HW9

2022-10-25

Question 12.1

I think one design of experiments would be trying to determine which combination of food storage methods results in the longest-lasting meat. Factors that could vary are: store in an original or sealed container, precooked or raw, and store in the fridge crisper drawer or another.

Question 12.2

Using the FrF2() function in R, I created the below fractional factorial design to see how to best varying the features across the survey. I set nruns to 16 (for 16 fictitious houses) and 10 for nfactors (for 10 different yes/no features).

```
rm(list = ls())
cat("\014")
```

```r
set.seed(1)

#install.packages("FrF2")

library(FrF2)
```

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

## Loading required package: grid

## Loading required package: conf.design

## Registered S3 method overwritten by 'DoE.base':
##   method           from
##   factorize.factor 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
```

```r
features = c(
  "A",
  "B",
  "C",
  "D",
  "E",
  "F",
  "G",
  "H",
  "I",
  "J",
  "K"
)

FrF2(nruns = 16, factor.names = features)
```

```
##     A  B  C  D  E  F  G  H  I  J  K
## 1  -1 -1 -1  1  1  1  1 -1 -1  1  1
## 2   1  1 -1 -1  1 -1 -1 -1 -1  1  1
```

```
## 3  -1  1  1 -1 -1 -1  1  1 -1  1 -1
## 4  -1 -1 -1 -1  1  1  1  1  1 -1 -1
## 5   1 -1 -1 -1 -1 -1  1 -1  1  1 -1
## 6   1 -1  1  1 -1  1 -1  1 -1  1 -1
## 7   1  1 -1  1  1 -1 -1  1  1 -1 -1
## 8  -1  1 -1 -1 -1  1 -1  1 -1 -1  1
## 9  -1 -1  1  1  1 -1 -1 -1 -1 -1 -1
## 10 -1 -1  1 -1  1 -1 -1  1  1  1  1
## 11 -1  1 -1  1 -1  1 -1 -1  1  1 -1
## 12  1 -1 -1  1 -1 -1  1  1 -1 -1  1
## 13  1 -1  1 -1 -1  1 -1 -1  1 -1  1
## 14 -1  1  1  1 -1 -1  1 -1  1 -1  1
## 15  1  1  1  1  1  1  1  1  1  1  1
## 16  1  1  1 -1  1  1  1 -1 -1 -1 -1
## class=design, type= FrF2
```

Question 13.1

a. Binomial: The answers to whether a sports team won their game
b. Geometric: The answers to a sports team's games played before winning a game
c. Poisson: The number of people who shop on Black Friday
d. Exponential: The time between shoppers lining up at a store for Black Friday
e. Weibull: The time before a computer motherboard fails

Question 13.2

I set up a SimPy simulation using a max time period of 10 hours and a total number of passengers of 500. Using these figures, I was able to determine that having 24 ID checks and 15 personal scanners results in an average wait time of 14.7 minutes.

Things that could be done to improve the results: 1. I was unable to figure out how to assign the passenger to the shortest personal scan queue. This would probably bring down the average times.

main.py

```python
import simpy as sy
import random

random.seed(1)
num_servers = 24
num_scanners = 15
runTime = 600
total_time = 0
total_passengers = 1
num_passengers = 500


class Airport(object):
    """ID/boarding pass check queue and security scanning
    """

    def __init__(self, env, num_servers, num_scanners):
        self.env = env
        self.server = sy.Resource(env, num_servers)
        self.scanner = sy.Resource(env, num_scanners)

    def IDcheck(self, passenger):
        """The ID check process. It takes a "person", checks their ID,then
        passes him/her to the next step."""
        ID_service_time = random.expovariate(.75)
        yield self.env.timeout(ID_service_time)

    def scan(self, passenger):
        """Security Scan. Passenergs are sent to shortest queue then scanned"""
        scan_time = random.uniform(0.5, 1)
        yield self.env.timeout(scan_time)


def Passenger(env, number, s):
    """Passnegers arrive, to the first available server for the ID check then
    sent to the first available scanner.
    """
    global total_time  # global average wait time
    global total_passengers
    Arrivaltime = env.now
    with s.server.request() as request:
        yield request
        yield env.process(s.IDcheck(number))

    with s.scanner.request() as request:
        yield request
        yield env.process(s.scan(number))

    pass_time = env.now - Arrivaltime
    total_time = total_time + pass_time
    total_passengers = total_passengers + 1


```

```
54  def setup(env, num, security):
55      arrival_int = random.expovariate(5)
56      yield env.timeout(arrival_int)
57      env.process(Passenger(env, num, security))
58
59
60  # Setup and start the simulation
61  print('Airport Security')
62  random.seed(1)
63  env = sy.Environment()
64  ap = Airport(env, num_servers, num_scanners)
65  # Start processes and run
66  for i in range(0, num_passengers):
67      env.process(setup(env, i, ap))
68
69  env.run()
70
71  avg_time = total_time / total_passengers
72  print("avg_time")
73  print(avg_time)
74
```