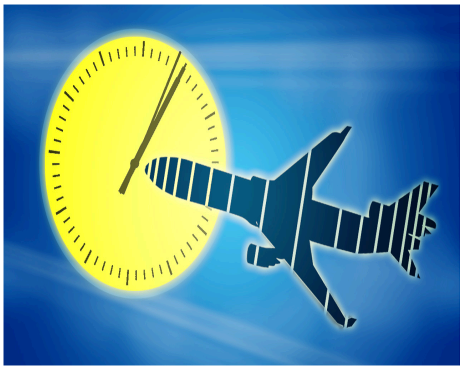
International Meeting Management Corporation Report



An algorithm to determine the best places to hold a meeting reducing the most the participant´s jetlag

Index of Contents

Executive Summary

It is a fact: in each and every day of the year, a different conference is held in a different part of the world. Effectively, these meetings gather people from all around the world, something that translates into a lot of travel. And inevitably, most of the people attending will suffer a certain degree of "jetlag". Incentivized by the International Meeting Management Corporation problem, we came up with a project idea: to develop an algorithm or method that could find a place for a meeting with the least amount of jetlag possible. Hence, we investigated about jetlag and all of the causes and factors that could potentially affect it.

We found out that some of the key factors that affected jetlag were distance travelled, number of time zones alternated, difference in altitude/elevation of the place of origin with respect to the place travelled, difference in daylight time between the origin and destination, among others.

When we started working, we noticed that the best way to get a city that would meet the criteria for a the "best meeting location" (reduces the most the jetlag in average for all of the participants) was by having a database with a lot of cities around the world with which we could compare the case study (the cities from which the participants would eventually departure from). This is why we created a database in Excel with fifty-six cities around the world (representative of all of the regions) and imputed data to each one of them, using the variables mentioned above. This was accompanied with a smaller database, the input, for the case study.

Essentially, the main algorithm we created is divided in four large parts that compares our four main variables between both databases: latitude/longitude, time zones, altitude, and daylight. For each variable category, it returns a series of cities from the larger database that would be the recommended ones for that specific case study. The more the cities overlap between the categories, the better. We measured this creating as well a "jetlag index", where the closer to 100, the least jetlag.

An important feature about the database and our algorithm is that it is completely flexible. This means that we could change any data (like cities, month of the year, coordinates, amount of people from each city) and would instantly change all the information and values connected to it.

We were asked to test our algorithm in different scenarios. For instance, in the "small meeting" case study, our algorithm returned that Tokyo, Japan and Beijing, China had the highest "jetlag index", making them the most suitable cities and regions for the meeting. For the "large meeting" case study, it gave that Istanbul, Turkey was the most suitable city/region to held the meeting at.

Introduction

This new era of globalization and worldwide connections has brought us wonderful things. TV shows, clothing brands, food brands, worldwide entertainment, 24-hour news and constant discoveries. Now, we are able to communicate faster with people across the globe, allowing us to efficiently exchange ideas. As intertwining ideas and sharing thoughts has become a quite a natural thing, it has caused one, not severe, but solvable problem: jetlag.

When we travel throughout the globe, and especially through time zones, our circadian rhythms (more commonly known as our "body clock"), the 24-hour cycle in the biological, physiological and behavioural processes of our bodies became unsynchronized with the time zone we could be in. Basically, this has disabled our organisms to reset themselves to their current time zones, and is the product of the increased intercontinental travel.

Today, the International Meeting Management Corporation has presented us the following problem: to find a place for a conference that gathers people from each and every corner of the globe, whilst reducing the most the average amount of jetlag for each participant. Hence, we have thought in creating an algorithm, that could account for the main factors that affect jetlag, which we consider are longitude/latitude, time zone, daylight hours, and altitude. This would permit us to input any location, and return the perfect spot to reduce both jetlag, as well as time and total distance travelled for everybody on the conference. This will aim as well to reduce the total costs for the conference.

Globalization is never a setback. Even if we think it may produce some problems, we believe the advantages outweigh the cost, helping us to communicate with people we wouldn't have been able to before. This is the true meaning of progress. And we think this that by using the algorithm we came up with, we would eventually be able to soften the impact of jetlag, and make globalization even better.

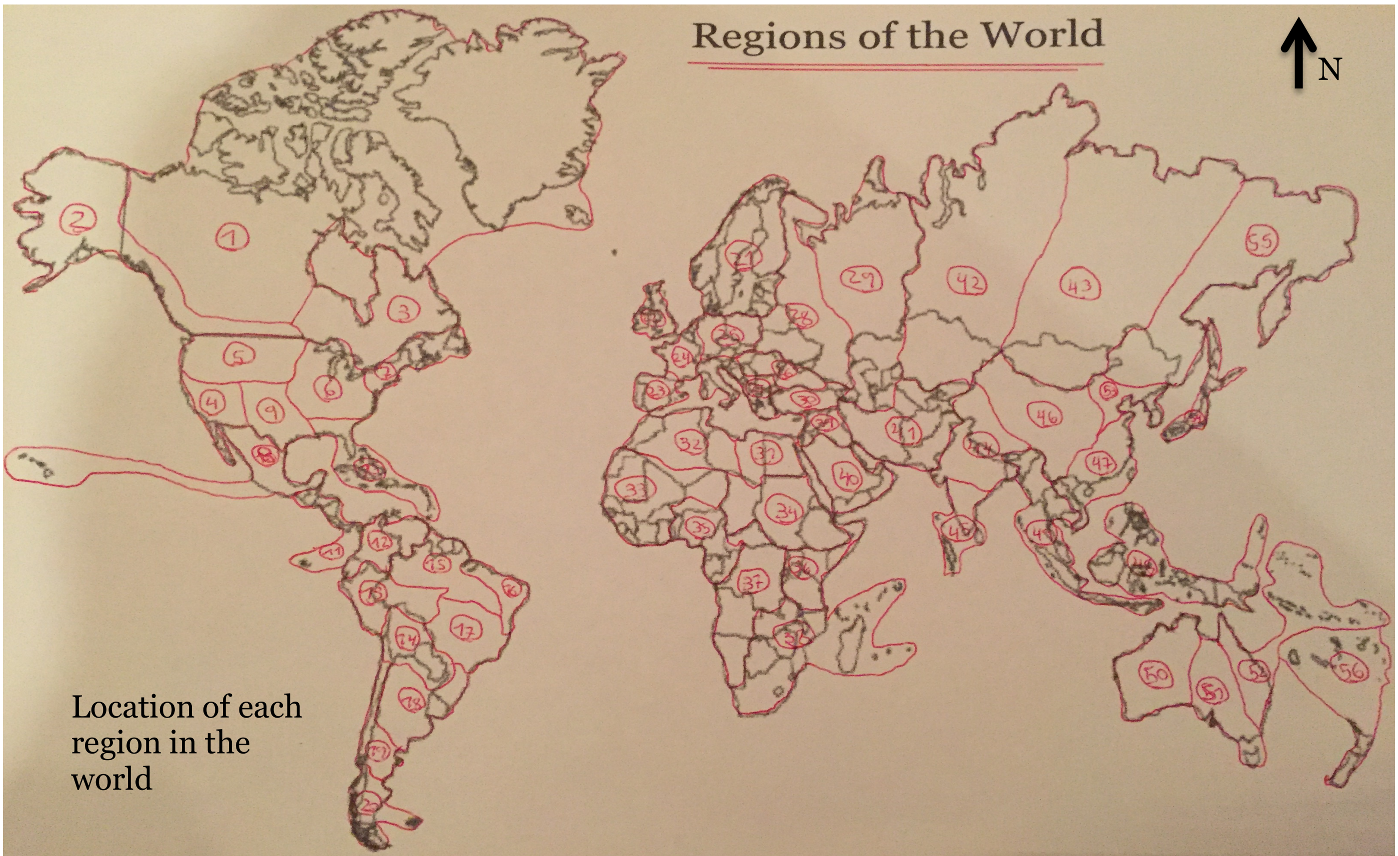
What is jetlag and why is it important?

