For office use only	Team Control Number	For office use only
T1 _____	**US-7054**	F1 _____
T2 _____		F2 _____
T3 _____		F3 _____
T4 _____		F4 _____

2017

USA Regional Round
The International Mathematical Modeling Challenge (IM^2C)
Summary Sheet

(Attach a copy of this page to the front of your solution paper.)

Anyone that has flown across timezones for business meetings, conferences, or vacations knows that many days of this extended visit are dedicated to simply recovering from fatigue and jet lag caused by traveling over multiple time zones. Companies that organize international events, such as the International Meeting Management Corporation (IMMC), are faced with the issue of sub-optimal productivity from their participants due to the fatigue associated with jet lag. In this paper, we aim to define the relationship between jet lag and productivity as well as find the best location to hold an international event with participants from all across the globe.

The first part of our model deals with quantifying the amount of productive hours that the participants lose traveling across multiple time zones. We used the fact that human circadian rhythms are sinusoidal to predict the degree by which a conference member was disjoint from the real time at the location of the conference, thus calculating the amount of hours in a day that they would be productive.

Next, we developed a productivity metric κ that takes into account the number of productive hours an individual can contribute given by the first part of the model, the fatigue caused by distance traveled calculated using the Vincenty formula, and the root mean square of the individual distances traveled. The productivity metric is calculated using a root mean square function of the individual productive hours. In addition, we created a cost metric M that combined together the costs of lodging and air-travel using empirical data.

Using these metrics, we created a Python script that derived data from the GeoPy and PyTZ libraries in addition to the TimezoneDB database to implement our algorithm using real-time data. For 282 predetermined locations around the world, we calculated the ratio between our two metrics ($\frac{M}{\kappa}$) and ranked the locations from least to greatest ratio. **We found that for the small conference in mid-June, IMMC should host a conference in Irkutsk, Russia. In the case of the larger conference in January, IMMC should host it in Samara, Russia.** In addition, we investigated the case that IMMC may not have data on the location of the members attending the conference, we used ran our algorithm and 1000 times with 10 participants from randomly chosen locations. The sampling was weighted by Gross Domestic Product so as to simulate the distribution of conference members from each country. **We found that when the location of members in a conference is not known, Dublin, Ireland is the ideal location to hold a conference.**

Though our model does not account for effects of climate on productivity or variable lodging costs and flight availabilities, it is robust against changes to the various parameters that control the metrics, thus making it an effective in pinpointing the optimal location to hold such international conferences.

Introduction

Every day, millions of airline passengers are affected by the phenomenon of jet lag, a common sleeping disorder caused by traveling through multiple time zones. The International Meeting Management Corporation (IMMC) is sponsoring a three-day long event with participants from all across the globe and wants to ensure that the participants are well-rested at the start of the event. In this study, we aim to model the presence of jet lag as well as its effect on the productivity of international participants doing intense intellectual work.

Originally thought to be an issue of the conscious state of mind, jet lag is now considered to be a biological disruption of our "internal clock", otherwise known as the circadian rhythm [1]. Previous attempts at modeling jet lag include using differential equations to describe the oscillatory change of the circadian rhythm and using experimental data from biological studies to determine the values of constants [2]. In addition to addressing these characteristics, our work directly parametrizes the effect of jet lag on productivity, and thus can be applied more realistically to organize and schedule global conferences.

To determine the optimal location for an international event, we developed a reproducible algorithm suitable to the needs of IMMC. In addition, we have conducted test cases to assess the viability and sensitivity of our model. In the case that the locations of members attending a conference are unknown, we have also used the algorithm to identify an ideal location to hold the global conference.

Restatement of Problem

The International Meeting Management Corporation requests us to develop an algorithm to find the best location for an international event that will last three days band test the effectiveness of our algorithm with the following scenarios:

Scenario 1 - Small Meeting in mid-June

- Six individual participants from: Monterey CA, USA; Zutphen, Netherlands; Melbourne, Australia; Shanghai, China; Hong Kong (SAR), China; Moscow, Russia.

Scenario 2 - Big Meeting in January

- Eleven individual participants from: Boston MA, USA (2 people); Singapore; Beijing, China; Hong Kong (SAR), China (2 people); Moscow, Russia; Utrecht, Netherlands; Warsaw, Poland; Copenhagen, Denmark; Melbourne, Australia.

In essence, this problem requires us to do 4 tasks:

1. Calculate the amount of jet lag a participant experiences travelling to the destination of the event.
2. Quantify the effects that jet lag has on productivity for each participant.
3. Find the airfare and lodging cost associated with the travel.
4. Considering the above factors, determine the optimal location to hold the event.

General Assumptions

- Seasonal variations in productivity and seasonal changes to airfares are difficult to quantify.
  - The effect of seasonal variance on productivity, such as fatigue caused by cultural or socioeconomic factors, is hard to quantify. Seasonal trends in airfare is a socioeconomic variable that is dependent on the cultural variance around the world.
- Conference members will be most productive in a climate that mirrors their climate of origin.
  - It is difficult to characterize the effects of climate, weather, and geography on productivity. Therefore, we postulated that conference members would be most productive and comfortable in a climate similar to their climate of origin. Our model is designed to average the latitudes of the conference members because the productivity metric is dependent on distance.
- Business travellers do not prepare to correct jet lag and follow their internal circadian clock until arrival.
  - Remedies to combat jet lag, such as sleeping at certain times or not sleeping at all, are difficult to parametrize. Instead, we assume that travellers sleep according to their circadian rhythms until they arrive at their destination.
- Participants only work when they are together, i.e. there is no individual work involved.
  - We assume that the purpose of this international conference is to have a diverse group of intellectuals to work collectively on an issue. Thus, we assume that they can only work when every participant is present at the event.
- There are flights available to fly to any given location at the times we specify.
  - Our model does not consider the availability of flights in different locations. To schedule the conference, we try to maximize the amount of communal productive time for each conference member, which requires scheduling flights at exact times for each member.

General Definitions

Circadian Rhythm: The circadian rhythm is a cyclic biological process that allows organisms to perform daily processes without external day-night cues. The circadian rhythm for a human, on average, is 24.5 hours.

External Clock: The external clock, ξ_e is the time at the given location given by physical timekeeping devices.

Internal Clock: The internal clock, ξ_i, is the internal state of any given member at the conference. That is, ξ_i describes what each member believes the time is, according to the circadian rhythm and level of fatigue. For instance, for a member that travels to a timezone 5 hours eastwards, ξ_i =12:00PM when the time is 5:00PM in the new time zone.

Jet Lag: Jet lag occurs when the internal clock and external clock are incongruous. The amount of jet lag is defined and calculated as the difference between the initial and final time zones of travel. That is, $\Delta J = Z_i - Z_f$.

Productivity (κ): A measure of the restfulness of the participant. This value decreases as jet lag increases.

UTC Offset: The amount of hours (forward or backward) from Greenwich Mean Time (Universal Time).

Model Design and Justification

Calculating Jet Lag

Since IMMC wants to organize an international event that consists of intense work over a course of three days, it is imperative that the penalty of jet lag is minimized for each participant.

