Analysis of Dwell Time Aggregations

A briefing on how dwell time can be aggregated including comparisons to Mainsail's method of aggregating dwell time. The report contains code snippets that may execute if the data files are in the same directory as the notebook. Executing the entire notebook took around 30 minutes on the author's machine.

Summary

The average dwell time is

$$\langle d \rangle_C = \frac{\sum_C d_c}{|C|}$$

where $\langle \ldots \rangle$ signifies an average; the subscript C on $\langle \ldots \rangle_C$ means the average is taken of the set of containers C; d_c is the dwell time of container c which is in the set of containers C; the set of containers C is the set of containers on the terminal at the time that the average dwell time is measured; and the count of containers in C is $|C|$. In short, it's the average amount of time that the containers have been on the terminal at a specific moment in time.

The dwell time d_c and the container count $|C|$ are both functions of time t. This means that the average dwell time $\langle d \rangle_C$ is a function of time too.

$$\langle d \rangle_C(t) = \frac{\sum_C d_c(t)}{|C(t)|}$$

The usual way to aggregate a metric that changes over time like the average dwell time $\langle d \rangle_C(t)$ is to average it over the period. This means that in this case the dwell time must be averaged over the set of containers C and over a period T to get the aggregated average dwell time $\langle d \rangle_{C,T}$ where the two subscripts C and T of $\langle \ldots \rangle_{C,T}$ mean that the dwell time has been averaged over two dimensions: the set of containers C and the period T.

The word 'aggregate' will be used instead of 'average' when working with time to avoid confusion. Four methods for aggregating the average dwell time over a period are examined:

- Sampling: This method samples the average dwell time at regular intervals based on the `sampling_interval` and averages the values.
- Integration: This method computes the area under the average dwell time curve and divides by the duration of the period.
- The ratio of averages: This method aggregates the average dwell time by computing the average total dwell time for all the containers for the period and then dividing by the

average container count for the period.
- The Mainsail metric: This method examines only containers that arrive and depart during the period and averages their dwell times.

For an in depth description of each method using an illustrative example, please see the section, "Illustrative Example."

Notation

Mathematical notation is kept to a minimum. The notations are listed here for easy reference.

- d: the dwell time
- d_c: the dwell time for a single container c in the set of containers C
- $d_c(t)$: the dwell time for a single container c at a moment in time t
- C: the set of containers on the terminal
- $C(t)$: the set of container on the terminal at time t
- $|C|$: the count of containers on the terminal at a given moment in time
- $\langle d \rangle$: the average dwell time
- $\langle d \rangle_C$: the average dwell time over the set of containers C
- $\langle d \rangle_{C,T}$: the average dwell time over the set of containers C and over the period T
- $\sum_C d_c$: the sum of the dwell times of every container in the set of containers C, that is, the total dwell time
- $\left\langle \sum_C d_c \right\rangle_T$: the average of the total dwell time over the period T
- $\langle C \rangle_T$: the average count of containers in the set C over the period T
- d_n: the dwell time of the nth sample
- N: the number of samples taken when using the sampling method
- $\int_0^T \langle d \rangle_C$: the area under the plot of $\langle d \rangle_C$
- $\langle d \rangle_D$: the average dwell time for the containers that arrived and departed during the period
- $|D|$: the count of containers that arrive and depart during the period
- w_c: the weight for container c, that is, the portion of the period that container c was on terminal
- T: the duration of the period
- $\frac{\sum_C w_c \langle d_c \rangle_T}{T}$: the average total dwell during the period
- $\frac{\sum_C w_c}{T}$: the average container count during the period
- $\frac{\sum_C w_c \langle d_c \rangle_T}{\sum_v w_v}$: the approximation of $\langle d \rangle_{C,T}$ given by the ratio method

Average Dwell Time Aggregations

The methods for aggregating the average dwell time $\langle d \rangle_C$ over a period T considered are:

- Sampling
- Integration
- The ratio of the averages
- The Mainsail metric

The preferred aggregation of the Crow's Nest team is the ratio of the averages. It permits the end user to filter the container set at runtime and is an excellent approximation to the exact value of the aggregated average dwell time $\langle d \rangle_{C,T}$.

Sampling is examined because it is easy to understand and because it is also an excellent approximation to the exact value of $\langle d \rangle_{C,T}$.

Integration is examined because it computes an exact value for $\langle d \rangle_{C,T}$. It is diffcult to understand, complex to implement, and does not permit the end user to filter the container set at runtime. However, for this document it provides significant value by providing the correct answer.

The Mainsail metric is examined to understand the differences between it and the values produced by the other three aggregations.

Methods

All aggregations have been written in Python as part of this notebook. All cells may be executed. Executing the entire notebook takes approximately thirty minutes.

The following cell imports some libraries that are used.

In [86]:
```
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import numpy as np
import datetime
```

The next cell defines the function that is used to perform the aggregations.

In [87]:
```
def dwell_times(container_times, sampling_interval, first_day, last_day, print_sett

    def plot_segments(segments, x1, x2, y1, y2, xTitle, yTitle, plotTitle, color, s
        # Create the plot object
        for i in range(len(segments)):
            plt.plot([segments.loc[i, x1], segments.loc[i, x2]]
                     , [segments.loc[i, y1], segments.loc[i, y2]]
                    , color=color)

        # Display the plot
        plt.xticks(rotation=90)
        plt.xlim(time_start, time_end)
        plt.xlabel(xTitle)
```

```
        plt.ylabel(yTitle)
        plt.title(plotTitle)
        if show:
            plt.show()
        return plt

    def return_zero_for_div_by_zero(numerators, divisors):
        return [num / div if div != 0 else 0 for num, div in zip(numerators, diviso

    def initial_dwell(box_schedule_df_row, start_time):
        if start_time >= box_schedule_df_row['departure']:
            return(0)
        elif start_time <= box_schedule_df_row['arrival']:
            return(0)
        else:
            return (start_time - box_schedule_df_row['arrival']).total_seconds()

    # The users can input a time table of arrivals and departures for containers. T
    # select dates that they are interested in. This is so that we can eventually s
    # all the containers that were on the terminal for those dates and report metri
    # dates instead of from the earliest event to the latest.

    # I had to remove Timedelta objects from some uses as they have a maximum value
    # days which is too small for all purposes. I use seconds as the unit for all t
    # and convert where necessary to days.

    # The integration method needs a dataframe with a grain of one row per interval
    # change events per day. If an interval stretches over two or more days, then i

    # The sampling method needs a dataframe with a grain of one row per sample.

    # The ratio method needs a dataframe with a grain of one row per container per

    # The mainsail method needs a dataframe with a grain of one row per container.
    # be used to get individual days.

    # Let's setup the user-input data.

    container_times_df = pd.DataFrame(container_times)
    if debug_setting:
        print(container_times_df)
    container_times_df['arrival'] = container_times_df['arrival'].apply(lambda x: p
    container_times_df['departure'] = container_times_df['departure'].apply(lambda
    container_times_df['dwell'] = (container_times_df['departure'] - container_time

    # And the rest of the user input
    sampling_timedelta = pd.Timedelta(hours=sampling_interval)
    start_time = pd.to_datetime(first_day)
    end_time = pd.to_datetime(last_day) + pd.Timedelta(days=1) - pd.Timedelta(secon
    if debug_setting:
        print('start_time: ', start_time)
        print('end_time: ', end_time)
    first_day = start_time.date()
    last_day = end_time.date()
    date_df = pd.DataFrame(pd.date_range(start=first_day, end=last_day), columns=['
    date_df['day_actual'] = date_df['day_actual'].apply(lambda x: x.date())
```

```
    period = (end_time - start_time).total_seconds()

    # To create the integration dataframe with a grain of one row per interval betw
    # events per day, I need to create an event list and then join it to itself, sh
    # to get intervals.

    # Building the integration dataframe first is a good choice. We'll use it to pr

    # Let's unpivot the input time table to get arrival and departure events.
    integration_df = pd.melt(
        pd.DataFrame(container_times_df)
        , id_vars=['container']
        , value_vars=['arrival', 'departure']
        , var_name='event_type'
        , value_name='event_time'
    )
    integration_df = integration_df.sort_values(by='event_time', ascending=True)
    integration_df['event_index'] = range(1, len(integration_df) + 1)
    integration_df['box_count_incr'] = (
        integration_df['event_type'].apply(lambda x: 1 if x == 'arrival' else -1)
    )
    integration_df['box_count'] = integration_df['box_count_incr'].cumsum()
    integration_df['event_start_day'] = pd.to_datetime(integration_df['event_time']
    integration_df['next_container'] = integration_df['container'].shift(-1)
    integration_df['next_event_type'] = integration_df['event_type'].shift(-1)
    integration_df['next_event_time'] = integration_df['event_time'].shift(-1)
    integration_df['event_end_day'] = pd.to_datetime(integration_df['next_event_tim
    # At this point, let's split the intervals by day.

    conn = sqlite3.connect(':memory:')

    integration_df.to_sql('integration_df', conn, index=False)
    date_df.to_sql('date_df', conn, index=False)
    query = '''
        select *
        from integration_df
        inner join date_df on
            date_df.day_actual between integration_df.event_start_day and integrati
    '''
    integration_df = pd.read_sql_query(query, conn)
    integration_df['event_index'] = integration_df['event_index'].astype(int)
    integration_df['event_time'] = integration_df['event_time'].apply(lambda x: pd.
    integration_df['event_start_day'] = integration_df['event_start_day'].apply(lam
    integration_df['next_event_time'] = integration_df['next_event_time'].apply(lam
    integration_df['event_end_day'] = integration_df['event_end_day'].apply(lambda
    integration_df['day_actual'] = integration_df['day_actual'].apply(lambda x: pd.
    integration_df = integration_df.sort_values(by=['event_index','day_actual'], as
    integration_df['event_index'] = range(1, len(integration_df) + 1)
    integration_df['interval_start_time'] = np.maximum(integration_df['event_time']
    integration_df['interval_end_time'] = np.minimum(integration_df['next_event_tim
    integration_df['duration'] = (integration_df['interval_end_time'] - integration
    # The dwell_rise is the amount of dwell time gained during the intervals. It's
    integration_df['dwell_rise'] = integration_df['duration'] * integration_df['box
    # Here I need to merge the container_times_df so that I can get the dwell for e
    # I need the dwell for each container for the entire visit so that I can subtra
    # when the container departs.
```

```
    integration_df = pd.merge(integration_df, container_times_df, on='container', h
    integration_df = integration_df.sort_values(by='event_index', ascending=True)
    integration_df['dwell_loss'] = (
        integration_df.apply(
            lambda row: -1 * row['dwell']
                if (row['event_type'] == 'departure') & (row['event_start_day'] ==
                else 0
            , axis=1
        )
    )
    # The dwell time increment for the interval is the sum of the dwell_rise and th
    integration_df['dwell_incr'] = integration_df['dwell_rise'] + integration_df['d
    # The cumulative sum is a handy way to get the total dwell at the end of the in
    # Since we don't start with an empty yard, we need to add the initial total dwe
    container_times_df['initial_dwell'] = container_times_df.apply(lambda row: init
    initial_total_dwell = container_times_df['initial_dwell'].sum()
    if debug_setting:
        print('integration_df:\n', integration_df, '\n')
        print('initial_total_dwell:\n', initial_total_dwell, '\n')
    integration_df['ending_dwell'] = integration_df['dwell_incr'].cumsum() + initia
    # And the starting dwell can be obtained easily enough be subtracting the dwell
    integration_df['starting_dwell'] = (integration_df['ending_dwell'] - integratio
    # The average dwell at the end points of the interval can be computed from the
    # by dividing by the box count.
    integration_df['starting_avg_dwell'] = integration_df['starting_dwell'] / integ
    integration_df['ending_avg_dwell'] = integration_df['ending_dwell'] / integrati
    integration_df['avg_dwell_rise'] = integration_df['ending_avg_dwell'] - integra

    # Let's produce the plots now.
    segments = integration_df.copy()
    segments = segments.drop(columns=[
                        'container','event_type','box_count_incr'
                        ,'next_container','next_event_type','duration','dwell_rise'
                        ,'arrival','departure','dwell','dwell_loss'
                        ,'dwell_incr','day_actual'
                        ,'event_time','event_start_day','next_event_time','event_en

    # Convert to days
    segments['starting_dwell_days'] = segments['starting_dwell']/86400
    segments['ending_dwell_days'] = segments['ending_dwell']/86400


    if print_setting:
        # Total dwell time plot
        plot_segments(segments, 'interval_start_time', 'interval_end_time', 'starti
                      'Time', 'Total Dwell (days)', 'Total Dwell Time Over Time', '

        # Box count plot
        plot_segments(segments, 'interval_start_time', 'interval_end_time', 'box_co
                      'Time', 'Container Count', 'Container Count Over Time', '#364
        #print('avg box count:\n', segments['box_count'].mean())

    # Let's produce the sampling times and sampled avg dwells for overlay on the av
    # The sample_offset let's us sample in the middle of the interval instead of at
    sample_offset = sampling_timedelta / 2
    sampling_times = pd.date_range(start=(start_time + sample_offset), end=end_time
```

```
    sampling_df = pd.DataFrame(sampling_times, columns=['sampling_time'])
    # Joining the samples to the segments
    sampling_df.to_sql('sampling_df', conn, index=False)
    if debug_setting:
        print('segments:\n', segments, '\n')
    segments.to_sql('segments', conn, index=False)
    query = '''
        select *
        from sampling_df
        left join segments on
            sampling_df.sampling_time >= segments.interval_start_time
            and sampling_df.sampling_time < segments.interval_end_time
    '''
    sampling_df = pd.read_sql_query(query, conn)
    # Use linear interpolation to obtain the sampled avg dwell time
    sampling_df['sample_dwell'] = (
        sampling_df['starting_avg_dwell'] +
        (sampling_df['sampling_time'].apply(pd.to_datetime) -
         sampling_df['interval_start_time'].apply(pd.to_datetime)).apply(lambda x:
    )
    sampling_df['sampling_time'] = pd.to_datetime(sampling_df['sampling_time'])
    # If there's no data, let the sample be 0.
    sampling_df['sample_dwell'] = sampling_df['sample_dwell'].fillna(0)

    # Avg dwell time plot
    if print_setting:
        segments['starting_avg_dwell_days'] = segments['starting_avg_dwell']/86400
        segments['ending_avg_dwell_days'] = segments['ending_avg_dwell']/86400
        avg_dwell_plot = plot_segments(segments, 'interval_start_time', 'interval_e
                       'Time', 'Avg Dwell (days)', 'Avg Dwell Time Over Time', '#364

        sampling_df['sample_dwell_days'] = sampling_df['sample_dwell']/86400
        if overlay_setting:
            avg_dwell_plot.scatter(sampling_df['sampling_time'], sampling_df['sampl
        else:
            avg_dwell_plot.show()

    ## Now time to compute results
    # results_df = pd.DataFrame(columns=['Method', 'Timeframe', 'Value'])

    # Integration results
    integration_df['area'] = (
        integration_df['starting_avg_dwell'] * integration_df['duration']
            + 0.5 * integration_df['duration'] * integration_df['avg_dwell_rise']
        # Need avg_dwell_rise here because if there are not boxes, it's zero.
    )
    if debug_setting:
        with pd.option_context('display.max_columns', None):
            print('integration_df:\n')
            print(integration_df)
    integration_result_all = integration_df['area'].sum() / (end_time - start_time)

    daily_integration_results = integration_df.groupby('day_actual')['area'].sum().
    daily_integration_results['avg_dwell'] = daily_integration_results['area']/8640
    daily_integration_results['Method'] = 'Integration'
    daily_integration_results['Timeframe'] = daily_integration_results['day_actual'
```

```
    daily_integration_results['Avg Dwell'] = daily_integration_results['avg_dwell']
    daily_integration_results = daily_integration_results.drop(columns=['day_actual

    integration_result_avg = daily_integration_results['Avg Dwell'].mean()

    results_df = pd.DataFrame([{'Method': 'Integration', 'Timeframe': 'Entire Perio
    results_df = pd.concat([results_df, daily_integration_results], ignore_index =
    results_df = pd.concat(
        [results_df
         , pd.DataFrame([{
             'Method': 'Integration'
             , 'Timeframe': 'Avg Over Days'
             , 'Avg Dwell': integration_result_avg
         }])]
        , ignore_index = True
    )

    # Sampling results
    sampling_result = sampling_df['sample_dwell'].mean()

    sampling_df['day_actual'] = sampling_df['sampling_time'].apply(lambda x: x.date
    daily_sampling_results = sampling_df.groupby('day_actual')['sample_dwell'].mean
    daily_sampling_results = daily_sampling_results.rename(columns={'day_actual': '
    daily_sampling_results['Avg Dwell'] = daily_sampling_results['Avg Dwell']
    daily_sampling_results['Method'] = 'Sampling'

    daily_sampling_avg = daily_sampling_results['Avg Dwell'].mean()
    if debug_setting:
        print('sampling_df:\n', sampling_df)

    results_df = pd.concat(
        [results_df
         , pd.DataFrame([{
             'Method': 'Sampling'
             , 'Timeframe': 'Entire Period'
             , 'Avg Dwell': sampling_result
         }])]
        , ignore_index = True
    )
    results_df = pd.concat([results_df, daily_sampling_results], ignore_index = Tru
    results_df = pd.concat(
        [results_df
         , pd.DataFrame([{
             'Method': 'Sampling'
             , 'Timeframe': 'Avg Over Days'
             , 'Avg Dwell': daily_sampling_avg
         }])]
        , ignore_index = True
    )

    # Ratio results
    # The ratio method needs a dataframe with a grain of one row per container per
    ratio_df = container_times_df.copy()
    ratio_df['dwell'] = ratio_df['dwell']
    ratio_df['dwell_start_day'] = ratio_df['arrival'].apply(lambda x: x.date())
    ratio_df['dwell_end_day'] = ratio_df['departure'].apply(lambda x: x.date())
```

```
    if debug_setting:
        print('ratio_df:\n', ratio_df)
    ratio_df.to_sql('ratio_df', conn, index=False)
    query = '''
        select ratio_df.*, date(date_df.day_actual) as day_actual
        from ratio_df
        inner join date_df on date(date_df.day_actual) between ratio_df.dwell_start
    '''
    ratio_df = pd.read_sql_query(query, conn)
    ratio_df['arrival'] = ratio_df['arrival'].apply(lambda x: pd.to_datetime(x))
    ratio_df['departure'] = ratio_df['departure'].apply(lambda x: pd.to_datetime(x)
    ratio_df['dwell'] = ratio_df['dwell']
    ratio_df['dwell_start_day'] = ratio_df['dwell_start_day'].apply(lambda x: pd.to
    ratio_df['dwell_end_day'] = ratio_df['dwell_end_day'].apply(lambda x: pd.to_dat
    ratio_df['day_actual'] = ratio_df['day_actual'].apply(lambda x: pd.to_datetime(
    ratio_df['end_of_day_actual'] = ratio_df['day_actual'] + pd.Timedelta(days=1) -
    ratio_df['daily_duration_start'] = ratio_df[['arrival', 'day_actual']].max(axis
    ratio_df['daily_duration_end'] = ratio_df[['departure', 'end_of_day_actual']].m
    ratio_df['daily_initial_dwell'] = (ratio_df['daily_duration_start'] - ratio_df[
    ratio_df['daily_box_duration'] = (ratio_df['daily_duration_end'] - ratio_df['da
    # The avg dwell is the (initial dwell + the final dwell) / 2.
    # The final dwell is the (initial dwell + the daily box duration).
    # Substituting, the avg dwell is the initial dwell + the daily box duration / 2
    ratio_df['daily_avg_dwell'] = ratio_df['daily_initial_dwell'] + ratio_df['daily
    ratio_df['daily_box_dwell_areas'] = ratio_df['daily_avg_dwell'] * ratio_df['dai
    if debug_setting:
        print('ratio_df:\n', ratio_df)

    ratio_result = ratio_df['daily_box_dwell_areas'].sum() / ratio_df['daily_box_du
    results_df = pd.concat(
        [results_df
         , pd.DataFrame([{
             'Method': 'Ratio'
             , 'Timeframe': 'Entire Period'
             , 'Avg Dwell': ratio_result
         }])]
        , ignore_index = True
    )

    daily_ratio_results = ratio_df.groupby('day_actual').agg({'daily_box_dwell_area
    daily_ratio_results['Avg Dwell'] = daily_ratio_results['daily_box_dwell_areas']
    daily_ratio_results = daily_ratio_results.drop(columns=['daily_box_dwell_areas'
    daily_ratio_results = daily_ratio_results.rename(columns={'day_actual': 'Timefr
    daily_ratio_results['Method'] = 'Ratio'
    daily_ratio_results['Timeframe'] = daily_ratio_results['Timeframe'].apply(lambd
    if debug_setting:
        print('daily_ratio_results:\n', daily_ratio_results)
    results_df = pd.concat([results_df, daily_ratio_results], ignore_index = True)

    daily_ratio_avg = daily_ratio_results['Avg Dwell'].mean()
    results_df = pd.concat(
        [results_df
         , pd.DataFrame([{
             'Method': 'Ratio'
             , 'Timeframe': 'Avg Over Days'
             , 'Avg Dwell': daily_ratio_avg
```

```
        }])]
    , ignore_index = True
)

# Mainsail metric computations
mainsail_df = container_times_df.copy()
# Mainsail truncates the arrivals and departures to compute the dwell
mainsail_df['arrival'] = mainsail_df['arrival'].apply(lambda x: x.normalize())
mainsail_df['departure'] = mainsail_df['departure'].apply(lambda x: x.normalize
mainsail_df['dwell'] = (mainsail_df['departure'] - mainsail_df['arrival']).appl
if debug_setting:
    print('mainsail_df:\n', mainsail_df)
mainsail_df = (
    mainsail_df[
        (mainsail_df['arrival'] >= start_time)
        & (mainsail_df['departure'] <= end_time)
    ]
)
if debug_setting:
    print('mainsail_df:\n', mainsail_df)
    print('mainsail total dwell = ', mainsail_df['dwell'].sum())
    print('mainsail box count = ', len(mainsail_df))
mainsail_result = mainsail_df['dwell'].mean()
results_df = pd.concat(
    [results_df
     , pd.DataFrame([{
         'Method': 'Mainsail'
         , 'Timeframe': 'Entire Period'
         , 'Avg Dwell': mainsail_result
     }])]
    , ignore_index = True
)

mainsail_df = (
    mainsail_df[
        (mainsail_df['arrival'].apply(lambda x: x.date()) == mainsail_df['depar
        & (mainsail_df['arrival'] >= start_time)
        & (mainsail_df['departure'] <= end_time)
    ]
)
mainsail_df['day_actual'] = mainsail_df['arrival'].apply(lambda x: x.date())
mainsail_daily_results = mainsail_df.groupby('day_actual')['dwell'].mean().rese
# I want missed days to be zero.
mainsail_daily_results = date_df.merge(mainsail_daily_results, on='day_actual',
mainsail_daily_results['dwell'] = mainsail_daily_results['dwell'].fillna(0)
mainsail_daily_results = mainsail_daily_results.rename(columns={'day_actual': '
mainsail_daily_results['Method'] = 'Mainsail'
mainsail_daily_results['Avg Dwell'] = mainsail_daily_results['Avg Dwell']
results_df = pd.concat([results_df, mainsail_daily_results], ignore_index = Tru
if debug_setting:
    print('mainsail_daily_results:\n', mainsail_daily_results)
daily_mainsail_avg = mainsail_daily_results['Avg Dwell'].mean()
results_df = pd.concat(
    [results_df
     , pd.DataFrame([{
         'Method': 'Mainsail'
```

```
            , 'Timeframe': 'Avg Over Days'
            , 'Avg Dwell': daily_mainsail_avg
        }])]
        , ignore_index = True
    )

    results_df['Avg Dwell'] = results_df['Avg Dwell']/86400

    if debug_setting:
        print('mainsail_daily_results:\n', mainsail_daily_results)

    return(results_df)
```

