Effect of Back-to-Back Games on NBA Player Performance

Darko Trifunovski

May 25, 2022

1 Introduction

One of the most popular professional sport leagues in North America is the National Basketball Association. In a regular NBA season[1], 30 teams divided into 2 conferences, each of which is further subdivided into 3 divisions, play 82 games over a roughly 6 month time span.

The large number of regular season games relative to the length of the season requires the league to schedule a number of so-called back-to-back games—games played by a team that has also played on the previous day. These games are somewhat controversial: many players say that such schedules are overly fatiguing and needlessly increase their risk of injury, while others point out that the travel schedule used to be even more grueling in the past.

The goal of our analysis will be to evaluate the impact of playing in a back-to-back game on player performance.

1.1 Player Performance

To be able to answer this question, we first need a way to quantify the performance of a player in a given game. We will use John Hollinger's game score (GS) formula, a commonly used player evaluation metric in basketball statistics. It is given by

$$\begin{aligned}\mathrm{GS} = {} & \mathrm{PTS} + 0.4 \cdot \mathrm{FG} - 0.7 \cdot \mathrm{FGA} - 0.4 \cdot (\mathrm{FTA} - \mathrm{FT}) \\ & + 0.7 \cdot \mathrm{ORB} + 0.3 \cdot \mathrm{DRB} + \mathrm{STL} + 0.7 \cdot \mathrm{AST} + 0.7 \cdot \mathrm{BLK} - 0.4 \cdot \mathrm{PF} - \mathrm{TOV},\end{aligned}$$

where

Abbreviation	Definition	Abbreviation	Definition
PTS	Points scored	DRB	Defensive rebounds
FG	Field goals made	STL	Steals
FGA	Field goals attempted	AST	Assists
FTA	Free throws attempted	BLK	Blocked shots
FT	Free throws made	PF	Personal fouls
ORB	Offensive rebounds	TOV	Turnovers

The score is scaled so that it is interpretable on the same scale as the number of points scored, i.e. so that a score of around 10 is average, while a score of over 40 represents an exceptional game performance.

In addition to being a widely used formula, GS has the advantage of using only statistics that are available in official modern NBA box score data.

[1]Of recent seasons, 2019–20 and 2020–21 are notable exceptions due to the disruption caused by the COVID-19 pandemic.

1.2 Dataset

As our dataset we use the official box scores scraped from `nba.com/stats` by Nathan Lauga and uploaded as CSV files on Kaggle (`www.kaggle.com/datasets/nathanlauga/nba-games`).

The dataset contains all NBA games from the 2004 season to December 2020, but for the purposes of this study we restrict ourselves to the 2018–2019 regular season only.

```
season_start = datetime.strptime("2018-10-16", "%Y-%m-%d")
season_end = datetime.strptime("2019-04-10", "%Y-%m-%d")

games = nbadata.games
season_games = games[(nbadata.games.GAME_DATE_EST >= season_start)
                     & (nbadata.games.GAME_DATE_EST <= season_end)].copy()
```

Next, we will discard all rows representing players that did not play in a particular game, and use the box score data to compute the game score for the rest.

```
player_details = nbadata.game_details.copy()
player_details = player_details[player_details.MIN > 0]
gs_fn = lambda x: (x.PTS + 0.4*x.FGM - 0.7*x.FGA
                    - 0.4*(x.FTA - x.FTM)
                    + 0.7*x.OREB + 0.3*x.DREB
                    + x.STL + 0.7 * x.AST + 0.7 * x.BLK
                    - 0.4 * x.PF - x.TO)
player_details["GS"] = gs_fn(player_details)
```