This part of the model serves to find the amount of jet lag a participant will experience as he or she travels through different time zones. The objectives of this part of the model are to quantify the amount of productive hours lost for the participants as a function of jet lag and, secondly, to investigate the implications of jet lag on productivity.

Specific Assumptions

- Business hours occur between 8:00 AM and 6:00 PM in the location's timezone. Buildings are closed between midnight and 8:00AM.
  - Business hours typically follow a 9-5 schedule, and as this conference is an intensive work session, it is safe to assume that proceedings may extend 1 hour in either direction. In addition, nighttime for most locations is between midnight and 8AM, so it is safe to assume that most buildings are closed during this time.
- We assume that the circadian rhythms of all participants are oscillatory with an average period of 24.5 hours.
  - Antonsen *et al* discovered that the neurons responsible for managing the circadian rhythm behave in an oscillatory fashion. In addition they found that without any light cues, the normal human circadian rhythm is around 24.5 hours. This is why the jet lag experienced by travelling eastwards is often worse than traveling a commensurate distance westwards [2].

Specific Definitions

Productive Hours: The hours between 8AM and 6PM in which the participants are able to do intensive work.

***S*:** The amount of productive hours lost due to jet lag.

Quantifying the Loss of Productive Hours

We assume that the participants will work during the business hours of 8AM to 6PM. Ideally, every participant will arrive to the event with their internal clocks in sync with the external clocks. However, jet lag imposes a phase shift in the circadian rhythm, and thus there is an inevitable loss of productive hours since the participants will not "feel" like their productive hours vary from those of their location. To quantify the exact loss in productive hours, S, we have to consider the effects of jet lag.

In regards to constructing the function $S(\Delta J)$, we know that the function must be sinusoidal, since the oscillatory behavior of the participants' circadian rhythms causes a periodic fall and rise in the perception of productive hours lost. Initial jet lag should cause a small loss in productive hours, while a ΔJ of 10 and ΔJ of 11 should cause a large loss in productive hours but not have a noticeable difference between them. Additionally the function will be bounded by $S = 0$ (since there can never be a negative amount of hours) and $S = 12$ which represents a complete reversal in the perception of night and day. Thus far, we have the following components:

$$S = 6 - 6\cos\left(\omega\Delta J - \phi\right)$$

Finally, we know that S should be repeat every day ($\frac{2\pi}{\omega} = 24$ hours) and that the phase shift ϕ should be 0.5 hours. This has been found to be a result of the circadian cycles of humans being equal to 24.5 hours instead of the expected 24 hours. Thus when a person experiences a day of 24 hours, that person is $24.5 - 24$ hours off their circadian cycle. Including all of these elements, we can predict the amount of productive hours lost as a function of jet lag.

$$S = 6 - 6\cos\left(\frac{\pi\Delta J_m}{12} - \frac{\pi}{24}\right)$$

Implications on Productivity

Because a typical working environment is typically closed for 8 hours during the night, we have a total of 16 hours that the participants are awake. During those active hours, we can represent the additional time an individual is unable to work during the day as S. Thus, the remaining $16 - S$ hours are available for the individual to use towards the group effort up to a maximum of 10 hours which is the length of our business day. The figure below summarizes the typical breakdown of the participants' day.

Figure 1: Jet lag causes a shift S of 4 hours

In the example above, the participant loses 4 productive hours due to jet lag ($S = 4$), and therefore is still able to find 10 hours in the day to do productive work, with an excess of 2 hours for non-work related activities. However the worst case would be a 12 loss of productive hours due to jet lag (when the internal and external clocks are 180° out of phase), and the participant will only be able to work 4 hours as a result. Therefore, the amount of productive hours of an individual–the quantity we seek to maximize–is $\min(10, 16 - S)$.

The number of productive hours of each individual m is thus

$$P_m = \min\left\{10, 10 + 6\cos\left(\frac{\pi\Delta J_m}{12} - \frac{\pi}{24}\right)\right\}$$

P_m is the measure of the total hours an individual is expected to be productive. However we reasoned that the overall productivity of all participants is more important than the individual productivity. We assume that the conference is a group effort, and thus it is not efficient if one member is productive for the full ten hours while others are only productive for one hour. To measure the overall jet lag productivity, J, we can use a root mean square (RMS) on all of the P_m:

$$J = \sqrt{\frac{1}{|M|}\sum_{m\in M} {P_m}^2}$$

Maximizing Productivity κ

The objective of this part of the model is to define the productivity metric, κ. We reasoned that two factors can hinder a participant's productivity: the amount of jet lag experienced (which we accounted for in the previous section) and the distance traveled. In regard to distance, we find the mean traveled distance of all participants and consider it a multiplicative factor of productivity.

Specific Assumptions

- Productivity is solely a function of distance traveled and jet lag.
  - We do not consider other hard-to-quantify effects on productivity.
- We assume that the amount of jet lag decreases by one hour with every one night of 8-hour sleep time.
  - Unlike sleep deprivation, circadian rhythm requires external light cues and behavioral changes to readjust to a new time zone. Every night of proper rest allows for the circadian rhythm of the body to shift 1 hour in the proper direction for the given time zone [1].

Adjusting for Distance Traveled

The distance component of the productivity metric, D, parametrizes the effect of traveling long distances on fatigue and ultimately productivity. We reasoned that even if participants are not jet lagged upon arrival to the event, they may very well be fatigued if they have traveled thousands of miles. Therefore the cost of jet lag does not necessarily outweigh the inconvenience of long-distance travel, and thus the distance must be factored into the productivity metric. Specifically, we want the distance component to count more significantly as every mile is added (convexity of the metric) and to decrease the productivity κ as the amount of miles traveled increases.

First, we found that the maximum distance the participants can traverse (assuming no round trips) is half the circumference of the Earth [1]. In comparison, we expressed the total distance traveled as the RMS of each individual distance, L_i, that every participant traveled. Note that squaring the individual distances will implement convexity into the distance component of the productivity metric as desired. Finally we define the distance component to be the ratio of the maximum possible distance that a participant can travel over the RMS of every individually traversed distance.

[1] We used the Vincenty approximation of the circumference of the Earth, which is 24,901 miles

Given distances L_1, L_2, ... , L_n, we can therefore model the distance component as:

$$D = \frac{\frac{1}{2}C_{\oplus}}{\sqrt{\frac{1}{|M|}\sum_{d \in L} d^2}}$$

Here $C_{\oplus}$ represents the earth's circumference, the maximum flight distance. As the flight distances increase, the smaller the value of D. Hence by multiplying the distance and jet lag components together, productivity will decrease as the total distance traveled increase, fulfilling our expectations of the distance component. In summary, we have defined the jet lag component, J, and the distance component, D, and expressed the productivity as the product of the two components.

