# S&DS 355/555
# Midterm 1 Review Session

Wendy Luo
wenjing.luo@yale.edu

## Supervised learning: Given X, predict y

**Regression: Predict values**

- Linear regression
- KNN regression
- Random forest regression

**Classification: Predict labels**

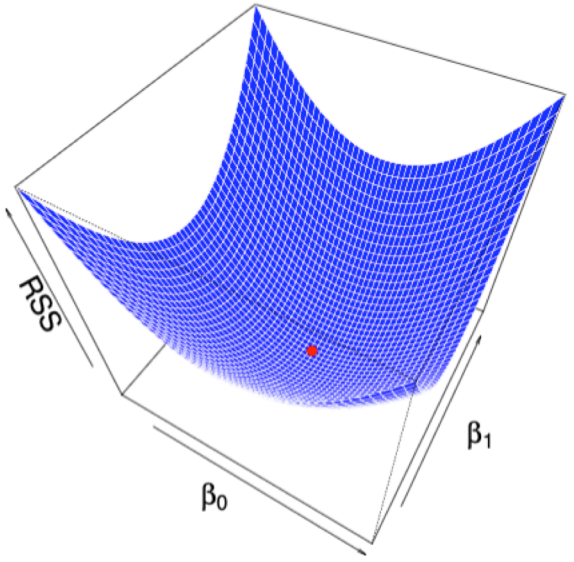
- Logistic regression
- KNN classification
- Random forest classification

## Unsupervised learning: Learn underlying structure or relationship in X

# Linear regression

**Criterion:** Least square, i.e. minimize residual sum of squares (RSS)

$$RSS = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$




**Coefficient estimation (model fitting) approach:**

- Gradient descent (stochastic gradient descent)

- Closed-form solution (matrix expression)

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ 1 & x_{2,1} & \ddots & & x_{2,p} \\ \vdots & & \ddots & \vdots & \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

$$y = X\beta + \epsilon$$

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

**Linear regression inference:**

Randomness in sampling -> variance in coefficient estimations

Standard errors of the coefficients describe how the coefficients vary under repeated sampling

**Sums of squares and $R^2$:**

Partitioning the sums of squares:

$$\underbrace{\sum (y_i - \bar{y})^2}_{\text{total sum of squares}(TSS)} = \underbrace{\sum (\hat{y}_i - \bar{y})^2}_{\text{explained sum of squares}(ESS)} + \underbrace{\sum (y_i - \hat{y}_i)^2}_{\text{residual sum of squares}(RSS)}$$

for least squares linear regression (some algebra shows this)

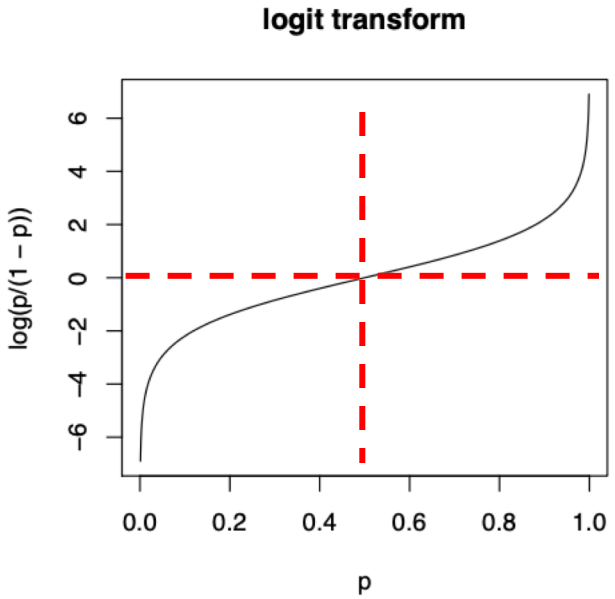
$$R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}$$

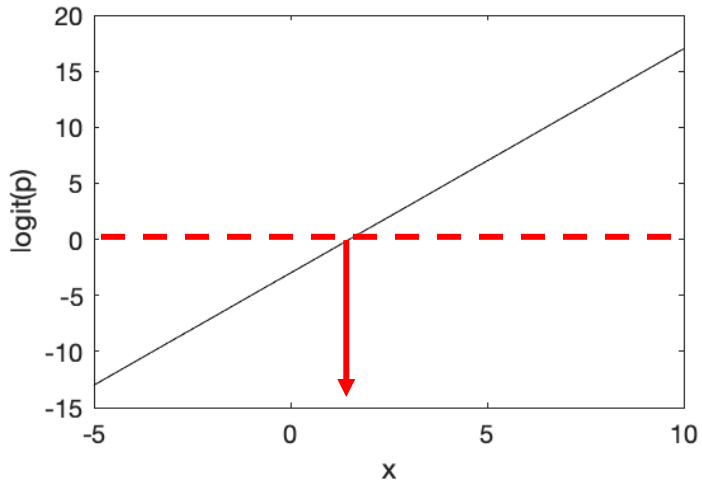
We can interpret $R^2$ (**multiple R-squared**) as the proportion of variability in $y$ explained by the model.

# Logistic regression

**Concepts:**

Linear discriminant function; Decision boundary; Classification risk/error rate; Empirical classification error/training error; Test error rate; Bayes risk; Bayes decision boundary; odds



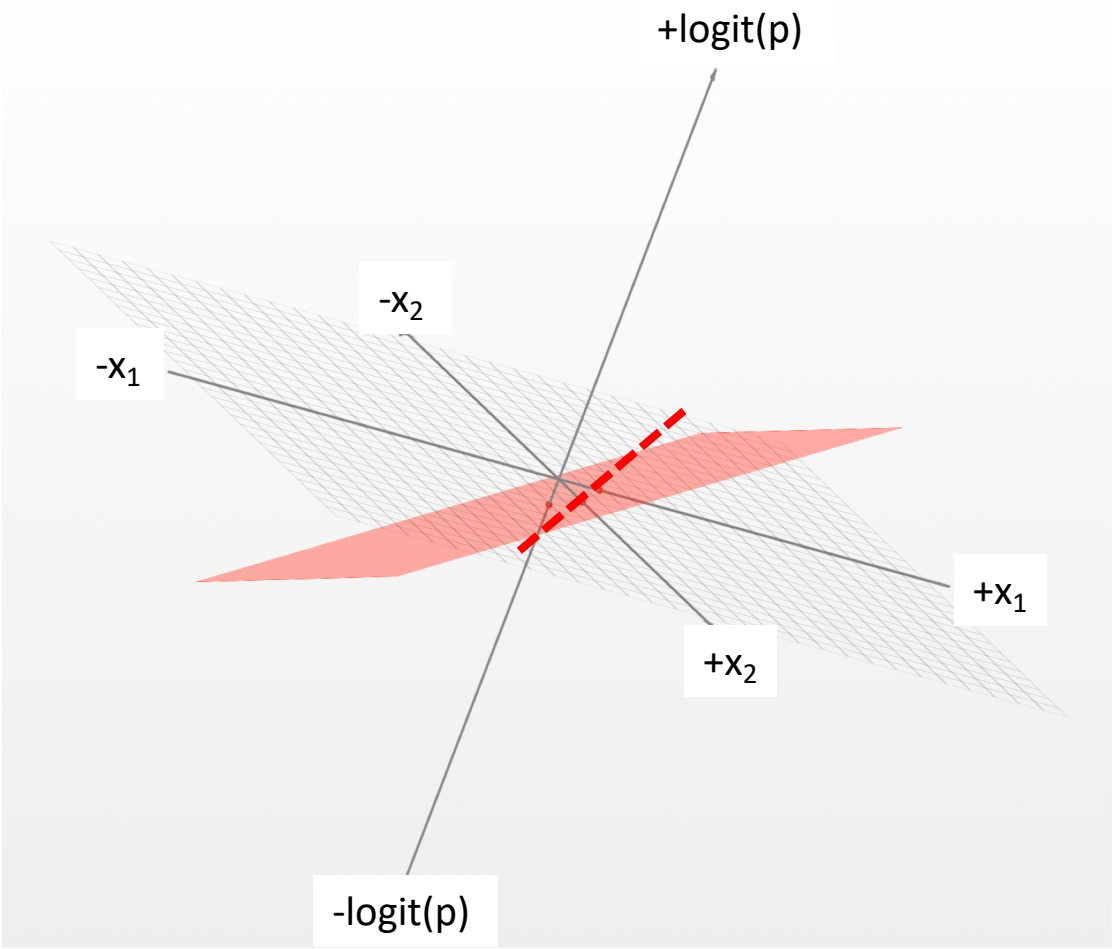

$$logit(p) = \log\left(\frac{p}{1-p}\right)$$

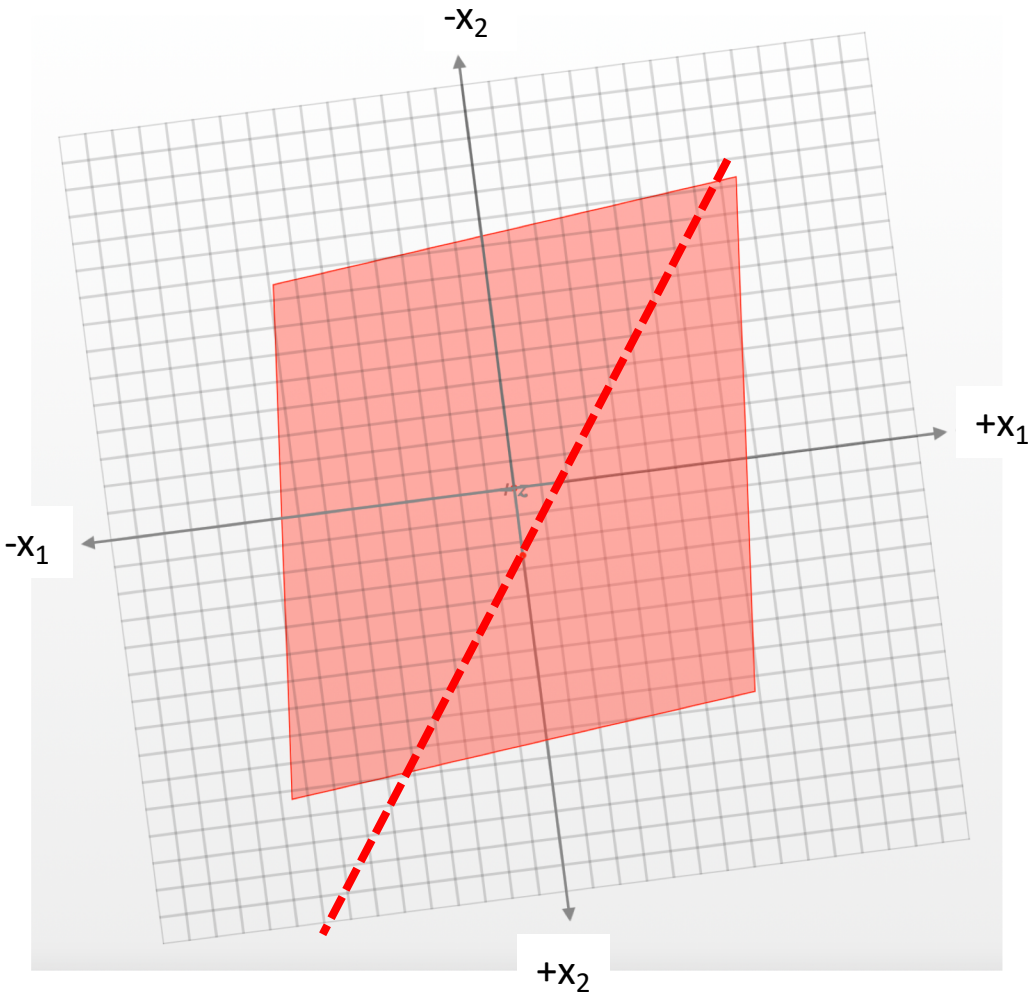
$$logit(\hat{p}(x)) = \hat{\beta}_0 + \hat{\beta}_1 x$$

$$\hat{p}(x) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 x}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 x}} = logistic(x^T\hat{\beta}) \equiv softmax(x^T\hat{\beta})$$

Decision boundary: p = 0.5; logit(p) = 0; linear boundary in X: $x = -\frac{\hat{\beta}_0}{\hat{\beta}_1}$

Decision boundary: p = 0.5; logit(p) = 0; linear boundary when there are two predictors $X_1$ and $X_2$




Decision boundary: logit(p) = 0; linear boundary in X: $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0$

**Coefficient estimation (model fitting) approach:**

Maximum likelihood estimation:

- Likelihood of a single observation $(x_i, y_i)$:

$$L_i(\beta) = p_i^{y_i} \cdot (1 - p_i)^{1-y_i} = \left( \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}} \right)^{y_i} \cdot \left( 1 - \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}} \right)^{1-y_i}$$

- Log-likelihood of a single observation:

$$\begin{aligned} \ell_i(\beta) &= y_i \log \left( \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}} \right) + (1 - y_i) \log \left( \frac{1}{1 + e^{x_i^T \beta}} \right) \\ &= y_i x_i^T \beta - \log(1 + e^{x_i^T \beta}) \end{aligned}$$

- Aggregate log-likelihood:

$$\ell(\beta) = \sum \left( y_i x_i^T \beta - \log(1 + e^{x_i^T \beta}) \right)$$

