Adding p-values with ggpubr

Augustus Pendleton

Today's Goal: Automatically add p-values to our plots using ggpubr

Often, we want to make statistical comparisons between groups in our data. When you're starting out in R, this can feel like a huge headache. You need to run the statistical tests, coerce the statistical results into a useable format, and arrange them onto a plot. For this reason, people will often avoid annotating their plots in R, instead doing it in an outside program like Illustrator or using a different software like Prism. However, learning to add these tests t your plots programmatically can make your workflow more flexible. What if you add new data to your project? What if you decide to divide the data into new groups? What if you decide to remove an outlier? Adding these annotations via R code means you don't need to redo all these steps everytime something changes upstream of the plot.

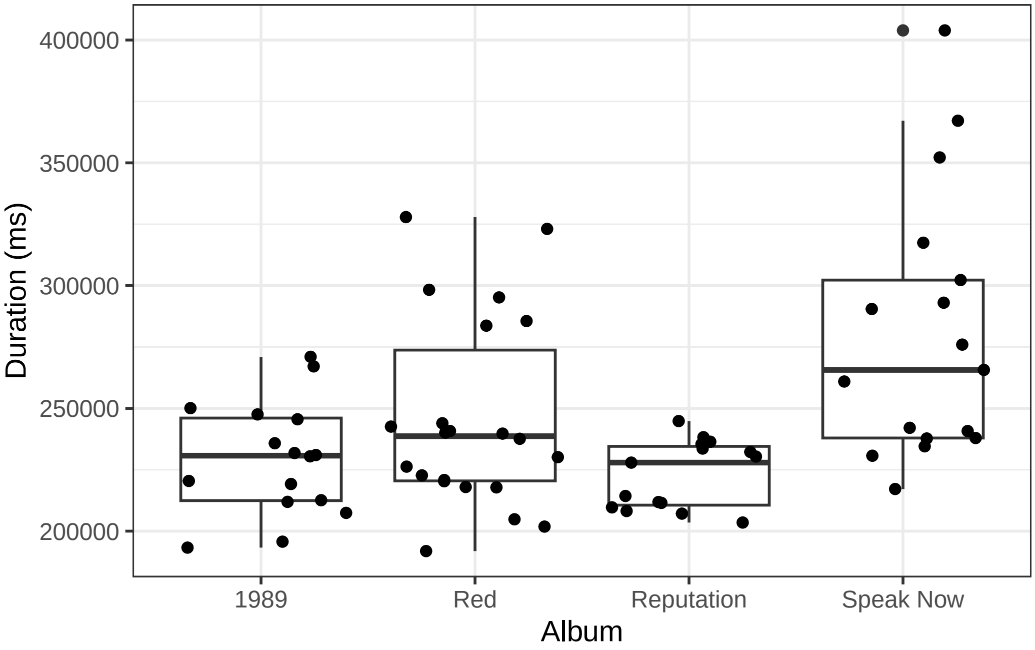
Loading packagtes and reading in the data

```r
library(tidyverse)
library(ggpubr)
library(rstatix)

taylor_data <- read_csv("data/taylor_albums.csv")
```

