第 11 回大阪 SAS 勉強会

「 SAS データを解析する」アプリを streamlit で作ってみる

武田薬品工業株式会社
舟尾 暢男

資料中のプログラム：https://nfunao.web.fc2.com/files/python_intro2_06.zip

メニュー

- インストール
- アプリの作成 → 実行 → 終了
- 「 SAS データを解析するアプリ」を streamlit で作成
- 各種出力と widget
- session_state
- レイアウト
- その他

python と streamlit のインストール

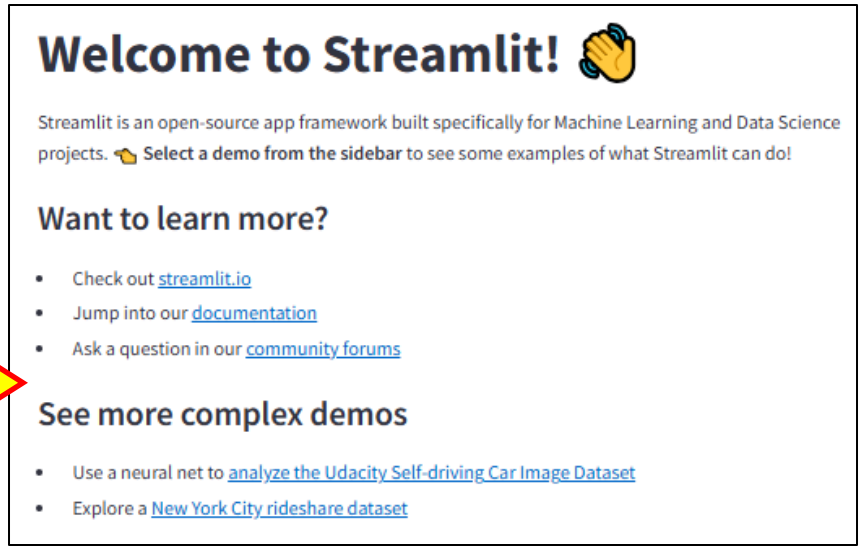
- python と Anaconda をインストール
  https://www.python.org/
  https://www.anaconda.com/download

- Anaconda Prompt 等で、モジュールをいくつかインストール

```
pip install streamlit
pip install pandas
pip install matplotlib
```

- インストールできたか確認

```
streamlit hello
```



- PC SAS は既に入っている前提（SAS v9.4）

メニュー

- インストール
- **アプリの作成 → 実行 → 終了**
- 「 SAS データを解析するアプリ」を streamlit で作成
- 各種出力と widget
- session_state
- レイアウト
- その他

1. PC 上でのアプリの作成 → 実行 → 終了

- アプリを作成 → C:¥temp に **app.py** というテキストファイルにて保存

```
import streamlit as st

st.title('Sample App.')
```

- Anaconda Prompt 等で、アプリのファイル（app.py）の階層まで移動後、実行
  - 【コマンドプロンプト】目的別コマンド一覧：https://techmania.jp/blog/cmd0002/

```
cd c:\temp
streamlit run app.py
```

streamlit 用のコマンド

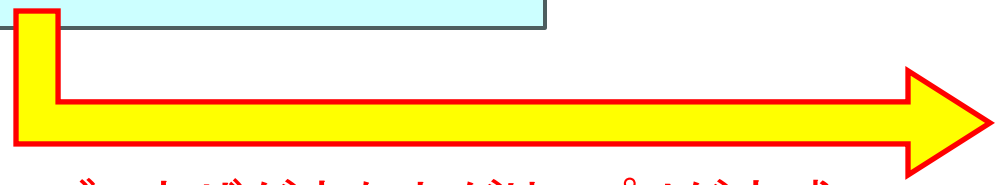
```
streamlit --help
streamlit --version
streamlit hello
streamlit docs
streamlit config show
streamlit cache clear
```

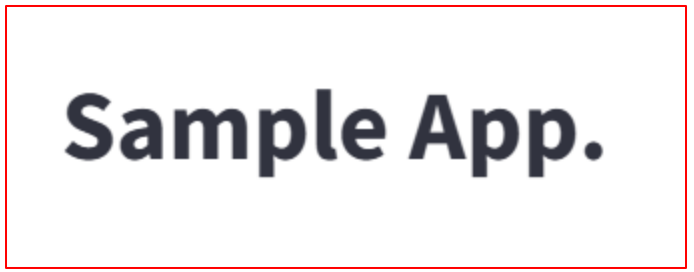
1. PC 上でのアプリの作成 → 実行 → 終了

- C:¥temp¥app.py

```
import streamlit as st

st.title('Sample App.')
```





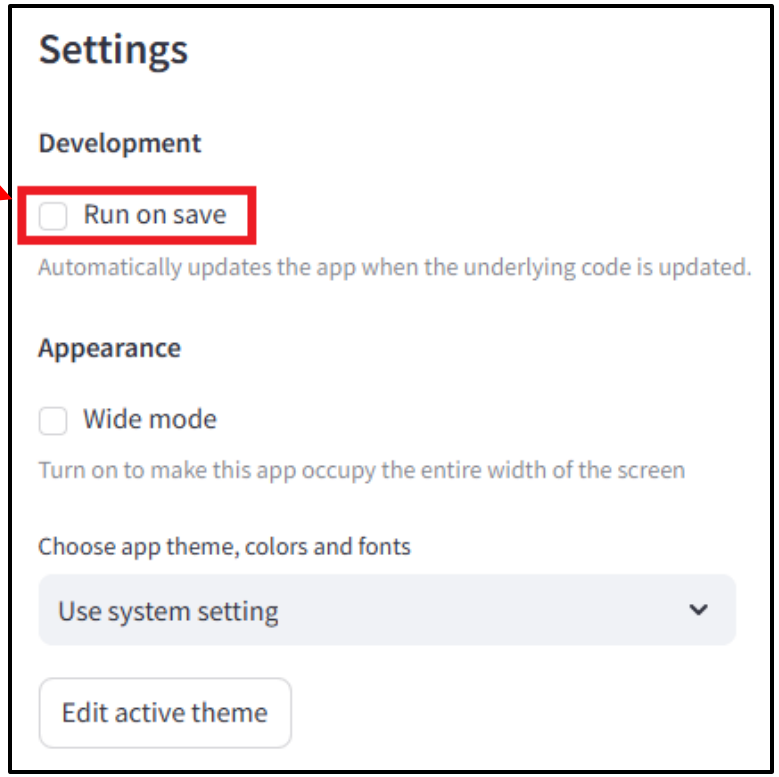
たった 2 行の命令でブラウザが立ち上がりアプリが完成！

【言い忘れ】

- 2025 年 1 月のナニワデータサイエンス研究会にて R shiny アプリの講演後、森岡さんより「 SAS で、この様なアプリを作成する方法はないのでしょうか？」とご質問いただく
  1. R shiny
  2. python shiny
  3. **python streamlit** ⇐
- 回答：python の streamlit を使うことで、簡単なアプリを作成することが出来るので、SAS データを解析するアプリを streamlit で作ってみました。

1. PC 上でのアプリの作成 → 実行 → 終了

- アプリの右上の Settings より [Run on save] をチェック
  - プログラムが修正・保存されると自動で更新される
  - 自動更新しない場合、コマンド上でアプリを【終了⇒起動】しなければいけなくなり、非常に面倒



- 終了する場合は、Anaconda Prompt 上で [Ctrl] + [C]

- アプリ作成時は以下を参照しつつ作業すると楽？
  - Streamlit Magic Cheat Sheets

    https://cheat-sheets.streamlit.app/
  - Streamlit documentation

    https://docs.streamlit.io/

メニュー

- インストール
- アプリの作成 → 実行 → 終了
- **「 SAS データを解析するアプリ」を streamlit で作成**
- 各種出力と widget
- session_state
- レイアウト
- その他

2. SAS プログラムを実行

- C:¥temp¥app.py → 黄色マーカーがバッチ実行部分

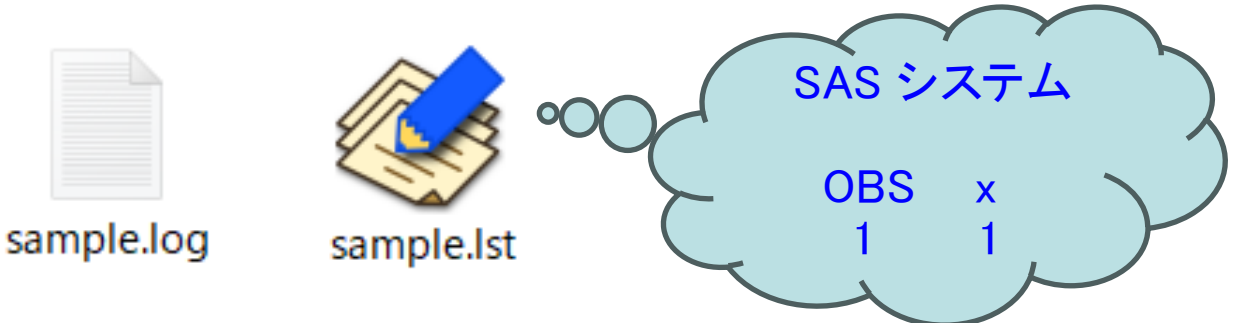
```
import streamlit as st
import subprocess

subprocess.Popen(r'"C:\Program Files\SAS_9.4\SASFoundation\9.4\sas.exe" -sysin
"C:\temp\sample.sas" -log "C:\temp" -print "C:\temp" -nosplash -icon',
shell=True)
```

- C:¥temp¥sample.sas

```
data x      ; x=1 ;
proc print ; run ;
```

- Anaconda Prompt 等でアプリ(app.py)を起動 → 結果とログファイルが出力



3. SAS データの読み込み＆解析

- C:¥temp¥app.py → 黄色マーカーが SAS ファイルの読み込み部分

```
import streamlit as st
import pandas    as pd
import matplotlib.pyplot as plt

st.title("Upload a SAS and Analyze Data")
uploaded_file = st.file_uploader("Choose a SAS file", type="sas7bdat")

if uploaded_file is not None:
  df = pd.read_sas(uploaded_file, format='sas7bdat', encoding='cp932')
  st.write("DataFrame Preview:")
  st.write(df.head(7))
  vars  = [c for c in df.columns]
  value = st.selectbox("Variable", vars)

  summary_stats = df[value].describe()
  st.subheader("Summary Statistics")
  st.write(summary_stats)

  st.subheader("Histogram")
  fig, ax = plt.subplots(1, 1, figsize=(10, 5))
  ax.hist(df[value], bins=20, alpha=0.7, color='blue', edgecolor='black')
  plt.tight_layout()
  st.pyplot(fig)
```

3. SAS データの読み込み＆解析

Upload a SAS and Analyze Data

Choose a SAS file

Drag and drop file here
Limit 200MB per file • SAS7BDAT

Browse files

tg.sas7bdat 128.0KB

DataFrame Preview:

	len	supp	dose
0	4.2	VC	0.5
1	11.5	VC	0.5
2	7.3	VC	0.5
3	5.8	VC	0.5
4	6.4	VC	0.5
5	10	VC	0.5
6	11.2	VC	0.5

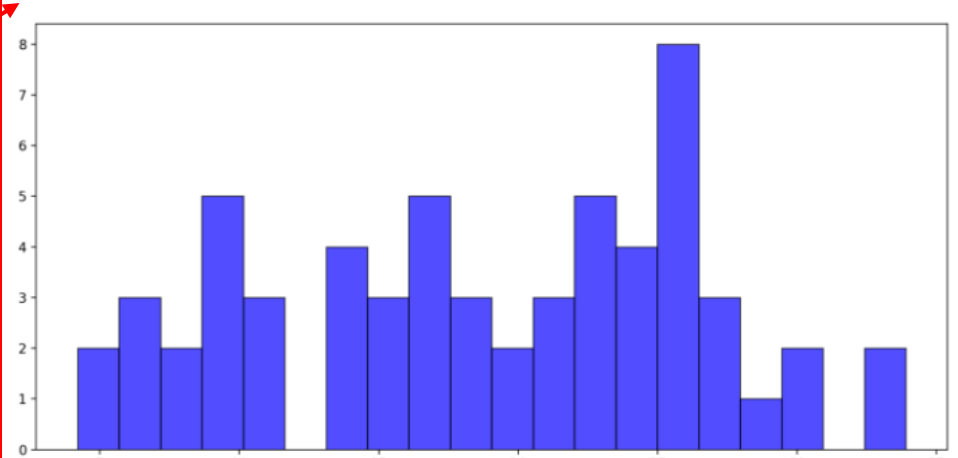
Variable

len

Summary Statistics

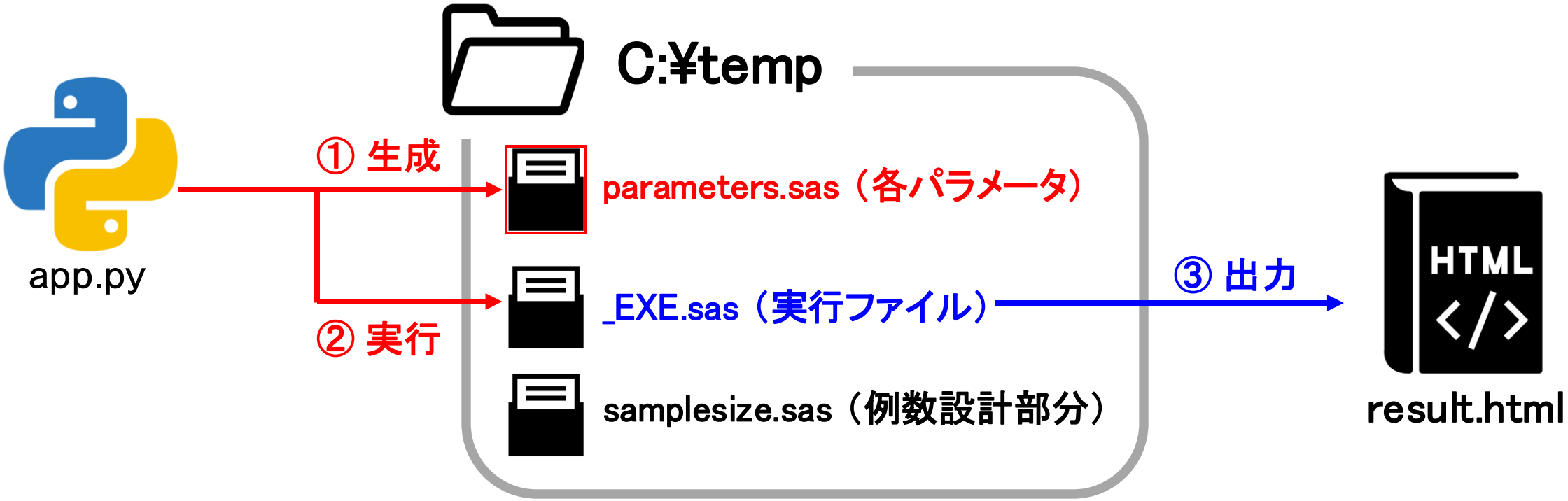
	len
count	60
mean	18.8133
std	7.6493
min	4.2
25%	13.075
50%	19.25
75%	25.275
max	33.9

