日学んだこと

1. **Pandas基礎**: データの読み込み・観察・フィルタリング
2. **データ分析**: 集計・ランキング・複数条件処理
3. **Streamlit連携**: インタラクティブなUI作成
4. **データ可視化**: 基本グラフ + Plotly Express
5. **統合技術**: 全機能を組み合わせたダッシュボード

作成した成果物

「インタラクティブなビザ発行数分析ダッシュボード」

- 実用的なデータ分析ツール
- 美しく分かりやすい可視化
- ユーザーフレンドリーなインターフェース

次回以降への活用

- 他のデータセットでも同じ手法が使える
- より高度な分析への拡張が可能
- データサイエンスの基礎スキルとして活用

継続学習のヒント

- 興味のあるデータを見つけて同じように分析してみる
- Plotlyの他のグラフタイプを試してみる
- 機械学習ライブラリ（scikit-learn）への発展

お疲れさまでした！

データの力で世界を理解する楽しさを感じていただけましたか？

次回もお楽しみに！