Project #2 Real-Time Analysis (St. Louis City SC)

—Feng Jiang, Yuyao Zhuge

Description:

Major League Soccer (MLS) is the top professional league in the United States and Canada comprising 29 teams — 26 in the U.S. and 3 in Canada — since the 2023 season. In this project, we are focusing on analyzing the performance of the team St. Louis City SC in 5 of the recent MLS games which chronically are ATX VS STL (2:3), STL VS CLT (3:1), RSL VS STL(0:4), SEA VS STL (3:0), and STL VS CIN (5:1).

Until May 1st, 2023, St. Louis CITY SC (STL) achieved a score of 6 wins, 1 draw, and 3 losses, with a total of 19 points in second place in the Western Conference. STL is a young team who just joined in 2023 as an expansion team. It is impressive for their achievements in the MLS among all the strong and experienced teams.

Problem Statement:

For the core field strategy, STL has developed and implemented a well-established pressing strategy. It refers to the strategy in which a team tries to win the ball back quickly by putting pressure on the opposing team's players. Typically, pressing involves a high defensive line and lots of movement by the attacking team's players to disrupt the opposing team's passing and possession. In this project, the major goal is to study and analyze the performance of the STL relating to their pressing strategy against their opponents, and the other way around such that

a. How does STL City SC compare against itself in each of these games?
b. Which game does STL City SC perform the best as it relates to their pressing strategy?
c. Which team showed the most resistance to STL City SC in their pressing strategy?

Then, give individual players' comparisons and team comparisons over general metrics.

Methodology:

Original Methodology:

For the main problem of analyzing the pressing strategy, since the datasets are missing a significant column of the ball including the ball position and ball speed over time, it limits the feasibility and accuracy of solving this problem. Nevertheless, after thinking deeply about the conceptual idea of pressing strategy, we catch the key of it which is to win the ball back swiftly by pressing pressure on the opposing team players. From this perspective, although we do not know who is controlling the ball, we can analyze if our players are moving toward the opposing players in a certain range at a high speed. If so, then we can possibly guess this opposing player is controlling the ball or is standing in an important position. Therefore, we can identify that our players are performing a pressing strategy.

More specifically, here are the steps we take to analyze how teams perform a pressing strategy:

1. Identify the pressing events: we start by looking for periods of the game where one team's players consistently occupy the opposing team's half of the field. This could indicate that they are employing a pressing strategy.
2. Analyze the X-Y positions: Once we have identified pressing events, we look at the X-Y positions of the pressing team's players at those times. Specifically, we look for

instances where the pressing team has more players in the opposing team's half of the field, and where those players are positioned to cut off passing lanes and apply pressure to the opposing team's players.

3. Analyze the speed: Another important factor in pressing is player speed. We look at the speed of the pressing team's players during pressing events to see if they are consistently moving quickly to close down space and put pressure on the opposing team's players.
4. Compare the two teams: Once we have analyzed the pressing strategy for one team, we repeat the same analysis for the other team and compare the results. Specifically, we look for differences in the number of press events, the X-Y positions of the players, and the speed of the players between the two teams.

The above are the main steps of our original methodology. For the programming implementation, we first grouped “FrameCount” and “TeamType” to categorize that at each moment, for each opposing player, calculate our team members' and the opposing player’s distance. If the distance is within a certain range, count and record the data. Continue taking this step until the last data of “FrameCount”. To implement this idea, we implemented the function is_within_range to return if the two points are within a certain distance.

```
In [ ]: def is_within_range(center_x, center_y, x, y, radius=600):
            """
            Check if a given point (x, y) is within a given radius from a center point (center_x, center_y)
            """
            distance = np.sqrt((x - center_x) ** 2 + (y - center_y) ** 2)
            return distance <= radius
```

Code 1

Then, we looped through each row in the groupby object below. For each row with “TeamType” = 0, we found its corresponding distance of data with “TeamType” = 1 and checked if the distance was less than 1000 (this is 10 meters in real value, and it is set just as a reasonable threshold) by calling the function above. Then recorded the row if the function returns True.

```
In [115]: df8_play.groupby(['FrameCount','TeamType']).count()
```

Out[115]:

		JerseyNumber	XPosition	YPosition	PlayerSpeed
FrameCount	**TeamType**				
2355083	**0**	5	5	5	5
	1	9	9	9	9
2355084	**0**	5	5	5	5
	1	9	9	9	9
2355085	**0**	5	5	5	5
...	**...**	...	...	...	...
2555624	**1**	11	11	11	11
2555625	**0**	11	11	11	11
	1	11	11	11	11
2555626	**0**	11	11	11	11
	1	11	11	11	11

352907 rows × 4 columns

Code 2

Nevertheless, after we had implemented the code, due to the tremendous amount of data, we could not get an output in 4 hours (It could take even much longer). Truly, if there are 11 players on each side at one moment, it will calculate and match 11*11 = 121 times of distances for each “FrameCount”. Then, it would be 352907 * 121 = 42701747 times of calculation for sqrt((X1-X2)^2+(Y1-Y2)^2), which is very time-consuming, not to mention the concatenate process at the end of each loop.

```
# create a new empty dataframe to store the results
result_df = pd.DataFrame()
radius = 1000
for group,data in agg:
    if group[1] == 0:
        df_0 = data
        continue
    if group[1] == 1:
        df_1 = data
#     if p < 5:
#         p +=1
    else:
        break
        # loop over each row in df_0
    for i, row in df_0.iterrows():
        # get the center coordinates from df_0
        center_x, center_y = row['XPosition'], row['YPosition']

            # check if any points in df_1 are within a radius of 1000 units from the center point
            mask = df_1.apply(lambda x: is_within_range(center_x, center_y, x['XPosition'], x['YPosition'], radius=1000

            # create a new dataframe with the rows from df_1 that are within the radius
            within_radius_df = df_1[mask]

            # add a new column to the within_radius_df dataframe with the distance from the center point
            within_radius_df['distance'] = within_radius_df.apply(lambda x: np.sqrt((x['XPosition'] - center_x) ** 2 +

            # add a new column to the within_radius_df dataframe with the name of the center point
            within_radius_df['opponent_center_name'] = row['JerseyNumber']

            # add the within_radius_df dataframe to the result dataframe
            result_df = pd.concat([result_df, within_radius_df])

    # reset the index of the result dataframe
result_df = result_df.reset_index(drop=True)

# print the result dataframe
result_df
```

