



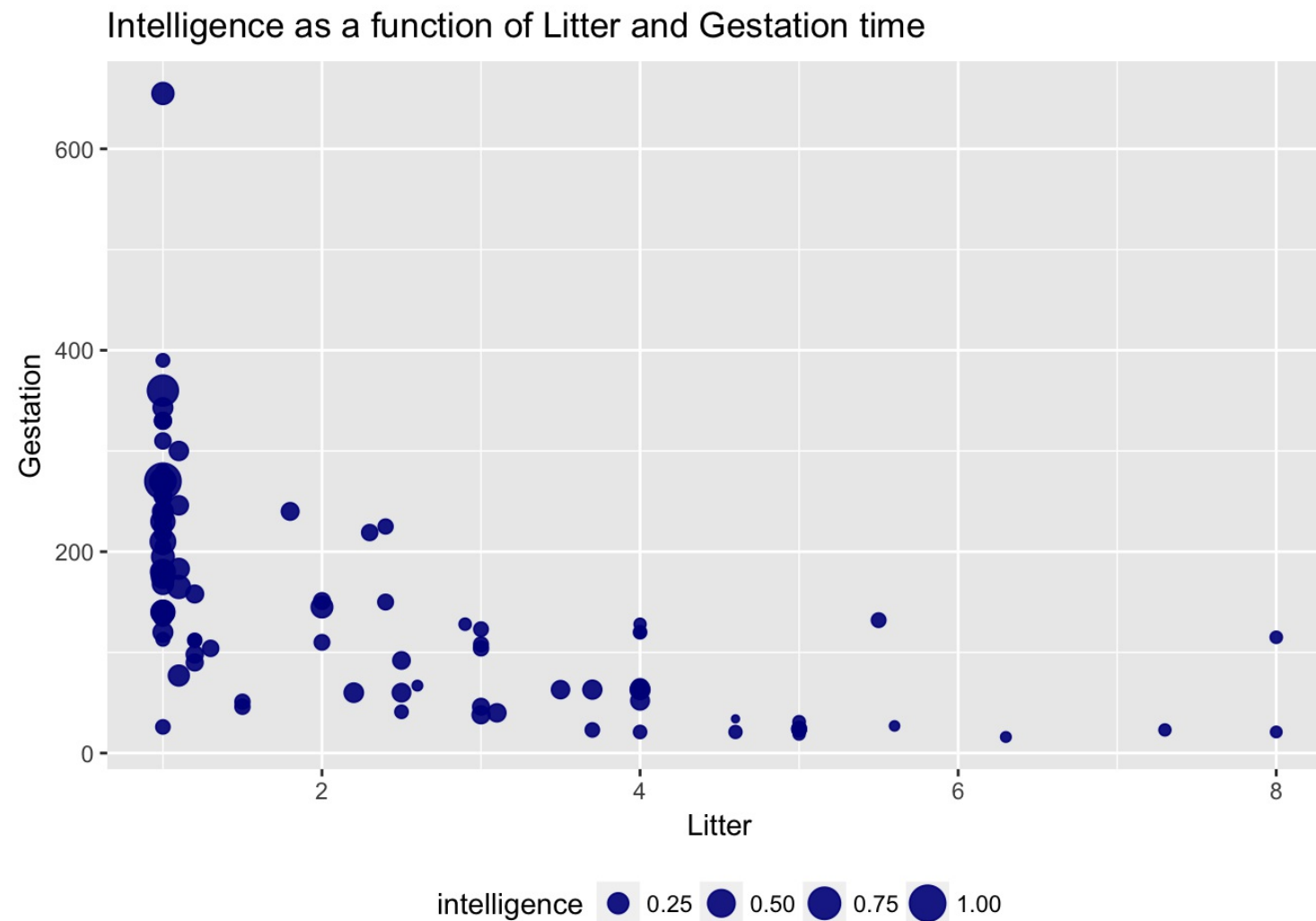
SUPERVISED LEARNING IN R: REGRESSION

# **The intuition behind tree-based methods**

Nina Zumel and John Mount  
Win-Vector, LLC

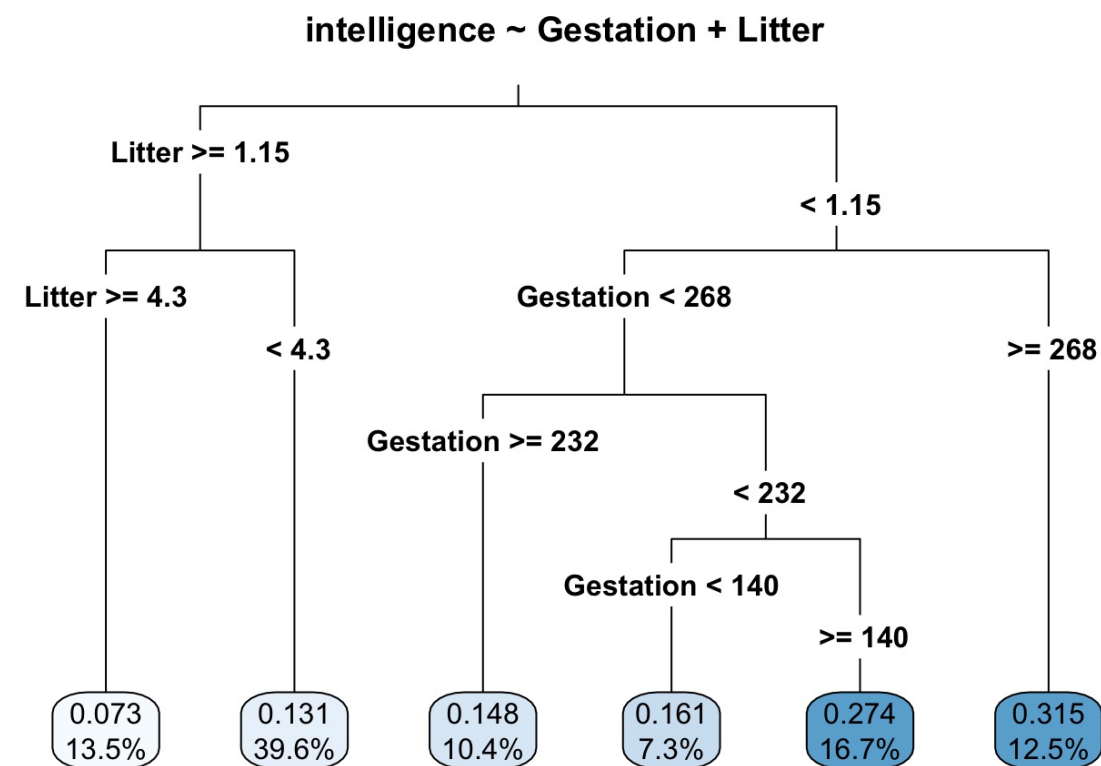


# Example: Predict animal intelligence from Gestation Time and Litter Size





# Decision Trees



Rules of the form:

- *if a AND b AND c THEN y*

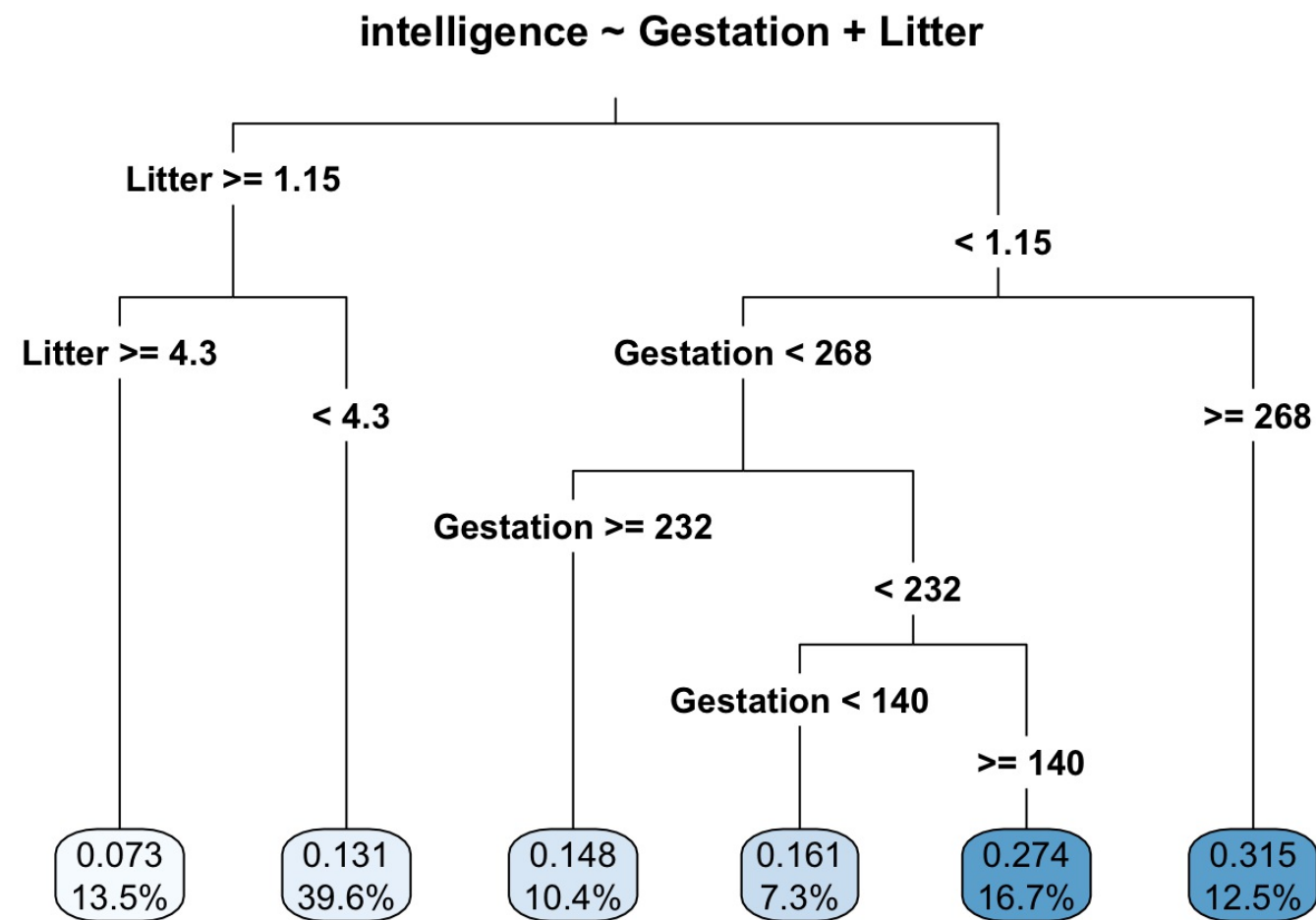
Non-linear concepts

- intervals
- non-monotonic relationships

non-additive interactions

- AND: similar to multiplication

# Decision Trees



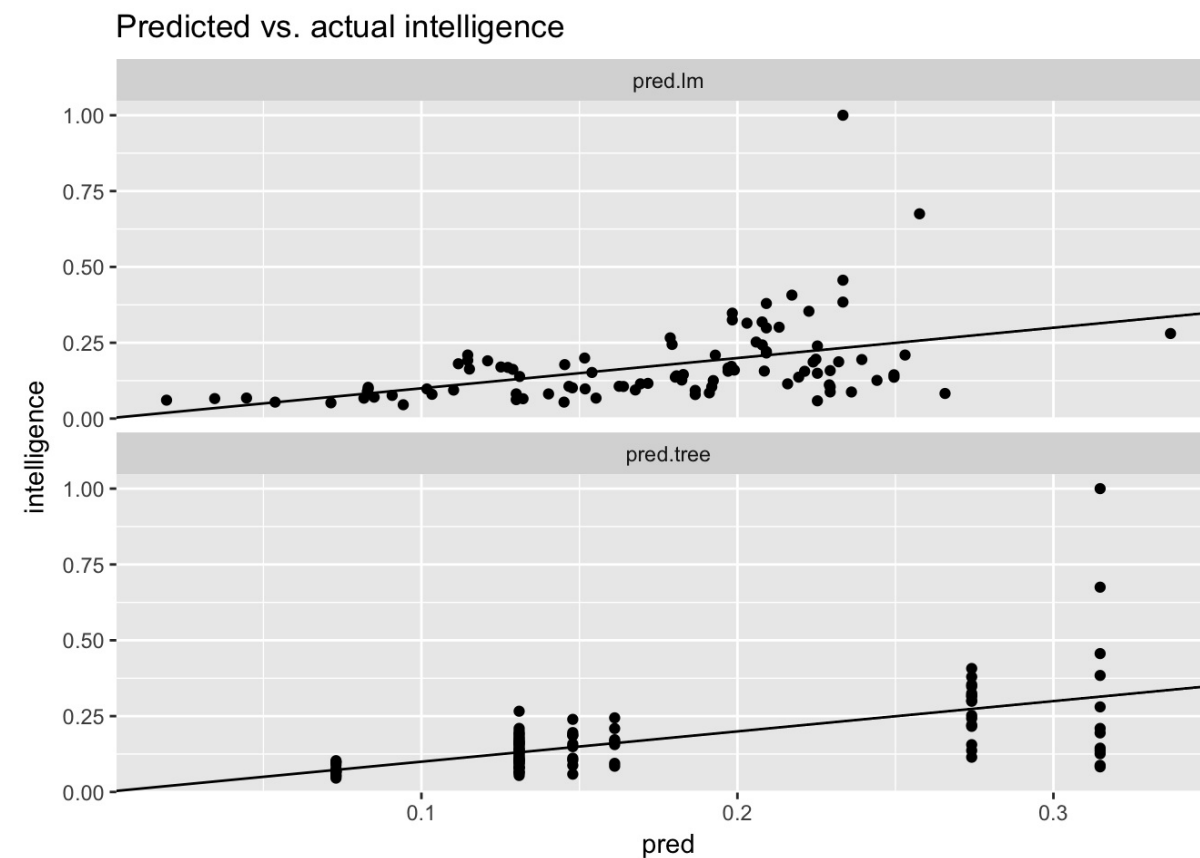
- IF Litter  $< 1.15$  AND Gestation  $\geq 268 \rightarrow$  intelligence = 0.315
- IF Litter IN  $[1.15, 4.3) \rightarrow$  intelligence = 0.131