To use the function a schedule of arrivals and departures for containers must be defined in `container_times`; a time interval in hours between successive samples must be defined as `sampling_interval`; and the first and last day of the period of interest must be defined as `first_day` and `last_day`.

The settings determine how verbose the output is.

- `print_setting` causes the charts to be output if set to `True`
- `debug_setting` prints out a large amount of internal variable data if set to `True`
- `overlay_setting` causes the samples to be overlaid on the average dwell time chart if set to `True`

Production Data Example

Six weeks of production data was obtained from the T18 TOS via two methods. The data was obtained by running the YRD120R Mainsail report for the dates from September 29, 2024 through November 9, 2024. The container number and arrival timestamps were concatenated to provide a unique identifier and only the arrival and departure timestamps were retained from the report's dataset. The data is available as `.\YRD120R.csv`.

In [88]:
```
yrd120r_data = pd.read_csv('file:///synfs/nb_resource/builtin/YRD120R.csv', dtype=s
yrd120r_data
```

Out[88]:

	container	arrival	departure
0	AAMU2609903_2024-10-04 00:50:45	10/4/2024 00:50:45	10/13/2024 14:36:03
1	AAMU2609950_2024-10-18 08:53:43	10/18/2024 08:53:43	10/18/2024 13:25:19
2	ACCU2313141_2024-10-25 01:51:50	10/25/2024 01:51:50	10/29/2024 13:11:34
3	ACCU2315632_2024-10-20 17:46:54	10/20/2024 17:46:54	10/24/2024 12:35:40
4	AFFU2000420_2024-10-20 21:44:50	10/20/2024 21:44:50	10/23/2024 17:05:14
...	...	...	...
42797	ZTCU4504280_2024-10-23 20:41:34	10/23/2024 20:41:34	10/26/2024 12:16:55
42798	ZTCU4504295_2024-10-23 21:40:14	10/23/2024 21:40:14	10/26/2024 12:16:56
42799	ZXJU0118489_2024-09-30 12:16:02	9/30/2024 12:16:02	10/10/2024 13:31:33
42800	ZXJU0146726_2024-10-27 11:16:19	10/27/2024 11:16:19	10/30/2024 10:56:52
42801	ZXJU0146808_2024-10-27 13:31:33	10/27/2024 13:31:33	10/28/2024 16:40:27

42802 rows × 3 columns

The data was also obtained by running the following query against the `gold_container_yard_inventory_daily` table in the Crow's Nest data warehouse.

```
select distinct
    equipment_id || '_' || in_yard_at as container
    , in_yard_at as arrival
    , coalesce(out_yard_at, dateadd('day', 1, current_date())) as
departure
from gold.operations.gold_container_yard_inventory_daily
where terminal_key = 'T18'
    and date_actual between to_date('2024-09-29') and to_date('2024-11-
09')
order by container
;
```

The query selects all records of containers on T18 during the period from September 29, 2024, and November 9, 2024. Each container is represented once for each day it is present on the terminal. The results set must therefore be narrowed to only the distinct values of containers, arrivals and departures. The Crow's Nest data is identified by the concatenation of the container number and the arrival time and only the arrival and departure timestamps are kept as fields in the same fashion as the YRD120R data. The data is available as `.\crows_nest_t18_yard.csv`.

In [89]:
```
crows_nest_data = pd.read_csv('file:///synfs/nb_resource/builtin/crows_nest_t18_yar
crows_nest_data
```

Out[89]:

	container	arrival	departure
0	7262620_2023-05-19 09:50:07.000	2023-05-19 09:50:07.000	2024-12-17 00:00:00.000
1	AACU4141663_2024-09-23 10:29:28.000	2024-09-23 10:29:28.000	2024-10-05 22:39:49.000
2	AAMU2609903_2024-10-04 00:50:45.000	2024-10-04 00:50:45.000	2024-10-13 14:36:08.000
3	AAMU2609950_2024-10-18 08:53:43.000	2024-10-18 08:53:43.000	2024-10-18 13:25:27.000
4	AAMU2612152_2024-09-18 23:14:49.000	2024-09-18 23:14:49.000	2024-10-04 15:20:11.000
...	...	...	...
61283	ZXJU0142994_2024-08-31 15:49:09.000	2024-08-31 15:49:09.000	2024-09-30 14:09:04.000
61284	ZXJU0146726_2024-10-27 14:37:02.000	2024-10-27 14:37:02.000	2024-10-30 10:56:53.000
61285	ZXJU0146808_2024-10-27 14:39:58.000	2024-10-27 14:39:58.000	2024-10-28 16:40:27.000
61286	ZXJU0154932_2024-08-20 09:46:15.000	2024-08-20 09:46:15.000	2024-09-29 14:49:21.000
61287	ZXJU0155498_2024-08-20 09:48:35.000	2024-08-20 09:48:35.000	2024-09-29 14:49:21.000

61288 rows × 3 columns

Note that the Crow's Nest data has 61,288 rows and the Mainsail data has 42,802 rows. Approximately 20,000 container arrivals have been excluded from the Mainsail data that are included in the Crow's Nest data. These are containers that arrived before and departed after September 29 or have not yet departed at all or are containers that arrived before and departed after November 9 or have not yet departed at all. Some of the containers are very long dwelling containers that have a strong impact on the average dwell time if included.

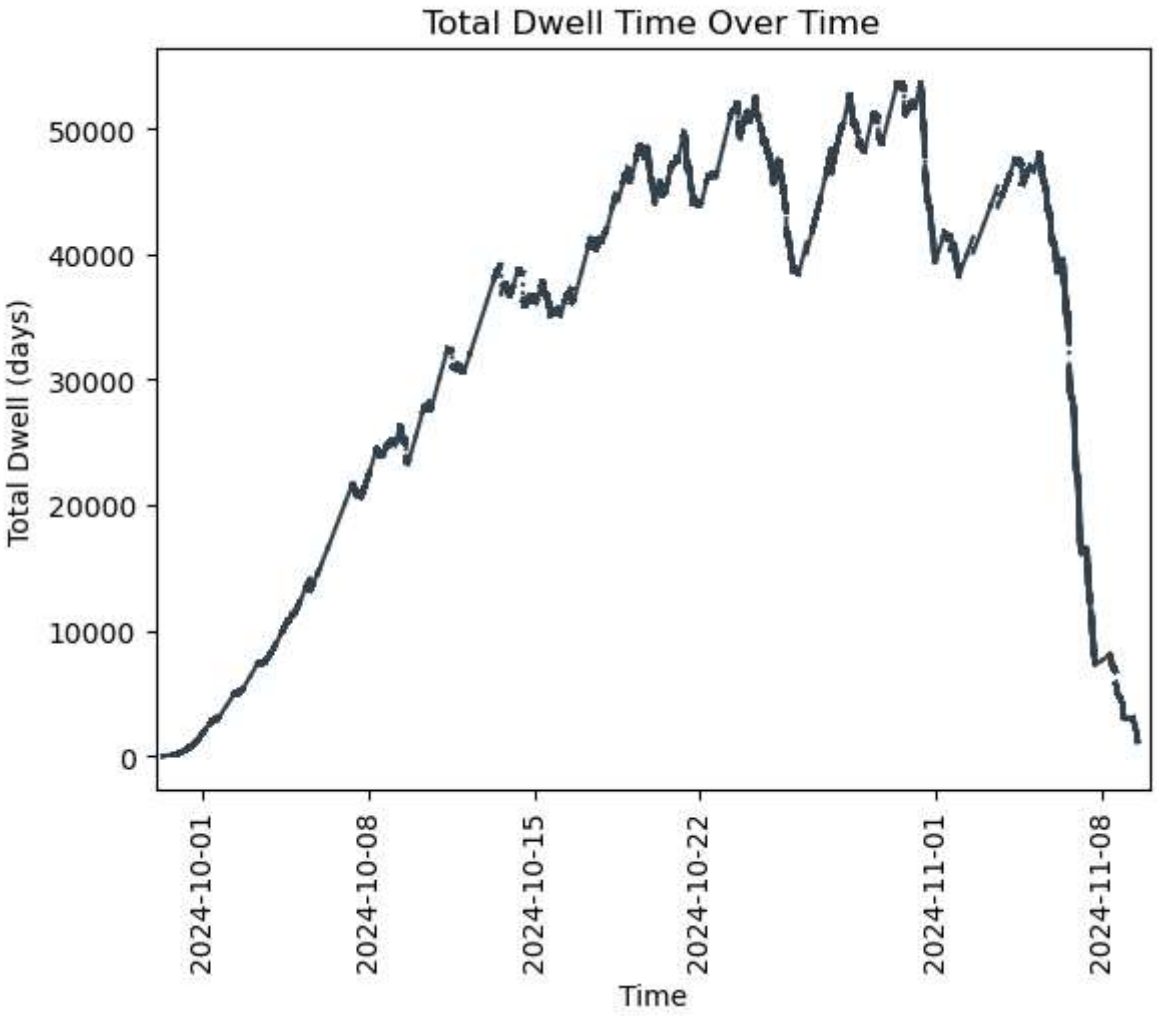
YRD120R data analysis

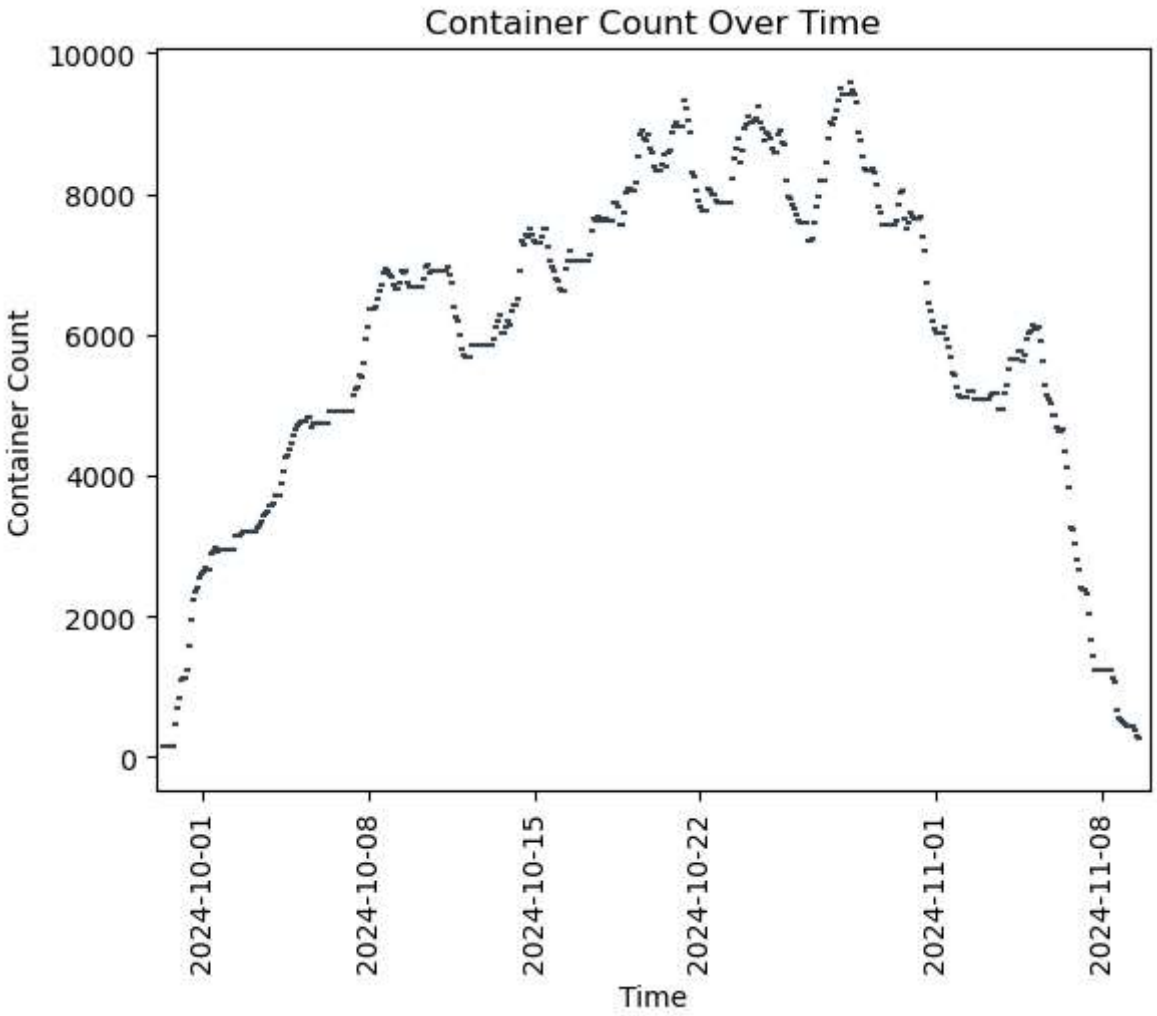
To run the YRD120R data through the methods in this notebook, set the parameters for calling the function and then call it.

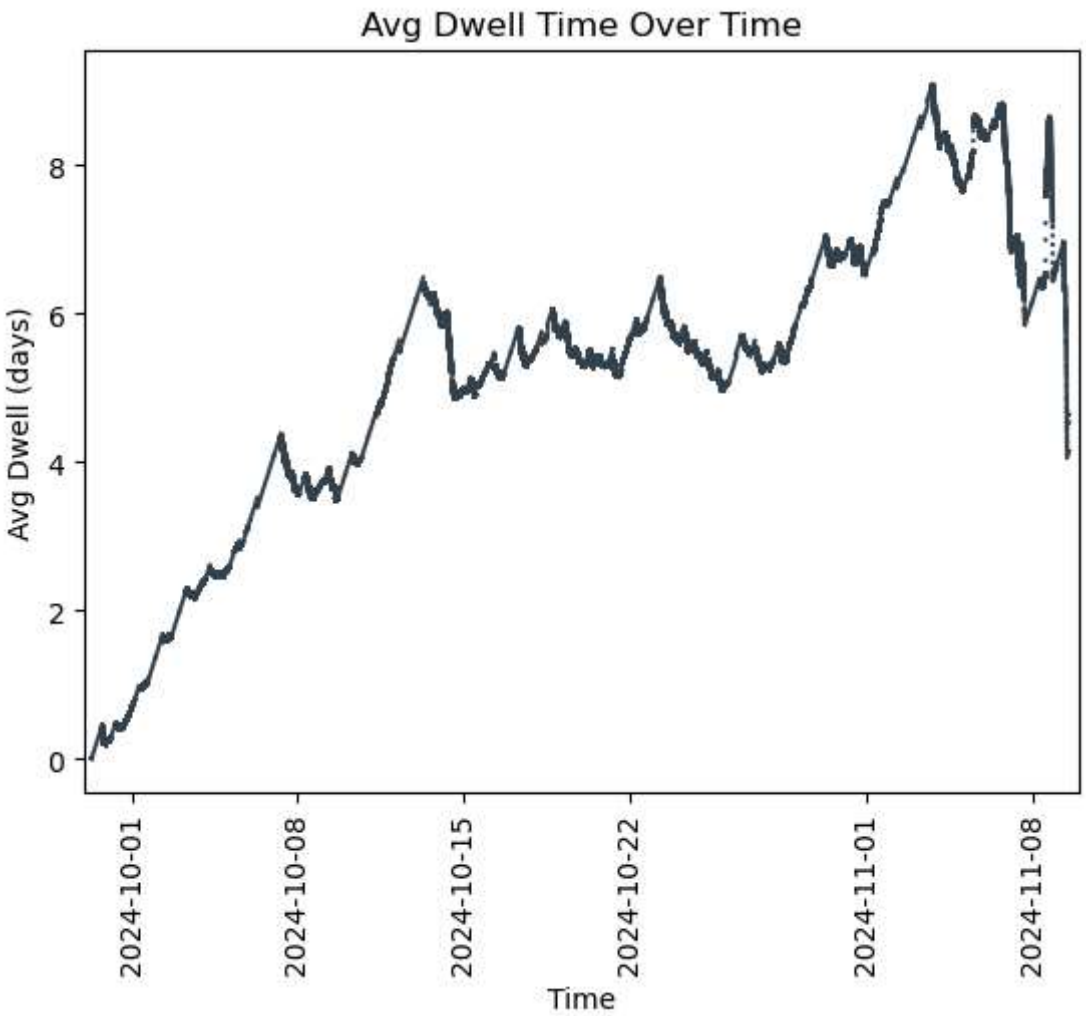
In [90]:
```
container_times = yrd120r_data
sampling_interval = 12.0
first_day = '2024-09-29'
last_day = '2024-11-09'
print_setting = True
```

```
debug_setting = False
overlay_setting = False
```

In [91]:
```
yrd120r_results = dwell_times(
    container_times, #.sample(n=100),
    sampling_interval, first_day, last_day,
    print_setting, debug_setting, overlay_setting
)
```




Container Count Over Time
Container Count
10000
8000
6000
4000
2000
0
2024-10-01
2024-10-08
2024-10-15
2024-10-22
2024-11-01
2024-11-08
Time



Some characteristics of the plots of total dwell time, container count, and average dwell time warrant discussion. All three plots begin and end with 0 value. This is due to the Mainsail methodology requirement that the containers arrive and depart during the reporting period. The report starts with no containers. As containers arrive, the container count increases. Since all containers must also depart, the container count returns to zero. Dwell time can only accrue from containers that are on terminal; so, dwell time also starts at zero and returns to zero. The average dwell time, the ratio of total dwell time and container count, also rises from zero and returns to zero but with very different rates. The rise is slower than the return. This is because the containers at the beginning of the period must necessarily be short dwelling containers, having only just arrived. In contrast, the containers at the end of the period may have dwelled for 6-weeks, causing a rapid crash to zero as they all depart before the end of the period.

