The VoteHub 2026 Midterm Forecasting Model

Zachary Donnini, Ryan McComb, Varun Vishwanath,
Austin Park, Luke Wines, and Garrett Herrin
VoteHub

April 2026

Abstract

We present VoteHub's 2026 midterm forecasting methodology for U.S. House, Senate, and gubernatorial elections. The model combines fundamentals, polling, expert ratings, prediction markets, and correlated simulation into a modular forecasting system designed to produce coherent race-level and chamber-level probabilistic forecasts. We estimate two model specifications: a Legacy model that incorporates fundamentals and polling data, and a Complete model that additionally integrates expert ratings and prediction market information. Back-testing suggests that prediction markets have demonstrated a strong recent track record in forecasting electoral outcomes.

The structural core of the system is a stacked regression framework trained and validated by holding out entire election years rather than random individual races, better reflecting the real forecasting problem of generalizing from past cycles to a new electoral environment. Polling is incorporated through a weighting scheme that accounts for recency, pollster quality, repeated releases, sponsorship type, and the effective number of polls (ENOP), which measures the information content of each race's polling average. Point estimates are then refined using expert race ratings and liquidity-adjusted prediction market data, with market-implied win probabilities transformed into implied margin estimates. To model joint uncertainty across races, the model includes a continuous similarity-based correlated simulation framework. The result is a temporally validated forecasting model that remains responsive to current-cycle information while preserving interpretable structure across races, offices, and chambers.

1 Fundamentals

We model fundamentals using a stacked machine-learning ensemble designed to generalize across election cycles rather than merely across individual races. The central idea is to treat *year* as the fundamental validation unit because there are different dynamics in each year. In every validation split, the model is trained on all years except one and then evaluated on the held-out year. This produces a procedure that is much closer to the real forecasting problem than ordinary race-level cross-validation, since the model must extrapolate from prior cycles to a new electoral environment.

The framework is implemented separately for House, Senate, and gubernatorial elections. In each case, the target variable is a continuous election outcome (two-party Republican margin of victory), denoted by

$$Y_i \in \mathbb{R},$$

and the predictors are a vector of pre-election fundamentals and lagged electoral indicators,

$$X_i \in \mathbb{R}^p.$$

For each race, the feature set includes prior electoral performance in the district at the congressional, gubernatorial, and presidential levels, adjusted for the estimated change in the race-specific political environment. It also incorporates structural indicators such as fundraising, incumbency status, recent changes in party control, and other relevant district- and cycle-level characteristics.

The method proceeds in two layers. First, a collection of base learners is trained and validated in a year-heldout framework. Second, the predictions from the strongest base learners are combined through a meta-model, again using only out-of-fold information. The resulting system is therefore a temporally cross-validated stacked machine-learning model.

1.1 Data

Most inputs to the base learners are derived from relatively objective quantities, such as incumbency, fundraising metrics, and other structural information about the race. These variables require little subjective interpretation and are directly observable in historical data. A central challenge, however, is translating past electoral outcomes into expectations for a future cycle when the broader political environment is changing. This is key, as the three most influential variables in the base learners' feature importance are the adjusted results of the two most recent presidential elections in a district, along with candidates' prior performance in that race.

Past election results must be interpreted relative to the expected political environment in 2026. A district that voted Republican by two points in 2024, for example, would not necessarily imply a favorable Republican baseline in 2026 if the national environment is expected to shift substantially toward Democrats. Accordingly, a key task of the fundamentals layer is to estimate, for each race, the shift in the race political environment (RPE) between 2026 and the baseline elections used to anchor its partisan fundamentals.

The first step is to estimate the 2026 national political environment (NPE). This is difficult, especially far in advance of Election Day. Generic ballot polling provides a useful signal of the eventual national environment, but it must be interpreted carefully. In midterm cycles, wave dynamics often develop gradually, with the out-party frequently gaining strength over the course of the year. At the same time, polling samples evolve as likely-voter screens are introduced and refined. Early-cycle polls may therefore misrepresent the eventual electorate, particularly when turnout dynamics differ across coalitions.

In addition to polling, the model incorporates broader priors about midterm behavior. Historically, the out-party tends to perform quite well in midterm elections, with the magnitude of that advantage inversely correlating with presidential approval. These priors provide a baseline expectation for the direction and approximate scale of the national environment, which is then updated as new information becomes available. To avoid overreacting to short-term polling fluctuations far out from Election Day, the estimated environment is partially regularized toward this broader prior, with the weight on the prior declining to zero as Election Day approaches. At the time of the model's launch for the 2026 midterm elections, Donald Trump had a net approval rating of -17.5 in our polling average. This level of unpopularity shifts the model's baseline expectations in a more Democratic direction.

Once an estimate of the NPE is obtained, it must be translated into an RPE using elasticity. The first component is geographic elasticity, which captures how responsive voters in a given geographic unit are to changes in the national environment. Each geography is assigned an elasticity parameter $x_i^{(g)}$, where values near one indicate average responsiveness and values above or below one indicate more or less elastic electorates, respectively. These estimates are constructed using survey data from the ANES and CCES together with demographic data from the ACS. They are intended to capture systematic geographic variation in the responsiveness of voter partisan preferences to national political shifts.

A second component accounts for race-specific conditions that have recently become more salient determinants of elasticity, particularly competitiveness and the resulting turnout dynamics. Let $x_{i,t}^{(r)}$ denote the race-level elasticity adjustment for

race i in election t. Total elasticity is then given by

$$x_{i,t} = x_i^{(g)} \cdot x_{i,t}^{(r)},$$

so that the race political environment for race i in election t is

$$\mathrm{RPE}_{i,t} = x_{i,t} \cdot \mathrm{NPE}_t.$$

This framework also implies a clean way to compare environment-adjusted baselines across races and election years. For race i in election year t, total elasticity is

$$x_{i,t} = x_i^{(g)} x_{i,t}^{(r)},$$

so the race political environment is

$$\mathrm{RPE}_{i,t} = x_{i,t} \cdot \mathrm{NPE}_t.$$

For two races i and j, possibly anchored to different baseline elections, the difference in race political environments is therefore

$$\mathrm{RPE}_{i,t_i} - \mathrm{RPE}_{j,t_j} = x_{i,t_i}\mathrm{NPE}_{t_i} - x_{j,t_j}\mathrm{NPE}_{t_j}.$$

This formulation allows each race to have its own geographic elasticity, race-level elasticity adjustment, and baseline national environment, while preserving a consistent signed-margin interpretation.

As an example, suppose the 2024 presidential result in Kansas was $R + 16.1$, the 2024 national political environment was $R + 1.5$, and the expected 2026 national political environment is $D + 5$. Since outcomes are measured on the Republican-margin scale, $D + 5$ is written as -5. Suppose further that Kansas has geographic elasticity $x_i^{(g)} = 1.05$, and that the relevant race-level elasticity adjustments are $x_i^{(r,2024)} = 1.03$ and $x_i^{(r,2026)} = 1.02$.

Define the total adjustment from the 2024 baseline to the expected 2026 environment as

$$\Delta_i(2024 \to 2026) = (\mathrm{NPE}_{2024} - \mathrm{NPE}_{2026})\, x_i^{(g)} x_i^{(r,2024)} x_i^{(r,2026)}.$$

The adjusted 2024 presidential baseline is then

$$B_{i,2024}^{\mathrm{adj}} = B_{i,2024} - \Delta_i(2024 \to 2026).$$

Substituting values gives

$$B_{i,2024}^{\mathrm{adj}} = 16.1 - (1.5 - (-5))\,(1.05)(1.03)(1.02) \approx 8.93.$$

Thus, after adjusting for the expected national shift and Kansas's elasticity, the 2024 presidential result contributes an estimated $R + 8.93$ partisan baseline to the 2026 Kansas Senate race.

1.2 Base Learners

Fundamentals are formed by a diverse collection of regression models, each intended to capture a somewhat different type of signal-response relationship.

1. **Elastic Net.** A penalized linear model combining ℓ_1 and ℓ_2 regularization. This allows sparse or nearly sparse linear fits while retaining shrinkage and stability when predictors are correlated.

2. **Bayesian Ridge.** A fully regularized linear model with Gaussian priors on coefficients. Relative to ordinary least squares, this imposes stronger shrinkage and can be useful when the number of races is not large relative to the flexibility one might otherwise allow.

3. **Shallow Neural Network.** A feedforward multilayer perceptron with a small hidden architecture, intended to capture low-order nonlinearities and interactions beyond what a linear model can express.

