Required Libraries

- Numpy
- Pandas
- Matplotlib

Functions

smooth(data_frame, time_series_name, dt = 1, filter_width, plot_result = False)

Finds and smooths the distribution of a series representing arrival or departure times using filtering of the Fourier transform of the data. Plotting the before and after is optional. It returns a series showing the assumed number of trips that started or ended (depending on **time_series_name**) at each time period at intervals of **dt**.

- **data_frame** = Data frame to operate on
- **time_series_name** = Series to plot the distribution for (e.g. "trip_start_mam")
- **dt** = Minimum amount of time between time values
- **filter_width** = Width of Gaussian filter
- **plot_result** = Determines whether or not to plot the result

find_peaks(t, x, ranges = [('AM', 0, 12), ('PM', 12, 24)])

Finds the peaks within the specific time ranges using the maximum values in those ranges. Returns a dictionary with the keys being the time period name and the values being the time at the maximum, the maximum value itself (used for plotting), and the hh:mm time of the peak.

- **t**—Array of time values
- **x**—Array of travel demand values
- **ranges**—List of 3-element tuples. Each tuple represents a different time of day period. The first element is the name of the time period, the second is the start time (in hours), and the third is the end time.

plot_peaks(peaks, line_color, text_height)

Adds the peaks to a time distribution plot as dotted vertical lines with the times labeled.

- **peaks**—A dictionary describing varous peaks. The output of **find_peaks** works as an input to this function.
- **line_color**—A string specifying a line color
- **text_height**—Specifies the heights of the peak label over the x-axis

get_dt(array)

Finds the greatest common denominator of all of the elements in array, and assumes that as the time difference value. If an integer does not result an error is raised.

- **Array**—Array of numbers to compute the greatest common denominator of.

create_time_plot(df_column_pairs, series_titles, plot_title, series_colors, series_linestyles, file_name, filter_width, x_limits, y_limits, peak_ranges = None, text_heights = None, peaks_to_use = None)

This is sort of the "main" function of the script. It takes a specified list of series and plots the distributions of all of those on a single plot, along with peaks if a user wishes. The indices of all

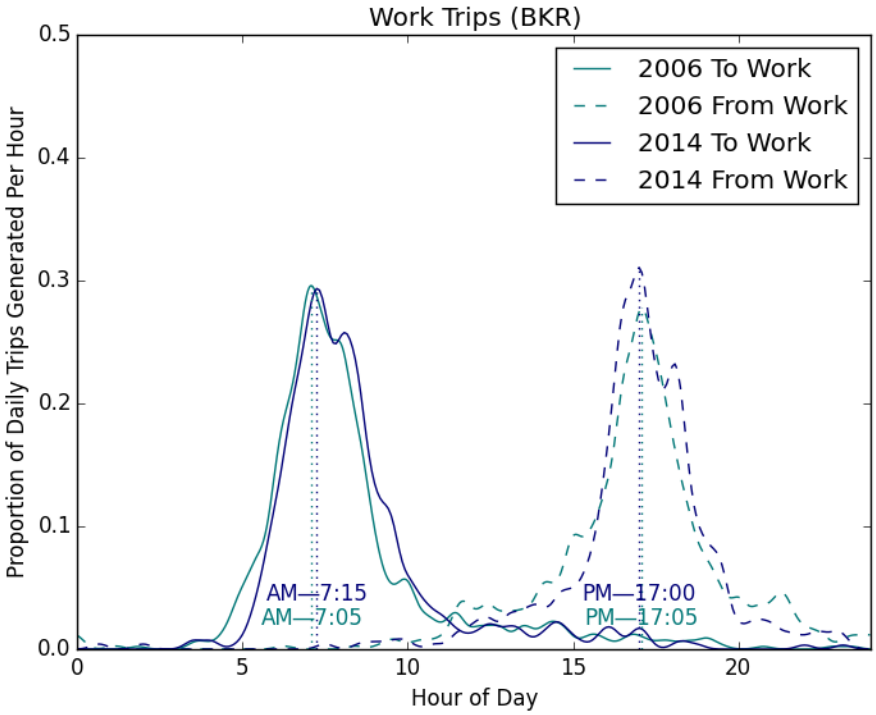
inputs in lists (other that **peaks_ranges** correspond to each other (there is the potential to make a single data frame as an input). The plot will be saved in a specified location.

