HKJournalist

Mar 30, 2023

Home:

- Slides for sharing: Download Link

It is a light and useful Python module, helping you generate a small size, pretty report as PDF slides (or any other format documents which human can directly read and hand out) each time after your programs finish. All you need to do is to customize a report template using `MarkDown` with variables name which used in your Python program, and maintain a `dict` to store those variables. Then, A few lines of code added before end of programs can automatically fetch and display them in final report file. Also, code deal with frequent structure/arguments changes or data source changes can benefit from the package if the report can play a role of 'snapshot' (with timestamp) of each code version.

CHAPTER 1

Quick Start

1.1 Pre-requirements

Before installing `hkjournalist`, please make sure `pandoc` and `pdflatex` are already properly installed in the environment.

- `pandoc`: pandoc.org/installing.html
- `texlive`/`mactex`(for MacOS): www.tug.org/texlive/ http://www.tug.org/mactex/

1.2 Install

```
pip install hkjournalist
```

1.3 Customize your report template

Write such a `md` file, use a pair of `{}` to wrap every variable which will be assigned specified value in your code. save it to `template.md`

```
% Hello World
% Xinyi Li
% 2019-12-19

---

### sine plot

![]({sin_plot})
```

(continues on next page)

(continued from previous page)

```
### sine table

{sin_table}

### sine function

```{{.python}}
{sin_func}
```
```

1.4 Run a `Journalist()` in your code to fetch variables

First, you should define a `dict` to record mapping with variable names and their value

```
from hkjournalist import Journalist

config = {}
```

Then, start your programming, and do not forget to assign value to corresponding variable names in `config`:

```
def sin_2x_and_cos_2x(x):
    y = np.sin(x) * np.sin(x) + np.cos(x) *  np.cos(x)
    return y

x = np.arange(0, 4 * np.pi, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)

df = pd.DataFrame({'x': x, 'sin(x)': y1, 'cos(x)': y2})
df['sin^2^(x)+cos^2^(x)'] = sin_2x_and_cos_2x(df['x']).values
df = df.set_index('x')

# plot sine curve as sin_plot
ax = df.plot()
plt.tight_layout()
config['sin_plot'] = ax

# random select 5 point (x,y) as sin_table
config['sin_table'] = df.sample(5)

# sin_2x_and_cons_2x as sin_func
config['sin_func'] = sin_2x_and_cos_2x
```

1.5 Invite a journalist to make a big news report

Last but not least, attach 3 lines **critical** code below to have your `Journalist` make a report and save it to `big_news.pdf` (you can get all code above in demo and the output file big_news.pdf)

```
# HK journalist runs faster than everyone! hear variable and make a report
reporter = Journalist(template_file='template.md')
```

(continues on next page)

(continued from previous page)

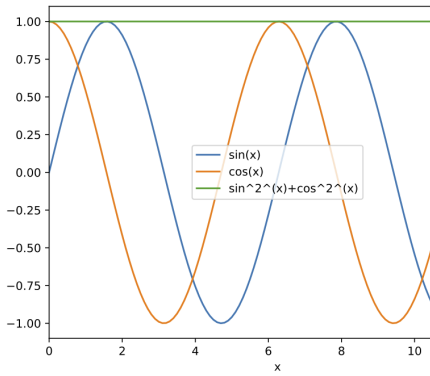
```
reporter.hear(config)
reporter.report(output_file='big_news.pdf', beamer=True, overwrite=True)
```

Hello World

Xinyi Li

2019-12-19

Sine Plot



Sine Table

x	sin(x)	cos(x)	$\sin^2(x)+\cos^2(x)$
6.6	0.31	0.95	1
5.6	-0.63	0.78	1
7.3	0.85	0.53	1
5.9	-0.37	0.93	1
4.7	-1	-0.01	1

Sine Function

```
1 def sin_2x_and_cos_2x(x):
2     y = np.sin(x) * np.sin(x) + np.cos(x) *
          np.cos(x)
3     return y
```

Report slides display as below:

1.6 Final question: What will my variables on slides look like?

All variables pass to `Journalist` via `hear` will display as strings just like what their `__str__` method do.

Except for 4 types with special support:

- `pandas.DataFrame`: -> a 3-line table (`TeX` default style)
- `matplotlib.axes.SubplotBase` (known as base figure object `ax` in `matplotlib`): -> a figure print on report (with high quality and small size as `pdf`)
- `function`: -> its full definition
- `list(str)`: -> `len(list)` followed by a sentance with all words concatenated.

