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.

ABC Corp, a starter general merchandise retailer stores selling products through its stores and digital channels is interested to decide if the customers are more excited with ‘free gifts’ or ‘discounts’ to establish its strategic plan. As a starter corporation, it has never leveraged a popular personality for advertising or offered a free gift to make a data driven decision. In this scenario, the design of experiments will be best suited for the Retail store to ‘ask the customer’ through design of experiments to obtain answer from the best judges, the customer for their question of interest.

A/B Approach: A/B tests can be used to decide between two options ‘free gifts’ or ‘discounts’ using digital advertisement for two set of customers, both being representative set of the overall customers of the retail store. Response can be collected and tracked for each of the approach and through hypothesis testing, we can understand the best strategy to move forward.

Factorial Approach: Alternatively, as we extend additional options to understand the impact of interest through more than two options, say leveraging a popular personality, discount, and free gifts comes into play, we can take advantage of Full factorial or balanced fractional factorial approach. In this scenario, the combination though mostly independent of each other could include Interaction terms of ‘free gifts’ + ’discounts of certain Top selling products’ to collect the combination that captures more interest

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.

Approach

1) set up Fractional factorial for 16 houses as number of runs and 10 yes/no as factor of the house to obtain 2 level fractional factorial design (-1 or +1) 2) for readability, I have tagged the features as features of the house (large yard, solar roof, etc.) as noted in the requirement and the response of the impact of the factors as yes/no 3) Though readibility is better wuith Approach#2, the summary of the model was possible only with binary output Using d as binary to intepret the combinations (as yes/no doesnt provide combinations) 4) Summary shows which main effects are aliased with two factor interactions (2fi). 5) resolution III design was identified from generators $generators

[1] E=AB F=AC G=BC H=AD J=BCD K=ABCD

As this have 3 factors

This design Identities ABE =ACF=GBC=HAD=JBCD=KBCD

For usage of less factors, we can choose one of ABE =ACF=GBC=HAD

Combination include

A=Large.Yard;B=Pool;E=X3.Car.Garage

A=Large.Yard;C=Solar.Roof;F=Walk.In.Closet

G=Basement;B=Pool; C=Solar.Roof

H=Backyard.Deck;A=Large.Yard;D=Office.Room

```
In [1]: # Clear environment

        rm(list = ls())
```

In [2]:

```r
# Call the library 'FrF2'
#install.packages("FrF2")
library("FrF2")
library(tidyr) # to use tibble function
library(tibble) #use rowname to column
library("dplyr")
```

```
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



Attaching package: 'dplyr'


The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

In [4]:
```
# Set seed = 1 to produce reproducible result