- **df_column pairs**—A list of the series to be plotted
- **series_titles**—A list of the names of series (e.g. "HBW to Work")
- **plot_title**—The Title of the Plot
- **series_colors**—A list of colors that each series will be when plotted
- **series_linestyles**—A list of line styles that each series will be when plotted
- **file_name**—The output filepath
- **filter_width**—Width of the Gaussian filter while smoothing the plots
- **x_limits**—tuple describing the x-limits of the plot
- **y_limits**—tuple describing the y-limits of the plot
- **peak_ranges** (optional)—List of tuples describing the peaks (input to **find_peaks**)
- **text_heights** (optional)—List of the heights of the peak labels above the x-axis for each series
- **peaks_to_use** (optional)—If not all of the peaks want to be found for one of the series being plotted, this list specifies which peaks are to be found for that series. Example: if the specified peaks are [('AM', 0, 12), ('PM', 12, 24)], than if distributions for four series were being plotted, a **peaks_to_use** value of [0, 1, 0, 1] would mean to only find the AM peak for the first and third series and to only find the PM peak for the second and fourth.

Example code:

```
time_smooth.create_time_plot([(to_work_bkr_2006, 'begtime'), (from_work_bkr_2006,
'begtime'), (to_work_bkr_2014, 'time_start_mam'), (from_work_bkr_2014,
'time_start_mam')],
                             series_titles = ['2006 To Work', '2006 From Work', '2014 To
Work', '2014 From Work'],
                             plot_title = 'Work Trips (BKR)',
                             series_colors = ['#007777', '#007777', '#000077',
'#000077'],
                             series_linestyles = ['-', '--', '-', '--'],
                             file_name = figure_location + 'Work_Trips_BKR.png',
                             filter_width = 15.5,
                             x_limits = (0, 24), y_limits = (0, 0.5),
                             peak_ranges = [('AM', 6, 9), ('PM', 15, 18)],
                             text_heights = [0.02, 0.02, 0.04, 0.04],
                             peaks_to_use = [0, 1, 0, 1])
```

Output:



Demonstration of smoothing process

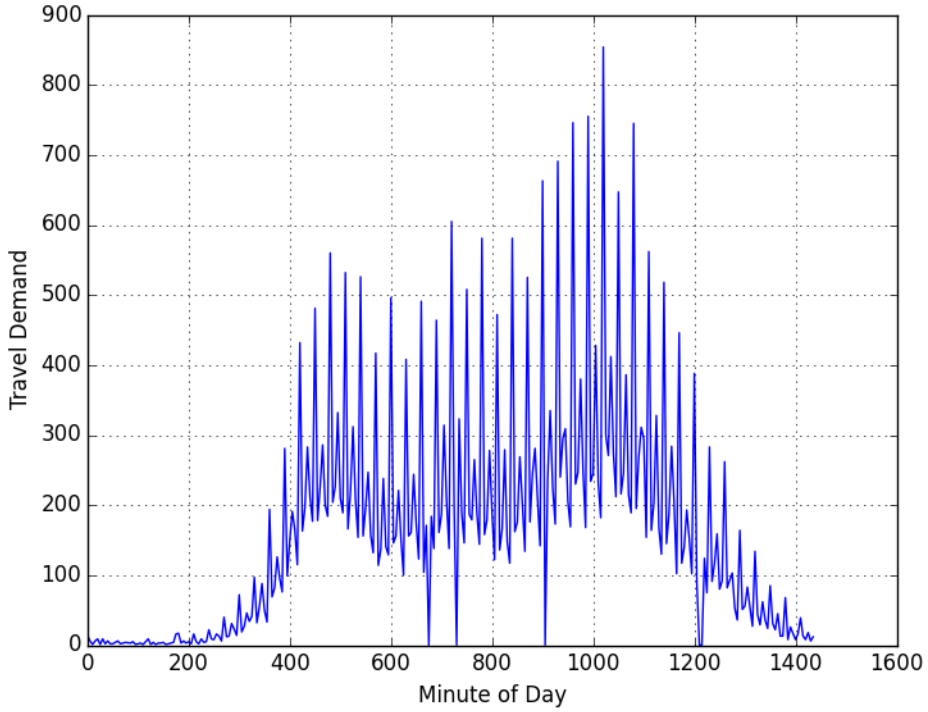


Figure 1: Unfiltered time distribution

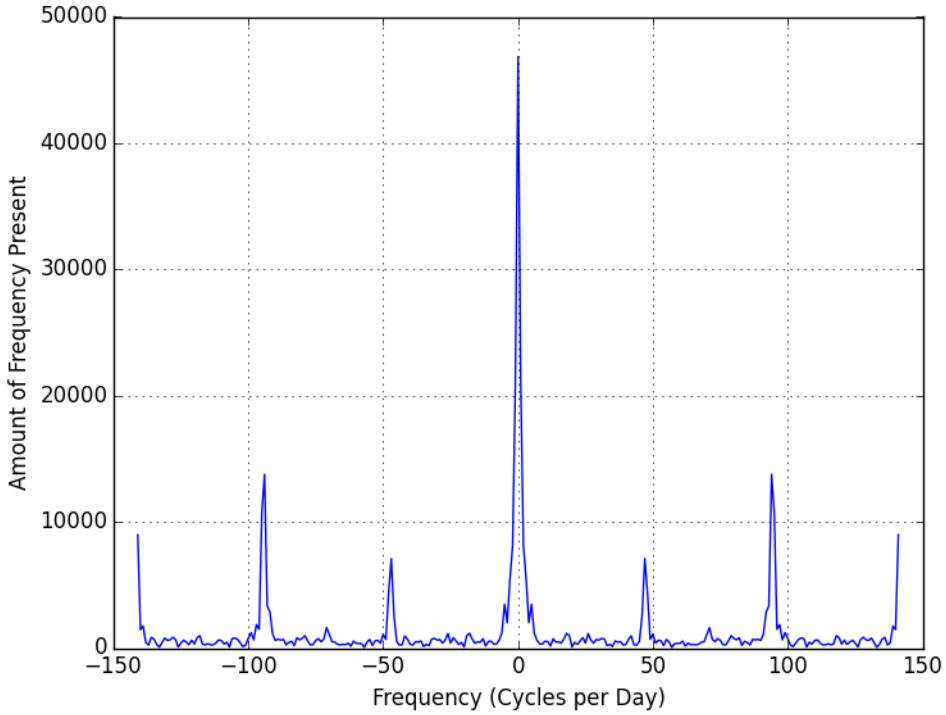


Figure 2: Data plotted in the frequency domain

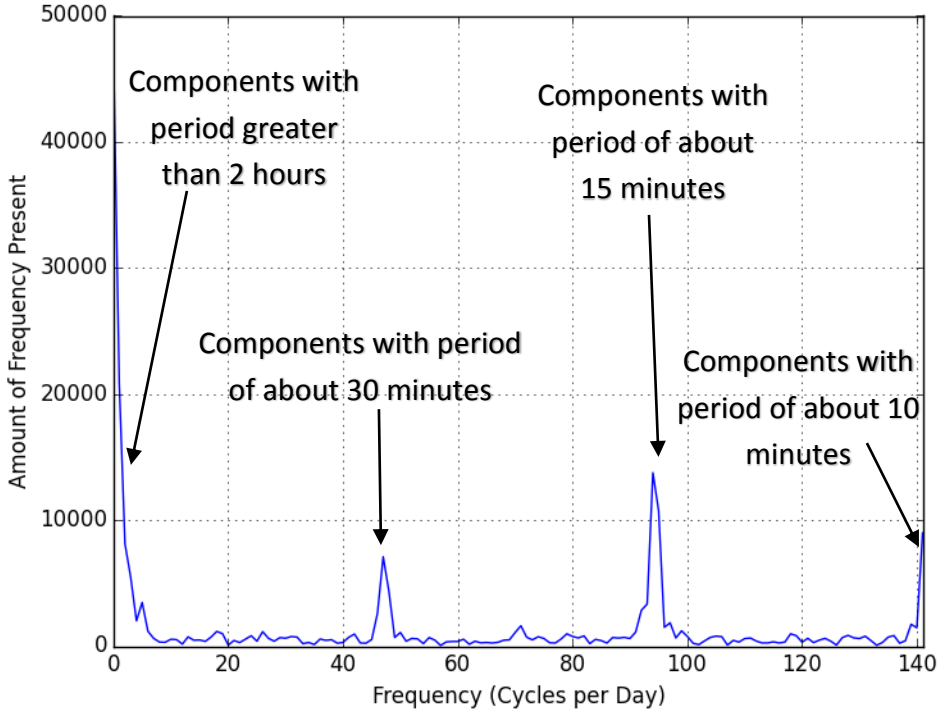


