HOMEWORK 9. ISYE 6501

Guillermo de la Hera Casado

October 21th, 2019

Fractional factorial design

First we will set up the seed and run the FrF2 function. As parameters we will pass:

- the number of ficticious houses to show (16) as *nruns*
- the number of factors to consider (10) as *nfactors*. All the factors are binomial (large yard vs small yard, solar roof vs no solar roof, etc)
- the names of the factors as *factor.names*

The point of using this technique is to avoid having to test all possible combinations of the factors per house and still achieve good representativity. Otherwise we would need to explore 2^(10) options!

```
set.seed(101) # Set Seed so that same sample can be reproduced in future also
fr <- FrF2(nruns=16, nfactors=10,
           factor.names = c("large yard","solar roof","centric","bright",
                            "energy efficient","big kitchen","brand new",
                            "price increased last month",
                            "price increased last 12 months","without tenants"))
fr
```

```
##    large.yard solar.roof centric bright energy.efficient big.kitchen
## 1          -1         -1      -1      1                 1           1
## 2           1          1       1      1                 1           1
## 3           1         -1       1      1                -1           1
## 4          -1          1       1     -1                -1          -1
## 5          -1         -1      -1     -1                 1           1
## 6           1         -1      -1      1                -1          -1
## 7           1         -1       1     -1                -1           1
## 8          -1          1      -1     -1                -1           1
## 9           1          1       1     -1                 1           1
## 10         -1          1       1      1                -1          -1
## 11          1          1      -1      1                 1          -1
## 12         -1         -1       1      1                 1          -1
## 13         -1         -1       1     -1                 1          -1
## 14          1         -1      -1     -1                -1          -1
## 15          1          1      -1     -1                 1          -1
## 16         -1          1      -1      1                -1           1
##    brand.new price.increased.last.month price.increased.last.12.months
## 1          1                         -1                              1
## 2          1                          1                              1
## 3         -1                          1                             -1
## 4          1                          1                             -1
## 5          1                          1                             -1
## 6          1                          1                              1
## 7         -1                         -1                              1
## 8         -1                          1                              1
## 9          1                         -1                             -1
## 10         1                         -1                              1
```

```
## 11        -1                          1                          -1
## 12        -1                         -1                          -1
## 13        -1                          1                           1
## 14         1                         -1                          -1
## 15        -1                         -1                           1
## 16        -1                         -1                          -1
##    without.tenants
## 1               -1
## 2                1
## 3               -1
## 4                1
## 5                1
## 6                1
## 7                1
## 8               -1
## 9               -1
## 10              -1
## 11              -1
## 12               1
## 13              -1
## 14              -1
## 15               1
## 16               1
## class=design, type= FrF2
```

By looking at the result we can see, for each house, which factors are taken. Let's explain the first two houses as an example:

- House 1. Marked with value=1, we have: *bright, energy efficient, big kitchen, brand new, house price increased in the last 12 months*. Those will be the 5 factored considered for the house.
- House 2. All factors are marked with value=1, therefore we take all of them for this sample.

Real life applications of probability distributions

Plese see below an example for each type:

- Binomial: Probability of five students passing the exam, out of a sample of 20 students. A student can either fail/pass an exam (binomial)
- Geometric: Probability of seeing four students passing the exam before the first student failure.
- Poisson: Probability of selling 5 houses in a day by a real state company, being the average 2 houses per day.
- Exponential: Assuming that the total number of house sold per day follows Poisson, the exponential distribution would model the inter-selling time between one house and the next one
- Weibull: How long for a new aircraft engine to fail after it has been installed.

Airport Security System with ARENA

I have decided to use ARENA for this assignment, with student version. Based on the parameters specified in the homework, I managed both to get rid of the error warning about 150 instances and also keep the avg total time $<$ 15min with configuration as per *Figure 1.* This is the overall picture of the airport simulation:

- One module with type=create, that serves 5 passengers per minute on average according to a Poisson distribution. Therefore the entity produced is: *passenger*.
- One module with type=process, that uses **a set of four servers (resources)** to process ID/boarding pass checks with EXP(0.75min). It selects the resource from the set to be used for each passenger at

