quarto_pandas

Using Pandas style with Quarto

In this example I'll try to use the builtin Styler object's methos of pandas for formatting pandas dataframe in Quarto. There will be some snags.

Generate a pandas dataframe

I'll generate a table of data, which will include some string, numeric and math expression:

```
import pandas as pd
import numpy as np

V_R = 50
sl = pd.Series(['SLR', 'SLC', 'SLV', 'SLD', 'SLO', 'SLID'], name='SL')
pvr = pd.Series([0.02, 0.05, 0.1, 0.63, 0.81, 0.995], name='P_V_R')

df = pd.DataFrame([sl, pvr]).transpose()
df['T_R'] = df['P_V_R'].map(lambda x: - V_R/np.log(1-x))
df['lambda_i'] = 1/df['T_R']
```

The above code results in the following textual data:

```
print(df)
```

```
    SL  P_V_R          T_R  lambda_i
0   SLR   0.02  2474.915823  0.000404
1   SLC   0.05   974.786287  0.001026
2   SLV    0.1   474.561079  0.002107
3   SLD   0.63    50.289048  0.019885
4   SLO   0.81    30.107220  0.033215
```

Table 1: df rendered without any custom formatting

	SL	P_V_R	T_R	lambda_i
0	SLR	0.02	2474.915823	0.000404
1	SLC	0.05	974.786287	0.001026
2	SLV	0.1	474.561079	0.002107
3	SLD	0.63	50.289048	0.019885
4	SLO	0.81	30.107220	0.033215
5	SLID	0.995	9.436958	0.105966

```
5  SLID  0.995     9.436958  0.105966
```

which get rendered as html table in quarto:

```
df
```

```
c:\Users\s.follador\Documents\github\quarto_pandas\.venv\lib\site-packages\IPython\core\forma
  return method()
```

Desired Output

The desired output I want to achieve is the one rapresented in Table 2, which is a manually consrtructed markdown table

Table 2: Desired output

SL	P_{V_R}	T_R	λ_i
SLR	0.02	2474.915823	0.000404
SLC	0.05	974.786287	0.001026
SLV	0.1	474.561079	0.002107
SLD	0.63	50.289048	0.019885
SLO	0.81	30.107220	0.033215
SLID	0.995	9.436958	0.105966

Format the dataframe using pandas' styler

I want to apply some different style to each columns which rapresenta different type of data:

(1) I want that the columns `P_V_R` to be formatted as `%` with 1 decimal place of precision

```
dfs = df.style.format({'P_V_R': '{:.1%}'}) # dfs is a styler object
dfs
```

Table 3: dataframe after formatting one columns

	SL	P_V_R	T_R	lambda_i
0	SLR	2.0%	2474.915823	0.000404
1	SLC	5.0%	974.786287	0.001026
2	SLV	10.0%	474.561079	0.002107
3	SLD	63.0%	50.289048	0.019885
4	SLO	81.0%	30.107220	0.033215
5	SLID	99.5%	9.436958	0.105966

The columns get correctly formatted, but **the whole table has lost the formatting**, it doesn't looks good.

Thanks to this workaround on the github I can get a nice looking table:

```
dfs.set_table_attributes('class=dataframe')
```

Table 4: class=dataframe

	SL	P_V_R	T_R	lambda_i
0	SLR	2.0%	2474.915823	0.000404
1	SLC	5.0%	974.786287	0.001026
2	SLV	10.0%	474.561079	0.002107
3	SLD	63.0%	50.289048	0.019885
4	SLO	81.0%	30.107220	0.033215
5	SLID	99.5%	9.436958	0.105966

(2) Now I want to center align the content of the cell; according to pandas documentation I have to use `set_table_styles`

```
dfs.set_table_styles(
    [{
        'selector': 'tr',
        'props': [('text-align', 'center')]
    }]
)
```

Table 5: center align failed

	SL	P_V_R	T_R	lambda_i
0	SLR	2.0%	2474.915823	0.000404
1	SLC	5.0%	974.786287	0.001026
2	SLV	10.0%	474.561079	0.002107
3	SLD	63.0%	50.289048	0.019885
4	SLO	81.0%	30.107220	0.033215
5	SLID	99.5%	9.436958	0.105966

As you can see, the content doesn't get centered, but I discovered that if change the `class` to `table` instead of `dataframe`, it works again:

```
# I can't use the same styler object (dfs) as before otherwise
# it will affect the precedent output

dfs2 = (df.style
            .set_table_attributes('class=table')
            # I reapply the precedent format
            .format({'P_V_R': '{:.1%}'})
            .set_table_styles(
                [{
                    'selector': 'tr',
                    'props': [('text-align', 'center')]
                }]
            )
)
dfs2
```

Table 6: class=table

	SL	P_V_R	T_R	lambda_i
0	SLR	2.0%	2474.915823	0.000404
1	SLC	5.0%	974.786287	0.001026
2	SLV	10.0%	474.561079	0.002107
3	SLD	63.0%	50.289048	0.019885
4	SLO	81.0%	30.107220	0.033215
5	SLID	99.5%	9.436958	0.105966

Notice that *the rows are not longer alternating colors*, but that is compatible with the output of a Markdown Table such as Table 2

```
# format the data
dfs2.format(formatter={
    'P_V_R': '{:.1%}',
    'T_R': '{:.0f}',
    'lambda_i': '{:0.3%}'
    })
#format the index
dfs2.format_index(
    formatter='${}$',
    axis=1
    )
dfs2.hide()
```

Table 7: fully formatting tabled using the styler

SL	P_V_R	T_R	$lambda_i$
SLR	2.0%	2475	0.040%
SLC	5.0%	975	0.103%
SLV	10.0%	475	0.211%
SLD	63.0%	50	1.989%
SLO	81.0%	30	3.321%
SLID	99.5%	9	10.597%

df.index