Histogram



データ表示や解析は python にて実施

4. 例数設計アプリ



- C:¥temp に **_EXE.sas**（実行ファイル）と **samplesize.sas**（例数設計部分の SAS ）を保存
（ python から自動作成したり、サブフォルダに入れておいても良いが簡単のため）
- app.py から **parameters.sas**（例数設計の各パラメータを設定したマクロ変数定義部分）を作成後、**_EXE.sas** を実行する

4. 例数設計アプリ

- C:¥temp¥ _EXE.sas

```
options source source2 symbolgen ;

%let _path = C:/temp ;

%inc "./parameters.sas" ;

%inc "./samplesize.sas" ;
```

- C:¥temp¥samplesize.sas

```
ods html file='result.html'
path="&_path" style=styles.sasweb ;
proc power;
  twosamplemeans
  test      = diff
  meandiff  = &_meandiff.
  stddev    = &_stddev.
  alpha     = &_alpha.
  power     = &_power.
  ntotal    = . ;
run ;
ods html close;

x "&_path\result.html" ;
```

- C:¥temp に _EXE.sas（実行ファイル）と **samplesize.sas**（例数設計部分の SAS ）を保存（ python から自動作成したり、サブフォルダに入れておいても良いが簡単のため）
- app.py から **parameters.sas**（例数設計の各パラメータを設定したマクロ変数定義部分）を作成後、_EXE.sas を実行する

4. 例数設計アプリ

- C:¥temp¥app.py → まず parameters.sas を作成後、_EXE.sas を実行し例数設計

```
import streamlit as st
import subprocess

st.title("Sample Size - 2-sample t-test")
meandiff = st.number_input('Meandiff:', 1)
stddev   = st.number_input('Stddev:',   2)
alpha    = st.number_input('Alpha:', 0.05)
power    = st.number_input('Power:',  0.8)

if st.button("Save Parameters"):
  with open("C:/temp/parameters.sas", "w") as f:
   f.write("%let _meandiff = " + str(meandiff) +" ; ")
   f.write("%let _stddev   = " + str(stddev  ) +" ; ")
   f.write("%let _alpha    = " + str(alpha   ) +" ; ")
   f.write("%let _power    = " + str(power   ) +" ; ")
   st.success(f"Parameters has been saved.")

if st.button("Calculate"):
  subprocess.Popen(r'"C:\Program Files\SAS_9.4\SASFoundation\9.4\sas.exe" -
sysin "C:\temp\_EXE.sas" -nosplash -icon', shell=True)
```

4. 例数設計アプリ

Sample Size - 2-sample t-test

Meandiff:

1

Stddev:

2

Alpha:

0.05

Power:

0.80

Save Parameters

Parameters has been saved.

Calculate

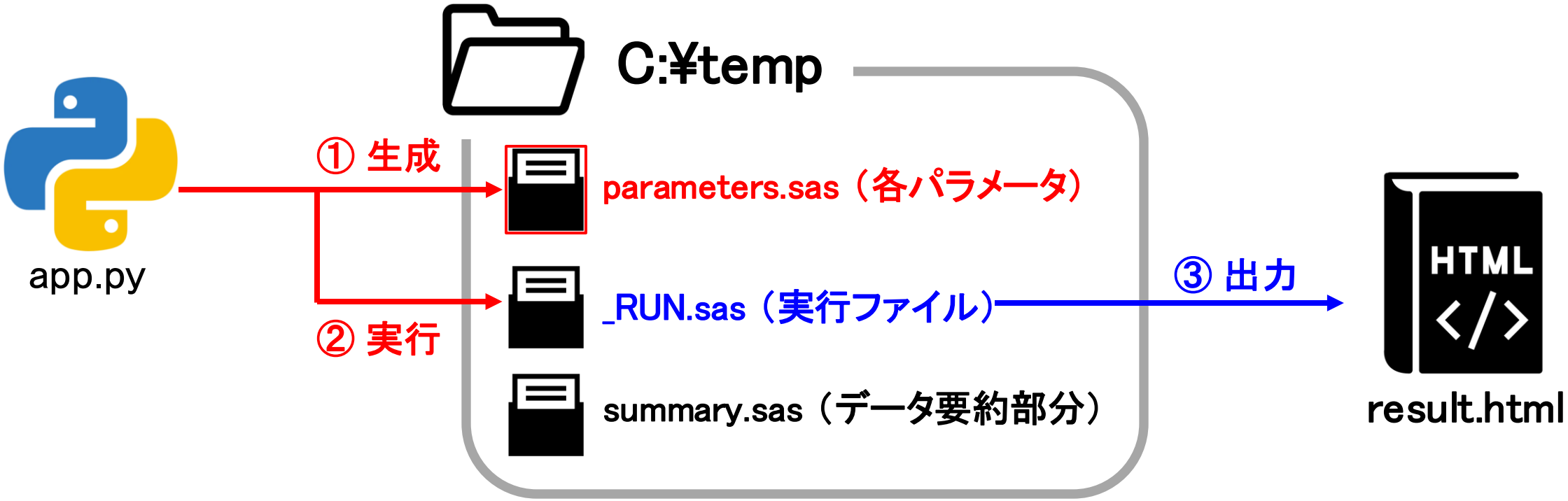
SAS システム

POWER プロシジャ
平均差に対する 2 標本 t 検定

固定シナリオ要素	
分布	正規
方法	正確
有意水準	0.05
平均差	1
標準偏差	2
名目の検出力	0.8
裾の数	2
帰無仮説の差	0
群 1 の重み	1
群 2 の重み	1

計算された N Total	
Actual Power	N Total
0.801	128

5. SAS データを解析するアプリ



- C:¥temp に **_RUN.sas**（実行ファイル）と **summary.sas**（データを要約する部分の SAS ）を保存（ python から自動作成したり、サブフォルダに入れておいても良いが簡単のため）
- app.py から**データを読み込んだ後**、**parameters.sas**（データ名と変数名を設定したマクロ変数定義部分）を作成、最後に **_RUN.sas** を実行する

5. SAS データを解析するアプリ

- C:¥temp¥_RUN.sas

```
options source source2 symbolgen ;

%let _path = C:/temp ;

%inc "./parameters.sas" ;

%inc "./summary.sas" ;
```

- C:¥temp¥summary.sas

```
libname IN "&_path" ;

ods html file='result.html'
path="&_path" style=styles.sasweb ;

proc means data=IN.&_data. n mean
stddev min q1 median q3 max ;
  var &_var. ;
run ;
proc sgplot data=IN.&_data. ;
  histogram &_var. / scale=count ;
  density   &_var. / type=normal ;
run;

ods html close;

x "&_path\result.html" ;
```

- C:¥temp に _RUN.sas（実行ファイル）と **summary.sas**（データを要約する部分の SAS ）を保存（ python から自動作成したり、サブフォルダに入れておいても良いが簡単のため）
- app.py から**データを読み込んだ後**、**parameters.sas**（データ名と変数名を設定したマクロ変数定義部分）を作成、最後に _RUN.sas を実行する

5. SAS データを解析するアプリ

- C:¥temp¥app.py → まず parameters.sas を作成後、_RUN.sas を実行しデータ解析

```
import streamlit as st
import pandas    as pd
import subprocess

st.title("Upload a SAS and Analyze Data")
uploaded_file = st.file_uploader("Choose a SAS file", type="sas7bdat")

if uploaded_file is not None:
  df = pd.read_sas(uploaded_file, format='sas7bdat', encoding='cp932')
  filename = uploaded_file.name.replace('.sas7bdat', '')
  st.write("DataFrame Preview:")
  st.write(df.head(7))
  vars = [c for c in df.columns]
  var  = st.selectbox("Variable", vars)
  with open("C:/temp/parameters.sas", "w") as f:
    f.write("%let _data = " + filename + " ; ")
    f.write("%let _var  = " + var      + " ; ")
    st.success(f"Parameters has been saved.")

if st.button("Run"):
  subprocess.Popen(r'"C:\Program Files\SAS_9.4\SASFoundation\9.4\sas.exe" -
sysin "C:\temp\_RUN.sas" -nosplash -icon', shell=True)
```

5. SAS データを解析するアプリ

Upload a SAS and Analyze Data

Choose a SAS file

Drag and drop file here
Limit 200MB per file • SAS7BDAT

Browse files

tg.sas7bdat 128.0KB

DataFrame Preview:

	len	supp	dose
0	4.2	VC	0.5
1	11.5	VC	0.5
2	7.3	VC	0.5
3	5.8	VC	0.5
4	6.4	VC	0.5
5	10	VC	0.5
6	11.2	VC	0.5

Variable

len

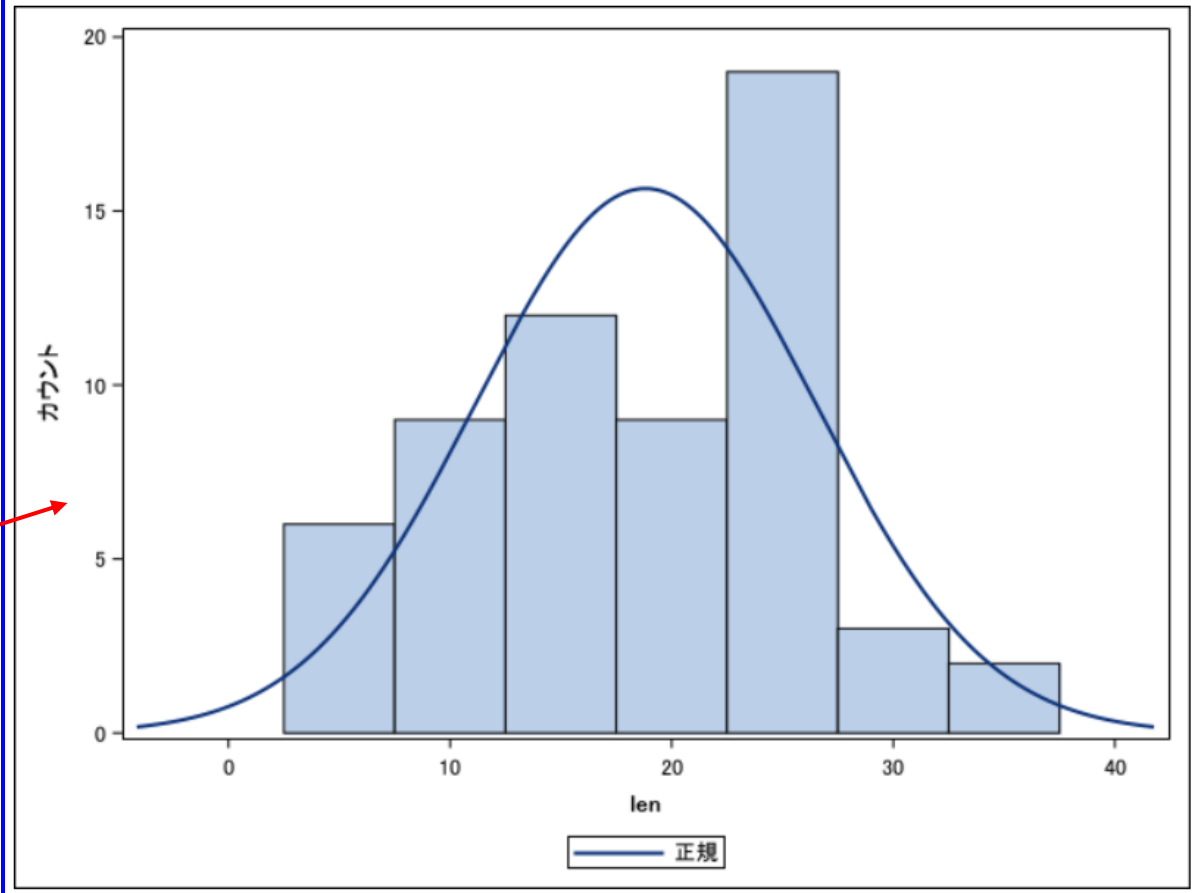
Parameters has been saved.