In [92]:
```
yrd120r_results.pivot_table(index='Method', columns='Timeframe', values='Avg Dwell'
```

Out[92]:

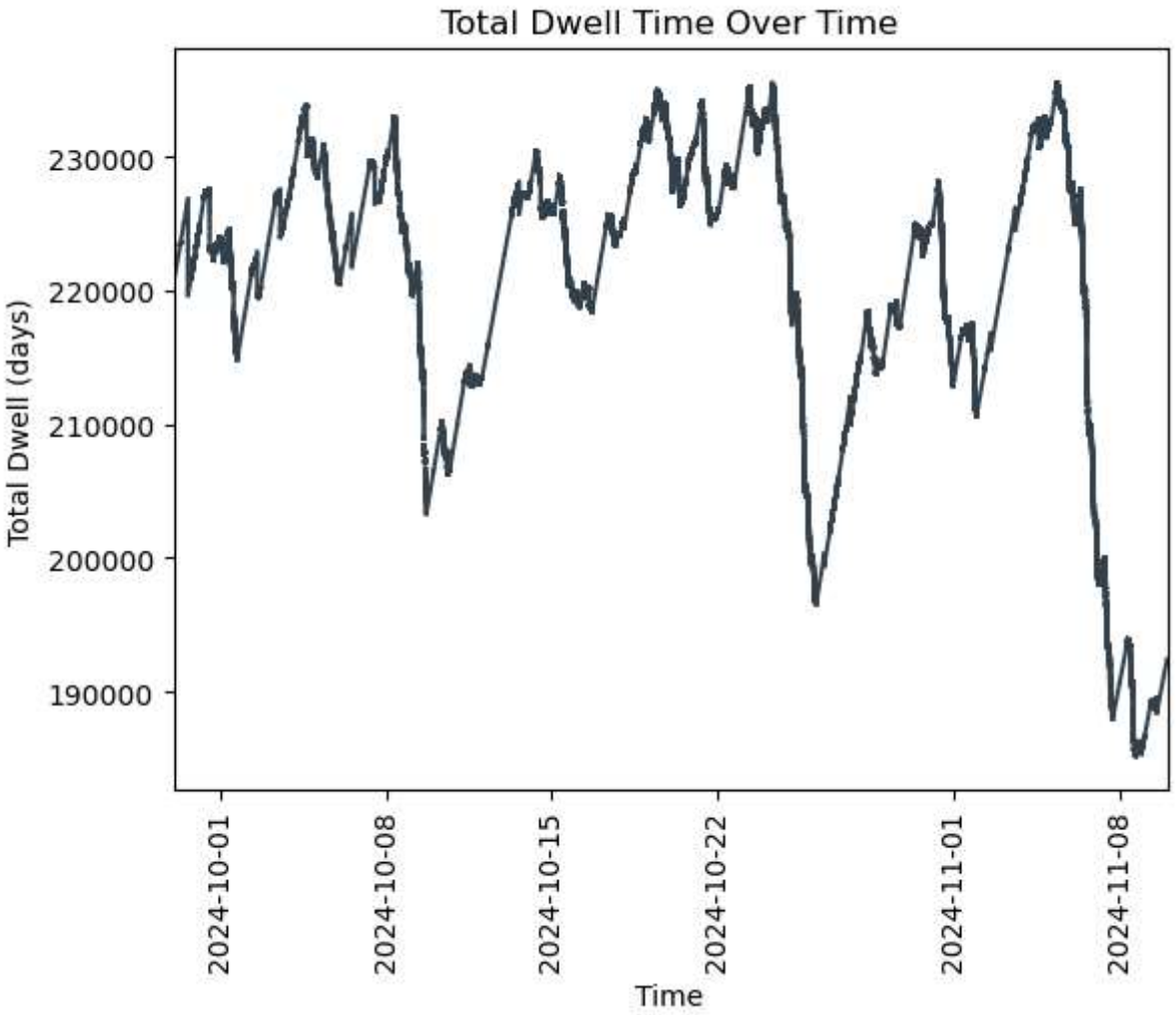
Timeframe Method	Avg Over Days	Entire Period
Integration	5.098386	5.098387
Mainsail	0.000000	5.739288
Ratio	5.162030	5.434108
Sampling	5.163540	5.163540

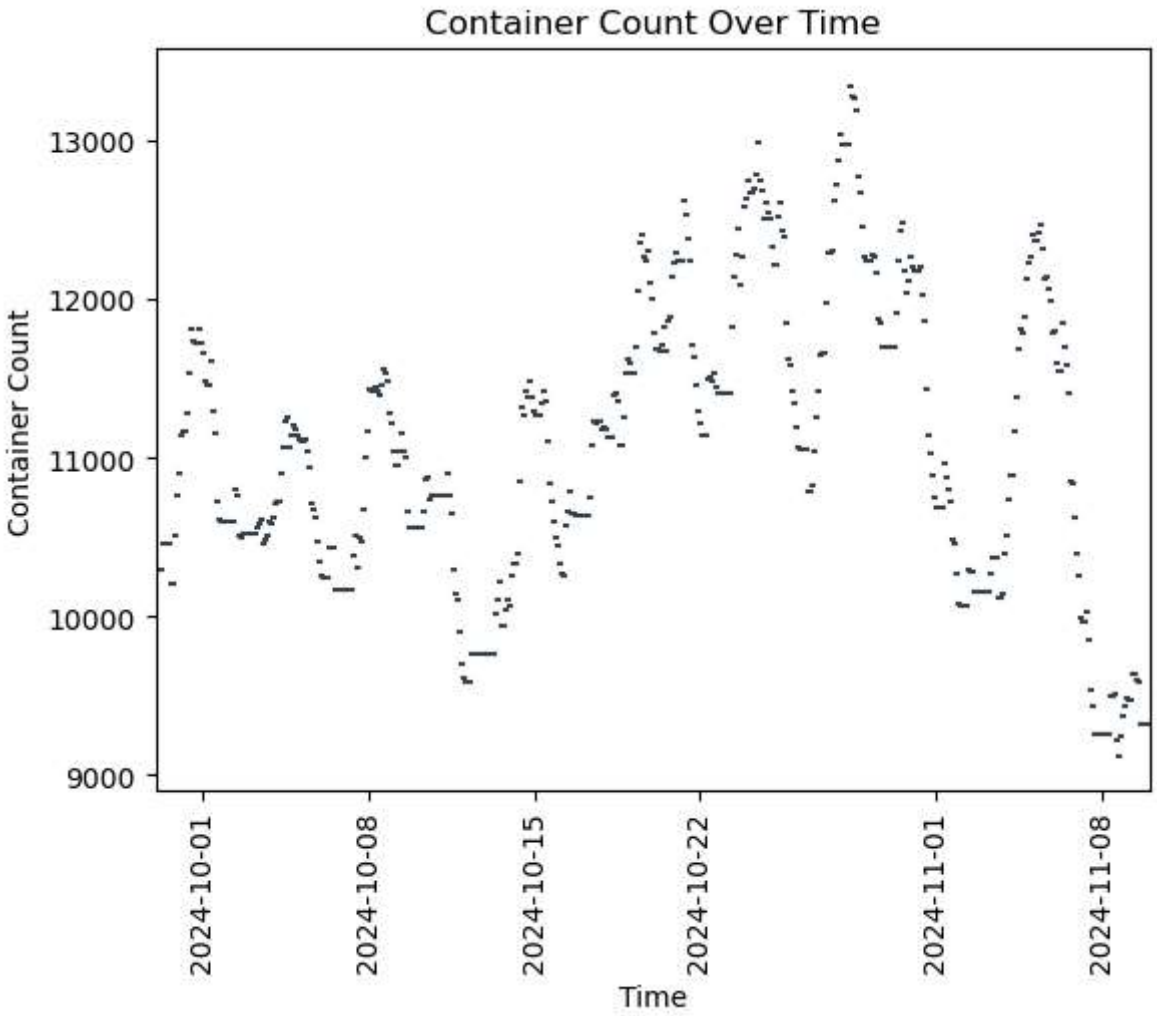
The integration and sampling method both have matching values for the average of the daily averages and for averaging over the entire period directly. They are in reasonably close agreement with each other, differing by 65 millidays or an hour and a half. The average over the daily averages for the ratio method is also very slightly closer to the exact answer provided by the integration method, differing by only 64 millidays or 1.3%. The difference between the ratio method's average over the daily averages and the average over the entire period is larger, a little over a quarter of a day. I do not have an explanation for the sign or the magnitude of the variance. The Mainsail result has the largest difference of any result. It overestimates the actual average dwell time for this selection of containers by almost two-thirds of a day or 18 hours. Given an average value of nearly 5 days, that is nearly a 13% error or about an order of magnitude larger error than the ratio or sampling methods.

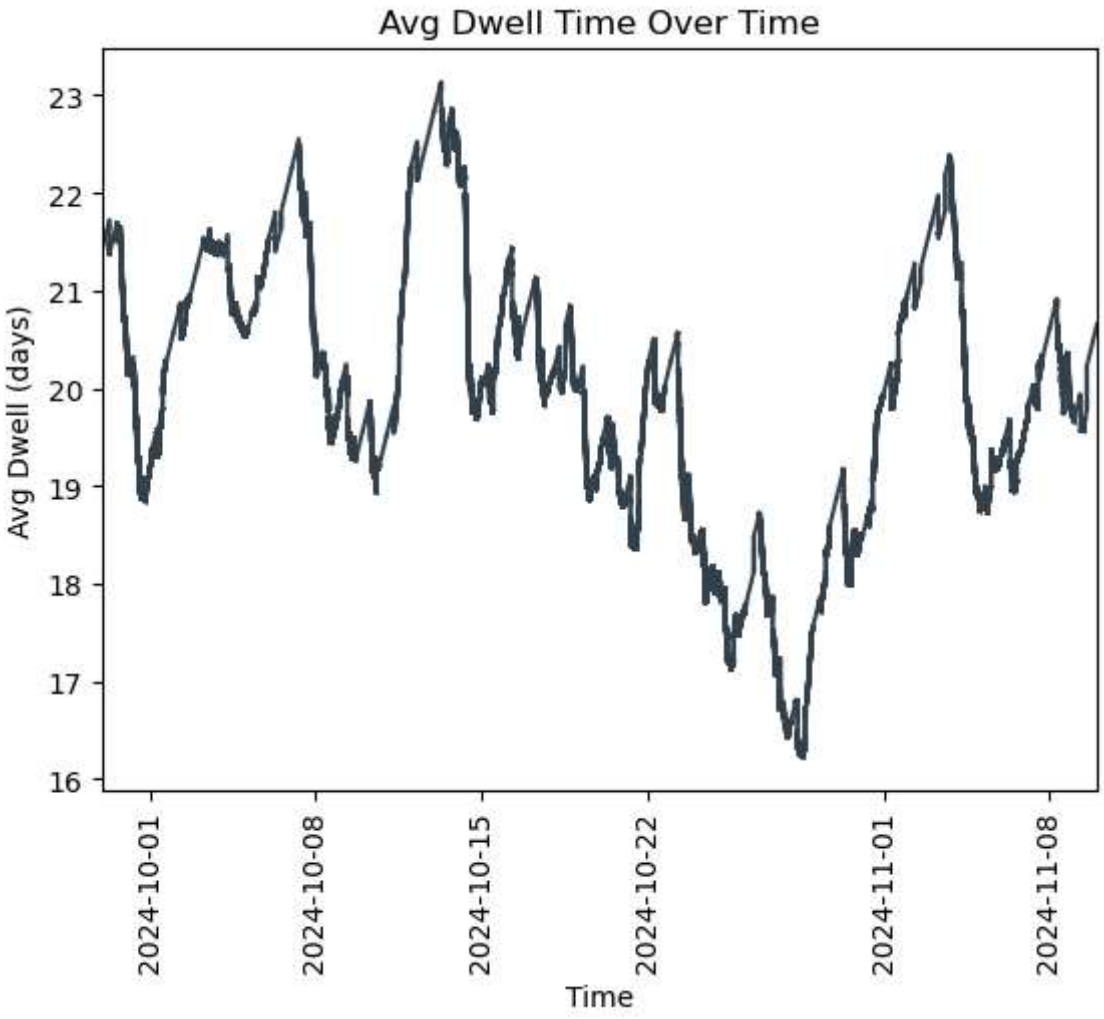
The data available in the Crow's Nest allows us to examine all the containers on T18 from September 29, 2024 through November9, 2024.

In [93]:
```
container_times = crows_nest_data
```

In [94]:
```
crows_nest_results = dwell_times(
    container_times, #.sample(n=100),
    sampling_interval, first_day, last_day,
    print_setting, debug_setting, overlay_setting
)
```


Total Dwell Time Over Time
Total Dwell (days)
230000
220000
210000
200000
190000
2024-10-01
2024-10-08
2024-10-15
2024-10-22
2024-11-01
2024-11-08
Time


Container Count Over Time
Container Count
13000
12000
11000
10000
9000
2024-10-01
2024-10-08
2024-10-15
2024-10-22
2024-11-01
2024-11-08
Time



The first notable characteristic of the plots that consider all containers is that they do not begin and end with 0. The container count starts in excess of 10,000 containers and ends in excess of 9,000 containers. The containers at the start and end are omitted from the Mainsail method's dataset. As a consequence of including these containers, the total dwell time starts in excess of 220,000 days and finishes with approximately 190,000 days of total dwell. Taking the ratio of the two shows a 6-week slice of the average dwell time. It begins from an initial value between 21 and 22 days, meanders about an average near 20 days, and finishes between 20 and 21 days.

The container count graphs from both datasets shows that the container count in the Mainsail dataset never reaches the average value. The average value for the container count appears to be around 11,000 containers. Estimating a 1-week moving average for the container count by eye for the Mainsail selection of data yields a maximum value of around 8,000 containers, approximately 30% shy of the actual count. These containers are missed because they arrived too early or departed too late to be included in the Mainsail dataset.

In [95]:
```
crows_nest_results.pivot_table(index='Method', columns='Timeframe', values='Avg Dwe
```

Out[95]:

Timeframe Method	Avg Over Days	Entire Period
Integration	19.979816	19.979822
Mainsail	0.000000	5.741526
Ratio	19.970304	19.889106
Sampling	20.124223	20.124223

The integration method has matching values for the two methods for computing the same average value, either by using the entire period directly or by averaging the average value for each day. The two values differ by 6 microdays or 0.5 seconds. Sampling agrees well with integration, differing by 144 millidays or approximate 3.5 hours for roughly 0.7% error. The ratio method outperforms the sampling method differing from integration by only 9.5 millidays or approximately 14 minutes for roughly 0.05% error. The difference for the ratio method between the average of the daily averages and the direct average for the entire period is in the opposite direction of the previous dataset and smaller in magnitude. I continue to have no explanation for the variance. The Mainsail metric is very similar to what was obtained with the first dataset because the Mainsail metric of course continues to enforce that the containers arrive and depart during the period. The Mainsail metric differs between the two datasets. With the Crow's Nest dataset a value of 5.7415 is obtained and with the YRD120R a value of 5.7393 is found. The difference is 2 millidays or 3 minutes for a 0.04% difference from the YRD120R value. There are a small number of disagreements about what containers are on the terminal and some small variations in when the containers arrive and depart between the two datasets.

The Mainsail measure of 5.7 days is distant from the apparently correct value found using the integration method of 20.0 days, corroborated by the other two methods to within $\pm$ 0.1 days. The reason for the difference is that the Mainsail measure includes on average over the course of the period only about half the containers. The average container count given by $\sum_C \frac{w_c}{T}$ is 11070.06 containers. The container count included in the Mainsail measure is given by $\sum_D \frac{w_d}{T}$ and amounts to 5876.26 containers. The formulas used here are derived and explained in the later sections of this document. On average, 5193.80 containers are excluded from the Mainsail dataset over the course of the period. The included containers have an average dwell time given by the ratio method of $\frac{\sum_D w_d \langle d_d \rangle_T}{\sum_D w_d}$, amounting to 5.43 days. The excluded containers have an average dwell time of 36.2 days. The selection of containers in the Mainsail dataset captures only around half of the containers on average and those containers are the shortest dwelling containers, resulting in a small estimate for the average dwell time, in this case, an estimate that is only approximately 30% of the correct value. The values for the quantitative analysis presented in this paragraph are developed in `.\ms_metric_quant_analysis.xlsx`.

Differences Between the Mainsail Method and the Other Methods and Their Impacts

The Mainsail metric does not work with all the containers on the terminal.

The selection of only a subset of the containers opens the Mainsail metric to bias that depends on how the selection is made. The selection is biased toward including only the shortest dwelling containers. The longest possible dwell that can be included in the dataset selected by the Mainsail metric is the length of the period. The container must arrive on the first day and depart on the last day.

The impact of the selection is a bias toward producing short average dwell times. To combat the bias, users have reported using large periods such as six weeks to include longer dwelling containers. However, using long periods such as six weeks impacts the usefulness of the Mainsail metric for a weekly report like the flash report. When using a period of six weeks, two successive weeks of the flash report will have a large number of containers in common in both weeks, reducing the sensitivity of the weekly metric to change. The reduction in sensitivity is similar to that experienced when using a moving average.

The Mainsail metric computes the full dwell time and not the average dwell time.

Sampling, integration, and the ratio method all compute the average value over the period T of the average dwell time. For the duration of the container's stay, the average dwell time is half of the full dwell time, that is, if a container arrives at noon and leaves at midnight, then its full dwell was 12 hours and its average dwell over that time was 6 hours. This means that if you selected many points from noon to midnight and computed the container's dwell at those points in time, you would get an average value of 6 hours.

The Mainsail metric has no connection to time except through the selector D. Once selected, the full dwell time is used to compute $\langle d \rangle_D$.

$$\langle d \rangle_D = \frac{\sum_D d_d}{|D|}$$

This method is analogous to how the average dwell time is computed at a single instant in time.

$$\langle d \rangle_C(t) = \frac{\sum_C d_c(t)}{|C(t)|}$$

However, to use the full dwell time is to choose a separate moment in time for every container and to bias that choice to be the longest possible dwell for the container. To be used in this manner, the above formula for $\langle d \rangle_D$ must be re-written as

$$\langle d \rangle_D = \frac{\sum_D d_d(t_d)}{|D|}$$

Instead of being computed for a single value of t, $\langle d \rangle_D$ is computed for D values of t_d which are selected to maximize $\sum_D d_d$.

The impact of this approach is to introduce a positive bias that competes with the preceding negative bias. The impact can be seen in the overestimate of the average dwell time for the entire period T of the illustrative example described later in this document. The bias toward shorter dwells from selecting only containers that arrive and depart during the period is much stronger than the bias toward reporting longer dwells from using the full dwell time instead of the average dwell time.

The Mainsail metric truncates the arrival and departure dates before computing the dwell.

The impact of truncating the arrival and departure dates is to shorten the average dwell times. For containers that arrive and depart on different days, the impact to the computed average is near zero if the arrival and departure times are uniformly distributed throughout the day. However, a container that stays on terminal for a single day is common and will not contribute a fraction of a day to the total dwell time, instead it contributes nothing because its arrival and departure are truncated to the same momement in time - midnight. Removing the dwell times of all containers that arrive and depart on the same day biases the average toward smaller values. This is the weakest of the biases in the Mainsail metric.

