MkDocs Exporter

6.2.0

Getting started

Getting started

Introduction

MkDocs Exporter is a plugin for MkDocs, it allows you to export your documentation to various formats such as PDF. If you're familiar with Python, you can install the plugin with `pip` (or your favourite package manager).

Examples

- **Read** this documentation in the PDF format
- **View** this page in the PDF format

Prerequisites

- Python `>= 3.9`
- MkDocs `>= 1.4`
- A compatible theme
  - `material` (🌠 *used by this documentation*)
  - `readthedocs`

Installation

You can start by installing the plugin with the package manager of your choice:

```
pip install mkdocs-exporter
```

You can now register the plugin in your configuration file:

```yaml
plugins:
  - exporter
```

Check out the configuration guides for more details about how to use and configure the plugin.

MkDocs Exporter

6.2.0

Generating PDF documents

Generating PDF documents

Prerequisites

Under-the-hood, this library depends on a web browser controlled by *Playwright* to generate PDF documents.

At the time of writing, *Playwright* has the following requirements:

- **Python**
  - Version 3.9 or higher
- **Operating system**
  - Windows 10+, Windows Server 2016+ or Windows Subsystem for Linux (WSL)
  - macOS 12 *Monterey*, macOS 13 *Ventura* or macOS 14 *Sonoma*
  - Debian 11, Debian 12, Ubuntu 20.04 or Ubuntu 22.04, with `x86-64` or `arm64` architecture

Is your operating system not compatible?

You can use *Docker* to build your documentation from any operating system.
Feel free to explore the Dockerfile used by this documentation for guidance.

To install the browser and its required dependencies, run:

```
playwright install chrome --with-deps
```

Configuration

Inside your `mkdocs.yml` configuration file, make sure to enable the PDF format:

```yaml
plugins:
  - exporter:
      formats:
        pdf:
          enabled: !ENV [MKDOCS_EXPORTER_PDF_ENABLED, true]
```

With this configuration, the `MKDOCS_EXPORTER_PDF_ENABLED` environment variable can be used as a toggle for the generation process.
By default, PDF documents will be generated.

Controlling PDF generation

In *MkDocs Exporter*, you can control whether a specific Markdown file should be included in the PDF generation process using the front matter configuration.
This feature provides flexibility in managing which documents are exported to PDF format based on your project's requirements.

Enabling PDF generation

Info

By default, all pages are included in the PDF generation process.

If you enable the explicit mode in your configuration, you'll have to explicitly enable PDF generation for your pages. This explicit configuration ensures that each page's inclusion in the PDF output is deliberate and controlled.

```yaml
plugins:
  - exporter:
      formats:
        pdf:
          explicit: true
```

To enable PDF generation for a specific page, add the following front matter at the beginning of the file:

```yaml
---
pdf: true
---
```

Disabling PDF generation

Conversely, if you want to exclude a pages from the PDF generation process, you can specify:

```yaml
---
pdf: false
---
```

MkDocs Exporter will skip including this file when creating PDF documents.
This capability allows you to manage and customize the content that appears in your PDF exports with precision.

Enhancing performance with concurrency

Concurreny enhances the efficiency of PDF generation by enabling the plugin to utilize multiple threads concurrently. This means that instead of processing one page at a time sequentially, the plugin can distribute the workload across multiple threads, thereby speeding up the generation process significantly.

This option is particularly beneficial for larger documentation projects with numerous pages.

```yaml
plugins:
  - exporter:
      formats:
        pdf:
          concurrency: 16
```

This configuration instructs *MkDocs Exporter* to generate up to 16 PDF documents concurrently.

Cover pages

In this section, you will learn how to configure and customize cover pages for your PDF documents using the *MkDocs Exporter* plugin.
Cover pages add a professional touch to your documentation and can include titles, authors, dates, logos, and other relevant information.

Unleashing the power of templates

The following examples use the `macros` plugin to introduce templating into pages.

While using macros can enhance the customization of your cover pages, it is purely facultative.
You can achieve a fully functional and visually appealing cover pages using standard Markdown without the need for additional plugins.

Global cover pages

The *MkDocs Exporter* plugin allows you to define global cover pages that apply to your entire documentation project. This ensures a consistent look and feel across all your PDF documents without needing to set up cover pages individually for each file.

`mkdocs.yml`

```yaml
plugins:
  - exporter:
    formats:
      pdf:
        stylesheets:
          - resources/stylesheets/pdf.scss
        covers:
          front: resources/templates/covers/front.html.j2
          back: resources/templates/covers/back.html.j2
```

> View the full content of this file here.

`resources/templates/covers/front.html.j2`

```html
<div class="front-cover">
  <img src="/assets/images/background.png">
  <section>
    <div class="brand">{{ config.site_name }}</div>
    <div class="title">{{ page.title }}</div>
  </section>
</div>
```

> View the full content of this file here.

