Logistic regression to predict probabilities

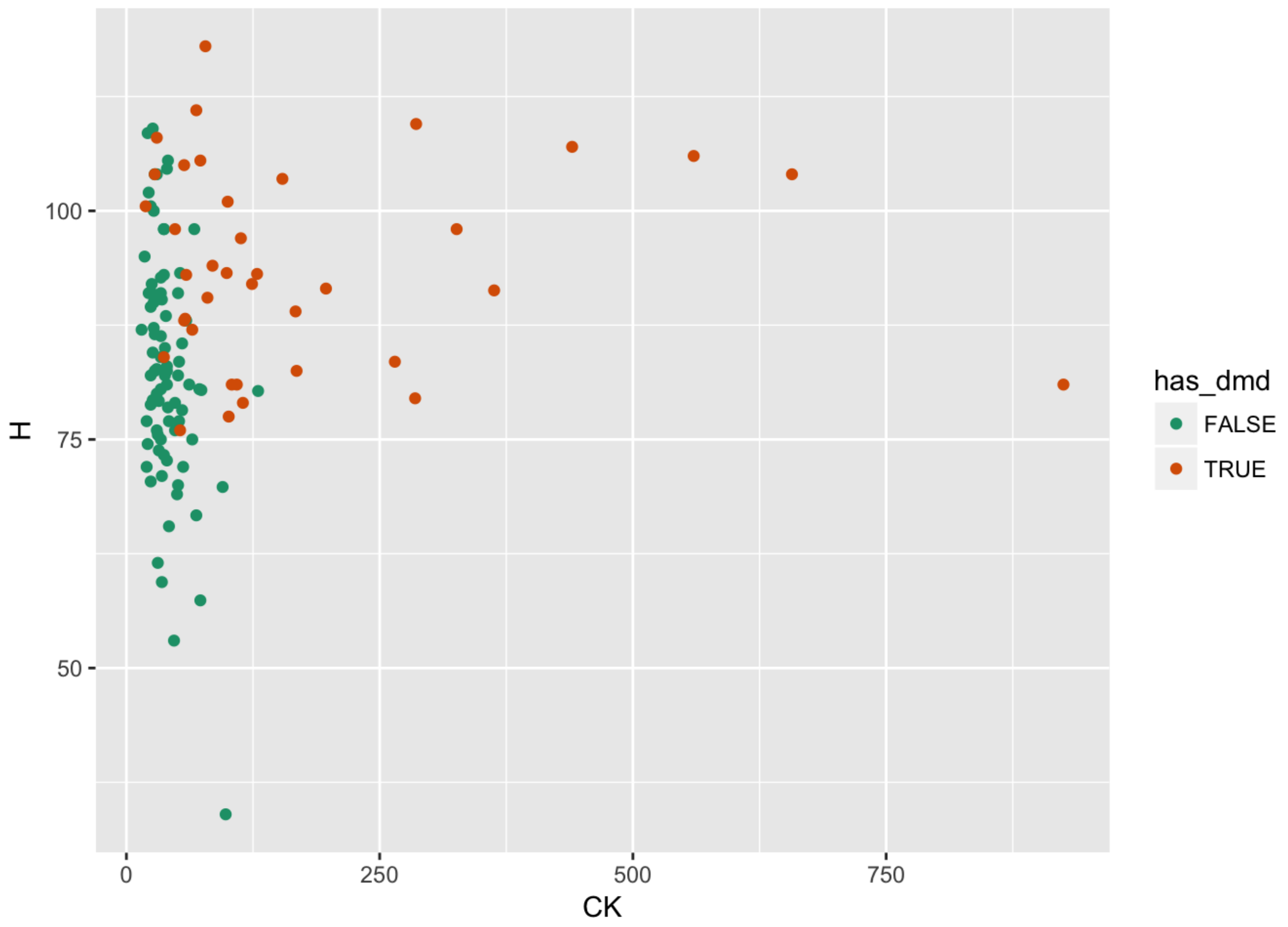
SUPERVISED LEARNING IN R: REGRESSION

Nina Zumel and John Mount
Win-Vector LLC

Predicting Probabilities

- Predicting *whether* an event occurs (yes/no): **classification**
- Predicting *the probability* that an event occurs: **regression**
- Linear regression: predicts values in $[-\infty, \infty]$
- Probabilities: limited to [0,1] interval
  - So we'll call it non-linear

Example: Predicting Duchenne Muscular Dystrophy (DMD)



- outcome: `has_dmd`   inputs: `CK` , `H`

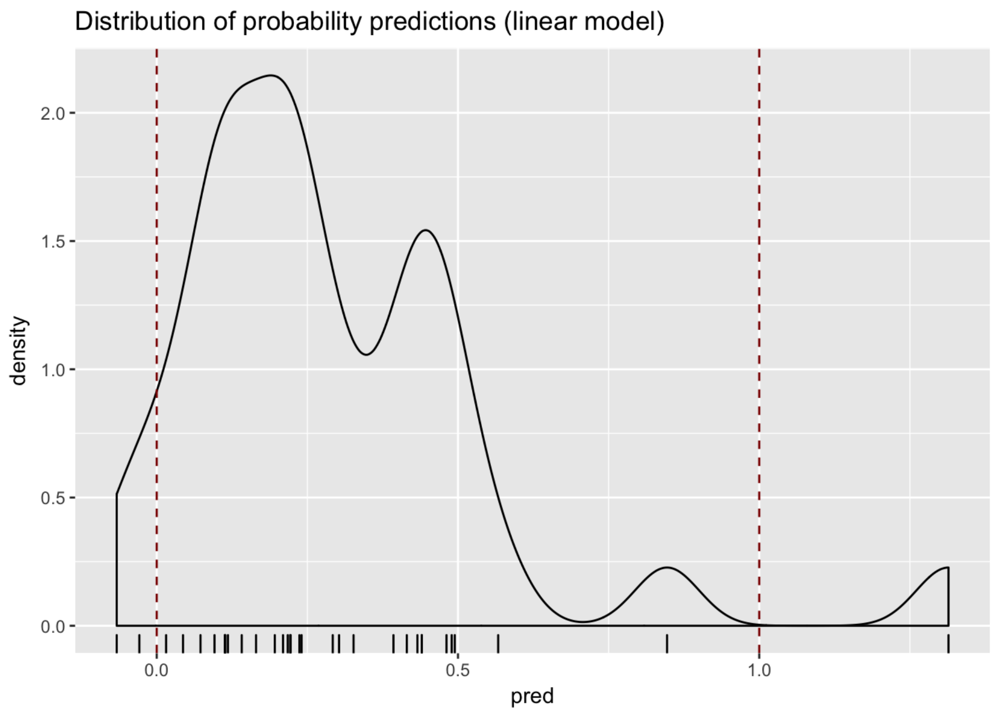
A Linear Regression Model

```
model <- lm(has_dmd ~ CK + H,
            data = train)

test$pred <- predict(
    model,
    newdata = test
)
```

