Applying SIR Math Modeling to Regular Season Portland Trailblazers Home Game Attendance

Carlos Fuentes, Nick Kuntz, Victor Tu

Prof. Goldwyn

Math Modeling

Fall 2021

Purpose:

For our final project, we created a model of fan attendance for Trailblazer home games at the Moda Center over one regular season (41 home games).

Background/Methods:

Using a time step of one game, we created a model by hand and found an equilibrium using Python modeling. First, we created an SIR model and produced equations which depicted attendance based on various categories to organize the susceptible fan population with. Using these four equations, we created a Leslie matrix to find the relationship between each variable. Next, we used these equations in Python to create three different models with alternative situations: for the first, we kept all variables constant throughout the season, meaning that we found an equilibrium. For the second, we introduced stochasticity to the model, meaning that we made one of the variables have a degree of randomness at each time step, which is a more realistic model than the first scenario because of the many possible influences that we couldn't capture within the scope of our project. Thirdly, we changed one of the variables midway through the season, which was representative of a major shift in fan attendance, which might be due to many reasons, including a winning/losing streak, new coaching personnel, or player injuries.

Assumptions:

As with any model, we had to make several assumptions to model such a complex system, which could include infinite variables. Some of our basic assumptions were:

- Attendance was independent of weather, time of year, team's record, possible injuries, winning streaks, rating of opposing team, etc.
- Fan capacity: 100,000
- Time step: 1 game
- Susceptible fan base: Portland Trailblazer fans on Facebook
- Fans will go to two games if they go to one
- Opposing teams' fans are negligible

To get a sense of how many fans might be attending these games, we decided to look at Portland Trailblazer's Facebook page. Due to complications with restraints on Python, we decided to raise the Moda Center's capacity from 20,000 to 100,000. Next, our SIR model was set up in a way that meant fans who go to their first game during the season will also go to the following game, which is due to our altered SIR model, which makes fans go to two games before having the option to leave. Although we did not account for the variables listed above (which would be near impossible to completely capture), we decided to introduce stochasticity to our model to account for some of the possible errors in our model.

SIR Model:

F (fans): Fans who are not at a game at *t*

W (willing): fans who are at the game at *t* and are going to go to next game (*t+1*)

U (unwilling): fans who are at the game at *t* and are not going to next game (*t+1*)

N (never): Fans who went to last game at (*t-1*) and are never going to another game

We decided to have four categories to provide several options for fans. We knew we wanted to differentiate between fans who were going to the next game and fans who weren't, and to do this, we created the two different categories (W and U). We also knew that we wanted to account for fans who will never go to another game versus fans who might go to another game, which is why we created N, but allowed the fans an option between going back into the susceptible category F or entering N. One important note from the diagram is that U empties at every time step, because fans who are at a game and not going to the next game are either going to re-enter F or enter N.

The primary issue with our model in Python is that fans must first enter W to later enter U. In real life, this is not realistic because many people just attend one game in a row, and attending more than one in a row is only the case for a small minority. However, because we knew we wanted to differentiate between W and U, and because of restraints on Python, we kept the model as is.

SIR equations:

$$F_{t+1} = F_t - \beta F_t + \psi U_t$$
$$W_{t+1} = W_t + \beta F_t - \mu W_t$$
$$U_{t+1} = U_t + \mu W_t - \psi U_t - (1 - \psi)\ U_t$$
$$N_{t+1} = N_t + (1 - \psi) U_t$$

Beta: Proportion of fans who didn't go to game (*t-1*) but attend at game *t*

Psi: Proportion of fans who attend game at *t* and decide not to attend the next game (*t-1*), but might attend a future game

Mu: Proportion of fans who attend at game *t* and will attend the next game (*t+1*), but will not attend the following game (*t+2*)

As stated in the previous section, U empties out at every time step, while N only gets bigger. Meanwhile, F and W consistently contain people at every time step and both lose/gain different numbers of fans after each game.

Leslie Matrix:

$$\begin{bmatrix} F_{t+1} \\ W_{t+1} \\ U_{t+1} \\ N_{t+1} \end{bmatrix} = \begin{bmatrix} 1-\beta & 0 & \psi & 0 \\ \beta & 1-\mu & 0 & 0 \\ 0 & \mu & 1-\psi-(1-\psi) & 0 \\ 0 & 0 & 1-\psi & 1 \end{bmatrix} \begin{bmatrix} F \\ W \\ U \\ N \end{bmatrix}$$

After creating the SIR equations, we decided to input them into a Leslie Matrix and attempt to find eigenvalues/vectors to provide further insight on the relationships between each variable. For this matrix, an equilibrium, along with stability/unstability exists for each variable, but the eigenvalues we calculated did not give us a mathematically viable answer.