Run

SAS システム

MEANS プロシジャ

分析変数：len len							
N	平均	標準偏差	最小値	下側四分位点	中央値	上側四分位点	最大値
60	18.8133333	7.6493152	4.2000000	12.5500000	19.2500000	25.3500000	33.9000000



おまけ：アプリをデスクトップアプリ化

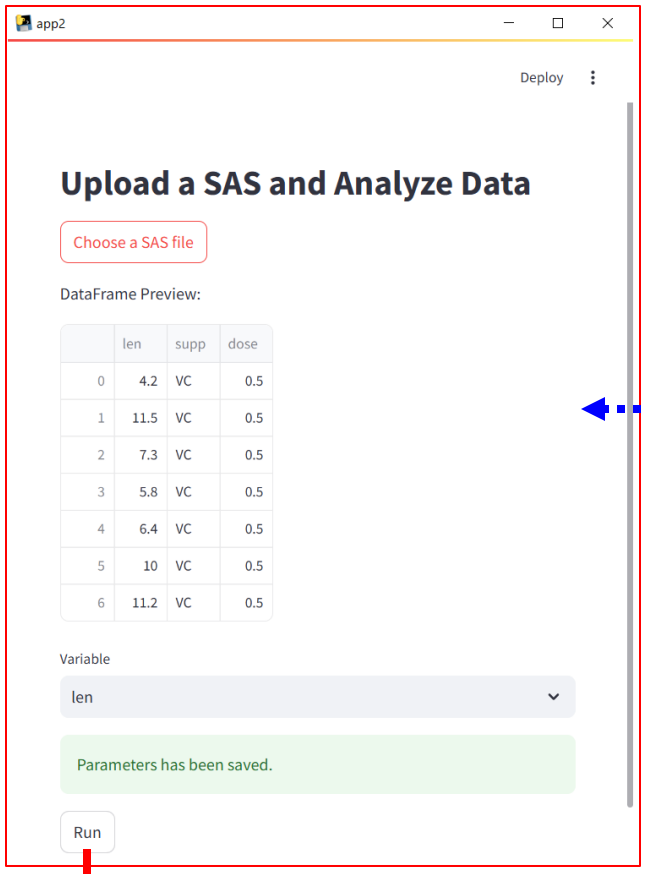
- **Streamlit アプリを簡単にデスクトップアプリ化するコマンドを作った** ⇦
  https://zenn.dev/ohtaman/articles/streamlit-desktop-app

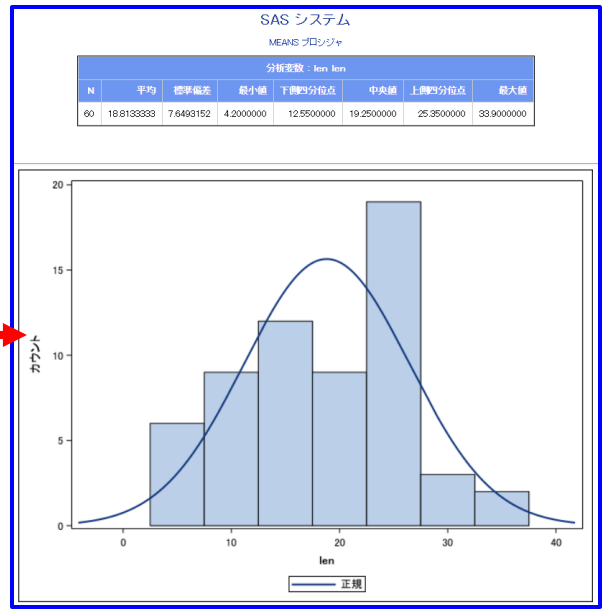
  - Anaconda Prompt 等で、モジュールをいくつかインストール

```
pip install streamlit-desktop-app
pip install easygui
```

- その他の方法：Streamlit の EXE 化
  https://qiita.com/Gyutan/items/c158920f0693fb099e92

おまけ：アプリをデスクトップアプリ化





app2.exe

_internal

app.py

_RUN.sas

summary.sas

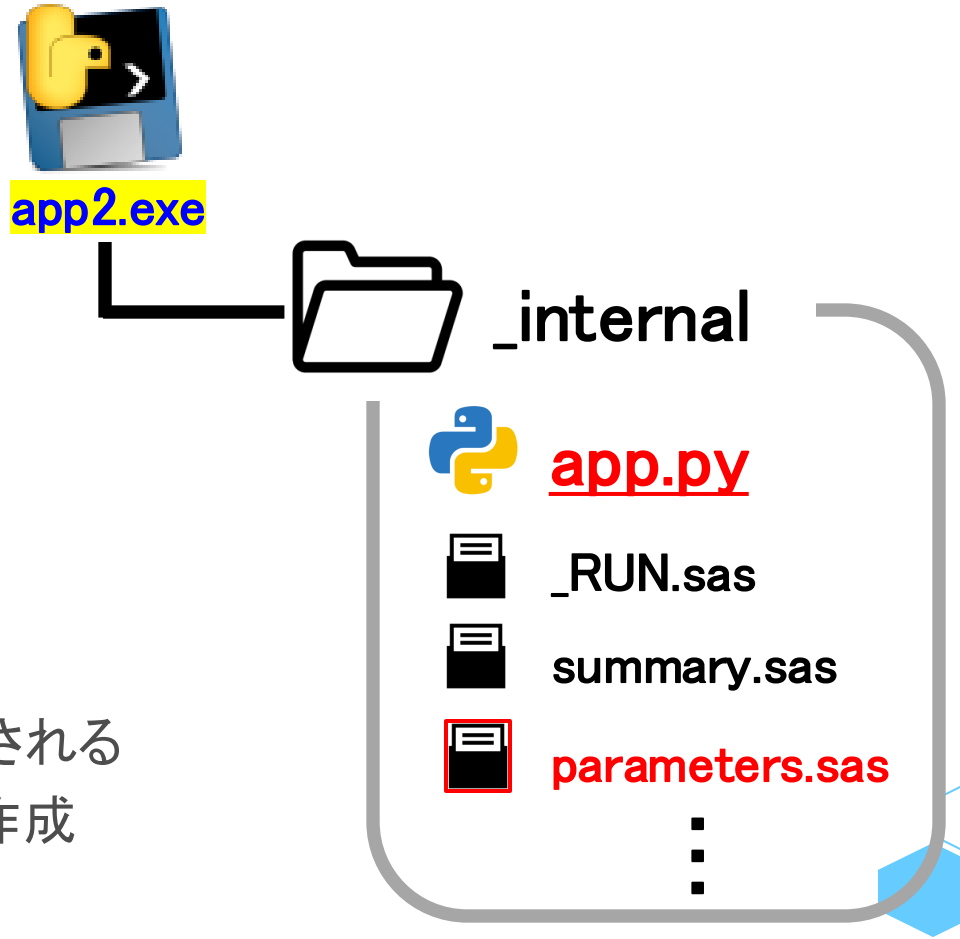
parameters.sas

おまけ：アプリをデスクトップアプリ化

- C:¥temp¥app.py をデスクトップアプリ化 → C:¥temp¥dist¥app¥app2.exe が作成される

```
streamlit-desktop-app build app.py --name "app2"
```

  - app2.exe と _internal フォルダ（依存ライブラリ等が格納）が作成される
  - 容量は大きめです…（ 1 GB 以上）
  - app.py（次頁、メインの python プログラム）は _internal フォルダに格納されるので、ここがカレントディレクトリとなる



- app2.exe 作成後、_RUN.sas と summary.sas を _internal フォルダに手動で格納して完成
  - 配布の際は _internal を含めて zip 等に圧縮
  - parameter.sas はアプリ（app.py）実行時に自動作成される
  - 結果の html ファイルは解析用データと同じ階層に作成（もちろん本仕様は変更可能）

おまけ：アプリをデスクトップアプリ化

- C:¥temp¥app.py

```
import streamlit as st
import pandas    as pd
import subprocess
import easygui
import os

if "is_button" not in st.session_state:
  st.session_state["is_button"]     = False

if "uploaded_file" not in st.session_state:
  st.session_state["uploaded_file"] = ""

def get_local_file_path():
  file_path = easygui.fileopenbox()
  return file_path

os.chdir(os.path.dirname(os.path.abspath(__file__)))

st.title("Upload a SAS and Analyze Data")
if st.button("Choose a SAS file"):
  st.session_state["is_button"]     = True
  st.session_state["uploaded_file"] = get_local_file_path()

if st.session_state["is_button"]:
  file_path     = st.session_state["uploaded_file"]
  _, extension  = os.path.splitext(file_path)
  if extension.lower() == ".sas7bdat":
    folder_name = os.path.dirname(file_path)
    file_name   = os.path.basename(file_path).replace('.sas7bdat', '')
    df = pd.read_sas(file_path, format='sas7bdat', encoding='cp932')
    st.write("DataFrame Preview:")
    st.write(df.head(7))
    vars = [c for c in df.columns]
    var  = st.selectbox("Variable", vars)
    with open("parameters.sas", "w") as f:
      f.write("%let _path = " + folder_name + " ; ")
      f.write("%let _data = " + file_name   + " ; ")
      f.write("%let _var  = " + var         + " ; ")
      st.success(f"Parameters has been saved.")
  else:
    st.write("This file is NOT a SAS file.")

if st.button("Run") and st.session_state["is_button"]:
  sas_exe = r'"C:\Program Files\SAS_9.4\SASFoundation\9.4\sas.exe"'
  sas_run = r'_RUN.sas'
  command = f'{sas_exe} -sysin "{sas_run}" -nosplash -icon'
  subprocess.Popen(command, shell=True)
```

- os.chdir() でカレントディレクトリを _internal に移動
  - _RUN.sas と parameter.sas のパスについて明示的に処理する必要がなくなる
- streamlit のモジュールは、セキュリティの観点でローカルファイルへのアクセスを制限している
  - 緑の部分にて別機能で SAS ファイルの情報を取得
- streamlit では session_state という概念がある
  - 赤の部分にて処理
  - session_state は後ろのスライドにて言及

おまけ：アプリをデスクトップアプリ化

- C:¥temp¥_RUN.sas

```
options source source2 symbolgen ;

%inc "./parameters.sas" ;

%inc "./summary.sas" ;
```

- C:¥temp¥summary.sas

```
libname IN "&_path" ;

ods html file='result.html'
path="&_path" style=styles.sasweb ;
proc means data=IN.&_data. n mean
stddev min q1 median q3 max ;
  var &_var. ;
run ;
proc sgplot data=IN.&_data. ;
  histogram &_var. / scale=count ;
  density   &_var. / type=normal ;
run;
ods html close;

x """&_path\result.html""";
```

お遊び：✦ Gemini に SAS データを解析させるアプリ

注：機密データを入力しないで下さい！

【事前準備】

- Gemini にログイン（ Google アカウントがあればすんなり）、当方は無料版を試用
  https://gemini.google.com/app?hl=ja

- Gemini API を使用するための API キーを取得
  参考： https://monomonotech.jp/kurage/memo/m240725_get_gemini_api_key.html

- C:¥temp に .env というテキストファイルを作成し、以下を書き込む
  （ XXXXX は、ご自身の Gemini API キー）

```
GOOGLE_API_KEY=XXXXXX
```

- Anaconda Prompt 等で、モジュール google-generativeai と dotenv を入れる

```
pip install google-generativeai python-dotenv
```