We learned that "desynchronization" or jetlag is a disorder that we get when we travel. As we go through different time zones, our bodies lose their natural cycle and we get desynchronized. Naturally, this temporary disorder brings several symptoms, like fatigue and insomnia. This happens because our body is still synchronized with the hour of the place we departure from and looks forwards to eat or sleep, again, at this same time, rather than the time of the place arriving in. Gratefully, jetlag is temporary, so most of the people will recover within few days. These symptoms depend in the amount of time zones crossed, time of day and direction travelled. Here are a couple of them: sleeping disturbance[1], indigestion, constipation, diarrhea, nausea, loss of appetite or hunger, disorientation and anxiety, among others.

<u>Assumptions</u>

These are the assumptions we made to find the algorithm:

1) The meeting has to take place in a big city or in the capital of a country (this enables easy air travel access, reducing costs and travel times)
2) We can divide the world into 56 regions, representative of all the corners of the globe. Also, a large city could be chosen from each one of these, that would be representative of that particular region.
3) If there were more people from one country or from the same region (more weight is added to that city), then the city chosen by the algorithm would be closer to them because it would minimize jetlag.
4) The amount of maximum day daylight and minimum daylight in a city could be modelled using a sine curve.
5) The participants would fly in business/first class, something that would enable them to rest while on road to the conference.

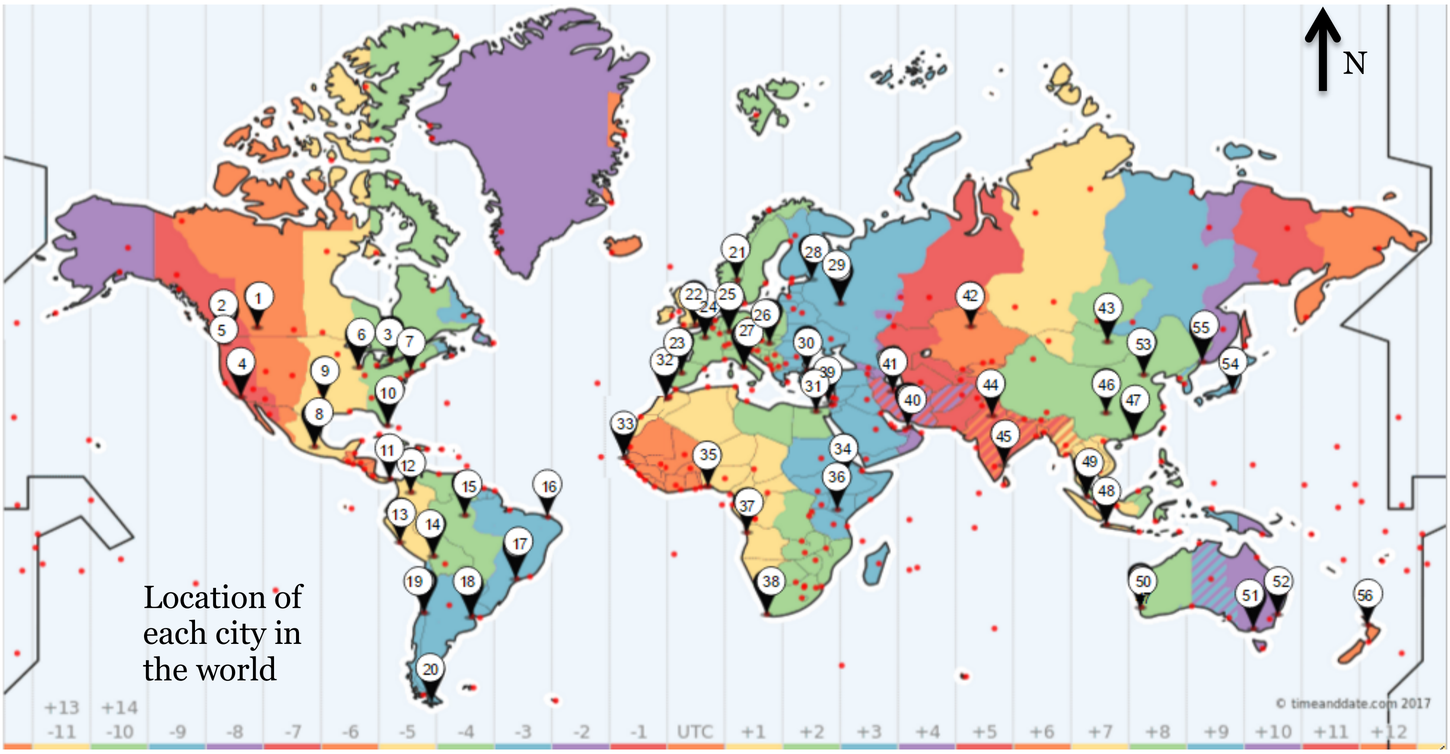


Location of each region in the world

[1] By traveling west, we gain hours, so our day is longer, leading to more hours than what our body naturally wants. If there is too much of a difference, we might might fall asleep. On the other hand, by traveling east, we lose hours, so
might might fall asleep. On the other hand, by traveling east, we lose hours, so
our day is shortened and our body has to stop it's natural cycle to sleep earlier, affecting us even more.

