Who wins the slot?

Forecasting audiences of big TV shows as a competition between channels: XGBoost, a neural network, Mistral Timestral and a joint competitive share model

Matheus Hoffmann ORCID

November 2026 · Preprint v0 (draft, not peer reviewed) · rev. f5fd12d

Abstract. A TV audience forecast for one channel ignores a plain fact: the viewers who watch a big show on one channel are not watching the rival channels in the same hour. We compare five forecasters of prime-time audiences, seven days ahead: XGBoost, a multilayer perceptron (DNN), Mistral's Timestral time-series foundation model, a Joint Competitive Share (JCS) model, and a neurosymbolic version of JCS. JCS predicts all channels of a slot together. It forecasts the pool of TV viewers in the hour and splits it between the channels and an outside option with a multinomial logit that sees every rival's line-up. We test the models on a simulated benchmark of three markets with three channels each: 28,620 training and 8,208 test channel-slots, with 129 big shows in the test period. All data in the benchmark are synthetic. XGBoost with rival features has the lowest error over all slots (15.1% WAPE). JCS has the lowest error on big shows (20.1% against 21.3%) and on big shows in contested slots (20.9% against 23.5%), but the paired 95% interval of the big-show difference (-3.3 to +0.6 points) includes zero. The DNN is worse than both (20.7% overall). The Timestral arm did not run, because our compute environment could not download its weights; its runner is ready. We name nine channels with public ratings (BBC One, ITV1 and Channel 4 in the UK; ABC, CBS and NBC in the US; Globo, Record and SBT in Brazil) and a protocol that scores frozen forecasts 14 days after broadcast. We also show where the Nixtla libraries fit.

Keywords: audience forecasting; television ratings; discrete choice; XGBoost; neurosymbolic models; forecast reconciliation.

1 Introduction

A broadcaster sells a big show before it airs. The price of an advertising break depends on a forecast of its audience, and the forecast decides how the channel schedules the rest of the night. The usual approach treats each channel as its own time series: last week's audience in the same hour, the programme's genre, the calendar and a model on top.

That approach misses the competition. At 21h on a Sunday, the viewers who watch a football final on one channel do not watch the drama on the next. When two channels put their strongest shows in the same hour, both lose viewers to each other. A model that sees only its own channel cannot know this. It can also predict that three channels together draw more viewers than the number of people in front of a TV set.

This paper asks a narrow question. For big shows on the main free-to-air channels, does a model that predicts all channels of a slot together beat strong single-channel models? We compare five forecasters:

1. **XGBoost** (Chen and Guestrin 2016), gradient-boosted trees on one row per channel and slot;
2. **a deep neural network (DNN)**, a multilayer perceptron on the same features;
3. **Mistral Timestral** (Mistral AI 2026), a pretrained time-series foundation model used zero-shot;

4. **the Joint Competitive Share (JCS) model**, which forecasts the pool of viewers in the hour and splits it between the channels with a discrete-choice model;
5. **a neurosymbolic JCS**, in which a neural network learns the utilities inside the same symbolic structure.

We did not call the fourth approach "game theory". The channels do play a scheduling game: each one chooses what to air knowing that its rivals do the same. But the forecaster does not solve that game. It takes the published line-ups as given and predicts how viewers split between them. "Joint competitive share" names what the model does: one joint forecast for all competitors of a slot, built from shares of a common pool.

Real minute-by-minute ratings are sold under licence, so we test the models on a simulated benchmark (Section 3). Every number in Section 5 comes from synthetic data. To move from simulation to real channels, Section 6 names nine channels whose ratings are published and a protocol to score frozen forecasts against them. Section 7 shows how the open-source Nixtla libraries would shorten that work.

2 The slot as a market

Take one market (a country), one day and one hour of prime time. We call this a *slot* j. In the slot, C channels each air one programme. Let P_j be the number of people watching any TV set in the hour: the *pool*. Panels measure it as total TV usage. Each viewer in the pool watches one of the C channels or something else: another channel, a streaming service or a game console. We call that last group the *outside option*, with index 0.