Code 3

Besides, due to the single-node process with my Python kernel, we could not do anything in parallel. Later, we tested not using concatenate in the code but just printed out the result of the final FrameCount. It took roughly one and a half hours. Since we did not have any high-performance computing devices for a stronger calculation, we ultimately gave up this method and tried a bunch of other methods such as mimicking the “geohash” method by dividing all the field area into grids and measuring the distance by grids.

```
In [182]:
# p=0
# create a new empty dataframe to store the results
result_df = pd.DataFrame()
radius = 1000
for group,data in agg:
    if group[1] == 0:
        df_0 = data
        continue
    if group[1] == 1:
        df_1 = data
#     if p < 5:
#         p +=1
    else:
        break
        # loop over each row in df_0
    for i, row in df_0.iterrows():
        # get the center coordinates from df_0
        center_x, center_y = row['XPosition'], row['YPosition']
        upper_x, lower_x, upper_y, lower_y = row['XPosition'] + radius,row['XPosition'] - radius,row['YPosition'] + rad

        df_result = df_1.loc[(df_1['XPosition'] > lower_x)&(df_1['XPosition'] < upper_x)&(df_1['YPosition'] > lower_y)&
        df_result['opponent_center_name'] = row['JerseyNumber']

        result_df = pd.concat([result_df, df_result])

    # reset the index of the result dataframe
result_df = result_df.reset_index(drop=True)

# print the result dataframe
result_df
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/var/folders/y9/tpf4g1_15nqbl3j7qy_tnvzm0000gn/T/ipykernel_43758/2006422634.py in <module>
     22         df_result['opponent_center_name'] = row['JerseyNumber']
     23
---> 24         result_df = pd.concat([result_df, df_result])
     25
     26 #             # check if any points in df_1 are within a radius of 1000 units from the center point

KeyboardInterrupt:
```

Code 4

It also took more than 4 hours and eventually, we could only turn to a simpler analysis.

Final Methodology:

Without analyzing the entire game frame by frame, we could only observe and consider the entire game as a whole. Another way of thinking about pressing strategy, it aims to attack and induce the opposing player to make mistakes to grab the ball back. Therefore, it can also be counted as pressing if players proactively seek for chances to attack instead of controlling and passing at the back of their own half-court.

Based on this, we decided to filter out the “FrameCount” when players spend time on the opposing team’s half-court. Furthermore, we laid down the measurement of individual speed with a high speed greater than 5.5 meters per second as they are using the pressing strategy. In addition, we filtered out the time that players spent on the opponents’ field Versus the total time they spent on the field. With the speed and time data, we could analyze if teams are performing a harsh pressing strategy.

Because teams switched courts in the middle of the game, the first thing was to capture the “FrameCount” when the first half ended and the second half started. To achieve this goal, we caught the point that the goalkeepers basically never went across the middle field. Therefore, his X position will always be positive or negative in the first half and then switched to the other. According to this, we implemented the function find_game_frame_count to return the four parts in each game respectively the first half game for the away team’s attack, the second half game for the away team’s attack, the first half game for the home team’s attack, and the second half game for the home team’s attack. Then, here are the problems we tried to solve.

1. Find the length of data of players on the opponents' field at a high speed. Compare the data for both teams. Compare the data for STL in all 5 matches.
2. Find the percentage of time that players spent on the opponents' field VS total time. Compare the data for both teams. Compare the data for STL in all 5 matches.

To better visualize and compare the data between teams and among all 5 matches, we created graphs and labeled them with their Jersey colors. Here shows the attack_high_speed_count VS each team in each game.

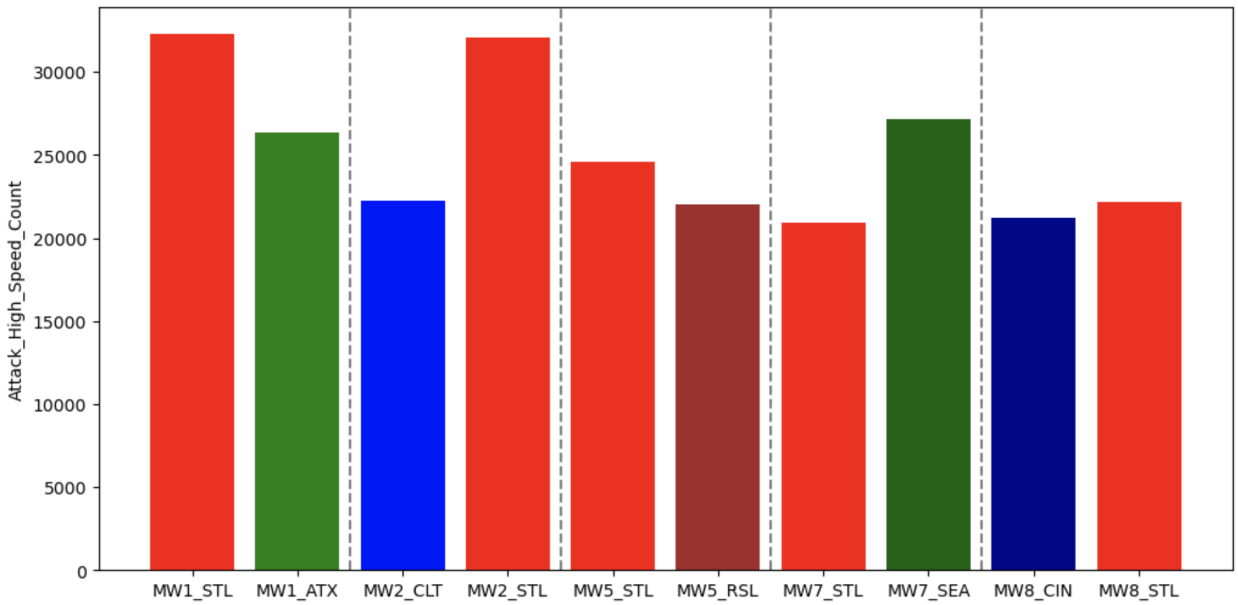


Figure 1

Each pair divided by the dotted line represents one single game respectively STL VS ATX (3:2), CLT VS STL (3:1), STL VS RSL (4:0), STL VS SEA (0:3), and CIN VS STL (5:1). All STL are in red, ATX is in green, CLT is in blue, RSL is in scarlet, SEA is in deep green, and CIN is in deep blue.

