

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: <u>https://nfunao.web.fc2.com/files/python_intro2_06.zip</u>



- ・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 <u>https://www.python.org/</u> <u>https://www.anaconda.com/download</u>
- 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)



Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects. The Select a demo from the sidebar to see some examples of what Streamlit can do!

Want to learn more?

- Check out <u>streamlit.io</u>
- Jump into our <u>documentation</u>
- Ask a question in our <u>community forums</u>

See more complex demos

- Use a neural net to <u>analyze the Udacity Self-driving Car Image Dataset</u>
- Explore a <u>New York City rideshare dataset</u>





python



×____

- ・ 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 <u>app.py</u> 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: <u>https://techmania.jp/blog/cmd0002/</u>

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¥<u>app.py</u>



【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 「<u>SAS で</u>、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?
 - <u>Streamlit Magic Cheat Sheets</u>
 <u>https://cheat-sheets.streamlit.app/</u>
 - Streamlit documentation

https://docs.streamlit.io/

Settings
Development
Run on save
Automatically updates the app when the underlying code is updated.
Appearance
Uide mode
Turn on to make this app occupy the entire width of the screen
Choose app theme, colors and fonts
Use system setting
Edit active theme



×

- ・ 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¥<u>app.py</u> → <mark>黄色マーカー</mark>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¥<u>sample.sas</u>

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¥<u>app.py</u> → 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



11







- C:¥temp に <u>EXE.sas</u>(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





• C:¥temp¥<u>EXE.sas</u>

```
options source source2 symbolgen ;
%let _path = C:/temp ;
%inc "./parameters.sas" ;
%inc "./samplesize.sas" ;
```

C:¥temp¥<u>samplesize.sas</u>

```
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 <u>EXE.sas</u>(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¥<u>app.py</u> → 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)
```





Sample Size - 2-sample	e t-test	SAS シス	テム
		POWER プロ3 平均差に対する 2 ³	ンジャ 標本 t 検定
Meandiff:		固定シナリオ	要素
1	- +	分布	正規
Stddev:		方法	正確
2	- +	有意水準	0.05
Alpha:		平均差	1
0.05	- +	標準偏差	2
		名目の検出力	0.8
Power:		裾の数	2
0.80	- +	帰無仮説の差	0
Save Parameters		群1の重み	1
		群 2 の重み	1
Parameters has been saved.		計算された N	Total
		Actual Power	N Total
Calculate		0.801	128







- C:¥temp に <u>RUN.sas</u>(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





• C:¥temp¥<u>RUN.sas</u>

```
options source source2 symbolgen ;
%let _path = C:/temp ;
%inc "./parameters.sas" ;
%inc "./summary.sas" ;
```

• C:¥temp¥<u>summary.sas</u>

```
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 <u>RUN.sas</u>(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¥<u>app.py</u> → 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)
```



19

SAS システム

5. SAS データを解析するアプリ

Upload a SAS and Analyze Data



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 <u>https://qiita.com/Gyutan/items/c158920f0693fb099e92</u>

おまけ: アプリをデスクトップアプリ化





おまけ: アプリをデスクトップアプリ化

• C:¥temp¥<u>app.py</u> をデスクトップアプリ化 → C:¥temp¥dist¥app¥<mark>app2.exe</mark> 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 _internal app.py **RUN**.sas summary.sas parameters.sas
- <a>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¥<u>app.py</u>

```
import streamlit as st
import pandas
                 as pd
import subprocess
import easyqui
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 = easyqui.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¥<u>summary.sas</u>

