covid19-data-analyzer Documentation

Release 0.1.0

Sebastian Weigand

Jan 11, 2021

CONTENTS:

CHAPTER

ONE

COVID19-DATA-ANALYZER

Especially in times of crisis and panic, with many different opinions on the topic, people should try to derive their own conclusion. Luckily we live in a time, where the skills and tools to analyse data are open to everyone. Thus this little project tries to provide you with the data and a small toolset to understand, what the current state of the covid19 pandemic is right now.

The code of this repository is the basis for istcoronaexponentiell.de, where this projects interactive dashboard is hosted.

CHAPTER

TWO

USAGE

1. Clone or download this repository
2. Install the dependencies

```
pip install -r requirements.txt
pip install -e .
```

3. Analyse the data yourself using the jupyter notebook and play with the dashboard
    - Open the jupyter notebook

    ```
    jupyter lab
    ```

    - Start the dashboard server

    ```
    covid19_dashboard
    ```

CHAPTER

THREE

INSTALLATION

3.1 From sources

The sources for covid19-data-analyser can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/s-weigand/covid19-data-analyzer
```

Or download the tarball:

```
$ curl -OJL https://github.com/s-weigand/covid19-data-analyzer/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

FOUR

USED DATA SOURCES

The data used by this project, is scraped from other sources and transformed in a uniform pattern, which than can be used by the analysis code. To be transparent about where the data originates from and also give props to the sources we list them here.

4.1 funkeinteraktiv (Funke Media Gruppe)

The funkeinteraktiv datasets originate from the funkeinteraktiv API, which is used by morgenpost Corona Monitor and very detailed for germany (county resolution). Since it provides labels in german as well as english, we thought it would be best to split it two different datasets for the german and international audience.

4.2 JHU (Johns Hopkins University)

The JHU dataset comes directly from the JHU github repository, which provides the data for the JHU dashboard.

CHAPTER

FIVE

USED FIT MODELS

The currently implemented fit models are builtin models from lmfit. To understand what the parameters stand for in the actual equations, you can have a look at the documentation given for the models themselves.

5.1 Exponential curve

class ExponentialModel(*independent_vars=['x'], prefix='', nan_policy='raise', **kwargs*)

A model based on an exponential decay function (see https://en.wikipedia.org/wiki/Exponential_decay) with two Parameters: `amplitude` (A), and `decay` (τ), in:

$$f(x; A, \tau) = Ae^{-x/\tau}$$

Parameters

- **independent_vars** (*['x']*) – Arguments to func that are independent variables.
- **prefix** (*str, optional*) – String to prepend to parameter names, needed to add two Models that have parameter names in common.
- **nan_policy** (*str, optional*) – How to handle NaN and missing values in data. Must be one of: 'raise' (default), 'propagate', or 'omit'. See Notes below.
- ****kwargs** (*optional*) – Keyword arguments to pass to `Model`.

Notes

1. nan_policy sets what to do when a NaN or missing value is seen in the data. Should be one of:
   - 'raise' : Raise a ValueError (default)
   - 'propagate' : do nothing
   - 'omit' : drop missing data

5.2 Logistic curve

class StepModel(*independent_vars=['x'], prefix='', nan_policy='raise', form='linear', **kwargs*)

A model based on a Step function, with three Parameters: `amplitude` (A), `center` (μ) and `sigma` (σ).

There are four choices for `form`:

- `linear` (the default)
- `atan` or `arctan` for an arc-tangent function
- `erf` for an error function
- `logistic` for a logistic function (see https://en.wikipedia.org/wiki/Logistic_function)

The step function starts with a value 0, and ends with a value of A rising to $A/2$ at μ, with σ setting the characteristic width. The functional forms are defined as:

$$
\begin{aligned}
f(x; A, \mu, \sigma, \text{form} = \text{'linear'}) &= A \min[1, \max(0, \alpha)] \\
f(x; A, \mu, \sigma, \text{form} = \text{'arctan'}) &= A[1/2 + \arctan(\alpha)/\pi] \\
f(x; A, \mu, \sigma, \text{form} = \text{'erf'}) &= A[1 + \text{erf}(\alpha)]/2 \\
f(x; A, \mu, \sigma, \text{form} = \text{'logistic'}) &= A[1 - \frac{1}{1 + e^{\alpha}}]
\end{aligned}
$$

where $\alpha = (x - \mu)/\sigma$.