4. **Deeper Neural Network.** A larger multilayer perceptron with greater representational capacity, allowing more complex nonlinear structure when supported by the data.

5. **Random Forest.** An ensemble of regression trees fit on bootstrap samples with feature subsampling. This model is designed to capture threshold effects, nonlinearities, and interaction structures without requiring them to be specified parametrically.

Some of these models require predictor standardization, which is performed within the fold-specific tuning process so that no information from the held-out year enters preprocessing. The individual base learners are all reasonably competitive with one another: average held-out RMSE differs only modestly across models, with the worst learner performing only slightly worse than the best in both the Senate (2% worse) and House (4% worse) settings. Model rankings do vary somewhat by sample size and task. The shallow MLP tends to outperform the deeper MLP in the Senate and gubernatorial stacks, where the estimation sample is smaller, while the deeper MLP performs better in the House stack, where more training data are available. Across all specifications, Elastic Net is consistently among the strongest base learners.

1.3 Grouped Hyperparameter Selection

Each base learner has its own hyperparameter space. Rather than fixing these tuning parameters ex ante, they are selected by randomized search using *grouped* cross-validation inside the training sample. This means that when a year t is held out in the outer validation loop, hyperparameters are chosen using only the remaining years, with additional year-aware folds inside that reduced training set.

Let the data be indexed by $i = 1, \ldots, N$, and let g_i denote the election year associated with observation i. For each outer fold with held-out year t, define

$$\mathcal{I}_{\text{train}}(t) = \{i : g_i \neq t\}, \qquad \mathcal{I}_{\text{test}}(t) = \{i : g_i = t\}.$$

For each candidate hyperparameter vector λ for model m, an inner grouped cross-validation procedure estimates predictive error on $\mathcal{I}_{\text{train}}(t)$. The hyperparameter vector chosen for that fold is

$$\hat{\lambda}_{m,t} = \arg\min_{\lambda \in \Lambda_m} \widehat{\text{RMSE}}^{\text{inner}}_{m,t}(\lambda),$$

where Λ_m denotes the search space for model m, and the inner criterion is root mean squared error.

Thus, each base learner is not only fit separately for each held-out year, but also tuned separately for each held-out year. This is important because it prevents a form of leakage in which the tuning process indirectly learns about the held-out cycle.

1.4 Year-Heldout Out-of-Fold Prediction

After choosing $\hat{\lambda}_{m,t}$, the corresponding model is fit on all observations from years other than t, and predictions are generated for the held-out year. For each model m, this yields an out-of-fold prediction

$$\hat{Y}_i^{(m,-g_i)},$$

meaning the prediction for observation i from a model that excluded observation i's year from training.

Collecting these predictions over all years produces a complete out-of-fold prediction vector for each base learner:

$$\hat{Y}^{(m,\text{OOF})} = \left(\hat{Y}_1^{(m,-g_1)}, \ldots, \hat{Y}_N^{(m,-g_N)}\right).$$

These out-of-fold predictions serve two purposes. First, they provide a year-respecting estimate of each model's predictive performance. Second, they form the inputs to the stacking stage.

1.5 Model-Specific Training Samples

The year-heldout method is used in three related but distinct settings.

House model. The House stacker is estimated and evaluated only on House races. Every out-of-fold error reported for this model refers only to House observations.

Governor model. The gubernatorial stacker is estimated and evaluated only on gubernatorial races.

Senate model. The Senate stacker is built differently. It is trained on a pooled congressional sample containing both House and Senate races, but evaluation is restricted to Senate observations. The purpose is to let the model learn broader congressional relationships from a larger estimation sample, improving stability, while still optimizing for Senate-specific predictive accuracy.

To prevent the pooled sample from being dominated by House races, Senate observations are given greater fitting weight. Let w_i denote the sample weight for observation i. Then in the Senate model,

$$w_i = \begin{cases} \omega_S & \text{if observation } i \text{ is a Senate race,} \\ 1 & \text{otherwise,} \end{cases}$$

where $\omega_S > 1$ is a fixed Senate upweighting constant. The consequence is that House races help estimate common signal, but Senate races exert more influence on the fitted relationship and are the only races used when computing reported Senate validation error.

1.6 Evaluating the Base Learners

For each base learner m, fold-specific errors are computed on the held-out year. The main loss function is root mean squared error,

$$\mathrm{RMSE}_{m,t} = \sqrt{\frac{1}{|\mathcal{E}_t|} \sum_{i \in \mathcal{E}_t} \left(Y_i - \hat{Y}_i^{(m,-t)} \right)^2},$$

where $\mathcal{E}_t$ is the relevant evaluation subset in year t. In the House and gubernatorial settings, $\mathcal{E}_t$ is all held-out observations in that year. In the Senate setting, $\mathcal{E}_t$ is only the held-out Senate observations.

Mean absolute error is also computed,

$$\mathrm{MAE}_{m,t} = \frac{1}{|\mathcal{E}_t|} \sum_{i \in \mathcal{E}_t} \left| Y_i - \hat{Y}_i^{(m,-t)} \right|,$$

but the principal ranking criterion is average held-out RMSE across years:

$$\overline{\mathrm{RMSE}}_m = \frac{1}{T} \sum_{t=1}^{T} \mathrm{RMSE}_{m,t}.$$

Although base-learner evaluation does not directly determine the final forecast, it played an important role in model development by allowing us to compare many alternative specifications, refine the training process, and retain the strongest versions of each learner for the fundamentals ensemble.

1.7 Stacking Layer

Once the base learners have produced out-of-fold predictions, the method constructs a second-stage design matrix

$$Z_i = \left(\hat{Y}_i^{(m_1,\mathrm{OOF})}, \ldots, \hat{Y}_i^{(m_M,\mathrm{OOF})} \right)^\top.$$

This matrix summarizes, for each race, how the strongest first-stage models predicted that race when its year was excluded from training.

The stacking model is a ridge regression with no intercept:

$$Y_i \approx Z_i^\top \beta.$$

The no-intercept restriction ensures that the final forecast remains a weighted combination of the base learners' predictions rather than allowing an additional additive adjustment at the stacking stage. We considered including an intercept, but in practice this introduced a noticeable Democratic shift in Senate forecasts. Because Democrats overperformed in many battleground Senate races in both 2022 and 2024, we were concerned that allowing an intercept would overfit to a very limited number of recent cycles and effectively bake a persistent pro-Democratic bias into the Senate fundamentals model. Given the small sample of recent Senate environments, we did not view two cycles as sufficient evidence to justify a systematic 1 to 2 point Democratic adjustment in key races across the map. Although Democratic Senate candidates have overperformed by a statistically significant amount in the last two

cycles, primary dynamics are fluid. We therefore did not want to hard-code that recent pattern too aggressively into the fundamentals model, particularly because incumbent quality can be inferred from observed electoral history and challenger quality can be approached through other indicators discussed later. This was one of the more consequential modeling choices in the framework, and it illustrates the broader set of subjective modeling choices involved in building an election forecast. On launch, allowing an intercept would have made Democratic Senate control odds slightly higher.

The ridge coefficients are chosen by minimizing

$$\sum_i \left(Y_i - Z_i^\top \beta\right)^2 + \alpha \|\beta\|_2^2,$$

where the penalty parameter α is itself selected by leave-one-year-out validation on the out-of-fold meta-design matrix. In other words, the stacking stage is tuned using the same temporal logic as the base learners.

After fitting, the coefficient vector is normalized to sum to one:

$$\tilde{\beta} = \frac{\hat{\beta}}{\sum_{m=1}^{M} \hat{\beta}_m},$$

whenever the denominator is nonzero. Thus the final stacked prediction is

$$\hat{Y}_i^{\text{stack}} = Z_i^\top \tilde{\beta} = \sum_{m=1}^{M} \tilde{\beta}_m \hat{Y}_i^{(m,\text{OOF})}.$$

This normalization makes the final predictor easier to interpret as an affine combination of base learners and prevents the scale of the meta-learner from drifting arbitrarily. As ridge coefficients are all nonnegative in federal models, this is also a weighted average of the base learner predictions.

1.8 Refitting for Prediction

The out-of-fold procedure is used for evaluation and for constructing the stacking layer, but the model ultimately intended for forecasting is trained using all available observations in the relevant sample. After the strongest base learners have been identified, each of them is re-tuned using grouped cross-validation on the full estimation sample and then refit on that full sample. The stacking coefficients, however, continue to come from the out-of-fold meta-stage, since those are the coefficients estimated from leakage-free first-stage predictions.