Metric 1: Productivity Metric

The productivity metric κ is calculated by multiplying the Jet Lag Component J with the distance component D. The Jet Lag Component J is defined for a set of members M:

$$J = \sqrt{\frac{1}{|M|}\sum_{m \in M} {P_m}^2}$$

Where P_m is defined as follows and given that ΔJ_m is the shift in time zones for a given member m:

$$P_m = \min\left\{10, 10 + 6\cos\left(\frac{\pi \Delta J_m}{12} - \frac{\pi}{24}\right)\right\}$$

The distance component D is defined as follows, where L is the set of distances traveled by individual members and $C_{\oplus}$ is the circumference of the Earth:

$$D = \frac{C_{\oplus} \cdot \sqrt{|M|}}{2\sqrt{\sum_{d \in L} d^2}}$$

The productivity metric κ is therefore defined as:

$$\kappa = J \times D$$

$$\kappa = \sqrt{\frac{1}{|M|}\sum_{m \in M} {P_m}^2} \times \frac{C_{\oplus} \cdot \sqrt{|M|}}{2\sqrt{\sum_{d \in L} d^2}}$$

Finding Cost

Ideally, we would like to be able to book flights without regard to cost, but like with every organization, IMMC has a budget to maintain. The task of this part of the model is to define the cost of hosting the participants at the international event.

Specific Assumptions

- The average fare for a ticket, given a distance d, can be calculated as $50 + 0.11d$ dollars. Furthermore, we assume that miscellaneous costs such as food will be the same at all locations, so we do not consider them in our cost calculations.
  - Ticket prices and flight distance have been found to follow this linear relationship on average [3]. It is difficult to quantify the cost of food and other miscellaneous costs such as flying business class, and since these costs will not affect the relative cost metrics, we choose to ignore them.
- We estimate the cost of lodging in any given country by using the average lodging expense of a 4-star hotel, which is \$150 [4].
  - Business professional conferences will likely occur in urban locations and therefore must provide the participants with a suitable place of stay. We assume that IMMC will want to provide a 4-star hotel to accommodate the lodging of the participants.
- The participants will leave at end of the final day of the conference.
  - In order to prevent an extra night of lodging costs, we assume that IMMC will not cover lodging fees after the third day of the conference.

Model Design of Cost

Using the assumption above, total cost of a flight, C, traveling distances, $d_1, d_2, ..., d_n$, can be represented as:

$$C_{\text{tot}} = \sum_{i \in P} d_i \cdot 0.11 + 50$$

Although, we used an empirical source of data to provide the flight rates and the airfare constant (\$50), we wanted the principal parameter of $C(d)$ to be the distances, d_i, that the participants are travelling [3]. We reasoned that airfares vary unpredictably as time progresses and economies change; thus, we used prior data to only provide an estimation of the rate and the airfare constant respectively. Therefore, when considering the ranking of locations, we made a note to make sure the cost model is robust against changes to flight rates and the airfare constant.

Once the participants arrive at the location, they will eventually require a place to stay. We assumed that IMMC wants to provide a four-star hotel for their guests, thus we used the average cost of \$150 per night for a four-star hotel [4]. To be safe, we used the average cost of a four-star hotel because if a country charges a higher price, then the participants will still be able to lodge at a hotel that is one or two stars lower.

Since the competitions is three days long, the participants will be staying three nights (the first night will the night before the conference). Thus if the number of people is $|P|$, the lodging cost, L is:

$$L = 3 \cdot |P| \cdot 150$$

.

Metric 2: Cost Metric

The cost metric M of holding the meeting at a given location is given by the cost of the air tickets to that location (C) plus the lodging cost at that location (L).

The total air ticket cost (for the set P of people) is given by:

$$C_{\text{tot}} = \sum_{i \in P} d_i \cdot 0.11 + 50$$

Where $d(i)$ is the distance from each person's location to the location of the meeting.

The lodging cost per night c is calculated by averaging the cost of 4-star hotels [4]. Given that $c = 150$,

So,

$$L = 3 \cdot |P| \cdot 150$$

$$L = 450 \cdot |P|$$

and

$$M = C + L$$

Hence, in our sensitivity analysis, we will specifically ensure that our model is not sensitive to large changes in the flight rates and airfare constants.

Model Implementation and Algorithm

We used the Python programming language to implement our model and generate an algorithm that would output a list of ranked locations to hold a global conference. From a list of around 480 unique time standards from TimezoneDB [5], we assembled a list of 282 unique locations (regions or cities). Since some locations have more than one timezone associated with them and other locations are inaccessible, we chose to reduce the list rather than burden the algorithm with unnecessary. However, for the sake of providing flexibility to IMMC, adding new locations in our algorithm can be easily accomplished.

A python library called GeoPy [6] was used to measure distances between two locations, given their latitude and longitude. In conjunction with another library called PyTZ [7], we were also able to determine the timezone of the location, and subsequently the UTC offset of that location at a given month and year. Therefore, although we simulated our model for the year of 2017, our program can be used for any given time and month.

For the 17 conference members of concern to IMMC, we began by using GeoPy and PyTZ to extract the latitude, longitude, and UTC offset for each member. We also used GeoPy and PyTZ to extract the same data for all 282 predetermined locations. We also extracted Gross Domestic Product (GDP) data for each country associated with each timezone from the International Monetary Fund [8]. The data was compiled into one source, and used for two processes. The data was used to (1) calculate the preferred locations for both conferences that IMMC asked us to determine, and (2) to determine an optimal location for global conferences when the number or location of participants is unknown.

To calculate preferred locations for each conference for IMMC, we iterated over all 282 locations for each conference and determined which produced the lowest $\frac{M}{\kappa}$ value. We used this ratio to rank our locations because we wanted to maximize the productivity metric (κ) and minimize the cost metric (M). To do so, for any given location, we calculated the ΔJ value for all members using their UTC offset and comparing it to the UTC offset of the location. We also calculated the distance between the conference member and the location by using a Vincenty algorithm that was implemented in GeoPy, which utilized an ellipsoid to approximate the distances between two points on Earth.

To determine an optimal location for global conferences when the number or location of participants is unknown, we took a weighted sample of our 282 locations to simulate a random draw of 10 conference participants. The sample selection was weighted by the GDP of their location. That is, for a country that has a higher GDP, a person from that country is more likely to be sampled for this section of the algorithm. This allowed us to determine a general location that approximated the general economic trends of different countries. We calculated M and κ for every location, and ran the algorithm 1000 times to determine which location was chosen the most often as the location for the global conference for the random weighted sample. The full Python code can be found in the appendix.