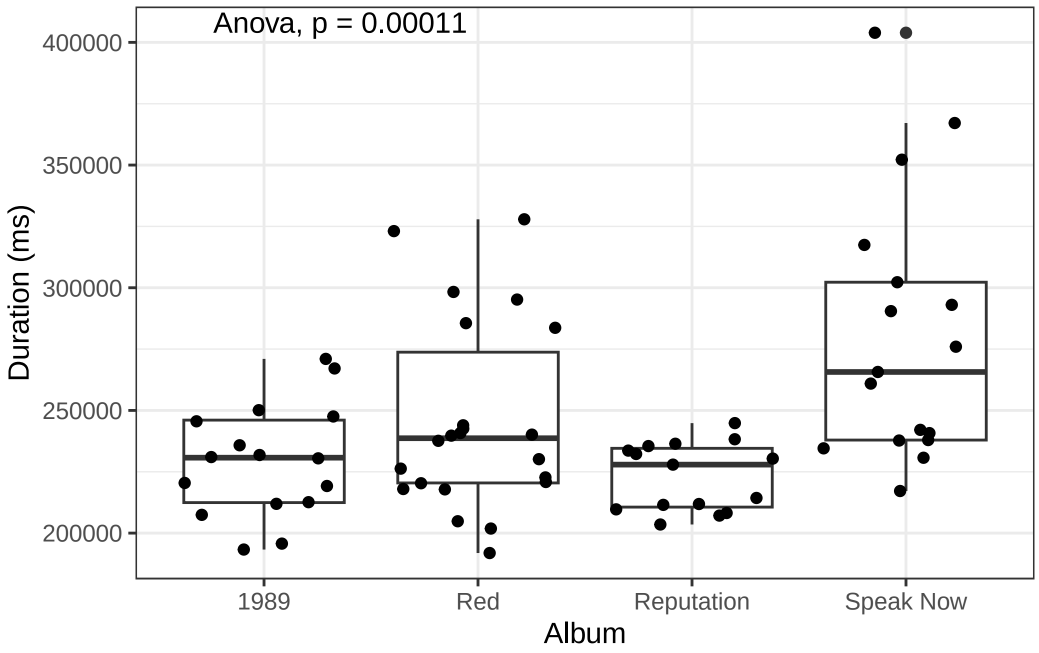
Our data holds statistics for four Taylor Swift albums. In our research, we might ask how the duration of songs compares between the albums. Let's make a boxplot to visualize that comparison.

```
ggplot(taylor_data, aes(x = album, y = duration_ms)) +
  geom_boxplot() +
  geom_jitter() +
  theme_bw() +
  labs(x = "Album", y = "Duration (ms)")
```



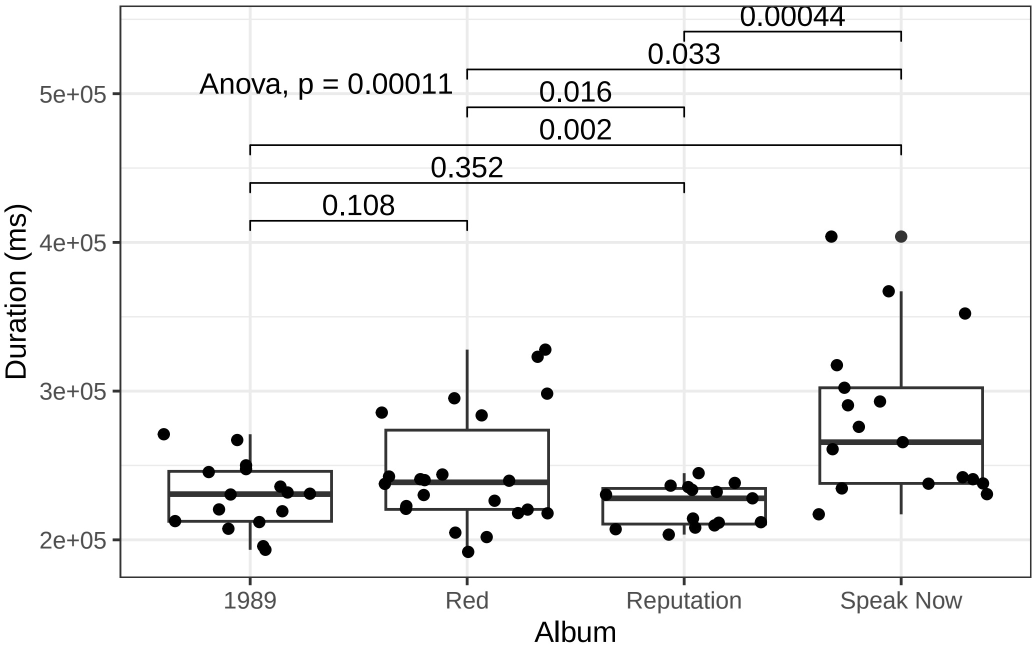
Now, we want to test when there are significant differences in duration between these four groups. First, we should probably run an ANOVA, to confirm there is a difference between at least two groups

```
ggplot(taylor_data, aes(x = album, y = duration_ms)) +
  geom_boxplot() +
  geom_jitter() +
  theme_bw() +
  labs(x = "Album", y = "Duration (ms)") +
  stat_compare_means(method = "anova")
```



