Online Supplement for "Accurate ED Wait Time Prediction"

I. R Code for obtaining Q-Lasso Predictor Variables

The method to obtain the Q-Lasso Predictor Variables in the R statistical software will be shown here. Assume that the time stamp data is in the format:

Table A1: Sample of raw time stamp data

Arrival_time	Mode_of_arrival	Triage	Triage_time	Treatment_time
2013-08-02 10:23:00	Walk-in	4	2013-08-02 10:45:00	2013-08-02 10:52:00
2013-08-02 10:42:00	Walk-in	3	2013-08-02 10:58:00	2013-08-02 11:31:00
2013-08-02 10:47:00	Ambulance	5	2013-08-02 11:10:00	2013-08-02 11:58:00

1) Obtaining count data

Suppose that the time stamp data is stored in a dataframe called **data** and we are interested in getting the candidate predictor variables for specific time periods stored in a vector called **periods**. Suppose this vector contains the time stamps of 2 periods we are interested in: '2013-08-02 10:30:00' and '2013-08-11:00:00' and we want the counts of (1) the number of patients who have arrived and not yet been Triaged in total and (2) the number of patients who have been Triaged and not received treatment in total and from each triage level in 3,4,5 (3) the number of patients who arrived in an ambulance who has registered but not yet been treated.
The function countNumber will perform a count of (1):

```
# Function: countNumber
# Usage: countNumber(periods,timeStamps1,timeStamps2)
# ------------------------------------------------------------
# Returns the count of the number of patients who at period time
# are between two stages in the ED given by timeStamps1 and timeStamps2

countNumber <- function(time,timeStamps1,timeStamps2){
  indices = which((time >= timeStamps1) & (time < timeStamps2))
  return(length(indices))
}
```

Example:

```
> countNumber(periods[1],data$Arrival_time,data$Triage_time)
```

```
> 1
> countNumber(periods[2],data$Arrival_time,data$Triage_time)
> 1
```

The second function countNumPerTriageTotal will perform a count of (2):

```
# Function: countNumPerTriageTotal
# Usage:
countNumPerTriageTotal(periods,timeStamps1,timeStamps2,triageLevel)
# ------------------------------------------------------------------------
# Returns the count of patients at each triage level and the total number
# of patients who at period time are between two stages in the ED given
# timeStamps1 and timeStamps2

countNumberPerTriageTotal <-
function(time,timeStamps1,timeStamps2,triage){
  indices = which((time >= timeStamps1) & (time < timeStamps2))
  output_vec = rep(0,6)
  if(length(indices) == 0) return(output_vec)
  output_vec[1] = sum(condData[indices] == 1)
  output_vec[2] = sum(condData[indices] == 2)
  output_vec[3] = sum(condData[indices] == 3)
  output_vec[4] = sum(condData[indices] == 4)
  output_vec[5] = sum(condData[indices] == 5)
  output_vec[6] = length(indices) # total
  return(output_vec)
}
```

Example:

```
> countNumberPerTriageTotal(periods[2],data$Triage_time,
data$Treatment_time)
> 0 0 1 0 0 1
```

The third function countNumberCond will perform the count for (3):

```
# Function: countNumberCond
# Usage:
countNumberCond(periods,timeStamps1,timeStamps2,Mode_of_arrival,
”Ambulance”)
# ------------------------------------------------------------------------
```

```
# Same function as countNumber but with added condition check based on
condData being part of condToMeet

countCond<- function(time,timeStamps1,timeStamps2,condData,condToMeet){
  indices = which((time >= timeStamps1) & (time < timeStamps2))
  return(sum(condData[indices] %in% condToMeet))
}
```

Example:

```
> countCond(periods[2],data$Arrival_time,data$Treatment_time,
Mode_of_arrival,"Ambulance")
> 1
```

2) Getting from count data to combined model predictors

Suppose after performing calculations above, one could get to a per period data, dpp, set of the form:

Table A2: Sample of per-period count data

period	num_wait_triage	fast_track_on	num_doctors
2013-08-02 10:30:00	1	1	2
2013-08-02 11:00:00	1	1	1

Then to get a Q-Lasso predictor variable that is given by:
(Fast Track On) x (Number of Patients waiting for triage) x (Number of Doctors)$^{-1}$
one could simply run the R command:

```
> dpp$fast_track_on * dpp$num_wait_triage * dpp$num_doctors
> 2 1
```

