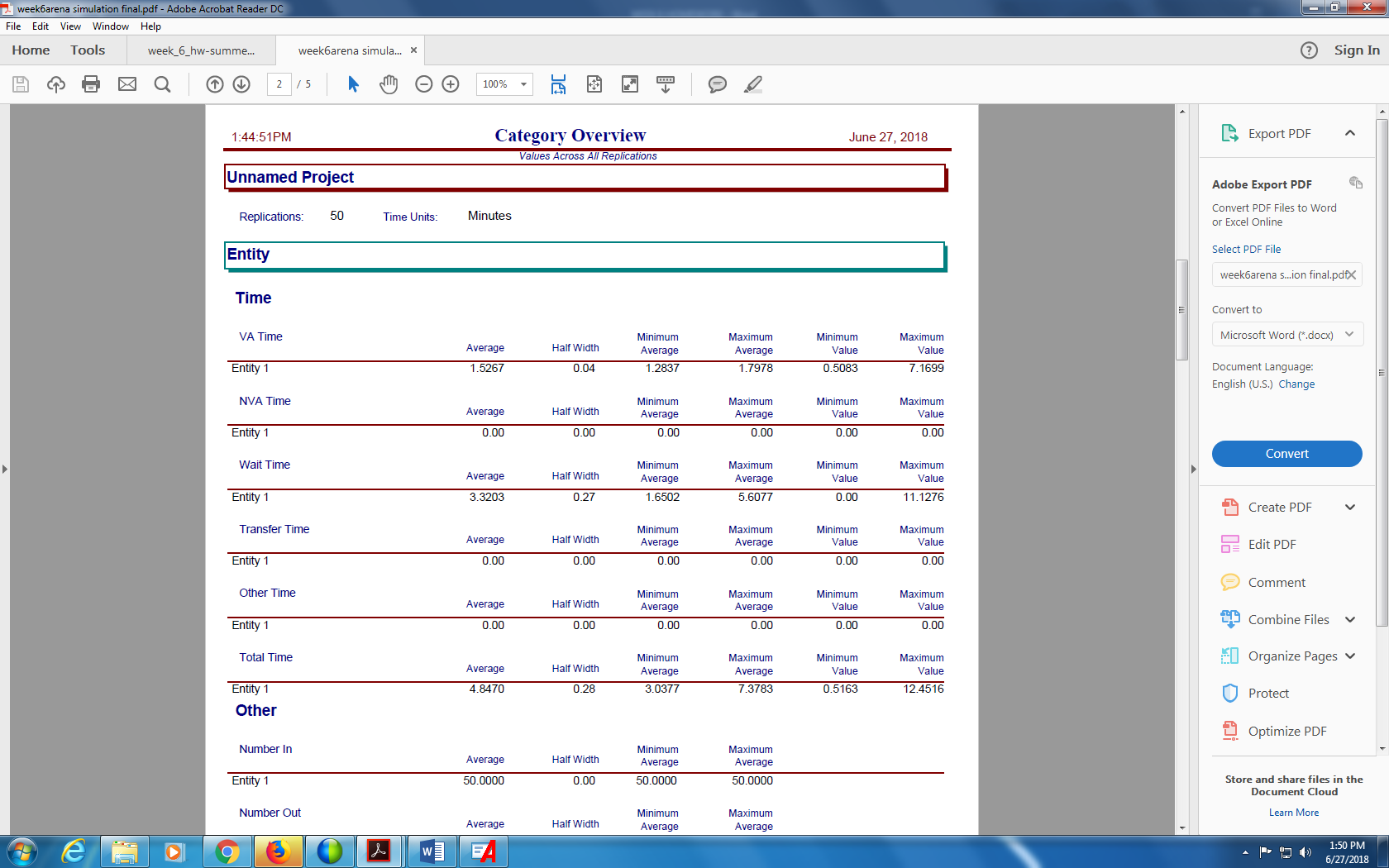
WEEK 6 HOMEWORK

Chunxia Cao

Question 13.2

*In this problem you, can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ1 = 5 per minute (i.e., mean interarrival rate µ1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate µ2 = 0.75 minutesAfter that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (time is uniformly distributed between 0.5 minutes and 1 minute). Use the Arena software (PC users) or Python with SimPy (PC or Mac users) to build a simulation of the system, and then vary the number of ID/boarding-pass checkers and personal-check queues to determine how many are needed to keep average wait times below 15 minutes. [If you’re using SimPy, or if you have access to a non-student version of Arena, you can use λ1 = 50 to simulate a busier airport.]*

