



SUPERVISED LEARNING IN R: REGRESSION

Welcome and Introduction

Nina Zumel and John Mount
Data Scientists, Win Vector LLC



What is Regression?

Regression: Predict a numerical outcome ("dependent variable") from a set of inputs ("independent variables").

- *Statistical Sense*: Predicting the expected value of the outcome.
- *Casual Sense*: Predicting a numerical outcome, rather than a discrete one.

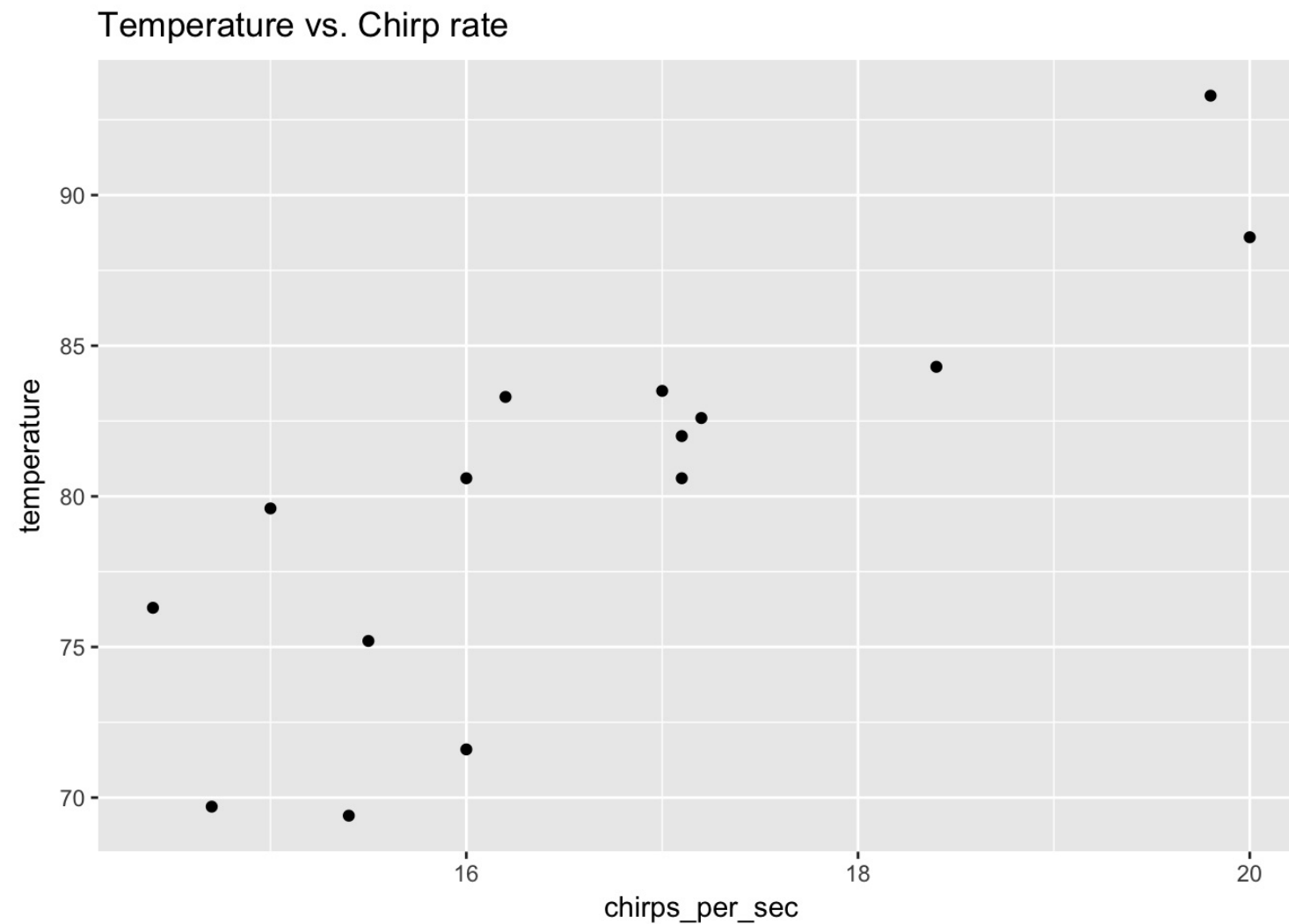


What is Regression?