お遊び: ✦ Gemini に SAS データを解析させるアプリ

注：機密データを入力しないで下さい！

① SASデータの読み込み

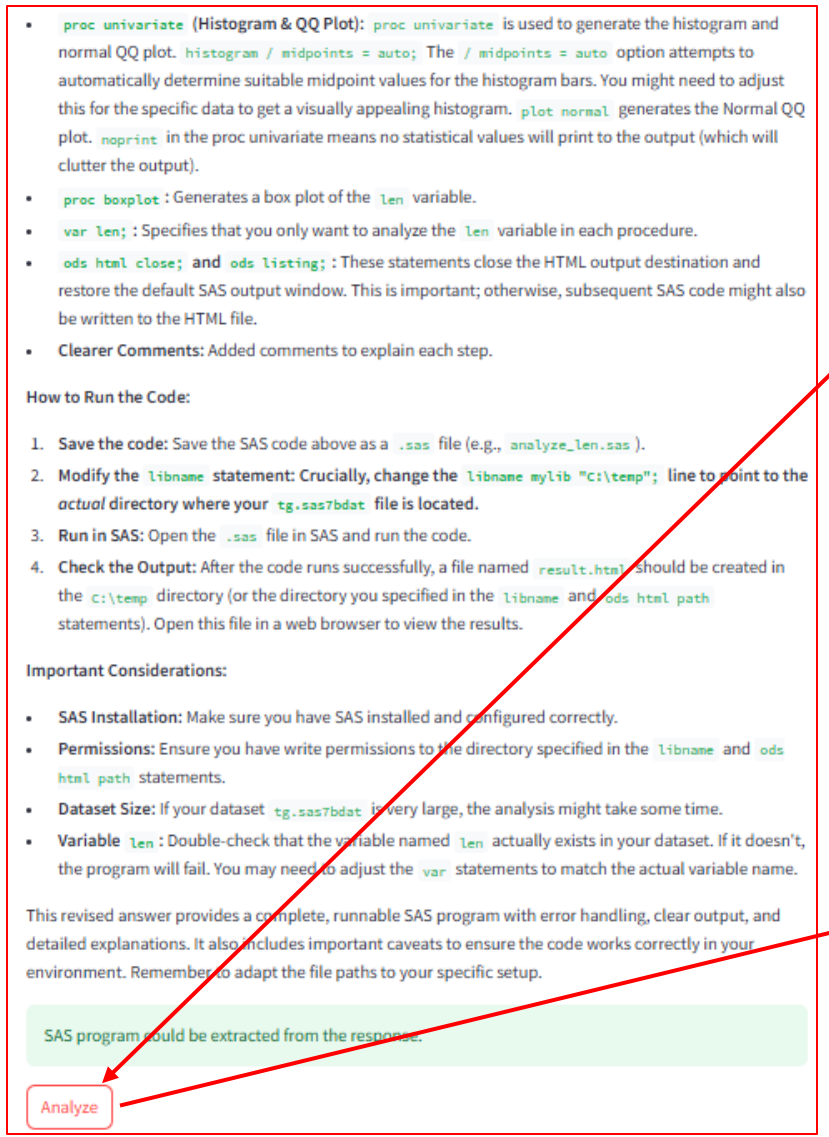
② 選択した変数に関する解析プログラムをGenemiが生成（ファイルも生成）

③ 補足説明も表示

④ [Analyze] で、生成されたSASプログラムを実行

結果と Gemini による要約が出力

結果はお楽しみ…



お遊び：✦Gemini に SAS データを解析させるアプリ

注：機密データを入力しないで下さい！

- C:¥temp¥**app.py**

```
import streamlit as st
import pandas    as pd
import subprocess
import codecs
import time
import os
import google.generativeai as genai
from   dotenv import load_dotenv

st.title("Upload a SAS and Analyze Data")
uploaded_file = st.file_uploader("Choose a SAS file", type="sas7bdat")

if uploaded_file is not None:
  df = pd.read_sas(uploaded_file, format='sas7bdat', encoding='cp932')
  filename = uploaded_file.name.replace('.sas7bdat', '')
  st.write("DataFrame Preview:")
  st.write(df.head(7))
  vars = [c for c in df.columns]
  var  = st.selectbox("Variable", vars)

  load_dotenv()
  GOOGLE_API_KEY = os.getenv('GOOGLE_API_KEY')
  genai.configure(api_key=GOOGLE_API_KEY)
  model = genai.GenerativeModel('gemini-2.0-flash')
  chat  = model.start_chat()
  query = "generate SAS program to analyze the variable " + var + " in
SAS dataset " + uploaded_file.name + " read by this app, and generate
summary statistics and graphs, and then output result.html, using the
path C:\\temp as libname."
  response = chat.send_message(query)
  st.write("Gemini, " + query)
  st.write(response.text)

  sas_code_start = response.text.find("```sas")
  sas_code_end   = response.text.find("```", sas_code_start +
                                      len("```sas"))
  if sas_code_start != -1 and sas_code_end != -1:
    sas_program = response.text[sas_code_start +
                  len("```sas"):sas_code_end].strip()
    try:
      with open(r"C:/temp/program.sas", "w") as file:
        file.write(sas_program)
        st.success(f"SAS program could be extracted from
                     the response.")
    except Exception as e:
      st.error(f"Error: {e}")
  else:
    st.write("SAS program could not be extracted from
              the response.")

if st.button("Analyze"):
  subprocess.Popen(r'"C:\Program
Files\SAS_9.4\SASFoundation\9.4\sas.exe" -sysin
"C:\temp\program.sas" -nosplash -icon', shell=True)
  time.sleep(10)
  try:
    with codecs.open(r"C:/temp/result.html", 'r', 'cp932') as file:
      html_content = file.read()
    summary_query = (
      "Summarize the following content of an HTML report. "
      "Provide the main insights and conclusions:\n" + html_content
    )
    summary_response = chat.send_message(summary_query)
    st.subheader("Summary of SAS Results:")
    st.write(summary_response.text)
    subprocess.Popen(["start", "C:/temp/result.html"], shell=True)
  except Exception as e:
    st.error(f"Error reading or summarizing result.html: {e}")
```

- **赤字部分**で Genemi を呼び出し
- **青字部分**で Genemi が SAS や要約を生成
  - C:¥temp にデータあり、ここに結果を出力
  - 読み込んだデータと指定した変数を解析

まとめ

- streamlit にて
  1. 超簡単にアプリが作れる
  2. SAS を（バッチにて）実行出来る
  3. SAS データも読み込める
  4. なのでアプリが作れたり遊べたりする

- SAS を読み込む方法として saspy や sas7bdat というモジュールもあり

- 次頁以降は streamlit 自体の説明です
  - データ: https://nfunao.web.fc2.com/files/python_intro2.zip

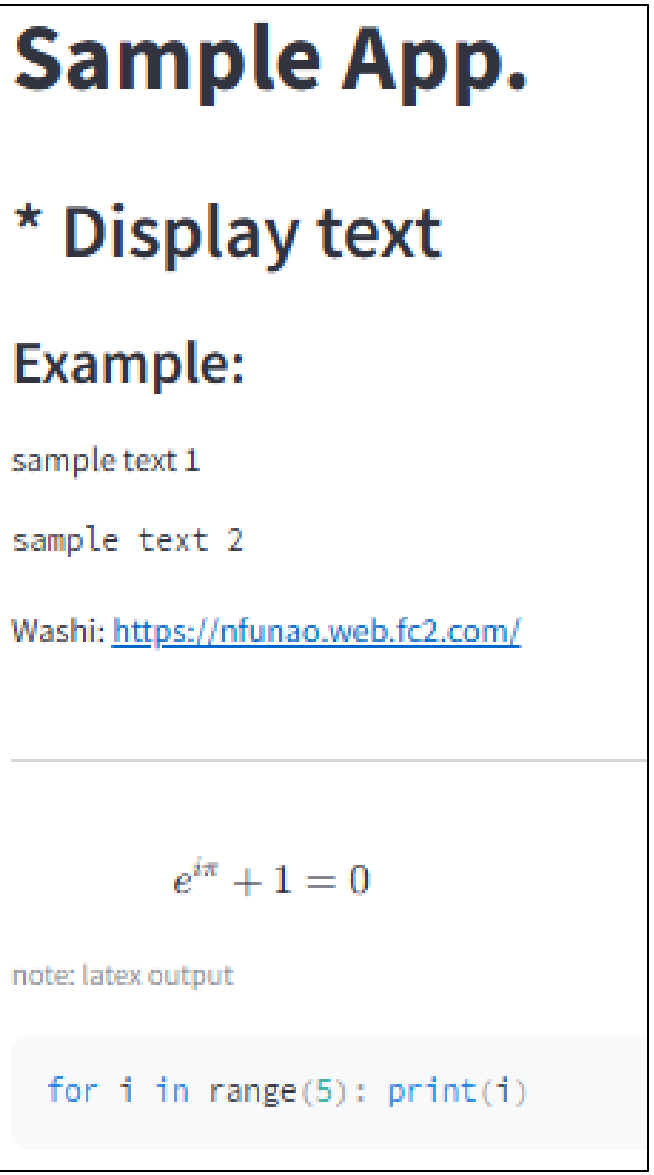
メニュー

- インストール
- アプリの作成 → 実行 → 終了
- 「 SAS データを解析するアプリ」を streamlit で作成
- **各種出力と widget**
- session_state
- レイアウト
- その他

テキスト出力

```
# app.py
import streamlit as st


st.title('Sample App.')
st.header('* Display text')
st.subheader('Example:')
st.write('sample text 1')
st.text('sample text 2')
st.markdown('Washi: https://nfunao.web.fc2.com/')
st.divider()
st.latex(r''' e^{i\pi} + 1 = 0 ''')
st.caption('note: latex output')
st.code('for i in range(5): print(i)')
```



テキスト出力：st.write() が便利

```
# app.py
import streamlit as st
import pandas    as pd

st.title('Sample App.')

markdown = """
#   Title
##  Subtitle
### sample text
```python
for i in range(5): print(i)
```
"""

st.markdown(markdown)

df = pd.DataFrame([[0,1,2], [3,4,5]])
st.write(df)
```

データ出力

```
# app.py
import streamlit as st
import pandas    as pd

st.title('Sample App.')

df = pd.DataFrame([[0,1,2], [3,4,5]])

df # magic command

st.write(df)

st.dataframe(df)

st.table(df)

st.json({'col1':'red','col2':'green'})
```

Sample App.

	0	1	2
0	0	1	2
1	3	4	5

	0	1	2
0	0	1	2
1	3	4	5

	0	1	2
0	0	1	2
1	3	4	5

	0	1	2
0	0	1	2
1	3	4	5


```
{
  "col1" : "red"
  "col2" : "green"
}
```

グラフ出力

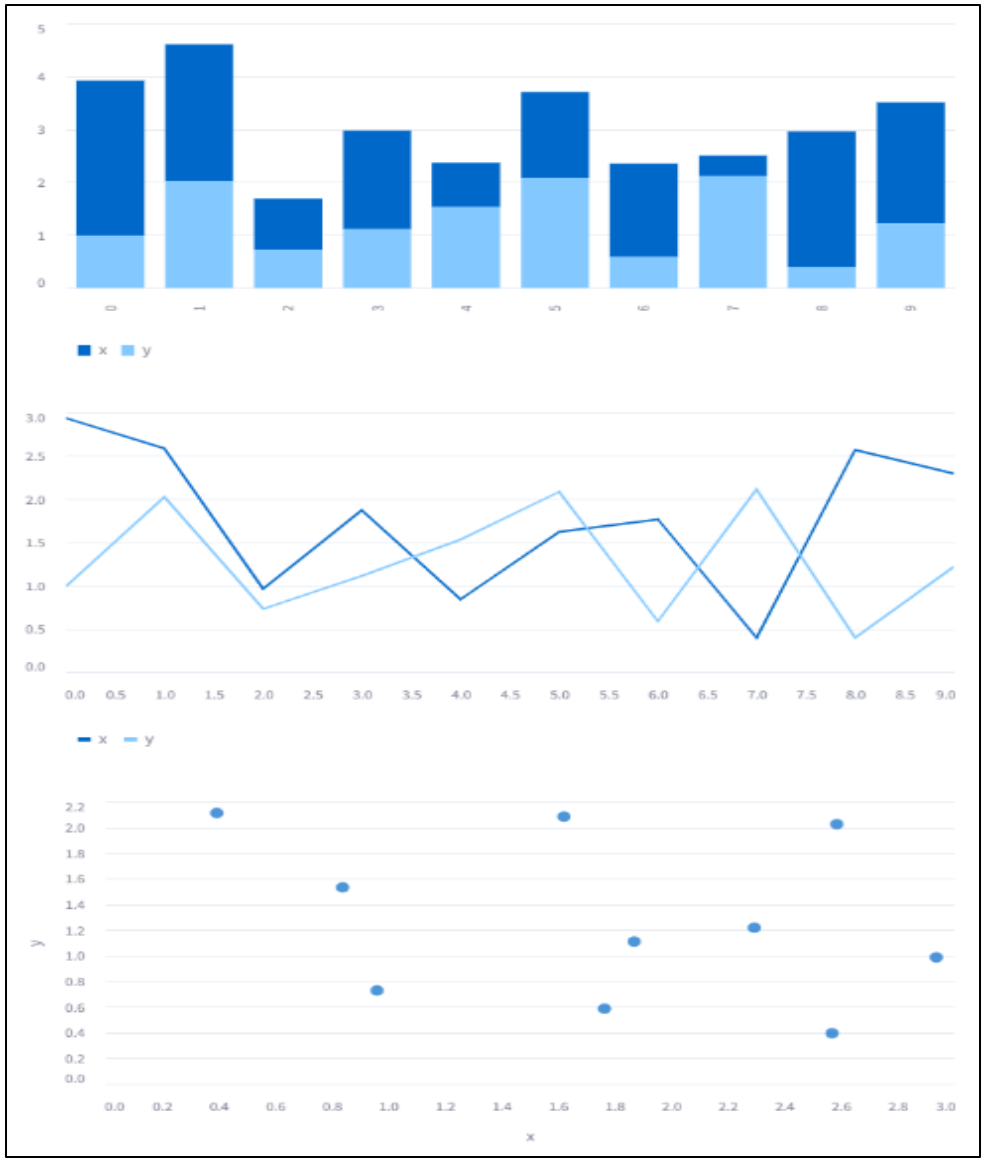
```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

df = pd.DataFrame({
   "x": 3*np.random.rand(10),
   "y": 3*np.random.rand(10),
   "g": np.random.choice(["A", "B"],
10)})

st.bar_chart(df[["x","y"]])

st.line_chart(df[["x","y"]])

st.scatter_chart(df,x="x",y="y")
```



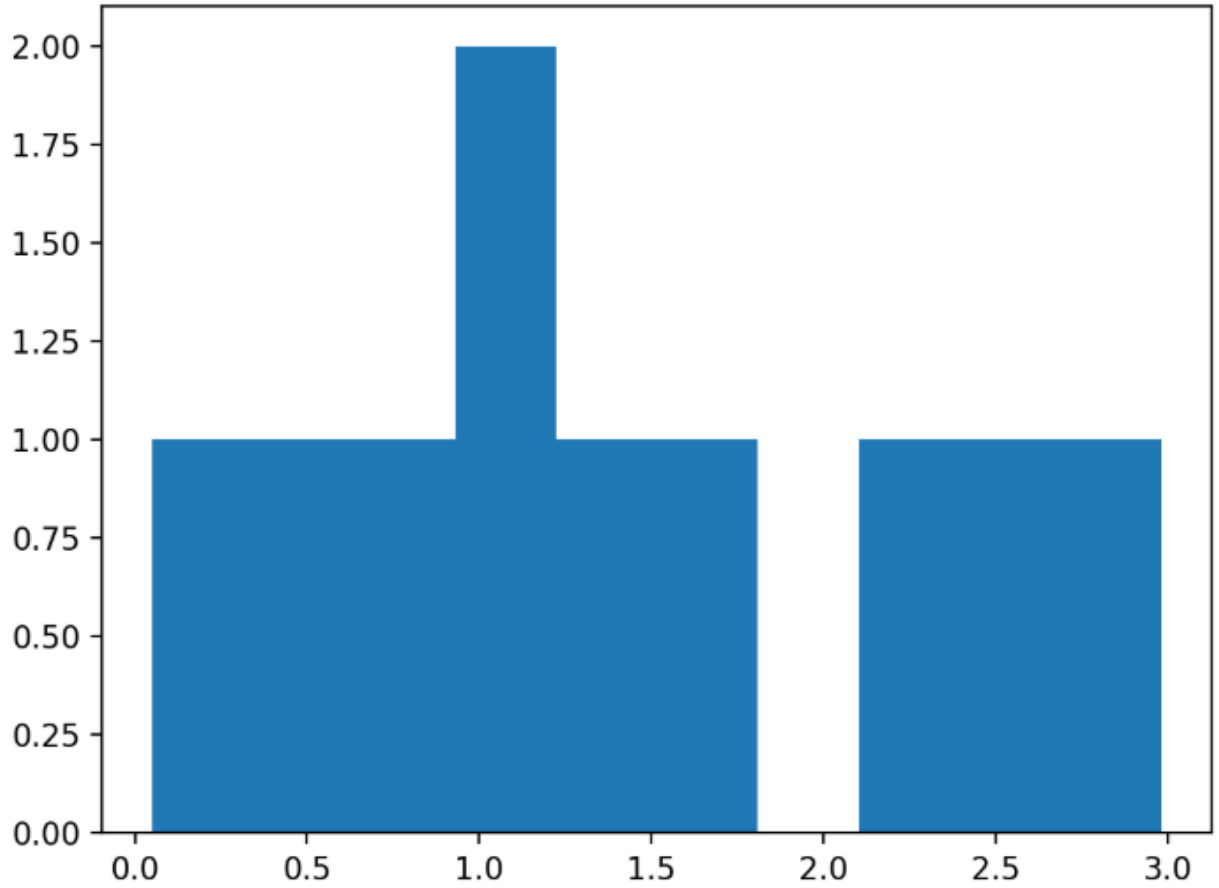
グラフ出力：pyplot 経由

```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np
import matplotlib.pyplot as plt

df = pd.DataFrame({
   "x": 3*np.random.rand(10),
   "y": 3*np.random.rand(10),
   "g": np.random.choice(["A", "B"],
10)})

fig, ax = plt.subplots()
ax.hist(df["x"])

st.pyplot(fig)
```



