ROBUST NONPARAMETRIC DISTRIBUTION FORECAST WITH BACKTEST-BASED BOOTSTRAP AND ADAPTIVE RESIDUAL SELECTION


*Amazon †University of Illinois at Urbana-Champaign

longsha@amazon.com

1. INTRODUCTION

Time series forecasting is crucial in many industrial applications and enables data-driven planning [1–3], such as making production capacity or inventory allocation decisions based on demand forecast [4]. Planners or optimization systems that consume the forecast often require the estimated distribution of the response variable (referred to as distribution forecast, or DF) instead of only the estimated mean/median (referred to as point forecast, or PF) to make informed and nuanced decisions. An accurate DF method should ideally factor in different sources of forecast uncertainty, including uncertainty associated with parameter estimates and model misspecification [2]. Furthermore, when deploying a DF method in industrial applications, there are other important practical considerations such as ease of adoption, latency, interpretability, and robustness to model misspecification. To this end, we propose a practical and robust DF framework that uses backtesting [5] to build a collection of predictive residuals and an adaptive residual selector to pick the relevant residuals for bootstrapping DF. The proposed framework incorporates different sources of forecast uncertainty by design, integrates well with an arbitrary PF model to produce DF, is robust to model misspecification, has negligible inference time latency, retains interpretability for model diagnostics, and achieves more accurate coverage than the current State-of-the-Art DF methods in our experiments.

The contributions of this paper are as follows:

1. We propose a flexible plug-and-play framework that can extend an arbitrary PF model to produce DF.
2. We extend the existing approach of bootstrapping predictive residuals [6, 7] by using backtest and covariate sampling to improve efficiency and account for uncertainty in input covariates.
3. We propose an adaptive residual selector, which relaxes the independence between residuals and covariates assumption and boosts model performance.
4. We propose a new formula on how bootstrapped residuals are applied during forecasting, which scales the residuals w.r.t. the PF.
5. Lastly, we empirically evaluate the performance of various DF approaches on our in-house product sales data and M4-hourly competition data. The proposed DF approach reduces the Absolute Coverage Error by more than 63% compared to the classic bootstrap approaches and by 2% – 32% compared to a variety of State-of-the-Art deep learning approaches.

2. RELATED WORK

The existing approaches for DF include the following categories: 1. Using models that make parametric distribution assumptions around the response variable and estimate the associated parameters, such as state space models [4, 10–12] and estimating model uncertainty through posterior distributions in a Bayesian setting [13–18]; 2. Using models that explicitly minimize quantile loss and generate quantile forecast, such as Quantile Gradient Boosting [19], MQ-R(C)NN [20], and Temporal Fusion Transformers [21]; and 3. Using variations of bootstrap methods that sample residuals to generate multiple bootstrap forecasts and then compute sample quantiles of the bootstrap forecasts [6, 7, 22–25]. The bootstrap methods have the practical advantage of being able to integrate with an arbitrary PF model to obtain DF, but the classic bootstrap methods are usually designed under strong assumptions around the PF model and dataset, which might not work well with complex real-world data or modern machine learning PF models. The “delete-x” approach of bootstrapping predictive residuals [6, 7] by design has less assumptions on
the data and is more robust to the choice of PF model; however, it only computes one predictive residual for each model training, ignores the uncertainty in covariates, and assumes that the residuals are independent from the covariates and PF. Our proposed DF framework is a significant generalization of the “delete-x” approach and addresses each of the aforementioned limitations. Another closely related approach is the Level Set Forecaster [26], which is similar to the special case of our approach where the in-sample forecast is used for computing and selecting residuals.