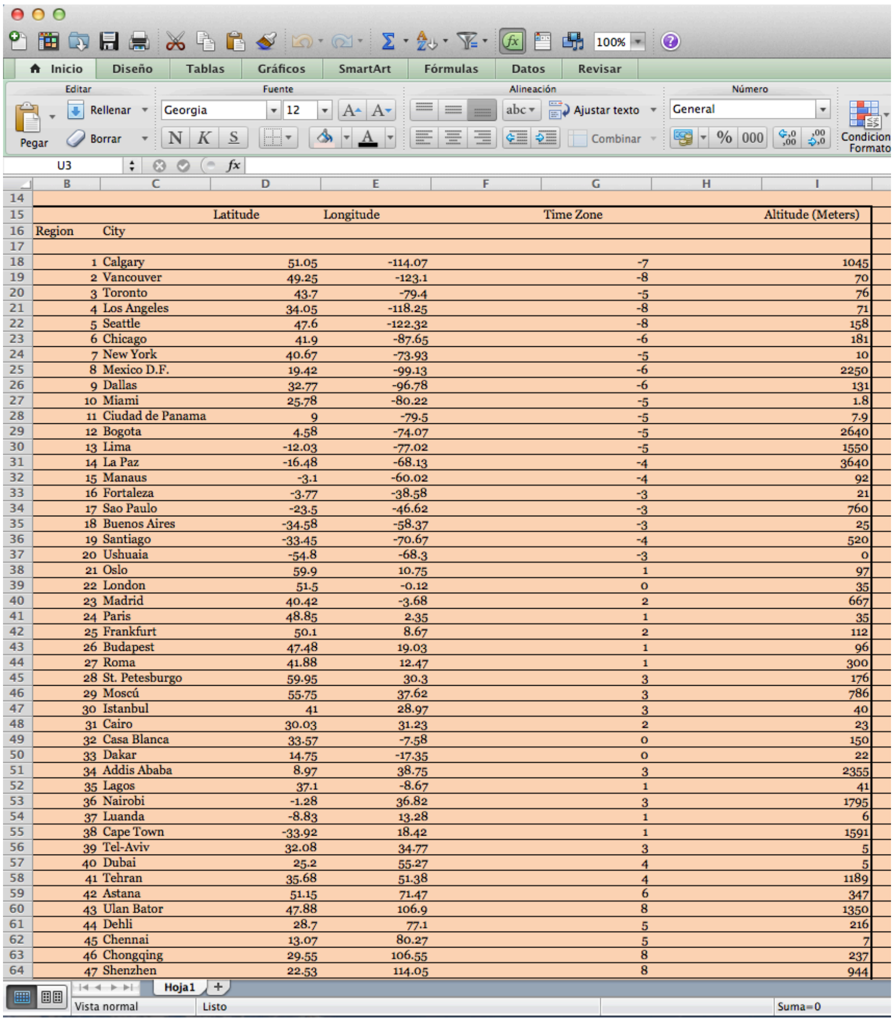
The following table lists the regions of the world, and their corresponding cities:

Region	Corresponding City	Region	Corresponding City
1	Calgary	29	Moscow
2	Vancouver	30	Istanbul
3	Toronto	31	Cairo
4	Los Angeles	32	Casablanca
5	Seattle	33	Dakar
6	Chicago	34	Addis Ababa
7	New York	35	Lagos
8	Mexico D.F.	36	Nairobi
9	Dallas	37	Luanda
10	Miami	38	Cape Town
11	Panama City	39	Tel Aviv
12	Bogota	40	Dubai
13	Lima	41	Tehran
14	La Paz	42	Astana
15	Manaus	43	Ulan Bator
16	Fortaleza	44	Delhi
17	Sao Paulo	45	Chennai
18	Buenos Aires	46	Chongqing
19	Santiago de Chile	47	Shenzhen
20	Ushuaia	48	Kuala Lumpur
21	Oslo	49	Jakarta
22	London	50	Perth
23	Madrid	51	Melbourne
24	Paris	52	Sydney
25	Frankfurt	53	Beijing
26	Budapest	54	Tokyo
27	Rome	55	Vladivostok
28	St. Petersburg	56	Auckland



Location of each city in the world

The algorithm



Region	City	Latitude	Longitude	Time Zone	Altitude (Meters)
1	Calgary	51.05	-114.07	-7	1045
2	Vancouver	49.25	-123.1	-8	70
3	Toronto	43.7	-79.4	-5	76
4	Los Angeles	34.05	-118.25	-8	71
5	Seattle	47.6	-122.32	-8	158
6	Chicago	41.9	-87.65	-6	181
7	New York	40.67	-73.93	-5	10
8	Mexico D.F.	19.42	-99.13	-6	2250
9	Dallas	32.77	-96.78	-6	131
10	Miami	25.78	-80.22	-5	1.8
11	Ciudad de Panama	9	-79.5	-5	7.9
12	Bogota	4.58	-74.07	-5	2640
13	Lima	-12.03	-77.02	-5	1550
14	La Paz	-16.48	-68.13	-4	3640
15	Manaus	-3.1	-60.02	-4	92
16	Fortaleza	-3.77	-38.58	-3	21
17	Sao Paulo	-23.5	-46.62	-3	760
18	Buenos Aires	-34.58	-58.37	-3	25
19	Santiago	-33.45	-70.67	-4	520
20	Ushuaia	-54.8	-68.3	-3	0
21	Oslo	59.9	10.75	1	97
22	London	51.5	-0.12	0	35
23	Madrid	40.42	-3.68	2	667
24	Paris	48.85	2.35	1	35
25	Frankfurt	50.1	8.67	2	112
26	Budapest	47.48	19.03	1	96
27	Roma	41.88	12.47	1	300
28	St. Petesburgo	59.95	30.3	3	176
29	Moscú	55.75	37.62	3	786
30	Istanbul	41	28.97	3	40
31	Cairo	30.03	31.23	2	23
32	Casa Blanca	33.57	-7.58	0	150
33	Dakar	14.75	-17.35	0	22
34	Addis Ababa	8.97	38.75	3	2355
35	Lagos	37.1	-8.67	1	41
36	Nairobi	-1.28	36.82	3	1795
37	Luanda	-8.83	13.28	1	6
38	Cape Town	-33.92	18.42	1	1591
39	Tel-Aviv	32.08	34.77	3	5
40	Dubai	25.2	55.27	4	5
41	Tehran	35.68	51.38	4	1189
42	Astana	51.15	71.47	6	347
43	Ulan Bator	47.88	106.9	8	1350
44	Dehli	28.7	77.1	5	216
45	Chennai	13.07	80.27	5	7
46	Chongqing	29.55	106.55	8	237
47	Shenzhen	22.53	114.05	8	944