Parameters

- **independent_vars** (*['x']*) – Arguments to func that are independent variables.
- **prefix** (*str, optional*) – String to prepend to parameter names, needed to add two Models that have parameter names in common.
- **nan_policy** (*str, optional*) – How to handle NaN and missing values in data. Must be one of: 'raise' (default), 'propagate', or 'omit'. See Notes below.
- ****kwargs** (*optional*) – Keyword arguments to pass to `Model`.

Notes

1. nan_policy sets what to do when a NaN or missing value is seen in the data. Should be one of:

- 'raise' : Raise a ValueError (default)
- 'propagate' : do nothing
- 'omit' : drop missing data

CHAPTER

SIX

PYTHON SOURCE DOCUMENTATION

Documentation of the Python source code. The auto generated Documentation mainly contains methods and attributes which are inherited from the corresponding classes of *django-cms*, but since the actual source code is very short you can just click the *[source]* link on top of each class.

covid19_data_analyzer.dashboard
covid19_data_analyzer.data_functions

6.1 dashboard

6.1.1 Submodules

covid19_data_analyzer.dashboard.app
covid19_data_analyzer.dashboard.components
covid19_data_analyzer.dashboard.index
covid19_data_analyzer.dashboard.utils

app

Submodules

—

components

Submodules

covid19_data_analyzer.dashboard.components.data_plots
covid19_data_analyzer.dashboard.components.data_selection
covid19_data_analyzer.dashboard.components.download_area
covid19_data_analyzer.dashboard.components.footer

data_plots

Submodules

—

Functions

Summary

update_data_plot
update_fit_param_table
update_growth_plot
update_growth_rate_plot

update_data_plot

update_data_plot(*data_source*, *parent_regions*, *regions*, *subsets*, *plot_settings*, *fit_model*)

update_fit_param_table

update_fit_param_table(*data_source*, *parent_regions*, *regions*, *subsets*, *plot_settings*, *fit_model*)

update_growth_plot

update_growth_plot(*data_source*, *parent_regions*, *regions*, *subsets*, *plot_settings*, *fit_model*)

update_growth_rate_plot

update_growth_rate_plot(*data_source*, *parent_regions*, *regions*, *subsets*, *plot_settings*, *fit_model*)

data_selection

Submodules

Functions

Summary

update_parent_regions
update_regions
update_subsets

update_parent_regions

update_parent_regions(*data_source*)

update_regions

update_regions(*data_source*, *values*)

update_subsets

update_subsets(*data_source*)

download_area

Submodules

Functions

Summary

download_data
update_download_link

download_data

download_data()

update_download_link

update_download_link(*data_source*, *dl_format*)

footer

Submodules

---

index

Submodules

Functions

Summary

run_dashboard_server

run_dashboard_server

run_dashboard_server()

utils

Submodules

covid19_data_analyzer.dashboard.utils.controls
covid19_data_analyzer.dashboard.utils.data_loader
covid19_data_analyzer.dashboard.utils.download
covid19_data_analyzer.dashboard.utils.plot

controls

Submodules

Functions

Summary

generate_dropdown_options	Generates Dropdown options from a list
generate_selector	Generates a selector based on values are in df_columns
get_available_subsets	Returns a list of subsets which are available on the dataset

generate_dropdown_options

generate_dropdown_options(*values: Iterable[str]*)

Generates Dropdown options from a list

Parameters **values** (*Iterable[str]*) – Iterable with the option values

Returns Options for the Dropdown

Return type Dict

generate_selector

generate_selector(*df_column: pandas.core.series.Series, values: Iterable[str]*)

Generates a selector based on values are in df_columns

Parameters

- **df_column** (*pd.Series*) – DataFrame colum which values should be compared to the values of values
- **values** (*Iterable[str]*) – Iterable of column values

Returns Boolean pd.Series which can be used as selector for the dataframe

Return type pd.Series

get_available_subsets

get_available_subsets(*covid19_data: pandas.core.frame.DataFrame*) → list

Returns a list of subsets which are available on the dataset

Parameters **covid19_data** (*pd.DataFrame*) – [description]

Returns List of available subsets

Return type list

data_loader

Submodules

—

Functions

Summary

get_fit_data_dict

get_fit_data_dict

get_fit_data_dict(*kind: str*) → Dict[str, Dict[str, pandas.core.frame.DataFrame]]

download

Submodules

—

Functions

Summary

generate_download_buffer	Transdorms a dataframe to a given fileformat as buffer, which can be used for flask.send_file

generate_download_buffer

generate_download_buffer(*data_source: str*, *file_format: str*) → Dict

Transdorms a dataframe to a given fileformat as buffer, which can be used for flask.send_file