The audience of channel c is

$$A_{cj} = P_j \, s_{cj}, \qquad \sum_{c=1}^{C} s_{cj} \leq 1. \tag{1}$$

The shares s_{cj} come from a multinomial logit, the standard model of choice between differentiated products (Berry 1994):

$$s_{cj} = \frac{\exp(u_{cj})}{1 + \sum_{k=1}^{C} \exp(u_{kj})}, \qquad u_{0j} = 0. \tag{2}$$

The utility u_{cj} of a channel depends on its own programme and on its rivals' programmes. Viewing choice models of this kind go back to the audience flow model of Rust and Alpert (Rust and Alpert 1984), which also added *lead-in*: viewers tend to stay on the channel they watched in the previous hour.

Eq. 1 and eq. 2 encode three facts that a single-channel model must learn from data, if it can learn them at all:

1. **Conservation.** Channel audiences cannot add up to more than the pool.
2. **Cannibalisation.** A stronger rival programme lowers every other channel's share.
3. **Market expansion.** A big show also raises P_j: people turn the TV on for a final.

A big show is where all three act at once. That is why we score the models separately on big shows and on *contested* slots, where a big show meets another big show or a programme of the same genre.

3 A simulated benchmark

Every audience number in this section and in Section 5 is synthetic. The generator, `tv-audience/python/simulate.py`, follows eq. 1 and eq. 2 with known parameters. A simulation lets us check each model against a known truth before we spend real data on it. It also has a cost: a model that matches the generator's structure has an advantage that real data may not give it. Section 8 returns to this.

3.1 Markets, channels and slots

The benchmark has three markets, labelled UK, US and BR, with three channels each. The channel names (UK-1 to BR-3) are placeholders, not real broadcasters. Each day has three one-hour prime-time slots, at 20h, 21h and 22h. The data run from 2 January 2023 to 31 October 2026.

Table 1: The simulated markets. UK has a clear leader, US three near-equal networks, BR one dominant channel.

Market	Mean pool (millions)	Mean shares of channels 1, 2, 3	Mean outside share	Test big shows
UK	15.1	0.28, 0.15, 0.12	0.46	39
US	50.3	0.17, 0.19, 0.17	0.47	43
BR	33.9	0.37, 0.13, 0.11	0.39	47

3.2 How the generator works

The pool. The pool of each slot starts from a market level (18, 60 and 40 million viewers at 21h on an average 2023 weekday). Multipliers set the hour (0.90, 1.00, 0.80), the day of the week (0.88 on Saturday to 1.04 on Sunday), the season (plus or minus 10% in UK and US, 6% in BR with a mid-year peak) and the year-end holidays (0.85). Linear viewing falls by 3.5% a year. A big show raises the pool by $0.35 \log(1 + \text{strength})$.

Programmes. Each channel and slot rotates six series in 13-week seasons. A series has a genre (news, drama, reality, sport, comedy or film), a base appeal and an appeal loss per season. Genres have a different appeal at each hour.

Big shows. Each channel airs about 31 events a year: premieres, finales, specials and live finals. Event strength is log-normal with a heavy right tail, capped at 4. A *big show* is an event with strength of at least 0.8; in the test period it has a median audience 2.2 times the channel's usual audience in that hour. Channels also counter-program: when two events fall in the same slot, the weaker one moves 1 to 3 days later with probability 0.5.

Shares. The utility of a channel adds its brand, the genre appeal at that hour, the series appeal, $0.9 \log(1 + \text{strength})$ for an event, a lead-in term from the channel's share in the previous hour, and a penalty of 0.45 for each rival that airs the same genre. The outside option starts at utility 0.9 and gains 0.08 to 0.12 a year, as streaming takes viewers. Each audience gets a 6% multiplicative noise.

What the models see. Forecasts are made 7 days ahead, the lead time of a published schedule. A model sees the line-up of every channel (genre, episode type), audiences and pools at least 7 days old, the calendar, and a *hype* signal for each event. Hype is $\log(1 + \text{strength})$ plus Gaussian noise with standard deviation 0.35. It stands for pre-broadcast signals such

as trailers, social media buzz or a previous season's finale. The panel reports the pool with a 2% error. The models never see the true strengths, the series appeal or the same-night lead-in.

3.3 Train and test split

We train on 6 February 2023 to 31 December 2025 (the first five weeks only feed the lag features) and test on 1 January to 31 October 2026. That gives 28,620 training and 8,208 test channel-slots, in 2,736 test slots. The test set has 129 big shows. 46 of them air in a contested slot.

