fip_0014

May 2, 2021

1 FIP 0014 thematic notebook

The goal of this notebook is to explore the impact of the upcoming sector expiration wave on April / May, especially when considering the ones with V1 proofs.

1.1 Setup and dependences

[1]:
```
%load_ext autotime

%load_ext autoreload

%autoreload 2
```

```
time: 8.38 ms (started: 2021-05-02 00:53:18 +00:00)
```

[2]:
```
# External dependences
import pandas as pd
import numpy as np
import plotly.express as px
from prophet import Prophet
import matplotlib.pyplot as plt

# Move path to parent folder
import sys
sys.path.insert(1, '../')

import plotly
plotly.offline.init_notebook_mode()
```

```
time: 814 ms (started: 2021-05-02 00:53:18 +00:00)
```

[3]:
```
# Create a connection object from a conn string
from filecoin_metrics.connection import get_connection, get_connection_string
conn_string = get_connection_string('../config/sentinel-conn-string.txt')
connection = get_connection(conn_string)
```

```
time: 293 ms (started: 2021-05-02 00:53:19 +00:00)
```

[4]:
```
QUERY = """
/* Get the last state of the sectors */
with sector_states as (
        select
        msi.*,
        max(msi.height) over (partition by msi.sector_id, msi.miner_id) as␣
 ↪max_height
        from miner_sector_infos msi
        where msi.activation_epoch > 0
        and msi.expiration_epoch > msi.height /* Get only active sectors */
        order by max_height
)
select
count(*) as sector_count,
sum(ss.initial_pledge::numeric) / 1e18 as initial_pledge_in_fil,
count(*) * 32 as network_power_in_gb,
date_trunc('DAY', to_timestamp(height_to_unix(ss.activation_epoch))) as␣
 ↪activation_date,
date_trunc('DAY', to_timestamp(height_to_unix(ss.expiration_epoch))) as␣
 ↪expiration_date
from sector_states as ss
where ss.max_height = ss.height /* get the last state of the info */
group by activation_date, expiration_date
order by activation_date, expiration_date
"""

query_df = (pd.read_sql(QUERY, connection)
              .assign(network_power_in_pib=lambda df: df.network_power_in_gb /␣
 ↪(1024 ** 2))
              .assign(initial_pledge_in_thousand_fil=lambda df: df.
 ↪initial_pledge_in_fil / 1000))
```

```
time: 11min 23s (started: 2021-05-02 00:53:20 +00:00)
```

[5]:
```
# Maximum date for V1 sectors
UPGRADE_DATE = '2020-11-25 00:00:00'

metrics = {'is_v1': lambda x:  x['activation_date'] < UPGRADE_DATE}

query_df = query_df.assign(**metrics)
```

```
time: 15.5 ms (started: 2021-05-02 01:04:43 +00:00)
```

[6]:
```
def resample_and_bar_plot(df, resample_rule, time_column, value_column, title,␣
 ↪**kwargs):
    fig_df = df.resample(resample_rule, on=time_column, label='left').sum()
    fig = px.bar(fig_df.reset_index(),
```

```
                x=time_column,
                y=value_column,
                title=title,
                **kwargs)
    return fig


def resample_and_bar_plot_relative(df, resample_rule, time_column,␣
 ↪value_column, title, **kwargs):
    fig_df = df.resample(resample_rule, on=time_column, label='left').sum()
    y = fig_df.groupby(time_column).sum()
    fig_df /= y
    fig = px.bar(fig_df.reset_index(),
                x=time_column,
                y=value_column,
                title=title,
                **kwargs)
    return fig
```

```
time: 14.4 ms (started: 2021-05-02 01:04:43 +00:00)
```

1.2 Sector Count

[7]:
```
df = query_df.copy()

print("Basic stats")
print("---")
print(f"Total sectors (#): {df.sector_count.sum()}")
print(f"Raw bytes power (PiB): {df.network_power_in_gb.sum() / (1024 ** 2) :.
 ↪3g}")
print(f"Initial pledge (FIL): {df.initial_pledge_in_fil.sum()}")
print("---")
```

```
Basic stats
---
Total sectors (#): 114944604
Raw bytes power (PiB): 3.51e+03
Initial pledge (FIL): 38894671.7302351
---
time: 16.6 ms (started: 2021-05-02 01:04:44 +00:00)
```

