SurvivalEVAL: A Comprehensive Open-Source Python Package for Evaluating Individual Survival Distributions

Shi-ang Qi, Weijie Sun, Russell Greiner

Computing Science, University of Alberta
{shiang, weijie2, rgreiner}@ualberta.ca

Abstract

Survival analysis is widely employed across medicine, business, and the social sciences. However, the absence of a unified and standardized software for evaluating survival analysis models impedes its broader application by researchers. In this research, we fill this gap by providing a comprehensive Python package, `SurvivalEVAL`, which implements seven evaluation metrics specific to survival analysis. The `SurvivalEVAL` package is designed to serve as a convenient and straightforward toolkit for individual survival distribution models. The package is publicly available on GitHub at https://github.com/shi-ang/SurvivalEVAL.

Introduction and Background

Survival analysis models have been extensively used across diverse fields such as medicine, business, and social sciences. Of these models, those producing individual survival distributions (ISDs) (Haider et al. 2020) have demonstrated superior versatility. An ISD represents a survival probability curve given an instance's description, $S(t \mid \boldsymbol{x}_i)$, over all future times $t > 0$. The utility of ISDs extends beyond mere risk score prediction; they also excel in predicting single-time survival probabilities and time-to-event estimations.

The standardization of evaluation metrics is essential in model development and benchmarking. Over the past few decades, a myriad of ISD models have been introduced. Researchers use many evaluation metrics to quantify and compare the new ISD modeling approaches against each others, and also prior works. Although researchers understandably opt for metrics that align with their specific objectives, there is a marked inconsistency in metric selection across studies due to the different approaches to handling censored subjects in the evaluation. Even in the context of the widely adopted concordance index (C-index), multiple variations exist, ranging from the standard C-index (Harrell Jr, Lee, and Mark 1996) to the time-dependent C-index (Antolini, Boracchi, and Biganzoli 2005), IPCW C-index (Uno et al. 2011), margin C-index (Kumar et al. 2022), and many other variants designed for addressing ties. Such disparities complicate the reproduction of empirical findings in survival analysis and impede model comparisons.

One of the main reasons for this is the lack of a complete, standardized survival evaluation toolkit to evaluate ISD models. In contrast to classification or regression tasks, where evaluation tools are standardized and abundant, only a few Python packages focus on evaluating the predictive power of survival models. To make things worse, most of the available packages have implemented only a few evaluation metrics (see the comparison in Table **??**). As authors could not find a suitable package for the desired evaluation metrics, they needed to write the evaluation metrics by themselves to satisfy their interests. Furthermore, calibration metrics have been playing an important role in evaluating survival models in recent years. Manuscripts lacking calibration metrics in their evaluations are prone to criticism during peer review processes. However, none of the existing survival packages have included any calibration metrics.

In response, this paper introduces a Python package, `SurvivalEVAL`. This package includes a comprehensive scope of metrics for the evaluation of the ISD model. The package operates independently of the specific model types. Additionally, `SurvivalEVAL` features APIs tailored to seamlessly interface with outputs from widely used Python packages. We illustrate the convenience and effectiveness of our package by applying it to a popular survival model using a real-world dataset. With our `SurvivalEVAL` package, one can compare and evaluate different survival models in a convenient and comprehensive way.

Package Overview

This section will present seven evaluation metrics and necessary implementation details in `SurvivalEVAL` package.

Notation and Definition

We consider a survival dataset with N time-to-event tuples, $\mathcal{D} = \{(\boldsymbol{x}_i, t_i, \delta_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ represents the observed features for the i-th subject, t_i denotes the event or censoring time, and $\delta_i \in \{0, 1\}$ is a censor/event indicator where $\delta_i = 0$ means the subject is right-censored (the subject has not experienced an event at time t_i) and $\delta_i = 1$ means subject experienced the event at time t_i.

ISD models are designed to model the individual survival distribution $S(t|\boldsymbol{x}_i) = P(T > t|\mathbf{X} = \boldsymbol{x}_i)$. Further, $f(t|\boldsymbol{x}_i) = -\partial S(t|\boldsymbol{x}_i)/\partial t$ denotes the (conditional) probability density function (PDF) of the event time T.