Initial conditions:

To model this system, we used numbers from basic research to come up with initial conditions at $t=0$, which are as follows:

F = 100,000

W = 0

U = 0

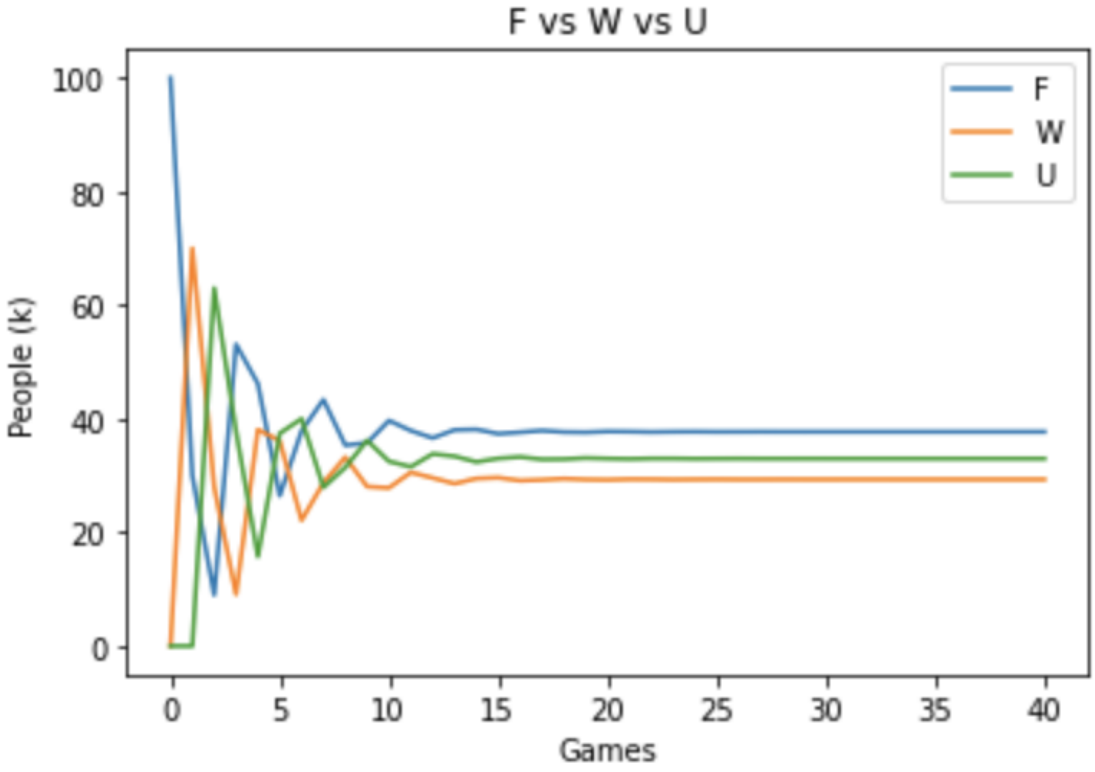
N = 0

These are the conditions immediately prior to the first game, which means all of the fans are going to be in F. However, as soon as the season begins, F dramatically drops and the numbers become more dynamic, as seen in the following graphs.

Python modeling:

Hypothetical 1: All variables are held constant throughout season

$\beta = .7$, $\mu = .9$, $\Psi = .8$

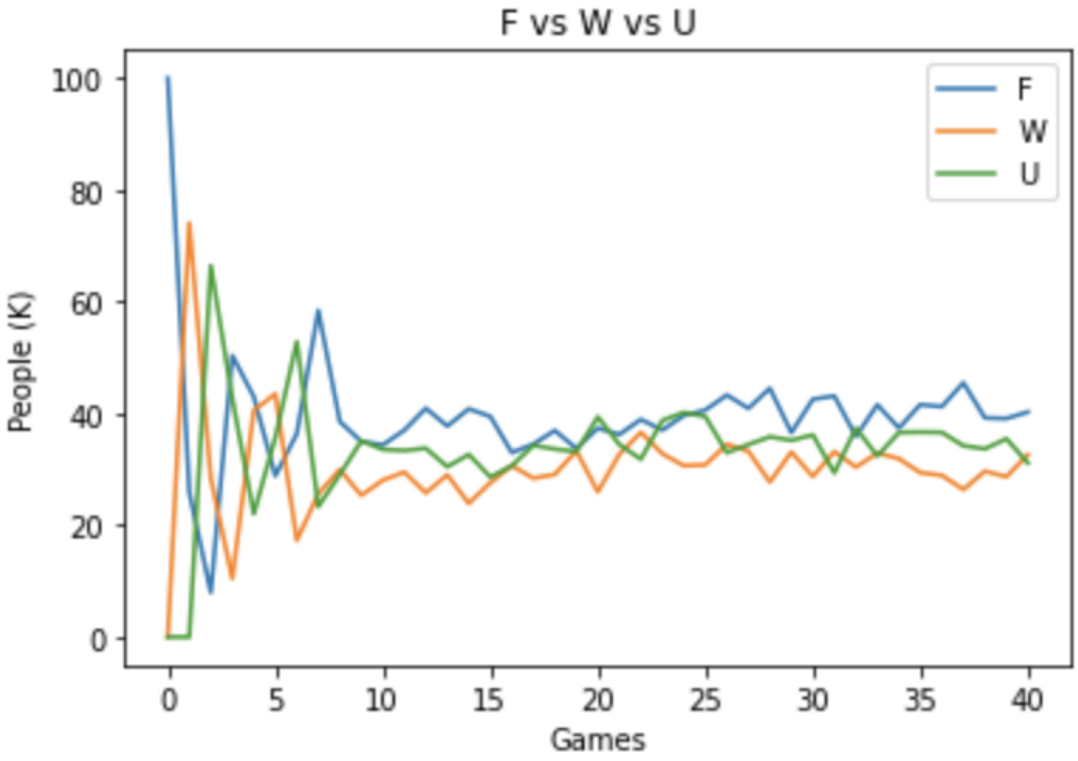


In this basic model, we kept all variables constant to find an equilibrium for each category. We used beta at 0.7, mu at 0.9, and psi at 0.8, which we found through a trial-and-error process to find equilibria that we deemed reasonable.

It is important to note that this kind of scenario is completely unrealistic, especially considering the fact that for the first 10 games, F, W, and U all drastically fluctuate. The reason this graph is so unrealistic is because the initial conditions are completely skewed so that F is full, and all the other categories are at 0. If we started midway through the season, or after the first few games, this model would look more realistic.

Hypothetical 2: Introducing stochasticity to the variables

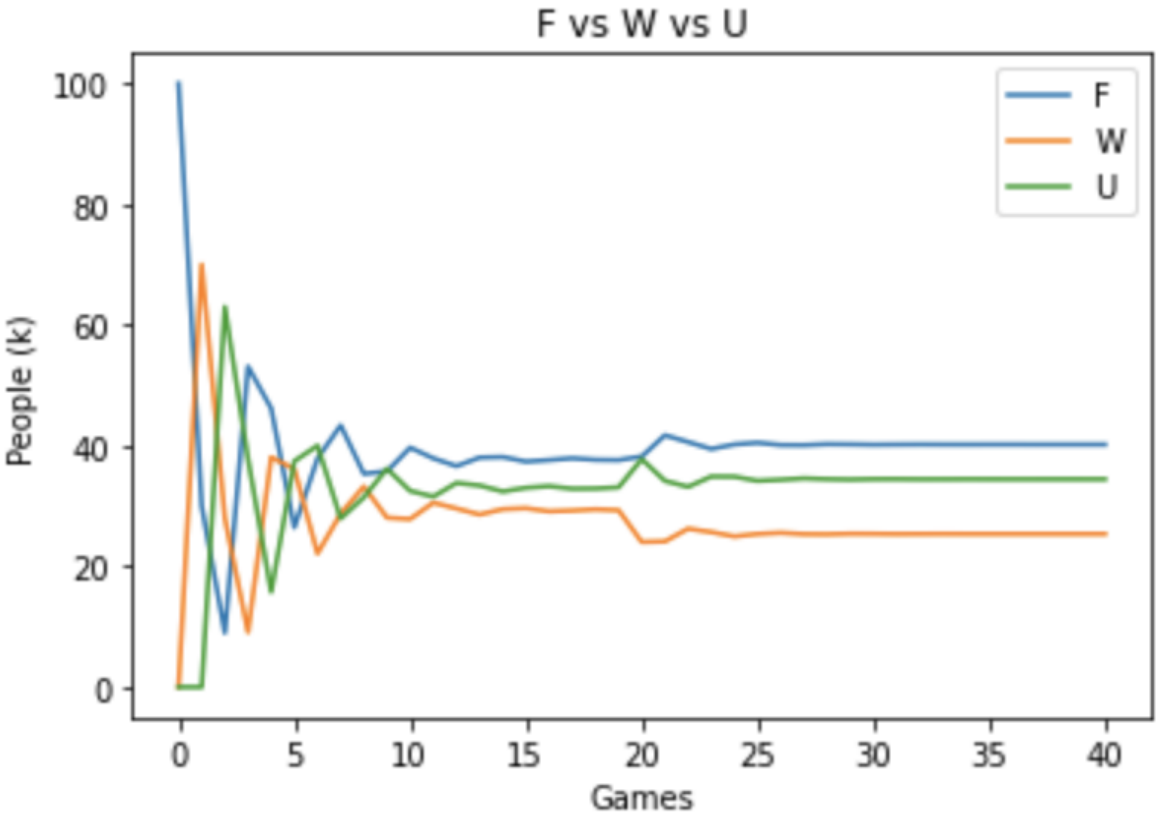
- Beta: Uniform Distribution between (0.6,0.8)
- Mu: Uniform Distribution between (0.9,1)
- Psi: Uniform Distribution between (0.7,0.9)