# Decision Trees

Pro: Trees Have an *Expressive Concept Space*

Model	RMSE
linear	0.1200419
tree	0.1072732

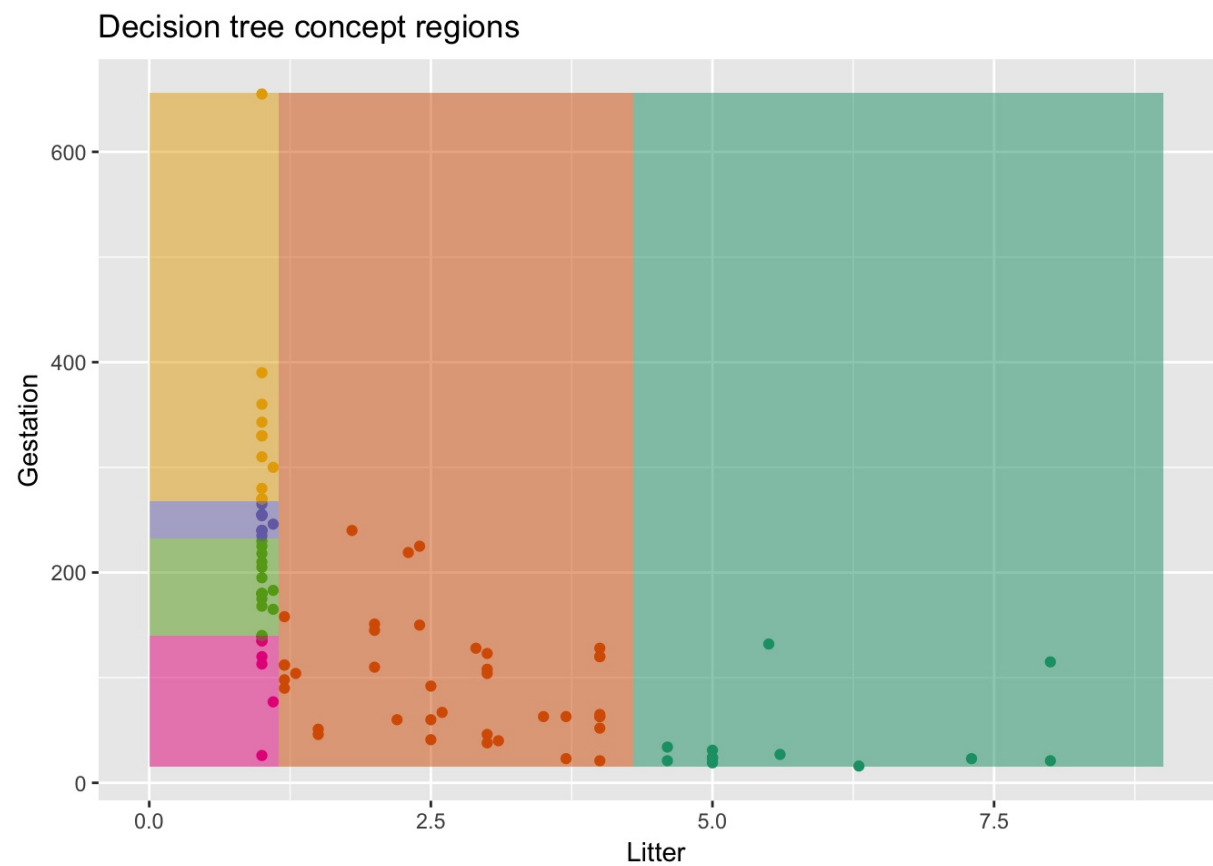
Con: *Coarse-Grained Predictions*





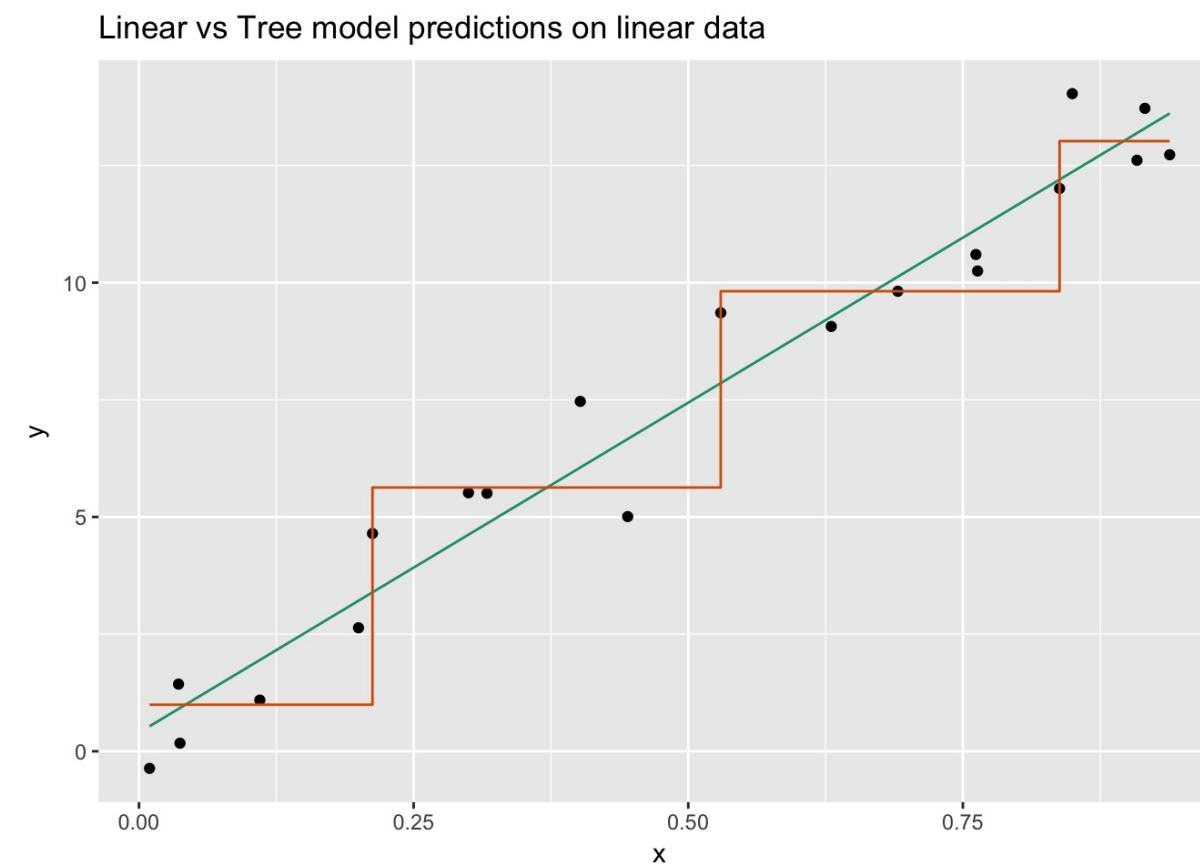
# It's Hard for Trees to Express Linear Relationships

## Trees Predict Axis-Aligned Regions



Each color is a different predicted value

## It's Hard to Express Lines with Steps





# Other Issues with Trees

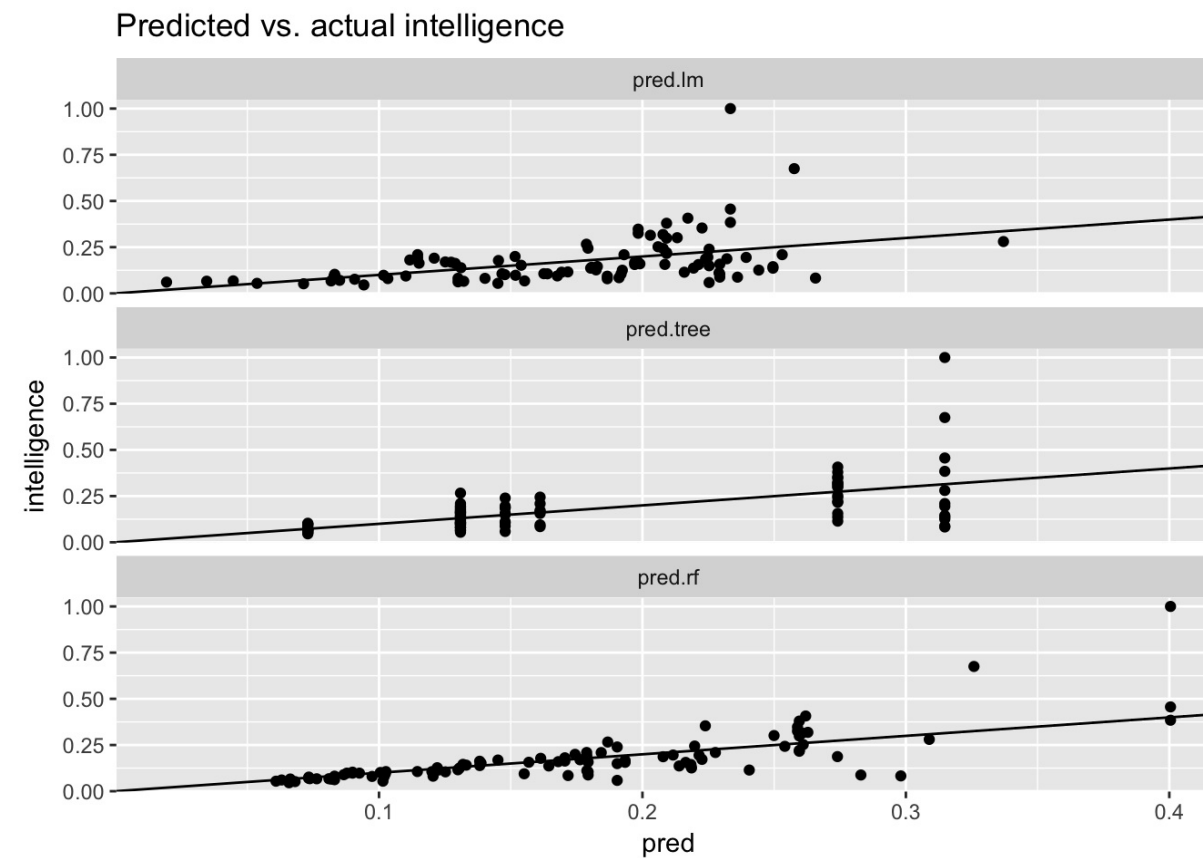
- Tree with too many splits (deep tree):
  - Too complex - danger of overfit
- Tree with too few splits (shallow tree):
  - Predictions too coarse-grained