Packages	C-index	MAE/MSE	IBS	D-Cal	AUC	BS	1-Cal
`lifelines` (Davidson-Pilon 2019)	✓	-	-	-	-	-	-
`pycox` (Kvamme, Borgan, and Scheel 2019)	✓†	-	✓	-	-	✓	-
`PySurvival` (Fotso et al. 2019)	✓	-	✓	-	-	✓	-
`scikit-survival` (Pölsterl 2020)	✓	-	✓	-	✓	✓	-
`auton-survival` (Nagpal, Potosnak, and Dubrawski 2022)	✓†	-	✓	-	✓	✓	-
`SurvivalEVAL`	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparisons of Python packages that are commonly used to evaluate survival predictions. †C-index implemented in `pycox` and `auton-survival` is the time-dependent C-index (Antolini, Boracchi, and Biganzoli 2005), which is different from the standard C-index proposed by Harrell Jr, Lee, and Mark (1996).

A predicted event time $\hat{t}_i$ can then be represented by either mean or median survival time, respectively:

$$\hat{t}_{i,\text{mean}} = \mathbb{E}_t[S(t|\boldsymbol{x}_i)] = \int_0^\infty S(t|\boldsymbol{x}_i)\, dt,$$
$$\hat{t}_{i,\text{median}} = \text{median}(S(t|\boldsymbol{x}_i)) = S^{-1}(0.5|\boldsymbol{x}_i).$$

Users have the flexibility to select their preferred method for calculating predicted times from these two options.

Calculating the mean and median survival time requires interpolation and extrapolation of the ISD curve. For this, we adopt specific interpolation methods to maintain the monotonic decreasing nature of the survival curves. Specifically, users have the option of selecting from (1) the linear method, (2) Forsythe, Malcolm and Moler method (Forsythe et al. 1977) combined with Hyman filtering (Hyman) (Hyman 1983), or (3) the piecewise cubic Hermite interpolating polynomial (PCHIP) method (Fritsch and Butland 1984). Among the three methods, the linear method is the fastest, but suffers from non-smooth interpolation. Hyman method is integrated within R packages, and our choice of Hyman ensures that interpolation results align with R's survival packages. Meanwhile, the PCHIP method is implemented in Python, offering faster execution times (2x faster than Hyman) albeit with a minor deviation in interpolation results compared to Hyman. For practical purposes, users may select any interpolation method, provided they apply it consistently (across different models) throughout the entire experiment.

For extrapolation, we need the ISD curves to be monotonic decreasing functions after the last time point, so we take a linear fit[1] of $(0, 1)$ and the last ISD point $(t_{\text{last}}, S(t_{\text{last}}|x))$, and then apply this linear function from the last time point to the time for which survival probability equals 0.

Concordance Index

The concordance index (C-index) is the most commonly used metric in survival analysis, measuring the model's ability to correctly rank the risks of the subjects. It is described as the proportion of all comparable pairs of subjects where the predicted and observed risk are concordant. A pair is comparable if we can determine who has the event first. The C-index is defined by Harrell Jr, Lee, and Mark (1996):

$$\text{C-index} = \frac{\sum_{i,j\in\mathcal{D}} \mathbb{1}_{t_i<t_j} \cdot \mathbb{1}_{\eta_i>\eta_j} \cdot \delta_i}{\sum_{i,j\in\mathcal{D}} \mathbb{1}_{t_i<t_j} \cdot \delta_i},$$

where η_i represents the risk score of subject i. In this `SurvivalEVAL` package, we use the negative of predicted times (mean or median) as risk scores for C-index. In addition to this standard C-index, we also implemented the margin C-index proposed in Kumar et al. (2022).

Mean Absolute/Squared Error

Survival prediction is like regression as it predicts a real number from a description of that subject. Therefore, it is natural to use the mean absolute or squared error (MAE/MSE) or mean squared error to evaluate the distance between the true event times and the predicted times. Several MAE/MSE variant has been proposed to handle right-censoring in survival analysis:

1. *Hinge* is a one-sided metric that considers only if the predicted time is earlier than the censored time, *i.e.*, $(t_i - \hat{t}_i) \cdot \mathbb{1}_{t_i>\hat{t}_i}$ and $(t_i - \hat{t}_i)^2 \cdot \mathbb{1}_{t_i>\hat{t}_i}$.
2. *Margin* (Haider et al. 2020) calculates a surrogate value for each censored subject, using the Kaplan Meier (KM) estimator (Kaplan and Meier 1958):
$$e_{\text{margin}}(t_i) = t_i + \frac{\int_{t_i}^\infty S_{\text{KM}(\mathcal{D})}(t)dt}{S_{\text{KM}(\mathcal{D})}(t)}.$$
3. *PO* (Qi et al. 2023) also calculates the surrogate values for each censored subject, but using pseudo-observation (PO) techniques. The PO value for the censored subject calculates how much this subject counts toward the group-level KM estimator, using jackknife resampling:
$$e_{\text{PO}}(t_i) = N \cdot \mathbb{E}_t[S_{\text{KM}}(t)] - (N-1) \cdot \mathbb{E}_t[S_{\text{KM}^{-i}}(t)].$$

For *Margin* and *PO*, once we calculate the surrogate value e_{margin} or e_{PO} for all the censored subjects, we can then replace the censor times label with the surrogate times and calculate MAE/MSE as a standard regression task. We also provide the option of using the weighting scheme to represent the confidence in the surrogate value for censored subjects as introduced in Qi et al. (2023). For completeness, we also implemented the *uncensored*, *IPCW-D*, and *IPCW-T* versions of MAE/MSE in Qi et al. (2023).

[1] If the last two time points on the ISD curves have the same probability, the cubic fitting (Hyman and PCHIP methods) will remain constant beyond the last time range, never reaching 0. This can result in an inaccurate calculation of mean and median survival times, which tends toward infinity.

Brier Score

Brier score (BS) is another commonly used metric in survival analysis, which measures the mean squared error between the observed binary status and the predicted survival probability at a target time t^*. In the `SurvivalEVAL` package, we will take the most classical method to calculate the BS by weighting the prediction residuals by inverse probability censoring weights (IPCW) (Graf et al. 1999):

$$\text{BS}(t^*) = \frac{1}{N}\sum_{i\in\mathcal{D}}\left[\frac{S(t^*|\boldsymbol{x}_i)^2\cdot\mathbb{1}_{t_i\leq t^*,\delta_i=1}}{G(t_i)} + \frac{(1-S(t^*|\boldsymbol{x}_i))^2\cdot\mathbb{1}_{t_i>t^*}}{G(t^*)}\right],$$

where $G(t)$ is the non-censoring probability at time t, which is estimated with KM on the censoring distribution (flip the censoring indicator of data), and its reciprocal is referred to as the IPCW. For BS calculation, researchers will need to manually choose t^* based on the research objectives. Otherwise, median time of the dataset will be used as default.

Integrated Brier Score

The integrated Brier score (IBS) is the expectation of single-time BS over time. IBS for survival prediction is typically defined as:

$$\text{IBS} = \frac{1}{N}\sum_{i\in\mathcal{D}}\frac{1}{t_{\text{max}}}\cdot\int_0^{t_{\text{max}}}\text{BS}(t)\,dt,$$

where t_{max} is defined as the maximum event time of the combined training and validation datasets.

Distribution Calibration

Distribution calibration (D-Cal) (Haider et al. 2020) is a popular calibration metric for evaluating ISD models. It is a statistical test to evaluate the reliability of the predicted ISD curves. As to the notation, for any probability interval $[a, b] \subset [0, 1]$, let

$$\mathcal{D}(a,b) = \{[\boldsymbol{x}_i, t_i, \delta_i = 1] \in \mathcal{D} \mid S(t_i|\boldsymbol{x}_i) \in [a,b]\}$$

be the subset of the subjects in the dataset $\mathcal{D}$ whose predicted probability at its event time, $S(t_i|\boldsymbol{x}_i)$, is in the interval $[a, b]$. The model is D-calibrated if the proportion of patients $|D(a,b)|/|D|$ is statistically similar to the proportion $b - a$, for any choice of a and b. It is common to use equal-sized, mutually exclusive intervals with Pearson's χ^2 test to examine if the proportion of patients in each bin is uniformly distributed. To handle censored subjects, we uniformly "split" each censored subject into subsequent probability intervals after $S(t_i|\boldsymbol{x}_i)$ (Haider et al. 2020).