The Database of cities

The algorithm we computed analyses four main variables that affect jetlag. These are latitudes and longitudes, time zones, altitude and daylight hours. Thus, we decided to divide the algorithm into four smaller parts that would analyse each one of these separately.

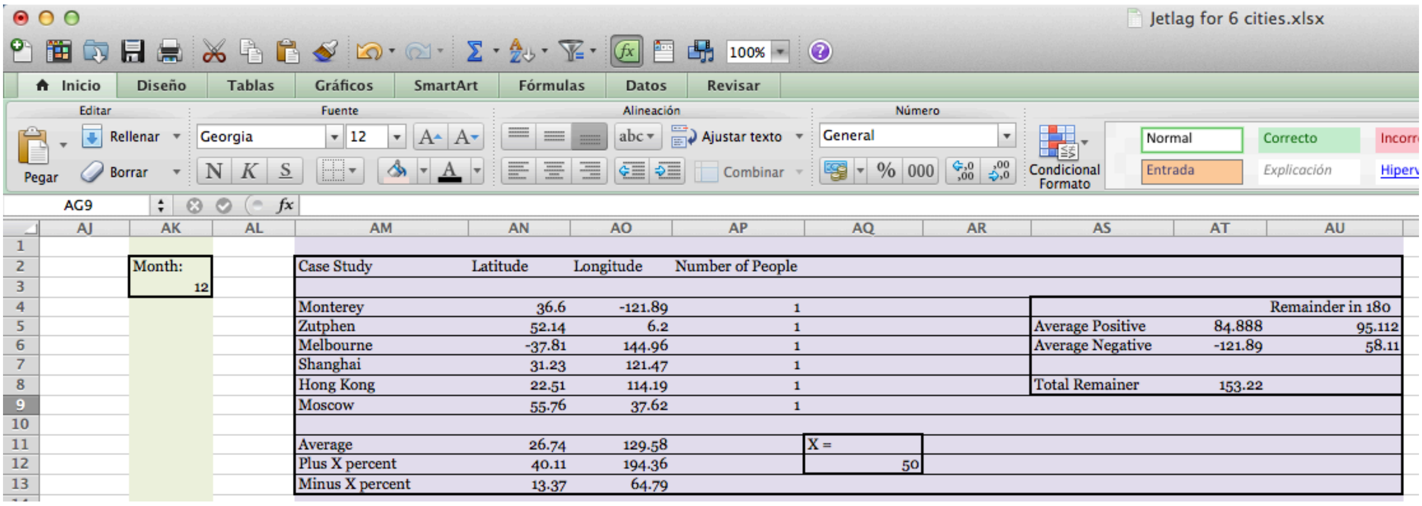
The first thing we did to begin with our algorithm was completing our database. The picture to the left shows a portion of the database, with the converted latitude/longitudes (from GPS coordinates into the decimal scale), time zone and altitude (elevation above sea level in meters) for each one of the cities we chose above. We also used "timeanddate.com" website software to find the month where these cities had their minimum and maximum daylight hours, and using those values (which we converted again into our decimal scale), we determined that these could be modelled with a sine curve, dependant on the month of the year. The formula for sine modelling is:

$$A\sin\left(\frac{2\pi x}{B} + \frac{2\pi}{C}\right) + D$$

Where A is the amplitude (the differences between the maximum and minimum over 2), B is the desired period (12 for us), $2\pi/C$ is the shift to the left (depending on the month, it gave us $+2\pi/4$ for peaks in December, or $-2\pi/4$ for peaks in June), and D is the shift upwards. Please feel free to see our excel formula in the Formula Appendix below. As seen, by changing the month cell above, the whole sine formula would return a different value (this is important because it gave our algorithm flexibility by adapting to different times of the year).

We only show part of the database, because most of it is just calculations and values we had to input (these are the results).

After doing this, it was time to build the input database, for the case study.



Month:	Case Study	Latitude	Longitude	Number of People				
12								
	Monterey	36.6	-121.89	1				Remainder in 180
	Zutphen	52.14	6.2	1		Average Positive	84.888	95.112
	Melbourne	-37.81	144.96	1		Average Negative	-121.89	58.11
	Shanghai	31.23	121.47	1				
	Hong Kong	22.51	114.19	1		Total Remainer	153.22	
	Moscow	55.76	37.62	1				
	Average	26.74	129.58		X =			
	Plus X percent	40.11	194.36		50			
	Minus X percent	13.37	64.79					

