Causal Relationship Between Active Players And Twitch Viewership

Team 2

Chun Yat (Jeco) Cheung

Yixuan (Ethan) Liu

Yu (Sherry) Shi

July 12, 2024

Table of Content

Objective

To analyze the causal relationship between a game's Twitch viewership and its active player engagement, using the case of Tom Clancy's Rainbow Six Siege as a primary example, and extending the analysis to Counter-Strike 2 and Rust.

Background

In February 2024, Ubisoft's Six Invitational 2024, co-streamed by Twitch's most subscribed channel, Jynxzi, set a record for the highest peak viewership (521,349 peak viewers) in the event's history. Subsequently, Rainbow Six Siege achieved a record high in average player count (over 60,000) since April 2021. This event highlights the potential influence of Twitch viewership on player engagement. Given the rapid growth of the live-streaming industry, understanding this relationship can offer valuable insights for gaming companies to strategize their marketing and engagement efforts.

Data Description

Due to limited access to historical data from Twitch and Steam APIs, we used a combination of datasets from two third-party websites: SteamDB.com and Sullygnome.com. These websites are online statistics and analytics databases for Steam and Twitch, respectively. According to the platforms' websites, APIs are used to collect information and are polled every 5-10 minutes to ensure data completeness and accuracy.

Tom Clancy's Rainbow Six Siege is the primary example for this project. Additionally, we included other games for reference based on the following criteria:

1. Must be online multiplayer shooting games.
2. Must have more than 5 years of data to ensure data sufficiency.
3. Must be available on the Steam platform.

Based on these criteria, Counter-Strike 2 and Rust were selected as reference games for our analysis.

The original data from SteamDB and Sullygnome provided 26 columns with daily player engagement and Twitch streaming information for each game, such as Daily Peak Players, Twitch Daily Peak Viewers, and Number of Channels. We manually added 6 new columns to account for potential confounding variables that could affect Twitch viewership and active player counts: "Day of the Week", "Original Price", "Discount" (the amount of discount in percentage), "Free Week/Weekend" (a binary variable that indicates whether the game is free on that day), “Events” (a string variable that contains information of special events, such as marketing campaigns), and "Tournament (INTL)" (a binary variable that indicates whether there is international tournament). This resulted in three datasets, each with 32 columns, one for each target game. Below is a sample of the list of variables.

['DateTime', 'Day of the Week', 'Followers', 'Players', 'Average Players', 'Twitch Viewers', 'Positive reviews', 'Negative reviews', 'Rating', 'Final price', 'Historical low', 'Original price', 'discount', 'Free Weekend / Free Week', 'Tournament (INTL)', 'Events', 'Peak viewers', 'Average viewers', 'Peak channels', 'Average channels', 'Viewer ratio (Average view / Average Channel)', 'Top channel', '2nd channel', '3rd channel', '4th channel', '5th channel', 'Channels 6-10', 'Channels 11-25', 'Channels 26-50', 'Channels 51-100', 'Channels 101-250']

Data Preprocessing

Exploratory Data Analysis

The ideal KPI for study would be Daily Active Users (DAU), a commonly used metric in the gaming industry counting unique daily users, but due to privacy policies, it is not usually disclosed to the public. In our datasets, “Players”, daily peak players, and “Average Players” are two potential KPIs to be considered, and they display similar trends across the whole dataset. However, “Average Players” only have values for less than two years, and because it is unclear how API derives the average value, it is impossible for us to impute the average values. Therefore, we decided to use “Players”, the poxy that is similar to DAU and has no missing values, as our KPI although the values are slightly lower than real DAU.

Since there are missing values in some columns in our raw data, we omitted columns that are not helpful in this study or have better substitutes, such as “Followers”, “Average Players”, “Positive reviews”, and “Negative reviews”. Other columns, “Free Weekend / Free Week”, “Tournament (INTL), and “Events”, were converted into binary values, where NAs were denoted as 0 (not happening) and non-NAs were

turned into 1 (happening). Besides, for the purpose of controlling a specific day's influence on players, we encoded "Day of the Week" into seven separate dummy variables. Moreover, the lowest game price is an important factor leading to users' behaviors, so to control for the price's impact, we created a new binary variable "is_historical_low" by incorporating "Final price" and "Historical low".

Stationarity and Deseasonality

It is crucial to understand the trend of time series data and the autocorrelation, implying seasonality, within variables. Figure 1-3 plot the trend of "Peak viewers" and "Players" across the data. From a descriptive perspective, when "Peak viewers" reaches a spike, "Players" seems to have a peak immediately or within a short period of time, which laid the foundation of our study.

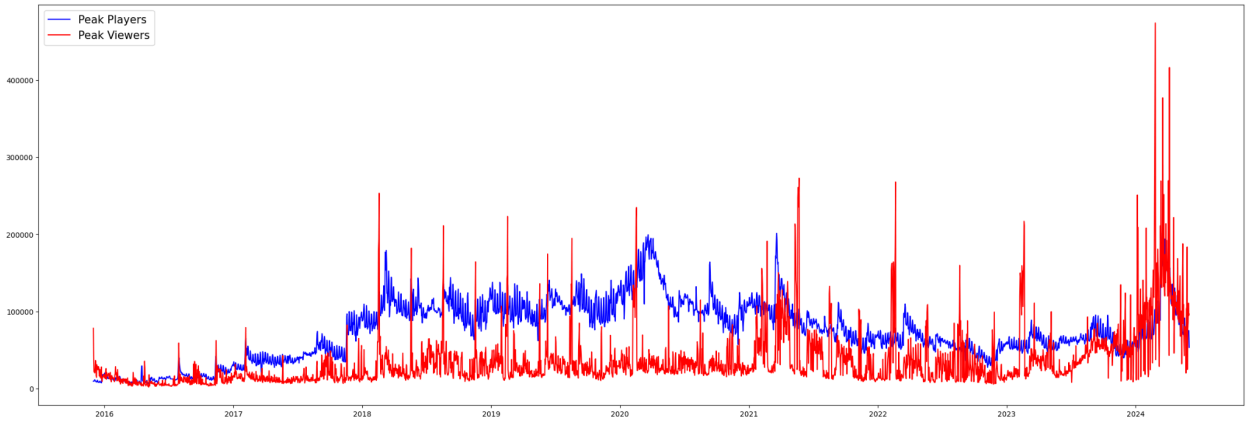

Figure 1: Plot of "Peak viewers" and "Players" of Raw Rainbow Six Siege Data

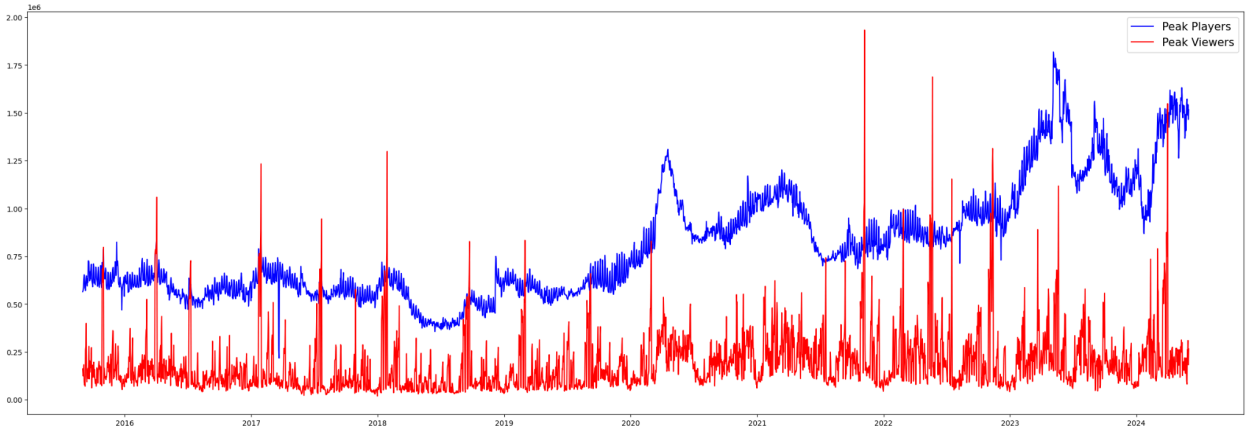

Figure 2: Plot of "Peak viewers" and "Players" of Raw Counter-Strike 2 Data

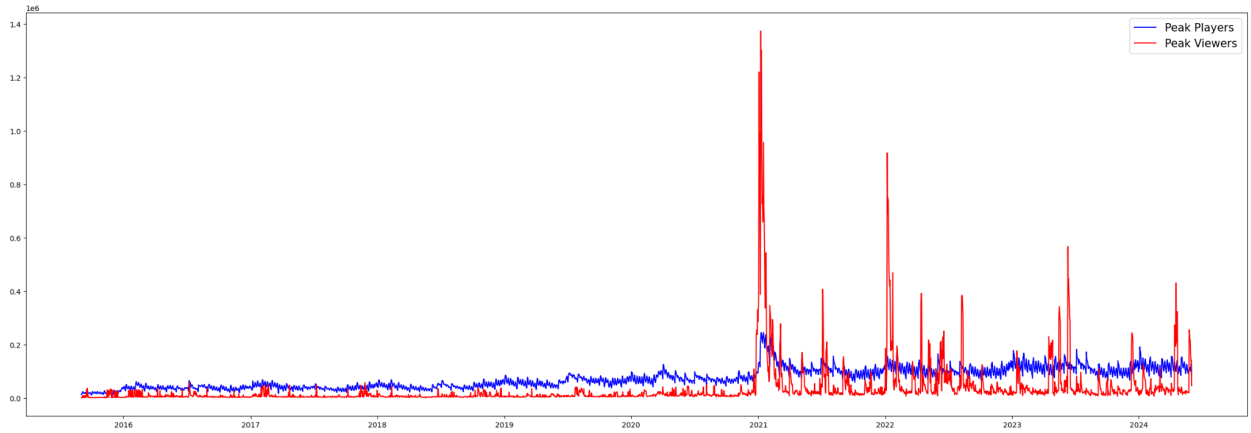

Figure 3: Plot of "Peak viewers" and "Players" of Raw Rust Data

Time series data are expected to be stationary during any modeling, which refers to constant means and constant variance. We applied the Augmented Dicky-Fuller Test (ADF Test), a statistical significance test used to test time series' stationarity, to all of the numerical features. The test result showed that the target variable "Players" is not stationary in all three games with a higher-than-threshold p-value. "Final price" in Counter-Strike 2 and "Rating" in Rust are also not stationary. Therefore, to keep scales on the same level, we took the first difference for all the numerical variables, and they became stationary when we reapplied the ADF Test. Figure 4-6 displays the plot of "Peak viewers" and "Players" after first differencing.

