Hot Hand in Basketball: A Time Series Analysis

Anonymous (Blinded for Peer Review)

1 Introduction and Motivation

Basketball is often described as a game of runs, meaning that each team will go on a run where they score several points in a row, mutliple times a game. On an individual level, players are thought to have hot streaks, where they perform better than usual for a small period of time, in-game or across them. This is referred to as "having a hot hand" or on a longer scale, "a linsanity run". Inspired by a past project that modeled baseball team momentum as a POMP model (Author 2025a), this project aims to model the hot hand phenomenon in basketball using a similar POMP modeling approach. It will answer the question:

Is there statistical evidence for the hot hand phenomenon across games in basketball?

As stated in the previous project, baseball has the distinct advantage of more data points and a simpler game structure to model, unlike basketball, which has more factors influencing the outcome of a game, and fewer observations. However, basketball has the advantage of players having more stable within-game performances, as they usually play most of the game and have a lot of chances to score. This higher volume of invovlement helps reduce variability and noise at the game level, making it feasible to model the hot hand phenomenon based on observable statistics, like points.

2 Data Description and Exploration

We decided to intially focus on the MVP (Most Valuable Player) of the 2024-25 NBA season, Shai Gilgeous-Alexander (SGA). Data was collected from the 'nba_api' Python package (Patel 2018), which provides easy access to a wide range of NBA statistics. We extracted game logs for SGA's season, including both regular season and playoff games. For each game, we collected the following variables:

- **Points (PTS)**: The number of points SGA scored in the game, which serves as our primary measure of performance and our observed variable.
- **Rolling Defensive Rating**: A centered two-week rolling average, leading up to the game, of the opponent's defensive rating, which estimates the number of points the opponent allows per 100 possessions. This serves as a covariate to account for the ability of the opponent to stop scoring. *Note: Due to the rolling nature of this variable, the first few games of the season will have less data points in the calculation, and thus may be less accurate. Similarly, in the*

playoffs when the teams play each other multiple times in a row, the rating might be influenced by the games played against SGA himself, which is a limitation to consider.

- **Home/Away Indicator**: A binary variable indicating whether the game was played at home (1) or away (0), which accounts for the home-court advantage that can influence player performance.
- **Usage % Missing**: Usage % measures how often a player ends a possession or shoots the ball relative to the rest of the players on the court (Bill 2023). We calculated the missing usage % for each game, which is the difference between the total usage % of SGA's team, subset to only players who averaged 12 minutes a game (a quarter) or more, and the usage % of the active players in that game. Usage % is related to a player's scoring behavior, so it could be another endogenous variable, but calculated over a season, it should be mostly exogenous to the player's hotness in a given game. This variable accounts for the fact that in some games, SGA might have more or less opportunity to score based on the availability of his teammates.

These are indexed by game number instead of game date, which is usually around every 2-3 days, but can be more or less frequent depending on the schedule. This is done to get a clean time series for modeling, but it is important to note that the time gaps between games are not uniform, and could be a factor in the player's performance and hotness. For example, a longer gap between games could lead to rustiness, while a shorter gap could lead to fatigue. Gaps long enough to cause significant changes in the player's performance are few and far between, but they could still be an interesting area for future exploration.

Figure 1 shows the our observed variable, points scored, over the course of the season.

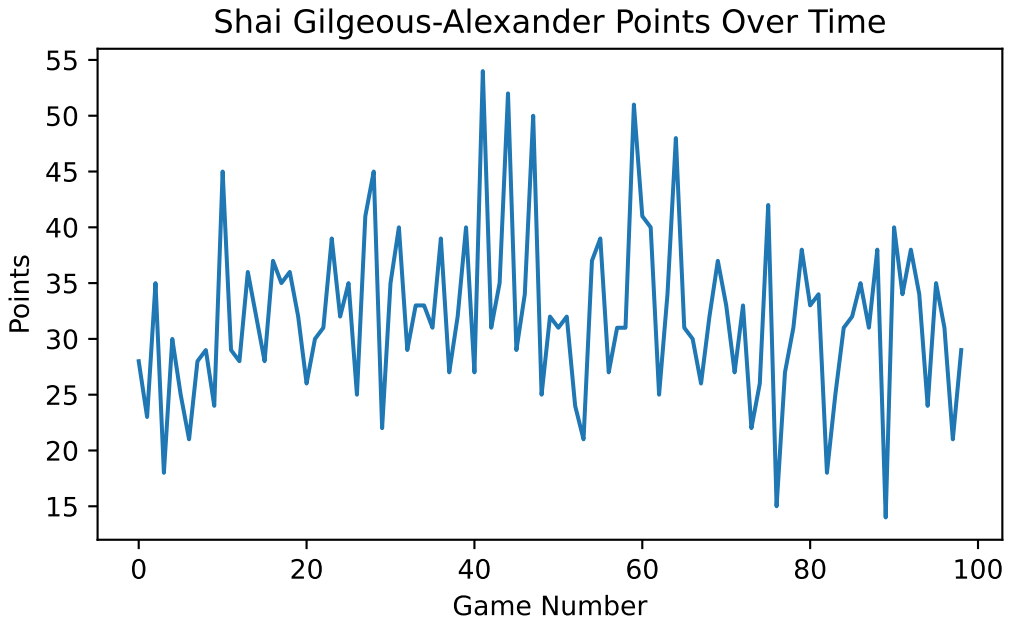

Figure 1: Points scored by SGA in the 2024-25 season, including regular season and playoffs.

99 data points corresponding to games played for SGA in the 2024-25 season, at an average of 32.04

points per game. Admittedly, this is less than the 100 data points asked for, but it is close enough to where our diagnostics and approximations should still be valid.

3 ARMA Modeling

Since Figure 1 above does not show any clear violations of stationarity, we can start by creating a benchmark ARMA model that would model the points scored as an ARMA process, without any covariates or latent states. This would give us a baseline to compare our POMP model to. The ARMA model is specified as:

$$\phi(B)((1-B)P_t - \mu) = \theta(B)\epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

where B is the backshift operator, $\phi(B)$ and $\theta(B)$ are polynomials of order p and q respectively, and μ is the mean of the process (Ionides 2026a).

