Jarkus Analysis Toolbox

Release 0.0

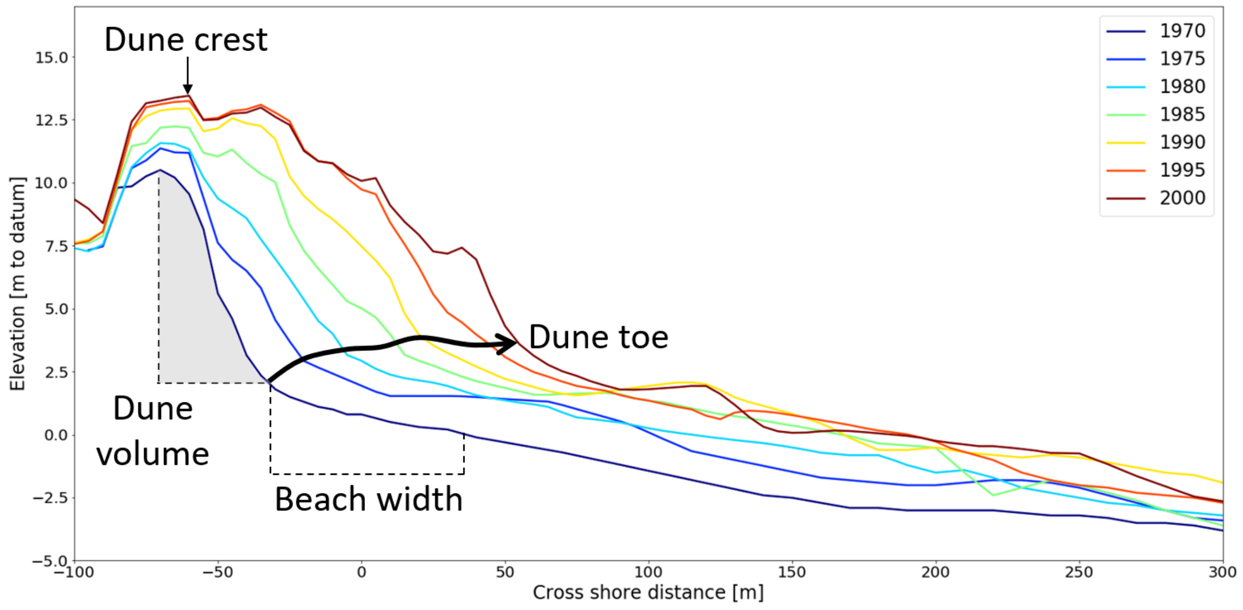
Christa van IJzendoorn

Jun 17, 2021

CONTENTS

The Jarkus Analysis Toolbox (JAT) is a Python-based open-source software, that can be used to analyze the Jarkus dataset. The Jarkus dataset is one of the most elaborate coastal datasets in the world and consists of coastal profiles of the entire Dutch coast, spaced about 250-500 m apart, which have been measured yearly since 1965. The main purpose of the JAT is to provide stakeholders (e.g. scientists, engineers and coastal managers) with the techniques that are necessary to study the spatial and temporal variations in characteristic parameters like dune height, dune volume, dune foot, beach width and closure depth. Different available definitions for extracting these characteristic parameters were collected and implemented in the JAT.

The software that is described in this documentation can be found in this Github repository. Additionally, the extracted parameters for the entire Jarkus dataset are made available through the 4TU repository. Please use the issues page to raise questions and suggest improvements.



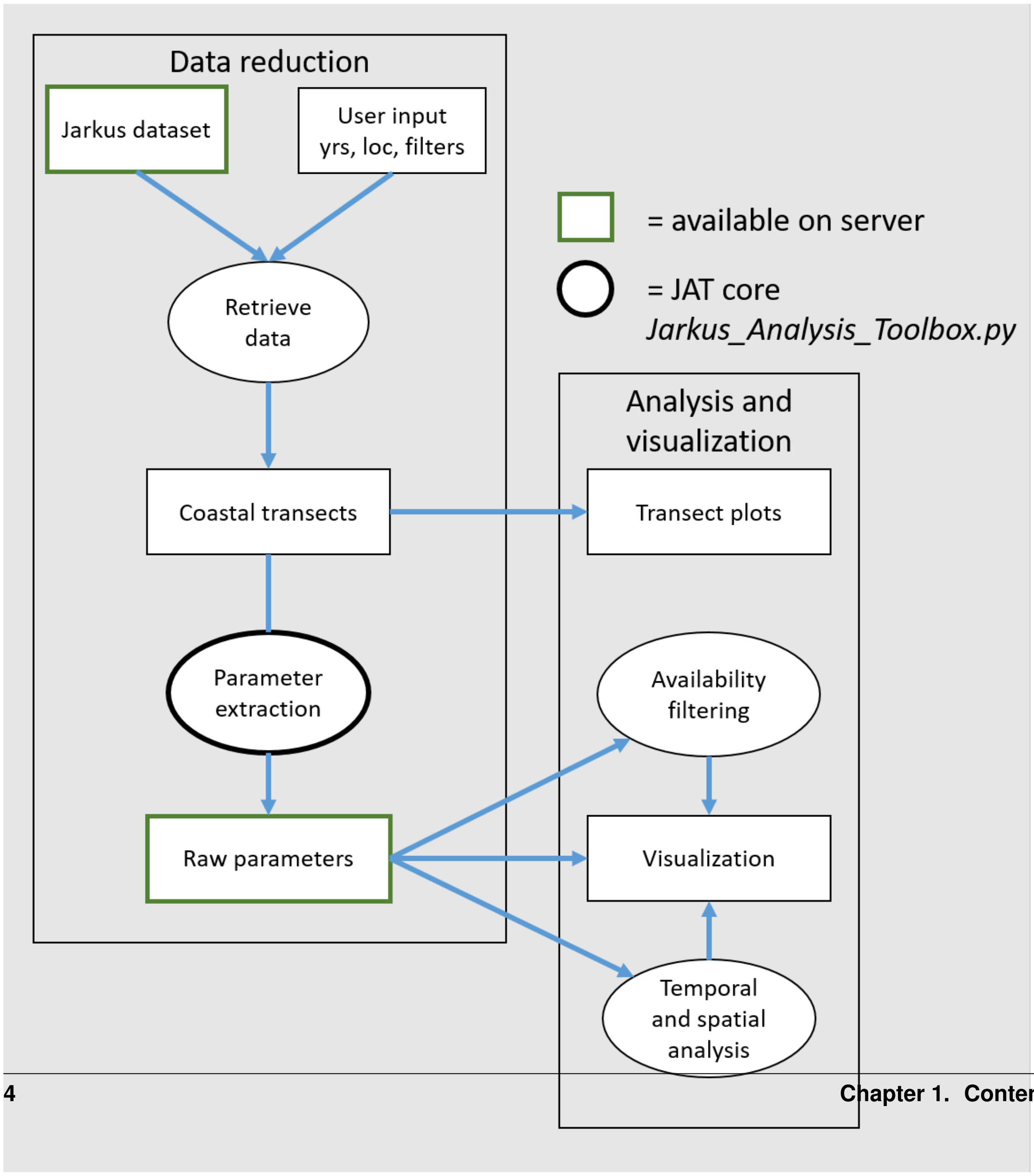
Example of characteristic parameters that can be extracted using the **JAT**.