[8]:
```
resample_rule = '1m'
time_column = 'expiration_date'
value_column = 'sector_count'
title = 'Count of Expiring  Sectors (#)'
```

```
groups = [pd.Grouper(key='expiration_date', freq=resample_rule),
          'is_v1']

fig_df = (df.groupby(groups)
            .sum()
         )


fig = px.bar(fig_df.reset_index(),
             x=time_column,
             y=value_column,
             color='is_v1',
             title=title)
fig.show()
```

```
time: 340 ms (started: 2021-05-02 01:04:44 +00:00)
```

[9]:
```
resample_rule = '1d'
time_column = 'expiration_date'
value_column = 'sector_count'
title = 'Count of Expiring  Sectors Before 15Jun2021 (log #)'


groups = [pd.Grouper(key='expiration_date', freq=resample_rule),
          'is_v1']

fig_df = (df.query("expiration_date < '2021-06-15 00:00+00:00'")
            .groupby(groups)
            .sum()
            .reset_index()
         )


fig = px.bar(fig_df,
             x=time_column,
             y=value_column,
             facet_row=fig_df.is_v1,
             title=title,
             log_y=True)
fig.show()
```

```
time: 108 ms (started: 2021-05-02 01:04:44 +00:00)
```

[10]:
```
sector_count = df.sector_count.sum()
v1_count = df.query("is_v1 == True").sector_count.sum()
```

```
v1_6mo_count = (df.query("is_v1 == True & expiration_date < '2021-06-01 00:
↪00+00:00'")
                            .sector_count
                            .sum())

v1_12mo_count = (df.query("is_v1 == True & expiration_date < '2021-12-01 00:
↪00+00:00' & expiration_date >= '2021-06-01 00:00+00:00'")
                            .sector_count
                            .sum())


v1_18mo_count = (df.query("is_v1 == True & expiration_date < '2022-06-01 00:
↪00+00:00' & expiration_date >= '2021-12-01 00:00+00:00'")
                            .sector_count
                            .sum())

print("---")
print(f"Total sectors (#): {sector_count}")
print(f"V1 sectors (#): {v1_count}")
print(f"V1 sectors share (%): {v1_count / sector_count :.1%}")
print("---")
print(f"6mo V1 sectors share (%) of total sectors: {v1_6mo_count / sector_count␣
↪:.2%}")
print(f"12mo V1 sectors share (%) of total sectors: {v1_12mo_count /␣
↪sector_count :.2%}")
print(f"18mo V1 sectors share (%) of total sectors: {v1_18mo_count /␣
↪sector_count :.2%}")
print("---")
print(f"6mo V1 sectors share (%) of total sectors: {v1_6mo_count / v1_count :.
↪2%}")
print(f"12mo V1 sectors share (%) of total sectors: {v1_12mo_count / v1_count :.
↪2%}")
print(f"18mo V1 sectors share (%) of total sectors: {v1_18mo_count / v1_count :.
↪2%}")
print("---")
```

```
---
Total sectors (#): 114944604
V1 sectors (#): 26189790
V1 sectors share (%): 22.8%
---
6mo V1 sectors share (%) of total sectors: 5.08%
12mo V1 sectors share (%) of total sectors: 1.09%
18mo V1 sectors share (%) of total sectors: 16.62%
---
6mo V1 sectors share (%) of total sectors: 22.28%
12mo V1 sectors share (%) of total sectors: 4.80%
```

```
18mo V1 sectors share (%) of total sectors: 72.92%
---
time: 35.8 ms (started: 2021-05-02 01:04:44 +00:00)
```

[11]:
```
resample_rule = '1m'
time_column = 'expiration_date'
value_column = 'sector_count'
title = 'Upcoming Sector Expiration Count, grouped by sector version (#)'


groups = [pd.Grouper(key='expiration_date', freq=resample_rule),
          'is_v1']

fig_df = (df.groupby(groups)
            .sum()
            .reset_index()
         )

fig = px.bar(fig_df.reset_index(),
             x=time_column,
             y=value_column,
             facet_row='is_v1',
             title=title)
fig.show()
```

```
time: 98.9 ms (started: 2021-05-02 01:04:44 +00:00)
```

[12]:
```
resample_rule = '1w'
time_column = 'activation_date'
value_column = 'sector_count'
title = 'Activated Sector Count, grouped by sector version (#)'


groups = [pd.Grouper(key='activation_date', freq=resample_rule),
          'is_v1']

fig_df = (df.groupby(groups)
            .sum()
            .reset_index()
         )

fig = px.bar(fig_df.reset_index(),
             x=time_column,
             y=value_column,
             facet_row='is_v1',
             title=title)
fig.show()
```

time: 93.7 ms (started: 2021-05-02 01:04:44 +00:00)

