Infovis group 2: Global Wind Power Tracker

Contents

1 Introduction

1.1 Dataset selection

The Global Wind Power Tracker (GWPT)[1] is a worldwide dataset of utility-scale, on and offshore wind facilities. It includes wind farm phases with capacities of 10 megawatts (MW) or more. A wind project phase is generally defined as a group of one or more wind turbines that are installed under one permit, one power purchase agreement, and typically come online at the same time. The GWPT dataset catalogs every wind farm phase at this capacity threshold of any status, including operating, announced, pre-construction, under construction, shelved, cancelled, mothballed, or retired. The dataset itself is subdivided into a 'large' (wind farms larger than 10MW) and 'small' (wind farms between 1 and 10MW) subset. The most recent release of this data was in December 2023.

1.2 Dataset license

All Global Energy Monitor data are freely available under a Creative Commons Attribution 4.0 International Public License unless otherwise noted. Under this license, you are free to:

- Share: Copy and redistribute data in any medium or format
- Adapt: remix, transform, and build upon the data for any purpose

Under the following terms:

- Attribution: You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

[1] https://globalenergymonitor.org/projects/global-wind-power-tracker/

- No Additional Restrictions: You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

1.3 Target user

Our target user is anyone interested in wind power, with at least a high school level understanding of STEM subjects. They should be familiar with the concept of electrical power measured in Watts, the prefix "M" for million, and that wind turbines generate electrical energy. They should also know that wind farms can be located both onshore and offshore, and have a basic knowledge of world geography, including continents, regions, and countries.

1.4 Goal

The goal is to provide our target users with an interactive web dashboard to help them explore various wind power-related questions, such as:

1. How is offshore wind currently distributed across different continents?
2. In future years, will the world invest more in onshore or offshore wind?
3. Can you list the top 20 largest operating wind farms in India commissioned between 2010 and 2020?
4. Can you locate the 3rd largest operating onshore wind farm in South Africa and retrieve its capacity value?

This list is non exhaustive. The dashboard is designed as an general educational tool for exploring wind power data.

2 Data preprocessing

2.1 Exploratory data analysis

To gain a better understanding of the data, exploratory data analysis (EDA) was performed using both a Python Jupyter notebook (with pandas) and Excel workbooks. During EDA, an initial assessment of the raw data was conducted to understand its structure. Here is what was found:

- The ‘large’ subset contains 26,523 wind farm phases.
- The ‘small’ subset contains 899 additional wind farm phases.
- Each subset has 29 features.
- The dataset contains wind farms from 155 distinct countries.
- Features are expressed either as strings (e.g., project name, country, owner) or floats (e.g., capacity, latitude, longitude).

Additionally, during EDA, missing values, outliers, duplicates, and inconsistencies within key features were identified:

- There were no missing data in any key features except for the start year. 9,919 observations (36.17% of the dataset) did not have a start year.
- The distinct values for 'Installation Type' were 'onshore', 'unknown', 'offshore mount unknown', 'offshore hard mount', and 'offshore floating', which are more detailed than necessary for our target users.
- Similarly, the distinct values for 'Status' were 'operating', 'cancelled', 'pre-construction', 'announced', 'construction', 'shelved', 'retired', and 'mothballed', which are also too detailed for our target users.

Finally, some visuals, like the one in Figure 1 were created to get a feeling for the distribution of the data.

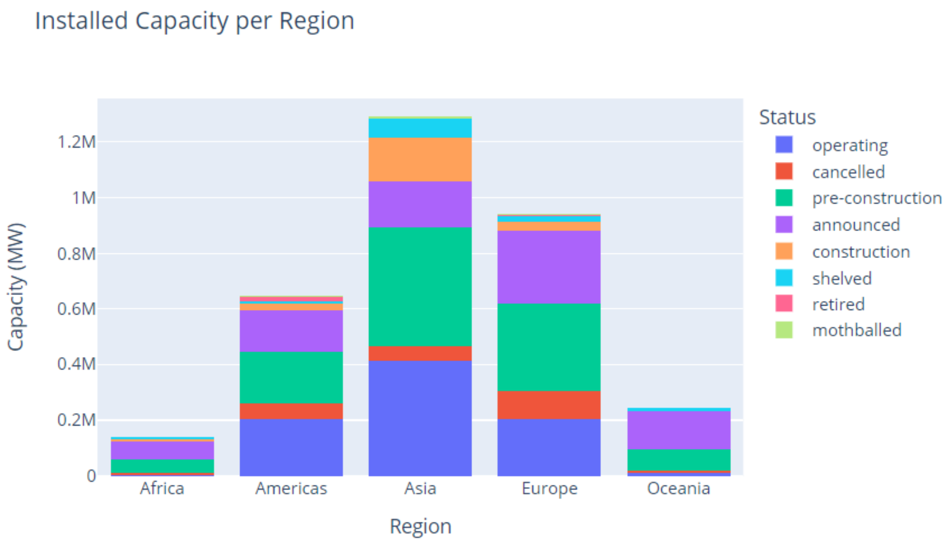


Figure 1 First visualization of the dataset made during EDA.

2.2 Data wrangling

Following EDA, data wrangling techniques were used to improve the quality of the data and create clean data ready for use in the dashboard.

Simplification of categorical features: The type column was reduced to 'onshore' and 'offshore'. 'Unknown' units were visualized and put under 'onshore' as they were all located on land. The status column was reduced to 'operating', 'future' and 'retired'. Cancelled and shelved projects were removed from the dataset as they provide no value for our target user.

Handling missing values: Missing values were handled through imputation[2] based on the nature of the data. Analysis on complete records has shown that the average life span of a wind farm is around 15 years as shown in Figure 2. This knowledge, combined with the 'status' field, which is always filled in, was used to make an educated guess about the missing 'start year' data.

Abbreviation: The project names were shortened to avoid visual clutter in the dashboard. Common occurrences like 'wind farm', 'wind project', ... were removed as this is clear from the context. This reduced the average project name length from 28 to 18 characters.

Handling outliers: Records with outliers in terms of installed capacity were removed. Based on business expertise[3] and internet research on a sample of these records we deduced that units with a capacity of more than 10GW are likely unrealistic and should be removed.

Data type harmonization. Data types were converted into the desired format (datetime, integer, ...) before storing the final cleaned records in '.parquet' files.