4 Five forecasters

All models forecast 7 days ahead and share one feature set (Table 2). The *own* features describe the channel's programme and history. The *rival* features describe the other channels in the same slot. A seasonal naive forecast, the audience of the same channel and hour 7 days earlier, sets the floor.

Table 2: Features available 7 days before broadcast.

Group	Features
Own	market, channel, genre, episode type, hype, $\log(1 + \text{hype})$, log audience 7, 14, 21 and 28 days earlier and their mean, last audience of the same series, new-series flag, hour, day of week, day of year (sine and cosine), holiday flag
Rival	number of rivals with the same genre, highest and summed rival hype, rivals' log audience 7 days earlier, log pool 7 days earlier and its 4-week mean

4.1 XGBoost

XGBoost (Chen and Guestrin 2016) fits gradient-boosted regression trees. Its target is the log audience minus the channel's 4-week mean log audience in that hour, so the trees learn changes from a level, not the level itself. We use 600 trees of depth 6, learning rate 0.03, row and column subsampling of 0.8 and L2 penalty 5. We fit it twice: with the own features only, and with the rival features added. The second version tells us how much a single-channel model gains from seeing its rivals, without any structure.

4.2 Deep neural network

The DNN is a multilayer perceptron with three hidden layers (128, 64 and 32 units), the same target and the own features, trained with Adam, an L2 penalty of 10^{-3} and early stopping on a 10% validation split (scikit-learn).

4.3 Mistral Timestral

Timestral is Mistral AI's family of time-series foundation models, released on the Hugging Face Hub in four sizes from 5 to 209 million parameters (Mistral AI 2026). It forecasts zero-shot: it is pretrained on many series and not fitted to ours. It returns 21 quantiles per step, takes up to 32,768 past steps and accepts past, known-future and categorical covariates. We use the largest model, `mistralai/Timestral-209M`. For each forecast origin, its context is the full log-audience history of each channel and hour up to 7 days before the target. We take the median of step 7.

Timestral is the foundation-model counterpart of the earlier idea of prompting a general language model with a series written as text (Gruver et al. 2023). We first planned that route with a Mistral chat model (Jiang et al. 2023). A model trained on time series is the fairer comparison.

This arm did not run for this draft. The compute environment blocked the Hugging Face Hub, which hosts the weights, and the PyTorch package index. The runner (`run_timestral` in `tv-audience/python/experiment.py`) is in place and skips itself when the model cannot load, so Section 5 reports Timestral as "not run". We do not stand in a guess for its numbers. Our runner also uses the history alone. Feeding the rivals' line-up as known-future covariates is the natural next step, and the one that would test Timestral on the competition question.

4.4 Joint Competitive Share (JCS) model

JCS forecasts the two factors of eq. 1 separately and multiplies them.

1. **Pool.** XGBoost, with the settings above, forecasts the log pool of the slot as a change from its 4-week mean. Its features are the calendar, the pool history and the highest and summed hype of the line-up, so a big show can raise the pool.
2. **Shares.** A multinomial logit (eq. 2) gives the shares of all C channels and the outside option at once. The utility is linear in: channel dummies (brand), genre-by-hour dummies, episode-type dummies, $\log(1 + \text{hype})$, the channel's log-odds share 7 days earlier and over 4 weeks (lead-in and habit), the number of same-genre rivals, the highest and summed rival hype, the new-series flag, the holiday flag and a market-specific time trend for the outside option. We fit it by maximum likelihood on the observed shares (a fractional multinomial logit), with L-BFGS and a small L2 penalty.

The forecast for channel c is $\hat{A}_{cj} = \hat{P}_j \hat{s}_{cj}$. By construction, the three forecasts of a slot never add up to more than the forecast pool, and a stronger rival lowers every other channel's forecast.

4.5 Neurosymbolic JCS

Neurosymbolic models join a neural network to symbolic knowledge or rules (Garcez and Lamb 2020). JCS is a natural host. The symbolic part stays fixed: one pool, one softmax per slot with an outside option, and audience equal to pool times share. The neural part learns the utility. We add a one-layer network (32 tanh units) to the linear utility of JCS: $u_{cj} = \beta^\top x_{cj} + w^\top \tanh(W x_{cj} + b)$. We train it through the softmax with the same likelihood, using Adam for 60 epochs on mini-batches of 512 slots, in plain NumPy. The network can find interactions, for example between genre, hour and hype, that the linear utility misses. It cannot break conservation or the competition between channels.