3. METHOD

The proposed DF framework is composed of a backtester, a residual selector, and a PF model (Figure 1). To summarize how it works: During training: 1. Backtest [5] is performed on the training data with the PF model to build a collection of predictive residuals (Figure 2); for covariates that need to be estimated for future time points (e.g., future price of a product), their values can be sampled from estimated distributions during backtest to account for the uncertainty in covariates. 2. The residual selector is pre-specified or learned from the training data as a set of rules or separate machine learning model that selects the most relevant subset of predictive residuals given a future data point based on their meta information. 3. Lastly, the PF model is trained on the entire training data. During forecasting: 1. For each future data point of interest, the trained PF model generates the PF. 2. The residual selector selects a subset of residuals. 3. Lastly, random samples of the trained PF model generates the PF. 2. The residual selector is pre-specified or learned from the training data with the PF model to build a collection of the selected predictive residuals from backtest to estimate the distribution of the future predictive residuals and thus the distribution of predictive residuals from backtest to account for uncertainty in covariates. 3.2. Selecting Residuals

Common PF models typically assume that the residuals are i.i.d. and independent from the covariates and the PF itself [2]. However, such assumption doesn’t always hold in practice for the predictive residuals. E.g., the variance of residuals can increase as we forecast further into the future or as the magnitude of PF increases. To relax the commonly imposed independence assumption between residuals and covariates (or more generally any meta information which can include the PF or other variables not in the original covariates), an adaptive residual selector can be learned from the training data to select a subset of residuals based on the meta information of the predictive residuals from backtest and the future data point, \( \bar{g}(\mathcal{E}, \mathcal{M}, \mathcal{M}^{(\text{future})}) \), so that the selected residuals are conditionally i.i.d.. The residual selector should ideally be based on the meta information that has a non-negligible impact on the predictive residuals. We mention two options for learning the residual selector here: 1. Compute distance correlation (which can detect both linear and non-linear dependence) [27,28] between the predictive residuals from backtest and their corresponding meta information to identify variables with the highest distance correlation to the residuals. Then design rules (e.g., set simple thresholds) around these variables to select residuals that have a different distribution from the distribution of the entire collection of residuals, which can be verified by the Kolmogorov-Smirnov test [29]. Note that if the residual selector has no impact, the selected residuals should have the same distribution as the entire collection. 2. Fit a machine learning model, such as a regression decision tree, to predict residuals from their meta information, then apply the model to the meta information of future data points to select the corresponding residuals. The performance of this model can also be used to check dependence between meta information and residuals – if the residuals are already independent from the meta information pre-selection, then the model should perform poorly.

3.3. Bootstrapping

We describe two formulae of generating bootstrap forecasts, Backtest-Additive and Backtest-Multiplicative. They can be applied to either iterative or direct PF models (an iterative model recursively consumes the forecast from the previous time point to forecast the next, whereas a direct model generates forecast for a future time point directly from covariates [30]). For Backtest-Additive, to generate bootstrap forecasts for the next time point \( d_i + 1 \), after obtaining the PF \( \hat{Y}^{d_i+1} = f(\hat{Y}^{d_i}; X^{d_i}; X^{d_i+1}) \) and the selected predictive residuals from backtest \( \mathcal{G} = \bar{g}(\mathcal{E}, \mathcal{M}, \mathcal{M}^{(d_i+1)}) \), random samples are drawn from the selected residuals \( \varepsilon_b \in \mathcal{G} \) for
Both the backtest step and the residual selection step can be
= R
 endorse forecast (or response variable) during backtest,
computed by dividing the extracted residuals over their cor-
in the same way as Backtest-Additive, the error ratios are
Backtest-Multiplicative, which scales the residuals based on
response variable seen during backtest. Hence we also propose
Additive can degrade if the variance of residuals increases
novel in Backtest-Additive). The performance of Backtest-
als [6, 7] (while the backtest and residual selection steps are
ual to the PF. The formula for Backtest-Additive is similar
step and adding the sample quantiles of the selected resid-
for the next time point are given by sampling
PF model, bootstrap forecasts are recursively generated for
the test split is used to compute the predictive residuals.

Fig. 1: Overview of the proposed DF framework. The backtester generates a collection of predictive residuals; the residual selector selects a subset of residuals for each future data point; the bootstrapping step combines the PF and selected residuals to generate DF.