An AIC table over different values of p and q is shown in Table 2 in the Supplementary Materials, which suggests that a ARMA(4,4) model is the best fit for the data. This is somewhat surprising given the small number of data points, but it could be due to the fact that the points scored in a game can be influenced by a variety of factors, creating complex relationships in the data.

The ARMA(4,4) model gives us a log-likelihood of -330.18, which we will use as a benchmark to compare our POMP model to. The parameters of the ARMA(4,4) model and the ACF plot for the residuals is shown in Figure 6 in the Supplementary Materials.

4 POMP Modeling

4.1 Model Specification

The latent state X_t represents the player's hotness in a given game, which is an unobserved variable that captures the player's underlying performance level that can fluctuate from game to game. We model this latent state as an AR(1) process, (Author 2025a):

$$X_t = \phi X_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

The process thus has the following transition density:

$$f_{X_t|X_{t-1}}(x_t|x_{t-1}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_t - \phi x_{t-1})^2}{2\sigma^2}\right)$$

The observations P_t represent the points scored in a game, which we model as Poisson, as the simplest way to model count data, but negative binomial will be another option considered later:

$$P_t|X_t \sim \text{Poisson}(\lambda_t)$$

The rate parameter λ_t is modeled as a log-linear function of the latent state X_t and the covariates, which allows us to capture the effect of the player's hotness and the other factors on the points scored:

$$\lambda_t = \exp(\mu + X_t + \beta_1 R_t + \beta_2 H_t + \beta_3 M_t)$$

where R_t is the centered rolling opponent's defensive rating, H_t is the home indicator variable, and M_t is the usage % missing in that game. Just like our predecessors in the baseball project (Author 2025a), we include the μ term to capture the baseline scoring level of the player, for which $\exp(\mu)$ can be interpreted as the expected points scored when the player is at neutral hotness ($X_t = 0$), against an average opponent ($R_t = 0$), playing without the home court boost ($H_t = 0$), and with nobody missing from the lineup ($M_t = 0$).

We would expect β_1 to be negative, as a higher defensive rating implies worse defense and thus more points scored, β_2 to be positive, as playing at home usually gives a boost to performance, and β_3 to be positive, as having more missing usage % means that the player has more opportunity to score. We can assume that the player starts the season at a neutral hotness level, so we can set $X_0 = 0$ for the initial state.

4.2 Local Search

We will start searching around the following initial parameter values:

- $\phi = 0.8$: This suggests a fairly persistent hotness state.
- $\sigma = 0.05$: This suggests that the hotness state does not fluctuate too wildly from game to game.
- $\mu = 3.4$: This is the baseline scoring level, which corresponds to about $e^{3.4} \approx 30$ points per game, around SGA's average points per game.
- $\beta_1 = -0.1$: This suggests that a one standard deviation increase in the opponent's defensive rating would lead to about a 10% ($e^{-0.1} \approx 0.9$) decrease in points scored. For Shai, this would mean that playing against a top defensive team would lead to about 6 points less scored.
- $\beta_2 = 0.05$: This suggests that playing at home would lead to about a 5% ($e^{0.05} \approx 1.05$) increase in points scored. Star players are largely unaffected by this home court boost, but it could still be a factor.
- $\beta_3 = 0.2$: This suggests that a 1% increase in missing usage would lead to about a 20% ($e^{0.2} \approx 1.22$) increase in points scored, which is quite significant, and suggests that the player's opportunity to score is a major factor in his performance.

Some simulations from this model with initial parameters are shown in Figure 2.

The log-likelihood of the model with the initial parameters is -348.83. If we search around these parameters using a local search (Ionides 2026b) we get a maximum log-likelihood of -335.77, which is an improvement over the intial parameters, but still worse than the ARMA model.

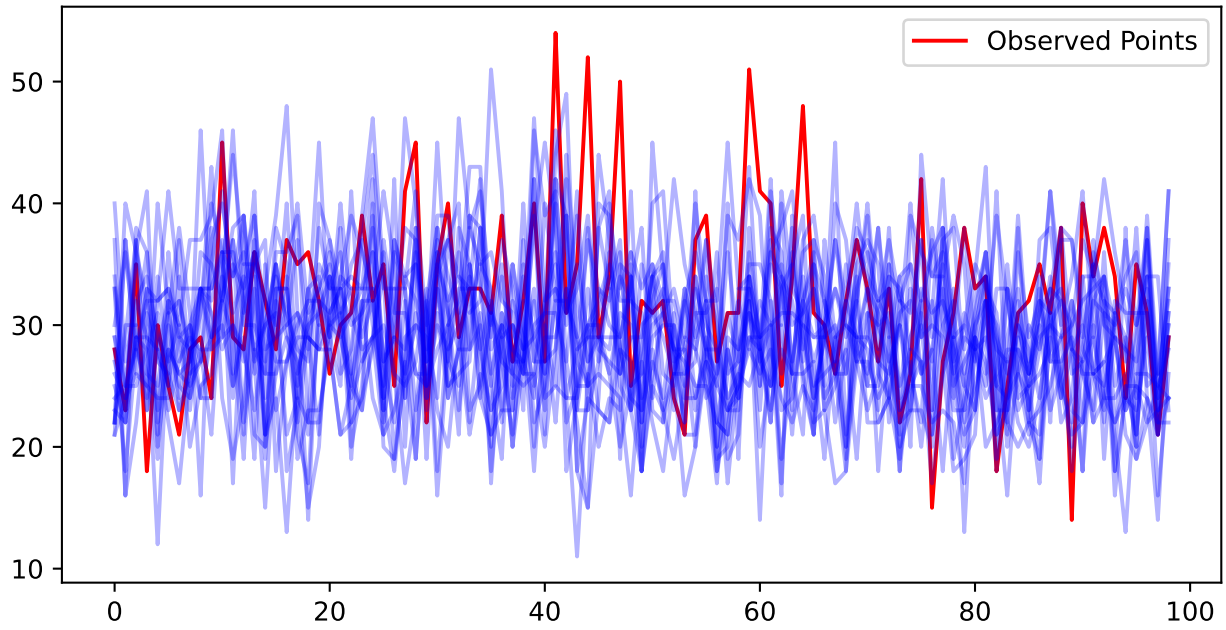

Figure 2: Simulated points scored from the POMP model with initial parameters, compared to observed points.

