Lecture 3- A Bracing Tour of Simple Linear Regression with R

Peter Shaffery

1/21/2020

PDFs, CDFs, and R

Last lecture I used functions like:

`rnorm, dnorm, pnorm, qnorm`

R provides these functions as part of its "base" set of functions, no package needed.

These functions represent four core utilities for dealing with normally distributed random variables:

- `rnorm(n, m, s)` - draw n **r**andom samples from $N(\mathrm{m}, \mathrm{s}^2)$
- `dnorm(x, m, s)` - compute the probability **d**ensity function $P[x|\mathrm{m}, \mathrm{s}] = N(x; \mathrm{m}, \mathrm{s}^2)$
- `pnorm(x, m, s)` - compute the **p**robability $P[X \leq x|\mathrm{m}, \mathrm{s}] = \int_{-\infty}^{x} N(x; \mathrm{m}, \mathrm{s}^2)dx$
- `qnorm(p, m, s)` - compute the **q**uantile q, where q is the value such that $P[X \leq q|\mathrm{m}, \mathrm{s}] = p$

Such functions are defined for a number of other common distributions:

`rbinom, rt, rchisq, rgamma, rpoisson`

Example: Penguins! (Con't again)

```
library(palmerpenguins)
```

Say that we are researchers studying penguins in the Palmer Archipelago of Antarctica. We have collected data on three different penguin species (Adelie, Chinstrap, and Gentoo) on three different islands (Torgersen, Biscoe, and Dream). For each subject in our dataset, we have recorded the following:

- length and depth of the bill (mm)
- length of the flippers (mm)
- body mass (g)
- sex
- year of the measurement

The data looks like this:

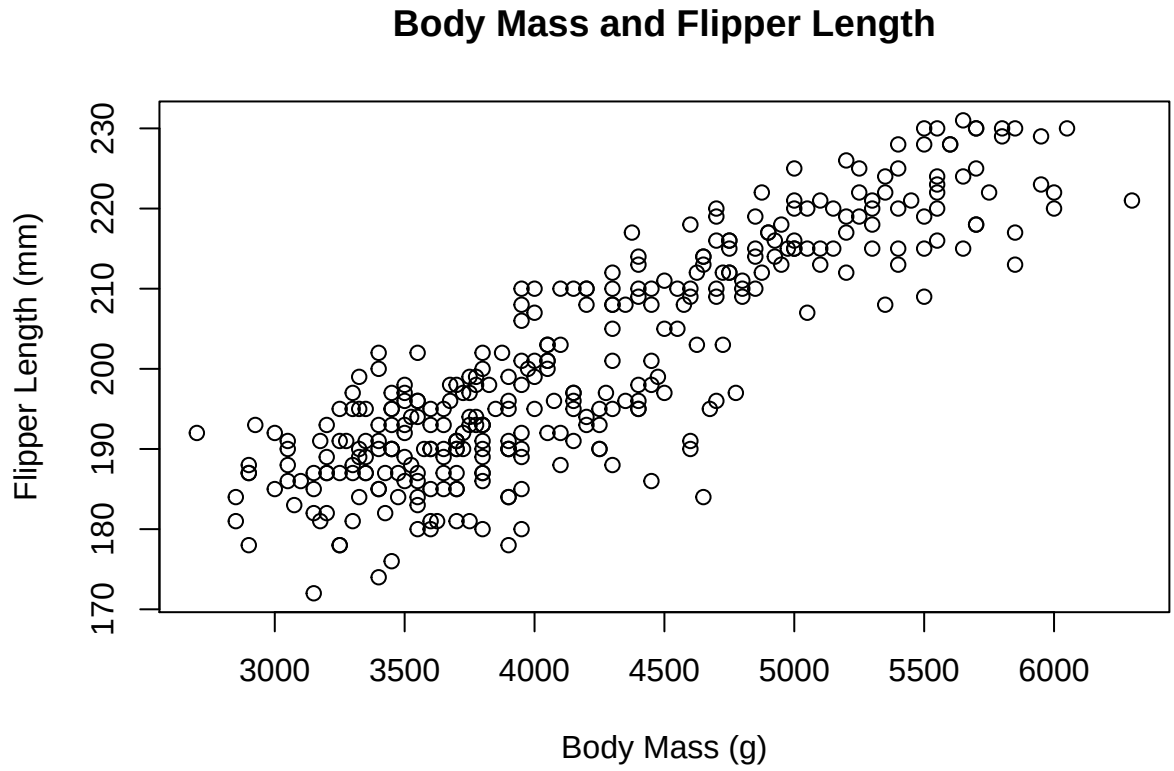
```
penguins
```

```
## # A tibble: 344 x 8
##    species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g
##    <fct>   <fct>           <dbl>         <dbl>            <int>       <int>
##  1 Adelie  Torge~           39.1          18.7              181        3750
##  2 Adelie  Torge~           39.5          17.4              186        3800
##  3 Adelie  Torge~           40.3          18                195        3250
##  4 Adelie  Torge~           NA            NA                 NA          NA
##  5 Adelie  Torge~           36.7          19.3              193        3450
```

```
##  6 Adelie  Torge~           39.3          20.6             190        3650
##  7 Adelie  Torge~           38.9          17.8             181        3625
##  8 Adelie  Torge~           39.2          19.6             195        4675
##  9 Adelie  Torge~           34.1          18.1             193        3475
## 10 Adelie  Torge~           42            20.2             190        4250
## # ... with 334 more rows, and 2 more variables: sex <fct>, year <int>
```

We are interested in the relationship between body mass (x) and flipper length (y)

```
dat = tidyr::drop_na(penguins)
plot(dat$body_mass_g,
     dat$flipper_length_mm,
     main='Body Mass and Flipper Length',
     xlab='Body Mass (g)',
     ylab='Flipper Length (mm)')
```




```
linear.model = lm(flipper_length_mm~body_mass_g,data=dat)
```

Evaluating the Fit of Linear Regression

Last lecture talked about *t-statistics* and *p-values*:

- **t-statistic:** $\hat{t} = \frac{\hat{\beta}_1}{\hat{\tau}}$, a "normalized" version of $\hat{\beta}_1$, scaled by its standard error $\hat{\tau}$
- **p-value:** $p = P[|t| \geq |\hat{t}||H_0]$, the probability (under the null hypothesis H_0) of getting data "more extreme" than the data you observed

Here's a question:

> If a linear regression produces least-squares estimators (LSEs) $\hat{\beta}_0$ and $\hat{\beta}_1$ with low p-values, does that mean that the model $y = \hat{\beta}_0 + \hat{\beta}_1 x$ is a "good fit" for the data?

Think about it for a second!

thinking.

thinking..

thinking...

