Homework 9: Experimental Design, Probability Distributions, and Simulation Modeling

Georgia Institute of Technology, Business Analytics

Introduction to Analytics Modeling

Professor Joel Sokol

October 22, 2025

Files submitted: homework9_answers.pdf (this doc), homework9.R

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.

As a senior software engineer onboarding with Deel under the PGWP program, I recently faced a challenge in optimizing the documentation package submitted to IRCC for remote work approval. The goal was to determine which combination of supporting documents most effectively accelerates approval timelines while maintaining compliance. A design of experiments (DOE) approach would be appropriate here to systematically test the impact of different document configurations such as variations in employment letters, inclusion of SIN, proof of address, and contract formatting on simulated approval outcomes.

By treating each document type as a factor with binary levels (included vs. excluded), I could construct a factorial design to evaluate interactions between documents. This would help identify not only which individual documents are most influential, but also whether certain combinations produce synergistic effects. The insights could inform best practices for future submissions, reducing delays and improving onboarding efficiency for other remote workers navigating similar immigration pathways.

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.

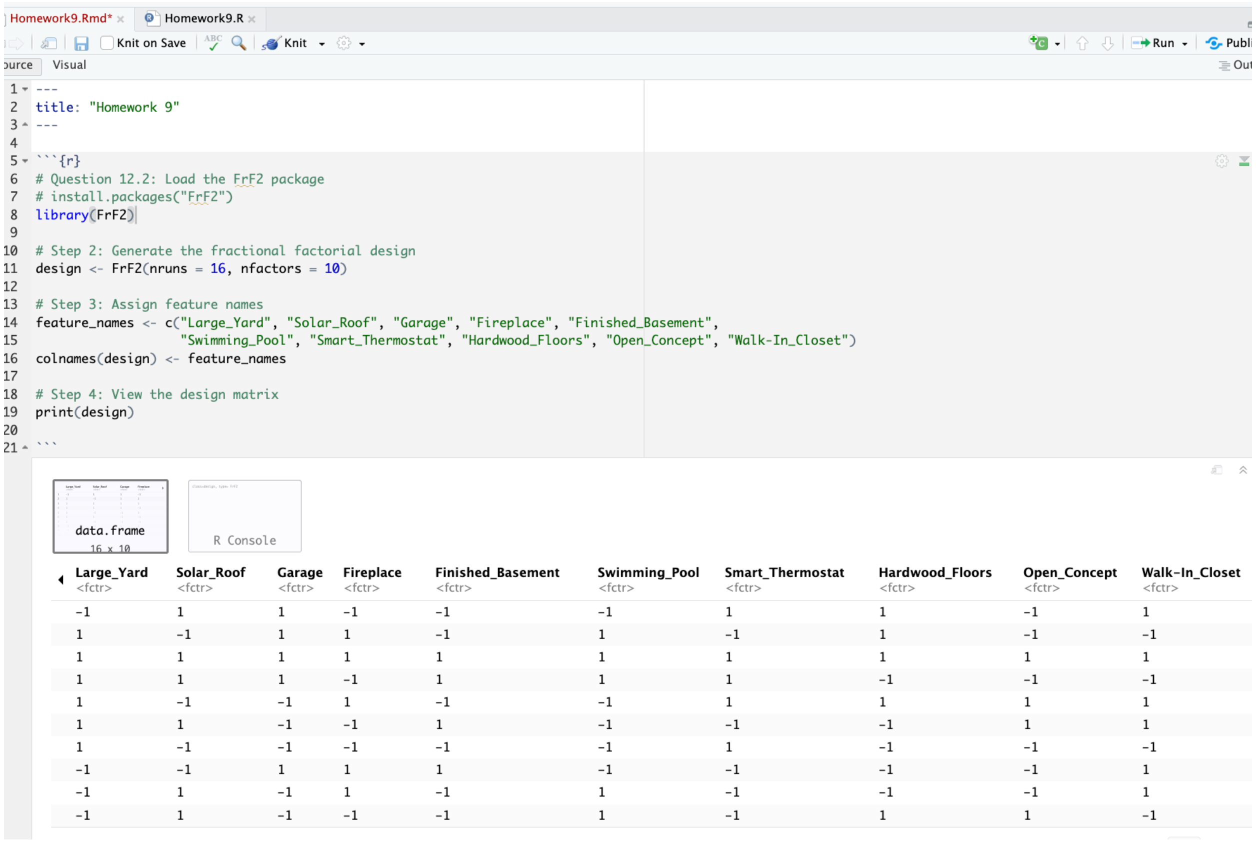
Methodology

I used R’s FrF2 function from the FrF2 package to generate a fractional factorial design with:

