gtfs Documentation

The Public Knowledge Workshop

Jun 05, 2021

Output specifications

GTFS Utils is a utility for reading, parsing and aggregating GTFS data from Israel MOT.
GTFS Utils is part of Open-Bus project in The Public Knowledge Workshop (“ ”).

CHAPTER 1

Usage

To use GTFS Stats script, run:

```
python setup.py install
run_gtfs_stats
```

CHAPTER 2

Configuration

GTFS Utils requires a configuration file `config.json`.
An example file can be found in `gtfs_utils/config_example.json`.

2.1 bucket_valid_file_name_regexp

Use `bucket_valid_file_name_regexp` field in configuration to choose which dates to run the script on.
You can use a value such as `"2019-03-07.zip"` to run on a single date, or `"2019-05-\d\d\.zip"` to run on a full month, for example.

2.2 forward_fill

Set `forward_fill` field in configuration to `true` for performing forward fill for missing dates using existing files.

CHAPTER 3

Output format

GTFS Stats outputs stats for trips and for routes (*trip_stats* and *route_stats*, accordingly).
Each of them is output as a `pkl.gz` file (a gzipped pickle) of a Pandas DataFrame, and is located in the `output` directory, as defined in the configuration file.
The fields in each one of them are described in the output specifications below.

3.1 Output format as it is in Splunk

This section will be written soon.

3.1.1 Trip Stats

`gtfs_utils.core_computations.`**`compute_trip_stats`**(*feed: partridge.gtfs.feed, zones: pandas.core.frame.DataFrame, clusters: pandas.core.frame.DataFrame, trip_to_date: pandas.core.frame.DataFrame, date: datetime.date, source_files_base_name: List[str]*) → pandas.core.frame.DataFrame

Parameters

- **`feed`** – Partridge feed for the specific date
- **`zones`** – DataFrame with stop_code to zone_name mapping
- **`trip_to_date`** – trip_id_to_date information to match with the feed data
- **`date`** – The original schedule date
- **`source_files_base_name`** – The original zips the data is based on (GTFS, Tariff, etc.)

Raise pandas.MergeError if trip_id_to_date will not merge as 1:1 with trip data

Returns A DataFrame with columns as described below

Trip stats table has the following columns:

- `agency_id` - Agency identifier, as specified in *agency.txt* file.
- `agency_name` - The full name of the agency, as specified in *agency.txt* file.
- `all_stop_code` - All stop codes (as specified in *stops.txt* file), separated by semicolons.
- `all_stop_desc_city` - Cities of all stops of the trip (as described in *stop_desc* field in *stops.txt* file), separated by semicolons.
- `all_stop_id` - All stop identifiers (as specified in *stops.txt* file), separated by semicolons.
- `all_stop_latlon` - All stop waypoints (*stop_lat* and *stop_lon* as specified in *stops.txt* file), formatted as *lat,lon* and separated by semicolons.
- `cluster_id` - Cluster code, as in *ClusterId* in *ClusterToLine* file.
- `cluster_name` - The name of the cluster to which the line belongs, as in *ClusterName* in *ClusterToLine* file.
- `cluster_sub_desc` - A sub-cluster name to which the line is associated, as in *ClusterSubDesc* in *ClusterToLine* file.
- `date` - The original schedule date
- `direction_id` - Indicates the direction of travel for a trip, as specified in *trips.txt* file.
- `distance` - The full travel distance of the trip in meters, which is the maximal *shape_dist_traveled*, as specified in *stop_times.txt* file.
- `duration` - Duration of the trip in hours
- `end_stop_city` - The city of the last stop of the trip, as described in *stop_desc* field in *stops.txt* file.
- `end_stop_code` - Stop code of the last stop of the trip
- `end_stop_desc` - The description of the last stop of the trip, as described as *stop_desc* field in *stops.txt* file.
- `end_stop_id` - Stop ID of the last stop of the trip
- `end_stop_lat` - Latitude of the last stop of the trip
- `end_stop_lon` - Longitude of the last stop of the trip
- `end_stop_name` - Stop name of the last stop of the trip
- `end_time` - Departure time of the last stop of the trip
- `end_zone` - Zone name of the last stop of the trip
- `source_files` - The original the data is based on (GTFS, Tariff, etc.)
- `is_loop` - 1 if the start and end stop are less than 400m apart, otherwise 0
- `line_type` - Line type code, as in *LineType* in *ClusterToLine* file.
- **`line_type_desc` - Line type description, as in *LineTypeDesc* in *ClusterToLine* file. The options for this fields are:**
  - "" - Urban
  - "" - Intercity
  - "" - Regional

