So now let's build an equation to predict points scored using some common basketball statistics.

So our dependent variable would now be points, and our independent variables would be some of the common basketball statistics that we have in our data set.

So for example, the number of two-point field goal attempts, the number of three-point field goal attempts, offensive rebounds, defensive rebounds, assists, steals, blocks, turnovers, free throw attempts-- we can use all of these.

So let's build this regression and call it PointsReg.

And that will just be equal to lm of PTS regressing on all those variables we just talked about.

So X2PA for two-point attempts, plus X3PA for three-point attempts, plus FTA for free throw attempts, AST for assists, ORB offensive rebounds, DRB for defensive rebounds, TOV for turnovers, and STL for steals.

And let's also throw in blocks, BLK.

Okay.

And as always, the data is from the NBA data set.

So we can go ahead and run this command.

So now that we've created our points regression model, let's take a look at the summary.

This is always the first thing you should do.

Okay, so taking a look at this, we can see that some of our variables are indeed very, very, significant.

Others are less significant.

For example, steals only has one significance star.

And some don't seem to be significant at all.

For example, defensive rebounds, turnovers, and blocks.

We do have a pretty good R-squared value, 0.8992, so it shows that there really is a linear relationship between points and all of these basketball statistics.

Let's compute the residuals here.

We can just do that by directly calling them, so that's PointsReg$residuals.

So there's this giant list of them, and we'll use this to compute the sum of squared errors.

SSE, standing for sum of squared errors, is just equal to the sum(PointsReg$residuals^2).

So what is the sum of squared errors here?

It's quite a lot-- 28,394,314.

So the sum of squared errors number is not really a very interpretable quantity.

But remember, we can also calculate the root mean squared error, which is much more interpretable.

It's more like the average error we make in our predictions.

So the root mean squared error, RMSE-- let's calculate it here.

So RMSE is just equal to the square root of the sum of squared errors divided by n, where n here is the number of rows in our data set.

So the RMSE in our case is 184.4.

So on average, we make an error of about 184.4 points.

That seems like quite a lot, until you remember that the average number of points in a season is, let's see, mean(NBA$PTS).

This will give us the average number of points in a season, and it's 8,370.

So, okay, if we have an average number of points of 8,370, being off by about 184.4 points is really not so bad.

But I think we still have room for improvement in this model.

If you recall, not all the variables were significant.

Let's see if we can remove some of the insignificant variables one at a time.

We'll take a look again at our model, summary(PointsReg), in order to figure out which variable we should remove first.

The first variable we would want to remove is probably turnovers.

And why do I say turnovers?

It's because the p value for turnovers, which you see here in this column, 0.6859, is the highest of all of the p values.

So that means that turnovers is the least statistically significant variable in our model.

So let's create a new regression model without turnovers.

An easy way to do this is just to use your up arrow on your keyboard, and scroll up through all of your previous commands until you find the command where you defined the regression model.

Okay, so this is the command where we defined the model.

Now let's delete turnovers, and then we can rename the model, and we'll call this one just PointsReg2.

So in our first regression model, PointsReg, we had an R-squared of 0.8992.

Let's take a look at the R-squared of PointsReg2.

And we see that it's 0.8991.

So almost exactly identical.

It does go down, as we would expect, but very, very slightly.

So it seems that we're justified in removing turnovers.

Let's see if we can remove another one of the insignificant variables.

The next one, based on p-value, that we would want to remove is defensive rebounds.

So again, let's create our model, taking out defensive rebounds, and calling this PointsReg3.

We'll just scroll up again.

Take out DRB, for defensive rebounds, and change the name of this to PointsReg3 so we don't overwrite PointsReg2.

Let's look at the summary again to see if the R-squared has changed.

And it's the same, it's 0.8991.

So I think we're justified again in removing defensive rebounds.

Let's try this one more time and see if we can remove blocks.

So we'll remove blocks, and call it PointsReg4.

Take a look at the summary of PointsReg4.

And again, the R-squared value stayed the same.

So now we've gotten down to a model which is a bit simpler.

All the variables are significant.

We've still got an R-squared 0.899.

And let's take a look now at the sum of squared errors and the root mean square error, just to make sure we didn't inflate those too much by removing a few variables.

So, remember that the sum of squared errors that we had in the original model was this giant number, 28,394,314.

And the root mean squared error, the much more interpretable number, was 184.4.

So those are the numbers we'll be comparing against when we calculate the new sum of squared errors of the new model, PointsReg4.

So let's call this SSE_4.

And that will just be equal to sum(PointsReg4$residuals^2).

And then RMSE_4 will just be the square root of SSE_4 divided by n, the number of rows in our data set.

Okay, so let's take a look at these.

The new sum of squared errors is now 28,421,465.

Again, I find this very difficult to interpret.

I like to look at the root mean squared error instead.

So the root mean squared error here is just RMSE_4, and so it's 184.5.

So although we've increased the root mean squared error a little bit by removing those variables, it's really a very, very, small amount.

Essentially, we've kept the root mean squared error the same.

So it seems like we've narrowed down on a much better model because it's simpler, it's more interpretable, and it's got just about the same amount of error.