CHAPTER

ONE

CONTENTS

1.1 Method



Flowchart of Jarkus Analysis Toolbox functionalities.

The Jarkus Analysis Toolbox helps to analyse the Jarkus dataset. This dataset is stored on an online repository and made available by Rijkswaterstaat and Deltares.

Based on user input the necessary data is retrieved from this dataset by the JAT for certain years and locations. The JAT contains the option to save the elevation information of the requested coastal transects and to create a quickplot that shows all measured years per requested transect.

The core of the JAT is in the parameter extraction. This means that characteristic parameters are extracted from the elevation profile of each requested coastal transect. User input determines which characteristic parameters are extracted and it is expected that more extraction methods will be added to the JAT in the future. A guide on how to add a new method is provided in the *Development* section.

The raw output parameters that were extracted by using the currently available methods are made avaiable through 4TU repository.

Within the examples provided along with the JAT there are examples of filtering and visualisation that can be executed based on the raw output parameters. These examples provide suggestions which help to kick-start further analysis, but this is where the user can apply their own methods.

1.2 Getting Started

1.2.1 Installation

Download the JAT from https://github.com/christavanijzendoorn/JAT.git and save the JAT to a convenient location on your computer.

Or use git and navigate to a convenient location and clone the repository:

```
$ git clone https://github.com/christavanijzendoorn/JAT.git
```

Open anaconda prompt and activate the environment you created or want to use (are you not able to follow? Go to *Help*). The JAT requires Python 3.7 and is not compatible with Python 3.8, so make sure to use the right version in your environment.

Navigate to the directory where the Jarkus Ananlysis Toolbox is located and the *setup.py* file is present. Use the following command to install the JAT:

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

1.2.2 Using the JAT

To use the JAT you will need to create two files (the names are suggestions based on the provided *Examples*:

1. *jarkus.yml*
2. *JAT_use.py*

The *jarkus.yml* file contains all the settings that are used to analyse the jarkus data.

These settings include:

- **years and transects** - Fill in the requested years and transects
- **inputdir** - Fill in where the input data is stored
- **outputdir** - Fill in where you want to store the JAT output

- **data locations** - Specify the name of the input files or specify their online location
- **save locations** - Specify the names of the folders in which the JAT output is saved
- **user defined** - Specify the user defined values
- **dimensions:**
  - **setting** - Specify the characteristic parameters that should be extracted
  - **variables** - No action needed, this is included to create a list of the requested parameters

The functionalities that you can use in the *JAT_use.py* file are explained in the *Functionalities* section. The best way to get an introduction into these functionalities is by using the *Examples*. These examples provide information on how to prepare transects, extract dimensions from these transects and show how to filter, analyse and visualize the extracted dimensions. Do not forget to change the directory of the *jarkus.yml* file in *JAT_use.py*.

Below you can find information that helps to understand (how to fill in) the settings in the *jarkus.yml* file.

1.2.3 Jarkus transect numbers

To be able to decide what transects you want to analyse with the JAT, you need to know the way in which the transects are numbered. The convention that is used in the JAT is as follows:

Vaknummer + raainummer = VNNNNNN:

- always 6 transect (raai) related numbers
- 1 or 2 coastal section (kustvak) related numbers, 2 in case of kustvak of 10+

Example Sand Engine: Vak 9, raai 11109 = 9011109 Example Meijendel: Vak 8, raai 9325 = 8009325 Example Westenschouwen: Vak 13, raai 1465 = 13001465

To check which transects are present in the area you want to analyse use the following sources:

- Overview of transects: https://maps.rijkswaterstaat.nl/geoweb55/index.html?viewer=Kustlijnkaart
- Overview of transects and 'kustvakken': https://puc.overheid.nl/rijkswaterstaat/doc/PUC_629858_31/

In the *jarkus.yml* file you can choose how many transects you want to analyse. First, you choose the type of analysis:

- **single** - analyse just one transect
- **multiple** - analyse a selection of tansects, these do not have to be next to each other spatially
- **range** - analyse transects between certain transect numbers. Especially around the boundaries of kustvakken, make sure to check whether the transects you want are indeed in increasing order
- **all** - analyse al available transect in the Jarkus dataset

In all cases, the JAT will automatically filter transect numbers that do not exist.

1.2.4 Input files

Jarkus

The Jarkus Analysis Toolbox was developed to make the analysis of the Jarkus dataset more accessible. To work with the JAT, the Jarkus data has to be accessed through this link.

When you want to access large amounts of data (i.e. many transects and years) or want to be independent of internet access it is advisable to download the dataset (approx. 3 GB). Make sure to include their directory in the settings file (*jarkus.yml*).

Dunetoe

When you want to work with the *dune toes* that were extracted using the second derivative method. These can be found here.

Nourishment

This is where the *nourishment* database can be found.

LocFilter

The *location_filter.yml* file is used to remove transects that contain, for instance, dams and dikes. It is used in Example 4 with `JAT.Filtering_functions.locations_filter`. This file can be rewritten and used with the `JAT.Filtering_functions.locations_filter` to do other types of filtering.

Titles

This file is used to automatically create figures that show the distribution through time and space of all available characteristic parameters, see Example 3.

1.2.5 User-defined settings

Below you can find a list of all user-defined settings that are included in the *jarkus.yml* file. For each setting a link to the documentation of the corresponding function is provided which explains how the setting is used.

- filter1: `JAT.Jarkus_Analysis_Toolbox.Transects.save_elevation_dataframes`
- filter2: `JAT.Filtering_functions.availability_locations_filter`
- primary dune: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top`
- secondary dune: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_secondary_dune_top`
- mean sea level: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level`
- mean high water: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_water_fixed`
- mean low water: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_water_fixed`
- landward variance threshold: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_variance`
- landward derivative: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_derivative`
- landward bma: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_bma`
- seaward foreshore: `JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point_foreshore`

- seaward active profile: *JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point_activeprofile*
- seaward DoC: *JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point_doc*
- dune toe fixed: *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*
- dune toe classifier: *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*
- normalization: `JAT.Jarkus_Analysis_Toolbox.Extraction.normalize_dimensions(`

1.2.6 Dependencies

The JAT has specific dependencies that are managed through the *setup.py* file, the packages needed are as follows:

```
* numpy =1.17.2
* pandas = 0.25.1
* netCDF4
* scipy = 1.3.1
* matplotlib
* cftime = 1.0.3.4
* joblib = 0.13.2
* pybeach
```

1.2.7 License

The JAT is free software made available under the GPL-3.0 License. For details see the license file.

1.3 Examples

After going through the *Getting Started* section you can try out the Jarkus Analysis Toolbox with the following examples.

1.3.1 1. Single transect

This example provides the code necessary to extract all characteristic parameters from one single transect. The default settings in the *jarkus_01.yml* file look at the years from 1980 to 2021 and at transect location 8009325. It is adviced to play around with these settings and extract the characteristic parameters for different periods and location. To extract the characteristic parameters open *JAT_use_single_transect.py* in the Python IDE of your choice (Spyder is recommended, see *Help*) and run the commands step by step. This should show you the steps necessary to extract the characteristic parameters for one transect and gives examples how these data can be visualized. Note that the plotting functions of *pandas* were used in this example, for more elaborate visualization use *matplotlib*.