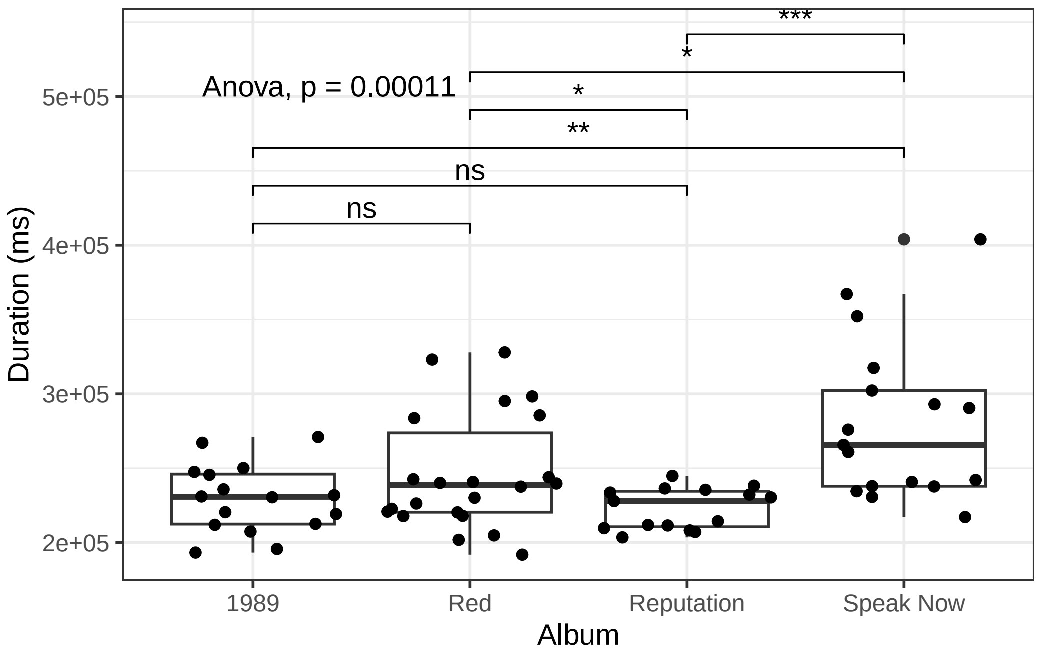
Great! There is a difference between at least two groups. Let's add pairwise t-tests to find out which groups are different.

```
ggplot(taylor_data, aes(x = album, y = duration_ms)) +
  geom_boxplot() +
  geom_jitter() +
  theme_bw() +
  labs(x = "Album", y = "Duration (ms)") +
  stat_compare_means(method = "anova", label.y =500000) +
  geom_pwc(method = "t_test")
```