Then, we compute whether each team is playing a back-to-back game, and merge the game and player data.

```
def had_prev_home_game(games_df, date, team_id):
    previous_day = date - timedelta(days=1)
    game = games_df[(games_df.GAME_DATE_EST == previous_day) &
         (games_df.HOME_TEAM_ID == team_id)]
    return game.shape[0]

def had_prev_away_game(games_df, date, team_id):
    previous_day = date - timedelta(days=1)
    game = games_df[(games_df.GAME_DATE_EST == previous_day) &
         (games_df.VISITOR_TEAM_ID == team_id)]
    return game.shape[0]

season_games["VISITOR_B2B_HOME"] = season_games.apply(
    lambda x: had_prev_home_game(games, x.GAME_DATE_EST, x.VISITOR_TEAM_ID),
    axis=1
)
season_games["VISITOR_B2B_AWAY"] = season_games.apply(lambda x:
    had_prev_away_game(games, x.GAME_DATE_EST, x.VISITOR_TEAM_ID),
    axis=1
)
season_games["HOME_B2B_AWAY"] = season_games.apply(lambda x:
    had_prev_away_game(games, x.GAME_DATE_EST, x.HOME_TEAM_ID),
    axis=1
)
```

```python
df = pd.merge(season_games, player_details, on="GAME_ID", how="inner").copy()
df["AWAY"] = pd.Categorical(df.TEAM_ID == df.VISITOR_TEAM_ID).codes
df["HA_B2B"] = (df.AWAY == False) & df.HOME_B2B_AWAY
df["AH_B2B"] = (df.AWAY == True) & df.VISITOR_B2B_HOME
df["AA_B2B"] = (df.AWAY == True) & df.VISITOR_B2B_AWAY
df["B2B"] = pd.Categorical(df.HA_B2B | df.AH_B2B | df.AA_B2B).codes

df["PLAYER_ID"] = pd.Categorical(df.PLAYER_ID).codes
num_players = len(df.PLAYER_ID.value_counts())
```

Finally, we plot some of the data.

```python
total_gs = df.groupby(["GAME_DATE_EST", "GAME_ID", "AWAY", "B2B"])["GS"].sum()

regular_game_dates = []
regular_game_gss = []
b2b_game_dates = []
b2b_game_gss = []
for idx, gs in total_gs.iteritems():
    date, _, _, b2b = idx
    if b2b:
        b2b_game_dates.append(date)
        b2b_game_gss.append(gs)
    else:
        regular_game_dates.append(date)
        regular_game_gss.append(gs)

fig, ax = plt.subplots(figsize=(13, 7))
plt.scatter(b2b_game_dates, b2b_game_gss, color="black", alpha=0.7, label="Back-to-Back")
plt.scatter(regular_game_dates, regular_game_gss, color="C1", alpha=0.3, label="Regular")
plt.xlabel("Date")
plt.ylabel("Team sum of player game scores")
plt.legend()
```