- *How many units will we sell?* (**Regression**)
- *Will this customer buy our product (yes/no)?* (**Classification**)
- *What price will the customer pay for our product?* (**Regression**)

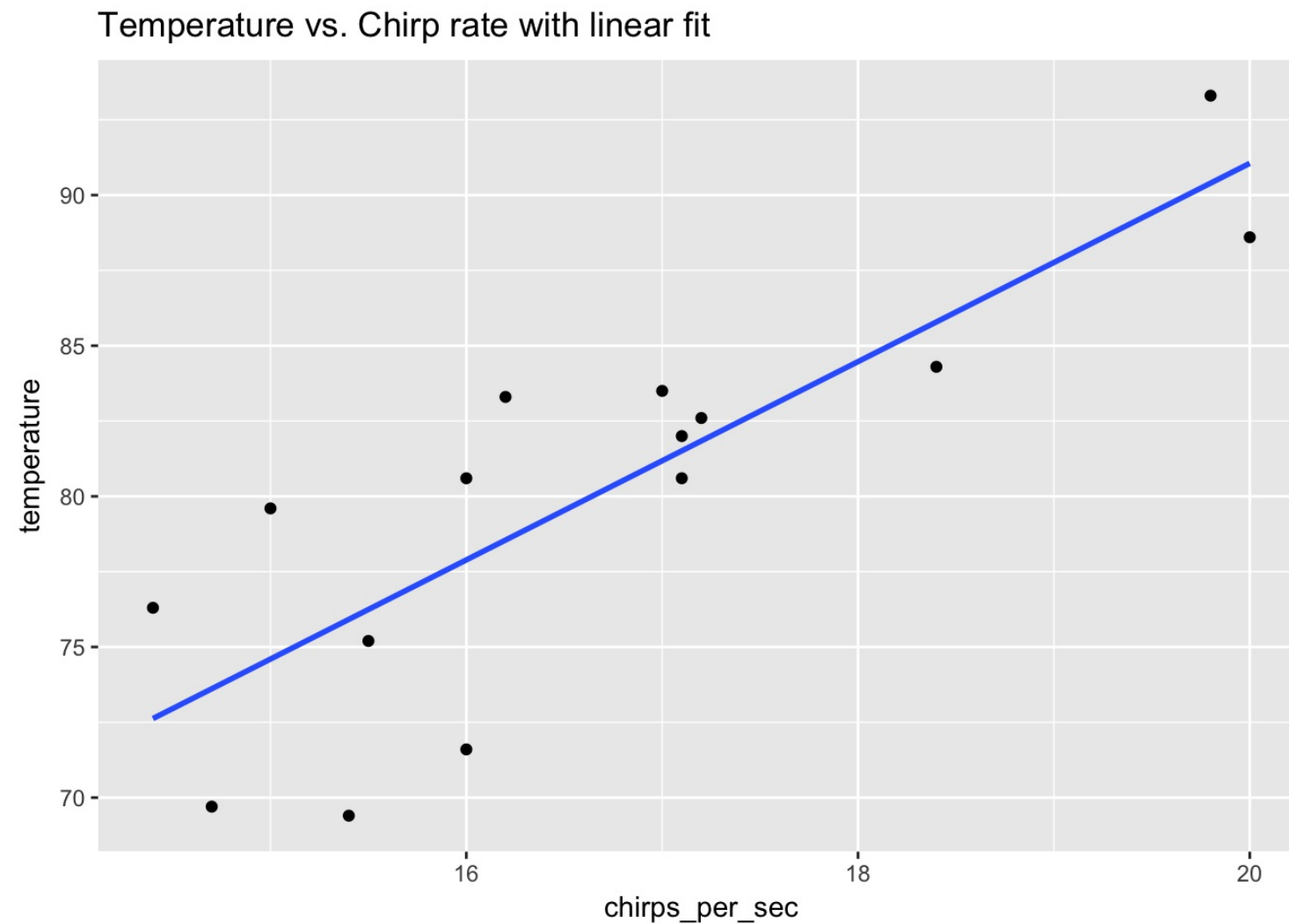


Example: Predict Temperature from Chirp Rate



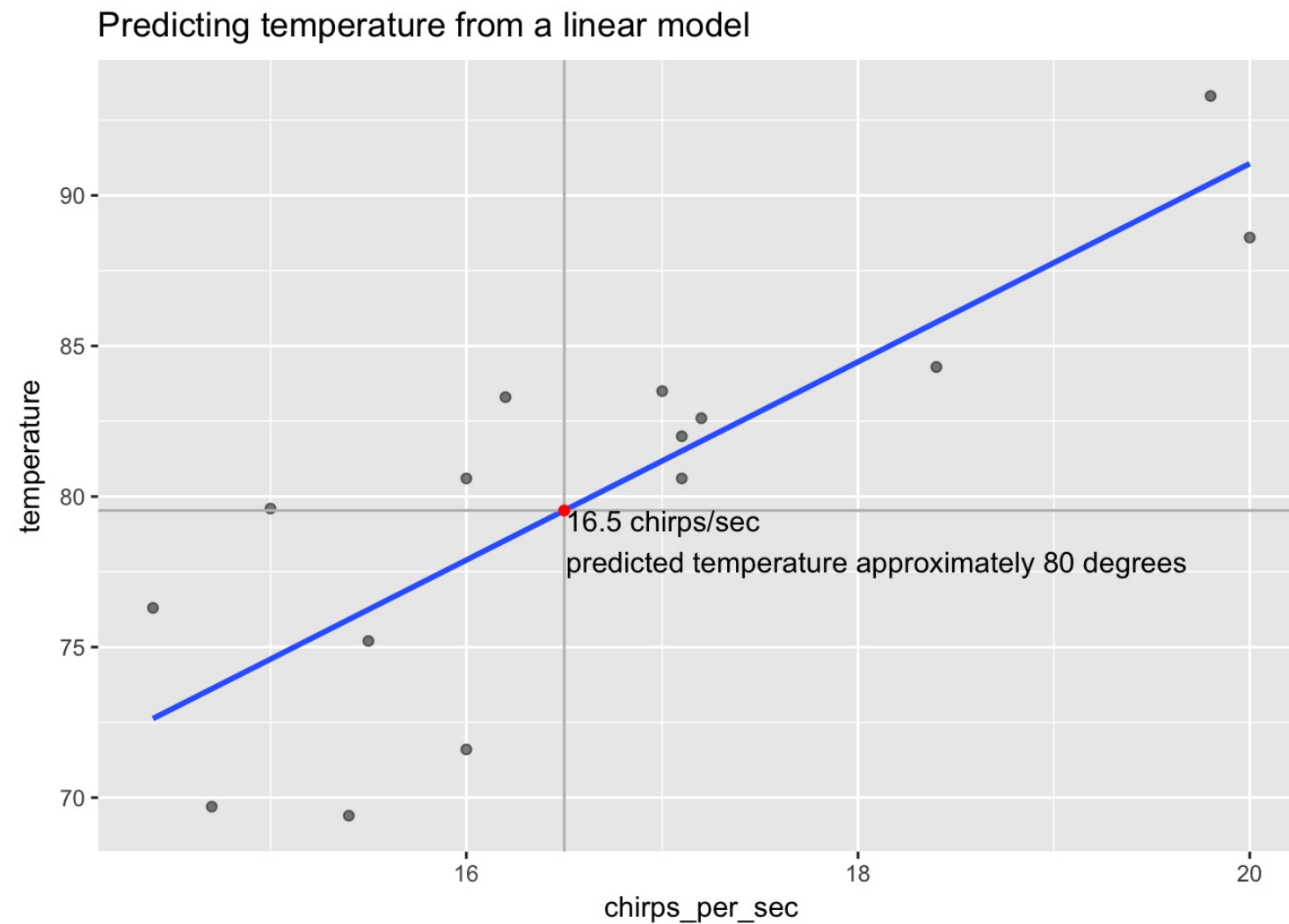


Predict Temperature from Chirp Rate





Predict Temperature from Chirp Rate





Regression from a Machine Learning Perspective

- *Scientific mindset*: Modeling to understand the data generation process
- *Engineering mindset*: *Modeling to predict accurately

Machine Learning: Engineering mindset



SUPERVISED LEARNING IN R: REGRESSION

Let's practice!