For a new observation with feature vector X_{new}, each retained base learner produces a prediction

$$\hat{Y}_{\text{new}}^{(m_j)}, \qquad j = 1, \ldots, M.$$

These predictions are stacked into a vector

$$Z_{\text{new}} = \left(\hat{Y}_{\text{new}}^{(m_1)}, \ldots, \hat{Y}_{\text{new}}^{(m_M)} \right)^{\top},$$

and the ensemble forecast is

$$\hat{Y}_{\text{new}}^{\text{stack}} = Z_{\text{new}}^{\top} \tilde{\beta}.$$

1.9 2026 Predicted Dynamics Adjustment

For the 2026 midterms, the method allows a cycle-specific structural adjustment that is imposed on top of the stacked fundamentals forecast. This adjustment is not estimated directly by the machine-learning pipeline. Instead, it reflects the VoteHub data science team's prior about systemic shifts from 2024 to 2026 based on recent data.

This is necessarily a judgmental component of the model. Because the task is to forecast a future electoral environment that has not yet fully occurred, there is no fully objective machine-learning procedure that can learn these shifts directly from past labeled data alone. Any attempt to do so would either rely too heavily on a very small number of recent cycles or would implicitly assume that the relationships observed in prior elections carry forward unchanged. We do not view either approach as satisfactory in a period when the coalition structure of the electorate appears to be rapidly evolving.

Instead, the model incorporates an explicit adjustment meant to encode the team's beliefs about how the electoral environment in 2026 may differ from the environment observed in 2024. Those beliefs are informed by a broad collection of evidence, including current polling, 2025 gubernatorial elections, special elections, and historical precedent from prior midterm cycles. We recognize that this represents a sizable subjective modeling choice. However, the alternative is not the absence of an assumption but rather the implicit and less defensible assumption that no structural change has occurred.

In substantive terms, the VoteHub data science team has several high-conviction views about the direction of likely coalition movement between 2024 and 2026. For example, the team has a high-conviction view that Hispanic voters will trend toward Democrats over this period. The difficult question is not the direction of that trend so much as its magnitude. Similar considerations apply to other dimensions of the

electoral environment. The challenge is to translate a genuine shift in the underlying 2026 midterm data-generating process, albeit one with uncertain magnitude, into a disciplined structural adjustment rather than to pretend that it can be learned mechanically from historical training data alone.

The stacked model is being used as the core forecasting engine, while the 2026 adjustment is an explicit structural overlay intended to encode analyst beliefs about changes in the relationship between presidential partisanship, demographics, and downballot performance in the current cycle.

1.10 Race Adjustments

We also incorporate a set of small but substantively meaningful post-model adjustments. In certain cases, binary indicators are excluded from the base learners because the share of observations taking a non-zero value is extremely small, which can lead to instability in estimation. Instead, these effects are applied after the machine learning stage. Examples include whether a candidate is running as an independent against a single major-party opponent, is subject to an ethics committee investigation, or is included on the "List of federal political scandals in the United States" Wikipedia article.

In addition, we implement modest adjustments for challenger experience or quality for candidates running in statewide races. Larger adjustments are applied based on a candidate's historical performance in previous state or federal contests for different offices. By contrast, performance in prior races for the same office is directly incorporated into the base model's training data.

1.11 Summary

In practical terms, this is a year-heldout stacked ensemble. It learns from several model classes at once, judges them by how well they predict entirely unseen election years, combines them into a single ensemble forecast, and then overlays a 2026-specific structural shift and light race-level adjustments when producing current-cycle predictions.

2 From Fundamentals to Predictions

The forecasting model proceeds in layers. The legacy model only includes Stages I and II. The complete model includes stages I–VI. The first layer produces an office-

specific structural prior from fundamentals. This section takes that prior and describes how it is transformed into the live point-estimate framework.

Let i index races, and let

$$y_i$$

denote the Republican two-party margin in race i, measured in percentage points. Let $o(i) \in \{\text{House}, \text{Senate}, \text{Governor}\}$ denote office type.

2.1 Stage 1: Fundamentals

For each race i, the model begins with an imported structural forecast,

$$m_i^{\mathrm{F}},$$

which is which is produced upstream by the fundamentals model described in Section 1 and serves as the baseline expectation for the race before other information is incorporated.

Everything that follows should therefore be understood as an update to m_i^{F}.

2.2 Polling Average Construction and Effective Number of Polls

Before the structural forecast is updated with polling in Stage II, VoteHub computes for each race a current polling average and an associated effective number of polls (ENOP). These quantities are produced by a poll-level weighting procedure that combines recency, pollster quality, repeated-release control, and sponsorship discounts. The resulting average is designed to be responsive to new information without allowing stale or low-quality polling to 'flood the zone' and dominate the estimate.

Fix a target date t and a response option or candidate share indexed by c. Let $\mathcal{P}_c(t)$ denote the set of usable polls for that quantity. The averaging function operates on the post-processed poll matrix supplied to it, so any upstream sample-priority rules or house-effect adjustments are already embedded in the poll values by the time this stage begins. For each poll $p \in \mathcal{P}_c(t)$, let d_p denote its field end date, let $j(p)$ denote its pollster, and let y_{pc} denote its reported value for choice c. Polls are first ordered by recency, with rank

$$r_p \in \{1, 2, \dots\},$$

where $r_p = 1$ denotes the most recent poll in the usable set. The model then assigns each recency rank an ordinal decay base

$$b(r),$$

where $b(r)$ is a decreasing concave function of rank with a positive lower floor. Thus, more recent ranks receive a decay base closer to one, while less recent ranks receive smaller bases. The within-poll weight is then

$$u_{pc}(t) = b(r_p)^{\,t-d_p}.$$

This produces standard time decay, but with an important twist: two polls that are equally old in calendar time do not necessarily decay at the same rate. If one poll is the third most recent usable poll in the race and the other is the thirtieth most recent usable poll, the latter is discounted more aggressively because its rank-specific base is lower.

This rank-dependent decay is intentional and reflects a bias-variance tradeoff. When only a few polls exist for a given race, variance reduction is valuable: even an older third-most-recent poll still provides information, because averaging across a small set of polls helps stabilize the estimate against the noise of any one release. In contrast, when a race has a very large number of polls, the marginal variance-reduction benefit from retaining a very old thirtieth poll is minimal, while the bias cost from including data gathered under a potentially outdated race environment can be substantial.

2.2.1 Within-pollster aggregation

The first stage of aggregation occurs within pollster. For each pollster j and choice c, the code forms a weighted pollster-specific mean

$$\bar{y}_{jc}(t) = \frac{\sum_{p:\, j(p)=j} u_{pc}(t)\, y_{pc}}{\sum_{p:\, j(p)=j} u_{pc}(t)},$$

where the sum runs over usable polls from pollster j that report choice c. This step prevents pollsters that release many polls in the same race from exerting disproportionate influence simply by fielding more often. Multiple releases from the same pollster are still used, but they are first compressed into a single pollster-level signal.

Let

$$\Delta_{jc}(t) = \min_{p:\, j(p)=j} (t - d_p)$$

denote the age of pollster j's most recent usable poll for choice c, and let

$$r^{\star}_{jc}$$

denote the global recency rank of that most recent usable poll. Define a raw pollster-level weight

$$g_{jc}(t) = q_j \, b\big(r^{\star}_{jc}\big)^{\Delta_{jc}(t)},$$

where q_j is the pollster quality weight. Pollsters with stronger empirical track records receive larger q_j, and unrated or low-information pollsters receive a default fallback weight. These raw weights are then normalized across pollsters:

$$\pi_{jc}(t) = \frac{g_{jc}(t)}{\sum_{j'} g_{j'c}(t)}.$$

Two features of this construction are important. First, recency enters at the pollster level through the most recent usable release from that pollster, so a pollster that has recently been in the field is treated as more informative than one whose latest release is old. Second, repeated releases by the same pollster do not linearly increase total influence, because the normalization is performed after pollster-level compression. This is one of the main ways the procedure avoids overstating the information content of many highly correlated polls from the same source.

