Lecture 08: Simple Regression

Taylor Arnold

Statistical models

Statistical models

- Today we turn our attention to modeling, which complements the graphical and data manipulation techniques that we have so far covered.
- The first model we will look at is linear regression. You have, in fact, already seen this graphically when fitting a smoothing line to data using `geom_smooth("lm")`. -Today we will start to see how to implement this model directly in R and how to analyze how well it fits the data at hand.

Statistical models

Statistical models are often characterized as an abstraction of an underlying truth or generative process, and in some cases this is probably the best way to think of them. I, however, view statistical models as a way of explaining variation in a dataset. In this framework there are three desirable properties of any model:

- captures as much variation as possible
- has minimal amount of complexity
- is generalizable to new data

Linear regression

The regression model

A regression model assumes that the value of one variable, which we will call y_i, is on average equal to a function of another variable x_i. In this set-up the variable x is called the **predictor** variable and y is called the **response**.

There are many other terms for these depending on the field of application. Examples include independent / dependent, regressor / regressand, explanatory / explained, feature / output, risk / outcome.

The regression model

We can write this relationship mathematically as:

$$\text{mean}(y_i) = f(x_i)$$

A common use of statistical modeling is to estimate the value of the function f given a set of data.

The linear regression model

One way to do this is to assume that the function f is a linear function and can therefore be described with simply an intercept α and slope β:

$$\text{mean}(y_i) = \alpha + x_i \cdot \beta$$

This model is called a **linear regression**; it requires only estimating two values. There are many different algorithms for estimating these parameters.

lm_basic()

The `lm_basic` function in R uses the classical and most common of these, which we will define shortly. An example of how to do this is given by:

```
model <- lm_basic(awake ~ sleep_rem, data = msleep)
```

Here we using treating lung cancer as the response and poverty as the predictor variable.

coef()

To see the estimated value of the slope and intercept, we use the `coef` function:

```
coef(model)
```

```
## (Intercept)   sleep_rem
##   18.536491   -2.625664
```

coef()

Using this output, the relationship implied by the model is:

$$\text{mean}(awake) = 18.536 + \text{sleep\_rem} \cdot -2.626$$

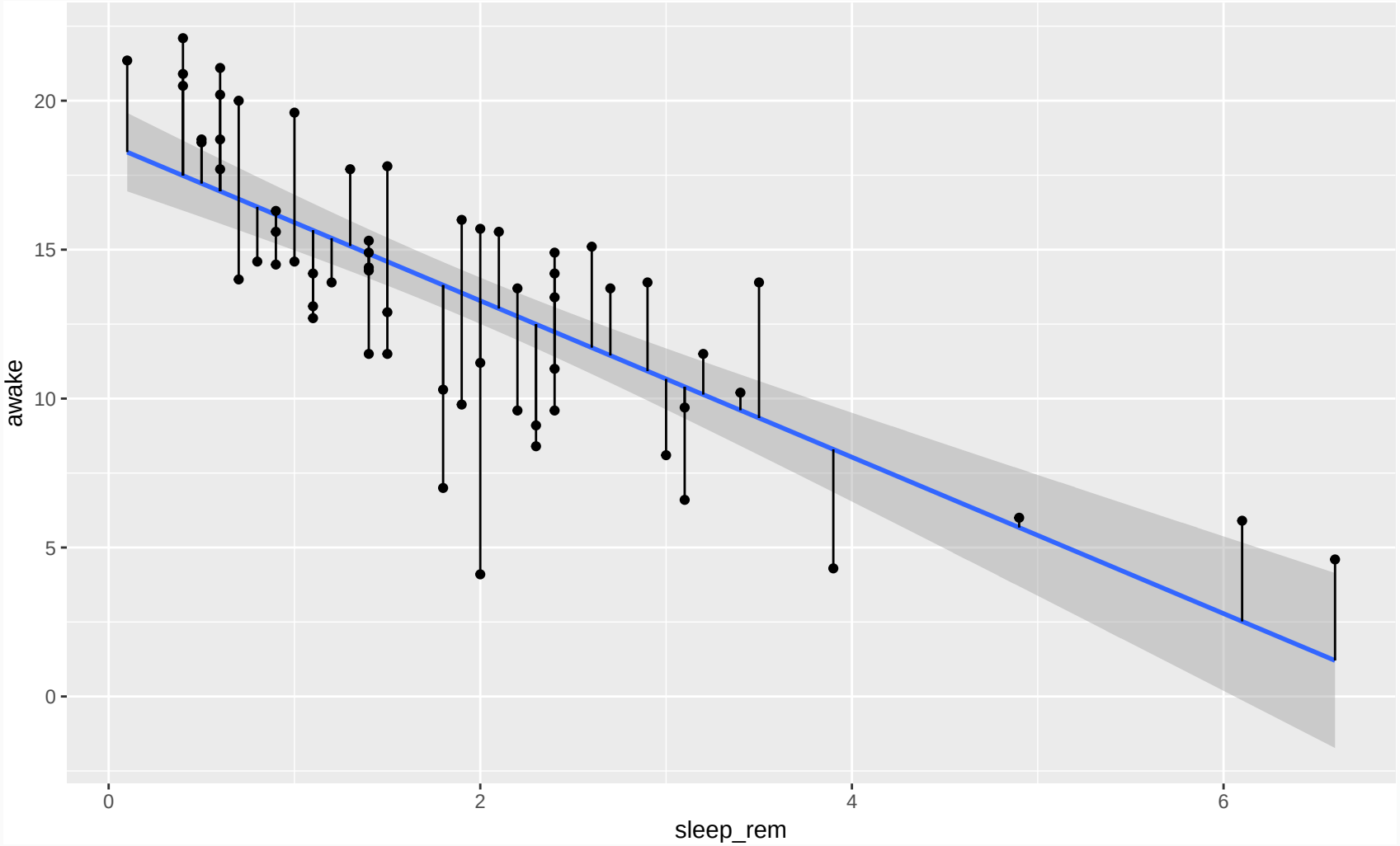
residuals

The **residuals** of a regression model are defined as the difference between the predicted value from the model and the observed value. It is very similar to the deviance that we saw earlier:

$$
\begin{aligned}
r_1 &= y_1 - f(x_1) \\
r_2 &= y_2 - f(x_2) \\
&\vdots \\
r_n &= y_n - f(x_n)
\end{aligned}
$$

residuals

We can also think of these residuals graphically:



ols

The `lm` function is choosing an intercept and slope in order to minimize the sum of the squared residuals. The technique is called **ordinary least squares**; it has a number of very nice computational and theoretical properties.

goodness of fit

sums of squares

We can calculate the sum of squared deviances (often called the **total sum of squares**, or TSS) as:

```
tss <- sum((msleep$awake - mean(msleep$awake))^2)
```

And the sum of squared residuals (RSS).

```
msleep <- mutate(msleep,
       awake_residual = awake - 18.536 + sleep_rem * 2.626)
rss <- sum(msleep$awake_residual^2)
```

r-squared

The ratio of these indicates how much of the variation is captured in the regression line.

```
rss / tss
```

```
## [1] 0.43484
```

Usually, we take the value `1 - rss/tss` so that numbers close to 1 indicate a great fit and values near 0 indicate a relatively poor fit. This quantity is formally called the coefficient of determination, or more commonly **r-squared**.

regression table

reg_table

The most useful way for us to summarize a regression object is to use the function `reg_table`:

```
reg_table(model)
```

The output includes a description of the residuals, the model coefficients, as well as the R-squared value.

reg_table

```
reg_table(model)
```

```
##
## Call:
## lm_basic(formula = awake ~ sleep_rem, data = msleep)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1852 -2.5483  0.3293  2.5774  4.6138
##
## Coefficients:
##             Estimate
## (Intercept)   18.536
## sleep_rem     -2.626
##
## Residual standard error: 3.015 on 59 degrees of freedom
## Multiple R-squared:  0.5652, Adjusted R-squared:  0.5578
## F-statistic: 76.68 on 1 and 59 DF,  p-value: 2.911e-12
```

reg_table

There are various options to the function that will be very handy as we move forward.