# Stacked Inverse Probability of Censoring Weighted Bagging: a case study in the InfCareHIV Register


Pablo Gonzalez Ginestet

Karolinska Institutet - MEB

*pablo.gonzalez.ginestet@ki.se*


**ISCB 41, Krakow 2020**
August 25, 2020

# Motivation

- Predicting the risk of an event during a specific time frame can play an important role in public health (example: identifying patients who are at high risk of developing an adverse outcome)

# Motivation

- Predicting the risk of an event during a specific time frame can play an important role in public health (example: identifying patients who are at high risk of developing an adverse outcome)
- Supervised machine learning (ML) algorithms (ex. deep neural network, ensemble methods like bagging & stacking) are useful for building risk prediction models

# Motivation

- Predicting the risk of an event during a specific time frame can play an important role in public health (example: identifying patients who are at high risk of developing an adverse outcome)
- Supervised machine learning (ML) algorithms (ex. deep neural network, ensemble methods like bagging & stacking) are useful for building risk prediction models
- Survival data (right-censoring & competing risk) is an obstacle to the direct applicability of many ML algorithms

# Motivation

*Attempts to adapt ML to censored data*

- i) Inverse probability of censoring weighting (IPCW) (Molinaro et al. (2004); Hothorn et al. (2006); Goldberg and Kosorok (2017); olfson et al. (2015), Bandyopadhyay et al. (2015) and Vock et al. (2016))

*Attempts to adapt ML to censored data*

- i) Inverse probability of censoring weighting (IPCW) (Molinaro et al. (2004); Hothorn et al. (2006); Goldberg and Kosorok (2017); olfson et al. (2015), Bandyopadhyay et al. (2015) and Vock et al. (2016))
- ii) other approaches (Hothorn et al. (2004), Ishwaran et al. (2014); Shivaswamy et al. (2007), Zubek and Khan (2008), and Van Belle et al. (2011); Binder et al. (2009))

# Motivation

*Attempts to adapt ML to censored data*

- i) Inverse probability of censoring weighting (IPCW) (Molinaro et al. (2004); Hothorn et al. (2006); Goldberg and Kosorok (2017); olfson et al. (2015), Bandyopadhyay et al. (2015) and Vock et al. (2016))
- ii) other approaches (Hothorn et al. (2004), Ishwaran et al. (2014); Shivaswamy et al. (2007), Zubek and Khan (2008), and Van Belle et al. (2011); Binder et al. (2009))

*Remark* Specific to the algorithm so limits the generalizability

# Motivation

*Attempts to adapt ML to censored data*

- i) Inverse probability of censoring weighting (IPCW) (Molinaro et al. (2004); Hothorn et al. (2006); Goldberg and Kosorok (2017); olfson et al. (2015), Bandyopadhyay et al. (2015) and Vock et al. (2016))
- ii) other approaches (Hothorn et al. (2004), Ishwaran et al. (2014); Shivaswamy et al. (2007), Zubek and Khan (2008), and Van Belle et al. (2011); Binder et al. (2009))

*Remark* Specific to the algorithm so limits the generalizability

*Remark* Many of these developments do not have software implementations

# Setup and Notation

- $K$ competing events
- $T_i$ event-time of type $\eta_i \in 1, \dots, K$
- $C_i$ censoring time
- $\tilde{T}_i = \min(T_i, C_i)$
- $\Delta_i = 1(T_i \leq C_i)$ and the censored event type $\tilde{\eta}_i = \Delta_i \eta_i$.
- Observed data $(\tilde{T}_i, \Delta_i, \tilde{\eta}, \mathbf{X}_i)_{i=1,\dots,n}$

# Setup and Notation

- $K$ competing events
- $T_i$ event-time of type $\eta_i \in 1, \ldots, K$
- $C_i$ censoring time
- $\tilde{T}_i = \min(T_i, C_i)$
- $\Delta_i = 1(T_i \leq C_i)$ and the censored event type $\tilde{\eta}_i = \Delta_i \eta_i$.
- Observed data $(\tilde{T}_i, \Delta_i, \tilde{\eta}, \mathbf{X}_i)_{i=1,\ldots,n}$
- Outcome

$$E_{k,i}(\tau) = \begin{cases} 1 & \text{if } \tilde{T}_i \leq \tau \text{ and } \tilde{\eta}_i = k \\ 0 & \text{if } \tilde{T}_i \leq \tau \text{ and } \tilde{\eta}_i \notin \{0, k\} \text{ or } \tilde{T}_i > \tau \\ \text{missing} & \text{otherwise.} \end{cases}$$

# Our Approach

- *Pre-processing step that combines IPCW and bagging*, which allows for all existing and any newly developed ML methods for classification to be applied to right-censored data with or without competing risks for the *TASK:* $\mathbb{P}(T_i \leq \tau, \eta_i = k | \mathbf{X}_i)$

# Our Approach

- *Pre-processing step that combines IPCW and bagging*, which allows for all existing and any newly developed ML methods for classification to be applied to right-censored data with or without competing risks for the *TASK:* $\mathbb{P}(T_i \leq \tau, \eta_i = k | \mathbf{X}_i)$
- *PERFORMANCE* evaluation metric: $Risk(f_k) = 1 - AUC_{IPCW}(E_k, f_k, \tau)$ Blanche et al.[2013]

# Our Approach

- *Pre-processing step that combines IPCW and bagging*, which allows for all existing and any newly developed ML methods for classification to be applied to right-censored data with or without competing risks for the *TASK:* $\mathbb{P}(T_i \leq \tau, \eta_i = k|\mathbf{X}_i)$
- *PERFORMANCE* evaluation metric: $Risk(f_k) = 1 - AUC_{IPCW}(E_k, f_k, \tau)$ Blanche et al.[2013]
- Possibility to optimally stack predictions from any IPCW bagged ML methods

## *Inverse Probability of Censoring Weighting (IPCW)*

- Due to censoring,
  $\mathbb{P}(E_{k,i}(\tau) = 1|\mathbf{X}_i) \neq \mathbb{P}(T_i \leq \tau, \eta_i = k|\mathbf{X}_i)$