Area under the ROC Curve

The area under the receiver operating characteristic curve (AUC) can be extended to the survival dataset. To be specific, given the individual risk scores $\{\eta_i\}_{i\in\mathcal{D}}$ at a target time t^*, the AUC can be calculated by:

$$\text{AUC}(t^*) = \frac{\sum_{i,j\in\mathcal{D}}\mathbb{1}_{t_i<t^*}\cdot\mathbb{1}_{t_j>t^*}\cdot\mathbb{1}_{\eta_i>\eta_j}\cdot\delta_i}{\sum_{i,j}\mathbb{1}_{t_i<t^*}\cdot\mathbb{1}_{t_j>t^*}\cdot\delta_i}.$$

For AUC calculation, we use the predicted survival probabilities at time t^* (*i.e.*, $S(t^*|\boldsymbol{x}_i)$) as the risk scores.

Hosmer-Lemeshow Calibration

Hosmer-Lemeshow calibration (1-Cal) (Hosmer and Lemesbow 1980) is a statistical test to evaluate the calibration ability of the risk predictions at time t^*. We define the risk score at time t^* as the survival probability, $S(t^*|\boldsymbol{x}_i)$, just like the AUC score.

To calculate 1-Cal statistics, we first sort the risk scores and group them into K bins. Within each bin, we calculate the expected number of events and compare it to the observed event number. We use the Hosmer-Lemeshow test to assess if the expected and observed event rates are statistically similar. To handle censored subjects, we can use the KM estimator to approximate the observed number of events for each bin at time t^* (D'Agostino and Nam 2003).

Other Metrics

The current `SurvivalEVAL` only includes the above mentioned evaluation metrics. However, there are many other existing evaluation variants that have been used by other research. For example, time-dependent C-index (Antolini, Boracchi, and Biganzoli 2005), IPCW C-index (Uno et al. 2011), IPCW AUC (Hung and Chiang 2010), administrative BS (Kvamme and Borgan 2023), etc. We will deliver the implementation for these variants in the future. Furthermore, the current packages only deal with the right-censoring format. We will improve the package by incorporating other forms of censoring, such as interval censoring.

Usage Example

In this section, we use an example with the corresponding Python code to illustrate the effectiveness of the metrics provided in `SurvivalEVAL` package.

The dataset we use in this example is the German Breast Cancer Study Group 2 dataset (GBSG2) (Schumacher et al. 1994). The dataset contains 686 samples and 8 features. It has a censoring rate of 43.6% with the target event of recurrence-free survival. We first perform a preprocessing on the GBSG2 dataset and split the training and testing datasets using the following code:

```
from lifelines.datasets import load_gbsg2
# Load the data and split train/test set
gbsg2 = load_gbsg2()
gbsg2 = gbsg2.replace({"horTh": {"no": 0, "yes": 1}, "menostat": {"Pre": 0, "Post": 1}, "tgrade": {"I": 1, "II": 2, "III": 3}})
train, test = gbsg2.iloc[:400, :], gbsg2.iloc[400:, :]
```

Then we choose a model from the `lifelines` package, and then train the model using the training set and perform inference on the testing set. To be specific, we choose the most popular Cox proportional hazard (CoxPH) (Cox 1972) model to check and test the provided functionalities.

```
from lifelines import CoxPHFitter
isd_curves = CoxPHFitter().fit(train, duration_col="time", event_col="cens").predict_survival_function(test)
```