SUPERVISED LEARNING IN R: REGRESSION

Linear regression - the fundamental method

Nina Zumel and John Mount
Win-Vector LLC



Linear Regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots$$

- y is *linearly* related to each x_i
- Each x_i contributes *additively* to y



Linear Regression in R: lm()

```
cmodel <- lm(temperature ~ chirps_per_sec, data = cricket)
```

- formula: temperature ~ chirps_per_sec
- data frame: cricket

Formulas

```
fmla <- temperature ~ chirps_per_sec  
fmla <- blood_pressure ~ age + weight
```

- LHS: outcome
- RHS: inputs
 - use + for multiple inputs

```
fmla <- as.formula("temperature ~ chirps_per_sec")
```



Looking at the Model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots$$

```
cmodel

##
## Call:
## lm(formula = temperature ~ chirps_per_sec, data = cricket)
##
## Coefficients:
## (Intercept) chirps_per_sec
##      25.232      3.291
```

More Information about the Model

```
summary(cmodel)
```

```
## Call:
## lm(formula = fmla, data = cricket)
##
## Residuals:
##    Min     1Q  Median     3Q    Max
## -6.515 -1.971  0.490  2.807  5.001
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   25.2323    10.0601   2.508 0.026183 *
## chirps_per_sec  3.2911     0.6012   5.475 0.000107 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.829 on 13 degrees of freedom
## Multiple R-squared:  0.6975, Adjusted R-squared:  0.6742
## F-statistic: 29.97 on 1 and 13 DF, p-value: 0.0001067
```

```
broom::glance(cmodel)
```

```
sigr::wrapFTest(cmodel)
```



SUPERVISED LEARNING IN R: REGRESSION

Let's practice!



SUPERVISED LEARNING IN R: REGRESSION

Predicting once you fit a model

Nina Zumel and John Mount
Win-Vector LLC



Predicting From the Training Data

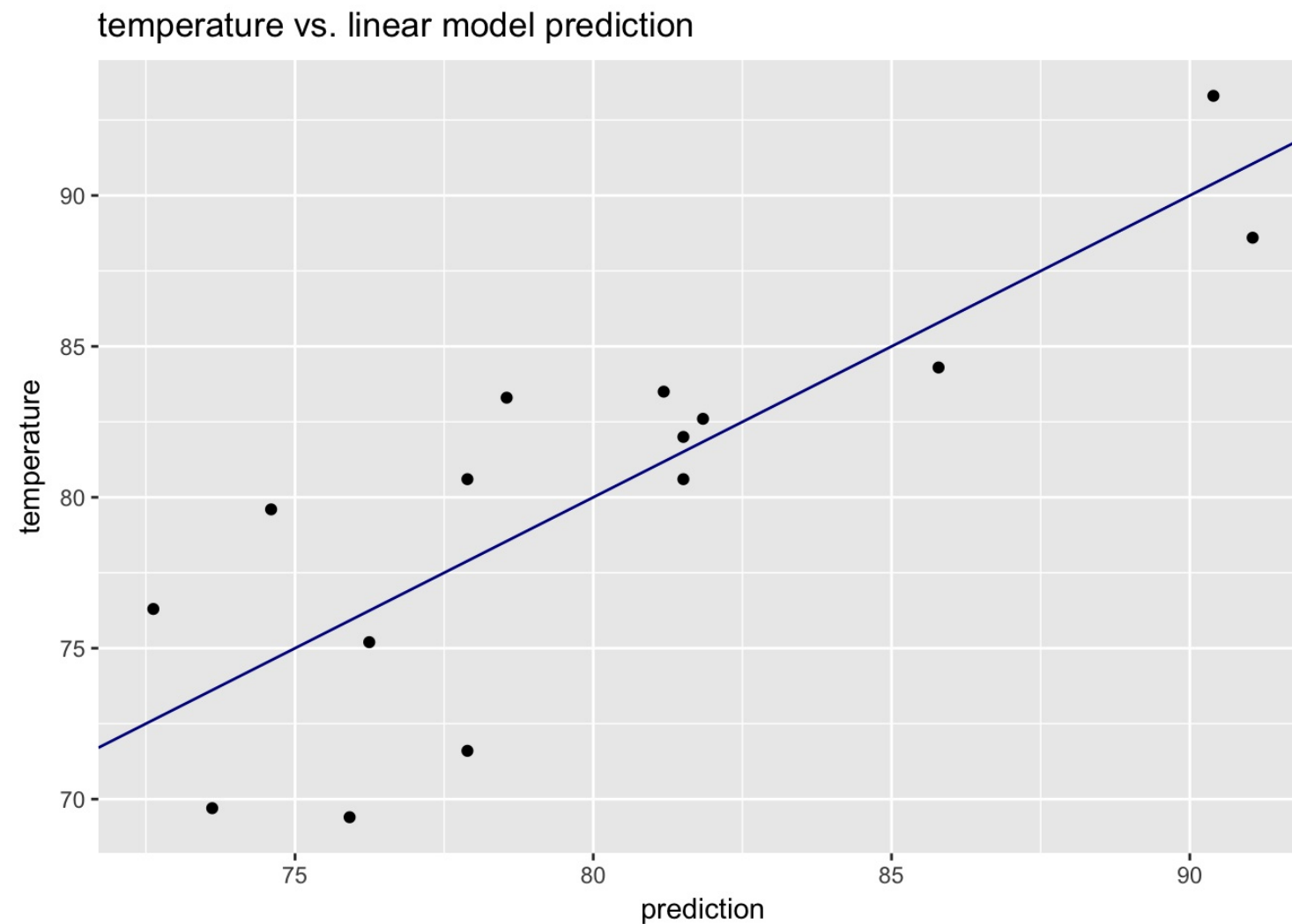
```
cricket$prediction <- predict(cmodel)
```