Here the following shows the time percentage of attacking VS total time among all the teams in the matches.

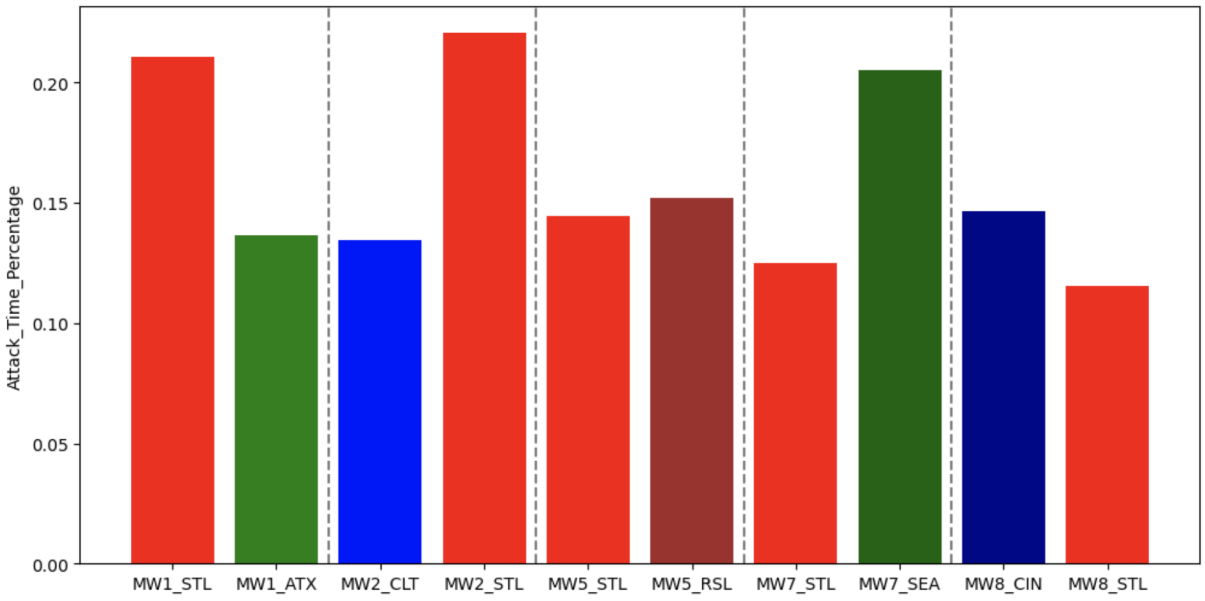


Figure 2

Each pair divided by the dotted line represents one single game respectively STL VS ATX (3:2), CLT VS STL (3:1), STL VS RSL (4:0), STL VS SEA (0:3), and CIN VS STL (5:1). All STL are in red, ATX is in green, CLT is in blue, RSL is in scarlet, SEA is in deep green, and CIN is in deep blue.

Team Comparison (ATX VS STL) in strategy

According to *Figures 1 and 2*, STL strived the most in MW1 and MW2. In MW1, STL once was 1:2 behind. Their pressing strategy worked and saved the game back. Not long after the second goal of ATX, because of the force and push from an STL striker, an ATX defender made a mistake and passed the ball to the back. However, he did not recognize the player he passed was his opponent until passed the ball out. The striker Jared Stroud went for an easy break. Furthermore, the third goal from STL was also the achievement of their pressing strategy. By pressing hard on their opponent, ATX lost the ball, and by excellent cooperation of long passes among several players, Joao Klauss eventually wrote a period with a striking goal for this game. It is definitely a win for their pressing strategy.

Team Comparison (STL VS CLT) in strategy

In addition, in MW2, it was more aggressive in attacking for STL as we can discover in both attacking speed and attacking time. An interesting discovery is that STL does not depend on ball possession too much. Basically, in each of the five games, STL maintains a ball possession below 50 percent. In MW2, although in terms of the data, STL had a greater advantage than CLT, STL only had 39% ball possession, which emphasized their pressing strategy. Because they did not seek for controlling the ball, and seek for opportunities to break through. Instead, they proactively sought attacking chances by exerting their strength of speed and pressing hard. Although CLT possessed 61% of ball control, they did not seem to adapt to STL's press. When they made a mistake at their half-field, STL would pursue a goal with speed and outstanding cooperation.

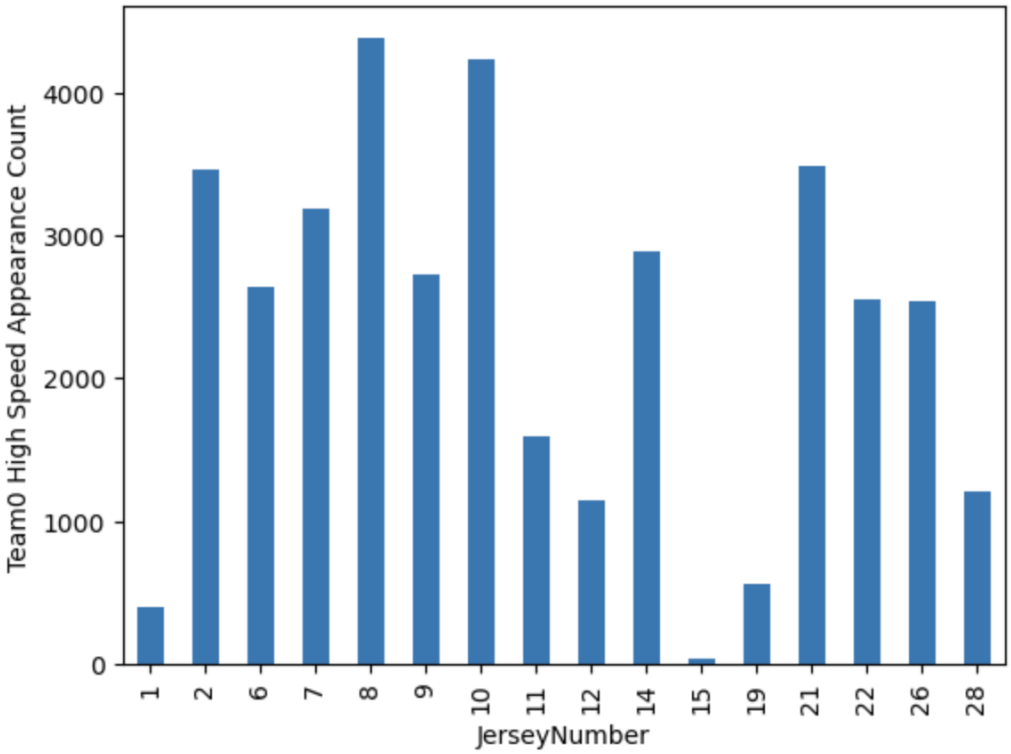
Among the two figures, we can discover that SEA had better data than STL in both figures. This could be due to the reasons for their line-up. STL had a 5 defender line-up. It seemed that at the beginning of the game, the main goal of the team was to attack with a defensive counterattack. Thus, the attacking data did not seem to have a well-performed pressing strategy implementation. Besides, the final score of that game also validated the importance of implementing their pressing strategy. Accordion to the data above, STL is a team strengthened in attacking.