outcome: `has_dmd` $\in \{0,1\}$

- 0: FALSE
- 1: TRUE

Model predicts values outside the range [0:1]



Logistic Regression

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...$$

```
glm(formula, data, family = binomial)
```

- Generalized linear model
- Assumes inputs additive, linear in *log-odds*: $log(p/(1-p))$
- family: describes error distribution of the model
  - logistic regression: `family = binomial`

DMD model

```
model <- glm(has_dmd ~ CK + H, data = train, family = binomial)
```

- outcome: two classes, e.g. a and b
- model returns $Prob(b)$
  - Recommend: 0/1 or FALSE/TRUE

Interpreting Logistic Regression Models

```
model
```

```
Call:  glm(formula = has_dmd ~ CK + H, family = binomial, data = train)

Coefficients:
(Intercept)           CK            H
  -16.22046      0.07128      0.12552

Degrees of Freedom: 86 Total (i.e. Null);  84 Residual
Null Deviance:       110.8
Residual Deviance: 45.16     AIC: 51.16
```

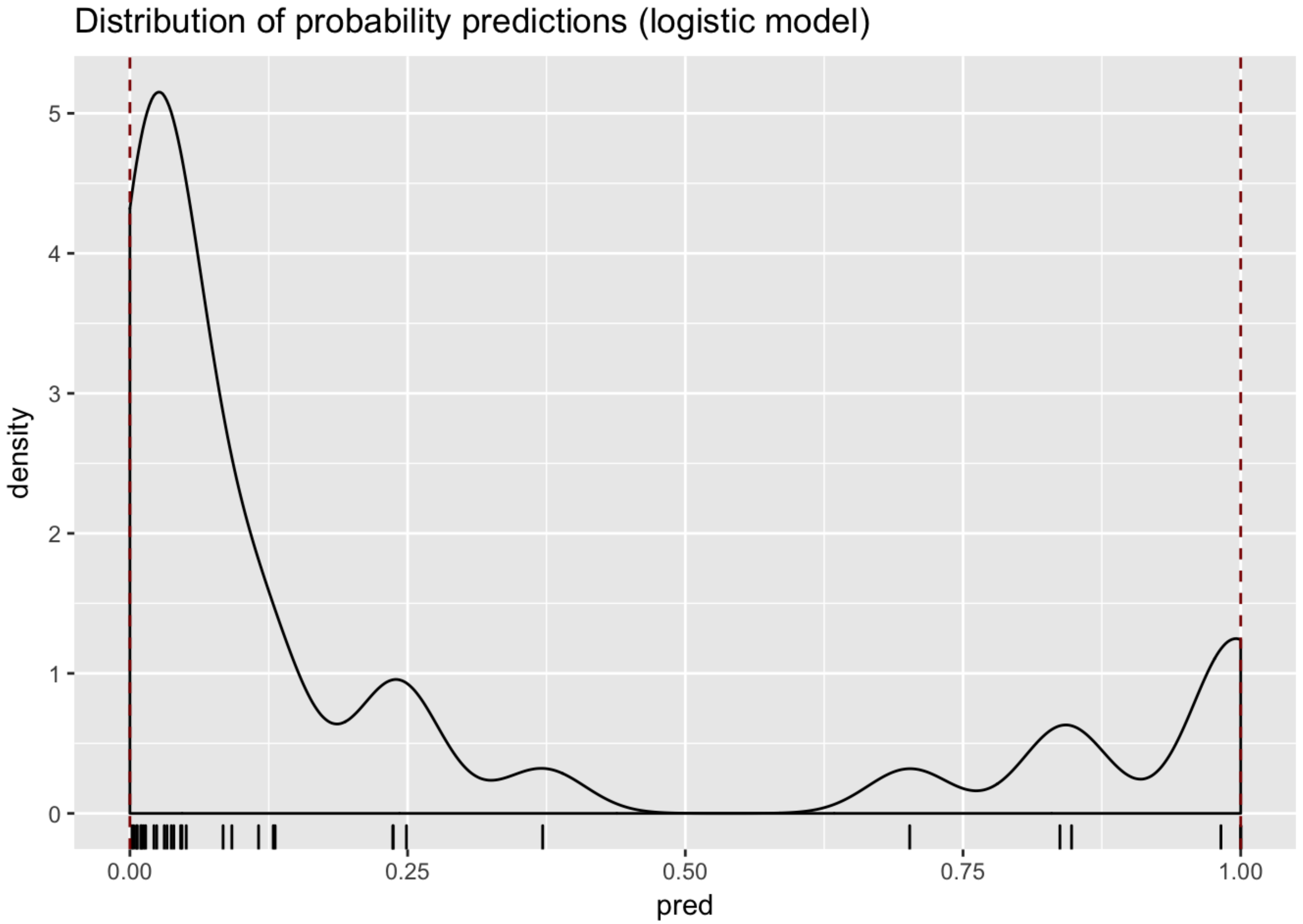
Predicting with a glm() model

```
predict(model, newdata, type = "response")
```