Figure 3: The positive half of the previous graph, with spikes highlighted.

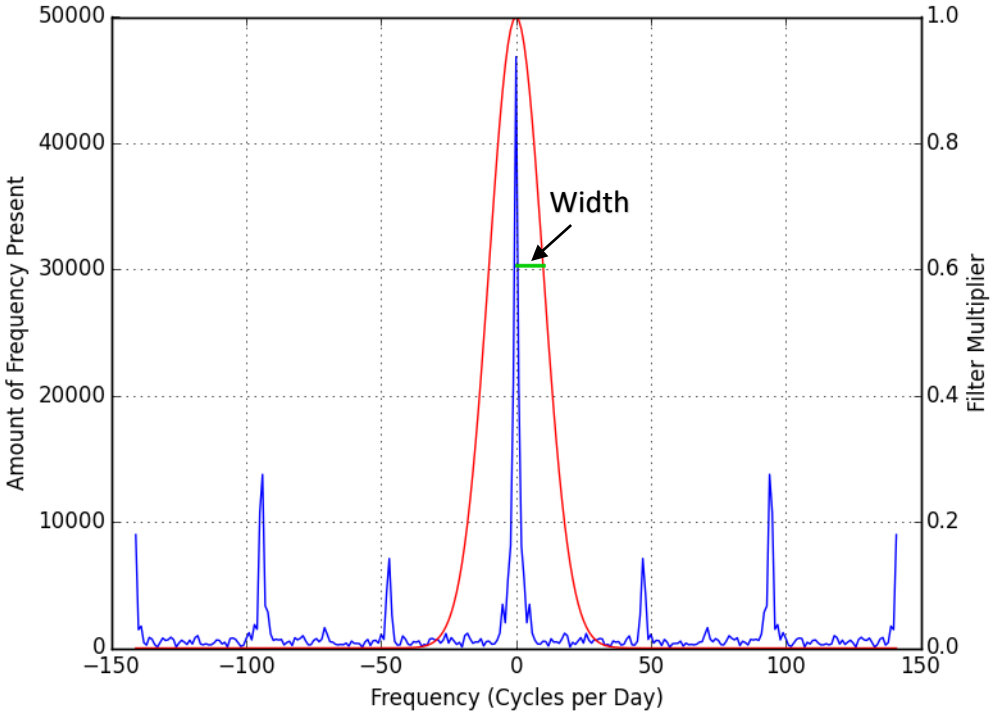


Figure 4: Data in the frequency domain with a Gaussian filter of width 10. The width parameter is shown with a green line segment. Width = 10 in this example.