Parameters

- **data_source** (*str*) – Name of the data source
- **file_format** (*"csv" | "xls*) – Format the file should be downloaded in

Returns dict containing the buffer, mimetype and filename

Return type Dict

plot

Submodules

—

Functions

Summary

create_trace	Function to generate the Plot trace of a single dataset, with the appropriate style (raw data -> marker, fit -> line)
generate_figure	Creates the Figure data for a plot
generate_plot_sub_data	Function to generate the data used to plot raw data and/or its fit
plotly_color_cycler	Colorcycler on base of the plotly default colors, to ensure that raw data and fits have the same color

create_trace

create_trace(*data: pandas.core.frame.DataFrame*, *region: str*, *subset: str*, *color: str*, *is_fit: bool = False*) → Dict

Function to generate the Plot trace of a single dataset, with the appropriate style (raw data -> marker, fit -> line)

Parameters

- **data** (*pd.DataFrame*) – Data which should be plotted
- **region** (*str*) – region name of the data, needed to generate the legend
- **subset** (*str*) – subset name of the data, needed to generate the legend
- **color** (*str*) – color of the trace, needed so raw data and fits have the same color
- **is_fit** (*bool, optional*) – Whether or not the data are from a fit, by default False

Returns Trace data to generate a plot with dash

Return type Dict

generate_figure

generate_figure(*data_source: str, parent_regions: Iterable[str], regions: Iterable[str], title: str, y_title: str, data_transform_fuction: Callable = None, subsets: Iterable[str] = ['confirmed'], plot_settings: Iterable[str] = [], fit_model: str = None*) → Dict

Creates the Figure data for a plot

Parameters

- **data_source** (*str*) – name of the data source
- **parent_regions** (*Iterable[str]*) – names of the parent_regions which should be plotted
- **regions** (*Iterable[str]*) – names of the regions which should be plotted
- **title** (*str*) – Title of the plot
- **y_title** (*str*) – caption of the y-axis
- **data_transform_fuction** (*Callable, optional*) – function which should be applied on the data, by default None
- **subsets** (*Iterable[str], optional*) – subsets of the data which should be ploted, by default ["confirmed"]
- **plot_settings** (*Iterable[str], optional*) – settings which should be used for the plot, by default []
- **fit_model** (*str, optional*) – name of the fit model which should be used, by default None

Returns figure data of the plot

Return type Dict

generate_plot_sub_data

generate_plot_sub_data(*raw_data: pandas.core.frame.DataFrame, region: str, subset: str, color: str, hide_raw_data: bool = False, fit_data: pandas.core.frame.DataFrame = None*) → List[Dict]

Function to generate the data used to plot raw data and/or its fit

Parameters

- **raw_data** (*pd.DataFrame*) – Actual case data
- **region** (*str*) – region name of the data, needed to generate the legend
- **subset** (*str*) – subset name of the data, needed to generate the legend
- **color** (*str*) – color of the trace, needed so raw data and fits have the same color
- **hide_raw_data** (*bool, optional*) – Whether or not to hide the raw data, by default False
- **fit_data** (*pd.DataFrame, optional*) – Data used to plot a fit, by default None

Returns List of traces which should be ploted

Return type List[Dict]

plotly_color_cycler

plotly_color_cycler(*plot_index: int*) → str
Colorcycler on base of the plotly default colors, to ensure that raw data and fits have the same color
Parameters **plot_index** (*int*) – index of the current plot
Returns Color to be used for a plot
Return type str

6.2 data_functions

6.2.1 Submodules

covid19_data_analyzer.data_functions.analysis
covid19_data_analyzer.data_functions.data_utils
covid19_data_analyzer.data_functions.scrapers

analysis

Submodules

covid19_data_analyzer.data_functions.analysis.exponential_curve
covid19_data_analyzer.data_functions.analysis.factory_functions
covid19_data_analyzer.data_functions.analysis.logistic_curve

exponential_curve

Submodules

—

Functions

Summary

batch_fit_exponential_curve	Implementation of batch_fit_model, for the exponential curve model.
fit_data_exponential_curve	Implementation of fit_data_model, with setting specific to the exponential curve model

batch_fit_exponential_curve

batch_fit_exponential_curve()

Implementation of batch_fit_model, for the exponential curve model.