# Ensembles of Trees

Ensemble Model Fits Animal Intelligence Data Better than Single Tree

Model	RMSE
linear	0.1200419
tree	0.1072732
random forest	0.0901681

Ensembles Give Finer-grained Predictions than Single Trees







## SUPERVISED LEARNING IN R: REGRESSION

**Let's practice!**



## SUPERVISED LEARNING IN R: REGRESSION

# Random forests

Nina Zumel and John Mount  
Win-Vector, LCC



# Random Forests

Multiple diverse decision trees averaged together

- Reduces overfit
- Increases model expressiveness
- Finer grain predictions



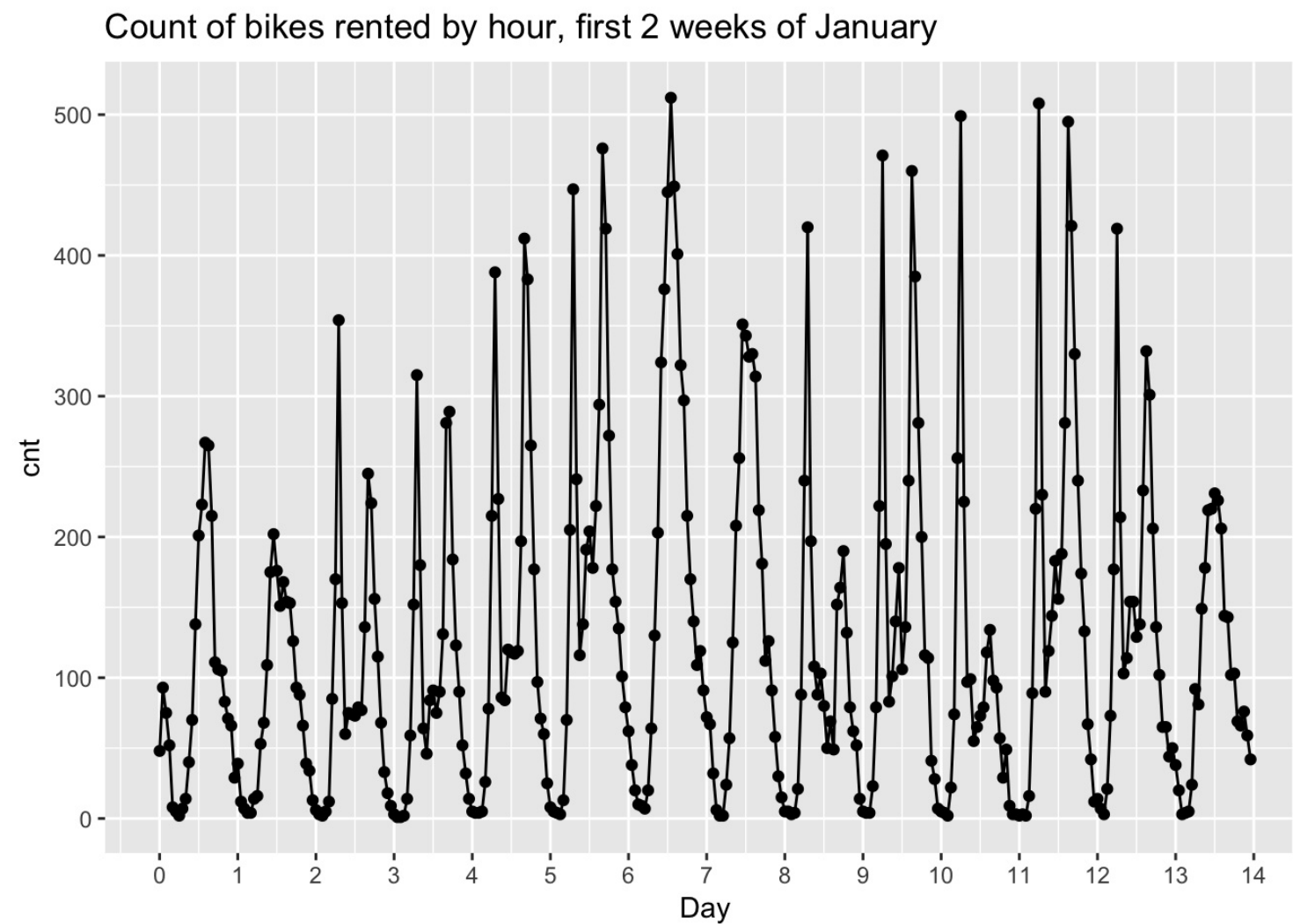
# Building a Random Forest Model

1. Draw bootstrapped sample from training data
2. For each sample grow a tree
  - At each node, pick best variable to split on (from a random subset of all variables)
  - Continue until tree is grown
3. To score a datum, evaluate it with all the trees and average the results.



# Example: Bike Rental Data

```
> cnt ~ hr + holiday + workingday +  
+     weathersit + temp + atemp + hum + windspeed
```



# Random Forests with ranger()

```
> model <- ranger(fmla, bikesJan,  
+               num.trees = 500,  
+               respect.unordered.factors = "order")
```

- formula, data
- num.trees (default 500) - use at least 200
- mtry - number of variables to try at each node
  - default: square root of the total number of variables
- respect.unordered.factors - recommend set to "order"
  - "safe" hashing of categorical variables



# Random Forests with ranger()

```
> model

## Ranger result
## ...
## OOB prediction error (MSE):      3103.623
## R squared (OOB):                0.7837386
```

Random forest algorithm returns estimates of out-of-sample performance.



# Predicting with a ranger() model

```
> bikesFeb$pred <- predict(model, bikesFeb)$predictions
```

predict() inputs:

- model
- data

Predictions can be accessed in the element predictions.





# Evaluating the model

Calculate RMSE:

```
> bikesFeb %>%  
+   mutate(residual = pred - cnt) %>%  
+   summarize(rmse = sqrt(mean(residual^2)))
```

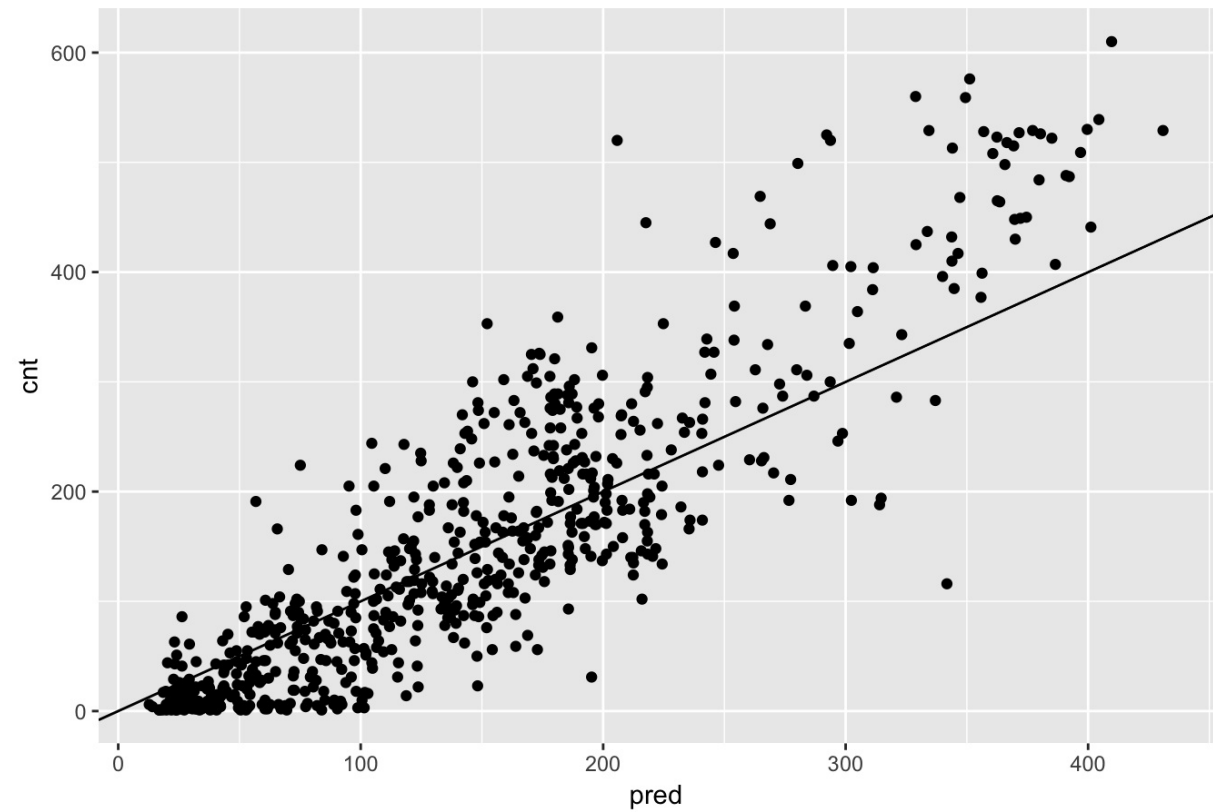
```
##           rmse  
## 1 67.15169
```

Model	RMSE
Quasipoisson	69.3
Random forests	67.15

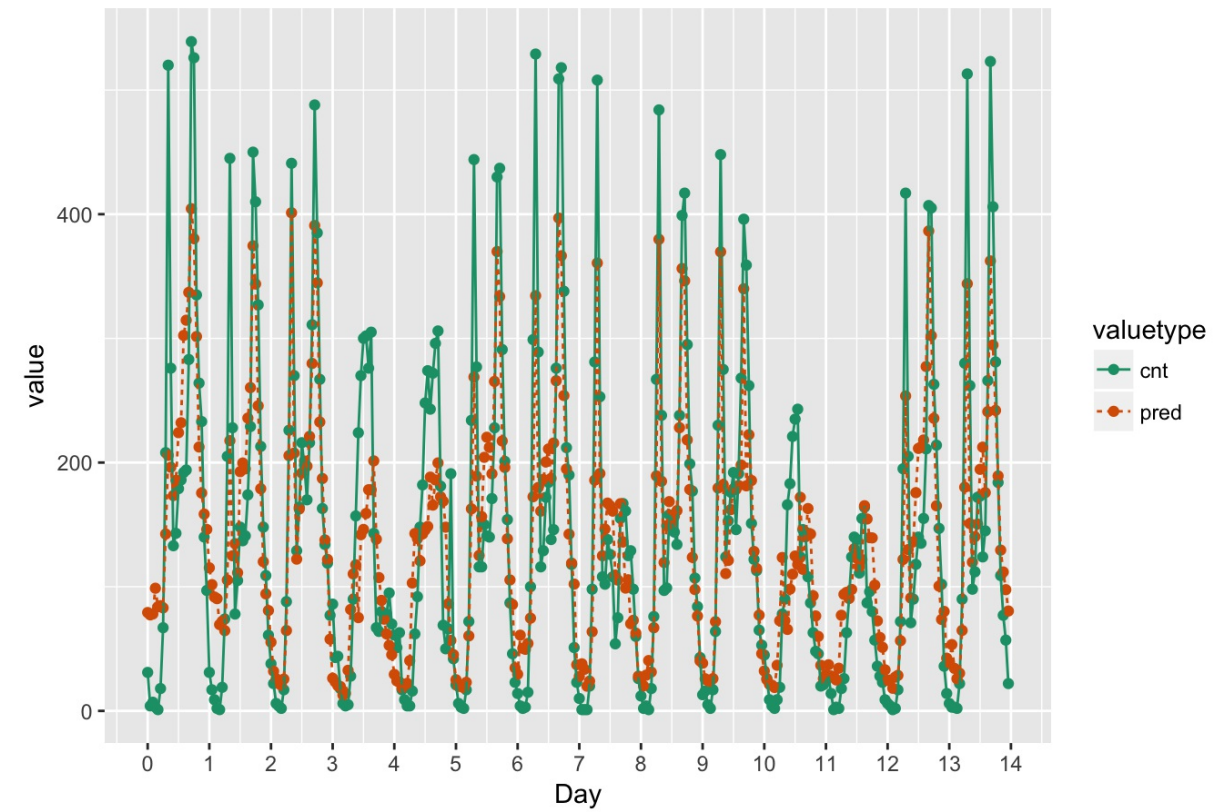


# Evaluating the model

Bike rentals, predictions vs actual, February - Random Forest



Predicted and Actual Hourly Bike Rentals, February - Random Forest





## SUPERVISED LEARNING IN R: REGRESSION

**Let's practice!**



SUPERVISED LEARNING IN R: REGRESSION

# One-Hot-Encoding Categorical Variables

Nina Zumel and John Mount  
Win-Vector, LLC



# Why Convert Categoricals Manually?

- Most R functions manage the conversion for you
  - `model.matrix()`
- `xgboost()` does not
  - Must convert categorical variables to numeric representation
- Conversion to indicators: *one-hot encoding*