Let's look at an example:

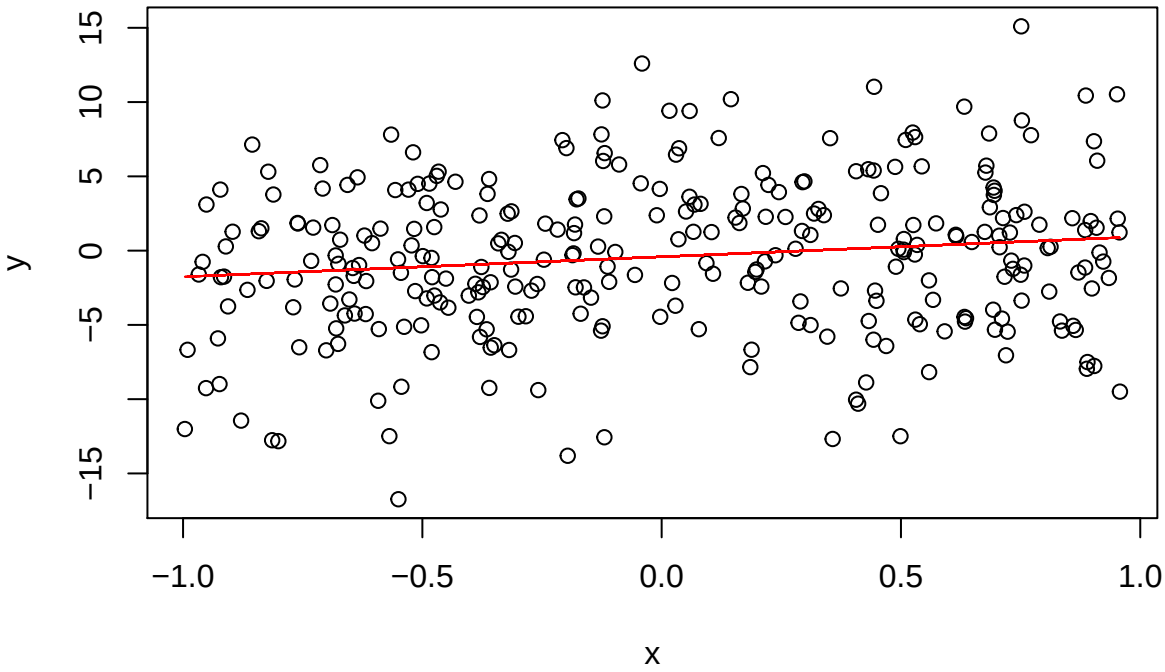
```
set.seed(8675309)
n.example = 300
x.example = runif(n.example,-1,1)
y.example = rnorm(n.example,0 + 1*x.example,5) # beta_0 = 0, beta_1 = 1, sigma^2 = 5

mod = lm(y.example ~ x.example)
summary(mod)
```

```
##
## Call:
## lm(formula = y.example ~ x.example)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.5816  -3.3614   0.1269   3.3081  14.4979
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4132     0.2958  -1.397   0.1635
## x.example     1.3497     0.5214   2.589   0.0101 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.123 on 298 degrees of freedom
## Multiple R-squared:  0.02199,    Adjusted R-squared:  0.01871
## F-statistic: 6.702 on 1 and 298 DF,  p-value: 0.01011
```

```
plot(x.example, y.example,xlab='x',ylab='y',main='Example')
lines(x.example,predict(mod),col='red') # plot best fit line
```

Example



We've got a significant relationship, but the trend-line really does a poor job of fitting the data! What's going on here?

P-values tell you one thing: how well the null hypothesis fits the data. A poorly fitting null model may *suggest* that our LSE model is a good fit, but it doesn't guarantee it.

R-squared

To measure model fit, we need to go back to the *residuals*:

$$\hat{\epsilon_i} = y_i - \hat{y_i}$$

Recall, in our original linear regression model, that residuals represent the part of the model that we can't explain,

So, rearranging:

$$y_i = \hat{y_i} + \hat{\epsilon_i}$$
$$y_i - \bar{y} = (\hat{y_i} - \bar{y}) + \hat{\epsilon_i}$$
$$\text{Deviation from mean} = \text{Deviation due to x} + \text{Deviation due to error}$$

Some fancy algebra (which we will come back to), can be used to show:

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y_i} - \bar{y})^2 + \sum_{i=1}^{n}\hat{\epsilon_i}^2$$
$$\text{SS}_{\text{total}} = \text{SS}_{\text{regression}} + \text{SSE}$$
$$\text{SST} = \text{SSR} + \text{SSE}$$

We can rewrite this one more time:

$$\frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$\frac{\text{SS}_{\text{regression}}}{\text{SS}_{\text{total}}} = 1 - \frac{\text{SSE}}{\text{SS}_{\text{total}}}$$

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$$

R^2 is sometimes referred to as the **coefficient of determination**, however it is easier to remember it as *percent variance explained*

It can be shown that:

$$R^2 = \text{Cor}(x, y)^2$$

R^2 is an example of *effect size*, a way to measure **how much change in y is due to a change in x**.

Checking Assumptions

Recall that simple linear regression made some assumptions:

1. $E[y_i|\beta_0, \beta_1, x_i] = \beta_0 + \beta_1 x_i$
2. The error terms are **iid** normal, $\epsilon_i \sim N(0, \sigma^2)$

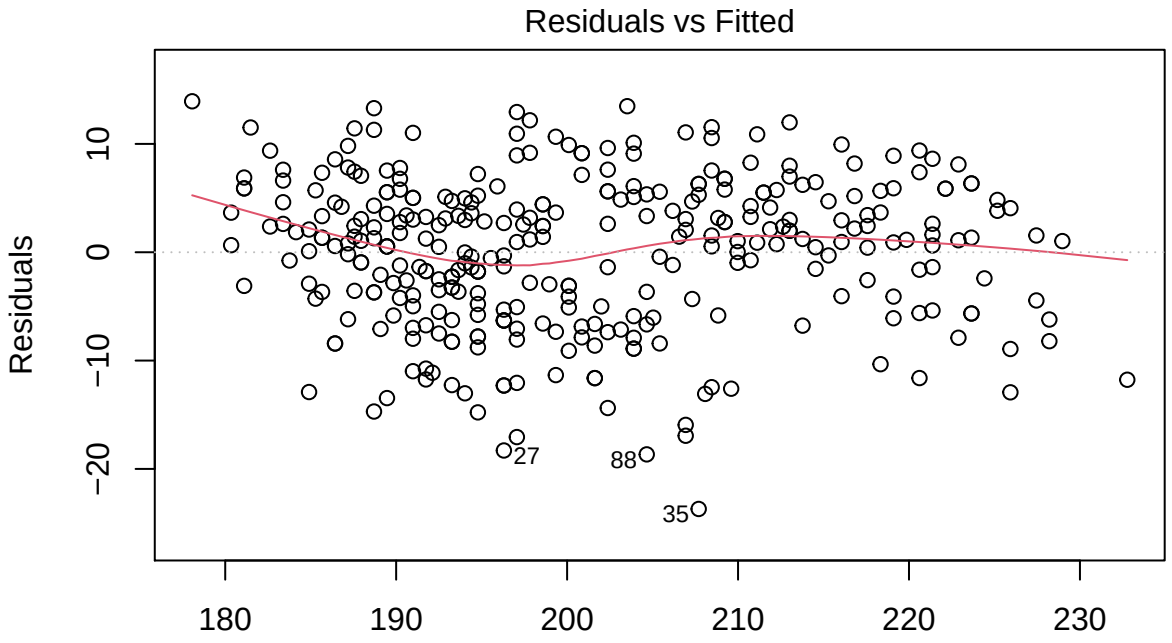
Violations of these assumptions have varying levels of severity:

- **The ϵ_i are not independent**: everything goes out the window, very hard to test for. Might be resolvable with multiple regression.
- **The variance σ^2 changes with x:** moderately affects confidence intervals and hypothesis testing, but easy to check for and resolve
- **The ϵ_i are not normally distributed:** only slightly affects confidence intervals and hypothesis testing, also easy to check for and sometimes easy to resolve

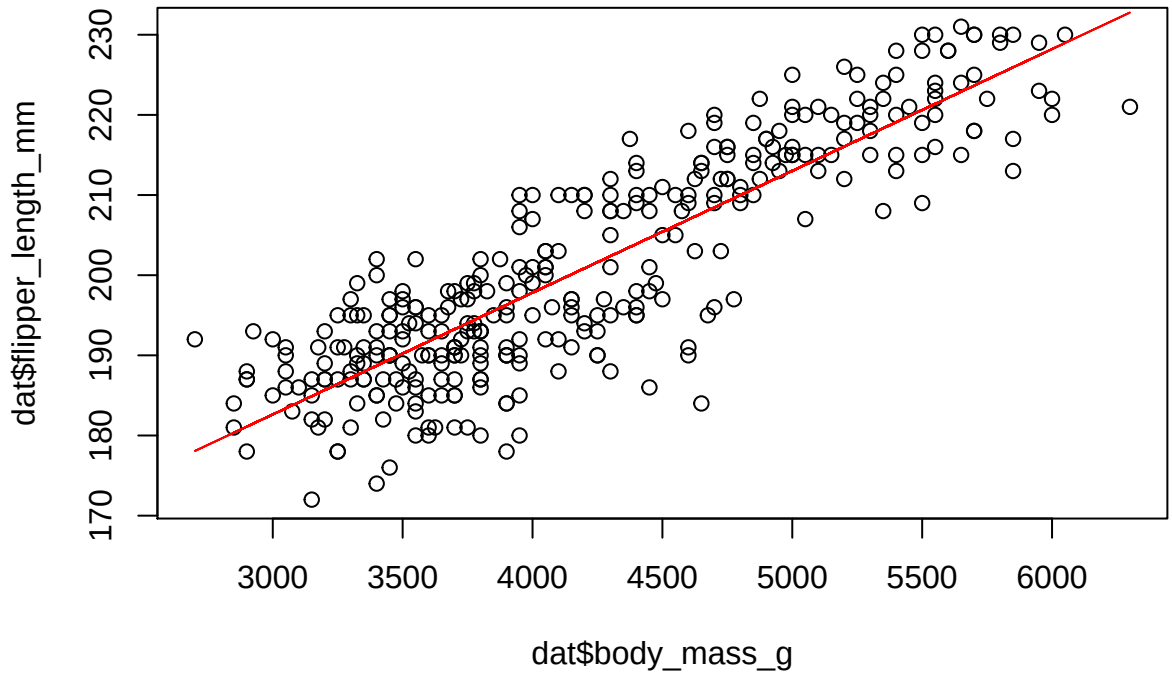
Often it is easiest to check for these assumptions using some visual tools, rather than with a hypothesis test. For linear models produced by `lm()`, R has some automated diagnostic plots:

- **Residuals vs. Fit plots:** a scatter plot of the values $(\hat{y}_i, \hat{\epsilon}_i)$, basically a rotated version of the original scatterplot so that the line of best fit is flat:

```
plot(linear.model,which=1)
```



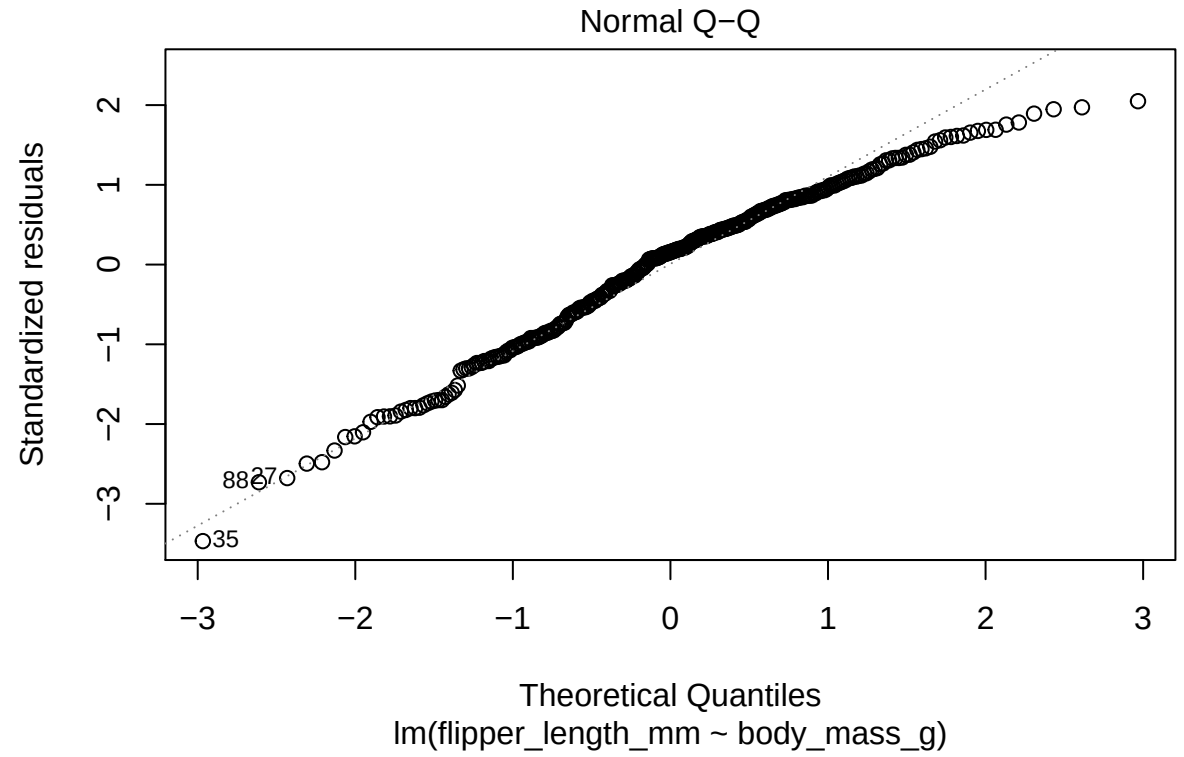

```
plot(dat$body_mass_g,dat$flipper_length_mm)
lines(dat$body_mass_g,predict(linear.model),col='red')
```



This plot helps us get an overall impression of the regression quality: are there even amounts of residuals above and below 0? Is there any overall trend in the residuals? Can we see any tapering or changes of variance?