Widget

Widget	Function
Button	st.button('Label')
Checkbox	st.checkbox('Label')
Color Picker	st.color_picker('Label', '#00ff00')
Date Editor	st.data_editor('Label', data-frame)
Date Input	st.date_input('Label')
Download Button	st.download_button('Label', contents)
File Uploader	st.file_uploader('Label')
Link Button	st.link_button("Label", "https://xxx.com")
Multi-line Text Input	st.text_area('Label')
Multiselect Widget	st.multiselect('Label', [1,2])
Numeric Input	st.number_input('Label')
Radio Button	st.radio('Label:', ['option 1','option 2'])
Select widget	st.selectbox('Label', [1,2])
Single-line Text Input	st.text_input('Label')
Slider	st.slider('Label', min_value=0, max_value=5)
Slider to Select Items	st.select_slider('Label', options=[1,'2'])
Time Input	st.time_input('Label')

Widget: button & checkbox

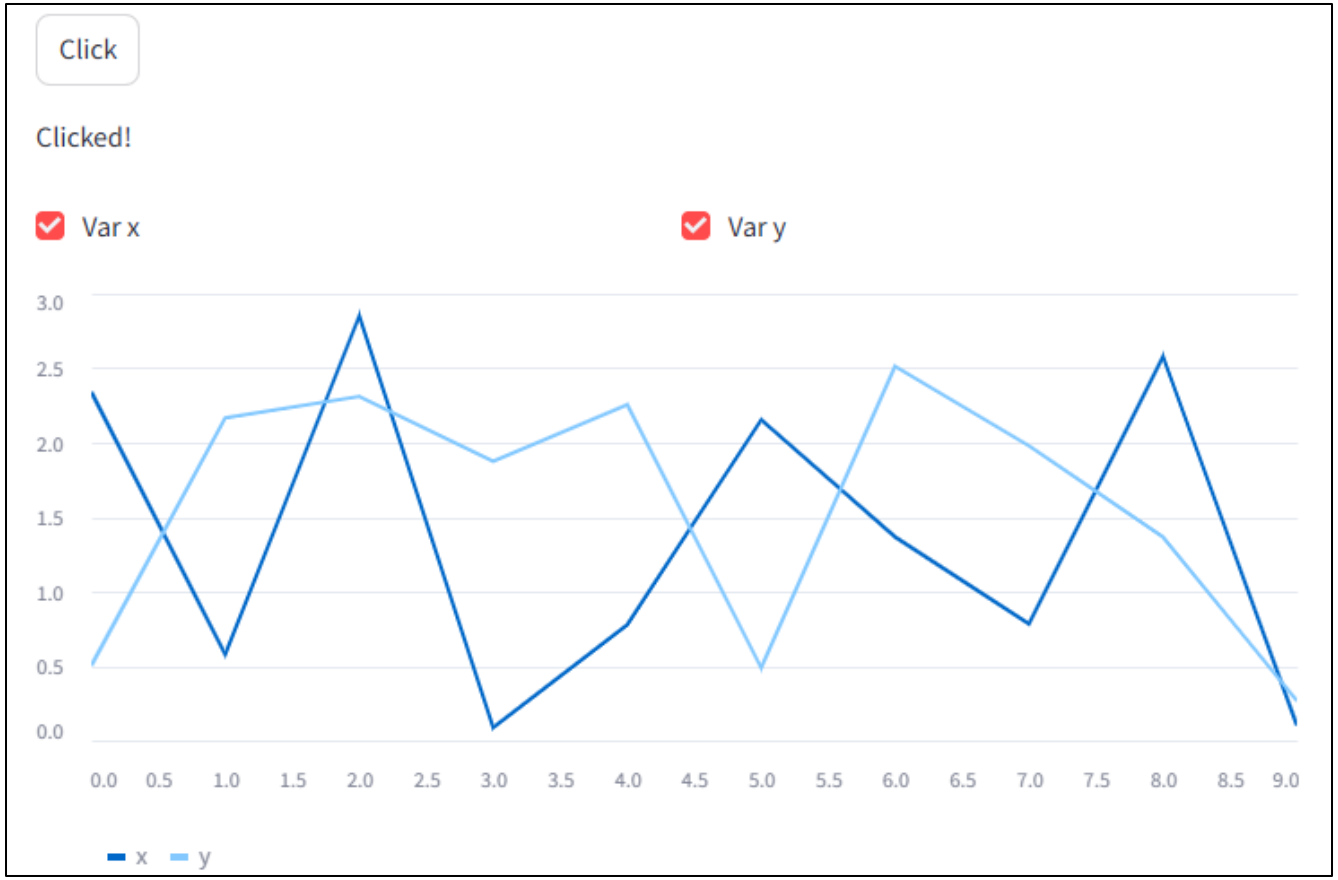
```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

df = pd.DataFrame({
  "x": 3*np.random.rand(10),
  "y": 3*np.random.rand(10),
  "g": np.random.choice(["A", "B"], 10)})

if st.button(label='Click'):
  st.write('Clicked!')

var = st.columns(2)
x   = var[0].checkbox(label='Var x')
y   = var[1].checkbox(label='Var y')

vars = []
if x:
  vars.append('x')
if y:
  vars.append('y')
st.line_chart(df[vars])
```

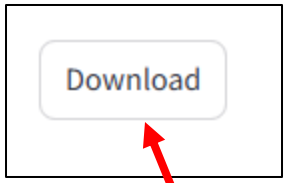


Widget: ファイルのダウンロード

```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

df = pd.DataFrame({
  "x": 3*np.random.rand(10),
  "y": 3*np.random.rand(10),
  "g": np.random.choice(["A", "B"], 10)})
csv = df.to_csv().encode('SHIFT-JIS')

st.download_button(label='Download',
  data=csv, file_name='output.csv',
  mime='text/csv')
```



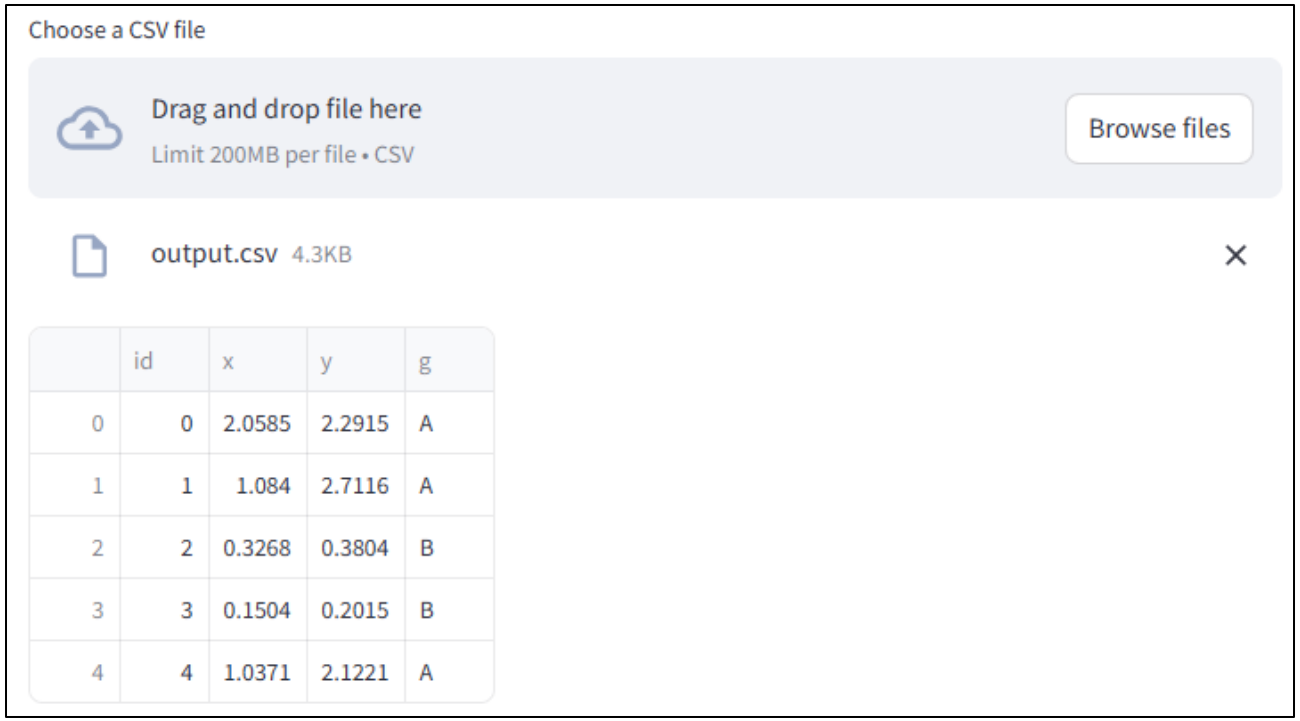
ボタンをクリックするとデータフレームが
CSV形式にてダウンロードされる

Widget: ファイルのアップロード

```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

file = st.file_uploader("Choose a CSV file",
  type='csv')

if file is not None:
  df = pd.read_csv(file)
  st.write(df)
```



	id	x	y	g
0	0	2.0585	2.2915	A
1	1	1.084	2.7116	A
2	2	0.3268	0.3804	B
3	3	0.1504	0.2015	B
4	4	1.0371	2.1221	A

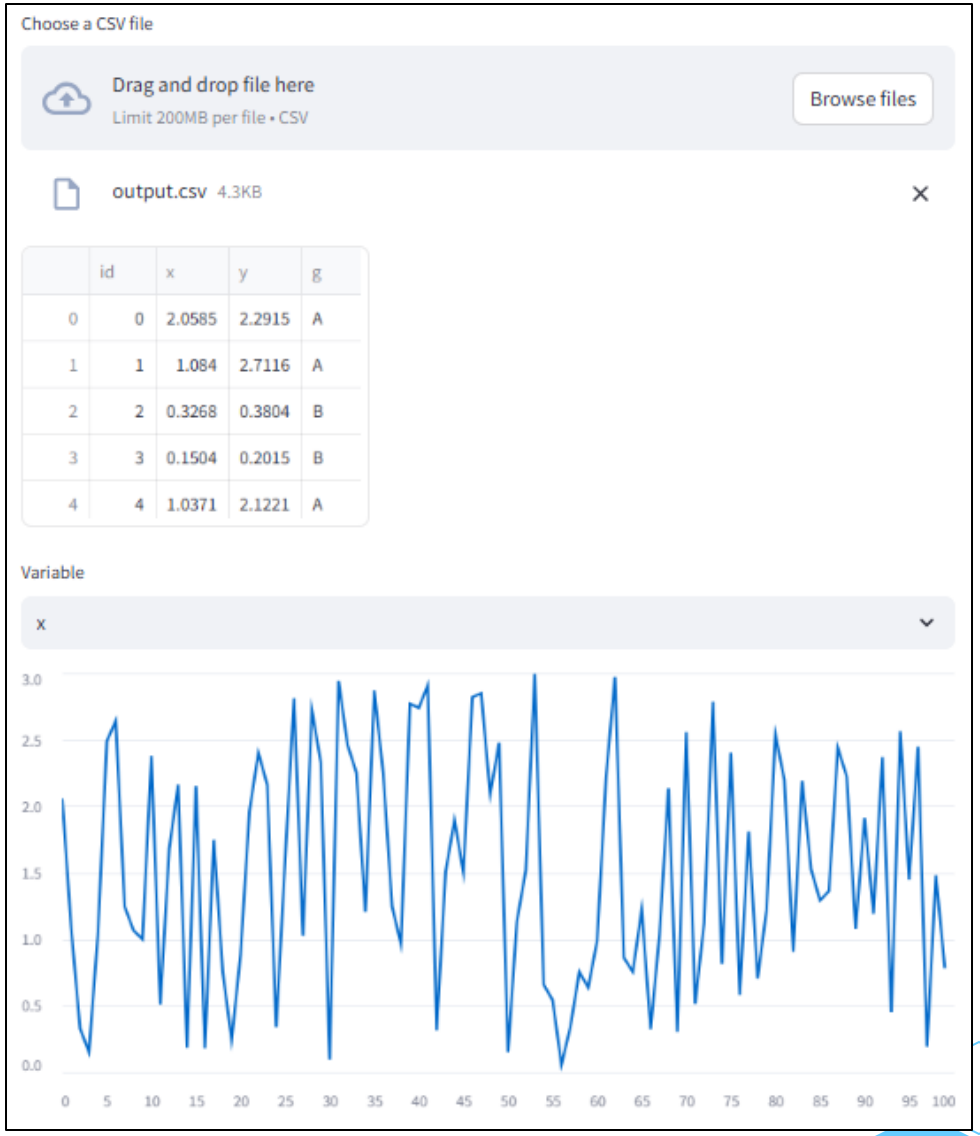
Widget: CSV のアップロード & 数値変数のグラフ

```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

file = st.file_uploader("Choose a CSV file",
  type='csv')

if file is not None:
  df = pd.read_csv(file)
  st.write(df.head())
  vars = [c for c in df.columns]
  column = st.selectbox("Variable", vars)

  numeric_vars = df.select_dtypes("number").columns
  if column in numeric_vars:
    st.line_chart(df[column])
```