- `newdata` : by default, training data
- To get probabilities: use `type = "response"`
  - By default: returns log-odds

DMD Model

```
model <- glm(has_dmd ~ CK + H, data = train, family = binomial)
test$pred <- predict(model, newdata = test, type = "response")
```



Evaluating a logistic regression model: pseudo-R^2

$$R^2 = 1 - \frac{RSS}{SS_{Tot}}$$

$$pseudoR^2 = 1 - \frac{deviance}{null.deviance}$$

- Deviance: analogous to variance (RSS)
- Null deviance: Similar to SS_{Tot}
- pseudo R^2: Deviance explained

Pseudo-R^2 on Training data

Using `broom::glance()`

```
glance(model) %>%
  summarize(pR2 = 1 - deviance/null.deviance)
```

```
   pseudoR2
1 0.5922402
```

Using `sigr::wrapChiSqTest()`

```
wrapChiSqTest(model)
```

```
"... pseudo-R2=0.59 ..."
```

Pseudo-R^2 on Test data

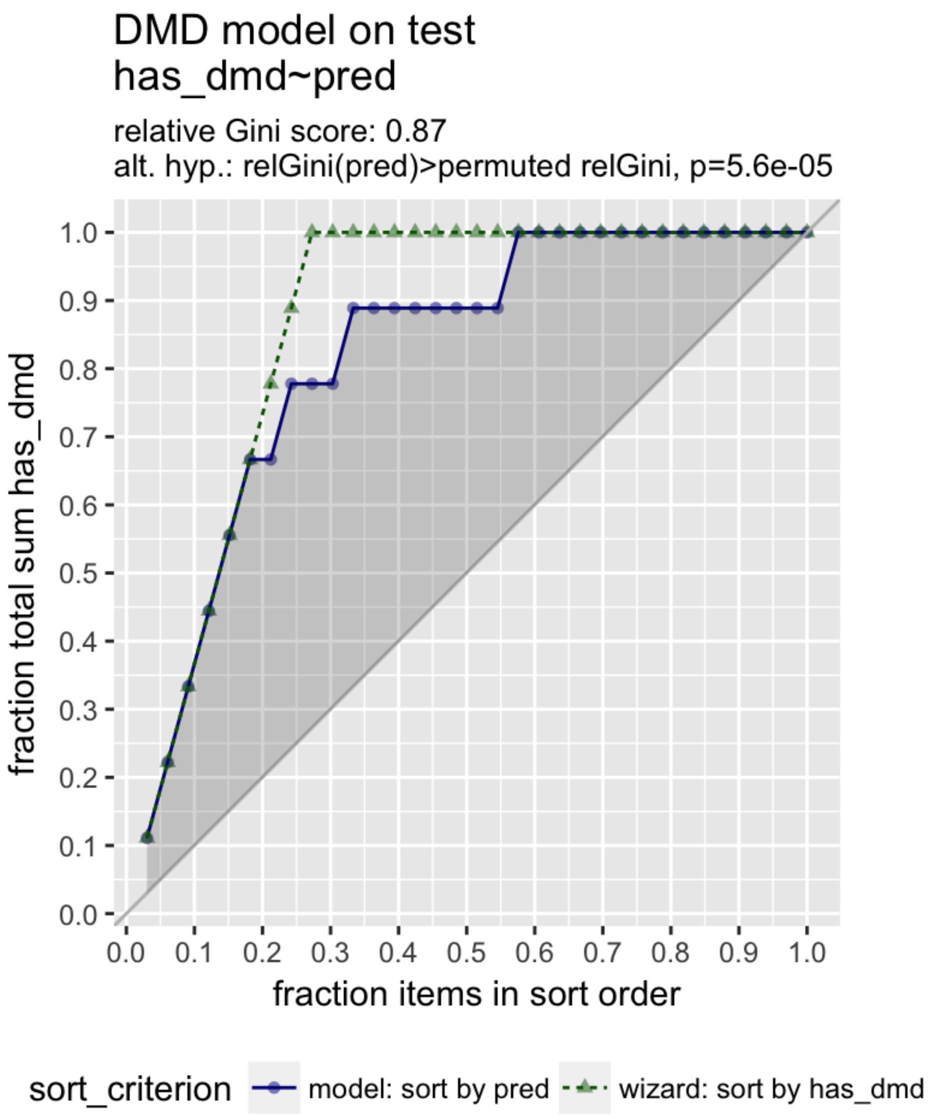
```
# Test data
test %>%
  mutate(pred = predict(model, newdata = test, type = "response")) %>%
  wrapChiSqTest("pred", "has_dmd", TRUE)
```

Arguments:

- data frame
- prediction column name
- outcome column name
- target value (target event)

The Gain Curve Plot

```
GainCurvePlot(test, "pred","has_dmd", "DMD model on test")
```



Let's practice!

SUPERVISED LEARNING IN R: REGRESSION

Poisson and quasipoisson regression to predict counts

SUPERVISED LEARNING IN R: REGRESSION

Nina Zumel and John Mount
Win-Vector, LLC

Predicting Counts

- Linear regression: predicts values in $[-\infty, \infty]$
- Counts: integers in range $[0, \infty]$

Poisson/Quasipoisson Regression

```
glm(formula, data, family)
```

- family: either `poisson` or `quasipoisson`
- inputs additive and linear in log(count)

