Lab 8: Introduction to Simple Linear Regression

Objectives for this Lab

- Graph a scatterplot.
- Estimate simple linear regression parameters β_0 and β_1.
- Add a fitted line, a confidence band, and a prediction band to the scatterplot.
- Estimate the mean of Y for a given value of X.
- Produce a residual plot.

1. As usual, start up RStudio and open Lab8.R. Load the `Sleuth3` and `ggplot2` R packages.

```
> library(Sleuth3)
> library(ggplot2)
```

2. View the first few lines of the extra-galactic nebulae data, and create a scatterplot.

```
> head(case0701)
> qplot(Velocity, Distance, data=case0701)
```

Note that this `qplot()` command looks very similar to the one we have used to produce side-by-side boxplots. The difference here is the lack of the `geom="boxplot"` and that `Velocity` is a numeric variable, not a categorical or "factor" variable. You can check this using `is.numeric()`.

```
> is.numeric(case0701$Velocity)
```

3. Perform the regression, that is, fit a line to the points in the scatterplot. As we did when performing a one-way ANOVA with `aov()`, we'll save an "object" called `case0701_lm` that contains the analysis.

```
> case0701_lm <- lm(Distance~Velocity, data=case0701)
```

The function `lm()` is useful for any kind of "linear model" analysis, of which simple linear regression and one-way ANOVA are two examples. If you take ST 412/512, you will spend a lot of time with `lm()`.

4. Get the parameter estimates and their standard errors.

```
> summary(case0701_lm)
```

Compare your R output to the table of parameter estimates in Display 7.9. The row labeled "Constant" in Display 7.9 is labeled `(Intercept)` in the R output. This row refers to the parameter β_0 which is the y-intercept in the linear equation

$$\mu\{Y|X\} = \beta_0 + \beta_1 X. \tag{1}$$

Equation (1) is the model equation on page 181 of the *Sleuth*. The notation $\mu\{Y|X\}$ on the left-hand side of the equation is read "the population mean of Y given X," so this model says that the population mean of Y is a linear function of X with slope β_1 and y-intercept β_0. For the data of Case Study 7.1.1, Y denotes `Distance` and X denotes `Velocity`. (You can tell by which variable is on which axis in Display 7.1.)

Find the point estimates and standard errors for β_0 and β_1 in the R output.

5. In Lab 7, we used the `confint()` to get Dunnett's confidence intervals from a `glht` object. If we give `confint()` our `lm` object, we will get confidence intervals for regression parameters β_0 and β_1.

```
> confint(case0701_lm)
```

6. The output from `lm()` in item 4 contains point estimates and standard errors for intercept β_0 and slope β_1. We stored the output in the `lm` object `case0701_lm`. In particular, R calls the point estimates "coefficients."

```
> case0701_lm$coefficients
```

   We can use these estimated coefficients to draw the estimated regression line on the scatterplot by adding a layer via `geom_abline()` to the `qplot()` command. `geom_abline()` needs two arguments, the intercept and slope.

```
> qplot(Velocity,Distance, data=case0701) +
+   geom_abline(slope=case0701_lm$coefficients["Velocity"],
+               intercept=case0701_lm$coefficients["(Intercept)"])
```

   Your scatterplot should now have a beautiful regression line.

   Note: You could replace `case0701_lm$coefficients["Velocity"]` and `case0701_lm$coefficients["(Intercept)"]` with their values:

```
> qplot(Velocity,Distance, data=case0701) +
+   geom_abline(slope=0.001372408,
+               intercept=0.3991704)
```

7. The `ggplot()` function is capable of drawing the fitted regression line automatically. Here's the command.

```
> ggplot(case0701, aes(x=Velocity, y=Distance)) +
+   geom_point() +
+   geom_smooth(method=lm, se=FALSE)
```

   Notes: First, there are four `"+"` symbols on the above code. The two on the far left of the second and third lines are continuation prompts and are not in the code in Lab8.r. The two `"+"` symbols on the right of lines one and two are part of the command. They are telling `ggplot()` to *add layers*.