4.6 Reconciliation of independent forecasts

A third way to make independent forecasts coherent is to reconcile them after the fact. We forecast the outside audience of each slot with XGBoost, take the XGBoost-with-rivals channel forecasts and the pool forecast, and reconcile the hierarchy pool $= \sum_c A_c +$ outside with MinTrace (Wickramasuriya, Athanasopoulos, and Hyndman 2019) in its structurally weighted form, as implemented in Nixtla's HierarchicalForecast (Olivares et al. 2022).

4.7 Metrics

We report the weighted absolute percentage error, WAPE $= \sum |A - \hat{A}| / \sum A$, over all test channel-slots, over the 129 big shows, and over the 46 big shows in contested slots. A 95% interval for the big-show WAPE comes from 500 bootstrap resamples of test days. The same resamples give paired intervals for the difference between each model and XGBoost with rivals. *Winner accuracy* is the share of slots in which the model ranks the true leader first. A broadcaster cares about this number when two big shows meet.

5 Results on simulated data

Table 3 gives the test errors. Figure 1 shows the same numbers. All of them come from the synthetic benchmark of Section 3.

Table 3: Test errors, 1 January to 31 October 2026, simulated data. All slots: 8,208 channel-slots. Big shows: 129. Contested: 46 big shows that meet a rival big show or a same-genre programme. Winner accuracy: share of the 2,736 slots in which the model ranks the true leader first; event slots are the 209 slots in which at least one channel airs an event with a positive hype signal. Bold: best in the column.

Model	WAPE all	WAPE big shows (95% CI)	WAPE contested	Winner acc.	Winner acc., event slots
Seasonal naive	19.0%	55.1% (51.7 to 58.6)	52.8%	87.6%	73.2%
XGBoost	15.5%	21.7% (18.0 to 25.3)	25.0%	89.9%	84.7%
XGBoost + rivals	**15.1%**	21.3% (17.9 to 24.6)	23.5%	89.9%	88.0%
DNN (MLP)	20.7%	27.6% (23.3 to 32.0)	25.8%	87.4%	85.2%
Mistral Timestral	not run	not run	not run	not run	not run
JCS	16.2%	**20.1%** (16.9 to 23.2)	**20.9%**	89.3%	87.1%
Neurosymbolic JCS	15.9%	22.5% (19.3 to 26.2)	24.1%	**90.3%**	**88.5%**
XGBoost + rivals + MinT	16.6%	21.4% (18.0 to 24.7)	24.7%	89.9%	88.0%

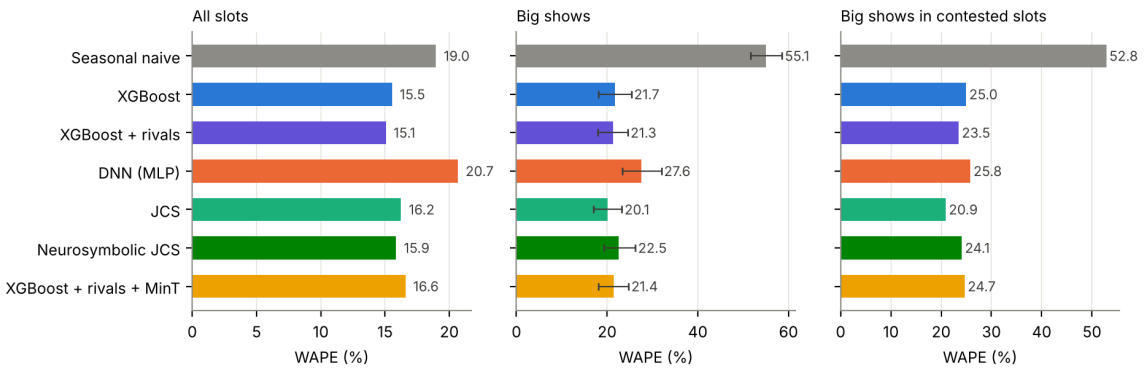


Figure 1: Weighted absolute percentage error by model. Left: all test channel-slots. Middle: big shows, with 95% day-bootstrap intervals. Right: big shows in contested slots. Simulated data. Timestral did not run and is not shown.