2.2.2 Final poll-level weighting

After pollster-level weights are determined, the function returns to the individual-poll level to compute the final average and the dispersion measure. Within each pollster, poll p receives a share of that pollster's total influence proportional to its within-pollster decay weight:

$$s_{pc}(t) = \frac{u_{pc}(t)}{\sum_{p' :\, j(p')=j(p)} u_{p'c}(t)}.$$

The code then applies an additional sponsorship adjustment

$$a_p \in (0, 1],$$

which downweights internal polls most heavily, downweights partisan public polls more mildly, and leaves ordinary nonpartisan public polls at full weight. The final unnormalized poll-level weight is therefore

$$\omega_{pc}(t) = \pi_{j(p)c}(t) \, s_{pc}(t) \, a_p.$$

Normalizing across all usable polls for choice c gives

$$\tilde{\omega}_{pc}(t) = \frac{\omega_{pc}(t)}{\sum_{p' \in \mathcal{P}_c(t)} \omega_{p'c}(t)}.$$

The reported polling average is then

$$\hat{y}_c(t) = \sum_{p \in \mathcal{P}_c(t)} \tilde{\omega}_{pc}(t)\, y_{pc}.$$

This formulation makes clear that the average is not a simple poll-level recency average. It is instead a nested weighting system. Polls are first pooled within pollster, pollsters are then weighted by the recency and quality of their latest usable contribution, and finally individual polls reclaim only their share of that pollster's total influence. The result is an average that is robust to repeated releases by the same organization, yet still allows multiple waves from that organization to contribute updated information.

2.2.3 Effective number of polls

For each choice c, the function also computes

$$\mathrm{ENOP}_c(t) = \sum_j g_{jc}(t),$$

that is, the sum of the *raw* pollster-level weights before normalization.

ENOP is not a literal count of polls, nor is it merely the number of polls with positive weight. Rather, it is an information-content measure. A recent high-quality pollster contributes close to one full unit, while an older or lower-quality pollster contributes only a small fraction of a unit. Because ENOP is computed from pollster-level raw weights rather than from the final normalized poll weights, ten old or low-quality pollsters do not increase ENOP, and multiple releases from the same pollster do not mechanically create higher ENOP. ENOP is best interpreted as the amount of information in the polling average, expressed in units roughly equivalent to the number of high-quality polls that would need to be released today to provide the same level of information.

2.2.4 Polling Average Rationalization

The polling average produced by this procedure is therefore shaped by four principles. First, newer information should matter more than older information. Second,

a poll's rate of decay should depend not only on its age but also on how deep the polling bench already is, so that thinly polled races preserve older information longer than heavily polled races. Third, distinct pollsters should contribute more independent information than repeated releases from the same pollster. Fourth, polling from stronger pollsters and from less strategically motivated sponsors should receive greater influence. The ENOP statistic is the scalar summary of how much effective polling information survives those adjustments, and it is precisely this quantity that governs how heavily the point-estimate layer leans on polling relative to fundamentals in Stage II.

2.3 Stage II: Poll Integration

The fundamentals are combined into a weighted average with the current polling average margin. The model computes this weighted average using the latest published average margin, denoted $\bar{p}_i$, and the effective number of polls, denoted n_i^{ENOP}. $\bar{p}_i$ is always put on the same signed margin scale as the fundamentals forecast.

Polling influence is governed by a weight function that depends on both time to election and ENOP. Let d_i denote days until the general election. The code first applies a diminishing-returns transformation to ENOP,

$$u_i = \frac{\min\{\sqrt{n_i^{\text{ENOP}}},\, \bar{n}\}}{\bar{n}},$$

where $\bar{n}$ is a fixed cap. It then multiplies this by a smooth recency term of the form

$$r(d_i) = a_0 + a_1 e^{-d_i/\tau},$$

and by an overall poll-share cap $\bar{w}$, producing a raw polling weight

$$\tilde{w}_i^{\text{poll}} = \bar{w}\, u_i\, r(d_i).$$

This construction has three important implications. First, poll influence increases as election day approaches. Second, the marginal contribution of additional polling information is positive and decreasing, reflecting a concave transformation of ENOP, with polling weight proportional to $\sqrt{\text{ENOP}}$ rather than ENOP itself. Third, no amount of polling is allowed to fully dominate and remove fundamentals from the model.

Governor races receive an additional transformation. Because the raw weight lies in $(0, 1)$, the code raises it to an exponent $\gamma_G \in (0, 1)$:

$$w_i^{\text{poll}} = \begin{cases} (\tilde{w}_i^{\text{poll}})^{\gamma_G}, & o(i) = \text{Governor}, \\ \tilde{w}_i^{\text{poll}}, & \text{otherwise}. \end{cases}$$

Since $\gamma_G < 1$, this increases the effective polling share for governors relative to equally polled House or Senate races. This reflects the fact that fundamentals exhibit lower predictive accuracy for gubernatorial outcomes than for federal races in backtesting, increasing the relative value of polling information.

The polling-updated structural estimate is therefore

$$m_i^{\mathrm{FP}} = \begin{cases} m_i^{\mathrm{F}}, & \text{if polling is unavailable,} \\ (1 - w_i^{\mathrm{poll}}) m_i^{\mathrm{F}} + w_i^{\mathrm{poll}} \bar{p}_i, & \text{otherwise.} \end{cases}$$

2.4 Stage III: Expert Rating Integration

After polling is blended in, the script applies a soft expert rating adjustment using Cook Political Report, Inside Elections, and Sabato's Crystal Ball. Experts have strong historical track records in election forecasting and provide valuable additional information. These ratings are not treated as direct numerical forecasts. Instead, each categorical label is interpreted as an interval on the margin scale. Let I_{ij} denote the interval associated with outlet $j \in \{\mathrm{Cook}, \mathrm{Inside}, \mathrm{Sabato}\}$ for race i. These intervals correspond to the usual ordered classes from *Solid R* through *Solid D*.

Define the interval-projection operator

$$\Pi_I(x) = \begin{cases} L_I, & x < L_I, \\ x, & x \in I, \\ U_I, & x > U_I, \end{cases}$$

with the obvious interpretation for open-ended extreme categories. In words, if the current continuous estimate already lies inside the outlet's category, that outlet contributes no adjustment. If it lies outside, the outlet pulls the estimate only toward the *nearest boundary* of its implied interval, not all the way to the center of the category and not to an outlet-specific numerical point estimate.

Let $\eta_j(t)$ denote the time-varying influence weight for outlet j, where t indexes calendar time within the campaign. $\eta_j(t)$ is strictly increasing in t, reflecting greater weight placed on expert ratings as election day approaches. The code makes these outlet weights rise linearly over the cycle. Let

$$\kappa_i = \begin{cases} \kappa_G > 1, & o(i) = \text{Governor}, \\ 1, & \text{otherwise}, \end{cases}$$

so that expert ratings have greater influence in gubernatorial races, where fundamentals-based models exhibit lower predictive accuracy. The outlet-specific adjustment is

$$A_{ij} = \eta_j(t)\,\kappa_i\Big(\Pi_{I_{ij}}(m_i^{\mathrm{FP}}) - m_i^{\mathrm{FP}}\Big).$$

The expert-adjusted estimate is then

$$m_i^{\mathrm{FPE}} = m_i^{\mathrm{FP}} + \sum_j A_{ij}.$$

Two features of this step are worth emphasizing. First, expert ratings act as a *regularizer* or boundary correction, not as a replacement forecast. Second, because each outlet compares its category to the same baseline quantity m_i^{fp}, the three adjustments are additive relative to a common reference point.

The time-varying weights $\eta_j(t)$ are designed to reflect the empirical observation that expert ratings become more accurate as election day approaches. In backtesting, these ratings tend to function as lagging indicators of the political environment, often underreacting to early-cycle shifts and converging toward realized outcomes later in the campaign. As a result, their influence is intentionally limited in the spring and summer, when structural and polling-based signals are more informative, and increased gradually over time as expert assessments incorporate more complete information about the state of each race.

2.5 Stage IV: Kalshi Liquidity-Weighted Probability Aggregation

The final information source is Kalshi. For the point-estimate calculation itself, however, the operative market quantities are drawn from the market-level orderbook data.

For each contract c within race i, let

$$p_{ic}^{\mathrm{trade}}$$

denote the last-trade price, and let

$$p_{ic}^{\mathrm{bid}}, \qquad p_{ic}^{\mathrm{ask}}$$

denote the best current bid and ask. If both sides of the book are present, the midpoint is

$$p_{ic}^{\mathrm{mid}} = \frac{p_{ic}^{\mathrm{bid}} + p_{ic}^{\mathrm{ask}}}{2}.$$

The code also records trading volume V_{ic} and open interest O_{ic}, and defines the bid-ask spread

$$s_{ic} = p_{ic}^{\text{ask}} - p_{ic}^{\text{bid}}.$$

Liquidity is summarized by a within-office-type composite score. Specifically, the code computes percentile ranks, separately for House, Senate, and Governor markets, for

$$V_{ic}, \qquad s_{ic}^{-1}, \qquad O_{ic},$$

and combines them as

$$\ell_{ic} = \omega_V \rho_V(V_{ic}) + \omega_S \rho_S(s_{ic}^{-1}) + \omega_O \rho_O(O_{ic}),$$

where the weights form a fixed convex combination with spread tightness receiving the greatest emphasis. Ranking within office type prevents low-profile House markets from being penalized merely because raw Senate-market volumes are mechanically larger.