See also:

fit_data_exponential_curve(), batch_fit_model()

fit_data_exponential_curve

fit_data_exponential_curve(*covid19_data: pandas.core.frame.DataFrame*, *parent_region: str*, *region: str*, *data_set: str = 'confirmed'*) → Dict[str, Union[lmfit.model.ModelResult, pandas.core.frame.DataFrame]]

Implementation of fit_data_model, with setting specific to the exponential curve model

Parameters

- **covid19_data** (*pd.DataFrame*) – Full covid19 data from a data_source
- **parent_region** (*str*) – parent_region of the data which should be fitted, needs to be in covid19_region_data.parent_region
- **region** (*str*) – region which data should be fired, needs to be in covid19_region_data.region
- **data_set** (*str, optional*) – which subdata schold be fitted, need to be of value [“confirmed”, “recovered”, deaths], by default “confirmed”

Returns

Result dict with keys “model_result” and “plot_data”.

model_result: lmfit.model.ModelResult result of the fit, with optimized parameters

plot_data: pd.DataFrame Same as covid19_region_data, but with an resetted index and and added fir result

Return type Dict[str, Union[lmfit.model.ModelResult, pd.DataFrame]]

See also:

fit_data_model()

factory_functions

Submodules

—

Functions

Summary

batch_fit_model	Generic function to fit a fit_function to the data of all data sources and save them to file
calc_extrema	Calculates the supremum and infimum of a given function func, with the parameters and their errors given by param_df, over the values of x.
fit_data_model	Generic function to fit lmfit.Model models, onto a regional subset covid data.
fit_regions	Function to fit all regions of a covid dataset
fit_subsets	Function to fit the subsets of a regional covid data
predict_trend	Generic function to predict a trend from fitted data

batch_fit_model

batch_fit_model (*fit_function: Callable*, *model_name: str*, *fit_func_kwargs: dict = {}*) → None

Generic function to fit a fit_function to the data of all data sources and save them to file

Parameters

- **fit_function** (*Callable*) – Implementation of a model with fit_data_model
- **model_name** (*str*) – Name of the model which is fitted, used to generate the path
- **fit_func_kwargs** (*dict, optional*) – Additional kwargs passed to fit_function, by default {}

See also:

fit_subsets(), *fit_regions()*

calc_extrema

calc_extrema(*x: numpy.ndarray*, *func: Callable*, *param_df: pandas.core.frame.DataFrame*, *func_options: dict = {}*, *brute_force_extrema: bool = False*) → Tuple[numpy.ndarray]

Calculates the supremum and infimum of a given function func, with the parameters and their errors given by param_df, over the values of x.

Parameters

- **x** (*np.ndarray*) – Values the supremum and infimum should be calculated over
- **func** (*Callable*) – Functions used to calculate the supremum and infimum
- **param_df** (*pd.DataFrame*) – DataFrame with parameters and errors
- **func_options** (*dict, optional*) – options for func, by default {}
- **brute_force_extrema** (*bool, optional*) – Whether or not to calculate supremum and infimum from all permutations of adding and subtracting the errors from the parameters. For some functions, i.e. the logistic curve, this is needed, since simply adding or subtracting the errors from the parameter can lead to supremum and/or infimum to cross the result with the exact parameters., by default False

Returns supremum, infimum

Return type Tuple[np.ndarray]

fit_data_model

fit_data_model(*covid19_region_data: pandas.core.frame.DataFrame*, *model: lmfit.model.Model*, *data_set: str = 'confirmed'*, *init_params: dict = {}*, *free_var_name: str = 'x'*) → Dict[str, Union[lmfit.model.ModelResult, pandas.core.frame.DataFrame]]

Generic function to fit lmfit.Model models, onto a regional subset covid data.