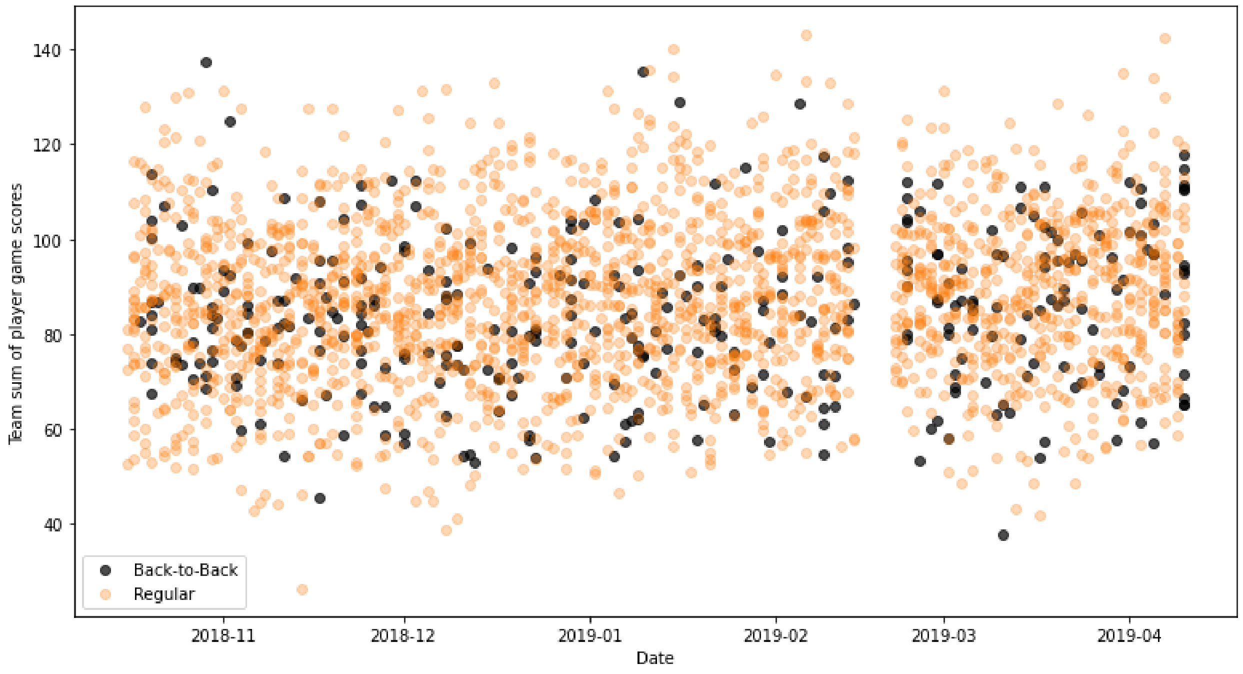

Figure 1: Each circle represents the sum of game scores of all players on a team in a game in the 2018 season.

```
az.plot_kde(df.GS)
plt.ylabel("Density")
plt.xlabel("Game Score")
```

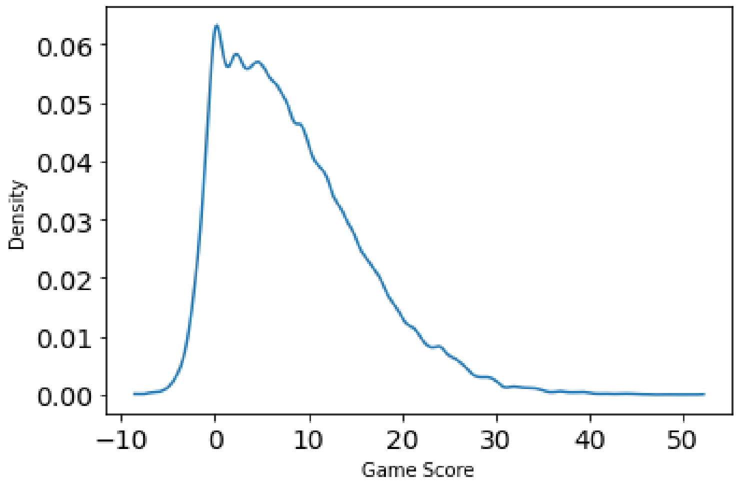

Figure 2: Distribution of game scores.

2 Models

Each observation in our dataset is the performance of a player in a game of the 2018–2019 NBA regular season. We model them as samples from a normal distribution, with mean depending on a mean parameter unique to each player, a parameter corresponding to whether the player is playing a home or away game (in the second and third models) and a parameter corresponding to whether the player is playing in a back-to-back game (in the third model only). Since back-to-back games are more likely to be away than home, we have to add the home/away parameter to the third model as a control.

2.1 Model with Player-only Effects

In the first model, we attempt to analyze the performance of each player, while ignoring whether they are playing an away or back-to-back game. We use a hierarchical model, where we model the mean performance of each player i as $\mu_i \sim \text{Normal}(\overline{\mu}, \sigma_\mu)$, where $\overline{\mu}$ is a hyperparameter with an informative prior of $\text{Normal}(10, 1)$. Recall that the game score is on a similar scale as points scored, and thus we know that it should be around 10 on average.