By using a Jupyter notebook, this ETL process is completely self-documenting and maintainable.

[2] As discussed during the mid-term presentation, we did not simply remove these records but found a way to guess a reasonable value.
[3] Two of our team members are active in the energy industry.

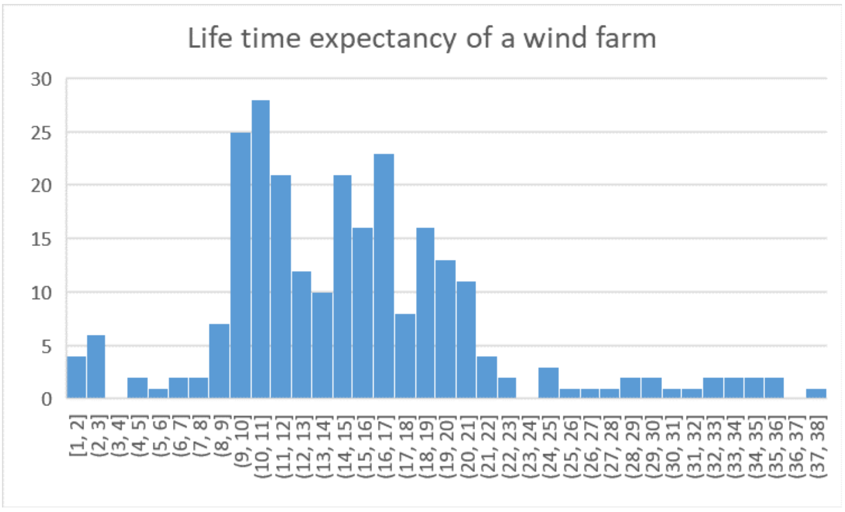


Figure 2 Distribution of the life expectancy of wind farms for which there was no missing data.

3 Validation

3.1 Domain situation and data/task abstraction

The target user – as described in section 1.3 – was considered early on in the design phase and some key questions were created that should be answered by using the developed visualization. Questions are combining 'analyse', 'search' and 'query' aspects. After development, those same questions were asked to a few potential users with different backgrounds.

3.2 Visual encoding/interaction idiom

To support the discussion on the selection and layout of the different visualization idioms, mockups were elaborated to support the validation and obtain consensus. During the mid-term presentation, having a mockup allowed the team to collect early feedback from the teaching team. This is described more in depth in section 4.1.

The different visual components are linked to each other, so changes on the filters will update all different visualization components. Alphabetical combo boxes are used for filtering on categorical values and a slider is used for filtering on numerical (time) values.

For the visualizations human capabilities are considered when selecting the channels used in the design. Markers on the map use position for location, size for capacity and hue for status. Sensible hues are selected for the different status categories (e.g. red-orange = retired) without inferring any order. Choropleth was not used for the map, as offshore concessions are often too small to have a benefit of using hue as channel (interference of channels).

For the horizontal bar chart, position is used as an extra channel for capacity and hue as a channel for status.

During validation, human time was measured to analyze how quick a user could find the answers to the test questions.

3.3 Algorithm

The dataset is static, which makes it possible to pre-compute query and summary statistics to improve performance. The map visualization was expected to be the slowest element and therefore multiple map frameworks were tested until one was found with a good performance for the amount of data (Plotly-Mapbox). It was found to take less than a second too fully load all data on the map.

4 Product

4.1 Mockups

For inspiration we drew from the John Hopkins Covid dashboard Course example. The team then drafted a mockup (Figure 3) to obtain consensus on the final product design choices. From these discussions we concluded:

- Big aggregate numbers (BANs) should be used to show aggregate results, and be placed on the left.
- The filters should be on the top of the screen. They should impact the BAN, map and bar chart.
- The map should be the central piece of the dashboard and take up sufficient space.
- A ranked bar chart should be placed on the right and allow the user to explore the largest units.

Having made this mockup allowed us to collect feedback before creating the first prototype (e.g. some filters are missing).

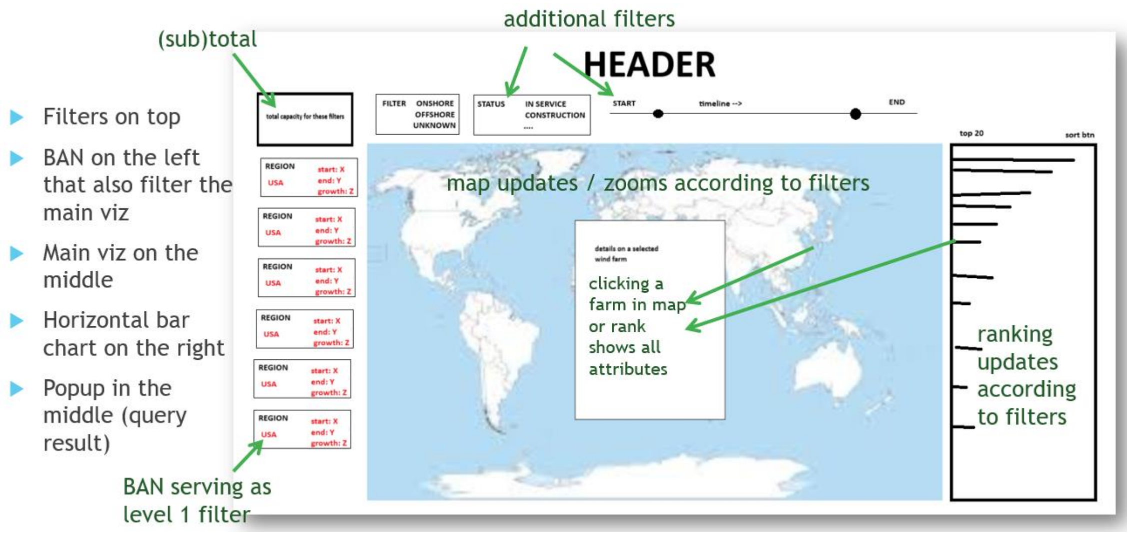

additional filters
(sub)total
HEADER
Filters on top
BAN on the left that also filter the main viz
Main viz on the middle
Horizontal bar chart on the right
Popup in the middle (query result)
total capacity for these filters
FILTER ONSHORE OFFSHORE UNKNOWN
STATUS IN SERVICE CONSTRUCTION ....
START
timeline -->
END
top 20
sort btn
REGION USA start: X end: Y growth: Z
map updates / zooms according to filters
details on a selected wind farm
clicking a farm in map or rank shows all attributes
ranking updates according to filters
BAN serving as level 1 filter

Figure 3 Most detailed mockup and basis for the final product.