- num_stops - Number of stops in trip
- num_zones - Number of zones where the trip stops are. Zones are defined in the files in *Tariff.zip*.
- num_zones_missing - Number of stops whose identifier is missing from the files in *Tariff.zip*.
- route_alternative - A route's alternative identifier. Constructs a route identifier together with route_direction and route_mkt.
- route_direction - A route's direction identifier. Constructs a route identifier together with route_alternative and route_mkt.
- route_id - Route identifier, as specified in *routes.txt* file.
- route_long_name - The full name of a route, as specified in *routes.txt* file.
- route_mkt - MOT Line's 5-digit catalog number (""), a unique number at the line level, but not unique at the alternative level. Constructs a route identifier together with route_direction and route_alternative.
- route_short_name - The short name of a route, as specified in *routes.txt* file.
- **route_type - The type of transportation used on a route, as specified in *routes.txt*. In Israel, MOT uses:**
  - 0 for light train (Jerusalem Light Rail)
  - 2 for train (Israel Railways)
  - 3 for bus
  - 715 for Flexible Service Line (" ")
- shape_id - Shape identifier, as specified in *shapes.txt* file.
- source_files - base name of the files the data is based on (as they are saved on S3).
- speed - Average speed of the trip in meters per hour (calculated as *distance/duration*).
- start_stop_city - The city of the first stop of the trip, as specified in *stop_desc* field in *stops.txt* file.
- start_stop_code - Stop code of the first stop of the trip
- start_stop_desc - The description of the first stop of the trip, as described as *stop_desc* field in *stops.txt* file.
- start_stop_id - Stop ID of the first stop of the trip
- start_stop_lat - Latitude of the first stop of the trip
- start_stop_lon - Longitude of the first stop of the trip
- start_stop_name - Stop name of the first stop of the trip
- start_time - Departure time of the first stop of the trip
- start_zone - Zone name of the first stop of the trip
- trip_id - Trip identifier, as specified in *trips.txt* file.
- trip_id_to_date - Trip identifier that is unique for each day in week and departure hour.

3.1.2 Route Stats

gtfs_utils.core_computations.**compute_route_stats**(*trip_stats_subset: pandas.core.frame.DataFrame, date: datetime.date, source_files_base_name: List[str], headway_start_time: str = '07:00:00', headway_end_time: str = '19:00:00'*) → pandas.core.frame.DataFrame

Compute stats for the given subset of trips stats.

Parameters

- **trip_stats_subset** – Subset of the output of *compute_trip_stats()*
- **date** – The original schedule date
- **source_files_base_name** – The original zips the data is based on (GTFS, Tariff, etc.)
- **headway_start_time** – HH:MM:SS time string indicating the start time for computing headway stats
- **headway_end_time** – HH:MM:SS time string indicating the end time for computing headway stats

Returns A DataFrame with columns as described below

Route stats table has the following columns:

- `agency_id` - Same as in `gtfs_utils.compute_trip_stats()`
- `agency_name` - Same as in `gtfs_utils.compute_trip_stats()`
- `all_start_time` - All of the start times (formatted as HH:MM:SS) in which the trips in the route start, separated by semicolons
- `all_stop_code` - Same as in `gtfs_utils.compute_trip_stats()`
- `all_stop_desc_city` - Same as in `gtfs_utils.compute_trip_stats()`
- `all_stop_id` - Same as in `gtfs_utils.compute_trip_stats()`
- `all_stop_latlon` - Same as in `gtfs_utils.compute_trip_stats()`
- `all_stop_name` - Names of all stops of the trip (as described in *stop_name* field in *stops.txt* file), separated by semicolons
- `all_trip_id` - All of the identifiers (`trip_id`, as specified in *trips.txt* file) of the trips in the route, separated by semicolons
- `all_trip_id_to_date` - all the `trip_id_to_date` ids that match this route, separated by semicolon
- `cluster_id` - Same as in `gtfs_utils.compute_trip_stats()`
- `cluster_name` - Same as in `gtfs_utils.compute_trip_stats()`
- `cluster_sub_desc` - Same as in `gtfs_utils.compute_trip_stats()`
- `date` - Same as in `gtfs_utils.compute_trip_stats()`
- `end_stop_city` - Same as in `gtfs_utils.compute_trip_stats()`
- `end_stop_desc` - Same as in `gtfs_utils.compute_trip_stats()`
- `end_stop_id` - Same as in `gtfs_utils.compute_trip_stats()`

