Homework 9

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.

I work for a company that develops reporting software. We can use a design of experiments approach when doing A|B testing of out products. Whenever we are developing new features, we create multiple options for the feature. We than A|B test these features to see which ones are going to have the most positive impacts for our business.

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.

First, we can import the required library.

```
library(FrF2)
```

```
## Loading required package: DoE.base
```

```
## Loading required package: grid
```

```
## Loading required package: conf.design
```

```
##
## Attaching package: 'DoE.base'
```

```
## The following objects are masked from 'package:stats':
##
##     aov, lm
```

```
## The following object is masked from 'package:graphics':
##
##     plot.design
```

```
## The following object is masked from 'package:base':
##
##     lengths
```

Below I set up two iterations of FrF2 with 10 yes/no factors. After I did this I printed the results and saw that in each iteration that results are slightly different which is showing the randomness when setting up these two experiments.

```
one <- FrF2(16,10, factor.names=list(color=c("light","dark"),flood_zone=c("yes","no"),
     age=c("old","new"),state=c("new","aged"), pool=c("yes", "no"), garage=c("yes","no"),heat=c("yes","no"
),
     AC=c("yes","no"),yard=c("yes","no"), dog_friendly=c("yes", "no")))
two <-  FrF2(16,10, factor.names=list(color=c("light","dark"),flood_zone=c("yes","no"),
     age=c("old","new"),state=c("new","aged"), pool=c("yes", "no"), garage=c("yes","no"),heat=c("yes","no"
),
     AC=c("yes","no"),yard=c("yes","no"), dog_friendly=c("yes", "no")))
```

Print the results of one

```
one
```

	color	flood_zone	age	state	pool	garage	heat	AC	yard
	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>
1	light	yes	new	aged	no	yes	yes	yes	yes
2	dark	no	old	new	no	yes	yes	yes	no
3	dark	no	old	aged	no	yes	yes	no	yes
4	dark	yes	new	new	yes	no	yes	yes	no
5	light	yes	new	new	no	yes	yes	no	no
6	light	no	old	new	yes	no	yes	no	no
7	light	no	old	aged	yes	no	yes	yes	yes
8	light	no	new	aged	yes	yes	no	yes	no
9	light	yes	old	new	no	no	no	no	yes
10	dark	yes	old	aged	yes	yes	no	no	no

1-10 of 16 rows | 1-10 of 11 columns Previous **1** 2 Next

Print the results of one

```
two
```

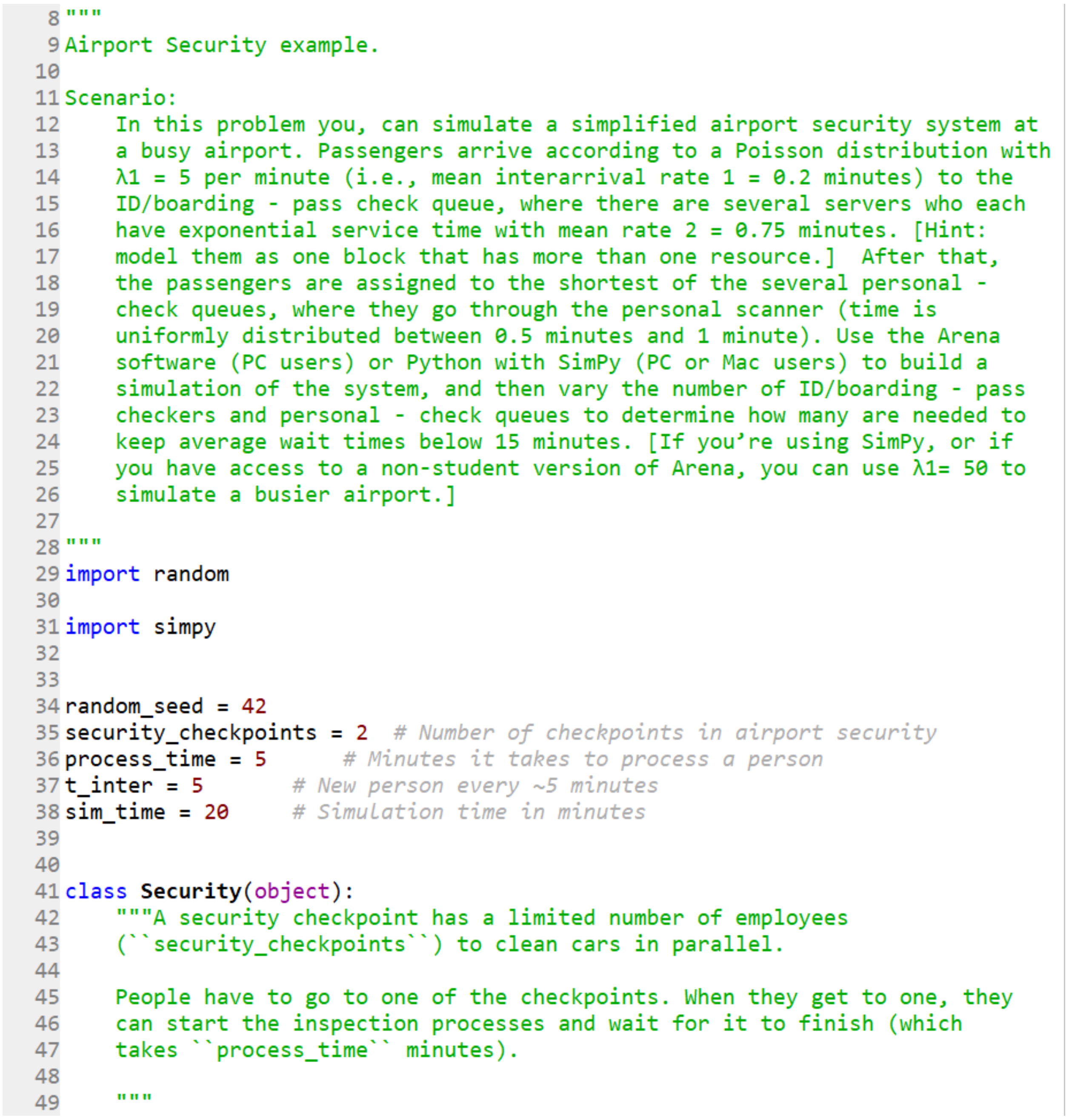
	color	flood_zone	age	state	pool	garage	heat	AC	yard
	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>
1	dark	yes	old	new	yes	yes	no	yes	yes
2	light	yes	old	new	no	no	no	no	yes
3	dark	yes	old	aged	yes	yes	no	no	no
4	light	no	old	new	yes	no	yes	no	no
5	dark	yes	new	aged	yes	no	yes	no	yes
6	light	no	old	aged	yes	no	yes	yes	yes
7	light	yes	new	aged	no	yes	yes	yes	yes
8	dark	yes	new	new	yes	no	yes	yes	no
9	light	no	new	aged	yes	yes	no	yes	no
10	dark	no	old	new	no	yes	yes	yes	no