4.2 Framework choices

The interactive visualization is built using the Dash and Plotly graphics.

Dash[4] is a web framework for building interactive, web-based data visualization applications using Python. Dash is stateless, meaning that each user interaction with the application does not rely on the previous interactions, allowing for scalable and consistent performance regardless of the number of users. Dash is popular because it simplifies the process of building interactive, web-based data visualizations and applications with Python, making it accessible even for those with limited web development experience.

Plotly[5] is an open-source graphing library that enables the creation of interactive, web-based visualizations with ease and flexibility. Plotly is versatile, offering numerous ways to create a wide range of interactive visualizations. Given its many bindings, Python developers can use Plotly without needing to know JavaScript. Plotly and Dash integrate seamlessly.

[4] https://dash.plotly.com/

[5] https://plotly.com/python/

Parquet
Dash
plotly

4.3 Map visualizations

With a map as the centerpiece of the app, some research time was devoted to finding and testing the optimal map visualization framework. While a 3D representation may give off a modern first impression, it is inferior to a 2D representation due to occlusion. E.g. in Figure 4 it is impossible to see what countries the circles on the north relate to.

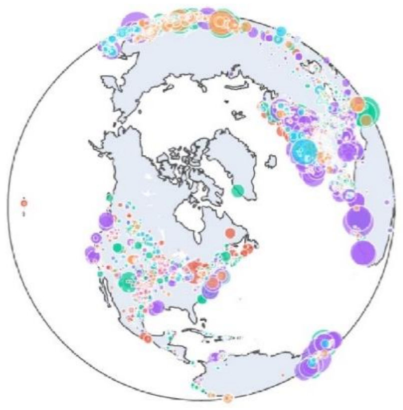


Figure 4 Map visualization experiment using 3D globe.

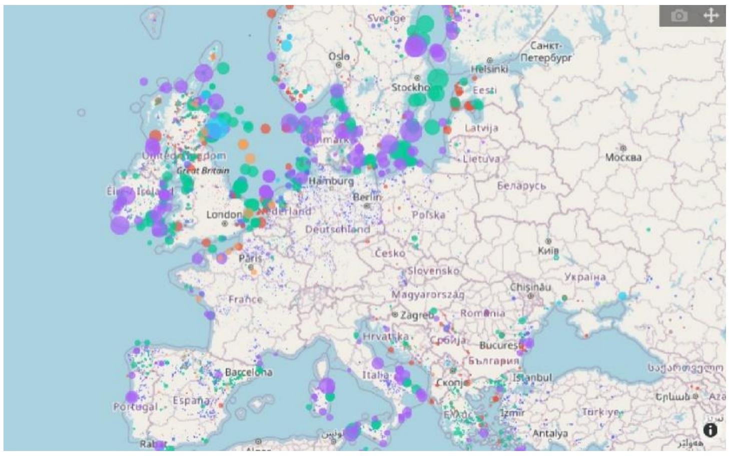


Figure 5 Map visualization experiment using 2D projection.

A comparative table was found on Plotly's website and copied in appendix B. After testing, the choice was made to go forward with Plotly's built-in support for map visualizations based on mapbox. Mapbox is a platform that provides tools and APIs for creating custom, interactive maps and geospatial visualizations for web and mobile applications.

During testing, special attention was given to built-in semantic zooming support to avoid showing all data as separate markers on a global level for overview and performance reasons. Available clustering options of the Plotly Mapbox package was properly tested, but found to be too limited in customization (e.g. size of markers is fixed and not related to total capacity as shown in Figure 6).

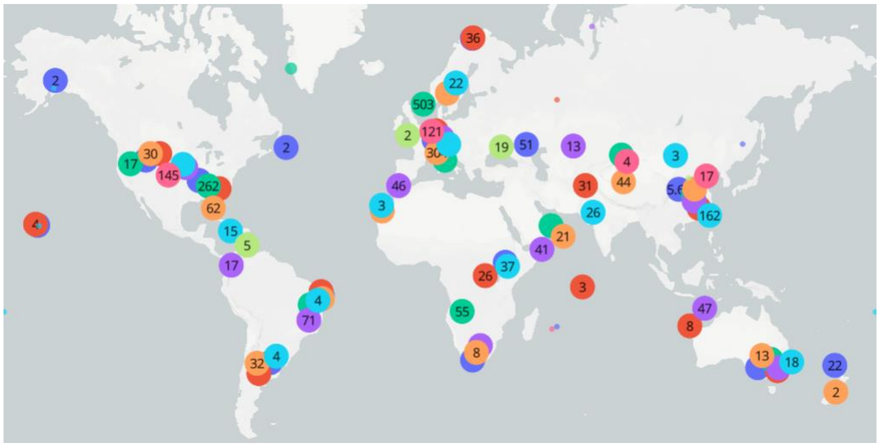


Figure 6: Available clustering in Plotly Mapbox package.

Therefore, our own clustering method was developed. Two semantic zoom levels were implemented, to increase performance and maintain fine control over the location of the aggregate circles. When zoomed out, wind farm project data are aggregated on a country-level (Figure 7). When zoomed in, detailed information of the projects is visualized. Also aggregation on subregion and region level was considered (as the hierarchical information is available in the data), but found to be losing too much information about location and size of individual projects, while not needed for performance.

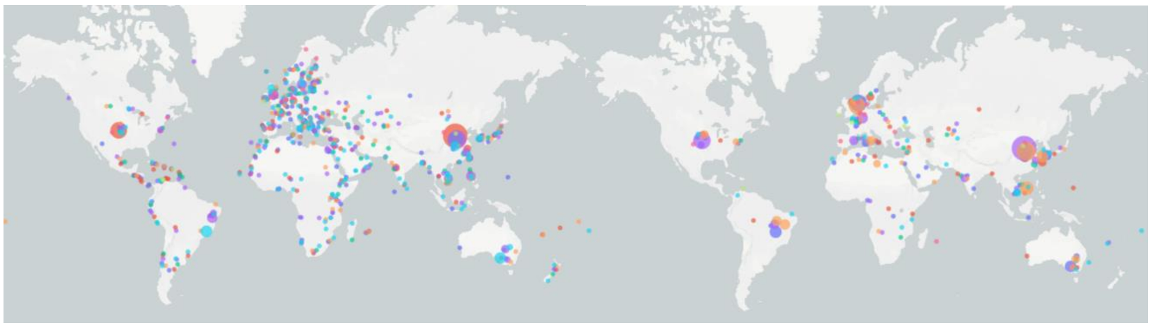


Figure 7: Clustering on country-level (left) and subregion-level (right). .