**Split 1:**
- data to train
- point forecast model
- time
- data to compute predictive residuals

**Split 2:**
- data to train
- point forecast model
- time
- data to compute predictive residuals

**Split 3:**
- data to train
- point forecast model
- time
- data to compute predictive residuals

\[ b = 1, 2, \ldots, B, \text{ then the bootstrap forecasts are given by} \]
\[ \hat{Y}_{i,b,\text{Add.}}^{d_i+1} = \hat{Y}_{i}^{d_i+1} + \varepsilon_b. \]
Quantile forecasts are obtained by taking sample quantiles of the bootstrap forecasts. Generalizing to arbitrary future time point \( d_i + k_i \), for an iterative PF model, bootstrap forecasts are recursively generated for the next time point until \( d_i + k_i \); for a direct PF model, the calculation remains the same as 1-step forecast with \( d_i + 1 \) replaced by \( d_i + k_i \). Note that for a direct PF model, quantile forecasts can be obtained by skipping the residual sampling step and adding the sample quantiles of the selected residuals to the PF. The formula for Backtest-Additive is similar to the existing approach to bootstrapping predictive residuals [6, 7] (while the backtest and residual selection steps are novel in Backtest-Additive). The performance of Backtest-Additive can degrade if the variance of residuals increases with the magnitude of the PF, or if the magnitude of the future PF is very different from the magnitude of the response variable seen during backtest. Hence we also propose Backtest-Multiplicative, which scales the residuals based on the PF: After obtaining the PF and the selected residuals in the same way as Backtest-Additive, the error ratios are computed by dividing the extracted residuals over their corresponding forecast (or response variable) during backtest,

\[ R = \{ \varepsilon_{i,j}/Y_{i,j} | \varepsilon_{i,j} \in \mathcal{E} \}; \]
then the bootstrap forecasts for the next time point are given by sampling \( r_b \in R \) and
\[ \hat{Y}_{i,b,\text{Multi.}}^{d_i+1} = \hat{Y}_{i}^{d_i+1} \cdot (1 + r_b). \]
The rest remains the same.

**3.4. Practical Considerations**
Both the backtest step and the residual selection step can be efficiently parallelized across multiple CPU’s/GPU’s. The backtest step requires multiple model training, but it is more efficient than the previous “delete-x” approach of bootstrapping predictive residuals [6, 7] and can be done offline at a lower frequency than updating the PF model. The only computational overhead during inference time is the (optional) residual selection given the PF, so the additional latency of obtaining DF is negligible. Furthermore, once a residual collection from backtest is built, quantile forecast for any target quantile can be generated without re-running backtest or re-training the PF model, whereas DF methods that explicitly minimize quantile loss typically require the target quantile to be specified before model training. The backtest-based methods are also relatively interpretable: They retain the interpretability of the underlying PF model if the PF model is interpretable; even with a less interpretable PF model, one can check the predictive residual distribution and model performance on the test split (and model coefficients if applicable) during the backtest step to help identify which data points or covariates tend to contribute to large predictive residuals and whether the model has systematic bias during out-of-sample forecasting. The choices of bootstrap formula (Backtest-Additive vs Backtest-Multiplicative), denominator of error ratios (backtest forecast vs observed response variable), residual selector variation, and PF model can be tuned as hyperparameters.