Case study input of "mini" algorithm number 1

The picture above is the first "mini" algorithm, for latitude and longitude. Hence, we filled it up, with the converted latitudes/longitudes (decimal scale) and the number of people that was coming from each city. To make the averages, for instance, for latitude, we made the algorithm multiply the number of people of each city by its respective latitude, add all of the values, and divide by the number of people (like in cumulative frequency). But then we encountered a problem with the longitudes. For instance, if we input a city in far Russia with longitude 170, and one in far USA, with longitude -160, the average will be 5, which is in the other corner of the world. Effectively, the average should be -175, which is closer to both points. So we made the algorithm make an average of all positive longitudes, and another average for the negative longitudes (positive is from Greenwich to the east, while negative is to the west). Then, we took the modulus (positive form) by subtracting them from 180, and then we added these values (this would give us the average separation in degrees, but across the pacific). Thus, if this value were more than 180, we would continue using the regular average. Nonetheless, if this value were less than 180, then we would take the average of all of the longitudes as before, but would add to them 180 degrees, to give the real effective average. Having done this, we computed a variable "X" that would establish the percentage parameters from the calculated average. For instance, if X is 30%, then the algorithm would return the average latitude/longitude plus its 30%, or minus its 30%. For simplicity, we established X as 50%, though the algorithm allows us to input any value. Next, we defined a function that basically said that if the latitude and longitude of the cities of the large database both where inside the defined parameters respectively, then we would assign them a "Yes", otherwise "No". We converted each "Yes" into a 1, and each "No" into a 0. This would enable the algorithm to add up all of the 1s, and account for the number of potential regions for the meeting. Then we defined another value called "Expectation", which we defined as "Yes". This means that the algorithm will look up in the list of all the "yes and no" which match this expectation. If they do match, then it would lookup in the large database the name of the city, its respective latitude and longitude, and its representative region of the world. This list shows the number of cities that are recommended for the meeting, following the flexible parameters established before.

			Potential Cities	5	Expectation: Yes	
City	Yes/No	1/0				
				Latitude	Longitude	Region
Calgary	No	0	Dehli	28.7	77.1	44
Vancouver	No	0	Chongqing	29.55	106.55	46
Toronto	No	0	Shenzhen	22.53	114.05	47
Los Angeles	No	0	Beijing	39.9	116.4	53
Seattle	No	0	Tokyo	35.68	139.68	54
Chicago	No	0	#¡REF!			
New York	No	0				
Mexico D.F.	No	0				
Dallas	No	0				
Miami	No	0				
Ciudad de Panama	No	0				
Bogota	No	0				
Lima	No	0				
La Paz	No	0				
Manaus	No	0				
Fortaleza	No	0				
Sao Paulo	No	0				
Buenos Aires	No	0				
Santiago	No	0				
Ushuaia	No	0				
Oslo	No	0				
London	No	0				
Madrid	No	0				
Paris	No	0				
Frankfurt	No	0				
Budapest	No	0				
Roma	No	0				
St. Petesburgo	No	0				
Moscú	No	0				
Istanbul	No	0				
Cairo	No	0				
Casa Blanca	No	0				
Dakar	No	0				
Addis Ababa	No	0				

Suma=0

Results for "mini" algorithm number 1

Then we began with the second "mini" algorithm, where we continued to fill up the small case study input database, but this time with the minimum and maximum daylight. The algorithm found the difference between each other, and then divided by 2 to obtain the amplitude of the sine curve. The column "equation" returns the respective daylight hours, using the computed sine curve, as well as the "month" variable defined before, of which all daylight equations in the database depend upon. To calculate the average, made the algorithm again multiply the number of people of each city by its respective calculated daylight, add all of the values, and divide by the number of people. We defined again another “X” variable, to set the parameters from the mean daylight time. In this case we set X to be 30%. Then, the algorithm tested again for the cities in the database that had a daylight value between these parameters, returned a “Yes” in a positive case, or “No” in negative case. We converted again the positives in 1 and negatives into 0, and added all of the 1 to find the number of potential cities.

Then, the algorithm displayed once again the list of cities, with their respective values and regions.

Minimum Daylight	Month	Maximum	Month	Difference	Over 2	Equation	Number of Peopl
9.72	12	14.63	6	4.91	2.46	9.72	
7.82	12	16.75	6	8.93	4.47	7.82	
9.33	6	14.60	12	5.27	2.64	14.60	
10.15	12	14.17	6	4.02	2.01	10.15	
10.78	12	13.50	6	2.72	1.36	10.78	
7.15	12	17.55	6	10.40	5.20	7.15	
X=					Average	10.04	
30					Plus X percent	13.05	
					Minus X percent	7.03	

City	Yes/No	1/0
Calgary	Yes	1
Vancouver	Yes	1
Toronto	Yes	1
Los Angeles	Yes	1
Seattle	Yes	1
Chicago	Yes	1
New York	Yes	1
Mexico D.F.	Yes	1
Dallas	Yes	1
Miami	Yes	1
Ciudad de Panama	Yes	1
Bogota	Yes	1
Lima	Yes	1
La Paz	No	0
Manaus	Yes	1
Fortaleza	Yes	1
Sao Paulo	No	0
Buenos Aires	No	0
Santiago	No	0
Ushuaia	No	0
Oslo	No	0
London	Yes	1
Madrid	Yes	1
Paris	Yes	1
Frankfurt	Yes	1
Budapest	Yes	1
Roma	Yes	1
St. Petesburgo	No	0
Moscú	No	0
Istanbul	Yes	1
Cairo	Yes	1

Potential Cities	44	Expectation:
	Daylight	Region
Fortaleza	12.33	16
London	7.90	22
Madrid	9.28	23
Paris	8.23	24
Frankfurt	8.13	25
Budapest	8.43	26
Roma	9.18	27
Istanbul	9.22	30
Cairo	10.20	31
Casa Blanca	9.92	32
Dakar	11.27	33
Addis Ababa	11.60	34
Lagos	11.75	35
Nairobi	12.18	36
Luanda	12.63	37
Tel-Aviv	10.10	39
Dubai	10.57	40
Tehran	9.73	41
Astana	7.98	42
Ulan Bator	8.38	43
Dehli	10.32	44
Chennai	11.35	45
Chongqing	10.28	46
Shenzhen	10.75	47
Jakarta	12.37	48
Kuala Lumpur	11.93	49
Perth	10.05	50
Beijing	9.33	53
Tokyo	9.78	54
Vadivostok	9.00	55
#¡REF!	#¡REF!	#¡REF!

Case study input and results of “mini” algorithm number 2

Following the second came the third “mini” algorithm, for the time zones. The time zones we used where all of them with reference to the UTC (Coordinated Universal Times), where negative means to the west, and positive to the east, of the 0 time zone in the Greenwich meridian zone. While averaging the time zones (by multiplying the number of people by the respective time, add all of them, and then divide by total amount of people), we encountered a similar problem than before with the latitudes and longitudes. Effectively, time zones are a representation of the globe in 2 dimensions, while the globe has 3, making it difficult to account the limits. For instance, if the case study is a city in Alaska with UTC –9 and a city in Russia with UTC + 10, then the average would be 0.5, which is far from these, in the opposite end of the world. The real average is actually -11.5, which is closer to them. Thus, we made the algorithm make an average of the positive time zones and the negative ones, and subtract them from 12, while taking the modulus (positive form). We added these values, and obtained an averaged difference in time zones, but across the pacific. If this value were larger than 12, then the algorithm would continue using the previous average. Nonetheless, if this were smaller than 12, then the algorithm would subtract 12 (if the value surpasses -12, the algorithm would convert the difference into a positive number, and then subtract it from 12). Once again, our algorithm repeated the same process, finding the list of cities that matched the parameters established by another “X” variable, which we defined as 40%.