`resources/templates/covers/back.html.j2`

```html
<div class="back-cover">
  <section>
    <div class="title">{{ config.site_name }}</div>
  </section>
</div>
```

If you use the template **front cover page**, do not forget to copy the background image `docs/assets/images/background.png` to your project.

Per-page cover pages

While global cover pages provide a uniform look across your documentation, you can override them on a per-page basis using front matter.
This flexibility allows you to customize cover pages for specific documents when needed.

```
---
covers:
  front: ./resources/templates/covers/special.html.j2
---

# My special page

This configuration specifies a custom front cover page template located at
`./resources/templates/covers/special.html.j2` for this page.
You can follow the same operation to specify a custom back cover page.
```

Hiding cover pages

In some cases, you might want to hide cover pages for specific documents within your project.

The *MkDocs Exporter* plugin allows you to easily hide cover pages on a per-page basis using the front matter in your Markdown files. This flexibility ensures that you can maintain a clean and streamlined appearance for certain pages without the need for cover pages.

```
---
hidden:
  - covers
---

# Appendix A

This appendix contains supplementary information but wont have cover pages...
```

Buttons

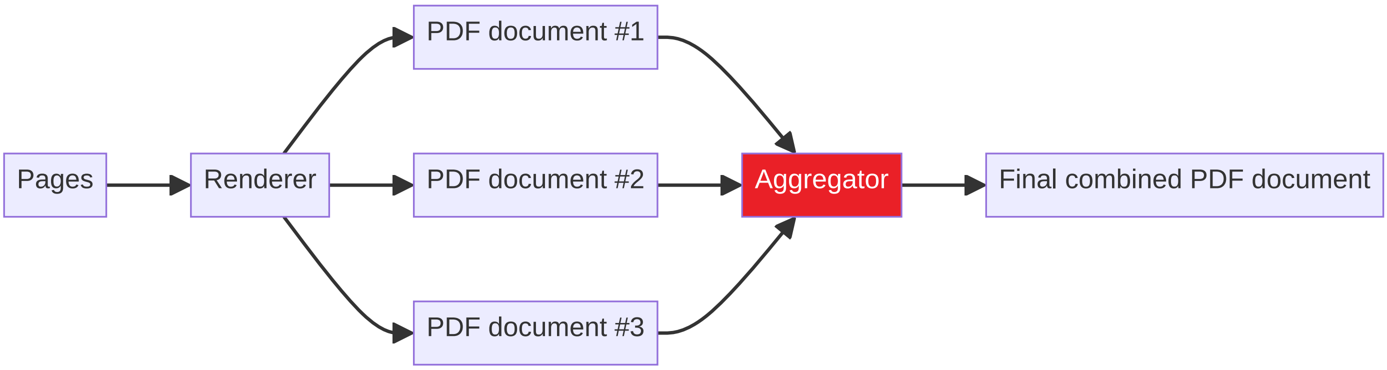
Adding a download button

To provide users with the ability to download your page as a PDF document, you can set up a download button using the *MkDocs Exporter* plugin.
This section outlines the configuration needed in your `mkdocs.yml` file to enable and customize the download button.

```
plugins:
  - exporter:
      buttons:
        - title: Download as PDF
          icon: material-file-download-outline
          enabled: !!python/name:mkdocs_exporter.formats.pdf.buttons.download.enabled
          attributes: !!python/name:mkdocs_exporter.formats.pdf.buttons.download.attributes
```

Combining pages into a single PDF document

In *MkDocs Exporter*, the aggregator feature allows you to consolidate multiple individual PDF documents into a single cohesive PDF file.
This functionality is particularly useful for compiling comprehensive documentation or reports where it's beneficial to have all content consolidated into one easily distributable format.



Here's how you can set it up in your `mkdocs.yml` configuration file:

```
plugins:
  - exporter:
      formats:
        pdf:
          aggregator:
            enabled: true
            output: documentation.pdf
            covers: all
```

Configuring cover pages behavior

When aggregating PDF documents, you have the flexibility to configure the behavior of cover pages to suit your needs.
There are five available options for managing cover pages:

- `all` (*default*): retains every cover page from all documents.
- `none` : removes every cover page from the aggregated document, resulting in a compilation that includes only the main content of each PDF.
- `limits` : retains the front cover of the first document and the back cover of the last document, while removing all other cover pages in between.
- `book` : retains the front cover of the first document and the back cover of the last document, while keeping only the front cover pages in between.
- `front` : preserves every front cover page from all documents but removes all back cover pages.
- `back` : preserves every back cover page from all documents but removes all front cover pages.

