15.1: SLR CODE SUPPLEMENT

Prof Amanda Luby

```r
library(patchwork)
library(tidyverse)
theme_set(theme_minimal())
```

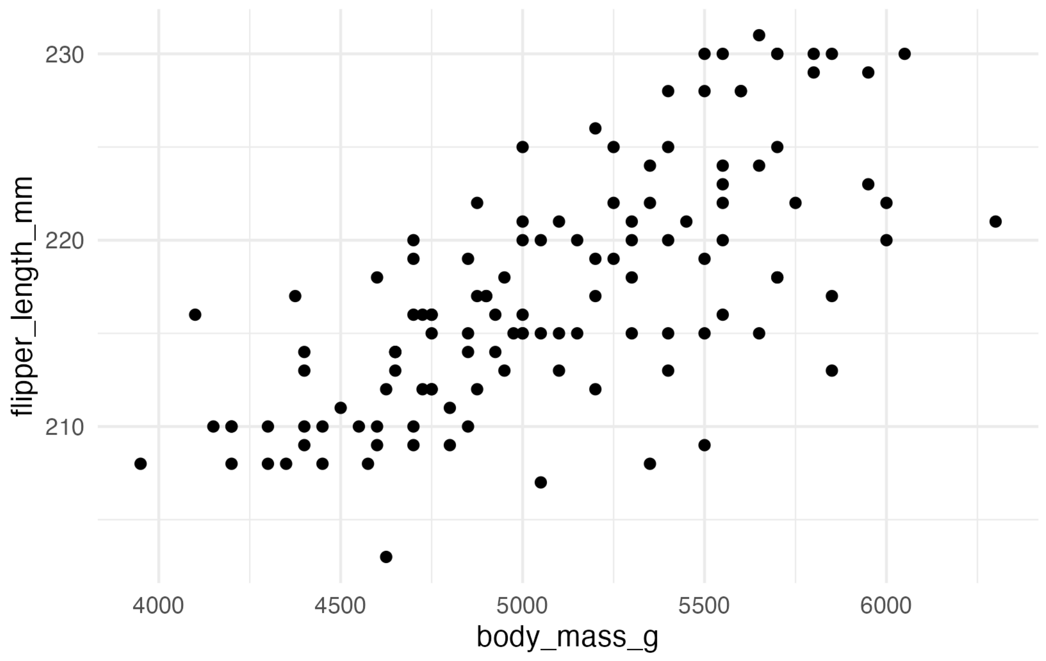
This .qmd document provides some example code for fitting linear regression models and doing corresponding inference in R. As is often the case, there are *many* different ways that you can do this correctly. I've chosen one approach that prioritizes code readability. You are welcome to take different approaches in your homework, but you should fully understand how everything works before submitting.

First, create the `gentoo` dataset, which contains only the `gentoo` penguins from the `palmerpenguins` data:

```r
gentoo = palmerpenguins::penguins %>%
  filter(species == "Gentoo")
```

Next, it's best practice to create a scatterplot of the relationship you're interested in *before* fitting the model:

```r
ggplot(gentoo, aes(x = body_mass_g, y = flipper_length_mm)) +
  geom_point()
```



Looks pretty good! There are no huge red flags about fitting a linear model. Next, we use `lm()` to fit the `linear` model, using the "formula" syntax (which we've seen before in `t.test()`). First, we create `gentoo_lm` and then we print out the summary:

```r
gentoo_lm = lm(flipper_length_mm ~ body_mass_g, data = gentoo)
gentoo_lm
```

```

Call:
lm(formula = flipper_length_mm ~ body_mass_g, data = gentoo)

Coefficients:
(Intercept)  body_mass_g
  1.713e+02    9.039e-03
```

This gives us the $\hat{\beta}_0$ and $\hat{\beta}_1$ estimates. Let's say we want to run a t-test for β_1 - how would we get the relevant quantities? There's actually a *lot* more information stored in the `gentoo_lm` R object (run `names(gentoo_lm)` to see what they are), and we can print a nice summary table with `summary()`:

```r
summary(gentoo_lm)
```

```

Call:
lm(formula = flipper_length_mm ~ body_mass_g, data = gentoo)

Residuals:
     Min       1Q   Median       3Q      Max
-12.0194  -2.7401   0.1781   2.9859   8.9806

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.713e+02  4.244e+00   40.36   <2e-16 ***
body_mass_g 9.039e-03  8.321e-04   10.86   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.633 on 121 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.4937,    Adjusted R-squared:  0.4896
F-statistic:   118 on 1 and 121 DF,  p-value: < 2.2e-16
```