Parameters

- **covid19_region_data** (*pd.DataFrame*) – covid19 DataFrame for one region
- **model** (*lmfit.Model*) – [description]
- **data_set** (*str, optional*) – which subdata schold be fitted, need to be of value [“confirmed” | “recovered” | deaths], by default “confirmed”
- **init_params** (*dict, optional*) – initial parameters for a fit, they depend on the model, by default {}
- **free_var_name** (*str, optional*) – name of the free variable used by the model, by default “x”

Returns

Result dict with keys “model_result” and “plot_data”.

model_result: lmfit.model.ModelResult result of the fit, with optimized parameters

plot_data: pd.DataFrame Same as covid19_region_data, but with an resetted index and and added fir result

Return type Dict[str, Union[lmfit.model.ModelResult, pd.DataFrame]]

fit_regions

fit_regions(*covid19_data: pandas.core.frame.DataFrame*, *fit_function: Callable*, *data_source: str*, *fit_func_kwargs: dict = {}*) → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Function to fit all regions of a covid dataset

Parameters

- **covid19_data** (*pd.DataFrame*) – Full covid19 data from a data_source
- **fit_function** (*Callable*) – Implementation of a model with fit_data_model
- **data_source** (*str*) – name of the data source, only needed to print debug information
- **fit_func_kwargs** (*dict, optional*) – Additional kwargs passed to fit_function, by default {}

Returns

fitted_plot_data, fitted_param_results

fitted_plot_data: actual fitted data, which can be used for plotting

fitted_param_results: fit parameters

Return type Tuple[pd.DataFrame, pd.DataFrame]

See also:

fit_subsets(), *batch_fit_model()*

fit_subsets

fit_subsets(*fit_function: Callable*, *covid19_data: pandas.core.frame.DataFrame*, *row: pandas.core.series.Series*, *subsets: Iterable*, *data_source: str*, *fit_func_kwargs: dict = {}*) → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Function to fit the subsets of a regional covid data

Parameters

- **fit_function** (*Callable*) – Implementation of a model with fit_data_model
- **covid19_data** (*pd.DataFrame*) – Full covid19 data from a data_source
- **row** (*pd.Series*) – Row of of a dataframe only containing unique value pairs for "region" and "parent_region"
- **subsets** (*Iterable*) – Iterable of subset names
- **data_source** (*str*) – name of the data source, only needed to print debug information
- **fit_func_kwargs** (*dict, optional*) – Additional kwargs passed to fit_function, by default {}

Returns

fitted_region_plot_data, fitted_param_subset

fitted_region_plot_data: actual fitted data, which can be used for plotting
fitted_param_subset: fit parameters for a region
Return type Tuple[pd.DataFrame, pd.DataFrame]
See also:

fit_data_model(), *fit_regions()*, *batch_fit_model()*

predict_trend

predict_trend(*fit_result: Dict[str, Union[lmfit.model.ModelResult, pandas.core.frame.DataFrame]], days_to_predict: int = 30, func_options: dict = {}, param_inverted_stderr: Iterable[str] = [], brute_force_extrema: bool = False*) → pandas.core.frame.DataFrame

Generic function to predict a trend from fitted data

Parameters

- **fit_result** (*Dict[str, Union[lmfit.model.ModelResult, pd.DataFrame]]*) – result of fit_data_model or its implementation
- **days_to_predict** (*int, optional*) – number of days to predict a trend for, by default 30
- **func_options** (*dict, optional*) – options for the function of model, by default {}
- **param_inverted_stderr** (*Iterable[str], optional*) – iterable of parameternames with should be inverted, to calculate the extrema. , by default []
- **brute_force_extrema** (*bool, optional*) – Whether or not to calculate supremum and infimum from all permutations of adding and subtracting the errors from the parameters. For some functions, i.e. the logistic curve, this is needed, since simply adding or subtracting the errors from the parameter can lead to supremum and/or infimum to cross the result with the exact parameters., by default False

Returns

DataFrame with columns “date”, “trend”, “trend_sup” and “trend_inf”
date: pd.Datetime date of the values
trend: float predicted trend
trend_sup: float supremum of the trend
trend_inf: float infimum of the trend
Return type pd.DataFrame
See also:

fit_data_model(), *calc_extrema()*, params_to_df()

logistic_curve

Submodules

—

Functions

Summary

batch_fit_logistic_curve	Implementation of batch_fit_model, for the logistic curve model.
fit_data_logistic_curve	Implementation of fit_data_model, with setting specific to the logistic curve model
predict_trend_logistic_curve	Implementation of fit_data_model, with setting specific to the logistic curve model

batch_fit_logistic_curve

batch_fit_logistic_curve()
Implementation of batch_fit_model, for the logistic curve model.