Choose the behavior that best aligns with your document aggregation needs to ensure the final PDF meets your requirements.

MkDocs Exporter

6.2.0

Adding buttons to pages

Adding buttons to pages

You can define custom buttons at the top of your pages.

Try it out

A custom button is featured on this page, check it out!
You can find its configuration below.

Usage

Adding a download button

The following configuration excerpt will add a download button to pages that have a PDF document:

```yaml
plugins:
  - exporter:
      buttons:
        - title: Download as PDF
          icon: material-file-download-outline
          enabled: !!python/name:mkdocs_exporter.formats.pdf.buttons.download.enabled
          attributes: !!python/name:mkdocs_exporter.formats.pdf.buttons.download.attributes
```

Defining a dynamic button

As you saw in the previous example, Python functions can be used to dynamically resolve button attributes.
Now, let's write a button that, when clicked, initiates a Google search using the current page's title as the query.

To begin with, let's write the function that will return the button's `href` attribute:

```python
from urllib.parse import urlencode
from mkdocs_exporter.page import Page

def href(page: Page, **kwargs) -> str:
  """The button's 'href' attribute."""

  return 'https://google.com/search?' + urlencode({'q': page.title})
```

Next, we can define the button and specify the path to the function previously defined (assuming it has been saved in the `button.py` file under the `my_module` module):

```yaml
plugins:
  - exporter:
      buttons:
        - title: Download as PDF
          icon: material-file-download-outline
          attributes:
            href: !!python/name:my_module.button.href
```

Repeat this process as needed; you can apply this method to any property of a button.

Adding buttons to a specific page

You can add buttons to pages using the buttons field in your page's front matter, enabling you to define buttons specific to each page.

Here's how it's configured for this page:

```
---
buttons:
  - title: I'm Feeling Lucky
    icon: material-star-outline
    attributes:
      class: md-content__button md-icon md-icon-spin
      href: https://www.youtube.com/watch?v=dQw4w9WgXcQ
      target: _blank
---

# Adding buttons to pages
```

MkDocs Exporter
6.2.0

Configuration

Configuration

`mkdocs_exporter.config.Config`

The plugin's configuration.

Source code in `mkdocs_exporter/config.py`

```python
class Config(BaseConfig):
  """The plugin's configuration."""

  theme = c.Optional(c.Theme(default=None))
  """Override the theme used by your MkDocs instance."""

  formats = c.SubConfig(FormatsConfig)
  """The formats to generate."""

  buttons = c.ListOfItems(c.SubConfig(ButtonConfig), default=[])
  """The buttons to add."""

  logging = c.SubConfig(LoggingConfig)
  """The logging configuration."""
```

`buttons = c.ListOfItems(c.SubConfig(ButtonConfig), default=[])`

> The buttons to add.

`formats = c.SubConfig(FormatsConfig)`

> The formats to generate.

`logging = c.SubConfig(LoggingConfig)`

> The logging configuration.

`theme = c.Optional(c.Theme(default=None))`

> Override the theme used by your MkDocs instance.

`mkdocs_exporter.config.ButtonConfig`

The configuration of a button.

Source code in `mkdocs_exporter/config.py`

```python
class ButtonConfig(BaseConfig):
  """The configuration of a button."""

  enabled = c.Type((bool, Callable), default=True)
  """Is the button enabled?"""

  title = c.Type((str, Callable))
  """The button's title."""

  icon = c.Type((str, Callable))
  """The button's icon (typically, an SVG element)."""

  attributes = c.Type((dict, Callable), default={})
  """Some extra attributes to add to the button."""
```

`attributes = c.Type((dict, Callable), default={})`

> Some extra attributes to add to the button.

`enabled = c.Type((bool, Callable), default=True)`

> Is the button enabled?

`icon = c.Type((str, Callable))`

> The button's icon (typically, an SVG element).

`title = c.Type((str, Callable))`

> The button's title.

`mkdocs_exporter.config.FormatsConfig`

Source code in `mkdocs_exporter/config.py`

```python
class FormatsConfig(BaseConfig):

  pdf = c.SubConfig(PDFFormatConfig)
  """The PDF format configuration."""
```

`pdf = c.SubConfig(PDFFormatConfig)`

> The PDF format configuration.

`mkdocs_exporter.config.LoggingConfig`

The logging configuration.

Source code in `mkdocs_exporter/config.py`

```python
class LoggingConfig(BaseConfig):
  """The logging configuration."""

  level = c.Choice(['debug', 'info', 'warning', 'error', 'critical'], default='info')
  """The log level."""
```

`level = c.Choice(['debug', 'info', 'warning', 'error', 'critical'], default='info')`

> The log level.

MkDocs Exporter

6.2.0

PDF

PDF

`mkdocs_exporter.formats.pdf.config.Config`

The plugin's configuration.

