COMP529: Batch Analytics Assignment - How Variable Different Employees' Movements Are

Middleware Configuration

The Hadoop standalone mode has been used for this assignment. The rationale for this is: there is just a single computer available (not a cluster), it is easier to debug a program in standalone mode and the data is sufficiently small to be run on a single computer.

Data Analytic Design

The MapReduce program 'maps' a line of the data if the GPS coordinates are inside a specified circle. The key (in the key, value pair) consists of 6 digits: the first three represent the employee number (e.g.'000', '001'...'181'), the next two represent the hour and the final digit, the day of the week.

The majority of the data was recorded in Beijing, by Microsoft employees, and hence the centre of the circle was set to be a Microsoft office in Beijing (of which there are several). Microsoft issued a User Guide for this dataset (see Appendix A), but this does not include information regarding which office the employees are based.

However, the User Guide does contain a heat map representing the most travelled paths in Beijing. The office selected, to be the centre of the circle, is within the "hottest" part of the map (the area inside the green circle). There are two Microsoft Research offices in this area, so one was chosen, leaving the option to run the program for a second time, with the other office as the centre point.

The User Guide suggests that employees recorded data for different periods of time and at different frequencies. For example, some employees were tracked for a year and some for a week, and the frequency of readings were mainly between every 1 to 5 seconds. It is not easy to compensate for this variability with the data provided, and so the approach taken assumes the data to be homogeneous.

Please see Appendix H for the full Java code, including comments. Note: to reduce the runtime of the program, the code only calculates the key, for a line of data, if it is in the circle.

Results

The results are listed in Appendices B-F. Note: 29 of the employees in the dataset were not counted by the MapReduce program. The User Guide states that some of the data was obtained outside of Beijing or even outside of China, and hence it is expected that some employees will be excluded from the results. It is assumed that these employees are not relevant to the study, and hence they have not been included in the statistics calculations.

Discussion of Results

Count per hour of the day

Appendix B shows the results, sorted by total count, from highest to lowest. An initial observation is that there is a large difference in the number of counts for each employee; the highest being 141,782 and the lowest just 2. It could be that the majority of this difference is due to variations in the period of time each employee collected data for. The User Guide shows that just 1% of the employees collected data for over a year, while 25% collected data for less than a week. Also, noted above, there are various Microsoft offices in Beijing and some employees may work predominantly at another offices, and hence count relatively fewer times compared to those based at the office chosen here.

A bar chart of this data (see Appendix C) shows that the employees were counted more often during the night. It could be that most these employees work night shifts (although this seems improbable), or perhaps the data was recorded in a different time zone.

Looking at the two employees with the highest count (022 and 067, blue and orange in the bar chart), it can be seen that their movements are almost the opposite to each other. 022 counted most frequently during the night, but 067 counted most frequently in the afternoon and evening.

Appendix D shows the key statistics for the data. In answering the question "how variable are different employees' movements?" these results clearly show the variability is high, with a high degree of skewness. Each hour has a large standard deviation in comparison with the mean. As noted above, the approach does not correct for the data being recorded for different lengths of time and at different frequencies; this will exaggerate the variability between the employees.

Appendix E shows the distribution of the number of counts per employee. This looks a lot like an exponential distribution. Appendix F is a graph of the logarithm of the number of counts. Note that this produces a very straight line (apart from the two tails) and hence strongly suggests that this is an exponential distribution.

Count per day of the week

There is still a lot of variability between the employees, when the counts are split by the day of the week, with a high ratio between the standard deviation and the mean (see Appendix D and G).

Interestingly, there are more counts on a Sunday than on a Monday, and given this is a place of work, it could be expected to be other way around. Google maps shows that the office is very near (e.g. within 500m) to a main road and the area around the office has restaurants, banks, petrol stations, a Walmart etc. Hence, there are many other reasons an employee would be counted, other than going to work.

Issues with the data and context of the question

One of the key issues with this dataset is that each employee was tracked for different lengths of time and also at different frequencies. Without correcting for this, it is difficult to make direct comparisons.

The data is also biased in that everyone being tracked was an employee of Microsoft, so they are very likely to visit the office on a frequent basis and the profile of the rest of the population of Beijing is excluded.

The context of the question was to aid registration plate recognition in the UK. If we were to use this dataset to improve these systems, we are assuming that the movements of Microsoft employees in Beijing are similar to the general public in the UK, and that a single office is equivalent to an area in London or a toll bridge; however, the validity of this would need to be tested.

Further Investigations

As mentioned above, there is another Microsoft Research office in Beijing which could have been selected at the centre point of the circle. The program was run for a second time, changing the centre point. As expected, the results were similar to the initial results, and hence they are considered to give any further insight into the variability of the employees' movements.

Another option for further investigation could include changing the radius of the circle, perhaps to represent the London Congestion Charge zone. However, before doing this, it would be important to understand if there is an area in Beijing that is similar to this area, else the investigation will not help solve the overall problem.

This approach did not consider the number of times an employee transits the boundary of the circle. In the context of the London Congestion Charge, this would not be important as a vehicle is charged per day, not per transit. However, when considering the Dartford Crossing, for example, this could be important as a vehicle is charge per crossing.

Conclusion and Recommendations

The results show the employees' movements to be very variable. However, there are some key issues with the dataset, namely that employees recorded data for different periods and frequencies.

Most fundamentally, it is not clear that investigating the movements of Microsoft employees in Beijing would help to solve the problem of reading ambiguous registration plates in the UK. Ideally, the analysis would be performed on a more relevant dataset, for example, vehicles which have entered the London Congestion Charge Zone.

To avoid a memory error while running the program, the data had to be run in four separate parts. If a larger dataset were analysed, it would be beneficial to use a cluster of computers and set up Hadoop in fully distributed mode. However, it would be wise to debug the program in standalone mode.