4.3 Global Search

Implementing a global search (Ionides 2026b) across different starting parameters is crucial to ensure we are not just stuck in a local maximum, which increases our chances of finding a true maximum. We search across the following ranges:

- ϕ: [0.01, 0.99]
- σ: [0.01, 0.2]
- β_1: [-0.5, 0]
- β_2: [0, 0.5]
- β_3: [0, 0.5]
- μ: [2.5, 4.0]

The best likelihood found across is -337.2, which is actually worse than the local search, which suggests that the local search might have found a better maximum, or that the global search did not explore the parameter space effectively. We can show the results of the global search in Figure 3:

This seems to indicate that ϕ values are generally negative for better likelihoods, which is counterintuitive, as it would suggest that a higher hotness in the previous game would lead to a lower hotness in the current game. This may be because the model is misspecified and the latent state is capturing some other factor that has a negative relationship with points scored. Either way, we can expand to searches executed on the Great Lakes cluster. We can also implement a Negative Binomial measurement model, which adds an additional dispersion parameter θ_{nb} to account for

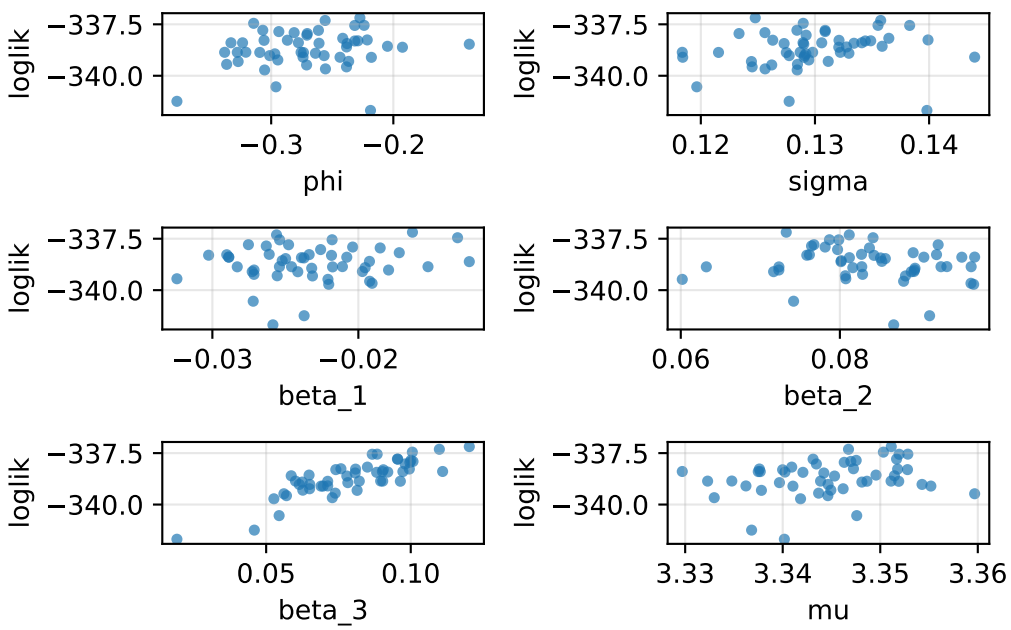

Figure 3: Results of global search across different starting parameters.

overdispersion in the points scored, which is common in basketball due to the varying point values of different shot types (free throws, 2-pointers, and 3-pointers). The formula for this model is:

$$P_t|X_t \sim \text{NegativeBinomial}(\lambda_t, \theta_{nb})$$

where λ_t is the same log-linear function of the latent state and covariates, and θ_{nb} is the dispersion parameter.

We generated multiple random starting points for our parameters: the autoregressive coefficient (ϕ), the latent state noise (σ), the covariate coefficients ($\beta_1, \beta_2, \beta_3$), the baseline scoring rate (μ) and the Negative Binomial dispersion parameter (θ_{nb}). For each starting point, we ran the `mif` algorithm to filter the particles and converge on the MLE. The parameter set that yielded the highest maximized log-likelihood after filtering was selected as the global MLE for each player.

4.3.1 Convergence Diagnostics: Global Search Trace Plots

To verify the convergence of our IF2 algorithm, we examined the parameter trace plots from our global search. The plots below display the trajectories of 15 randomly initialized starting points over the course of the filtering iterations for each player.

Despite the parameters being initialized across a wide and highly variable parameter space, the algorithm successfully forces the disparate starting points to converge toward consensus values.