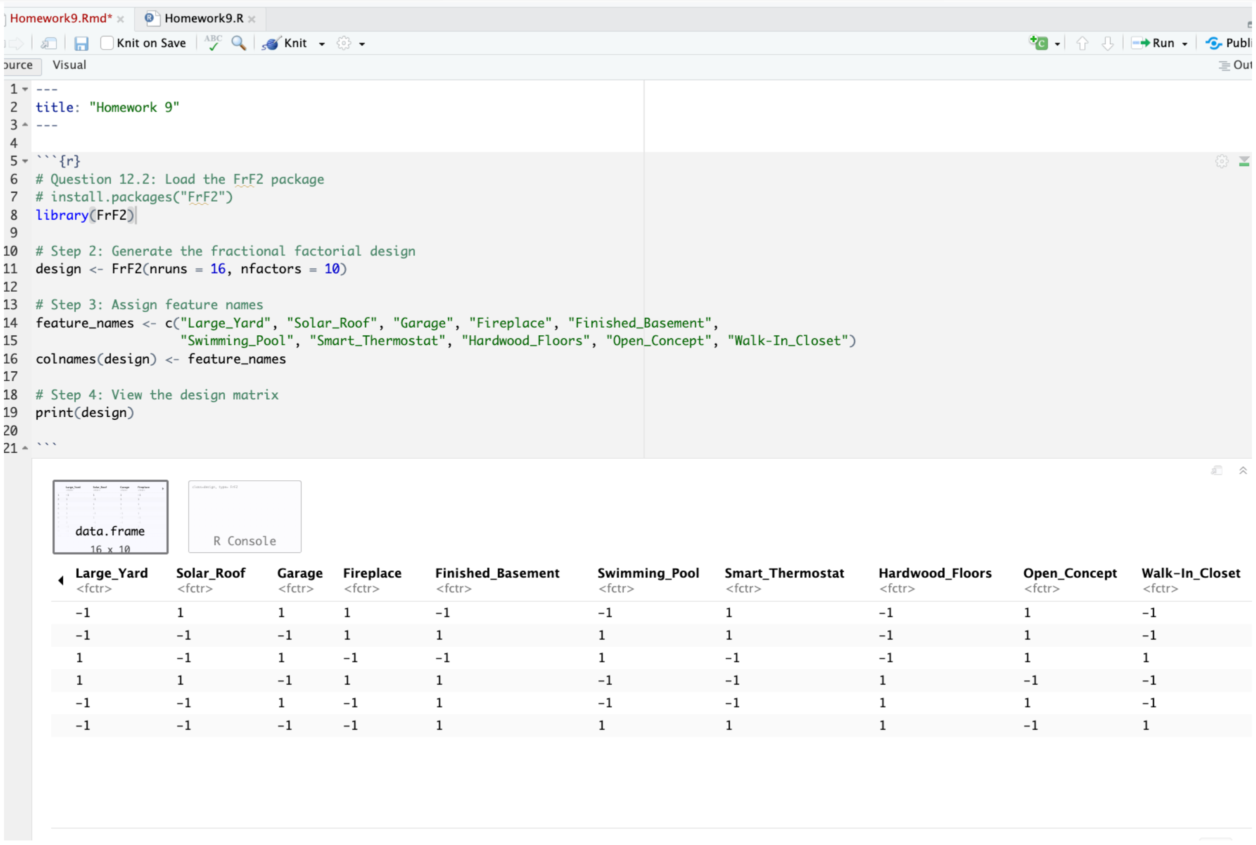
- 10 binary features
- 16 runs
- Resolution IV design

Results

The output was a 16x10 matrix with values of 1 (include feature) and -1 (exclude feature). Each row represents a fictitious house profile for survey purposes.



~~~r
---
title: "Homework 9"
---

```{r}
# Question 12.2: Load the FrF2 package
# install.packages("FrF2")
library(FrF2)

# Step 2: Generate the fractional factorial design
design <- FrF2(nruns = 16, nfactors = 10)

# Step 3: Assign feature names
feature_names <- c("Large_Yard", "Solar_Roof", "Garage", "Fireplace", "Finished_Basement",
                   "Swimming_Pool", "Smart_Thermostat", "Hardwood_Floors", "Open_Concept", "Walk-In_Closet")
colnames(design) <- feature_names

# Step 4: View the design matrix
print(design)

~~~
```

Large_Yard <fctr>	Solar_Roof <fctr>	Garage <fctr>	Fireplace <fctr>	Finished_Basement <fctr>	Swimming_Pool <fctr>	Smart_Thermostat <fctr>	Hardwood_Floors <fctr>	Open_Concept <fctr>	Walk-In_Closet <fctr>
-1	1	1	-1	-1	-1	1	1	-1	1
1	-1	1	1	-1	1	-1	1	-1	-1
1	1	1	1	1	1	1	1	1	1
1	1	1	-1	1	1	1	-1	-1	-1
1	-1	-1	1	-1	-1	1	1	1	1
1	1	-1	-1	1	-1	-1	-1	1	1
1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	1	-1	-1	-1	-1	1
-1	1	-1	1	-1	1	-1	-1	-1	1
-1	1	-1	-1	-1	1	-1	1	1	-1