Appendix A - User Guide

See http://research.microsoft.com/pubs/152176/%User20Guide-1.2.pdf for the full document

User Guide

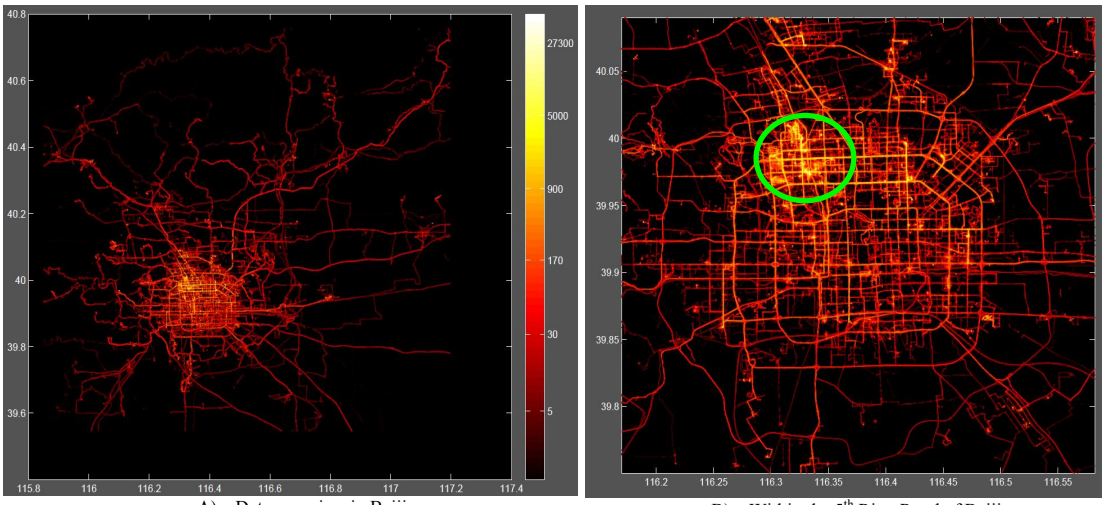
Version 1.2 (2011/10/31)

1. Data Description

This GPS trajectory dataset was collected in (Microsoft Research Asia) Geolife project by 178 users in a period of over four years (from April 2007 to October 2011). A GPS trajectory of this dataset is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. This dataset contains 17,621 trajectories with a total distance of 1,251,654 kilometers and a total duration of 48,203 hours. These trajectories were recorded by different GPS loggers and GPS-phones, and have a variety of sampling rates. 91 percent of the trajectories are logged in a dense representation, e.g. every 1~5 seconds or every 5~10 meters per point.

This dataset recoded a broad range of users' outdoor movements, including not only life routines like go home and go to work but also some entertainments and sports activities, such as shopping, sightseeing, dining, hiking, and cycling. This trajectory dataset can be used in many research fields, such as mobility pattern mining, user activity recognition, location-based social networks, location privacy, and location recommendation.

Although this dataset is wildly distributed in over 30 cities of China and even in some cities located in the USA and Europe, the majority of the data was created in Beijing, China. Figure 1 plots the distribution (heat map) of this dataset in Beijing. The figures standing on the right side of the heat bar denote the number of points generated in a location.



A) Data overview in Beijing

B) Within the 5th Ring Road of Beijing

Figure 1 Distribution of the dataset in Beijing city

The distributions of distance and duration of the trajectories are presented in Figure 2 and Figure 3.

In the data collection program, a portion of users have carried a GPS logger for years, while some of the others only have a trajectory dataset of a few weeks. This distribution is presented in Figure 4, and the distribution of the number of trajectories collected by each user is shown in Figure 5.

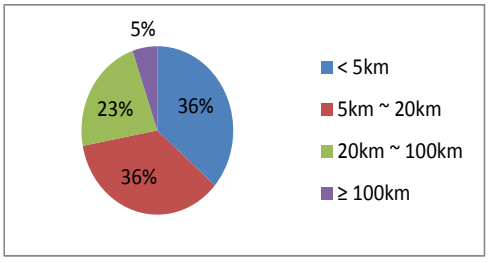


Figure 2 Distribution of trajectories by distance

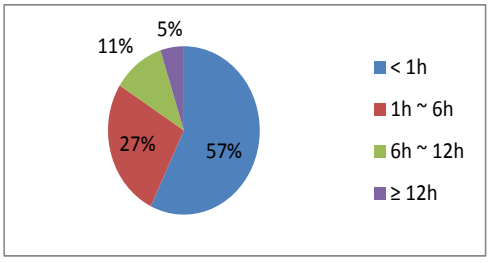
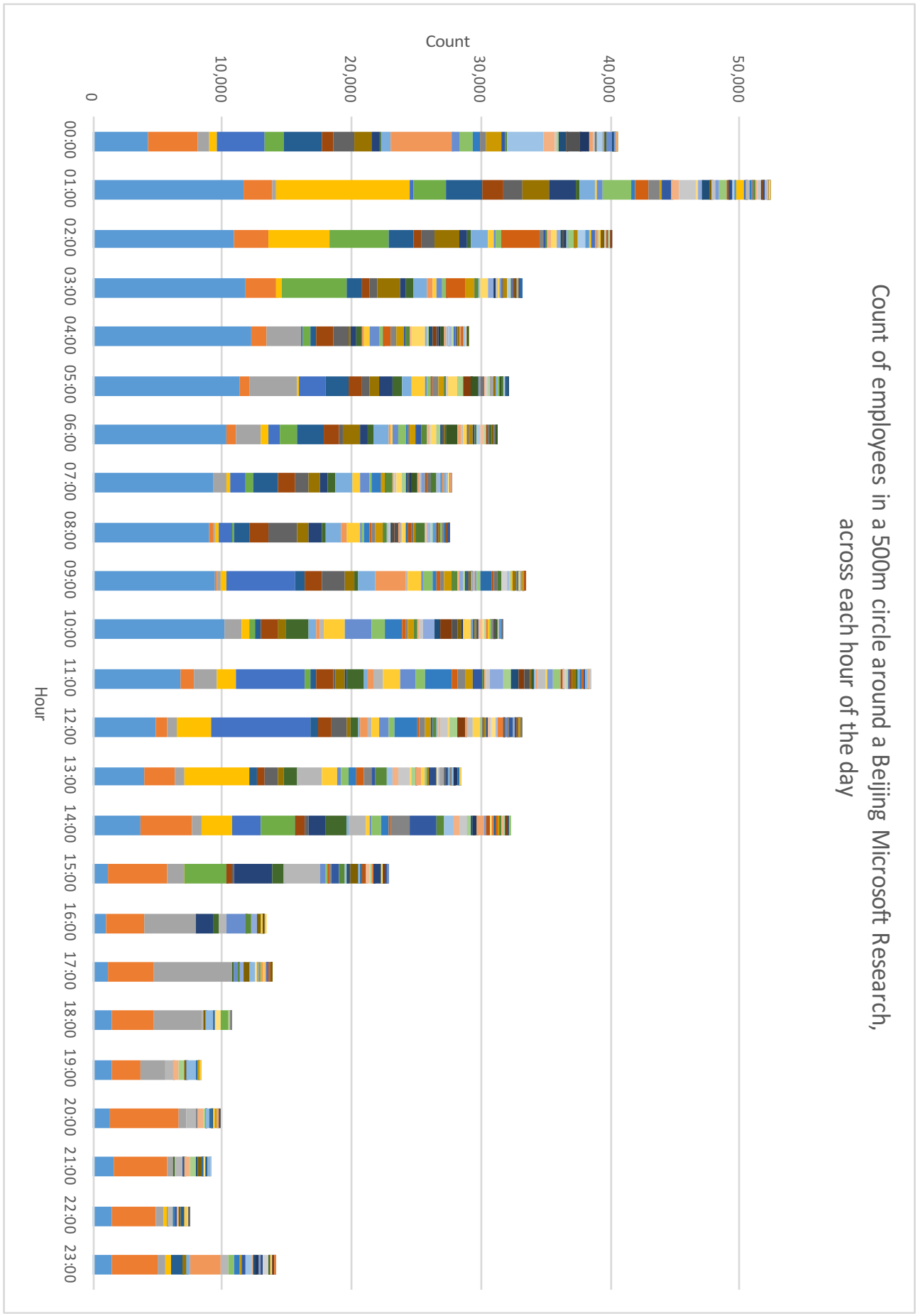