4.4 Colour choices

A colourblind-safe colour scheme for the different status categories was selected from ColorBrewer 2.0 (https://colorbrewer2.org/#type=qualitative&scheme=Dark2&n=3). The same colours are used for the map and the bar chart.

To optimize the contrast between the coloured markers and the background, a minimalistic light grey map was selected as background map.

4.5 Final product

Our final product is an interactive dashboard of global wind power. It consists of several components which are described below:

1. **Big aggregate numbers (BANs):** On the left side of the map BANs are placed representing the different continents. These BANs display the name and total wind power capacity for each continent. Clicking on the BAN will update the map and bar chart according. For example: clicking on "Africa" will display the wind farms on Africa only on the map and will display the data of the top 20 largest wind farms in Africa on the bar chart as shown in the Figure 8 below.

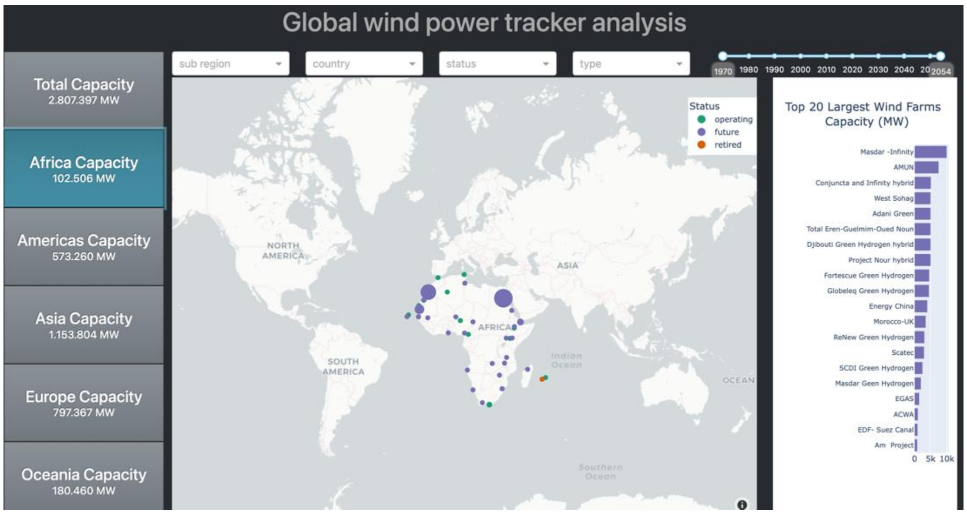


Figure 8 Displaying windfarms according to BAN selection

2. **Filters:** On the top of the main area are the filters which allows user to filter the map according to sub region, country, status and type. The bar chart also updates accordingly. Figure 9 depicts these filters.

- Sub-region: This filter allows the user to filter the map by sub-region.
- Country: This filter allows the user to update the visualization by country.
- Status: This filter allows the user to update the visualization by status (operating, future or retired).
- Type: This filter allows the user to update the visualization by type (onshore or offshore).
- Time Slider: The time slider allows the user to select a specific time range for the data displayed on the map and bar chart. The user can adjust the start and end points of the time range by dragging the handles on the slider. The map and bar chart will then update to display data that falls within the selected time range.

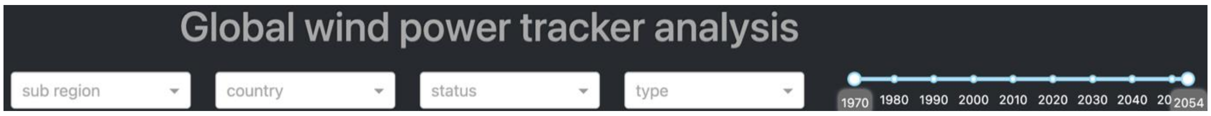


Figure 9 Filters

3. **Map**: The main visualization of this app is the interactive map that visually represents all the wind farms.

- Colour-coding: The map uses colour-coding to represent the status of the wind farms. Each project on the map is represented by a circle on the map and the color of the circle indicates it's status. For example: green is used for the operating status. Figure 10 shows a more detailed picture of how different wind farms are colour-coded according to their status.

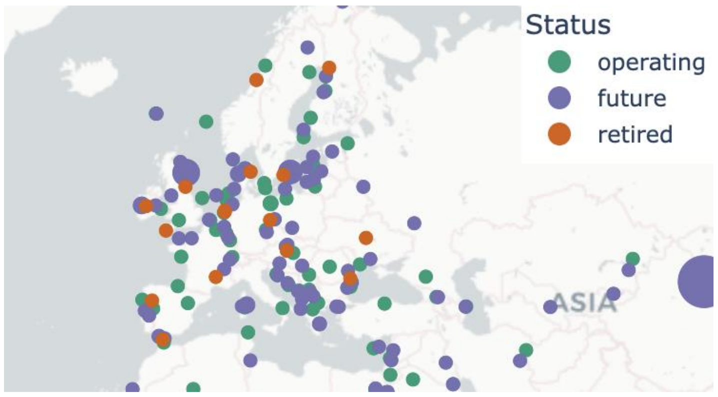
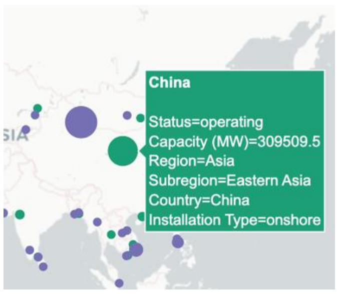


Figure 10 Map colour-coded according to status

- Circle Marker Size: The size of the marker corresponds to the capacity of the wind farm. The bigger the size of the marker, the bigger the capacity of the wind farm. In Figure 10 it can be seen that the circle markers vary in size.
- Zoom: The user can zoom in or zoom out on the map. The initial view shows aggregated information on a country-level. Zooming the map provides more detailed information of the specific wind farm projects.
- Hover Interaction: When the user hovers over the marker, additional information about the windfarm is displayed which is shown in Figure 11.