   The code `method=lm` tells R to fit a regression line. The code `se=FALSE` tells R *not* to include a confidence band.

8. Omitting `se=FALSE` will produce a plot with a 95% confidence band on the scatterplot. These are the curves labeled "95% confidence band for single mean distance" on the figure in Display 7.11, which is the narrowest confidence band. The slightly larger confidence band is for the entire regression line, and uses a Scheffé multiplier to account for writing essentially an infinite number of simultaneous confidence intervals. We will not calculate this larger confidence band.

9. Add the "95% prediction band for an unknown distance" in Display 7.11. This we'll have to do by hand–`ggplot()` doesn't have a way to add these automatically.

   (a) Create a data frame called `pred_intervals` containing predictions with lower and upper 95% prediction limits using `predict()`.

   ```
   > pred_intervals <- data.frame(predict(case0701_lm, interval="prediction"))
   ```

   R gives a warning message to remind you that prediction intervals are different from confidence intervals. We will discuss prediction vs. estimation in lecture (and see Section 7.4.3 of the textbook for details). If instead of `interval="prediction"`, you write `interval="confidence"`, you'll get confidence intervals.

   We put the `predict()` command within the `data.frame()` command to make `pred_intervals` a data frame. Otherwise it's a "matrix," and we can't refer to its columns by name.

   Speaking of column names, what are they?

   ```
   > head(pred_intervals)
   ```

   The lower and upper 95% prediction limits are called `lwr` and `upr`. The `fit` column contains the point predictions, the points on the estimated line corresponding to the `Velocity` values given in the original data. You can find these numbers in the fit_i column in Display 7.8.

   (b) Join the output of `predict()` to the original data frame. This puts the data and the prediction bounds in a single data frame which we can then give to `ggplot()`.

   ```
   > case0701_df2 <- cbind(case0701, pred_intervals)
   > head(case0701_df2)
   ```

   (c) Create the plot.

   ```
   > ggplot(case0701_df2, aes(x=Velocity, y=Distance)) +
   +   geom_point() +
   +   geom_smooth(method=lm) +
   +   geom_line(aes(y=lwr), color = "blue", linetype = "dashed") +
   +   geom_line(aes(y=upr), color = "blue", linetype = "dashed")
   ```

   This is the same code as in 7 with `se=FALSE` omitted and two extra layers for the two dotted lines.

10. The `predict()` function can calculate confidence intervals and prediction intervals for a given set of values of the explanatory variable.

    (a) Calculate confidence intervals for the population mean distance of extra-galactic nebulae with recession velocities of 600 km/sec and -200 km/sec (interpret negative recession velocity as traveling *toward* us rather than away from us). These are the points on the true regression line corresponding to `Velocity`$= 600$ and `Velocity`$= -200$.

    ```
    > predict(case0701_lm, data.frame(Velocity=c(600,-200)),
    +         interval="confidence")
    ```

    The lower and upper bounds of the confidence intervals are listed in the columns labeled `lwr` and `upr` respectively.

Note the second argument to `predict()`. This argument must be a data frame containing a column with exactly the same name as the predictor variable in the data frame in the `lm()` command used to get the lm object `case0701_lm` (refer to item 3).

(b) The confidence interval for $\mu\{\text{Distance}|\text{Velocity} = -200\}$ contains 0. What do you conclude?

11. When we did the regression back in 3, R calculated fitted values and residuals and stored them in the lm object `case0701_lm`.

(a) Compare the first few of R's residuals to those in Display 7.8.

```
> head(case0701_lm$residuals)
```

The discrepancies are due to rounding error in Display 7.8.

(b) When we discuss assumptions for regression in Chapter 8, we'll see that a plot of the residuals vs. fitted values is a useful diagnostic, just as it was for one-way ANOVA. You can get the plot in the same way that we did before.

```
> plot(case0701_lm, which=1)
```

In regression, the "Fitted values" are obtained by plugging the observed explanatory variables (here `Velocity`) into the estimated regression equation. These are the point predictions of item 9(a).