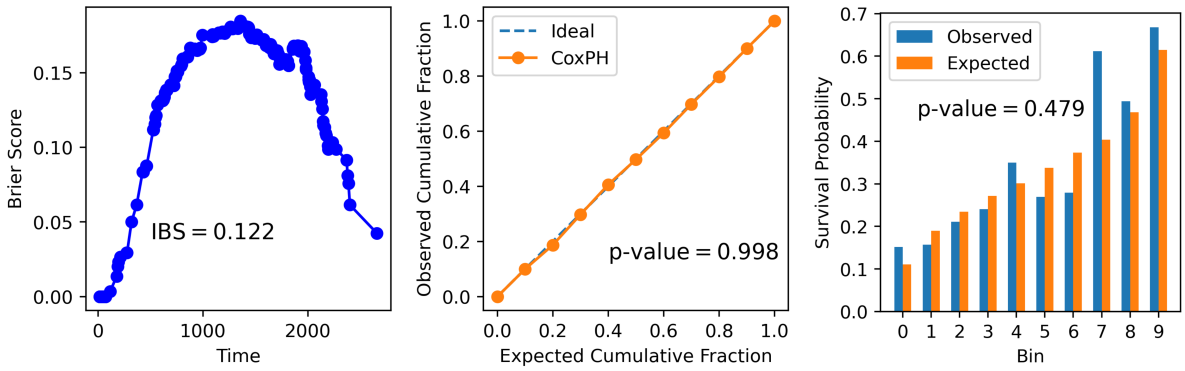

Figure 1: Visualization of the results of evaluation metrics. Left: Brier scores over all time points. Middle: the quantile-quantile plot of the observed histogram for D-Cal. Right: observed and expected event histograms of 1-Cal.

Once we get the predicted ISD curves for the test data from the model, we can then use the APIs in `SurvivalEVAL` to seamlessly plug in these ISD outputs. For example:

```
from Evaluator import LifelinesEvaluator
evl = LifelinesEvaluator(isd_curves,
    test.time, test.cens, train.time,
    train.cens)
```

`SurvivalEVAL` has tailored APIs for many popular survival Python packages, including `lifelines`, `pycox`, `PySurvival`, `scikit-survival`, and `auton-survival`. For other customized ISD models, users can use the `SurvivalEvaluator` API that simply takes the 2-D survival curve matrix, the 1-D time points vector, and the true labels as inputs, and follows the below guidelines to calculate the evaluation scores.

Once the user gets everything ready, s/he can perform the evaluation, starting from the C-index:

```
cindex, correct_pairs, total_pairs = evl
    .concordance()
```

This method returns three float values, representing the C-index (0.688), the number of correctly ordered pairs (13630), and the number of total comparable pairs (19821).

To calculate MAE, the user can simply type:

```
mae = evl.mae(method="Pseudo_obs",
    weighted=True, log_scale=False)
```

which calculates the MAE-PO value for the testing set. The user can also choose other methods like *hinge*, *margin*, etc, and choose to whether use a weighted scheme and logarithmic scale or not.

Then, to calculate the IBS score, one can use:

```
ibs = evl.integrated_brier_score(
    num_points=None, draw_figure=True)
```

`num_points` refers to the number of uniformly distributed points at which the BS is to be calculated. If None, the points are set to be the unique censored times from the test set. Users can also set the argument `draw_figure` to `True` to show Figure 1 (left), which represents the BS values at all the unique time points in the training data.

To calculate D-Cal, the user needs to decide how many bins they will use to split the quantile and performs the statistical test. A common choice could be 10 bins:

```
p_value, bin_hist = evl.d_calibration(
    num_bins=10)
```

This method returns a p-value for the χ^2 test and also the bin histogram for the predefined 10 bins. We can visualize the D-calibration results using the bin histogram to generate a quantile-quantile (q-q) plot (Figure 1 middle). The closer the model's q-q plot to the ideal one, the better the model is D-Calibrated.

If the user wants to evaluate the model's performance at a single time point, s/he can calculate the AUC, BS, and 1-Cal with a predefined target time (for example, 1000 days).

```
target_t = 1000
auc = evl.auc(target_t)
bs = evl.brier_score(target_t)
p,ob,exp = evl.one_calibration(target_t)
```

We can obtain an AUC score of 0.720 and a BS of 0.175 for the model. As to the 1-Cal, the p-value $(\mathrm{p}) = 0.479 > 0.05$ indicates that the predicted probabilities are 1-Calibrated at 1000 days. We can visualize the numbers of observed events and expected events using a histogram in Figure 1 (right).

Concluding Remarks

Evaluating the predictive capabilities of survival models presents challenges due to the absence of a universally accepted criterion for evaluation metrics. Current metrics are distributed among various Python or R packages with inconsistent interfaces, complicating the task for non-specialists.

In this study, we address this issue by introducing `SurvivalEVAL`, a comprehensive Python package that offers a uniform interface to a broad range of performance assessment and statistical comparison techniques. The package facilitates a straightforward interface for evaluating and comparing models. The current version of `SurvivalEVAL` incorporates seven evaluation metrics. More metrics and variants are under development and will be integrated in future iterations. Additionally, we are enhancing the package's documentation and interface to better assist non-computer science researchers in survival model evaluation.