China

Status=operating
Capacity (MW)=309509.5
Region=Asia
Subregion=Eastern Asia
Country=China
Installation Type=onshore

Figure 11 Hovering over a marker displays detailed information

4. **Bar Chart:** On the right side of the screen, there is a bar chart, displaying the top 20 largest wind farms and their total capacity.

- Colour-coding: Similar to the map, the bar chart is using the same colour-codes according to the status which can be seen in Figure 12.

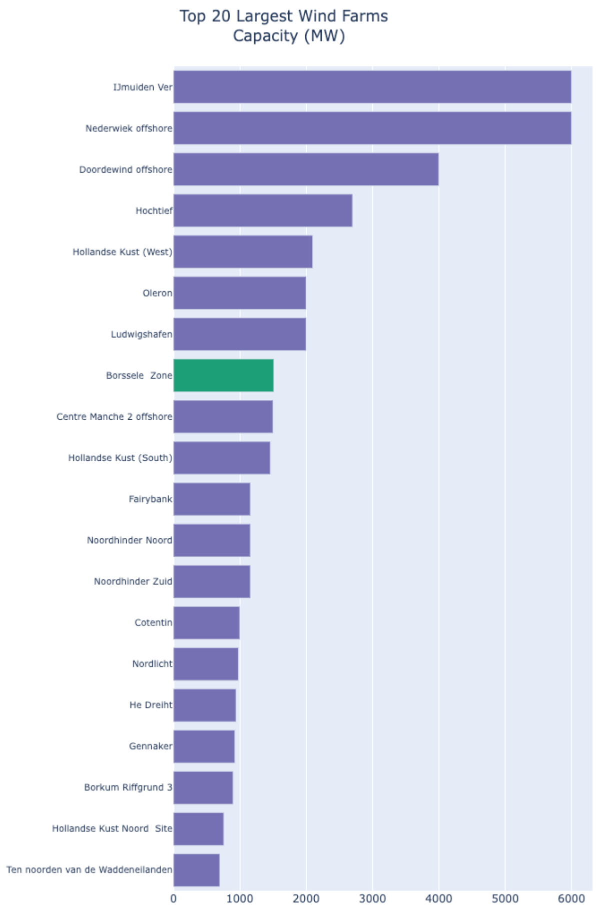


Figure 12 Bar chart colour-coded according to status

- Hover and Click Interaction: Like the map, the bar chart also displays detailed information when it is hovered over by the user which can be seen in Figure 13. Clicking on the bar chart will also center and zoom the map to the location of the selected project.

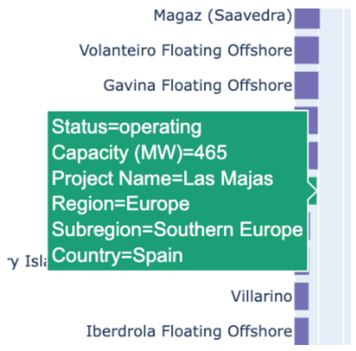


Figure 13 Hovering over the bar chart displays detailed information

5 Evaluation

5.1 Evaluation framework

The evaluation process for the Global Wind Power Tracker Dashboard was designed to assess both usability and functionality. A structured test was conducted followed by an interview and filling up a questionnaire to asses the ability of the end-user to understand and interact with the dashboard.

5.2 Evaluation Setup

1. Team Involvement: Each member of the team conducted one interview.
2. Selection of Canditate: Three canditates from diverse backgrounds participated in the interview. Each canditate was from a different age group and all of them possessd a masters degree.
3. Interview Setup: Four key questions were asked to the canditates. More information about the questions can be found under section 1.4.
4. Questionnaire focus: During the interview a structured questionnaire was filled out focussing on:
    - Understanding the question
    - Ability to answer the question without support
    - Correctness of the answer
    - General Feedback
5. Scoring Methodology:
    1. Understanding the questions:
        - Correct Understanding: 1 point
        - Partial Understanding: 0.5 points
        - Incorrect Understanding: -1 point
    2. Ability to Answer Without Support:
        - Answerwed without support: 1 point
        - Partially answered without support: 0.5 points
        - Could not answer without support: -1 point
    3. Correctness of the answer:
        - Correct Answer: 1 point
        - Partially Correct Answer: 0.5 points
        - Incorrect Answer: -1 point

5.3 Evaluation Results

The diverging stacked bar chart illustrates the scores of all 4 questions asked to the 3 canditates.