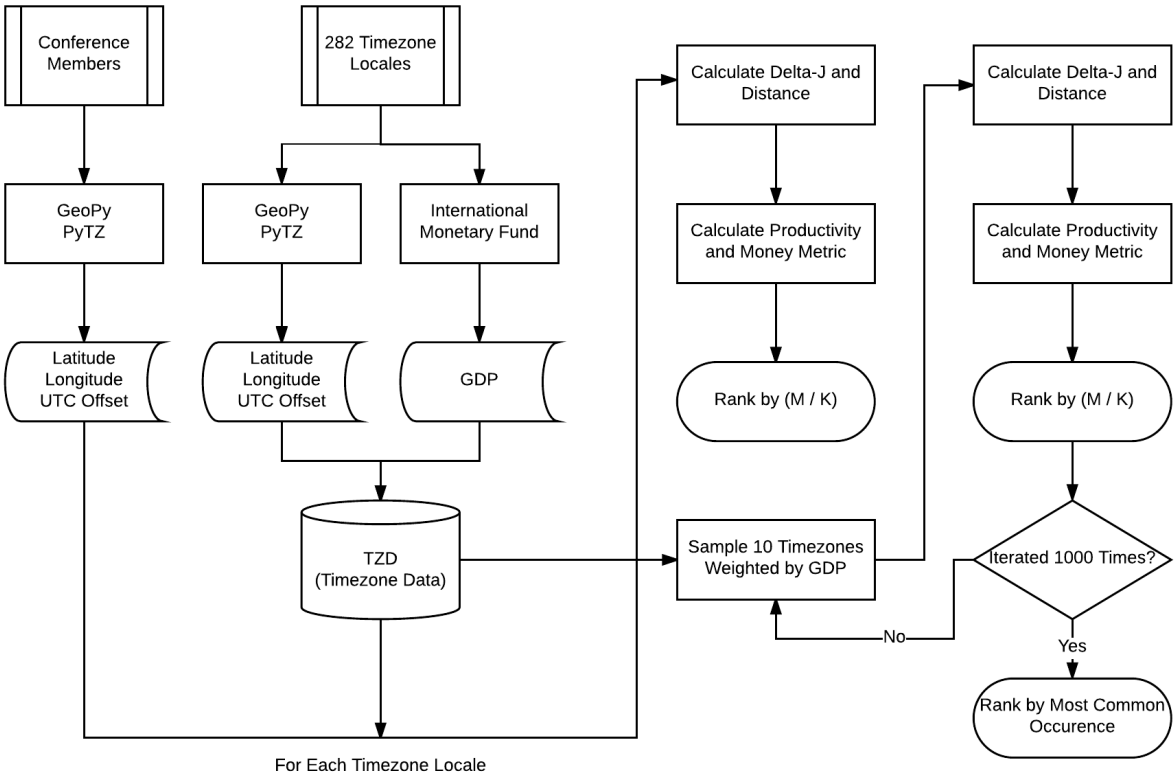

Figure 2: A flowchart of the algorithm that was implemented in Python.

Results

By running our algorithm on the given data sets for Conference 1 and Conference 2, we obtained the following results. Since we wish to maximize productivity and minimize cost, cities were ranked by the program using the heuristic $\frac{M}{K}$:

Conference 1: 6 Attendees; Mid-June The optimal location to hold this conference is Irkutsk, Russia, a historically significant city valued for its famous art museums and churches [9]. This location for the conference results in minimized inter-timezone travel for the attendees and maximizes the $\frac{m}{\kappa}$ metric.

Figure 3: Map showing the flight routes (calculated with Google Flights) of each of the attendees (starting cities are marked in green and Irkutsk is marked with the pin [10, 11].)

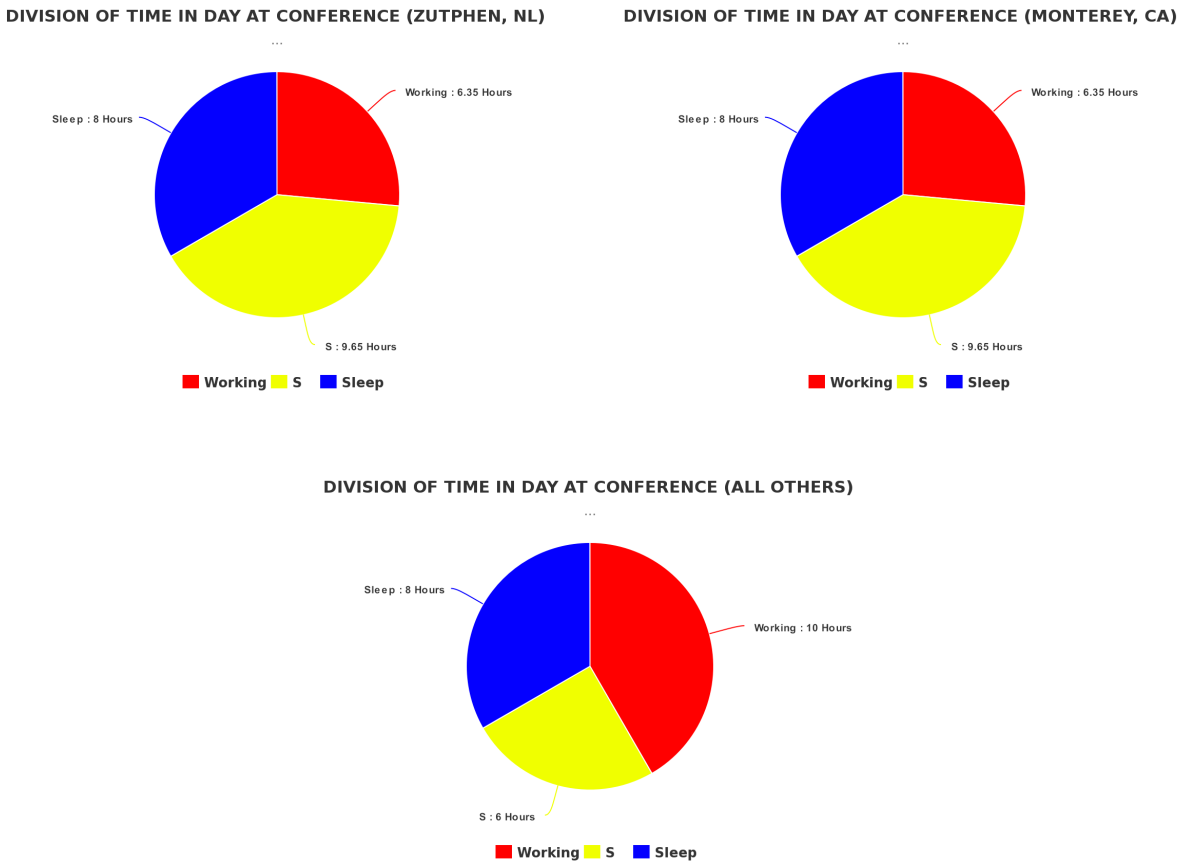

Figure 4: The Divisions of Time of Day for the members of Conference 1. [12]

By analyzing the map, we see that Irkutsk is the optimal location because it creates more inter-latitudinal movement than inter-timezone movement. The people having to face the largest timezone jumps are those from California and the Netherlands.

Conference 2: 11 Attendees; January The optimal location to hold this conference is Samara, Russia, the sixth largest city in Russia [13]. This location for the conference results in minimized inter-timezone travel for the attendees and maximizes the $\frac{m}{\kappa}$ metric.