**4. EXPERIMENTS**
We conduct experiments on two real-world time-series datasets: an in-house product sales dataset and the M4-hourly competition dataset [8, 9]. The product sales dataset consists of daily sales of 76 products between 01/01/2017 and 01/10/2021 and 147 covariates capturing information on pricing, supply constraints, trend, seasonality, special events, and product attributes. The standard Absolute Coverage Error (ACE) is used to evaluate the DF performance: The coverage (CO) of quantile forecast \( \hat{Y}_{i}^{\tau} \) for target quantile \( \tau \) over the test set \( D_{\text{test}} \) is defined as

\[ \text{CO}(D_{\text{test}}; \tau) = \frac{1}{|D_{\text{test}}|} \sum_{i \in D_{\text{test}}} I\{ Y_{i}^{\tau} \leq \hat{Y}_{i}^{\tau} \}; \]

and ACE is defined as

\[ \text{ACE}(D_{\text{test}}; \tau) = \lvert \text{CO}(D_{\text{test}}; \tau) - \tau \rvert. \]
(We also track other metrics such as quantile loss, weighted quantile loss, and weighted prediction interval width; the conclusions from different metrics are overall consistent.) A 100-fold backtest is used for evaluation, which is separate...
from the backtest used for computing predictive residuals – in each training-test split for evaluation, the latter half of the training split is used to perform a separate backtest to build the predictive residual collection without using information from the test split for a fair evaluation. The reported ACE is averaged across all training-test splits, 24-week forecast horizon for product sales and 48-hour horizon for M4-hourly, and the following range of target quantiles: $\tau = 0.1, 0.2, \ldots, 0.9$. For experiments with deep learning models, the reported ACE is also averaged across 10 trials due to the fluctuation in model performance.

Compared to other DF approaches, bootstrap approaches have the advantage of extending any PF model to produce DF, which makes them easy to adopt and able to potentially retain desired properties of the PF model. Thus, the first experiment focuses on comparing the proposed Backtest-Additive (BA) and Backtest-Multiplicative (BM) against classic bootstrap approaches for DF: bootstrap with fitted residuals (FR) [2] and bootstrap with fitted models (FM) [7, 22]. This experiment is performed on the product sales dataset, as it contains covariates which can accommodate the use of standard Machine Learning models as direct PF models. A variety of PF models are used to assess the bootstrap approaches’ robustness to the choice of PF model, including Ridge Regression [19], Support Vector Regression (SVR) [19], Random Forest (RF) [19], and Feed-forward Neural Networks (NN) [19]. The proposed bootstrap approaches outperform the classic approaches for all PF models (Table 1).

The second experiment compares against other State-of-the-Art DF approaches, including Quantile Lasso (QLasso) [19], Quantile Gradient Boosting (QGB) [19], DeepAR [3, 9], Deep Factors (DFact) [9, 31], MQ-CNN [9, 20], Deep State Space Models (DSSM) [4, 9], and Temporal Fusion Transformers (TFT) [9, 21]. Because the bootstrap approaches require an underlying PF model, for a fair comparison we use the median forecast from each of the aforementioned benchmarks as the PF models to be integrated with the backtest-based bootstrap, so they share the same model architecture and hyperparameters. The comparison against QGB and QLasso is performed on the product sales data and the comparison against the deep learning models is performed on the M4-hourly data. The proposed bootstrap approaches are integrated with the median forecast from each of the aforementioned benchmarks as the PF models to be integrated with the backtest-based bootstrap, so they share the same model architecture and hyperparameters. The comparison against QGB and QLasso is performed on the product sales dataset, and the comparison against the deep learning models is performed on the M4-hourly data. The proposed bootstrap approaches outperform the median forecast from each of the aforementioned benchmarks as the PF models to be integrated with the backtest-based bootstrap, so they share the same model architecture and hyperparameters. The comparison against QGB and QLasso is performed on the product sales dataset, and the comparison against the deep learning models is performed on the M4-hourly data.

The third experiment assesses the robustness of the proposed approaches to model assumptions/hyperparameters. DeepAR requires the output distribution to be specified prior to the model learning its parameters. In this experiment the backtest-based bootstrap approaches integrated with the

---

1Note on the data: QGB and QLasso are not traditional time series models and require engineered covariates available in the product sales data but not in M4-hourly, while the deep learning models implemented in GluonTS [9] package require the conditioning and forecast horizons fixed for all time series and the product sales data contain time series of varying lengths.