```
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""";
```



【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 /こ SAS データを解析させるアプリー

run:

を入力したいで 下さい! **Upload a SAS and Analyze Data** Choose a SAS file proc univariate (Histogram & QQ Plot): proc univariate is used to generate the histogram and ③ 補足説明も表示 ① SASデータ Drag and drop file here Browse files normal QO plot. histogram / midpoints = auto; The / midpoints = auto option attempts to automatically determine suitable midpoint values for the histogram bars. You might need to adjust の読み込み this for the specific data to get a visually appealing histogram. plot normal generates the Normal QQ tg.sas7bdat 128.0KB × plot. noprint in the procunivariate means no statistical values will print to the output (which will clutter the output). DataFrame Preview: proc boxplot : Generates a box plot of the len variable. len supp dose var len; : Specifies that you only want to analyze the len variable in each procedure. ④ [Analyze] で、 4.2 VC 0.5 ods html close; and ods listing; These statements close the HTML output destination and restore the default SAS output window. This is important; otherwise, subsequent SAS code might also 11.5 VC 0.5 生成されたSAS be written to the HTML file. 7.3 VC 0.5 Clearer Comments: Added comments to explain each step. 5.8 VC 0.5 プログラムを実行 How to Run the Code: 6.4 VC 0.5 10 VC 0.5 1. Save the code: Save the SAS code above as a .sas file (e.g., analyze_len.sas). 6 11.2 VC 0.5 2. Modify the libname statement: Crucially, change the libname mylib "C:\temp"; line to joint to the actual directory where your tg.sas7bdat file is located. 結果と Gemini に Variable 3. Run in SAS: Open the .sas file in SAS and run the code. 2 選択した変数 – 4. Check the Output: After the code runs successfully, a file named result.html should be created in よる要約が出力 the c:\temp directory (or the directory you specified in the libname and ods html path に関する解析 Gemini, generate SAS program to analyze the variable len in SAS dataset tg.sas7bdat read by this app, and statements). Open this file in a web browser to view the results. generate summary statistics and graphs, and then output result.html, using the path C:\temp as libname. Important Considerations: プログラムを /* Set the libname for the SAS dataset location */ SAS Installation: Make sure you have SAS installed and configured correctly. libname temp 'C:\temp': Genemiが生成 Permissions: Ensure you have write permissions to the directory specified in the libname and ods /* Check if the dataset exists */ 結果はお楽しみ… html path statements. proc contents data=temp.tg; (ファイルも生成) Dataset Size: If your dataset tg.sas7bdat is very large, the analysis might take some time. run; · Variable len : Double-check that the variable named len actually exists in your dataset. If it doesn't, /* Analyze the variable 'len' */ the program will fail. You may need to adjust the yar statements to match the actual variable name. ods html file="C:\temp\result.html"; This revised answer provides a complete, runnable SAS program with error handling, clear output, and proc means data=temp.tg mean median stddev min max q1 q3; detailed explanations. It also includes important caveats to ensure the code works correctly in yo var len; environment. Remember to adapt the file paths to your specific setup. run; /* Generate a histogram */ SAS program could be extracted from the response proc sgplot data=temp.tg; histogram len; title "Histogram of Variable Len"; Analyze

```
お遊び: 
◆ Gemini /こ SAS データを解析させるアプリ
```

• C:¥temp¥<u>app.py</u>

```
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 S\overline{A}S 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" -svsin
"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
                                                            27
        読み込んだデータと指定した変数を解析
```

thon



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: <u>https://nfunao.web.fc2.com/files/python_intro2.zip</u>



×____

- ・ 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)')
```

Sample App.
* Display text
Example:
sample text 1
sample text 2
Washi: <u>https://nfunao.web.fc2.com/</u>
$e^{i\pi} + 1 = 0$
note: latex output
<pre>for i in range(5): print(i)</pre>

テキスト出力: st.write() が便利



Sample App. Title Subtitle sample text for i in range(5): print(i) 0 1 2 1 0 0 2 1 3 4 5

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



<b># app.py</b> import streamlit as st	Sample
import pandas as pd	0 1
st.title('Sample App.')	1 3 4
df = pd.DataFrame([[0,1,2], [3,4,5]])	0 1
df # magic command	1 3 4
st.write(df)	0 1 0 0 1
st.dataframe(df)	1 3 4
st.table(df)	0
<pre>st.json({'col1':'red','col2':'green'})</pre>	<pre>     {         "col1": "red"         "col2": "green" } </pre>

Sa	m	p	l	e	A	p
	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	3		4		5
)						
{ "co	11"	: '	'rec	1"		

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







34





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

#### Widget: button & checkbox





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

Download



# Widget: ファイルのアップロード

	Choose a	a CSV file				
<b># app.py</b> import streamlit as st		Drag Limit	and dro 200MB pe	<b>p file he</b> r er file • CS	re V	Browse files
import pandas as pd	D	outp	ut.csv 4	.3KB		×
import numpy as np		id	x	у	g	
file = st.file uploader("Choose a CSV file",	0	0	2.0585	2.2915	A	
	1	1	1.084	2.7116	А	
type='csv')	2	2	0.3268	0.3804	В	
	3	3	0.1504	0.2015	В	
if file is not None:	4	4	1.0371	2.1221	Α	
<pre>df = pd.read_csv(file)</pre>						
st.write(df)						



## 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])
```

Choose a	CSV file										
夺	Drag Limit	and dro 200MB pe	<b>p file her</b> er file • CS\	e /					Br	owse f	iles
D	outp	ut.csv 4	.3KB								×
	id	х	у	g							
0	0	2.0585	2.2915	Α							
1	1	1.084	2.7116	Α							
2	2	0.3268	0.3804	В							
3	3	0.1504	0.2015	В							
4	4	1.0371	2.1221	Α							
Variable x 3.0 2.5 2.0 1.5 1.0 0.5 0.0 0			20 25			5	70	75			



## 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)
```

```
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)
```

Choose a	CSV file														
Ð	Drag Limit	and dro 200MB pe	<b>p file he</b> er file • CS	v V									Brows	e files	
D	outp	ut.csv 4	.3KB											>	<
	id	х	у	g											
0	0	2.0585	2.2915	Α											
1	1	1.084	2.7116	А											
2	2	0.3268	0.3804	В											
3	3	0.1504	0.2015	В											
4	4	1.0371	2.1221	Α											
'ary y															~
3.0 2.5	•		•	:.		•				•	•••	•	••		
2.0			••			•		•	•	• •		:*		•	
∽ 1.5				•				•			•		•	••	
1.0		•	•	•		•	• •	•	•				•	•	
0.5	•		•			••			•		•		•		•
0.0	0.0 (	0.2 0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.



