Predictions for "Simple" Linear Regression

(Sleuth 3 Sections 7.4.2 and 7.4.3)

Goals for today

- Get an estimate for the mean response at a particular value x of the explanatory variable by plugging in x in the equation
- Get a *confidence interval* for the mean response at a particular value x using t-based methods
- Make Bonferroni or Scheffe adjustments to get simultaneous confidence intervals for the mean response at multiple values of x.

Example

We have a data set with information about 152 flights by Endeavour Airlines that departed from JFK airport in New York to either Nashville (BNA), Cincinnati (CVG), or Minneapolis-Saint Paul (MSP) in January 2012.

```
head(flights)
```

```
## # A tibble: 6 x 3
##   distance air_time dest
##      <dbl>    <dbl> <chr>
## 1     1029      189 MSP
## 2      765      150 BNA
## 3     1029      173 MSP
## 4      589      118 CVG
## 5      589      115 CVG
## 6     1029      153 MSP
```

```
nrow(flights)
```

```
## [1] 152
```

R Code to get model fit

```
model_fit <- lm(air_time ~ distance, data = flights)
summary(model_fit)
```

```
##
## Call:
## lm(formula = air_time ~ distance, data = flights)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.022  -7.054  -1.086   6.170  24.170
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.567729   3.955477   3.683 0.000321 ***
## distance     0.146999   0.004372  33.624  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.881 on 150 degrees of freedom
## Multiple R-squared:  0.8829, Adjusted R-squared:  0.8821
## F-statistic:  1131 on 1 and 150 DF,  p-value: < 2.2e-16
```

Our Estimates for β_0 and β_1 are

$\hat{\beta}_0 =$

$\hat{\beta}_1 =$

Note: LaTeX code for $\hat{\beta}_1$ is `\hat{\beta}_1`.

Our estimated mean function is

$\hat{\mu}(Y|X) = \hat{\beta}_0 + \hat{\beta}_1 X =$

The fitted (predicted) mean air time at a flight distance of 589 miles is

```
14.568 + 0.147 * 589
```

```
## [1] 101.151
```

...or...

```
predict_df <- data.frame(
  distance = 589
)
predict_df
```

```
##   distance
## 1      589
```

```
predict(model_fit, newdata = predict_df)
```

```
##        1
## 101.1504
```

Standard Error of Estimated Mean

Here's a formula you will never use for the predicted mean at the value X_0:

$$SE\{\hat{\mu}(Y|X_0)\} = \hat{\sigma}\sqrt{\frac{1}{n} + \frac{(X_0 - \bar{X})^2}{(n-1)s_X^2}}$$

- $\bar{X}$ is the sample mean of the explanatory variable
- s_X^2 is the sample variance of the explanatory variable
- $\hat{\sigma} = \sqrt{\frac{\text{Sum of Squared Residuals}}{n-2}}$

Something to notice (don't need to memorize):

- This standard error depends on the value X_0 at which we are estimating the mean response
- $(X_0 - \bar{X})^2$ is smallest if $X_0 = \bar{X}$, so $SE\{\hat{\mu}(Y|X_0)\}$ is smallest at the sample mean.

Find and interpret a 95% confidence interval for the mean air time for flights traveling a distance of 589 miles.

```
predict(model_fit, newdata = predict_df, interval = "confidence", se.fit = TRUE)
```

```
## $fit
##        fit      lwr      upr
## 1 101.1504 98.13544 104.1653
##
## $se.fit
## [1] 1.525854
##
## $df
## [1] 150
##
## $residual.scale
## [1] 9.880835
```

This is just a t-based interval based on the estimate and its standard error (although the calculation of standard error is complicated...)

```
qt(0.975, df = 152 - 2)
```

```
## [1] 1.975905
```

```
101.150 - 1.976 * 1.526
```

```
## [1] 98.13462
```

```
101.150 + 1.976 * 1.526
```

```
## [1] 104.1654
```

Find and interpret Bonferroni adjusted confidence intervals for the mean air time at flight distances of 589 miles, 765 miles, and 1029 miles, with a familywise confidence level of 95%.

Approach 1 (easier): adjust confidence level we ask `predict` for.