# One-hot-encoding and data cleaning with vtreat

Basic idea:

- designTreatmentsZ() to design a *treatment plan* from the training data, then
- prepare() to create "clean" data
  - all numerical
  - no missing values
  - use prepare() with treatment plan for all future data

# A Small vtreat Example

## Training Data

x	u	y
one	44	0.4855671
two	24	1.3683726
three	66	2.0352837
two	22	1.6396267

## Test Data

x	u	y
one	5	2.6488148
three	12	1.5012938
one	56	0.1993731
two	28	1.2778516

# Create the Treatment Plan

```
> vars <- c("x", "u")  
> treatplan <- designTreatmentsZ(dframe, varslst, verbose = FALSE)
```

## Inputs to designTreatmentsZ()

- dframe: training data
- varlist: list of input variable names
- set verbose = FALSE to suppress progress messages



# Get the New Variables

The scoreFrame describes the variable mapping and types

```
> (scoreFrame <- treatplan$scoreFrame %>%  
+   select(varName, origName, code))  
  
##           varName origName  code  
## 1  x_lev_x.one        x    lev  
## 2 x_lev_x.three        x    lev  
## 3  x_lev_x.two        x    lev  
## 4      x_catP        x  catP  
## 5      u_clean        u  clean
```

Get the names of the new lev and clean variables

```
> (newvars <- scoreFrame %>%  
+   filter(code %in% c("clean", "lev")) %>%  
+   use_series(varName))  
[1] "x_lev_x.one"    "x_lev_x.three" "x_lev_x.two"    "u_clean"
```



# Prepare the Training Data for Modeling

```
> training.treat <- prepare(treatmentplan, dframe, varRestriction = newvars)
```

Inputs to `prepare()`:

- `treatmentplan`: treatment plan
- `dframe`: data frame
- `varRestriction`: list of variables to prepare (optional)
  - default: prepare all variables

# Before and After Data Treatment

## Training Data

x	u	y
one	44	0.4855671
two	24	1.3683726
three	66	2.0352837
two	22	1.6396267

## Treated Training Data

x_lev_x.one	x_lev_x.three	x_lev_x.two	u_clean
1	0	0	44
0	0	1	24
0	1	0	66
0	0	1	22



# Prepare the Test Data Before Model Application

```
> (test.treat <- prepare(treatplan, test, varRestriction = newvars))
```

```
##      x_lev_x.one x_lev_x.three x_lev_x.two u_clean
## 1             1             0             0         5
## 2             0             1             0        12
## 3             1             0             0        56
## 4             0             0             1        28
```

# vtreat Treatment is Robust

Previously unseen x level: *four*

x	u	y
one	4	0.2331301
two	14	1.9331760
three	66	3.1251029
<b>four</b>	25	4.0332491

*four* encodes to (0, 0, 0)

```
prepare(treatplan, toomany, ...)
```

x_lev_x.one	x_lev_x.three	x_lev_x.two	u_clean
1	0	0	4
0	0	1	14
0	1	0	66
<b>0</b>	<b>0</b>	<b>0</b>	25