5.1 Four findings

1. On ordinary slots, single-channel XGBoost wins. XGBoost with rival features has the lowest error over all slots, 15.1%. JCS is 1.1 points worse; the paired 95% interval of the difference is +0.9 to +1.5 points (Table 4). Most slots hold no event, and there the history of the channel and hour is the best predictor. Trees use that history more flexibly than the linear utility of JCS.

2. On big shows, JCS has the lowest error, but the margin is not yet clear. JCS cuts the big-show error from 21.3% to 20.1%, and the contested-slot error from 23.5% to 20.9%. With 129 big shows, the paired interval of the big-show difference runs from -3.3 to +0.6 points. It includes zero. A longer test period or a real panel with more events is needed to settle it.

3. Seeing the rivals helps every model, a little. Adding rival features to XGBoost lowers its contested-slot error from 25.0% to 23.5% and lifts winner accuracy in event slots from 84.7% to 88.0%. Structure adds more on contested slots (JCS, 20.9%) than features alone.

4. The DNN is the weakest learned model. On the same features and target, the multilayer perceptron has 20.7% error over all slots, worse than the seasonal naive forecast (19.0%). An early version with a calendar trend feature did worse still (37.6%): the network extrapolated the trend outside the training range. Tabular data with strong lag features favour trees, and our DNN had no structure to lean on.

Table 4: Paired 95% day-bootstrap intervals of the WAPE difference against XGBoost with rivals, in percentage points. Below zero means better than XGBoost with rivals.

Model	All slots	Big shows
Seasonal naive	+3.2 to +4.6	+29.5 to +37.3
XGBoost	+0.3 to +0.7	-0.7 to +1.5
DNN (MLP)	+5.1 to +6.1	+2.8 to +9.6
JCS	+0.9 to +1.5	-3.3 to +0.6
Neurosymbolic JCS	+0.4 to +1.2	-1.7 to +4.2
XGBoost + rivals + MinT	+1.3 to +1.8	-0.7 to +1.1

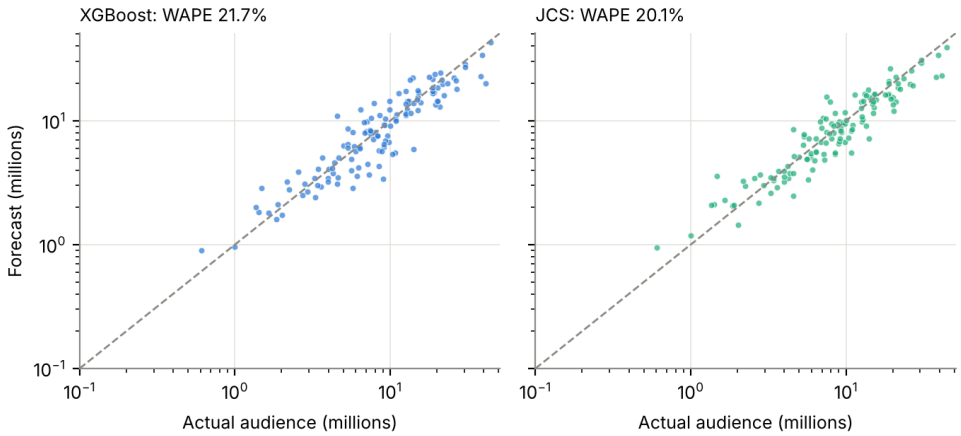


Figure 2: Big-show forecasts against actual audiences, log scales, simulated data. Left: XGBoost (own features). Right: JCS. The dashed line is a perfect forecast.

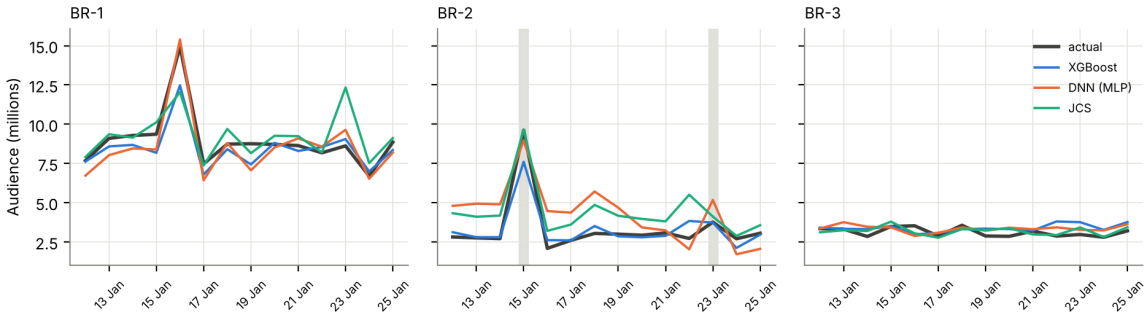


Figure 3: Two weeks of the simulated BR market at 21h. Black: actual audience. Grey bands: days on which the channel airs an event. Simulated data.