1.3 Forecasts

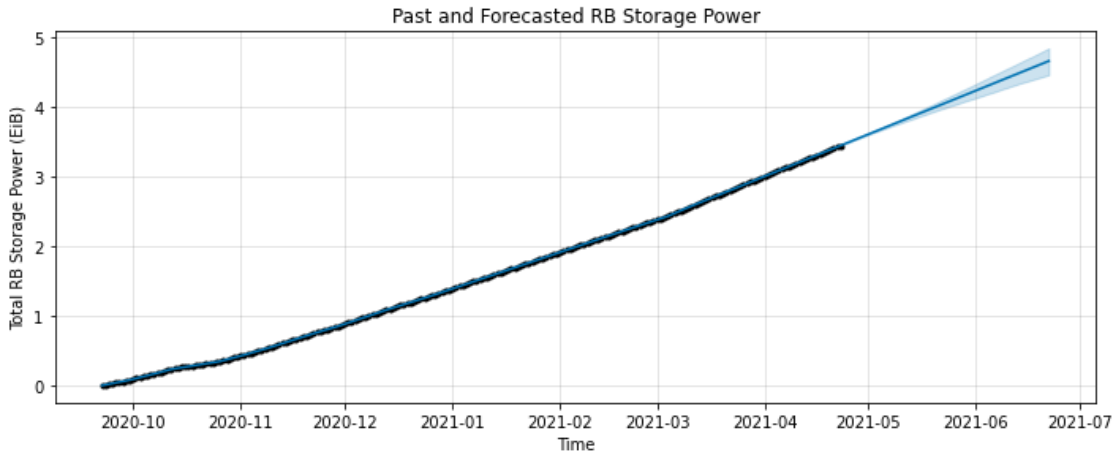
[13]:
```
rename_cols = {'activation_date': 'ds',
               'sector_count': 'y'}

proj_df = (df.resample("1d", on="activation_date")
             .sector_count
             .sum()
             .reset_index()
             .rename(columns=rename_cols)
             .assign(ds=lambda df: df.ds.dt.tz_localize(None))
             .assign(y=lambda df: df.y.cumsum() / (32 * 1024 * 1024)))

m = Prophet(changepoint_prior_scale=0.4)
m.fit(proj_df)
future = m.make_future_dataframe(periods=60)
forecast = m.predict(future)

fig = m.plot(forecast, figsize=(10, 4))
plt.title('Past and Forecasted RB Storage Power')
plt.xlabel("Time")
plt.ylabel("Total RB Storage Power (EiB)")
plt.show()
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```



time: 3.26 s (started: 2021-05-02 01:04:44 +00:00)

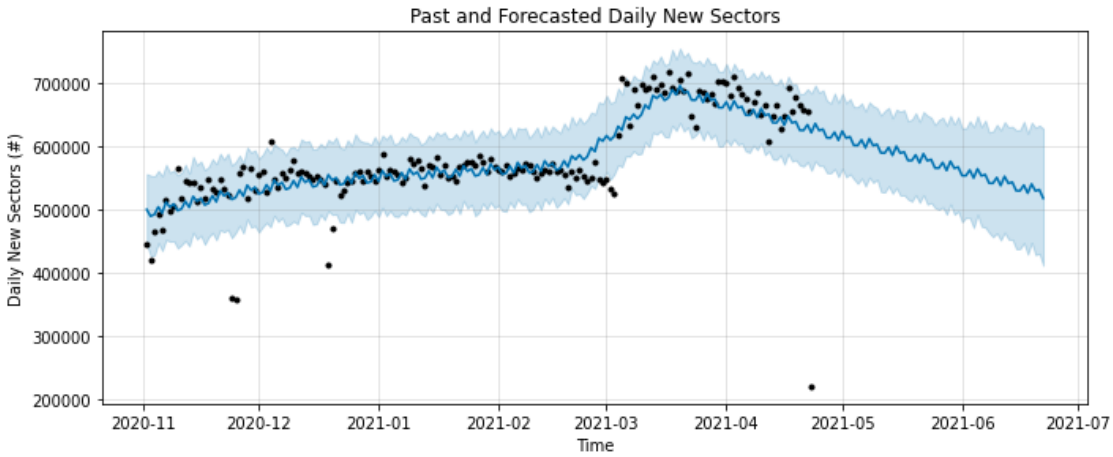
[14]:
```
rename_cols = {'activation_date': 'ds',
               'sector_count': 'y'}