Source code in `mkdocs_exporter/formats/pdf/config.py`

```python
class Config(BaseConfig):
  """The plugin's configuration."""

  enabled = c.Type(bool, default=True)
  """Is the generator enabled?"""

  explicit = c.Type(bool, default=False)
  """Should pages specify explicitly that they should be rendered as PDF?"""

  concurrency = c.Type(int, default=4)
  """The maximum number of concurrent PDF generation tasks."""

  stylesheets = c.ListOfItems(c.File(exists=True), default=[])
  """A list of custom stylesheets to apply before rendering documents."""

  scripts = c.ListOfItems(c.File(exists=True), default=[])
  """A list of custom scripts to inject before rendering documents."""

  covers = c.SubConfig(CoversConfig)
  """The document's cover pages."""

  browser = c.SubConfig(BrowserConfig)
  """The browser's configuration."""

  url = c.Optional(c.Type(str))
  """The base URL that'll be prefixed to links with a relative path."""

  aggregator = c.SubConfig(AggregatorConfig)
  """The aggregator's configuration."""
```

`aggregator = c.SubConfig(AggregatorConfig)`

> The aggregator's configuration.

`browser = c.SubConfig(BrowserConfig)`

> The browser's configuration.

`concurrency = c.Type(int, default=4)`

> The maximum number of concurrent PDF generation tasks.

`covers = c.SubConfig(CoversConfig)`

> The document's cover pages.

`enabled = c.Type(bool, default=True)`

> Is the generator enabled?

`explicit = c.Type(bool, default=False)`

> Should pages specify explicitly that they should be rendered as PDF?

`scripts = c.ListOfItems(c.File(exists=True), default=[])`

> A list of custom scripts to inject before rendering documents.

`stylesheets = c.ListOfItems(c.File(exists=True), default=[])`

> A list of custom stylesheets to apply before rendering documents.

`url = c.Optional(c.Type(str))`

> The base URL that'll be prefixed to links with a relative path.

`mkdocs_exporter.formats.pdf.config.CoversConfig`

The cover's configuration.

Source code in `mkdocs_exporter/formats/pdf/config.py`

```python
class CoversConfig(BaseConfig):
  """The cover's configuration."""

  front = c.Optional(c.File(exists=True))
  """The front cover template location."""

  back = c.Optional(c.File(exists=True))
  """The back cover template location."""
```

`back = c.Optional(c.File(exists=True))`

> The back cover template location.

`front = c.Optional(c.File(exists=True))`

> The front cover template location.

`mkdocs_exporter.formats.pdf.config.BrowserConfig`

The browser's configuration.

Source code in `mkdocs_exporter/formats/pdf/config.py`

```python
class BrowserConfig(BaseConfig):
  """The browser's configuration."""

  debug = c.Type(bool, default=False)
  """Should console messages sent to the browser be logged?"""

  headless = c.Type(bool, default=True)
  """Should the browser start in headless mode?"""

  timeout = c.Type(int, default=60_000)
  """The timeout when waiting for the PDF to render."""

  args = c.ListOfItems(c.Type(str), default=[])
  """Extra arguments to pass to the browser."""
```

`args = c.ListOfItems(c.Type(str), default=[])`

> Extra arguments to pass to the browser.

`debug = c.Type(bool, default=False)`

> Should console messages sent to the browser be logged?

`headless = c.Type(bool, default=True)`

> Should the browser start in headless mode?

`timeout = c.Type(int, default=60000)`

> The timeout when waiting for the PDF to render.

`mkdocs_exporter.formats.pdf.config.AggregatorConfig`

The aggregator's configuration.

Source code in `mkdocs_exporter/formats/pdf/config.py`

```
class AggregatorConfig(BaseConfig):
  """The aggregator's configuration."""

  enabled = c.Type(bool, default=False)
  """Is the aggregator enabled?"""

  output = c.Type(str, default='combined.pdf')
  """The aggregated PDF document output file path."""

  metadata = c.Type(dict, default={})
  """Some metadata to append to the PDF document."""

  covers = c.Choice(['all', 'none', 'limits', 'book', 'front', 'back'], default='all')
  """The behavior of cover pages."""
```

`covers = c.Choice(['all', 'none', 'limits', 'book', 'front', 'back'], default='all')`

> The behavior of cover pages.

`enabled = c.Type(bool, default=False)`

> Is the aggregator enabled?

`metadata = c.Type(dict, default={})`

> Some metadata to append to the PDF document.

`output = c.Type(str, default='combined.pdf')`

> The aggregated PDF document output file path.