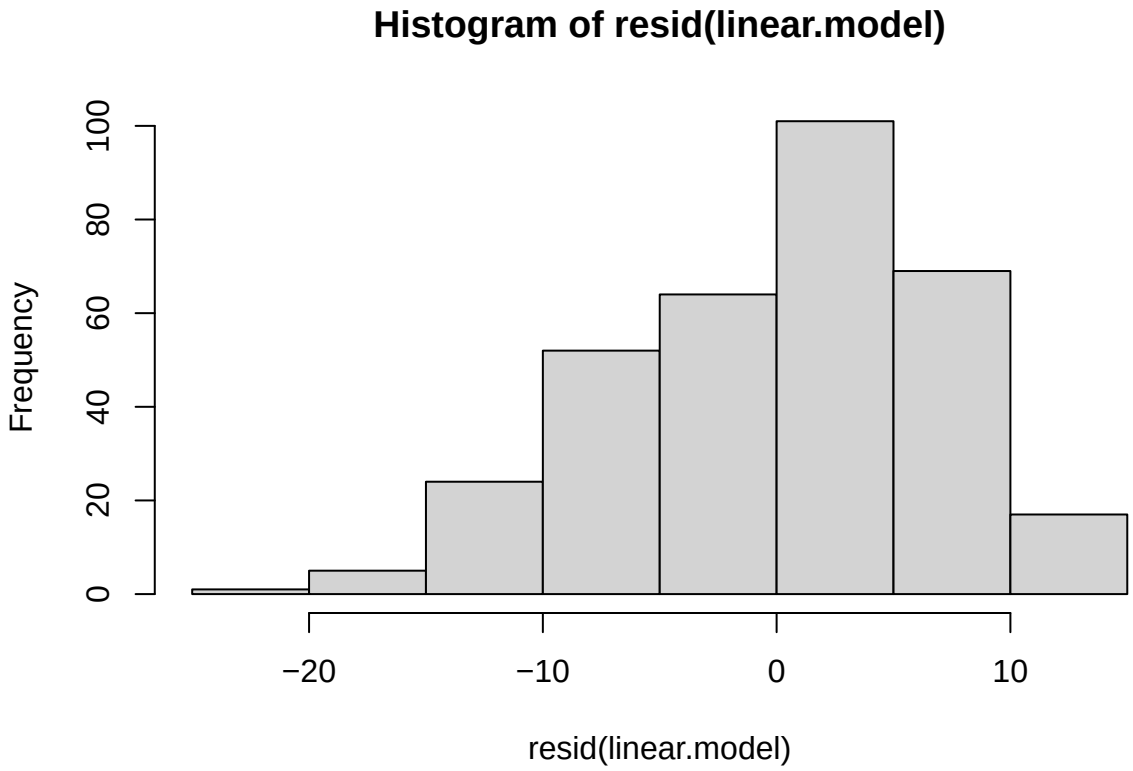
- **Q-Q plots:** plots the *quantiles* of the standardized residuals ($\frac{\epsilon_i}{\hat{\sigma}}$), against the values they *should have* if the ϵ_i were normal:

```
plot(linear.model,which=2)
```



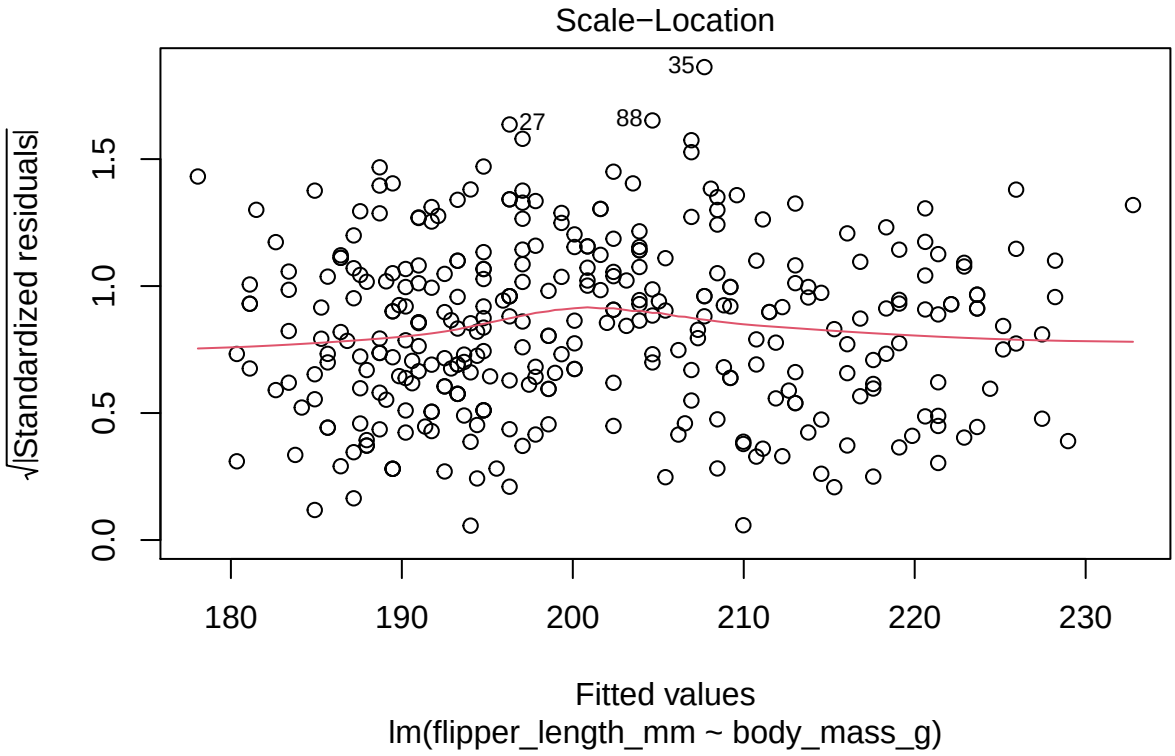
Another way to assess the same information would be with a histogram of the residuals:

```
hist(resid(linear.model))
```



- **Scale-Location plots:** similar to the **Residuals vs. Fit** plot, except instead of plotting $(\hat{y}_i, \hat{\epsilon}_i)$, we plot $(\hat{y}_i, \sqrt{|\frac{\hat{\epsilon}_i}{\hat{\sigma}}|})$. The idea here is to isolate the visual effect of heteroskedasticity. Since any negative residuals will be made positive, if variance is higher or lower at one end of the x-axis, then a trend will appear in the data

```
plot(linear.model,which=3)
```



Putting it All Together- Fish!

Let's perform a regression analysis from start to finish, using a new dataset. Say that we are researchers investigating mercury in wild fish populations. For two separate rivers (Lumber and Wacamaw), we set up a number of measuring stations. At each station we catch, weight, measure, and release fish back into the river. Additionally, from each fish we take a small tissue sample to determine blood mercury content. We want to know how a fish's weight relates to its blood mercury content.