Use the Arena software to do simulation. Passengers arrive according to a Poisson distribution with λ1 = 5 per minute to the ID/boarding-pass check queue, where the server have exponential service time with mean rate µ = 0.75 minutes. Set up 3 personal scanners (uniformly distributed between 0.5 minutes and 1 minute). The simulation result is as following:



Given the average wait time is 3.3203, 3 personal scanners would be enough for the airport.

***Question 14.1***

*The breast cancer data set breast-cancer-wisconsin.data.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29 ) has missing values.*

Check missing values:

> for (i in 2:11) {

+ print (paste0("v",i))

+ print(table(databc[,i]))

+ }

[1] "v2"

1 2 3 4 5 6 7 8 9 10

145 50 108 80 130 34 23 46 14 69

[1] "v3"

1 2 3 4 5 6 7 8 9 10

384 45 52 40 30 27 19 29 6 67

[1] "v4"

1 2 3 4 5 6 7 8 9 10

353 59 56 44 34 30 30 28 7 58

[1] "v5"

1 2 3 4 5 6 7 8 9 10

407 58 58 33 23 22 13 25 5 55

[1] "v6"

1 2 3 4 5 6 7 8 9 10

47 386 72 48 39 41 12 21 2 31

[1] "v7"

? 1 10 2 3 4 5 6 7 8 9

16 402 132 30 28 19 30 4 8 21 9

[1] "v8"

1 2 3 4 5 6 7 8 9 10

152 166 165 40 34 10 73 28 11 20

[1] "v9"

1 2 3 4 5 6 7 8 9 10

443 36 44 18 19 22 16 24 16 61

[1] "v10"

1 2 3 4 5 6 7 8 10

579 35 33 12 6 3 9 8 14

[1] "v11"

2 4

458 241

It can be seen that only V7 has missing values; to show observations with missing data:

> databc[which(databc$V7=="?"),]

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11

24 1057013 8 4 5 1 2 ? 7 3 1 4

41 1096800 6 6 6 9 6 ? 7 8 1 2

140 1183246 1 1 1 1 1 ? 2 1 1 2

146 1184840 1 1 3 1 2 ? 2 1 1 2

159 1193683 1 1 2 1 3 ? 1 1 1 2

165 1197510 5 1 1 1 2 ? 3 1 1 2

236 1241232 3 1 4 1 2 ? 3 1 1 2

250 169356 3 1 1 1 2 ? 3 1 1 2

276 432809 3 1 3 1 2 ? 2 1 1 2

293 563649 8 8 8 1 2 ? 6 10 1 4

295 606140 1 1 1 1 2 ? 2 1 1 2

298 61634 5 4 3 1 2 ? 2 3 1 2

316 704168 4 6 5 6 7 ? 4 9 1 2

322 733639 3 1 1 1 2 ? 3 1 1 2

412 1238464 1 1 1 1 1 ? 2 1 1 2

618 1057067 1 1 1 1 1 ? 1 1 1 2

*1. Use the mean/mode imputation method to impute values for the missing data.*

First, create function to find the mode of a vector, then find the find the mode and mean of v7 and impute v7 for observations with missing data:

> getmode=function(v){

+ uniqv=unique(v)

+ uniqv[which.max(tabulate(match(v,uniqv)))]

+ }

> (mode\_v7=as.numeric(getmode(databc[-missing,"V7"])))

[1] 1

> (mean\_v7=mean(as.numeric(getmode(databc[-missing,"V7"]))))

[1] 1

> modeimp=databc

> modeimp[missing,]$V7=mode\_v7

> modeimp$V7=as.integer(modeimp$V7)

Check if the missing data has been replaced successfully:

> modeimp[which(modeimp$V7=="?"),]

[1] V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11

<0 rows> (or 0-length row.names)

It can be seen that missing data of V7 has been replaced.

