Welcome to your Jupyter Book

Contents

This is a small sample book to give you a feel for how book content is structured. It shows off a few of the major file types, as well as some sample content. It does not go in-depth into any particular topic - check out the Jupyter Book documentation for more information.

Check out the content pages bundled with this sample book to see more.

Governance In DeFi

This article reviews the

Introduction

With centralisation the main theme of traditional finance, DeFi's emerged as a viable alternative to the centralised financal institution building upon the peer-to-peer transactions functionality of cryptocurrencies.

Governance Token

Governance tokens are units that are interchangeable and serve to implement a voting mechanism among a group of participants. Through majority-voting schemes, holders can express their intentions for the development of the protocol [].

Skip to main content

Evolution of Governance in DeFi Protocols

The evolution of governance in a number of DeFi protocols often follows a pattern with three major stages team-centred governance and took [].

1. Centralised Team-based Governance:
2. Decentralised Token-based Governance:
3. Centralised Token-based Governance:

Centrality in Token Distrubution

Effective token distribution strategies are of paramount importance for governance in DeFi. This is because the distribution process determines the number of individuals who can exercise control over a project, as well as the extent of their voting power. Hence, a well-planned and executed token distribution strategy is a critical aspect of the success of voting rights tokens.

Maker DAO

Findings

In conducting case studies, DeFi protocols analysed demonstrated a high degree of centrality []. A combination of lending protocols, decentralised exchanges and yield aggregators were analysed to understand the governance centralisation. Another study conducted an empirical analysis of the top DeFi protocols by examining the wallet addresses and token holdings of participants and found evidence of centrality with voting power controlled by the top 5, top 100 and top 1000 addresses []. The protocols analysed in this study included Compound, Uniswap, Compound, Balancer and Yearn Finance. Compoound showed the highest evidence of centrality and Uniswap the least with the top 5 address for both constituting 42.1% and 12.05% respectively. Another unique approach involved the survey of users of DeFi protocols through interviews []. Voter collusion, low participation and voter apathy were identified as the main challenges in decentralised governance. In factoring the change in voting power over time, the centraliaiton of DeFI protocols increased with time with the majority token holders purchasing more tokens over a period of time [].

Content with notebooks

You can also create content with Jupyter Notebooks. This means that you can include code blocks and their outputs in your book.

Markdown + notebooks

As it is markdown, you can embed images, HTML, etc into your posts!

You can also add_{math} and

$$math^{blocks}$$

or

$$\text{mean} la_{tex}$$

$$mathblocks$$

But make sure you \$Escape \$your \$dollar signs \$you want to keep!

Skip to main content

MyST markdown

MyST markdown works in Jupyter Notebooks as well. For more information about MyST markdown, check out the MyST guide in Jupyter Book, or see the MyST markdown documentation.

Code blocks and outputs

Jupyter Book will also embed your code blocks and output in your book. For example, here's some sample Matplotlib code:

```
from matplotlib import rcParams, cycler
import matplotlib.pyplot as plt
import numpy as np
plt.ion()
```