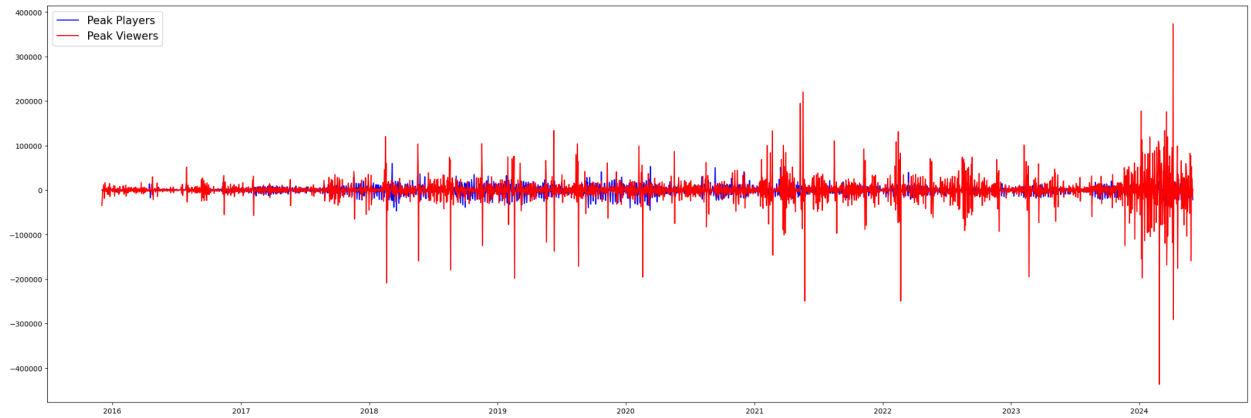

Figure 4: Plot of "Peak viewers" and "Players" of Rainbow Six Siege Data After First Difference

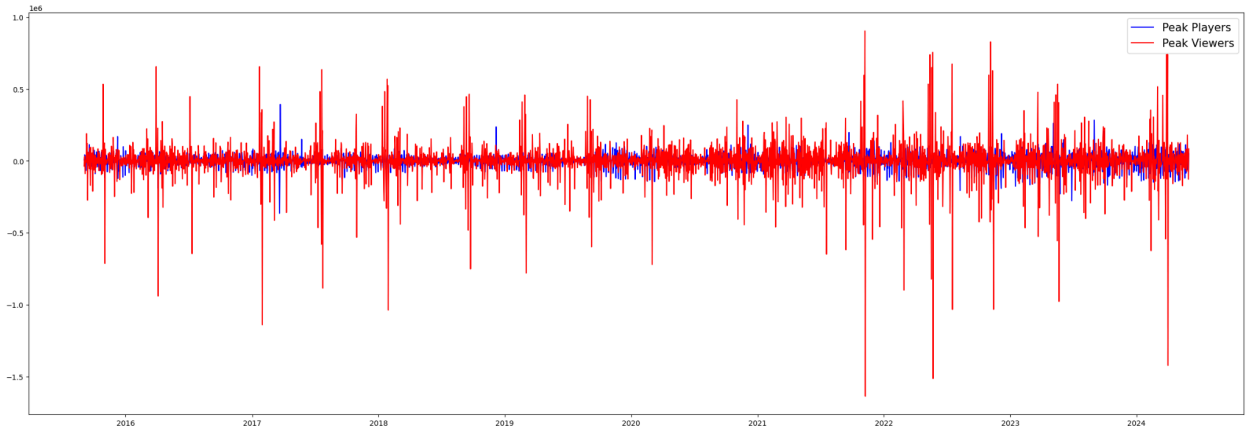

Figure 5: Plot of “Peak viewers” and “Players” of Counter-Strike 2 Data After First Difference

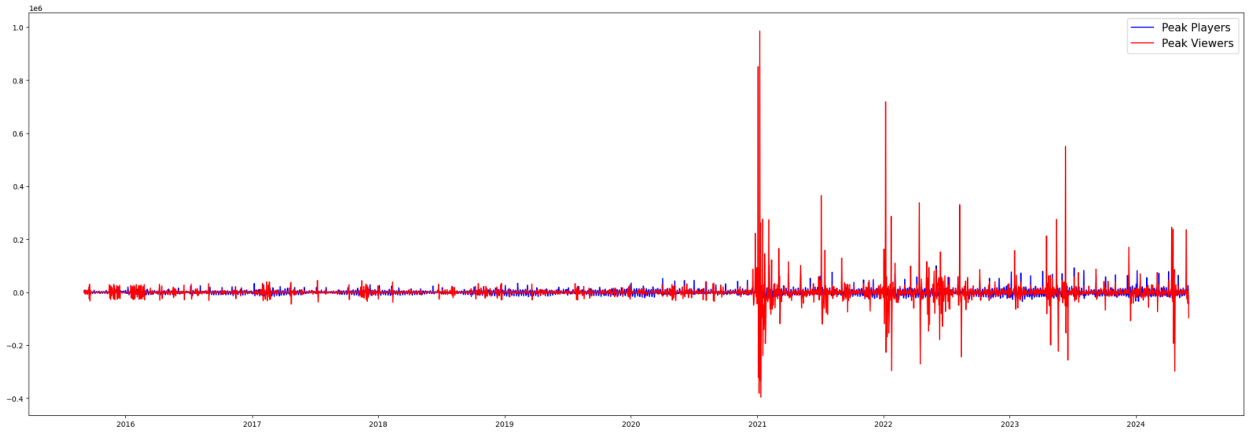

Figure 6: Plot of “Peak viewers” and “Players” of Rust Data After First Difference

Moreover, seasonality is common in time series but it is not preferred when performing modeling analyses. Seasonality can be checked by autocorrelation scores ranging from -1 to 1, where the closer the magnitude to 1, the stronger the seasonality. We detected strong seasonality in “Players” in all three games. There are seasonal trends approximately every seven days with an autocorrelation score higher than 0.6 (Figure 5-7 in Appendix). Therefore, to remove the seasonality in “Players”, we derived a new variable “deseasonlized_players” by subtracting the decomposed seasonal value of players from “Players”. The new variable has no strong seasonality as most of the autocorrelation scores dropped under 0.25 (Figure 8-10 in Appendix).

Lag

Not only can factors at time t have causal relationships with players, but historical data can also have potential causal relationships with players. To perform thorough analyses of causality, we shifted “Peak

viewers” from t-1 to t-14 and “Rating”, “Final Price”, and “is_historical_low” from t-1 to t-7 to mimic the lag effect. These lag variables are used selectively in the following modeling section.

Modeling

Linear Regression

Before implementing deeper analyses of the causal relationship between Players and Twitch viewers, we performed linear regressions on players to understand how Twitch viewership is potentially quantitatively associated with Players under the influence of confounding variables. To keep the results' robustness and reflect the true relationship between two variables, we used cleaned data to fit models for all three games even though outputs showed weaker associations.

Rainbow Six Siege

To control the confounding effect, we included “Rating”, “Final price”, Days of the Week, “Free Weekend / Free Week (bool)”, “Tournament (INTL)”, “is_historical_low_shift_5”, “Final price_shift_3”, and “Final price_shift_4” as our confounding variables. Trying a full model with “Peak viewers” and all its lags variables on “deaseaonalized_players”, we decided to check the association with “Peak viewers” and its 1, 6, and 13 lags respectively since they appear to have more significant results.

As the OLS outputs shown in Table 1, p-values of “Peak viewers” with shifts 1, 6, and 13 indicate that these three variables are statistically significant by using 0.05 as the threshold. Their positive coefficients and 95% confidence intervals imply there might be a potential positive effect of around 0.008 on the number of players. However, “Peak viewers” proves a weaker relationship. The p-value of 0.069 suggests the result is marginally significant with a likely positive influence but it is not convincing enough to be considered significant at the 5% significance level. The [-0.001, 0.017] confidence interval for the coefficient of “Peak viewers” also suggests that there might be some uncertainty about the true effects. Thus, we concluded that there is possibly a positive correlation between players and peak viewers but needs further analysis to test the magnitude.

Table 1: OLS Regression Results for Rainbow Six Siege

Dep. Variable:	deseasonalized_players	R-squared:	0.018
Model:	OLS	Adj. R-squared::	0.012
Method:	Least Squares	Prob (F-statistics):	8.47e-06

	coef	P > | t |	[0.025	0.975]
const	-100.7951	0.391	-330.978	129.388
Peak viewers	0.0083	0.069	-0.001	0.017
Rating	49.6415	0.976	-3171.857	3271.14
Final price	-13.0917	0.735	-88.818	62.635
Monday	-49.1228	0.873	-653.769	555.524
Tuesday	154.4082	0.612	-442.405	751.222
Wednesday	147.4945	0.625	-444.085	739.074
Thursday	-37.8509	0.901	-636.195	560.493
Friday	-22.0609	0.942	-618.656	574.534
Saturday	-69.2453	0.819	-662.78	524.29
Sunday	-224.4179	0.464	-824.746	375.91
Free Weekend / Free Week (bool)	2758.2477	0.000	1345.384	4171.112
Tournament (INTL)	997.6599	0.197	-518.007	2513.327
is_historical_low_shift_5	59.1223	0.876	-685.896	804.143
Peak viewers_shift_1	0.0097	0.032	0.001	0.019
Peak viewers_shift_6	0.0086	0.044	0.000	0.017
Peak viewers_shift_13	0.0104	0.016	0.002	0.019
Final price_shift_3	-100.8643	0.009	-176.771	-24.958
Final price_shift_4	-104.1269	0.007	-179.92	-28.334

Counter-Strike 2

In Counter-Strike 2's linear regression model, we considered "Rating", "Final price", "Final price_shift_4", "Tournament (INTL)" "Events", Days of the Week, and "is_historical_low_shift_2" to control for confounding variables. "Peak viewers" and its shift of 1 and 2 days were included to test the association with players.