CHAPTER 2

Auto Generate `md` Template

2.1 I am too lazy to write a `md` template

If you have too many variables to report, which make write a template a big project, or simply don't want to write a `md` template, **No problem!** `hkjournalist` can generate a report template with each variable on one slide page automatically. With slight modification or directly using it as a template, you can get your real report.

2.2 Example

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from hkjournalist import Journalist

config = {}

for i in range(1, 4):
    uniform_data = np.random.rand(10, 12)
    plt.figure(figsize=(10, 8))
    ax = sns.heatmap(uniform_data, cmap='Blues', annot=True, linewidth=.5)
    plt.tight_layout()
    config[f'Plot_{i}'] = ax

reporter = Journalist(fig_height='80%')
reporter.hear(config)
reporter.generate_template('auto_generate_template.md')
reporter.report(output_file='auto_report.pdf', beamer=True, overwrite=True)
```

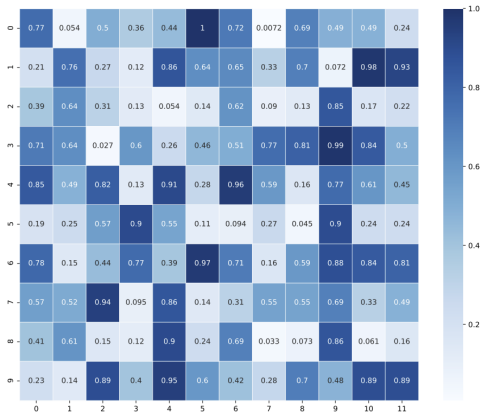
Output (raw file):

Report template

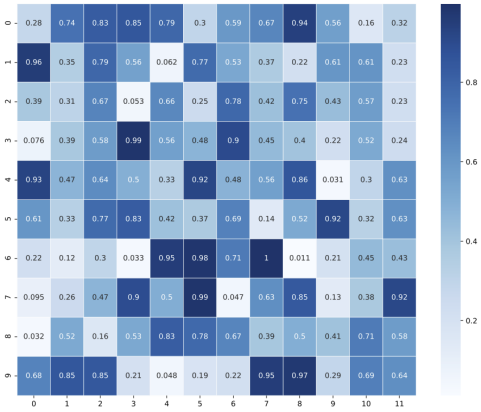
Author

2019-12-19

Plot_1



Plot_2



Plot_3

CHAPTER 3

Tutorial 1: Run a prophet model on time series

- Code and output: example/1_prophet_model_evalute.py
- Data and model usage from `fbprophet` tutorials

3.1 Prepare a `dict` to store variables

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from hkjournalist import Journalist
from fbprophet import Prophet

sns.set(style="darkgrid")

# maintain a dict to record all variables used in reports
config = {}
```

3.2 Load data

```
df = pd.read_csv('./data/example_wp_log_peyton_manning.csv')
df['ds'] = pd.to_datetime(df['ds'])
```

3.3 Plot data

```
ax = df.set_index('ds').plot(figsize=(20, 10))
plt.tight_layout()
config['data_plot'] = ax
```

3.4 Split train and test set

```
train_end_date = '20131231'
config['train_end_date'] = train_end_date
train_df = df[df['ds'] <= train_end_date]
test_df = df[df['ds'] > train_end_date]

test_df['year'] = test_df['ds'].dt.year
test_df['month'] = test_df['ds'].dt.month
```

3.5 Build a model

```
model = Prophet(weekly_seasonality=True, yearly_seasonality=True)
model.add_seasonality('monthly', period=30.5, fourier_order=12, prior_scale=10)
model.add_seasonality('quarter', period=364.5 / 4, fourier_order=10, prior_scale=5)
```

Seasonalities are import hyper-parameters of a `Prophet` model, which should be recorded every time.

3.6 Fit this model and plot the prediction

The final plot should also be reported.

```
model.fit(train_df)
config['seasonality'] = pd.DataFrame(model.seasonalities)
test_df['y_pred'] = model.predict(test_df[['ds']])['yhat'].values
ax = test_df[['ds', 'y', 'y_pred']].set_index('ds').plot(figsize=(20, 10))  # plot␣
↪predict result
plt.tight_layout()
config['pred_plot'] = ax
```

3.7 Define a custom metric to evaluate model

And the metric definition should be exposed to others

```
def kpi_mape(df, y_true, y_pred):
    # mape group by yearmonth
    df[y_pred] = df[y_pred].clip(0, None)
    df['diff'] = abs(df[y_true] - df[y_pred])
    mape_df = df.groupby(['year', 'month']).agg({'diff': 'sum', y_true: 'sum'}).reset_
↪index()
    mape_df['mape'] = mape_df['diff'] / mape_df[y_true]
    res_df = mape_df.pivot(index='month', columns='year', values='mape')
    return res_df

config['metric_func'] = kpi_mape
```