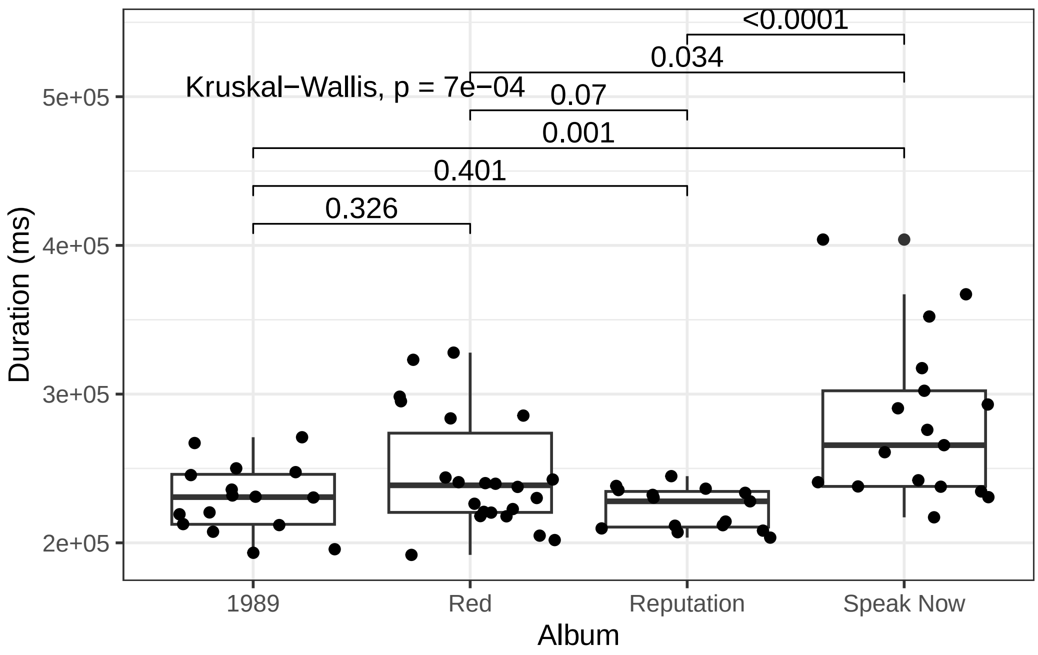
We can also add stars, instead of the p=value itself.

```
ggplot(taylor_data, aes(x = album, y = duration_ms)) +
  geom_boxplot() +
  geom_jitter() +
  theme_bw() +
  labs(x = "Album", y = "Duration (ms)") +
  stat_compare_means(method = "anova", label.y =500000) +
  geom_pwc(method = "t_test", label = "p.signif")
```



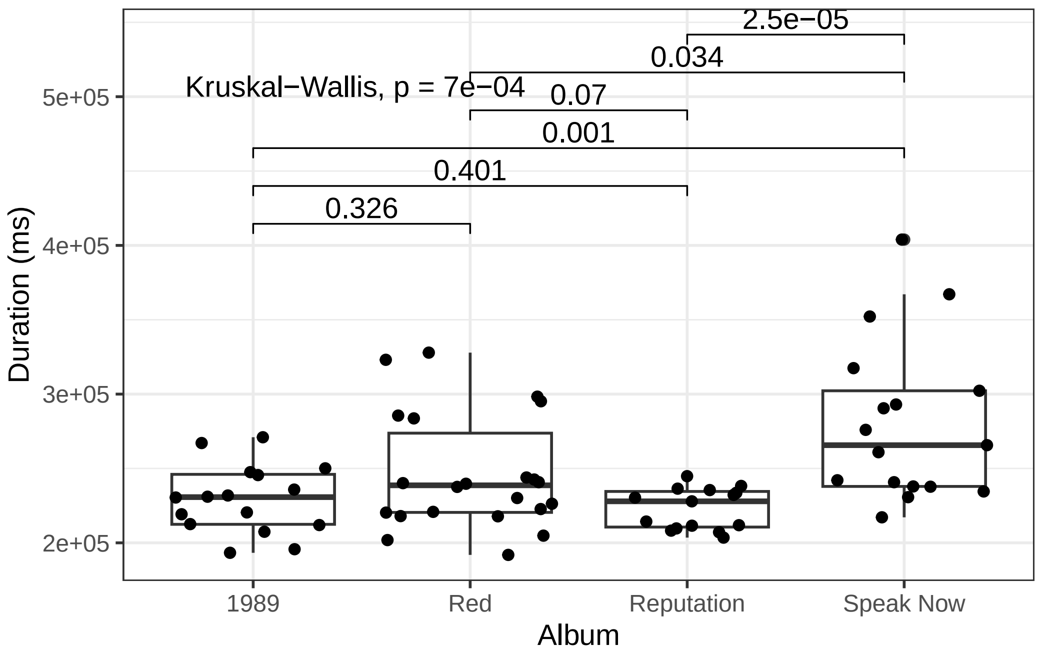
It's easy to use non-parametric tests, like Kruskal Wallace and Wilcoxon comparisons

```
ggplot(taylor_data, aes(x = album, y = duration_ms)) +
  geom_boxplot() +
  geom_jitter() +
  theme_bw() +
  labs(x = "Album", y = "Duration (ms)") +
  stat_compare_means(method = "kruskal.test", label.y =500000) +
  geom_pwc(method = "wilcox_test")
```