For the scale parameters σ_h and σ_μ we use a weakly-informative prior for scale-parameters: $\text{Exponential}(0.5)$.

$$\begin{aligned}
h_i &\sim \text{Normal}(\mu_{\text{PLAYER}[i]}, \sigma_h) \\
\sigma_h &\sim \text{Exponential}(0.5) \\
\mu_j &\sim \text{Normal}(\overline{\mu}, \sigma_\mu) \\
\overline{\mu} &\sim \text{Normal}(10, 1) \\
\sigma_\mu &\sim \text{Exponential}(0.5)
\end{aligned}$$

We express this model in PyMC code as follow:

```python
with pm.Model() as model1:
    mu_bar = pm.Normal("mu_bar", 10, 1)
    sigma_mu = pm.Exponential("sigma_mu", 0.5)
    mu = pm.Normal("mu", mu_bar, sigma_mu, shape=num_players)
    sigma_h = pm.Exponential("sigma_h", 0.5)
    h = pm.Normal("h", mu[df.PLAYER_ID], sigma_h, observed=df.GS)

    trace1 = pm.sample(draws=1_000, chains=4, cores=4, random_seed=42,
                       return_inferencedata=True)
```

2.2 Model with Home/Away Covariate

In the second model, we proceed identically to the first, but add to m_i a parameter α whose value depends on whether we are looking at a player's performance in a home or an away game. We use $\text{Normal}(0, 0.5)$ priors for our parameters, which is a weakly-informative prior relative to the scale of the response variable.

$$
\begin{aligned}
h_i &\sim \text{Normal}(m_i, \sigma_h)\\
\sigma_h &\sim \text{Exponential}(0.5)\\
m_i &= \mu_{\text{PLAYER}[i]} + \alpha_{\text{AWAY}[i]}\\
\mu_j &\sim \text{Normal}(\overline{\mu}, \sigma_\mu)\\
\overline{\mu} &\sim \text{Normal}(10, 1)\\
\sigma_\mu &\sim \text{Exponential}(0.5)\\
\alpha &\sim \text{Normal}(0, 0.5)
\end{aligned}
$$

```
with pm.Model() as model2:
    mu_bar = pm.Normal("mu_bar", 10, 1)
    sigma_mu = pm.Exponential("sigma_mu", 0.5)
    mu = pm.Normal("mu", 0, sigma_mu, shape=num_players) + mu_bar
    alpha = pm.Normal("alpha", 0, 0.5, shape=2)
    sigma_h = pm.Exponential("sigma_h", 0.5)
    h = pm.Normal("h", mu[df.PLAYER_ID] + alpha[df.AWAY], sigma_h, observed=df.GS)

    alpha_diff = pm.Deterministic("alpha_diff", alpha[0] - alpha[1])
    trace2 = pm.sample(draws=1_000, chains=4, cores=4, random_seed=42,
                       return_inferencedata=True)
```

2.3 Model with Back-to-Back Game Covariate

We proceed similarly to the previous model, but now add an additional pair of parameters β corresponding to whether the player is playing a back-to-back game or not. We once again use the weakly-informative prior Normal$(0, 0.5)$.

$$
\begin{aligned}
h_i &\sim \text{Normal}(m_i, \sigma)\\
m_i &= \mu_{\text{PLAYER}[i]} + \alpha_{\text{AWAY}[i]} + \beta_{\text{B2B}[i]}\\
\mu_j &\sim \text{Normal}(\overline{\mu}, \sigma_\mu)\\
\overline{\mu} &\sim \text{Normal}(10, 1)\\
\sigma_\mu &\sim \text{Exponential}(0.5)\\
\alpha &\sim \text{Normal}(0, 0.5)\\
\beta &\sim \text{Normal}(0, 0.5)
\end{aligned}
$$