Poisson/Quasipoisson Regression

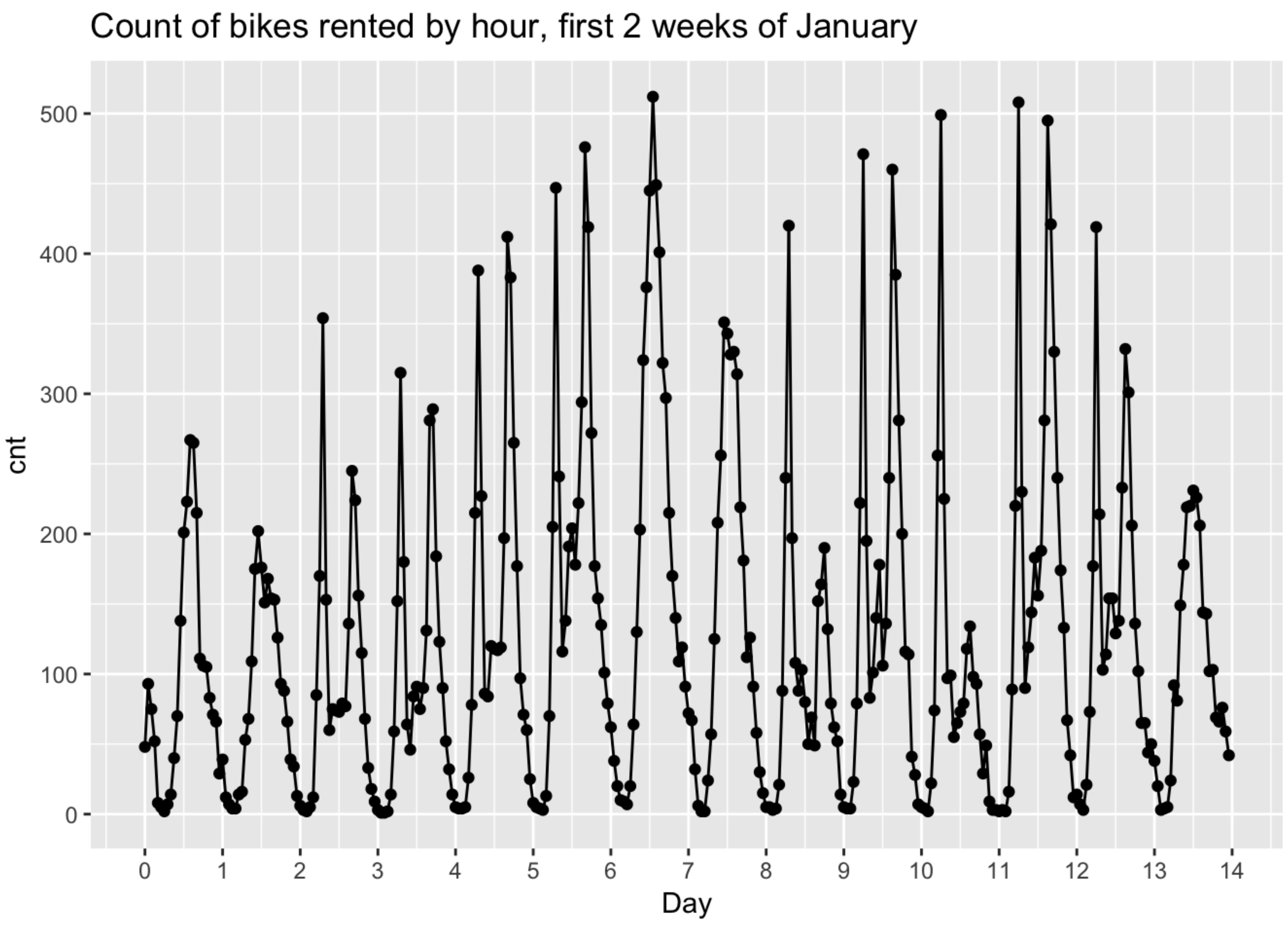
```
glm(formula, data, family)
```

- family: either `poisson` or `quasipoisson`
- inputs additive and linear in log(count)
- outcome: *integer*
  - counts: e.g. number of traffic tickets a driver gets
  - rates: e.g. number of website hits/day
- prediction: expected *rate* or *intensity* (not integral)
  - expected # traffic tickets; expected hits/day

Poisson vs. Quasipoisson

- Poisson assumes that `mean(y) = var(y)`
- If `var(y)` much different from `mean(y)` - quasipoisson
- Generally requires a large sample size
- If rates/counts >> 0 - regular regression is fine

Example: Predicting Bike Rentals



Fit the model

```
bikesJan %>%
  summarize(mean = mean(cnt), var = var(cnt))
```

```
      mean      var
1 130.5587 14351.25
```

Since `var(cnt)` >> `mean(cnt)` $\rightarrow$ *use quasipoisson*

```
fmla <- cnt ~ hr + holiday + workingday +
  weathersit + temp + atemp + hum + windspeed

model <- glm(fmla, data = bikesJan, family = quasipoisson)
```