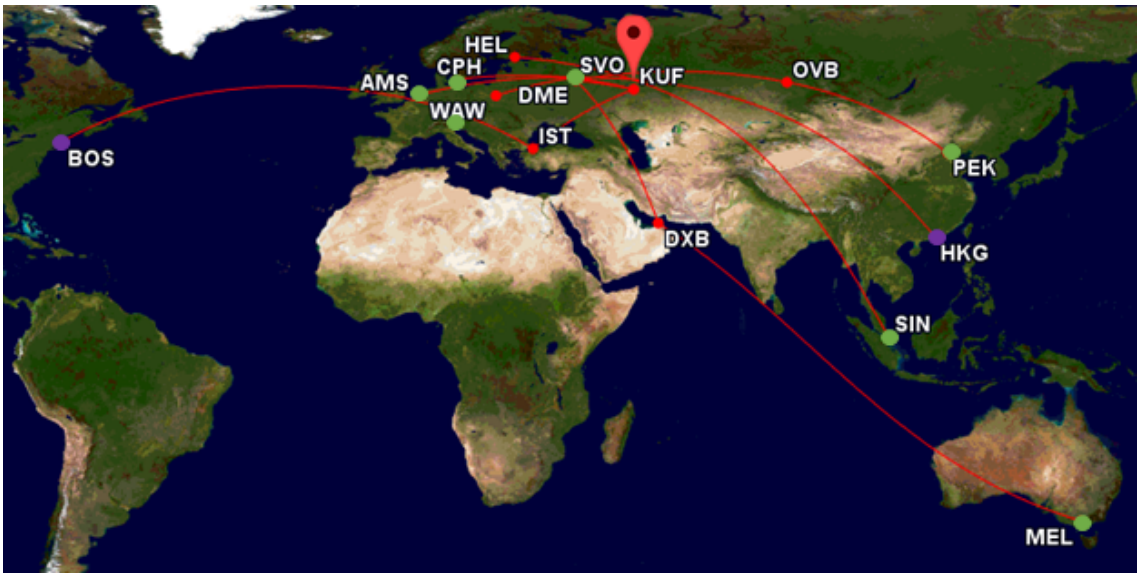

Figure 5: Map showing the flight routes (calculated with Google Flights) of each of the attendees (starting cities are marked in green, starting cities with 2 people are marked in purple, and Samara is marked with the pin) [10, 11].

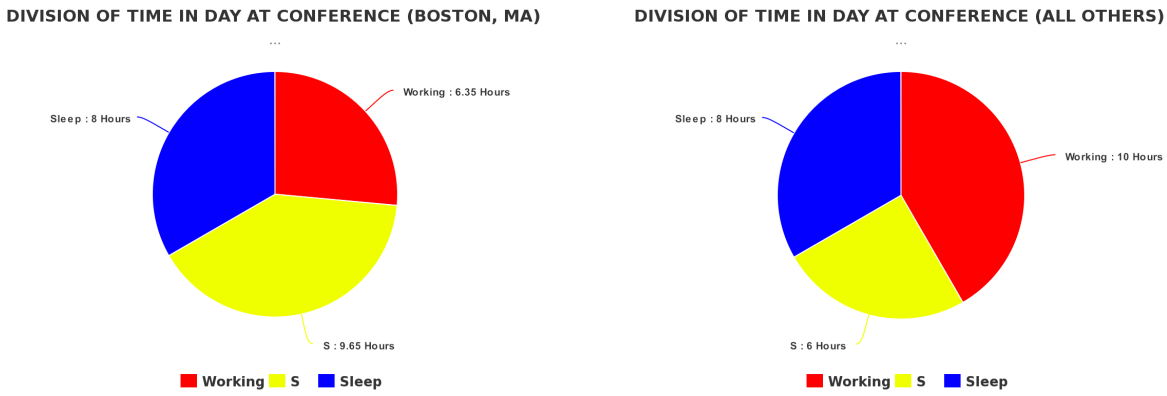

Figure 6: The Divisions of Time of Day for the members of Conference 2. [12]

This map shows that Samara is the optimal location to hold the conference. There is a considerable amount of inter-timezone travel, but this is unavoidable given the dispersed starting locations of the attendees. The three attendees having the largest timezone jumps are from Boston and Melbourne, whole the others primarily travel between the inner Asian timezones.

Model Extension: Where is the "Center of the World"?

Our model gives an ideal for a single meeting, but what about the long run? Having to constantly schedule meetings around the world is a costly endeavor that will drain funds from the IMMC. It is far easier to simply have one permanent headquarters around the world where IMMC workers can lodge and work as they please. So, where is the optimal location to place such a important meeting place?

Our model is capable of predicting a location for a global conference when the locations and number of people are known. However, IMMC may encounter situations where these factors are not

known due to large gatherings or unknown conference members during planning. In such a case, an ideal probabilistic location for a global conference can help maximize productivity without specifically knowing the locations of all members involved.

After analyzing the given data sets for the two conferences, we looked at the problem of coordinating a conference between randomly chosen set of 10 people at random locations around the world. First, we selected a random sample of 10 timezones for out attendees. Then, we ran our algorithm on this set of attendees 1000 times and produced a histogram of the most popular choices of conference location. We found that the most commonly selected optimal city was Sarajevo, Bosnia.

Though the random sample simulation was informative, it was not completely accurate. Timezones such as Vostok, Antarctica were equally likely to be selected as locations of attendees as New York City, USA. Thus, we developed a probabilistic sampling system that selected the locations of the attendees at random with a probability corresponding to the relative Gross Domestic Product (GDP) of the location (country in the surrounding area).

After testing our newly selected data with our algorithm and forming the same histogram as we had previously done, we found that Dublin, Ireland was the most commonly selected optimal city –the "Center of the World."

Sensitivity Analysis

We performed a comprehensive sensitivity analysis on our model, testing the model's robustness against changing the various parameters that determine the metrics. The results are described below.

Conference 1

Normal	-1 Outlier	-2 Outliers
Irkutsk	Thimphu	Almaty
Shanghai	Shanghai	Bishkek
Seoul	Macau	Urumqi

Table 1: The top three results from our algorithm for Conference 1 after performing outlier reduction are shown. The second column shows the results after removing Monterey, CA. The third column shows the results after removing Monterey, CA and Melbourne, AUS.

Normal	Air Fare: $d \cdot \$0.25 + \$100$	Air Fare: $d \cdot \$0.05 + \$25$
Irkutsk	Shanghai	Irkutsk
Shanghai	Irkutsk	Shanghai
Seoul	Seoul	Seoul

Table 2: The top three results from our algorithm for Conference 1 after changing the airfare are shown. The second column shows the results when the airfare is \$0.25 times the number of miles traveled + \$100, and the third column shows the results when the airfare is \$0.05 times the number of miles traveled + \$25.

Normal	Reduced Jet Lag (1 hour)
Irkutsk	Seoul
Shanghai	Irkutsk
Seoul	Shanghai

Table 3: The top three results from our algorithm for Conference 1 if the attendees' jet lag was reduced by 1 hour are shown in column 2.

6	8	10 (Normal)	12	14
Vladivostok	Seoul	Irkutsk	Irkutsk	Irkutsk
Seoul	Yakutsk	Shanghai	Shanghai	Shanghai
Irkutsk	Vladivostok	Seoul	Seoul	Seoul

Table 4: The top three results from our algorithm for Conference 1 if the maximum amount of working hours was 6, 8, 12, or 14 (instead of 10) are shown.

Normal	$\frac{1}{\kappa} + M$
Irkutsk	Shanghai
Shanghai	Seoul
Seoul	Taipei

Table 5: The top three results from our algorithm for Conference 1 using an alternate ranking based on $\frac{1}{\kappa} + M$ instead of $\frac{M}{\kappa}$. Though this ranking results in more well-known cities, it largely favors the money metric.