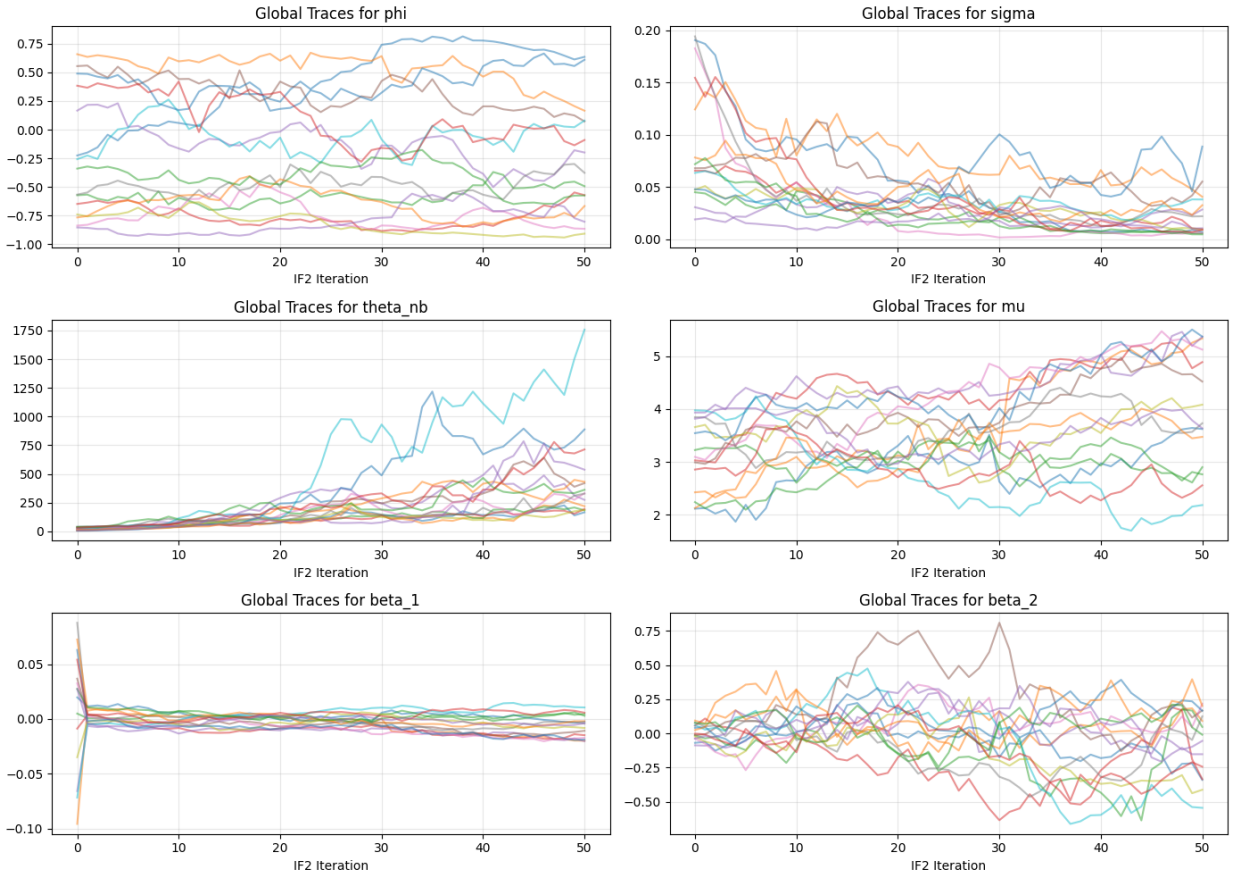

Figure 4: Global Search Trace Plots for Shai Gilgeous-Alexander

4.3.2 Measurement Model Comparison: Poisson vs. Negative Binomial

To determine the best measurement model for our POMP architecture, we evaluated both a standard Poisson distribution and a Negative Binomial distribution.

By extracting the MLE from our IF2 global search for both models, we can directly compare their goodness-of-fit:

Player	Poisson MLE Log-Likelihood	Negative Binomial MLE Log-Likelihood
Shai Gilgeous-Alexander	-355.24	-352.63
Steph Curry	-569.26	-518.03
Jayson Tatum	-780.32	-377.37

For all three players, the Negative Binomial model yields a higher maximum log-likelihood. The improvement is particularly drastic for Jayson Tatum and Steph Curry, high-volume 3-point shooters

whose game-to-game scoring variance is heavily influenced by perimeter shooting percentages. Another crucial point is that this log-likelihood is less than the one achieved by our local search with the Poisson model for Shai,

4.4 ϕ Profiling

To directly test for the presence of the "hot hand" phenomenon, we constructed a profile likelihood for the autoregressive parameter, ϕ. In our model formulation, ϕ dictates the persistence of the latent state X_t. A value of ϕ significantly greater than zero would indicate that a player's underlying "hot" or "cold" state carries over from game to game, providing mathematical evidence for the hot hand.

We evaluated the profile likelihood across a grid of fixed ϕ values. For each fixed value, we re-ran the iterated filtering process to maximize the likelihood over all other parameters.

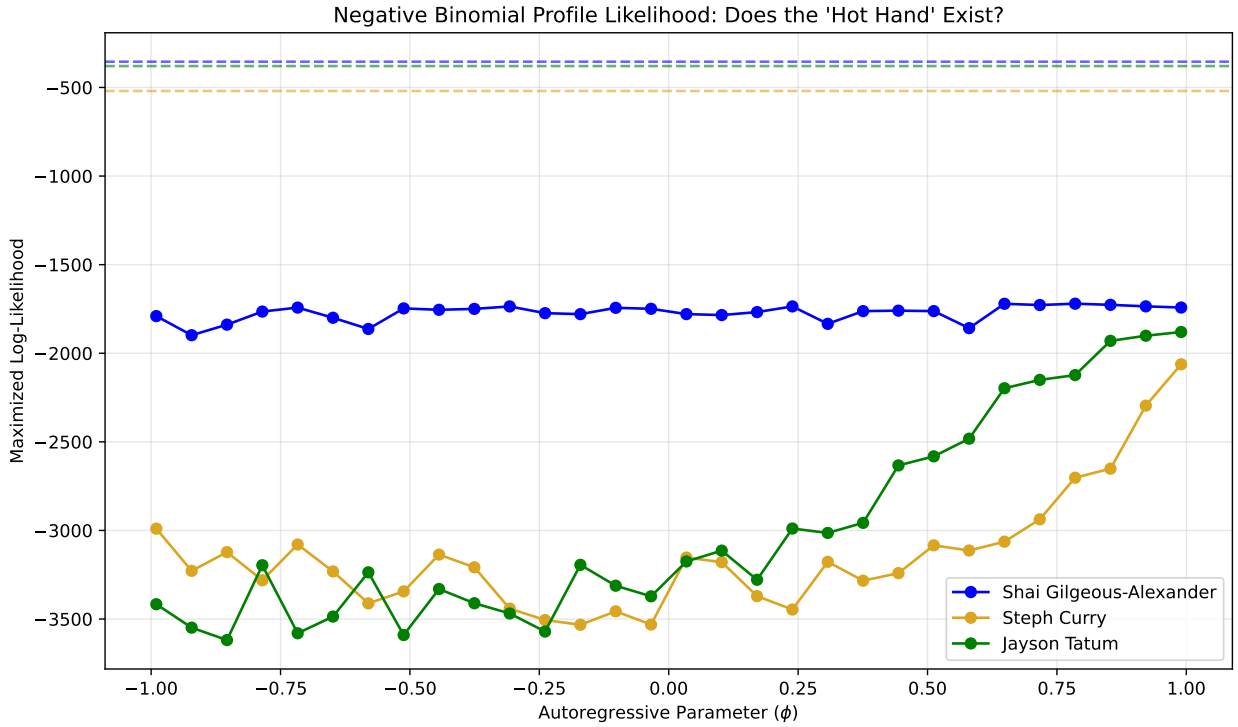

Figure 5: Profile Likelihood of the 'Hot Hand' parameter (phi) across three players using a Negative Binomial measurement model.