Conclusions

The mathematically rigorous method for computing the average of the ratio of two functions is to integrate the ratio and divide by the magnitude of the domain. For a ratio of functions of time, this means divding the two functions together, taking the area under the resulting curve, and dividing by the duration of the period. This method was implemented in this notebook as a reference.

Two approximate methods were implemented and justified through comparison to the results of the integration method, namely, the sampling method and the ratio method. Sampling is perhaps familar and intuitive and is implemented as corroboration for the integration method. The ratio method was implemented because it is the preferred method of the Crow's Nest team for aggregating the average dwell time. The quality of the approximation using the ratio method has been justified through comparison to the integration and sampling methods.

The Mainsail metric for aggregated average dwell time is shown to have differences from the other three methods, which largely agree upon the average dwell time for a dataset taken over six weeks from T18. The differences are shown to be primarily due to Mainsail's

exclusion of containers that do not arrive and depart during the period. This excludes approximately half the containers in the T18 dataset at any given moment in time during the analysis period. The excluded containers have an average dwell time that is approximately a factor of $7\times$ the average dwell of the included containers, resulting in the Mainsail metric being roughly only a third of the correct value given by the integration method and corroborated by the sampling and ratio methods.

The Crow's Nest team has decided to move forward with the ratio method and not implement the Mainsail metric. The sampling and integration methods are also not scheduled for development as the ratio method appears to be sufficiently accurate for the purposes of the data warehouse and its users.

Appendix: Illustrative Example

An example is described thoroughly to illustrate each method used to compute the average dwell time.

Sampling

To demonstrate the sampling method, consider an empty yard that receives three containers that depart on the following schedule:

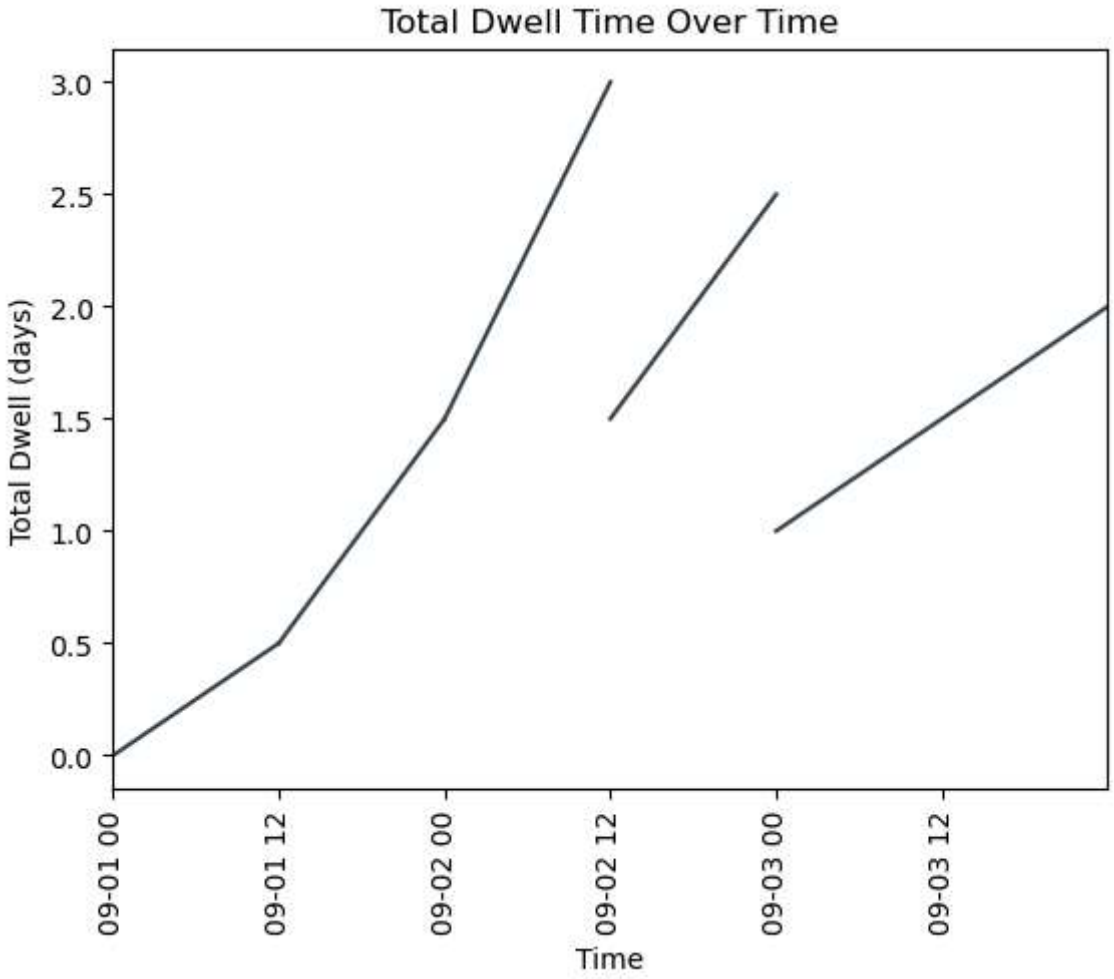
- Container A: Arrives at midnight on September 1; Departs at noon on September 2
- Container B: Arrives at noon on September 1; Departs at midnight on September 3
- Container C: Arrives at midnight on September 2; Departs the moment before midnight on September 3

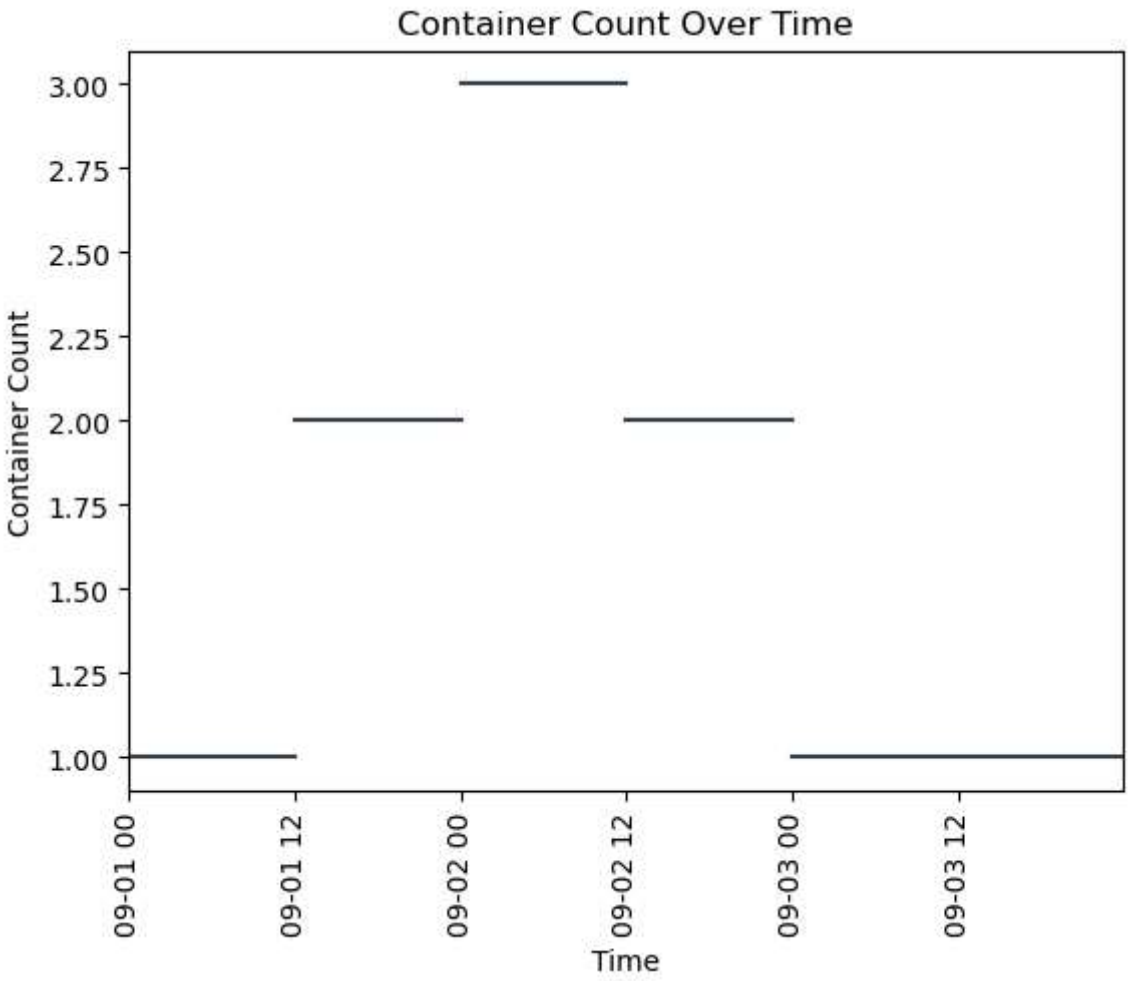
Let the gap in time between successive samples be 12 hours and let's set the period of interest to be from September 1 through September 3.

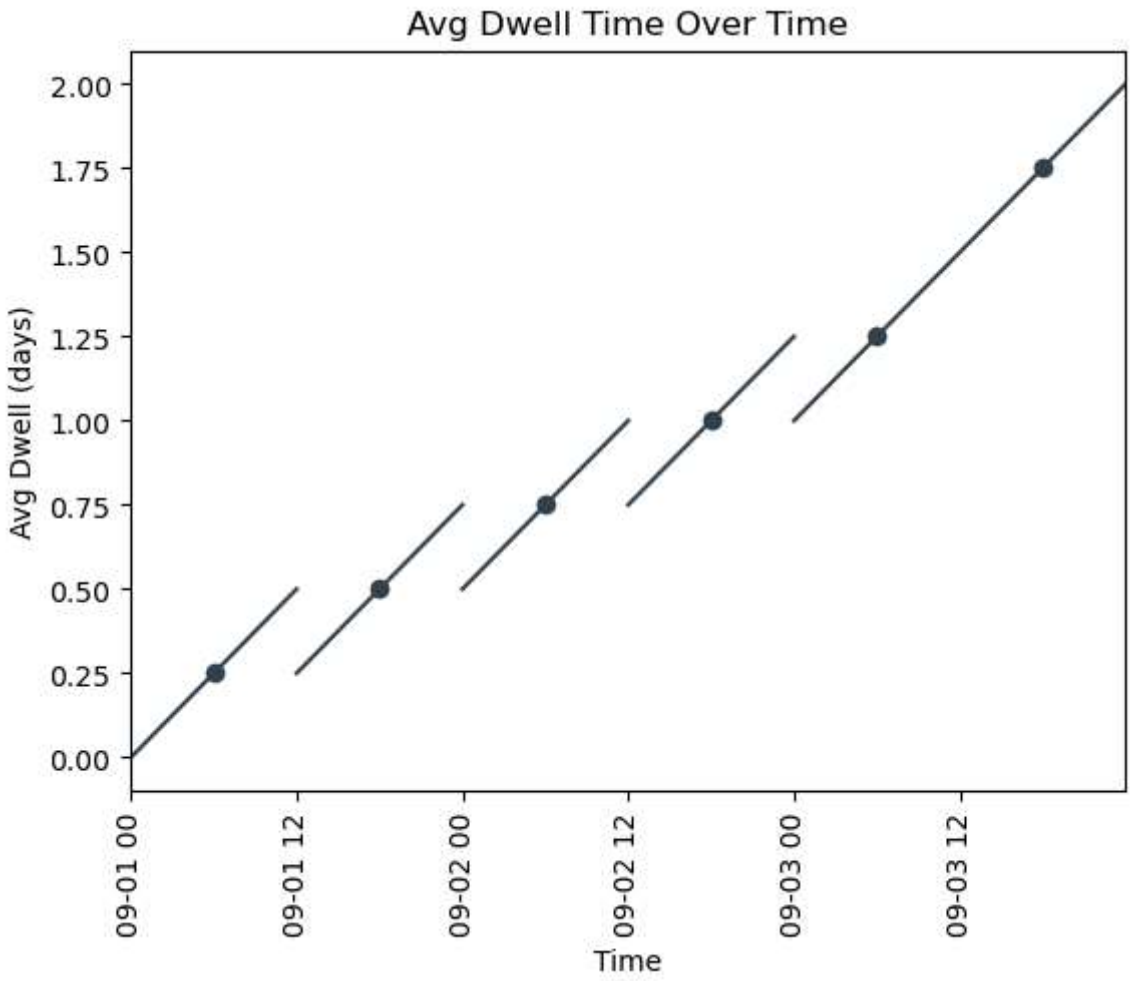
In [96]:
```
box_schedules = {
    'container': ['A', 'B', 'C'],
    'arrival': ['2024-09-01 00:00:00', '2024-09-01 12:00:00', '2024-09-02 00:00:00'
    'departure': ['2024-09-02 12:00:00', '2024-09-03 00:00:00', '2024-09-03 23:59:5
}
sample_gap = 12
first_day = '2024-09-01'
last_day = '2024-09-03'
```

The previous cell codifies the example. The next cell runs it.

In [97]:
```
results = dwell_times(box_schedules, sample_gap, first_day, last_day, True, False,
```


Total Dwell Time Over Time
Total Dwell (days)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
09-01 00
09-01 12
09-02 00
09-02 12
09-03 00
09-03 12
Time


Container Count Over Time
Container Count
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
09-01 00
09-01 12
09-02 00
09-02 12
09-03 00
09-03 12
Time



Explicit Example

The next few paragraphs work through the example explicitly so that the reader can verify the veracity of the charts and results. Skip to "How Sampling Works" to avoid this work. Skip to "Integration" if you know how sampling works.

The first chart shows how the total dwell time changes over time. From midnight to noon on the 1st, Container A is the only box on the terminal. It arrived at midnight and so the total dwell time trends upward from 0 to 12 hours (0.5 days) from midnight to noon. Container B arrives at noon. From noon on the 1st to midnight on the 2nd, there are two containers on the terminal and 24 hours of dwell are added over those 12 hours of time. At the end of the 1st, the containers have a combined total dwell of 36 hours (1.5 days). At midnight on the second, Container C arrives and over the next 12 hours 36 additional hours of dwell accrues. The total dwell increases from 36 to 72 hours (3 days).

At noon on the 2nd, Container A leaves, removing 36 hours of dwell time from the total. It drops from 72 to 36 hours (1.5 days). Two containers remain and over the next 12 hours another 24 hours of dwell accrues so that at midnight on the 2nd the total dwell time is 60 hours (2.5 days). At that time, Container B departs, removing another 36 hours and dropping the total dwell from 60 to 24 hours (1 day). Container C is the only remaining container. It

stays for 24 more hours (less one second) and so adds an additional 24 hours of dwell to make the final dwell time at the end of the 3rd 48 hours (2 days).

The second chart shows how the container count changes over time. From midnight to noon on the 1st, there's only 1 container, Container A. From noon to midnight on the 1st there are 2; from midnight on the 2nd to noon there are 3; from noon to midnight on the 2nd, there are only 2 again; then there's only 1 container, Container C, on the terminal all day on the 3rd.

The third chart shows the average dwell time. The average dwell time is the first chart divided by the second chart. This means that for every point at time t in the Total Dwell Time chart, it is divided by the point at the same time t on the Container Count chart to get the point shown at time t on the Average Dwell Time chart. For this example, the box count is either 1, 2, or 3, depending on the time. For the first half of the 1st and for all of the 3rd, the Total Dwell Time is divided by a container count of 1 to get the Average Dwell Time; that is, the Average Dwell Time equals the Total Dwell Time. For the second half of the 1st and the last half of the 2nd, the Total Dwell Time is divided by a container count of 2, and for the first half of the 2nd, the Total Dwell Time is divided by a container count of 3. This division results in the chart shown for the Average Dwell Time.

How Sampling Works

Sampling works by taking N measurements and then averaging the results of those measurement to obtain an average result for the period during which the samples were taken.

$$\langle d \rangle_T = \frac{\sum_N d_n}{N}$$

If the samples are distributed regularly in time, then it is a good approximation to the exact average. The accuracy increases with greater numbers of samples, but so does the work involved in obtaining the result.

The sampling interval for the example is set to 12 hours. The function centers the sampling time in its interval so that the samples occur at 6 AM and 6 PM each day. The expectations for the results are shown in Table 1.

Table 1. Sample times and sampled total dwell, container count, and average dwell time values. |Sample Time|Total Dwell (days)| Container Count| Sampled Average Dwell Time (days)| |--|--|--|--| |2024-09-01 0600| 0.25| 1| 0.25| |2024-09-01 1800| 1.00| 2| 0.50| |2024-09-02 0600| 2.25| 3| 0.75| |2024-09-02 1800| 2.00| 2| 1.00| |2024-09-03 0600| 1.25| 1| 1.25| |2024-09-03 1800| 1.75| 1| 1.75|

The sampled points have been marked with dots on the Average Dwell Time chart.

Here are the sampling results for the example:

In [98]:
```
sampling_results = results[results['Method']=='Sampling'].copy()
sampling_results['Avg Dwell'] = round(sampling_results['Avg Dwell'],4)
sampling_results
```

Out[98]:

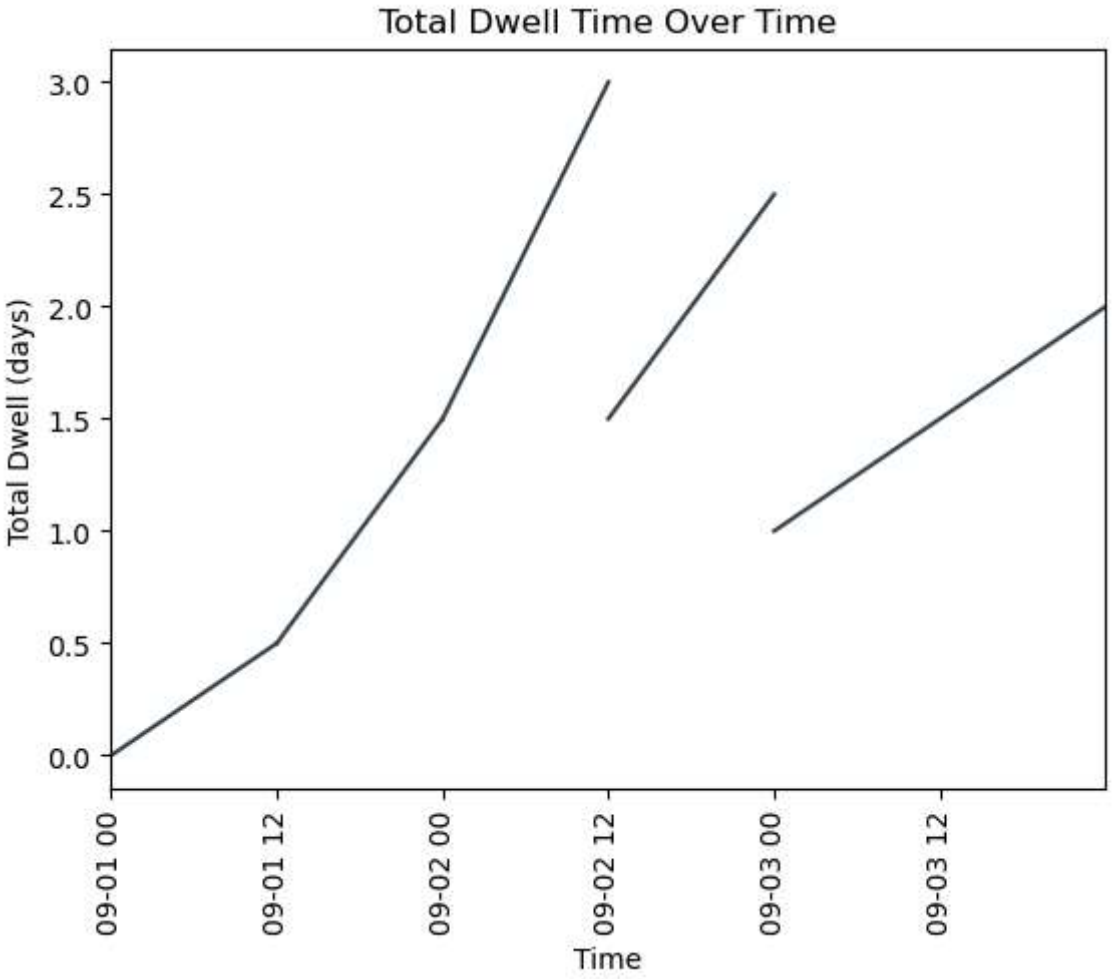
	Method	Timeframe	Avg Dwell
5	Sampling	Entire Period	0.9167
6	Sampling	2024-09-01	0.3750
7	Sampling	2024-09-02	0.8750
8	Sampling	2024-09-03	1.5000
9	Sampling	Avg Over Days	0.9167