## SUPERVISED LEARNING IN R: REGRESSION

**Let's practice!**



SUPERVISED LEARNING IN R: REGRESSION

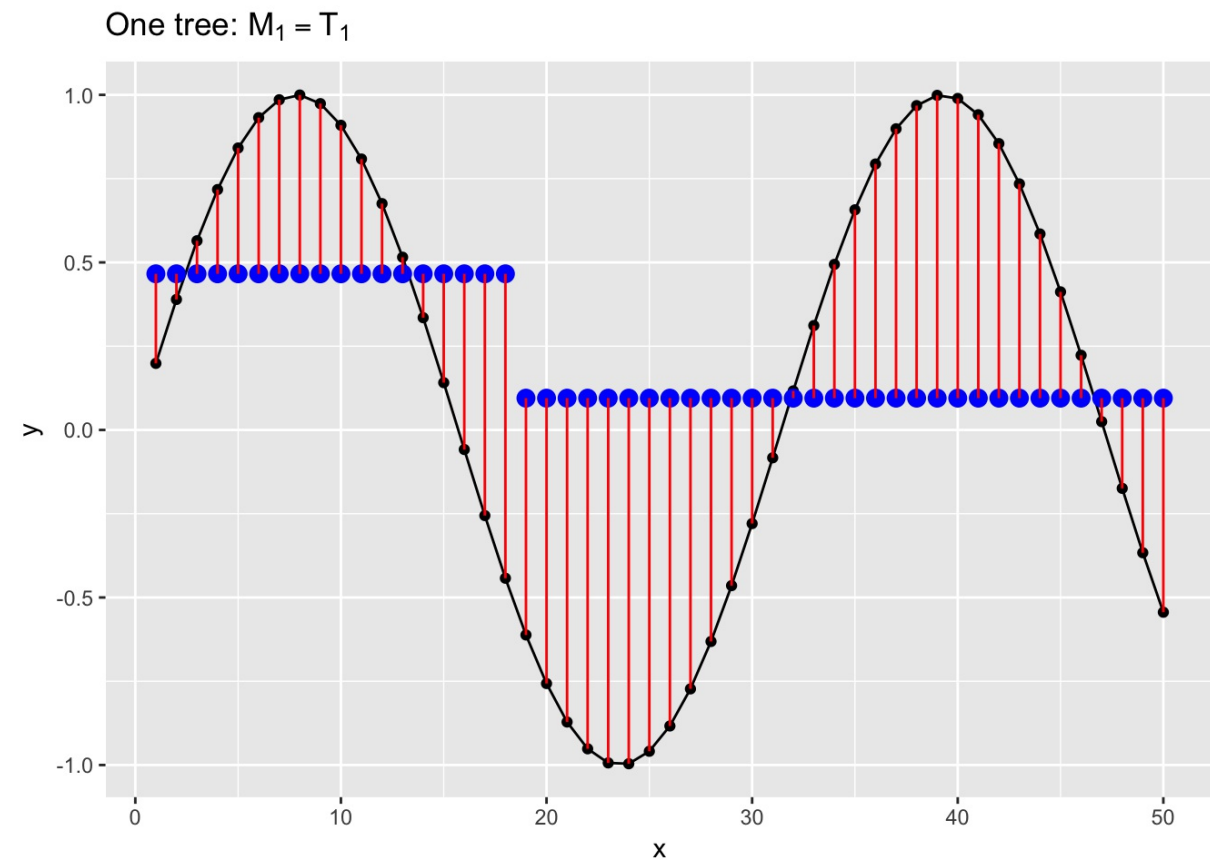
# Gradient boosting machines

Nina Zumel and John Mount  
Win-Vector, LLC



# How Gradient Boosting Works

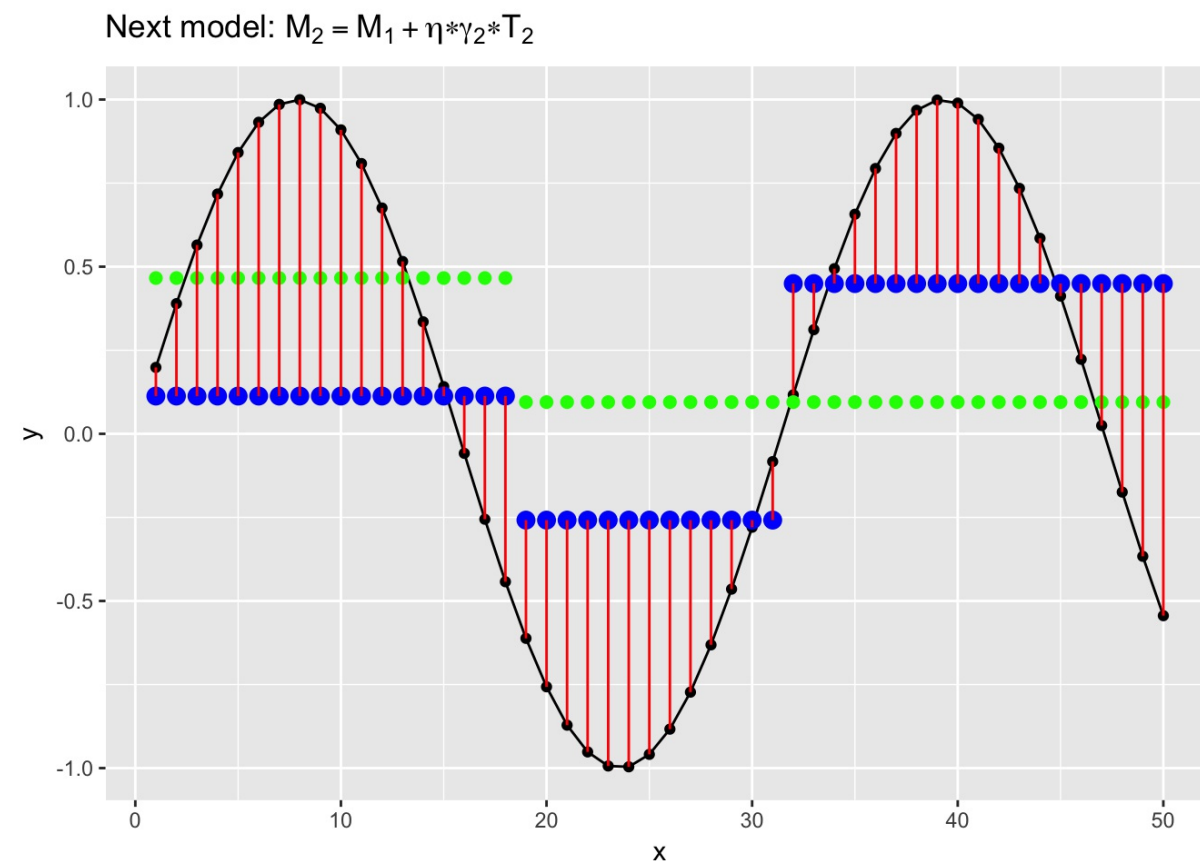
1. Fit a shallow tree  $T_1$  to the data:  $M_1 = T_1$







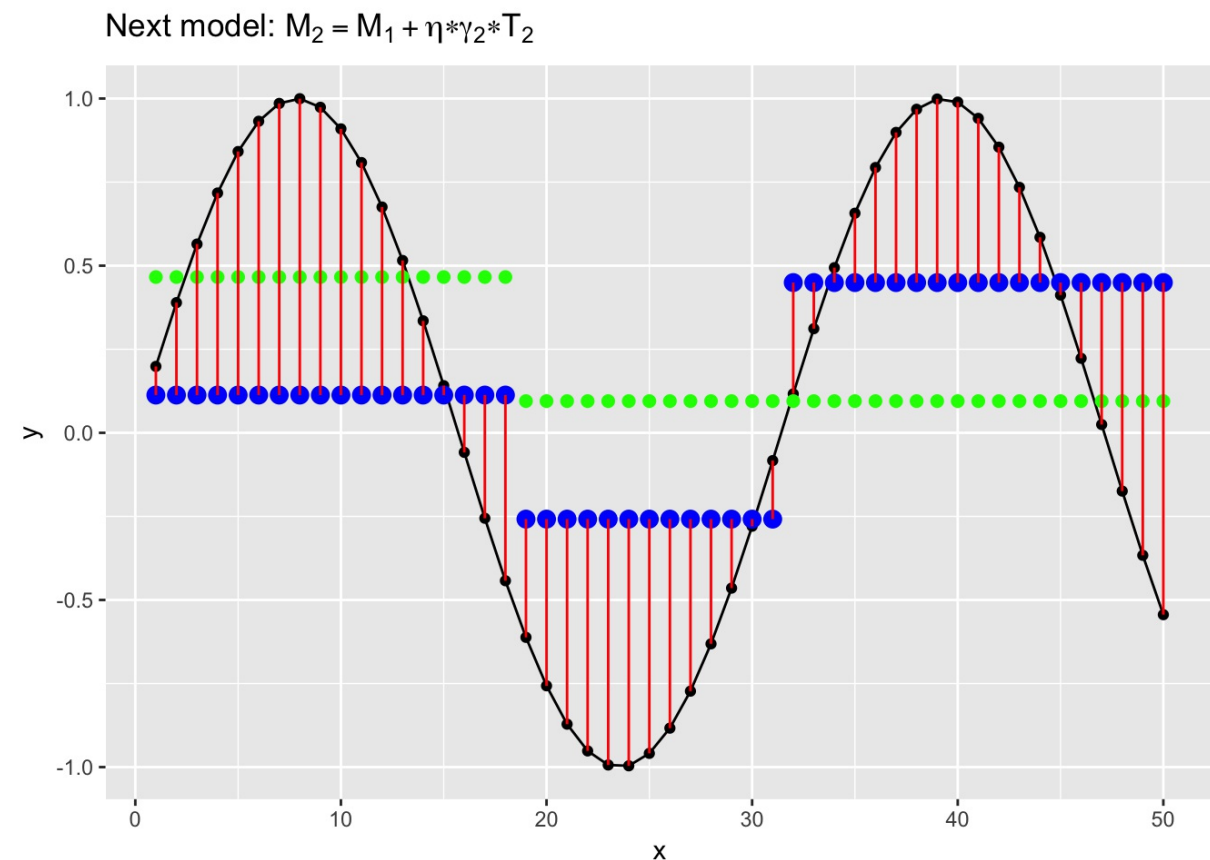
# How Gradient Boosting Works



1. Fit a shallow tree  $T_1$  to the data:  $M_1 = T_1$
2. Fit a tree  $T_2$  to the residuals. Find  $\gamma$  such that  $M_2 = M_1 + \gamma T_2$  is the best fit to data



# How Gradient Boosting Works



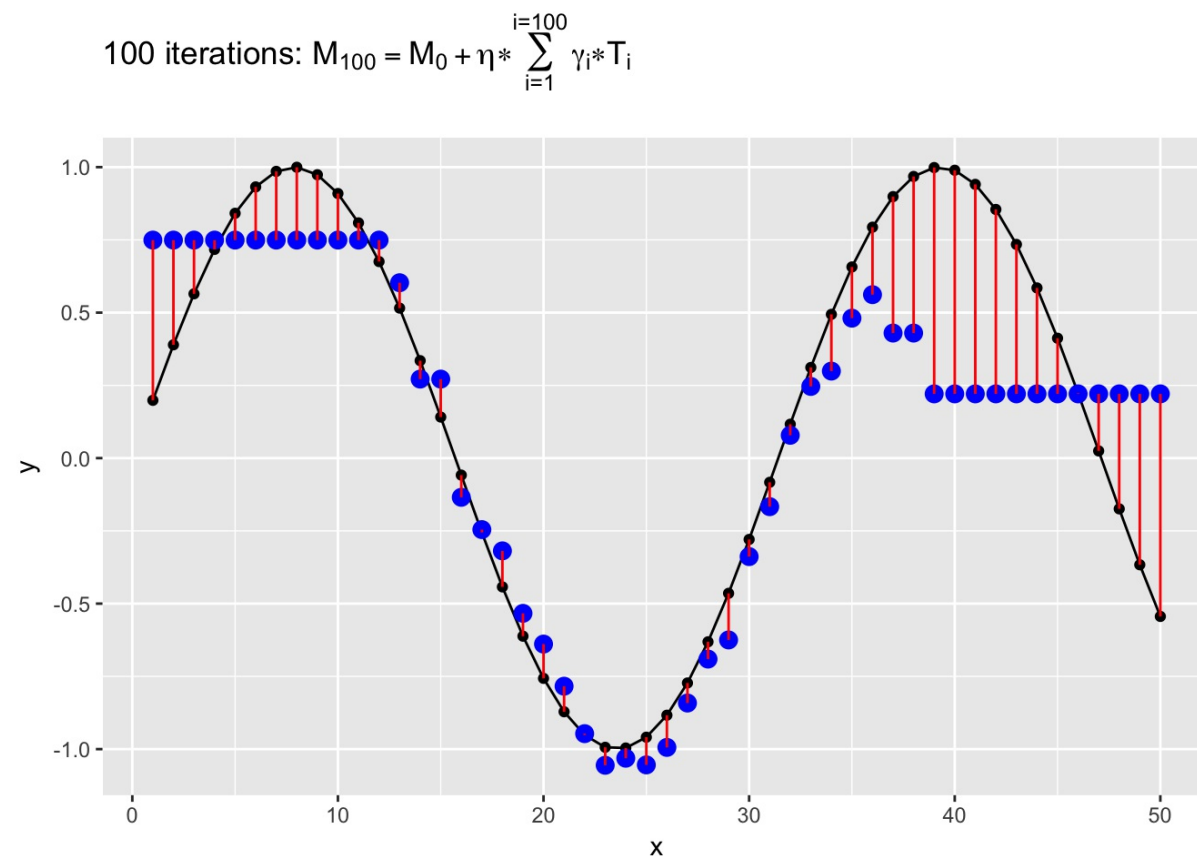
Regularization: learning rate  
 $\eta \in (0, 1)$