5 Conclusion

In this project, we aimed to rigorously quantify the "hot hand" phenomenon in the NBA by modeling player scoring as a Partially Observed Markov Process (POMP). By treating a player's underlying "form" or "momentum" as a hidden latent state driven by an autoregressive process, we attempted

to separate true psychological momentum from standard statistical variance and external covariates (such as opponent defensive ratings and home-court advantage).

We used a **Negative Binomial measurement model** to explicitly account for overdispersion in basketball scoring (which naturally arises due to the varying point values of free throws, 2-pointers, and 3-pointers), as it proved to perform better than a Poisson measurement model across all three players. Though it is important to note than an ARMA(4,4) still outperformed our POMP models in terms of log-likelihood, suggesting that there is room for improvement in our model specification.

5.1 Discussion of Findings

Based on our ϕ profiling using the Negative Binomial model, the data does not provide strong statistical evidence for a sustained "hot hand" effect that carries over from game to game.

For all three players (Shai Gilgeous-Alexander, Steph Curry, and Jayson Tatum), the profile likelihood surface is highly irregular, and the maximized log-likelihoods for the fixed ϕ values completely fail to cross the 95% confidence interval threshold derived from the global Maximum Likelihood Estimate. This suggests two main conclusions:

1. **Weak Latent Signal:** The variation in a player's game-to-game scoring is likely dominated by the measurement model, specifically the Negative Binomial overdispersion and the observable covariates, rather than a persistent, hidden autoregressive state. Even for historically explosive scorers like Curry, the model struggles to identify a consistent ϕ signal.
2. **Optimization Challenges:** The steep drop-off between the global MLE and the profile likelihood points indicates a highly jagged likelihood surface.

5.2 Future Work and Limitations

Outside of the limitation of time heterogeneity between games, and data restraints, the key limitation of this project is that there are many hidden factors that could be influencing the player's performance that may be currently absorbed into the latent state. For example, if a player gets hot, a defense will adjust their scheme to try to stop him, leading to less points scored in the next game. It would be nice to expand the model to account for some of these factors, potentially adding in game factors like shot distribution, but it would require more data. Future work could also explore modeling the hot hand phenomenon within games, as there are more points and a clearer time structure, but it would require a more complex model to account for all the different factors.

6 Scholarship and Acknowledgements

6.1 Contextual Integration

This project builds on work and review comments from the past midterm projects (Author 2025a), (Author 2025b). Our project was inspired by the future work suggestion from the baseball project, suggesting applying their work to individual momentum. Our modeling approach was similar to theirs, and we used their WAR suggestion to represent availability, and put it into related basketball context in *Missing Usage %*. Lastly, something echoed in both review comments was a desire to run on multiple teams, which in our case was on players. We implemented this to get a sense of whether our results are consistent across different players.

6.2 AI Usage

AI was primarily used for debugging code, especially helpful in the data extraction section, where we had to deal with a lot of API calls and data manipulation. The core modeling and analysis was done by us, with the help of class notes, past projects, and any citations mentioned.

7 Bibliography

Author, Anonymous. 2025a. "Examining Explanatory Role of Momentum in Baseball." https://ionides.github.io/531w25/final_project/project02/blinded.html.

———. 2025b. "NBA POMP-ELO: A Stochastic System Approach to Modeling Team Strength and Predicting Games." https://ionides.github.io/531w25/final_project/project09/blinded.html.

Bill, Coach. 2023. "All about the Basketball Usage Rate." https://blog.hoopsalytics.com/usage-rate-basketball-stat/.

Ionides, Edward. 2026a. "Notes for STATS 531, Modeling and Analysis of Time Series Data." https://ionides.github.io/531w26/.

———. 2026b. "PyPomp Tutorial Code." https://github.com/pypomp/tutorials/blob/main/sbied/chapter4/source.qmd.

Patel, Swar. 2018. "NBA API." https://github.com/swar/nba_api.

8 Supplementary Materials

AIC table for ARMA models with different values of p and q:

Table 2

	MA0	MA1	MA2	MA3	MA4	MA5
AR0	NaN	NaN	NaN	NaN	NaN	NaN
AR1	NaN	NaN	NaN	NaN	NaN	NaN
AR2	NaN	NaN	NaN	NaN	NaN	NaN
AR3	NaN	NaN	NaN	NaN	NaN	NaN
AR4	NaN	NaN	NaN	NaN	NaN	NaN
AR5	NaN	NaN	NaN	NaN	NaN	NaN

Fitting this model with 4 AR and 4 MA terms gives us the following parameters:

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                    PTS   No. Observations:                   99
Model:                 ARIMA(4, 0, 4)   Log Likelihood                -330.184
Date:                Tue, 21 Apr 2026   AIC                            680.368
Time:                        22:37:28   BIC                            706.319
Sample:                             0   HQIC                           690.868
                                  - 99
```

```
Covariance Type:                 opg
================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const         32.0008      0.886     36.098      0.000      30.263      33.738
ar.L1         -0.0860      0.138     -0.622      0.534      -0.357       0.185
ar.L2          0.3851      0.109      3.534      0.000       0.172       0.599
ar.L3          0.2568      0.114      2.262      0.024       0.034       0.479
ar.L4         -0.7502      0.132     -5.666      0.000      -1.010      -0.491
ma.L1         -0.0273      0.214     -0.127      0.899      -0.448       0.393
ma.L2         -0.4363      0.128     -3.422      0.001      -0.686      -0.186
ma.L3         -0.0220      0.194     -0.113      0.910      -0.403       0.359
ma.L4          0.9605      0.222      4.319      0.000       0.525       1.396
sigma2        43.5564      8.749      4.978      0.000      26.408      60.705
================================================================================
Ljung-Box (L1) (Q):                   0.14   Jarque-Bera (JB):                 1.60
Prob(Q):                              0.70   Prob(JB):                         0.45
Heteroskedasticity (H):               1.42   Skew:                             0.31
Prob(H) (two-sided):                  0.32   Kurtosis:                         2.94
================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