No closed-form solution; use gradient descent/stochastic gradient descent to find coefficients that maximize the aggregate log-likelihood

**Discriminative model:** model only the output y given the input X

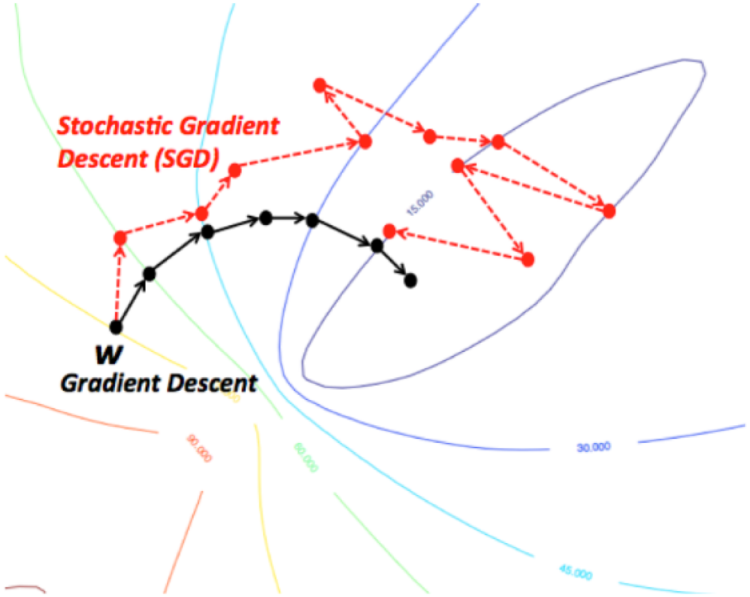
- Estimate by maximizing the conditional likelihood p(y | X)
- E.g. Logistic regression

**Generative model:** model both the input X and the output y

- Estimated by maximizing the joint likelihood p(y , X)
- E.g. Gaussian discriminant analysis (Linear or Quadratic)

**Batch gradient descent:**

1. Read all data points X
2. Make a prediction
3. Observe the true response/label y
4. Update the parameter to minimize the loss




**Stochastic gradient descent:**

1. Read only one data points $x_i$ or a small set of x
2. Make a prediction
3. Observe the true response/label $y_i$
4. Update the parameter to minimize the loss

**Advantage:**

- More efficient and cheap especially for big model and dataset with large n and p
- Fit the model in an online manner as data stream in

**Disadvantage:**

- May take more steps to converge due to randomness in sampling

**Coefficient update:**

SGD Update:

$$\beta_j \longleftarrow \beta_j + \delta(y - p(x))x_j$$

$$\beta_j x_j \longleftarrow \beta_j x_j + \delta(y - p(x))x_j^2$$

$$p(x) = \frac{1}{1 + \exp(-\beta^T x)}$$

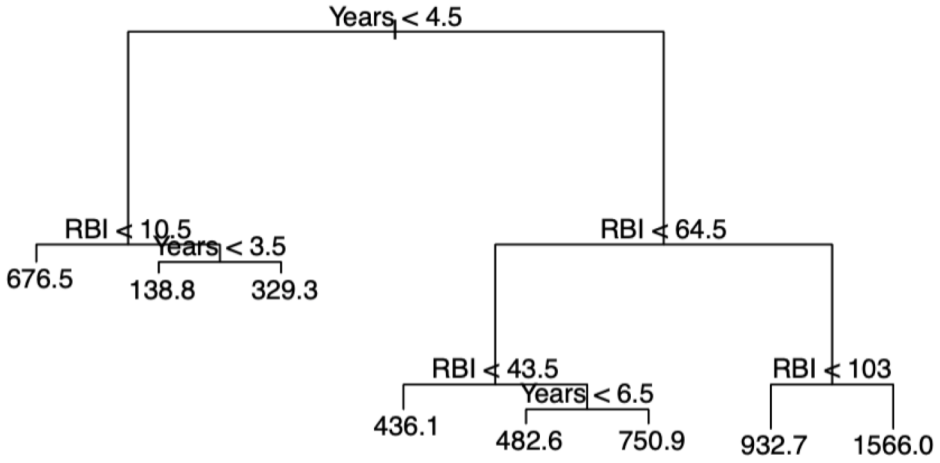
Case checking:

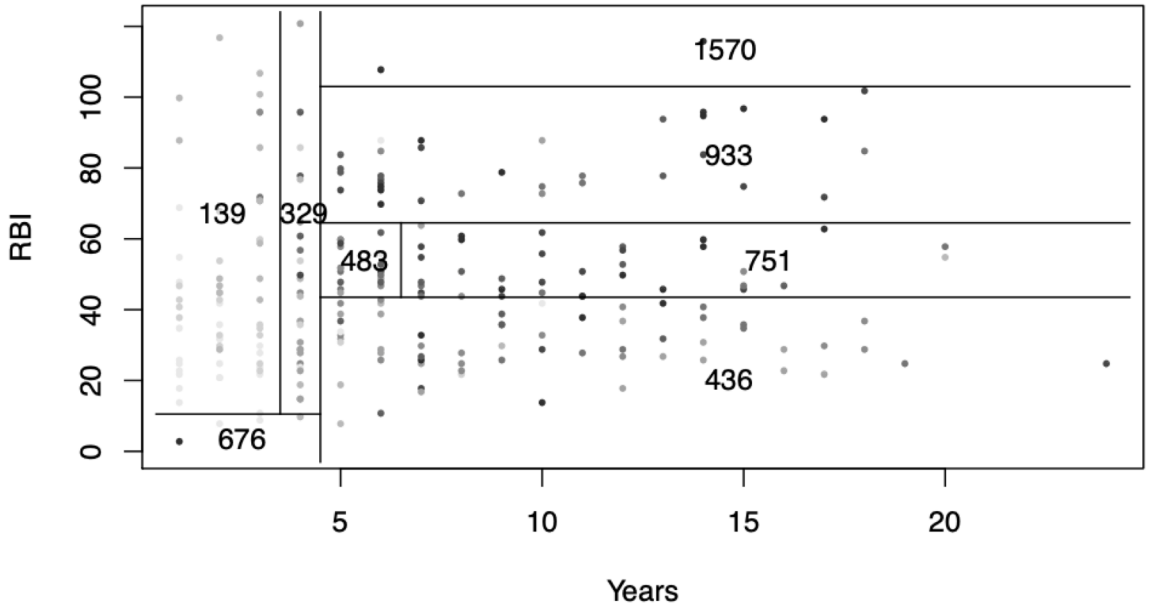
- Suppose $y = 1$ and probability $p(x)$ is high? *small change*
- Suppose $y = 1$ and probability $p(x)$ is small? *big change* ↑
- Suppose $y = 0$ and probability $p(x)$ is small? *small change*
- Suppose $y = 0$ and probability $p(x)$ is big? *big change* ↓

**Choice of learning rate:**

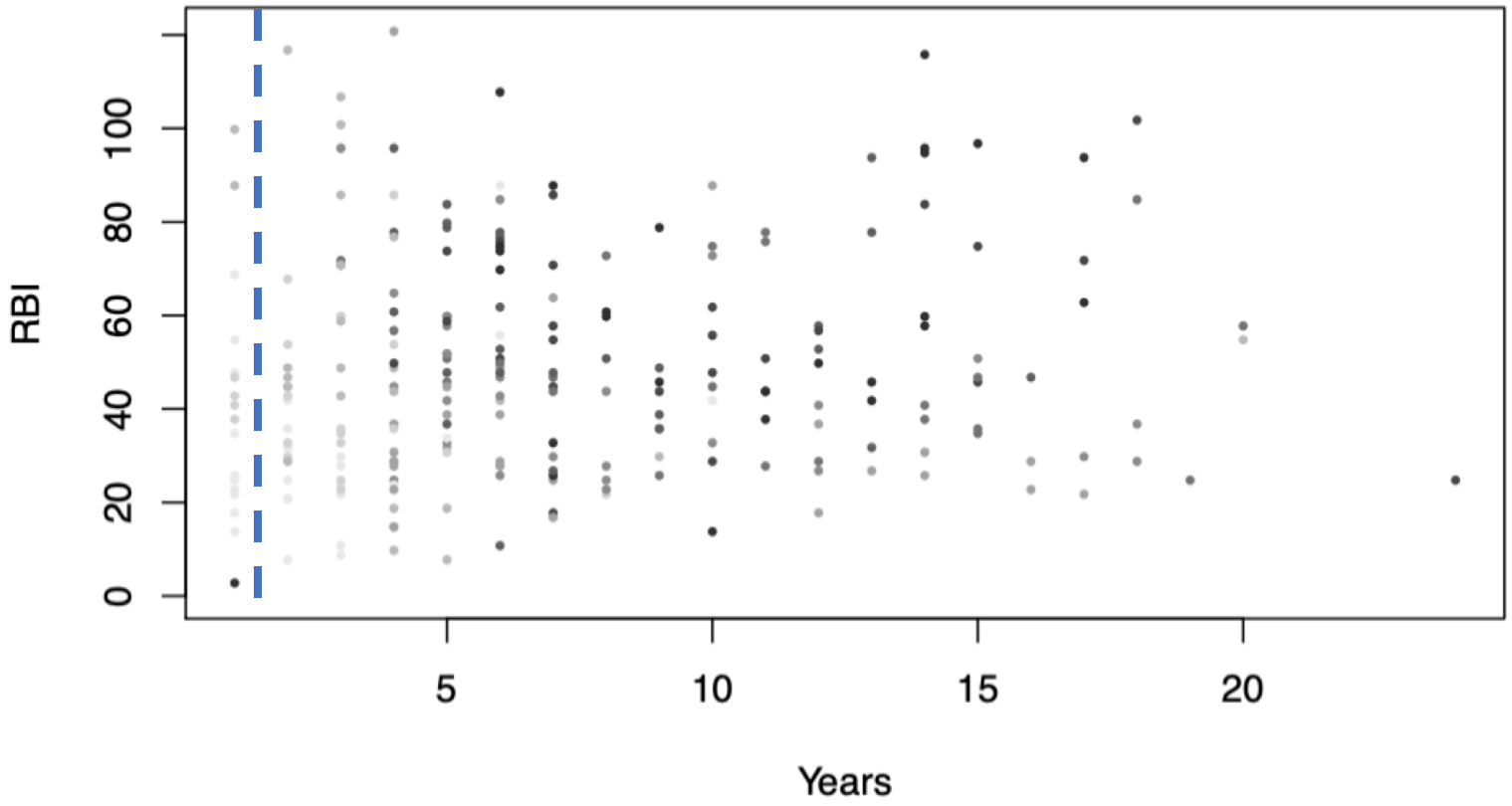
- Conservative: $\delta_t = \frac{1}{t}$
- More aggressive: $\delta_t = \frac{1}{\sqrt{t}}$
- Other rules
- General principle: decrease the learning rate/step size as we approach the true coefficients

# Trees

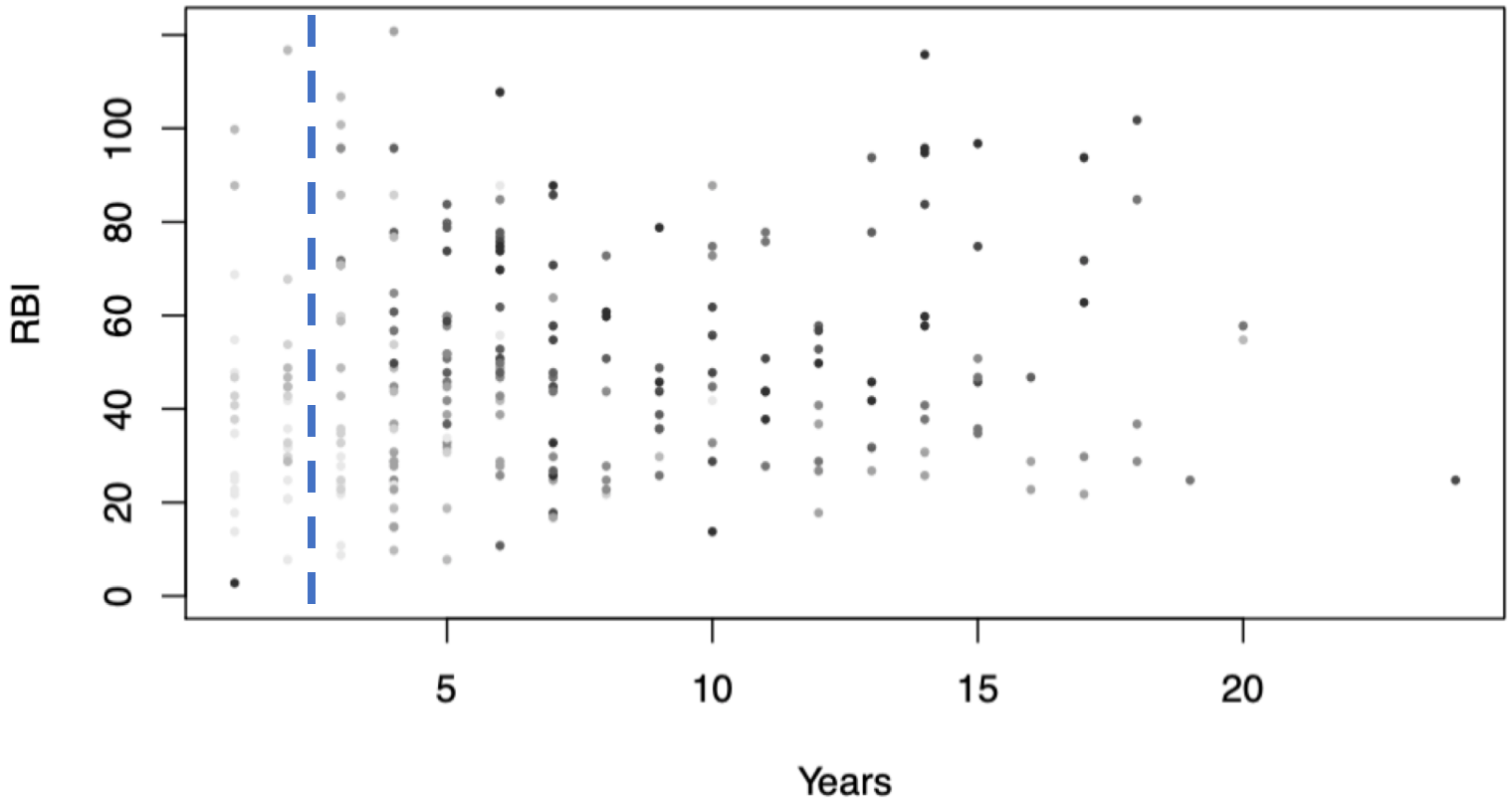

Years < 4.5
RBI < 10.5
Years < 3.5
676.5
138.8
329.3
RBI < 64.5
RBI < 43.5
Years < 6.5
RBI < 103
436.1
482.6
750.9
932.7
1566.0