- `end_stop_lat` - Same as in `gtfs_utils.compute_trip_stats()`
- `end_stop_lon` - Same as in `gtfs_utils.compute_trip_stats()`
- `end_stop_name` - Same as in `gtfs_utils.compute_trip_stats()`
- `end_time` - Same as in `gtfs_utils.compute_trip_stats()`, referring to the last trip of the route
- `end_zone` - Same as in `gtfs_utils.compute_trip_stats()`
- `source_files` - Same as in `gtfs_utils.compute_trip_stats()`
- `is_bidirectional` - 1 if the route has trips in both directions, otherwise 0
- `is_loop` - Same as in `gtfs_utils.compute_trip_stats()`
- `line_type` - Same as in `gtfs_utils.compute_trip_stats()`
- `line_type_desc` - Same as in `gtfs_utils.compute_trip_stats()`
- `max_headway` - The maximal duration (in minutes) between trip starts on the route between `headway_start_time` and `headway_end_time`
- `mean_headway` - The mean duration (in minutes) between trip starts on the route between `headway_start_time` and `headway_end_time`
- `mean_trip_distance` - The full travel distance of each trip on the route in meters, which is the maximal *shape_dist_traveled*, as specified in *stop_times.txt* file (calculated as *service_distance/num_trips*)
- `mean_trip_duration` - Duration of each trip on the route in hours (calculated as *service_duration/num_trips*)
- `min_headway` - The minimal duration (in minutes) between trip starts on the route between `headway_start_time` and `headway_end_time`
- `num_stops` - Same as in `gtfs_utils.compute_trip_stats()`
- `num_trip_ends` - Number of trips on the route in the subset with non-null end times before 23:59:59
- `num_trip_starts` - Number of trips on the route in the subset with non-null start times
- `num_trips` - Number of trips on the route in the subset
- `num_zones` - Same as in `gtfs_utils.compute_trip_stats()`
- `num_zones_missing` - Same as in `gtfs_utils.compute_trip_stats()`
- `peak_end_time` - End time of first longest period during which the peak number of trips (`peak_num_trips`) occurs
- `peak_num_trips` - Maximal number of simultaneous trips in the service (for a given direction)
- `peak_start_time` - Start time of first longest period during which the peak number of trips (`peak_num_trips`) occurs
- `route_alternative` - Same as in `gtfs_utils.compute_trip_stats()`
- `route_direction` - Same as in `gtfs_utils.compute_trip_stats()`
- `route_id` - Same as in `gtfs_utils.compute_trip_stats()`
- `route_long_name` - Same as in `gtfs_utils.compute_trip_stats()`
- `route_mkt` - Same as in `gtfs_utils.compute_trip_stats()`
- `route_short_name` - Same as in `gtfs_utils.compute_trip_stats()`
- `route_type` - Same as in `gtfs_utils.compute_trip_stats()`

- `service_distance` - The full travel distance of all trips on the route in meters, which is the maximal *shape_dist_traveled*, as specified in *stop_times.txt* file.
- `service_duration` - Total duration of all trips on the route in hours
- `service_speed` - Average speed each trip on the route in km/h
- `source_files` - base name of the files the data is based on (as they are saved on S3).
- `start_stop_city` - Same as in `gtfs_utils.compute_trip_stats()`
- `start_stop_desc` - Same as in `gtfs_utils.compute_trip_stats()`
- `start_stop_id` - Same as in `gtfs_utils.compute_trip_stats()`
- `start_stop_lat` - Same as in `gtfs_utils.compute_trip_stats()`
- `start_stop_lon` - Same as in `gtfs_utils.compute_trip_stats()`
- `start_stop_name` - Same as in `gtfs_utils.compute_trip_stats()`
- `start_time` - Same as in `gtfs_utils.compute_trip_stats()`, referring to the first trip of the route
- `start_zone` - Same as in `gtfs_utils.compute_trip_stats()`

If `trip_stats_subset` is empty, return an empty DataFrame.