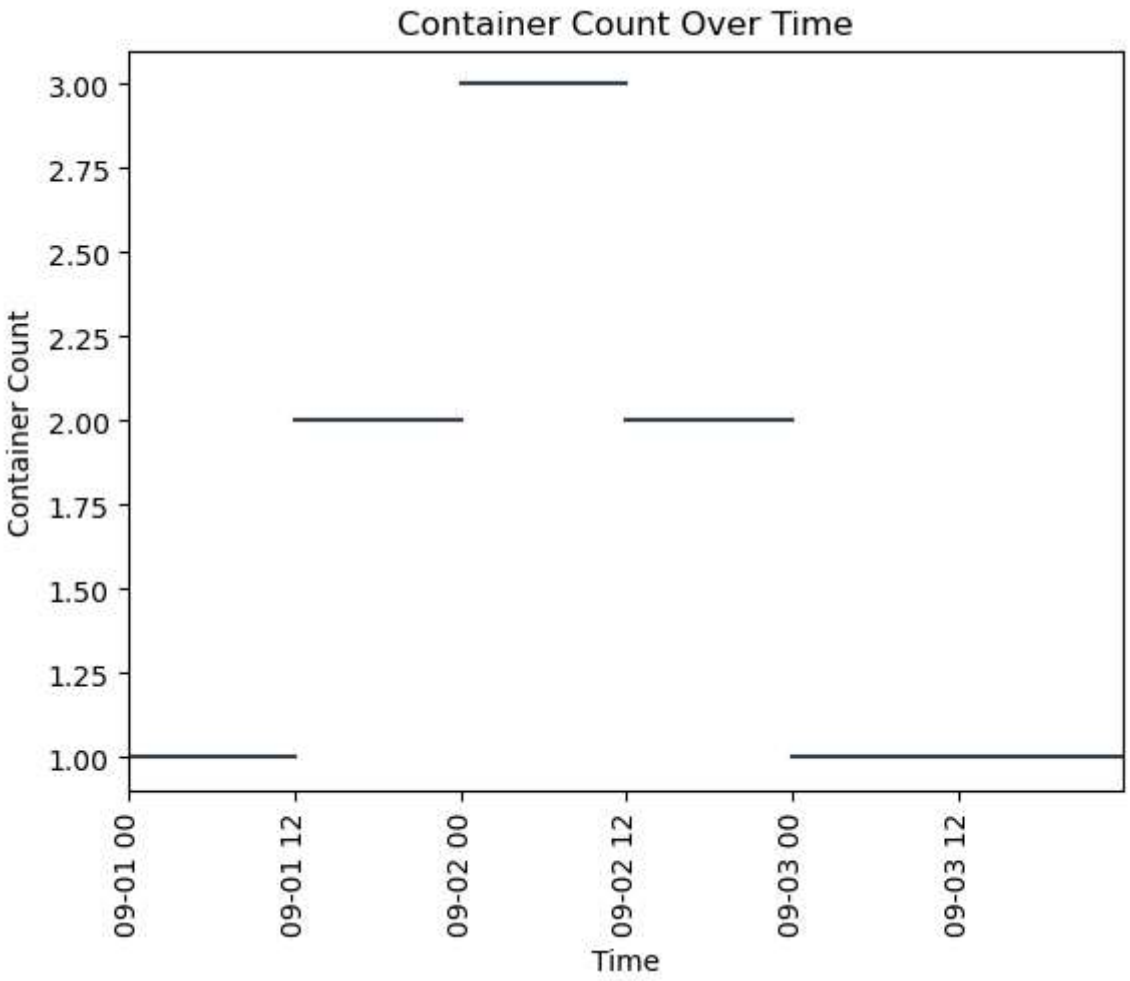
The results present averages instead of results for individual samples. The rows with a date for the Timeframe are the average for that day. They agree with the average of the expected values for each day. The row with the Timeframe "Entire Period" averages all the samples together. The row with the Timeframe "Avg Over Days" averages the averages obtained from each day, 0.375, 0.875, and 1.5, in this case.

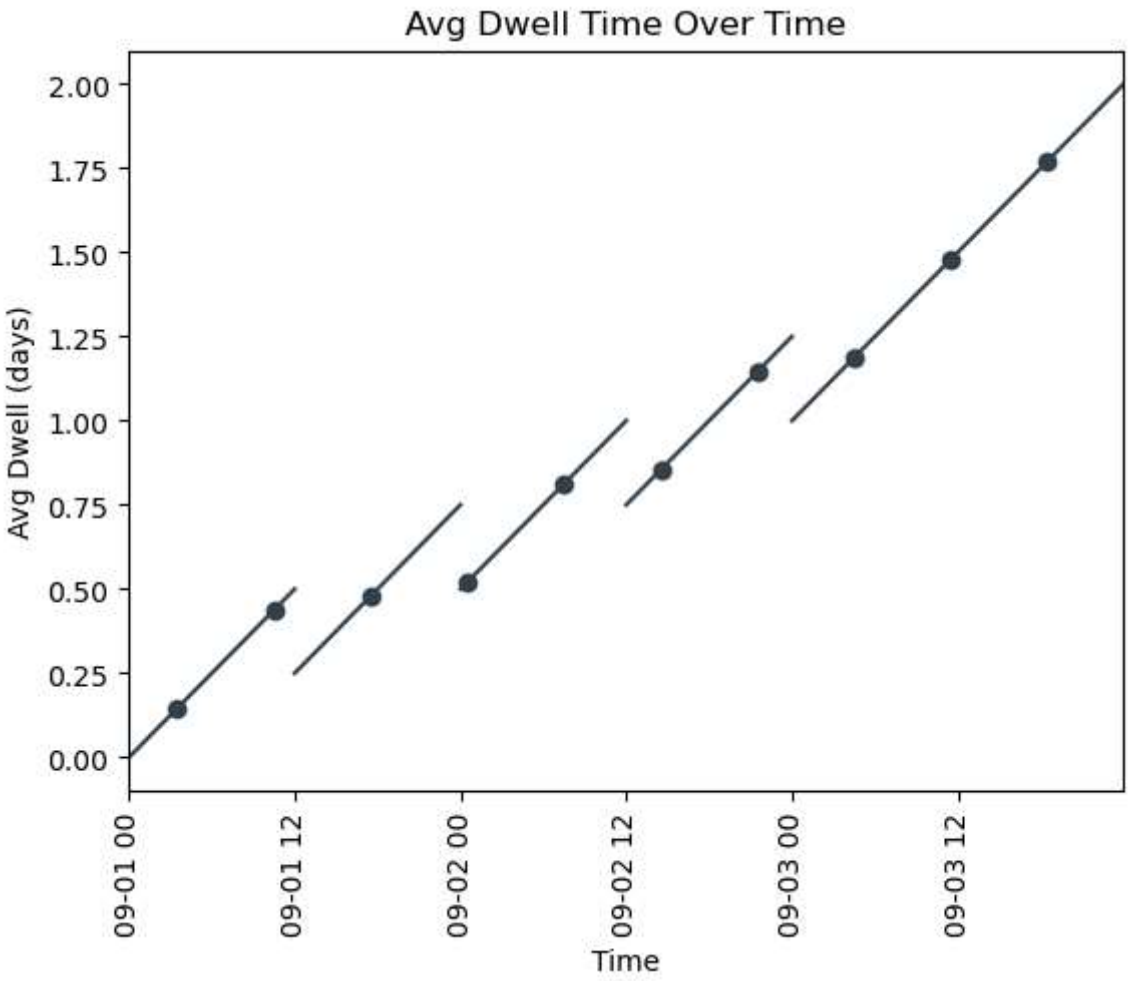
The two rows "Entire Period" and "Avg Over Days" are presented separately because not every aggregation method is an exact method and the value obtained from averaging the entire period directly may not be identical to the average of the daily averages. Because the samples taken are distributed in time, the average for the entire period and that obtained by averaging the average values of the days are identical in this case, but this is not always true. If some days have more samples than others, then the two aggregations will differ. The example below sets the `sample_gap` from 12 hours to 7 hours so that the days have differing numbers of samples.

In [99]:
```
box_schedules = {
    'container': ['A', 'B', 'C'],
    'arrival': ['2024-09-01 00:00:00', '2024-09-01 12:00:00', '2024-09-02 00:00:00'
    'departure': ['2024-09-02 12:00:00', '2024-09-03 00:00:00', '2024-09-03 23:59:5
}
sample_gap = 7
first_day = '2024-09-01'
last_day = '2024-09-03'
```

In [100…
```
results = dwell_times(box_schedules, sample_gap, first_day, last_day, True, False,
```


Total Dwell Time Over Time
Total Dwell (days)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
09-01 00
09-01 12
09-02 00
09-02 12
09-03 00
09-03 12
Time


Container Count Over Time
Container Count
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
09-01 00
09-01 12
09-02 00
09-02 12
09-03 00
09-03 12
Time



In [101…
```
uneven_sampling_results = results[results['Method']=='Sampling'].copy()
uneven_sampling_results['Avg Dwell'] = round(uneven_sampling_results['Avg Dwell'],4
uneven_sampling_results
```

Out[101]:

	Method	Timeframe	Avg Dwell
5	Sampling	Entire Period	0.8833
6	Sampling	2024-09-01	0.3542
7	Sampling	2024-09-02	0.8333
8	Sampling	2024-09-03	1.4792
9	Sampling	Avg Over Days	0.8889

The first and last days have three samples, but the middle day has four. With different numbers of samples over the days, the average of all the samples is not equal to the average of the average daily values.

Discussion of the Sampling Method

In conclusion, the sampling method is an approximate method. However, from the production data example, sampling twice a day appears to be often enough to capture an

accurate approximation to the true aggregate average dwell time. The approximation can be improved by taking more samples per day, but this also increase the amount of work and storage that is required to produce the approximation.

Integration

To average a function of time like the average dwell time, the integral of the function is divided by the period.

$$\langle d \rangle_{C,T} = \frac{\int_0^T \langle d \rangle_C}{T}$$

This means that the area under the curve of the average dwell time is computed and the area is divided by the duration of the period T.

The area under the average dwell time curve can be computed using geometry. For each piecewise continuous interval, that is, for each uninterrupted line segment, the area under the curve is the area of a rectangle and a trigangle. The box has a height of the initial average dwell time for the interval and a width that is the duration of the interval. Its area is the product of the width and height. The triangle is an isosceles right triangle. It is a right triangle with both legs the same length. The area of the triangle is $\frac{1}{2}bh$ with b and h both equal to the length of a leg. The length of a leg is the duration of the interval t_i. The area of the triangle is $\frac{1}{2}t_i^2$.

In [103…

```
from IPython.display import Image, display

# Display an image from a file
display(Image(filename='./builtin/area diagram v2.png', width=400))
```

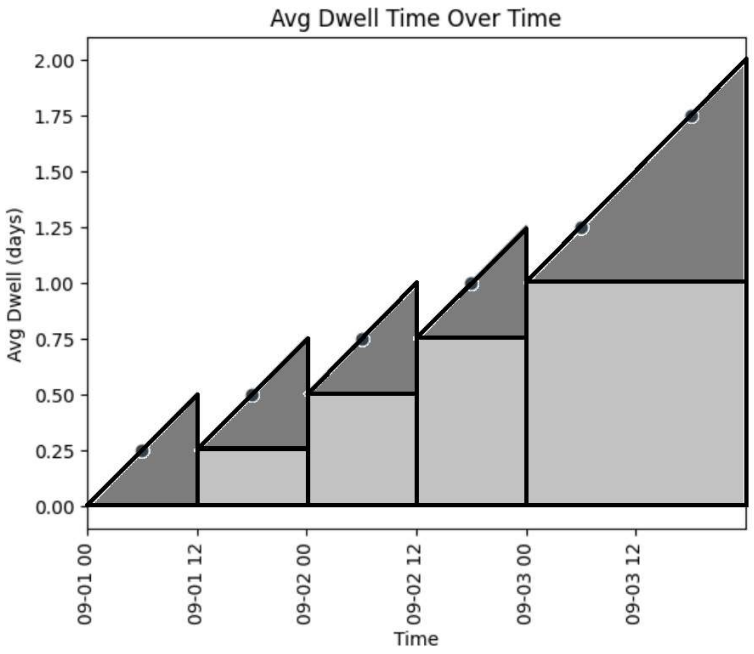


Figure 1. Area under the average dwell time curve.

The area under the average dwell time curve is shown in Table 1.

Table1. Areas under the Average Dwell Time curve. |Interval Start|Interval End|Initial Avg Dwell|Duration (h)|Box Area (h^2)|Triangle Area (h^2)|Total Area (h^2)| |--|--|--|--|--|--|--| |09-01 0000|09-01 1200|0|12|0|72|72| |09-01 1200|09-02 0000|6|12|72|72|144| |09-02 0000|09-02 1200|12|12|144|72|216|2| |09-02 1200|09-03 0000|18|12|216|72|288|2| |09-03 0000|09-03 2359|24|24|576|288|864|3|

In Table 1, metrics for the intervals defined by the uninterrupted line segments in *Figure 1* are shown. The initial avg dwell is the value of the average dwell time curve at the interval start. The duration is the width of the interval. The box area is the product of the initial avg dwell and the duration and the triangle area is one-half of the square of the duration. The total area for the interval is the sum of the box area and triangle area.

The intervals for each day can be summed and divided by the the duration of a day, 24 hours, to get the aggregated average dwell time for each day or for all three days.

Table 2. Aggregated average dwell times. |Day|Total Area (h^2)|Duration|Aggregated Avg Dwell Time (h)| Agg Avg Dwell Time (days) |--|--|--|--|--| |1|216|24|9|0.375| |2|504|24|21|0.875| |3|864|24|36|1.5 |1-3|1584|72|22|0.9167|

The integration method is an exact method of computing the aggregated average dwell time. This means that the average of the daily values must be equal to the aggregated value computed for the entire period. This is the case here; the average of 9, 21, and 36 is 22.

The results that we have computed here agree with those computed for the example. Note that Day 3 is one second short of 24 hours. However, we are rounding to the nearest 10,000 th of a day which is around 8.5 seconds. The loss of one second from Day 3 is smaller than the rounding error.

In [104…
```
integration_results = results[results['Method']=='Integration'].copy()
integration_results['Avg Dwell'] = round(integration_results['Avg Dwell'],4)
integration_results
```

Out[104]:

	Method	Timeframe	Avg Dwell
0	Integration	Entire Period	0.9167
1	Integration	2024-09-01	0.3750
2	Integration	2024-09-02	0.8750
3	Integration	2024-09-03	1.5000
4	Integration	Avg Over Days	0.9167

Discussion of the Integration Method

Implementing the integration method is significantly more complex in terms of development and maintenance of the code than any other method included in the brief. This means that this method, in comparison to the other methods, has a greater chance for errors and defects, will be more costly to build, will be more costly and risky to change, and will reach end-of-life sooner.

This method is significantly less flexible than any other method included in the brief. Any filtering on the set of containers analyzed must be done during development and not at runtime. This means that the Crow's Nest users would need to request development to provide the aggregate average dwell time for import containers, transship contianers, full containers, empty containers, and any other subgroup of containers that may be of interest. The delivery of new filters would likely be weeks to months and be dependent on the team's other priorities.

An excellent approximation to the exact answer that does not require the ongoing and ever-growing storage of the sampling method or the complexity and inflexibility of the integration method is wanted.

Ratio Method

The complexity of the integration method is removed by using the ratio method which is an approximation. Instead of carefully dividing the curve for the total dwell time by the curve for the box count point-by-point through time, the average of the total dwell time is divided by the average of the box count for the period of interest. The ratio of the average of two functions is a well-known and in our case excellent approximation for the average of the ratio of two functions like the aggregated average dwell time, $\langle d \rangle_{C,T}$.

$$\frac{\left\langle \sum_C d_c \right\rangle_T}{\langle C \rangle_T} \approx \left\langle \frac{\sum_C d_c(t)}{C(t)} \right\rangle_T = \langle d \rangle_{C,T}$$

The average of the total dwell time is obtained exactly by taking the weighted average of the average dwell time of each container during the period. The weights are the portion of time that the container was on the terminal during the period. When the container is not on the period, the dwell is 0.

$$\left\langle \sum_C d_c \right\rangle_T = \sum_C \frac{w_c}{T} \langle d_c \rangle_T$$

For our example, Table 3 shows metrics associated with the computation of the average total dwell time.

Table 3. Computing the average total dwell time. |Container|Day|Initial Dwell|Final Dwell|$\langle d_c \rangle_T$ (h)|w_c (h)|T (h)|$\frac{w_c}{T}\langle d_c \rangle_T$ (h)| |--|--|--|--|--|--|--|--| |A|1|0|24|12|24|24|12| |B|1|0|12|6|12|24|3| |A|2|24|36|30|12|24|15| |B|2|12|36|24|24|24|24| |C|2|0|24|12|24|24|12| |C|3|24|48|36|24|24|36|

In Table 3, metrics concerning the total dwell time for each container on each day are computed. Container A arrived on Day 1 and was on terminal the entire day. Its initial dwell upon arrival is 0 hours and the final dwell is 24 hours. This makes the average dwell for Container A over the course of the day 12 hours. The weight is the portion of the period that the container was on terminal. It was on terminal for 24 hours and the period for a day is 24 hours. The weight is therefore 1 and the weighted average dwell equals the average dwell, namely, 12 hours. For Container B and Day 1, it arrived at noon and stayed through the rest of the day for a total of 12 hours with an average dwell over that time of 6 hours. It was on terminal for only half the day; it's weight is 0.5 making its weighted average dwell 3 hours. On Day 2, Container A stars with an initial dwell of 24 hours and remains on terminal for another 12 hours, departing with a final dwell of 36 hours. The average dwell for Container A over that time is 30 hours. It was on terminal for only half a day making its weighted average dwell 15 hours. The values for the reamining rows of Table 3 are computed similarly.

Values for the average total dwell time for each day and for the entire period are shown in Table 4. The average total dwell is the sum of the weighted average dwells for each day.

Table 4. Aggregated average total dwell times. |Day|$\left\langle \sum_C d_c \right\rangle_T$ (h)| |--|--| |Entire Period|102| |1|15| |2|51| |3|36|

The average contianer count is computed by summing the amount of time for each day that each container was on the terminal.

$$\langle C \rangle_T = \sum_C \frac{w_c}{T}$$

In Table 5, metrics concerning the container count for each container on each day are computed. The values for w_c and T are reproduced from Table 3.

Table 5. Metrics associated with the average container count. |Container|Day|w_c|T|$\langle C \rangle_T$| |--|--|--|--|--| |A|1|24|24|1| |B|1|12|24|0.5| |A|2|12|24|0.5| |B|2|24|24|1| |C|2|24|24|1| |C|3|24|24|1|

In Table 6, the average container count $\langle C \rangle_T$ for each day is listed. The average container count is the sum of the contributions from each container during the day.

Table 6. Daily average container counts. |Day|$\langle C \rangle_T$| |--|--| |Entire Period|5| |1|1.5| |2|2.5| |3|1|

The ratio method divides the values in Table 4 by those in Table 6 to arrive at average values for the day and for the entire period. The average of the average values of the days is presented in the row labeled "Avg Over Days".

Table 6. Results for the ratio method for the example. |Period|$\left\langle \sum_C d_c \right\rangle_T$ (h)|$\langle C \rangle_T$ (h)|$\frac{\left\langle \sum_C d_c \right\rangle_T}{\langle C \rangle_T}$ (h)|$\frac{\left\langle \sum_C d_c \right\rangle_T}{\langle C \rangle_T}$ (days)| |--|--|--|--|--| |Entire Period|102|5|20.4|0.85| |Sep 1|15|1.5|10|0.4167| |Sep 2|51|2.5|20.4|0.85| |Sep 3|36|1|36|1.5| |Avg Over Days|--|--|22.13|0.9222|

Note that for this approximate method that the value for the average of the daily averages does not equal the average computed from examining the entire period. This occurs because this is an approximate method. Every method in the brief other than the integration method is an approximation.