RBI
Years
0
20
40
60
80
100
5
10
15
20
1570
933
139
329
483
751
436
676

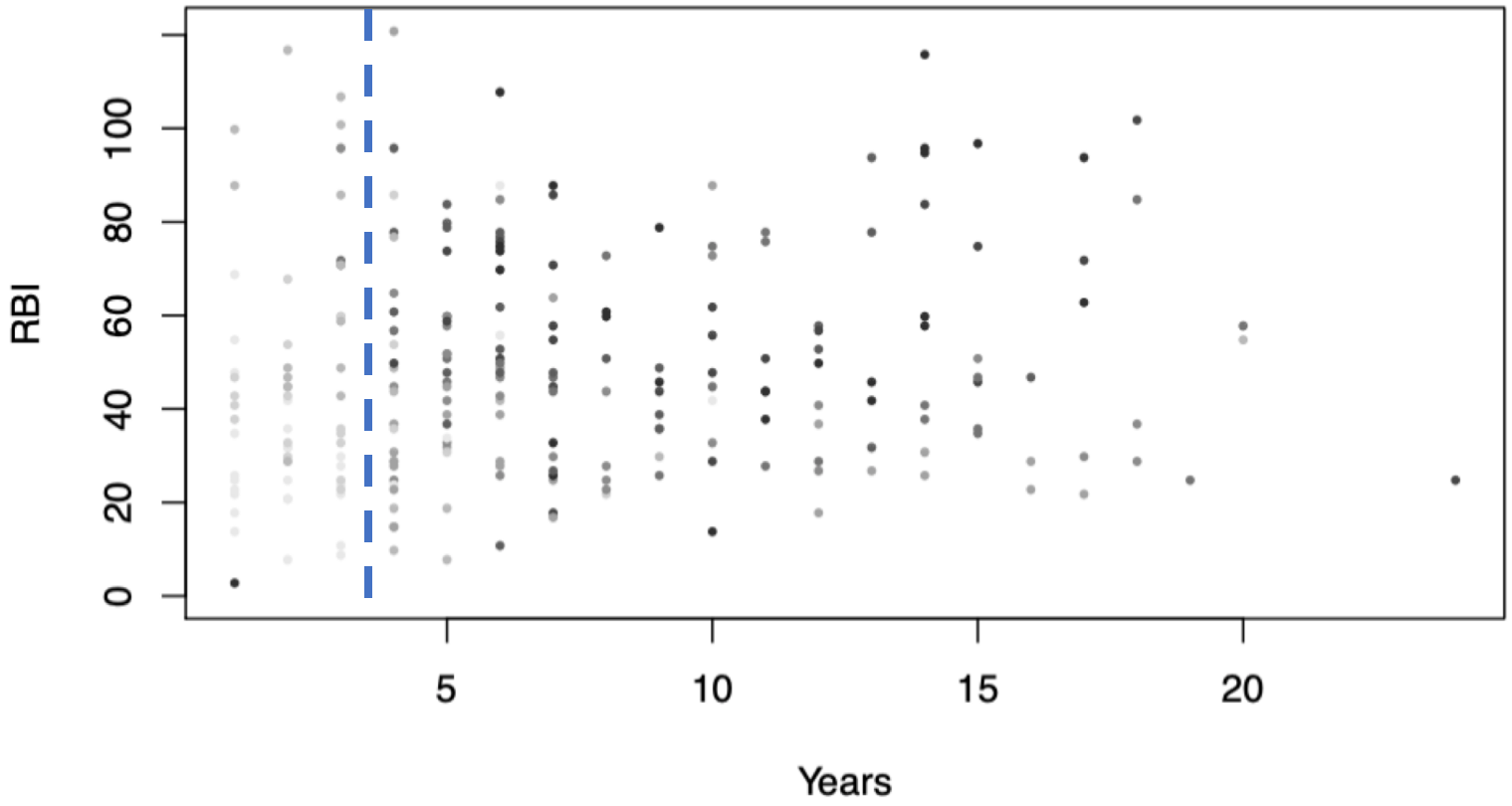
**Tree growth:**



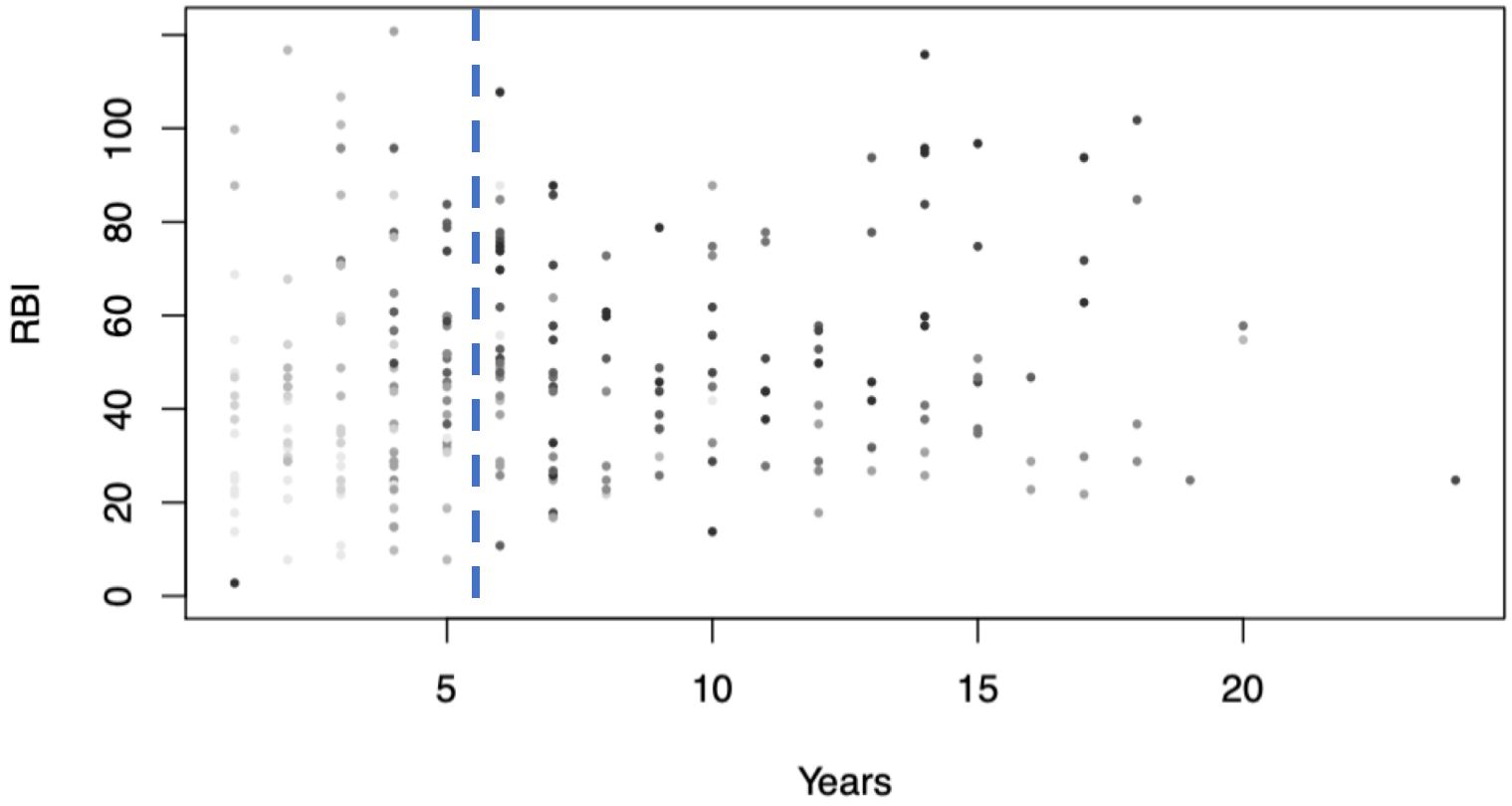
**Tree growth:**



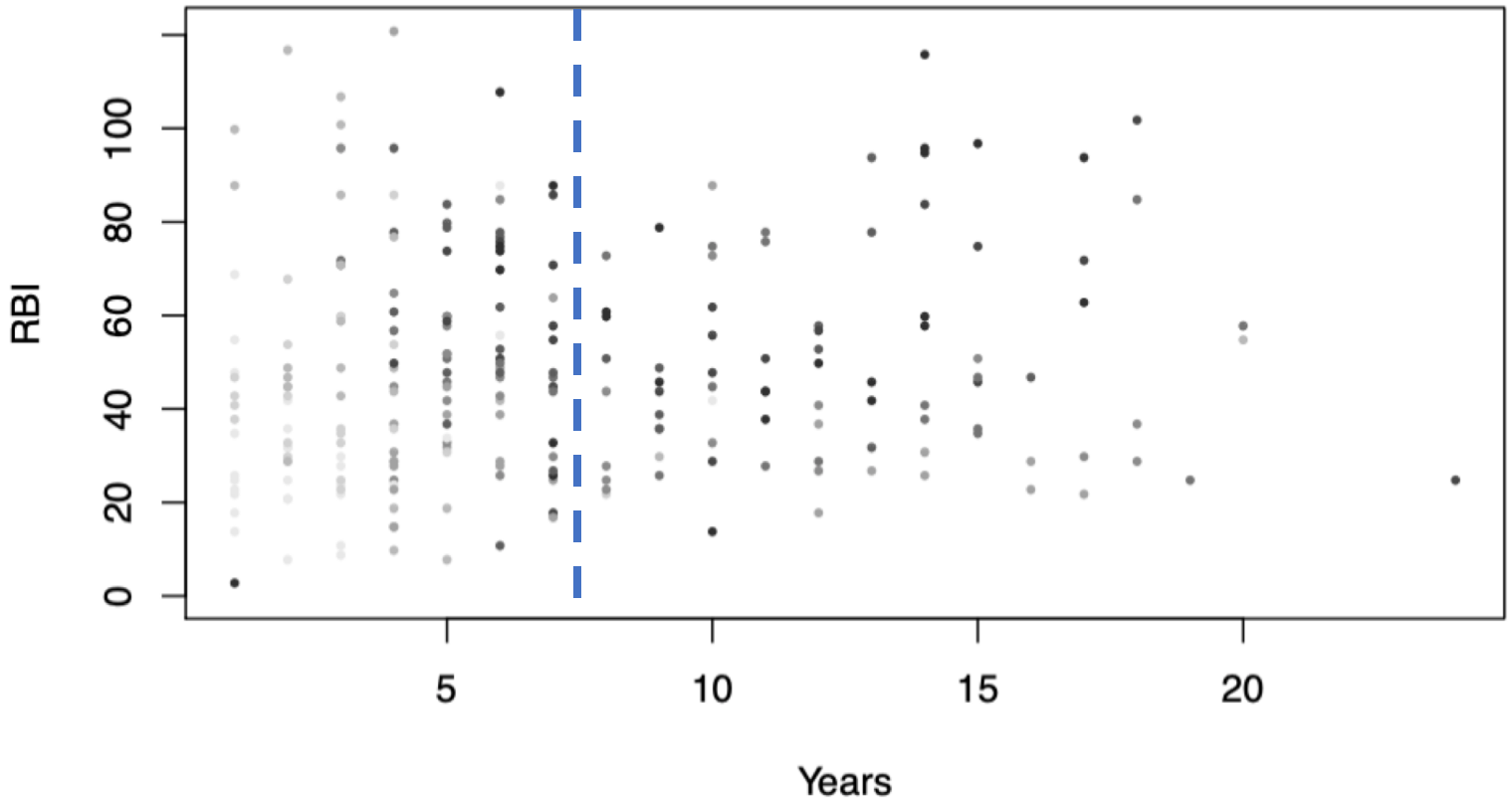
**Tree growth:**



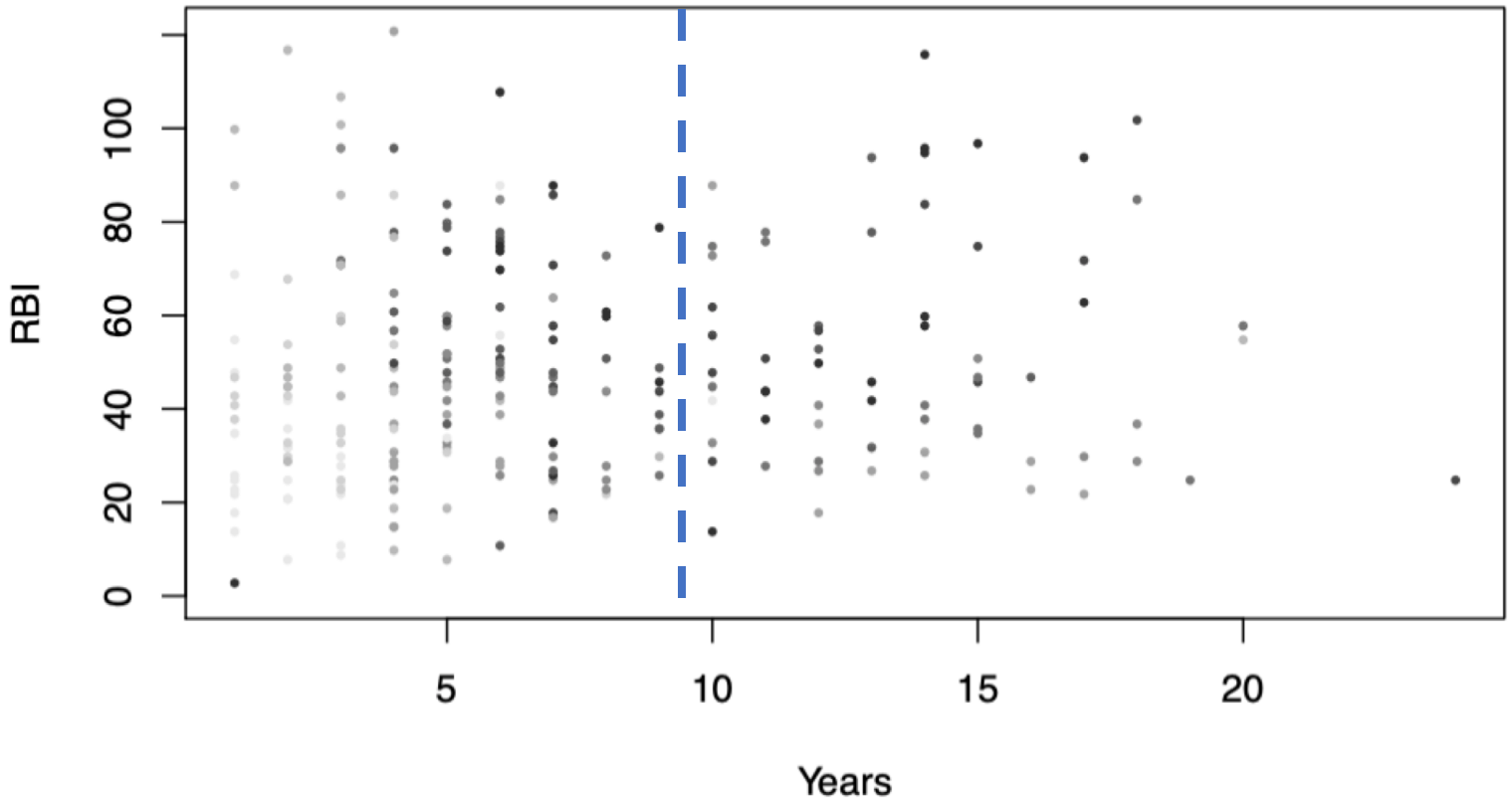
**Tree growth:**



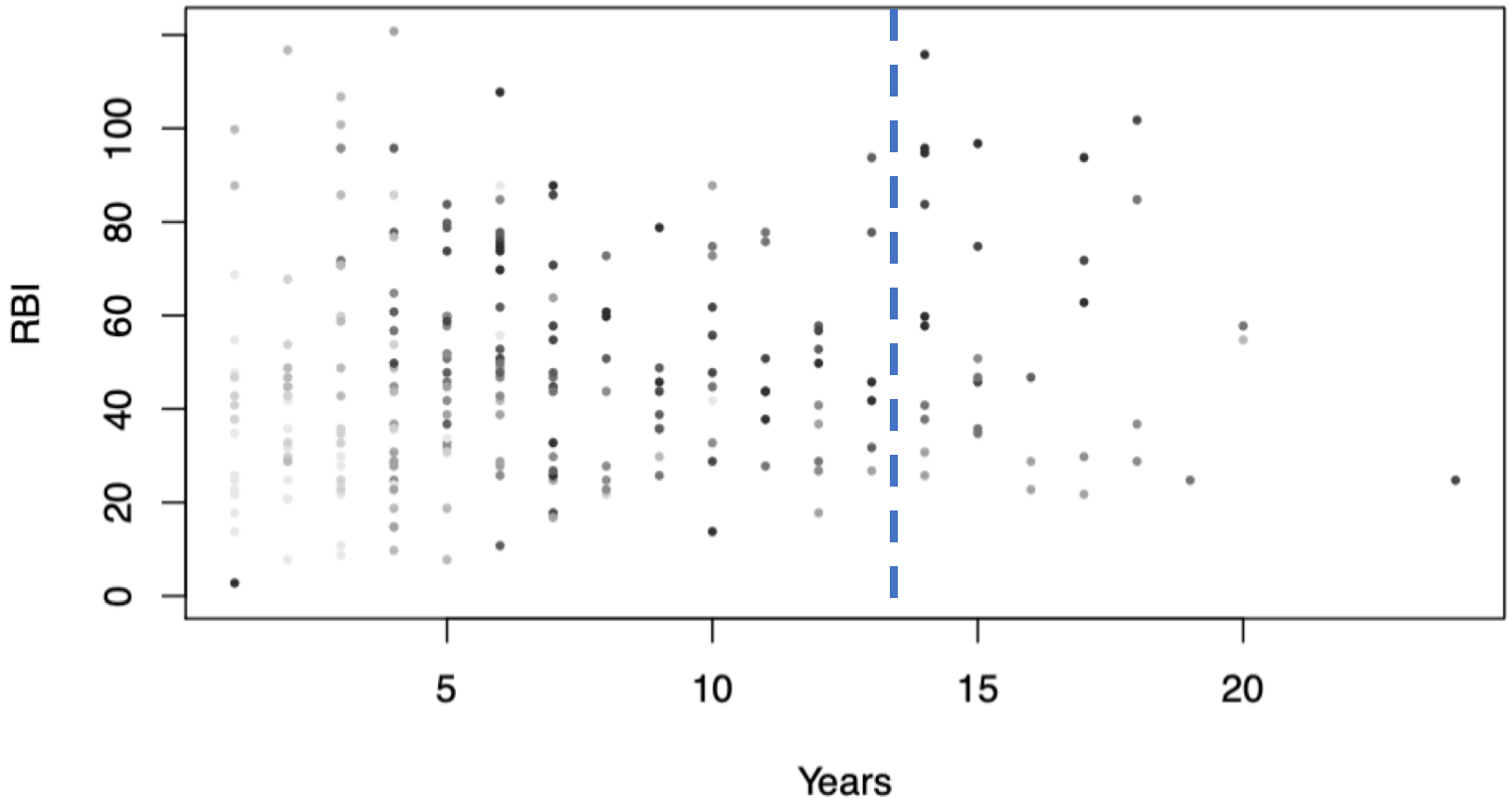
**Tree growth:**



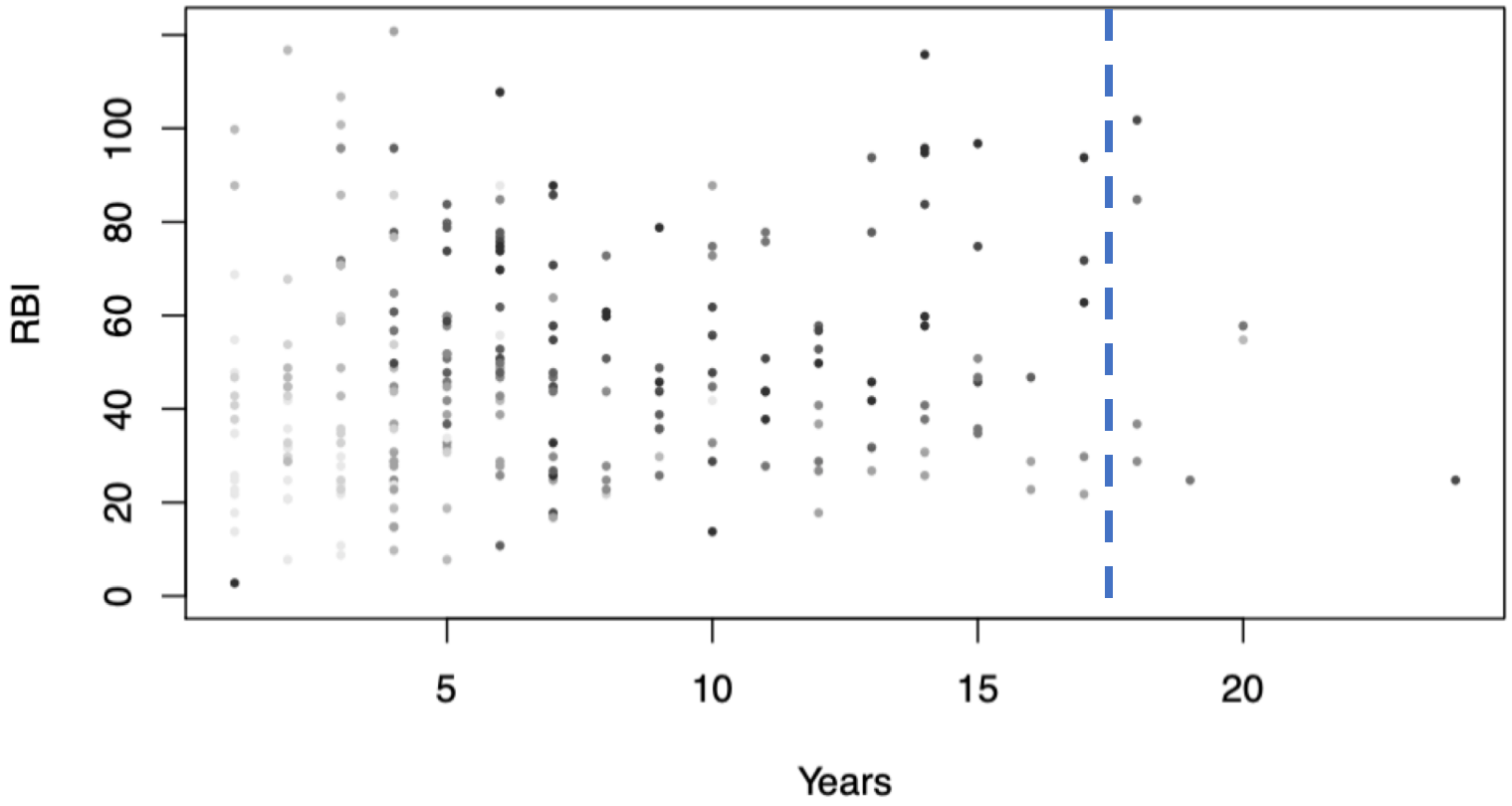
**Tree growth:**



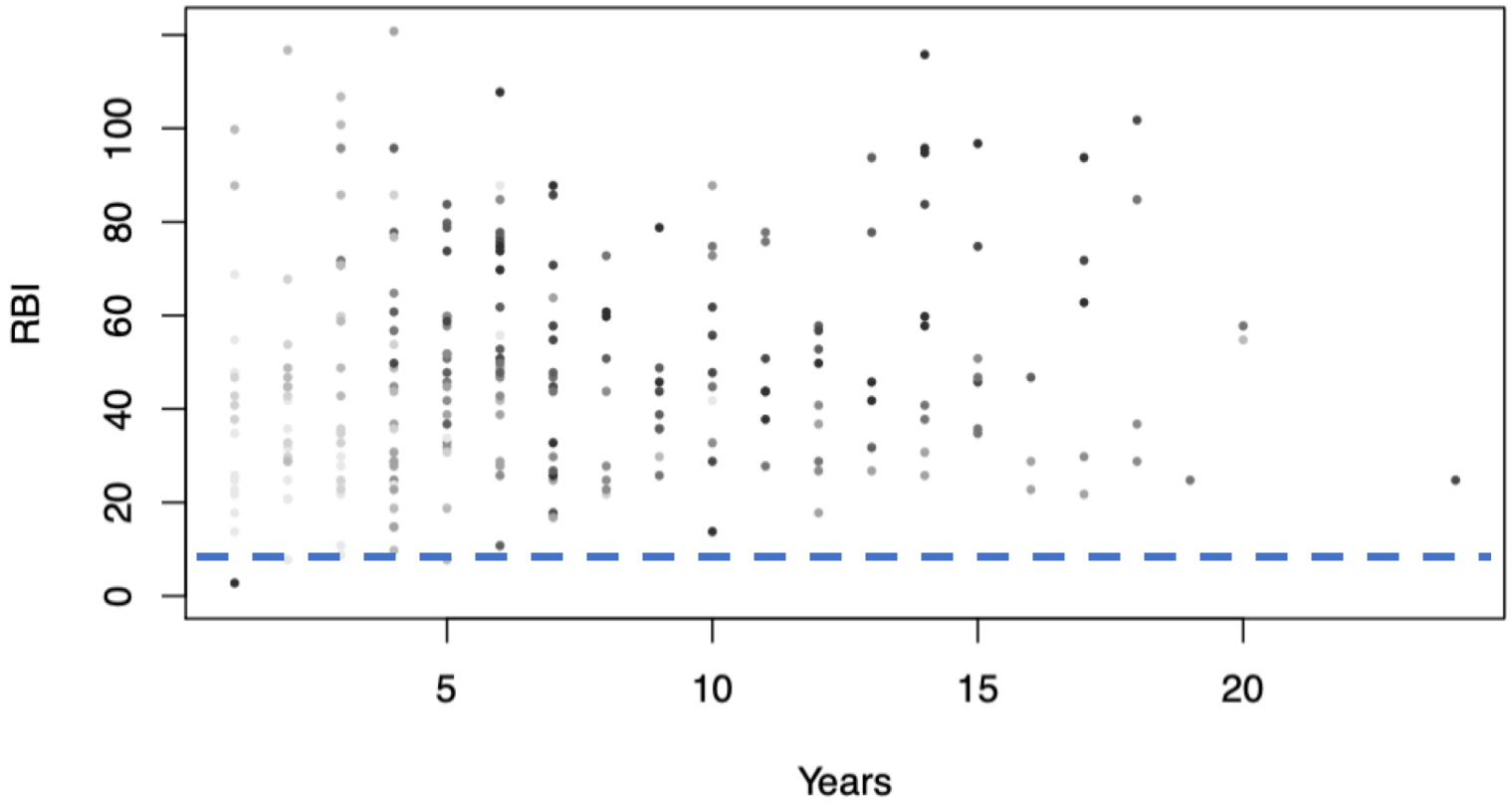
**Tree growth:**



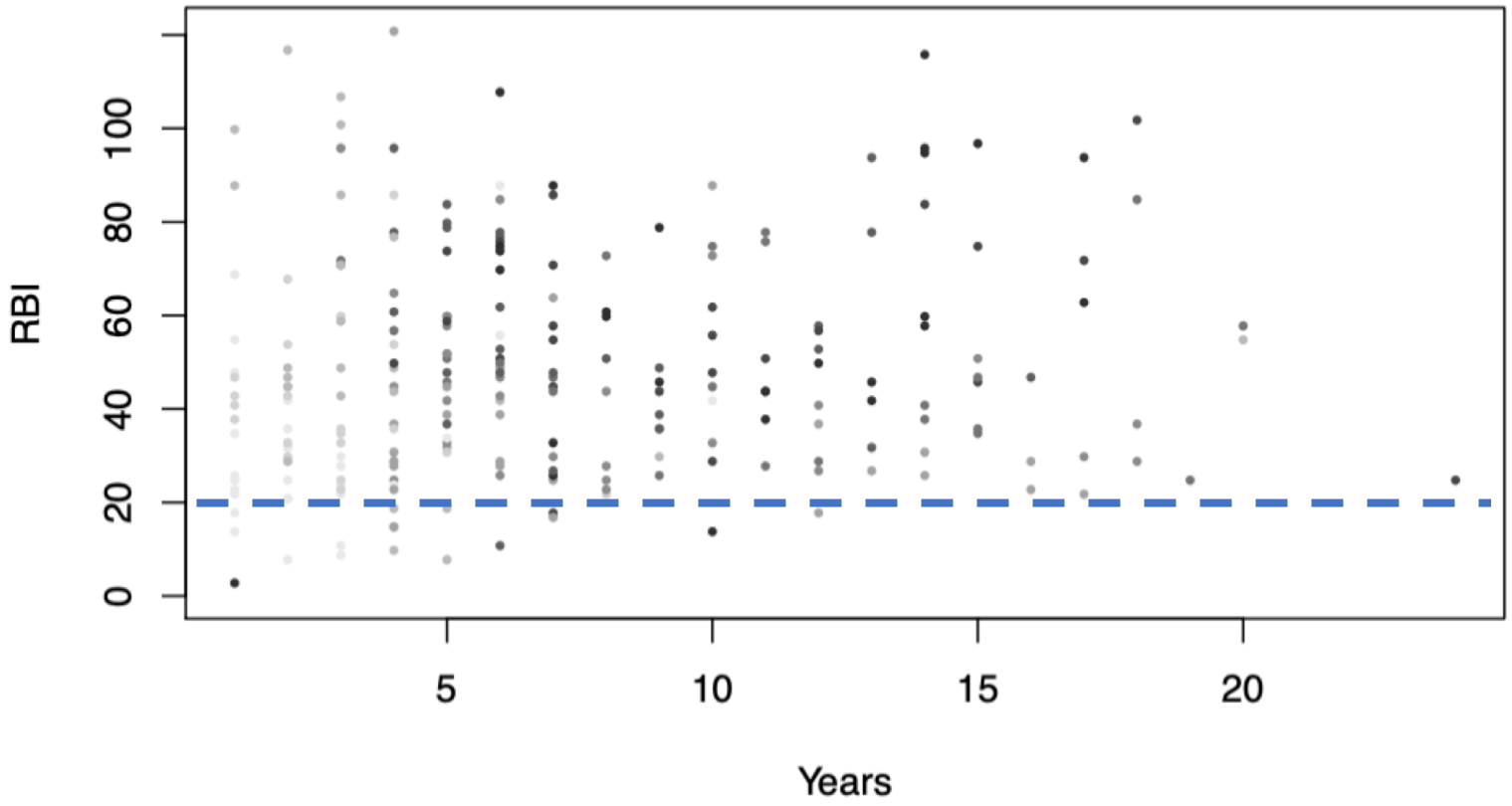