Check model fit

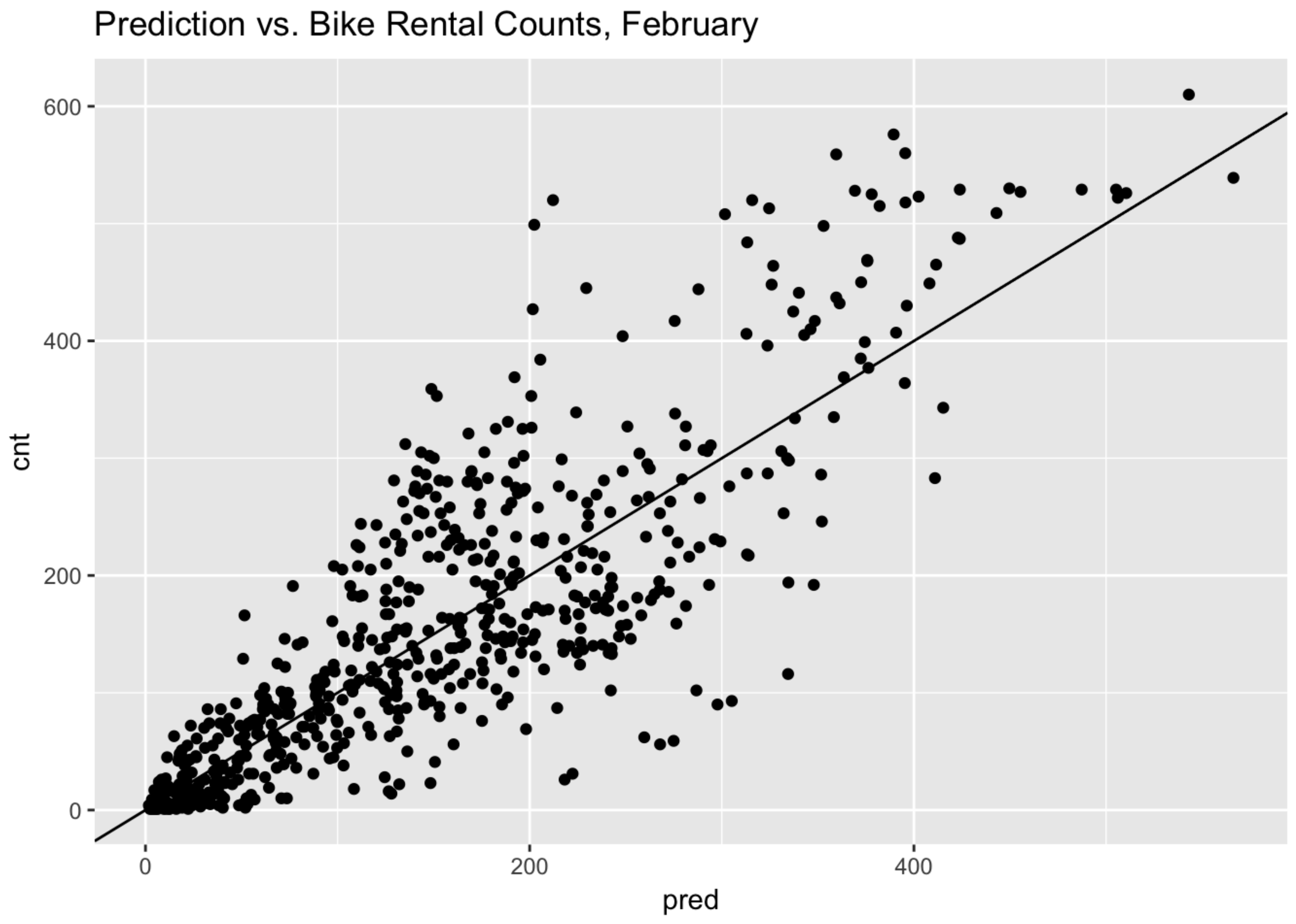
$$pseudoR^2 = 1 - \frac{deviance}{null.deviance}$$

```
glance(model) %>%
  summarize(pseudoR2 = 1 - deviance/null.deviance)
```

```
   pseudoR2
1 0.7654358
```

Predicting from the model

```
predict(model, newdata = bikesFeb, type = "response")
```



Evaluate the model

You can evaluate count models by RMSE

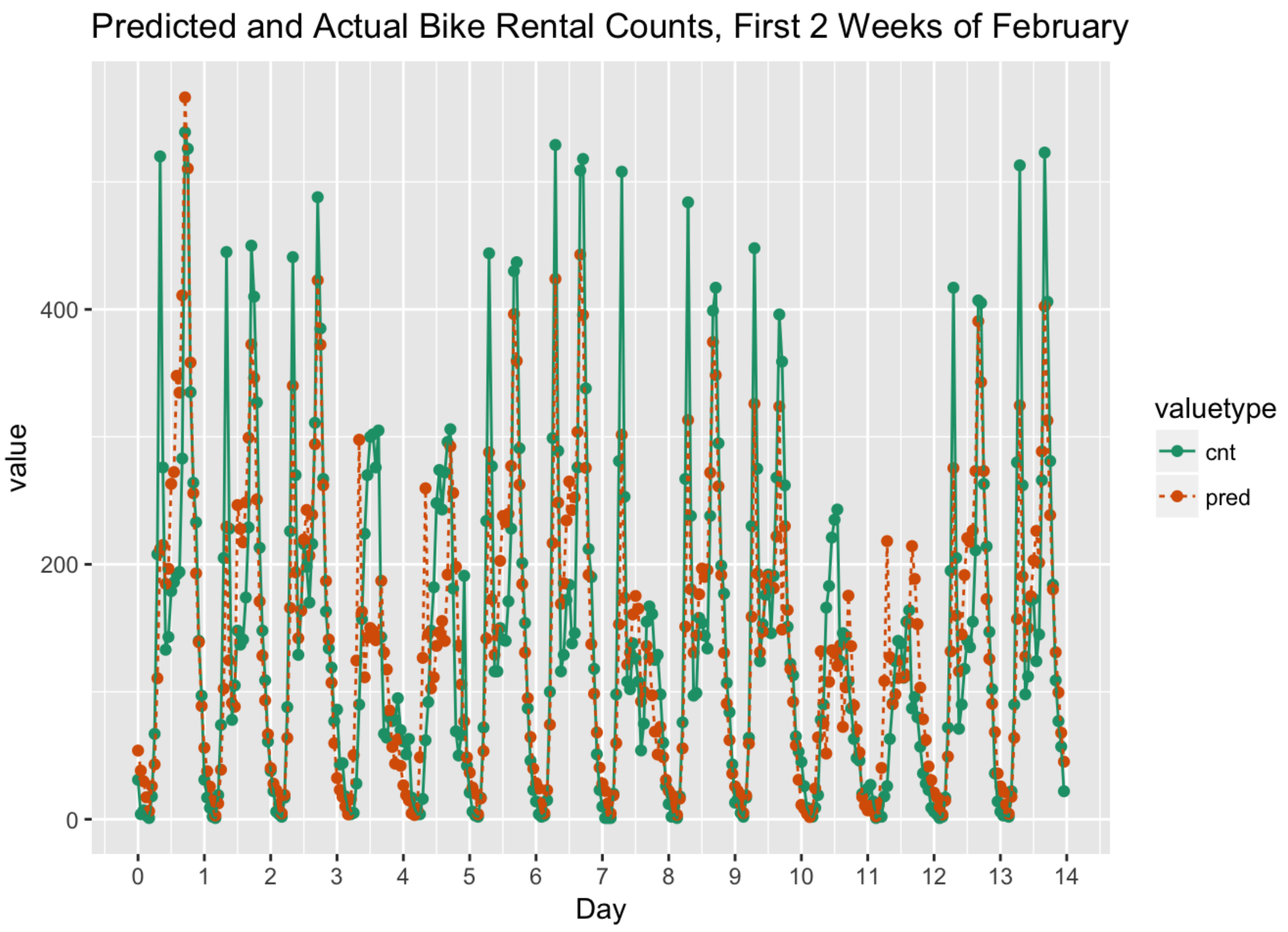
```
bikesFeb %>%
  mutate(residual = pred - cnt) %>%
  summarize(rmse = sqrt(mean(residual^2)))
```

```
      rmse
1 69.32869
```

```
sd(bikesFeb$cnt)
```

```
134.2865
```