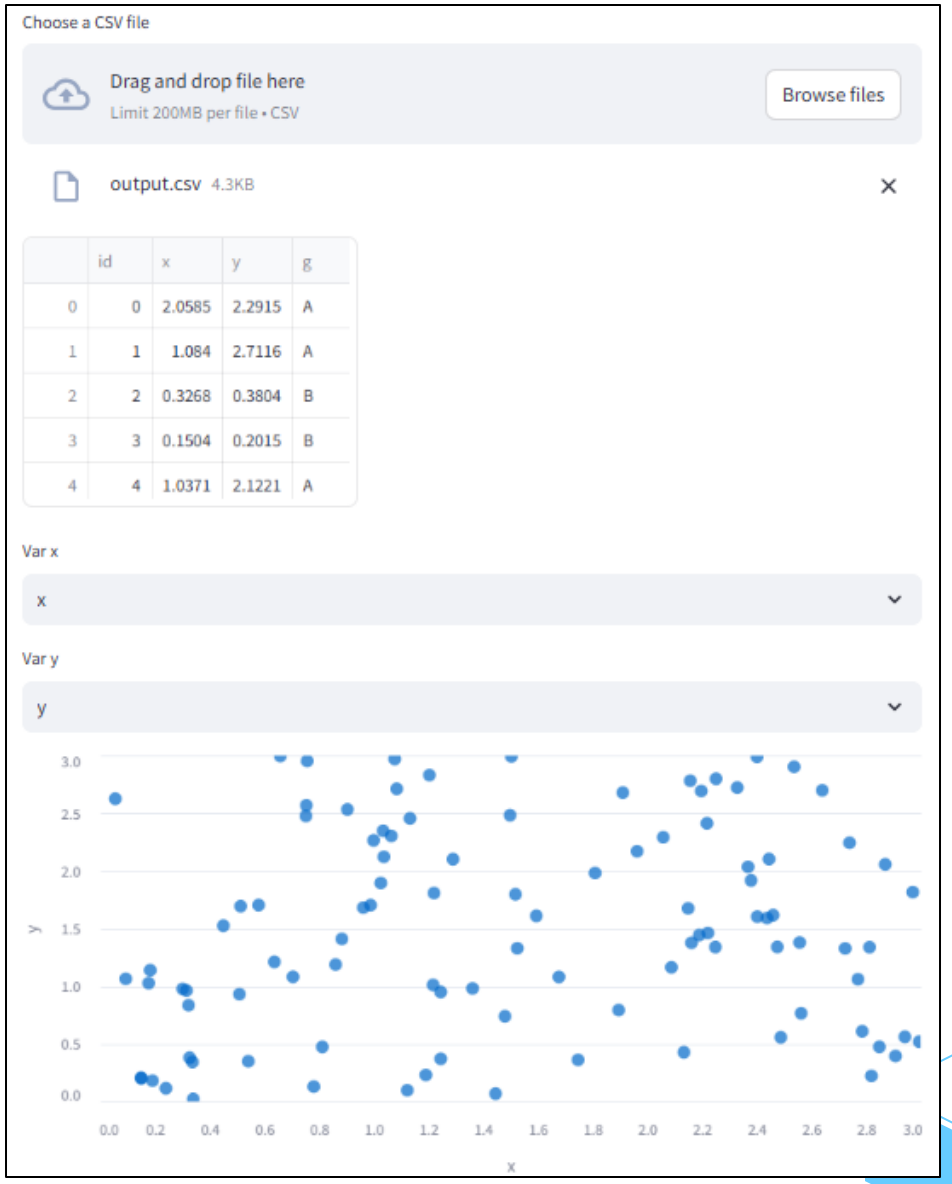
Widget: CSV のアップロード & 数値変数のグラフ

```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

file = st.file_uploader("Choose a CSV file",
  type='csv')

if file is not None:
  df = pd.read_csv(file)
  st.write(df.head())
  vars = [c for c in df.columns]
  x    = st.selectbox("Var x", vars)
  y    = st.selectbox("Var y", vars)

  numeric_vars = df.select_dtypes("number").columns
  if x in numeric_vars and y in numeric_vars:
    st.scatter_chart(df, x=x, y=y)
```



メニュー

- インストール
- アプリの作成 → 実行 → 終了
- 「 SAS データを解析するアプリ」を streamlit で作成
- 各種出力と widget
- **session_state**
- レイアウト
- その他

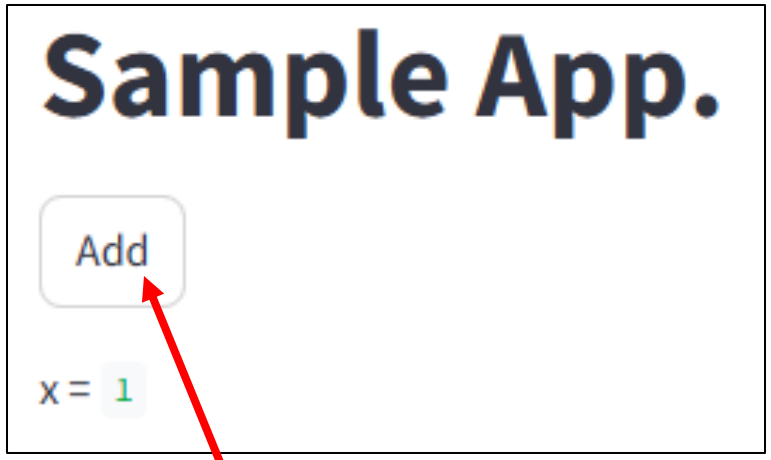
session_state

```
# app.py
import streamlit as st

st.title('Sample App.')
x = 0

if st.button("Add"):
  x = x + 1

st.write("x = ", x)
```



ボタンをクリックすると変数 x に
値がどんどん足されるはずだが
何回クリックしても 1 のまま…

session_state

- streamlit は widget からの入力 ⇒ 値が更新されると、プログラムが一番上から再実行される → 変数は再実行されると初期化されるものも
- widget からの入力があった際、変数の値を保持したい場面がある
  ⇒ session_stete を用いることで変数の値を保持することが出来る
- streamlit の各 widget は 1 つの key と 1 つの value を持ち（≒辞書型）、key が各 widget の ID の役割を果たす
  - session_state は streamlit で用意されている辞書型のオブジェクトで、key と value で各 widget の値を保持することが出来る
  - 以下では key=x と value=1 を session_state に追加している（同じ挙動）

```
st.session_state['x'] = 1
st.session_state.x    = 1
```

session_state：改善例

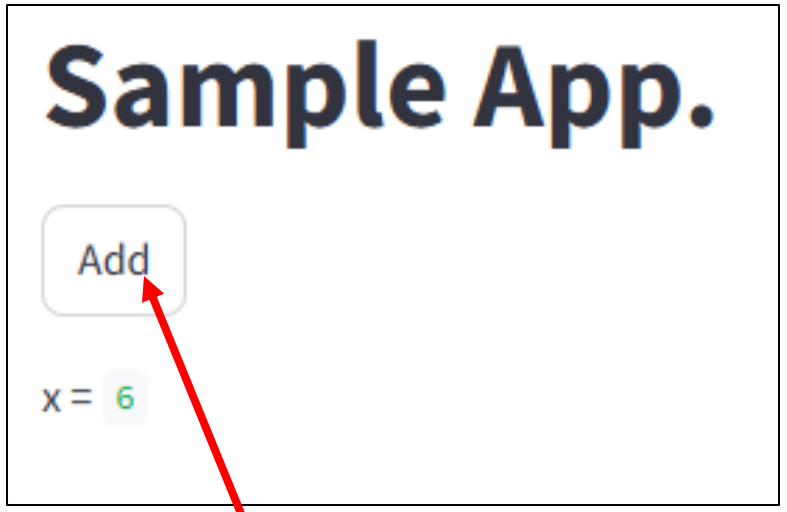
```
# app.py
import streamlit as st

st.title('Sample App.')

if "x" not in st.session_state:
  st.session_state["x"] = 0

if st.button('Add'):
  st.session_state["x"] += 1

st.write("x = ", st.session_state['x'])
```



ボタンをクリックすると変数 x に
値がどんどん足される

- ただし、widget（例えばkey="id"）のvalue は st.session_state["id"] で取得すること
（直接代入の形で取得するとエラーの原因になり得る）
- session_state は「変数を保管する」他に「 widget の value を呼び出す」機能もある
- **参考：覚えておきたい Streamlit での変数取り扱い（ session_stete の注意点）**
https://qiita.com/NewYorki/items/176c8861987d8d0b2794

session_state：数値変数のグラフ

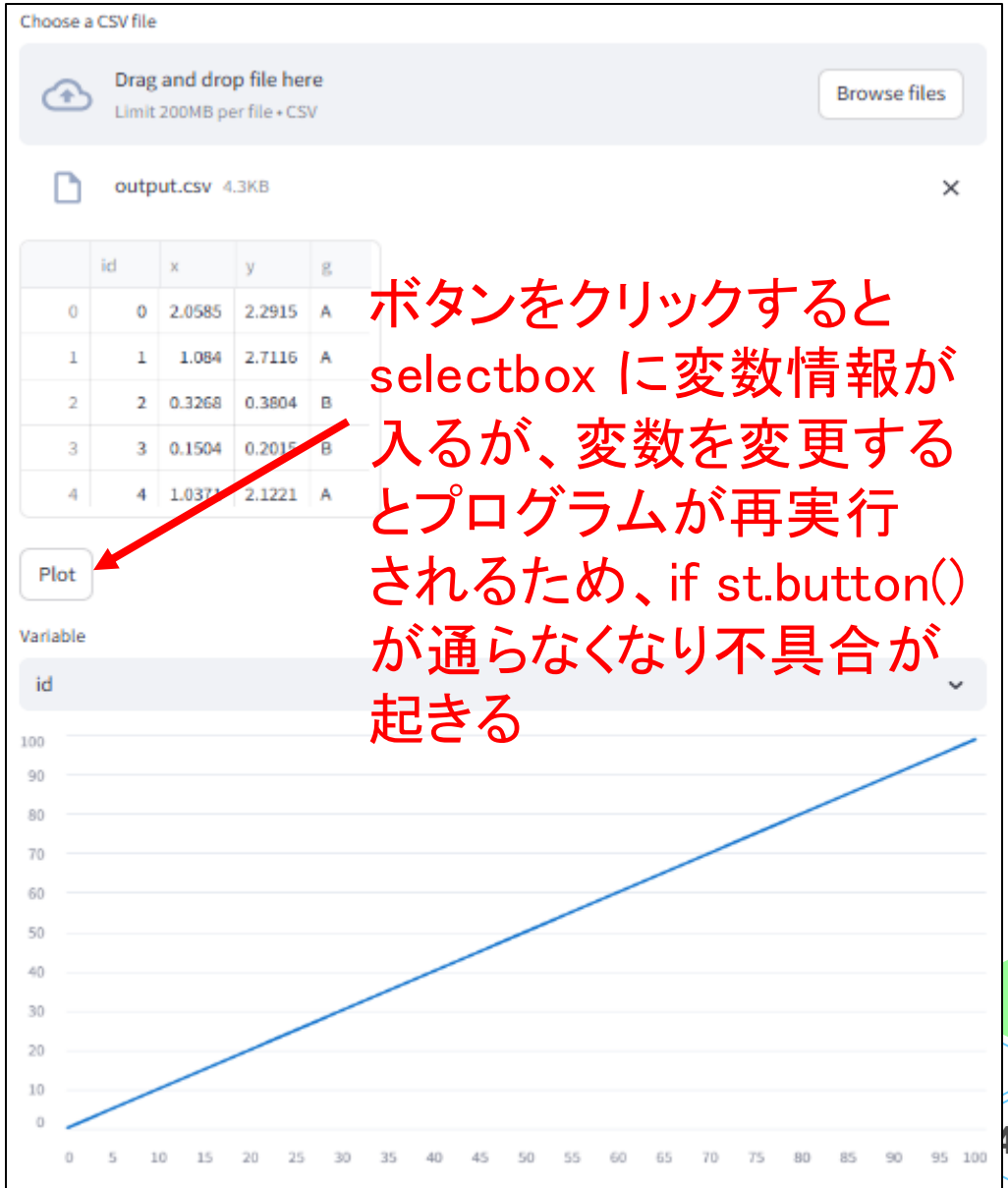
```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

file = st.file_uploader("Choose a CSV file",
type='csv')

if file is not None:
  df = pd.read_csv(file)
  st.write(df.head())

if st.button('Plot'):
  vars = [c for c in df.columns]
  column = st.selectbox("Variable", vars)

  numeric_vars = df.select_dtypes("number").columns
  if column in numeric_vars:
    st.line_chart(df[column])
```



session_state：数値変数のグラフ(改善例)

```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

if "is_button" not in st.session_state:
  st.session_state["is_button"] = False

file = st.file_uploader("Choose a CSV file",
  type='csv')

if file is not None:
  df = pd.read_csv(file)
  st.write(df.head())

if st.button('Plot') or st.session_state["is_button"]:
  st.session_state["is_button"] = True
  vars = [c for c in df.columns]
  column = st.selectbox("Variable", vars)

  numeric_vars = df.select_dtypes("number").columns
  if column in numeric_vars:
    st.line_chart(df[column])
```