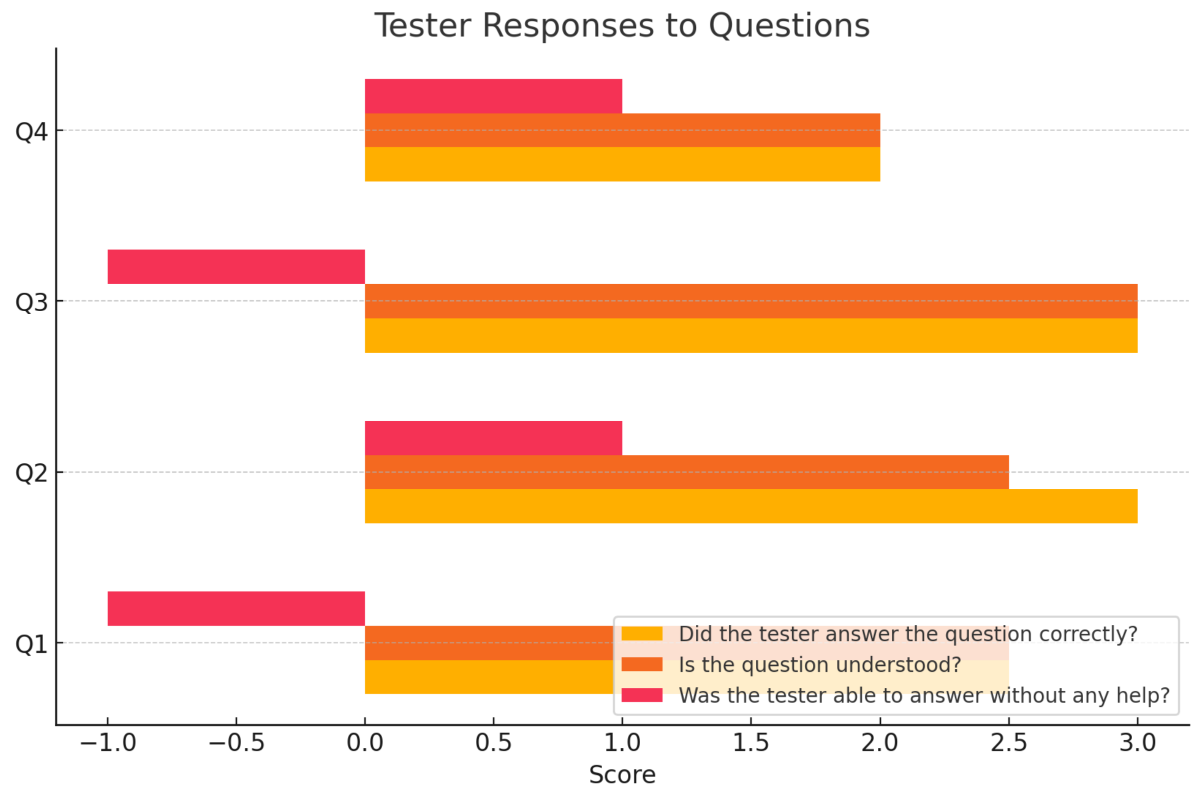


Figure 14 Result of the interviews

Result Analysis:

The results suggested that the question 2 and 4 were mostly understood well and correctly answerd by most candidates. However, question 1 and 3 posed some challenges where the users needed help which resulted the negatve score. Overall, we found positive performance from the users with a little difficulty in answering some questions without support for some candidates.

5.4 Appreciative feedback

- **Visual Appeal:** The dashboard was easy on the eyes of the canditates.
- **Layout:** The intuitive F-shape layout was appreciated by the users.
- **Colour Consistency:** Consistent use of colours for different status was highly appreciated by the users as it helped them to differentiate between operating, future and retired easily.
- **Big Aggregate Numbers (BANs):** Users found the BANs very helpful to extract the total capacity data quickly

5.5 Constructive feedback

- **Empty Filters:** Some users find it confusing when the filters appear empty if the parent levels were not set. For example: They were not able to directly filter a country. To filter a country they had to first filter a sub-region.
- **Slider Tooltip:** One of the user suggested to add a tooltip when hovered over the slider to know what exactly they are filtering.
- **Data Snapshot Timing:** Users suggested that it will be helpful to know when exactly the data wes updated, mainly to understand the “future” status.

6 Team organization

The team established collaboration practices within the first week. This section outlines those agreements.

6.1 Communication

For asynchronous communication, the team relied on a WhatsApp group. For synchronous communication, the team organized eight meetings via Microsoft Teams over the semester. Minutes from these meetings were recorded in OneNote and distributed by email.

6.2 Collaboration

The team decided to use Git for version control and collaborated on the codebase through a public GitHub repository (https://github.com/jorritvm/infovis). Mainline development was performed without specific branching rules, and branches were only created for specific purposes, such as in-depth refactoring or proof-of-concept work for map visualization.

6.3 Planning

The team established and executed a project plan, which is visualized as a Gantt chart in Appendix A. After the mid-term presentation, the team adopted an agile methodology, dividing the remaining four weeks into two sprints. Each sprint included planning and review sessions. During the sprint reviews, bugs were discussed and assigned.

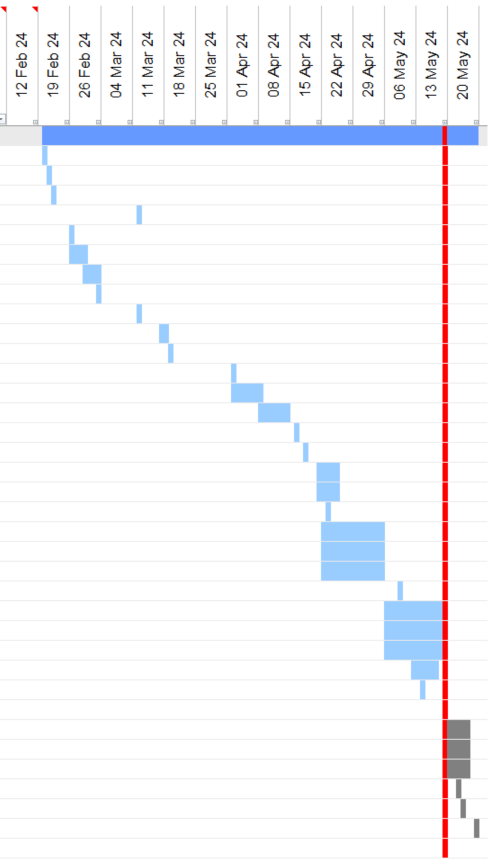
7 Authors

Jorrit Vander Mynsbrugge	0606134	jorrit.jules.vander.mynsbrugge@vub.be
Ruth Vandeputte	0591588	ruth.vandeputte@vub.be
Mishkat Haider Chowdhury	0594966	mishkat.haider.chowdhury@vub.be