Note that we have added in the PyMC code below the deterministic `beta_diff` parameter which represents our quantity of interest—the difference between game scores in regular games and back-to-back games.

```
with pm.Model() as model3:
    mu_bar = pm.Normal("mu_bar", 10, 1)
    sigma_mu = pm.Exponential("sigma_mu", 0.5)
    mu = pm.Normal("mu", 0, sigma_mu, shape=num_players) + mu_bar
    alpha = pm.Normal("alpha", 0, 0.5, shape=2)
    beta = pm.Normal("beta", 0, 0.5, shape=2)
    sigma_h = pm.Exponential("sigma_h", 0.5)
    h = pm.Normal("h", mu[df.PLAYER_ID] + alpha[df.AWAY] + beta[df.B2B], sigma_h,
                  observed=df.GS)

    beta_diff = pm.Deterministic("beta_diff", beta[0] - beta[1])
    trace3 = pm.sample(draws=1_000, chains=4, cores=4, random_seed=42,
                       return_inferencedata=True)
```

3 Model Comparison

We compare our models using PSIS-LOO.

```
round(az.compare({"player_only": trace1, "away": trace2, "b2b": trace3}), 3)
```

	rank	loo	p_loo	d_loo	weight	se	dse	warning	loo_scale
away	0	−83060.473	440.267	0.000	0.966	135.319	0.000	False	log
b2b	1	−83061.428	440.918	0.954	0.000	135.300	0.756	True	log
player_only	2	−83074.116	439.161	13.643	0.034	135.395	5.406	True	log

PSIS-LOO detected 6 observations with $\hat{k}$ scores between 0.5 and 0.7 in at least one of the models, and 2 observations between 0.7 and 1. Due to the large size of our dataset (26,095 observations), this is not a cause for concern.

Note that PSIS-LOO prefers the model using only the home/away covariate, as it performs almost identically to the back-to-back one, but using fewer parameters. This suggests that the effect of playing back-to-back games on game score is extremely weak, but we should be careful with drawing any conclusions from this as our dataset is very imbalanced in favor of non-back-to-back games, and so it should be expected that the contribution of this feature to improving predictions over the whole dataset will be extremely minor.

4 Conclusion

We are finally ready to plot the difference between game scores in regular games and back-to-back games using the data generated in the third model.

```
beta_diff_samples = az.extract_dataset(trace3).beta_diff.values
az.plot_kde(beta_diff_samples)
plt.xlabel("beta_diff")
plt.ylabel("Density")
```

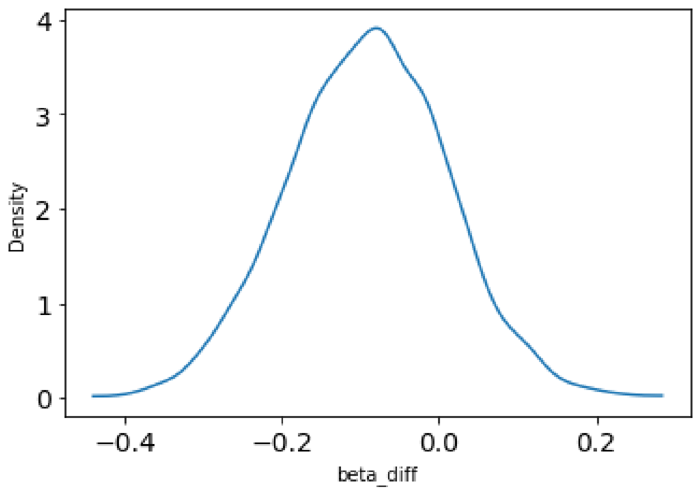

Figure 3: Distribution of difference between game scores in regular games and back-to-back games.

```
print("Probability that beta_diff is >0:",
    round(sum(beta_diff_samples > 0) / len(beta_diff_samples), 3))
print("89% HDI for beta_diff is:",
    np.round(az.hdi(beta_diff_samples, 0.98), 2))
```

```
Probability that beta_diff is >0: 0.179
89% HDI for beta_diff is: [-0.32  0.14]
```

Thus we may conclude that there is no evidence for a negative effect on player performance in playing back-to-back games.