The model summary indicates that all three variables are statistically significant with p-values of 0.002, 0.006, and 0.001 respectively. The coefficients of these three variables are greater than 0.01, and their

95% confidence intervals are within the range of positive numbers. Hence, we expect to achieve a positive causal effect estimation of Twitch viewers on players for this game in the following analyses.

Table 2: OLS Regression Results for Counter-Strike 2

Dep. Variable:	deseasonalized_players	R-squared:	0.009	
Model:	OLS	Adj. R-squared::	0.005	
Method:	Least Squares	Prob (F-statistics):	0.0144	
	coef	**P > | t |**	**[0.025**	**0.975]**
const	241.4805	0.802	-1648.611	2131.572
Peak viewers	0.0133	0.016	0.002	0.024
Rating	7.938e+04	0.051	-194.726	1.59e+05
Final price	676.8137	0.485	-1222.369	2575.996
Tournament (INTL)	1863.8389	0.545	-4175.751	7903.429
Events	1.715e+04	0.168	-7211.545	4.15e+04
Friday	-406.3806	0.8	-3549.538	2736.776
Monday	225.6946	0.895	-3135.037	3586.427
Saturday	-439.8242	0.785	-3605.914	2726.265
Sunday	-1097.3885	0.5	-4288.138	2093.361
Thursday	-585.899	0.716	-3737.498	2526.7
Tuesday	1609.403	0.346	-1740.213	4959.019
Wednesday	935.8752	0.584	-2419.323	4291.073
Peak viewers_shift_1	0.0172	0.002	0.006	0.028
Peak viewers_shift_2	0.0116	0.033	0.001	0.022
Final_price_shift_4	-2527.681	0.009	-4415.118	-640.244
is_historical_low_shift_2	38.6516	0.977	-2622.028	2699.331

Rust

We used similar confounding variables for Rust, except we substituted “is_historical_low_shift_2” with “is_historical_low”. “Peak viewers” with lags of 4, 6, 11, and 13 days were taken into account as variables in this model.

As all five peak viewers variables are statistically significant, “Peak viewers” at time t, t-4, t-6, and t-13 display positive coefficients in predicting players, as well as their corresponding 95% confidence intervals. On the contrary, “Peak viewers” at time t-11 shows a negative coefficient with the magnitude of 0.0124 on players. It is highly likely that “Peak viewer” at different lags can have distinct effects on players. According to the overall outputs we derived from this model, it is more likely that Twitch viewers of Rust will have a positive relationship with players.

Table 3: OLS Regression Results for Rust

Dep. Variable:	deseasonalized_players	R-squared:	0.025	
Model:	OLS	Adj. R-squared::	0.02	
Method:	Least Squares	Prob (F-statistics):	2.82e-10	
	coef	**P > | t |**	**[0.025**	**0.975]**
const	-14.0932	0.92	-288.583	260.396
Peak viewers	0.0227	0.000	0.016	0.03
Rating	1.172e+04	0.203	-6331.677	2.98e+04
Final price	-44.9266	0.461	-164.296	74.443
Tournament (INTL)	-1000.0289	0.631	-5076.666	3076.608
Events	3565.9543	0.317	-3424.556	1.06e+04
Friday	-86.8187	0.82	-835.123	661.486
Monday	70.5528	0.854	-679.756	820.862
Saturday	-50.5195	0.894	-793.339	692.3
Sunday	102.8997	0.787	-642.014	847.814
Thursday	-179.6083	0.637	-926.813	567.597
Tuesday	197.6223	0.604	-549.671	944.915
Wednesday	-68.2215	0.858	-814.583	678.14
Peak viewers_shift_4	0.0103	0.004	0.003	0.017
Peak viewers_shift_6	0.0129	0.000	0.006	0.02
Peak viewers_shift_11	-0.0124	0.001	-0.02	-0.005
Peak viewers_shift_13	0.0093	0.01	0.002	0.016
Final_price_shift_4	-151.6197	0.013	-271.001	-32.238

is_historical_low	318.0461	0.632	-982.618	1618.71

Granger Causality Test

Before proceeding to other more complex algorithms, we applied the Granger Causality Test to determine if there was any evidence implying the relationship between players and Twitch viewers. The Granger Causality Test is a statistical concept used to learn the predictability of one time series on another. It is noteworthy that this test does not provide insights into the true causal relationship, instead it can tell important lags of the predicting variable.

Since the Granger Causality Test requires data to be stationary and with no seasonality, we used stationary and deseasonalized players and peak viewers' values. In this study, we mainly test the formula $Players_t = const + \gamma_1 Players_{t-1} + \ldots + \gamma_p Players_{t-p} + \alpha_1 Viewers_{t-1} + \ldots + \alpha_p Viewers_{t-p} + \epsilon$ on the following hypotheses:

H_0: “Peak viewers” does not granger-cause "deseasonalized_players” ($\alpha_1 = \ldots = \alpha_p = 0$)

H_1: “Peak viewers” granger-causes “deseasonalized_players” (at least one of the $\alpha \neq 0$)

The maximum lag order of this test was determined by running the vector autoregressive model between “Peak viewers” and “deseasonalized_players” to select orders using AIC. Utilizing the significance level of 0.05, we can conclude from Table 4 that “Peak viewers” granger causes “deseasonalized_players” starting from a lag of 1 for Rainbow Six Siege and Counter-Strike 2 and from a lag of 2 for Rust. This test result helps us choose the optimal lag of variables in the CDMI algorithm.

Table 4: Granger Causality Test Results for Three Games

	Rainbow Six Siege	Counter-Strike 2	Rust
Number of lags 1	p = 0.045	p = 0.0203	p = 0.3242
Number of lags 2	p = 0.0024	p = 0.0049	p = 0.0053
Number of lags 3	p = 0.0001	p = 0.0005	p = 0.0001
Number of lags 4	p = 0.0002	p = 0.0009	p = 0.0014
Number of lags 5	p = 0.0016	p = 0.0013	p = 0.000
Number of lags 6	p = 0.000	p = 0.0014	p = 0.000
Number of lags 7	p = 0.000	p = 0.0002	p = 0.000
Number of lags 8	p = 0.000	p = 0.0002	p = 0.000

Number of lags 9	p = 0.0002	p = 0.0003	p = 0.000
Number of lags 10	p = 0.0003	p = 0.0006	p = 0.000
Number of lags 11	p = 0.001	p = 0.0011	p = 0.000
Number of lags 12	p = 0.0011	p = 0.0029	p = 0.000
Number of lags 13	p = 0.0004	p = 0.0046	p = 0.000
Number of lags 14	p = 0.0001	p = 0.0056	p = 0.000
Number of lags 15	p = 0.0002	-	p = 0.000

Causal Discovery Using Model Invariance(CDMI)

To infer causality between the predictor, Peak Viewers Diff, and the response variable, Peak Players Diff, we leveraged a methodology called Causal Discover Using Model Invariance (CDMI) via knockoffs.

Proposed by Ahmad, Shadaydeh, and Denzler (2022) in their paper, 'Causal Discovery using Model Invariance through Knockoff Interventions', the method uses DeepAR to model nonlinear interactions in multivariate time series and employs knockoff variables to create interventional environments, testing for causal relationships by comparing the invariance of the response residuals using statistical tests like the Kolmogorov-Smirnov test. If the residual distribution remains unchanged under intervention, the predictor is deemed non-causal; otherwise, it is considered causal.

In our implementation of the method, we chose to substitute DeepAR with simpler DNNs to model our time series due to the constraints of computational resources and various challenges during implementation attempts. The DNN was trained with the ADAM optimizer with an Exponential Decaying learning rate and its structure is as follows:

Table 5: DNN Structure

Layer Name	Type	Output Shape	Parameters
dense	Dense	(None, 64)	2,176
dropout	Dropout	(None, 64)	0
dense_1	Dense	(None, 32)	2,080
dropout_1	Dropout	(None, 32)	0
dense_2	Dense	(None, 32)	1,056

dropout_2	Dropout	(None, 32)	0
dense_3	Dense	(None, 1)	33

- Total params: 5345 (20.88 KB)
- Trainable params: 5345 (20.88 KB)
- Non-trainable params: 0 (0.00 Byte)

The other steps remain the same, as Python codes were adapted from the paper's Github pages.

The identical causal discovery pipeline was applied to all three games. However, input variables were inconsistent across each game as certain variables were dropped due to the mathematical constraint of the knockoff generating algorithm. Among the dropped variables, there were no main predictors of interest (i.e., Peak Viewers Diff and its lagged versions) involved, and dropped variables were highly correlated with other variables with different time lags, which we believe have little effect on our final results. The detailed list of variables could be found below in each game's section. Through CDMI via Knockoff, we identified several causal links between Peak Viewer Diff and Peak Player Diff with varied time lags across all 3 games.