1.3.2 2. Regional analysis

Example 2 shows how to extract the characteristic parameters from multiple transects at once. Tow work with this example, include the correct directories in the *jarkus_02.yml* file and run the code in *JAT_use_region_transects.py*.

1.3.3 3. Extract all

This Example shows how to extract all characteristic parameters from all transect locations. For this, include the correct directories in the *jarkus_03.yml* file and run the code in *JAT_use_extract_all.py*. The analysis can take a long time, around 10 hours. Thus, it is recommended to download the input files and store them locally to reduce the run time.

The *Filtering_execution.py* file provides an example of how the filtering functionalities of the JAT can be used.

To create distribution plots that show the values of the characteristic parameters through time and space use *Distribution_plots.py*. This script can only be used after the output of *JAT_use_extract_all.py* and *Filtering_execution.py* are available. *Distribution_plots.py* creates the distribution plots for both the filtered and unfiltered dataframes. The distribution plots of the unfiltered dataframes are available on the 4TU repository to show what the characteristic parameters look like.

Creation_netcdf.py was used to produce the netcdf file that is available on the 4TU repository. The output of *Creation_netcdf.py*, which is *extracted_parameters.nc* is saved in the Input directory because it serves as the input for Example 5.

1.3.4 4. Dune toe analysis

The Jarkus Analysis Toolbox was developed during the research that led to the publication of Van IJzendoorn et al. (2021)[1]. This example shows how the toolbox was used for the dune toe analysis. To replicate the results include the correct directories in the *jarkus_04.yml* file and run the code in *JAT_use_dune_toe_analysis.py*. Then, the following scripts produce the figures that are included in the paper.

- dunetoe_transect_figure.py - Figure 1
- dunetoe_transect_map.py - Figure 1
- dunetoe_trend_figure.py - Figure 2 and Supl. Figure 1
- dunetoe_alongshore_figure.py - Figure 3
- sea_level_rise_figure.py - Figure 4

The mapping executed in the dunetoe_transect_map.py uses the package basemap which is dependent on a specific version of matplotlib and is therefore not compatible with the jarkus dependencies. Thus, it is best to create a new environment to run this script. This can be done by using the dune_transect_map.yml file which includes all the dependencies necessary to run the mapping script. Use the anaconda prompt and go to directory where environment file (dune_transect_map.yml) is located, use the following commands:

```
$ conda env create -f dune_transect_map.yml
$ conda activate map
$ python dunetoe_transect_map.py
```

It should be noted that for the sea level rise figure, a specific dataset is used that can be found here.

The Figures folder includes all figures for reference so you can check whether your output matches the expectations.

[1] Van IJzendoorn, C.O., De Vries, S., Hallin, C. & Hesp, P.A. (2021). Sea level outpaced by coastal dune toe translation. *In review*

1.3.5 5. Use NetCDF file

The output of Example 3 was converted into a netcdf file that is publicly available. This makes sure that the characteristic parameters can be accessed directly without having to use the Jarkus Analysis Toolbox. Thus, to work with this example you can choose to work through example 3 or just simply download *extracted_parameters.nc* from the 4TU repository.

The *Load_data_from_netcdf.py* script shows how to load the extracted characteristic parameters from the netcdf file and gives a first glimpse of how to work with these data.

1.4 Characteristic parameters

1.4.1 Explanation of characteristic parameters

This table provides the user with a direct link to the explanation of the module that extracts the characteristic parameter from the coastal profile. Thus, it provides an explanation of the precise method used to derive each characteristic parameter.

Parameter	Type	Explanation
Dune top	Primary	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_*
	Secondary	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_secondary_dun*
Mean Sea Level	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_leve*
	Variable	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_leve*
Mean Low Water	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_wate*
	Variable	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_wate*
Mean High Water	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_wat*
	Variable	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_wat*
Intertidal area width	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_intertidal_wi*
	Variable	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_intertidal_wi*
Landward boundary	Variance	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_poin*
	Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_poin*
	BMA	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_poin*
Seaward boundary	Foreshore	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point*
	Active Profile	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point*
	Depth of Closure	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point*
Dune toe	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixe*
	Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_deri*
	Pybeach	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_pybe*
Beach width	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_beach_width_f*
	Variable	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_beach_width_v*
	Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_beach_width_d*
	Variable Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_beach_width_d*
Beach gradient	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_beach_gradien*
	Variable	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_beach_gradien*
	Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_beach_gradien*
Dune front width	Primary Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_wi*
	Primary Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_wi*
	Secondary Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_wi*
	Secondary Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_wi*
Dune front gradient	Primary Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_gr*
	Primary Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_gr*

co

Table 1 – continued from previous page

	Secondary Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_gr*
	Secondary Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_front_gr*
Dune volume	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_volume_f*
	Derivative	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_volume_d*
Intertidal area gradient	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_intertidal_gr*
Intertidal area volume	Fixed	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_intertidal_vo*
	Variable	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_intertidal_vo*
Foreshore gradient	BMA	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_foreshore_gra*
Foreshore volume	BMA	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_foreshore_vol*
Active profile gradient	BMA	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_active_profil*
Active profile volume	BMA	*JAT.Jarkus_Analysis_Toolbox.Extraction.get_active_profil*

1.4.2 Variable names and dependencies of characteristic parameters

This table shows the variable name of each characteristic parameter that is used in the jarkus.yml file to indicate which characteristic parameters should be extracted. Additionally, the corresponding variable names of the output that is produced for each characteristic parameter is given. The last column provides on which variable the extraction of the characteristic parameters depends. For instance, the mean sea level can only be extracted when the dune top location had already been extracted because the cross-shore location of the primary dune top is necessary. These dependencies are also indicated in the *Functionalities* with the *See also* sections.

Parameter	Variable name (in jarkus.yml)	Output variables	Dependent
Dune top	primary_dune_top	DuneTop_prim_x	
		DuneTop_prim_y	
	secondary_dune_top	DuneTop_sec_x	DuneTop_prim_x
		DuneTop_sec_y	DuneTop_prim_y
Mean Sea Level	mean_sea_level	MSL_x	DuneTop_prim_x
	mean_sea_level_variable	MSL_x_var	MLW_x_var
			MHW_x_var
Mean Low Water	mean_low_water_fixed	MLW_x_fix	MSL_x
	mean_low_water_variable	MLW_x_var	MSL_x
		MHW_y_var	
Mean High Water	mean_high_water_fixed	MHW_x_fix	MSL_x
	mean_high_water_variable	MHW_x_var	MSL_x
		MHW_y_var	
Intertidal area width	intertidal_width_fixed	Intertidal_width_fix	MLW_x_var
			MHW_x_var
	intertidal_width_var	Intertidal_width_var	MLW_x_var
			MHW_x_var
Landward boundary	landward_point_variance	Landward_x_variance	DuneTop_prim_x
	landward_point_derivative	Landward_x_der	MHW_y_var
	landward_point_bma	Landward_x_bma	
Seaward boundary	seaward_point_foreshore	Seaward_x_FS	
	seaward_point_activeprofile	Seaward_x_AP	
	seaward_point_doc	Seaward_x_mindepth	
		Seaward_x_DoC	
Dune toe	dune_toe_fixed	Dunetoe_x_fix	
	dune_toe_derivative	Dunetoe_y_der	
		Dunetoe_x_der	
	dune_toe_pybeach	Dunetoe_y_pybeach	DuneTop_prim_x