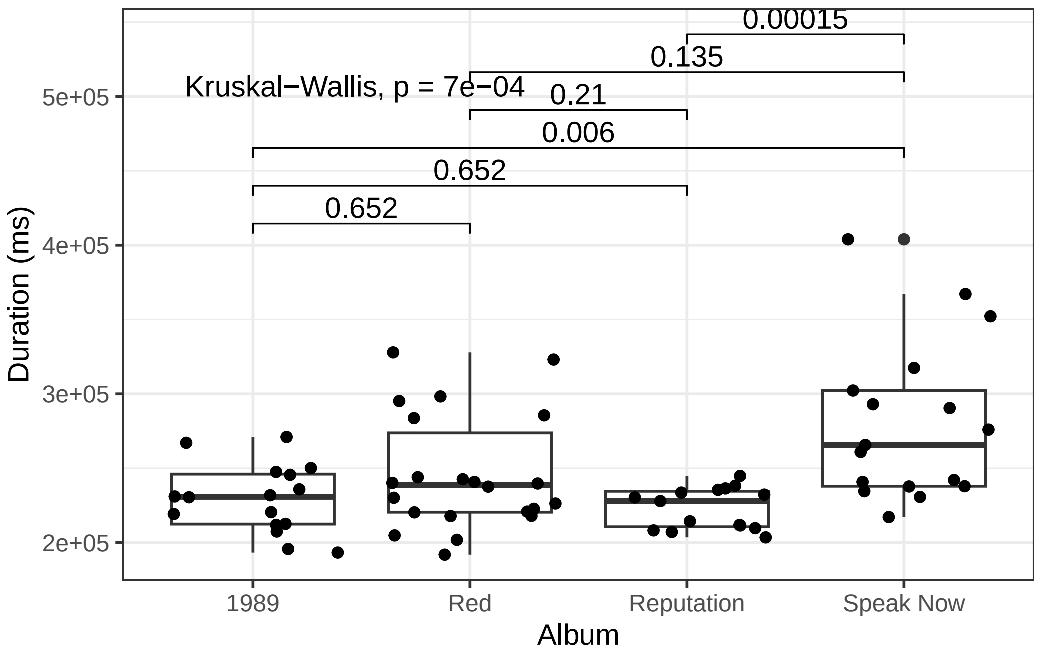


Finally, ggpubr can also adjusts your p-values automatically for multiple comparisons (which you should do!). It's easy to specify the adjustment method.

```
ggplot(taylor_data, aes(x = album, y = duration_ms)) +
  geom_boxplot() +
  geom_jitter() +
  theme_bw() +
  labs(x = "Album", y = "Duration (ms)") +
  stat_compare_means(method = "kruskal.test", label.y =500000) +
  geom_pwc(method = "wilcox_test", label = "p.adj", p.adjust.method = "none")
```



```
ggplot(taylor_data, aes(x = album, y = duration_ms)) +
  geom_boxplot() +
  geom_jitter() +
  theme_bw() +
  labs(x = "Album", y = "Duration (ms)") +
  stat_compare_means(method = "kruskal.test", label.y =500000) +
  geom_pwc(method = "wilcox_test", label = "p.adj", p.adjust.method = "holm")
```