## *Inverse Probability of Censoring Weighting (IPCW)*

- Due to censoring, $\mathbb{P}(E_{k,i}(\tau) = 1|\mathbf{X}_i) \neq \mathbb{P}(T_i \leq \tau, \eta_i = k|\mathbf{X}_i)$
- Inverse probability of censoring weighting (IPCW) estimation is a two-step procedure
- i) Fit a model to $G(t|\mathbf{X}_i) = P(C_i > t|\mathbf{X}_i)$ (ex. Cox proportional hazard model, Aalen's linear hazard model, boosting in Cox regression or random forest)

## *Inverse Probability of Censoring Weighting (IPCW)*

- Due to censoring, $\mathbb{P}(E_{k,i}(\tau) = 1|\mathbf{X}_i) \neq \mathbb{P}(T_i \leq \tau, \eta_i = k|\mathbf{X}_i)$
- Inverse probability of censoring weighting (IPCW) estimation is a two-step procedure
- i) Fit a model to $G(t|\mathbf{X}_i) = P(C_i > t|\mathbf{X}_i)$ (ex. Cox proportional hazard model, Aalen's linear hazard model, boosting in Cox regression or random forest)
- ii) Weight each individual by

$$w_i = \begin{cases} \frac{1}{G(\min(T_i,\tau)|\mathbf{X}_i))} & \text{if } \min(T_i, \tau) \leq C_i \\ 0 & \text{otherwise} \end{cases}$$

- Dependent censoring under assumption of coarsening at random (CAR).

## *Stacked IPCW Bagging*

1. Algorithm 1: to obtain predictions in a test set
2. Algorithm 2: to combine optimally algorithms based on a target loss function ($AUC_{IPCW}$)

## Stacked IPCW Bagging

1. Algorithm 1: to obtain predictions in a test set
2. Algorithm 2: to combine optimally algorithms based on a target loss function ($AUC_{IPCW}$)

- Training set $\mathcal{Q}$ and a validation or test set $\mathcal{V}$

## *Stacked IPCW Bagging*

1. Algorithm 1: to obtain predictions in a test set
2. Algorithm 2: to combine optimally algorithms based on a target loss function ($AUC_{IPCW}$)

- Training set $\mathcal{Q}$ and a validation or test set $\mathcal{V}$
- Library of ML algorithms: $\mathcal{L} = \{\varphi_a$ such that $a \in 1, \ldots, A\}$ where $\varphi_a$ is an algorithm , $A = \#$ algorithms

## *Stacked IPCW Bagging*

1. Algorithm 1: to obtain predictions in a test set
2. Algorithm 2: to combine optimally algorithms based on a target loss function ($AUC_{IPCW}$)

- Training set $\mathcal{Q}$ and a validation or test set $\mathcal{V}$
- Library of ML algorithms: $\mathcal{L} = \{\varphi_a$ such that $a \in 1, \ldots, A\}$ where $\varphi_a$ is an algorithm , $A = \#$ algorithms
- $\widehat{Y}_{a,i} = \hat{\varphi}_a(\mathbf{X}_i^{\mathcal{V}}, \mathcal{Q})$ be the prediction for subject $i$ in $\mathcal{V}$ using ML algorithm $\hat{\varphi}_a$ given their covariates $\mathbf{X}_i^{\mathcal{V}}$ and trained in the training sample $\mathcal{Q}$.

## *Stacked IPCW Bagging*

1. Algorithm 1: to obtain predictions in a test set
2. Algorithm 2: to combine optimally algorithms based on a target loss function ($AUC_{IPCW}$)

- Training set $\mathcal{Q}$ and a validation or test set $\mathcal{V}$
- Library of ML algorithms: $\mathcal{L} = \{\varphi_a$ such that $a \in 1, \ldots, A\}$ where $\varphi_a$ is an algorithm , $A = \#$ algorithms
- $\widehat{Y}_{a,i} = \hat{\varphi}_a(\mathbf{X}_i^{\mathcal{V}}, \mathcal{Q})$ be the prediction for subject $i$ in $\mathcal{V}$ using ML algorithm $\hat{\varphi}_a$ given their covariates $\mathbf{X}_i^{\mathcal{V}}$ and trained in the training sample $\mathcal{Q}$.
- $\hat{w}_i^n = \frac{\hat{w}_i}{\sum_i \hat{w}_i}$

**Algorithm 1** Stacked IPCW Bagging

**Input:** data $\mathbf{X}^{\mathcal{V}}$ and $\mathcal{Q}$; $w^n$, $\mathcal{L}$, $\varsigma$ and $\underline{\widehat{\beta}}$.

**Algorithm 2** Stacked IPCW Bagging

1: **Input:** data $\mathbf{X}^{\mathcal{V}}$ and $\mathcal{Q}$; $w^n$, $\mathcal{L}$, $\varsigma$ and $\underline{\widehat{\beta}}$.

2: Produce $B$ training sets $Q_j^{\omega}, j = \overline{1}, \ldots, B$ by performing weighted resampling with replacement from the original training set $\mathcal{Q}$, using the normalized IPCW-weights $w_i^n$.

**Algorithm 3** Stacked IPCW Bagging

1: **Input:** data $\mathbf{X}^{\mathcal{V}}$ and $\mathcal{Q}$; $w^n$, $\mathcal{L}$, $\varsigma$ and $\underline{\widehat{\beta}}$.

2: Produce $B$ training sets $Q_j^{\omega}, j = \overline{1}, \ldots, B$ by performing weighted resampling with replacement from the original training set $\mathcal{Q}$, using the normalized IPCW-weights $w_i^n$.