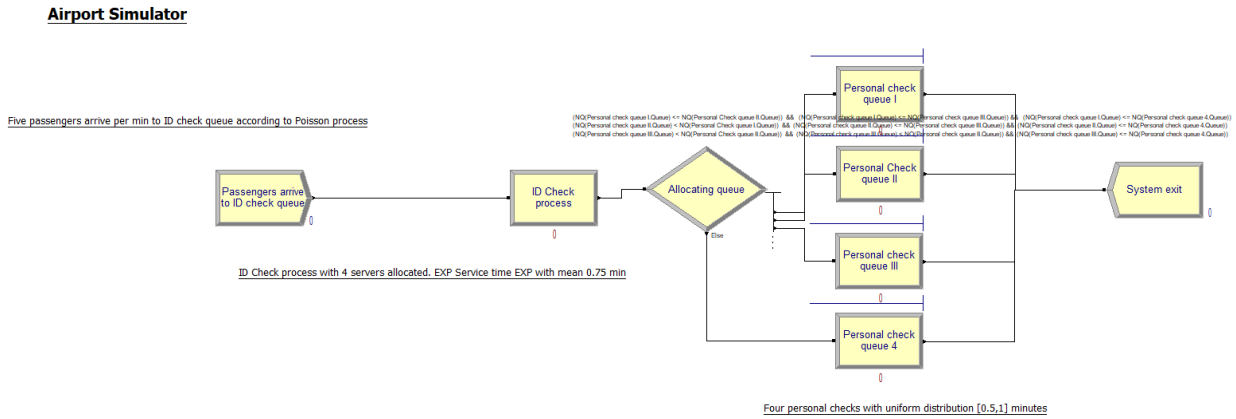

Figure 1: Overall airport simulation process.

random to ensure all the resources are used at the same rate. It follows the pattern: *Seize, Delay, Release* as the passenger is assigned to a resource, there is a processing delay and after that it gets automatically released.

- One module with type=decide. **It will allocate each passenger to the shortest personal check queue**. This was acomplished by using Expression Builder and allocating a condition for each one of the queues, as seen in the picture. Else, the passenger goes to queue 4. As a learning, it was key to check how to use the *NQ()* function, as that provides the current number of people waiting on each queue.
- Four modules with type=process representing each queue. All of them follow uniform distribution [0.5, 1] min. To avoid the common mistake pointed out during 21-10-2019 office hours, each queue has it's own resource (Personal Check 1, Personal Check 2...). All follow the pattern: *Seize, Delay, Release.*
- One module with type=dispose where the process ends.

In order to run the simulation and achieve good representativity, I changed few parameters in the Run setup. The details can be seen in *Figure 2*

- Number of replications = 100
- Each replication runs for 24 hours. We also assume that the airport is open 24h per day.
- The base time unit has been changed to minutes in order to make the report easier to read.

In *Figure 3* we can see the average total time spent for each passenger in the process, an average of 5.56 min. As a breakdown, we can see that most of the time the passenger is waiting (avg 4.06 min) and then the time spent interacting with resources (Value Added time) is very little: 1.49 min.

With regards to the actual number of passsengers seized by resource, we can also see the breakdown in *Figure 4*. It's interesting to see how the ID-check servers distribute their workload almost evenly due to random allocation. We can also see that the first personal-check queues are more used than the last ones. However, all of them are needed in order to stop the "student trial version" error from showing up and achieving the objective.