*2. Use regression to impute values for the missing data.*

Set up the dataset exclude the missing values and establish linear regression model use all other factors except v7.

> datare=databc[-missing,2:10]

> datare$V7=as.integer(datare$V7)

> modere=lm(V7~., data=datare)

> summary(modere)

Call:

lm(formula = V7 ~ ., data = datare)

Residuals:

Min 1Q Median 3Q Max

-9.7316 -0.9426 -0.3002 0.6725 8.6998

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.616652 0.194975 -3.163 0.00163 \*\*

V2 0.230156 0.041691 5.521 4.83e-08 \*\*\*

V3 -0.067980 0.076170 -0.892 0.37246

V4 0.340442 0.073420 4.637 4.25e-06 \*\*\*

V5 0.339705 0.045919 7.398 4.13e-13 \*\*\*

V6 0.090392 0.062541 1.445 0.14883

V8 0.320577 0.059047 5.429 7.91e-08 \*\*\*

V9 0.007293 0.044486 0.164 0.86983

V10 -0.075230 0.059331 -1.268 0.20524

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.274 on 674 degrees of freedom

Multiple R-squared: 0.615, Adjusted R-squared: 0.6104

F-statistic: 134.6 on 8 and 674 DF, p-value: < 2.2e-16

Since not all the predictors are significant, use the step() function to select a reduced model, then get the final model :

> moderef=lm(formula = V7 ~ V2 + V4 + V5 + V8, data = datare)

> summary(moderef)

Call:

lm(formula = V7 ~ V2 + V4 + V5 + V8, data = datare)

Residuals:

Min 1Q Median 3Q Max

-9.8115 -0.9531 -0.3111 0.6678 8.6889

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.53601 0.17514 -3.060 0.0023 \*\*

V2 0.22617 0.04121 5.488 5.75e-08 \*\*\*

V4 0.31729 0.05086 6.239 7.76e-10 \*\*\*

V5 0.33227 0.04431 7.499 2.03e-13 \*\*\*

V8 0.32378 0.05606 5.775 1.17e-08 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.274 on 678 degrees of freedom

Multiple R-squared: 0.6129, Adjusted R-squared: 0.6107

F-statistic: 268.4 on 4 and 678 DF, p-value: < 2.2e-16

get prediction for missing v7 values, impute v7 for the observations with missing data for v7 and round the values since the original are all integers:

> (V7hat=predict(moderef,newdata=databc[missing,]))

24 41 140 146 159 165 236 250 276 293 295

5.4585352 7.9816106 0.9872832 1.6218560 0.9807851 2.2157441 2.7152652 1.7634059 2.0741942 6.0866099 0.9872832

298 316 322 412 618

2.5265324 5.2438347 1.7634059 0.9872832 0.6634986

> datareimp=databc

> datareimp[missing,]$V7=round(V7hat)

> datareimp$V7=as.integer(datareimp$V7)

Check if the missing data has been replaced successfully, and make sure no v7 values are out of the original range:

> datareimp [which(datareimp$V7=="?"),]

[1] V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11

<0 rows> (or 0-length row.names)

> datareimp$V7[datareimp$V7>10]=10

> datareimp$V7[datareimp$V7<1]=1

> datareimp$V7

[1] 1 10 2 4 1 10 10 1 1 1 1 1 3 3 9 1 1 1 10 1 10 7 1 5 1 7 1 1 1 1 1 1 5 1 1 1

[37] 1 1 10 7 8 3 10 1 1 1 9 1 1 8 3 4 5 8 8 5 6 1 10 2 3 2 8 2 1 2 1 10 9 1 1 2

[73] 1 10 4 2 1 1 3 1 1 1 1 2 9 4 8 10 1 1 1 1 1 1 1 1 1 1 6 10 5 5 1 3 1 3 10 10[109] 1 9 2 9 10 8 3 5 2 10 3 2 1 2 10 10 7 1 10 1 10 1 1 1 10 1 1 2 1 1 1 1 1 1 5 5

[145] 1 2 8 2 1 10 1 10 5 3 1 10 1 1 1 10 10 1 1 3 2 2 10 1 1 1 1 1 1 10 10 10 1 1 1 10