Rainbow Six Siege

A total number of 33 variables were included in the model. The variables are categorized as follows:

Primary Variables:

- Peak viewers
- Final price
- Is_historical_low

Lagged Variables:

- Peak viewers (shifted by 1 to 14 days):
    - Peak viewers_shift_1
    - Peak viewers_shift_2
    - Peak viewers_shift_3
    - Peak viewers_shift_4
    - Peak viewers_shift_5
    - Peak viewers_shift_6
    - Peak viewers_shift_7
    - Peak viewers_shift_8

    - Peak viewers_shift_9
    - Peak viewers_shift_10
    - Peak viewers_shift_11
    - Peak viewers_shift_12
    - Peak viewers_shift_13
    - Peak viewers_shift_14
- Rating (shifted by 1 and 7 days):
    - Rating_shift_1
    - Rating_shift_7
- Final price (shifted by 1 to 7 days):
    - Final price_shift_1
    - Final price_shift_2
    - Final price_shift_3
    - Final price_shift_4
    - Final price_shift_5
    - Final price_shift_6
    - Final price_shift_7
- is_historical_low (shifted by 1 to 7 days):
    - is_historical_low_shift_1
    - is_historical_low_shift_2
    - is_historical_low_shift_3
    - is_historical_low_shift_4
    - is_historical_low_shift_5
    - is_historical_low_shift_6
    - Is_historical_low_shift_7

The data was split into training and test sets using TimeSeriesSplit from the scikit-learn library with a 3:1 ratio. The DNN model was then trained on the training set and validated on the test set, resulting in a validation MAPE of 117.40%. After DNN training, knockoff variables were generated using the DeepKnockoff package. Knockoff and original variables were compared using the compute_diagnostics function predefined by Ahmad et al. (2022) to check if they are statistically similar but decorrelated pairwise. The results are as follows:

Table 6: Knockoff Results (Rainbow Six Siege)

Metric	Swap	Value
Covariance	self	5.647262e-01
Covariance	full	4.172939e+19
MMD	full	0.000000e+00
KNN	full	8.521036e-01

Energy	full	4.373484e+03
Covariance	partial	4.269365e+19
MMD	partial	0.000000e+00
KNN	partial	7.779935e-01
Energy	partial	4.218953e+03

The causal discovery was conducted using the causal_inference_knockoff function, adapted from the original research paper, by comparing the residual (MAPE) distributions before and after knockoff intervention. Five significant causal predictors were identified at a significance level of 0.05. Specifically, Peak viewers_shift_5 had a p-value of 0.033542, Peak viewers_shift_6 had a p-value of 0.000270, Peak viewers_shift_7 had a p-value of 0.012299, Peak viewers_shift_10 had a p-value of 0.033542, and Peak viewers_shift_11 had a p-value of 0.003967.

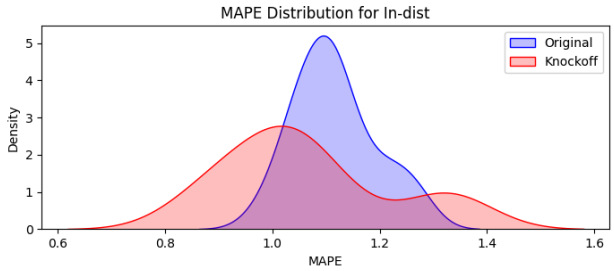

Figure 7: MAPE Distributions for ‘Peak viewers_shift_5’

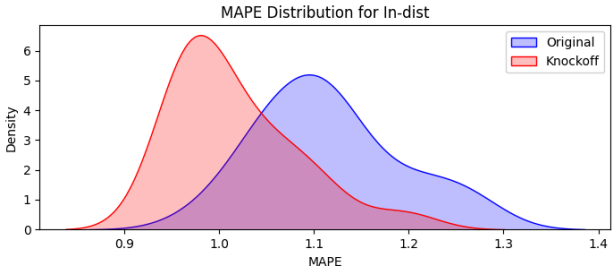

Figure 8: MAPE Distributions for ‘Peak viewers_shift_6’

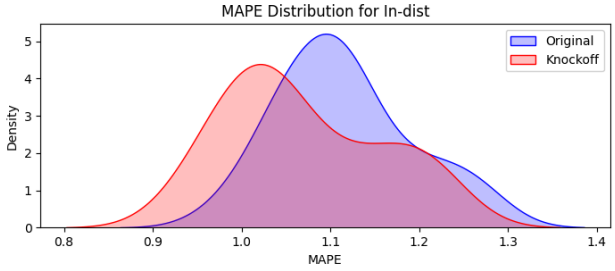

Figure 9: MAPE Distributions for ‘Peak viewers_shift_7’

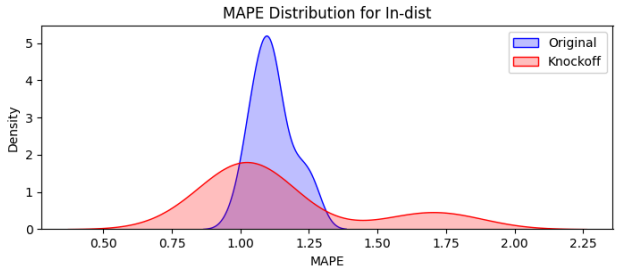

Figure 10: MAPE Distributions for 'Peak viewers_shift_10'

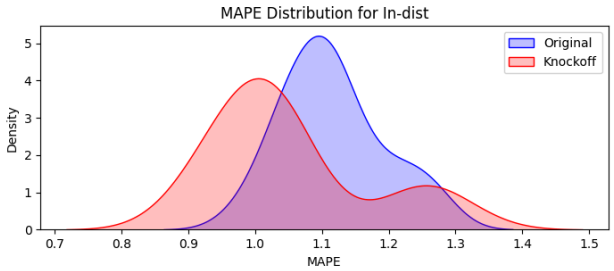

Figure 11: MAPE Distributions for 'Peak viewers_shift_11'

Counter-Strike 2

For Counter-Strike 2, a total of 28 variables were included in the model, some variables of rating and is_historical_low were dropped due to the failure to produce knockoffs by the algorithm. The remaining variables are categorized and listed as follows:

Primary Variables:

- Peak viewers
- Final price
- is_historical_low

Lagged Variables:

- **Peak viewers (shifted by 1 to 14 days):**
  - Peak viewers_shift_1
  - Peak viewers_shift_2
  - Peak viewers_shift_3
  - Peak viewers_shift_4
  - Peak viewers_shift_5
  - Peak viewers_shift_6
  - Peak viewers_shift_7
  - Peak viewers_shift_8
  - Peak viewers_shift_9
  - Peak viewers_shift_10
  - Peak viewers_shift_11
  - Peak viewers_shift_12

    - Peak viewers_shift_13
    - Peak viewers_shift_14
- **Rating (shifted by 1 and 7 days):**
    - Rating_shift_1
    - Rating_shift_7
- **Final price (shifted by 1 to 7 days):**
    - Final price_shift_1
    - Final price_shift_2
    - Final price_shift_3
    - Final price_shift_4
    - Final price_shift_5
    - Final price_shift_6
    - Final price_shift_7
- **is_historical_low (shifted by 2, 4, and 6 days):**
    - is_historical_low_shift_2
    - is_historical_low_shift_4
    - is_historical_low_shift_6

The data was then split into training and test sets using TimeSeriesSplit from the scikit-learn library in a 3:1 ratio. A new DNN model was trained, producing a validation MAPE of 114.20%. Knockoff variables were generated again using the DeepKnockoff package. After obtaining the knockoff variables, a comparison was made to get the following metrics results:

Table 7: Knockoff Results (Counter-Strike 2)

Metric	Swap	Value
Covariance	self	5.252502e-01
Covariance	full	7.163853e+21
MMD	full	0.000000e+00
KNN	full	8.188679e-01
Energy	full	9.874250e+03
Covariance	partial	7.392684e+21
MMD	partial	0.000000e+00
KNN	partial	7.264151e-01
Energy	partial	9.287500e+03

However, Causal discovery using Model Invariance with Counter-Strike 2 data did not identify any significant causal variables. From a business perspective as well as visual observation of the data, this result is understandable in the case of Counter-Strike. As one of the most played games on Steam, Counter-Strike's long-standing popularity predates the emergence of Twitch as a streaming platform. Consequently, the effect of Twitch in bringing new activity to the game is likely very limited.

Rust

For Rust, a total of 32 variables were included in the model. These variables are categorized and listed as follows:

Primary Variables:

- Peak viewers
- Final price
- is_historical_low

Lagged Variables:

- **Peak viewers (shifted by 1 to 14 days):**
    - Peak viewers_shift_1
    - Peak viewers_shift_2
    - Peak viewers_shift_3
    - Peak viewers_shift_4
    - Peak viewers_shift_5
    - Peak viewers_shift_6
    - Peak viewers_shift_7
    - Peak viewers_shift_8
    - Peak viewers_shift_9
    - Peak viewers_shift_10
    - Peak viewers_shift_11
    - Peak viewers_shift_12
    - Peak viewers_shift_13
    - Peak viewers_shift_14
- **Rating (shifted by 1 and 7 days):**
    - Rating_shift_1
    - Rating_shift_7
- **Final price (shifted by 1 to 7 days):**
    - Final price_shift_1
    - Final price_shift_2
    - Final price_shift_3
    - Final price_shift_4
    - Final price_shift_5

    - Final price_shift_6
    - Final price_shift_7
- **is_historical_low (shifted by 1 to 7 days):**
    - is_historical_low_shift_1
    - is_historical_low_shift_2
    - is_historical_low_shift_3
    - is_historical_low_shift_4
    - is_historical_low_shift_5
    - is_historical_low_shift_6
    - Is_historical_low_shift_7

The same process was applied to the data, comparison metrics between the knockoff and original variables are as follows:

Table 8: Knockoff Results (Rust)

Metric	Swap	Value
Covariance	self	3.565873e-01
Covariance	full	2.542798e+20
MMD	full	0.000000e+00
KNN	full	9.654088e-01
Energy	full	8.171688e+03
Covariance	partial	2.920133e+20
MMD	partial	0.000000e+00
KNN	partial	8.927673e-01
Energy	partial	6.987219e+03

For the Rust data, Peak viewers_shift_5 (p = 0.033542), Peak viewers_shift_6 (p = 0.012299), and Peak viewers_shift_14 (p = 0.001116) were identified as significant causal predictors for Peak Player Diff.