continues on next page

Table 2 – continued from previous page

		Dunetoe_x_pybeach	MSL_x
			MHW_x_var
			Landward_x_der
Beach width	beach_width_fix	Beach_width_fix	MSL_x
			Dunetoe_x_fix
	beach_width_var	Beach_width_var	MSL_x_var
			Dunetoe_x_fix
	beach_width_der	Beach_width_der	MSL_x
			Dunetoe_x_der
	beach_width_der_var	Beach_width_der_var	MSL_x_var
			Dunetoe_x_der
Beach gradient	beach_gradient_fix	Beach_gradient_fix	MSL_x
			Dunetoe_x_fix
	beach_gradient_var	Beach_gradient_var	MSL_x_var
			Dunetoe_x_fix
	beach_gradient_der	Beach_gradient_der	MSL_x
			Dunetoe_x_der
Dune front width	dune_front_width_prim_fix	Dune-front_width_prim_fix	DuneTop_prim_x
			Dunetoe_x_fix
	dune_front_width_prim_der	Dune-front_width_prim_der	DuneTop_prim_x
			Dunetoe_x_der
	dune_front_width_sec_fix	Dunefront_width_sec_fix	DuneTop_prim_x
			DuneTop_prim_y
			DuneTop_sec_x
			Dunetoe_x_fix
	dune_front_width_sec_der	Dunefront_width_sec_der	DuneTop_prim_x
			DuneTop_prim_y
			DuneTop_sec_x
			Dunetoe_x_der
Dune front gradient	dune_front_gradient_prim_fix	Dune-front_gradient_prim_fix	DuneTop_prim_x
			Dunetoe_x_fix
	dune_front_gradient_prim_der	Dune-front_gradient_prim_der	DuneTop_prim_x
			Dunetoe_x_der
	dune_front_gradient_sec_fix	Dune-front_gradient_sec_fix	DuneTop_prim_x
			DuneTop_prim_y
			DuneTop_sec_x
			Dunetoe_x_fix
	dune_front_gradient_sec_der	Dune-front_gradient_sec_der	DuneTop_prim_x
			DuneTop_prim_y
			DuneTop_sec_x
			Dunetoe_x_der
Dune volume	dune_volume_fix	DuneVol_fix	DuneTop_prim_x
			Landward_x_variance
			Dunetoe_x_fix
	dune_volume_der	DuneVol_der	DuneTop_prim_x
			Landward_x_variance
			Dunetoe_x_der
Intertidal area gradient	intertidal_gradient	Intertidal_gradient_fix	MSL_x
			MLW_x_fix
			MHW_x_fix
Intertidal area volume	intertidal_volume_fix	Intertidal_volume_fix	MSL_x
			MLW_x_fix

continues on next page

Table 2 – continued from previous page

			MHW_x_fix
	intertidal_volume_var	Intertidal_volume_var	MSL_x
			MLW_x_var
			MHW_x_var
Foreshore gradient	foreshore_gradient	Foreshore_gradient	Seaward_x_FS
			Landward_x_bma
Foreshore volume	foreshore_volume	Foreshore_volume	Seaward_x_FS
			Landward_x_bma
Active profile gradient	active_profile_gradient	Active_profile_gradient	Seaward_x_AP
			Landward_x_bma
Active profile volume	active_profile_volume	Active_profile_volume	Seaward_x_AP
			Landward_x_bma

1.5 Functionalities

The Jarkus Analysis Toolbox provides many functionalities. Here, all available modules are explained per category.

1.5.1 Geometric functions

Provides basic geometric functions that execute calculations based on the coastal profile.

`JAT.Geometric_functions.`**`find_intersections`**(*elevation*, *crossshore*, *y_value*)

Find cross-shore location of intersection between profile and horizontal line at a fixed elevation.

Parameters

- **elevation** (*np.array*) – np.array containing the elevation of the coastal profile in meters.
- **crossshore** (*np.array*) – np.array containing the crossshore location in meters.
- **y_value** (*float*) – Elevation of the horizontal line in meters.

Returns intersection_x: Cross-shore location of the intersection between the coastal profile and horizontal line.

Return type int

`JAT.Geometric_functions.`**`get_gradient`**(*elevation*, *seaward_x*, *landward_x*)

Find gradient of a profile between two points

The gradient of a coastal profile is determined by finding the slope of the line of best fit along the elevation between a landward and seaward boundary.

Parameters

- **elevation** (*np.array*) – np.array containing the elevation of the coastal profile in meters
- **seaward_x** (*float or int*) – Cross-shore seaward boundary
- **landward_x** (*float or int*) – Cross-shore landward boundary

Returns gradient: slope of the best fit line

Return type float

`JAT.Geometric_functions.`**`get_volume`**(*elevation*, *seaward_x*, *landward_x*)

Determine volume under coastal profile between two two points

The volume of the coastal profile between a landward and seaward boundary is determined by integrating over the surface beneath the coastal profile between those two points.

Parameters

- **`elevation`** (*`np.array`*) – np.array containing the elevation of the coastal profile in meters
- **`seaward_x`** (*`float or int`*) – Cross-shore seaward boundary
- **`landward_x`** (*`float or int`*) – Cross-shore landward boundary

Returns volume: surface under the graph in m^2. Can be interpreted as m^3 by assuming the profile is 1 m wide.

Return type float

1.5.2 Jarkus Analysis Toolbox

Includes the most important functionalities of the JAT including retrieving data and extracting profile dimensions

`class` `JAT.Jarkus_Analysis_Toolbox.`**`Extraction`**(*data*, *config*)

Extracting characteristic parameters from coastal profiles.

This class provides the functionalities to extract the characteristic parameters requested by the user from transects of the jarkus dataset. Additionally, it provides functions to post-process the outcome of the extraction.

`get_active_profile_gradient`(*trsct_idx*)

Extract the gradient of the active profile (Active_profile_gradient).

The gradient of the active profile is determined by finding the slope of the line of best fit along the coastal profile between the seaward active profile boundary and the landward boundary between the marine and aeolian zone (BMA).

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point_activeprofile`, *`JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_bma`*, *`JAT.Geometric_functions.get_gradient`*

`get_active_profile_volume`(*trsct_idx*)

Extract the volume of the active profile (Active_profile_volume).

The volume of the active profile is determined by finding the surface under the coastal profile between the seaward active profile boundary and the landward boundary between the marine and aeolian zone (BMA).

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point_activeprofile`, *`JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_bma`*, *`JAT.Geometric_functions.get_volume`*

get_all_dimensions()

Extracts all requested characteristic parameters for all requested years and transects.

Checks whether the saving directory is in place and proceeds to go through all requested transects. Per characteristic parameter it is checked whether it was requested, and, if so, the values for all requested years are extracted. Ultimately, per transect a dataframe is saved that includes the values of all requested characteristic parameters for all years at that location.

get_beach_gradient_der(*trsct_idx*)

Extract the gradient of the beach (Beach_gradient_der).