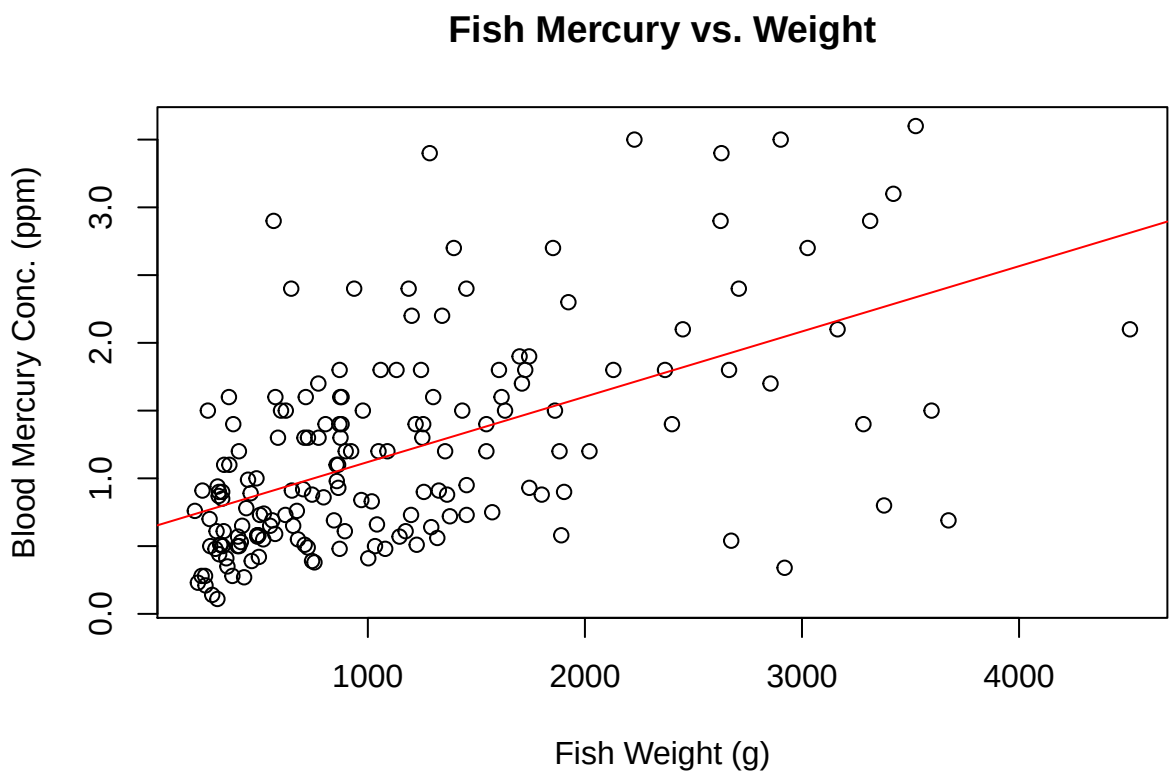
```
fish = read.csv('../../data/fish_hg.tsv', sep=' ')
fish = tidyr::drop_na(fish)

head(fish)
```

```
##    river station length_cm weight_g HG_conc_ppm
## 1 lumber      l1      47.0     1616        1.60
## 2 lumber      l1      48.7     1862        1.50
## 3 lumber      l1      55.7     2855        1.70
## 4 lumber      l1      45.2     1199        0.73
## 5 lumber      l1      44.7     1320        0.56
## 6 lumber      l1      43.8     1225        0.51
```

```
mod = lm(HG_conc_ppm ~ weight_g, data=fish)
```

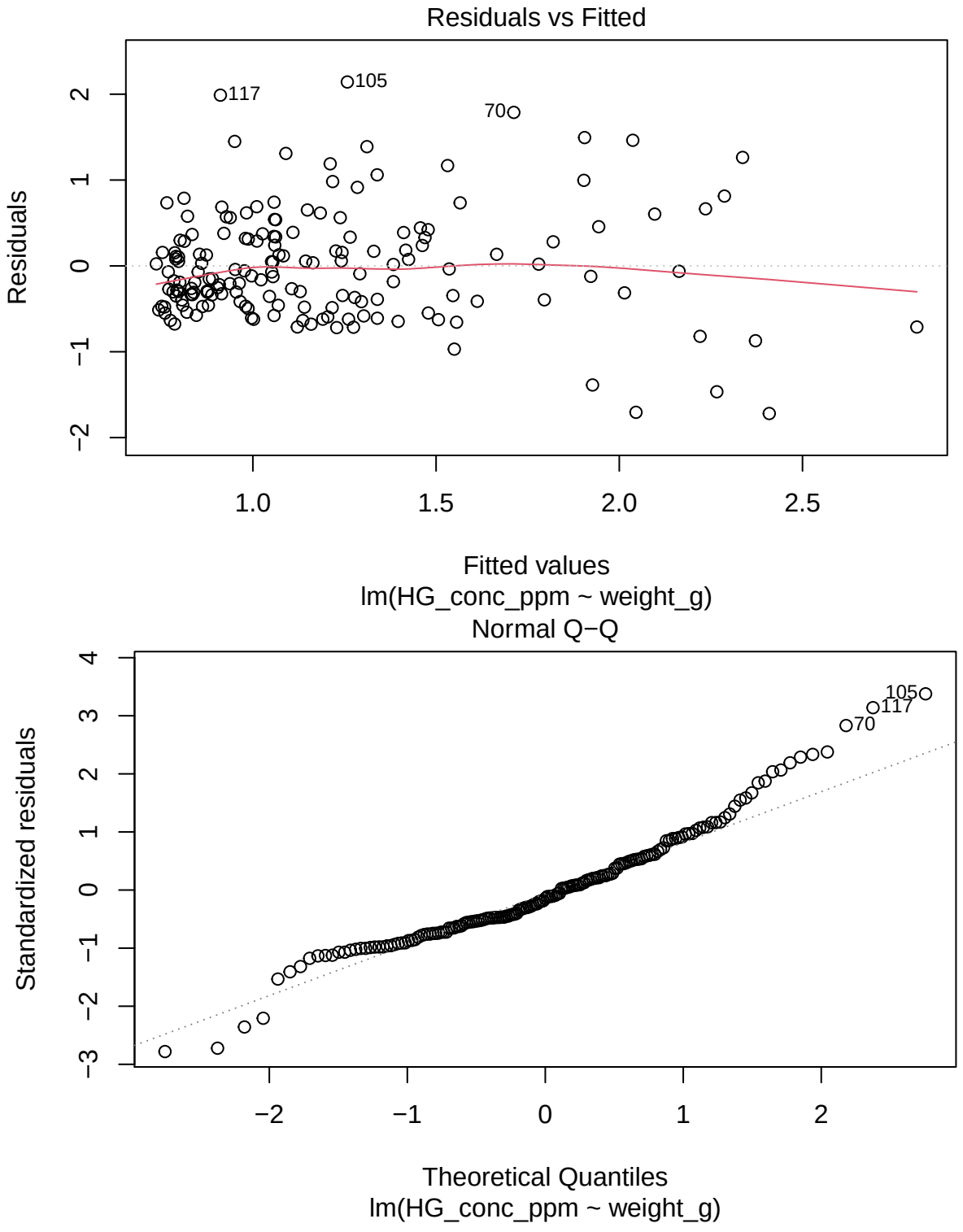
```r
plot(fish$weight_g,
     fish$HG_conc_ppm,
     xlab='Fish Weight (g)',
     ylab='Blood Mercury Conc. (ppm)',
     main='Fish Mercury vs. Weight')
abline(mod,col='red')
```