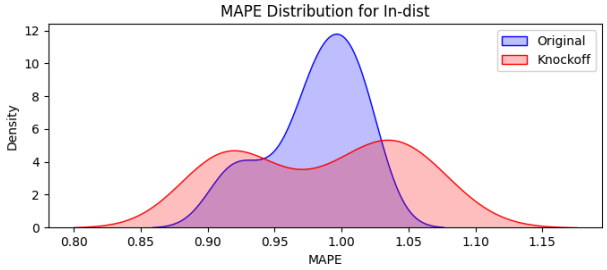

Figure 12: MAPE Distributions for ‘Peak viewers_shift_5’

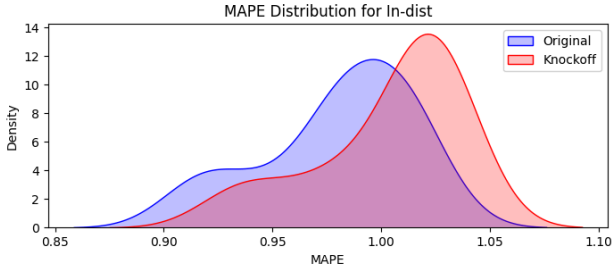

Figure 13: MAPE Distributions for ‘Peak viewers_shift_6’

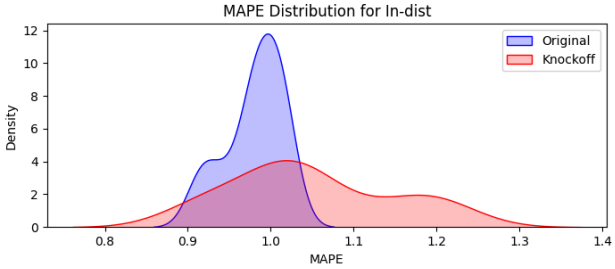

Figure 14: MAPE Distributions for ‘Peak viewers_shift_14’

Latent Peter and Clark Momentary Conditional Independence (LPCMCI)

To cross-validate the CDMI results and further estimate the causal effect, we employed the PCMCI algorithm to identify causal relationships and interactions between multiple time series. Specifically, we used Latent-PCMCI (LPCMCI) to handle unobserved confounders. The algorithm was implemented with the TIGRAMITE library in Python.

The inputs for the algorithm were two time series: Peak Viewer Diff and Peak Player Diff. For the conditional independence test, we used the partial correlation test. We chose tau_max = 14 to look back 14 days (i.e., lagged by 1 to 14 days) and used the default pc_alpha = 0.05 as the significance threshold.

After obtaining the LPCMCI results for causal link identification, we proceeded to estimate the causal effect of our predictor on the response using the CausalEffects class from the TIGRAMITE library with adjustment sets. However, 2 out of the 3 outputs (specifically, the causal graphs from t-14 to t for Rainbow Six Siege and Rust) could not be solved by the estimation algorithm due to the graph’s complexity involving too many nodes and edges.

To quantify the causal effect as much as possible, we simplified the graphs to retain as many causal relationships as possible while allowing the algorithm to solve for an estimation. The simplification was done by pruning links with higher p-values, one by one, until optimal adjustment sets could be found, minimizing potential bias introduced to the model. If links with relatively higher p-values were pruned and the optimal set still could not be found, we further pruned randomly selected links until a solution was

found. Throughout this process, causal links of interest (i.e., links from Peak Viewer Diff to Peak Player Diff) remained untouched.

Rainbow Six Siege

Two significant causal links from Peak Viewer Diff to Peak Player Diff were found. Specifically, Peak Viewer Diff at time t-1 was identified to cause Peak Player Diff at time t with a p-value of 0.04696, and Peak Viewer Diff at time t-14 was identified to cause Peak Player Diff at time t with a p-value of 0.00173. The time series DAG plot of the algorithm output is shown below:

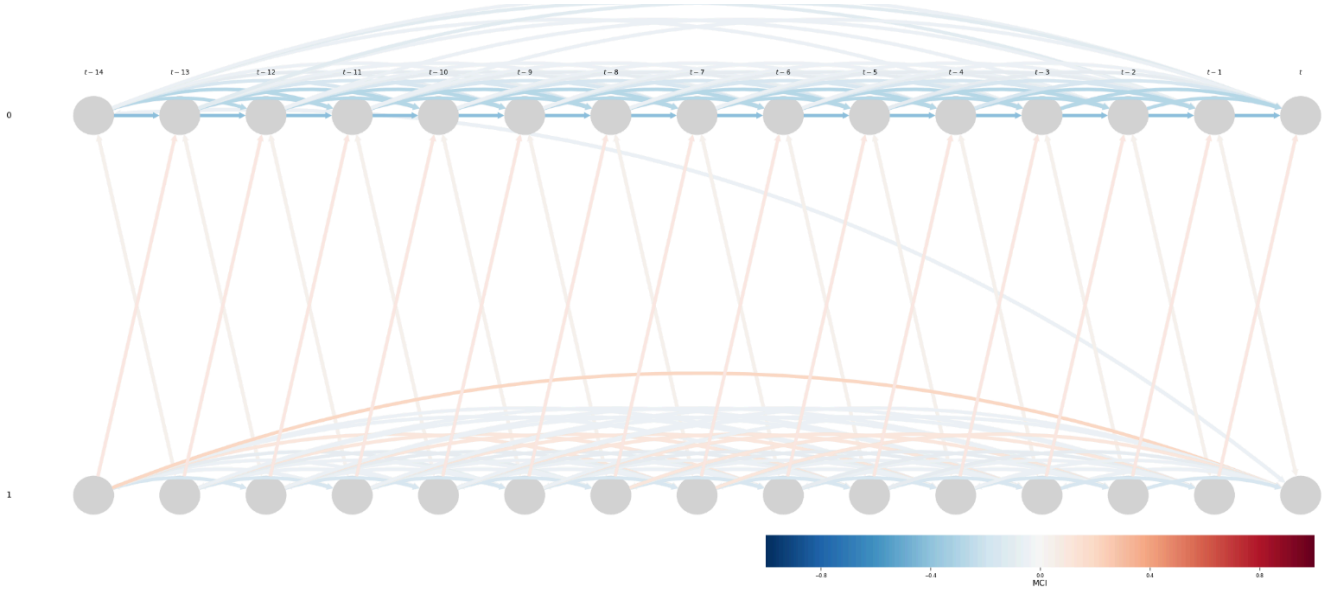

Figure 15: Time Series DAG for Peak Viewer Diff (0) and Peak Player Diff (1) in Rainbow Six Siege

Causal graph pruning was applied to the time series DAG because the algorithm failed to find an optimal adjustment set for causal effect identification and estimation. A total of 15 links were pruned to obtain a solution, leaving only 5 links, including the 2 targeted links. This pruning could heavily impact the estimation validity. The pruned causal DAG is shown below:

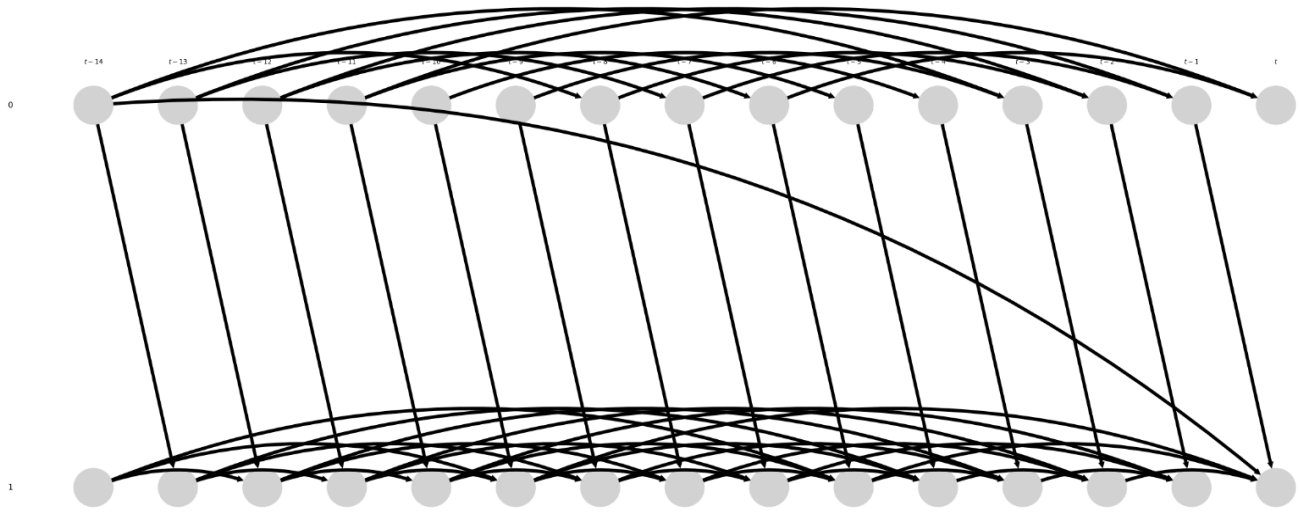

Figure 16: Pruned Time Series DAG for Peak Viewer Diff (0) and Peak Player Diff (1) in Rainbow Six Siege

The causal effect was then estimated using a linear regression approximation by observing the change in the response variable when incrementing the inputs by 1. The combined effect of Peak Viewer Diff lag 1 and Peak Viewer Diff lag 14 is -0.0030. The individual effects of the two variables were 0.0119 and -0.0130, respectively. This unexpected result, which contradicts findings from the other two games, might be attributed to the heavily pruned DAG and the resulting loss of significant links that could otherwise alter the results.