Acknowledgments

SQ, WS, and RG are grateful for the funding from NSERC (Canada's Natural Science and Engineering Research Council), CIFAR (Canadian Institute for Advanced Research) and Amii (Alberta Machine Intelligence Institute).

References

Antolini, L.; Boracchi, P.; and Biganzoli, E. 2005. A time-dependent discrimination index for survival data. *Statistics in medicine*, 24(24): 3927–3944.

Cox, D. R. 1972. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2): 187–202.

D'Agostino, R. B.; and Nam, B.-H. 2003. Evaluation of the performance of survival analysis models: discrimination and calibration measures. *Handbook of statistics*, 23: 1–25.

Davidson-Pilon, C. 2019. lifelines: survival analysis in Python. *Journal of Open Source Software*, 4(40): 1317.

Forsythe, G. E.; et al. 1977. *Computer methods for mathematical computations*. Prentice-hall.

Fotso, S.; et al. 2019. PySurvival: open source package for survival analysis modeling. *URL: https://www. pysurvival. io*.

Fritsch, F. N.; and Butland, J. 1984. A method for constructing local monotone piecewise cubic interpolants. *SIAM journal on scientific and statistical computing*, 5(2): 300–304.

Graf, E.; Schmoor, C.; Sauerbrei, W.; and Schumacher, M. 1999. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, 18(17-18): 2529–2545.

Haider, H.; Hoehn, B.; Davis, S.; and Greiner, R. 2020. Effective ways to build and evaluate individual survival distributions. *The Journal of Machine Learning Research*, 21(1): 3289–3351.

Harrell Jr, F. E.; Lee, K. L.; and Mark, D. B. 1996. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in medicine*, 15(4): 361–387.

Hosmer, D. W.; and Lemesbow, S. 1980. Goodness of fit tests for the multiple logistic regression model. *Communications in statistics-Theory and Methods*, 9(10): 1043–1069.

Hung, H.; and Chiang, C.-T. 2010. Estimation methods for time-dependent AUC models with survival data. *Canadian Journal of Statistics*, 38(1): 8–26.

Hyman, J. M. 1983. Accurate monotonicity preserving cubic interpolation. *SIAM Journal on Scientific and Statistical Computing*, 4(4): 645–654.

Kaplan, E. L.; and Meier, P. 1958. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282): 457–481.

Kumar, N.; Qi, S.-a.; Kuan, L.-H.; Sun, W.; Zhang, J.; and Greiner, R. 2022. Learning accurate personalized survival models for predicting hospital discharge and mortality of COVID-19 patients. *Scientific reports*, 12(1): 4472.

Kvamme, H.; and Borgan, Ø. 2023. The Brier Score under Administrative Censoring: Problems and a Solution. *Journal of Machine Learning Research*, 24(2): 1–26.

Kvamme, H.; Borgan, Ø.; and Scheel, I. 2019. Time-to-Event Prediction with Neural Networks and Cox Regression. *Journal of Machine Learning Research*, 20: 1–30.

Nagpal, C.; Potosnak, W.; and Dubrawski, A. 2022. autonsurvival: An open-source package for regression, counterfactual estimation, evaluation and phenotyping with censored time-to-event data. In *Machine Learning for Healthcare Conference*, 585–608. PMLR.

Pölsterl, S. 2020. scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn. *Journal of Machine Learning Research*, 21(212): 1–6.

Qi, S.-A.; Kumar, N.; Farrokh, M.; Sun, W.; Kuan, L.-H.; Ranganath, R.; Henao, R.; and Greiner, R. 2023. An Effective Meaningful Way to Evaluate Survival Models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, 28244–28276. PMLR.

Schumacher, M.; Bastert, G.; Bojar, H.; Hübner, K.; Olschewski, M.; Sauerbrei, W.; Schmoor, C.; Beyerle, C.; Neumann, R.; and Rauschecker, H. 1994. Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. German Breast Cancer Study Group. *Journal of Clinical Oncology*, 12(10): 2086–2093.

Uno, H.; Cai, T.; Pencina, M. J.; D'Agostino, R. B.; and Wei, L.-J. 2011. On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in medicine*, 30(10): 1105–1117.