Compare Predictions and Actual Outcomes



Let's practice!

SUPERVISED LEARNING IN R: REGRESSION

GAM to learn non-linear transformations

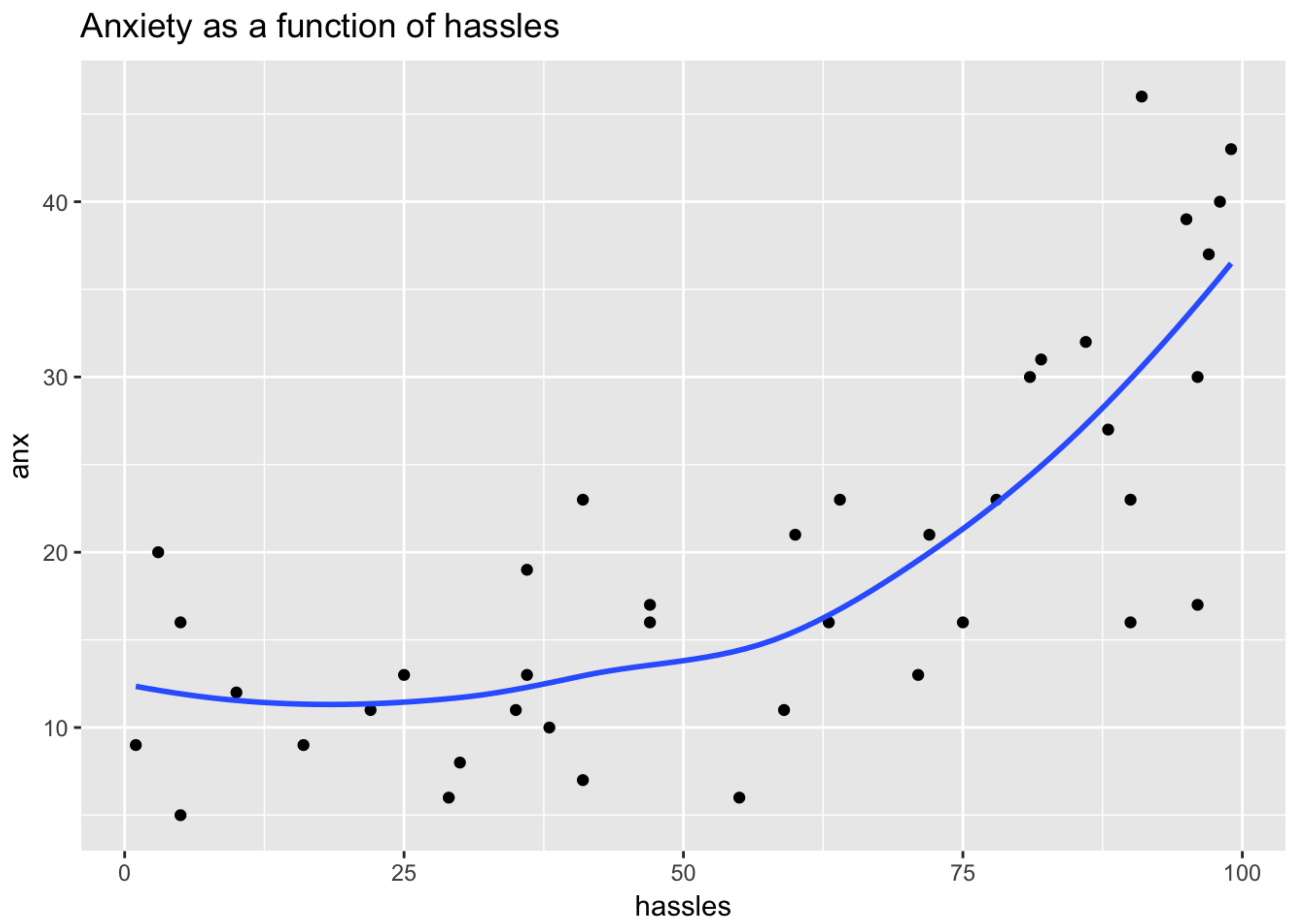
SUPERVISED LEARNING IN R: REGRESSION

Nina Zumel and John Mount
Win-Vector, LLC

Generalized Additive Models (GAMs)

$$y \sim b0 + s1(x1) + s2(x2) + ...$$

Learning Non-linear Relationships



gam() in the mgcv package

```
gam(formula, family, data)
```

family:

- gaussian (default): "regular" regression
- binomial: probabilities
- poisson/quasipoisson: counts