The trade-versus-midpoint blending weight is then a linear function of liquidity:

$$\lambda_{ic} = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min})\ell_{ic}.$$

So, more liquid contracts rely more heavily on the last trade, while thinner markets rely more heavily on the midpoint.

The model handles three edge cases.

First, if a contract has no usable two-sided order book, it is discarded. Second, if the contract has no last-trade evidence and essentially no trading activity, it is discarded. Third, if the last trade lies outside the current bid-ask interval, the trade is treated as stale; in that case the model resets the blending share to its minimum level and replaces the stale trade price by the nearest live quote.

Thus, the contract-level probability used downstream is

$$p_{ic} = \begin{cases} \text{missing}, & \text{if the market is unusable,} \\ \lambda_{ic} p_{ic}^{\text{trade}} + (1 - \lambda_{ic}) p_{ic}^{\text{mid}}, & \text{if the market is usable and not stale,} \\ \lambda_{\min} \tilde{p}_{ic}^{\text{live}} + (1 - \lambda_{\min}) p_{ic}^{\text{mid}}, & \text{if the last trade is stale,} \end{cases}$$

where $\tilde{p}_{ic}^{\text{live}}$ denotes the nearest active quote.

Within each race, the usable contract probabilities are normalized to sum to one:

$$p_{ic}^{\text{norm}} = \frac{p_{ic}}{\sum_{c' \in i} p_{ic'}}.$$

The race-level Kalshi signal retained by the code is the Republican win probability

$$q_i,$$

obtained either directly from the Republican contract or, if necessary, by subtraction from the non-Republican side after remapping. The model also stores an average liquidity score by race.

2.6 Stage V: Probability-to-Margin

The model does not combine Kalshi probabilities directly with margin forecasts. Instead, it translates the Republican win probability into a Republican two-party margin estimate.

First, degenerate probabilities of exactly 0 or 1 are removed, and remaining probabilities are clipped slightly away from the endpoints to avoid infinite inverse-normal values. Second, the script applies a mild sharpening transform,

$$q_i^\star = \frac{q_i^\alpha}{q_i^\alpha + (1-q_i)^\alpha}, \qquad \alpha = 1+\varepsilon, \quad \varepsilon > 0 \text{ small},$$

which pushes favorites somewhat farther above 0.5 and longshots somewhat farther below 0.5. This is a light favorite-longshot correction based on our analysis of past prediction market calibration.

The transformed probability is then mapped to a Gaussian latent score

$$z_i = \Phi^{-1}(q_i^\star).$$

Next, the code constructs an office-specific margin scale

$$\sigma_i = s(d_i)\, r_{o(i)},$$

where $s(d_i)$ is a smooth function of time to election with the form

$$s(d_i) = s_0 + s_1 d_i^\nu, \qquad 0 < \nu < 1,$$

and $r_{o(i)}$ is an office-specific multiplier. This implies that the same market probability corresponds to a somewhat wider implied vote margin earlier in the cycle than later.

The raw Kalshi-implied margin is

$$\tilde{m}_i^{\text{Kalshi}} = 200\, \sigma_i z_i.$$

Finally, the implementation applies a mild nonlinear stretch to this raw margin:

$$m_i^{\text{Kalshi}} = \text{sign}\big(\tilde{m}_i^{\text{Kalshi}}\big)\,\big|\tilde{m}_i^{\text{Kalshi}}\big|^{\delta}, \qquad \delta > 1.$$

This last step preserves sign but modestly amplifies larger market-implied leads relative to very small ones based on observed market patterns. In a setting where highly liquid prediction markets exist for vote margins, Stage V could be removed from the forecasting framework, which would be preferable. Mapping continuous electoral outcomes to binary win probabilities necessarily discards information about the underlying distribution of market expectations, an unavoidable limitation under current market constraints.

2.7 Stage VI: Dynamic Weighting

Kalshi is not assigned a fixed global weight. Instead, its influence declines as the expert-adjusted structural estimate becomes less competitive. This reflects the fact that probabilities are most informative on the logit scale near 0.5, where uncertainty is highest, and less informative as outcomes approach certainty.

Let

$$x_i = \big|m_i^{\text{FPE}}\big|.$$

The code first defines a base Kalshi share

$$\bar{w}_i^{\text{Kalshi}} = c_0 + c_1 e^{-c_2 x_i},$$

which is largest near tossups and declines as the structural margin moves away from zero.

This base weight is then rescaled according to polling availability. Let w_i^{poll} denote the polling weight from the earlier step. The code uses a multiplier

$$g_i = \begin{cases} g_{\max}, & \text{if polling is unavailable,} \\ 1, & \text{if } w_i^{\text{poll}} \geq \bar{w}_{\text{thresh}}, \\ g_{\max} - (g_{\max} - 1)\dfrac{w_i^{\text{poll}}}{\bar{w}_{\text{thresh}}}, & \text{if } 0 \leq w_i^{\text{poll}} < \bar{w}_{\text{thresh}}. \end{cases}$$

Thus, Kalshi receives more influence when polling is sparse or unavailable, and less additional influence when polling is already informative. This reflects the fact that prediction markets incorporate race-specific information and dynamics not captured by fundamentals, and are particularly valuable when polling data are limited.

The model includes an office multiplier

$$h_{o(i)},$$

with the largest factor for governors, an intermediate factor for Senate races, and the smallest factor for House races. Gubernatorial races exhibit weaker fundamentals-based predictive performance, so greater weight is placed on supplementary information sources in those contests. In contrast, House races generally have lower overall market volume than Senate races, but this is already accounted for by normalizing volume measures within race type. The realized Kalshi share is therefore

$$w_i^{\text{Kalshi}} = \min\left\{\bar{w}_{\max}^{\text{Kalshi}},\ \bar{w}_i^{\text{Kalshi}} g_i h_{o(i)}\right\},$$

where $0 < \bar{w}_{\max}^{\text{Kalshi}} < 1$ is a fixed cap that prevents prediction markets from replacing the fundementals too much. The structural share is the complement,

$$w_i^{\text{struct}} = 1 - w_i^{\text{Kalshi}}.$$

2.8 Final point estimate

The final point estimate is

$$m_i^{\text{PE}} = \begin{cases} m_i^{\text{FPE}}, & \text{if Kalshi is unavailable,} \\ w_i^{\text{struct}} m_i^{\text{FPE}} + w_i^{\text{Kalshi}} m_i^{\text{Kalshi}}, & \text{otherwise.} \end{cases}$$

This is the quantity written out as the point estimate for race i.

Conceptually, the entire system can therefore be read as a layered update:

$$m_i^{\text{F}} \longrightarrow m_i^{\text{FP}} \longrightarrow m_i^{\text{FPE}} \longrightarrow m_i^{\text{PE}}.$$

The first transition adds polling in proportion to information content and time to election. The second regularizes the result against categorical expert ranges. The third allows prediction markets to move the estimate, especially in competitive, lightly polled, and higher-profile races. m_i^{PE} is the Complete forecast, while m_i^{FP} is the Legacy forecast.

Why do we release both a Legacy model and a Complete model? The primary reason is that the two models serve different purposes. The Legacy model is designed to be a more traditional election forecasting model. It relies on fundamentals and polling, but does not incorporate outside sources such as expert ratings or prediction markets. This makes it easier to compare directly against other election models, such

as those historically produced by Nate Silver, G. Elliott Morris, The Economist, and Decision Desk HQ. For most apples-to-apples comparisons with other forecasters, the Legacy model is likely the more appropriate benchmark.

There is also an important endogeneity concern with prediction markets. If prediction markets are included in our only forecast, and market participants use our forecast to inform their trades, then a feedback loop can emerge: our model could move the market, and the market could then move our model. Maintaining a Legacy model that is fully independent of prediction market activity helps reduce this concern and provides traders, analysts, and readers with a forecast that is exogenous to market prices.

At the same time, our goal is not only to build a traditional model, but also to build the most accurate forecasting system we can. That is the purpose of the Complete model. The Complete model incorporates fundamentals, polling, expert ratings, and prediction markets. While the inclusion of expert ratings and prediction markets is less traditional, we believe these sources may contain meaningful predictive information.