The gradient of the beach is determined by finding the slope of the line of best fit along the coastal profile between the fixed mean sea level and the second derivative dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*, *JAT.Geometric_functions.get_gradient*

get_beach_gradient_fix(*trsct_idx*)

Extract the gradient of the beach (Beach_gradient_fix).

The gradient of the beach is determined by finding the slope of the line of best fit along the coastal profile between the fixed mean sea level and the fixed dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*, *JAT.Geometric_functions.get_gradient*

get_beach_gradient_var(*trsct_idx*)

Extract the gradient of the beach (Beach_gradient_var).

The gradient of the beach is determined by finding the slope of the line of best fit along the coastal profile between the variable mean sea level and the fixed dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level_variable, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*, *JAT.Geometric_functions.get_gradient*

get_beach_width_der()

Extract the width of the beach (Beach_width_der).

The width of the beach is determined by calculating the cross-shore distance between the fixed mean sea level and the dune toe location based on the second derivative method.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*

get_beach_width_der_var()
Extract the width of the beach (Beach_width_der_var).

The width of the beach is determined by calculating the cross-shore distance between the variable mean sea level and the dune toe location based on the second derivative method.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level_variable, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*

get_beach_width_fix()
Extract the width of the beach (Beach_width_fix).

The width of the beach is determined by calculating the cross-shore distance between the fixed mean sea level and the fixed dune toe location.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*

get_beach_width_var()
Extract the width of the beach (Beach_width_var).

The width of the beach is determined by calculating the cross-shore distance between the variable mean sea level and the fixed dune toe location.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level_variable, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*

get_dataframe_per_dimension()
Creates and saves a dataframe per characteristic parameter from the dataframes with all requested characteristic parameters per transect.

get_dune_front_gradient_prim_der(*trsct_idx*)
Extract the gradient of the primary dune front (Dunefront_gradient_prim_der).

The gradient of the dune front is determined by finding the slope of the line of best fit along the coastal profile between the primary dune top and the derivative dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*, *JAT.Geometric_functions.get_gradient*

get_dune_front_gradient_prim_fix(*trsct_idx*)
Extract the gradient of the primary dune front (Dunefront_gradient_prim_fix).

The gradient of the dune front is determined by finding the slope of the line of best fit along the coastal profile between the primary dune top and the fixed dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*, *JAT.Geometric_functions.get_gradient*

get_dune_front_gradient_sec_der(*trsct_idx*)

Extract the gradient of the secondary dune front (Dunefront_gradient_sec_der).

The gradient of the dune front is determined by finding the slope of the line of best fit along the coastal profile between the secondary dune top and the derivative dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_secondary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*, *JAT.Geometric_functions.get_gradient*

get_dune_front_gradient_sec_fix(*trsct_idx*)

Extract the gradient of the secondary dune front (Dunefront_gradient_sec_fix).

The gradient of the dune front is determined by finding the slope of the line of best fit along the coastal profile between the secondary dune top and the fixed dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_secondary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*, *JAT.Geometric_functions.get_gradient*

get_dune_front_width_prim_der()

Extract the width of the primary dune front (Dunefront_width_prim_der).

The width of the primary dune front is determined by calculating the cross-shore distance between the cross-shore location of the primary dune top and the derivative dune toe location.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*

get_dune_front_width_prim_fix()

Extract the width of the primary dune front (Dunefront_width_prim_fix).

The width of the primary dune front is determined by calculating the cross-shore distance between the cross-shore location of the primary dune top and the fixed dune toe location.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*

get_dune_front_width_sec_der()

Extract the width of the secondary dune front (Dunefront_width_sec_der).

The width of the secondary dune front is determined by calculating the cross-shore distance between the cross-shore location of the secondary dune top and the derivative dune toe location.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_secondary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*

`get_dune_front_width_sec_fix`()

Extract the width of the secondary dune front (Dunefront_width_sec_fix).

The width of the secondary dune front is determined by calculating the cross-shore distance between the cross-shore location of the secondary dune top and the fixed dune toe location.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_secondary_dune_top`, *`JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed`*

`get_dune_toe_derivative`(*trsct_idx*)

Extract the elevation (Dunetoe_y_der) and cross-shore location (Dunetoe_x_der) of the dune toe based on the second derivative method [3].

The dune toe elevation is extracted from the repository where the work of Diamantidou et al. [3] is saved. First, the method selects part of the coastal profile. This selection is between the landward boundary (get_landward_point_derivative) and the variable MHW. Then, the first and second derivative of this part of the coastal profile is calculated. The most seaward location where the first derivative is lower than 0.001 and the second derivative is lower than 0.01 is selected as the dune toe [3].

Parameters **`trsct_idx`** (*`index of the transect necessary to extract the elevation of the profiles.`*) –

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_derivative`

`get_dune_toe_fixed`(*trsct_idx*)

Extract the cross-shore location of the dune toe (Dunetoe_x_fix).

The dune toe is defined as a fixed, user-defined elevation (default = +3 m). The intersections between this elevation and the coastal profile are determined. Then, the cross-shore location of the most seaward intersection is selected as the dune toe.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

`get_dune_toe_pybeach`(*trsct_idx*)

Extract the elevation (Dunetoe_y_pybeach) and cross-shore location (Dunetoe_x_pybeach) of the dune toe based on pybeach machine learning method[7].

Pybeach provides three different pre-trained machine learning models (barrier-island, wave-embayed and mixed) that can extract the dune toe location. Here, the user can define which model to use (default = 'mixed') These models were based on the identification of the dune toe by experts. To make the applicaiton of the pybeach machine learning method comparable to the second derivative method a similar reduction of the coastal profile (with a landward and seaward boundary) is executed.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_derivative`, *`JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative`*

`get_dune_volume_der`(*trsct_idx*)

Extract the dune volume (DuneVol_der).

The dune volume is determined by finding the surface under the coastal profile between the primary dune top and the derivative dune toe location.

[7] Beuzen, Tomas. "pybeach: A Python package for extracting the location of dune toes on beach profile transects." Journal of Open Source Software 4(44) (2019): 1890. https://doi.org/10.21105/joss.01890

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_derivative*, *JAT.Geometric_functions.get_volume*

get_dune_volume_fix (*trsct_idx*)

Extract the dune volume (DuneVol_fix).

The dune volume is determined by finding the surface under the coastal profile between the primary dune top and the fixed dune toe location.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_dune_toe_fixed*, *JAT.Geometric_functions.get_volume*

get_foreshore_gradient (*trsct_idx*)

Extract the gradient of the foreshore (Foreshore_gradient).

The gradient of the foreshore is determined by finding the slope of the line of best fit along the coastal profile between the seaward foreshore boundary and the landward boundary between the marine and aeolian zone (BMA).

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point_foreshore, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_bma*, *JAT.Geometric_functions.get_gradient*

get_foreshore_volume (*trsct_idx*)

Extract the volume of the foreshore (Foreshore_volume).

The foreshore volume is determined by finding the surface under the coastal profile between the seaward foreshore boundary and the landward boundary between the marine and aeolian zone (BMA).