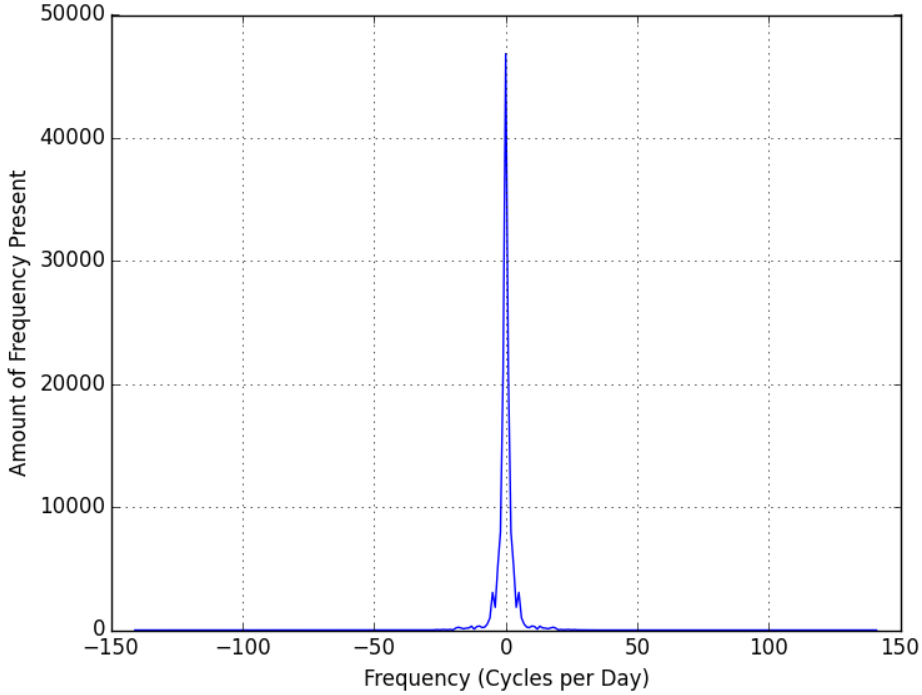


Figure 5: Data in the frequency domain multiplied by the filter. The components with low period are now gone.

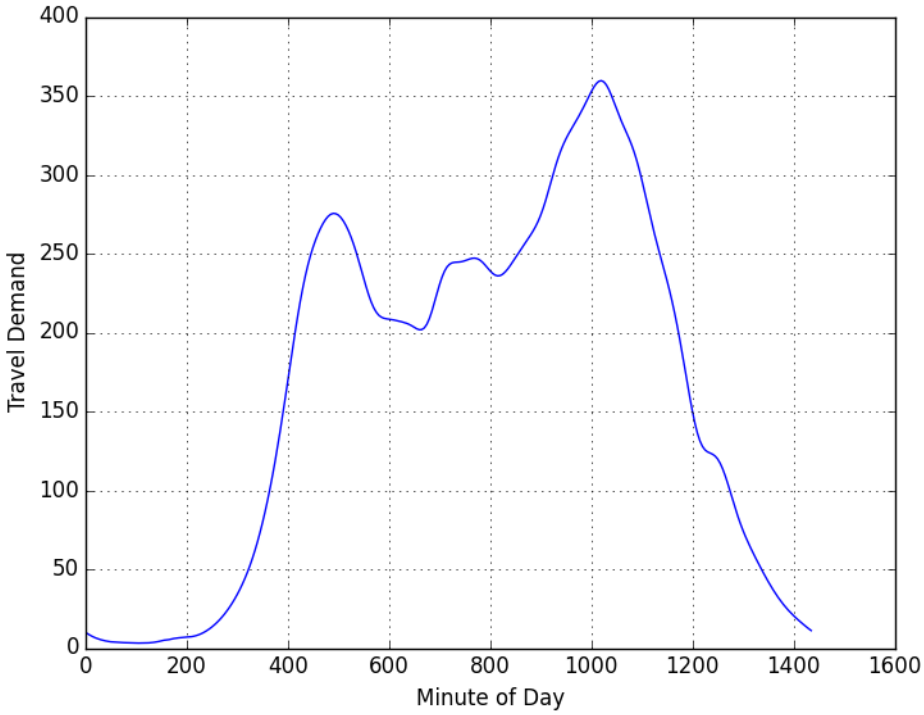


Figure 6: Filtered time distribution data