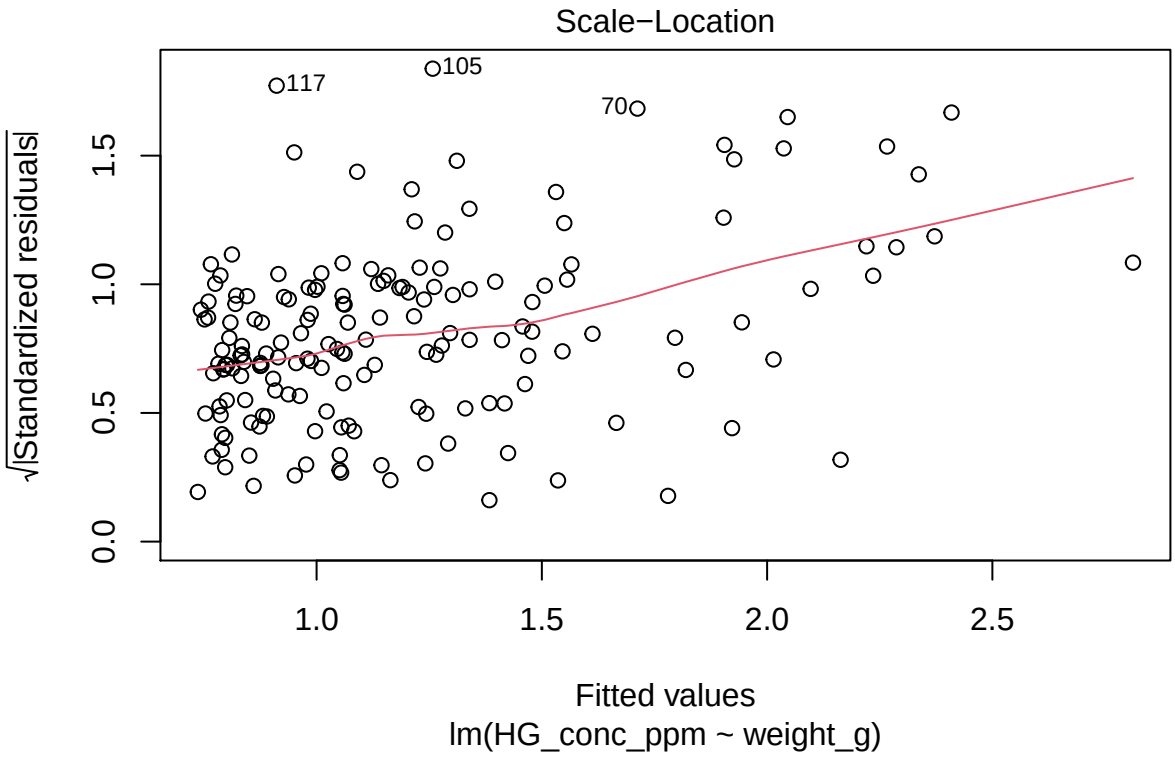



```r
summary(mod)
```

```
##
## Call:
## lm(formula = HG_conc_ppm ~ weight_g, data = fish)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.71933 -0.41388 -0.09201  0.33615  2.14220
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.387e-01  8.035e-02   7.948 2.56e-13 ***
## weight_g    4.818e-04  5.572e-05   8.647 3.93e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6361 on 169 degrees of freedom
## Multiple R-squared:  0.3067, Adjusted R-squared:  0.3026
## F-statistic: 74.77 on 1 and 169 DF,  p-value: 3.929e-15
```

```
plot(mod, which=c(1,2,3))
```





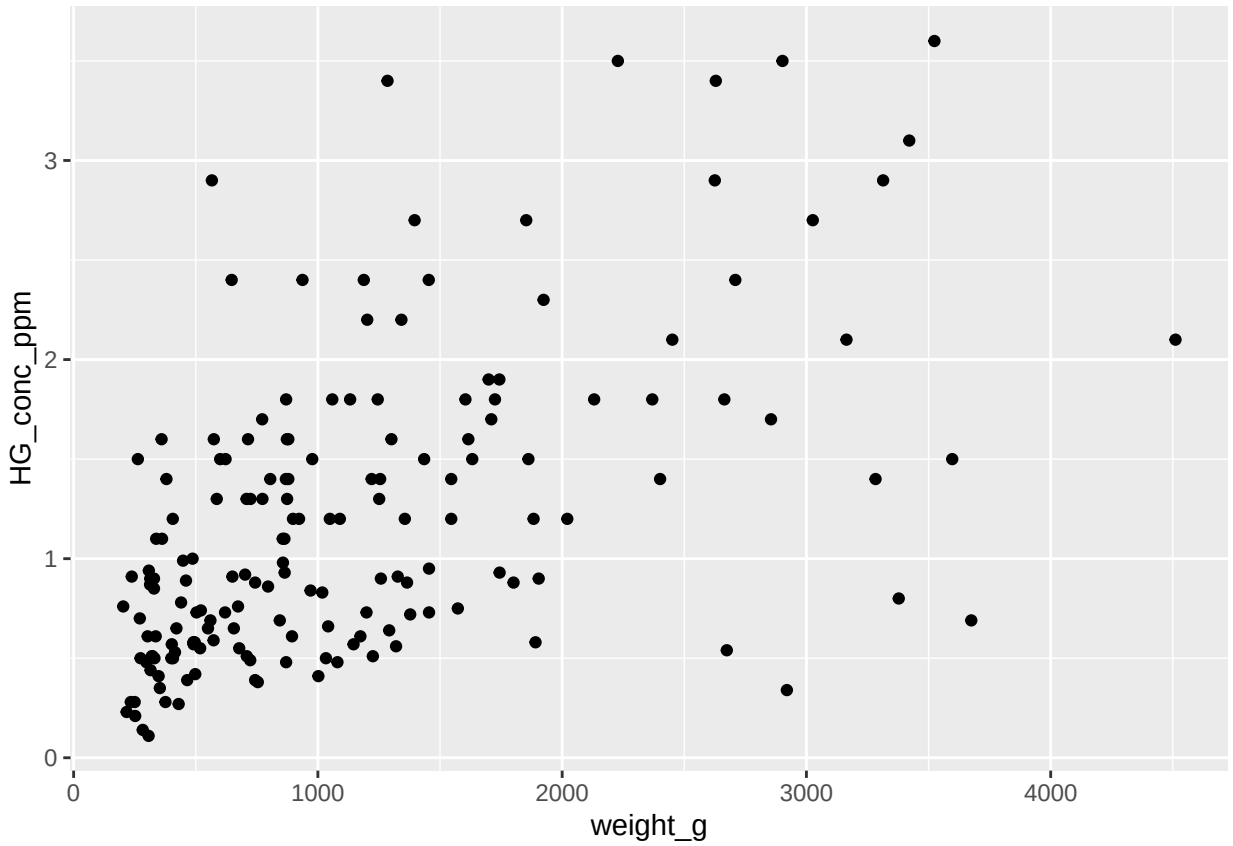
Intro ggplot2

```
library(ggplot2) # gg stands for "Grammar of Graphics" (Leland Wilkinson, 1991)
```

Base R's `plot` function is not great, particularly once you want to deal with color or something in a smart way.