Conference 2

Normal	-1 Outlier	-2 Outliers	-3 Outliers
Samara	Almaty	Almaty	Almaty
Moscow	Bishkek	Urumqi	Bishkek
Volgograd	Urumqi	Thimphu	Tashkent

Table 6: The top three results from our algorithm for Conference 2 after performing outlier reduction are shown. The second column shows the results after removing 1 person from Boston, MA. The third column shows the results after removing the other person from Boston, and the fourth shows the results after removing the person from Melbourne, AUS in addition to the others.

Normal	Air Fare: $d \cdot \$0.25 + \$100$	Air Fare: $d \cdot \$0.05 + \$25$
Samara	Samara	Samara
Moscow	Moscow	Almaty
Volgograd	Volgograd	Volgograd

Table 7: The top three results from our algorithm for Conference 2 after changing the airfare. The second column shows the results when the airfare is \$0.25 times the number of miles traveled + \$100, and the third column shows the results when the airfare is \$0.05 times the number of miles traveled + \$25.

Normal	Reduced Jet Lag (1 hour)
Samara	Samara
Moscow	Moscow
Volgograd	Volgograd

Table 8: The top three results from our algorithm for Conference 2 if the attendees' jet lag was reduced by 1 hour are shown in column 2.

6	8	10 (Normal)	12	14
Samara	Moscow	Samara	Samara	Samara
Moscow	Samara	Moscow	Tashkent	Tashkent
Volgograd	Volgograd	Volgograd	Almaty	Almaty

Table 9: The top three results from our algorithm for Conference 2 if the maximum amount of working hours was 6, 8, 12, or 14 (instead of 10) are shown.

Normal	$\frac{1}{\kappa} + M$
Irkutsk	Moscow
Shanghai	Samara
Seoul	Taipei

Table 10: The top three results from our algorithm for Conference 2 using an alternate ranking based on $\frac{1}{\kappa} + M$ instead of $\frac{M}{\kappa}$.

Analysis

From the sensitivity analysis, we can see that our model is relatively insusceptible to changes in airfare, jet lag, and number working hours in the day. Conference 2 is slightly more conducive to changes in these parameters, a result of the larger amount of inter-timezone travel. Removing outliers from the set of attendees results in increases in the values of the metrics, but generally does not change the top three results of the algorithm. Finally, changing the ranking method to $\frac{1}{\kappa} + M$ does not drastically alter the results and produces more well-known cities. However, as both M and κ are greater than one, this ranking weighs the cost metric much more heavily than the productivity metric. As a result, the regular metric of $\frac{M}{\kappa}$ is preferred, because it is insensitive to drastic changes in the parameters that determine the metrics.

Strengths and Weaknesses

Strengths

- Our model is robust against changes to the parameters involved in calculating the metrics.
  - Even after changing the metric to $\frac{1}{\kappa} + M$, replacing the amount of productivity hours to a higher value, decreasing the amount of jet lag by 1 hour, or changing the airfare, the top three cities remained generally remained in the top three and always remained in the top 10.
- Our model considers both jet lag and distance traveled in our metric, allowing it to be an accurate measure of productivity.

  - While jet lag is the main cause of the loss of productivity over long flights, the fatigue caused by travel time (proportional to travel distance) is also involved. Our model accounts for both to produce a more accurate measure of productivity.
- Our model uses a comprehensive list of timezones and analyzes the metric values for each cities in a manner that is able to produce very precise values for all 282 cities.
  - By using location data from Google Maps and timezone data from live-updating databases, we ensure that our algorithm does not become deprecated.
- Our model is versatile and can calculate the optimal meeting location given any set of attendees. This versatility allowed us to calculate the "Center of the World."
  - The only inputs to our model are the initial set of members and the time of the conference, which can be arbitrary. Our code will be able to output results as long as all of the people are not from the same location (in which case the solution is trivial).
- Our model takes into account Daylight Savings Time (DST) at each timezone, allowing for more accurate jet lag calculations.
  - The timezone calculations can be done at any time of the year, and the use of live databases will ensure that DST calculations are done appropriately.
- Our model uses the Vincenty distance formula, treating the Earth like an ellipse and allowing for more accurate distance calculations.
  - The Vincenty distance formula is an iterative method to compute the distance between two points on an ellipsoid. Our model uses this formula as opposed to Great Circle distance and thus achieves more accurate results.
- Our algorithm is not computationally intensive and can be calculated for large data sets in a relatively short amount of time.
  - Our algorithm is $O(n)$, where n is the number of people.

Weaknesses

- Our model does not consider variations in the prices of lodging between different locations due to the unavailability of lodging price data. This could have caused our model to undervalue less urban locations where lodging is cheaper. However, since the model is not sensitive to large changes in airfare (which is much larger than the lodging cost), it is safe to say that lodging costs would not drastically affect the results.
- Our model does not consider the availability of flights between locations. More rural areas, which have fewer flight connections, are subsequently overvalued by our model. Overall, however, Google Flights has virtually guaranteed direct travel from each starting location to the conference location [11]
- Our model does not consider losses in productivity caused by the lower availability of high-quality working areas in rural regions of the world. Therefore our model may overestimate the practicality of some of the 282 cities for intensive work.
- Our model does not consider the loss in productivity that an individual experiences after he/she leaves the meeting location. Thus our model may be overlooking hidden implicit costs that are not covered by the flight ticket and lodging expenses because it is difficult to quantify how the individual schedules of the attendees will play out, and because the model is not sensitive to cost changes.
- Our model does not consider the effects of climate on productivity. Although climate may play a role on productivity, the effects of jet lag and fatigue have a far greater effect on productivity. Furthermore, it is difficult to quantify the effects of climate on different individuals.

Conclusion and Future Work

Our goal was to create an algorithm to determine the optimal location that maximizes the productivity of a group of individuals during a three-day conference. To accomplish this, we created a productivity metric that is computed using the fatigue that results from the jet lag of attendees. This metric employed a sinusoidal productivity function and the inverse RMS of distances traveled by the attendees. In addition, we incorporated the estimated airfare and lodging into a cost metric. Using timezone data from TimezoneDB as well as the GeoPy and PyTZ libraries, we were able to create a Python script that computed our metric for 282 populated cities around the world. For the first conference, which consisted of 6 participants, we found the optimal location to be Irkutsk, Russia. For the second conference, which consisted of 11 participants, the optimal location was Samara, Russia. To test the sensitivity of our model, we adjusted the various parameters that were involved in determining the metrics and found that overall, the model was robust against these changes.

We also considered the problem of finding the "Center of the World", which is the ideal location to hold a conference for any group of individuals with unknown locations. By running our algorithm many times on sets of individuals located around the world based on the GDP of their location, we found that Dublin, Ireland was the optimal place to hold such a conference. Though our model does not account for variations in lodging costs and the effects of climate on productivity, it is highly precise due to accurate consideration of jet lag shifts, versatility of usage (the model only requires a set of attendees and a conference time as inputs), and low computational complexity.

Given enough time, we can extend the model to include lodging price variations, air ticket availability, and climate effects on productivity through an agent based model.