$$M_2 = M_1 + \eta \gamma T_2$$

- Larger  $\eta$ : faster learning
- Smaller  $\eta$ : less risk of overfit



# How Gradient Boosting Works



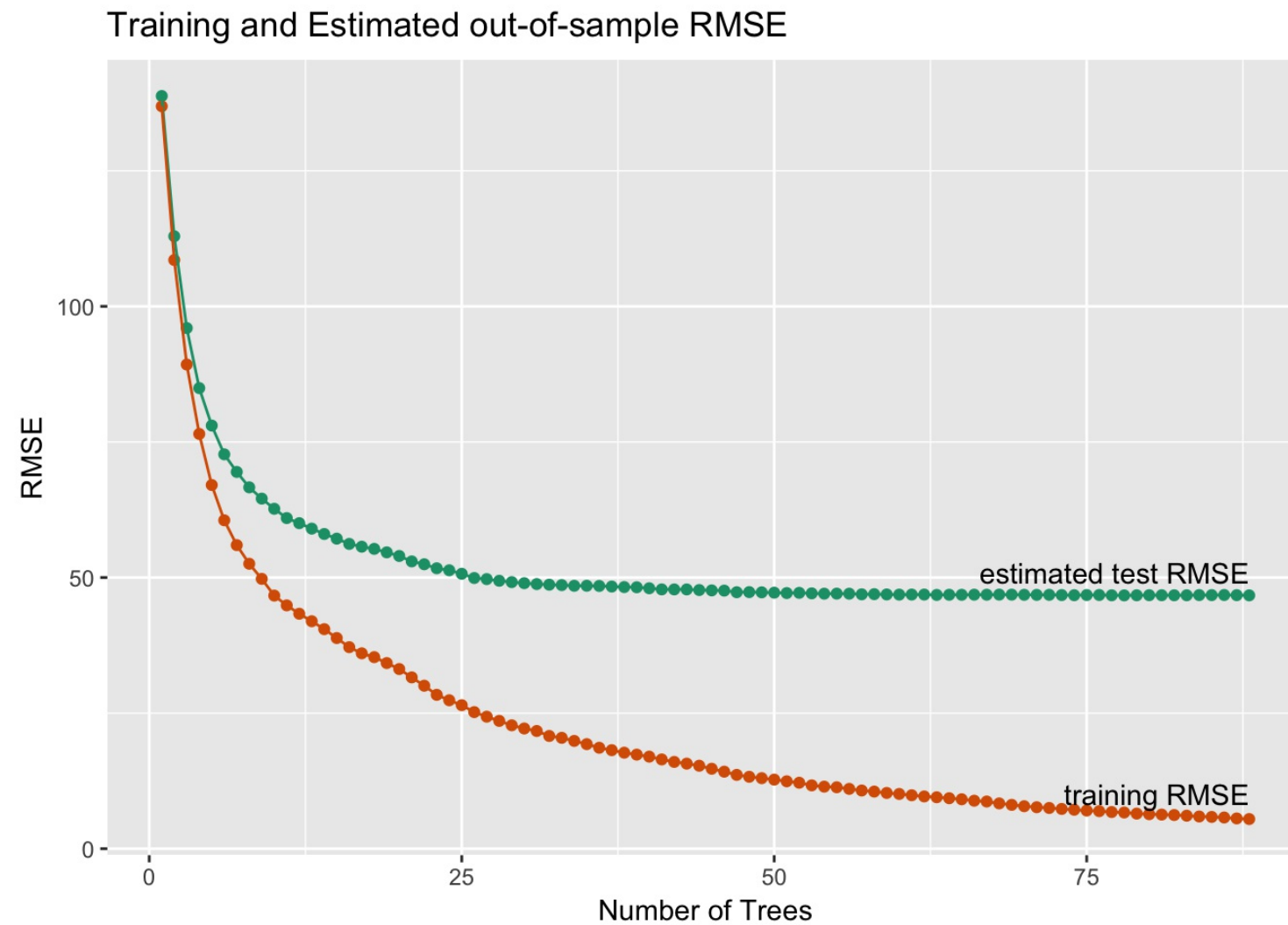
1. Fit a shallow tree  $T_1$  to the data
  - $M_1 = T_1$
2. Fit a tree  $T_2$  to the residuals.
  - $M_2 = M_1 + \eta \gamma_2 T_2$
3. Repeat (2) until stopping condition met

**Final Model:**

$$M = M_1 + \eta \sum \gamma_i T_i$$



# Cross-validation to Guard Against Overfit



Training error keeps decreasing, but test error doesn't



# Best Practice (with `xgboost()`)

1. Run `xgb.cv()` with a large number of rounds (trees).



# Best Practice (with `xgboost()`)

1. Run `xgb.cv()` with a large number of rounds (trees).
2. `xgb.cv()`\$`evaluation_log`: records estimated RMSE for each round.
  - Find the number of trees that minimizes estimated RMSE:  $n_{best}$



# Best Practice (with `xgboost()`)

1. Run `xgb.cv()` with a large number of rounds (trees).
2. `xgb.cv()`\$`evaluation_log`: records estimated RMSE for each round.
  - Find the number of trees that minimizes estimated RMSE:  $n_{best}$
3. Run `xgboost()`, setting `nrounds =  $n_{best}$`

# Example: Bike Rental Model

First, prepare the data

```
> treatplan <- designTreatmentsZ(bikesJan, vars)
> newvars <- treatplan$scoreFrame %>%
+   filter(code %in% c("clean", "lev")) %>%
+   use_series(varName)

> bikesJan.treat <- prepare(treatplan, bikesJan, varRestriction = newvars)
```

For `xgboost()`:

- Input data: `as.matrix(bikesJan.treat)`
- Outcome: `bikesJan$cnt`





# Training a model with `xgboost()` / `xgb.cv()`

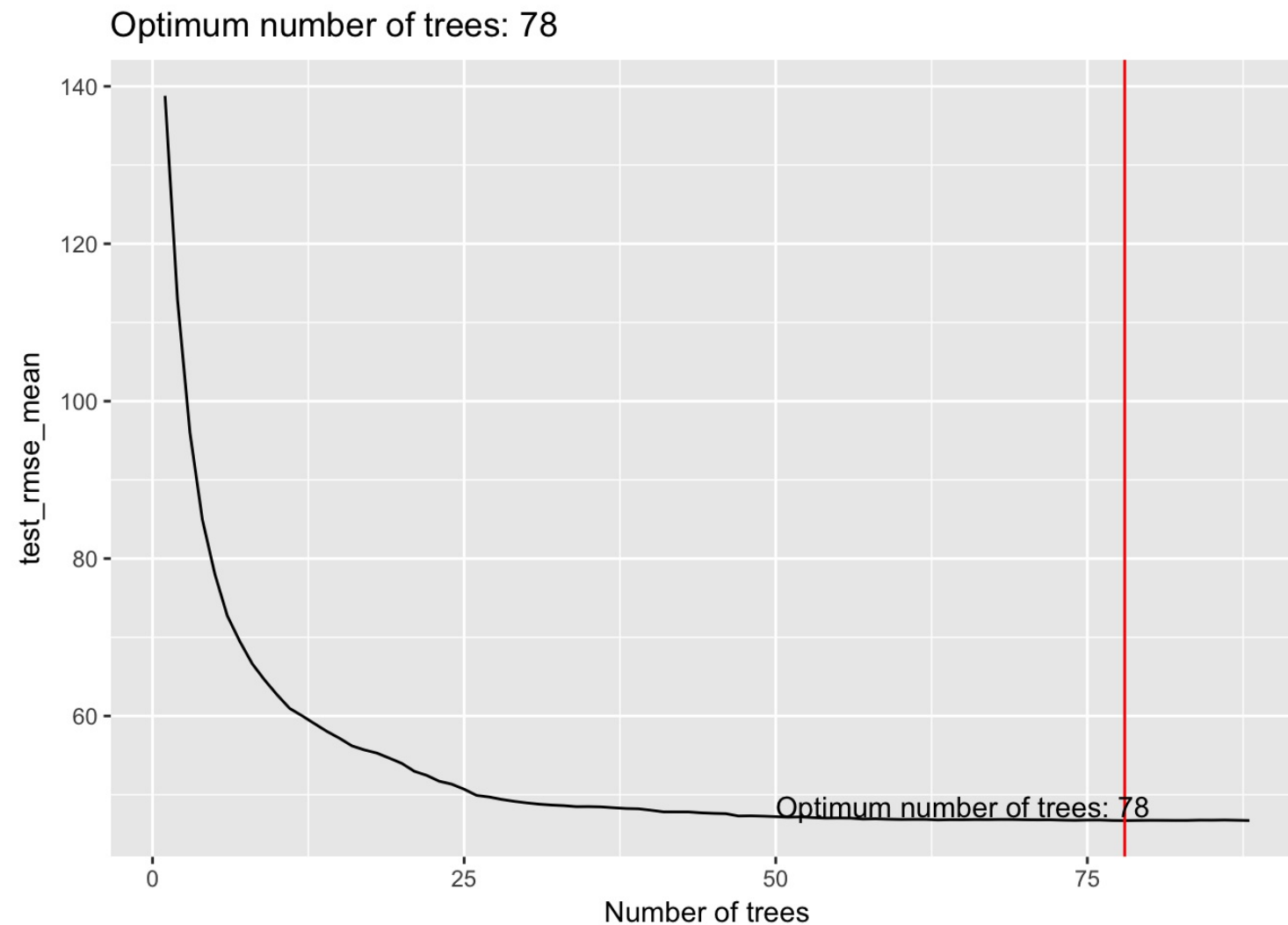
```
> cv <- xgb.cv(data = as.matrix(bikesJan.treat),  
+             label = bikesJan$cnt,  
+             objective = "reg:linear",  
+             nrounds = 100, nfold = 5, eta = 0.3, depth = 6)
```