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_seaward_point_foreshore, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_landward_point_bma*, *JAT.Geometric_functions.get_volume*

get_intertidal_gradient_fix (*trsct_idx*)

Extract the gradient of the intertidal area (Intertidal_gradient_fix).

The gradient of the intertidal area is determined by finding the slope of the line of best fit along the coastal profile between the fixed mean low water and the fixed mean high water.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_water_fixed, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_water_fixed*, *JAT.Geometric_functions.get_gradient*

get_intertidal_volume_fix(*trsct_idx*)

Extract the volume of the intertidal area (Intertidal_volume_fix).

The intertidal area volume is determined by finding the surface under the coastal profile between the fixed mean low water and the fixed mean high water.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_water_fixed, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_water_fixed*, *JAT.Geometric_functions.get_volume*

get_intertidal_volume_var(*trsct_idx*)

Extract the volume of the intertidal area (Intertidal_volume_var).

The intertidal area volume is determined by finding the surface under the coastal profile between the variable mean low water and the variable mean high water.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_water_variable, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_water_variable*, *JAT.Geometric_functions.get_volume*

get_intertidal_width_fixed()

Extract the width of the intertidal area (Intertidal_width_fix).

The width of the intertidal area is determined by calculating the cross-shore distance between the fixed mean low water and the fixed mean high water.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_water_fixed, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_water_fixed*

get_intertidal_width_variable()

Extract the width of the intertidal area (Intertidal_width_var).

The width of the intertidal area is determined by calculating the cross-shore distance between the variable mean low water and the variable mean high water.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_water_variable, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_water_variable*

get_landward_point_bma(*trsct_idx*)

Extract the cross-shore location of the landward boundary based on the boundary between the marine and aeolian zone (Landward_x_bma) [4].

The landward boundary is defined as a fixed, user-defined elevation (default = +2 m). The intersections between this elevation and the coastal profile are determined. Then, the cross-shore location of the most seaward intersection is selected as the landward boundary.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

`get_landward_point_derivative`(*trsct_idx*)

Extract the cross-shore location of the landward boundary based on steps in the second derivative method (Landward_x_der) [3].

The landward boundary is determined by finding the peaks with a prominence larger than a fixed threshold (default = +2.4 m). If peaks are found and those peaks are larger than a user-defined elevation (default = 6.0), the cross-shore location of the intersection of this elevation with the coastal profile is the landward boundary. Otherwise, the peaks above the peaks threshold (variable MHW + prominence threshold) are selected and the most seaward selected peak is the landward boundary. If none of these selection cannot be applied a NaN value is inserted. This function uses scipy.signal.find_peaks [1]. The prominence of a peak measures how much a peak stands out from the surrounding baseline of the signal and is defined as the vertical distance between the peak and its lowest contour line [2].

Parameters **`trsct_idx`** (*`index of the transect necessary to extract the elevation of the profiles.`*) –

`get_landward_point_variance`(*trsct_idx*)

Extract the cross-shore location of the landward boundary based on variance (Landward_x_variance).

The landward boundary is determined by calculating the variance of the elevation of a transect location throughout all available years. Stable points are located based on where the variance of the elevation through time is below a user-defined threshold (default = 0.1). The stable points landward of the primary dune top are filtered out and the cross-shore location and elevation of the most seaward stable point is are selected as the landward boundary.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top`

`get_mean_high_water_fixed`(*trsct_idx*)

Extract the cross-shore location of mean high water (MHW_x_fix).

The mean high water is defined as a fixed, user-defined elevation (default = + 1 m). The intersections between this elevation and the coastal profile are determined. Then, intersections that are further than 250 m landward of the location of the mean sea level (MSL_x) are excluded. This filtering is necessary to make sure landward intersections behind the dunes are not selected as the MHW location.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level`

`get_mean_high_water_variable`(*trsct_idx*)

Extract the elevation (MHW_y_var) and cross-shore location (MHW_x_var) of mean low water.

The mean high water is defined as a spatially variable elevation. This elevation is provided per transect location in the jarkus database (determined with tidal modeling). The intersections between this elevation and the coastal profile are determined. Then, intersections that are further than 250 m landward of the location of the mean sea level (MSL_x) are excluded. This filtering is necessary to make sure landward intersections behind the dunes are not selected as the MHW location. Both the cross-shore location and variable elevation are saved. Note, that the spatially variable elevation is not variable in time, so each transect has a constant MHW elevation assigned that is constant throughout the years.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level`

`get_mean_low_water_fixed`(*trsct_idx*)

Extract the cross-shore location of mean low water (MLW_x_fix).

The mean low water is defined as a fixed, user-defined elevation (default = -1 m). The intersections between this elevation and the coastal profile are determined. Then, intersections that are further than 250 m seaward of the location of the mean sea level (MSL_x) are excluded. This filtering is necessary to make sure seaward intersections caused by for instance the presence of shoals are not selected as the MLW location.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level`

`get_mean_low_water_variable`(*trsct_idx*)

Extract the elevation (MLW_y_var) and cross-shore location (MLW_x_var) of mean low water.

The mean low water is defined as a spatially variable elevation. This elevation is provided per transect location in the jarkus database (determined with tidal modeling). The intersections between this elevation and the coastal profile are determined. Then, intersections that are further than 250 m seaward of the location of the mean sea level (MSL_x) are excluded. This filtering is necessary to make sure seaward intersections caused by for instance the presence of shoals are not selected as the MLW location. Both the cross-shore location and variable elevation are saved. Note, that the spatially variable elevation is not variable in time, so each transect has a constant MLW elevation assigned that is constant throughout the years.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

`JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_sea_level`

`get_mean_sea_level`(*trsct_idx*)

Extract the cross-shore location of mean sea level (MSL_x).

The mean sea level is defined as a fixed, user-defined elevation (default = 0 m). The intersections between this elevation and the coastal profile are determined. The most seaward intersection is selected as the cross-shore location if no primary dune top is available. Otherwise, all intersections landward of the primary dune top are filtered out. Then, if the distance between the most seaward and landward intersection is equal or smaller than 100 m the most seaward intersection is selected as the cross-shore MSL location. Otherwise, if the distance is larger than 100 m, only the intersections 100 m landward of the most seaward intersection are selected. Of this selection, the most seaward intersection is selected as the cross-shore MSL location. This filtering is necessary to make sure landward intersections behind the dunes and seaward intersections due to the presence of shoals are not selected as the MSL location.

Parameters **`trsct_idx`** (*`int`*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top

get_mean_sea_level_variable()

Extract the cross-shore mean sea level location (MSL_x_var) based on the variable mean high and low water.

The mean sea level location is determined by calculating the middle point between the cross-shore location of the variable mean high water and the variable mean low water.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_low_water_variable, *JAT.Jarkus_Analysis_Toolbox.Extraction.get_mean_high_water_variable*

get_primary_dune_top(*trsct_idx*)

Extract the primary dune top height (DuneTop_prim_y) and cross-shore location (DuneTop_prim_x).

The primary dune top is defined as the most seaward dune top that has a height that is larger than a user-defined threshold (default = 5 m) and a prominence that is larger than a user-defined value (default = 2.0). This function uses scipy.signal.find_peaks [1]. The prominence of a peak measures how much a peak stands out from the surrounding baseline of the signal and is defined as the vertical distance between the peak and its lowest contour line [2].