Figure 3 Distribution of trajectories by effective duration

Appendix B - Results

Employee/ Hour	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	Total
022	4,302	11,671	10,949	11,830	12,159	11,296	10,258	9,215	8,975	9,482	10,153	6,836	4,885	3,991	3,692	1,228	988	1,101	1,514	1,486	1,278	1,573	1,390	1,530	141,782
067	3,820	2,166	2,568	2,287	1,300	768	775	0	324	68	0	910	786	2,374	3,927	4,524	2,911	3,557	3,199	2,175	5,298	4,134	3,407	3,463	54,741
036	827	347	0	28	2,594	3,664	1,921	1,074	193	258	1,317	1,802	776	714	828	1,326	4,029	6,028	3,747	1,884	687	513	698	653	35,908
128	640	10,361	4,813	463	40	140	556	374	279	515	576	1,436	2,651	5,064	2,243	19	2	0	0	0	0	0	173	398	30,743
141	3,683	174	0	0	89	2,143	981	1,149	1,017	5,255	0	5,371	7,668	0	2,298	0	0	0	0	0	0	0	0	0	29,828
165	1,506	2,566	4,534	5,043	669	0	1,331	628	143	0	432	480	0	0	2,567	3,199	0	0	0	0	0	0	0	0	23,098
044	2,903	2,781	1,871	1,092	447	1,696	1,953	1,855	1,161	833	553	459	579	522	122	85	0	0	0	0	0	0	0	863	19,775
033	939	1,610	681	638	1,270	1,040	1,306	1,332	1,419	1,242	1,209	1,328	1,085	610	615	411	0	0	0	0	0	0	0	0	16,735
016	1,573	1,550	978	653	1,117	625	275	1,061	2,214	1,834	73	143	1,212	1,036	385	73	0	0	0	0	0	0	0	22	14,824
032	1,405	2,077	1,935	1,740	158	729	1,287	908	996	775	571	651	247	390	0	0	0	0	0	0	0	0	0	262	14,131
020	529	2,061	506	369	502	1,003	576	529	906	0	0	145	0	0	1,370	2,985	1,419	0	0	0	0	0	46	0	12,946
167	84	185	321	692	376	820	471	553	315	274	1,746	1,447	616	1,034	1,599	836	357	152	0	0	0	90	46	67	12,081
015	778	1,202	1,409	1,019	89	715	1,111	1,307	1,241	1,341	546	264	164	85	81	124	0	0	0	0	0	0	0	262	11,738
013	4,670	0	0	403	84	0	143	0	434	2,310	384	354	547	0	0	0	0	0	0	0	0	0	0	2,390	11,719
005	52	0	0	0	0	0	0	0	55	138	277	857	274	1,812	1,325	2,689	639	69	14	627	669	544	356	562	10,959
153	0	170	402	235	559	1,083	278	675	1,041	984	1,627	1,322	620	1,310	388	64	0	0	0	0	0	0	0	0	10,758
007	560	480	169	423	600	142	404	658	34	216	2,126	1,063	817	275	151	477	1,421	302	0	0	0	26	73	0	10,417
085	1,087	2,157	474	311	301	199	485	278	190	714	967	849	385	516	747	37	0	6	0	0	0	39	24	462	10,228
084	518	329	24	55	39	13	198	611	465	378	1,337	1,943	1,691	616	499	109	36	0	0	0	0	0	100	366	9,327
140	85	992	2,832	1,551	609	189	72	82	138	170	309	530	254	576	198	45	41	29	49	0	0	0	0	0	8,751
163	453	963	294	0	453	372	95	1	247	371	152	585	330	567	1,379	164	0	2	0	68	131	54	60	13	6,754
040	1,104	149	119	669	591	519	456	340	554	621	428	527	443	0	0	0	0	0	0	0	0	0	0	150	6,670
065	409	654	151	24	181	28	502	1	30	3	0	760	171	357	2,179	682	0	0	0	0	0	149	90	273	6,644
126	20	35	88	308	287	137	333	582	327	354	295	244	397	842	530	406	396	150	0	0	0	5	0	0	5,736
074	2,933	82	0	126	0	0	93	0	131	81	3	0	91	526	744	64	0	0	0	0	0	0	0	448	5,322
039	794	470	280	0	72	8	56	150	96	41	174	128	116	346	507	0	0	0	41	385	521	444	137	146	4,912
082	164	1,364	33	0	0	77	159	60	59	0	222	307	640	917	553	0	0	0	0	0	0	0	0	0	4,555
043	67	138	467	624	1,099	700	522	435	57	19	0	0	63	162	0	0	28	0	0	0	0	0	8	21	4,410
073	0	267	215	350	70	4	15	21	2	345	871	1,030	138	20	9	53	469	206	11	0	0	0	0	21	4,117
112	73	68	2	4	201	455	198	290	1	111	31	488	457	239	325	48	9	33	28	478	51	386	0	44	4,020
010	616	525	142	55	223	0	111	220	146	124	529	600	135	0	24	106	0	0	0	0	0	114	0	53	3,723
078	1	102	1	0	304	652	0	0	144	0	832	527	489	0	0	0	0	0	0	21	0	0	0	0	3,073
017	965	0	113	0	169	21	257	26	186	256	469	371	66	0	0	118	0	0	0	0	0	0	0	0	3,017
159	99	100	0	0	0	0	60	14	29	25	186	125	0	58	127	570	203	423	118	112	2	252	204	13	2,720
115	670	70	206	82	209	16	0	126	47	1	177	74	0	25	174	58	2	29	22	43	97	98	0	340	2,566
026	0	0	64	43	204	559	924	477	0	127	0	65	0	0	32	0	0	0	0	0	0	0	0	20	2,515
024	0	0	26	10	27	0	0	20	0	132	22	256	0	0	0	108	0	465	464	615	323	0	0	0	2,468
125	343	0	17	57	263	52	262	123	96	32	8	66	140	368	580	0	0	0	0	0	0	0	0	0	2,407
030	102	246	57	42	72	0	220	167	171	84	37	556	464	0	28	0	0	0	0	0	0	0	0	20	2,266