MkDocs Exporter
6.2.0

Diagrams

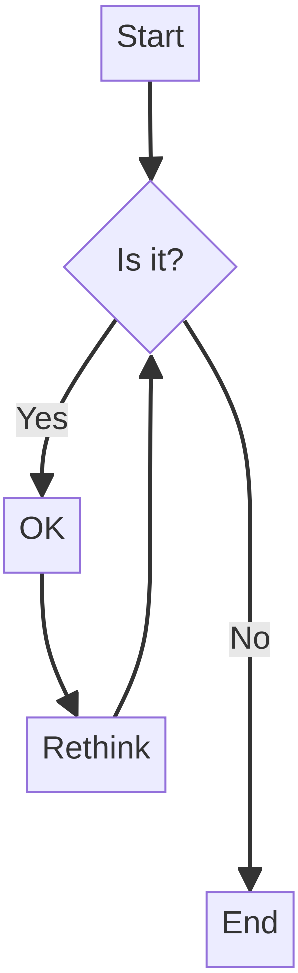
Diagrams

Mermaid

Check out Mermaid's documentation for a reference of all supported diagrams.

Flowchart

Pellentesque nec lacus est. Phasellus pulvinar volutpat nisl aliquet auctor.
Nunc tincidunt molestie mi, et porttitor lectus congue in. Aliquam eu tortor viverra libero imperdiet suscipit.



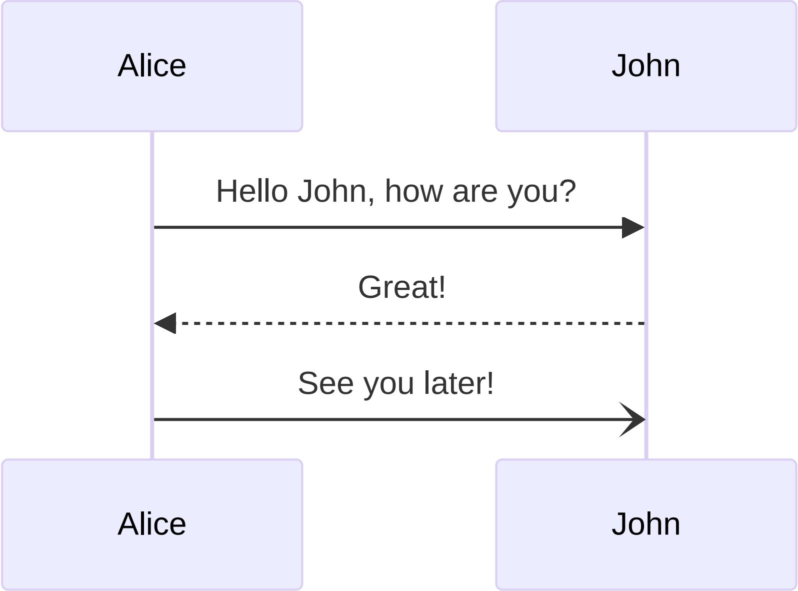
Phasellus auctor consectetur diam et ullamcorper. Ut vel lacus massa. Quisque magna magna, semper quis feugiat at, lobortis non leo. Quisque vitae sollicitudin ex. Nulla ut laoreet purus, ut porttitor dui. Suspendisse et ornare erat. Nam id ornare lorem.

Nam quam justo, commodo eu lobortis vestibulum, molestie sed dolor.