Brief Individual Analysis by Graph:

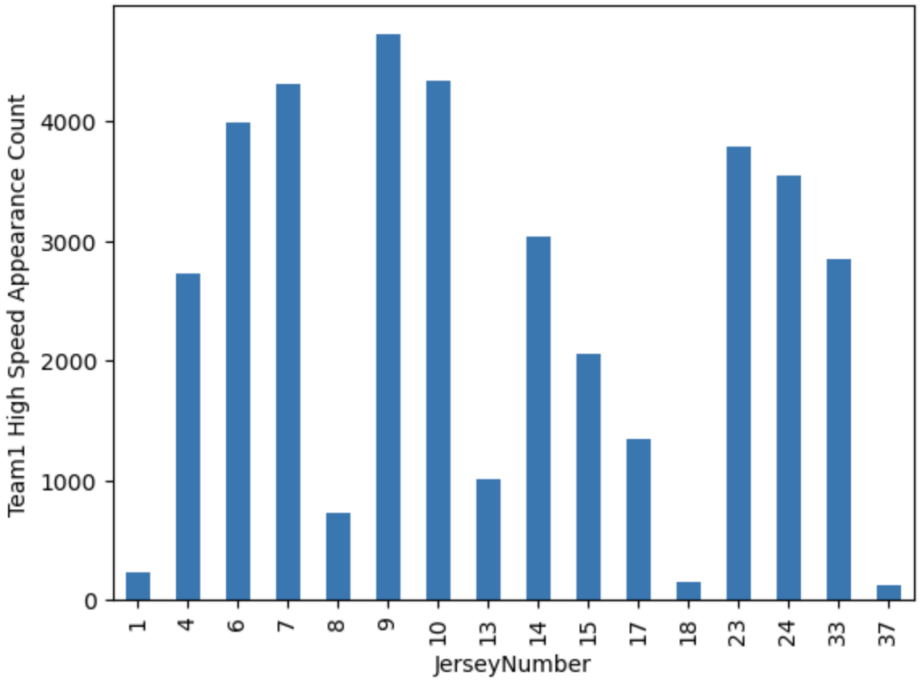
Since we take high considerations of high speed in this project. Here we include graphs of each team VS Jersey Number in each game. Based on the High-Speed Appearance count, we can discover which team performed well in running and grabbing the ball. Therefore, they could make abundant contributions to the team's pressing strategy.

ATX VS STL (2:3):

ATX:

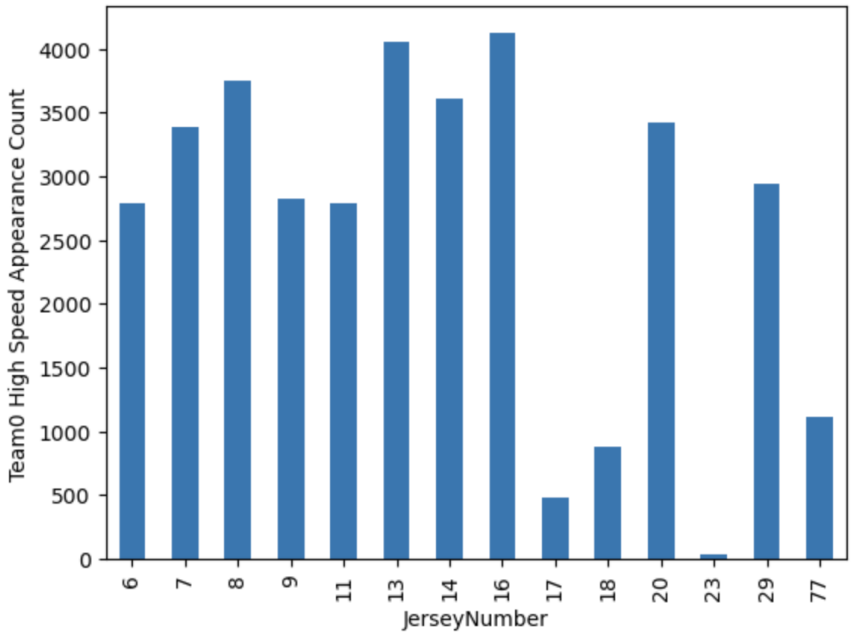


STL:

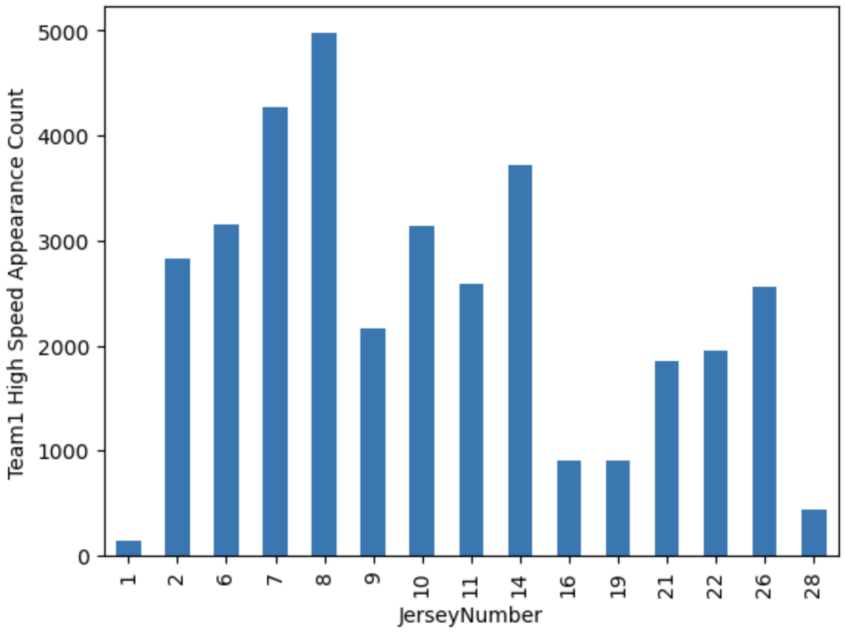


STL VS CLT (3:1):

STL:

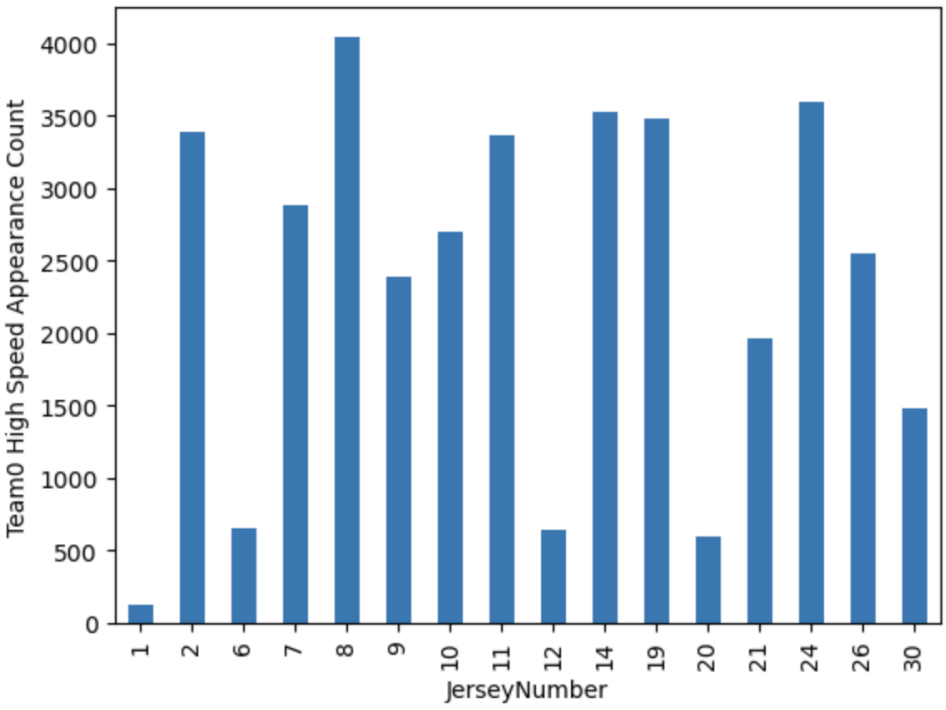


CLT:

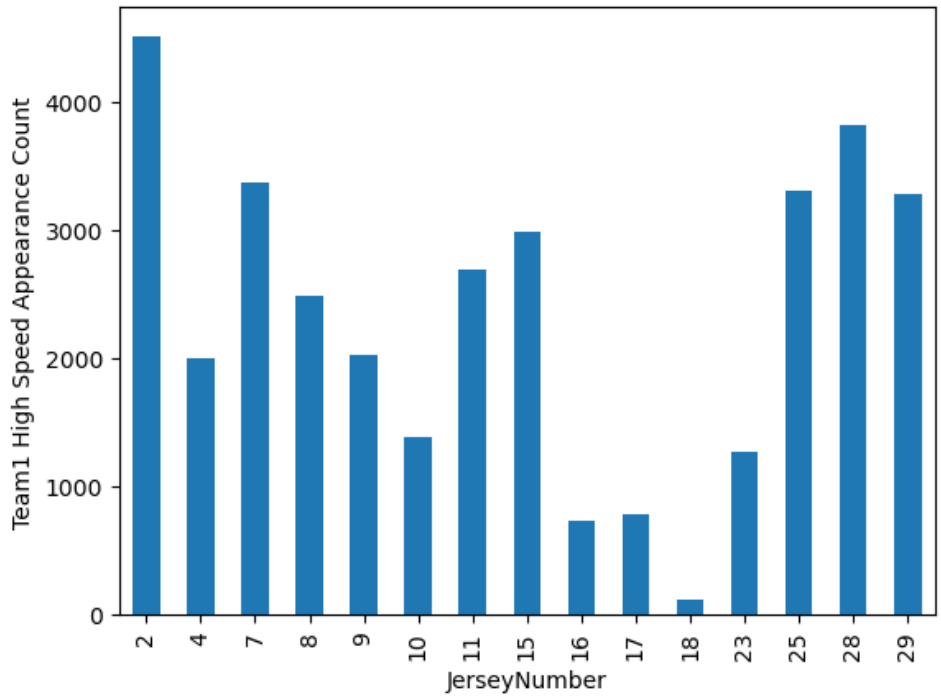


RSL VS STL(0:4):

RSL:

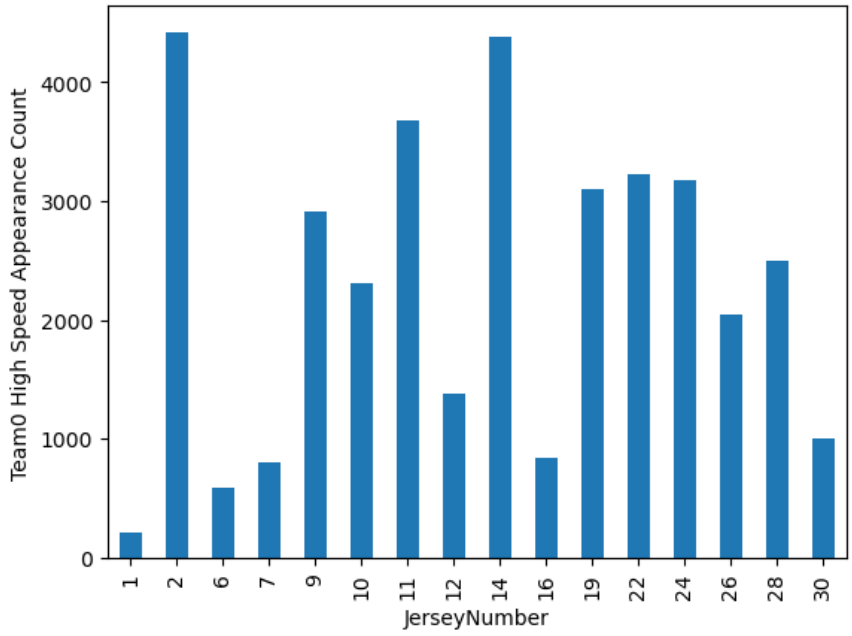


STL:

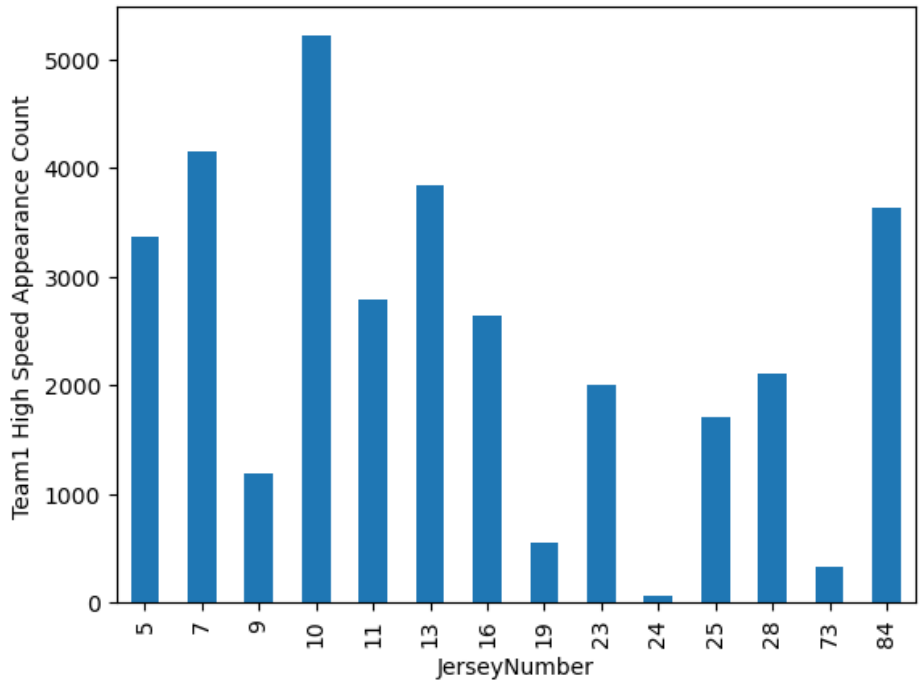


SEA VS STL (3:0):

SEA:

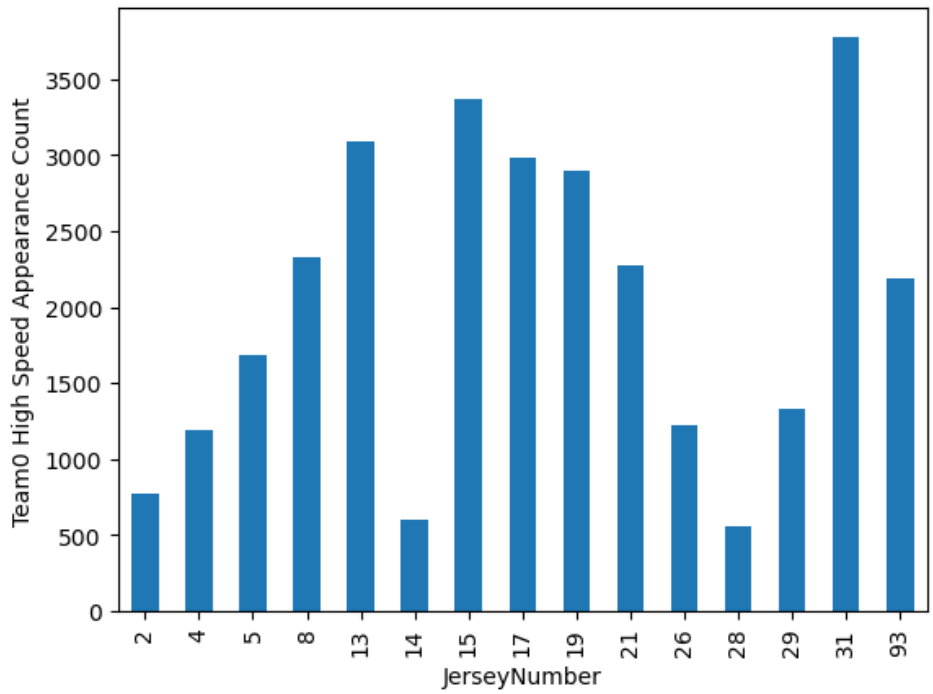


STL:

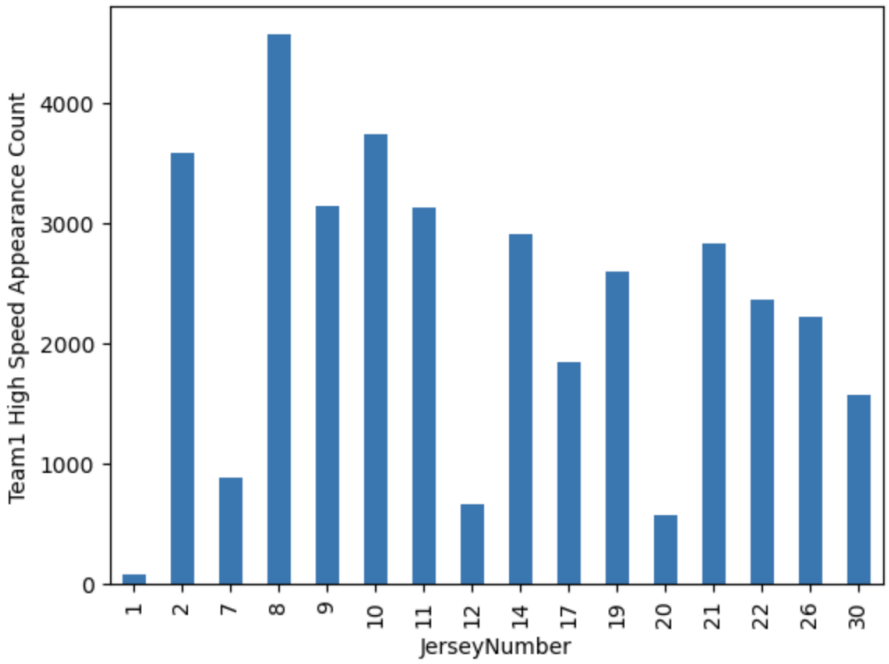


STL VS CIN (5:1):