In [105…
```
ratio_results = results[results['Method']=='Ratio'].copy()
ratio_results['Avg Dwell'] = round(ratio_results['Avg Dwell'],4)
ratio_results
```

Out[105]:

	Method	Timeframe	Avg Dwell
10	Ratio	Entire Period	0.8500
11	Ratio	2024-09-01	0.4167
12	Ratio	2024-09-02	0.8500
13	Ratio	2024-09-03	1.5000
14	Ratio	Avg Over Days	0.9222

Discussion of the Ratio Method

The ratio method avoids the complexity that is necessary when using the integration method. We do not need to compute the areas of intervals between successive inventory change events. The ratio method also avoids the storage issues associated with capturing snapshots of the yard for each sample. We do not need to make copies of data that we already store just to capture various moments in time. However, the ratio method is an approximation. Comparison of its results with that of the sampling and integration methods is warranted.

In [106…
```
combined_results = pd.merge(ratio_results, integration_results, on='Timeframe')
combined_results = pd.merge(combined_results, sampling_results, on='Timeframe')
combined_results.columns = ['Ratio Method','Timeframe','Ratio Avg Dwell','Integrati
combined_results
```

Out[106]:

	Ratio Method	Timeframe	Ratio Avg Dwell	Integration Method	Integration Avg Dwell	Sampling Method	Sampling Avg Dwell
0	Ratio	Entire Period	0.8500	Integration	0.9167	Sampling	0.9167
1	Ratio	2024-09-01	0.4167	Integration	0.3750	Sampling	0.3750
2	Ratio	2024-09-02	0.8500	Integration	0.8750	Sampling	0.8750
3	Ratio	2024-09-03	1.5000	Integration	1.5000	Sampling	1.5000
4	Ratio	Avg Over Days	0.9222	Integration	0.9167	Sampling	0.9167

While the sampling and integration methods agree very well for the illustrative example, some disagreement in the results between the ratio method and the integration and sampling methods is present. The averge dwell for the first day is high by 4 hundreths of a day which is around 1 hour. The average value for day 2 is around half an hour low while day 3 luckily agrees exactly. Because the ratio method is an approximate method, the value obtained from computing the entire period directly need not equal the value obtained from computing each day and averaging the daily values. These two approaches differ by 7 hundreths of a day or just under 2 hours.

Errors of 0 to 2 hours on 5 values that average 0.9 days or 22 hours are observed for a percentage error of 0% to 11%. As has been shown, the errors from a six-week dataset taken from production are substantially smaller than what is found in this illustrative example.

Mainsail Metric

The Mainsail metric that is part of the YRD120R report produces an aggregate average dwell time by selecting the containers that have arrived or departed during the reporting period. For the selected containers only, the Mainsail metric computes the average dwell time.

$$\langle d \rangle_D = \frac{\sum_D d_d}{|D|}$$

The set of containers D is a subset of the set of containers C. The set D comprises the containers that arrived and departed during the reporting period. The set C contains all the containers that were on the terminal at all during the reporting period.

Table 7 shows metrics associated with computing the Mainsail metric. The Mainsail metric truncates the arrival and departure times. This means that the dwell times are computed as if the containers arrived or departed at midnight on the date. Note that in this example Container B, which was on terminal for 36 hours from noon on Sep 1 to midnight on Sep 3, is computed as having 48 hours of dwell while Container C, which was on terminal from 48

hours from midnight Sep 2 to 2359 hours on Sep 3, is computed as having 24 hours of dwell. See Table 7.

*Table 7. Metrics associated with computing the Mainsail dwell time. Note that no container arrives and departs on the same day and no container has 'Yes' for a value of $D{DayX}. However, all containers arrive and depart during the period*T.| Container|Arrival Date|Departure Date|In D_{Day1}|In D_{Day2}|In D_{Day3}|In D_T$|Truncated Arrival|Truncated Departure|Total Dwell (h)| |--|--|--|--|--|--|--|--|--|--| |A|Sep. 1|Sep. 2|No|No|No|Yes|2024-09-01 0000|2024-09-02 0000|24| |B|Sep. 1|Sep. 2|No|No|No|Yes|2024-09-01 0000|2024-09-03 0000|48| |C|Sep. 2|Sep. 3|No|No|No|Yes|2024-09-02 0000|2024-09-03 0000|24|

Since no container arrives and departs during the same day, a value of 0 has been defined for the daily average values. The average over the entire period is computed to be 32 hours. See Table 8.

Table 8. Aggregated average dwell times using the Mainsail metric. |Period|$\langle d \rangle_D$ (h)|$\langle d \rangle_D$ (h)| |--|--|--| |Entire Period|32|1.3333| |Sep 1|0|0| |Sep 2|0|0| |Sep 3|0|0| |Avg Over Days|0|0|

In [107…
```
mainsail_results = results[results['Method']=='Mainsail'].copy()
mainsail_results['Avg Dwell'] = round(mainsail_results['Avg Dwell'],4)
mainsail_results
```

Out[107]:

	Method	Timeframe	Avg Dwell
15	Mainsail	Entire Period	1.3333
16	Mainsail	2024-09-01	0.0000
17	Mainsail	2024-09-02	0.0000
18	Mainsail	2024-09-03	0.0000
19	Mainsail	Avg Over Days	0.0000

Discussion of the Mainsail Metric

Comparison of the Mainsail metric to the other metrics is warranted.

In [108…
```
combined_results = pd.merge(combined_results, mainsail_results, on='Timeframe')
combined_results = combined_results.rename(columns={'Method': 'Mainsail Method', 'A
combined_results
```

Out[108]:

	Ratio Method	Timeframe	Ratio Avg Dwell	Integration Method	Integration Avg Dwell	Sampling Method	Sampling Avg Dwell	Mainsail Method	M
0	Ratio	Entire Period	0.8500	Integration	0.9167	Sampling	0.9167	Mainsail	
1	Ratio	2024-09-01	0.4167	Integration	0.3750	Sampling	0.3750	Mainsail	
2	Ratio	2024-09-02	0.8500	Integration	0.8750	Sampling	0.8750	Mainsail	
3	Ratio	2024-09-03	1.5000	Integration	1.5000	Sampling	1.5000	Mainsail	
4	Ratio	Avg Over Days	0.9222	Integration	0.9167	Sampling	0.9167	Mainsail	

The daily values and their average are not valid. The period chosen is too small for any containers to be included in the set D. When computing the entire period directly, the selection difference is avoided. However, the Mainsail metric computes the average dwell using the maximum dwell for each container as if it were computing the average dwell for a single moment in time. This results in an overestimate of the aggregated average dwell time of 0.9167 days obtained via integration and sampling. The difference due to using the full dwell instead of the average dwell is partially cancelled through Mainsail's use of truncated days instead of timestamps. This final difference drops the average dwell over the entire period from 40 hours ((36+36+48)/3 = 40) to 32 hours or from 1.6667 days to 1.3333. The combination of differences results in a difference of 0.4166 days or 10 hours or a 45% difference from the integration result.

Appendix: Testing

In [109…

```
def fetch(df, method, timeframe, timeframe_is_date):
    if timeframe_is_date:
        return_value = (
            df.loc[
                (df['Method'] == method)
                & (df['Timeframe'] == pd.to_datetime(timeframe).date())
                , 'Avg Dwell'
            ].values[0]
        )
    else:
        return_value = (
            df.loc[
                (df['Method'] == method)
                & (df['Timeframe'] == timeframe)
                , 'Avg Dwell'
            ].values[0]
```

```
        )
    return(return_value)
```

In [110…

```
def compose_test_result(method, timeframe, correct_value, actual_value):
    test_passes = (correct_value == actual_value)
    if test_passes:
        pass_fail_string = ' Passes'
    else:
        pass_fail_string = ' Fails'
    test_result_string = (
        pass_fail_string + ': '
        + 'Correct: ' + str(correct_value)
        + '\tObtained: ' + str(actual_value)
        + '\t' + method + ' ' + timeframe
    )
    return(test_passes, test_result_string)
```

In [112…

```
def test(df, method, timeframe, is_date, correct_value):
    test_value = fetch(df, method, timeframe, is_date)
    (test_bool, test_string) = compose_test_result(method, timeframe, correct_value
    return(test_bool, test_string)
```

In [113…

```
def test_general(test_name, box_schedules, sample_gap, first_day, last_day, tests,
    if debug_setting:
        print('\n' + test_name + ' begin\n')
        print('sample_gap: ' + str(sample_gap))
        print('first_day: ' + first_day)
        print('last_day: ' + last_day)
    results = dwell_times(box_schedules, sample_gap, first_day, last_day, print_set
    results['Avg Dwell'] = round(results['Avg Dwell'],1)
    if debug_setting:
        print('results:')
        print(results)
        print('\n')
    num_tests = len(tests)
    test_results = [None] * num_tests
    test_strings = [None] * num_tests
    for i in range(num_tests):
        (test_results[i], test_strings[i]) = test(results, tests[i][0], tests[i][1]
    test_result = all(test_results)
    if debug_setting:
        for test_string in test_strings:
            print(test_string)
        print(str(test_result) + ': ' + test_name)
    return(test_result)
```

In [114…

```
def test_primary(print_setting, debug_setting, overlay_setting):
    box_schedules = {
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 00:00:00', '2024-09-01 12:00:00', '2024-09-02 00:00
        'departure': ['2024-09-02 12:00:00', '2024-09-03 00:00:00', '2024-09-03 23:
    }
    sample_gap = 12
    first_day = '2024-09-01'
```

```
    last_day = '2024-09-03'
    tests = [
        ['Integration', 'Entire Period', False, 0.9],
        ['Integration', '2024-09-01', True, 0.4],
        ['Integration', '2024-09-02', True, 0.9],
        ['Integration', '2024-09-03', True, 1.5],
        ['Integration', 'Avg Over Days', False, 0.9],
        ['Sampling', 'Entire Period', False, 0.9],
        ['Sampling', '2024-09-01', True, 0.4],
        ['Sampling', '2024-09-02', True, 0.9],
        ['Sampling', '2024-09-03', True, 1.5],
        ['Sampling', 'Avg Over Days', False, 0.9],
        ['Ratio', 'Entire Period', False, 0.8],
        ['Ratio', '2024-09-01', True, 0.4],
        ['Ratio', '2024-09-02', True, 0.8],
        ['Ratio', '2024-09-03', True, 1.5],
        ['Ratio', 'Avg Over Days', False, 0.9],
        ['Mainsail', 'Entire Period', False, 1.3],
        ['Mainsail', '2024-09-01', True, 0.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', '2024-09-03', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_primary', box_schedules, sample_gap, first_day
    return(test_result)
```

In [115…

```
def test_primary_first_two_days(print_setting, debug_setting, overlay_setting):
    box_schedules = {
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 00:00:00', '2024-09-01 12:00:00', '2024-09-02 00:00
        'departure': ['2024-09-02 12:00:00', '2024-09-03 00:00:00', '2024-09-03 23:
    }
    sample_gap = 3
    first_day = '2024-09-01'
    last_day = '2024-09-02'
    tests = [
        ['Integration', 'Entire Period', False, 0.6],
        ['Integration', '2024-09-01', True, 0.4],
        ['Integration', '2024-09-02', True, 0.9],
        ['Integration', 'Avg Over Days', False, 0.6],
        ['Sampling', 'Entire Period', False, 0.6],
        ['Sampling', '2024-09-01', True, 0.4],
        ['Sampling', '2024-09-02', True, 0.9],
        ['Sampling', 'Avg Over Days', False, 0.6],
        ['Ratio', 'Entire Period', False, 0.7],
        ['Ratio', '2024-09-01', True, 0.4],
        ['Ratio', '2024-09-02', True, 0.8],
        ['Ratio', 'Avg Over Days', False, 0.6],
        ['Mainsail', 'Entire Period', False, 1.0],
        ['Mainsail', '2024-09-01', True, 0.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_primary_first_two_days', box_schedules, sample
    return(test_result)
```

In [116…

```
def test_primary_last_two_days(print_setting, debug_setting, overlay_setting):
    box_schedules = {
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 00:00:00', '2024-09-01 12:00:00', '2024-09-02 00:00
        'departure': ['2024-09-02 12:00:00', '2024-09-03 00:00:00', '2024-09-03 23:
    }
    sample_gap = 3
    first_day = '2024-09-02'
    last_day = '2024-09-03'
    tests = [
        ['Integration', 'Entire Period', False, 1.2],
        ['Integration', '2024-09-02', True, 0.9],
        ['Integration', '2024-09-03', True, 1.5],
        ['Integration', 'Avg Over Days', False, 1.2],
        ['Sampling', 'Entire Period', False, 1.2],
        ['Sampling', '2024-09-02', True, 0.9],
        ['Sampling', '2024-09-03', True, 1.5],
        ['Sampling', 'Avg Over Days', False, 1.2],
        ['Ratio', 'Entire Period', False, 1.0],
        ['Ratio', '2024-09-02', True, 0.8],
        ['Ratio', '2024-09-03', True, 1.5],
        ['Ratio', 'Avg Over Days', False, 1.2],
        ['Mainsail', 'Entire Period', False, 1.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', '2024-09-03', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_primary_last_two_days', box_schedules, sample_
    return(test_result)
```

In [117…

```
def test_some_no_inventory(print_setting, debug_setting, overlay_setting):
    box_schedules = {
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 00:00:00', '2024-09-02 12:00:00', '2024-09-03 12:00
        'departure': ['2024-09-01 12:00:00', '2024-09-03 00:00:00', '2024-09-03 23:
    }
    sample_gap = 3
    first_day = '2024-09-01'
    last_day = '2024-09-03'
    tests = [
        ['Integration', 'Entire Period', False, 0.1],
        ['Integration', '2024-09-01', True, 0.1],
        ['Integration', '2024-09-02', True, 0.1],
        ['Integration', '2024-09-03', True, 0.1],
        ['Integration', 'Avg Over Days', False, 0.1],
        ['Sampling', 'Entire Period', False, 0.1],
        ['Sampling', '2024-09-01', True, 0.1],
        ['Sampling', '2024-09-02', True, 0.1],
        ['Sampling', '2024-09-03', True, 0.1],
        ['Sampling', 'Avg Over Days', False, 0.1],
        ['Ratio', 'Entire Period', False, 0.2],
        ['Ratio', '2024-09-01', True, 0.2],
        ['Ratio', '2024-09-02', True, 0.2],
        ['Ratio', '2024-09-03', True, 0.2],
```

```
        ['Ratio', 'Avg Over Days', False, 0.2],
        ['Mainsail', 'Entire Period', False, 0.3],
        ['Mainsail', '2024-09-01', True, 0.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', '2024-09-03', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_some_no_inventory', box_schedules, sample_gap,
    return(test_result)
```

In [118…

```
def test_no_starting_inventory(print_setting, debug_setting, overlay_setting):
    box_schedules = {
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 03:00:00', '2024-09-02 12:00:00', '2024-09-03 12:00
        'departure': ['2024-09-01 15:00:00', '2024-09-03 00:00:00', '2024-09-03 23:
    }
    sample_gap = 3
    first_day = '2024-09-01'
    last_day = '2024-09-03'
    tests = [
        ['Integration', 'Entire Period', False, 0.1],
        ['Integration', '2024-09-01', True, 0.1],
        ['Integration', '2024-09-02', True, 0.1],
        ['Integration', '2024-09-03', True, 0.1],
        ['Integration', 'Avg Over Days', False, 0.1],
        ['Sampling', 'Entire Period', False, 0.1],
        ['Sampling', '2024-09-01', True, 0.1],
        ['Sampling', '2024-09-02', True, 0.1],
        ['Sampling', '2024-09-03', True, 0.1],
        ['Sampling', 'Avg Over Days', False, 0.1],
        ['Ratio', 'Entire Period', False, 0.2],
        ['Ratio', '2024-09-01', True, 0.2],
        ['Ratio', '2024-09-02', True, 0.2],
        ['Ratio', '2024-09-03', True, 0.2],
        ['Ratio', 'Avg Over Days', False, 0.2],
        ['Mainsail', 'Entire Period', False, 0.3],
        ['Mainsail', '2024-09-01', True, 0.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', '2024-09-03', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_no_starting_inventory', box_schedules, sample_
    return(test_result)
```

In [119…

```
def test_no_ending_inventory(print_setting, debug_setting, overlay_setting):
    box_schedules = {
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 00:00:00', '2024-09-02 12:00:00', '2024-09-03 09:00
        'departure': ['2024-09-01 12:00:00', '2024-09-03 00:00:00', '2024-09-03 21:
    }
    sample_gap = 3
    first_day = '2024-09-01'
    last_day = '2024-09-03'
    tests = [
        ['Integration', 'Entire Period', False, 0.1],
```

```
        ['Integration', '2024-09-01', True, 0.1],
        ['Integration', '2024-09-02', True, 0.1],
        ['Integration', '2024-09-03', True, 0.1],
        ['Integration', 'Avg Over Days', False, 0.1],
        ['Sampling', 'Entire Period', False, 0.1],
        ['Sampling', '2024-09-01', True, 0.1],
        ['Sampling', '2024-09-02', True, 0.1],
        ['Sampling', '2024-09-03', True, 0.1],
        ['Sampling', 'Avg Over Days', False, 0.1],
        ['Ratio', 'Entire Period', False, 0.2],
        ['Ratio', '2024-09-01', True, 0.2],
        ['Ratio', '2024-09-02', True, 0.2],
        ['Ratio', '2024-09-03', True, 0.2],
        ['Ratio', 'Avg Over Days', False, 0.2],
        ['Mainsail', 'Entire Period', False, 0.3],
        ['Mainsail', '2024-09-01', True, 0.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', '2024-09-03', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_no_ending_inventory', box_schedules, sample_ga
    return(test_result)
```

In [120…

```
def test_no_inventory_first_two_days(print_setting, debug_setting, overlay_setting)
    box_schedules = {
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 03:00:00', '2024-09-02 09:00:00', '2024-09-03 09:00
        'departure': ['2024-09-01 15:00:00', '2024-09-02 21:00:00', '2024-09-03 21:
    }
    sample_gap = 3
    first_day = '2024-09-01'
    last_day = '2024-09-02'
    tests = [
        ['Integration', 'Entire Period', False, 0.1],
        ['Integration', '2024-09-01', True, 0.1],
        ['Integration', '2024-09-02', True, 0.1],
        ['Integration', 'Avg Over Days', False, 0.1],
        ['Sampling', 'Entire Period', False, 0.1],
        ['Sampling', '2024-09-01', True, 0.1],
        ['Sampling', '2024-09-02', True, 0.1],
        ['Sampling', 'Avg Over Days', False, 0.1],
        ['Ratio', 'Entire Period', False, 0.2],
        ['Ratio', '2024-09-01', True, 0.2],
        ['Ratio', '2024-09-02', True, 0.2],
        ['Ratio', 'Avg Over Days', False, 0.2],
        ['Mainsail', 'Entire Period', False, 0.0],
        ['Mainsail', '2024-09-01', True, 0.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_no_inventory_first_two_days', box_schedules, s
    return(test_result)
```

In [121…

```
def test_no_inventory_last_two_days(print_setting, debug_setting, overlay_setting):
    box_schedules = {
```

```
        'container': ['A', 'B', 'C'],
        'arrival': ['2024-09-01 03:00:00', '2024-09-02 09:00:00', '2024-09-03 09:00
        'departure': ['2024-09-01 15:00:00', '2024-09-02 21:00:00', '2024-09-03 21:
    }
    sample_gap = 3
    first_day = '2024-09-02'
    last_day = '2024-09-03'
    tests = [
        ['Integration', 'Entire Period', False, 0.1],
        ['Integration', '2024-09-02', True, 0.1],
        ['Integration', '2024-09-03', True, 0.1],
        ['Integration', 'Avg Over Days', False, 0.1],
        ['Sampling', 'Entire Period', False, 0.1],
        ['Sampling', '2024-09-02', True, 0.1],
        ['Sampling', '2024-09-03', True, 0.1],
        ['Sampling', 'Avg Over Days', False, 0.1],
        ['Ratio', 'Entire Period', False, 0.2],
        ['Ratio', '2024-09-02', True, 0.2],
        ['Ratio', '2024-09-03', True, 0.2],
        ['Ratio', 'Avg Over Days', False, 0.2],
        ['Mainsail', 'Entire Period', False, 0.0],
        ['Mainsail', '2024-09-02', True, 0.0],
        ['Mainsail', '2024-09-03', True, 0.0],
        ['Mainsail', 'Avg Over Days', False, 0.0]
    ]
    test_result = test_general('test_no_inventory_last_two_days', box_schedules, sa
    return(test_result)
```