---

### Table 1: ACE comparison of different bootstrap DF approaches integrated with different PF models.

<table>
<thead>
<tr>
<th>Bootstrap\PF Model</th>
<th>Ridge</th>
<th>SVR</th>
<th>RF</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR</td>
<td>0.102</td>
<td>0.195</td>
<td>0.207</td>
<td>0.176</td>
</tr>
<tr>
<td>FM</td>
<td>0.095</td>
<td>0.218</td>
<td>0.171</td>
<td>0.125</td>
</tr>
<tr>
<td>BA</td>
<td>0.069</td>
<td>0.065</td>
<td>0.055</td>
<td>0.077</td>
</tr>
<tr>
<td>BM</td>
<td>0.038</td>
<td>0.061</td>
<td>0.027</td>
<td>0.048</td>
</tr>
</tbody>
</table>

### Table 2: ACE comparison of backtest-based bootstrap integrated with the median forecast vs the default DF.

<table>
<thead>
<tr>
<th>DF\Model</th>
<th>QLasso</th>
<th>QGB</th>
<th>DeepAR</th>
<th>DFact</th>
<th>MQCNN</th>
<th>DSSM</th>
<th>TFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0.188</td>
<td>0.119</td>
<td>0.102</td>
<td>0.098</td>
<td>0.092</td>
<td>0.136</td>
<td>0.067</td>
</tr>
<tr>
<td>Median + BA</td>
<td>0.114</td>
<td>0.078</td>
<td>0.100</td>
<td>0.067</td>
<td>0.078</td>
<td>0.124</td>
<td>0.058</td>
</tr>
<tr>
<td>Median + BM</td>
<td>0.039</td>
<td>0.036</td>
<td>0.104</td>
<td>0.070</td>
<td>0.071</td>
<td>0.112</td>
<td>0.060</td>
</tr>
</tbody>
</table>

### Table 3: ACE comparison of backtest-based bootstrap integrated with the median forecast vs the default DF from DeepAR under different pre-specified output distributions.

<table>
<thead>
<tr>
<th>DF\Output Dist.</th>
<th>Neg. Bin.</th>
<th>Normal</th>
<th>Gamma</th>
<th>Laplace</th>
<th>Poisson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0.102</td>
<td>0.192</td>
<td>0.162</td>
<td>0.138</td>
<td>0.114</td>
</tr>
<tr>
<td>Median + BA</td>
<td>0.100</td>
<td>0.109</td>
<td>0.116</td>
<td>0.157</td>
<td>0.094</td>
</tr>
<tr>
<td>Median + BM</td>
<td>0.104</td>
<td>0.165</td>
<td>0.111</td>
<td>0.156</td>
<td>0.088</td>
</tr>
</tbody>
</table>

### Table 4: Relative change in MAPE for Bagging PF compared to the original PF.

<table>
<thead>
<tr>
<th>Bootstrap\PF Model</th>
<th>Ridge</th>
<th>SVR</th>
<th>RF</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR</td>
<td>+0.8%</td>
<td>+6.5%</td>
<td>+0.2%</td>
<td>+0.7%</td>
</tr>
<tr>
<td>FM</td>
<td>+0.4%</td>
<td>+6.6%</td>
<td>-3.8%</td>
<td>+2.6%</td>
</tr>
<tr>
<td>BA</td>
<td>-12.3%</td>
<td>-21.0%</td>
<td>-10.0%</td>
<td>+1.5%</td>
</tr>
<tr>
<td>BM</td>
<td>-22.1%</td>
<td>-31.8%</td>
<td>-5.3%</td>
<td>-13.4%</td>
</tr>
</tbody>
</table>

---

5. CONCLUSION

This paper proposes a robust DF framework with backtest-based bootstrap and adaptive residual selection. It can efficiently extend an arbitrary PF model to generate DF, is robust to the choice of model, and outperforms a variety of benchmark DF methods on real-world data, making the proposed framework well-suited for industrial applications.
6. REFERENCES


