# Chartbook_Example

April 4, 2025

[1]:
```
# Import libraries
import pandas as pd
import subprocess
```

## 0.1 Chartbook Minimal Example - US Wage Growth

Three parts:

1) Gather raw data, apply functions, store cleaned data;
2) Generate descriptive text from cleaned data; and
3) Compile LaTeX file to update pdf file (or other format) with new data and text.

Tools: Python, R, Stata or similar for parts (1) and (2). LaTeX (pdf) or web-based for part (3). In theory, AI (Gemini) could do part (2) as well.

This example is written in Python. If you are new to Python, I suggest using 'miniconda'.

The example shows a way to download wage data from FRED, apply calculations, and save the results. Next, these results are entered into a set of functions and script to generate two sentences of descriptive text. This text is saved locally as a .txt file. Finally, a separate LaTex (.tex) file is used as a template. The .tex file has a "hard-keyed" portion, a placeholder for the .txt file, and a placeholder chart waiting to read the .csv wage growth data. When the .tex file is compiled, it generates/updates a PDF file.

**Step 1: Gather data, apply functions, store cleaned data** Download wage data from FRED and calculate the three-month over three-month growth rate. Save the results locally as a .csv file.

[2]:
```
# Download AHETPI series from FRED
url = 'https://fred.stlouisfed.org/data/AHETPI'
df = pd.read_html(url, parse_dates=True)[1].set_index('DATE')
```

[3]:
```
# Show last 5 observations
df.tail()
```

[3]:

| DATE | VALUE |
|---|---|
| 2024-11-01 | 30.58 |
| 2024-12-01 | 30.67 |
| 2025-01-01 | 30.80 |

```
2025-02-01  30.91
2025-03-01  30.96
```

[4]:
```
# Calculate 3-month moving average
df['MA'] = df['VALUE'].rolling(3).mean()
```

[5]:
```
df.tail()
```

[5]:
```
            VALUE         MA
DATE
2024-11-01  30.58  30.483333
2024-12-01  30.67  30.580000
2025-01-01  30.80  30.683333
2025-02-01  30.91  30.793333
2025-03-01  30.96  30.890000
```

[6]:
```
# Calculate 3M/3M change
df['CH3M'] = (((df['MA'] / df['MA'].shift(3))**4) - 1) * 100
```

[7]:
```
df.tail()
```

[7]:
```
            VALUE         MA      CH3M
DATE
2024-11-01  30.58  30.483333  4.219403
2024-12-01  30.67  30.580000  4.113957
2025-01-01  30.80  30.683333  4.054071
2025-02-01  30.91  30.793333  4.130270
2025-03-01  30.96  30.890000  4.117015
```

[8]:
```
# Save result to csv
df.loc['2017':].to_csv('wages.csv')
```

[ ]:
```

```

[ ]:
```

```

**Step 2: Generate text** Two sentences: the curent monthly and 3-month average values; the 3M/3M growth rate.

[9]:
```
# Open csv file
data = pd.read_csv('wages.csv', parse_dates=['DATE'], index_col='DATE')
# Format latest date
ltdt = data.index[-1].strftime('%B %Y')
# Format wages with dollar sign and two decimals
lt = data.map('\\${:.2f}'.format)

# Describe the change: If small, "virtually unchanged", else "increase/decrease␣
 ↪of X percent"
```

```
chval = data['CH3M'].iloc[-1]
chdir = 'decrease' if chval < 0 else 'increase' # Direction of change
if abs(chval) <= 0.2:   # Small values
    chtxt = 'virtually no change'
else:
    chtxt = f'an annualized {chdir} of {abs(chval):.1f} percent'

text = (f'As of {ltdt}, the earnings of production and non-supervisory '+
        f'workers average {lt.VALUE.iloc[-1]} per hour. Over the past '+
        f'three months, earnings average {lt.MA.iloc[-1]} per hour, '+
        f'{chtxt} from the previous three months.')
with open('wages.txt', 'w') as text_file:
        text_file.write(text)
print(text)
```

```
As of March 2025, the earnings of production and non-supervisory workers average
\$30.96 per hour. Over the past three months, earnings average \$30.89 per hour,
an annualized increase of 4.1 percent from the previous three months.
```

[ ]:

[ ]:

**Step 3: Update template** This block of code will compile the latex file chartbook_example.tex with the latest data and text generated above. If the result is successful, the process will return code "0".

[10]:
```
# Compile latex template with latest data to generate pdf file
runcb = subprocess.run(['latex', 'chartbook_example.tex'], stdout=subprocess.
 ↪DEVNULL)
if runcb.returncode == 0:
    print('Success! PDF file generated.')
else:
    print(runcb.returncode)
```

```
Success! PDF file generated.
```

[ ]:

**Scaling up the process** To scale up this process, I've written a few custom functions. For example value_text() will generate text from a single numerical value, and has lots of options to handle different use cases.

[11]:
```
from utils import value_text # Custom function to handle economic statistics
```

[12]:
```
# Apply custom function to wage growth data
value_text(chval, 'increase_of', adj='annualized')
```

[12]:
```
'an annualized increase of 4.1 percent'
```

[13]:
```
# Basic output
value_text(chval)
```

[13]:
```
'increased 4.1 percent'
```

[14]:
```
# Negative values
value_text(-8.27748)
```

[14]:
```
'decreased 8.3 percent'
```

[15]:
```
# Options for text
value_text(87.7628, 'contribution')
```

[15]:
```
'contributed 87.8 percent'
```

[16]:
```
# More casual text
value_text(73.311, 'contribution', casual=True)
```

[16]:
```
'added 73.3 percent'
```

[17]:
```
# Format as currency
value_text(6282.84346, 'plain', ptype=None, dollar=True, digits=2)
```

[17]:
```
'\\$6,282.84'
```

[ ]:
```

```