proj_df = (df.query("activation_date > '2020-11-01 00:00+00:00'")
             .resample("1d", on="activation_date")
             .sector_count
             .sum()
             .reset_index()
             .rename(columns=rename_cols)
             .assign(ds=lambda df: df.ds.dt.tz_localize(None)))

m = Prophet(changepoint_prior_scale=0.2)
m.fit(proj_df)
future = m.make_future_dataframe(periods=60)
forecast = m.predict(future)

fig = m.plot(forecast, figsize=(10, 4))
plt.title('Past and Forecasted Daily New Sectors')
plt.xlabel("Time")
plt.ylabel("Daily New Sectors (#)")
plt.show()
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```




```
time: 1.95 s (started: 2021-05-02 01:04:48 +00:00)
```

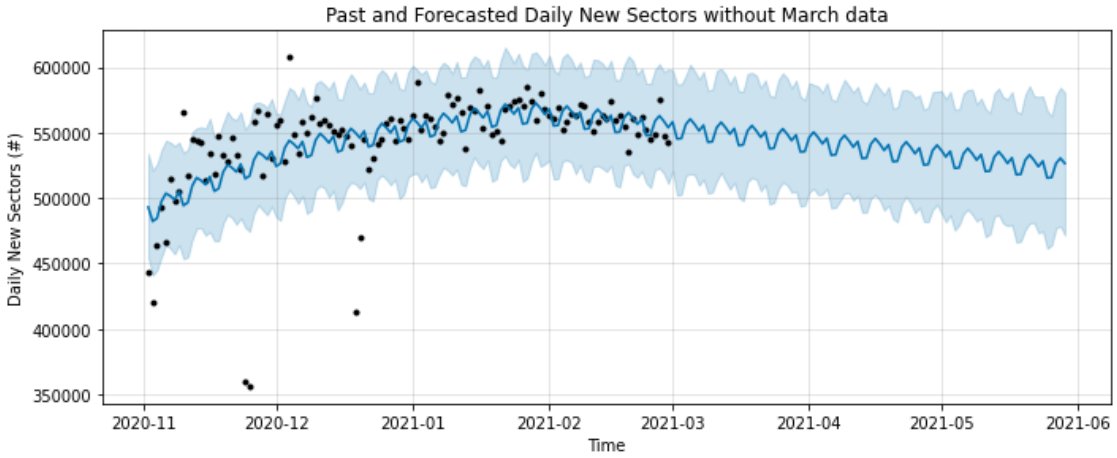
[15]:
```
rename_cols = {'activation_date': 'ds',
               'sector_count': 'y'}
```

```
proj_df = (df.query("activation_date > '2020-11-01 00:00+00:00' &␣
 ↪activation_date < '2021-03-01 00:00+00:00'")
             .resample("1d", on="activation_date")
             .sector_count
             .sum()
             .reset_index()
             .rename(columns=rename_cols)
             .assign(ds=lambda df: df.ds.dt.tz_localize(None)))

m = Prophet(changepoint_prior_scale=0.2)
m.fit(proj_df)
future = m.make_future_dataframe(periods=90)
forecast = m.predict(future)

fig = m.plot(forecast, figsize=(10, 4))
plt.title('Past and Forecasted Daily New Sectors without March data')
plt.xlabel("Time")
plt.ylabel("Daily New Sectors (#)")
plt.show()
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```




```
time: 2.11 s (started: 2021-05-02 01:04:49 +00:00)
```

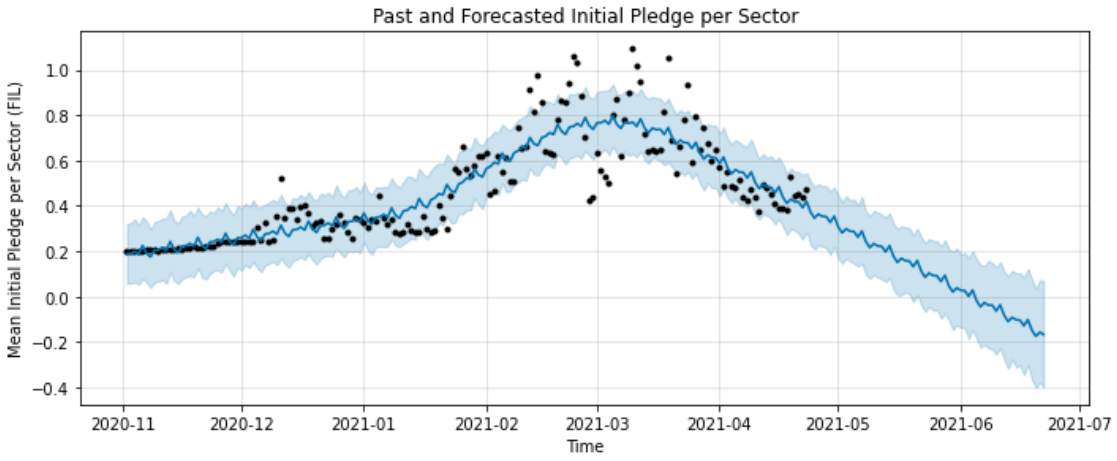
[16]:
```
rename_cols = {'activation_date': 'ds',
               0: 'y'}
```