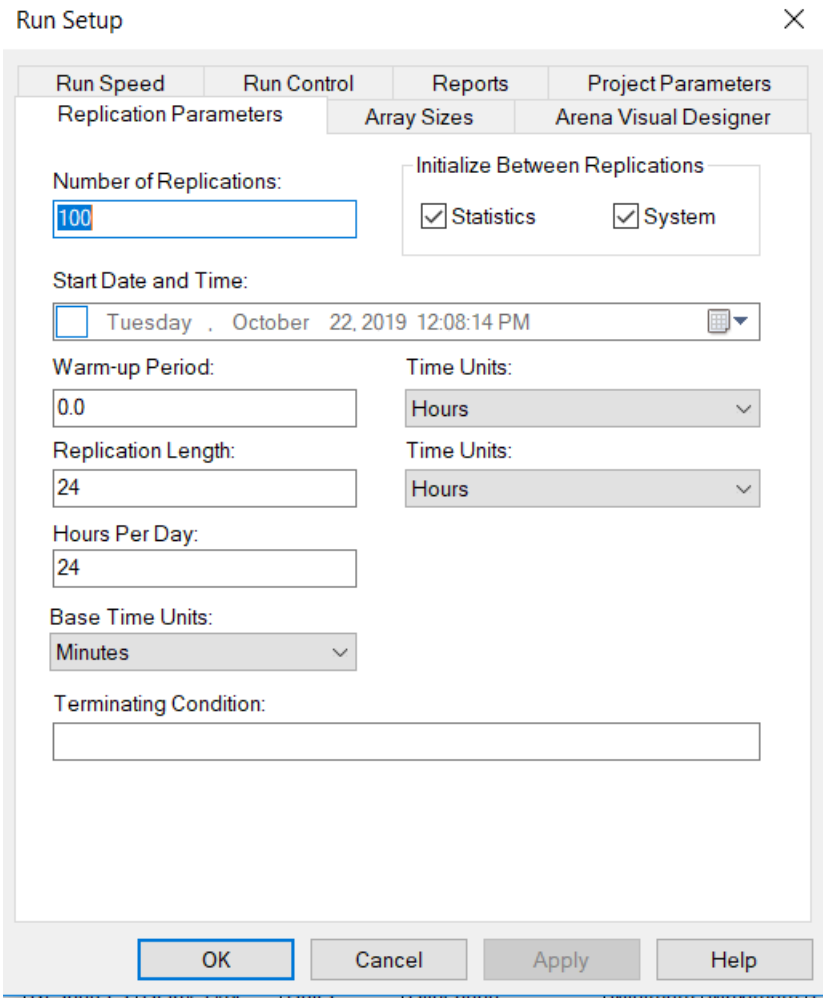

Figure 2: Run Setup.

Unnamed Project

Replications: 100 Time Units: Minutes

Entity

Time

VA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Passenger	1.4994	0.00	1.4802	1.5211	0.5014	10.4390

NVA Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Passenger	0.00	0.00	0.00	0.00	0.00	0.00

Wait Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Passenger	4.0642	0.21	2.0042	8.3407	0.00	21.7854

Transfer Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Passenger	0.00	0.00	0.00	0.00	0.00	0.00

Other Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Passenger	0.00	0.00	0.00	0.00	0.00	0.00

Total Time	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
Passenger	5.5636	0.21	3.4906	9.8404	0.5170	25.1764

Figure 3: Overall time per passenger.

Replications: 100 Time Units: Minutes

Resource

Usage

Total Number Seized	Average	Half Width	Minimum Average	Maximum Average
ID server 1	1802.62	9.01	1682.00	1938.00
ID server 2	1791.40	9.26	1665.00	1913.00
ID server 3	1797.15	8.95	1698.00	1929.00
ID server 4	1796.80	7.40	1706.00	1895.00
Personal Check 1	1915.66	1.79	1893.00	1933.00
Personal check II	1883.88	2.12	1862.00	1909.00
Personal check III	1786.30	5.36	1720.00	1853.00
Personal check queue IV	1591.07	11.55	1425.00	1756.00

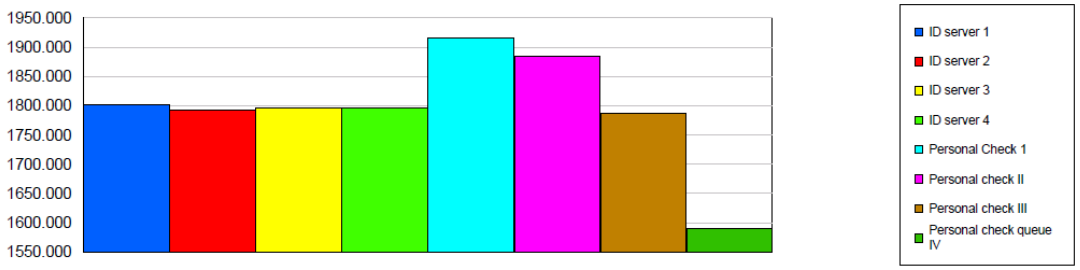

Figure 4: Total passengers seized by resource.