**Tree growth:**



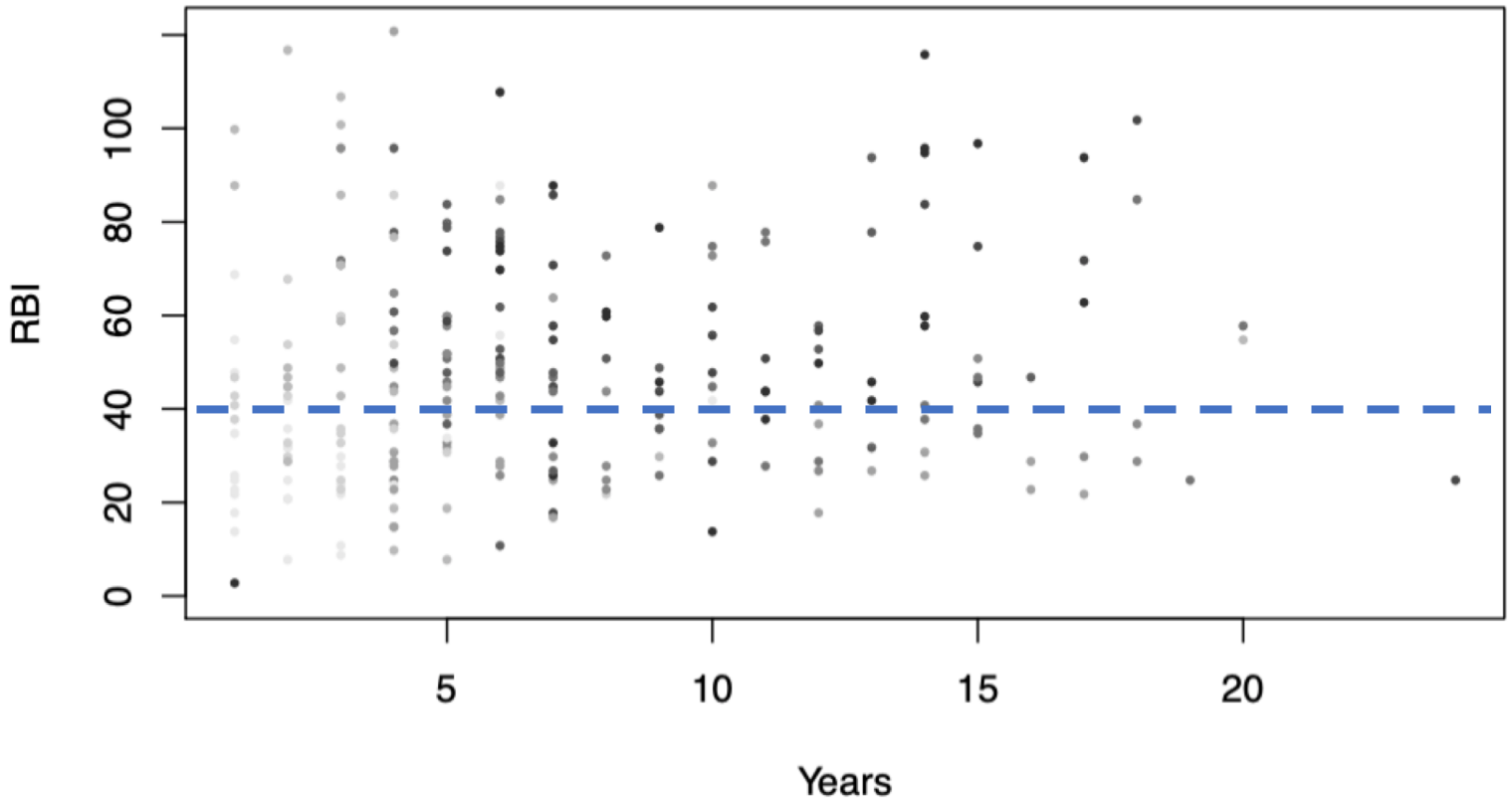
**Tree growth:**



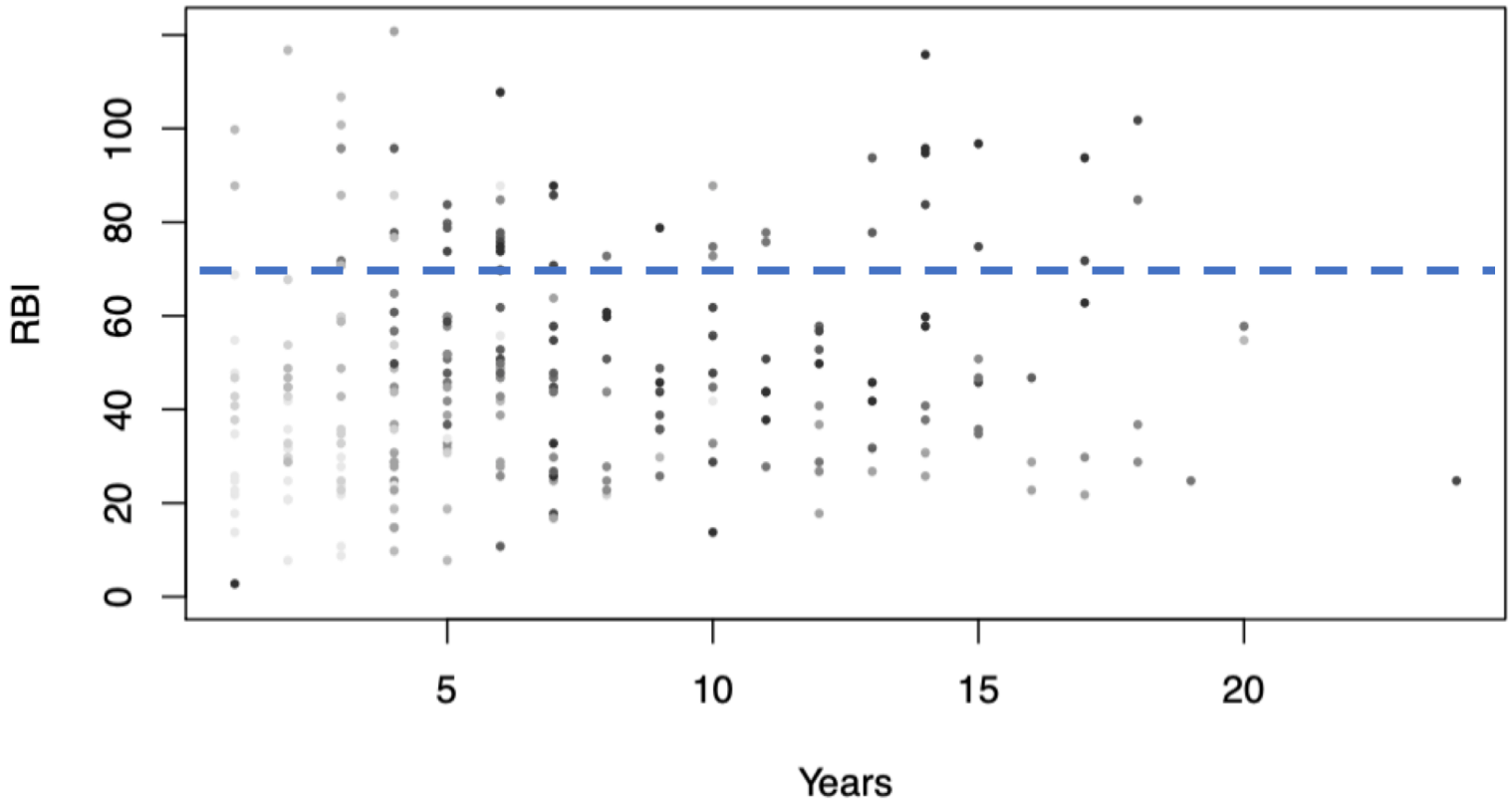
**Tree growth:**



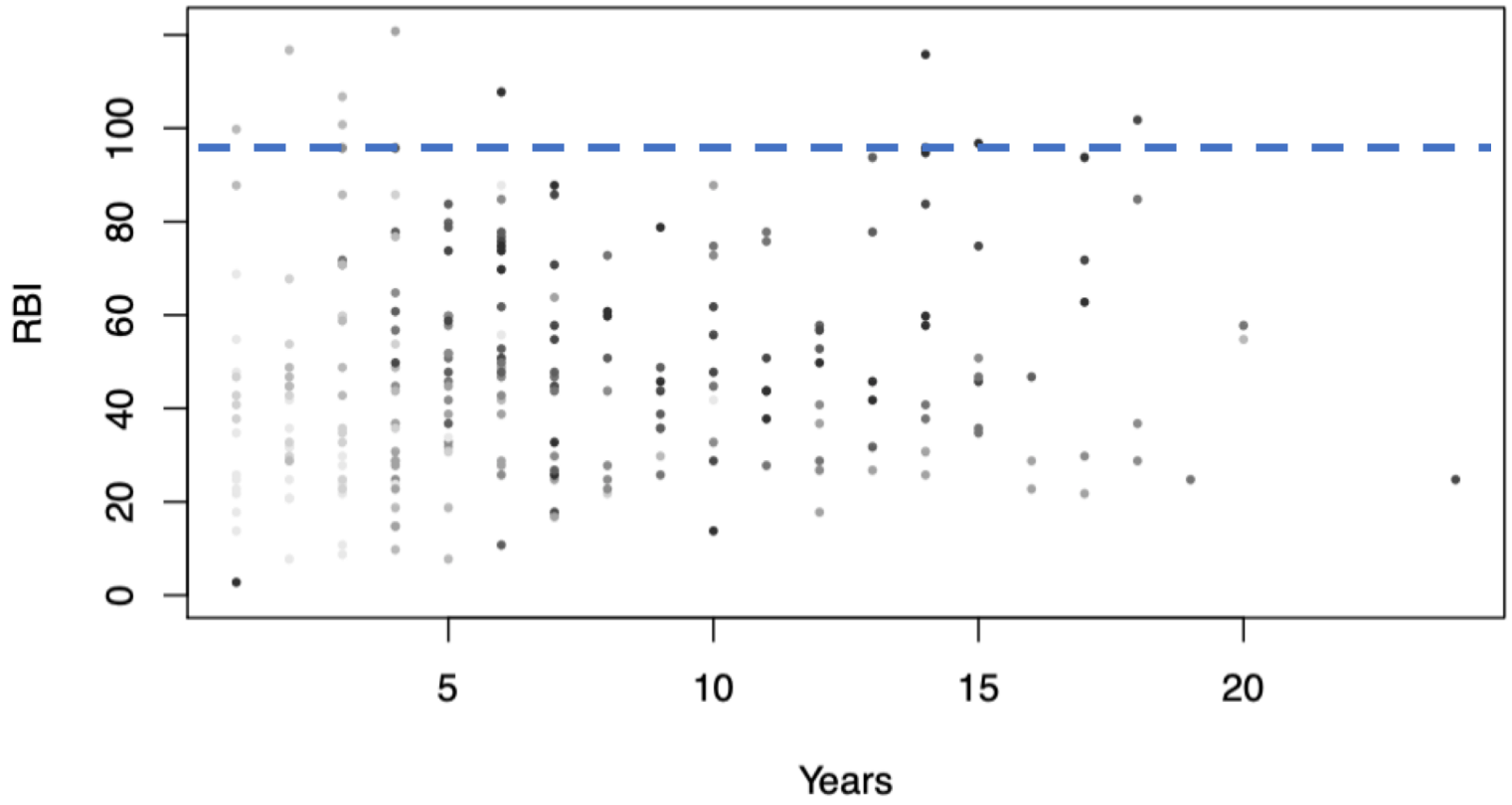
**Tree growth:**



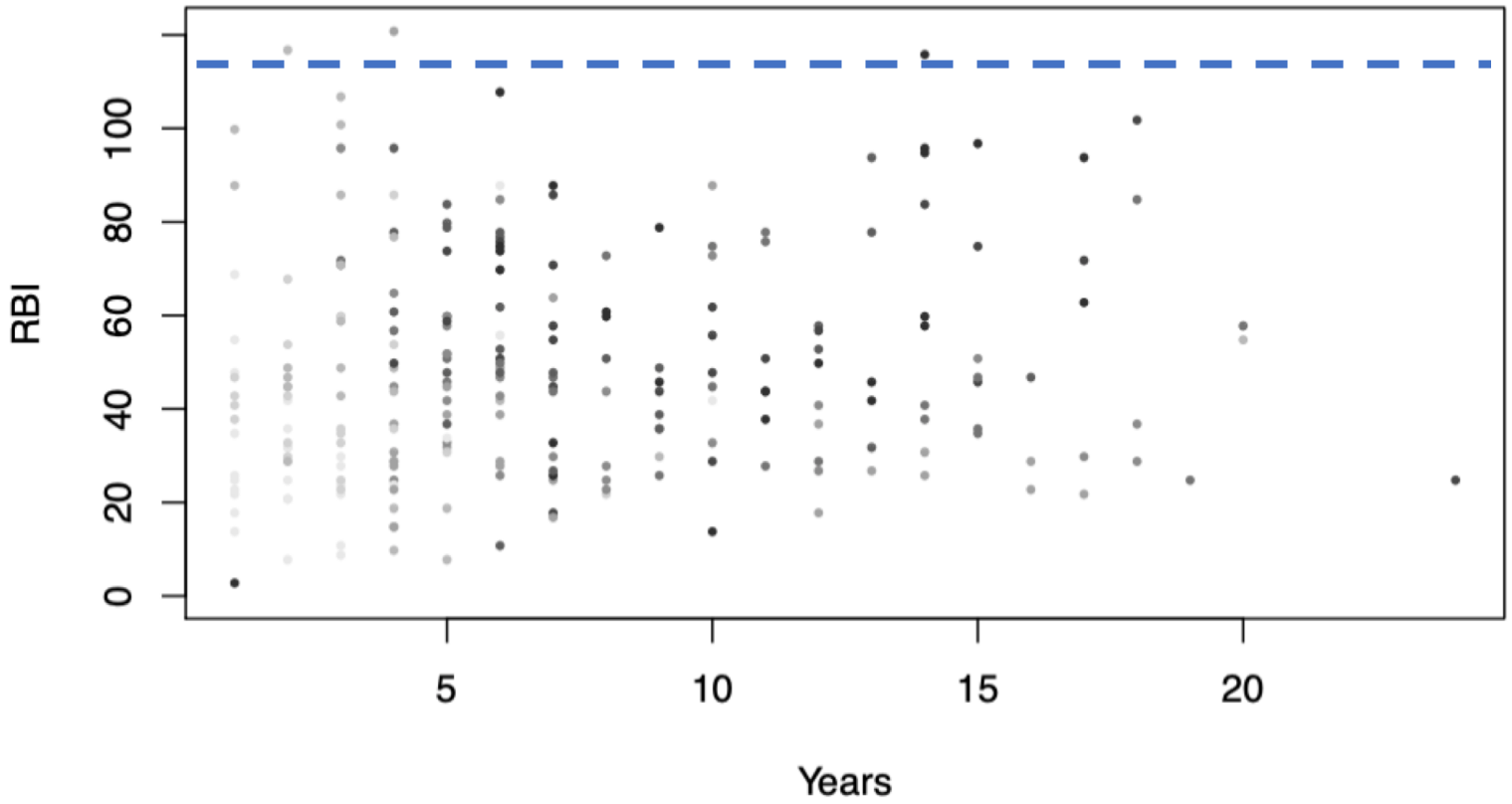
**Tree growth:**



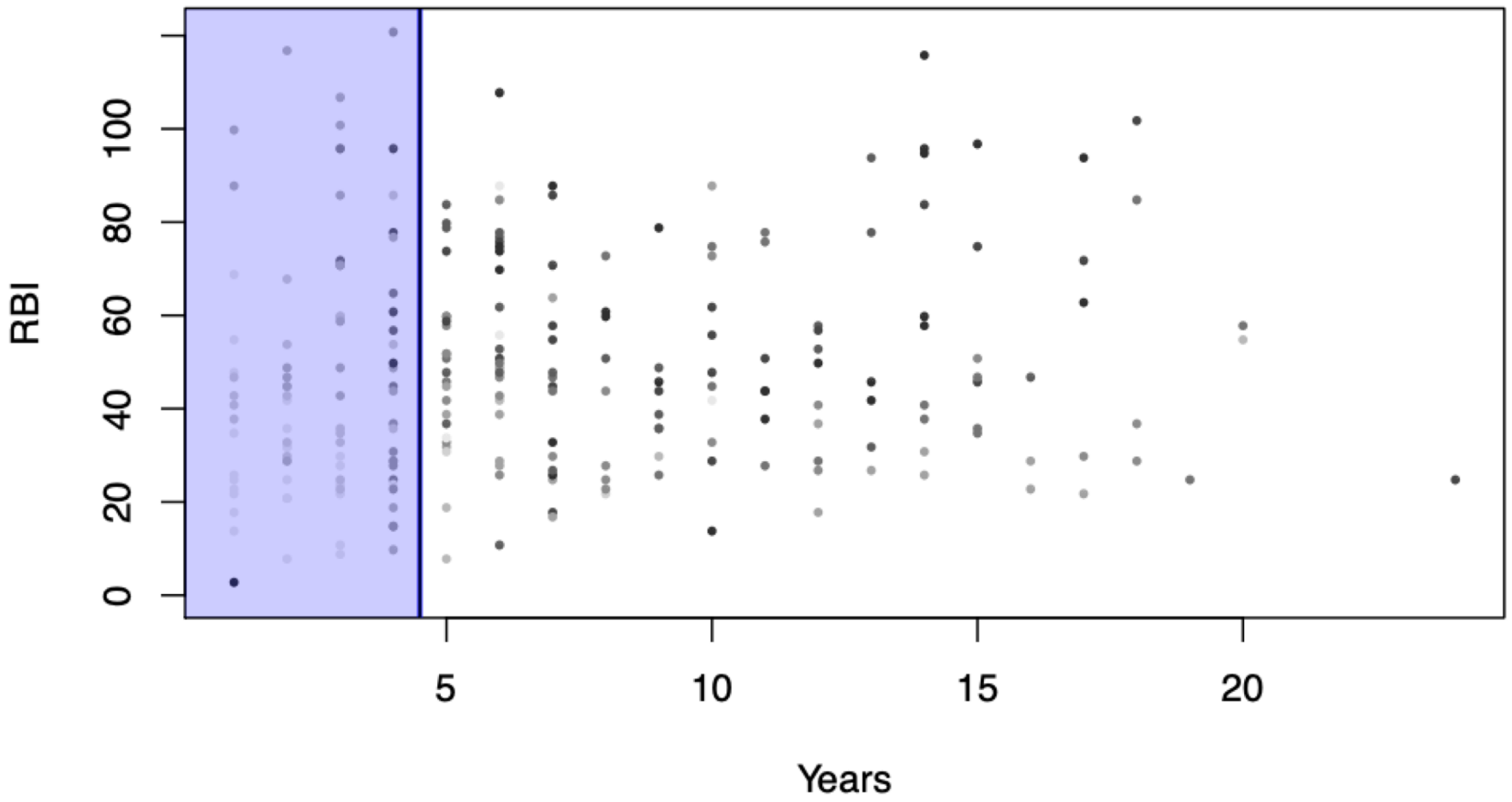
**Tree growth:**



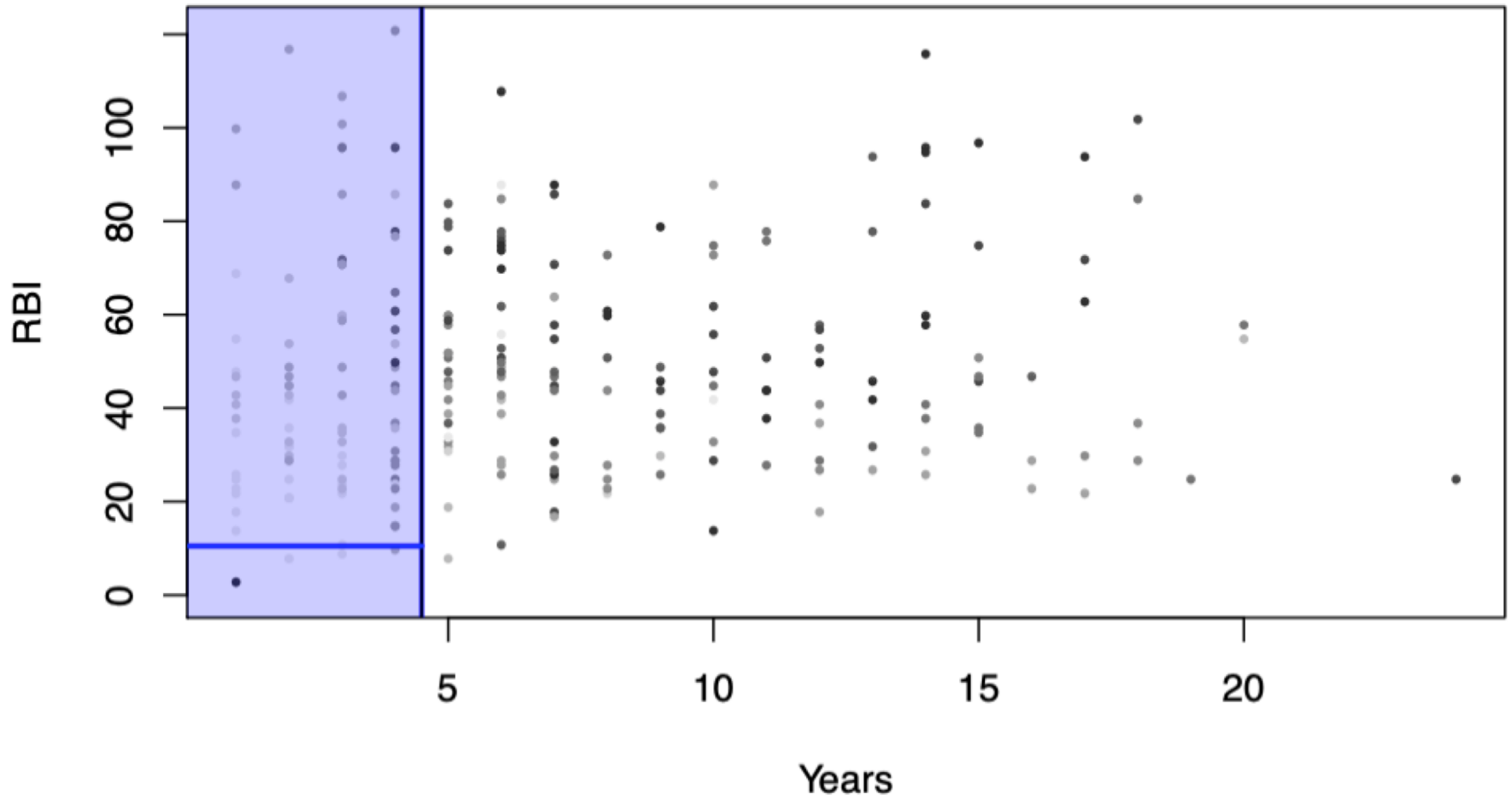
**Tree growth:**



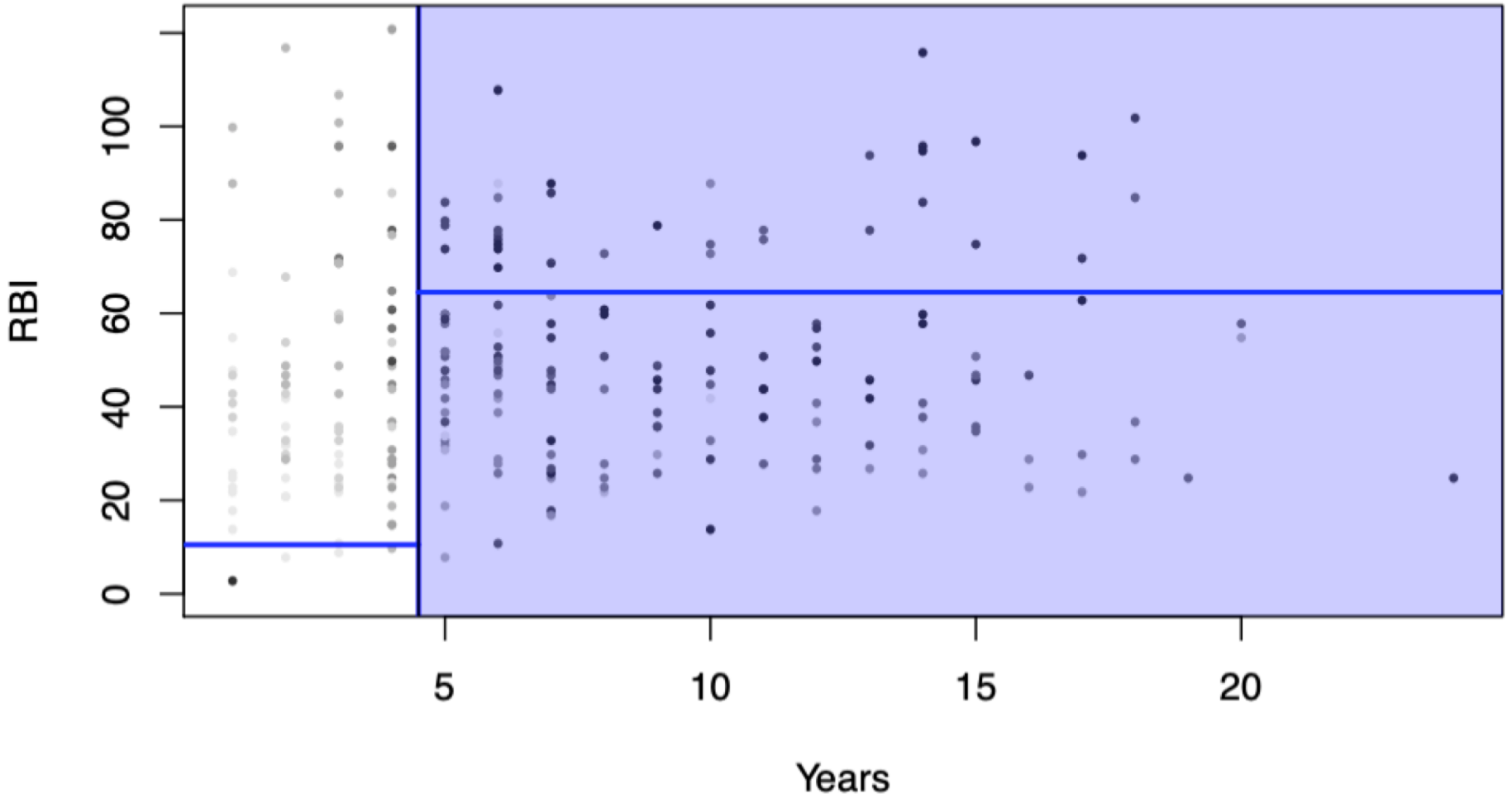
**Tree growth:**



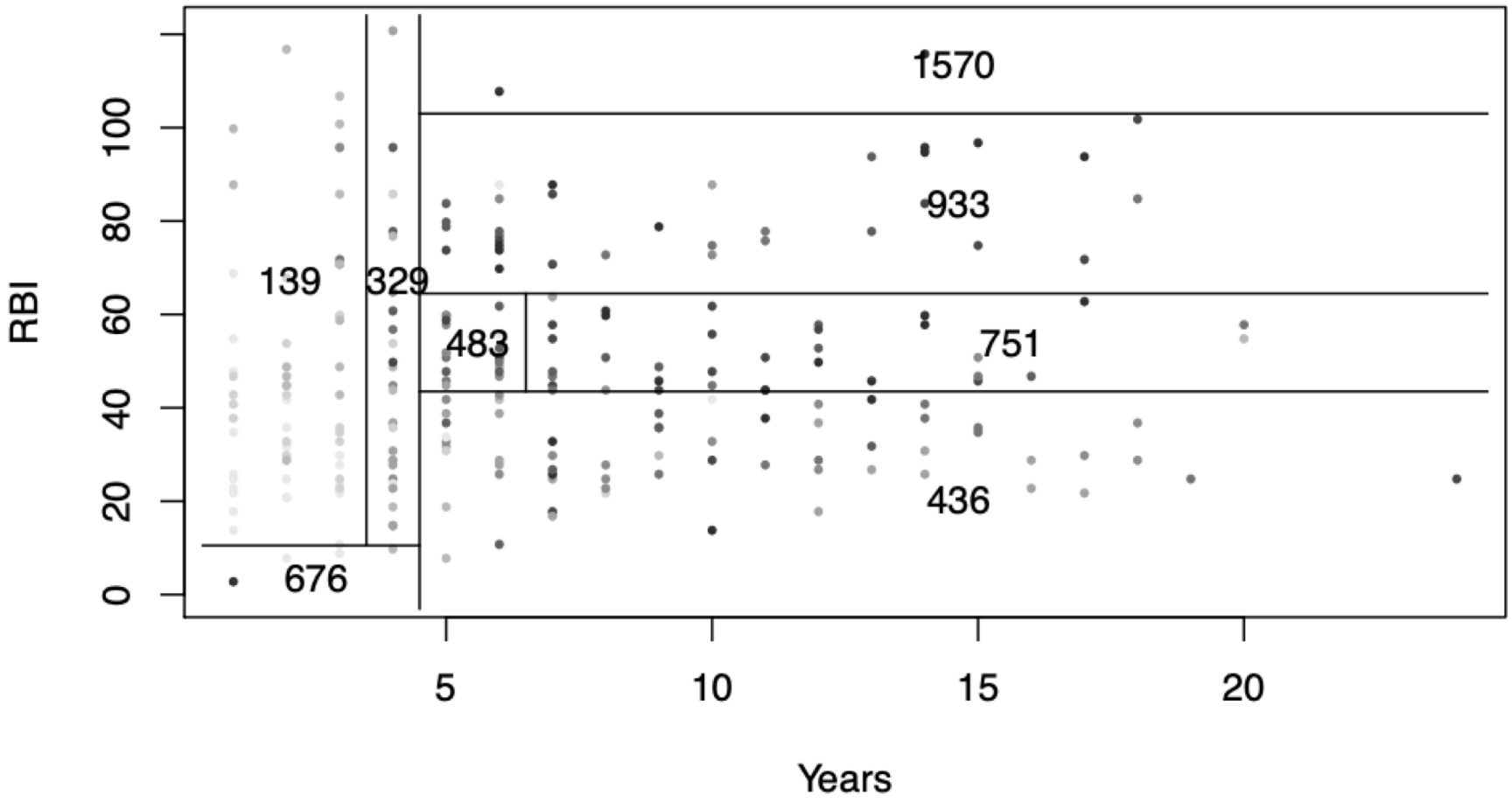
**Tree growth:**



**Tree growth:**



**Tree growth:**


RBI