1-10 of 16 rows | 1-10 of 11 columns Previous **1** 2 Next

Question 13.1 For each of the following distributions, give an example of data that you would expect to follow this distribution (besides the examples already discussed in class). a.Binomial
b.Geometric
c.Poisson
d.Exponential e.Weibull

Binomial- A chart showing people's IQ will tend to have a binomial distributions. Geometric - If you ask the number of people outside a voting booth who they voted for, the geometric distribution would tell you the number of one votes until you got a vote for another party. Poisson - Restaurants can use a poisson distribution to determine the number of customers they will receive over an hour long period. Exponential - This can be used to determine the average amount of time that someone spends with a customer. Weibull - Can be used if you have data in two different data sets and you want to model how closely their distribution match.

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 ??1 = 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 ??1= 50 to simulate a busier airport.]



```
"""
Airport Security example.

Scenario:
    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.]

"""
import random

import simpy


random_seed = 42
security_checkpoints = 2  # Number of checkpoints in airport security
process_time = 5      # Minutes it takes to process a person
t_inter = 5       # New person every ~5 minutes
sim_time = 20     # Simulation time in minutes


class Security(object):
    """A security checkpoint has a limited number of employees
    (``security_checkpoints``) to clean cars in parallel.

    People have to go to one of the checkpoints. When they get to one, they
    can start the inspection processes and wait for it to finish (which
    takes ``process_time`` minutes).

    """
```

```python
def person(env, name, ap):
    """The person (each person has a ``name``) arrives at airport security
    (``ap``) and requests an inspection.

    It then starts the inspection process, waits for it to finish and
    leaves to never come back ...

    """
    print('%s arrives at airport security at %.2f.' % (name, env.now))
    with ap.checkpoint.request() as request:
        yield request

        print('%s enters the security inspection station at %.2f.' %
              (name, env.now))
        yield env.process(ap.checkpoint(name))

        print('%s leaves the airport at %.2f.' % (name, env.now))


def setup(env, security_checkpoints, process_time, t_inter):
    """Create a airport, a number of initial people and keep creating people
    approx. every ``t_inter`` minutes."""
    # Create airport security
    airport_security = Security(env, security_checkpoints, process_time)

    # Create 4 initial people
    for i in range(4):
        env.process(person(env, 'Person %d' % i, airport_security))

    # Create more people while the simulation is running
    while True:
        yield env.timeout(random.randint(t_inter - 2, t_inter + 2))
        i += 1
        env.process(car(env, 'Person %d' % i, airport_security))


# Setup and start the simulation
print('airport_security')

random.seed(random_seed)  # This helps reproducing the results

# Create an environment and start the setup process
env = simpy.Environment()
env.process(setup(env, security_checkpoints, process_time, t_inter))

# Execute!
env.run(until=sim_time)
```