II. Predictor variables with nonzero coefficients assigned by Q-Lasso

SMMC:

- Periods of the day: 70 out of 143 periods
- Indicator variable that takes value 1 when Fast Track is in Operation
- Number of doctors/ Number of nurses
- Day of the week: Tuesday, Saturday and Sunday
- Rolling average time to treatment of all patients who started treatment in the last 2 hours

- Flu cases in previous week
- Temperature in the last 2 hours
- Variables motivated by fluid models: 35 out of 408 variables

Hospital 1:

- Periods of the day: 74 out of 143 periods
- Number of doctors
- Day of the week: Monday, Wednesday, Thursday, Friday
- Rolling average time to treatment of all patients who started treatment in the last 4 hours
- Variables motivated by fluid models: 10 out of 96 variables

Hospital 2:

- Periods of the day: 4 out of 143 periods
- Day of the week: Monday
- Temperature in the last 2 hours
- Rolling average time to treatment of all patients who started treatment in the last 6 hours
- Variables motivated by fluid models: 10 out of 120 variables

Hospital 3:

- Periods of the day: 5 out of 143 periods
- Rolling average time to treatment of all patients who started treatment in the last 6 hours
- Variables motivated by fluid models: 8 out of 120 variables

III. Statistical test for out-of-sample test set prediction comparisons

As the models, with different candidate predictor variables, being compared in Table 5 of the main text are "nested," the squared errors generated from the test set prediction are correlated with each other, violating the iid assumption required by a normal t-test. Therefore, the authors used the following modified scheme – which accounts for the covariance of the predictions – to establish statistical significance:

Suppose we are comparing two models M1 and M2 with different candidate predictor variables. The first 80% of data (training data) is used to fit the prediction functions P1 and P2, for M1 and M2 respectively. The last 20% of data (test data) is partitioned into 100 non-overlapping subsets. P1 and P2 are used to generate predictions for all the subsets, and the mean squared error for each

subset is found. Suppose the MSEs obtained for M1 are $E_{1,1}$,$E_{1,2}$ … $E_{1,100}$ and those for M2 are $E_{2,1}$,$E_{2,2}$ … $E_{2,100}$.

Let:

$$\bar{E}_A = \frac{1}{100}\sum_{i=1}^{100} E_{A,i} \text{ for } A = 1,2$$

$$VAR_A = \frac{1}{99}\sum_{i=1}^{100}\left(E_{A,i} - \bar{E}_A\right)^2 \text{for } A = 1,2$$

$$COV_{12} = \frac{1}{99}\sum_{i=1}^{100}\left(E_{1,i} - \bar{E}_1\right)\left(E_{2,i} - \bar{E}_2\right)$$

The test statistic to use is then calculated by:

$$TS = \frac{\bar{E}_2 - \bar{E}_1}{\frac{(VAR_1 + VAR_2 - 2COV_{12})^{0.5}}{100^{0.5}}}$$

which was then compared to a t-distribution with 99 degrees of freedom.

As a robustness check, the authors used a Diebold-Mariano test (DM test) for comparing prediction accuracy of forecasts; the results were consistent with the modified t-test above. The procedure was carried out via command `dm.test()` from the R package `forecast`. However, it is important to note that the DM test may not be valid for comparing nested models.

IV. Preliminary Analysis of Initial LWBS Rates

As a first-pass analysis of the decrease in left-without-being-seen (LWBS) rates observed post-implementation, the authors ran a stepwise forward regression on data spanning December 11, 2014 to January 12, 2015 to identify (prior to implementation) which variables were most explanatory in predicting whether or not a patient would LWBS. Predictive variables included, for example, the age of the patient, as well as indicator variables *1{patient is of low-acuity}* and *1{patient was triaged between 6am and 10am}*. The authors then modeled whether or not a patient who visited between January 13 and March 15, 2015 left without being seen as:

$$P(LWBS_i \mid \boldsymbol{X}_i, TR_i) = \rho_0 + \rho_1 TR_i + \rho_{\mathbf{X}} \boldsymbol{X}_i$$

where $\mathbf{X}_i$ comprises the LWBS-predictive variables mentioned above, $LWBS_i$ takes value 1 if patient i left without being seen (0 otherwise) and TR_i takes value 1 if patient i visited after February 13, 2015 and was exposed to the *T*riage *R*oom wait time system (0 otherwise). The estimated value of ρ_1 was -0.121, with a p-value of 0.03.