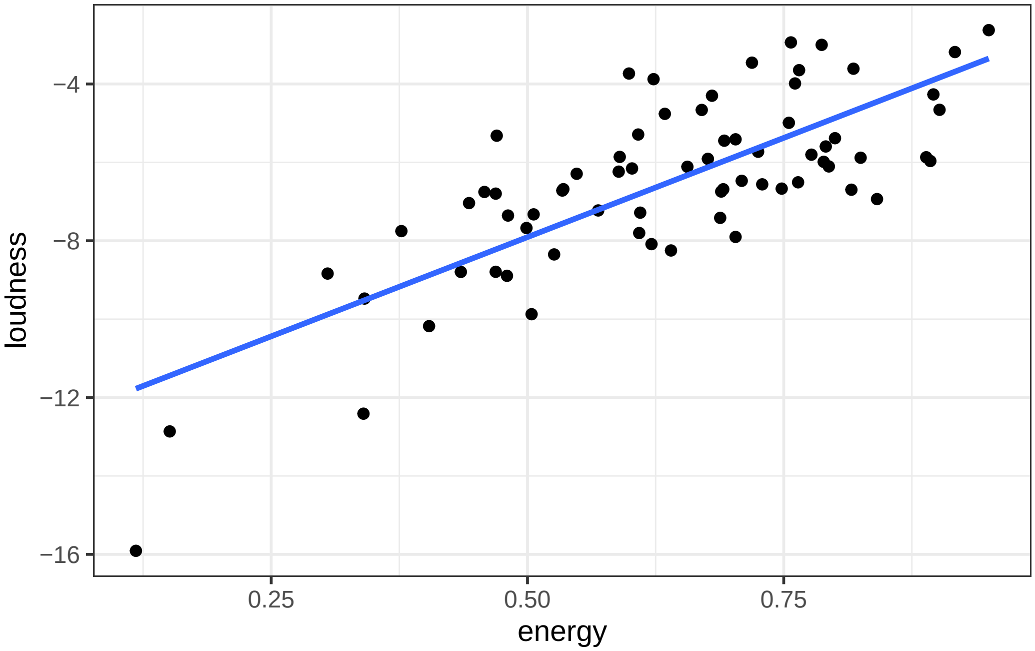


These two commands (`stat_compare_means` and `geom_pwc`) are very flexible in how the tests are run and how the labels are displayed; I recommend the ggpubr Reference log for more information.

Adding regression statistics

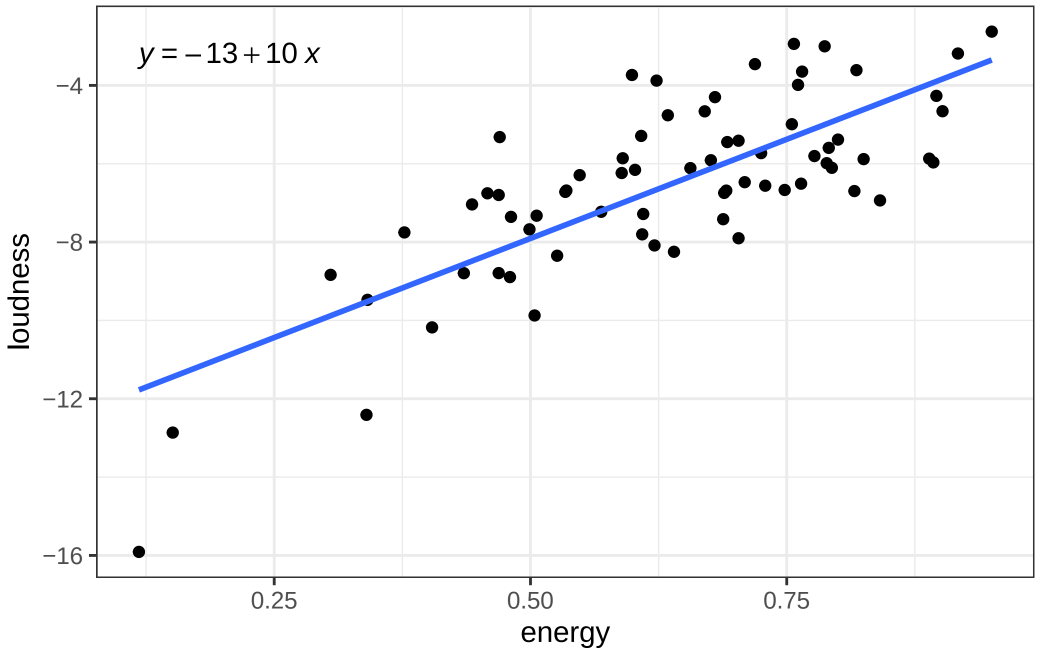
Another common area where we want to compare our plotting and statistical analysis is add information about a regression which we've run. Let's start by making a a scatterplot with a linear model on top.

```
ggplot(taylor_data, aes(x = energy, y = loudness)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw()
```