Sequence diagram

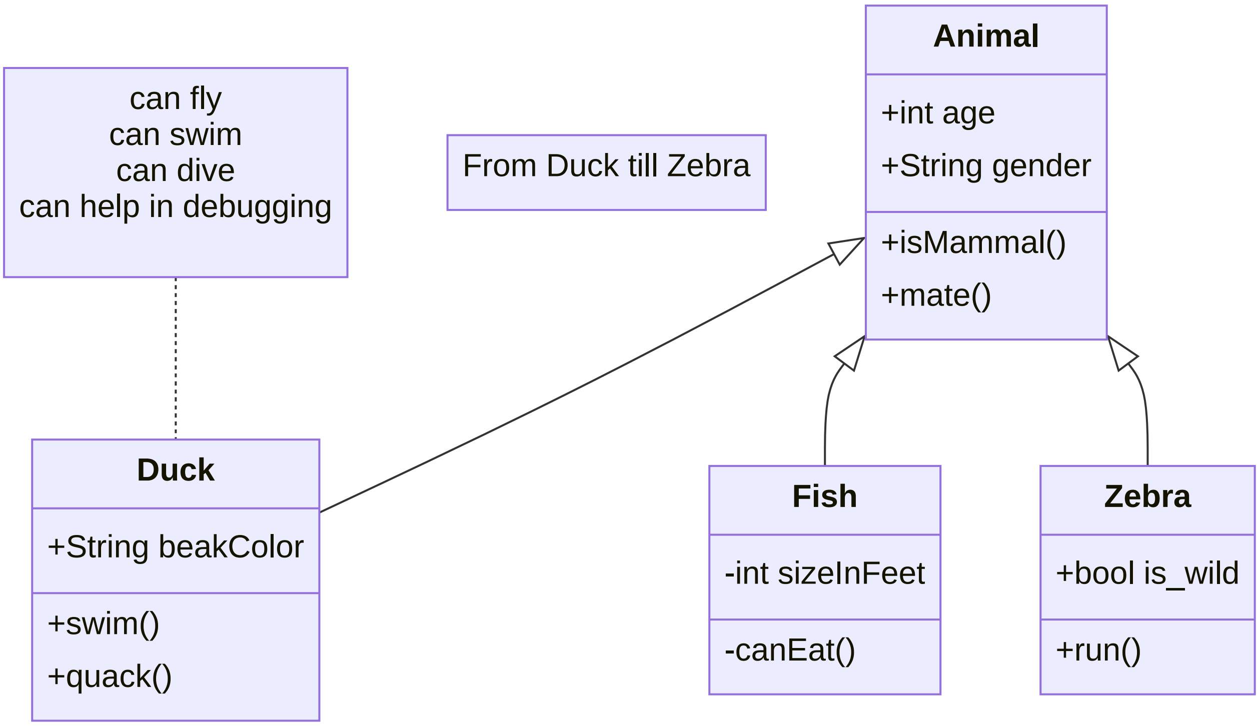
Morbi justo enim, rhoncus nec dictum vitae, porttitor ac odio. Pellentesque ac malesuada neque, quis mollis purus. Aliquam interdum, est a mattis vestibulum, nibh nisl pulvinar orci, a faucibus ante massa et diam.

Phasellus sed velit ex. Proin condimentum dolor ac felis pellentesque imperdiet. Nulla porta est lacus. Cursus tincidunt cursus diam, sit amet condimentum risus convallis scelerisque.



Class diagram

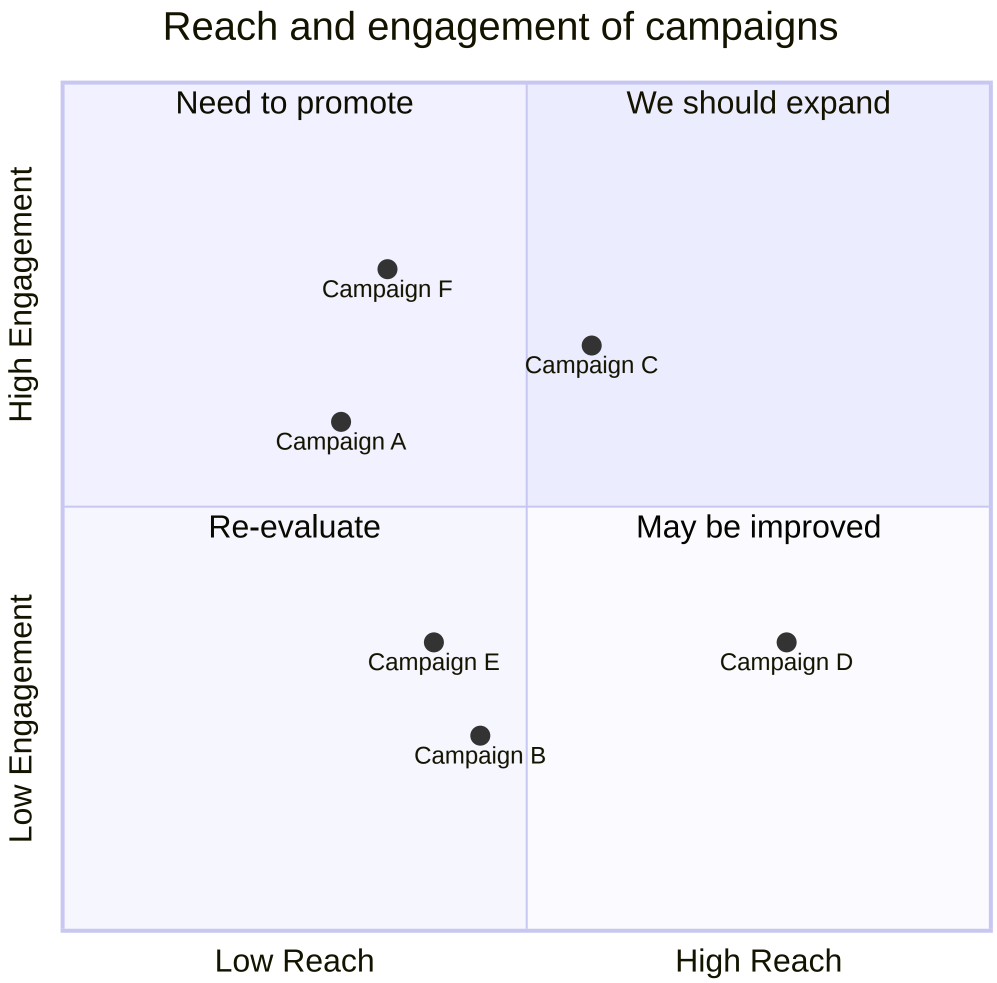
Integer iaculis in sem in porttitor. Duis tempus ullamcorper purus, ac efficitur massa aliquam sed. Nam et varius diam. Duis suscipit ultrices odio, non volutpat elit bibendum a. In sit amet ultrices metus. Pellentesque vitae malesuada libero. Phasellus tincidunt cursus diam, sit amet condimentum risus convallis scelerisque. Donec eu luctus tellus, ac porta ex. Nullam dictum eros a consectetur posuere. Donec sed ligula auctor diam faucibus pulvinar nec ut erat. Integer accumsan laoreet mollis. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aliquam pellentesque tempor pulvinar.



