Paper 11482-2016

The Scheduler: Give Your Punch Data a New Dimension

Ariel Kumpinsky, The Claro Group, LLC[1]

ABSTRACT

The Scheduler is an innovative tool that maps linear data (i.e., time stamps) to an intuitive 3-dimensional representation. This transformation allows the user to identify relationships, conflicts, gaps, etc. that were not readily apparent in the data's native form. This tool has applications in operations research and litigation related analyses. This paper will discuss why *The Scheduler* was created, the data available to analyze the issue, and how this code can be used in other types of applications. Specifically, this paper will discuss how *The Scheduler* can be used by supervisors to maintain the presence of three or more employees at all times, which can help ensure all federal and state mandated work breaks are taken. Additional examples for *The Scheduler* include assisting construction foreman to create a schedule that visualizes the presence of contractors in a logical sequence while eliminating overlap and/or ensuring cushions of time between contractors and matching items to non-uniform information.

INTRODUCTION

Many businesses, such as retail stores, factories, and grocery stores, must generate employee work schedules that result in smooth operational output and that alert employees to take mandated work breaks. These schedules can also be used by attorneys and consulting experts to evaluate potential work break violations (i.e., workers either did not or were not allowed to take breaks). *The Scheduler* can create an employee schedule as well as identify any potential scheduling conflicts. This paper discusses the specific problem that resulted in the development of *The Scheduler*. It then addresses how the tool can be implemented to solve a variety of application challenges in operations research and litigation related analyses.

DEVELOPMENT OF THE TOOL

CONTEXT:

The Claro Group is a multidisciplinary consulting firm and its professionals regularly provide consulting services in the area of labor and employment-related disputes. In connection with such services, Claro professionals are sometimes asked to analyze employee work log data, often in the form of "punch-in" and "punch-out" records. The underlying datasets can contain millions of records, which in turn may require the development and implementation of various methodologies to help efficiently process such large datasets.

---

[1] *The views expressed are those of the author and do not necessarily reflect the views of The Claro Group or its clients. The processes and approach described in this article were based upon the available data and may not be equally applicable given the existence of other types/formats of data. This article is for general information purposes and is not intended to be and should not be taken as legal or accounting advice.*

DATA:

In one particular matter, Claro professionals evaluated the number of instances an employee potentially was unable to take a state mandated 30 minute meal break due to insufficient staffing. The data contained over six years of "punch-in" and "punch-out" records involving more than 25,000 employees across hundreds of different work locations (nearly 50 million observations in all). The data were organized such that each observation consisted of one punch-in and punch-out record ("Punch Data"), the employee's ID number, the employee's name, and the work location, among other variables. In other words, the Punch Data is a time log of each employee's actual working hours (i.e., non-break hours). Table 1 shows a sample of the data:

VIEWTABLE: Work.Punch5loc2

	emplid	date	time1	time2
1	9	02/25/2005	25FEB05:07:00:00	25FEB05:17:00:00
2	15	02/25/2005	25FEB05:12:57:00	25FEB05:17:00:00
3	15	02/25/2005	25FEB05:11:57:00	25FEB05:12:57:00
4	15	02/25/2005	25FEB05:08:00:00	25FEB05:11:57:00
5	17	02/25/2005	25FEB05:02:00:00	25FEB05:09:00:00
6	18	02/25/2005	25FEB05:09:00:00	25FEB05:13:03:00
7	18	02/25/2005	25FEB05:13:03:00	25FEB05:14:03:00
8	18	02/25/2005	25FEB05:14:03:00	25FEB05:17:59:00
9	18	02/25/2005	25FEB05:17:59:00	25FEB05:18:00:00
10	18	02/25/2005	25FEB05:18:00:00	25FEB05:18:01:00
11	18	02/25/2005	25FEB05:08:59:00	25FEB05:09:00:00
12	21	02/25/2005	25FEB05:14:06:00	25FEB05:18:22:00
13	21	02/25/2005	25FEB05:18:22:00	25FEB05:19:00:00
14	21	02/25/2005	25FEB05:14:00:00	25FEB05:14:06:00
15	22	02/25/2005	25FEB05:12:32:00	25FEB05:14:43:00
16	22	02/25/2005	25FEB05:14:43:00	25FEB05:15:55:00
17	22	02/25/2005	25FEB05:15:55:00	25FEB05:21:00:00
18	22	02/25/2005	25FEB05:21:00:00	25FEB05:21:02:00
19	22	02/25/2005	25FEB05:12:30:00	25FEB05:12:32:00
20	30	02/25/2005	25FEB05:08:22:00	25FEB05:09:30:00
21	30	02/25/2005	25FEB05:06:34:00	25FEB05:08:22:00
22	30	02/25/2005	25FEB05:06:30:00	25FEB05:06:34:00
23	31	02/25/2005	25FEB05:11:13:00	25FEB05:15:27:00
24	31	02/25/2005	25FEB05:15:27:00	25FEB05:16:33:00
25	31	02/25/2005	25FEB05:16:33:00	25FEB05:20:00:00

CONTSTRAINTS AND ASSUMPTIONS:

Under the existing facts, we assumed the locations could only remain open and operational if at least two employees had to be able to perform work duties (i.e., at work that day and not on break) at any given time. Thus, an employee would only be able to go on break if three or more employees were scheduled when a break was required.