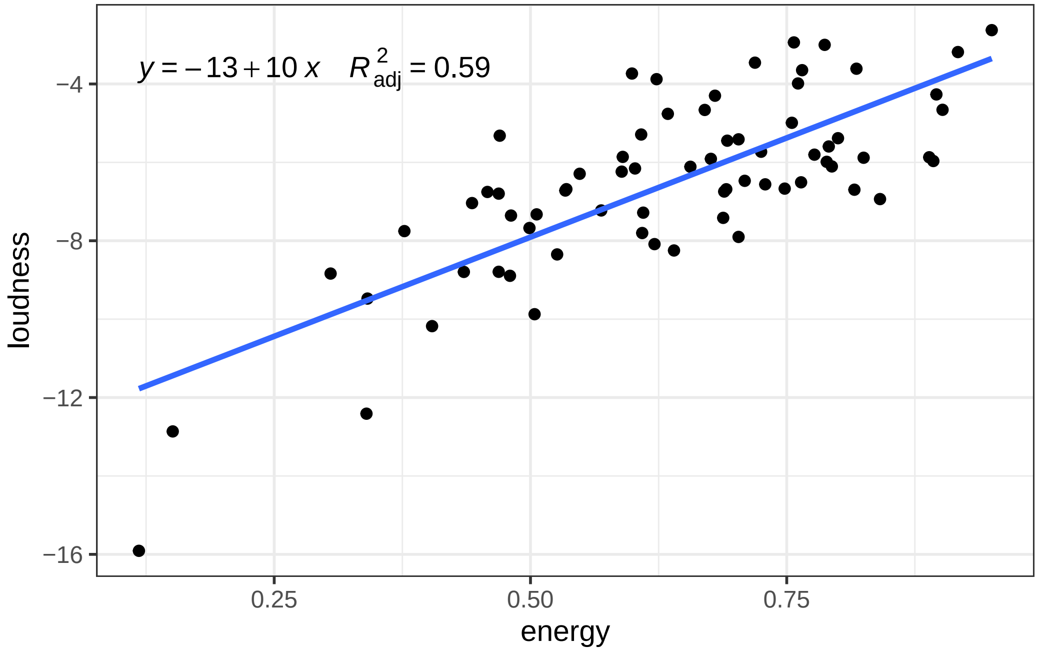
Now, let's use a `ggpubr::stat_regline_equation` to add the regression equation onto our plot.

```
ggplot(taylor_data, aes(x = energy, y = loudness)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw() +
  stat_regline_equation()
```



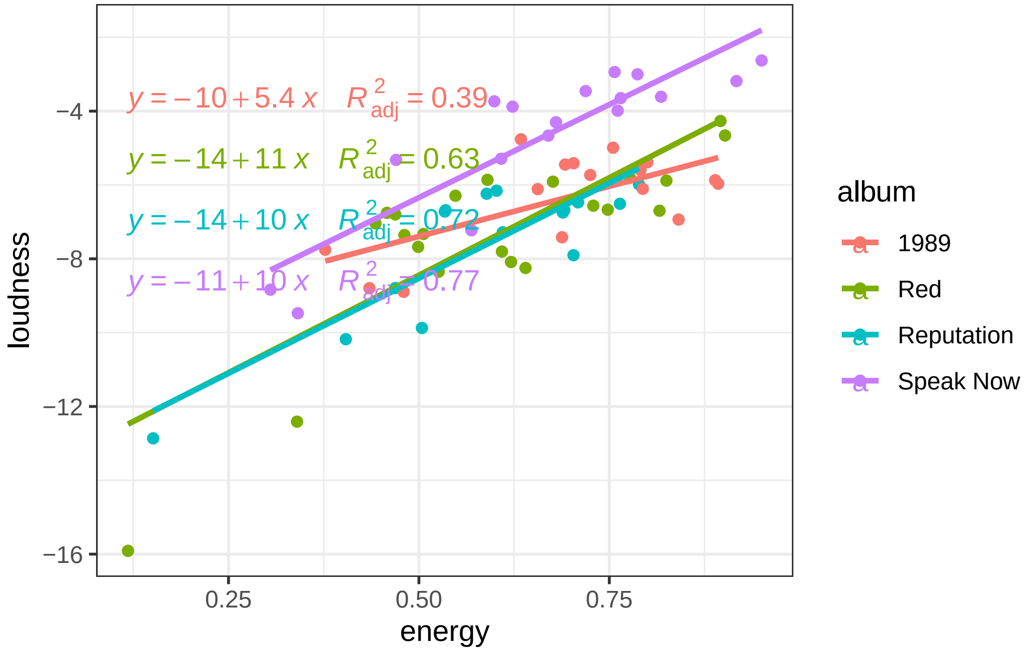
This tells us that for every increase in 1 unit of energy, loudness tends to increase by 10. What if we want to add the R-squared? In this case, we'll have to be more explicit about what we want to include in our label

```
ggplot(taylor_data, aes(x = energy, y = loudness)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw() +
  stat_regline_equation(aes(label = paste(after_stat(eq.label), after_stat(adj.rr.label),
```