Time Zone	Number of People
-8	1
2	1
10	1
8	1
8	1
3	1

		Remainder in 12
Average Positive	6.2	5.8
Average Negative	-8	4
Total Remainer	9.8	

X=		
40	Real Average	9.83
	Plus X percent	13.77
	Minus X percent	5.90

City	Yes/No	1/0	Potential Cities	13	Expectation: Yes
				Time Zone	Region
Calgary	No	0	Astana	6.00	42
Vancouver	No	0	Ulan Bator	8.00	43
Toronto	No	0	Chongqing	8.00	46
Los Angeles	No	0	Shenzhen	8.00	47
Seattle	No	0	Jakarta	7.00	48
Chicago	No	0	Kuala Lumpur	7.00	49
New York	No	0	Perth	8.00	50
Mexico D.F.	No	0	Melbourne	10.00	51
Dallas	No	0	Sydney	10.00	52
Miami	No	0	Beijing	8.00	53
Ciudad de Panama	No	0	Tokyo	9.00	54
Bogota	No	0	Vadivostok	10.00	55
Lima	No	0	Auckland	12.00	56
La Paz	No	0	#¡REF!		
Manaus	No	0			
Fortaleza	No	0			
Sao Paulo	No	0			
Buenos Aires	No	0			
Santiago	No	0			
Ushuaia	No	0			
Oslo	No	0			
London	No	0			
Madrid	No	0			
Paris	No	0			
Frankfurt	No	0			
Budapest	No	0			
Roma	No	0			
St. Petesburgo	No	0			
Moscú	No	0			

Case study input and results of “mini” algorithm number 3

Finally, we computed the forth a "mini" algorithm, which would compare the average elevation (above sea level in meters) between the cities in the database and the case study. Following the procedures established before, we created another "X" variable at 40% to establish the parameters, and made the algorithm return the list of cities that were inside this cut.

Altitude (Meters)	Number of People
8	1
10	1
31	1
4	1
4	1
151	1

X=		
	Average	34.67
40	Plus X percent	48.53
	Minus X percent	20.80

City	Yes/No	1/0
Calgary	No	0
Vancouver	No	0
Toronto	No	0
Los Angeles	No	0
Seattle	No	0
Chicago	No	0
New York	No	0
Mexico D.F.	No	0
Dallas	No	0
Miami	No	0
Ciudad de Panama	No	0
Bogota	No	0
Lima	No	0
La Paz	No	0
Manaus	No	0
Fortaleza	Yes	1
Sao Paulo	No	0
Buenos Aires	Yes	1
Santiago	No	0
Ushuaia	No	0
Oslo	No	0
London	Yes	1
Madrid	No	0
Paris	Yes	1
Frankfurt	No	0
Budapest	No	0
Roma	No	0
St. Petesburgo	No	0
Moscú	No	0
Istanbul	Yes	1
Cairo	Yes	1
Casa Blanca	No	0
Dakar	Yes	1
Addis Ababa	No	0

Potential Cities	11	Expectation: Yes
	Altitude	Region
Fortaleza	21.00	16
Buenos Aires	25.00	18
London	35.00	22
Paris	35.00	24
Istanbul	40.00	30
Cairo	23.00	31
Dakar	22.00	33
Lagos	41.00	35
Perth	32.00	50
Beijing	44.00	53
Tokyo	37.00	54
#¡REF!		

Listo Suma=0

Case study input and results of "mini" algorithm number 4

What remained after all this analysis was to compare the list of cities that each "mini" algorithm returned. To do this, we defined a weight for each category. This means that if a city were inside of the list returned, then it would have something similar to a score. For this case, the weights we defined where 25 points for each category. In the end, the ability to change the weights of the algorithm is what adds more flexibility to it.

Thus, we created one last database that for each category looked up which city of the large database was inside of the returned list. In the positive case, it would assign the already established weight. In negative case, it would assign a 0. Hence, for each category for each city, we made the algorithm add each one of the weights, to obtain what we defined as the "Jetlag Index". This final value ranges from 0 to 100, and easily enough, the closer to 100, the better the city to held a meeting.