3.8 Detail data for error analysis

```
kpi_df = kpi_mape(test_df, 'y', 'y_pred')
plt.figure(figsize=(4, 6))
ax = sns.heatmap(kpi_df, annot=True, cmap='YlGn', linewidth=.5, fmt='.2f')
plt.tight_layout()
config['error_plot'] = ax
```

3.9 Add some personal note to describe your model

```
config['note'] = "Prophet with no holidays info"
```

3.10 Finally, here comes the `Journalist`!

```
report_journalist = Journalist(template_file='./reports/1_prophet_report_template.md')
report_journalist.hear(config)
report_journalist.report(output_file='./reports/1_prophet_report.pdf', beamer=True,␣
↪overwrite=False)
```

3.11 Output slides

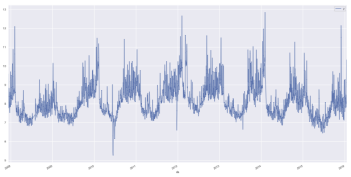
raw file: examples/reports/1_prophet_report_2019-12-18_22:06:18.pdf

Simple Model Test

Author

2019-12-18

Data Plot

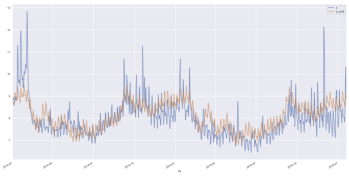


Train set

▶ **train end date** 20131231
▶ **seasonality**

	monthly	quarter	yearly	weekly
condition_name				
fourier_order	12	10	10	3
mode	additive	additive	additive	additive
period	30.5	91.125	365.25	7
prior_scale	10.0	5.0	10.0	10.0

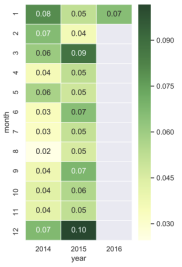
Prediction Result



Error analysis

```
def kpi_mape(df, y_true, y_pred):
    # mape group by yearmonth
    df[y_pred] = df[y_pred].clip(0, None)
    df['diff'] = abs(df[y_true] - df[y_pred])
    mape_df = df.groupby(['year',
        'month']).agg({'diff': 'sum', y_true:
        'sum'}).reset_index()
    mape_df['mape'] = mape_df['diff'] /
        mape_df[y_true]
    res_df = mape_df.pivot(index='month',
        columns='year', values='mape')
    return res_df
```

Error analysis (plot)



Note

Prophet with no holidays info

CHAPTER 4

Tutorial 2: EDA and select feature

- Code and output: example/2_feature_selection.py
- Data from Kaggle Contest

4.1 Load Data and pass basic statistics data to `config`

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from hkjournalist import Journalist

config = {}

df = pd.read_csv('./data/train.csv')
df = df.iloc[:, 1:]
feature_df = df.loc[:, :'Horizontal_Distance_To_Fire_Points']

# basic statistics feature
config['Cover_type'] = df.groupby('Cover_Type').size()
config['shape'] = df.shape
config['dtypes'] = feature_df.dtypes
config['skew'] = feature_df.skew()
config['description'] = feature_df.describe()
```

4.2 Plot the correlation matrix

```
data_corr = feature_df.corr()
plt.figure(figsize=(8, 6))
```

(continues on next page)

(continued from previous page)

```
grid = sns.heatmap(data_corr, annot=True, fmt='.2f', cmap='RdBu', center=0,␣
↪linewidth=.5)
plt.tight_layout()
config['Corr'] = grid
```

4.3 Filter features with high correlation coefficient

```
def drop_by_cor(feature_df, threshold=0.9):
    corr_matrix = feature_df.corr().abs()
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.
↪bool))
    to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
    return to_drop


config['select_method'] = drop_by_cor
threshold = 0.7

valid_features = list(set(feature_df.columns) - set(drop_by_cor(feature_df,␣
↪threshold)))
config[f'features_remain_with_thres_as_{int(threshold * 100)}'] = valid_features
```

Too many variables to write a template by hand? don't worry, just call `generate_template` method!

4.4 HKJournalist can use auto-generated template before making a big report!