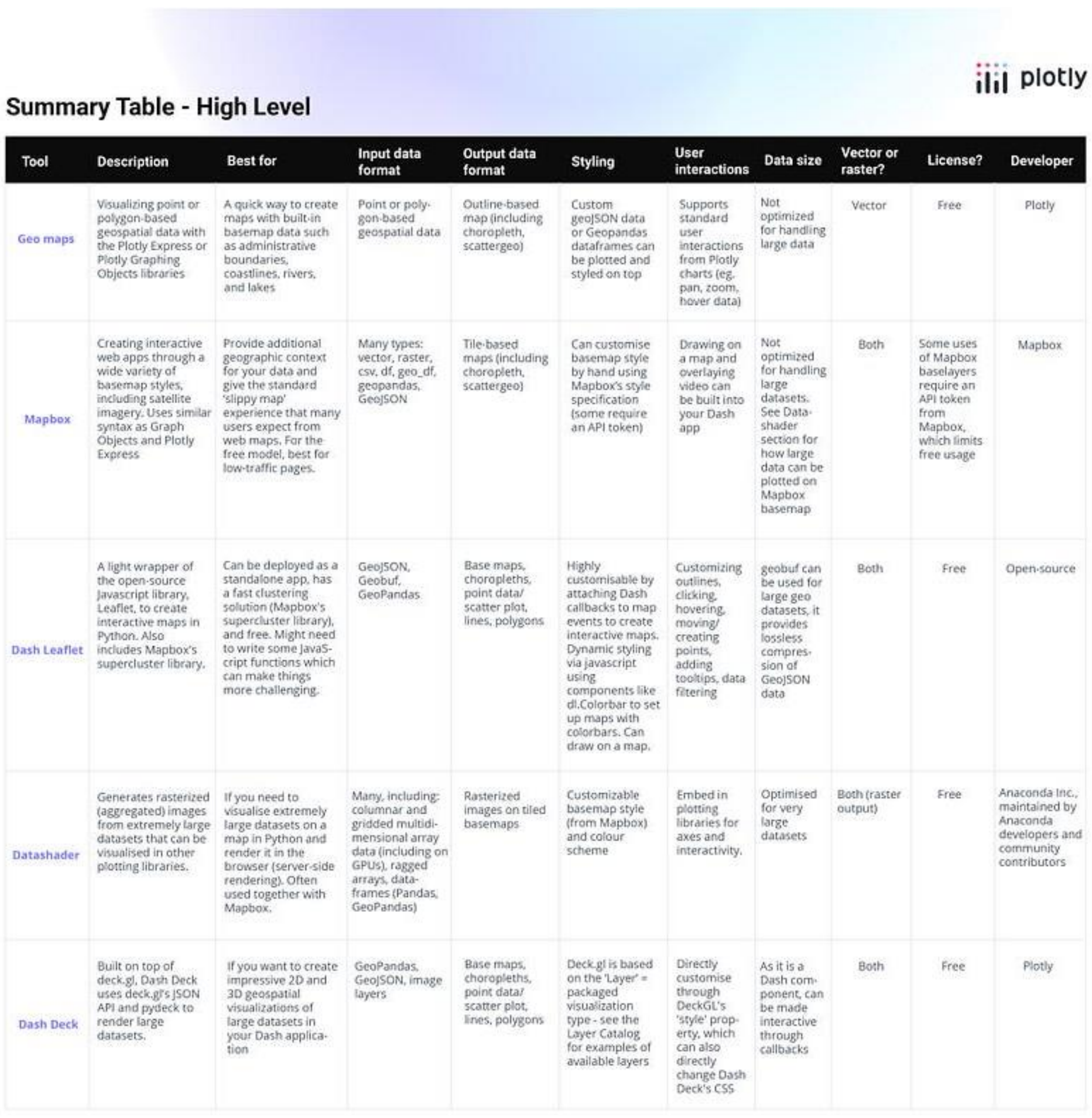
Appendix A Project plan

Infovis global wind power exploration

Task	Wh	Start	End	Duration	%Complete
Taken		Tue 20-02-24	Sun 26-05-24	97	100%
clinic 1: intro project	J+M	Tue 20-02-24	Tue 20-02-24	1	100%
team call: debrief clinic	J+M+R	Wed 21-02-24	Wed 21-02-24	1	100%
setting up team collab channels (github, whatsapp)	J	Thu 22-02-24	Thu 22-02-24	1	100%
setting up dash example code	J+M	Tue 12-03-24	Tue 12-03-24	1	100%
team call: dataset discussion	J+M+R	Mon 26-02-24	Mon 26-02-24	1	100%
drafting dataset doc	J	Mon 26-02-24	Thu 29-02-24	4	100%
review dataset doc	M+R	Thu 29-02-24	Sun 03-03-24	4	100%
dataset doc submission	J	Sun 03-03-24	Sun 03-03-24	1	100%
clinic 2	J+M	Tue 12-03-24	Tue 12-03-24	1	100%
data preprocessing + eda notebook	J	Sun 17-03-24	Mon 18-03-24	2	100%
team call: dataset and framework	J+M+R	Tue 19-03-24	Tue 19-03-24	1	100%
team call: data validation	J+M+R	Tue 02-04-24	Tue 02-04-24	1	100%
prepare interim presentation	J	Tue 02-04-24	Mon 08-04-24	7	100%
review interim presentation	M+R	Mon 08-04-24	Sun 14-04-24	7	100%
team call: prepare interim presentation	J+M+R	Tue 16-04-24	Tue 16-04-24	1	100%
interim presentation	J+M+R	Thu 18-04-24	Thu 18-04-24	1	100%
unavailable (abroad)	J	Sun 21-04-24	Thu 25-04-24	5	100%
unavailable (abroad)	R	Sun 21-04-24	Thu 25-04-24	5	100%
clinic 3	M	Tue 23-04-24	Tue 23-04-24	1	100%
sprint 1: data processing / aggregation / filters / interactivi	J	Mon 22-04-24	Sun 05-05-24	14	100%
sprint 1: bar charts, BAN, popups	M	Mon 22-04-24	Sun 05-05-24	14	100%
sprint 1: map plot (framework selection, notebook impl)	R	Mon 22-04-24	Sun 05-05-24	14	100%
team call: sprint review	J+M+R	Thu 09-05-24	Thu 09-05-24	1	100%
sprint 2: cross validation + tweaking + bugfixing	J	Mon 06-05-24	Sun 19-05-24	14	100%
sprint 2: cross validation + tweaking + bugfixing	M	Mon 06-05-24	Sun 19-05-24	14	100%
sprint 2: cross validation + tweaking + bugfixing	R	Mon 06-05-24	Sun 19-05-24	14	100%
unavailable (abroad)	J	Sun 12-05-24	Fri 17-05-24	6	100%
clinic 4	M+R	Tue 14-05-24	Tue 14-05-24	1	100%
team call: sprint review + final actions	J+M+R	Sun 19-05-24	Sun 19-05-24	1	100%
final ppt + report drafting	J	Mon 20-05-24	Fri 24-05-24	5	0%
final ppt + report drafting	M	Mon 20-05-24	Fri 24-05-24	5	0%
final ppt + report drafting	R	Mon 20-05-24	Fri 24-05-24	5	0%
team call: consolidation	J+M+R	Wed 22-05-24	Wed 22-05-24	1	0%
final presentation	J+M+R	Thu 23-05-24	Thu 23-05-24	1	0%
deadline final report	J+M+R	Sun 26-05-24	Sun 26-05-24	1	0%