See also:

fit_data_logistic_curve(), batch_fit_model()

fit_data_logistic_curve

fit_data_logistic_curve(*covid19_data: pandas.core.frame.DataFrame*, *parent_region: str*, *region: str*, *data_set: str = 'confirmed'*, *sigma: Union[int, float] = 5*) → Dict[str, Union[lmfit.model.ModelResult, pandas.core.frame.DataFrame]]
Implementation of fit_data_model, with setting specific to the logistic curve model

Parameters

- **covid19_data** (*pd.DataFrame*) – Full covid19 data from a data_source
- **parent_region** (*str*) – parent_region of the data which should be fitted, needs to be in covid19_region_data.parent_region
- **region** (*str*) – region which data should be fitted, needs to be in covid19_region_data.region

- **data_set** (*str, optional*) – which subdata schold be fitted, need to be of value ["confirmed", "recovered", deaths], by default "confirmed"
- **sigma** (*int, optional*) – initial value for the parameter 'sigma' of the logistic curve model, by default 14

Returns

Result dict with keys "model_result" and "plot_data".
model_result: lmfit.model.ModelResult result of the fit, with optimized parameters
plot_data: pd.DataFrame Same as covid19_region_data, but with an resetted index and and added fir result

Return type Dict[str, Union[lmfit.model.ModelResult, pd.DataFrame]]

See also:

`fit_data_model()`

predict_trend_logistic_curve

predict_trend_logistic_curve(*fit_result: Dict[str, Union[lmfit.model.ModelResult, pandas.core.frame.DataFrame]], days_to_predict: int = 30*) → pandas.core.frame.DataFrame

Implementation of fit_data_model, with setting specific to the logistic curve model

Parameters

- **fit_result** (*Dict[str, Union[lmfit.model.ModelResult, pd.DataFrame]]*) – result of fit_data_model or its implementation
- **days_to_predict** (*int, optional*) – number of days to predict a trend for, by default 30

Returns

DataFrame with columns "date", "trend", "trend_sup" and "trend_inf"
date: pd.Datetime date of the values
trend: float predicted trend
trend_sup: float supremum of the trend
trend_inf: float infimum of the trend

Return type pd.DataFrame

See also:

`predict_trend()`, `fit_data_model()`, `calc_extrema()`, `params_to_df()`

Functions

Summary

get_fit_data	Convenience function to quickly get the fitted data from the supported sources.

get_fit_data

get_fit_data(*data_source: str*, *model_name='logistic_curve'*, *kind='plot'*) → pandas.core.frame.DataFrame

Convenience function to quickly get the fitted data from the supported sources.

Parameters

- **data_source** (*str*) – Name of the source the fitted data was fetched from.
- **model_name** (*str, optional*) – Name of the model which was used for fitting, by default "logistic_curve"
- **kind** (*str, optional*) – kind of data you want to retrieve, by default "plot"

Returns Plot data or parameters, depending on 'kind'

Return type pd.DataFrame

Raises

- **ValueError** – If the model_name isn't supported
- **ValueError** – If the data_source isn't supported

data_utils

Submodules

—

Functions

Summary

calc_country_total	Calculates the total for each country from the covid_df, where only data for regions was present before
calc_worldwide_total	Calculates the worldwide total.
get_daily_growth	Calculates the daily growth values

Continued on next page

Table 33 – continued from previous page

get_data_path	Returns the Path object of a path in data and creates the parent folders if they don't exist already
get_fit_param_results_row	Returns a row containing all fitted parameters for a region, which can than be combined to a fit param results dataframe
get_growth_rate	Calculates the growth rate values
get_infectious	Calculates the number of still infectious people.
get_shifted_dfs	Helper function to shift the date of the covid data by a given time and gain DataFrames which can be used to calculate the growth and growth rate.
params_to_df	Returns a DataFrame with the values and stderr of the params
params_to_dict	Converts fit result parameters to a dict
translate_funkeinteraktiv_fit_df	Helperfunction to prevent Fitting overhead, which would be caused if the same dataset with de and en region names would be fitted.

calc_country_total

calc_country_total(*covid_df: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

Calculates the total for each country from the covid_df, where only data for regions was present before

Parameters **covid_df** (*pd.DataFrame*) – covid19 DataFrame (needs to be in uniform style)

Returns Dataframe containing the totals for countries, which before only had their regions listed.

Return type pd.DataFrame

calc_worldwide_total

calc_worldwide_total(*covid_df: pandas.core.frame.DataFrame*, *parent_region_label='parent_region'*, *region_label='region'*) → pandas.core.frame.DataFrame

Calculates the worldwide total.