3: **for** each $a \in \mathcal{L}$ **do**

**Algorithm 4** Stacked IPCW Bagging

**Input:** data $\mathbf{X}^{\mathcal{V}}$ and $\mathcal{Q}$; $w^n$, $\mathcal{L}$, $\varsigma$ and $\underline{\widehat{\beta}}$.
Produce $B$ training sets $Q^{\omega}_j, j = \overline{1}, \ldots, B$ by performing weighted resampling with replacement from the original training set $\mathcal{Q}$, using the normalized IPCW-weights $w^n_i$.
**for** each $a \in \mathcal{L}$ **do**
    Train the ML technique $\varphi_a$ on each bootstrap sample and obtain B fits: $\hat{\varphi}_{a,j}$ for $j = 1, \ldots, B$.

**Algorithm 5** Stacked IPCW Bagging

1: **Input:** data $\mathbf{X}^{\mathcal{V}}$ and $\mathcal{Q}$; $w^n$, $\mathcal{L}$, $\varsigma$ and $\underline{\widehat{\beta}}$.

2: Produce $B$ training sets $Q_j^{\omega}, j = 1, \ldots, B$ by performing weighted resampling with replacement from the original training set $\mathcal{Q}$, using the normalized IPCW-weights $w_i^n$.

3: **for** each $a \in \mathcal{L}$ **do**

4: Train the ML technique $\varphi_a$ on each bootstrap sample and obtain B fits: $\hat{\varphi}_{a,j}$ for $j = 1, \ldots, B$.

5: Compute $\hat{Y}_{a,i,j} = \hat{\varphi}_{a,j}(\mathbf{X}_i^{\mathcal{V}}, Q_j^{\omega})$ for $j = 1, \ldots, B$.

**Algorithm 6** Stacked IPCW Bagging

**Input:** data $\mathbf{X}^{\mathcal{V}}$ and $\mathcal{Q}$; $w^n$, $\mathcal{L}$, $\varsigma$ and $\widehat{\underline{\beta}}$.

Produce $B$ training sets $Q_j^{\omega}, j = 1, \ldots, B$ by performing weighted resampling with replacement from the original training set $\mathcal{Q}$, using the normalized IPCW-weights $w_i^n$.

**for** each $a \in \mathcal{L}$ **do**

  Train the ML technique $\varphi_a$ on each bootstrap sample and obtain B fits: $\hat{\varphi}_{a,j}$ for $j = 1, \ldots, B$.

  Compute $\hat{Y}_{a,i,j} = \hat{\varphi}_{a,j}(\mathbf{X}_i^{\mathcal{V}}, Q_j^{\omega})$ for $j = 1, \ldots, B$.

  Output: $\hat{Y}_{a,i} = \sum_{j=1}^{B} \hat{Y}_{a,i,j}$ for $i = 1, \ldots, N$

**Algorithm 7** Stacked IPCW Bagging

**Input:** data $\mathbf{X}^{\mathcal{V}}$ and $\mathcal{Q}$; $w^n$, $\mathcal{L}$, $\varsigma$ and $\underline{\widehat{\beta}}$.

Produce $B$ training sets $Q_j^{\omega}, j = 1, \ldots, B$ by performing weighted resampling with replacement from the original training set $\mathcal{Q}$, using the normalized IPCW-weights $w_i^n$.

**for** each $a \in \mathcal{L}$ **do**

  Train the ML technique $\varphi_a$ on each bootstrap sample and obtain B fits: $\hat{\varphi}_{a,j}$ for $j = 1, \ldots, B$.

  Compute $\hat{Y}_{a,i,j} = \hat{\varphi}_{a,j}(\mathbf{X}_i^{\mathcal{V}}, Q_j^{\omega})$ for $j = 1, \ldots, B$.

  Output: $\hat{Y}_{a,i} = \sum_{j=1}^{B} \hat{Y}_{a,i,j}$ for $i = 1, \ldots, N$

**end for**

Output: $\widehat{Y}_i(\underline{\widehat{\beta}}) = \varsigma(\hat{Y}_{1,i}, \ldots, \hat{Y}_{A,i} | \hat{\beta}_1, \ldots, \hat{\beta}_A)$ for $i = 1, \ldots, N$. One can apply, potentially IPCW, performance metrics to $\widehat{Y}_i$ in the validation sample to obtain an accurate estimate of prediction accuracy.

**Algorithm 8** Obtaining optimal coefficients for stacking

1: **Input:** data $\mathcal{Q}$; $w^n$, $\mathcal{L}$ and $\varsigma$

**Algorithm 9** Obtaining optimal coefficients for stacking

1: **Input:** data $\mathcal{Q}$; $w^n$, $\mathcal{L}$ and $\varsigma$ .

2: Partition the training sample $\mathcal{Q}$ into $V$-fold. Denote them $Q_1, ..., Q_V$

**Algorithm 10** Obtaining optimal coefficients for stacking

**Input:** data $\mathcal{Q}$; $w^n$, $\mathcal{L}$ and $\varsigma$ .

Partition the training sample $\mathcal{Q}$ into $V$-fold. Denote them $Q_1, ..., Q_V$

**for** $v = 1$ to $V$ **do**

  **for** $j = 1$ to $B$ **do**

    Obtain a weighted bootstrap sample from the training set $\mathcal{Q} \setminus Q_v$

**Algorithm 11** Obtaining optimal coefficients for stacking

**Input:** data $\mathcal{Q}$; $w^n$, $\mathcal{L}$ and $\varsigma$ .

Partition the training sample $\mathcal{Q}$ into $V$-fold. Denote them $Q_1, ..., Q_V$

**for** $v = 1$ to $V$ **do**

  **for** $j = 1$ to $B$ **do**

    Obtain a weighted bootstrap sample from the training set $\mathcal{Q} \setminus Q_v$

    Train each $\varphi_a \in \mathcal{L}$ on the same bootstrap sample and compute prediction for the validation set $Q_v$: $\{\hat{Y}^v_{j,1}, .., \hat{Y}^v_{j,A}\}$

**Algorithm 12** Obtaining optimal coefficients for stacking

**Input:** data $\mathcal{Q}$; $w^n$, $\mathcal{L}$ and $\varsigma$ .

Partition the training sample $\mathcal{Q}$ into $V$-fold. Denote them $Q_1, ..., Q_V$