ACF plot for residuals of ARMA(4,4) model:

The following code was used on the Great Lakes cluster to perform the global Maximum Likelihood Estimation and the profile likelihood generation:

```python
import pandas as pd
import numpy as np
import jax
import jax.numpy as jnp
import jax.scipy.special as jspecial
import pypomp as pp
from pypomp import RWSigma
import matplotlib.pyplot as plt
import gc
import os

def rproc(X_, theta_, key, covars, t, dt):
    X_new = theta_["phi"] * X_["X"] + theta_["sigma"] * jax.random.normal(key)
    return {"X": X_new}
```

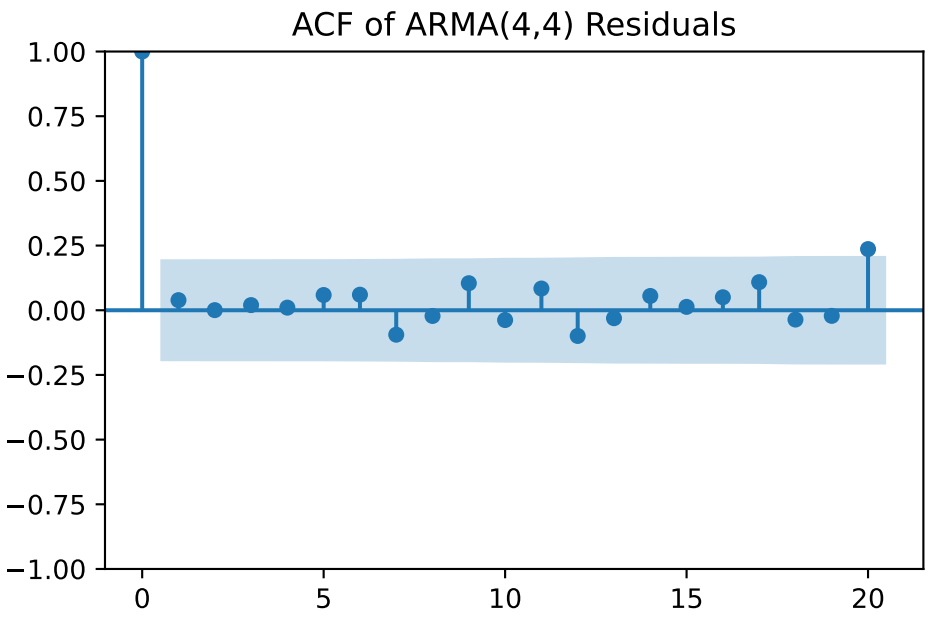

Figure 6: ACF plot for residuals of ARMA(4,4) model.

```
def dmeas(Y_, X_, theta_, covars, t):
    lam = jnp.exp(theta_["mu"] + X_["X"] +
                  theta_["beta_1"] * covars["Rolling_DEF_RATING"] +
                  theta_["beta_2"] * covars["Usage_Missing"] +
                  theta_["beta_3"] * covars["home"])

    n = theta_["theta_nb"]
    p = n / (n + lam)

    return jax.scipy.stats.nbinom.logpmf(Y_["PTS"], n, p)

def rmeas(X_, theta_, key, covars, t):
    lam = jnp.exp(theta_["mu"] + X_["X"] +
                  theta_["beta_1"] * covars["Rolling_DEF_RATING"] +
                  theta_["beta_2"] * covars["Usage_Missing"] +
                  theta_["beta_3"] * covars["home"])

    n = theta_["theta_nb"]
    key1, key2 = jax.random.split(key)
    gamma_sample = jax.random.gamma(key1, n)
    rate = gamma_sample * (lam / n)
```

```
    return jnp.array([jax.random.poisson(key2, rate)])

rinit = lambda theta_, key, covars, t0: {"X": 0.0}

def to_est(theta):

    phi_mapped = (theta["phi"] + 1.0) / 2.0
    return {"phi": jspecial.logit(phi_mapped), "sigma": jnp.log(theta["sigma"]),
            "theta_nb": jnp.log(theta["theta_nb"]),
            "beta_1": theta["beta_1"], "beta_2": theta["beta_2"], "beta_3":
             ↪  theta["beta_3"], "mu": theta["mu"]}

def from_est(theta_est):

    phi_unmapped = 2.0 * jspecial.expit(theta_est["phi"]) - 1.0
    return {"phi": phi_unmapped, "sigma": jnp.exp(theta_est["sigma"]),
            "theta_nb": jnp.exp(theta_est["theta_nb"]),
            "beta_1": theta_est["beta_1"], "beta_2": theta_est["beta_2"],
             ↪  "beta_3": theta_est["beta_3"], "mu": theta_est["mu"]}

par_trans = pp.ParTrans(to_est=to_est, from_est=from_est)

players = [
    "Shai_Gilgeous-Alexander_data.csv",
    "Steph_Curry_data.csv",
    "Jayson_Tatum_data.csv"
]

for player_file in players:
    player_name = player_file.split('_data')[0]
    print(f"\nStarting Negative Binomial Analysis for {player_name}")

    try:
        df = pd.read_csv(player_file)
    except FileNotFoundError:
        print(f"Could not find {player_file}, skipping...")
        continue

    ys = df[['PTS']]
    covars = df[['Rolling_DEF_RATING', 'Usage_Missing', 'home']]

    # Global Search Setup
    n_starts = 15
    np.random.seed(42)
```

```
    global_starts = [{
        "phi": np.random.uniform(-0.9, 0.9), "sigma": np.random.uniform(0.01,
        ↪  0.2),
        "theta_nb": np.random.uniform(5.0, 50.0),
        "beta_1": np.random.uniform(-0.1, 0.1), "beta_2": np.random.uniform(-0.1,
        ↪  0.1),
        "beta_3": np.random.uniform(-0.1, 0.1), "mu": np.random.uniform(2.0, 4.0)
    } for _ in range(n_starts)]

    global_pomp = pp.Pomp(ys=ys, t0=0, rinit=rinit, rproc=rproc, dmeas=dmeas,
↪   rmeas=rmeas,
                          theta=global_starts, covars=covars, statenames=["X"],
↪   nstep=1, par_trans=par_trans)

    rw_sd = RWSigma(sigmas={"phi": 0.02, "sigma": 0.02, "theta_nb": 0.02,
                            "beta_1": 0.01, "beta_2": 0.01, "beta_3": 0.01, "mu":
                             ↪  0.02})

    print("Running Global Search")