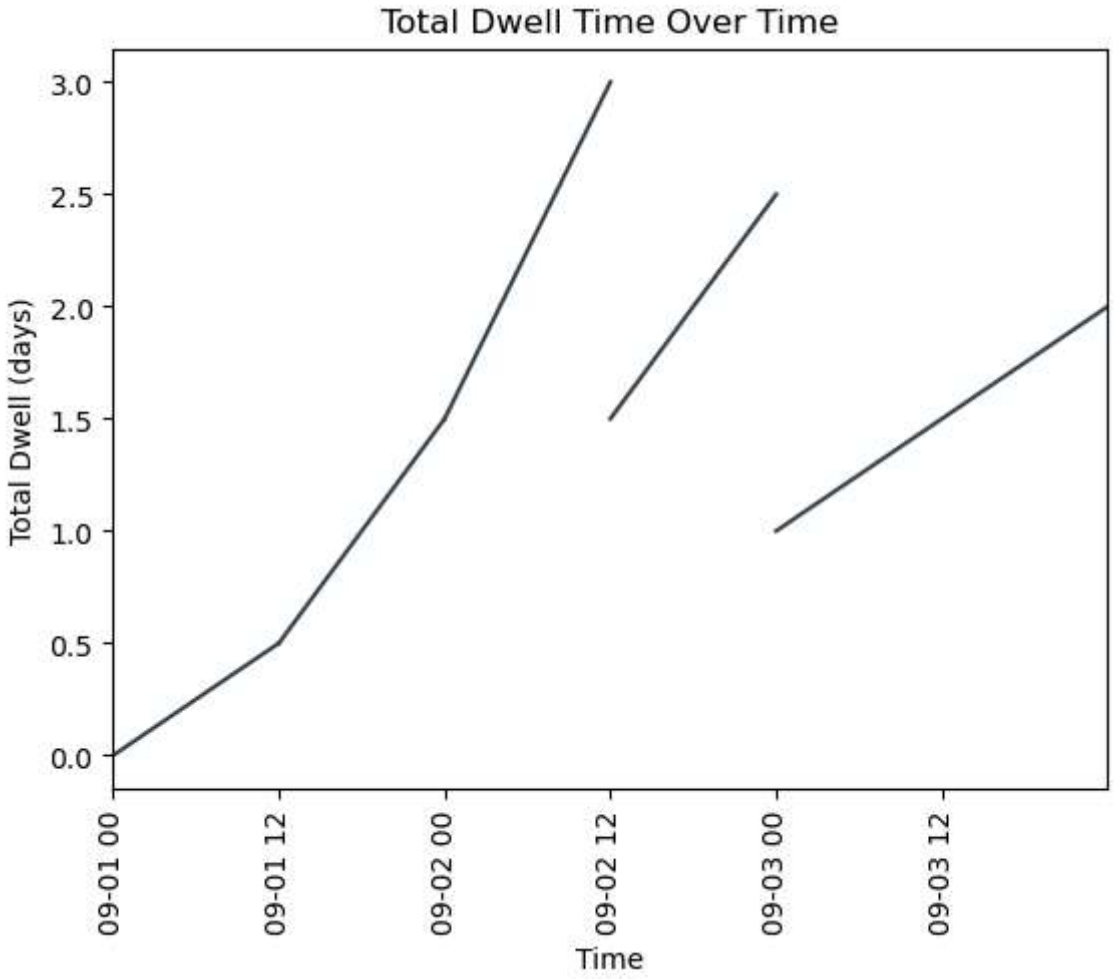
In [122…

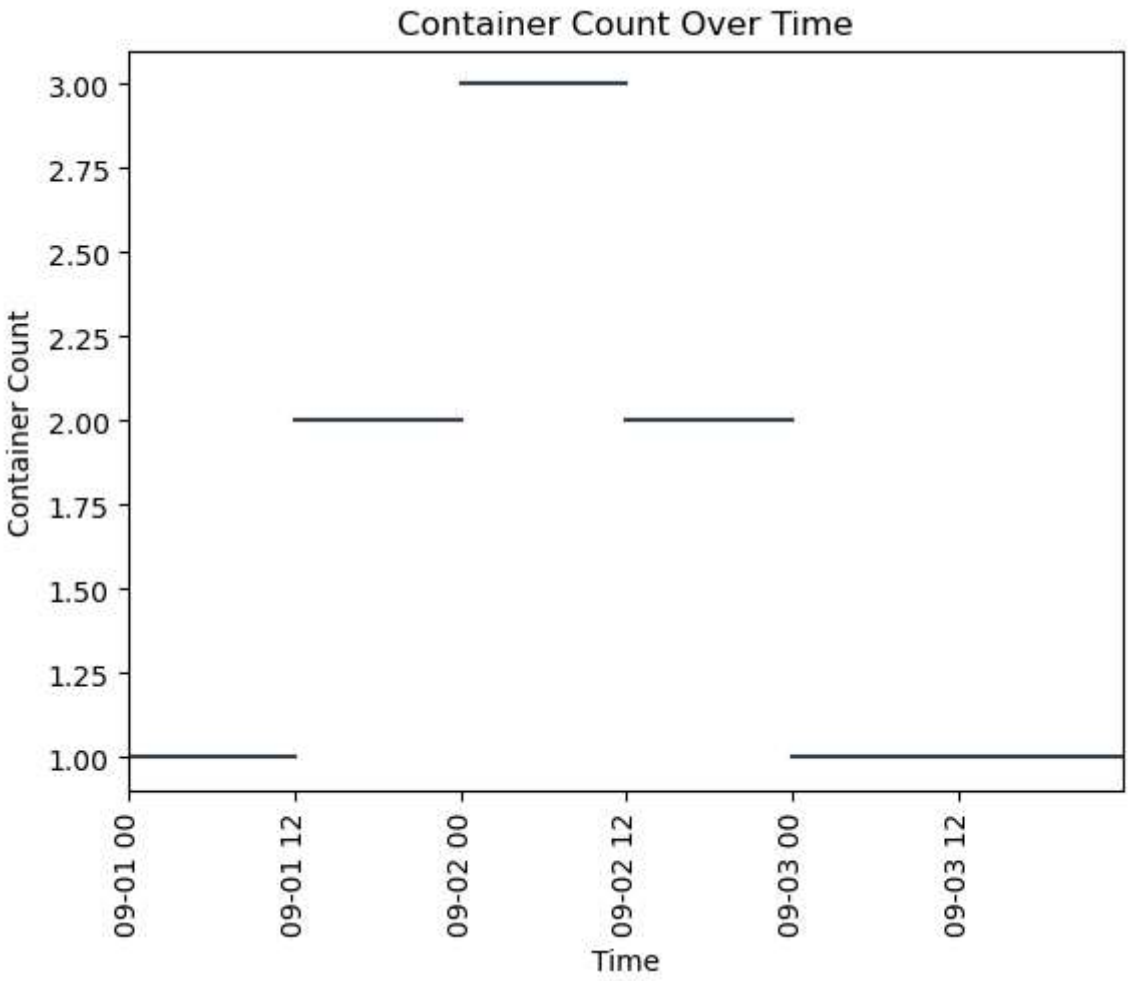
```
print(str(test_primary(False, False, False))+ ' test_primary')
print(str(test_primary_first_two_days(False, False, False)) + ' test_primary_first_
print(str(test_primary_last_two_days(False, False, False)) + ' test_primary_last_tw
print(str(test_some_no_inventory(False, False, False)) + ' test_some_no_inventory')
print(str(test_no_starting_inventory(False, False, False)) + ' test_no_starting_inv
print(str(test_no_ending_inventory(False, False, False)) + ' test_no_ending_invento
print(str(test_no_inventory_first_two_days(False, False, False)) + ' test_no_invent
print(str(test_no_inventory_last_two_days(False, False, False)) + ' test_no_invento
```

```
True test_primary
True test_primary_first_two_days
True test_primary_last_two_days
True test_some_no_inventory
True test_no_starting_inventory
True test_no_ending_inventory
True test_no_inventory_first_two_days
True test_no_inventory_last_two_days
```

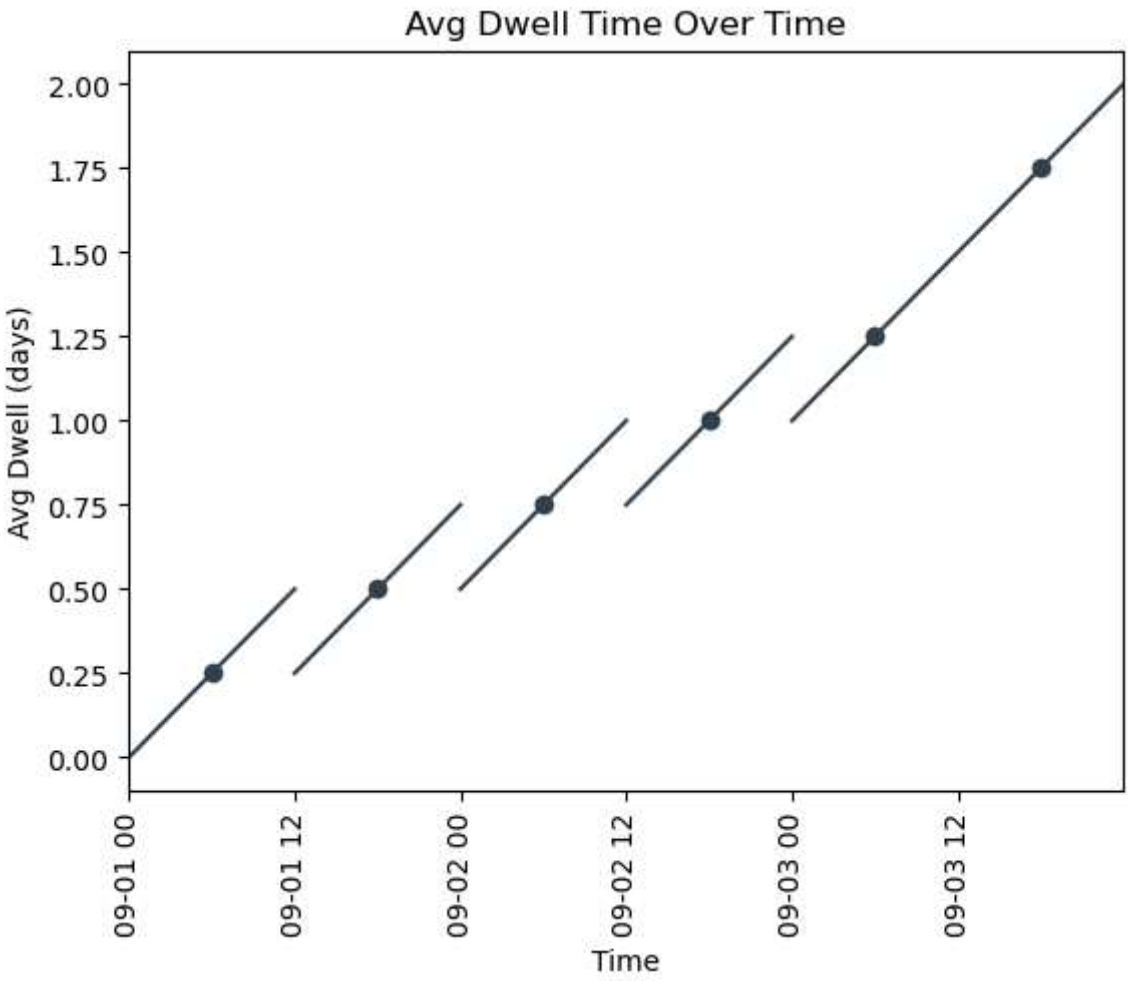
In [123…

```
test_primary(True, False, True)
```


Total Dwell Time Over Time
Total Dwell (days)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
09-01 00
09-01 12
09-02 00
09-02 12
09-03 00
09-03 12
Time


Container Count Over Time
Container Count
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
09-01 00
09-01 12
09-02 00
09-02 12
09-03 00
09-03 12
Time

Out[123]: True



Appendix: Derivations

The following has been reproduced from the Crow's Nest Wiki. This link provides access to the maintained and updated document.

Summary

To help make aggregation of functions clearer, faster, and less error prone, the math behind how to average a curve is discussed. Continuous and piecewise continuous functions are briefly introduced. A correct method for averaging functions taken from calculus is described and applied. Good approximations are also introduced and applied. Finally, rules for averaging functions simply, easily, and correctly are described and applied. The guidelines are reproduced here:

- If the metric is a ratio of functions, that is, both the numerator and the denominator change, then use an approximate method like the ratio of averages unless an exact value is truly needed. If so, use the integration method.
- If the metric is not a ratio of functions, the average can be computed by summing the average value for the container weighted by the time the container was present during

the period T. Some applied examples are:

- For yard utilization, the average value for a container visit during the day is the TEU times the portion of the day the container was on the terminal. Sum these and divide by the yard capacity and you have the average yard utilization over the period.
- For container count, the average value for a container visit during the day is 1 times the portion of the day the container was on the terminal. Sum these and you have the average container count over the period.
- For total dwell time, the average value for a container visit during the day is the difference between the container's maximum and minimum dwell times during the day divided by 2 times the portion of the day the container was on the terminal. Sum these and you have the average over the period of the total dwell time.

- Finally, averages for larger periods can be obtained by averaging the average values of smaller periods. For an approximate method like the ratio method, computing the average of a large period directly can give a different result than averaging the average values of smaller periods, but the difference is typically small for the metrics we compute as part of the Crow's Nest.

Introduction

A function is a curve. It has one and only one value for each point on the horizontal axis. Functions are common. One example is the temperature at a location over the course of the day. Another is the Dow Jones Industrial Average.

Additional examples of functions from container terminals are

- the average yard utilization
- the average container count
- the average dwell time

The yard utilization u is the sum of the TEU for every container visit v on the terminal at time t divided by the total capacity of the terminal in TEU.

$$u(t) = \frac{\sum_v s_v(t)}{C}$$

Equation 1. Yard utiliztion.

The (t) after the u and s_v mean that those values change over time. They are functions of time. As each s_v changes over time, so will u. To apply the equation, we have to find out what the values of every s_v are at a time t. Equation 1 states that the yard utilization at time t is the sum over every container visit v of the size of the container s_v divided by the yard capacity C. The notation tells us at a glance that the yard capacity is modeled as a constant. That's not strictly true, but that's how it's modeled in Equation 1. The notation also tells us

the size of the container can change over time. That's because a container that isn't on the terminal has a size of 0 in this model. When it is on the terminal it has a size equal to its TEU.

Yard utilization is a function of time. See Figure 1.

Figure 1. Yard utilization over time.

The yard utilization jumps to new values when a container enters or exits. The graph has flaws. Only the horizontal lines in the graph represent the function. The vertical lines are artifacts of how the graph was created. Because the segments of the curve don't all connect, the curve is discontinuous. However, it has pieces that are continuous and cover the entire horizontal axis. This means that it is piecewise continuous. When we average the yard utilization, the piecewise continuous nature of the function adds complications, but they get resolved.

The container count on the terminal is also piecewise continuous like the function shown in Figure 1. When a container enters, the function jumps up by 1. When a container leaves, it drops immediately by 1. This makes averaging the function potentially more complex, but the complexity can be simplified and removed.

The average dwell time $\langle d \rangle_C(t)$ is defined as:

$$\langle d \rangle_C(t) = \frac{\sum_v d_v(t)}{n(t)}$$

Equation 2. Average dwell time.

The brackets $\langle \ldots \rangle$ around the d are a notation that indicates this is an average value and the subscript C means that the average is taken over the set of containers C that are on the terminal. Equation 2 shows that the average dwell time is itself a function of time. The average dwell time may change from moment to moment. It is the ratio of the sum over every container visit v of the dwell time for container visit v at time t divided by the number of containers n on the terminal at time t. Note that the denominator in Equation 2 is a function of time whereas in Equation 1 the denominator - the yard capacity C - was modeled as a constant. The function in the denominator adds complications when averaging that get resolved by using an approximation.

Averaging Functions

From calculus, it has been shown that the average of a function is its definite integral divided by the domain. The functions we're interested in are functions of time which means that the domain is an interval of time. The domain defines the period of time over which we want to average the function. The definite integral is the area under the curve. It is the gray area in

Figure 1. The domain is the length from the left edge of the left-most bar to the right edge of the right-most bar.

Computing the average of a function using the integration method yields the expected value of the function for the period. The expected value is defined from probability and statistics. It is the value that is expected when a single value of the averaged values is selected. This means that if many values are selected that their average converges to the expected value as the number of selections grows.

To apply the integration method from calculus, the area under the curve must be computed. In the architecture of our data model, the domain is the minimum period to average over. It is set by the developer and is assumed to be a day. Larger averages are obtained at runtime by averaging over many domains, that is, averaging the average values from many days. Average values for periods smaller than a day are not available.

Application of Integration Method to Yard Utilization

Let A_i be the interval of each interval i in Figure 1 and let T be the period. We denote the average yard utilization over the domain defined by the interval T as $\langle u \rangle_T$. The integration method states that $\langle u \rangle_T$ is:

$$\langle u \rangle_T = \frac{1}{T} \sum_i A_i$$

Equation 3. Averaging yard utilization using integration.

Within each interval Equation 1 tells us that the height of the curve is $\frac{\sum_v s_v(t)}{C}$. Let the duration of interval i be t_i. The area of the rectangle is the height times the width:

$$A_i = \frac{\sum_v s_v(t)}{C} t_i$$

Equation 4. The area of interval i for yard utilization.

The notation $s_v(t)$ can be simplified when considering a single interval. When no containers enter or leave, $s_v(t)$ is constant. It is either 0 or the TEU of container visit v. The notation when discussing a single interval for $s_v(t)$ is changed to $s_{v,i}$ to show that the value is constant within interval i.

$$A_i = \frac{\sum_v s_{v,i}}{C} t_i$$

Equation 5. The area of interval i with notation for constants.

Substituting Equation 5 into Equation 3 gives:

$$\langle u \rangle_T = \frac{1}{T} \sum_i \frac{\sum_v s_{v,i}}{C} t_i$$

Equation 6. The average yard utilization unsimplified.

Equation 6 can be simplified. The interval durations t_i can be distributed into the second sum, the constant C can be factored out of the sums, and the associative property of addition permits the interchange of the summation operators.

$$\langle u \rangle_T = \frac{1}{T} \sum_i \frac{\sum_v s_{v,i}}{C} t_i =$$

$$\frac{1}{T} \sum_i \frac{\sum_v t_i s_{v,i}}{C} =$$

$$\frac{1}{TC} \sum_i \sum_v t_i s_{v,i} =$$

$$\frac{1}{TC} \sum_v \sum_i t_i s_{v,i}$$

Equation 7. Swapped summation operators for average yard utilization.

The double sum can be simplified. The value $s_{v,i}$ is the TEU of container visit v when it is on the terminal and 0 when it is off. This means that $\sum_i t_i s_{v,i} = t_v s_v$ where t_v is the duration of container visit v on the terminal during the period and s_v is the TEU of container visit v. Equation 7 simplifies the expression for the average yard utilization to be:

$$\langle u \rangle_T = \frac{1}{TC} \sum_v t_v s_v = \frac{1}{T} \sum_v \frac{t_v s_v}{C}$$

Equation 8. Simplified expression for average yard utilization.

Note that Equation 8 no longer has any dependence on the intervals of the piecewise continuous function for yard utilization. This is convenient for data modeling because interval information is complex, large, and varies if the user filters the yard utilization by fulls/emtpies, import/exports/transships/storage, etc. Using interval information requires knowing at development time what filters the user wants. Also note that the index of the sum is over the container visits v. This is a convenient index for data modeling because container visits are a fact that is already tracked. The values $\frac{t_v s_v}{C}$ can be stored by the container visit and by the day. They can be summed and divided by the user-defined period to obtain the average yard utilization for any period larger than a day. Further, none of these simplifications have been approximate. No accuracy has been lost.