179	0	0	0	228	0	0	239	0	261	0	515	194	572	240	0	0	0	0	0	0	0	0	0	0	2,249
008	0	333	51	43	0	2	0	282	43	145	140	481	37	113	222	0	0	0	0	0	0	0	0	110	2,002
142	53	537	393	0	10	0	5	8	3	229	44	537	50	70	0	39	0	0	0	0	0	0	0	0	1,978
147	0	0	0	0	0	0	0	0	131	914	128	0	0	0	0	100	0	0	178	132	55	90	0	0	1,728
004	0	72	27	50	26	0	19	205	366	103	46	110	93	0	256	268	0	0	0	0	0	24	0	61	1,726
014	85	2	15	130	41	335	81	237	126	424	2	15	39	85	33	57	9	0	1	0	0	0	0	0	1,717
096	0	35	228	336	0	0	400	0	193	5	88	13	161	161	0	0	0	0	0	0	0	0	0	0	1,620
155	0	258	0	0	0	0	70	60	0	0	228	31	243	594	0	0	0	0	0	0	0	0	0	94	1,578
023	0	0	0	0	0	0	251	423	675	179	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,528
035	406	167	546	70	26	0	265	1	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,511
006	0	0	0	22	104	195	198	0	132	13	261	175	84	0	7	282	0	0	0	0	0	0	0	0	1,473
052	0	0	27	95	0	74	89	45	5	472	40	63	160	0	0	0	0	0	0	0	0	0	0	392	1,462
130	0	0	0	0	0	0	0	0	0	0	157	0	331	61	51	0	0	49	536	74	0	82	30	0	1,371
158	0	28	0	0	81	100	0	243	313	204	2	48	97	122	0	0	0	0	0	0	0	0	0	0	1,238
003	0	53	4	52	113	97	107	32	144	76	45	98	42	61	23	227	0	0	0	0	0	26	0	11	1,211
028	71	174	279	93	0	33	0	105	246	44	0	0	0	0	0	0	0	0	0	0	0	0	0	12	1,057
071	0	0	0	0	0	2	0	0	0	0	232	48	335	2	174	87	4	55	0	28	23	0	0	15	1,005
093	106	0	0	0	0	278	0	0	0	0	0	0	0	301	0	0	63	148	0	0	24	0	0	0	920
169	0	578	121	0	0	0	0	0	0	63	118	0	0	0	0	0	0	0	0	0	0	0	0	0	880
144	0	73	295	129	156	17	0	0	0	0	11	114	77	0	0	0	0	0	0	0	0	0	0	0	872
122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	2	202	523	9	33	0	0	0	806
091	0	6	0	0	0	1	0	0	28	100	37	0	20	9	66	20	0	0	18	0	49	247	204	0	805
001	66	0	0	132	0	11	140	0	201	0	0	191	29	0	0	0	0	0	0	0	0	0	0	0	770
081	0	0	0	0	248	29	206	11	0	0	0	0	85	119	72	0	0	0	0	0	0	0	0	0	770
025	0	0	0	50	55	94	0	53	9	195	0	277	0	0	0	0	0	0	0	0	0	0	0	0	733
100	0	0	0	0	0	0	0	0	0	0	0	0	0	197	0	501	0	0	0	0	0	0	0	0	698
168	88	74	112	76	0	0	60	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	81	643
068	0	0	0	69	67	121	0	0	89	0	0	141	0	83	47	0	0	0	0	0	0	0	0	0	617
102	78	6	83	18	63	0	0	18	0	98	0	0	0	0	0	0	0	163	0	10	26	0	30	0	593
170	0	289	185	0	0	0	0	0	0	0	0	49	0	0	0	0	0	0	0	0	0	0	0	0	523
083	0	0	90	0	0	0	0	0	0	0	0	40	0	6	0	0	31	126	0	0	89	17	103	20	522
070	367	0	0	0	0	0	0	0	0	37	0	0	99	0	0	0	0	0	0	0	0	0	0	0	503
011	86	0	0	0	0	94	0	117	98	0	99	0	0	0	0	0	0	0	0	0	0	0	0	0	494
089	0	59	11	60	111	3	0	0	0	3	14	132	77	0	1	4	0	0	0	2	0	0	0	3	480
000	0	59	0	40	24	0	68	0	103	16	26	65	42	5	23	0	0	0	0	0	0	0	0	0	471
058	0	149	0	0	0	0	0	0	3	11	30	7	24	97	137	0	0	0	0	0	0	0	0	0	458
095	0	7	0	0	0	0	0	0	0	0	0	92	18	0	83	2	0	0	0	66	183	0	0	0	451
042	4	0	0	0	0	0	0	0	0	21	0	62	349	5	2	0	0	0	0	0	0	0	0	0	443
154	0	157	0	0	31	0	0	0	0	0	0	29	58	31	128	8	0	0	0	0	0	0	0	0	442
038	0	0	0	0	211	113	17	0	0	0	0	0	0	87	0	0	0	0	0	0	0	0	0	0	428
059	0	144	0	0	0	0	0	0	0	0	30	7	0	97	137	0	0	0	0	0	0	0	0	0	415
046	0	286	57	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	391
131	0	0	0	0	0	0	0	0	0	0	0	9	0	0	7	99	1	0	9	0	70	0	43	143	381
092	0	0	0	0	0	0	0	106	118	0	47	0	39	0	70	0	0	0	0	0	0	0	0	0	380
066	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	0	64	0	190	0	0	0	0	0	373
057	32	28	0	0	0	0	0	0	0	0	24	0	53	1	24	82	27	0	0	0	0	0	0	74	345
094	32	28	0	0	0	0	0	0	0	0	24	0	53	1	24	82	27	0	0	0	0	0	0	74	345
150	32	28	0	0	0	0	0	0	0	0	24	0	53	1	24	82	27	0	0	0	0	0	0	74	345
148	0	0	251	0	0	0	62	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	318
166	11	0	0	0	0	0	0	0	0	0	0	0	19	277	8	0	0	0	0	0	0	0	0	0	315
027	0	0	0	0	25	61	0	0	0	0	227	0	0	0	0	0	0	0	0	0	0	0	0	0	313
045	0	0	0	0	0	0	14	0	0	0	0	73	41	0	14	44	0	0	0	26	35	12	48	0	307
177	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	189	0	0	113	0	0	0	302
012	0	0	105	0	23	0	0	44	0	5	89	0	0	0	0	0	0	0	0	0	0	0	0	3	269
075	28	0	0	0	0	0	0	0	0	188	0	0	0	0	4	1	0	0	0	2	0	0	44	0	267
055	28	0	0	0	0	0	0	0	1	144	0	5	0	0	4	1	0	0	0	2	0	0	44	0	229