[181] 1 1 1 10 10 1 8 10 8 1 8 10 1 1 1 1 7 1 1 1 10 10 1 1 1 10 5 1 1 1 10 8 1 10 10 5

[217] 1 1 4 1 1 10 5 8 10 1 10 5 1 10 7 8 1 10 1 3 10 2 9 10 2 1 1 5 1 2 10 9 1 2 1 10

[253] 10 10 8 10 1 1 1 8 10 10 10 10 3 1 10 10 4 1 10 1 10 4 1 2 1 1 1 7 1 1 10 10 10 10 10 1

[289] 5 10 1 1 6 10 1 10 5 3 1 10 4 1 10 1 10 10 1 1 3 5 1 1 1 1 1 5 10 8 1 5 10 2 1 10

[325] 1 1 10 1 4 10 8 1 1 10 10 1 10 1 1 10 10 1 1 1 10 1 1 1 1 8 1 1 3 10 1 1 3 10 4 7

[361] 10 10 3 3 1 1 10 10 1 1 1 1 1 1 1 1 1 1 1 1 1 10 1 1 1 1 10 1 1 2 1 10 1 1 1 1

[397] 1 1 1 1 9 1 1 4 1 1 1 1 2 1 1 1 4 1 10 3 10 1 2 1 3 10 1 1 1 10 1 2 1 1 1 1

[433] 1 1 8 10 1 1 1 1 10 4 3 2 1 1 1 1 1 10 1 1 1 10 1 6 10 3 1 1 1 5 1 1 1 4 10 1

[469] 1 1 1 1 1 1 1 1 1 1 1 10 1 1 5 10 1 3 1 10 3 4 1 10 1 10 5 1 1 1 1 1 1 1 1 1

[505] 1 1 5 4 1 1 1 1 1 1 10 10 1 1 1 10 1 1 5 10 1 1 1 1 1 1 10 1 1 1 1 1 1 1 1 1

[541] 2 1 1 1 1 1 10 1 1 5 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 10 1 3 10 5 10 10 1 1 2 1

[577] 1 1 1 1 1 10 10 1 1 1 10 1 3 1 1 10 10 1 10 1 1 1 1 1 1 1 1 1 10 8 1 1 10 1 10 2

[613] 10 1 1 1 1 1 1 1 1 2 1 1 1 4 6 5 1 1 1 1 1 3 1 1 1 2 1 1 1 1 1 1 1 1 1 1

[649] 2 1 4 1 1 1 1 1 1 1 10 1 1 1 1 1 1 1 1 1 1 5 8 1 1 1 1 1 1 1 1 1 10 10 1 1

[685] 1 1 1 1 1 1 1 5 1 1 2 1 3 4 5

It can be seen that the imputation is success.

*3. Use regression with perturbation to impute values for the missing data.*

Add error terms to the previous predicted value, impute the missing values and round the values since the original are all integers, make sure no v7 values are out of the original range and check if the missing data has been replaced successfully:

> (v7pert=rnorm(nrow(databc[missing,]),V7hat,sd(V7hat)))

[1] 4.1419532 3.1677681 1.5180863 1.0501916 2.9656700 4.2917880 5.9509085 3.3212302 3.8794323

[10] 5.4397249 4.1141002 5.8300905 3.7955115 -0.1163035 1.6836149 3.1094576

> datapertimp=databc

> datapertimp[missing,]$V7=round(v7pert)

> datapertimp$V7=as.integer(datapertimp$V7)

> datapertimp$V7[datapertimp$V7>10]=10

> datapertimp$V7[datapertimp$V7<1]=1

> datapertimp$V7

[1] 1 10 2 4 1 10 10 1 1 1 1 1 3 3 9 1 1 1 10 1 10 7 1 4 1 7 1 1 1 1 1 1 5 1 1 1

[37] 1 1 10 7 3 3 10 1 1 1 9 1 1 8 3 4 5 8 8 5 6 1 10 2 3 2 8 2 1 2 1 10 9 1 1 2

[73] 1 10 4 2 1 1 3 1 1 1 1 2 9 4 8 10 1 1 1 1 1 1 1 1 1 1 6 10 5 5 1 3 1 3 10 10