Expert ratings, particularly those from the Cook Political Report, Sabato's Crystal Ball, and Inside Elections, have a strong track record in election forecasting. Their value also tends to increase as Election Day approaches, as analysts incorporate more candidate-specific information, fundraising data, district-level context, and campaign developments into their ratings. For this reason, expert ratings receive greater weight as the cycle progresses.

Finally, we believe our model is among the first election forecasting models to incorporate prediction market data directly, and we want to explain why we chose to do so. We do not yet have enough backtesting data to conclude that the modern prediction market landscape is categorically "better" than polling. We also reject that framing to some degree, since polling is itself one of the central inputs that shapes prediction market prices. However, the data we have examined are promising.

For pre-2024 elections, we tested PredictIt data; for 2024, we tested Kalshi data. Across 79 Senate and presidential races from 2016 to 2022, we converted election-morning market prices into an implied predicted margin using the transformation (this is a rough approximation of the process described above):

$$\widehat{\text{margin}} = \begin{cases} 4.5 \cdot \text{logit}(p), & \text{for statewide races,} \\ 6 \cdot \text{logit}(p), & \text{for U.S. House races.} \end{cases}$$

where p is the implied Republican win probability, calculated as the Republican contract price divided by the combined Democratic and Republican contract prices.

This market-implied margin produced a lower RMSE against the actual election margin than polling averages alone (Appendix Figure 1).

Appendix Figure 3 provides another way of evaluating the relationship. In a linear regression of actual margin on both the PredictIt-implied margin and the polling-based margin, with the intercept constrained to zero, only the PredictIt-implied margin had a statistically significant coefficient. The polling-based margin did not. This is not sufficient evidence to conclude that prediction markets are superior to polling, but it does suggest that market prices contained meaningful predictive information beyond what was captured by polling averages alone.

In 2024, Kalshi performed substantially better than polling averages in statewide presidential and Senate races when using the same transformation from order book pricing to predicted margin. However, we are cautious about overinterpreting that result. Kalshi was consistently more favorable to Republicans than polling across the board, and 2024 was a cycle with a pro-Republican polling error. Because statewide errors are highly correlated across races, this makes it difficult to distinguish genuine predictive superiority from a cycle-specific alignment between market prices and the direction of polling error.

The 2024 House data provides a somewhat cleaner test. Kalshi had 23 House races with sufficient market volume where Race to the White House also had a polling average. Because House races generally have a less correlated error structure than statewide races, we are more comfortable analyzing these results in greater detail. In this subset, the Kalshi-implied model produced slightly lower error than polling averages (Appendix Figure 4).

Overall, prediction markets have a promising recent track record and appear to contain useful information for election forecasting. As discussed above, we believe prediction markets are especially valuable in more competitive races, where order book prices are more expressive; in races with limited polling, particularly U.S. House contests; and in gubernatorial races, where fundamentals-based models tend to perform far worse than they do for federal races.

At the same time, the available sample remains limited, and more data is needed before we can make strong claims about their precise predictive value. For this reason, we incorporate prediction markets as one input in the Complete model while preserving a Legacy model that excludes them. After the 2026 cycle, we will be able to compare the performance of the Legacy and Complete models directly, allowing us to better evaluate how much predictive value prediction markets add when controlling for fundamentals, polling, and expert ratings. The existence of liquid Kalshi markets for all federal races in 2026 should substantially improve the quality and scope of future testing.

3 Correlated Simulations

A central problem in election forecasting is that forecast errors are not independent across contests. If the model underestimates Republican strength in one competitive Senate race, it is more likely than not that it is also understating Republican strength in other races exposed to the same national environment, the same regional environment, or the same state-specific political conditions.

For that reason, simulating every race independently would produce implausibly thin tails and badly understate the probability of coherent multi-race scenarios. At the same time, directly estimating a separate pairwise correlation for every pair of races is neither statistically stable nor substantively appealing in a cycle with relatively few recent elections from which to learn. A guiding principle of the correlated simulation literature is therefore to estimate a small number of correlation levels well rather than thousands of pairwise correlations poorly (Louit et al., 2022). We extend this into a more continuous direction. Rather than partitioning races into a small number of discrete clusters and assigning correlations based on shared bucket membership, we construct a continuous similarity score for every pair of races using demographic, political, socioeconomic, and geographic features. We then map those similarities into an office-pair-specific correlation matrix.

Let $i = 1, \ldots, N$ index forecasted races. For each race i, let

$$m_i$$

denote the Republican two-party point estimate in percentage points and let

$$\mu_i = \frac{50 + m_i/2}{100}$$

denote the corresponding Republican two-party vote share on the unit interval. The simulation engine seeks to generate a joint distribution for the vector of vote shares

$$Y = (Y_1, \ldots, Y_N)^\top$$

such that

$$Y \sim \mathcal{N}(\mu, \Sigma),$$

where $\mu = (\mu_1, \ldots, \mu_N)^\top$ and Σ is a covariance matrix encoding both marginal uncertainty and cross-race dependence.

3.1 Similarity construction from race-level features

The first step is to construct a continuous similarity matrix across all races to be forecast. The code reads in a race-level feature table, denoted here by

$$X_i = (X_{i1}, \ldots, X_{ip}),$$

where the default feature set includes demographic composition, socioeconomic structure, partisan baseline, and geographic location. In the current implementation, these variables include race, education, partisanship, and geography. We assign a fixed importance weight to each feature, with some dimensions deliberately given more influence than others in the similarity calculation. Let $w_k > 0$ denote the feature weight for dimension k.

Each feature is first standardized across races,

$$Z_{ik} = \frac{X_{ik} - \bar{X}_k}{s_k},$$

and then reweighted as

$$\tilde{Z}_{ik} = \sqrt{w_k}\, Z_{ik}.$$

The use of $\sqrt{w_k}$ means that Euclidean distance in the transformed space is equivalent to a weighted Euclidean distance in the original standardized space:

$$d_{ij} = \left(\sum_{k=1}^{p} w_k (Z_{ik} - Z_{jk})^2 \right)^{1/2}.$$

Thus, two House districts that are more similar will have a smaller distance than two districts that differ sharply on the chosen dimensions, even before any office-type rules are applied.

The code then converts distances into a raw similarity score on $[0, 1]$ by an affine rescaling:

$$s_{ij}^{\text{raw}} = 1 - \frac{d_{ij} - d_{\min}}{d_{\max} - d_{\min}},$$

where $d_{\min}$ and $d_{\max}$ are the minimum and maximum pairwise distances observed in the current cycle's race set. By construction, larger values indicate greater similarity.

This raw similarity is not used directly. Instead, the code rescales the off-diagonal similarities so that the distribution of pairwise resemblance has a controlled center and dispersion. Writing $\mathcal{S} = \{s_{ij}^{\text{raw}} : i < j\}$ for the set of off-diagonal raw similarities, with empirical mean $\bar{s}$ and standard deviation σ_s, the code computes

$$s_{ij}^{*} = \text{clip}_{[0,1]} \left(\mu_s + \kappa_s \frac{s_{ij}^{\text{raw}} - \bar{s}}{\sigma_s} \right), \qquad i \neq j,$$

for fixed tuning constants μ_s and κ_s, and sets $s^*_{ii} = 1$. This step is important because the similarity matrix is not meant to reproduce raw geometric distances literally. Rather, it is intended to provide a stable and cycle-specific ordering of race-pair resemblance that can then be translated into plausible electoral correlations.

3.2 From similarity to pairwise correlation

The next step maps continuous similarity into a correlation matrix. Let $t_i \in \{H, S, G\}$ denote the office type of race i and let u_i denote its state. For each office-pair type (a, b), the code specifies a low anchor $\underline{\rho}_{ab}$ and a high anchor $\overline{\rho}_{ab}$, with

$$0 < \underline{\rho}_{ab} < \overline{\rho}_{ab} < 1.$$

These anchors differ across pair types. A House–House pair is not treated the same as a Senate–Senate pair, a House–Senate pair, or a Governor–Senate pair. Importantly, the code does *not* attempt to learn a unique correlation for every one of the $\binom{N}{2}$ race pairs. Instead, it estimates correlation more parsimoniously by saying, in effect, that a given pair belongs to one broad office-pair class, and that within that class the exact correlation should move smoothly between a lower and upper bound as the two races become more or less similar.