135	0	0	72	0	0	0	0	0	0	0	0	0	0	62	70	0	0	0	0	0	0	0	0	0	204
133	0	0	0	0	0	0	0	0	140	0	0	0	0	57	1	0	0	0	0	0	0	0	0	0	198
079	0	12	87	15	14	0	0	0	0	0	0	14	51	0	0	0	0	0	0	0	0	0	0	0	193
009	0	0	0	0	0	53	133	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186
124	17	74	0	0	0	0	0	0	0	0	19	63	0	0	0	0	0	0	0	0	0	0	0	0	173
136	0	36	6	0	14	8	3	11	6	9	0	0	29	2	15	25	0	0	0	0	0	0	0	0	164
029	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	152
037	0	0	0	0	0	60	24	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	143
062	0	0	0	0	0	10	0	0	0	3	23	89	0	0	0	0	0	0	0	0	0	0	0	0	125
114	0	50	64	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	121
063	0	0	0	0	0	0	0	117	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	117
113	11	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	115
152	0	42	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	104
088	0	0	0	0	0	0	0	0	0	0	99	0	0	0	0	0	0	0	0	0	0	0	0	0	99
134	0	0	2	1	1	29	2	0	0	9	0	25	18	0	0	0	0	0	0	0	0	0	0	4	91
031	0	0	0	0	0	67	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	84
111	0	50	0	12	0	0	1	0	0	10	2	0	0	1	1	0	0	0	0	0	0	0	0	0	77
178	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	75	0	0	0	0	0	0	75
107	16	3	0	0	0	0	0	26	5	12	0	0	12	0	0	0	0	0	0	0	0	0	0	0	74
002	0	0	38	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	72
105	0	0	0	0	0	0	8	6	46	6	0	0	5	0	0	0	0	0	0	0	0	0	0	0	71
149	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	14	28	0	66
119	0	0	0	0	0	0	0	0	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	63
056	0	0	0	0	0	0	0	4	0	0	5	1	2	27	15	6	0	0	0	0	0	0	0	0	60
019	0	0	0	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	0	57
103	0	0	0	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	52
121	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0	0	0	48
080	0	15	0	0	0	0	0	0	0	7	16	0	6	0	0	0	0	0	0	0	0	0	0	0	44
104	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	19	2	9	0	0	0	0	44
117	0	10	9	0	0	0	0	0	0	0	0	0	0	0	9	16	0	0	0	0	0	0	0	0	44
047	1	3	0	6	10	1	1	0	0	0	3	9	2	7	0	0	0	0	0	0	0	0	0	0	43
101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	6	0	0	0	0	0	42
109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	6	0	0	0	0	0	42
151	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	37
157	0	0	0	0	2	2	3	0	0	1	0	0	26	0	0	0	0	0	0	0	0	0	0	0	34
097	0	0	0	20	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31
098	0	0	0	0	0	0	0	0	0	0	12	0	0	17	0	0	0	0	0	0	0	0	0	0	29
077	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	25
162	6	14	2	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	24
172	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	24
127	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23
138	0	0	0	0	0	0	0	0	0	0	3	0	0	19	0	0	0	0	0	0	0	0	0	0	22
053	10	5	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	21
090	0	8	0	0	0	8	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20
076	0	8	7	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	18
110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	4	0	0	0	18
048	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	17
123	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
161	0	0	0	0	10	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
106	0	0	0	0	0	0	0	0	0	1	0	0	0	8	0	0	0	0	0	0	0	0	0	0	9
139	5	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
021	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	4
146	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	4
061	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2
072	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2
086	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
116	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
181	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	2
Total	40,413	52,173	39,970	32,975	28,964	31,949	31,191	27,587	27,452	33,278	31,610	38,382	32,984	28,380	32,120	22,678	13,219	13,667	10,713	8,254	9,761	8,958	7,386	14,038	618,102