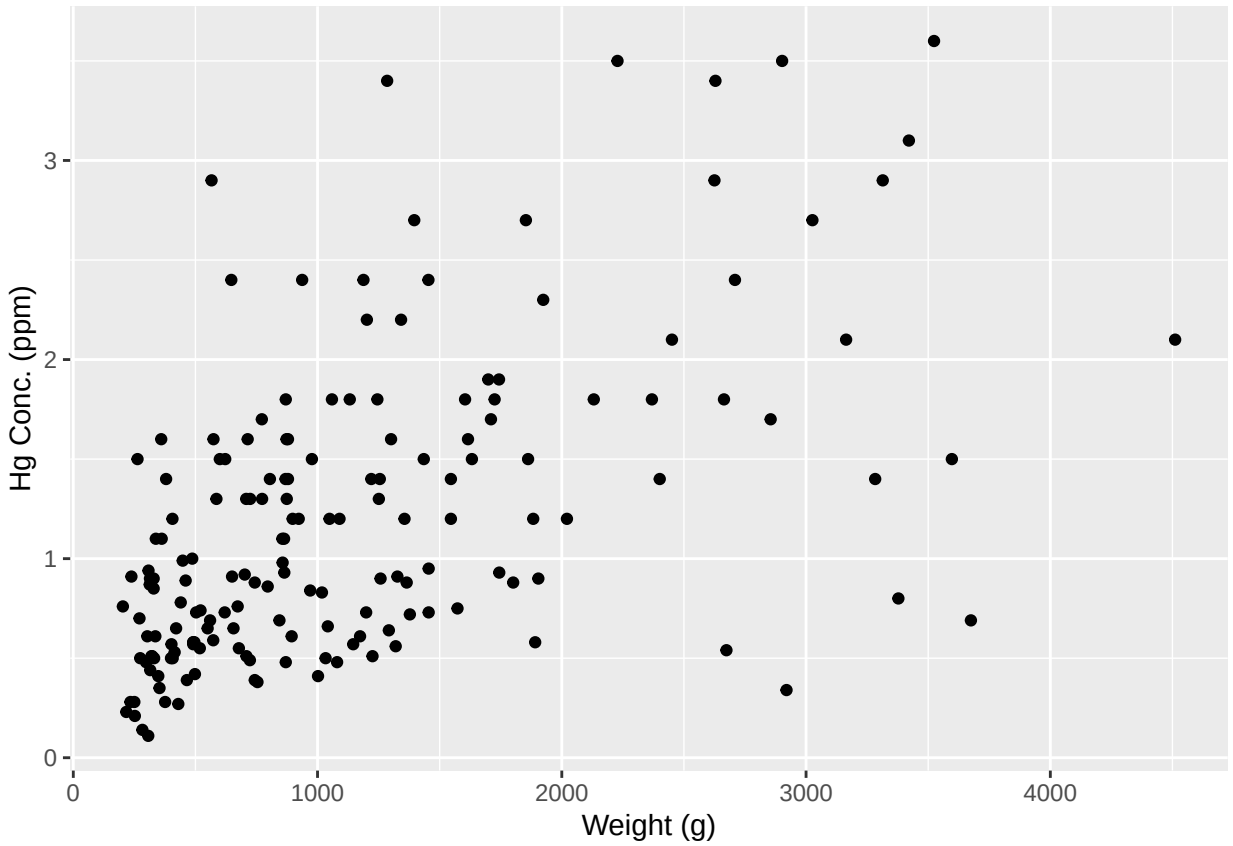
Enter 'ggplot2":

```
plt = ggplot(fish, aes(x=weight_g, y=HG_conc_ppm))
plt + geom_point()
```



In `ggplot2` plots are built out of *layers*, where each layer can inherit data from the ones that came before:

```
plt.scatter = plt + geom_point()
plt.scatter + labs(x='Weight (g)', y='Hg Conc. (ppm)')
```

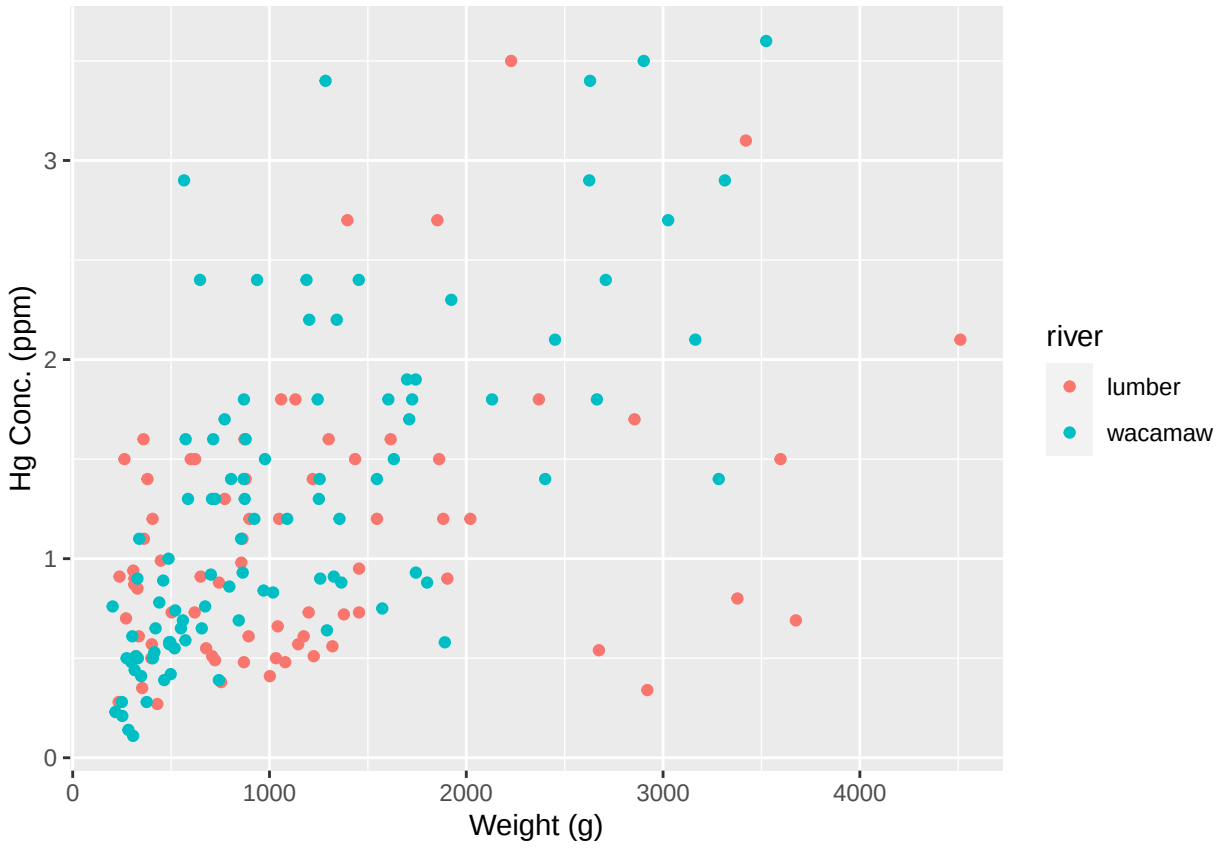


In data visualization, we often talk about **visual channels**, the aspects of a plot which can be used to communicate information to the reader. For a scatterplot, some visual channels might be:

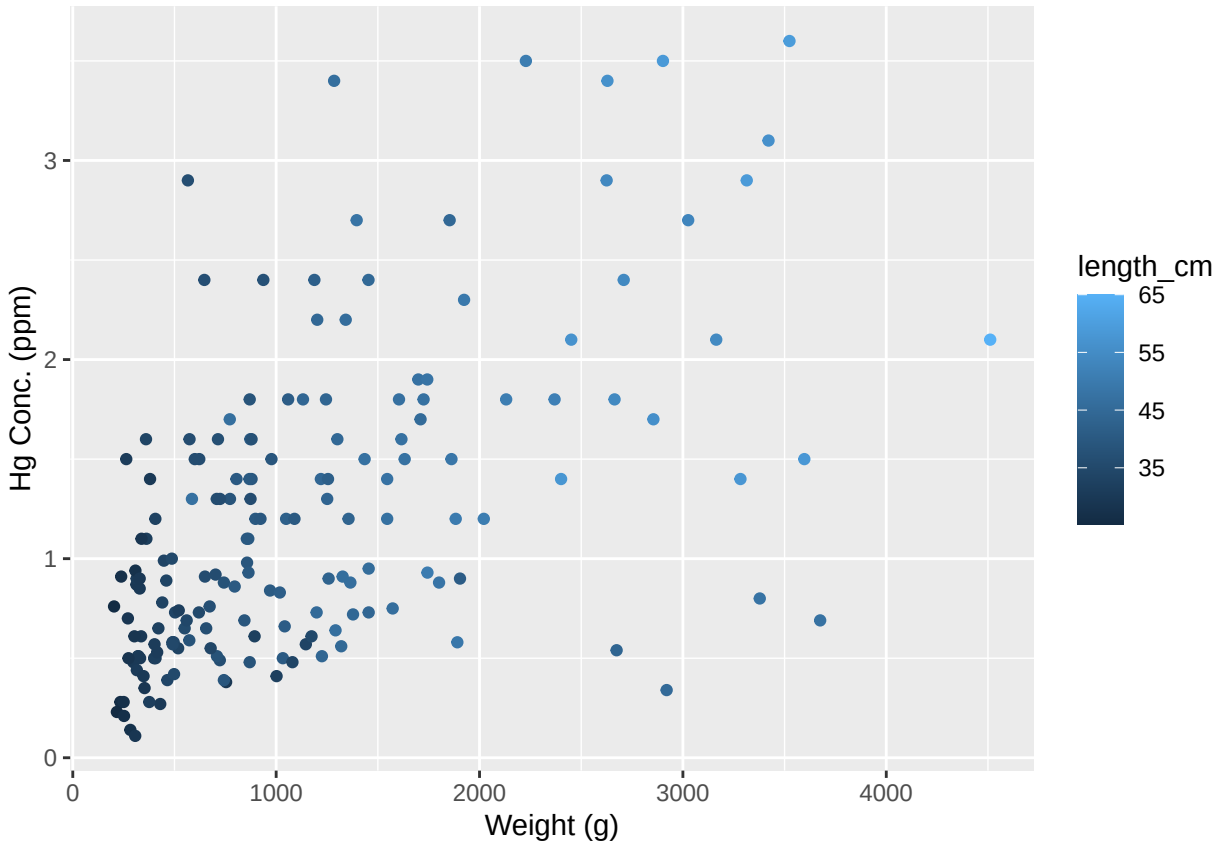
- Point color
- Point shape
- Point size

`ggplot2` makes it very easy to relate visual channels to information in the dataset:

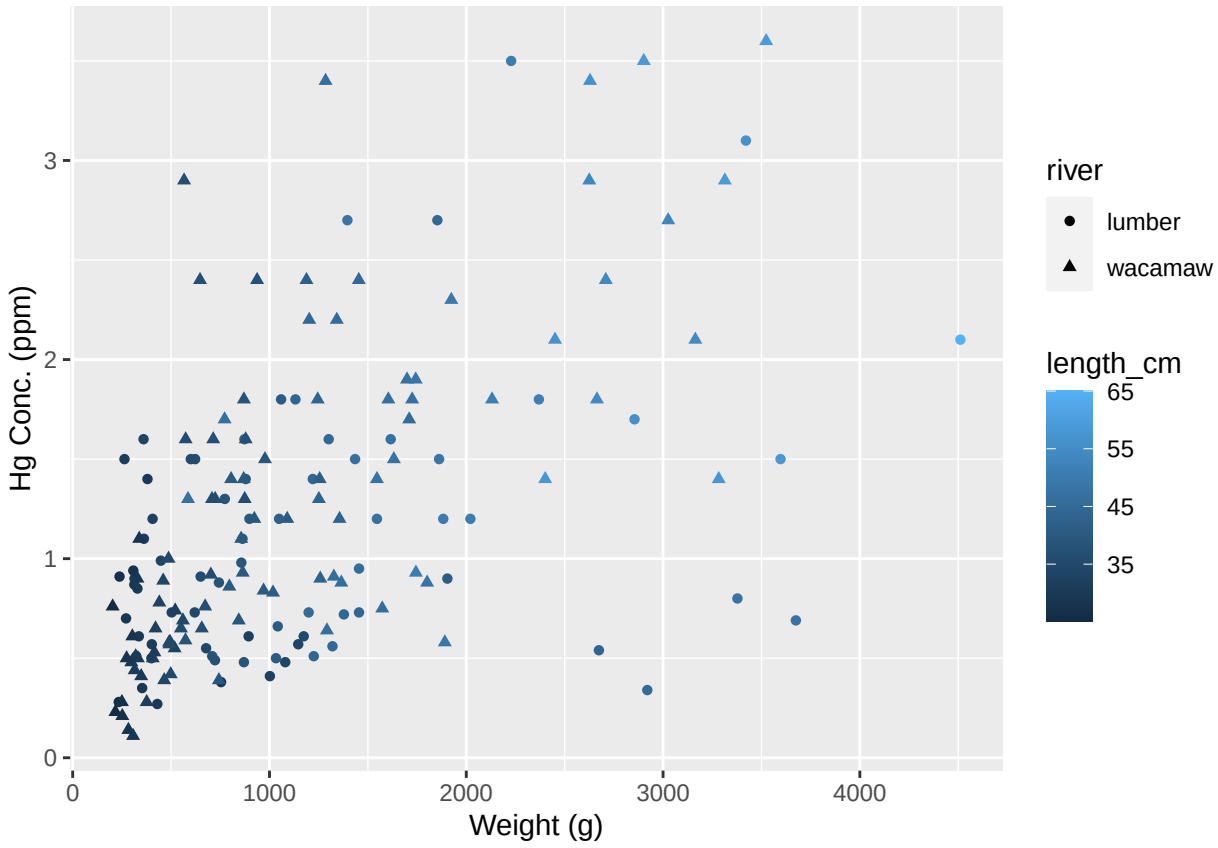
```
plt = ggplot(fish, aes(x=weight_g, y=HG_conc_ppm, color=river))
plt + geom_point() + labs(x='Weight (g)', y='Hg Conc. (ppm)')
```




```
plt = ggplot(fish, aes(x=weight_g, y=HG_conc_ppm, color=length_cm))
plt + geom_point() + labs(x='Weight (g)', y='Hg Conc. (ppm)')
```




```
plt = ggplot(fish, aes(x=weight_g, y=HG_conc_ppm, shape=river, color=length_cm))
plt + geom_point() + labs(x='Weight (g)', y='Hg Conc. (ppm)')
```


Hg Conc. (ppm)
0
1
2
3
0
1000
2000
3000
4000
Weight (g)
river
lumber
wacamaw
length_cm
65
55
45
35