**for** $v = 1$ to $V$ **do**

  **for** $j = 1$ to $B$ **do**

    Obtain a weighted bootstrap sample from the training set $\mathcal{Q} \setminus Q_v$

    Train each $\varphi_a \in \mathcal{L}$ on the same bootstrap sample and compute prediction for the validation set $Q_v$: $\{\hat{Y}^v_{j,1}, .., \hat{Y}^v_{j,A}\}$

  **end for**

  Bag the predictions: $\hat{Y}^v_a = \frac{1}{B} \sum_{j=1}^{B} \hat{Y}^v_{j,a}$ for $a = 1, \ldots A$

**end for**

**Algorithm 13** Obtaining optimal coefficients for stacking

**Input:** data $\mathcal{Q}$; $w^n$, $\mathcal{L}$ and $\varsigma$ .
Partition the training sample $\mathcal{Q}$ into $V$-fold. Denote them $Q_1, ..., Q_V$
**for** $v = 1$ to $V$ **do**
  **for** $j = 1$ to $B$ **do**
    Obtain a weighted bootstrap sample from the training set $\mathcal{Q} \setminus Q_v$
    Train each $\varphi_a \in \mathcal{L}$ on the same bootstrap sample and compute prediction for the validation set $Q_v$: $\{\hat{Y}^v_{j,1}, .., \hat{Y}^v_{j,A}\}$
  **end for**
  Bag the predictions: $\hat{Y}^v_a = \frac{1}{B}\sum_{j=1}^{B} \hat{Y}^v_{j,a}$ for $a = 1, \ldots A$
**end for**
Stack them: $\hat{\mathbf{Y}}^Q_a = [\hat{Y}^{v=1}_a, \ldots, \hat{Y}^{v=V}_a]$ for $a = 1, \ldots A$

**Algorithm 14** Obtaining optimal coefficients for stacking

**Input:** data $\mathcal{Q}$; $w^n$, $\mathcal{L}$ and $\varsigma$ .

Partition the training sample $\mathcal{Q}$ into $V$-fold. Denote them $Q_1, ..., Q_V$

**for** $v = 1$ to $V$ **do**

  **for** $j = 1$ to $B$ **do**

    Obtain a weighted bootstrap sample from the training set $\mathcal{Q} \setminus Q_v$

    Train each $\varphi_a \in \mathcal{L}$ on the same bootstrap sample and compute prediction for the validation set $Q_v$: $\{\hat{Y}^v_{j,1}, .., \hat{Y}^v_{j,A}\}$

  **end for**

  Bag the predictions: $\hat{Y}^v_a = \frac{1}{B}\sum_{j=1}^{B} \hat{Y}^v_{j,a}$ for $a = 1, \ldots A$

**end for**

Stack them: $\hat{\mathbf{Y}}^Q_a = [\hat{Y}^{v=1}_a, \ldots, \hat{Y}^{v=V}_a]$ for $a = 1, \ldots A$

Minimize IPCW loss function over the validation sample predictions $\mathbb{L}(E_i, \hat{\mathbf{Y}}^{\mathcal{Q}}_{i,1}, \ldots, \hat{\mathbf{Y}}^{\mathcal{Q}}_{i,A}, w^n_i, \varsigma | \beta_1, \ldots, \beta_A)$ and obtain $\underline{\widehat{\beta}}$.

## $Performance : \quad AUC_{IPCW}(E_k, f_k, \tau)$

- Zheng et al. [2012], Blanche et al.[2013]
  $AUC(t) =$

$$\mathbb{P}(\widehat{Y}_i(\underline{\beta}) > \widehat{Y}_j(\underline{\beta}) | \underbrace{T_i \leq t, \eta_i = k}_{case}, \underbrace{\{(T_j > t) \cup (T_j < t, \eta_j \notin \{0, k\}}_{control}$$

## $Performance: \quad AUC_{IPCW}(E_k, f_k, \tau)$

- Zheng et al. [2012], Blanche et al.[2013]
  $AUC(t) =$
  $$\mathbb{P}(\widehat{Y}_i(\underline{\beta}) > \widehat{Y}_j(\underline{\beta}) | \underbrace{T_i \leq t, \eta_i = k}_{case}, \underbrace{\{(T_j > t) \cup (T_j < t, \eta_j \notin \{0, k\}}_{control}$$
- For evaluating the predictive performance (Blanche et al.[2013]) in the validation/test data $\widehat{AUC}_{IPCW}(\tau, \mathbf{E}_k, \widehat{\mathbf{Y}}(\underline{\beta}), \mathbf{w}^n)$

$$\frac{\sum_{i=1}^n \sum_{j=1}^n E_{k,i}\hat{w}_i^n (1 - E_{k,j})\hat{w}_j^n \mathbb{I}[\widehat{Y}_i(\underline{\beta}) > \widehat{Y}_j(\underline{\beta})]}{(\sum_{i=1}^n E_{k,i}\hat{w}_i^n)(\sum_{j=1}^n (1 - E_{k,j})\hat{w}_j^n)}$$

## $Performance: \quad AUC_{IPCW}(E_k, f_k, \tau)$

- Zheng et al. [2012], Blanche et al.[2013]

  $AUC(t) =$

  $$\mathbb{P}(\widehat{Y}_i(\underline{\beta}) > \widehat{Y}_j(\underline{\beta}) | \underbrace{T_i \leq t, \eta_i = k}_{case}, \underbrace{\{(T_j > t) \cup (T_j < t, \eta_j \notin \{0, k\}}_{control}$$

- For evaluating the predictive performance (Blanche et al.[2013]) in the validation/test data $\widehat{AUC}_{IPCW}(\tau, \mathbf{E}_k, \widehat{\mathbf{Y}}(\underline{\beta}), \mathbf{w}^n)$

  $$\frac{\sum_{i=1}^n \sum_{j=1}^n E_{k,i}\hat{w}_i^n(1 - E_{k,j})\hat{w}_j^n \mathbb{I}[\widehat{Y}_i(\underline{\beta}) > \widehat{Y}_j(\underline{\beta})]}{(\sum_{i=1}^n E_{k,i}\hat{w}_i^n)(\sum_{j=1}^n (1 - E_{k,j})\hat{w}_j^n)}$$