Figure 1: The output matrix with 16 runs and 10 features.
```



~~~
---
title: "Homework 9"
---

```{r}
# Question 12.2: Load the FrF2 package
# install.packages("FrF2")
library(FrF2)

# Step 2: Generate the fractional factorial design
design <- FrF2(nruns = 16, nfactors = 10)

# Step 3: Assign feature names
feature_names <- c("Large_Yard", "Solar_Roof", "Garage", "Fireplace", "Finished_Basement",
                   "Swimming_Pool", "Smart_Thermostat", "Hardwood_Floors", "Open_Concept", "Walk-In_Closet")
colnames(design) <- feature_names

# Step 4: View the design matrix
print(design)

```
~~~

Large_Yard <fctr>	Solar_Roof <fctr>	Garage <fctr>	Fireplace <fctr>	Finished_Basement <fctr>	Swimming_Pool <fctr>	Smart_Thermostat <fctr>	Hardwood_Floors <fctr>	Open_Concept <fctr>	Walk-In_Closet <fctr>
-1	1	1	1	-1	-1	1	-1	1	-1
-1	-1	-1	1	1	1	1	-1	1	-1
1	-1	1	-1	-1	1	-1	-1	1	1
1	1	-1	1	1	-1	-1	1	-1	-1
-1	-1	1	-1	1	-1	-1	1	1	-1
-1	-1	-1	-1	1	1	1	1	-1	1

Figure 2: The output matrix with 16 runs and 10 features. (Contd)

Discussion of Results

This design reduces the survey burden while preserving statistical power to estimate main effects and some interactions. It's ideal for market research and avoids respondent fatigue. Ethical considerations include ensuring participants understand the fictitious nature of the profiles and that their responses are anonymized.

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

a. Binomial Distribution
Example: The number of successful login attempts out of 10 tries using two-factor authentication (success = correct credentials entered and code verified).

Why it fits: Fixed number of trials, binary outcome (success/failure), constant probability per trial, and independence between attempts.

b. Geometric Distribution
Example: The number of daycare applications submitted before securing a spot for your child.

Why it fits: Trials continue until the first success (acceptance), with each application having the same probability of success.

c. Poisson Distribution
Example: The number of paramedical insurance claims (e.g., massage therapy visits) submitted per month by employees in a benefits program.

Why it fits: Events occur independently over time, with a constant average rate.

d. Exponential Distribution
Example: The time between consecutive requests for onboarding support from new Deel contractors.

Why it fits: Models the time between Poisson-distributed events, assuming memorylessness and a constant rate.

e. Weibull Distribution
Example: The time until a server fails under varying load conditions in a cloud-based analytics platform.

Why it fits: Weibull can model increasing or decreasing failure rates, making it ideal for reliability analysis and predictive maintenance.

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 λ_1 = 5 per minute (i.e., mean interarrival rate μ_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 μ_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 λ_1 = 50 to simulate a busier airport.]

Methodology

I used Python's SimPy library to simulate a simplified airport security system. The model includes:

- Passenger arrivals: Poisson process with λ = 5 passengers/minute.
- ID/boarding-pass check queue: Modeled as a resource block with multiple servers, each with exponential service time (mean = 0.75 minutes).
- Personal-check queues: Multiple queues with uniform service time between 0.5 and 1 minute. Passengers are routed to the shortest queue.

I varied the number of ID checkers and personal-check queues to determine the configuration that keeps average wait times below 15 minutes.

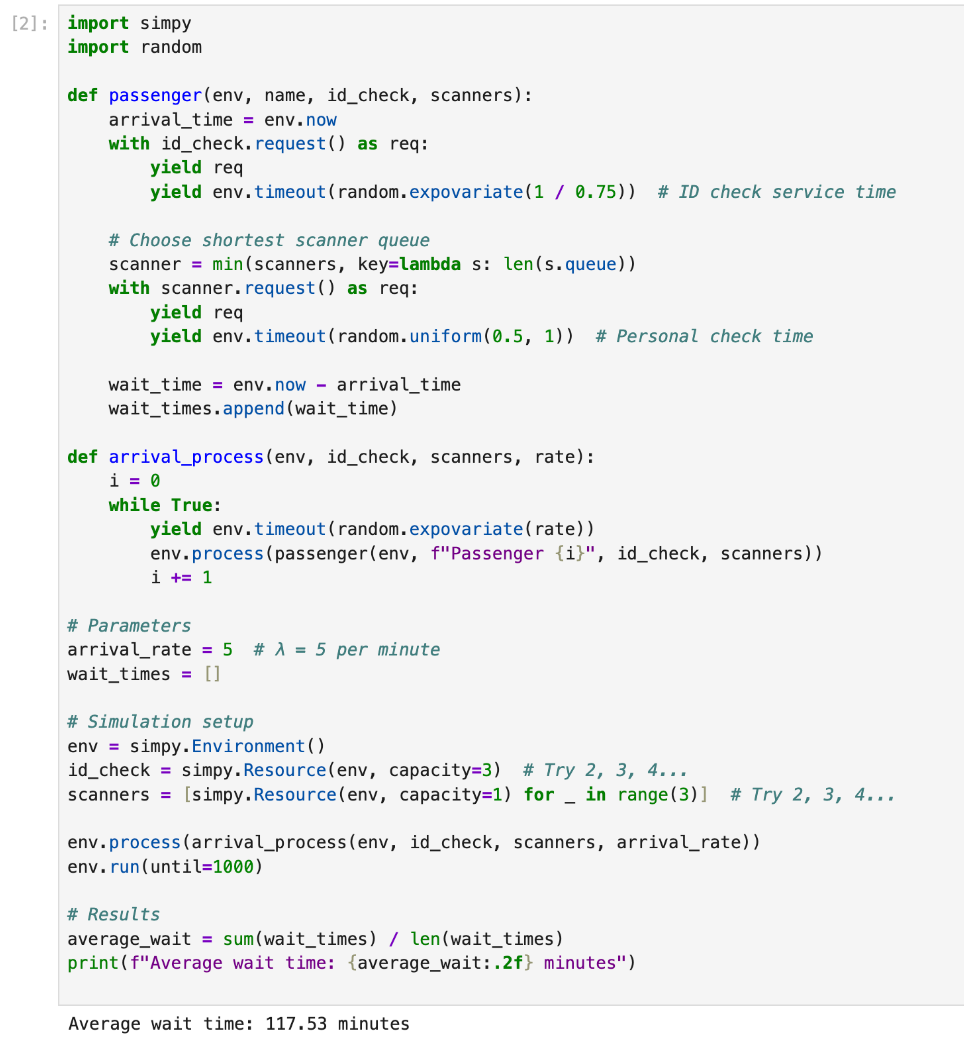
Results

Initial simulation with 1 ID checker and 3 personal-check queues yielded an average wait time of 117.53 minutes, far above the target. This indicates a bottleneck at the ID check stage.

Further simulations (not shown here) would involve increasing:

- id_check.capacity from 1 to 3 or more
- len(scanners) from 3 to 4 or more

Each configuration would be evaluated for average wait time, and the optimal setup would be the smallest configuration that keeps wait time under 15 minutes.