3.1.3 Configuration file

Config Example

An example for a minimal working config

```
{
  "files": {
    "base_directory": "",  # Fill with a full path to the download dir
    "child_directories": {
      "gtfs_feeds": "gtfs_feeds",
      "output": "output",
      "filtered_feeds": "filtered_feeds",
      "logs": "logs"
    },
    "output_file_name_regexp": "^(?P<date_str>[^_]+?)_(?P<type>\\w+)",
    "output_file_type": "csv.gz"
  },

  "s3": {
    "access_key_id": "Your Access key id",  # Fill with your key parameters
    "secret_access_key": "Your secret access key",  # Fill with your key parameters
    "s3_endpoint_url": "https://ams3.digitaloceanspaces.com",
    "bucket_name": "obus-do2",
  },

  "use_data_from_today": false,
  "date_range": ["2019-03-07", "2019-03-07"],
}
```

Parameters description

Main configuration object parameter

The configuration for a gtfs_stats run			
type	*object*		
properties			
• **files**	*Files object parameters*		
• **s3**	*S3 object parameters*		
• date_range	First and last date of the gtfs files to be download from bucket. only relevant if `use_data_from_today` is set to false. {Format: *YYYY-MM-DD*}		
	type	*array*	
	maxLength	2	
	minLength	2	
	items		
	•	type	*string*
		pattern	^d{4}-d{2}-d{2}$
	First and last date of the gtfs files to be download from bucket. only relevant if `use_data_from_today` is set to false. {Format: *YYYY-MM-DD*}		
	type	*array*	
	maxLength	2	
	minLength	2	
• use_data_from_today	Download only gtfs data from today (overrides date_range).		
	type	*boolean*	
	default	False	
• override_source_data_date	If set use the data from the same date for all analyzed dates		
	type	*string*	
	pattern	^d{4}-d{2}-d{2}$	
	default		
• max_gtfs_size_in_mb	Limit the maximum size of the downloaded gtfs files (in MB). If not set, the limit is only free disk space.		
	type	*integer*	
	default	sys.maxsize	
• display_download_progress_bar	If true, displays a proggress bar while downloading.		
	type	*boolean*	
	default	False	
• display_size_on_progress_bar	If true, displays file size on the download status bar.		
	type	*boolean*	
	default	False	
• delete_downloaded_gtfs_zip_files	If true, delete the gtfs files after parsing them.		
	type	*boolean*	
	default	True	
• force_existing_files	If true, force downloading files that already exist on disk.		
	type	*boolean*	
	default	False	
• write_filtered_feed	If true, writes a filtered version of the gtfs for the specific dates.		
	type	*boolean*	

Continued on next page

Table 1 – continued from previous page

	default	False
• console_verbosity	Lowest logging level to be printed to console.	
	type	*string*
	enum	DEBUG, INFO, WARNING, ERROR, CRITICAL
	default	'ERROR'
additionalProperties	False	
The configuration for a gtfs_stats run		
type	*object*	
anyOf	•	
	•	

Files object parameters

files info would be under the *files* tag.		
type	*object*	
properties		
• **base_directory**	base directory for the created files.	
	type	*string*
• **output_file_name_regexp**	A regular expression used to find existing output files.	
	type	*string*
• **output_file_type**	The file type for the outputs.	
	type	*string*
	enum	pkl.gz, csv.gz, csv
• **child_directories**	Names of dirs that will be used (name is relative to `base_directory`	
	type	*object*
	properties	
	• **gtfs_feeds**	The name of the directory the GTFS files would be downloaded to.
		type *string*
	• **output**	The name of the directory for the output files.
		type *string*
	• **filtered_feeds**	The name of the directory for the filtered feeds, if exists.
		type *string*
	• **logs**	The name of the directory for the log files.
		type *string*
	additionalProperties	False
	Names of dirs that will be used (name is relative to `base_directory`	
	type	*object*
additionalProperties	False	
files info would be under the *files* tag.		
type	*object*	

S3 object parameters

All the info about S3 connection parameters would come here (under *s3* tag).		
type	*object*	
properties		
• **access_key_id**	Authorization access key id for S3.	
	type	*string*
• **secret_access_key**	Authorization secret access key for S3.	
	type	*string*
• **s3_endpoint_url**	Connection endpoint url for S3.	
	type	*string*
• **bucket_name**	Bucket name for S3.	
	type	*string*
• upload_results	If true, upload the analyzed results back to S3.	
	type	*boolean*
• results_path_prefix	Prefix path on S3 for the uploaded results.	
	type	*string*
additionalProperties	False	
All the info about S3 connection parameters would come here (under *s3* tag).		
type	*object*	

Index

C