Parameters

- **covid_df** (*pd.DataFrame*) – covid19 DataFrame (needs to be in uniform style)
- **parent_region_label** (*str*) – name of the parent_region column
- **region_label** (*str*) – name of the region column

Returns Dataframe containing the worldwide totals.

Return type pd.DataFrame

get_daily_growth

get_daily_growth(*covid_df: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

Calculates the daily growth values

Parameters **covid_df** (*pd.DataFrame*) – Full covid19 data from a data_source

Returns covid19 DataFrame, with daily growth values instead of totals.

Return type pd.DataFrame

get_data_path

get_data_path(*sub_path: str*) → pathlib.Path

Returns the Path object of a path in data and creates the parent folders if they don't exist already

Parameters **sub_path** (*str*) – subpath in data directory

Returns Path to a file in data

Return type Path

get_fit_param_results_row

get_fit_param_results_row(*region: str, parent_region: str, subset: str, fit_result: Dict[str, Union[lmfit.model.ModelResult, pandas.core.frame.DataFrame]]*) → pandas.core.frame.DataFrame

Returns a row containing all fitted parameters for a region, which can than be combined to a fit param results dataframe

Parameters

- **region** (*str*) – Value of the fitted region
- **parent_region** (*str*) – Parent region of the fitted region
- **subset** (*str*) – Subset of the regions data which was fitted
- **fit_result** (*Dict[str, Union[lmfit.model.ModelResult, pd.DataFrame]]*) – Result of fit_data_model or its implementation

Returns Row of fit param results dataframe, for the fitted region

Return type pd.DataFrame

See also:

covid19_data_analyzer.data_functions.analysis.factory_functions.fit_data_model()

get_growth_rate

get_growth_rate(*covid_df: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

Calculates the growth rate values

Parameters **covid_df** (*pd.DataFrame*) – Full covid19 data from a data_source

Returns covid19 DataFrame, with growth rate values instead of totals.

Return type pd.DataFrame

get_infectious

get_infectious(*covid_df: pandas.core.frame.DataFrame*) → None

Calculates the number of still infectious people. This function uses the mutability of DataFrames, which is why it doesn't have a return value

Parameters **covid_df** (*pd.DataFrame*) – Dataframe containing all covid19 data

get_shifted_dfs

get_shifted_dfs(*covid_df: pandas.core.frame.DataFrame*, *time_shift: Union[int, float] = 1*, *time_shift_unit: str = 'D'*) → Tuple[pandas.core.frame.DataFrame, pandas.core.frame.DataFrame]

Helper function to shift the date of the covid data by a given time and gain DataFrames which can be used to calculate the growth and growth rate.

Parameters

- **covid_df** (*pd.DataFrame*) – Full covid19 data from a data_source
- **time_shift** (*[int,float], optional*) – value by which the time should be shifted, by default 1
- **time_shift_unit** (*str, optional*) – unit of the time shift , by default "D"

Returns shifted and unshifted covid19 data, with date, parent_region and region as index

Return type Tuple[pd.DataFrame, pd.DataFrame]

params_to_df

params_to_df(*params: lmfit.parameter.Parameters*, *param_inverted_stderr: Iterable[str] = []*) → pandas.core.frame.DataFrame

Returns a DataFrame with the values and stderr of the params

Parameters

- **params** (*lmfit.Parameters*) – fit result parameters

- **param_inverted_stderr** (*Iterable[str], optional*) – iterable of parameternames with should be inverted, to calculate the extrema. , by default []

Returns DataFrame with columns “value” and “stderr”, parameternames as index

Return type pd.DataFrame

params_to_dict

params_to_dict(*params: lmfit.parameter.Parameters*, *kind: str = 'values'*) → dict

Converts fit result parameters to a dict

Parameters

- **params** (*lmfit.Parameters*) – fit result parameters
- **kind** (*str, optional*) – [“values”, “stderr”], by default “values”

Returns Dict containing the parameternames as key and the values or stderr as values

Return type dict

translate_funkeinteraktiv_fit_data

translate_funkeinteraktiv_fit_data()

Helperfunction to prevent Fitting overhead, which would be caused if the same dataset with de and en region names would be fitted. Rather than fitting twice, this function simply translates the german region names to the english ones, which were both extracted by ‘get_funkeinteraktiv_data’.

scrapers

Submodules

covid19_data_analyzer.data_functions.scrapers.JHU
covid19_data_analyzer.data_functions.scrapers.funkeinteraktiv