set.seed(1)
```

In [7]:
```
#Information
# 16 houses , 10 yes/no factor, 2 level fractional factorial design (-1 or +1)
Fact<-FrF2(nruns = 16,nfactors = 10)
Fact
```

A design: 16 × 10

	A	B	C	D	E	F	G	H	J	K
	<fct>	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**
1	-1	1	-1	-1	-1	1	-1	1	1	-1
2	1	-1	1	-1	-1	1	-1	-1	1	1
3	1	-1	-1	1	-1	-1	1	1	1	1
4	1	-1	1	1	-1	1	-1	1	-1	-1
5	-1	1	1	1	-1	-1	1	-1	1	-1
6	1	1	-1	-1	1	-1	-1	-1	1	1
7	-1	1	-1	1	-1	1	-1	-1	-1	1
8	-1	-1	-1	1	1	1	1	-1	1	-1
9	-1	1	1	-1	-1	-1	1	1	-1	1
10	1	1	-1	1	1	-1	-1	1	-1	-1
11	-1	-1	-1	-1	1	1	1	1	-1	1
12	-1	-1	1	1	1	-1	-1	-1	-1	1
13	1	1	1	-1	1	1	1	-1	-1	-1
14	1	1	1	1	1	1	1	1	1	1
15	1	-1	-1	-1	-1	-1	1	-1	-1	-1
16	-1	-1	1	-1	1	-1	-1	1	1	-1

In [51]:
```
#choosing few features to better check the data
#Changing -1 to "no" and +1 to "yes"
(fracFact <- FrF2(
  nruns = 16, nfactors =10,
  factor.names = c(
    'Large Yard', 'Pool', 'Solar Roof', 'Office Room', '3 Car Garage',
    'Walk-In Closet', 'Basement', 'Backyard Deck', 'Modular Kitchen', 'Sunroom'
    ),
  default.levels = c('Yes', 'No'),
    randomize = FALSE
  ) %>%
  as_tibble()
 )
```

A tibble: 16 × 10

Large.Yard	Pool	Solar.Roof	Office.Room	X3.Car.Garage	Walk.In.Closet	Basement	Backyard.Deck	Modular.Kitchen	Sunroom
<fct>	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**	**<fct>**
Yes	Yes	Yes	Yes	No	No	No	No	Yes	No
No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Yes	No	Yes	Yes	Yes	No	Yes	No	No	Yes
No	No	Yes	Yes	No	Yes	Yes	Yes	No	No
Yes	Yes	No	Yes	No	Yes	Yes	No	No	Yes
No	Yes	No	Yes	Yes	No	Yes	Yes	No	No
Yes	No	No	Yes	Yes	Yes	No	No	Yes	No
No	No	No	Yes	No	No	No	Yes	Yes	Yes
Yes	Yes	Yes	No	No	No	No	Yes	No	Yes
No	Yes	Yes	No	Yes	Yes	No	No	No	No
Yes	No	Yes	No	Yes	No	Yes	Yes	Yes	No
No	No	Yes	No	No	Yes	Yes	No	Yes	Yes
Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No
No	Yes	No	No	Yes	No	Yes	No	Yes	Yes
Yes	No	No	No	Yes	Yes	No	Yes	No	Yes
No	No	No	No	No	No	No	No	No	No

```
In [52]: #understanding the results
         data <- fracFact %>% rownames_to_column('fracFact')
         #column_to_rownames(var = "Contas.Resultado")
         data
```

A tibble: 16 × 11

fracFact	Large.Yard	Pool	Solar.Roof	Office.Room	X3.Car.Garage	Walk.In.Closet	Basement	Backyard.Deck	Modular.Kitchen	Sunro
<chr>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>
1	Yes	Yes	Yes	Yes	No	No	No	No	Yes	No
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
3	Yes	No	Yes	Yes	Yes	No	Yes	No	No	Yes
4	No	No	Yes	Yes	No	Yes	Yes	Yes	No	No
5	Yes	Yes	No	Yes	No	Yes	Yes	No	No	Yes
6	No	Yes	No	Yes	Yes	No	Yes	Yes	No	No
7	Yes	No	No	Yes	Yes	Yes	No	No	Yes	No
8	No	No	No	Yes	No	No	No	Yes	Yes	Yes
9	Yes	Yes	Yes	No	No	No	No	Yes	No	Yes
10	No	Yes	Yes	No	Yes	Yes	No	No	No	No
11	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes	No
12	No	No	Yes	No	No	Yes	Yes	No	Yes	Yes
13	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No
14	No	Yes	No	No	Yes	No	Yes	No	Yes	Yes
15	Yes	No	No	No	Yes	Yes	No	Yes	No	Yes
16	No	No	No	No	No	No	No	No	No	No

In [50]:
```
fracFact %>% select(Large.Yard) %>%
group_by(Large.Yard) %>%
  summarize(n())
```

A tibble: 2 × 2

Large.Yard	n()
<fct>	<int>
Yes	8
No	8

my_range <- 1:10

my_range

for (i in 1) { fracFact %>% select(fracFact[1]) %>% group_by(fracFact[1]) %>% summarize(n()) }

In [59]:
```
#Using d as binary to intepret the combinations (as yes/no doesnt provide combinations)
d <- FrF2(16,10, res3 = T, factor.names = c(
    'Large Yard', 'Pool', 'Solar Roof', 'Office Room', '3 Car Garage',
    'Walk-In Closet', 'Basement', 'Backyard Deck', 'Modular Kitchen', 'Sunroom'
    ), randomize = FALSE)
#print(d)
aliasprint(d)
```