Appendix B   Map visualization comparison table



Summary Table - High Level

Tool	Description	Best for	Input data format	Output data format	Styling	User interactions	Data size	Vector or raster?	License?	Developer
Geo maps	Visualizing point or polygon-based geospatial data with the Plotly Express or Plotly Graphing Objects libraries	A quick way to create maps with built-in basemap data such as administrative boundaries, coastlines, rivers, and lakes	Point or polygon-based geospatial data	Outline-based map (including choropleth, scattergeo)	Custom geoJSON data or Geopandas dataframes can be plotted and styled on top	Supports standard user interactions from Plotly charts (eg. pan, zoom, hover data)	Not optimized for handling large data	Vector	Free	Plotly
Mapbox	Creating interactive web apps through a wide variety of basemap styles, including satellite imagery. Uses similar syntax as Graph Objects and Plotly Express	Provide additional geographic context for your data and give the standard 'slippy map' experience that many users expect from web maps. For the free model, best for low-traffic pages.	Many types: vector, raster, csv, df, geo_df, geopandas, GeoJSON	Tile-based maps (including choropleth, scattergeo)	Can customise basemap style by hand using Mapbox's style specification (some require an API token)	Drawing on a map and overlaying video can be built into your Dash app	Not optimized for handling large datasets. See Datashader section for how large data can be plotted on Mapbox basemap	Both	Some uses of Mapbox baselayers require an API token from Mapbox, which limits free usage	Mapbox
Dash Leaflet	A light wrapper of the open-source Javascript library, Leaflet, to create interactive maps in Python. Also includes Mapbox's supercluster library.	Can be deployed as a standalone app, has a fast clustering solution (Mapbox's supercluster library), and free. Might need to write some JavaScript functions which can make things more challenging.	GeoJSON, Geobuf, GeoPandas	Base maps, choropleths, point data/ scatter plot, lines, polygons	Highly customisable by attaching Dash callbacks to map events to create interactive maps. Dynamic styling via javascript using components like dl.Colorbar to set up maps with colorbars. Can draw on a map.	Customizing outlines, clicking, hovering, moving/ creating points, adding tooltips, data filtering	geobuf can be used for large geo datasets, it provides lossless compression of GeoJSON data	Both	Free	Open-source
Datashader	Generates rasterized (aggregated) images from extremely large datasets that can be visualised in other plotting libraries.	If you need to visualise extremely large datasets on a map in Python and render it in the browser (server-side rendering). Often used together with Mapbox.	Many, including: columnar and gridded multidimensional array data (including on GPUs), ragged arrays, dataframes (Pandas, GeoPandas)	Rasterized images on tiled basemaps	Customizable basemap style (from Mapbox) and colour scheme	Embed in plotting libraries for axes and interactivity.	Optimised for very large datasets	Both (raster output)	Free	Anaconda Inc., maintained by Anaconda developers and community contributors
Dash Deck	Built on top of deck.gl, Dash Deck uses deck.gl's JSON API and pydeck to render large datasets.	If you want to create impressive 2D and 3D geospatial visualizations of large datasets in your Dash application	GeoPandas, GeoJSON, image layers	Base maps, choropleths, point data/ scatter plot, lines, polygons	Deck.gl is based on the 'Layer' = packaged visualization type - see the Layer Catalog for examples of available layers	Directly customise through DeckGL's 'style' property, which can also directly change Dash Deck's CSS	As it is a Dash component, can be made interactive through callbacks	Both	Free	Plotly

Appendix C How to deploy?

Setting up your local copy

1. Clone the repo
   ```
   git clone https://github.com/jorritvm/infovis.git
   ```
   or unzip the provided archive
2. Make sure you're using python 3.11
3. Navigate to your project directory
   ```
   cd infovis
   ```
4. Set up a virtual environment
   ```
   python -m venv venv
   ```
5. Restore the dependencies in that virtual environment
   ```
   pip install -r requirements.txt
   ```
6. Copy `.env.template` to `.env` and configure the required secrets
   (optional if you started from the zip archive)

Running the dashboard from the CLI

1. Activate the venv
   ```
   venv\Scripts\activate
   ```
2. Navigate to the app directory:
   ```
   cd app
   ```
3. Run the dash app:
   ```
   python app.py
   ```
4. Explore the app using your web browser.

Appendix D   Repository structure

```
infovis
¦   .env                                    .env file read by app
¦   .env.template                           .env template file
¦   README.md
¦   requirements.txt
¦
+---app
¦   ¦   app.py                              main dashboard app file
¦   ¦
¦   +---assets
¦   ¦   +---css
¦   ¦   ¦       styles.css                  custom css definitions
¦   ¦
¦   +---callbacks                           callback handlers grouped by
¦   ¦   ¦   cb_bar_chart.py                 output
¦   ¦   ¦   cb_continent.py
¦   ¦   ¦   cb_country_filter.py
¦   ¦   ¦   cb_map.py
¦   ¦   ¦   cb_status_type.py
¦   ¦   ¦   cb_sub_region.py
¦   ¦
¦   +---components                          visual components of the
¦   ¦   ¦   filters.py                      dashboard like bootstrap
¦   ¦   ¦   visualisations.py               dropdown filters and visuals
¦   ¦
¦   +---utils                               utility functions like
¦       ¦   utils.py                        filtering
¦
+---data
¦   +---analysis                            folder containing manual
¦   ¦       ...                             data analysis
¦   ¦
¦   +---clean                               folder containing cleaned
¦   ¦       ...                             data in parquet & xlsx
¦   ¦
¦   +---etl                                 folder containing the ETL
¦   ¦       process_gwpt_data.ipynb         pipeline in jupyter
¦   ¦
¦   +---raw                                 folder containing the raw
¦           Global-Wind-Power-              input data
Tracker-December-2023.xlsx
¦
+---doc                                     self explanatory
¦   +---assignment                          documentation folder
¦   +---dataset_selection
¦   +---evaluation
¦   +---final_presentation
¦   +---midterm_presentation
¦   +---mockup
¦   +---planning
¦   +---report
¦   +---video
¦                                           folder containing some map
+---experimentation                         experiments
¦       ...
```