AnalyticsModeling_HW9

Fall 2024

Question 12.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a design of experiments approach would be appropriate.

At work, we run an application that users can log into and interact with. A DOE approach would be appropriate if we have a few different versions of the app's homepage and we want to know which version gets users to interact with the app the most (i.e., for the longest amount of time).

Question 12.2

To determine the value of 10 different yes/no features to the market value of a house (large yard, solar roof, etc.), a real estate agent plans to survey 50 potential buyers, showing a fictitious house with different combinations of features. To reduce the survey size, the agent wants to show just 16 fictitious houses. Use R's FrF2 function (in the FrF2 package) to find a fractional factorial design for this experiment: what set of features should each of the 16 fictitious houses have? Note: the output of FrF2 is "1" (include) or "-1" (don't include) for each feature.

The fractional factorial design can be used to test a subset of feature combinations for each house, so that the real estate agent can show a representative and balanced variety of homes without having to show every available home.

```
# list of house features
features <- c('large yard', 'solar roof', 'garage', 'basement', 'attic', 'pool', 'garden', 'num floors', 'sqft', 'new applia
nces')

# Fractional Factorial Design of 16 houses and 10 features
FrF2(nruns = 16
     , nfactors = 10
     , factor.names=features
     )
```

```
##    large.yard solar.roof garage basement attic pool garden num.floors sqft
## 1           1          1     -1        1     1   -1     -1          1   -1
## 2          -1          1     -1       -1    -1    1     -1          1    1
## 3           1         -1     -1        1    -1   -1      1          1    1
## 4           1          1      1        1     1    1      1          1    1
## 5          -1          1     -1        1    -1    1     -1         -1   -1
## 6          -1         -1      1       -1     1   -1     -1          1    1
## 7           1          1      1       -1     1    1      1         -1   -1
## 8           1         -1      1        1    -1    1     -1          1   -1
## 9          -1          1      1       -1    -1   -1      1          1   -1
## 10         -1          1      1        1    -1    1      1         -1    1
## 11         -1         -1      1        1     1   -1     -1         -1   -1
## 12         -1         -1     -1       -1     1    1      1          1   -1
## 13          1          1     -1       -1     1   -1     -1         -1    1
## 14         -1         -1     -1        1     1    1      1         -1    1
## 15          1         -1      1       -1    -1    1     -1         -1    1
## 16          1         -1     -1       -1    -1   -1      1         -1   -1
##    new.appliances
## 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
```

Question 13.1

For each of the following distributions, give an example of data that you would expect to follow this distribution

1. Binomial: any data that has a binary response follows the Binomial distribution, such as if a patient is healthy or sick or if a job-seeker is hired or rejected.
2. Geometric: say I want to calculate how many dice rolls it takes for me to land on 1. This follows a Geometric distribution.
3. Poisson: say I have data for car accidents at a certain intersection. The number of accidents that occur at this intersection in a week follows a Poisson distribution.
4. Exponential: the time between bus arrivals at a bus stop follows an Exponential distribution.
5. Weibull: the time it takes until a machine fails follows a Weibull distribution.

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 □1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate □2 = 0.75 minutes. [Hint: model them as one block that has more than one resource.] After 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.]

To do this simulation with Simpy, I had to rely on some outside resources to teach me how to use this package properly. References are below!

- Simpy CarWash Example: http://simpy.readthedocs.io/en/latest/examples/carwash.html
- Simpy Movie Example: https://simpy.readthedocs.io/en/latest/examples/movie_renege.html
- Simpy Documentation: https://simpy.readthedocs.io/en/latest/simpy_intro/installation.html

Simpy is a discrete-event simulation library. In this problem, I had to simulate an airport system in which:

- Passengers arrive according to `X=POI(mean = 5)` per minute or an arrival rate of `X=EXP(mean = 0.2)` minutes
- Passengers queue at the ID/Boarding-pass line, where several servers will assist them when it's their turn
- Passengers are served with a service time of `Y=EXP(mean = 0.75)` minutes
- Passengers queue at the personal-check line, where they will go through scanner
- Passengers are scanned according to `Z=UNIFORM(.5, 1.0)`

The goal is to figure out how many servers and personal queues are needed to keep average wait times to less than 15 minutes.

First, I had to establish the random variables given in the prompt and start with some random number of employees to check IDs/boarding passes and some number of scanners to do the personal checks.

```
n_servers = 2
n_scanners = 4
served_time = .75
scanned_time = [.5,1]
arrival_time = .02
sim_time = 500
```

Next, I had to create the airport object and define the processes involved with each step of this simulation.

```
class Airport(object):
    def __init__(self, env, num_servers, num_scanners, serve_time, scan_time):
        self.env = env
        self.IDcheck = simpy.Resource(env, num_servers)
        self.Scancheck = simpy.Resource(env, num_scanners )
        self.serve_time = serve_time
        self.scan_time = scan_time

    def ID_check(self, passenger):
        rand_ID_time = random.expovariate(served_time)
        yield self.env.timeout(rand_ID_time)

    def scan_check(self, passenger):
        yield self.env.timeout(random.uniform(self.scan_time[0],self.scan_time[1]))


def security(env, passenger, airport):
    #passenger arrives
    arrival = env.now

    #goes through ID check and scan
    with airport.IDcheck.request() as request:
        yield request
        yield env.process(airport.ID_check(passenger))

    with airport.Scancheck.request() as request:
        yield request
        yield env.process(airport.scan_check(passenger))
```

From this point, I could run the simulation and calculate the average wait time for each person to complete the security process from start to finish. For the sake of reproducibility, I set the seed to a consistent value.

```
def run_sim(num_servers= n_servers, num_scanners = n_scanners):

    random.seed(47)

  # Run the simulation
    env = simpy.Environment()
    env.process(run_simulation(env,num_servers,num_scanners,served_time, scanned_time))
    env.run(until=sim_time)

    return average_time(wait_times)
```

It's perhaps not too surprising that the simulation concluded that more ID/boarding pass servers are needed than scanners, since scanner processing time is quicker. But after trying a variety of server and scanner numbers, the simulation indicated that 10 servers and 6 scanners are the minimum number required to keep processing times below 15 minutes total.

```
 The average wait time with 10 servers and 6 scanners is below 15 minutes!
11.588881934073045
```