In this model, we introduced a degree of randomness (stochasticity) to the variables, which we did by projecting the attendance for the season with uniform distributions for each variable. This model successfully captures the degree of randomness that each category would experience over a regular season, although we are unable to account for the factors that lead to this randomness. We decided to let each factor have a range of distribution between 0.1 and 0.2, which covers a substantial range of variability.

Hypothetical 3: Major change midway through season

- Games 1-19: β =.7, μ = .9, Ψ = .8
- Games 20-40: β =.6, μ = .95, Ψ = .7



For this model, we lowered beta by 0.1, raised mu by 0.05, and lowered psi by 0.1. These changes indicate that slightly fewer fans are going to games if they haven't already gone to one, and more fans will go to the game at (*t+1*) but will not attend at (*t+2*).

If this graph was modeling real fan attendance, this change in variables could be explained by a star player getting injured for the remainder of a season or ticket prices being significantly increased.

Conclusions:

We have many takeaways from completing this project. Our goal was to demonstrate how population sizes of Trailblazer home attendance changed over the season based on the SIR-type model that we created. After adopting this model to Python and simulating it over the season, we were able to visually demonstrate what patterns occur in the season based on the parameters we set. The most basic setup was the one where we kept our coefficients the same throughout the season, which we used to see where equilibrium would be reached by each compartment in our model and predict which compartments would generally have the most or least amount of people in them. While we recognize that this setup is not entirely accurate when it comes to modeling fan behaviour, it was extremely helpful to set a baseline for understanding how values of coefficients affect the size of each compartment. We were able to show that when we set beta at 0.7, mu at 0.9, and psi at 0.8, the overall fanbase compartment was the largest, the unwilling fans compartment was the second largest, and the willing fans compartment was the third largest. This makes sense because generally we'd expect there to be more people at a game that don't want to go to the next one than people who are not at a game that do want to go to a next one, which was mainly impacted by mu being higher than beta. Our next model introduced stochasticity in an attempt to more accurately show that there are a lot of factors at and before each game that affect someone's willingness to go to that game. To do this, we generated a uniform random number at every time step to show that someone's willingness to go to a game or not is going to still be close to a certain probability, but that there still could be some variability. What we learned from this model is that it's likely a more accurate model when looking at individual games, but overall, each compartment averages to about the same equilibria values from the first model, so it has a consistent pattern overall even with stochasticity

involved. The final conclusion came when we implemented our third model, which went back to using deterministic coefficient values, but assumed that there was a big event in the middle of a season (new player signing, multiple stars get injured, team involved in a scandal). We did this by updating the coefficients in the middle of the season and then letting the rest of that season simulate with the new coefficient values. We found this model to be really interesting, because even though we changed the coefficients pretty drastically, the equilibria that each compartment reached was pretty close to the original equilibria.

Overall, we found that using an SIR-type model for this problem was very useful and provided some interesting conclusions. It definitely had its limitations, and there are still some unique variations of the model to explore, but we believe that we achieved our goal of modeling how attendance at Blazer home games changes over a season.

Appendix 1:

We decided to implement our work into code in order to produce data and graphs that reflect our SIR model. Using Python we first provided initial conditions for beta, psi, mu, F, W,U, and N. Next, we created a for loop that iterated through for each time-step (41 games). This for loop allowed us to produce a list of data and plot it.