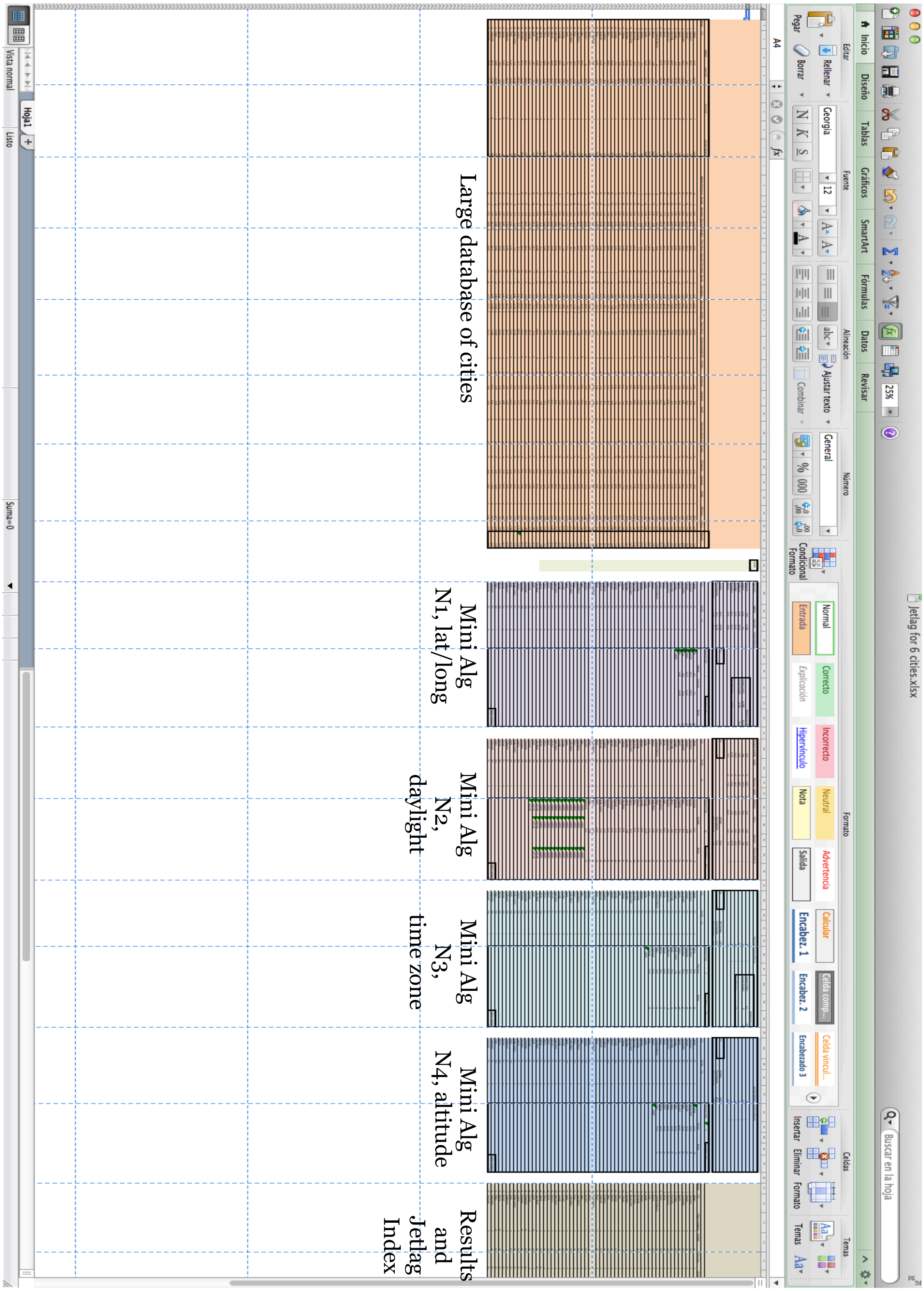
We then made the algorithm order the Total Jetlag Indeces from largest to smallest, as seen in the picture, in its right end. From the database of cities and regions around the world, this small table summarizes which cities are the best to hold a meeting at, for a particular case study.

Region	City	Weight Coordinates	Daylight	Time Zone	Altitude	Total Jetlag Index
1	Calgary	0	25	0	0	25
2	Vancouver	0	25	0	0	25
3	Toronto	0	25	0	0	25
4	Los Angeles	0	25	0	0	25
5	Seattle	0	25	0	0	25
6	Chicago	0	25	0	0	25
7	New York	0	25	0	0	25
8	Mexico D.F.	0	25	0	0	25
9	Dallas	0	25	0	0	25
10	Miami	0	25	0	0	25
11	Ciudad de Panama	0	25	0	0	25
12	Bogota	0	25	0	0	25
13	Lima	0	25	0	0	25
14	La Paz	0	0	0	0	0
15	Manaus	0	25	0	0	25
16	Fortaleza	0	25	0	25	50
17	Sao Paulo	0	0	0	0	0
18	Buenos Aires	0	0	0	25	25
19	Santiago	0	0	0	0	0
20	Ushuaia	0	0	0	0	0
21	Oslo	0	0	0	0	0
22	London	0	25	0	25	50
23	Madrid	0	25	0	0	25
24	Paris	0	25	0	25	50
25	Frankfurt	0	25	0	0	25
26	Budapest	0	25	0	0	25
27	Roma	0	25	0	0	25
28	St. Petesburgo	0	0	0	0	0
29	Moscú	0	0	0	0	0
30	Istanbul	0	25	0	25	50
31	Cairo	0	25	0	25	50
32	Casa Blanca	0	25	0	0	25
33	Dakar	0	25	0	25	50
34	Addis Ababa	0	25	0	0	25
35	Lagos	0	25	0	25	50
36	Nairobi	0	25	0	0	25
37	Luanda	0	25	0	0	25
38	Cape Town	0	0	0	0	0
39	Tel-Aviv	0	25	0	0	25
40	Dubai	0	25	0	0	25
41	Tehran	0	25	0	0	25
42	Astana	0	25	25	0	50
43	Ulan Bator	0	25	25	0	50
44	Dehli	25	25	0	0	50
45	Chennai	0	25	0	0	25
46	Chongqing	25	25	25	0	75
47	Shenzhen	25	25	25	0	75
48	Jakarta	0	25	25	0	50
49	Kuala Lumpur	0	25	25	0	50

Region	City	Total Jetlag Index
53	Beijing	100
54	Tokyo	100
46	Chongqing	75
47	Shenzhen	75
50	Perth	75
16	Fortaleza	50
22	London	50
24	Paris	50
30	Istanbul	50
31	Cairo	50
33	Dakar	50
35	Lagos	50
42	Astana	50
43	Ulan Bator	50
44	Dehli	50
48	Jakarta	50
49	Kuala Lumpur	50
55	Vadivostok	50
1	Calgary	25
2	Vancouver	25
3	Toronto	25
4	Los Angeles	25
5	Seattle	25
6	Chicago	25
7	New York	25
8	Mexico D.F.	25
9	Dallas	25
10	Miami	25
11	Ciudad de Panama	25
12	Bogota	25
13	Lima	25
15	Manaus	25
18	Buenos Aires	25
23	Madrid	25
25	Frankfurt	25
26	Budapest	25
27	Roma	25
32	Casa Blanca	25
34	Addis Ababa	25
36	Nairobi	25
37	Luanda	25
39	Tel-Aviv	25
40	Dubai	25
41	Tehran	25
45	Chennai	25
51	Melbourne	25
52	Sydney	25
56	Auckland	25
14	La Paz	0

Final results and jetlag index order

This is how the final algorithm in excel looks like, with 25 percent zoom size!