    global_pomp.mif(J=1500, M=50, rw_sd=rw_sd, a=0.95,
↪   key=jax.random.PRNGKey(789))
    global_pomp.pfilter(J=2500, reps=5, key=jax.random.PRNGKey(1011), CLL=True)

    pf_result = global_pomp.results_history[-1]
    df_cll = pf_result.CLL().reset_index()

    val_col = df_cll.select_dtypes(include=['float']).columns[0]
    cat_cols = df_cll.select_dtypes(exclude=['float']).columns

    theta_col = [c for c in cat_cols if 'theta' in c.lower() or
↪   df_cll[c].nunique() == n_starts][0]
    rep_col = [c for c in cat_cols if 'rep' in c.lower() or df_cll[c].nunique()
↪   == 5][0]

    summed_over_time = df_cll.groupby([theta_col,
↪   rep_col])[val_col].sum().reset_index()

    def lse(x):
        return float(jspecial.logsumexp(x.values) - jnp.log(5))

    mean_ll = summed_over_time.groupby(theta_col)[val_col].apply(lse)

    best_idx = int(mean_ll.idxmax())
    best_loglik = float(mean_ll.max())
```

```
    mif_results = global_pomp.results_history[-2].traces_da
    actual_params = ["phi", "sigma", "theta_nb", "beta_1", "beta_2", "beta_3",
↪   "mu"]
    best_params = {param: float(mif_results.isel(theta_idx=best_idx,
↪   iteration=-1).sel(variable=param).values)
                   for param in actual_params}

    print(f"MLE LogLik: {best_loglik:.2f}")

    print("Generating Global Trace Plots")
    params_to_plot = ["phi", "sigma", "theta_nb", "mu", "beta_1", "beta_2"]
    fig, axes = plt.subplots(3, 2, figsize=(14, 10))
    for idx, param in enumerate(params_to_plot):
        ax = axes[idx // 2, idx % 2]
        for start_idx in range(mif_results.sizes['theta_idx']):
            trace_vals =
↪   mif_results.isel(theta_idx=start_idx).sel(variable=param).values.flatten()
            ax.plot(trace_vals, alpha=0.5)
        ax.set_title(f"Global Traces for {param}")
        ax.set_xlabel("IF2 Iteration")
        ax.grid(alpha=0.3)

    plt.tight_layout()
    trace_filename = f"global_traces_{player_name}.png"
    plt.savefig(trace_filename)
    print(f"Saved {trace_filename}")
    plt.close()

    print("Running Profile Likelihood for phi (-0.99 to 0.99)")
    phi_vals = np.linspace(-0.99, 0.99, 30)
    profile_results = []

    rw_sd_prof = RWSigma(sigmas={"phi": 0.0, "sigma": 0.02, "theta_nb": 0.02,
                                 "beta_1": 0.01, "beta_2": 0.01, "beta_3": 0.01,
                                  ↪  "mu": 0.02}, init_names=["sigma"])

    for p_val in phi_vals:
        prof_starts = [{**best_params, "phi": p_val} for _ in range(5)]
        prof_pomp = pp.Pomp(ys=ys, t0=0, rinit=rinit, rproc=rproc, dmeas=dmeas,
↪   rmeas=rmeas,
                            theta=prof_starts, covars=covars, statenames=["X"],
↪   nstep=1, par_trans=par_trans)
```

```
        prof_pomp.mif(J=1500, M=50, rw_sd=rw_sd_prof, a=0.95,
↪   key=jax.random.PRNGKey(333))
        prof_pomp.pfilter(J=2500, key=jax.random.PRNGKey(444), reps=5, CLL=True)

        prof_df = prof_pomp.results_history[-1].CLL().reset_index()
        prof_val_col = prof_df.select_dtypes(include=['float']).columns[0]
        prof_cat_cols = prof_df.select_dtypes(exclude=['float']).columns

        prof_rep_col = [c for c in prof_cat_cols if 'rep' in c.lower() or
↪   prof_df[c].nunique() == 5][0]

        prof_ll_per_rep =
↪   prof_df.groupby(prof_rep_col)[prof_val_col].sum().values
        prof_mean_ll = float(jspecial.logsumexp(prof_ll_per_rep) - jnp.log(5))

        profile_results.append({"phi": p_val, "loglik": prof_mean_ll, "cutoff":
↪   best_loglik - 1.92})

        del prof_pomp
        gc.collect()

    pd.DataFrame(profile_results).to_csv(f"{player_name}_nb_profile.csv",
↪   index=False)
    print(f"Saved {player_name}_nb_profile.csv")

print("All processing complete!")
```

Trace Plots from the Global Search for each player:

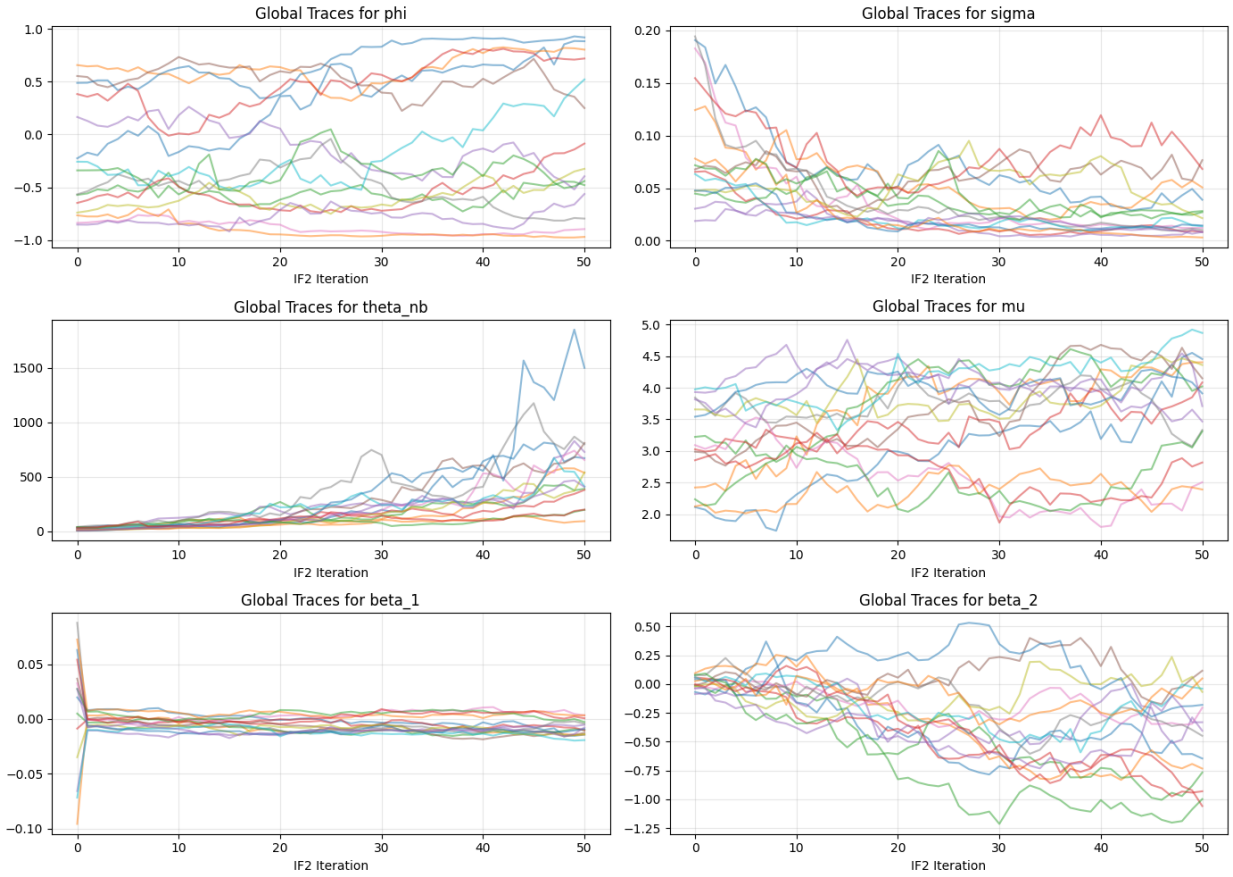

Figure 7: Global Search Trace Plots for Steph Curry

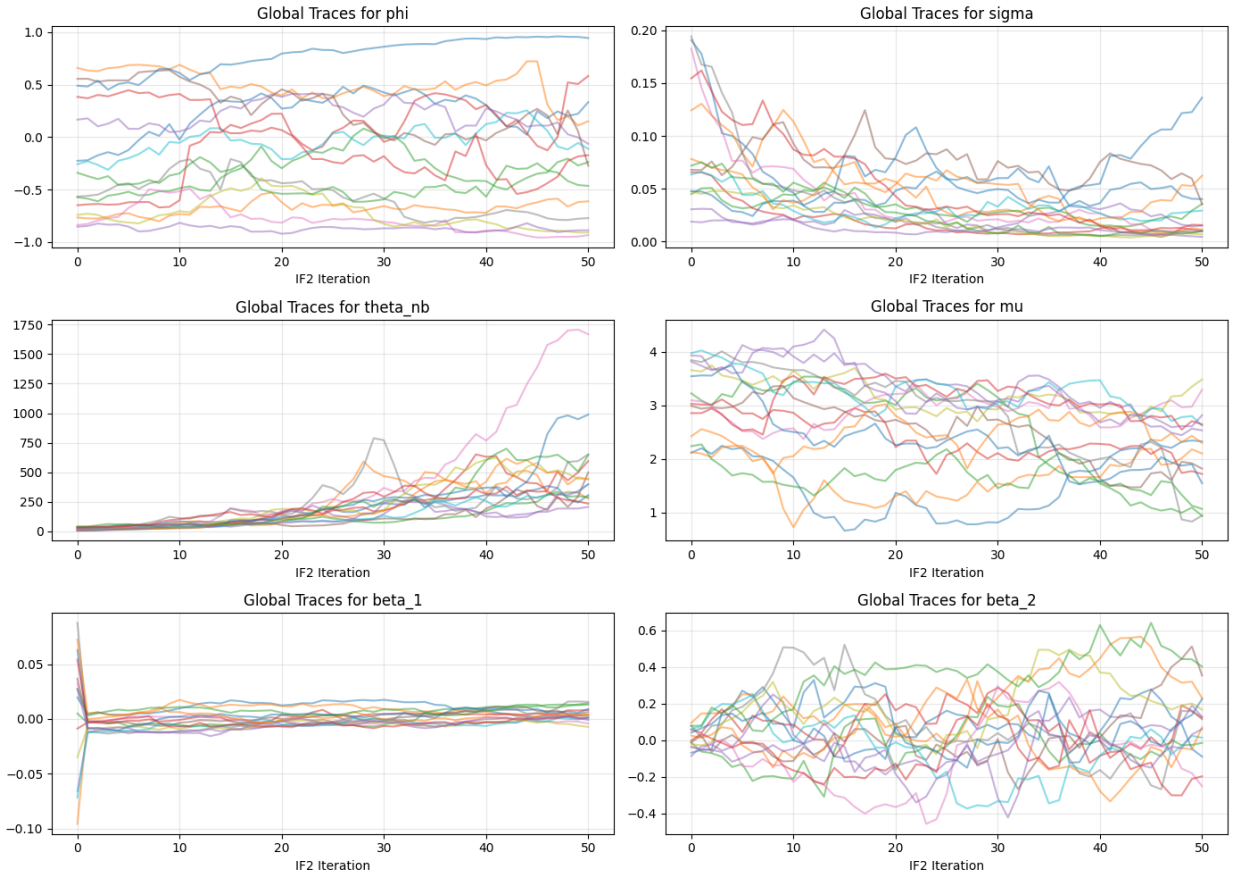

Figure 8: Global Search Trace Plots for Jayson Tatum