## Key inputs to `xgb.cv()` and `xgboost()`

- data: input data as matrix ; label: outcome
- objective: for regression - "reg:linear"
- nrounds: maximum number of trees to fit
- eta: learning rate
- depth: maximum depth of individual trees
- nfold (`xgb.cv()` only): number of folds for cross validation



# Find the Right Number of Trees



```
> elog <- as.data.frame(cv$evaluation_log)
> (nrounds <- which.min(elog$test_rmse_mean))
[1] 78
```



# Run `xgboost()` for final model

```
> nrounds <- 78

> model <- xgboost(data = as.matrix(bikesJan.treat),
+                 label = bikesJan$cnt,
+                 nrounds = nrounds,
+                 objective = "reg:linear",
+                 eta = 0.3,
+                 depth = 6)
```

# Predict with an `xgboost()` model

Prepare February data, and predict

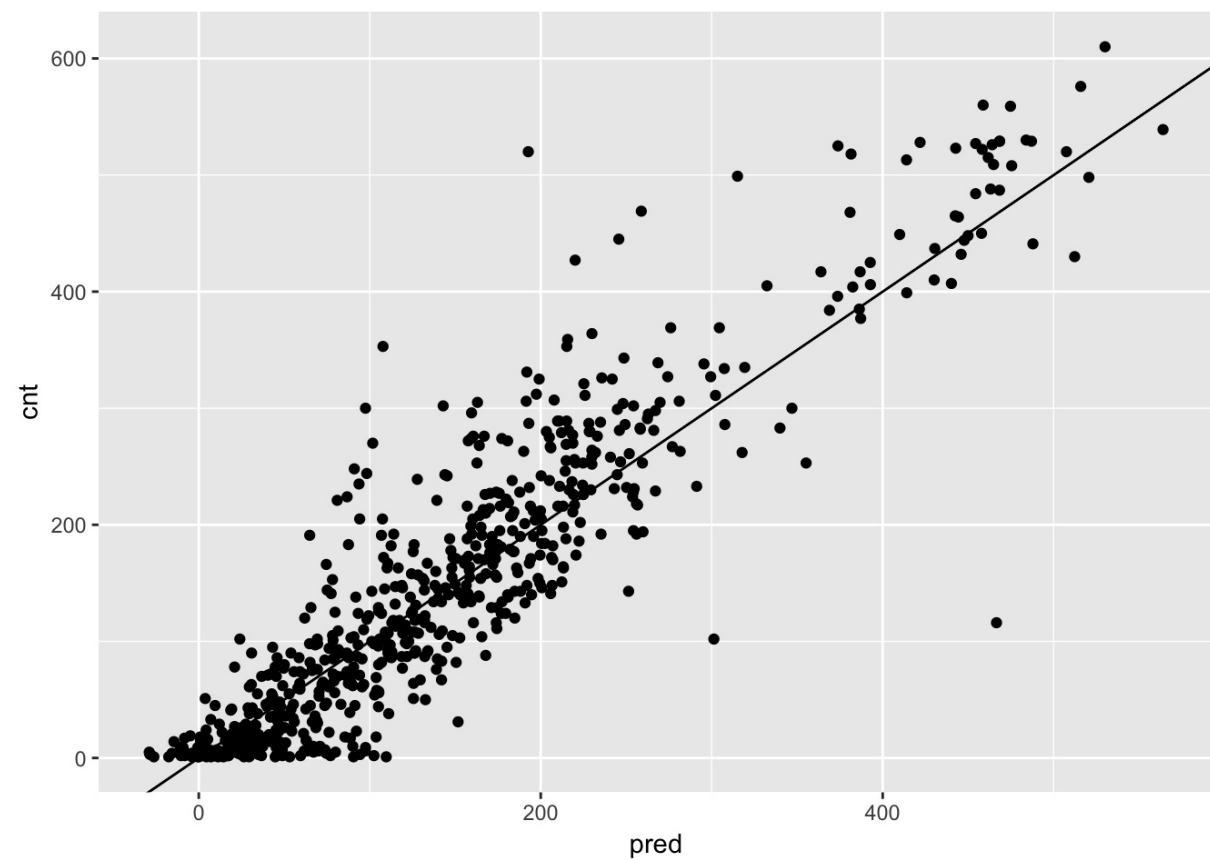
```
> bikesFeb.treat <- prepare(treatplan, bikesFeb, varRestriction = newvars)
> bikesFeb$pred <- predict(model, as.matrix(bikesFeb.treat))
```

## Model performances on February Data

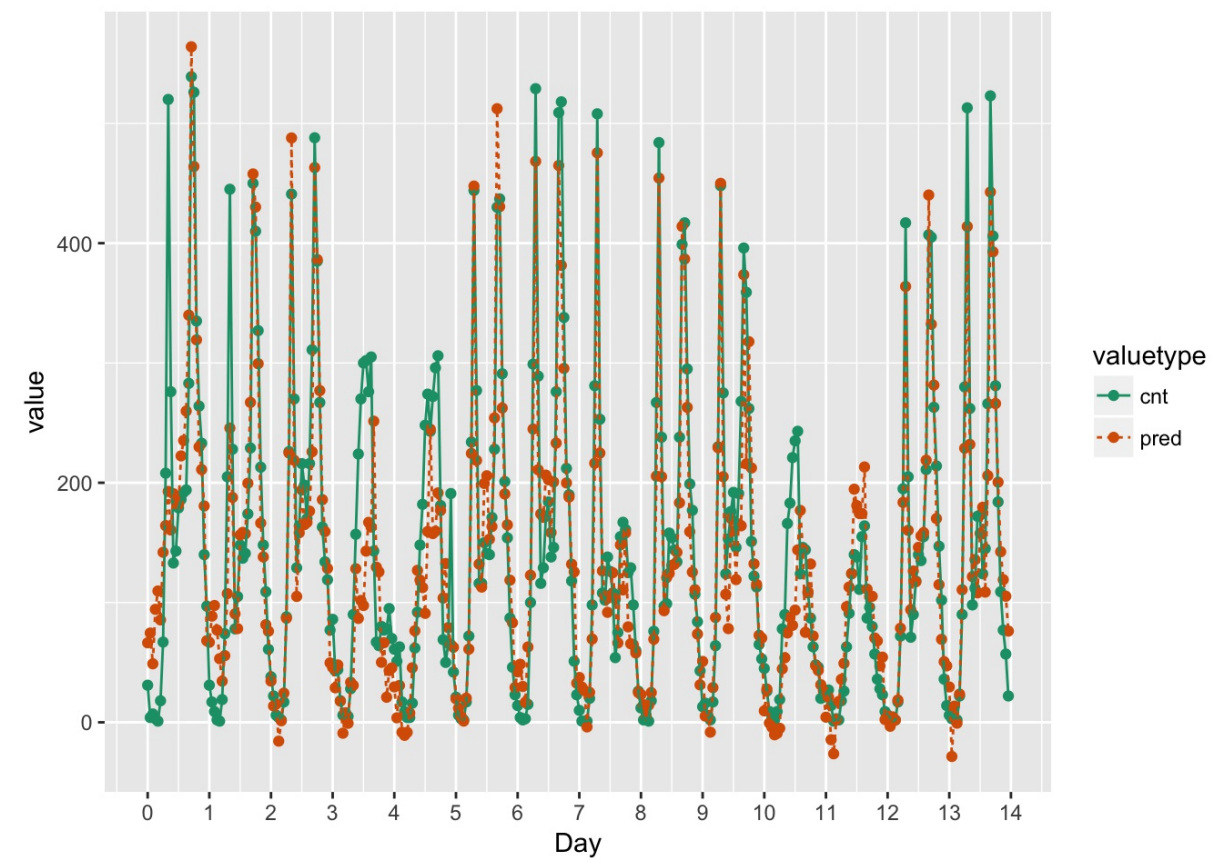
Model	RMSE
Quasipoisson	69.3
Random forests	67.15
<b>Gradient Boosting</b>	<b>54.0</b>

# Visualize the Results

Predictions vs. Actual Bike Rentals, February



Predictions and Hourly Bike Rentals, February





## SUPERVISED LEARNING IN R: REGRESSION

**Let's practice!**