The coefficients table gives us $\hat{\beta}_0$ and $\hat{\beta}_1$, their corresponding standard errors, the resulting t-test statistic assuming $\beta_0 = 0$ and $\beta_1 = 0$, and the p-value for the hypothesis test. The Residual standard error also gives us an estimate for s_ϵ, the unbiased estimator for σ^2.

Before moving on, let's check our residual plots. I like to do this via `broom::augment()` (you may need to download the `broom` package). The idea is we augment the existing `lm()` model to find the fitted values and residuals for each data point. This is also the command we will use to get confidence and prediction intervals for $\hat{y}$. This creates a new dataset called `gentoo_aug` that contains the x and y values, the predictions (`.fitted`) and residuals (.resid). It also gives a variety of other quantities that we may or may not talk about in this class.

```
gentoo_aug = broom::augment(gentoo_lm)
gentoo_aug
```

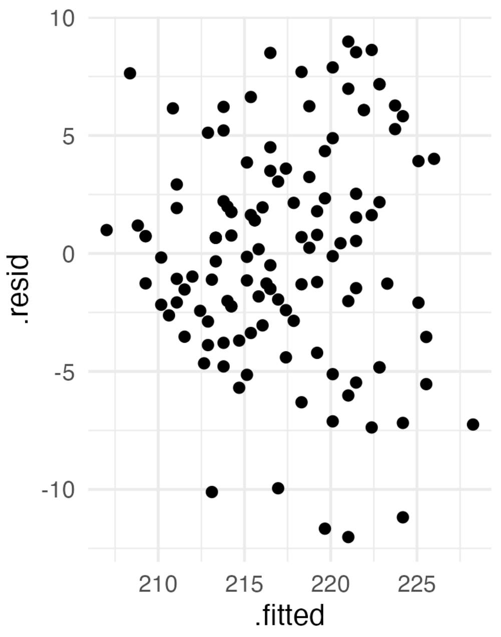
```
# A tibble: 123 x 9
  .rownames flipper_length_mm body_mass_g .fitted .resid   .hat .sigma .cooksd
  <chr>                 <int>       <int>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>
 1 1                      211        4500    212. -0.980 0.0188   4.65 4.38e-4
 2 2                      230        5700    223.  7.17  0.0207   4.61 2.58e-2
 3 3                      210        4450    212. -1.53  0.0208   4.65 1.18e-3
 4 4                      218        5700    223. -4.83  0.0207   4.63 1.17e-2
 5 5                      215        5400    220. -5.12  0.0115   4.63 7.18e-3
 6 6                      210        4550    212. -2.43  0.0171   4.65 2.43e-3
 7 7                      211        4800    215. -3.69  0.0106   4.64 3.43e-3
 8 8                      219        5200    218.  0.692 0.00863  4.65 9.80e-5
 9 9                      209        4400    211. -2.08  0.0229   4.65 2.41e-3
10 10                     215        5150    218. -2.86  0.00831  4.65 1.60e-3
# i 113 more rows
# i 1 more variable: .std.resid <dbl>
```

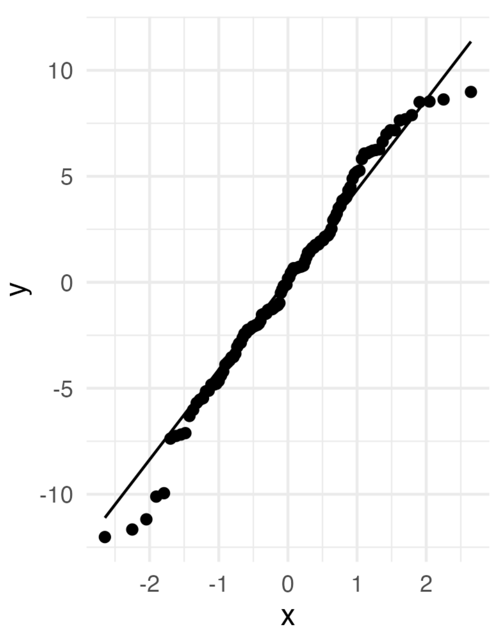
We can then use our new augmented dataset to make residual plots. Which assumptions do each of these residual plots tell us about?

```
p1 = ggplot(gentoo_aug, aes(x = .fitted, y = .resid)) +
  geom_point()

p2 = ggplot(gentoo_aug, aes(sample = .resid)) +
  geom_qq() +
  geom_qq_line()

p1 + p2
```