Counter-Strike 2

Two significant causal links from Peak Viewer Diff to Peak Player Diff were found. Specifically, Peak Viewer Diff at time t-1 was identified to cause Peak Player Diff at time t with a p-value of 0.00162, and Peak Viewer Diff at time t-2 was identified to cause Peak Player Diff at time t with a p-value of 0.02314. The time series DAG plot of the algorithm output is shown below:

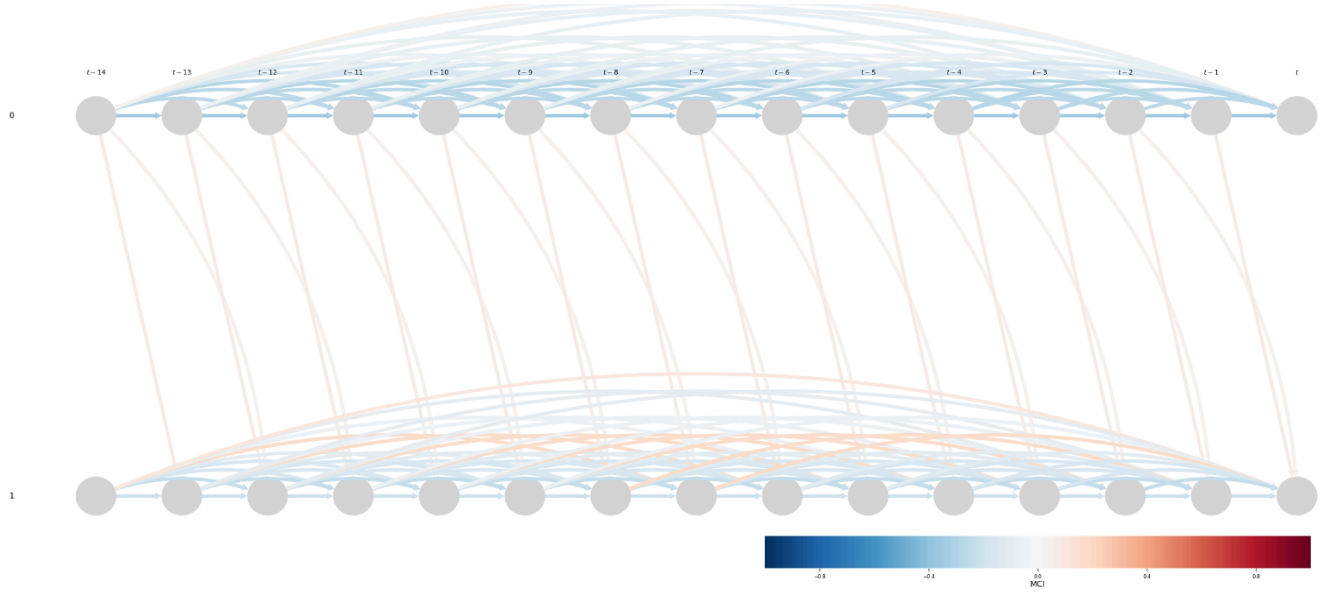

Figure 17: Time Series DAG for Peak Viewer Diff (0) and Peak Player Diff (1) in Counter-Strike 2

The causal effect estimation for Counter-Strike 2 ran successfully without any need for pruning. A combined effect of 0.0235 was identified, suggesting a 2% conversion ratio from Twitch viewership to active players for this game.

Rust

Again, two significant causal links from Peak Viewer Diff to Peak Player Diff were found. Specifically, Peak Viewer Diff at time t was identified to cause Peak Player Diff at time t with a p-value of 0.00000, and Peak Viewer Diff at time t-4 was identified to cause Peak Player Diff at time t with a p-value of 0.00105. The time series DAG plot of the algorithm output is shown below:

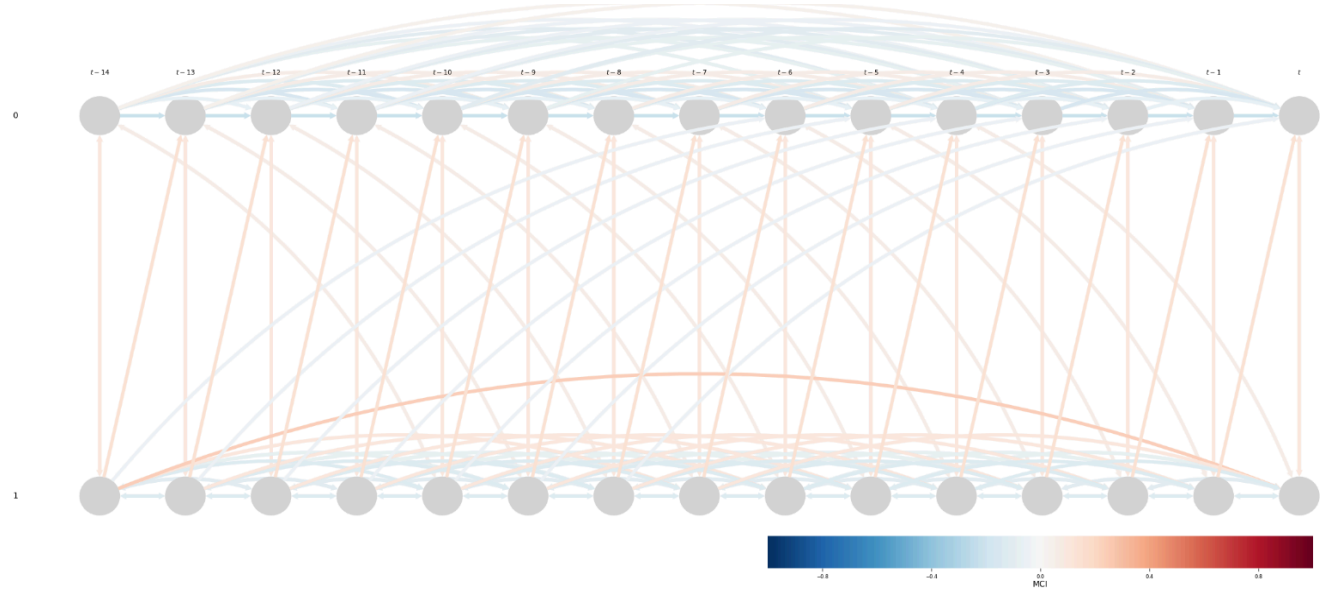

Figure 18: Time Series DAG for Peak Viewer Diff (0) and Peak Player Diff (1) in Rust

Causal graph pruning was again applied to the time series DAG because the algorithm failed to find an optimal adjustment set for causal effect identification and estimation. A total of 9 links were pruned to get a final solution, leaving 11 links untouched, including the 2 targeted links. The pruned causal DAG is shown below:

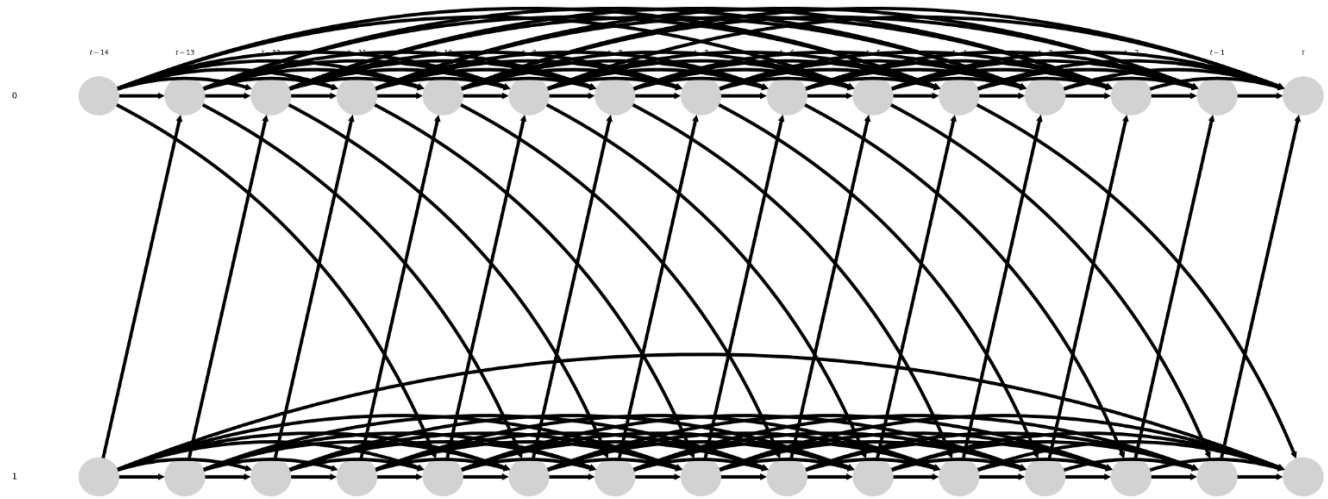

Figure 19: Pruned Time Series DAG for Peak Viewer Diff (0) and Peak Player Diff (1) in Rust

The causal effect estimation was conducted, and a combined causal effect of 0.0418 was identified, suggesting a 4% conversion ratio from Twitch viewership to active players for Rust.

Modeling Results

Based on comprehensive results from four different modeling approaches applied to three selected games, we found evidence of a causal relationship between Twitch peak viewers and Steam peak players. Our analysis suggests that increased viewership on Twitch tends to bring new active players to the games investigated.

We also discovered that this causal relationship usually exhibits a time delay, with the impact from viewership to players typically taking between 1 to 2 weeks to manifest. However, there is also evidence that an immediate impact can occur under certain circumstances, and the duration of these delays can vary between games.

Finally, we obtained estimates of the causal effect for each game. Despite the range of effect sizes spanning from -0.001 to 0.04, estimates between 0.02 and 0.04 seem more reliable, considering the significant pruning of the graph that could potentially lead to biased results. The varying results also suggest between-game variation, as each game possesses unique characteristics and player bases.

Follow-up Analysis on Twitch Viewership

After confirming the causal relationship between Twitch viewership and player engagement, the business question for companies becomes how to increase their viewership. To address this, we conducted a follow-up analysis based on the recent success of Rainbow Six Siege (RS6). Our case study focused on two key factors: Influence of Top Streamers and Impact of Tournaments.

1. **Influence of Top Streamers:** To understand the influence of top streamers over the years, we analyzed the annual total viewership from the top channel compared to all channels, calculating the ratio. Among our three target games, Rainbow Six Siege exhibited a significantly increasing trend in top channel influence starting from 2020, rising from 27.2% to 47.5% (corresponding visualizations are attached in the Appendix). This suggests that top streamers have had a notable impact on RS6's Twitch viewership over years. This could potentially explain the huge success of the Six Invitational 2024, which was co-streamed with top streamer Jynxzi.
2. **Impact of Tournaments:** We examined whether holding tournaments effectively promotes a game.
    - **Player and Viewership Comparison:**

- During tournaments, we observed a 9.78% increase in players and a 319% increase in Twitch viewership compared to non-tournament periods.

- **Monthly Viewership Analysis:**

  - Grouping RS6's viewership data by month, we observed a noticeable spike in viewership during tournament periods. This pattern was consistent across the two other target games, Counter-Strike 2 and Rust, though not to the same extent.

3. These findings indicate that holding tournaments with top streamers is effective in drawing attention from both new and current players for RS6. However, similar events (co-streaming tournaments with top streamers) were not observed for the other target games, suggesting that the impact of tournaments and top streamers can vary significantly between games.