STL:



CIN:



Conclusion:

Based on the analysis above, we think STL performed the best in their pressing strategy in the game Versus CLT. SEA showed the most resistance to STL City SC in their pressing strategy. Compared to itself, STL performed well in the first two matches against ATX and CLT. In the other three games, there was not an extraordinary performance although they achieved an impressive record, especially in the last game against CIN with a 5:1 score.
For an individual comparison with STL and ATX

Appendix:

Player Physical OutPut

1. **ATX VS STL (2:3)**

Legend	Position	Jersey Number	Player Name	Minutes_Played(min)	Total_Distance(m)	High_Speed_Distance(m)	Sprint_Distance(m)	Sprint_Count
St. Louis CITY SC	**Central Striker**							
Austin FC	Center Forward	9	Klauss	92	9425	556	198	11
		9	Gyasi	90	10664	961	176	14

			Zardes					
	Advanced Midfielders							
	Left	8	Jared Stroud	84	9989	884	173	8
		14	Diego Fagundez	79	9280	618	191	10
	Mid	7	Tomas Ostrak	65	8268	639	101	4
		10	Sebastian Driussi	99	11041	883	248	12
	Right	21	Rasmus Alm	64	7947	709	432	6
		7	Emiliano Rigoni	99	11451	871	306	14
	Defensive Midfielders							
	Left	10	Eduard Lowen	100	11914	858	107	8
		6	Daniel Pereira	101	12215	807	250	10
	Right	19	Indiana Vassilev	14	1218	116	60	2
		33	Owen Wolff	80	9877	570	115	5
	Back Defenders							

	Left Outside Back	14	John Nelson	102	11185	586	356	10
		23	Zan Kolmanic	74	8487	768	162	8
	Center Back Left	22	Kyle Hiebert	101	10828	518	227	10
		15	Leo Vaisanen	100	10954	413	115	8
	Center Back Right	26	Tim Parker	101	10241	515	109	8
		18	Julio Cascante	9	978	31	4	1
	Right Outside Back	2	Jakob Nerwinski	100	11689	703	220	8
		24	Nick Lima	99	11337	710	79	6
	Goal Keeper							
		1	Roman Burki	101	6241	79	0	0
		1	Brad Stuver	101	5689	48	14	2
	Substitutes							
		(19)6	Njabulo Blom	84	10106	523	22	6
		(21)11	Nicholas Gioacchini	37	4498	322	161	7
		(8)12	Celio	15	1931	238	83	2

			Martins					
		(9)15	Joshua Yaro	8	1035	9	16	1
		(7)28	Miguel Perez	36	4350	242	70	2
		(18)4	Kipp Keller	91	10595	547	159	7
		(33)8	Alexander Ring	21	2297	149	40	2
		(14)13	Ethan Finlay	21	2599	206	67	3
		(23)17	Jon Gallagher	27	3273	271	157	3
		(9)37	Maximiliano Urruti	9	936	26	0	0

2. STL VS CLT (3:1)

Legend	Position	JerseyNumber	PlayerName	Minutes_Played(min)	Total_Distance(m)	High_Speed_Distance(m)	Sprint_Distance(m)	Sprint_Count
St. Louis CITY SC	**Central Striker**							
Charlotte FC		16	Samuel Adeniran	72	8168	821	74	4
		11	Jon Gallagher	77	8710	529	252	7
		9	Klauss	94	9459	440	217	8
		9	Will	96	9686	493	251	12

			Bruin					
	Advanced Midfielders	10	Sebastian Driussi	95	10889	625	105	7
		19	Indiana Vassilev	19	2287	185	53	2
		8	Jared Stroud	70	8306	1007	209	7
		21	Rasmus Alm	27	3359	376	95	4
		6	Daniel Pereira	97	10473	638	146	5
		7	Tomas Ostrak	76	8993	860	129	7
		28	Ethan Finlay	4	606	91	16	1
		2	Alex Roldan	96.115556	11297.73633	685.873333	165.942	8
		6	Xavier Arreaga	19.023333	2341.718667	129.135667	55.218	3
		7	Yeimar Gomez	62.525	7603.124333	592.138667	82.519667	5
		8	Joao Paulo	59.27	7924.114667	821.931667	158.313333	10
		9	Will Bruin	96.080556	9685.949333	492.854333	251.134667	12
		10	Jimmy Medranda	75.061111	9301.22	545.181667	91.14	7
		14	Nicolas Lodeiro	95.676111	10939.57167	709.481667	145.737	11

	Back Defenders							
		19	Raúl Ruidíaz	76.820556	8976.146667	712.070667	236.528	7
		20	Kelyn Rowe	19.061111	2470.498	118.192	0	0
		21	Josh Atencio	34.93	4264.940667	397.032333	43.782	4
		26	Tim Parker	99	9742	516	185	6
	Goal Keeper							
		1	Roman Burki	101	6241	79	0	0
		24	Shane O'Neill	98.235	11033.8806	735.223667	159.969333	11
		26	Abdoulaye Cissoko	95.888889	9986.569	513.823	79.587333	6
	Substitutes							
		39	Benjamin Lundt					
		17	Selmir Pidro					
		20	Akil Watts					
		15	Joshua Yaro					
		12	Celio Martins					