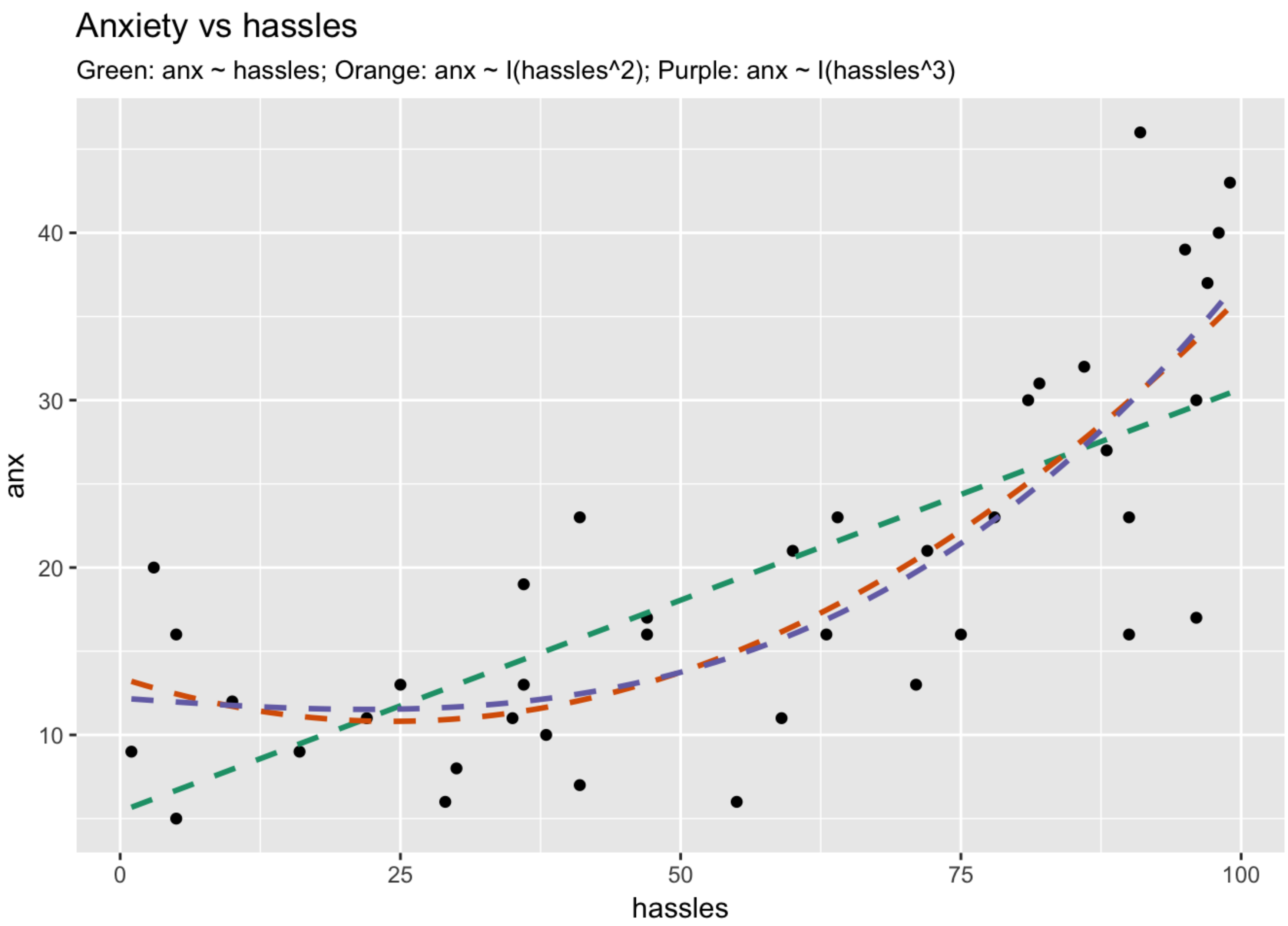
Best for larger datasets

The s() function

```
anx ~ s(hassles)
```

- `s()` designates that variable should be non-linear
- Use `s()` with continuous variables
  - More than about 10 unique values

Revisit the hassles data



Revisit the hassles data

Model	RMSE (cross-val)	R^2 (training)
Linear ($hassles$)	7.69	0.53
Quadratic ($hassles^2$)	6.89	0.63
Cubic ($hassles^3$)	**6.70**	**0.65**

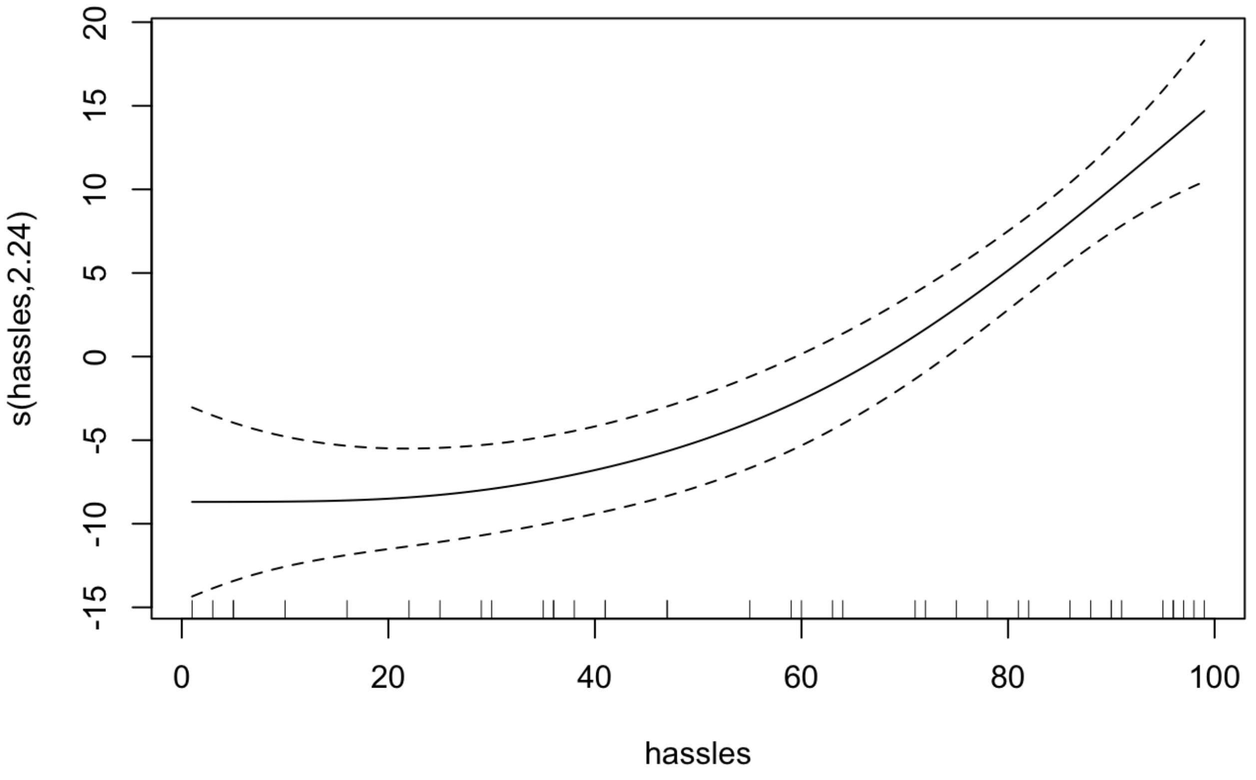
GAM of the hassles data

```
model <- gam(
  anx ~ s(hassles),
  data = hassleframe,
  family = gaussian
)

summary(model)
```

```
...

R-sq.(adj) =  0.619   Deviance explained = 64.1%
GCV = 49.132  Scale est. = 45.153    n = 40
```