Appendix

parser.R

```r
setwd("~/Desktop/IMMC/data")
rm(list=ls())

timezone <- read.csv("timezone.csv", header=F)
names(timezone) <- c("zone.id", "time.standard", "start.time", "gmt.offset", "dst
    ")
timezone <- timezone[,c("zone.id", "time.standard", "gmt.offset", "dst")]
timezone <- unique(timezone)

zone <- read.csv("zone.csv", header=F)
names(zone) <- c("zone.id", "abbreviation", "zone.name")

zonenames <- c()
for (i in 1:nrow(timezone)) {
  zonenames <- c(zonenames, as.character(zone[timezone[i,1],3]))
}

timezone["zone.names"] <- zonenames

write.csv(file="processed.timezone.csv", timezone, row.names=F, col.names=F)
```

datahandler.py

```python
import csv;
import os.path;
from geopy.geocoders import Nominatim;
from geopy.geocoders import GoogleV3;
from datetime import datetime;
from geopy.distance import vincenty;

time = datetime.strptime("Jan 1 2017", "%b %d %Y");
#time = datetime.strptime("Jun 15 2017", "%b %d %Y");

data = dict()

if not os.path.isfile("data/processed.zone.csv"):
    with  open("data/zone.csv") as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='"');
        for row in reader:
            data[int(row[0])] = dict();
            data[int(row[0])]["Zone.ID"] = int(row[0])
            data[int(row[0])]["Zone.Abbr"] = row[1]
            data[int(row[0])]["Zone.Name"] = row[2]
            data[int(row[0])]["Address"] = row[2].split("/")[1] + ", " + row[2].
               split("/")[0];

    geolocator = Nominatim()
    googleAPI = GoogleV3(api_key="REMOVED");

```

```
    for key in data:
        print(data[key]);
        try:
            addr = data[key]["Address"];
            location = geolocator.geocode(addr);
            coordinates = (location.latitude, location.longitude);
            timezone = googleAPI.timezone(coordinates, at_time=time);
            date_at_location = datetime.now(timezone);
            dJ = date_at_location.utcoffset().total_seconds() / 60 / 60;

            data[key]["Latitude"] = location.latitude;
            data[key]["Longitude"] = location.longitude;
            data[key]["Offset"] = dJ;
        except:
            continue;


    with open("data/processed.zone.csv", 'w', newline='') as csvfile:
        writer = csv.writer(csvfile, delimiter=',', quotechar='"', quoting=csv.
           QUOTE_MINIMAL)
        for key in data:
            try:
                writer.writerow([
                    data[key]["Zone.ID"],
                    data[key]["Zone.Abbr"],
                    data[key]["Zone.Name"],
                    data[key]["Address"],
                    data[key]["Latitude"],
                    data[key]["Longitude"],
                    data[key]["Offset"]
                ]);
            except:
                continue;
    with open("data/gdp.csv") as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='"');
        for row in reader:
            data[int(row[0])]["GDP"] = float(row[3]);
else:
    with  open("data/processed.zone.csv") as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='"');
        for row in reader:
            data[int(row[0])] = dict();
            data[int(row[0])]["Zone.ID"] = int(row[0])
            data[int(row[0])]["Zone.Abbr"] = row[1]
            data[int(row[0])]["Zone.Name"] = row[2]
            data[int(row[0])]["Address"] = row[3]
            data[int(row[0])]["Latitude"] = float(row[4])
            data[int(row[0])]["Longitude"] = float(row[5])
            data[int(row[0])]["Offset"] = float(row[6])
    with open("data/gdp.csv") as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='"');
        for row in reader:
            data[int(row[0])]["GDP"] = float(row[3]);

#with open("data/processed.timezone.csv", newline='') as csvfile:
#    reader = csv.reader(csvfile, delimiter=',', quotechar='"');
#    for row in reader:
```

```
#         if (row[0] == "zone.id"):
#             continue;
#         timezone[row[4]] = list();
#with open("data/processed.timezone.csv", newline='') as csvfile:
#    reader = csv.reader(csvfile, delimiter=',', quotechar='"');
#    for row in reader:
#         if (row[0] == "zone.id"):
#             continue;
#         row_add = dict();
#         row_add["Zone.ID"] = int(row[0]);
#         row_add["Time.Standard"] = row[1];
#         row_add["GMT.Offset"] = int(row[2]) / 60 / 60;
#         row_add["DST"] = bool(row[3]);
#         timezone[row[4]].append(row_add);

print("Loaded Timezone Data");
print(data);
```

algorithm.py

```
from datahandler import data as tzd;
from datahandler import time;
from geopy.geocoders import Nominatim;
from geopy.geocoders import GoogleV3;
from datetime import datetime;
from geopy.distance import vincenty;
import math;
import operator;

C_EARTH = 24901;

members = {
    0: { "Address": "Boston, Massachusetts, United States of America" },
    1: { "Address": "Boston, Massachusetts, United States of America" },
    2: { "Address": "Singapore, Republic of Singapore" },
    3: { "Address": "Beijing, China" },
    4: { "Address": "Hong Kong, China" },
    5: { "Address": "Hong Kong, China" },
    6: { "Address": "Moscow, Russia" },
    7: { "Address": "Utrecht, Netherlands" },
    8: { "Address": "Warsaw, Poland" },
    9: { "Address": "Copenhagen, Denmark" },
    10: { "Address": "Melbourne, Australia" }
};

# members = {
#     0: { "Address": "Monterey, California, United States" },
#     1: { "Address": "Zutphen, Netherlands" },
#     2: { "Address": "Melbourne, Australia" },
#     3: { "Address": "Shanghai, China" },
#     4: { "Address": "Hong Kong, China" },
#     5: { "Address": "Moscow, Russia" }
# };

```

```
locations = dict();
rank_locations = dict();

if __name__ == "__main__":

    geolocator = Nominatim()
    googleAPI = GoogleV3(api_key="REMOVED");

    for key in members:
        location = geolocator.geocode(members[key]["Address"]);
        coordinates = (location.latitude, location.longitude);
        timezone = googleAPI.timezone(coordinates, at_time=time);
        date_at_location = datetime.now(timezone);
        utc = date_at_location.utcoffset().total_seconds() / 60 / 60;
        members[key]["Latitude"] = location.latitude;
        members[key]["Longitude"] = location.longitude;
        members[key]["Offset"] = utc;

    for key in tzd:
        conf_coordinates = (float(tzd[key]["Latitude"]), float(tzd[key]["
           Longitude"]));
        pSum = 0;
        dSum = 0;
        travelCost = 0;
        for mem_key in members:
            coordinates = (float(members[mem_key]["Latitude"]), float(members[mem
               _key]["Longitude"]));
            distance = vincenty(coordinates, conf_coordinates).miles;
            dJ = members[mem_key]["Offset"] - tzd[key]["Offset"];

            P = min(10, 10 + (6 * math.cos((math.pi * dJ / 12) - (math.pi / 24)))
               );
            pSum += math.pow(P, 2);

            dSum += math.pow(distance, 2);

            travelCost += distance * 0.11 + 50;

        if dSum != 0:
            D = (C_EARTH * math.sqrt(len(members))) / (2 * math.sqrt(dSum));
        else:
            D = float("inf");
        J = math.sqrt((1 / len(members)) * pSum);
        L = 450 * len(members);
        M = travelCost + L;
        K = J * D;

        locations[tzd[key]["Zone.Name"]] = [K, M];
        rank_locations[tzd[key]["Zone.Name"]] = M / K;

    ranked = sorted(rank_locations.items(), key=operator.itemgetter(1));
    for tup in ranked:
        print(str(tup[0]) + str(locations[tup[0]]));
```