5.2 Big shows up close

Figure 2 plots each big-show forecast against the truth. The seasonal naive forecast fails on big shows (55.1%) because a big show has a median audience 2.2 times the usual one for its channel and hour. Every learned model reads the hype signal and the episode type, so all of them cut that error by more than half.

Figure 3 shows two weeks of the simulated BR market at 21h. The forecasts track the spikes. They differ most on the non-event days around them.

5.3 The neurosymbolic variant

The neural utility lowers the all-slot error of JCS from 16.2% to 15.9% and gives the best winner accuracy (90.3% of all slots, 88.5% of event slots). It does worse on big shows (22.5%). Big shows are rare (1.6% of rows), and the network spends its capacity on the common case. We read this as a promising but unfinished result: the structure works with a neural utility, but the network needs either more events or a loss that weights them.

5.4 Coherence

No learned model forecast more viewers for the three channels of a slot than the measured pool in our test: the outside option takes 44% of the pool on average, which leaves room. The seasonal naive forecast broke that bound in 0.5% of slots. JCS keeps the bound by construction. In a market where the main channels hold most of the pool, such as a big final in Brazil, independent forecasts would break it more often.

6 From simulation to real channels: a ground-truth protocol

A simulation can rank models but cannot prove that they work on real viewers. Minute-level panel data are licensed, but each of the three markets below publishes a free summary of its ratings. That is enough for an honest test of big shows: freeze a forecast before the show airs, wait for the published number, and score.

6.1 Nine channels with public ratings

Table 5: Channels with published audience figures, three per market.

Market	Channel	Public source of the actual audience	Metric	Published
UK	BBC One	Barb, weekly top 50 programmes (Barb 2026)	consolidated 7-day average audience	weekly
UK	ITV1	same	same	weekly
UK	Channel 4	same	same	weekly
US	ABC	Nielsen, Top 10 linear TV, broadcast (Nielsen 2026)	P2+ average audience and household rating	weekly
US	CBS	same	same	weekly
US	NBC	same	same	weekly
BR	Globo	Kantar IBOPE Media, weekly top 10 per channel (Fernandes 2016)	reach, individual and household audience, 15 markets	weekly, on Tuesday, for the week before
BR	Record	same	same	same
BR	SBT	same	same	same

We checked each source on 7 November 2026, except the Kantar IBOPE ranking. The Barb dashboard covers "all broadcast channels and SVOD services" each week, with broadcast data from 27 August 2018 (Barb 2026). Nielsen's page listed ABC, CBS and NBC programmes in its broadcast top 10 for the week of 26 October to 1 November 2026, six days before our check (Nielsen 2026). For Brazil we found only the 2016 announcement: Kantar IBOPE Media began to publish, every Tuesday, the top 10 programmes of Band, Globo, Record,

RedeTV! and SBT across its 15 measured markets (Fernandes 2016). **We did not confirm that this ranking is still published in 2026.** A reader should check it before relying on it. Brazilian newspapers also report Kantar's preliminary ratings for Grande São Paulo the morning after a show; those numbers are a weaker fallback, because they are preliminary and cover one city.

6.2 Protocol: score after 14 days

1. **Freeze.** Seven days before a big show airs, run every model on the published line-ups of all channels in the slot. Commit the forecasts to the repository. The commit time stamps them.
2. **Wait.** Barb's figure for a show includes 7 days of catch-up viewing, so it cannot exist before day 7. Nielsen showed a week's list 6 days after it ended. Kantar IBOPE publishes on the Tuesday after the week, at most 9 days after broadcast. We score every forecast on day 14 after broadcast, which covers all three sources with a margin.
3. **Score.** Compare each forecast with the published number. Report WAPE and winner accuracy per market, as in Section 5.