メニュー

- インストール
- アプリの作成 → 実行 → 終了
- 「 SAS データを解析するアプリ」を streamlit で作成
- 各種出力と widget
- session_state
- **レイアウト**
- その他

レイアウト：sidebar

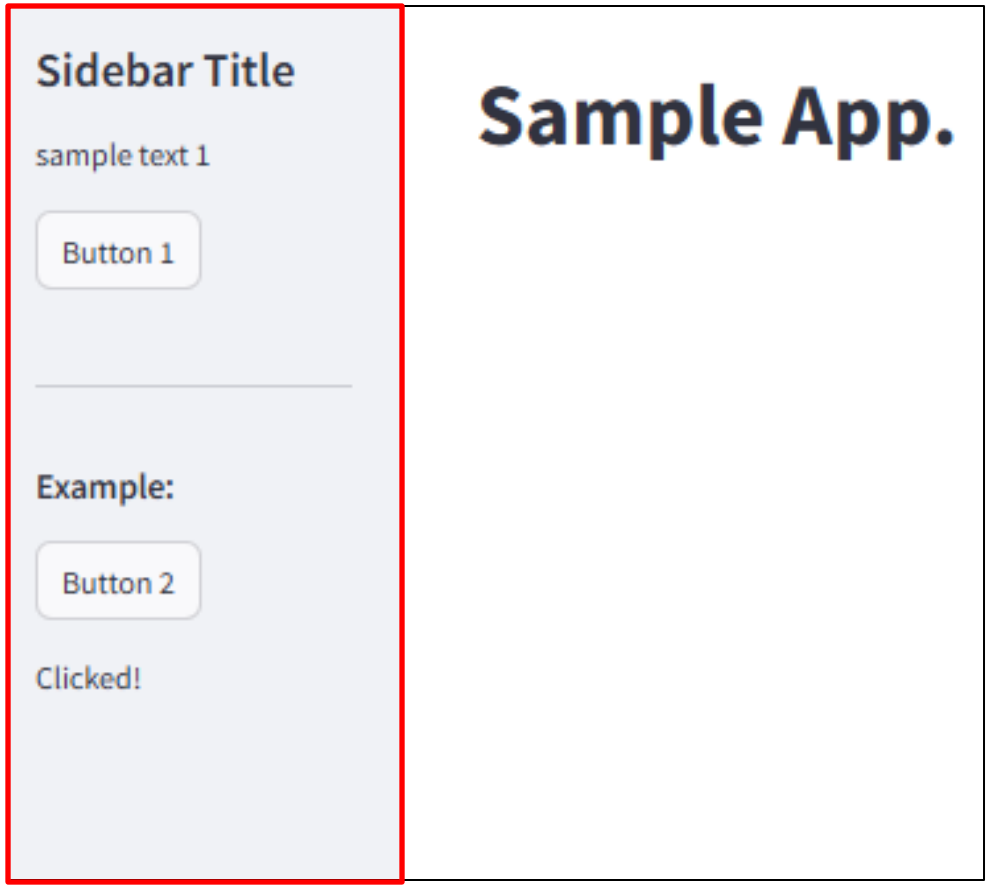
- レイアウトに関する操作（例えば sidebar ）は 2 通りの方法がある
  - st モジュールに対して行っていた操作を st.sidebar に対して行う
  - with st.sidebar の中で操作を行う（こちらの方が楽そう）

```
# app.py
import streamlit as st

st.title('Sample App.')

st.sidebar.title("Sidebar Title")
st.sidebar.write('sample text 1')
if st.sidebar.button(label='Button 1'):
  st.sidebar.write('Clicked!')

with st.sidebar:
  st.divider()
  st.subheader('Example:')
  if st.button(label='Button 2'):
    st.write('Clicked!')
```



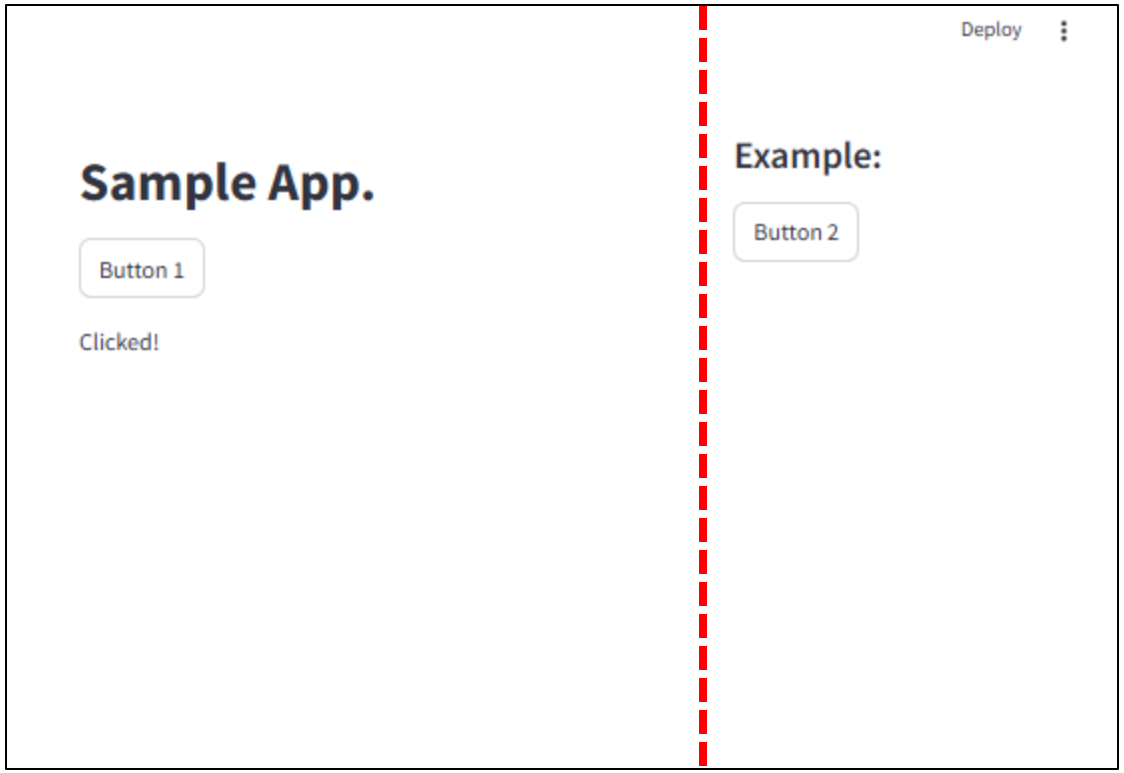
レイアウト: column

```
# app.py
import streamlit as st

# col1, col2 = st.columns(2) # equal witdth
col1, col2 = st.columns([2, 1])

with col1:
  st.title('Sample App.')
  if st.button(label='Button 1'):
    st.write('Clicked!')

with col2:
  st.subheader('Example:')
  if st.button(label='Button 2'):
    st.write('Clicked!')
```



レイアウト：tab

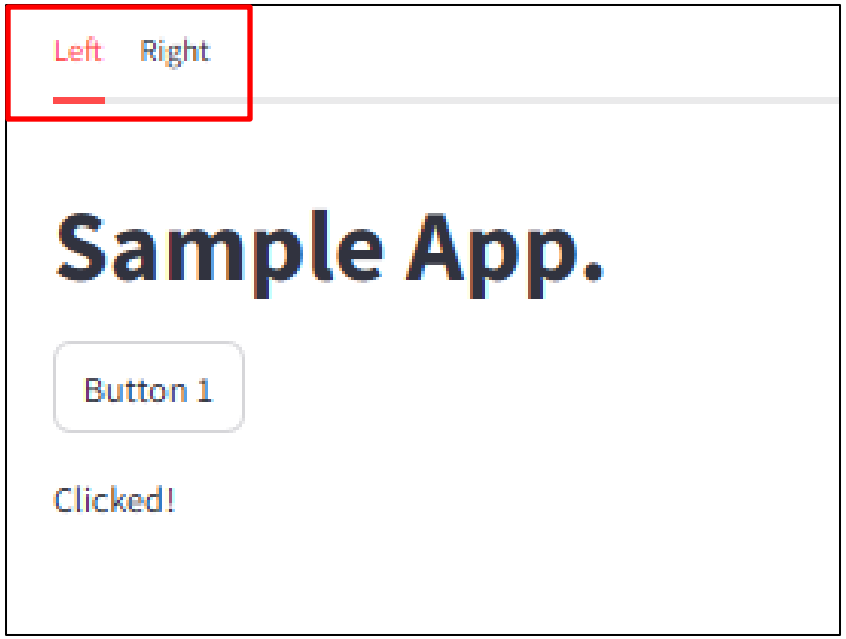
```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

file = st.file_uploader("Choose a CSV file",
type='csv')

if file is not None:
  df = pd.read_csv(file)
  st.write(df.head())

if st.button('Plot'):
  vars = [c for c in df.columns]
  column = st.selectbox("Variable", vars)

  numeric_vars = df.select_dtypes("number").columns
  if column in numeric_vars:
    st.line_chart(df[column])
```

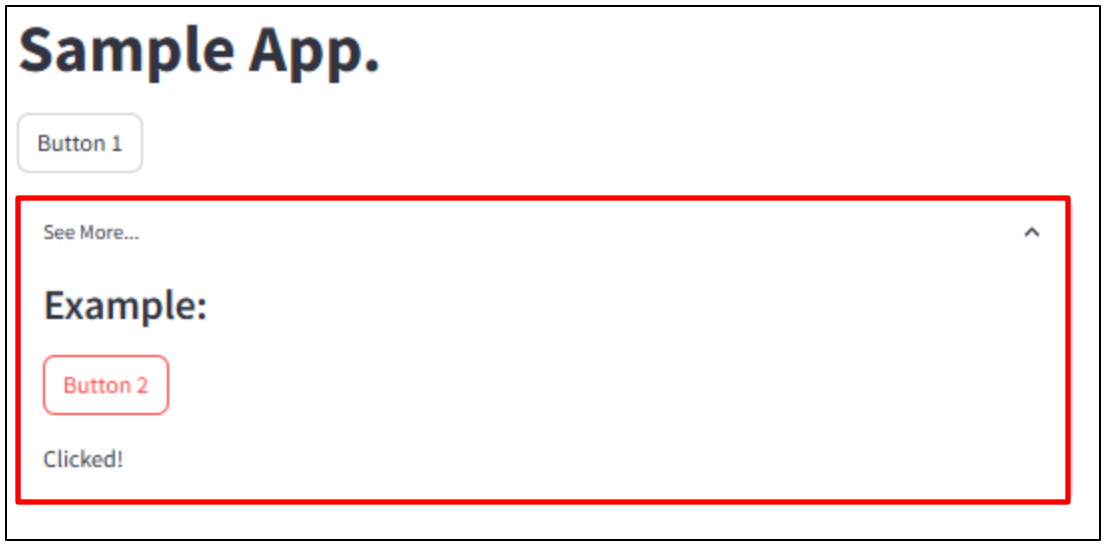


レイアウト：expander（部品の表示・非表示切り替え）

```
# app.py
import streamlit as st

st.title('Sample App.')
if st.button(label='Button 1'):
  st.write('Clicked!')

with st.expander("See More..."):
  st.subheader('Example:')
  if st.button(label='Button 2'):
    st.write('Clicked!')
```