If the pair type of races (i, j) is (t_i, t_j), the baseline correlation is

$$\rho^{(0)}_{ij} = \underline{\rho}_{t_i t_j} + \left(\overline{\rho}_{t_i t_j} - \underline{\rho}_{t_i t_j}\right) s^*_{ij}.$$

This is the key continuous extension relative to simpler clustered models. In a binary cluster system, two races would often receive one correlation if they are "in the same region" and another if they are not. Here, two races can be moderately similar, very similar, or barely similar, and that distinction matters continuously.

The code then applies an additional same-state boost. Let

$$\delta_{ij} = \mathbf{1}\{u_i = u_j\}$$

indicate whether the two races are from the same state. Then the pre-projection correlation takes the form

$$\tilde{\rho}_{ij} = \rho^{(0)}_{ij} + b_{t_i t_j}\delta_{ij}, \qquad i \neq j,$$

where $b_{t_i t_j} \geq 0$ is an office-pair-specific within-state increment. This feature is quite important in a midterm model. In a presidential year, there is a national race at the top of the ballot that strongly ties together down-ballot outcomes and absorbs

much of the common environment. In a midterm, there is more room for genuinely state-level political climate to matter: gubernatorial candidate quality, state-specific incumbency patterns, local partisan brands, issue environments, differential turnout, and media markets can create a state or regional shock that simultaneously moves a Senate race, one or more House races, and the gubernatorial contest in the same direction. For that reason, the code intentionally gives extra covariance to same-state race pairs, especially when the pair contains statewide contests.

After these pairwise correlations are constructed, the code clips each off-diagonal element into a safe interval strictly inside $[-1, 1]$ and places ones on the diagonal. Letting $\tilde{R}$ denote the resulting symmetric matrix, the code then enforces positive semidefiniteness by eigenvalue clipping. If

$$\tilde{R} = Q\Lambda Q^\top$$

is the eigendecomposition of $\tilde{R}$, the code replaces all eigenvalues below a small positive threshold ε by ε, yielding

$$R^+ = Q\Lambda^+ Q^\top.$$

Because this perturbation can move the diagonal away from one, the matrix is then renormalized:

$$R_{ij} = \frac{R^+_{ij}}{\sqrt{R^+_{ii} R^+_{jj}}},$$

after which the diagonal is reset to one exactly. This produces the final correlation matrix

$$R = (\rho_{ij}),$$

which is guaranteed to be usable in downstream simulation even after same-state boosts and heterogeneous pairwise interpolation.

3.3 Race-level marginal uncertainty

Correlated simulations require not only a correlation matrix but also a standard deviation for each race. The code therefore builds a race-specific uncertainty vector

$$\sigma = (\sigma_1, \ldots, \sigma_N)^\top.$$

The base uncertainty schedule begins with a House-level standard deviation that declines smoothly as election day approaches. Writing d for days remaining until the general election, the code uses a function of the form

$$\sigma_H(d) = a + b\, d^\nu, \qquad 0 < \nu < 1,$$

with fixed constants a, b, ν. Uncertainty shrinks as election day approaches, and the decline is gradual and concave down with respect to time.

This House baseline is then multiplied by an office-specific factor. Gubernatorial races are treated as more volatile than House races, Senate races as somewhat less volatile.

Given R and σ, the total covariance matrix is constructed as usual:

$$\Sigma = DRD, \qquad D = \mathrm{diag}(\sigma_1, \ldots, \sigma_N).$$

3.4 National-plus-structured covariance decomposition

We decompose the covariance into a rank-one national component plus a structured residual component:

$$\Sigma = \Sigma_{\mathrm{nat}} + \Sigma_{\mathrm{struct}}.$$

This is conceptually very close to the correlated simulation logic used in prior election models (Louit et al., 2022), where outcomes are driven by national, regional, and unit-level components. The present code, however, uses a more flexible variant. Instead of requiring the structured piece to arise from a small number of discrete clusters, it allows the structured covariance to inherit the entire continuous similarity matrix described above.

The national component is parameterized as

$$\Sigma_{\mathrm{nat}} = c\,\sigma\sigma^{\top},$$

where σ is the vector of marginal standard deviations. This is a rank-one matrix, and it implies that the national component contributes the *same correlation* c to every pair of races:

$$\mathrm{Corr}_{\mathrm{nat}}(Y_i, Y_j) = c.$$

To preserve positive semidefiniteness of the residual component, the code computes the largest admissible value of c such that

$$\Sigma_{\mathrm{struct}} = \Sigma - c\,\sigma\sigma^{\top}$$

remains positive semidefinite. When Σ is positive definite, the maximal admissible value is

$$c_{\max} = \frac{1}{\sigma^{\top}\Sigma^{-1}\sigma}.$$

The code then chooses

$$c = \eta c_{\max}, \qquad 0 \le \eta \le 1,$$

for a fixed tuning parameter η. It extracts some fraction of the maximum possible common national factor while leaving the remainder inside the structured residual covariance.

This is a pure reparameterization of the same total covariance matrix, not a change to the implied joint distribution. So,

$$\Sigma_{\text{nat}} + \Sigma_{\text{struct}} = \Sigma.$$

Thus, final simulated standard deviations and correlations are unchanged. The benefit is interpretive and computational: one can now think of each simulation draw as the sum of a national shock shared by all races and a structured shock carrying the more nuanced continuous cross-race dependence.

The factor loadings associated with the national shock are

$$\lambda_i = \sqrt{c}\,\sigma_i.$$

So if $Z_0 \sim \mathcal{N}(0,1)$ denotes the scalar national draw, the national contribution to race i is simply $\lambda_i Z_0$.

3.5 Monte Carlo simulation

Let L denote a matrix square root of the structured residual covariance matrix, so that

$$LL^\top = \Sigma_{\text{struct}}.$$

For each simulation draw $r = 1, \ldots, M$, the model generates

$$Z_0^{(r)} \sim \mathcal{N}(0,1), \qquad Z^{(r)} \sim \mathcal{N}(0, I_N),$$

independently, and forms

$$Y^{(r)} = \mu + \lambda Z_0^{(r)} + LZ^{(r)},$$

where $\lambda = (\lambda_1, \ldots, \lambda_N)^\top$. By construction,

$$\text{Var}(Y^{(r)}) = \Sigma_{\text{nat}} + \Sigma_{\text{struct}} = \Sigma,$$

so the simulation exactly reproduces the intended mean vector and covariance matrix.

3.6 Chamber Control and Race-Level Probabilities

Once the simulation matrix is constructed, each race's win probability is estimated by thresholding at 0.5:

$$\hat{\pi}_i = \frac{1}{M} \sum_{r=1}^{M} \mathbf{1}\{Y_i^{(r)} > 0.5\}.$$

These race-level probabilities are then aggregated into chamber control summaries. For Senate races, the code counts the number of simulated Republican wins and adds the fixed number of non-cycle Republican seats that are not on the ballot in 2026. For House and gubernatorial races, the code simply counts simulated Republican wins:

$$H^{(r)} = \sum_{i \in \mathcal{H}} \mathbf{1}\{Y_i^{(r)} > 0.5\}, \qquad S^{(r)} = S_0 + \sum_{i \in \mathcal{S}} \mathbf{1}\{Y_i^{(r)} > 0.5\}, \qquad G^{(r)} = \sum_{i \in \mathcal{G}} \mathbf{1}\{Y_i^{(r)} > 0.5\},$$

where $\mathcal{H}, \mathcal{S}, \mathcal{G}$ denote the House, Senate, and gubernatorial race sets and S_0 is the fixed baseline number of Republican Senate seats not up in 2026.

From these simulated seat totals the script computes control probabilities, average seat counts, and chamber-level intervals. At the race level, it also computes vote-share quantiles from the simulated draws and packages those for downstream display. Separately, it constructs a "most correlated races" table by taking, for each race i, the ten largest off-diagonal entries of the final correlation matrix R. This is a useful explanatory artifact for the frontend: it makes the correlation structure visible and interpretable rather than burying it entirely inside a simulation engine.

Comparing correlated simulations in the Legacy and Complete models, the covariance matrix is held fixed, while the mean vector varies depending on the forecast layer being used. This reflects the distinction between the correlation structure, which governs how races move together, and the point-estimate layer, which determines the baseline level of each race prior to the application of common shocks in each simulation.