For comparison, we plot the effect of playing a home game versus an away one.

```
alpha_diff_samples = az.extract_dataset(trace2).alpha_diff.values
az.plot_kde(alpha_diff_samples)
plt.xlabel("alpha_diff")
plt.ylabel("Density")
```

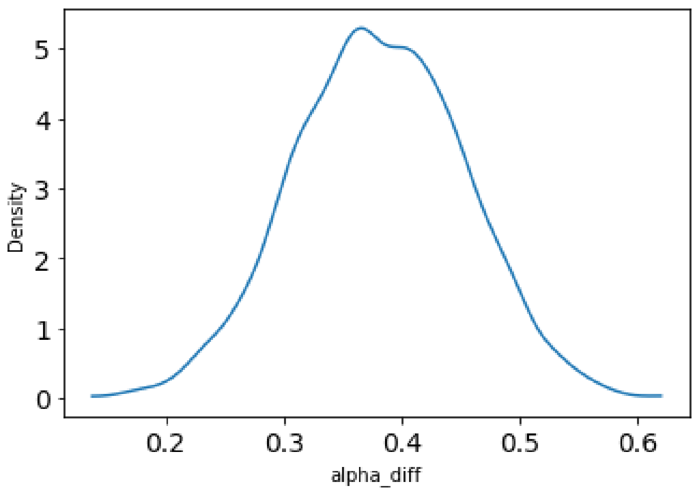

Figure 4: Distribution of difference between game scores in home games versus away ones.

```
print("Probability that alpha_diff is >0:",
    round(sum(alpha_diff_samples > 0) / len(alpha_diff_samples), 3))
print("89% HDI for alpha_diff is:",
    np.round(az.hdi(alpha_diff_samples, 0.89), 3))
```

```
Probability that alpha_diff is >0: 1.0
89% HDI for alpha_diff is: [0.269 0.497]
```

Note that there is robust evidence that playing an away game will hurt player performance, with the bulk of the distribution of the parameter being located away from zero.

4.1 Further Research

The main limitation of our analysis is assuming an additive model for the effect of playing a back-to-back game, i.e. that a player that averages 20 points per game should experience the same penalty in points as one that averages 5. One possible avenue of further research is to run more analysis of this kind but by instead assuming that the penalty is a percentage of the mean.

A Model Diagnostics

Due to the large number of parameters in each model (almost 600), we cannot visually inspect the trace plot for all parameters. We will instead check for problematic parameters which we define as ones with effective sample size less than 25% of the number of draws (i.e. less than 1000), or with $\widehat{R} > 1.01$.

```
def problematic(summary):
    problem = ((summary["r_hat"] > 1.01) | (summary["ess_bulk"] < 1000)
               | (summary["ess_tail"] < 1000))
    return problem[problem].index
```

A.1 Model with player-only effect

```
summary1 = az.summary(trace1)
problematic(summary1)
```

```
[]
```

There are no problematic parameters for the first model.

A.2 Model with home/away covariate only

```
summary2 = az.summary(trace2)
summary2.loc[problematic(summary2)]
```

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
mu_bar	7.364	0.379	6.652	8.058	0.013	0.009	819.0	1772.0	1.01

```
az.plot_trace(trace2, var_names=["mu_bar"], figsize=(13, 5), compact=False)
```