Nullam dictum eros a consectetur posuere. Donec sed ligula auctor diam faucibus pulvinar nec ut erat. Integer accumsan laoreet mollis.
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aliquam pellentesque tempor pulvinar.

Quadrant chart

Duis in consectetur metus, sed vestibulum orci. Quisque tincidunt, nunc eu tincidunt volutpat, risus mi ultricies elit, nec malesuada est ipsum eu lacus. Suspendisse vitae neque pulvinar, tristique lectus sit amet, aliquet arcu. Sed mollis tristique lacus non scelerisque. Suspendisse convallis consequat leo id placerat. Morbi sodales bibendum nibh, vitae interdum erat sagittis sed. Morbi consequat vulputate odio nec mollis. Fusce a neque lacus. Morbi lorem tortor, lobortis id magna imperdiet, malesuada euismod velit.



MkDocs Exporter

6.2.0

Mathematics

Mathematics

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc quis augue massa. Vestibulum suscipit et nisi in imperdiet. Nunc et mattis arcu, feugiat venenatis elit. Praesent hendrerit lacus felis, vel volutpat massa tristique vitae.

Quisque quam urna, vulputate quis malesuada quis, iaculis sit amet lectus. Vivamus malesuada dignissim lectus in condimentum. Integer et efficitur sapien. Aenean aliquam pharetra porttitor. Morbi rutrum placerat lectus. Aenean imperdiet nisi lobortis sagittis ultricies.

The homomorphism f is injective if and only if its kernel is only the singleton set e_G, because otherwise $\exists a, b \in G$ with $a \neq b$ such that $f(a) = f(b)$.

$$\ker f = \{g \in G : f(g) = e_H\}.$$

Mauris tincidunt, tortor eu convallis sollicitudin, ex risus blandit ligula, vel pulvinar lorem ante eget orci. Fusce pellentesque pulvinar velit, non molestie mi pellentesque sit amet. Sed convallis dolor vel sagittis rhoncus. Pellentesque at ullamcorper ex. Curabitur ut nisl quam. Vestibulum aliquet massa in odio tempor, quis maximus erat semper. Donec iaculis condimentum lacus, a tincidunt metus congue ut. Donec congue odio sit amet lacus semper, nec bibendum sapien accumsan.

> $$E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$
>
> Nullam ante felis, luctus et leo eget, rhoncus lacinia libero.

Curabitur in hendrerit metus, in ornare mauris. Fusce sodales odio at erat finibus, id congue nibh feugiat. Vestibulum lacinia, nulla et vestibulum ultricies, massa ex imperdiet tortor, eget mattis arcu dolor non risus. Curabitur vulputate diam vel nulla fermentum, non suscipit libero sodales. Praesent aliquet aliquam leo, at facilisis ante imperdiet ac. Nunc eleifend ligula nisi, vel aliquam sapien vehicula non. Mauris consectetur erat vel augue varius tempor.

Vivamus nec imperdiet sapien, at euismod urna. Proin tempor placerat mauris id aliquet. Phasellus lorem nunc, porta sed odio nec, aliquet aliquet velit. Nullam blandit tincidunt sollicitudin. Fusce non tincidunt felis. Integer laoreet dolor eget sodales tempor. Sed aliquet faucibus nulla et molestie. Nunc sit amet suscipit purus. Nunc turpis purus, pretium eleifend lobortis vel, vulputate ut sem. Vivamus consectetur ante id velit aliquet, ut vulputate augue feugiat. Maecenas consequat mi eget pretium maximus. Nam sed porta mi. Aliquam viverra sed libero sit amet eleifend.

$$\begin{bmatrix} x & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 + xy \\ y - 1 \end{bmatrix}.$$

In tortor velit, tincidunt at massa vitae, consequat euismod leo.