- 3 CI's at a familywise confidence level of 95%
- Overall, miss for 5% of samples, $\alpha = 0.05$
- Each individual CI has $\alpha = 0.05/3 = 0.0167$
- Each individual CI has confidence level $(1 - 0.0167) \times 100\% = 98.3\%$

```
predict_df <- data.frame(
  distance = c(589, 765, 1029)
)
predict_df
```

```
##   distance
## 1      589
## 2      765
## 3     1029
```

```
predict(model_fit,
  newdata = predict_df,
  interval = "confidence",
  se.fit = TRUE,
  level = 0.983
)
```

```
## $fit
##        fit       lwr      upr
## 1 101.1504  97.46754 104.8332
## 2 127.0223 124.70452 129.3400
## 3 165.8301 163.37682 168.2834
##
## $se.fit
##         1         2         3
## 1.5258544 0.9602809 1.0164383
##
## $df
## [1] 150
##
## $residual.scale
## [1] 9.880835
```

Approach 2: Manual calculation based on standard errors

```
(1 - 0.05/(2*3))
```

```
## [1] 0.9916667
```

```
qt(0.9917, df = 152 - 2)
```

```
## [1] 2.422641
```

```
# CI for X0 = 589
101.150 - 2.423 * 1.526
```

```
## [1] 97.4525
```

```
101.150 + 2.423 * 1.526
```

```
## [1] 104.8475
```

```
# CI for X0 = 765
127.0223 - 2.423 * 0.960
```

```
## [1] 124.6962
```

```
127.0223 + 2.423 * 0.960
```

```
## [1] 129.3484
```

```
# CI for X0 = 1029
165.8301 - 2.423 * 1.016
```

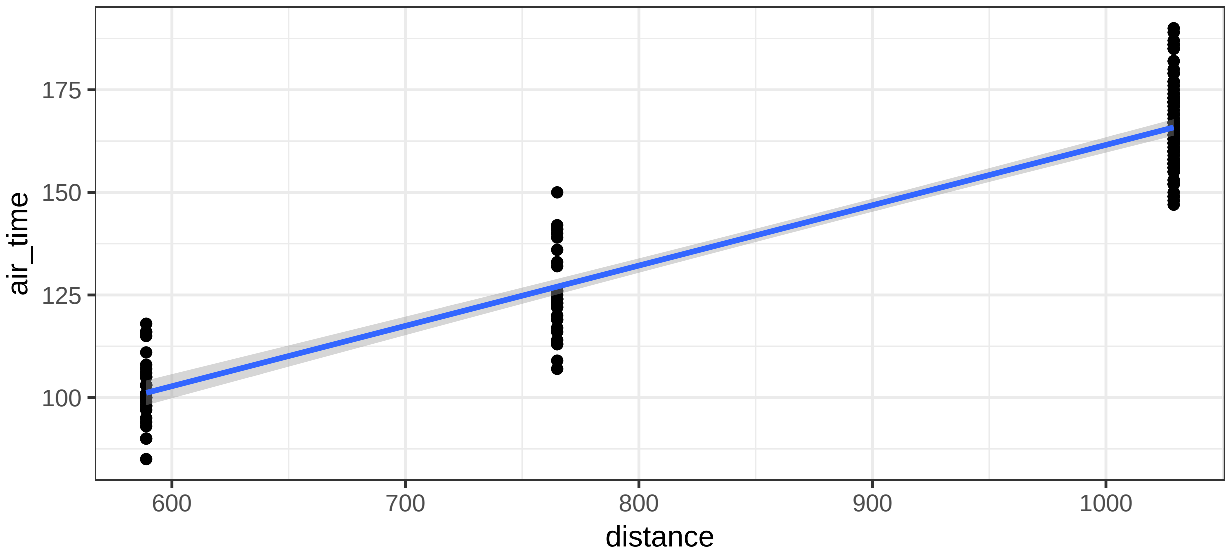
```
## [1] 163.3683
```

```
165.8301 + 2.423 * 1.016
```

```
## [1] 168.2919
```

Find and plot Scheffe adjusted CIs for the means at a grid of 101 values of x between 589 and 1029

```
ggplot(data = flights, mapping = aes(x = distance, y = air_time)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_bw()
```



I will not ask you to do this by hand, the following is just to illustrate

```
Scheffe_multiplier <- sqrt((2 - 1) * qf(0.95, df1 = 2 - 1, df2 = 152 - 2))

predict_df <- data.frame(
  distance = seq(from = 589, to = 1029, length = 101)
)
head(predict_df)
```

```
##   distance
## 1    589.0
## 2    593.4
## 3    597.8
## 4    602.2
## 5    606.6
## 6    611.0
```

```
preds_with_ses <- predict(model_fit,
  newdata = predict_df,
  se.fit = TRUE
)

predict_df <- predict_df %>%
  mutate(
    scheffe_lower = preds_with_ses$fit - Scheffe_multiplier * preds_with_ses$se.fit,
    scheffe_upper = preds_with_ses$fit + Scheffe_multiplier * preds_with_ses$se.fit
  )

ggplot(data = flights, mapping = aes(x = distance, y = air_time)) +
  geom_point() +
  geom_line(data = predict_df, mapping = aes(x = distance, y = scheffe_lower)) +
  geom_line(data = predict_df, mapping = aes(x = distance, y = scheffe_upper)) +
  theme_bw()
```