```
proj_df = (df.query("activation_date > '2020-11-01 00:00+00:00'")
              .resample("1d", on="activation_date")
              .apply(lambda x: (x.initial_pledge_in_fil / x.sector_count).mean())
              .reset_index()
              .rename(columns=rename_cols)
              .assign(ds=lambda df: df.ds.dt.tz_localize(None))
          )

m = Prophet(changepoint_prior_scale=0.2)
m.fit(proj_df)
future = m.make_future_dataframe(periods=60)
forecast = m.predict(future)

fig = m.plot(forecast, figsize=(10, 4))
plt.title('Past and Forecasted Initial Pledge per Sector')
plt.xlabel("Time")
plt.ylabel("Mean Initial Pledge per Sector (FIL)")
plt.show()
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```




```
time: 2 s (started: 2021-05-02 01:04:52 +00:00)
```

[17]:
```
m.fit
```

[17]:
```
<bound method Prophet.fit of <prophet.forecaster.Prophet object at
0x7f17c77f4310>>
```

```
time: 23.2 ms (started: 2021-05-02 01:04:54 +00:00)
```

[18]:
```
resample_rule = '7d'
time_column = 'activation_date'
value_column = 'sector_count'
title = 'Count of Sector Activation Date (#)'


groups = [pd.Grouper(key='activation_date', freq=resample_rule),
          pd.Grouper(key='expiration_date', freq=resample_rule)]
fig_df = df.groupby(groups).sum().reset_index()
px.density_heatmap(fig_df,
                   x='activation_date',
                   y='expiration_date',
                   z='sector_count')
```

```
time: 125 ms (started: 2021-05-02 01:04:54 +00:00)
```

[19]:
```
resample_rule = '1d'
time_column = 'activation_date'
value_column = 'sector_count'
title = 'Count of Sector Activation Date (#)'


groups = [pd.Grouper(key='activation_date', freq=resample_rule),
          pd.Grouper(key='expiration_date', freq=resample_rule)]
fig_df = df.groupby(groups).sum().reset_index()
fig = px.density_contour(fig_df,
                   x='activation_date',
                   y='expiration_date',
                   z='sector_count',
                   histfunc='sum')
fig.show()
```

```
time: 626 ms (started: 2021-05-02 01:04:54 +00:00)
```

1.4 Initial Pledge

[20]:
```
sector_count = df.initial_pledge_in_fil.sum()
v1_count = df.query("is_v1 == True").initial_pledge_in_fil.sum()

v1_6mo_count = (df.query("is_v1 == True & expiration_date < '2021-06-01 00:
↪00+00:00'")
                          .initial_pledge_in_fil
                          .sum())

v1_12mo_count = (df.query("is_v1 == True & expiration_date < '2021-12-01 00:
↪00+00:00' & expiration_date >= '2021-06-01 00:00+00:00'")
                          .initial_pledge_in_fil
```

```
                            .sum())


v1_18mo_count = (df.query("is_v1 == True & expiration_date < '2022-06-01 00:
↪00+00:00' & expiration_date >= '2021-12-01 00:00+00:00'")
                            .initial_pledge_in_fil
                            .sum())

print("---")
print(f"Total collateral (Million FIL): {sector_count / 1e6 :.3g}")
print(f"V1 collateral (Million FIL): {v1_count / 1e6 :.3g}")
print(f"V1 sectors share (%): {v1_count / sector_count :.1%}")
print("---")
print(f"6mo V1 sectors share (%) of total collateral: {v1_6mo_count /␣
↪sector_count :.2%}")
print(f"12mo V1 sectors share (%) of total collateral: {v1_12mo_count /␣
↪sector_count :.2%}")
print(f"18mo V1 sectors share (%) of total collateral: {v1_18mo_count /␣
↪sector_count :.2%}")
print("---")
print(f"6mo V1 sectors share (%) of total collateral: {v1_6mo_count / v1_count :
↪.2%}")
print(f"12mo V1 sectors share (%) of total collateral: {v1_12mo_count /␣
↪v1_count :.2%}")
print(f"18mo V1 sectors share (%) of total collateral: {v1_18mo_count /␣
↪v1_count :.2%}")
print("---")
```

```
---
Total collateral (Million FIL): 38.9
V1 collateral (Million FIL): 5.78
V1 sectors share (%): 14.9%
---
6mo V1 sectors share (%) of total collateral: 3.06%
12mo V1 sectors share (%) of total collateral: 0.70%
18mo V1 sectors share (%) of total collateral: 11.10%
---
6mo V1 sectors share (%) of total collateral: 20.58%
12mo V1 sectors share (%) of total collateral: 4.74%
18mo V1 sectors share (%) of total collateral: 74.68%
---
time: 36.5 ms (started: 2021-05-02 01:04:54 +00:00)
```