```
report_journalist = Journalist()
report_journalist.hear(config)
report_journalist.generate_template('./reports/2_feature_select_template.md')
report_journalist.report('./reports/2_feature_select.pdf', overwrite=True,␣
↪beamer=True, aspectratio=169)
```

4.5 Output

raw file: ./reports/2_feature_select.pdf

Report template

Author

2019-12-19

Cover_type

Cover_Type
1 2160
2 2160
3 2160
4 2160
5 2160
6 2160
7 2160
dtype: int64

shape

(15120, 55)

dtypes

Elevation int64
Aspect int64
Slope int64
Horizontal_Distance_To_Hydrology int64
Vertical_Distance_To_Hydrology int64
Horizontal_Distance_To_Roadways int64
Hillshade_9am int64
Hillshade_Noon int64
Hillshade_3pm int64
Horizontal_Distance_To_Fire_Points int64
dtype: object

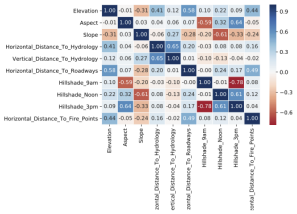
skew

Elevation 0.075640
Aspect 0.450935
Slope 0.523658
Horizontal_Distance_To_Hydrology 1.488052
Vertical_Distance_To_Hydrology 1.537776
Horizontal_Distance_To_Roadways 1.247811
Hillshade_9am -1.093681
Hillshade_Noon -0.953232
Hillshade_3pm -0.340827
Horizontal_Distance_To_Fire_Points 1.617099
dtype: float64

description

	Elevation	Aspect	Slope	Horizontal_...	Vertical_...	...	...	...	...	...	...
count	15120	15120	15120	15120	15120	15120	15120	15120	15120	15120	15120
mean	2749.32	156.68	16.5	227.2	51.08	1714.02	212.7	218.97	135.09	1511.15	
std	417.68	110.08	8.45	210.08	61.24	1325.07	30.56	22.8	45.9	1099.94	
min	1863	0	0	0	-146	0	0	99	0	0	
25%	2376	65	10	67	5	764	196	207	106	730	
50%	2752	126	15	180	32	1316	220	223	138	1256	
75%	3104	261	22	330	79	2270	235	235	167	1988.25	
max	3849	360	52	1343	554	6890	254	254	248	6993	

Corr



select_method

```
def drop_by_cor(feature_df, threshold=0.9):
    corr_matrix = feature_df.corr().abs()
    upper =
        corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
        k=1).astype(np.bool))
    to_drop = [column for column in upper.columns if
        any(upper[column] > threshold)]
    return to_drop
```

features_remain_with_thres_as_70

9 Hillshade_9am, Aspect, Horizontal_Distance_To_Roadways, Vertical_Distance_To_Hydrology, Horizontal_Distance_To_Hydrology, Hillshade_Noon, Elevation, Slope, Horizontal_Distance_To_Fire_Points

CHAPTER 5

Write templates with Chinese characters

(new feature from 0.0.5)

Now, `HKJournalist` can generate reports from templates that contain Chinese Characters. In other words, we can write reports using `zh_CN`.

5.1 Pre-requirements

First of all, you should check if XeLaTeX is installed properly in your environment. It is always integrated in TeXLive /MacTeX.

Then, make sure you have already installed target Chinese fonts.

5.2 Specify the header in `md` template

In the YAML metadata block, you should specify the CJK font used by the report. To simplify, For slides, we just specify `documnetclass` in which the font is already defined: *(it is what exactly `HKJournalist` do when asked to automatically give a chinese report template)*

```
---
documentclass: ctexbeamer
title:
author:
date: \today{{}}
---
```

Or you can replace the first line with a font declaration, such as:

```
# for windows
CJKmainfont: KaiTi
# Or for Linux
mainfont: WenQuanYi Micro Hei Mono
```

Note: the line about font or `documentclass` should be placed at first in the block.

Note: `\today{}` is a command that will be replaced by the actual date in proper format in the final report. To escape `{}`, it should be as `\today{{}}` in a template.

5.3 Write main part of your program

In general, there is nothing noteworthy in this part. Codes used in this example is exactly the same as the one in Quick Start.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from hkjournalist import Journalist

config = {}


def sin_2x_and_cos_2x(x):
    y = np.sin(x) * np.sin(x) + np.cos(x) * np.cos(x)
    return y


x = np.arange(0, 4 * np.pi, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)

df = pd.DataFrame({'x': x, 'sin(x)': y1, 'cos(x)': y2})
df['sin^2^(x)+cos^2^(x)'] = sin_2x_and_cos_2x(df['x']).values
df = df.set_index('x')

# plot sine curve as sin_plot
ax = df.plot()
plt.tight_layout()
config['sin_plot'] = ax

# random select 5 point (x,y) as sin_table
config['sin_table'] = df.sample(5)

config['sin_func'] = sin_2x_and_cos_2x
```

However, you should always be careful on the encoding of your code/text files, especially when chinese characters probably occur in strings. Please ensure your files are always read and written under `UTF-8` encoding.