JHU

Submodules

—

Functions

Summary

get_JHU_data	Retrives covid19 data from JHU (Johns Hopkins University) and transforms it to a uniform style https://github.com/CSSEGISandData/COVID-19
get_JHU_data_subset	Retrives covid19 data subset from JHU (Johns Hopkins University) https://github.com/CSSEGISandData/COVID-19

get_JHU_data

get_JHU_data(*update_data: bool = False*) → pandas.core.frame.DataFrame

Retrives covid19 data from JHU (Johns Hopkins University) and transforms it to a uniform style https://github.com/CSSEGISandData/COVID-19

Parameters **update_data** (*bool, optional*) – Whether to fetch updated data or not, if the locally saved data doesn't include today.

Returns Dataframe containing the covid19 data from JHU

Return type pd.DataFrame

See also:

get_JHU_data_subset()

get_JHU_data_subset

get_JHU_data_subset(*subset: str*) → pandas.core.frame.DataFrame

Retrives covid19 data subset from JHU (Johns Hopkins University) https://github.com/CSSEGISandData/COVID-19

Parameters **subset** (*str*) – Name of the subset, currently 'confirmed' or 'deaths' see: https://github.com/CSSEGISandData/COVID-19/issues/1250

Returns Dataframe containing the covid19 data subset from JHU

Return type pd.DataFrame

funkeinteraktiv

Submodules

—

Functions

Summary

get_funkeinteraktiv_data	Retrives covid19 data from morgenpost API, which is used to generate the following website: https://interaktiv.morgenpost.de/corona-virus-karte-infektionen-deutschland-weltweit/
get_funkeinteraktiv_language_data	Helperfunction to select in which language the region and parent_region values should be represented

get_funkeinteraktiv_data

get_funkeinteraktiv_data(*update_data: bool = False*, *language: str = 'de'*) → pandas.core.frame.DataFrame

Retrives covid19 data from morgenpost API, which is used to generate the following website: https://interaktiv.morgenpost.de/corona-virus-karte-infektionen-deutschland-weltweit/

Parameters

- **update_data** (*bool, optional*) – Whether to fetch updated data or not, if the locally saved data doesn't include today.
- **language** (*"de"|"en", optional*) – Language in which the region and parent_region values should be represented

Returns Dataframe containing the covid19 data from morgenpost.de

Return type pd.DataFrame

get_funkeinteraktiv_language_data

get_funkeinteraktiv_language_data(*covid19_data: pandas.core.frame.DataFrame*, *language: str*) → pandas.core.frame.DataFrame

Helperfunction to select in which language the region and parent_region values should be represented

Parameters

- **covid19_data** (*pd.DataFrame*) – covid19 DataFrame from "https://funkeinteraktiv.b-cdn.net/history.v4.csv"

- **language** (*"de" | "en"*) – Language in which the region and parent_region values should be represented

Returns [description]

Return type pd.DataFrame

See also:

get_funkeinteraktiv_data()

Functions

Summary

get_data	Convenience function to quickly get covid19 data from the supported sources.

get_data

get_data(*data_source: str = 'funkeinteraktiv_de'*, *update_data: bool = False*) → pandas.core.frame.DataFrame

Convenience function to quickly get covid19 data from the supported sources.

Parameters

- **data_source** (*"funkeinteraktiv_de" | "funkeinteraktiv_en" | "JHU", optional*) – source from which the data should be fetched, by default "funkeinteraktiv_de"
- **update_data** (*bool, optional*) – Whether to fetch updated data or not, if the locally saved data doesn't include today.

Returns covid19 DataFrame

Return type pd.DataFrame

Raises **ValueError** – If data_source is not supported

CHAPTER

SEVEN

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at https://github.com/s-weigand/covid19-data-analyzer/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

7.1.4 Write Documentation

covid19-data-analyser could always use more documentation, whether as part of the official covid19-data-analyser docs, in docstrings, or even on the web in blog posts, articles, and such.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/s-weigand/covid19-data-analyzer/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute? Here's how to set up *covid19_data_analyser* for local development.

1. Fork the *covid19_data_analyser* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/covid19_data_analyser.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv covid19_data_analyser
$ cd covid19_data_analyser/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8:

```
$ flake8 covid19_data_analyser
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/s-weigand/covid19-data-analyzer/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

G

P

R

S

T

U