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

get_requested_variables()

Retrieve all variables that are related to the requested characteristic parameters

Returns variables_req: List of all variable that are related to the requested characteristic parameters as included in the configuration file

Return type list

get_seaward_point_activeprofile(*trsct_idx*)

Extract the cross-shore location of the seaward active profile boundary (Seaward_x_AP).

The seaward boundary is defined as a fixed, user-defined elevation (default = -8 m). The intersections between this elevation and the coastal profile are determined. Then, the cross-shore location of the most seaward intersection is selected as the seaward boundary.

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

get_seaward_point_doc(*trsct_idx*)

Extract the cross-shore location (Seaward_x_DoC) of the depth of closure based on the method of Hinton [5].

Approximation of the depth of closure below a user-defined minimum (default = -5.0 m) (Seaward_x_mindepth) where the standard deviation of the elevation through time is below a user-defined value (default = 0.25) for at least a user-defined length (default = 200m), based on the method by Hinton [5].

Parameters **trsct_idx** (*int*) – index of the transect necessary to extract the elevation of the profiles.

get_seaward_point_foreshore(*trsct_idx*)

Extract the cross-shore location of the seaward foreshore boundary (Seaward_x_FS).

The seaward boundary is defined as a fixed, user-defined elevation (default = -4 m). The intersections between this elevation and the coastal profile are determined. Then, the cross-shore location of the most seaward intersection is selected as the seaward boundary.

Parameters **`trsct_idx`** (*int*) – index of the transect necessary to extract the elevation of the profiles.

`get_secondary_dune_top`(*trsct_idx*)
Extract the secondary dune top height (DuneTop_sec_y) and cross-shore location (DuneTop_sec_x).

The secondary dune top is defined as the most seaward dune top that has a height that is larger than a user-defined threshold (default = 3 m) and a prominence that is larger than a user-defined value (default = 0.5) and is located seaward of the primary dune top. This function uses scipy.signal.find_peaks [1]. The prominence of a peak measures how much a peak stands out from the surrounding baseline of the signal and is defined as the vertical distance between the peak and its lowest contour line [2]. The goal of this function is to be able to identify embryo dune formation.

Parameters **`trsct_idx`** (*int*) – index of the transect necessary to extract the elevation of the profiles.

See also:

JAT.Jarkus_Analysis_Toolbox.Extraction.get_primary_dune_top

`normalize_dimensions`()
Normalize the cross-shore location values of all requested characteristic parameters

Normalization of the cross-shore locations is done to make cross-shore values between transects comparable. The normalization is executed by subtracting a normalization value from the value of each year of a characteristic parameter. This function provides the option to apply a normalization based on the mean of all the years available for a transect. Additionally, a normalization based on the value of a fixed user-defined year is available. The normalized cross-shore locations are saved as a dataframe.

class JAT.Jarkus_Analysis_Toolbox.**Transects**(*config*)
Loading and plotting transects.

This class provides the functionalities to retrieve the jarkus dataset and filter out the years and locations requested by the user. This includes determining whether the user defined request is available. Additionally, the elevation of each requested transect can be saved and plotted to provide easy access for analysis, and the conversion of the transect number to the alongshore kilometer is provided.

`get_availability`(*config*)
Getting available years and transects

This function executes the get_years_filtered and get_transects_filtered functions based on a configuration file containing the requested years and transects.

Parameters **`config`** (*dict*) – The configuration file that contains the user-requested years and transects

See also:

Transects.get_years_filtered, *Transects.get_transects_filtered*

`get_conversion_dicts`()
Create conversion from transect number to alongshore meter and vice versa

For each transect number in the jarkus dataset the alongshore kilometer is calculated. A dictionary is created that relates each transect number to its alongshore kilometer. Additionally, a dictionary is created that does the reverse.

Returns

conversion_ids2alongshore: does the conversion from transect number to alongshore meter

conversion_alongshore2ids: does the conversion from alongshore meter to transect number

Return type dict

`get_transect_plot`(*config*)
Save plot with all coastal profiles for each requested transect

For each requested transect a quickplot is created and saved (as png and picle file) that shows all the requested years. The colors in the plot go from the start year in red to the end year in blue. Currently the axes are set automatically but this can be changed to user-defined limits in the future, which is mostly relevant for single transect plotting.

Parameters **`config`** (*`dict`*) – The configuration file that contains the user-requested years and transects, reference to the jarkus dataset and the save locations.

`get_transects_filtered`(*transects*)
Filtering requested transects

It is determined what type of request is made and which transects are associated with this request. Then all transects in the jarkus dataset are extracted and compared to the user-requested years. Only the available (requested) years and their indices are retained.

Parameters **`transects`** (*`dict`*) – Part of the configuration file that includes which type of transects are requested (single, multiple, range or all) and (if applicable) which transects are associated with this request.

`get_years_filtered`(*start_yr*, *end_yr*)
Filtering requested years

All years in the jarkus dataset are extracted and compared to the user-requested years. Only the available (requested) years and their indices are retained.

Parameters

- **`start_yr`** (*`int`*) – Starting year of the user-requested period
- **`end_yr`** (*`int`*) – Ending year of the user-requested period

`save_elevation_dataframes`(*config*)
Save elevation of all years for each transect as a dataframe

The elevation and corresponding cross-shore location of each requested year and requested transect location are saved as a dataframe. Note that each resulting file contains the profiles for all requested years of one requested transect. The function provides the option to use a filter that leaves out profiles when there is no elevation data present between a certain minimum and maximum elevation. This can, for instance, be useful when only the foreshore is studied and all transects that do not have elevation data in this region are redundant. The user-defined values for filter1 are included in the configuration file. Currently this filter does not have an effect on the extraction of the characteristic parameters because these are determined based on the elevation that is directly extracted from the jarkus dataset. Therefore, the default setting for filter1 is that is it not applied (config[‘user defined’][‘filter1’][‘apply’]=False), but this could be changed in the future.

Parameters **`config`** (*`dict`*) – The configuration file that contains the user-requested years and transects, reference to the jarkus dataset, the filter1 settings and the save locations.

1.5.3 Filtering functions

Provides functions that allow filtering of the extracted characteristic parameters.

`JAT.Filtering_functions.`**`availability_locations_filter`**`(`*config*, *dimension*`)`

Filter out transects based on data availability.

Filter out transects that have a data availability that is lower than the user-defined threshold.

Parameters

- **`config`** (*`dict`*) – configuration file that includes the user defined availability threshold in percentage (filter2, locations)
- **`dimension`** (*`pd.dataframe`*) – dataframe containing the values of a characteristic parameter through time and more multiple transect locations.

Returns dimension_filt: dataframe containing the values of a characteristic parameter where filtered transects have been removed.

Return type pd.dataframe

`JAT.Filtering_functions.`**`availability_years_filter`**`(`*config*, *dimension*`)`

Filter out years based on data availability.

Filter out years that have a data availability that is lower than the user-defined threshold.

Parameters

- **`config`** (*`dict`*) – configuration file that includes the user defined availability threshold in percentage (filter2, years)
- **`dimension`** (*`pd.dataframe`*) – dataframe containing the values of a characteristic parameter through time and for multiple transect locations.