Each plot and set of code for their respective hypotheticals were a bit different. For hypothetical 1, all values were kept constant. So the code is the same as mentioned above. For hypothetical 2, we got rid of the constants for beta, psi, and mu, and instead included a randomizer for each coefficient into our for loop. Then for hypothetical 3, we had two sets of constants and for loops in order to reflect the mid-season change and unwillingness of Blazers fans to attend a home game. The code for all three hypotheticals are included below.

Appendix 2:

Hypothetical 1:

```
#All values stay constant

#initial coefficients
beta = .7
psi = .8
mu = .9

F = {0:100}
W = {0:0}
U = {0:0}
N = {0:0}

#for loop
for i in range(1,41):
    F[i] = F[i-1] - beta*F[i-1] + psi*U[i-1]
    W[i] = W[i-1] + beta*F[i-1] - mu*W[i-1]
    U[i] = U[i-1] + mu*W[i-1] - psi*U[i-1]
    N[i] = (1-psi)*U[i-1]+N[i-1]

#organizing the data
data_list = [F,W,U,N]
data_list1 = [F,W,U]

df = pd.DataFrame.from_dict(data_list)
df1 = pd.DataFrame.from_dict(data_list1)

df.index = ['F','W','U','N']
df1.index = ['F','W','U']

df = df.transpose()
df1 = df1.transpose()

#print values of F,W,U,N
print(df)

#plot values F,W,U
plot = df1.plot.line()
plt.legend(loc = "upper right")
plt.title('F vs W vs U')
plt.xlabel('Games')
plt.ylabel('People (k)')

plt.show()
```

Hypothetical 2:

```
#stochasticity

#initial condtions

F = {0:100}
W = {0:0}
U = {0:0}
N = {0:0}

#for loop
for i in range(1,41):
    mu = np.random.uniform(.8,1) #randomizing mu between .8 and 1
    beta = np.random.uniform(.6,.8) #randomizing beta between .6 and .8
    psi = np.random.uniform(.7,.9) #randomizing psi between .7 and .9

    F[i] = F[i-1] - beta*F[i-1] + psi*U[i-1]
    W[i] = W[i-1] + beta*S[i-1] - mu*W[i-1]
    U[i] = U[i-1] + mu*W[i-1] - psi*U[i-1]
    b[i-1] = beta
    N[i] = N[i-1]+(1-psi)*U[i-1]

#organizing the data
data_list = [F,W,U,b,N]
data_list1 = [F,W,U]

df = pd.DataFrame.from_dict(data_list)
df1 = pd.DataFrame.from_dict(data_list1)

df.index = ['F','W','U', 'beta','N']
df1.index = ['F','W','U']

df = df.transpose()
df1 = df1.transpose()

#print values for F,W,U,beta,N
print(df)

#plot values for F,W,U
plot = df1.plot.line()
plt.legend(loc='upper right')
plt.title("F vs W vs U")
plt.xlabel('Games')
plt.ylabel('People (K)')

plt.show()
```

Hypothetical 3:

```
#Values changed mid-way through season

#initial conditions
beta = .7
psi = .8
mu = .9

F = {0:100}
W = {0:0}
U = {0:0}
N = {0:0}

#for loop for first half of the season
for i in range(1,41):
    F[i] = F[i-1] - beta*F[i-1] + psi*U[i-1]
    W[i] = W[i-1] + beta*F[i-1] - mu*W[i-1]
    U[i] = U[i-1] + mu*W[i-1] - psi*U[i-1]
    N[i] = (1-psi)*U[i-1]+N[i-1]

#constants change halfway through the season
beta = .6
mu = .95
psi = .7

#for loop for second half of the season
for i in range(20, 41):
    F[i] = F[i-1] - beta*F[i-1] + psi*U[i-1]
    W[i] = W[i-1] + beta*F[i-1] - mu*W[i-1]
    U[i] = U[i-1] + mu*W[i-1] - psi*U[i-1]
    N[i] = (1-psi)*U[i-1]+N[i-1]

#organize the data
data_list = [F,W,U,N]
data_list1 = [F,W,U]

df = pd.DataFrame.from_dict(data_list)
df1 = pd.DataFrame.from_dict(data_list1)

df.index = ['F','W','U','N']
df1.index = ['F','W','U']

df = df.transpose()
df1 = df1.transpose()

#print values for F,W,U,N
print(df)

#plot values for F,W,U
plot = df1.plot.line()
plt.legend(loc = "upper right")
plt.title('F vs W vs U')
plt.xlabel('Games')
plt.ylabel('People (k)')

plt.show()
```