```
$legend
 [1] A=Large.Yard      B=Pool            C=Solar.Roof      D=Office.Room    
 [5] E=X3.Car.Garage   F=Walk.In.Closet  G=Basement        H=Backyard.Deck  
 [9] J=Modular.Kitchen K=Sunroom        

$main
 [1] A=BE=CF=DH=JK B=AE=CG       C=AF=BG       D=AH=GJ       E=AB=FG      
 [6] F=AC=EG       G=BC=DJ=EF=HK H=AD=GK       J=AK=DG       K=AJ=GH      

$fi2
[1] AG=BF=CE=DK=HJ BD=CJ=EH=FK    BH=CK=DE=FJ    BJ=CD=EK=FH    BK=CH=DF=EJ
```

```
In [63]: #Above shows which main effects are aliased withtwo factor interactions (2fi).
         #Considering the set of features from the main effects
         #ABE =ACF =ADH =AJK..
```

In [88]:
```
#design.info(d)
summary(d)
```

```
Call:
FrF2(16, 10, res3 = T, factor.names = c("Large Yard", "Pool",
    "Solar Roof", "Office Room", "3 Car Garage", "Walk-In Closet",
    "Basement", "Backyard Deck", "Modular Kitchen", "Sunroom"),
    randomize = FALSE)

Experimental design of type  FrF2
16  runs

Factor settings (scale ends):
  Large.Yard Pool Solar.Roof Office.Room X3.Car.Garage Walk.In.Closet Basement
1         -1   -1         -1          -1            -1             -1       -1
2          1    1          1           1             1              1        1
  Backyard.Deck Modular.Kitchen Sunroom
1            -1              -1      -1
2             1               1       1

Design generating information:
$legend
 [1] A=Large.Yard      B=Pool            C=Solar.Roof      D=Office.Room
 [5] E=X3.Car.Garage   F=Walk.In.Closet  G=Basement        H=Backyard.Deck
 [9] J=Modular.Kitchen K=Sunroom

$generators
[1] E=AB   F=AC   G=BC   H=AD   J=BCD  K=ABCD


Alias structure:
$main
 [1] A=BE=CF=DH=JK B=AE=CG       C=AF=BG       D=AH=GJ       E=AB=FG
 [6] F=AC=EG       G=BC=DJ=EF=HK H=AD=GK       J=AK=DG       K=AJ=GH

$fi2
[1] AG=BF=CE=DK=HJ BD=CJ=EH=FK    BH=CK=DE=FJ    BJ=CD=EK=FH    BK=CH=DF=EJ


The design itself:
   Large.Yard Pool Solar.Roof Office.Room X3.Car.Garage Walk.In.Closet Basement
1          -1   -1         -1          -1             1              1        1
2           1   -1         -1          -1            -1             -1        1
3          -1    1         -1          -1            -1              1       -1
4           1    1         -1          -1             1             -1       -1
5          -1   -1          1          -1             1             -1       -1
6           1   -1          1          -1            -1              1       -1
7          -1    1          1          -1            -1             -1        1
8           1    1          1          -1             1              1        1
9          -1   -1         -1           1             1              1        1
```