3.7 Why this structure is substantively appealing

The logic of the model can be summarized as follows. First, cross-race dependence is real and must be modeled (Heidemanns et al., 2020; Isakov and Kuriwaki, 2020). Second, one should resist the temptation to estimate thousands of race-pair correlations directly from a tiny historical sample (Louit et al., 2022). Third, the omitted-variable critique of fundamentals forecasting implies that correlated structural misses are inevitable, especially when several races are exposed to the same broader environment

(Lauderdale and Linzer, 2015). The code aims to operationalize those three ideas in a way that is both parsimonious and realistic. It is parsimonious because it only estimates a small number of office-pair anchor schedules and then lets the rest of the matrix be generated mechanically. It is realistic because similarity is continuous rather than binary, allowing the model to say that two House races in demographically and politically similar settings should move together more than two House races that merely share the same chamber label.

This balance is especially valuable in a midterm setting. With no presidential race on the ballot, there is more room for genuinely state-level or region-specific environments to matter, and therefore more reason to allow same-state Senate, House, and gubernatorial contests to co-move strongly. The code reflects that belief directly. It does not assume that all within-state pairs are identical, nor does it collapse the country into a few rigid blocs. Instead, it combines office type, continuous similarity, and same-state reinforcement into a single positive-semidefinite covariance structure and then simulates from that structure in a transparent national-plus-structured form.

References

Heidemanns, M., Gelman, A., and Morris, G. E. (2020). An updated dynamic bayesian forecasting model for the US presidential election. *Harvard Data Science Review*, 2(4).

Isakov, M. and Kuriwaki, S. (2020). Towards principled unskewing: Viewing 2020 election polls through a corrective lens from 2016. *Harvard Data Science Review*, 2(4).

Lauderdale, B. E. and Linzer, D. (2015). Under-performing, over-performing, or just performing? the limitations of fundamentals-based presidential election forecasting. *International Journal of Forecasting*, 31(3):965–979.

Louit, S., Ram, M., Williams, K., Alduncin, A., McCaul, P., and Tranter, S. (2022). Psephological correlated simulation techniques with decision desk hq: For the 2022 midterms and beyond. *Harvard Data Science Review*, 4(4).

A Appendix Figures

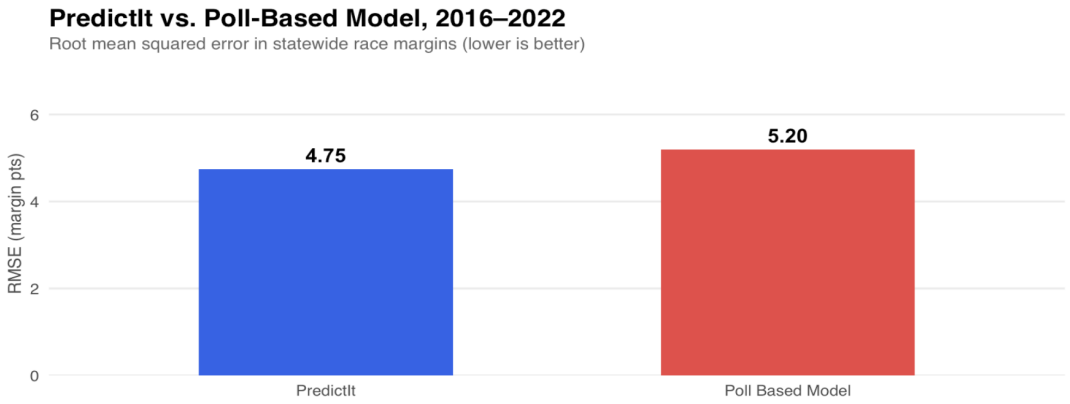


Figure 1: Prediction market-implied margins versus polling averages RMSE, 2016–2022.

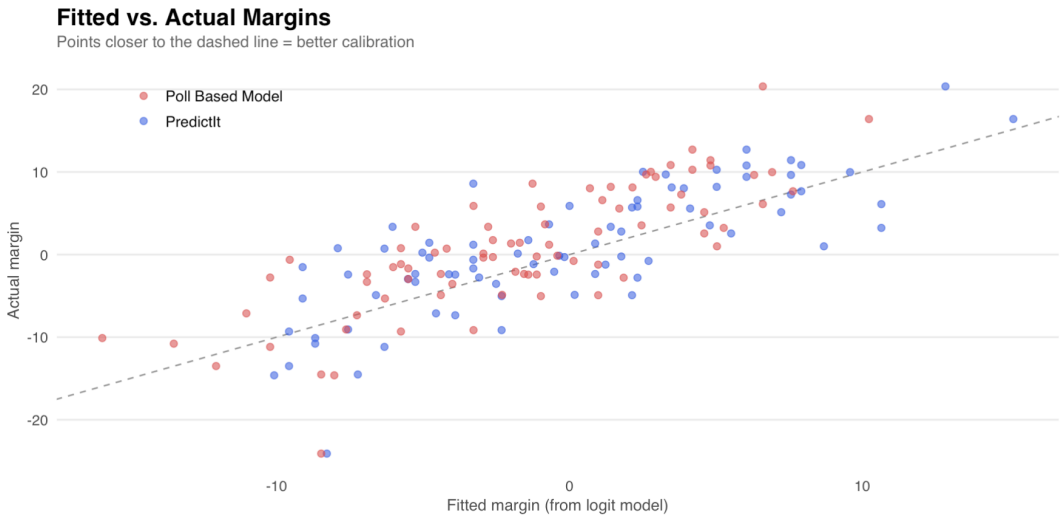


Figure 2: Fitted vs. Actual Margins for PMs and poll-based models.

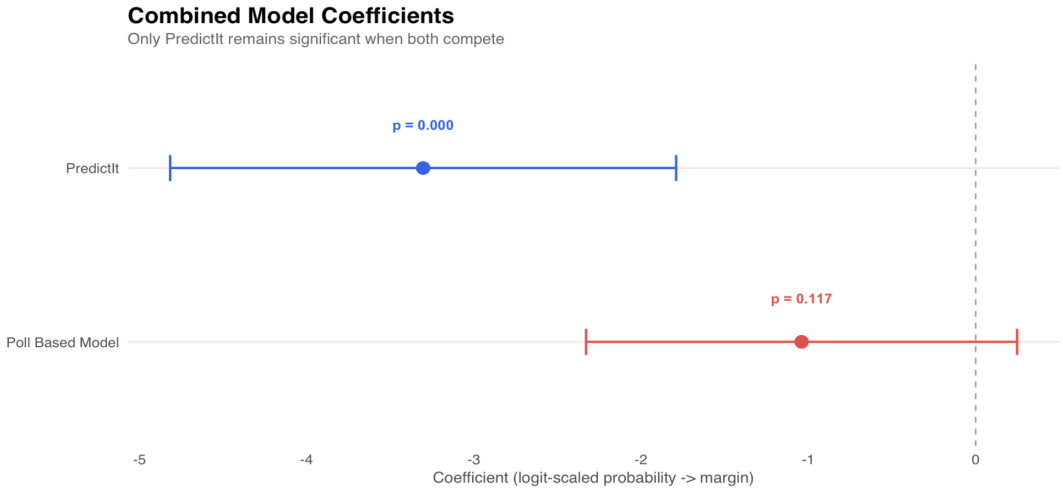


Figure 3: Regression comparison of actual margins on prediction market-implied and polling-based margins.

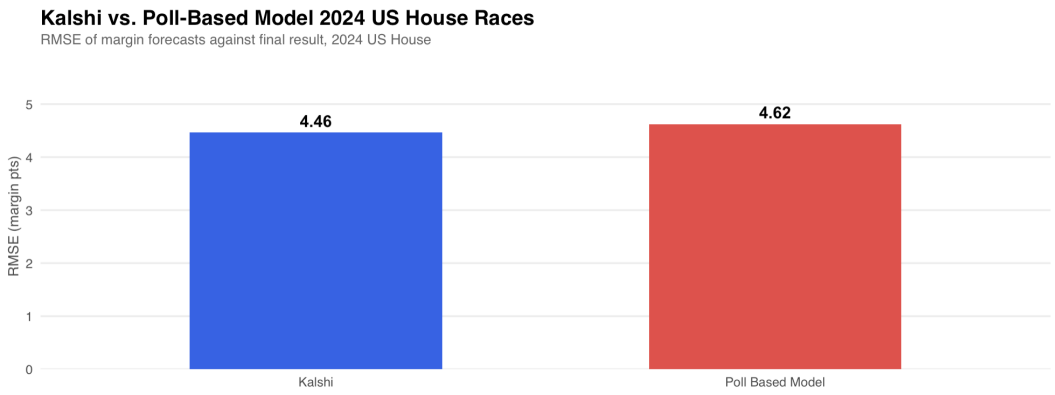


Figure 4: Kalshi-implied margins versus polling averages in 2024 U.S. House races.