And the rest part of `md` template is:

```
###

![]({sin_plot})

###

{sin_table}

###
```

(continues on next page)

(continued from previous page)

```
```{{.python}}
{sin_func}
```

5.4 Turn on `zh`

When constructing a `HKJournalist` instance, Turn on its `zh-CN` support by using `zh=True` argument.

```
reporter = Journalist(template_file='./reports/3_zh_cn_template.md', zh=True)
```

It is not too late to set `reporter.zh=True` by hand before call its `report()` method, of course, if you rely on automatically generated template with `generate_template()`, its `zh` property must be determined more early.

5.5 Final report

The following steps are as usual:

```
reporter.hear(config)
reporter.report(output_file='./reports/3_zh_report.pdf', beamer=True, overwrite=True)
```

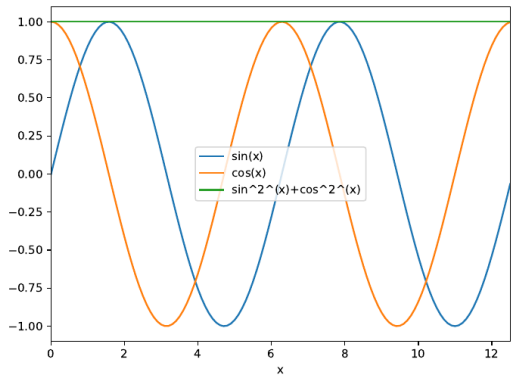
The output file 3_zh_report.pdf looks like:
```

中文报告

香港记者

2020 年 5 月 3 日

正弦曲线



正弦取值

x	sin(x)	cos(x)	$\sin^2(x)+\cos^2(x)$
1.3	0.96	0.27	1
5.9	-0.37	0.93	1
3.3	-0.16	-0.99	1
11.8	-0.69	0.72	1
2.4	0.68	-0.74	1

正弦函数

```
def sin_2x_and_cos_2x(x):
    y = np.sin(x) * np.sin(x) + np.cos(x) *  np.cos(x)
    return y
```

CHAPTER 6

hkjournalist package

6.1 Submodules

6.2 hkjournalist.journalist module

class hkjournalist.journalist.**Journalist**(*template_file=None*, *fig_width=None*, *fig_height=None*, *tmp_path='./temp'*, *zh=False*)

Bases: object

Class to record and generate reports

Parameters

- **template_file** (*str*) – file path of md template
- **fig_width** (*None, int*) – figure width (percentage of whole page width), prior to *fig_height* settings
- **fig_height** (*None, int*) – figure height (percentage of whole page width)‘
- **tmp_path** (*str*) – temporary directory path to store temporary files (such as figures)
- **zh** (*bool*) – if it supports chinese (zh_CN) usage

generate_template(*template_file='./template.md'*, *title='template'*, *author='Author'*, *append=False*)

Generate a *md* template according to mappings which previously passed to.

The output template will be structed as each variable on a single slide with variable name as its title

Noteit may overwrite the file with the address

Parameters

- **template_file** (*str*) – output template file path
- **title** (*str*) – report title
- **author** (*str*) – author name

- **append** (*bool*) – If use append mode to add new contents of report

Returns None

Return type None

Returns

hear(*config_dict: dict*)
Pass your variables mappings to the reporter

Parameters **config_dict** (*dict*) – variable mappings such as {'var_name':value}

Returns None

Return type None

report(*output_file='./final_report.pdf'*, *beamer=True*, *theme='default'*, *color_theme='seagull'*, *use_template_config=False*, *overwrite=True*, *aspectratio=43*)
Generate final pdf (or other format) report using previously heard config dict

Parameters

- **output_file** (*str*) – final output file path
- **beamer** (*bool*) – whether the output pdf will be a beamer slides ?
- **theme** (*str*) – the theme used to create beamer (see https://hartwork.org/beamer-theme-matrix/))
- **color_theme** (*str*) – the color theme used to create beamer (see https://hartwork.org/beamer-theme-matrix/)
- **use_template_config** (*bool*) – whether use metadata params of format in your custom template, if false, just use params in this function to produce a report. otherwise, please ref to https://pandoc.org/MANUAL.html to write a fine `md` template
- **overwrite** (*bool*) – whether use a timestamp of current time as a postfix for the final output filename. if false, a new file will occur every time the method call without overwriting previouly ones.
- **aspectratio** (*int*) – aspect ratio of slide page. only valid when *beamer* is turn on and output format is *pdf*

Returns execution return code (0 if succeed)

Return type int

6.3 Module contents

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index