- For Selecting the optimal $\widehat{\underline{\beta}}$ (Fong et al. [2016])

$$\widehat{\underline{\beta}} = arg\min \left\{ \left(1 - \widehat{AUC}_{IPCW}(\tau, \mathbf{E}_k, \widehat{\mathbf{Y}}(\underline{\beta}^*), \hat{\mathbf{w}}^n)\right) + \lambda \sum_{a=1}^{A} \beta_a^{*2} \right\}$$

## *Simulation studies*

- Two simulation studies (different covariates used to generate the scale parameter) and four scenarios (A;B;C;D)

# *Simulation studies*

- Two simulation studies (different covariates used to generate the scale parameter) and four scenarios (A;B;C;D)
- $T_i$ event-time of type $\eta_i \in 1, 2$ and $C_i \sim$ Weibull distribution

# *Simulation studies*

- Two simulation studies (different covariates used to generate the scale parameter) and four scenarios (A;B;C;D)
- $T_i$ event-time of type $\eta_i \in 1, 2$ and $C_i \sim$ Weibull distribution
- $\mathbf{X}_i$: 20 covariates normal distributed and correlated each other

# *Simulation studies*

- Two simulation studies (different covariates used to generate the scale parameter) and four scenarios (A;B;C;D)
- $T_i$ event-time of type $\eta_i \in 1, 2$ and $C_i \sim$ Weibull distribution
- $\mathbf{X}_i$: 20 covariates normal distributed and correlated each other
- $E_{k,i}(\tau)$ was created for $k = 1$ and $\tau = 26.5$

# *Simulation studies*

- Two simulation studies (different covariates used to generate the scale parameter) and four scenarios (A;B;C;D)
- $T_i$ event-time of type $\eta_i \in 1, 2$ and $C_i \sim$ Weibull distribution
- $\mathbf{X}_i$: 20 covariates normal distributed and correlated each other
- $E_{k,i}(\tau)$ was created for $k = 1$ and $\tau = 26.5$
- 500 simulated data sets, sample size 1250 (training set 1000 obs and test set 250 obs )
- *Scenario A*.Independent Censoring (no covariate)
- *Scenario B*. Independent Censoring (disjoint subset of the covariates)

# *Simulation studies*

- Two simulation studies (different covariates used to generate the scale parameter) and four scenarios (A;B;C;D)
- $T_i$ event-time of type $\eta_i \in 1, 2$ and $C_i \sim$ Weibull distribution
- $\mathbf{X}_i$: 20 covariates normal distributed and correlated each other
- $E_{k,i}(\tau)$ was created for $k = 1$ and $\tau = 26.5$
- 500 simulated data sets, sample size 1250 (training set 1000 obs and test set 250 obs )
- *Scenario A*.Independent Censoring (no covariate)
- *Scenario B*. Independent Censoring (disjoint subset of the covariates)
  *Scenario C*. Dependent Censoring (same subset of the covariates as those associated with the failure outcome)
  *Scenario D*. Dependent Censoring (same subset of the covariates)

# *Simulation studies*

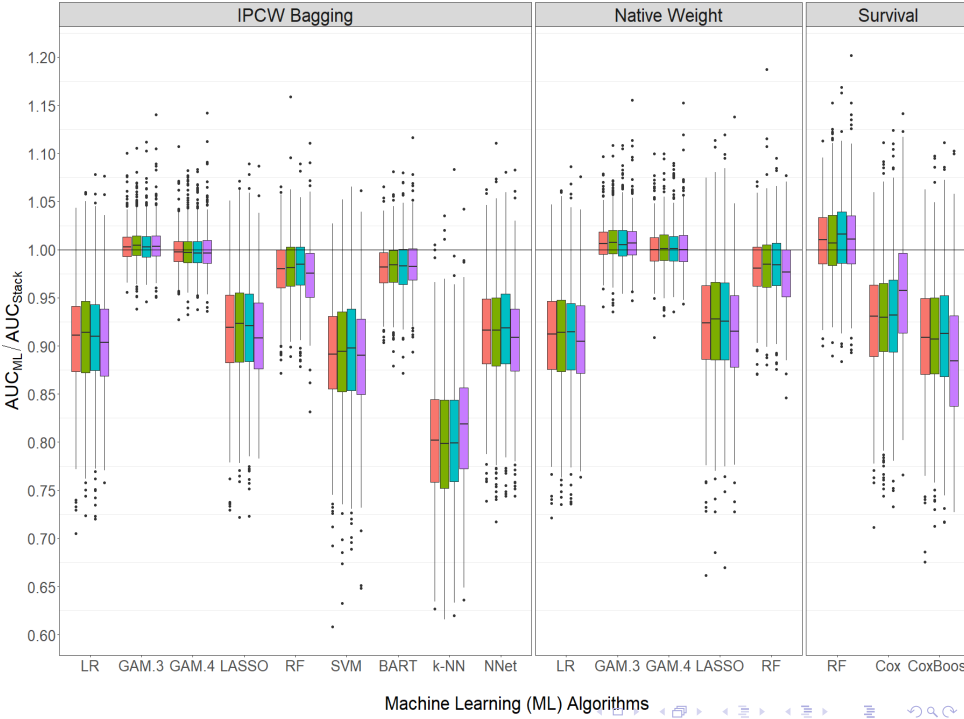
- We compared our IPCW Bagging procedure to those algorithms in the library that allow weights to be specified as an argument of the R function: logistic regression, GAM, LASSO and random forest (Native Weight)