From our analysis, we learned that Twitch viewership distribution, the streaming environment, and the nature of each game are unique. Therefore, it is not universally applicable to conclude that all gaming companies should collaborate with top Twitch streamers or organize tournaments. Instead, we developed a more general procedure to help companies devise effective marketing strategies tailored to their specific games.

Recommendations

General Procedure for Developing Marketing Strategies

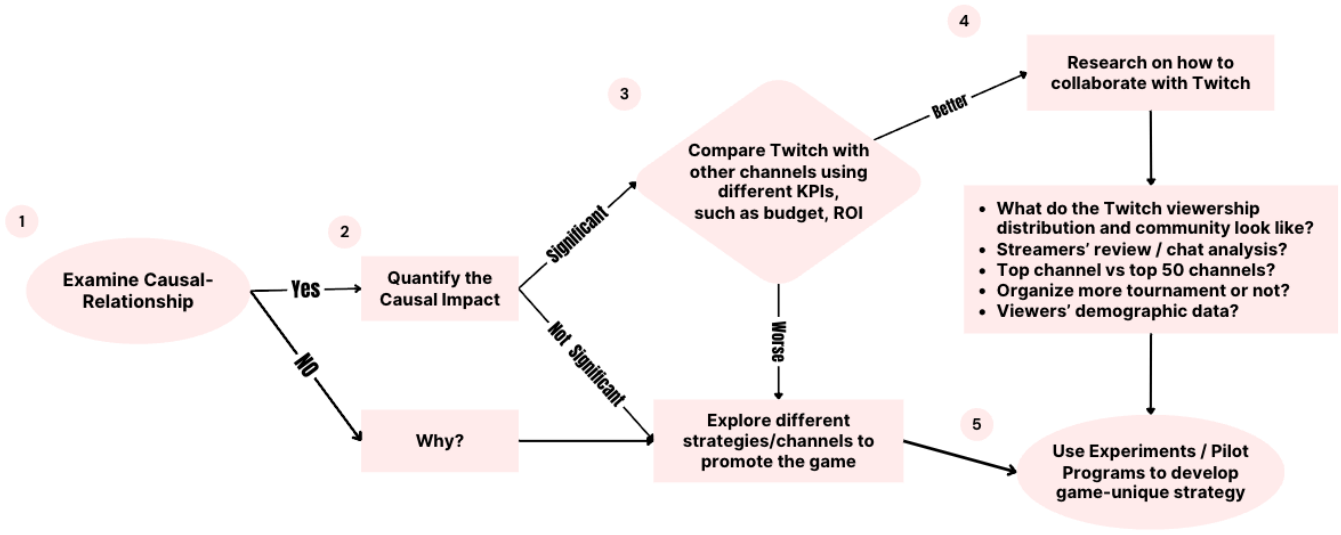

Figure 20: General Procedure for Developing Marketing Strategies

To develop a game-unique marketing strategy, companies should first examine the relationship between active players and Twitch viewership to determine if there is a causal relationship or correlation, and assess the strength of this relationship. If the relationship is significant, they should compare Twitch with other traditional channels for promoting the game using different KPIs, such as budget and ROI. In researching effective collaboration with Twitch, companies might consider partnering with top streamers who have a significant influence on viewership, as seen in Rainbow Six Siege, or collaborating with multiple channels if top streamer influence is less pronounced.

Focus on Game Development

When our team was researching the impact of different marketing strategies, we noticed that most events only create short-term effects (appendix). Therefore, alongside these efforts, companies must focus on game development by continuously improving user experience through new game modes, features, and robust anti-cheat systems to keep players engaged. Developing innovative content and ensuring a fair gaming environment are crucial for retaining and attracting players in the long term. Being responsive to user feedback is equally important; companies should actively collect and utilize feedback to guide game development and improvements, ensuring that player concerns and suggestions are addressed promptly.

Utilize Twitch / Streaming as a Feedback Channel

Additionally, companies can leverage Twitch and other streaming platforms as feedback channels, treating streamers' reviews and chat interactions as valuable sources of user insights. Compared to the traditional feedback channels, like players’ comments, ratings, and DAU, this approach allows for more direct and real-time feedback and helps identify areas for enhancement. By using these feedbacks to inform game updates and development strategies, companies can create a more engaging and satisfying experience for players, ultimately fostering a loyal and active player base.

Limitations

During our project, we identified several limitations that we believe could be improved upon with further opportunities. One significant limitation is the need for 'true' experiments to establish causal inference. Our analysis would benefit from more behavioral and demographic data on both viewers and players. This additional data would provide deeper insights into how different factors influence engagement and

viewership. Understanding the nuances of player behavior and preferences is crucial for refining marketing strategies and enhancing player experience.

Another area for improvement is the development of more complex and accurate models to quantify the relationship between viewers and players. While our current models provide valuable insights, more sophisticated modeling techniques could yield even more precise predictions and identify key factors influencing this relationship. Advanced models would allow us to better understand trends, predict future behavior, and make more informed decisions about marketing and game development.

Additionally, we recognize the need for more in-depth game information, including updates and features, to understand the impact of these elements on viewership. Analyzing the specific features and updates that drive player engagement and viewership can provide actionable insights for game developers. Furthermore, a detailed study of between-game discrepancies would help us understand why Twitch has a more significant causal influence on some games compared to others. Identifying these differences can help tailor marketing strategies to the unique dynamics of each game.

In conclusion, addressing these limitations would significantly enhance the robustness of our findings. By conducting true experiments for causal inference, gathering more behavioral and demographic data, developing more complex models, and conducting in-depth studies on game-specific factors and between-game discrepancies, we can gain a more comprehensive understanding of the relationship between Twitch viewership and player engagement. These improvements would allow us to provide more actionable recommendations for game developers and marketers, ultimately leading to more effective strategies and better gaming experiences.

Reference

Ahmad, W., Shadaydeh, M., & Denzler, J. (2022). Causal discovery using model invariance through knockoff interventions. *In Proceedings of the ICML 2022 Workshop on Spurious Correlations, Invariance, and Stability*, Baltimore, Maryland, USA. arXiv:2207.04055. https://doi.org/10.48550/arXiv.2207.04055

Runge, J. (n.d.). *Tigramite: Python package for causal inference in time series.* GitHub. Retrieved July 12, 2024, from https://github.com/jakobrunge/tigramite

Via, D. (2024, June 28). *Siege reaches its highest player peak on Steam in almost three years*. SiegeGG. https://siege.gg/news/siege-reaches-its-highest-player-peak-on-steam-in-almost-three-years

Appendix

Table 1: Dataset sample (Rainbow Six Siege)

```
[5] df
```

	DateTime	Day of the Week	Followers	Players	Average Players	Twitch Viewers	Peak viewers	Positive reviews	Negative reviews	Rating	...	Top channel	2nd channel	3rd channel	4th channel	5th channel	Channels 6-10	Channels 11-25	Channels 26-50	Channels 51-100	Channels 101-250
0	12/1/15	Tuesday	NaN	9375	NaN	78000	78000	34	-27	55.737705	...	10155.0	3511.0	2101.0	1128.0	946.0	3090.0	3047.0	1230.0	521.0	356.0
1	12/2/15	Wednesday	NaN	9644	NaN	43081	43081	384	-40	86.185567	...	6179.0	3902.0	2537.0	1858.0	1448.0	3878.0	3130.0	1087.0	534.0	380.0
2	12/3/15	Thursday	NaN	9574	NaN	21731	21731	331	-40	87.500000	...	4705.0	2314.0	1504.0	1052.0	682.0	1788.0	1903.0	844.0	416.0	323.0
3	12/4/15	Friday	NaN	9760	NaN	22240	22240	215	-61	85.159011	...	3047.0	1815.0	1205.0	859.0	542.0	1581.0	1531.0	586.0	323.0	270.0
4	12/5/15	Saturday	NaN	11185	NaN	20170	20170	145	-52	83.446200	...	4288.0	2093.0	1332.0	976.0	737.0	1924.0	1495.0	564.0	339.0	308.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3101	5/28/24	Tuesday	1234810.0	73587	59641.50000	63649	77159	212	-184	85.161084	...	4561.0	2642.0	2010.0	1398.0	1051.0	2708.0	2025.0	1058.0	693.0	658.0
3102	5/29/24	Wednesday	1234919.0	72059	56994.41667	85116	85042	198	-193	85.150607	...	11108.0	2167.0	1422.0	999.0	749.0	2172.0	1760.0	901.0	699.0	681.0
3103	5/30/24	Thursday	1235028.0	75042	57615.54167	109551	110558	243	-142	85.144024	...	17711.0	2189.0	1643.0	1150.0	872.0	2194.0	1914.0	1001.0	655.0	675.0
3104	5/31/24	Friday	1235145.0	74767	59003.37500	81217	94463	251	-90	85.140972	...	10498.0	2247.0	1602.0	1270.0	995.0	2504.0	1937.0	889.0	687.0	680.0
3105	6/1/24	Saturday	1235269.0	53164	59860.00000	84980	96354	178	-62	85.138929	...	11008.0	1721.0	1215.0	975.0	827.0	2554.0	1771.0	754.0	613.0	639.0