Results

This is what we obtained after computing the different case studies:

Small meeting case study	Large meeting case study
• **Time:** Mid June (month 6.5) • **Cities:** Monterey CA, USA Zutphen, Netherlands Melbourne, Australia Shanghai, China Hong Kong (SAR), China Moscow, Russia • **Participants:** One from each city • **Results:**	• **Time:** January (month 1) • **Cities:** Boston MA, USA Singapore Beijing, China Hong Kong (SAR), China Moscow, Russia Utrecht, Netherlands Warsaw, Poland Copenhagen, Denmark Melbourne, Australia • **Participants:** One from each city, except for both Hong Kong and Boston (with 2 participants) • **Results:**

Small meeting case study results:

Region	City	Total Jetlag Index
53	Beijing	100
54	Tokyo	100
46	Chongqing	75
47	Shenzhen	75
50	Perth	75
16	Fortaleza	50
22	London	50
24	Paris	50
30	Istanbul	50
31	Cairo	50
33	Dakar	50
35	Lagos	50
42	Astana	50
43	Ulan Bator	50
44	Dehli	50
48	Jakarta	50
49	Kuala Lumpur	50
55	Vadivostok	50
1	Calgary	25
2	Vancouver	25
3	Toronto	25
4	Los Angeles	25
5	Seattle	25
6	Chicago	25

Large meeting case study results:

Region	City	Total Jetlag Index
30	Istanbul	100
39	Tel-Aviv	75
40	Dubai	75
41	Tehran	75
53	Beijing	50
54	Tokyo	50
50	Perth	50
22	London	50
24	Paris	50
31	Cairo	50
35	Lagos	50

After inputting all of the values in our algorithm, we obtained that Beijing, China from region 53 and Tokyo, Japan from region 54 had the highest "Jetlag Index", meaning that the meeting most probably should be held somewhere near those places.	After inputting all of the values in our algorithm, we obtained that Istanbul, Turkey from region 30 had the highest "Jetlag Index", meaning that the meeting most probably should be held somewhere near those places.

Evaluation and other alternatives

The results we obtained help minimize the effect of Jetlag on the participants attending the conference. Nonetheless, these are by no means the only results that can be obtained. Our algorithm analyses the main 4 variables (latitude/longitude, time zones, daylight hours, and altitude) that affect jetlag, and then returns an Index, a region and a city where meeting could be held.

Different results could be obtained, for instance, if we changed the month of the year, or even the different "X" variables that establish the parameters. But these, at the end of the day, depend on the very specifications of the conference organizers. What we did just helps them in the process of selecting the city, or at least narrows their options.

Naturally, the results are based upon the assumptions we stated previously. As expected, assumptions might remove credibility to the whole process, making it not very real. Nonetheless, these do not stop us, and our algorithm, of obtaining fairly reliable results.

If we wanted to make the algorithm even more precise than what it is now, it should account for additional variables that include:

- Mean temperatures - Could be modelled again using sine curves, though it is more irregular

- The direction of travel

- Diet - Difficult to compute mathematically

- Cartesian coordinates in the XYZ area instead of latitudes and longitudes - Difficult to compute, given the earth is not a perfect sphere

- Dividing the world in more regions, and adding more cities to them - This would account for a larger city database, obtaining more precise results. Utopically, the best algorithm would have each and every city of the world!

- Include flights and costs – though the algorithm reduces the total distance travelled, meaning less costs.

In fair honesty, in the 5 days the International Meeting Management Corporation gave us, it would have been very difficult to consider all of these variables. Still, we are satisfied with what we obtained!

Perhaps next time, instead of having everybody flying to a conference place, we could just organize a Skype conference, with no Jetlag at all!

Formula Appendix

The following formulae are part of the main body of the designed algorithm:

1) Sine formula:

```
=AH22*(SIN((2*PI()/12)*$AK$3 - (PI()/2)))+AE22-AH22
```

AH22 is the amplitude, the blocked cell $AK$3 is the month cell, and AE22 is the maximum daylight.

2) IF formula for the latitude and longitudes:

```
=IF(AT8<180; 180 - (AO4*AP4 + AO5*AP5 + AO6*AP6 + AO7*AP7 + AO8*AP8 + AO9*AP9 + AP10*AO10 + AP11*AO11 + AP12*AO12)/SUM(AP4:AP12); (AO4*AP4 + AO5*AP5 + AO6*AP6 + AO7*AP7 + AO8*AP8 + AO9*AP9 + AP10*AO10 + AP11*AO11 + AP12*AO12)/SUM(AP4:AP12))
```

This basically says that if AT8, the separation in degrees across the pacific, is less than 180, then it should compute the cumulative average (each longitude multiplied the respective number of people), all divided the total number of people), and then subtract it from 180.

3) Look up formula:

```
=INDEX($BQ$22:$BQ$77; SMALL(INDEX(($BY$19=$BR$22:$BR$77)*(MATCH(ROW($BR$22:$BR$77); ROW($BR$22:$BR$77)))+($BY$19<>$BR$22:$BR$77)*1048577; 0; 0); ROW(A1)))
```

This formula was used to find the list of cities that matched the criteria. The INDEX would return the city, and would do that using a SMALL function, that would check for the for $BY$19, the expectation, in matrix $BR$22: $BR$77, where all the "Yes" and "No" were. The rest is just computational excel values that are required for the algorithm to work.

4) Look up formula 2:

```
=INDEX($D$22:$D$77;MATCH(AQ22;$C$22:$C$77;0))
```

This formula was used after the one above to find the respective values assigned for each city. AQ22 is the city name, $C$22:$C$77 is the matrix in which to find, and would return the row coordinate, to be looked up in matrix $D$22:$D$77.

Bibliography

- *http://www.nhs.uk/Conditions/Jet-lag/Pages/Symptoms.aspx*
- *http://www.medicalnewstoday.com/articles/165339.php*
- *https://en.wikipedia.org/wiki/Jet_lag*
- *http://www.foxnews.com/travel/2014/11/17/key-factors-that-affect-jet-lag.html*
- *https://www.timeanddate.com/sun*
- *http://www.jetlagrooster.com*
- *http://www.medicinenet.com/jet_lag/article.htm*
- *http://www.health.com/health/gallery/0,,20322187,00.html*
- *http://www.timetemperature.com/time-zone-maps/world-time-zone-map.shtml*
- *http://everytimezone.com*
- *https://www.geovista.psu.edu/grants/MapStatsKids/MSK_portal/concepts_latlg.html*
- *https://www-istp.gsfc.nasa.gov/stargaze/Slatlong.htmgo*