Two caveats apply. First, the public lists are top-N lists. A big show usually makes the list, but a weak rival in the same slot may not, so the protocol scores big shows well and their rivals poorly. Second, the three sources measure different things: 7-day consolidated viewers in the UK, live-plus-same-day viewers in the US lists, and audience over 15 markets in Brazil. Each market's models must be trained on the same metric they are scored on.

7 How Nixtla could help

Nixtla maintains a set of open-source Python libraries for forecasting, all under the Apache-2.0 licence, and a pretrained foundation model. Each one maps to a part of this study.

Table 6: Nixtla tools and the parts of this study they cover.

Need in this study	Nixtla tool	What it adds
Fast statistical baselines (seasonal naive, ETS, ARIMA) for thousands of channel-slot series	StatsForecast (Nixtla 2026c)	Baselines with cross-validation in a few lines; a fair floor for every model
XGBoost with lag features	MLForecast (Nixtla 2026a)	Lags, rolling means and exogenous variables built for any scikit-learn-style model, including XGBoost and LightGBM
Neural models	NeuralForecast (Nixtla 2026b)	More than 30 models (NHITS, TFT, PatchTST and others) with exogenous and static covariates; a stronger DNN arm than our perceptron

Need in this study	Nixtla tool	What it adds
Coherent forecasts across channels	HierarchicalForecast (Olivares et al. 2022)	Reconciliation (BottomUp, TopDown, MinTrace, ERM) and probabilistic coherent forecasts
A second foundation model	TimeGPT (Garza, Challu, and Mergenthaler-Canseco 2023)	Zero-shot forecasts from a pretrained model, a benchmark for Timestral

We used one of them. HierarchicalForecast reconciled the XGBoost forecasts so that the three channels and the outside option add up to the pool. The result is a fair test of "coherence after the fact" against JCS's "coherence by construction". In our benchmark reconciliation did not help. MinTrace with structural weights left the big-show error at 21.4% (21.3% before) and raised the all-slot error from 15.1% to 16.6% (Table 3). The pool and outside-option forecasts carry their own errors, and reconciliation spreads them over the channels.

Three uses look more promising than the one we tried.

1. **Reconcile JCS with real hierarchies.** Real panels report audiences by region and by demographic group. HierarchicalForecast can keep a national JCS forecast coherent with its regional parts, a hierarchy that JCS alone does not handle.
2. **Rebuild the baselines with less code.** StatsForecast and MLForecast would replace our hand-written lag features and seasonal naive forecast, and their cross-validation would give a rolling test over many origins instead of our single split.
3. **Stress-test the foundation-model arm.** Running TimeGPT next to Timestral, with the rivals' line-up as known-future covariates where the model accepts them, would show whether a foundation model can learn the competition that JCS writes down by hand.

8 Discussion and limits

The results suggest a division of labour. Trees win on ordinary slots, where the past of a channel and hour explains most of the future. The joint model wins on big shows and in contested slots, where the competition decides the outcome. The margin on big shows is 1.2 points and its interval includes zero, so we state it as a lead, not as a proof.

A hybrid follows from this: keep the JCS structure (pool times shares, one softmax per slot) and learn the utility with gradient-boosted trees instead of a linear function or a small network. The neurosymbolic variant shows that the structure accepts a learned utility. It also shows that a learned utility needs care on rare events.

8.1 Is a neurosymbolic approach suitable?

Yes, for this problem in particular. Audience forecasting has hard rules that hold for every channel and every night: shares of a pool cannot exceed one, viewers choose one programme at a time, and a stronger rival takes viewers from the others. These rules are cheap to write down and expensive to learn from data, above all for rare big shows. A neurosymbolic model writes them into the architecture and learns only what is uncertain, the utility of each programme. Our first version matches JCS over all slots and gives the best winner accuracy, but it does not yet beat JCS on big shows. The constraint layer could also carry

more rules: a channel's contractual broadcast of a match, regulatory news slots, or the rule that a programme that ends at 22h cannot lead into the 22h slot.

8.2 Limits

1. **Synthetic data.** The benchmark was built from the same structure that JCS assumes: a pool and a logit. That favours JCS. On real data the share structure may hold less well, for example when viewers record shows or watch on several screens.
2. **One split.** We test on one period with one seed. A rolling-origin evaluation over several seeds would give wider and more honest intervals.
3. **Few big shows.** 129 test big shows and 46 contested ones give wide intervals.
4. **The Timestral arm did not run.** Its result is the largest gap in this draft.
5. **Fixed schedules.** We take the line-ups as given. Channels react to each other's schedules, sometimes days before broadcast. A model of that scheduling game is a separate question from the forecast, and we leave it out.