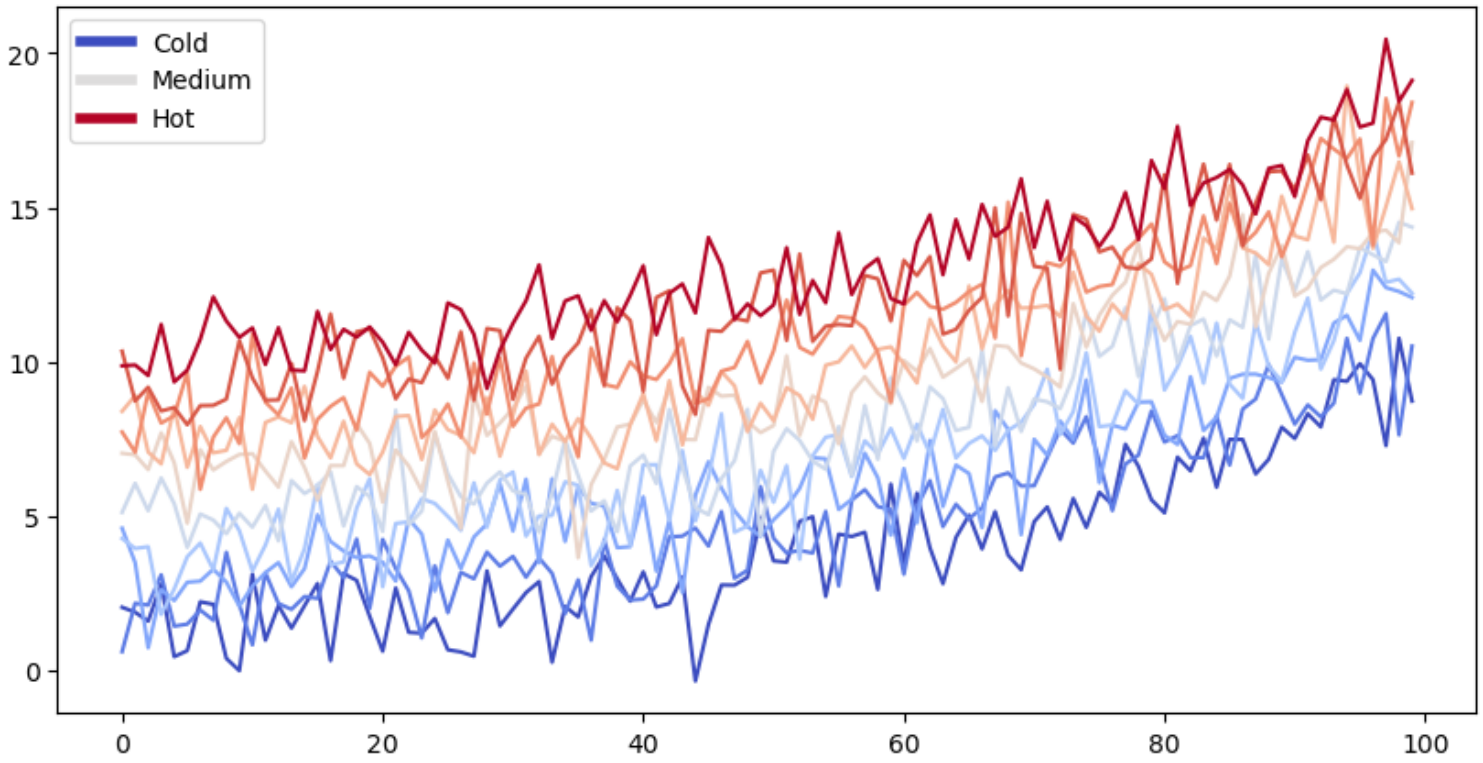
```
<matplotlib.pyplot._IonContext at 0x7ff631a2ca00>
```

```
# Fixing random state for reproducibility
np.random.seed(19680801)

N = 10
data = [np.logspace(0, 1, 100) + np.random.randn(100) + ii for ii in range(N)]
data = np.array(data).T
cmap = plt.cm.coolwarm
rcParams['axes.prop_cycle'] = cycler(color=cmap(np.linspace(0, 1, N)))


from matplotlib.lines import Line2D
custom_lines = [Line2D([0], [0], color=cmap(0.), lw=4),
                Line2D([0], [0], color=cmap(.5), lw=4),
                Line2D([0], [0], color=cmap(1.), lw=4)]

fig, ax = plt.subplots(figsize=(10, 5))
lines = ax.plot(data)
ax.legend(custom_lines, ['Cold', 'Medium', 'Hot']);
```



There is a lot more that you can do with outputs (such as including interactive outputs) with your book. For more information about this, see the Jupyter Book documentation

Notebooks with MyST Markdown

Jupyter Book also lets you write text-based notebooks using MyST Markdown. See the Notebooks with MyST Markdown documentation for more detailed instructions. This page shows off a notebook written in MyST Markdown.

An example cell

With MyST Markdown, you can define code cells with a directive like so:

```
print(2 + 2)
```

```
4
```

When your book is built, the contents of any {...} blocks will be executed with your default

Skip to main content

> **See also**
>
> Jupyter Book uses Jupytext to convert text-based files to notebooks, and can support many other text-based notebook files.

Create a notebook with MyST Markdown

MyST Markdown notebooks are defined by two things:

1. YAML metadata that is needed to understand if / how it should convert text files to notebooks (including information about the kernel needed). See the YAML at the top of this page for example.
2. The presence of `{code-cell}` directives, which will be executed with your book.

That's all that is needed to get started!

Quickly add YAML metadata for MyST Notebooks

If you have a markdown file and you'd like to quickly add YAML metadata to it, so that Jupyter Book will treat it as a MyST Markdown Notebook, run the following command:

```
jupyter-book myst init path/to/markdownfile.md
```