0
20
40
60
80
100
5
10
15
20
Years
1570
933
751
436
483
329
139
676

# Trees

**Advantages:**

- Response variables can be categorical or quantitative.
- Yields a set of interpretable decision rules.
- Predictive ability is mediocre, but can be improved by combining multiple trees (resampling, ensemble methods)

**Issues:**

- Instability. Trees can have high variance. As data change, tree topology can change dramatically, making interpretation difficult
- Lack of smoothness. The splits lead to a “jagged” decision boundary. More of a problem for regression than classification
- Difficulty capturing additive structure, where the regression function is a sum of terms

**Bias variance control:**

As tree is grown deeper, bias decreases, variance increases

- Stop growing
  - Depth
  - Minimum number of samples at the leaf
  - Minimum RSS reduction at next split
- Pruning: grow a fully-grown tree and then prune it back
  - Use cross validation to find the subtree with the best performance
  - Cost-complexity pruning

$$C(T) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

# Random forest

1. For $b = 1$ to $B$:
   - **(a)** Draw a bootstrap sample $Z^*$ of size $n$ from the training data
   - **(b)** Grow a random-forest tree $T_b$ to the bootstrapped data, recursively repeating following steps, until minimum node size reached:
     - **i.** Select $m$ variables at random from the $p$ variables
     - **ii.** Pick the best variable/split-point among the $m$
     - **iii.** Split the node into two children nodes
2. Output the ensemble fo trees $\{T_b\}_{b=1}^{B}$.