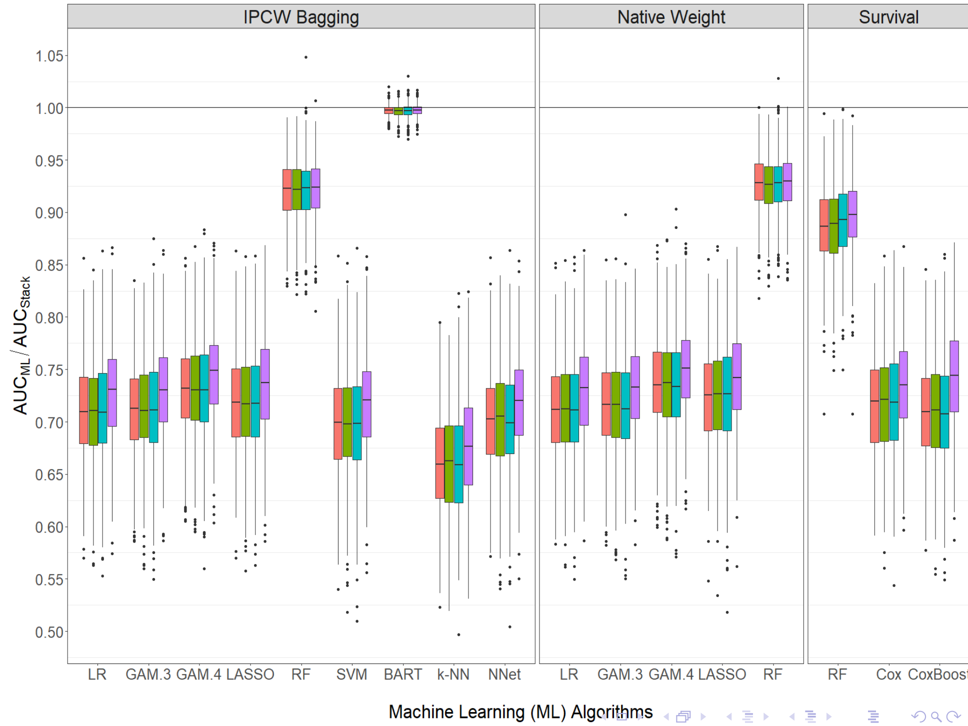
## *Simulation studies*

- We compared our IPCW Bagging procedure to those algorithms in the library that allow weights to be specified as an argument of the R function: logistic regression, GAM, LASSO and random forest (Native Weight)
- We also compare our procedure to three survival based methods:
  - cause-specific Cox proportional hazard model-based method (Ozenne et al., 2017)
  - Cox-Boost (Binder, 2013), which is a sub-distribution hazard model
  - and survival random forests for competing risk (Ishwaran and Kogalur, 2020)

# *Simulation studies*

- We compared our IPCW Bagging procedure to those algorithms in the library that allow weights to be specified as an argument of the R function: logistic regression, GAM, LASSO and random forest (Native Weight)
- We also compare our procedure to three survival based methods:
  - cause-specific Cox proportional hazard model-based method (Ozenne et al., 2017)
  - Cox-Boost (Binder, 2013), which is a sub-distribution hazard model
  - and survival random forests for competing risk (Ishwaran and Kogalur, 2020)


IPCW Bagging
Native Weight
Survival
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
$AUC_{ML} / AUC_{Stack}$
LR
GAM.3
GAM.4
LASSO
RF
SVM
BART
k-NN
NNet
LR
GAM.3
GAM.4
LASSO
RF
RF
Cox
CoxBoos
Machine Learning (ML) Algorithms



IPCW Bagging
Native Weight
Survival
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
$AUC_{ML} / AUC_{Stack}$
LR
GAM.3
GAM.4
LASSO
RF
SVM
BART
k-NN
NNet
LR
GAM.3
GAM.4
LASSO
RF
RF
Cox
CoxBoost
Machine Learning (ML) Algorithms

## *Real Data Application: InfCare HIV Register*

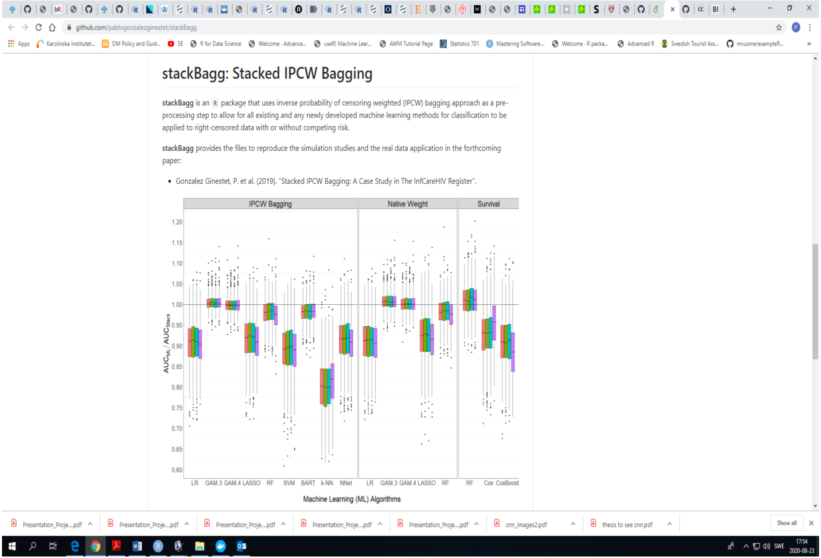
- Event of interest: failure of treatment to keep viral load below 50m/L (undetectable) within 2 years
- Final data consisted of 3,114 subjects who were diagnosed with HIV since 2009 and have at least some record of being suppressed
- 1,726 subjects did not have a record of death or a HIV RNA measurement$< 50$ copies/mL within two years from their date of first suppression ($E_{1,i}(2) = 0$)
- 17 subjects died within two years ($E_{1,i}(2) = 0$)
- 764 subjects experienced a rebound in their viral load within two years of suppression ($E_{1,i}(2) = 1$)
- 624 subjects were censored prior to two years after their first suppression and thus their outcomes were not known ($E_{1,i}(2) = NA$)
- Predictor variables used: age, gender, ethnicity, immigration status, infection route, number of languages spoken, if the subject has an infection (e.g., pneumonia), HIV medications.