[109] 1 9 2 9 10 8 3 5 2 10 3 2 1 2 10 10 7 1 10 1 10 1 1 1 10 1 1 2 1 1 1 2 1 1 5 5

[145] 1 1 8 2 1 10 1 10 5 3 1 10 1 1 3 10 10 1 1 3 4 2 10 1 1 1 1 1 1 10 10 10 1 1 1 10

[181] 1 1 1 10 10 1 8 10 8 1 8 10 1 1 1 1 7 1 1 1 10 10 1 1 1 10 5 1 1 1 10 8 1 10 10 5

[217] 1 1 4 1 1 10 5 8 10 1 10 5 1 10 7 8 1 10 1 6 10 2 9 10 2 1 1 5 1 2 10 9 1 3 1 10

[253] 10 10 8 10 1 1 1 8 10 10 10 10 3 1 10 10 4 1 10 1 10 4 1 4 1 1 1 7 1 1 10 10 10 10 10 1

[289] 5 10 1 1 5 10 4 10 5 6 1 10 4 1 10 1 10 10 1 1 3 5 1 1 1 1 1 4 10 8 1 5 10 1 1 10

[325] 1 1 10 1 4 10 8 1 1 10 10 1 10 1 1 10 10 1 1 1 10 1 1 1 1 8 1 1 3 10 1 1 3 10 4 7

[361] 10 10 3 3 1 1 10 10 1 1 1 1 1 1 1 1 1 1 1 1 1 10 1 1 1 1 10 1 1 2 1 10 1 1 1 1

[397] 1 1 1 1 9 1 1 4 1 1 1 1 2 1 1 2 4 1 10 3 10 1 2 1 3 10 1 1 1 10 1 2 1 1 1 1

[433] 1 1 8 10 1 1 1 1 10 4 3 2 1 1 1 1 1 10 1 1 1 10 1 6 10 3 1 1 1 5 1 1 1 4 10 10

[469] 1 1 1 1 1 1 1 1 1 1 1 10 1 1 5 10 1 3 1 10 3 4 1 10 1 10 5 1 1 1 1 1 1 1 1 1

[505] 1 1 5 4 1 1 1 1 1 1 10 10 1 1 1 10 1 1 5 10 1 1 1 1 1 1 10 1 1 1 1 1 1 1 1 1

[541] 2 1 1 1 1 1 10 1 1 5 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 10 1 3 10 5 10 10 1 1 2 1

[577] 1 1 1 1 1 10 10 1 1 1 10 1 3 1 1 10 10 1 10 1 1 1 1 1 1 1 1 1 10 8 1 1 10 1 10 2

[613] 10 1 1 1 1 3 1 1 1 2 1 1 1 4 6 5 1 1 1 1 1 3 1 1 1 2 1 1 1 1 1 1 1 1 1 1

[649] 2 1 4 1 1 1 1 1 1 1 10 1 1 1 1 1 1 1 1 1 1 5 8 1 1 1 1 1 1 1 1 1 10 10 1 1

[685] 1 1 1 1 1 1 1 5 1 1 2 1 3 4 5

> datapertimp[which(datapertimp$V7=="?"),]

[1] V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11

<0 rows> (or 0-length row.names)

*4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using :*

(*1) the data sets from questions 1,2,3;*

Running SVM for the data sets from questions 1,2,3 get the accuracy are:

accuracy on test data = 0.9619048 on Mean/mode Imputation data

accuracy on test data = 0.9714286 on regression imputation data

accuracy on test data = 0.9809524 on regression with perturbation imputation data.

*(2) the data that remains after data points with missing values are removed;*

accuracy on test data = 0.9901961 on the dataset with missing values are removed.

*(3) the data set when a binary variable is introduced to indicate missing values*

Running SVM for the data sets from questions 1,2,3 get the accuracy are:

accuracy on test data = 0.9333333

It seems that remove the missing values get better accuracy in test data.

(details shown in r markdown file)

***Question 15.1***

*Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?*

In the hospital, almost all employee work “shift”. The director should schedule employees for weekly "shifts" (seven works days plus night duty shifts) to minimize payroll costs while meeting varying demand for each day of the week, optionally taking into account employee seniority and preferences