レイアウト：container（複数の部品をまとめる）

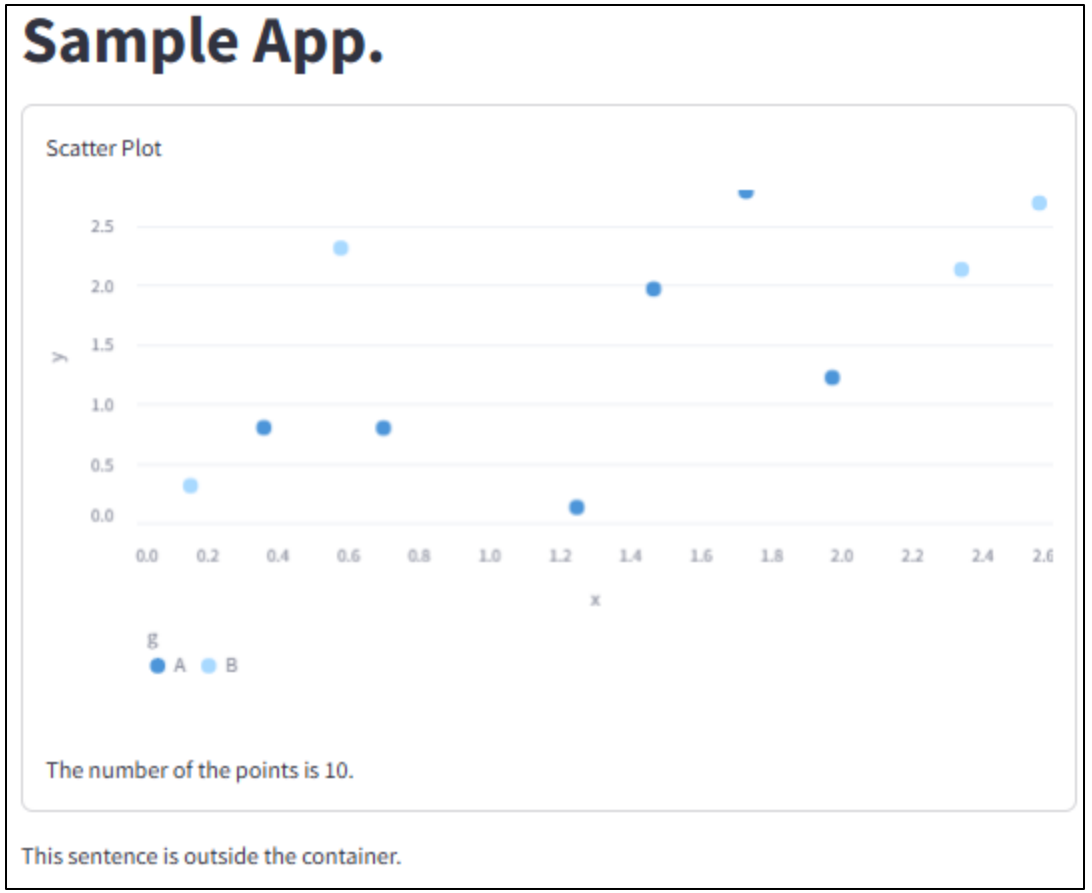
```
# app.py
import streamlit as st
import pandas    as pd
import numpy     as np

st.title('Sample App.')

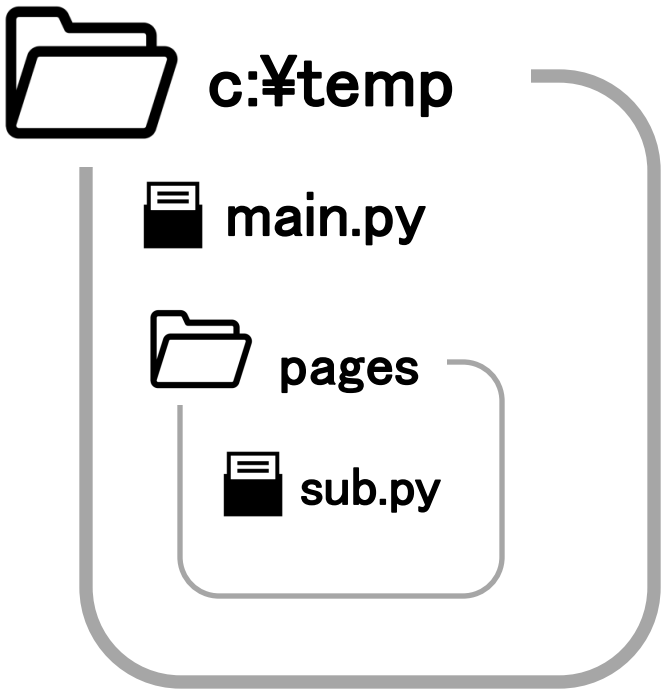
df = pd.DataFrame({
  "x": 3*np.random.rand(10),
  "y": 3*np.random.rand(10),
  "g": np.random.choice(["A", "B"], 10)})

with st.container(border=True):
  st.write("Scatter Plot")
  st.scatter_chart(df,x="x",y="y",color="g")
  st.write("The number of the points is 10.")

st.write("This sentence is outside the container.")
```



レイアウト：複数ページ



- C:¥temp¥**main.py** ← こちらを実行

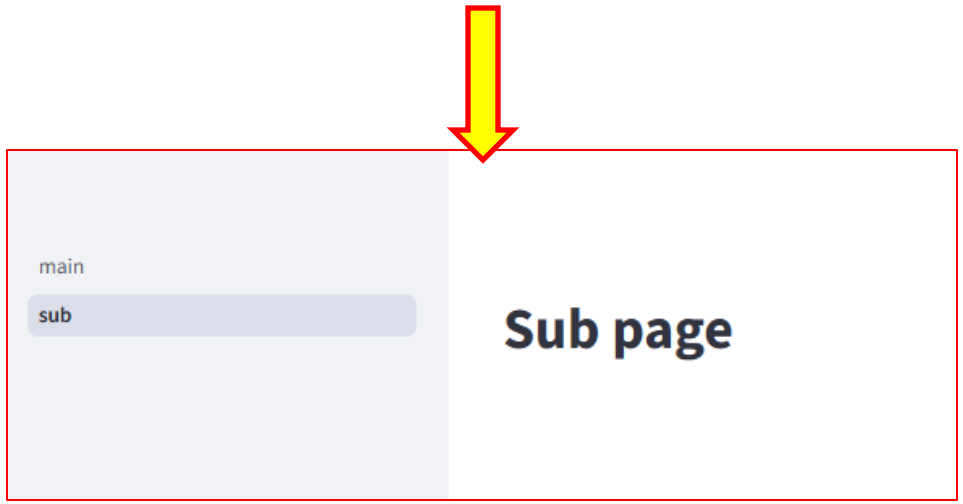
```
import streamlit as st

st.set_page_config(page_title="Main", page_icon="")
st.title("Main page")
```

- C:¥temp¥pages¥**sub.py**

```
import streamlit as st

st.set_page_config(page_title="Sub", page_icon="")
st.title("Sub page")
```



レイアウト：その他

- ページ設定： set_page_config
  https://docs.streamlit.io/develop/api-reference/configuration/st.set_page_config

- テーマの変更
  https://docs.streamlit.io/develop/concepts/configuration/theming

メニュー

- インストール
- アプリの作成 → 実行 → 終了
- 「 SAS データを解析するアプリ」を streamlit で作成
- 各種出力と widget
- session_state
- レイアウト
- **その他**

アプリ作成例：解析ソフト

```
# app.py
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

st.title("Upload a CSV and Analyze Data")
uploaded_file = st.file_uploader("Choose a CSV file",
type="csv")

if uploaded_file is not None:
  df = pd.read_csv(uploaded_file)
  st.write("DataFrame Preview:")
  st.write(df.head())
  vars  = [c for c in df.columns]
  value = st.selectbox("Variable", vars)
  group = st.selectbox("Group", vars)

  if st.checkbox('Grouped'):
    summary_stats = df.groupby(group)[value].describe()
    st.subheader("Summary Statistics by Group")
    st.write(summary_stats)

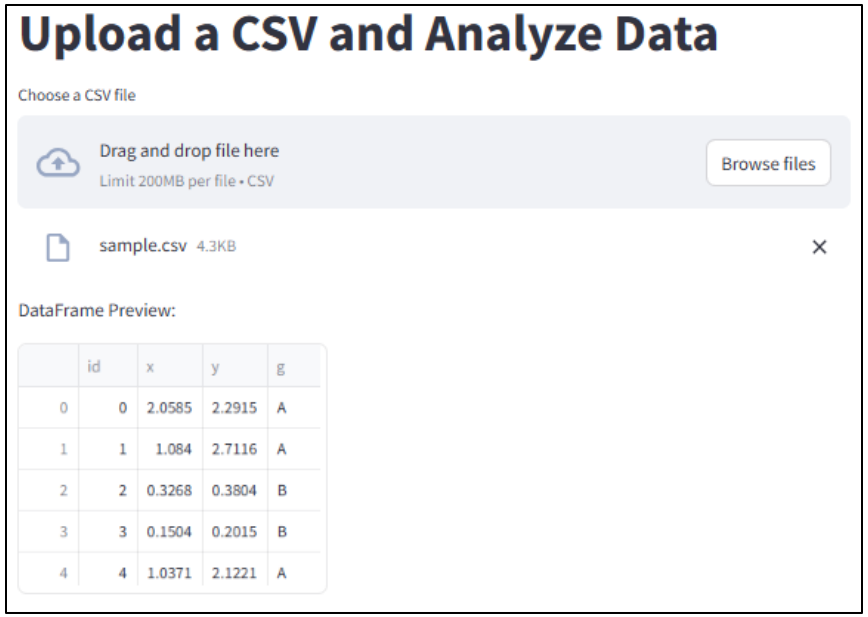
    st.subheader("Histogram by Group")
    groups = df[group].unique()
    num_groups = len(groups)
```

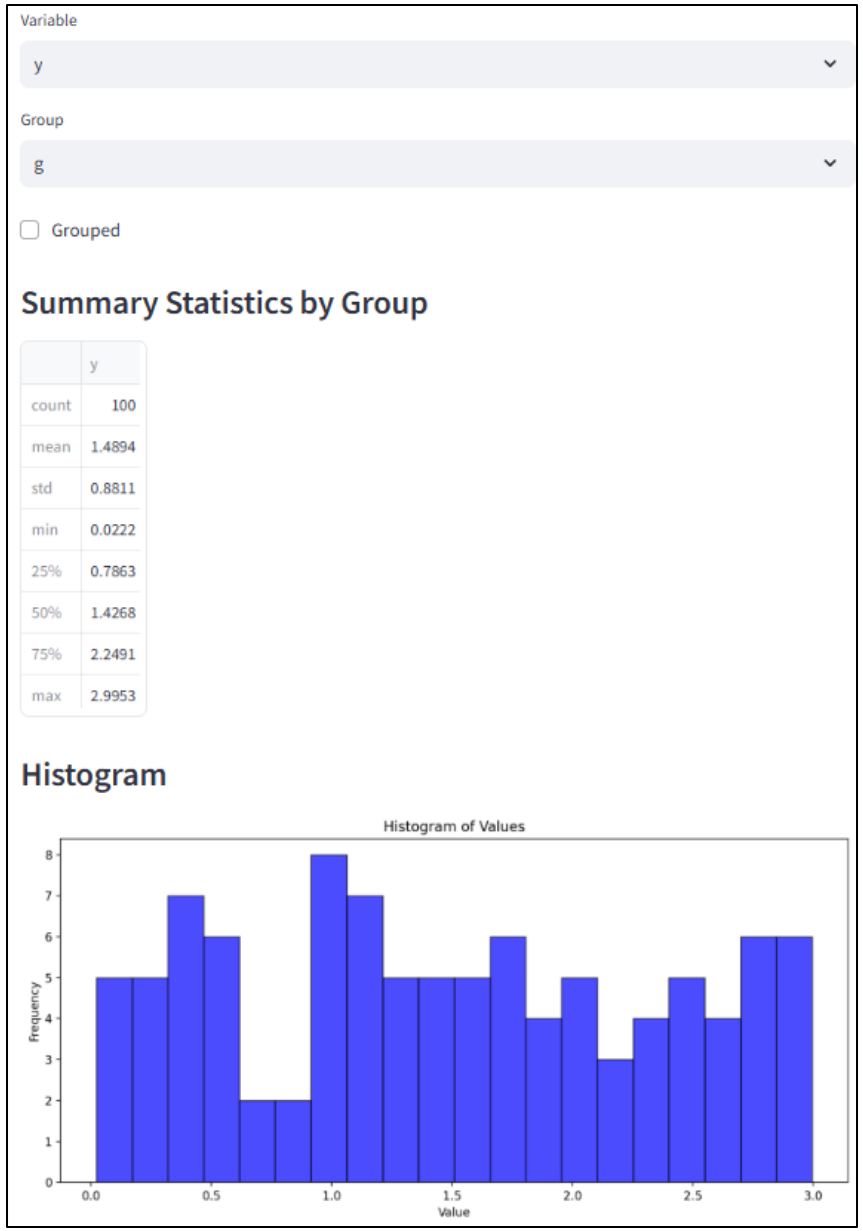
```
if num_groups <11:
     fig, axs = plt.subplots(num_groups, 1, figsize=(10, 5 * num_groups))
     if num_groups == 1:
       axs = [axs]
     for ax, grp in zip(axs, groups):
       group_data = df[df[group] == grp][value]
       ax.hist(group_data, bins=20, alpha=0.7, color='blue',
               edgecolor='black')
       ax.set_title(f'Histogram of Values for Group {grp}')
       ax.set_xlabel('Value')
       ax.set_ylabel('Frequency')
     plt.tight_layout()
     st.pyplot(fig)

 else:
   summary_stats = df[value].describe()
   st.subheader("Summary Statistics by Group")
   st.write(summary_stats)

   st.subheader("Histogram")
   fig, ax = plt.subplots(1, 1, figsize=(10, 5))
   ax.hist(df[value], bins=20, alpha=0.7, color='blue',
           edgecolor='black')
   ax.set_title(f'Histogram of Values')
   ax.set_xlabel('Value')
   ax.set_ylabel('Frequency')
   plt.tight_layout()
   st.pyplot(fig)
```

アプリ作成例：https://f5lppq2xsk877wluqd3gj3.streamlit.app/





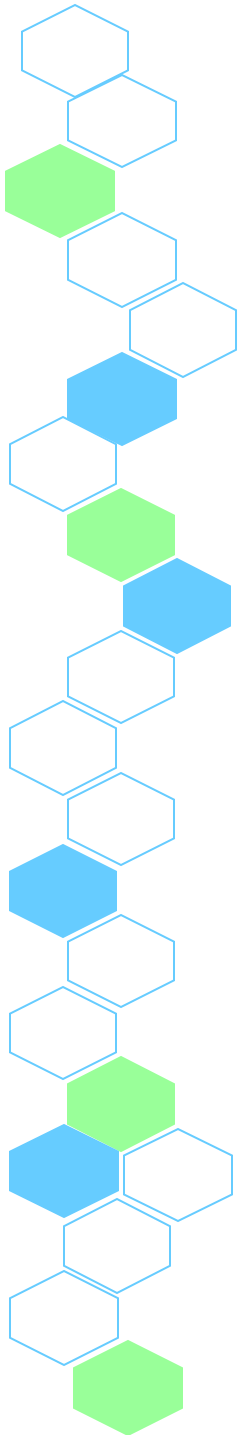
Streamly Streamlit Assistant

- https://streamly.streamlit.app/

  streamlit 用生成 AI がプログラム作成をサポート！

参考文献

- Streamlit documentation
  https://docs.streamlit.io/
- Streamlit API cheat sheet
  https://docs.streamlit.io/develop/quick-reference/cheat-sheet
- Streamlit Magic Cheat Sheets
  https://cheat-sheets.streamlit.app/
- 覚えておきたい Streamlit での変数取り扱い（ session_stete の注意点）
  https://qiita.com/NewYorki/items/176c8861987d8d0b2794
- Pandas User's Guide
  https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html
- 【コマンドプロンプト】目的別コマンド一覧
  https://techmania.jp/blog/cmd0002/
- 舟尾 暢男「streamlit でアプリ作成」
  https://nfunao.web.fc2.com/
  https://nfunao.web.fc2.com/files/python_intro2.zip



python
powered

– End of File –