- `predict()` by default returns training data predictions



Looking at the Predictions

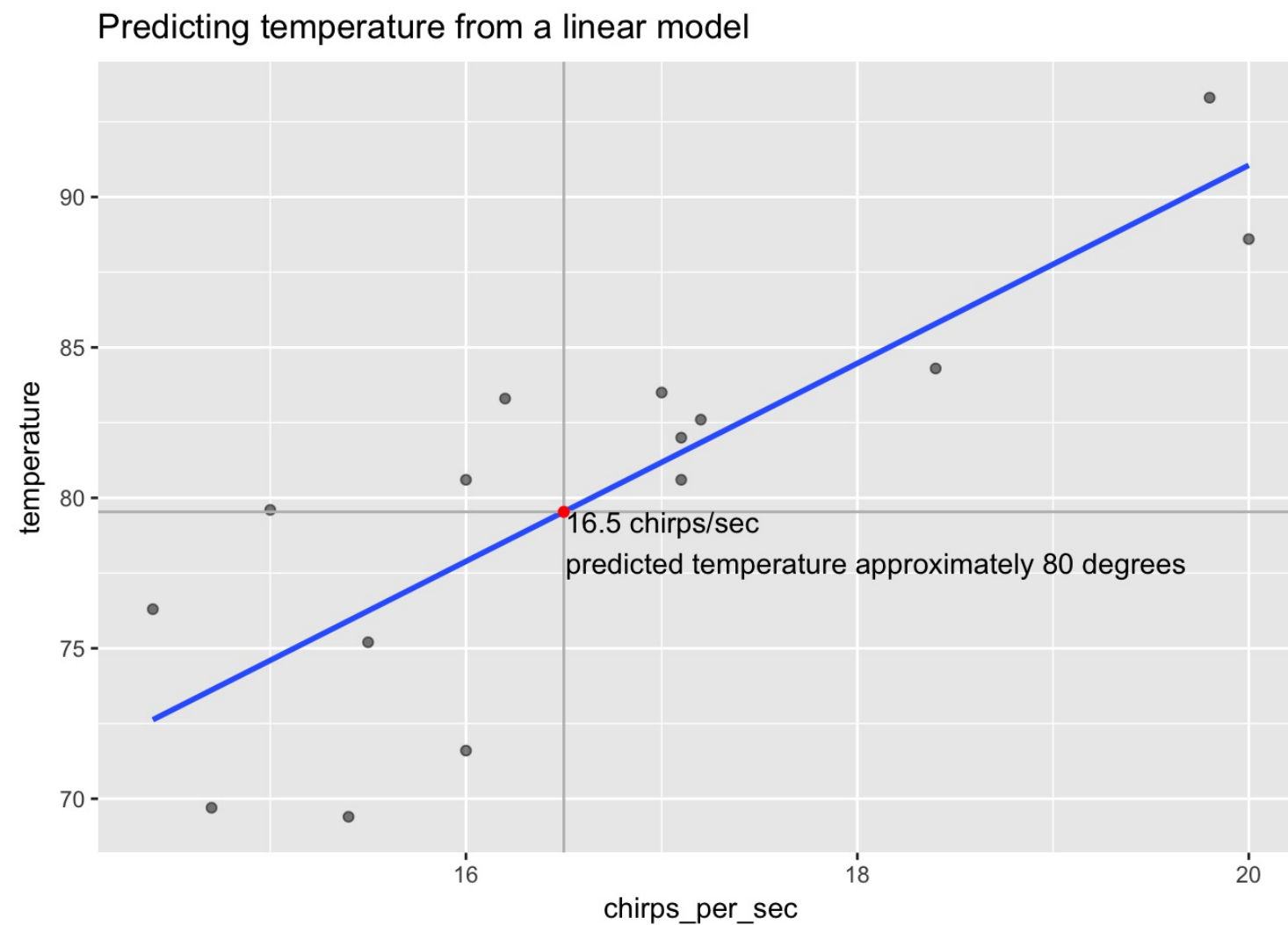
```
ggplot(cricket, aes(x = prediction, y = temperature)) +  
  geom_point() +  
  geom_abline(color = "darkblue") +  
  ggtitle("temperature vs. linear model prediction")
```