×\_\_\_\_

- ・ 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)
```

# Sample App. Add X = 1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…





- 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
```







- 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) <u>https://qiita.com/NewYorki/items/176c8861987d8d0b2794</u>

## 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])
```







46

## session\_state: 数値変数のグラフ(改善例)

```
hoose a CSV fil
app.py
 Drag and drop file here
import streamlit as st
 Browse files
 Limit 200MB net file + CSV
import pandas
 as pd
import numpy as np
 output.csv 4.3KB
 X
if "is button" not in st.session state:
 o 2.0585 2.2915 A ボタンをクリックすると
 st.session state["is button"] = False
 1 1.084 2.7116 A
 selectbox に変数情報が
 2 0.3268 0.3804 B
file = st.file uploader("Choose a CSV file",
 3 0.154 0.20 0 入り、変数を変更しても
 type='csv')
 * 12 2.1221 * 不具合は起きない
if file is not None:
 Plot
 df = pd.read csv(file)
 st.write(df.head())
 3.0
if st.button('Plot') or st.session state["is button"]:
 st.session state["is button"] = True
 2.5
 vars = [c for c in df.columns]
 2.0
 column = st.selectbox("Variable", vars)
 1.5
 numeric vars = df.select dtypes("number").columns
 1.0
 if column in numeric vars:
 0.5
 st.line chart(df[column])
```



×\_\_ 7

- ・ 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)

<b># app.py</b> import streamlit as st	Sidebar Title	Sample App.
<pre>st.title('Sample App.')</pre>	Button 1	
<pre>st.sidebar.title("Sidebar Title")</pre>		
<pre>st.sidebar.write('sample text 1')</pre>		
<pre>if st.sidebar.button(label='Button 1'):</pre>	Example:	
<pre>st.sidebar.write('Clicked!')</pre>	Button 2	
with st.sidebar:	Clicked!	
<pre>st.divider()</pre>		
<pre>st.subheader('Example:')</pre>		
<pre>if st.button(label='Button 2'):</pre>		
<pre>st.write('Clicked!')</pre>		

レイアウト: column



#### Deploy # app.py import streamlit as st Example: Sample App. Button 2 # col1, col2 = st.columns(2) # equal witdth Button 1 col1, col2 = st.columns([2, 1])Clicked! with coll: 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])
```





#### # 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!')
```

Sample App.	
Button 1	
See More	^
Example:	
Button 2	
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.")



52

レイアウト: 複数ページ





C:¥temp¥<u>main.py</u> ← 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¥<u>sub.py</u>

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 <u>https://docs.streamlit.io/develop/api-</u> <u>reference/configuration/st.set\_page\_config</u>
- テーマの変更

https://docs.streamlit.io/develop/concepts/configuration/theming



×\_\_\_\_

- ・ 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: <u>https://f5lppq2xsk877wluqd3gj3.streamlit.app/</u>



# Upload a CSV and Analyze Data Chose a CSV file Image and drop file here Limit 200MB per file - CSV Browse files Image and drop file here Limit 200MB per file - CSV Browse files Image and drop file here Limit 200MB per file - CSV Browse files Image and drop file here Limit 200MB per file - CSV Source file Image and drop file here Limit 200MB per file - CSV Source file Image and drop file here Limit 200MB per file - CSV Source file Image and drop file here Limit 200MB per file - CSV Source file Image and drop file here Limit 200MB per file - CSV Source file Image and drop file here Limit 200MB per file - CSV X DataFrame Preview: X Image and drop file here Limit 200MB per file - CSV X Image and drop file here Limit 200MB per file - CSV X DataFrame Preview: X Image and drop file here Limit 200MB per file - CSV X Image and drop file here Limit 200MB per file - CSV X Image and drop file here Limit 200MB per file - CSV X Image and drop file

y	
g	
Gro	uped
um	mar
	у
ount:	100
nean	1.4894
itd	0.8811
nin	0.0222
	0.7863
25%	
50%	1.4268
25% 30% 15%	1.4268 2.2491

#### Histogram







https://streamly.streamlit.app/

streamlit 用生成 AI がプログラム作成をサポート!





- Streamlit documentation <u>https://docs.streamlit.io/</u>
- Streamlit API cheat sheet

https://docs.streamlit.io/develop/quick-reference/cheat-sheet

- <u>Streamlit Magic Cheat Sheets</u>
   <u>https://cheat-sheets.streamlit.app/</u>
- 覚えておきたい Streamlit での変数取り扱い(session\_stete の注意点)
   <a href="https://giita.com/NewYorki/items/176c8861987d8d0b2794">https://giita.com/NewYorki/items/176c8861987d8d0b2794</a>
- 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」
 <a href="https://nfunao.web.fc2.com/">https://nfunao.web.fc2.com/</a>
 <a href="https://nfunao.web.fc2.com/files/python\_intro2.zip">https://nfunao.web.fc2.com/files/python\_intro2.zip</a>

#### python powered

#### – End of File –