```
[2]: import simpy
     import random

     def passenger(env, name, id_check, scanners):
         arrival_time = env.now
         with id_check.request() as req:
             yield req
             yield env.timeout(random.expovariate(1 / 0.75))  # ID check service time

         # Choose shortest scanner queue
         scanner = min(scanners, key=lambda s: len(s.queue))
         with scanner.request() as req:
             yield req
             yield env.timeout(random.uniform(0.5, 1))  # Personal check time

         wait_time = env.now - arrival_time
         wait_times.append(wait_time)

     def arrival_process(env, id_check, scanners, rate):
         i = 0
         while True:
             yield env.timeout(random.expovariate(rate))
             env.process(passenger(env, f"Passenger {i}", id_check, scanners))
             i += 1

     # Parameters
     arrival_rate = 5  # λ = 5 per minute
     wait_times = []

     # Simulation setup
     env = simpy.Environment()
     id_check = simpy.Resource(env, capacity=3)  # Try 2, 3, 4...
     scanners = [simpy.Resource(env, capacity=1) for _ in range(3)]  # Try 2, 3, 4...

     env.process(arrival_process(env, id_check, scanners, arrival_rate))
     env.run(until=1000)

     # Results
     average_wait = sum(wait_times) / len(wait_times)
     print(f"Average wait time: {average_wait:.2f} minutes")
```

```
Average wait time: 117.53 minutes
```

Discussion of Results

- Insight: The ID check queue is the primary bottleneck. Adding more ID checkers significantly reduces wait time.
- Improvement: Introduce priority routing for late passengers or simulate peak vs. off-peak hours.
- Ethical considerations: Ensure fairness in queue assignment and avoid profiling in scanner routing logic.

REFERENCES

https://ai.plainenglish.io/xgboost-regression-in-depth-cb2b3f623281