[21]:
```
resample_rule = '1m'
time_column = 'expiration_date'
value_column = 'initial_pledge_in_fil'
title = 'Initial Pledge (FIL) of Expiring  Sectors (#)'
```

```
groups = [pd.Grouper(key='expiration_date', freq=resample_rule),
          'is_v1']

fig_df = (df.groupby(groups)
            .sum()
         )


fig = px.bar(fig_df.reset_index(),
             x=time_column,
             y=value_column,
             color='is_v1',
             title=title)
fig.show()
```

```
time: 80.2 ms (started: 2021-05-02 01:04:54 +00:00)
```

[22]:
```
resample_rule = '1m'
time_column = 'expiration_date'
value_column = 'initial_pledge_in_fil'
title = 'Initial Pledge (FIL) of Expiring Sectors, grouped by Sector Version'


groups = [pd.Grouper(key='expiration_date', freq=resample_rule),
          'is_v1']

fig_df = (df.groupby(groups)
            .sum()
            .reset_index()
         )


fig = px.bar(fig_df,
             x=time_column,
             y=value_column,
             facet_col='is_v1',
             title=title)
fig.show()
```

```
time: 92.4 ms (started: 2021-05-02 01:04:55 +00:00)
```

[23]:
```
resample_rule = '1m'
time_column = 'activation_date'
value_column = ['initial_pledge_in_thousand_fil']
title = 'Sum of Initial Pledge (FIL) across activation dates'
```

```
resample_and_bar_plot(df, resample_rule, time_column, value_column, title).
↪show()
```

```
time: 74 ms (started: 2021-05-02 01:04:55 +00:00)
```

1.5 RB Network Power

[24]:
```
resample_rule = '1m'
time_column = 'expiration_date'
value_column = 'network_power_in_pib'
title = 'RB Network Power (PiB) of Expiring  Sectors (#)'


groups = [pd.Grouper(key='expiration_date', freq=resample_rule),
          'is_v1']

fig_df = (df.groupby(groups)
            .sum()
         )


fig = px.bar(fig_df.reset_index(),
             x=time_column,
             y=value_column,
             color='is_v1',
             title=title)
fig.show()
```

```
time: 77.9 ms (started: 2021-05-02 01:04:55 +00:00)
```

[25]:
```
resample_rule = '1m'
time_column = 'expiration_date'
value_column = 'network_power_in_pib'
title = 'RB Network Power (PiB) of Expiring Sectors, grouped by Sector Version'


groups = [pd.Grouper(key='expiration_date', freq=resample_rule),
          'is_v1']

fig_df = (df.groupby(groups)
            .sum()
            .reset_index()
         )


fig = px.bar(fig_df,
             x=time_column,
```

```
            y=value_column,
            facet_col='is_v1',
            title=title)
fig.show()
```

```
time: 92.5 ms (started: 2021-05-02 01:04:55 +00:00)
```

[26]:
```
resample_rule = '1m'
time_column = 'activation_date'
value_column = ['network_power_in_pib']
title = 'Sum of RB Network Power (PiB) across activation dates'
resample_and_bar_plot(df, resample_rule, time_column, value_column, title).
 ↪show()
```

```
time: 75.1 ms (started: 2021-05-02 01:04:55 +00:00)
```

[ ]:
```

```