Appendix C - Bar chart: hour of the day



Appendix D - Key statistics

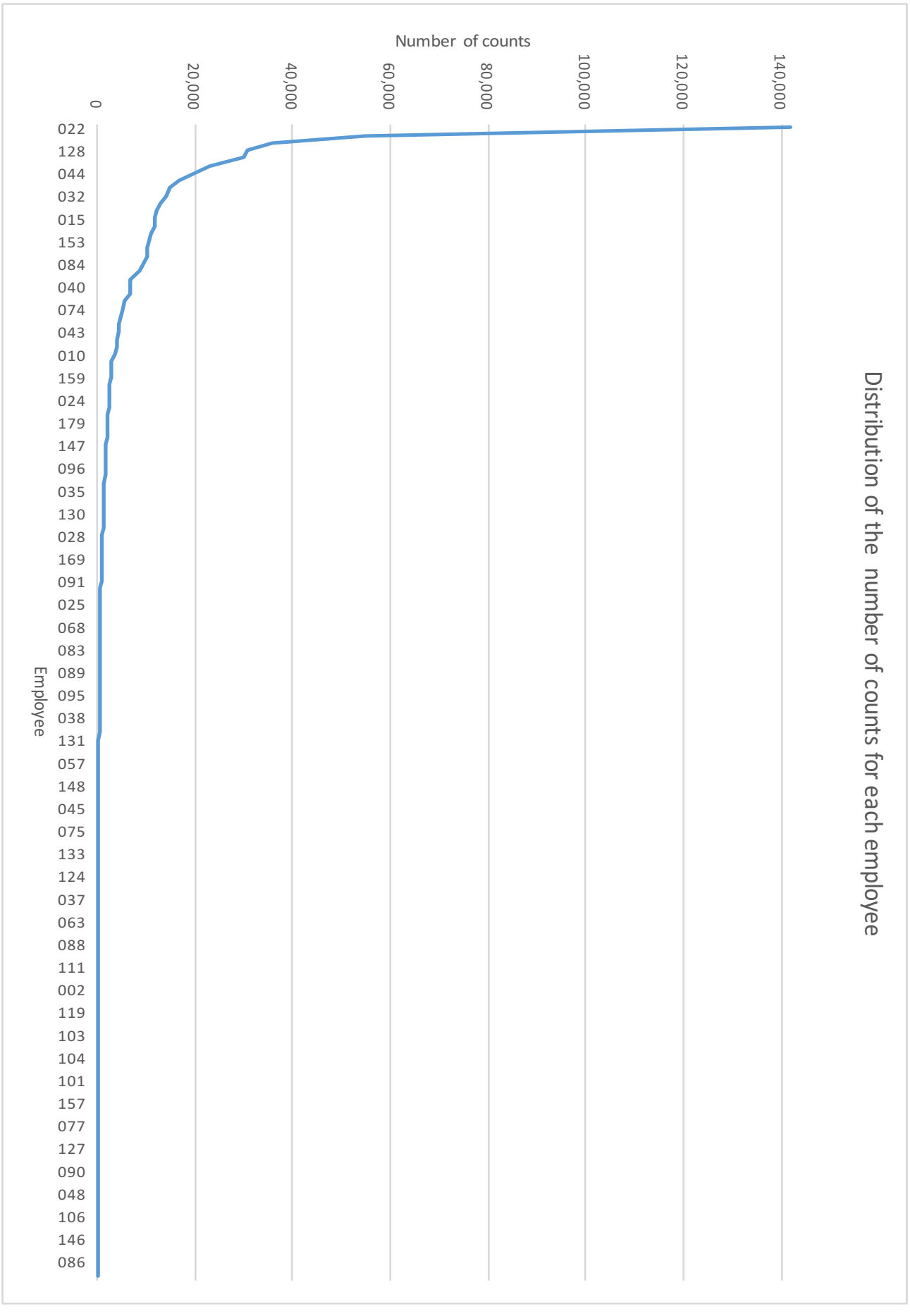
Statistics by the hour of the day

Hour in GMT	Total	Average	St. Dev.	St. Dev: Average	Variance	Min value	Lower quartile	Median	Upper quartile	Max value
00:00	13,219	86	441	5	194,360	0	0	0	0	4,029
01:00	13,667	89	572	6	327,089	0	0	0	0	6,028
02:00	10,713	70	419	6	175,533	0	0	0	0	3,747
03:00	8,254	54	272	5	73,944	0	0	0	0	2,175
04:00	9,761	64	448	7	200,477	0	0	0	0	5,298
05:00	8,958	59	364	6	132,532	0	0	0	0	4,134
06:00	7,386	48	303	6	91,988	0	0	0	0	3,407
07:00	14,038	92	377	4	141,820	0	0	0	11	3,463
08:00	40,413	264	772	3	596,703	0	0	0	71	4,670
09:00	52,173	341	1,335	4	1,783,158	0	0	0	102	11,671
10:00	39,970	261	1,090	4	1,188,218	0	0	0	72	10,949
11:00	32,975	216	1,072	5	1,149,217	0	0	0	50	11,830
12:00	28,964	189	1,021	5	1,041,552	0	0	0	70	12,159
13:00	31,949	209	994	5	987,528	0	0	0	53	11,296
14:00	31,191	204	882	4	778,161	0	0	0	95	10,258
15:00	27,587	180	792	4	627,156	0	0	0	59	9,215
16:00	27,452	179	777	4	604,122	0	0	0	118	8,975
17:00	33,278	218	918	4	843,311	0	0	0	98	9,482
18:00	31,610	207	879	4	772,853	0	0	0	89	10,153
19:00	38,382	251	770	3	592,948	0	0	0	143	6,836
20:00	32,984	216	788	4	620,172	0	0	0	97	7,668
21:00	28,380	185	601	3	360,643	0	0	0	83	5,064
22:00	32,120	210	609	3	371,315	0	0	0	70	3,927
23:00	22,678	148	570	4	325,204	0	0	0	39	4,524
Total	**618,102**	**4,040**	**13,320**	**3**	**177,423,972**	**2**	**72**	**443**	**2,266**	**141,782**