Predicting on New Data

```
newchirps <- data.frame(chirps_per_sec = 16.5)
newchirps$prediction <- predict(cmodel, newdata = newchirps)
newchirps
##  chirps_per_sec  pred
## 1      16.5 79.53537
```





SUPERVISED LEARNING IN R: REGRESSION

Let's practice!



SUPERVISED LEARNING IN R: REGRESSION

Wrapping up linear regression

Nina Zumel and John Mount
Win-Vector, LLC



Pros and Cons of Linear Regression

- Pros
 - Easy to fit and to apply
 - Concise
 - Less prone to overfitting



Pros and Cons of Linear Regression

- Pros
 - Easy to fit and to apply
 - Concise
 - Less prone to overfitting
 - **Interpretable**

```
## Call:
## lm(formula = blood_pressure ~ age + weight, data = bloodpressure)
##
## Coefficients:
## (Intercept)      age      weight
##   30.9941    0.8614    0.3349
```



Pros and Cons of Linear Regression

- Pros
 - Easy to fit and to apply
 - Concise
 - Less prone to overfitting
 - Interpretable
- **Cons**
 - Can only express linear and additive relationships



Collinearity

- **Collinearity** -- when input variables are partially correlated.

```
## Call:
## lm(formula = blood_pressure ~ age + weight, data = bloodpressure)
##
## Coefficients:
## (Intercept)      age      weight
##   30.9941    0.8614    0.3349
```



Collinearity

- **Collinearity** -- when variables are partially correlated.
- Coefficients might change sign

```
## Call:
## lm(formula = blood_pressure ~ age + weight, data = bloodpressure)
##
## Coefficients:
## (Intercept)      age      weight
##   30.9941    0.8614    0.3349
```



Collinearity

- **Collinearity** -- when variables are partially correlated.
- Coefficients might change sign
- High collinearity:
 - Coefficients (or standard errors) look too large
 - Model may be unstable

```
## Call:
## lm(formula = blood_pressure ~ age + weight, data = bloodpressure)
##
## Coefficients:
## (Intercept)      age      weight
##   30.9941    0.8614    0.3349
```



Coming Next

- Evaluating a regression model
- Properly training a model