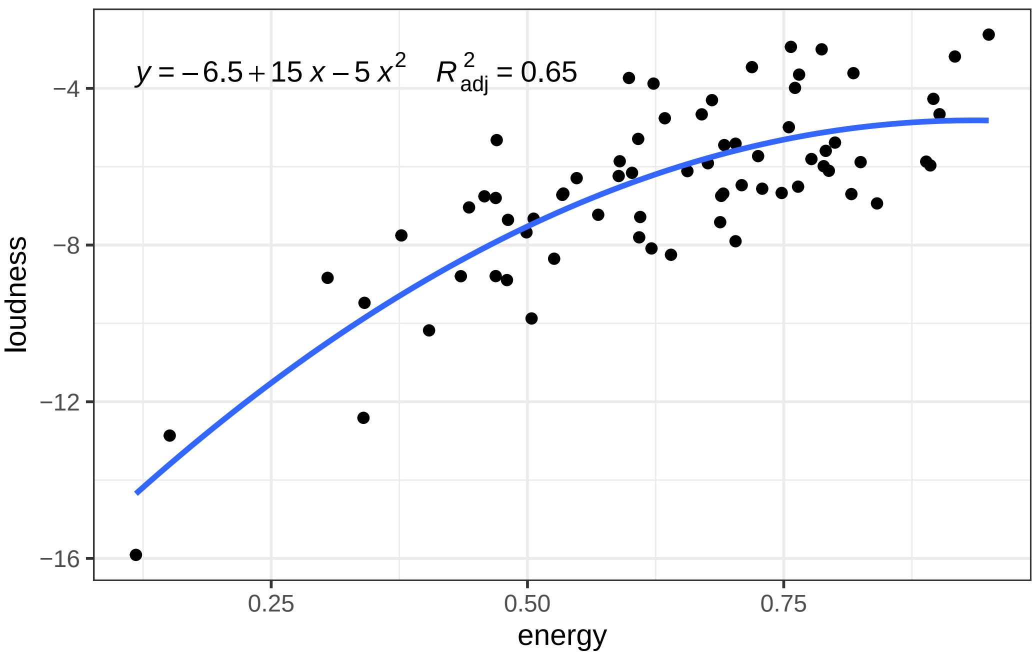
You can explore the other statistics calculated in this function on its help page. We can even add information for more complicated models. For instance, what if we want to calculate a line for each album?

```
ggplot(taylor_data, aes(x = energy, y = loudness, color = album)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw() +
  stat_regline_equation(aes(label = paste(after_stat(eq.label), after_stat(adj.rr.label),
```



Or what if we don't want a linear model?

```
ggplot(taylor_data, aes(x = energy, y = loudness)) +
  geom_point() +
  geom_smooth(formula = y~poly(x, 2), method = "lm", se = FALSE) +
  theme_bw() +
  stat_regline_equation(formula = y~poly(x, 2), aes(label = paste(after_stat(eq.label), af
```



Again, lots of flexibility here - feel free to play around on your own!