IMPLEMENTATION:

We needed to develop a piece of SAS code which would assess when an employee was

"punched-in" and working and whether or not they were able to take their state mandated breaks (i.e., two additional employees were working at that time). Additionally, we needed to limit the computational time in order to constrain cost to our client.

Table 2 below provides sample code implementing *The Scheduler* macro. The first data step (lines 3-9) generates a list of date-time combinations (containing the day, month, year, and clock time). This set is computed with respect to the user designated number of time intervals, length of time, as well as a starting date/time (in this case 193,152 time intervals, each 15 minutes long beginning at midnight on September 1, 2004, see line 5). With these given parameters the end date becomes March 5, 2010. The user sets these parameters depending on the number of segments desired.[2] Although more time segments will result in more precision, the tradeoff is an increase in SAS processing time as well as hard drive space usage. The user will have to determine what makes the most sense given his or her particular circumstances. In order to save processing time, each location was analyzed separately.

Table 2. *The Scheduler* Macro Sample Code

```
1   %MACRO MASSREADIN(store_no2);
2   (step 1 start)
3   data TIMES_&store_no2.(drop=i);
4   format TIME datetime.;
5   do i=0 to 193151;  TIME=intnx('MINUTE15','01Sep2004:00:00:00'dt,i);
6                 store_no2=&store_no2.;
7                 output times_&store_no2.;
8   end;
9   run;
10
11  (step 1 end)
12
13  data punch4_&store_no2.;
14  set punch5loc;
15  if store_no2=&store_no2.;
16  run;
17
18  (step 2 start)
19
20  data Working_&store_no2. out;
21                do pcc = 1 to ncc;
22                          set times_&store_no2. nobs=ncc point=pcc;
23                          do ppc = 1 to pcc;
24                                    set punch4_&store_no2. nobs=npc point=ppc;
25                                   if ((time1<=time & time2>time)) & store_no2=&store_no2. then do;
26                                                                                working=emplid;
27                                                                                output working_&store_no2.;
28                                                                      end;
29                                    end;
30                end;
31                stop;
32  run;
33  %MEND
```

Lines 13-16 designate the particular location we were interested in evaluating. Next, for each

[2] The number of segments was calculated by multiplying the number of days between September 1, 2004 and March 5, 2010 (2012) by the number of daily segments (4 15-minute periods per hour X 24 hours is 96).

time segment created in step one, the second data step (lines 20 - 32) evaluates each observation in the employee Punch Data and determines if the created segment is contained within the work interval. Each line of the Punch Data work interval is compared to all of the created time segments. In this example the total number of comparisons generated is equal to 193,152 (the number of segments between September 1, 2004 and March 5, 2010) multiplied by the employee records in the Punch Data. Even a small number of employee punch-in/punch-out records will result in a large number of comparisons (e.g., 10 employee records in the Punch Data would result in 193,152 X 10, or 1,931,520 comparisons). If the code determines that the created time segment is contained within the punch record, a flag is generated indicating that the employee was working during that particular interval, denoted with the 'working' variable in line 26. In this case, the working variable was an employee ID, which was used to track working segments, as shown below in Table 3:

Table 3. Example of Output Created By *The Scheduler*

	TIME	store_no2	emplid
105	25FEB05:13:45:00	550	18
106	25FEB05:13:45:00	550	22
107	25FEB05:14:00:00	550	9
108	25FEB05:14:00:00	550	15
109	25FEB05:14:00:00	550	18
110	25FEB05:14:00:00	550	21
111	25FEB05:14:00:00	550	22
112	25FEB05:14:15:00	550	9
113	25FEB05:14:15:00	550	15
114	25FEB05:14:15:00	550	18
115	25FEB05:14:15:00	550	21
116	25FEB05:14:15:00	550	22
117	25FEB05:14:30:00	550	9
118	25FEB05:14:30:00	550	15
119	25FEB05:14:30:00	550	18
120	25FEB05:14:30:00	550	21
121	25FEB05:14:30:00	550	22
122	25FEB05:14:45:00	550	9
123	25FEB05:14:45:00	550	15
124	25FEB05:14:45:00	550	18
125	25FEB05:14:45:00	550	21
126	25FEB05:14:45:00	550	22
127	25FEB05:15:00:00	550	9
128	25FEB05:15:00:00	550	15
129	25FEB05:15:00:00	550	18

The final steps include eliminating all 15 minute intervals where no employees were working. These intervals indicate when the work location was closed. Next, the "working" variable was

sorted by time and transposed to show how many employees were working at each particular 15 minute interval. The output appears as follows in Table 4:

Table 4. Example of Transposed Data

	store_no	TIME	workingemp_1	workingemp_2	workingemp_3	workingemp_4	workingemp_5
43	550	25FEB05:12:30:00	9	15	18	22	.
44	550	25FEB05:12:45:00	9	15	18	22	.
45	550	25FEB05:13:00:00	9	15	18	22	.
46	550	25FEB05:13:15:00	9	15	18	22	.
47	550	25FEB05:13:30:00	9	15	18	22	.
48	550	25FEB05:13:45:00	9	15	18	22	.
49	550	25FEB05:14:00:00	9	15	18	21	22
50	550	25FEB05:14:15:00	9	15	18	21	22
51	550	25FEB05:14:30:00	9	15	18	21	22
52	550	25FEB05:14:45:00	9	15	18	21	22
53	550	25FEB05:15:00:00	9	15	18	21	22
54	550	25FEB05:15:15:00	9	15	18	21	22
55	550	25FEB05:15:30:00	9	15	18	21	22
56	550	25FEB05:15:45:00	9	15	18	21	22
57	550	25FEB05:16:00:00	9	15	18	21	22
58	550	25FEB05:16:15:00	9	15	18	21	22
59	550	25FEB05:16:30:00	9	15	18	21	22
60	550	25FEB05:16:45:00	9	15	18	21	22
61	550	25FEB05:17:00:00	18	21	22	.	.
62	550	25FEB05:17:15:00	18	21	22	.	.
63	550	25FEB05:17:30:00	18	21	22	.	.
64	550	25FEB05:17:45:00	18	21	22	.	.
65	550	25FEB05:18:00:00	18	21	22	.	.
66	550	25FEB05:18:15:00	21	22	.	.	.
67	550	25FEB05:18:30:00	21	22	.	.	.
68	550	25FEB05:18:45:00	21	22	.	.	.
69	550	25FEB05:19:00:00	22	.	.	.	.
70	550	25FEB05:19:15:00	22	.	.	.	.
71	550	25FEB05:19:30:00	22	.	.	.	.
72	550	25FEB05:19:45:00	22	.	.	.	.
73	550	25FEB05:20:00:00	22	.	.	.	.
74	550	25FEB05:20:15:00	22	.	.	.	.
75	550	25FEB05:20:30:00	22	.	.	.	.

Once transposed, the data were collapsed, a count of instances where workers were unable to take a meal break was generated. Each location took between 30 and 60 minutes of computational time to loop through the code, depending on the number of observations in the Punch Data. This number is dependent on many factors including the computer hardware, the size of the created time segment dataset, and other computer processes running in the background.

OTHER USES

The Scheduler was created to help evaluate whether employees were able to take state-mandated meal breaks, but there are myriad other opportunities to use this code. Examples include allowing supervisors to determine if schedules efficiently allocate employee resources and

assisting construction foreman to create a schedule that visualizes the presence of contractors in a logical sequence while eliminating overlap and/or ensuring cushions of time between contractors. Finally, this code can be used to match items to a non-uniform period.

Using this code and with appropriate available data, a supervisor could potentially aggregate transaction dollars into the same time intervals as those used for the employee schedules, enabling any employer to evaluate whether its resources are being allocated efficiently. Another example where *The Scheduler* could prove useful involves the sequencing and scheduling of work trades in the construction of a high rise building. In this example, many different workers/tradespeople might need to occupy the same area to perform work. The work of plumbers, electricians, city inspectors, insulators, etc. generally should be scheduled in a way to minimize overlapping workers and thus minimize inefficiencies and interference. Using *The Scheduler*, a foreman could evaluate and determine, floor by floor, when and where these tradesmen should be scheduled so they may work harmoniously.

Finally, *The Scheduler* might be used to match items to non-uniform periods. For instance, employee expense receipts are typically reimbursed on employee paychecks. However, in instances where all employees are not paid on the same day, these receipts may not easily be assigned a standard week end date. A different week end date may be needed to match pay with expenses during that week. However, using this code, given just a list of employee week end dates, *The Scheduler* could be used to match each employee's week end date with his or her individual expense receipts.

CONCLUSION

The Scheduler is an innovative tool that maps linear data (i.e., time stamps) to an intuitive 3-dimensional representation. This transformation allows the user to identify relationships, conflicts, gaps, etc. that were not readily apparent in the data's native form. Even though the code was created to specifically tackle the issue of determining potential employee meal break violations, there are several other applications for this tool.

ACKNOWLEDGMENTS

The author would like to acknowledge Joseph Krock, Andrew Cook, and Susan Burkhauser for their helpful comments.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ariel Kumpinsky
The Claro Group, LLC
350 S. Grand Ave
Suite 2350
Los Angeles, CA 90071
Work Phone: 213-784-3083
E-mail: akumpinsky@theclarogroup.com
Web: www.theclarogroup.com