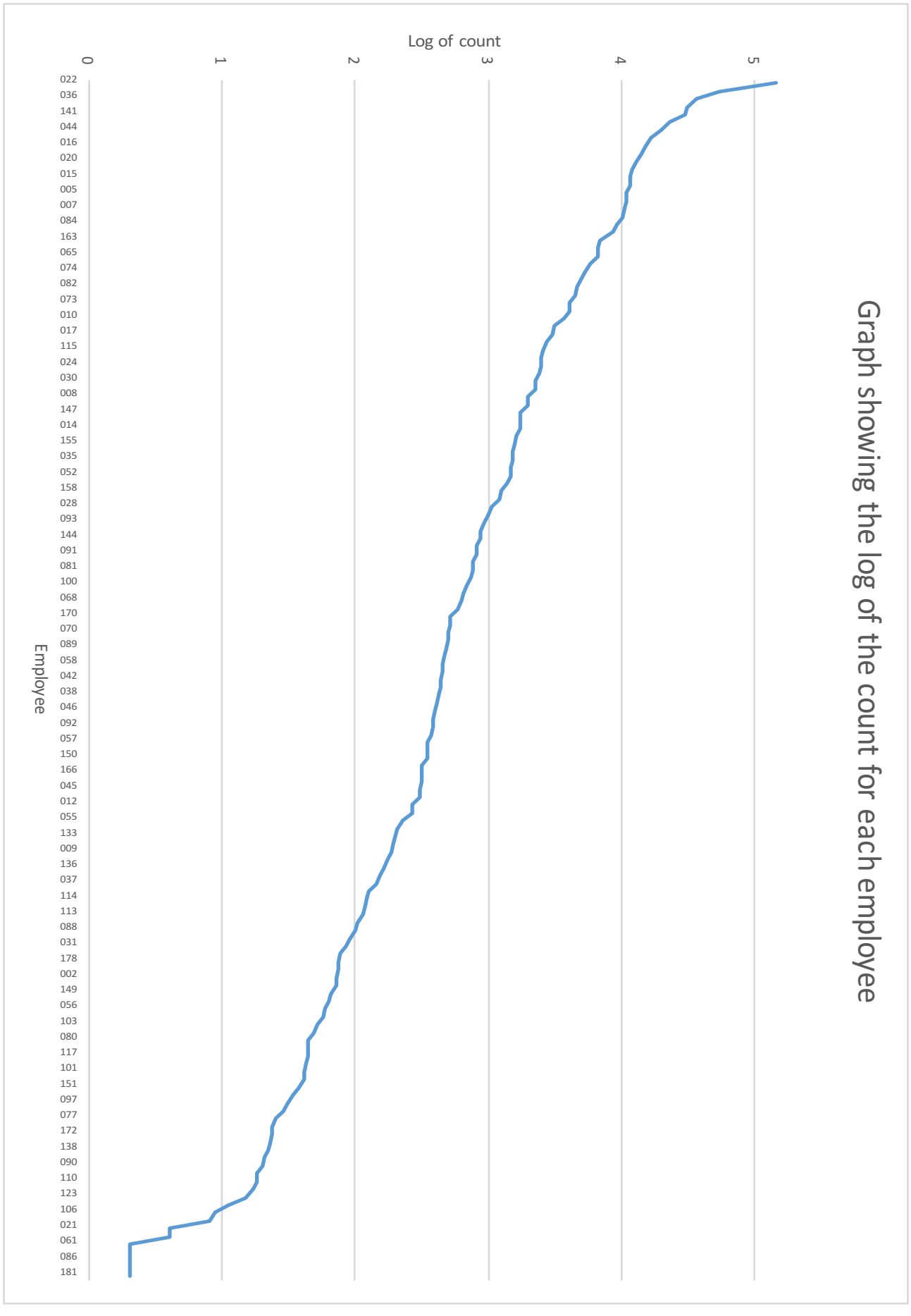
Statistics by the day of the week

Day	Total	Average	St. Dev.	St. Dev: Average	Variance	Min value	Lower quartile	Median	Upper quartile	Max value
Sunday	73,839	483	1,359	3	1,845,961	0	0	17	253	9,303
Monday	71,523	467	1,469	3	2,156,638	0	0	18	267	14,848
Tuesday	90,032	588	2,891	5	8,360,082	0	0	27	281	33,708
Wednesday	99,285	649	2,997	5	8,983,334	0	0	17	208	35,071
Thursday	105,691	691	2,968	4	8,808,442	0	0	21	231	33,187
Friday	98,574	644	1,796	3	3,224,418	0	0	54	314	11,591
Saturday	79,158	517	1,740	3	3,026,190	0	0	11	283	12,525
Total	**618,102**	**4,040**	**13,320**	**3**	**177,423,972**	**2**	**72**	**443**	**2,266**	**141,782**