The authors also performed a robustness check by fitting a probit regression on the same predictors as in the above equation. In the probit regression, ρ_1 had an estimated value of -0.009, with a p-value of 0.04. Thus, both tests *suggest* that the implemented triage wait time prediction system may have reduced the LWBS rate at SMMC. However, much more data must be collected, and randomized experimentation conducted, before concluding that this is indeed the case.

V. Re-Estimating the Coefficients of the Prediction Function

The prediction accuracy of Q-Lasso can be marginally improved by updating the prediction function using more recent data. The authors experimented with varying the frequency at which the prediction function is "re-estimated," as well as the amount of training data used to do so. Figure A1 shows the relationship between the test set MSE, the amount of data in the training set used for each re-estimation (t, measured in number of days) and the length of the interval between re-estimations (r, measured in number of days). The graph on the left panel shows the diminishing benefit from increasing t, for r = 1 and r = 7 days. The graphs for other values of t are similar in shape. Roughly, the optimal amount of data to use for re-estimating the prediction function is about 150 days. Further increasing t beyond 150 days does not improve the MSE. The graph on the right panel shows test set MSE as a function of r, when t = 150. It shows that the prediction function should be updated as frequently as possible, because the test set MSE increases monotonically with r. The graphs for other values of t are similar.

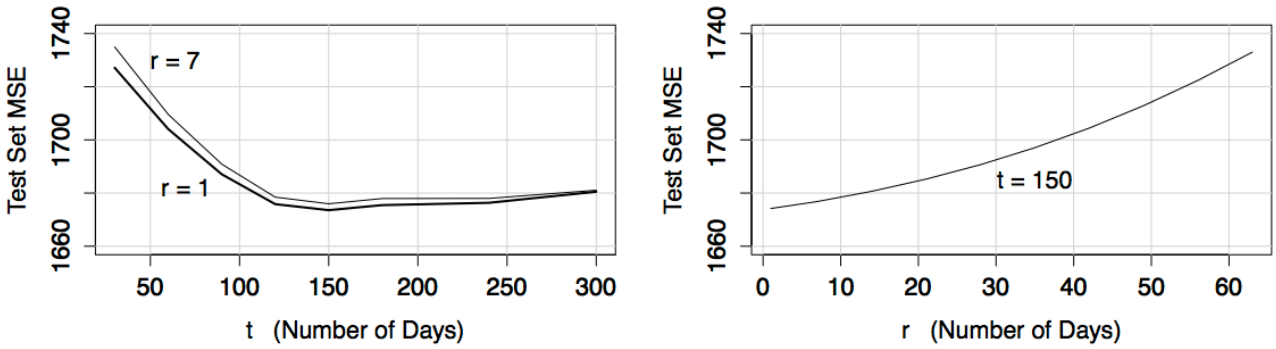

Figure A1 Test set MSE as a function of amount of training data used for re-estimation, *t* (left) and length of the interval between re-estimations, *r* (right).

Running Q-Lasso frequently may not be possible for some hospitals as the Lasso step recommended in §3.4 requires manual tuning of the parameter λ. On the other hand, running a simple linear regression to update the prediction function can be easily automated. Thus the authors explored an implementation shortcut. First, the authors ran Q-Lasso on the training set data and identified the subset of predictor variables with non-zero coefficients. Then, in the test data, the authors re-estimated the prediction function every r days using a simple linear regression on this subset of candidate predictor variables. The authors found that, for all combinations of t and r, re-estimating the prediction function using exclusively Q-Lasso every time improved the test set MSE by between 0.3% to 1.9%, with the majority by less than 1%, compared to using the implementation shortcut with simple linear regression. Thus, a hospital need not use the full Q-Lasso for re-estimating the prediction function each time.

The results presented above suggest three takeaways. First, Q-Lasso may not require large amounts of historical data (150 days is ideal for SMMC). Second, the prediction function should be re-estimated as frequently as possible – every day, ideally. Third, Q-Lasso, which requires manual tuning, need not be run to re-estimate the prediction function each day. Rather, a hospital can easily automate the retraining process by running Q-Lasso every few weeks, and identifying a subset of important candidate predictor variables S. Daily, in between Q-Lasso runs, the hospital can update the prediction function automatically using simple linear regression with only the variables in S, and new data.