Returns dimension_filt: dataframe containing the values of a characteristic parameter where filtered years have been removed.

Return type pd.dataframe

`JAT.Filtering_functions.`**`locations_filter`**`(`*dimension*, *filter_file*`)`

Filter out user-defined transects.

Filter out locations that are specified by the user from a dataframe of a characteristic parameter. Default settings filter out locations like the Hondsbossche Dunes and Maasvlakte, redundant transects at the outer edges of the Wadden Islands, and dams in Zeeland.

Parameters

- **`dimension`** (*`pd.dataframe`*) – dataframe containing the values of a characteristic parameter through time and more multiple transect locations.
- **`filter_file`** (*`dict`*) – Includes a numbered list of sections that should be excluded. The first transect number represents the start of the section, the second transect number the end.

Returns dimension_filt: dataframe containing the values of a characteristic parameter where filtered transects have been removed.

Return type pd.dataframe

`JAT.Filtering_functions.`**`nourishment_filter`**`(`*config*, *dimension*`)`

Split characteristic parameter values into nourished and not nourished transects.

Parameters

- **`config`** (*`dict`*) – configuration file that includes the directory where the nourishment database is stored.
- **`dimension`** (*`pd.dataframe`*) – dataframe containing the values of a characteristic parameter through time and more multiple transect locations.

Returns

- *pd.dataframe* – nourished_dataframe: dataframe containing the values of a characteristic parameter of only transect that have been nourished.
- *pd.dataframe* – not_nourished_dataframe: dataframe containing the values of a characteristic parameter of only transect that have not been nourished.

`JAT.Filtering_functions.`**`yrs_filter`**(*dimension*, *begin_year*, *end_year*)

Filter out user-defined years.

Filter values of a characteristic parameter that are associated with a range of years that is specified by the user.

Parameters

- **`dimension`** (*`pd.dataframe`*) – dataframe containing the values of a characteristic parameter through time and more multiple transect locations.
- **`begin_yr`** (*`int`*) – Start year of the range that should be filtered
- **`end_yr`** (*`int`*) – End year of the range that should be filtered

Returns dimension_filt: dataframe containing the values of a characteristic parameter where filtered years have been removed.

Return type pd.dataframe

1.6 Help

1.6.1 New to Python

If you are new to Python it might be that you are not able to follow the instructions in *Getting Started*. Here, a recommened set-up is presented, however, there are many different options and everyone has different needs and preferences so this is just a suggestion.

First of all, install Anaconda! You can find an installation guide here: https://docs.anaconda.com/anaconda/install/windows/.

As you may or may not know Python works with packages. These packages provide certain functionalities, for instance, installing the package matplotlib provides a library of functions that help to create graphs and plots. Often a package is dependent on other packages to be able to work. For instance, the package numpy provides scientific computing with Python and you can imagine that you will need this package for matplotlib to function. The packages that are needed are called the *dependencies*. The JAT has several specific dependencies which are listed in *Getting Started*:.

It might be that the specific packages that you need for the JAT do not go together with another project that you are working on that, for instance, works with a newer version of matplotlib. This can occur because the available packages for python are constantly evolving. To avoid these type of incompatibilities Anaconda works with environments. In this case, it is recommended to make a jarkus specific environment (for instance with the name *jarkus*). When you have created the environment make sure that you use python 3.7 because the JAT is not compatible with version 3.8. The best guide to follow is provided here: https://docs.anaconda.com/anaconda/navigator/getting-started/.

The explanation on this web page relates to the Anaconda Navigator which is a nice visual way of looking at the environments. However, what you will often see is that people install packages using conda in the Anaconda prompt. That is also what we will need to do to install the JAT as explained in *Getting Started*:.

You can find information on managing environments in the Anaconda prompt here: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html. But don't get too lost in this! The only things you need to do is open the Anaconda Prompt and type the following:

```
$  activate jarkus
$  cd C:/JAT/setup.py
$  python setup.py install
```

First, this activate the environment that you just created (in this case *jarkus*). Note, that the dollar sign represents the blinking cursor in the Anaconda Prompt. Subsequently, the 'cd' command indicates that you want to change the folder that is open in the Anaconda prompt. When typing the 'cd' command follow it with the directory (in this case C:\JAT\setup.py) where the *setup.py* file is located. For the JAT this directory is dependent on where you have saved the JAT files, but it should end in . . . \JAT\setup.py.

These commands will install the Jarkus Analysis Toolbox and all its dependencies so it works in one go.

To work with the examples, Spyder is recommended. To make sure it is installed open the Anaconda Navigator and browse to the jarkus environment (Applications on . . .). Here, install Spyder and then launch it. In Spyder load, from example 1, *jarkus_01.yml* to see what settings are used and then open *JAT_use_single_transect.py* to see the steps necessary for the analysis. Spyder's walkthrough to get a feeling for how it works is recommended: https://docs.spyder-ide.org/current/index.html.

Often used short-cuts and features are:

- F5 for running all the code in one script (green play button)
- crtl+Return for running current sections denoted by %## statements (green play button in yellow/white marking)
- F9 for running one line of code or selected code
- The variable explorer

This is the point where you can start using the *Getting Started* and *Examples* sections again. If you run into other issues, try using Google/Stack Overflow because the internet knows a lot and other people often have experienced the same issues that you are running into.

1.6.2 Contact

If you find problems in the code of the Jarkus Analysis Toolbox please create an issue on Github. There, you can also ask questions, indicate that you want to contribute to the code or share ideas on the improvement and application of the JAT.

1.7 Development

1.7.1 Suggested improvements and additions

- Include Momentary Coastline Calculation and BKL
- Include Depth of Closure - both cross-shore and elevation - check with Nicha's work
- Include 2nd derivative method python version - based on current matlab-based method
- Include example of nourishment filter use - should work with .nc file on opendap

- Add active profile calculation based on landward variance boundary and DoC
- Add extraction of lon and lat - currently only cross-shore values are available after extraction
- Add … many other extraction methods that are available

1.7.2 Adding new extraction methods

The JAT currently provides a large range of extraction methods, but many more could be introduced. Below you find the most important steps for adding a new method to the JAT:

- Add the extraction method in the *Jarkus_analysis_toolbox.py* in class *Extraction* as *def get_parameter* (fill in appropriate name *for parameter*). It is best to use the other available extraction methods as inspiration so the method is comparable. For the assigment of the output into the dimensions dataframe use an appropriate variable name.
- Add an *if statement* for the extraction of the parameter in *get_all_dimensions* of *Extraction* class in *Jarkus_analysis_toolbox.py*
- Add the parameter name in the configuration file so the *if statement* (reference above) will work. This means the parameter name should be added in both dimensions→setting→variablename with a True/False statement, and in dimensions→variables→ variablename with the variable names as assigned in the *get_parameter* function, in the jarkus.yml file.
- Add these assigned variable names to the plot_tiles.yml file, so the distribution figures can be generated automatically. Note, that for cross-shore variables it is crucial to add an 'x' in the variable name, so it is picked up in the automatic normalization (Extraction → normalize_dimensions). Then, add a suitable title and label name for in the figure.
- Make sure to update the documentation!

PYTHON MODULE INDEX

INDEX