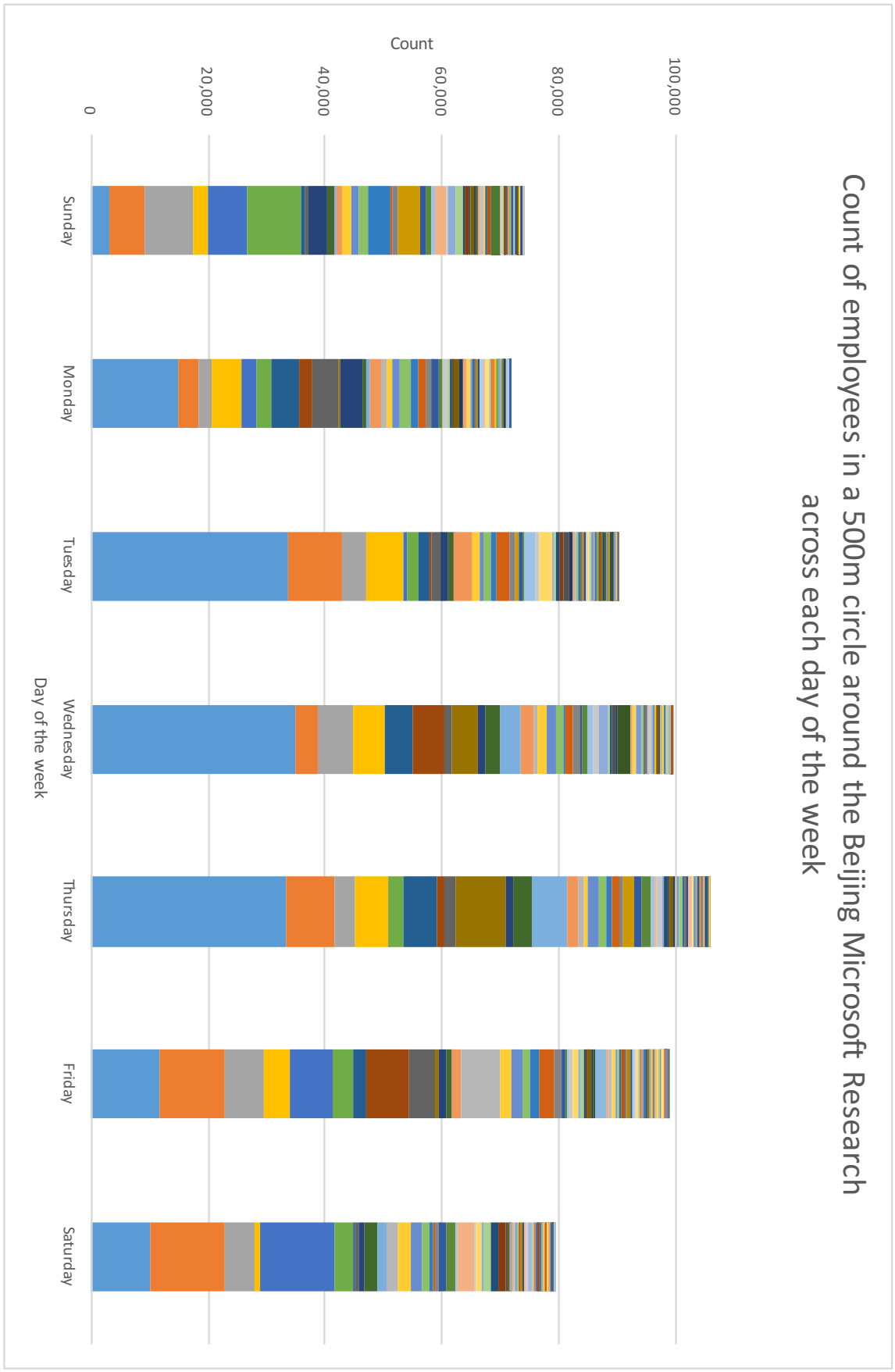
Appendix E - Graph of count distribution



Appendix F - Log of count distribution



Appendix G - Bar Chart: day of the week



Appendix H - Java code

```
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.GregorianCalendar;
import java.util.Date;
import java.util.Scanner;
import java.util.Calendar;

import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class GPScount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text GPS_key = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {

      //first check that the line is complete/contains data we what to read
      /*** count should be equal to 6 ***/
      int count_commas = StringUtils.countMatches(value.toString(), ",");

      if(count_commas == 6){
      //use scanner function to read along the line (with a , delimiter)
      Scanner line = new Scanner (value.toString());
      //set the delimiter
      line.useDelimiter(",");

      //set the centre point of the circle and radius
      double centre_x = 39.9765778;
      double centre_y = 116.3360511;
      double r = 0.0045; //assume one degree is equivalent to 111km =>
                         //500m = 0.0045 degrees

        //initiate variables
      double latitude, longitude;
      latitude = longitude = 0;
      String date_string, time;
      date_string = time = null;

      //iterate through the line and set the latitude, longitude, date and
      //time (e.g. skip some of the data)
      while (line.hasNext()){
            latitude = Double.parseDouble(line.next());
            longitude = Double.parseDouble(line.next());
            line.next(); line.next(); line.next(); // skip some of the fields
            date_string = line.next(); //get the date
            time = line.next(); // get the time
```

```
      }//end of while
      line.close();//close the scanner

            //check if coordinates are in the circle and if they are => MAP
            if ((longitude - centre_y)*(longitude - centre_y) +
            (latitude - centre_x)*(latitude - centre_x) < r*r){

            //get the file path of the line being read, to get the employee number
            Path filePath = ((FileSplit)context.getInputSplit()).getPath();

            //use scanner to read through the file path, with / as the delimiter
            Scanner employee = new Scanner (filePath.toString());
            employee.useDelimiter("/");

            String employee_number = null;
            while(employee.hasNext()){
                  employee.next();employee.next();employee.next();
                  employee.next();employee.next();employee.next();
                  employee_number = employee.next(); //get the employee number
                  employee.next();employee.next();
            }
            employee.close();//close the scanner

            //get the day of the week from the date
            String day_key = null;
            try{
                  SimpleDateFormat format = new SimpleDateFormat("yyy-MM-dd");
                  Date date = format.parse(date_string);
                  Calendar cal = GregorianCalendar.getInstance();
                  cal.setTime(date);
                  int date_int = cal.get(Calendar.DAY_OF_WEEK);
                  day_key = Integer.toString(date_int); // day of the week
            }
            catch (java.text.ParseException e){e.printStackTrace();
                              }

             //create the key to be mapped. It consists 6 digits
            //the first = the employee number,
            //next two = the hour and last digit = the day of the week
              String hour_key = time.substring(0,2);
              String line_key = employee_number + hour_key + day_key;

              //key is mapped to value 1
              GPS_key.set(line_key);
              context.write(GPS_key, one);

             }//end of if(in the circle) statement

      }//end of first if(6 commas in a line) statement

   }//end of map method

}//end of TokenizerMapper


  //reduce method (this is unchanged from the WordCount code)
  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
```

```
    public void reduce(Text key, Iterable<IntWritable> values,
    Context context) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }


  //main method
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    //delete the output file if it exists
    FileSystem fs = FileSystem.get(conf);
    if(fs.exists(new Path(args[0]))){
      fs.delete(new Path(args[0]), true);
    }

    //set up the job
    Job job = Job.getInstance(conf, "gps count");
    job.setJarByClass(GPScount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    //input files => set file path for all 182 employees
    for (int i = 0; i < 181; i++){
      String file_number = String.format("%03d", i);
      FileInputFormat.addInputPaths(job,
      "/home/ubuntu/workspace/GPScount/Data/"+ file_number + "/Trajectory");
    }

    //create output file
    FileOutputFormat.setOutputPath(job, new Path(args[0]));

    System.exit(job.waitForCompletion(true)?0:1);

  }//end of main

}//end of GPScount
```