```
10          1   -1         -1           1            -1             -1        1
11         -1    1         -1           1            -1              1       -1
12          1    1         -1           1             1             -1       -1
13         -1   -1          1           1             1             -1       -1
14          1   -1          1           1            -1              1       -1
15         -1    1          1           1            -1             -1        1
16          1    1          1           1             1              1        1
   Backyard.Deck Modular.Kitchen Sunroom
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
class=design, type= FrF2
```

In [ ]:
```
#Above is resolution III design which can be identified from generators
#$generators
#[1] E=AB   F=AC   G=BC   H=AD   J=BCD  K=ABCD
#As this have 3 factors
#This design Identities ABE =ACF=GBC=HAD=JBCD=KBCD

#For usage of less factors, we can choose one of  ABE =ACF=GBC=HAD
#Combination include

#A=Large.Yard;B=Pool;E=X3.Car.Garage
#A=Large.Yard;C=Solar.Roof;F=Walk.In.Closet
#G=Basement;B=Pool; C=Solar.Roof
#H=Backyard.Deck;A=Large.Yard;D=Office.Room
```

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

I. Binomial

A Tech organization contracts the usage of their software to its customers. To avoid work overload for contract extension, they have distributed the contract renewal process, follow-ups, policies setup, and contract term changes beginning of every quarter. Every contract is tagged as P => customer who successfully extended the software contract (1-p) => customer cancelled the Contract

Probability of getting “x” Contracts extension responses is independent on “n” Bernoulli responses between customers and is well suited for a Binomial distribution We assume every quarter will have equal number of contract renewals. If the p is low, it indicates less renewals and “P” is more indicates more than usual contract renewals. We can obtain the probability of the Tech organization reaching 70% of its quarterly customer contracts P(x>0.7) and use hypothesis testing to accept/reject its assumption.

II. Geometric

Use case: A data science analyst is attempting to compile his code and produce expected outcome for an assignment before its due date. Trial: Every time a code is compiled % probability of finding a bug whenever the code is executed.

P => Good outcome (Successful Code outcome) (1-P) => Failed attempts/bug

As geometric distribution can be used to identify probability of having “x” failures before success (or vice versa) (OR) used to find if the trials are independently distributed

Case 1: Is each Bernoulli trial of code compiles by a data scientist is independent of each other? Can a dependent underlying root cause contribute to the code not being independently distributed?

Case 2: If the code compilation trails prove i.i.d independent of each other, we can use geometric distribution to find how many trials of failed attempts are made before a successful outcome of the code

III. Weibull

I will use the above scenario of a data scientist trying to get his code up and running. Weibull can be used to identify how long it will take him to get the expected results. Assuming he starts working on his assignment at 8AM on a Wednesday morning, Weibull can be get the probability of getting the assignment completed before midnight?

As the failure rate decreases with time, following the rule of ‘worst things fail first’, “K” will be less than 1 for this scenario

IV. Poisson

Use case: I would like to understand how many hits I get on my blog. We can use binomial for this scenario. I can see if I get 0.5 hit in 1 hour or NOT. But it cannot contain more than 1 event in the unit of time (say, 5 hits in one hours). I have options to shrink my unit of time to find if I get a hit or not. But, I don’t have data about the number of hits (considering peak and lazy hours) to come up with the unit of time window (any why bother as we have Poisson)

As Poisson doesn’t require “n” or “p” and the only parameter is Lamda, this scenario will fit Poisson well. Assuming lamda =0.5 (average of 1 hit per hour), I can develop a probability of getting more than 5 or 10 hits from Poisson distribution

V. Exponential

As continuous Exponential distribution complements the discrete Poisson distribution , I will use the arrival time of the hits on my website to obtain the time between successive arrivals of my website visitors using the scale as 1/lamba. Considering 0.5 hit in 1 hour for Poisson, my parameter for lamda is (1/0.5) for exponential.

As one of the widely used use case of exponential distribution is anomaly detection, I can extend this use case to see if I haven’t received any hits for 5 hours on morning hours, then that will be an prompt for me to can check to see if the website is actually up for visitors to login

In [ ]:
```
######13.2 Simulation is documented in a separate file##
```