Application of Integration Method to Container Count

The above equations can be adapted to yield a simple and exact average of the container count on the terminal by the day. The average of the container count is:

$$\langle n \rangle_T = \frac{1}{T} \sum_i A_i$$

Equation 9. Computing average container count via the integration method.

Equation 4 computes the area for each interval i of yard utilization. To adapt it to container count we need to replace the value for the heights of the intervals $\frac{\sum_v s_v(t)}{C}$ with the value for the height of the container count curve $\sum_v n_v(t)$ where $n_v(t)$ is 1 if the container is on the terminal and 0 if it is not. The area for each interval of container count is:

$$A_i = t_i \sum_v n_v(t)$$

Equation 10. The area of interval i for container count.

Substituting Equation 10 into Equation 9 and simplifying as was done previously yields:

$$\langle n \rangle_T = \frac{1}{T} \sum_i t_i \sum_v n_v(t) =$$

$$\frac{1}{T} \sum_i \sum_v t_i n_{v,i} =$$

$$\frac{1}{T} \sum_v \sum_i t_i n_{v,i} =$$

$$\frac{1}{T} \sum_v t_v n_v = \frac{1}{T} \sum_v t_v$$

Equation 11. The average container count.

where t_v is the time spent on terminal during period T and $n_v = 1$. The average container count is the sum of the portion of the day each container visit v spent on the terminal.

Application of Integration Method to Average Dwell Time

Applying Equation 3 to average dwell time yields:

$$\langle d \rangle_{C,T} = \frac{1}{T} \sum_i A_i$$

Equation 12. Average dwell time via the integration method.

Note that the average dwell time has two subscripts, C and T. The average dwell time is averaged over two domains. The definition in Equation (2) is averaged over the set of containers C that are on the terminal at time t. The second subscript T denotes that the average dwell time $\langle d \rangle_C$ has been averaged again - this time over time, and specifically over the time interval defined by T.

The average dwell time has a constant slope of 1 within an interval. An example of the function for the average dwell time is shown in Figure 2.

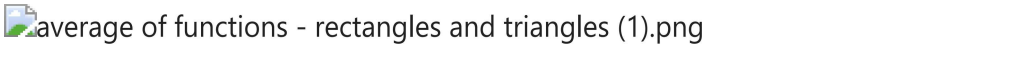


Figure 2. An example of the average dwell time changing over time.

Figure 2 has flaws. Only the top sloping segment of each interval are function values. The vertical lines and the horizontal lines dividing the triangles from the bars are not part of the function. They are artifacts from how the diagram was produced.

The area for each interval can be computed using geometry. The area of the bar is the value of the average dwell time at the start of the interval $\langle d \rangle_C(t_{i,\alpha})$ times the width of the interval. The triangle on top is an isosceles right triangle. The bottom leg and the right leg are the same length. They are both t_i in length. They are the same length because the slope of the function is always 1. Applying the formula for the area of a triangle $A = \frac{1}{2}bh$ where $b = h = t_i$ gives the area of the triangle as $\frac{1}{2}t_i^2$.

$$A_i = \langle d \rangle_C(t_{i,\alpha})t_i + \frac{1}{2}t_i^2$$

Equation 13. The area for the intervals in Figure 2.

Equation 13 can be factored and rewritten in terms of the average over the interval of the average dwell time which is written as $\langle d \rangle_{C,i}$. The average over the interval of the average dwell time is the average of the first point of the interval $\langle d \rangle_{C,i}(\alpha)$ and the last point $\langle d \rangle_{C,i}(\omega)$ where the notation $\langle d \rangle_{C,i}(t)$ denotes the value of $\langle d \rangle_{C,i}$ at a single point in time t and α and ω are the first and last points in time of the interval i.

$$\langle d \rangle_{C,i} = \frac{\langle d \rangle_{C,i}(\alpha) + \langle d \rangle_{C,i}(\omega)}{2}$$

Equation 14. The average over the interval of the average dwell time.

Since the slope is 1, the last point $\langle d \rangle_{C,i}(\omega)$ is equal to the value of the first point plus the width of the interval.

$$\langle d \rangle_{C,i}(\omega) = \langle d \rangle_{C,i}(\alpha) + t_i$$

Equation 15. The relationship between the last average dwell time in the interval and the first.

Substituting Equation 15 into Equation 14 gives:

$$\langle d \rangle_{C,i} = \frac{\langle d \rangle_{C,i}(\alpha) + \langle d \rangle_{C,i}(\omega)}{2} = \frac{\langle d \rangle_{C,i}(\alpha) + \langle d \rangle_{C,i}(\alpha) + t_i}{2} = \langle d \rangle_{C,i}(\alpha) + \frac{1}{2}t_i$$

Equation 16. The average over the interval of the average dwell time expressed without the final value.

Factoring t_i out of the expression for A_i in Equation 13 and substituting Equation 16 gives:

$$A_i = \langle d \rangle_{C,i}(\alpha)t_i + \frac{1}{2}t_i^2 = (\langle d \rangle_{C,i}(\alpha) + \frac{1}{2}t_i)t_i = \langle d \rangle_{C,i}t_i$$

Equation 17. The area of the interval expressed using the average over the interval of the average dwell time.

The definition of the average dwell time in Equation 2 is used to obtain the average over the interval of the average dwell time $\langle d \rangle_{C,i}$. The continuous intervals of the average dwell time are straight lines. This means that the average over an interval of the average dwell time always occurs at the midpoint of the interval. The total dwell time for a container is also a straight line; the midpoint is the average total dwell time for each container visit during the interval also. The average over the interval i of the dwell time for container visit d_v is denoted as $\langle d_v \rangle_i$. The value $\langle d_v \rangle_i$ is not divided by the number of containers. Summing $\langle d_v \rangle_i$ over v produces the average total dwell time during interval i.

Substituting the definition of average dwell time into Equation 17 gives:

$$A_i = \frac{\sum_v \langle d_v \rangle_i}{n_i} t_i$$

Equation 18. Final value for A_i for average dwell time.

Substituting Equation 18 into Equation 12 gives:

$$\langle d \rangle_{C,T} = \frac{1}{T} \sum_i \frac{\sum_v \langle d_v \rangle_i}{n_i} t_i$$

Equation 19. The average over the period of the average dwell time.

The n_i and t_i can be distributed into the sum over the container visits v and the summation operators swapped to give:

$$\langle d \rangle_{C,T} = \frac{1}{T} \sum_i \sum_v \frac{\langle d_v \rangle_i t_i}{n_i} = \frac{1}{T} \sum_v \sum_i \frac{\langle d_v \rangle_i t_i}{n_i}$$

Equation 20. Rearranging Equation 19.

This is as simple as the expression can be made. We cannot remove n_i from the inner sum over i because n_i depends on i. When the denominator is a function, using the integration method requires moment-by-moment pairing of the numerator and denominator. In the case of piecewise continuous functions, this means maintaining interval information.

Note that the outer sum is in a convenient grain, container visits. The values obtained from the inner sum could be conveniently stored. However, to compute the inner sum values would require knowledge of the intervals. This means that filters for the metric would need to be known in advance at development time. Also, a large amount of storage and

significant code complexity would be added which would be multiplied by the number of filters.

Application of Integration Method to the Average Total Dwell Time

To demonstrate how the function in the denominator n_i is preventing easy use of the integration method for all metrics, the average over a period of the total dwell time (as opposed to the average dwell time) is shown. The value of A_i loses the denominator n_i and becomes:

$$A_i = \sum_v \langle d_v \rangle_i t_i$$

Equation 21. The area of an interval for averaging the total dwell time (not the average dwell time) over a period.

Applying Equation 3 to compute the average over a period of the dwell time $\langle w \rangle_T$ gives:

$$\langle w \rangle_T = \frac{1}{T} \sum_i A_i =$$

$$\frac{1}{T} \sum_i \sum_v \langle d_v \rangle_i t_i =$$

$$\frac{1}{T} \sum_v \sum_i \langle d_v \rangle_i t_i$$

Equation 22. An intermediate form for the average over a period of the total dwell time.

The double sum in Equation 22 can be simplified. For each container visit v, the product of $\langle d_v \rangle_i$ and t_i is the area within interval i under the curve for the dwell time of the container visit. Summing these areas gives the total area under the curve for container visit v. By rearranging Equation 22 to obtain $\langle w \rangle_T T = \sum_i A_i$, the area under the curve is shown to equal the average value of the curve multiplied by the duration of the period. This means that for each container visit v:

$$\sum_i \langle d_v \rangle_i t_i = \langle d_v \rangle_T t_v$$

Equation 23. The sum of the areas of the intervals is the area of the domain.

In Equation 23 $\langle d_v \rangle_T$ is the average over the domain T of the dwell time for container visit v and t_v is the time that container visit v is on terminal during the domain T. Substituting Equation 23 and Equation 22 gives:

$$\langle w \rangle_T = \frac{1}{T} \sum_v \langle d \rangle_T t_v$$

Equation 24. The average over a period of the dwell time of the containers.

Choosing the average of the total dwell permits the simplification shown in Equation 23. It also causes Equation 24 to produce the expected value for the total dwell of the containers over the period T.

Alternative choices to represent the total dwell of the interval

The maximum dwell could also be chosen if resource consumption should be emphasized. However, the simplification in Equation 23 no longer applies and the resulting value cannot be considered as an average or an expected value in the usual senses of the terms. Substituting the maximum value for $\langle d_v \rangle_T$ in Equation 24 gives:

$$w_{max} = \frac{1}{T} \sum_v d_{max,v} t_v$$

Equation 25. Averaging the maximum dwells of the containers over a period.

The aggregate measure w_{max} is related to $\langle w \rangle_T$, but differs by $\sum_v t_v^2$. For example, consider a single container that is on the terminal for 8 days. See Table 1.

Period	$\langle w \rangle_T$	w_{max}	**Difference**
1	0.5	1	0.5
2	1.5	2	0.5
3	2.5	3	0.5
4	3.5	4	0.5
5	4.5	5	0.5
6	5.5	6	0.5
7	6.5	7	0.5
8	7.5	8	0.5
1-2	1	1.5	0.5
3-4	3	3.5	0.5
5-6	5	6.5	0.5
7-8	7	7.5	0.5
1-4	2	2.5	0.5
5-8	6	6.5	0.5
1-8	4	4.5	0.5

Table 1. Example values from Equations 24 and 25.

In Table 1 Equation 24 shows $\langle w \rangle_T$ and w_{max} for the smallest periods - single days - and for larger periods composed of days. For the largest period, the value of w_{max} of 4.5 is much closer to the average value of 4 than to the maximum value for the period of 8. The value of w_{max} does not produce the maximum value for periods larger than the minimum period. It is inconsistent, moving from the maximum value for the smallest periods to the close to the average value for large periods. This occurs because averaging the daily values is similar to averaging sampled values. Averaging greater numbers of samples causes the average to converge on the average value. The reason the w_{max} average differs from the true average is that the values it uses are biased - they are always the largest possible value. Had the average value been used instead of the maximum value, then the average of the average values would converge to the true average. The average of the maximum values does not converge to the true average. It is always biased with a positive error.

Any alternative to $\langle d_v \rangle_i$ to represent the total dwell for container visit v during interval i must assure that $\sum_i d_{v,i}^{alt} t_i = \langle d_v \rangle_T t_v$ or the aggregate result will not be the average of the total dwell over the period.

Approximate Methods

A common way to approximate the average of any function or combination of functions is to use sampling. Another common way to approximate the average of a ratio of two functions is to compute the average of the numerator and divide by the average of the denominator.

Sampling

The concept of sampling is straightforward. Compute the metric at several values of t and then average the computed values. The selected points are called samples. If the samples are representative of the period, then the average of the samples is accurate. Regardless, as the number of samples taken increases, the accuracy of the result is expected to increase. Sampling approaches the value produced by the integration method as the number of samples becomes large.

Ratio of Averages

When a metric has functions in the numerator and the denominator, a good approximate method is to compute the average of the numerator and divide by the average of the denominator. This will not yield the same value as the integration method, but it is good enough for most purposes and greatly simplifies the complexity of the method, reduces storage requirements, and permits filtering the metric at run time.

Applying the ratio of averages to the average over a period of the average dwell time means dividing Equation 24 by the final expression of Equation 11:

$$\langle d \rangle_{C,T} = \left\langle \frac{\sum_v d_v(t)}{n(t)} \right\rangle_T \approx \frac{\langle w \rangle_T}{\langle n \rangle_T} = \frac{\sum_v \langle d_v \rangle_T t_v}{\sum_v t_v}$$

Equation 26. Applying the ratio of averages to the average over a period of the average dwell time.

The numerator and the denominator can be computed without interval information which permits filtering to be done at runtime. The index of the sum is over the container visits v which is a convenient grain for the data model. However, this method is approximate and this means that computing averages for large period different ways may produce different results. Computing the monthly average directly may differ from computing it by averaging the daily averages.

Example

The average dwell time for an example is aggregated using the integration method, sampling, and using the ratio of averages. The example is for three containers entering and leaving on the same day. The containers have the entry and exit times shown in Table 2.

Containers	1	2	3
Entry	0	6	9
Exit	12	18	24

Table 2. Entry and exit times for the three containers in the example.

The entries and exits of the containers are shown in Figure 3 as part of the container count function. The total dwell summed across all containers is shown in Figure 4. The average dwell time is shown in Figure 5.

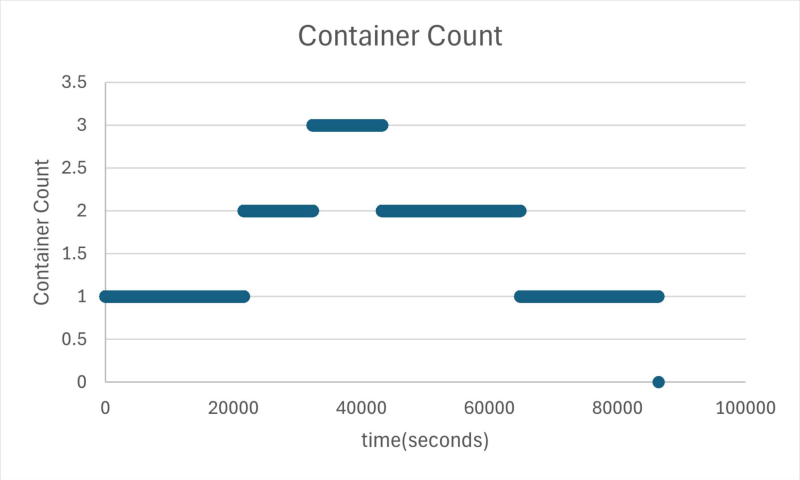


Figure 3. Container count as a function of time.

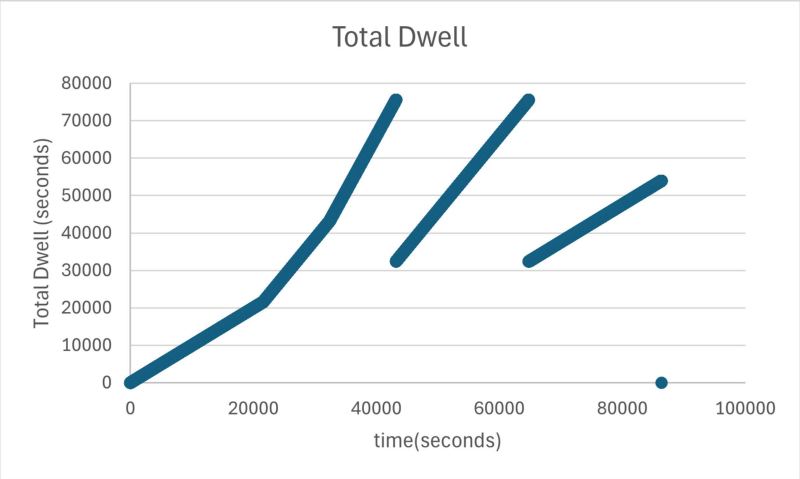


Figure 4. The total dwell summed across all three containers.

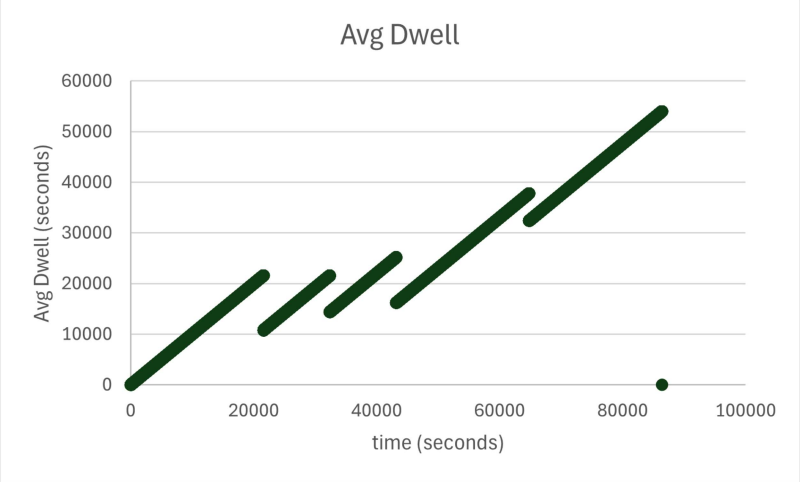


Figure 5. The average dwell as a function of time for the example.

Applying the integration method requires computing the area under the curve in Figure 5. There are five intervals of 6, 3, 3, 6, and 6 hours duration. This results in the following areas computed using the penultimate expression of Equation 17:

Interval	$\langle d \rangle_i (\alpha)$	t_i	**Area**	**Square hours**
1	0	6	$\frac{1}{2}6^2$	18
2	3	3	$3 \cdot 3 + \frac{1}{2}3^2$	13.5
3	4	3	$4 \cdot 3 + \frac{1}{2}3^2$	16.5
4	4.5	6		45

Interval	$\langle d \rangle_i (\alpha)$	t_i	Area	Square hours
			$4.5 \cdot 6 + \frac{1}{2}6^2$	
5	9	6	$9 \cdot 6 + \frac{1}{2}6^2$	72

Table 3. Interval areas for Figure 5.

The sum of the areas is 165 square hours. Dividing by 24 hours gives an average over the period for the average dwell time of 6.875 hours.

Sampling was applied to this example. Samples were taken hourly, every minute, and every second. The results were then averaged to get an average over the period for the average dwell time.

Frequency	Avg Value (hours)
Hour	6.12
Minute	6.861901
Second	6.874782

Table 4. Sampling results for the example.

The sampling results approach the expected value obtained from integration as the sampling rate increases.

The ratio of averages method was applied. The average total dwell time was computed and the average box count was computed and their ratio taken to obtain a value. The average total dwell is $\sum_v \langle d_v \rangle_T t_v$ (Equation 24) and the average box count is $\sum_v t_v$ (Equation 11). The factors of $\frac{1}{T}$ have been omitted because they cancel.

$$\sum_v \langle d_v \rangle_T t_v = 6 \cdot 12 + 6 \cdot 12 + 7.5 \cdot 15 = 256.5 \text{ hours}^2$$

$$\sum_v t_v = 12 + 12 + 15 = 39 \text{ hours}$$

The ratio $\frac{256.5}{39} = 6.576923$. The ratio of averages produces a value that is too low in this instance but is a reasonable approximation that is much simpler to obtain.

Guidelines

Guidelines surface from this analysis:

- If the metric is a ratio of functions, use an approximate method like sampling or the ratio of averages unless an exact value is truly needed. If so, use the integration method.
- If the metric is not a ratio of functions, the average can be computed by selecting a grain and then summing over the grain of the table the average value for the function over the period T. Some applied examples are:
  - For yard utilization, the average value for a container visit during the day is the TEU times the portion of the day the container was on the terminal.
  - For container count, the average value for a container visit during the day is 1 times the portion of the day the container was on the terminal.
  - For total dwell time, the average dwell time for a container visit during the day is the container's maximum dwell during the day divided by 2 times the portion of the day the container was on the terminal.
- Finally, averages for larger periods can be obtained by averaging the average values of smaller periods. For an approximate method like the ratio method, computing the average of a large period directly can give a different result than averaging the average values of smaller periods, but the difference is typically small for the metrics we compute as part of the Crow's Nest.

Conclusions

Weighted averages apply well to metrics that are not ratios of functions. Weighted averages can be applied to sums of constants and can be applied to sums of functions if the average value of the function is used. When a metric is a ratio of functions, apply an approximate method like sampling or the ratio of averages unless an exact value is truly needed.