# *Real Data Application: InfCare HIV Register*

**TABLE 2** Estimated AUCs for predicting a rebound in viral load in split sample test set based on Cox-PH-weig

| ML Algorithm | R package | IPCW Bagging | Native weight | Survival |
|---|---|---|---|---|
| Logistic Regression | stats::glm | 0.601 | 0.574 | |
| GAM | gam::gam | 0.575 | 0.585 | |
| LASSO | glmnet::glmnet | 0.572 | 0.599 | |
| Random Forest | ranger::ranger | 0.539 | 0.599 | 0.554 |
| SVM | e1071::svm | 0.603 | | |
| BART | bartMachine::bartMachine | 0.598 | | |
| k-NN | class::knn | 0.602 | | |
| Neural Network | neuralnet::neuralnet | 0.603 | | |
| Stacked IPCW bagging | | 0.602 | | |
| Cox-PH | riskRegression::CSC | | | 0.586 |
| Cox-Boost | CoxBoost::CoxBoost | | | 0.587 |

# *R package on GitHub:* *github.com/pablogonzalezginestet/stackBagg* ▸ *Link*



# *Conclusions*

- We generalized IPCW bagging approach to arbitrary algorithms (Hothorn et al. [2006] limited to random forest and boosting)

# *Conclusions*

- We generalized IPCW bagging approach to arbitrary algorithms (Hothorn et al. [2006] limited to random forest and boosting)
- We extendeded the IPCW bagging method to allow for the possibility of competing risks

# *Conclusions*

- We generalized IPCW bagging approach to arbitrary algorithms (Hothorn et al. [2006] limited to random forest and boosting)
- We extendeded the IPCW bagging method to allow for the possibility of competing risks
- We explored IPCW bagging in cases of dependent censoring (under CAR assumption)

# Conclusions

- We generalized IPCW bagging approach to arbitrary algorithms (Hothorn et al. [2006] limited to random forest and boosting)
- We extendeded the IPCW bagging method to allow for the possibility of competing risks
- We explored IPCW bagging in cases of dependent censoring (under CAR assumption)
- We suggested a procedure to optimally stack predictions from a set of ML procedures using as our target loss function the area under the time-dependent ROC curve (AUC) following the Super Learner technique (VanderLaan et al.[2007])

# *Conclusions*

- We generalized IPCW bagging approach to arbitrary algorithms (Hothorn et al. [2006] limited to random forest and boosting)
- We extendeded the IPCW bagging method to allow for the possibility of competing risks
- We explored IPCW bagging in cases of dependent censoring (under CAR assumption)
- We suggested a procedure to optimally stack predictions from a set of ML procedures using as our target loss function the area under the time-dependent ROC curve (AUC) following the Super Learner technique (VanderLaan et al.[2007])
- Computationally intensive limiting the the algorithms to add in the library

# *Conclusions*

- We generalized IPCW bagging approach to arbitrary algorithms (Hothorn et al. [2006] limited to random forest and boosting)
- We extendeded the IPCW bagging method to allow for the possibility of competing risks
- We explored IPCW bagging in cases of dependent censoring (under CAR assumption)
- We suggested a procedure to optimally stack predictions from a set of ML procedures using as our target loss function the area under the time-dependent ROC curve (AUC) following the Super Learner technique (VanderLaan et al.[2007])
- Computationally intensive limiting the the algorithms to add in the library
- We have also developed an R package (installed via the command remotes::install_github("pablogonzalezginestet/stackBagg").

KAROLINSKA INSTITUTET
ANNO 1810

Karolinska
Institutet

*Thank You!*

# *Simulation studies. Shape and scale parameter*

Shape parameter: main event type is 3.5, competing event type is 2.5, and is 1 for the censoring time.

**TABLE 1** List of covariates and transformations used to generate the scale parameter of the main event time ($\eta$ = 1), competing event time ($\eta$ = 2) and censoring time in each simulation and scenario †.

| (sim,scen) | $X_{T,\eta=1}$ | $X_{T,\eta=2}$ | $X_C$ |
|---|---|---|---|
| 1, A | $X_1, X_{16}, X_1X_{16}, bs(X_{11}), bs(X_{16})$ | $X_2$ | No covariate |
| 1, B | same as A | same as A | $X_3, X_4, X_9, X_{19}, X_{20}$ |
| 1, C | same as A | same as A | $X_1, X_6, X_{11}, X_{16}, X_{20}$ |
| 1, D | same as A | $X_1, X_6, X_{11}, X_{16}, X_{20}$ | same as C |
| 2, A | $X_1, X_6, X_{11}, X_{16}, X_{20}, \cos(X_{16})/.1, X_1X_6, X_{20}I(X_{20} > median(X_{20}))$ | $X_2$ | No covariate |
| 2, B | same as A | same as A | $I(X_3 > median(X_3)), X_4, X_9, cos(X_{19})/.7$ |
| 2, C | same as A | same as A | $I(X_1 > median(X_1)), X_6, X_{11}, cos(X_{16})/.7, X_{20}$ |
| 2, D | same as A | $X_1, X_6, X_{11}, X_{16}, X_{20}$ | same as C |

† $(sim, scen)$ denotes the pair simulation and scenario; $bs()$ denotes the basis matrix of cubic-spline function; cos() denotes cosine function and $I()$ denotes the indicator function.

# *Simulation studies*

- $\mathcal{L} = \{$ logistic regression (LR); GAM df$=$ 3, 4; LASSO; random forest (RF); SVM; bayesian additive regression trees (BART); k-nearest neighbors and neural networks$\}$
- $B = 10$ and 5-fold cross-validation
- grid of values for $\lambda \in \{0.01, 0.1, 0.5, 1, 5, 10, 15, 25, 50, 100\}$
- stacking function for ensemble: linear
- We compare IPCW Bagging with Native weight methods (LR, GAM, LASSO and RF) and survival based methods (CoxPH, CoxBoost, RF for competing risk).
- CoxPH and CoxBoost (not shown here) weights were used

# *Real Data Application: InfCare HIV Register. Tuning Parameter*

The following table shows the values of the tuning parameter that were selected among a grid search using 5-fold cross validation on the training dataset.