In sit amet nunc consectetur, commodo lectus vitae, consequat lacus. Duis eu odio mattis, porttitor nunc in, placerat massa. Vestibulum rhoncus libero libero, laoreet aliquam diam facilisis convallis. Etiam quis vestibulum est, id interdum odio. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Suspendisse posuere non leo ac varius. Integer hendrerit sed lacus nec hendrerit. Nunc porttitor porttitor aliquam.

$$\begin{aligned} F = \{F_x \in F_c : & (|S| > |C|) \\ & \cap (\text{minPixels} < |S| < \text{maxPixels}) \\ & \cap (|S_{\text{conected}}| > |S| - \epsilon)\} \end{aligned}$$

In sit amet nunc consectetur, commodo lectus vitae, consequat lacus. Duis eu odio mattis, porttitor nunc in, placerat massa. Vestibulum rhoncus libero libero, laoreet aliquam diam facilisis convallis. Etiam quis vestibulum est, id interdum odio. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Suspendisse posuere non leo ac varius. Integer hendrerit sed lacus nec hendrerit. Nunc porttitor porttitor aliquam.

MkDocs Exporter

6.2.0

Multimedia

Multimedia

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Images

Duis faucibus ante non lorem sollicitudin, vel dictum neque interdum. Nunc vel urna ipsum. Nullam urna massa, finibus sed porttitor nec, vehicula nec nibh. Proin mattis porttitor lectus ac elementum. Suspendisse faucibus aliquam sapien quis euismod. Vivamus justo elit, placerat at arcu vel, pulvinar finibus lectus. Nunc rhoncus fringilla enim, non molestie metus consequat nec. Curabitur volutpat sollicitudin purus eget fringilla. Sed congue mauris id mi semper, sit amet consectetur erat fringilla.

JPEG

Photo by Henry Be on Unsplash.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Vivamus ut convallis diam. Morbi laoreet fermentum auctor. Nulla nec iaculis nisi. Ut ultrices dolor id lacus vulputate, ac ultrices libero eleifend. Praesent nisi urna, lacinia non lectus et, egestas euismod sapien. Morbi eu nibh ut libero feugiat dignissim. Quisque bibendum, est vel iaculis interdum, lorem sem efficitur justo, vitae facilisis leo est in turpis. Etiam justo libero, commodo quis dolor in, feugiat pharetra sem. Nunc mauris magna, fermentum eu tristique ut, porttitor sit amet diam. Vivamus in hendrerit mauris. Sed facilisis semper enim, et consectetur justo ultricies ac. Aliquam ut felis nec eros blandit condimentum nec in mauris. Duis a viverra enim. Aliquam erat volutpat. Nam vitae diam egestas, hendrerit nulla et, viverra augue.

Aliquam vitae velit sit amet felis dapibus interdum vitae ac purus. Morbi vitae orci risus. Donec condimentum nibh id ex consectetur veleris.

Mauris vulputate lorem vitae mauris cursus porta.
Curabitur efficitur interdum massa id porttitor. Nunc tincidunt sem metus, at vulputate nunc facilisis eu. Nulla nec

massa eu nulla ornare cursus sed a est. Duis sit amet leo felis.

WebP

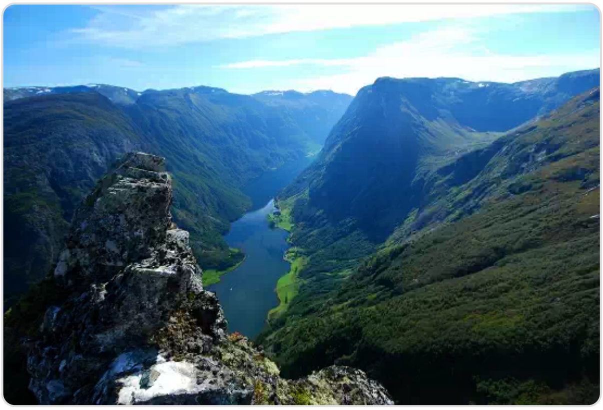


Photo by Kjetil Birkeland Moe.

In hac habitasse platea dictumst. Nullam porttitor a leo sit amet dictum. Ut a diam a turpis cursus iaculis. Sed tempus nisl quis hendrerit finibus. Duis vel mattis turpis. Suspendisse commodo, ante id luctus vestibulum, sem nunc mattis neque, at rutrum justo metus nec risus. Phasellus tincidunt tempor lacus vel dictum. Vivamus ac rutrum magna. Sed sit amet fermentum turpis. Donec molestie augue felis, sed blandit nisl vehicula at.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut a gravida turpis, at mattis nisi.