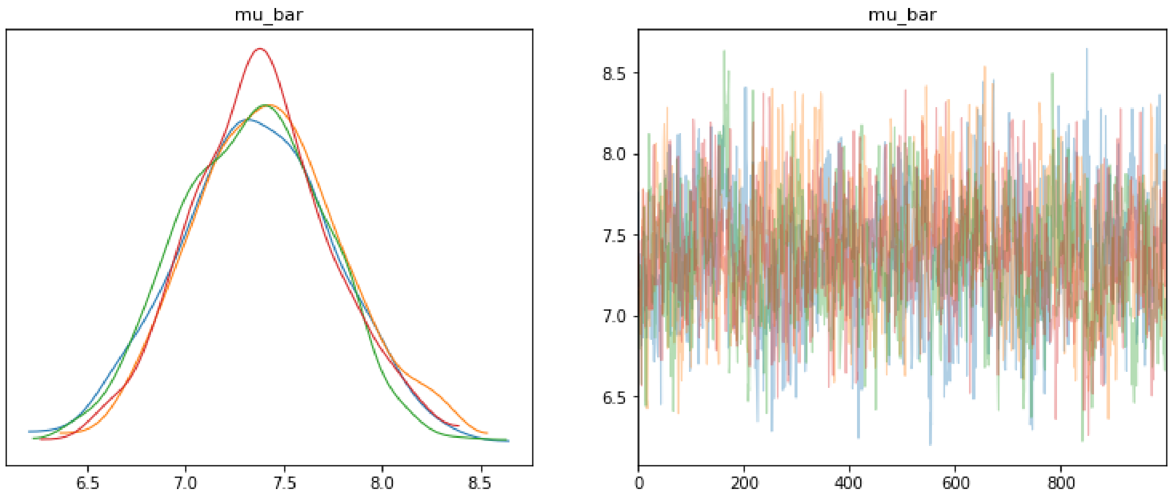

Figure 5: Left, a plot of the posterior probability distribution from each chain for $\overline{\mu}$ in the second model and right, a trace plot.

There is one potentially problematic parameter in the second model: $\overline{\mu}$. However, the effective sample size is still sufficiently large, the $\widehat{R}$ value is close enough to 1, and the trace plot does not exhibit any pathologies, so we are satisfied with the performance of this model.

A.3 Model with back-to-back game covariate

```
summary3 = az.summary(trace3)
problematic(summary3)
```

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
mu_bar	7.628	0.469	6.748	8.509	0.018	0.013	702.0	1796.0	1.01
mu[331]	1.527	0.698	0.285	2.881	0.022	0.016	969.0	2579.0	1.00

Once again, $\overline{\mu}$ is the parameter of greatest concern, so we focus on plotting its trace plot.

```
az.plot_trace(trace3, var_names=["mu_bar"], figsize=(13, 5), compact=False)
```

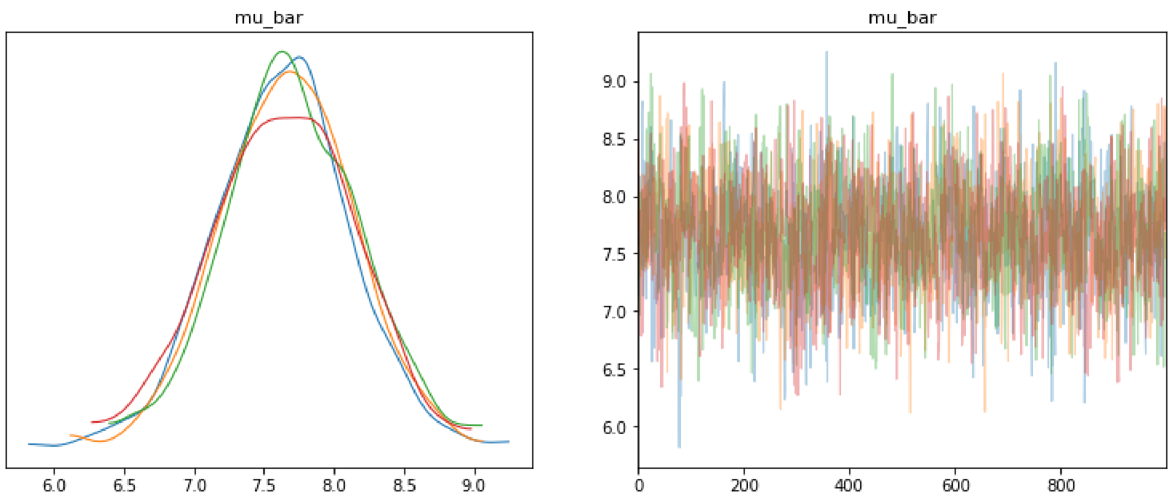

Figure 6: Left, a plot of the posterior probability distribution from each chain for $\overline{\mu}$ in the third model and right, its trace plot.

The trace plot does not appear pathological, so we are satisfied with the performance of the model.