TABLE: Tuning parameter selected using 5-fold cross validation on the training set for the InfCare HIV Registry.

| ML algorithm | Tuning parameter |
|---|---|
| Logistic Regression | $NA$ |
| Generalized Additive Models | $df = 3$ |
| LASSO | $\lambda_{CV} = 0.01685$ |
| Random Forest | $num.tree = 50$ , $mtry = 1$ |
| Support vector machine | $cost = 100$, $kernel = radial$,$gamma = 0.1$ |
| Bayesian additive regression trees | $num\_trees = 50$,$k = 2, q = 0.9$ |
| k-nearest neighbors | $k = 43$ |
| Neural Network | $hidden = 2$ |

**TABLE 3** Average Estimated AUCs across 500 data sets for Simulation 1 and 2 and their four scenarios (A, B, C and D) using all available covariates and a Cox-PH model for censoring for predicting the event of interest in the test data set.

| ML Algorithm | Simulation 1 | | | | | | | | Simulation 2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | A* | B | B* | C | C* | D | D* | A | A* | B | B* | C | C* | D | D* |
| True | 0.761 | | 0.761 | | 0.761 | | 0.781 | | 0.928 | | 0.928 | | 0.928 | | 0.940 | |
| *IPCW Bagging* | | | | | | | | | | | | | | | | |
| LogReg | 0.647 | 0.648 | 0.648 | 0.647 | 0.646 | 0.647 | 0.651 | 0.650 | 0.631 | 0.631 | 0.630 | 0.630 | 0.631 | 0.630 | 0.656 | 0.655 |
| GAM.3 | 0.718 | 0.718 | 0.718 | 0.717 | 0.715 | 0.715 | 0.725 | 0.725 | 0.635 | 0.634 | 0.633 | 0.632 | 0.635 | 0.633 | 0.659 | 0.657 |
| GAM.4 | 0.714 | 0.714 | 0.713 | 0.712 | 0.711 | 0.711 | 0.721 | 0.721 | 0.651 | 0.650 | 0.649 | 0.649 | 0.651 | 0.649 | 0.673 | 0.672 |
| LASSO | 0.654 | 0.655 | 0.655 | 0.655 | 0.654 | 0.654 | 0.656 | 0.655 | 0.639 | 0.638 | 0.637 | 0.637 | 0.639 | 0.637 | 0.663 | 0.662 |
| RF | 0.700 | 0.700 | 0.700 | 0.699 | 0.700 | 0.699 | 0.702 | 0.701 | 0.820 | 0.821 | 0.818 | 0.818 | 0.819 | 0.818 | 0.832 | 0.830 |
| SVM | 0.635 | 0.636 | 0.635 | 0.635 | 0.635 | 0.636 | 0.639 | 0.639 | 0.620 | 0.620 | 0.620 | 0.620 | 0.620 | 0.618 | 0.647 | 0.645 |
| BART | 0.702 | 0.701 | 0.702 | 0.701 | 0.700 | 0.700 | 0.710 | 0.709 | 0.888 | 0.889 | 0.886 | 0.887 | 0.887 | 0.886 | 0.899 | 0.898 |
| k-NN | 0.572 | 0.573 | 0.570 | 0.571 | 0.570 | 0.570 | 0.587 | 0.587 | 0.588 | 0.588 | 0.586 | 0.586 | 0.586 | 0.587 | 0.609 | 0.610 |
| NNet | 0.654 | 0.655 | 0.651 | 0.652 | 0.652 | 0.652 | 0.652 | 0.651 | 0.625 | 0.624 | 0.624 | 0.623 | 0.623 | 0.622 | 0.647 | 0.644 |
| Stacked IPCW bagging | 0.715 | 0.715 | 0.714 | 0.713 | 0.712 | 0.712 | 0.721 | 0.721 | 0.891 | 0.891 | 0.889 | 0.890 | 0.889 | 0.889 | 0.902 | 0.902 |
| *Native Weight* | | | | | | | | | | | | | | | | |
| LogReg | 0.649 | 0.650 | 0.649 | 0.648 | 0.647 | 0.648 | 0.653 | 0.652 | 0.633 | 0.633 | 0.632 | 0.632 | 0.633 | 0.632 | 0.658 | 0.656 |
| GAM.3 | 0.720 | 0.720 | 0.720 | 0.718 | 0.717 | 0.718 | 0.727 | 0.727 | 0.638 | 0.637 | 0.636 | 0.635 | 0.636 | 0.635 | 0.661 | 0.659 |
| GAM.4 | 0.716 | 0.716 | 0.715 | 0.714 | 0.713 | 0.713 | 0.723 | 0.723 | 0.655 | 0.655 | 0.653 | 0.653 | 0.653 | 0.652 | 0.677 | 0.675 |
| LASSO | 0.659 | 0.660 | 0.659 | 0.659 | 0.659 | 0.659 | 0.659 | 0.658 | 0.644 | 0.644 | 0.643 | 0.643 | 0.645 | 0.643 | 0.670 | 0.667 |
| RF | 0.701 | 0.701 | 0.702 | 0.700 | 0.701 | 0.700 | 0.704 | 0.702 | 0.826 | 0.826 | 0.822 | 0.822 | 0.823 | 0.822 | 0.837 | 0.835 |
| *Survival* | | | | | | | | | | | | | | | | |
| Cox-PH | 0.661 | 0.662 | 0.661 | 0.661 | 0.662 | 0.662 | 0.688 | 0.687 | 0.638 | 0.638 | 0.637 | 0.637 | 0.637 | 0.636 | 0.662 | 0.660 |
| Cox-Boost | 0.649 | 0.649 | 0.648 | 0.647 | 0.648 | 0.648 | 0.637 | 0.634 | 0.631 | 0.631 | 0.630 | 0.631 | 0.628 | 0.628 | 0.669 | 0.668 |
| RF | 0.721 | 0.721 | 0.721 | 0.720 | 0.721 | 0.720 | 0.729 | 0.727 | 0.790 | 0.790 | 0.787 | 0.788 | 0.793 | 0.791 | 0.808 | 0.806 |