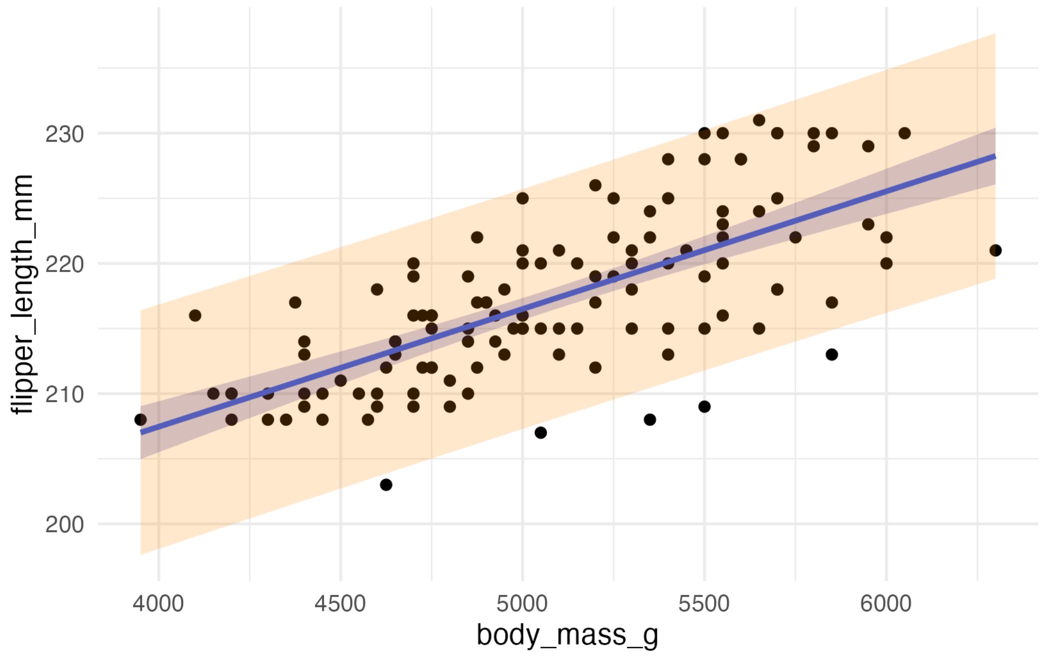




We can also use `augment` to create confidence and prediction intervals for $\hat{y}$. The code below (1) creates two augmented datasets, one with confidence intervals for each prediction and one for prediction intervals for each prediction, and (2) makes a scatterplot with each set of intervals overlaid.

```
gentoo_conf = broom::augment(gentoo_lm, interval = "confidence")
gentoo_prediction = broom::augment(gentoo_lm, interval = "prediction")

palmerpenguins::penguins %>%
  filter(species == "Gentoo") %>%
  ggplot(aes(x = body_mass_g, y = flipper_length_mm)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  geom_ribbon(data = gentoo_conf, aes(ymin = .lower, ymax = .upper),
              alpha = .2, fill = "darkblue") +
  geom_ribbon(data = gentoo_prediction, aes(ymin = .lower, ymax = .upper),
              alpha = .2, fill = "darkorange")
```



If we want to find the intervals for a specific point, we could either look through the augmented dataset, or tell R we want to find the intervals for new data. The code below tells R to create a confidence interval for a new penguin with body mass 7000g. (Note that it doesn't have a .resid column – why?)

```
broom::augment(gentoo_lm,
               interval = "confidence",
               newdata = tibble(body_mass_g = c(7000)))
```

```
# A tibble: 1 x 4
  body_mass_g .fitted .lower .upper
        <dbl>   <dbl>  <dbl>  <dbl>
1        7000    235.   231.   238.
```