generalize.py

```
from datahandler import data as tzd;
from datahandler import time;
from geopy.geocoders import Nominatim;
from geopy.geocoders import GoogleV3;
from datetime import datetime;
from geopy.distance import vincenty;
from geopy.distance import great_circle;
import math;
import operator;
import random;
import csv;

C_EARTH = 24901;

# members = {
#     0: { "Address": "Boston, Massachusetts, United States of America" },
#     1: { "Address": "Singapore, Republic of Singapore" },
#     2: { "Address": "Beijing, China" },
#     3: { "Address": "Hong Kong, China" },
#     4: { "Address": "Moscow, Russia" },
#     5: { "Address": "Utrecht, Netherlands" },
#     6: { "Address": "Warsaw, Poland" },
#     7: { "Address": "Copenhagen, Denmark" },
#     8: { "Address": "Melbourne, Australia" }
# };

members = {
    0: { "Address": "Monterey, California, United States" },
    1: { "Address": "Zutphen, Netherlands" },
    2: { "Address": "Melbourne, Australia" },
    3: { "Address": "Shanghai, China" },
    4: { "Address": "Hong Kong, China" },
    5: { "Address": "Moscow, Russia" }
};

histogram_data = dict();

def generate(members):

    locations = dict();
    rank_locations = dict();

    geolocator = Nominatim()
    googleAPI = GoogleV3(api_key="REMOVED");

    #for key in members:
    #    location = geolocator.geocode(members[key]["Address"]);
    #    coordinates = (location.latitude, location.longitude);
    #    timezone = googleAPI.timezone(coordinates, at_time=time);
    #    date_at_location = datetime.now(timezone);
    #    utc = date_at_location.utcoffset().total_seconds() / 60 / 60;
    #    members[key]["Latitude"] = location.latitude;
    #    members[key]["Longitude"] = location.longitude;
    #    members[key]["Offset"] = utc;

```

```
    for key in tzd:
        conf_coordinates = (float(tzd[key]["Latitude"]), float(tzd[key]["
            Longitude"]));
        pSum = 0;
        dSum = 0;
        travelCost = 0;
        for mem_key in members:
            coordinates = (float(members[mem_key]["Latitude"]), float(members[mem
                _key]["Longitude"]));
            distance = vincenty(coordinates, conf_coordinates, iterations=100).
                miles;
            dJ = members[mem_key]["Offset"] - tzd[key]["Offset"];

            P = min(10, 10 + (6 * math.cos((math.pi * dJ / 12) - (math.pi / 24)))
                );
            pSum += math.pow(P, 2);

            dSum += math.pow(distance, 2);

            travelCost += distance * 0.11 + 50;

        D = (C_EARTH * math.sqrt(len(members))) / (2 * math.sqrt(dSum));
        J = math.sqrt((1 / len(members)) * pSum);
        L = 450 * len(members);
        M = travelCost + L;
        K = J * D;

        locations[tzd[key]["Zone.Name"]] = [K, M];
        rank_locations[tzd[key]["Zone.Name"]] = M / K;

    ranked = sorted(rank_locations.items(), key=operator.itemgetter(1));
    for k in range(len(ranked)):
        histogram_data[ranked[k][0]] += k + 1;

if __name__ == "__main__":
    for key in tzd:
        histogram_data[tzd[key]["Zone.Name"]] = 0;

    GDPSum = 0;
    weights = [];
    for key in tzd:
        GDPSum += tzd[key]["GDP"];
    GDPWeights = 0;
    for key in tzd:
        GDPWeights += tzd[key]["GDP"];
        weights.append(GDPWeights / GDPSum);

    for i in range(1000):
        members = dict();
        for k in range(10):
            rnd = random.random();
            for j in range(len(weights) - 1):
                if rnd > weights[j] and rnd < weights[j+1]:
                    mmbr = list(tzd.values())[j];
                    members[k] = dict();
                    members[k]["Address"] = mmbr["Address"];
                    members[k]["Latitude"] = mmbr["Latitude"];
```

```
                    members[k]["Longitude"] = mmbr["Longitude"];
                    members[k]["Offset"] = mmbr["Offset"];
        generate(members);
        print("Iteration: %s, %s" % (i, members));
        pass;

    # iters = 1000;
    # i = 0;
    # while i < iters:
    #     try:
    #         members = dict();
    #         for k in range(10):
    #             id = random.randint(0, 281);
    #             members[k] = dict();
    #             members[k]["Address"] = list(tzd.values())[id]["Address"]
    #             members[k]["Latitude"] = list(tzd.values())[id]["Latitude"]
    #             members[k]["Longitude"] = list(tzd.values())[id]["Longitude"]
    #             members[k]["Offset"] = list(tzd.values())[id]["Offset"]
    #         generate(members);
    #         i += 1;
    #         print("Iteration: %s, Iterations: %s" % (i, iters));
    #     except KeyboardInterrupt:
    #         exit();
    #     #except:
    #     #    print("Iteration: %s, Iterations: %s" % (i, iters));
    #     #    continue;

    with open("data/histogram.csv", "w", newline='') as csvfile:
        writer = csv.writer(csvfile, delimiter=',', quotechar='"', quoting=csv.
            QUOTE_MINIMAL)
        for key in histogram_data:
            writer.writerow([key, histogram_data[key]]);
```

References

[1] Christian Nordqvist. *Jet Lag: What It Is and How to Beat It*. URL: http://www.medicalnewstoday.com/articles/165339.php.

[2] SIAM News. *Explaining the East/West Asymmetry of Jet Lag*. URL: https://sinews.siam.org/Details-Page/explaining-the-eastwest-asymmetry-of-jet-lag.

[3] *Airline fare analysis: comparing cost per mile*. 2013. URL: https://www.rome2rio.com/blog/2013/01/02/170779446/.

[4] *The Hotels.com Hotel Price Index*. URL: http://gb.hotels.com/hotel-price-index/7-average-room-prices-by-star-rating.html.

[5] *Free Time Zone Database & API*. URL: https://timezonedb.com/.

[6] *Geopy*. URL: https://pypi.python.org/pypi/geopy.

[7] *Pytz*. URL: https://pypi.python.org/pypi/pytz.

[8] *World Economic Outlook Database October 2016*. URL: http://www.imf.org/external/pubs/ft/weo/2016/02/weodata/index.aspx.

[9] Lonely Planet. *Irkutsk*. URL: https://www.lonelyplanet.com/russia/siberia/irkutsk.

[10] *Great Circle Mapper*. URL: http://www.gcmap.com/.

[11] *Google Flights*. URL: http://www.flights.google.com.

[12] *Graphing/Charting and General Data Visualization App*. URL: https://www.meta-chart.com/.

[13] *Samara city, Russia*. URL: http://russiatrek.org/samara-city.