Examining the Transformations

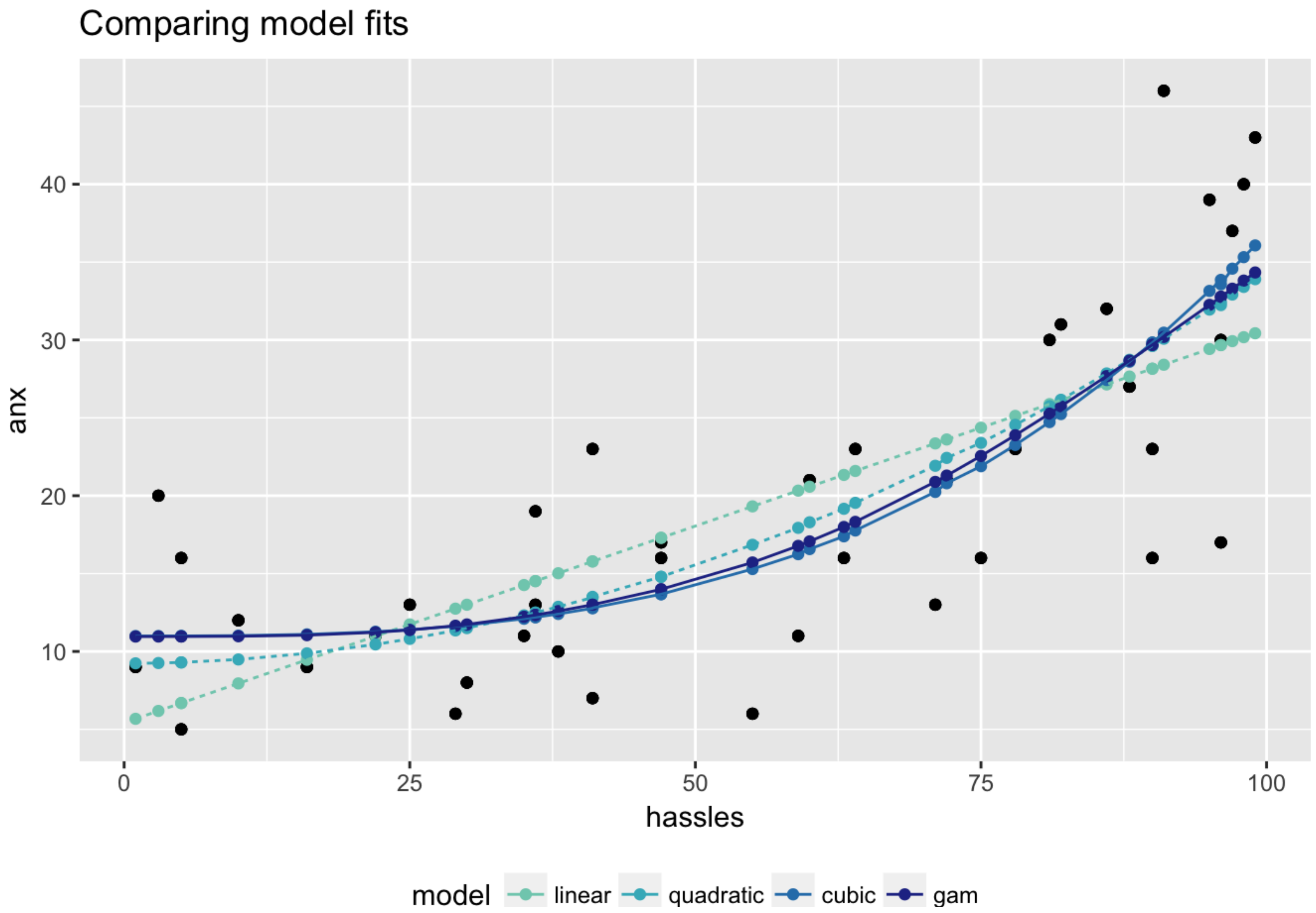
```
plot(model)
```



y values: `predict(model, type = "terms")`

Predicting with the Model

```
predict(model, newdata = hassleframe, type = "response")
```



Comparing out-of-sample performance

Knowing the correct transformation is best, but GAM is useful when transformation isn't known

Model	RMSE (cross-val)	R^2 (training)
Linear ($hassles$)	7.69	0.53
Quadratic ($hassles^2$)	6.89	0.63
Cubic ($hassles^3$)	6.70	0.65
GAM	**7.06**	0.64

- Small dataset $\rightarrow$ noisier GAM

Let's practice!

SUPERVISED LEARNING IN R: REGRESSION