The next step is the protocol of Section 6: frozen forecasts for real big shows on the nine channels, scored on day 14.

Statements

Synthetic data. Every audience figure in Section 3 and Section 5 comes from a simulation. None describes a real channel or programme. Channel names in Section 6 name public data sources only.

Use of AI. Claude (Anthropic), an AI assistant, wrote the simulator and the analysis code, ran it, and drafted this text at the author's request. The author is responsible for the content.

Reproducibility. `python3 tv-audience/python/experiment.py` regenerates every number, table and figure from a fixed seed. `tv-audience/README.md` lists the Python packages it needs.

References

Barb. 2026. "Weekly Top 50 Programmes." Barb viewing data dashboards (As Viewed, Jan 2024 onward; As Broadcast, Aug 2018 to Aug 2025), accessed 7 November 2026. https://www.barb.co.uk/viewing-data/as-viewed-top-50/.

Berry, Steven T. 1994. "Estimating Discrete-Choice Models of Product Differentiation." *The RAND Journal of Economics* 25 (2): 242–62. https://doi.org/10.2307/2555829.

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785.

Fernandes, Guilherme. 2016. "Ibope Abre Mais Dados de TV Aberta." Meio & Mensagem, 13 July 2016. https://www.meioemensagem.com.br/ultimas-noticias/ibope-abre-mais-dados-de-tv-aberta.

Garcez, Artur d'Avila, and Luís C. Lamb. 2020. "Neurosymbolic AI: The 3rd Wave." arXiv:2012.05876. https://doi.org/10.48550/arXiv.2012.05876.

Garza, Azul, Cristian Challu, and Max Mergenthaler-Canseco. 2023. "TimeGPT-1." arXiv:2310.03589. https://doi.org/10.48550/arXiv.2310.03589.

Gruver, Nate, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023. "Large Language Models Are Zero-Shot Time Series Forecasters." NeurIPS 2023; arXiv:2310.07820. https://doi.org/10.48550/arXiv.2310.07820.

Jiang, Albert Q., Alexandre Sablayrolles, Arthur Mensch, et al. 2023. "Mistral 7B." arXiv:2310.06825. https://doi.org/10.48550/arXiv.2310.06825.

Mistral AI. 2026. "Timestral: Time-Series Foundation Models." GitHub repository mistralai/timestral and Hugging Face models mistralai/Timestral-5M to Timestral-209M, accessed 7 November 2026. https://github.com/mistralai/timestral.

Nielsen. 2026. "Nielsen Top 10." Nielsen data center, linear TV list for the week of 26 October to 1 November 2026, accessed 7 November 2026. https://www.nielsen.com/data-center/top-ten/.

Nixtla. 2026a. "MLForecast: Scalable Machine Learning for Time Series Forecasting." GitHub repository, Apache-2.0, accessed 7 November 2026. https://github.com/Nixtla/mlforecast.

———. 2026b. "NeuralForecast: Scalable and User Friendly Neural Forecasting Algorithms." GitHub repository, Apache-2.0, accessed 7 November 2026. https://github.com/Nixtla/neuralforecast.

———. 2026c. "StatsForecast: Lightning Fast Forecasting with Statistical and Econometric Models." GitHub repository, Apache-2.0, accessed 7 November 2026. https://github.com/Nixtla/statsforecast.

Olivares, Kin G., Azul Garza, David Luo, Cristian Challú, Max Mergenthaler, Souhaib Ben Taieb, Shanika L. Wickramasuriya, and Artur Dubrawski. 2022. "HierarchicalForecast: A Reference Framework for Hierarchical Forecasting in Python." arXiv:2207.03517. https://doi.org/10.48550/arXiv.2207.03517.

Rust, Roland T., and Mark I. Alpert. 1984. "An Audience Flow Model of Television Viewing Choice." *Marketing Science* 3 (2): 113–24. https://doi.org/10.1287/mksc.3.2.113.

Wickramasuriya, Shanika L., George Athanasopoulos, and Rob J. Hyndman. 2019. "Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization." *Journal of the American Statistical Association* 114 (526): 804–19. https://doi.org/10.1080/01621459.2018.1448825.