To make a prediction at a new point $x$:

*Regression*: Average $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

*Classification*: Majority vote of the individual trees

Bagging of training samples

Subsampling predictors to further decorrelate the trees

Reduce variance by taking the average

Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**

Training samples: A, C, C, D, D, E

OOB samples: B, F

Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**

Training samples: A, C, C, D, D, E

OOB samples: B, F

$x_1, x_3, x_4$

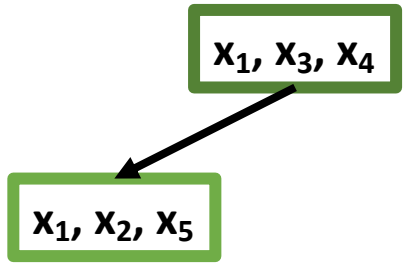
Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**

Training samples: A, C, C, D, D, E

OOB samples: B, F



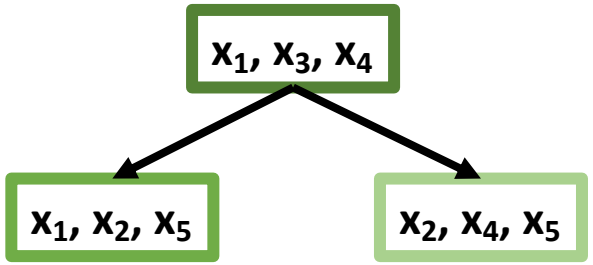
Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**

Training samples: A, C, C, D, D, E

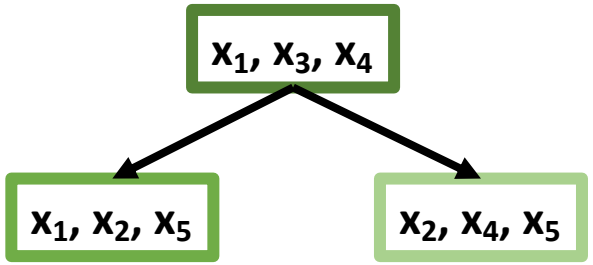
OOB samples: B, F



Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**
Training samples: A, C, C, D, D, E
OOB samples: B, F




**Tree 2**
Training samples: A, B, D, E, E, E
OOB samples: C, F

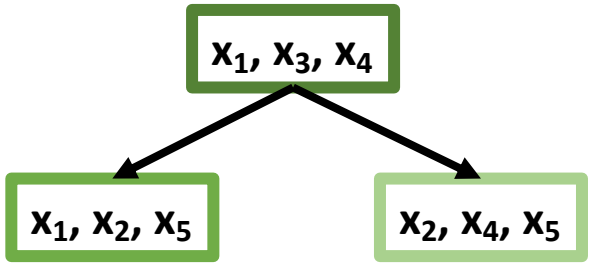
Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**

Training samples: A, C, C, D, D, E

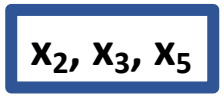
OOB samples: B, F




**Tree 2**

Training samples: A, B, D, E, E, E

OOB samples: C, F



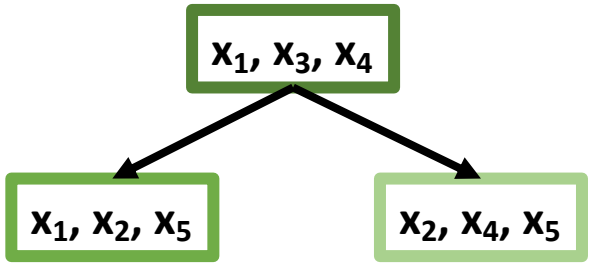
Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**

Training samples: A, C, C, D, D, E

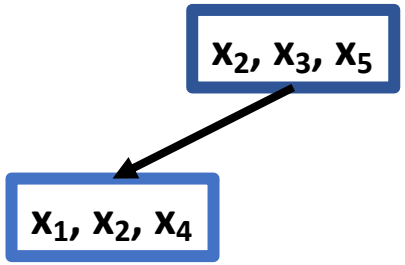
OOB samples: B, F




**Tree 2**

Training samples: A, B, D, E, E, E

OOB samples: C, F



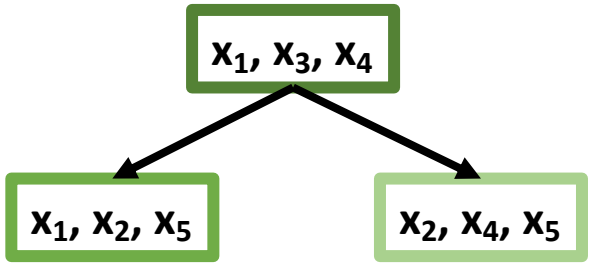
Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |

**Tree 1**

Training samples: A, C, C, D, D, E

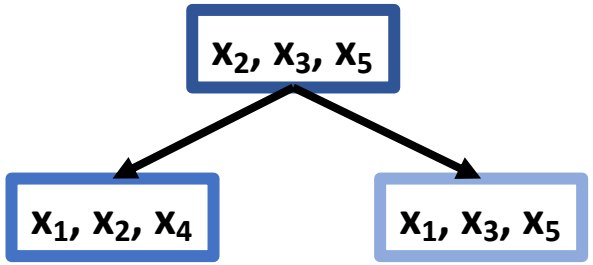
OOB samples: B, F




**Tree 2**

Training samples: A, B, D, E, E, E

OOB samples: C, F



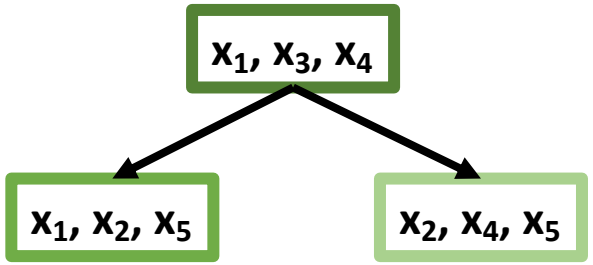
Training Data
n = 6
P = 5
m = 3

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $\hat{y}$ |
|---|---|---|---|---|---|---|
| A | | | | | | |
| B | | | | | | |
| C | | | | | | |
| D | | | | | | |
| E | | | | | | |
| F | | | | | | $(\hat{y}_{1,F}+\hat{y}_{2,F})/2$ |

**Tree 1**

Training samples: A, C, C, D, D, E

OOB samples: B, F

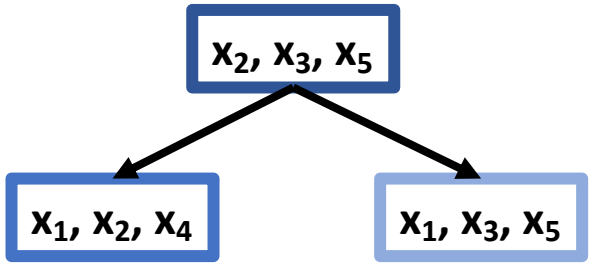



Predict OOB Sample: $\hat{y}_{1,B}$, $\hat{y}_{1,F}$

**Tree 2**

Training samples: A, B, D, E, E, E

OOB samples: C, F




Predict OOB Sample: $\hat{y}_{2,C}$ , $\hat{y}_{2,F}$

# Principal component analysis (PCA)

PCA finds the directions of greatest variability in the data.

- Unsupervised learning method for visualization and dimension reduction
- All the principal vectors are orthogonal

1. Center the data: $x_i \mapsto x_i - \frac{1}{n}\sum_{j=1}^{n} x_j = x_i - \overline{x}$

2. Compute the $d \times d$ sample covariance $S = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$. Note that

$$\frac{1}{n}(x_i - \overline{x})^2$$

is the sample variance of 1-dimensional data

3. Find the first $k$ eigenvectors of $S$,

$$\phi_1, \ldots, \phi_k \in \mathbb{R}^d, \qquad S\phi_j = \lambda_j \phi_j$$

**Principal vectors**

4. Project the data onto those $k$ vectors:

$$x_i \mapsto \overline{x} + (\phi_1^T x_i)\phi_1 + \ldots + (\phi_k^T x_i)\phi_k$$

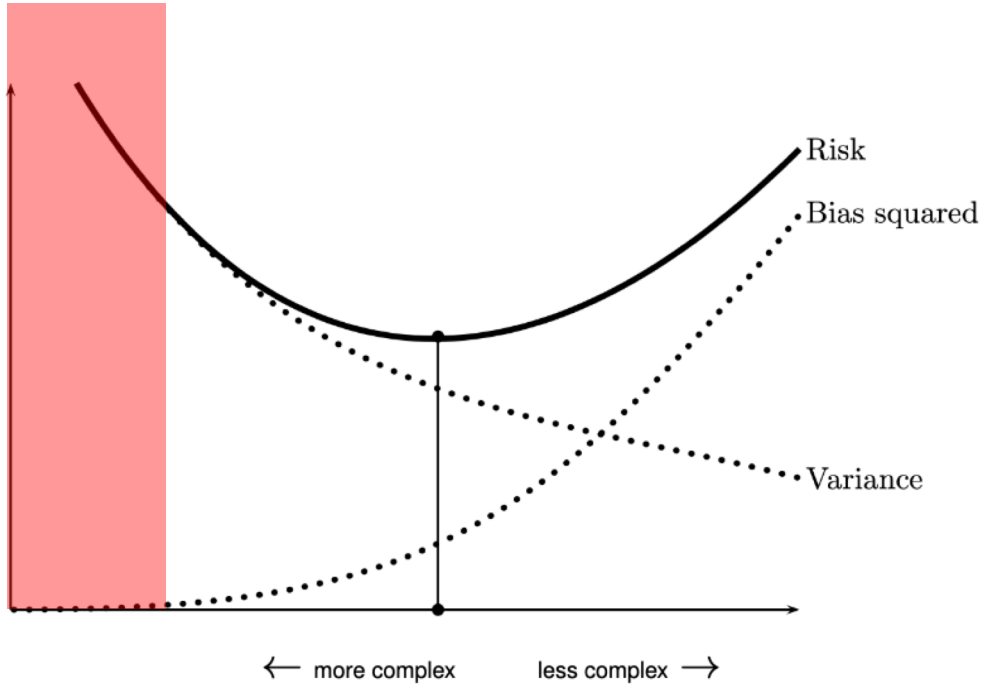
**Principal components**

# Bias-Variance tradeoffs & Overfitting

Bias and variance are two sides of the same coin: As squared bias goes up, variance goes down

Risk=Bias$^2$+ Variance

Overfitting




Overfitting: model has a small training MSE but a large test MSE.

Think about when overfitting will happen for different models (linear regression, logistic regression, KNN, tree/random forest, etc.)

# Model evaluation

**Evaluation with training data:**

- $R^2$, training MSE, misclassification rate on training set
- Minimize bias, may lead to high variance and cause overfitting

**Evaluation with test data:**

- Test MSE, misclassification rate on test set

What if we only have one data set?
Split the data set and treat one of the subset as test data.

1. Divide dataset randomly into a training set and a validation set.
2. Fit the model on the training set.
3. Use the validation set to obtain estimated test error.
4. Repeat!

**Cross validation:**

- leave-one-out cross validation: tiny validation set with only 1 sample
  - no randomness
- K-fold cross validation: randomly divide into k folds; k-1 folds for training, 1 fold for validation

# Penalization term

$$L'(\beta, X, y) = L(\beta, X, y) + \lambda \sum_{j=1}^{d} \beta_j^2$$

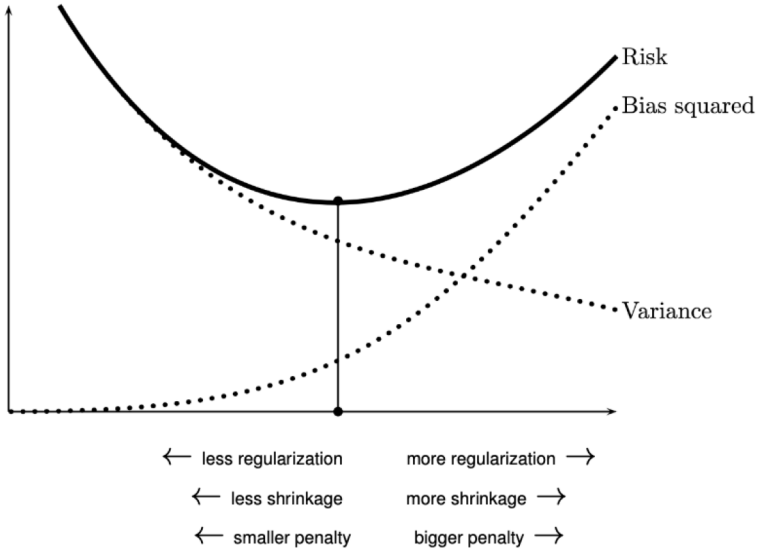
- E.g. logistic regression

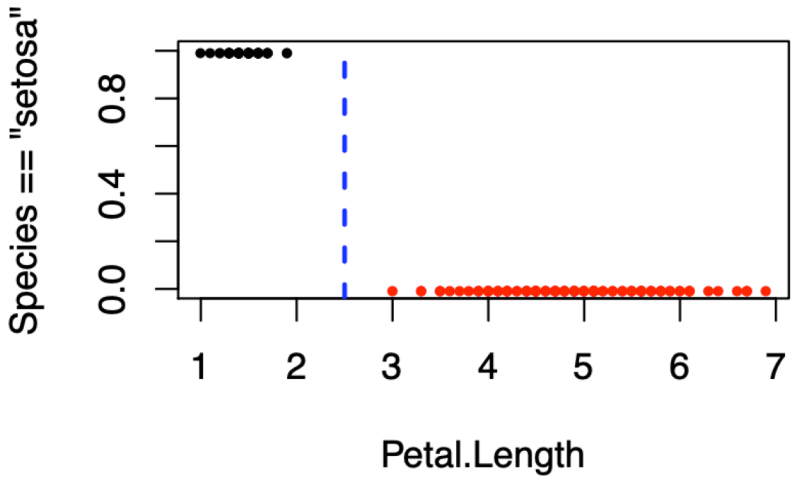
$$L(\beta_0, \beta_1) = \sum_{i=1}^{n} \left[ log\left(1 + e^{\beta_0 + X_i \beta_1}\right) - Y_i \left(\beta_0 + X_i \beta_1\right) \right] + \lambda \left(\beta_0^2 + \beta_1^2\right)$$

Tree

$$C(T) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- Penalization term leads to coefficient shrink
- Alleviate overfitting, avoid coefficient explosion in logistic regression (a special case of overfitting)



# Standardization / Scaling

- KNN: If the predictors are on very different scales, it is necessary to standardize the predictors. Otherwise, the prediction will be dominated by some of the predictors.
- Linear regression/Logistic regression: Standardization doesn’t make a difference
- Linear regression/Logistic regression with a penalization term: Without standardization, the coefficients for predictors at small scales with be more strongly penalized
- Linear regression/Logistic regression fitted with gradient descent/SGD: Standardization doesn’t change the final results but leads to a better convergence
- Tree/Random forest: Standardization doesn’t make a difference