		25	Harrison Afful					
		24	Jaylin Lindsey					
		33	Patrick Agyemang					
		28	Joseph Mora					
		31	Benjamin Bender					

3. **RSL VS STL (0:4)**

Legend	Position	JerseyNumber	PlayerName	Minutes_Played(min)	Total_Distance(m)	High_Speed_Distance(m)	Sprint_Distance(m)	Sprint_Count
St. Louis CITY SC	**Central Striker**							
Real Salt Lake		16	Maximiliano Urruti	21	2208	171	59	2
		11	Diego Fagundez	77	749	172	5	6
		9	Klauss	99	599	265	5	8
		2	Anton Tinnerholm	97	11173	925	286	7
		4	Maxime Chanot	96	9550	404	108	8
	Advanced Midfielders	6	Daniel Pereira	14	1372	120	14	1

		7	Emiliano Rigoni	22	2240	161	25	1
		8	Tomas Ostrak	72	8630	500	64	4
		10	Sebastian Driussi	77	9168	465	52	2
		19	Jesus Ferreira	98	10311	632	85	2
		7	Jesus Medina	96	10648	683	180	6
		8	Keaton Parks	72	8630	500	64	4
		16	Chris Gloster	23	2622	149	9	1
		17	Jesús López	24	2766	157	23	3
		18	Ismael Tajouri-Shradi	98	5031	21	0	0
		25	Sean Johnson	96	10183	679	95	9
		28	Gudmundur Thórarinsson	96	11721	779	135	7
	Back Defenders							
		15	Sam Fink	98	11009	611	176	10
		23	Zan Kolmanic	24	2980	256	23	2
		25	Paris Gee	96	10183	679	95	9
		2	Sean Reynolds	97	11173	925	286	7
		4	Julio	96	9550	404	108	8

			Cascante					
		18	Ismael Tajouri-Shradi	98	5031	21	0	0
		23	Malte Amundsen	24	2980	256	23	2
		9	Valentín Castellanos	61	6947	410	183	8
		10	Maximiliano Moralez	36	3398	281	130	3
		11	Thiago Andrade	73	7568	550	200	6
	Goal Keeper							
		15	Alexander Callens	98	11009	611	176	10
	Substitutes							
		33	Moses Nyeman					
		27	Bertin Jacquesson					
		24	Tomas Gomez					
		12	Celio Martins					
		12	Scott Caldwell					
		33	Moses Nyeman					
		27	Bertin Jacquesson					

		24	Tomas Gomez					
		22	Delentz Pierre					

4. **SEA VS STL (3:0)**

Legend	Position	Jersey Number	PlayerName	Minutes_Played(min)	Total_Distance(m)	High_Speed_Distance(m)	Sprint_Distance(m)	Sprint_Count
St. Louis CITY SC	**Central Striker**							
Seattle Sounders FC		16	Maximiliano Urruti	23	2622	149	9	1
		11	Diego Fagundez	73	7568	550	200	6
		9	Klauss	61	6947	410	183	8
		5	Justen Glad	97	9696	685	193	4
		7	Pablo Ruiz	98	11476	844	244	4
	Advanced Midfielders	7	Emiliano Rigoni	96	10648	683	180	6
		8	Tomas Ostrak	72	8630	500	64	4
		17	Indiana Vassilev	24	2766	157	23	3
		18	Kipp Keller	98	5031	21	22	4
		11	Andres Gomez	97	11242	564	136	5
		16	Maikel	84	9287	537	174	7

			Chang					
		23	Ilijah Paul	64	6210	407	167	3
		73	Zac MacMath	5	760	68	29	1
	Back Defenders							
		2	Sean Reynolds	99	11685	894	234	5
		12	Nicholas Hinds	18	2060	283	27	2
		14	Jared Stroud	99	11335	889	206	4
		2	Sean Reynolds	97	11173	925	286	7
		28	Jasper Löeffelsend	97	9742	426	111	5
		9	Emeka Eneli	35	3581	240	95	1
	Goal Keeper							
		10	Sebastian Driussi	36	3398	281	130	3
		10	Jefferson Savarino	99	12778	1052	173	6
	Substitutes							
		77	Sota Kitahara					
		3	Xavier					

			Arreaga					
		12	Fredy Montero					
		45	Ethan Dobbelaere					
		12	Scott Caldwell					
		33	Moses Nyeman					
		27	Bertin Jacquesson					
		24	Tomas Gomez					
		22	Delentz Pierre					

5. **STL VS CIN (5:1)**

Legend	Position	JerseyNumber	Player Name	Minutes_Played (min)	Total_Distance(m)	High_Speed_Distance(m)	Sprint_Distance(m)	Sprint_Count
St. Louis CITY SC	**Central Striker**							
		9	Klauss	94	9184	641	437	14
FC Cincinnati		11	Jon Gallagher	76	9236	638	236	11
		21	Matt Miazga	96	9544	464	171	8
		19	Brandon Vázque	96	9814	587	190	8

			z					
	Advanced Midfielders	10	Sebastian Driussi	95	11409	749	72	5
		20	N/A	18	2404	117	15	1
		19	Indiana Vassilev	77	8468	529	158	7
		8	Jared Stroud	82	9887	928	237	10
		21	Rasmus Alm	63	7265	577	267	9
		30	Miguel Perez	32	3832	321	163	3
		4	Nick Hagglund	49	4869	239	91	4
		5	Obinna Nwobodo	62	6165	342	117	2
		8	Marco Angulo	47	5390	479	123	2
		13	Santiago Arias	90	9126	631	246	6
		15	Yerson Mosquera	95	9388	685	314	10
	Back Defenders							
		26	Tim Parker	95	9531	448	102	4
		14	Diego F	64	6979	591	215	9

		2	Sean Reynolds	97	11173	925	286	7
		28	Raymon Gaddis	6	729	119	58	1
		29	Arquimides Ordonez	49	4702	270	102	6
	Goal Keeper							
		7	Tomas Ostrak	18	2243	183	104	2
		31	Álvaro Barreal	95	9967	772	225	7
		93	Júnior Moreno	93	9267	437	105	6
	Substitutes							
		24	Lucas Bartlett					
		39	Benjamin Lundt					
		28	Miguel Perez					
		15	Joshua Yaro					
		33	Isaiah Foster					
		3	Joey Akpunonu					
		32	Ian					

			Murphy					
		1	Alec Kann					