3106 rows × 31 columns

Figure 1-3: Top channel influence analysis

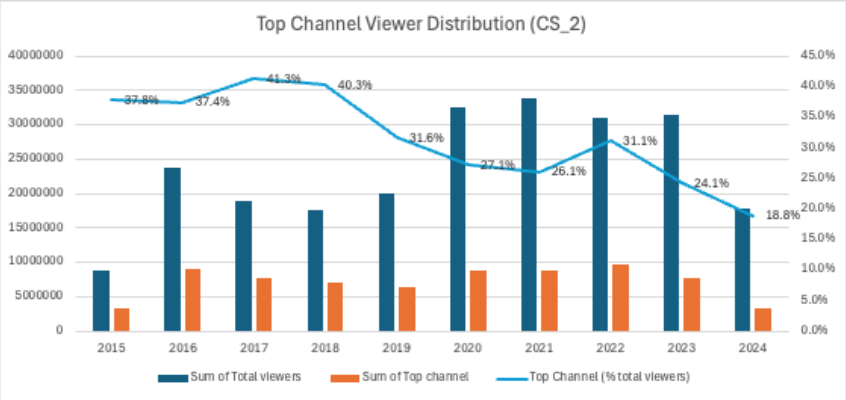

Figure 4: Other marketing strategies analysis

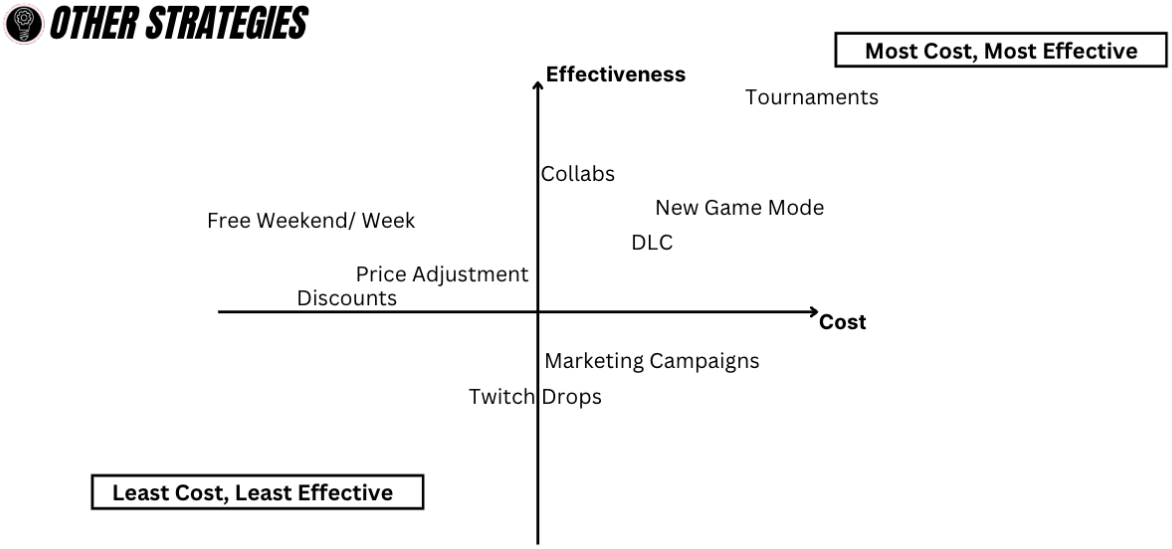

Figure 5: ACF and PACF for “Players” Before Deseasonalization (Rainbow Six Siege)

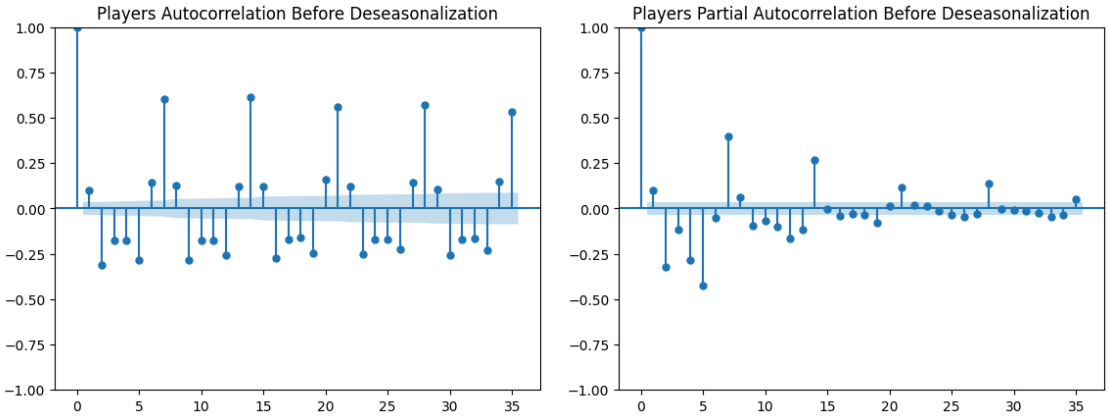

Figure 6: ACF and PACF for “Players” Before Deseasonalization (Counter-Strike 2)

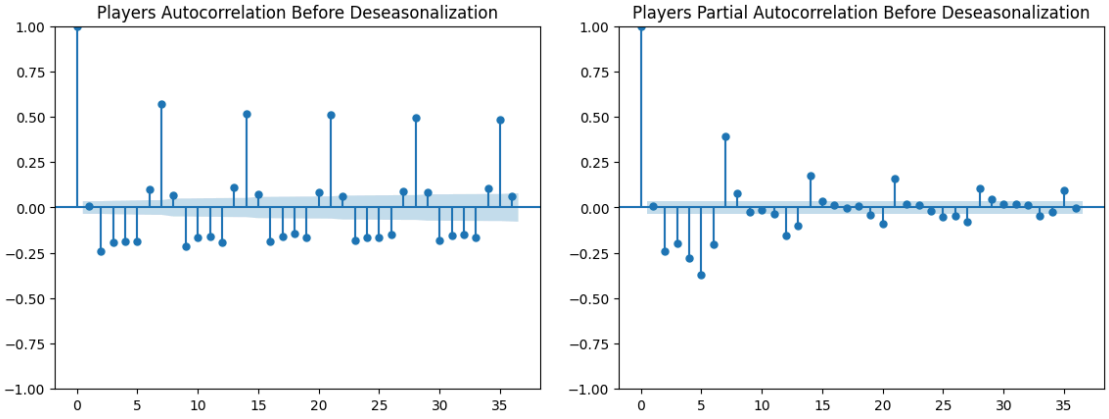

Figure 7: ACF and PACF for “Players” Before Deseasonalization (Rust)

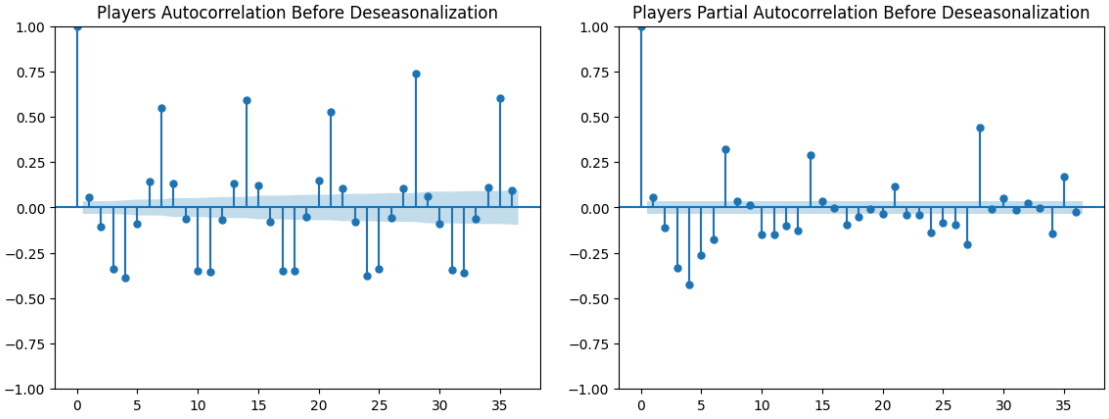

Figure 8: ACF and PACF for “deseasonalized_players” (Rainbow Six Siege)

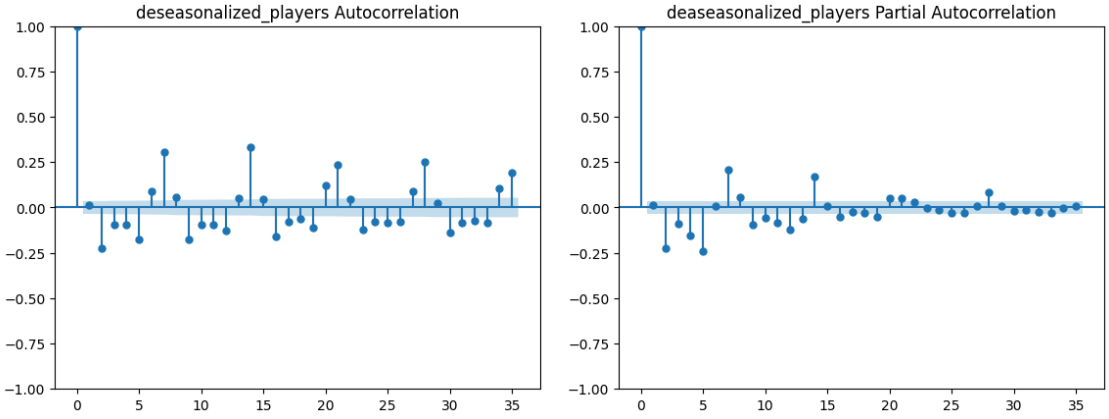

Figure 9: ACF and PACF for “deseasonalized_players” (Counter-Strike 2)

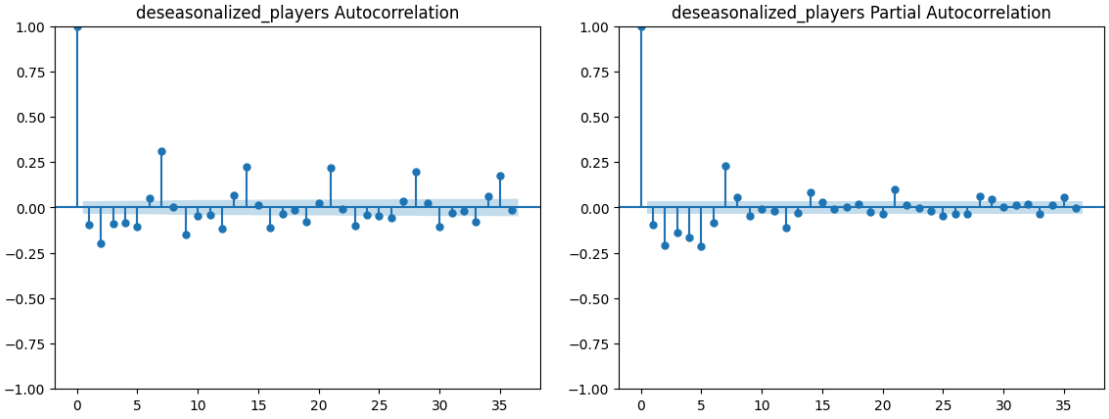

Figure 10: ACF and PACF for “deseasonalized_players” (Rust)