Class 27: Linear Modeling III

Dr. Glasbrenner

December 4, 2017

Setup

```
# Configure settings for compiling to HTML and PDF
knitr::opts_chunk$set(
  echo = TRUE, eval = TRUE, fig.width = 5, fig.asp = 0.618, out.width = "70%",
  dpi = 120, fig.align = "center", cache = TRUE)
# Load required packages
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(modelr))
suppressPackageStartupMessages(library(broom))
load(url("http://fall17.cds101.com/R/mariokart_cross_validation.RData"))
# Load datasets
mariokart <- read_rds(url("http://fall17.cds101.com/datasets/mariokart.rds"))
```

Get a blank working RMarkdown file:

```
download.file("http://fall17.cds101.com/documents/class27.txt",
              destfile = "class27.Rmd")
```

Exploration

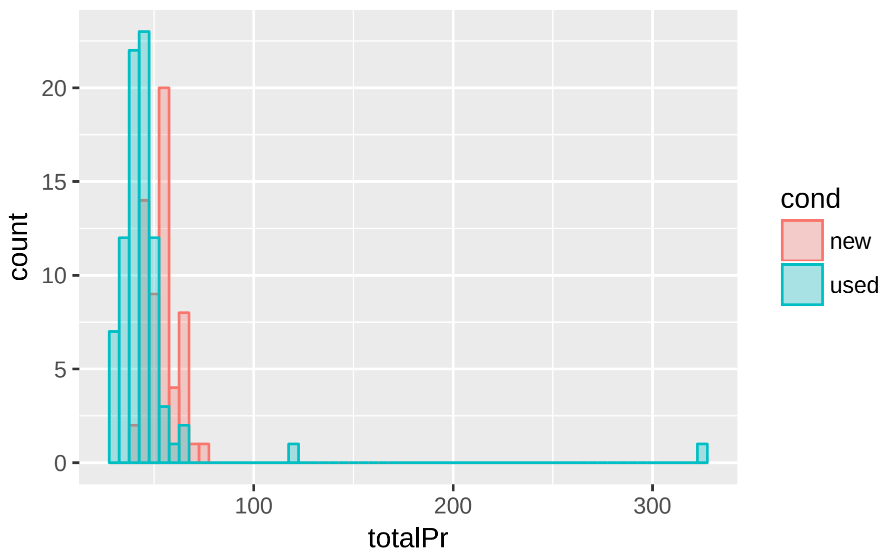
View the first several entries of the *Mario Kart* dataset:

```
glimpse(mariokart, width = 80)
```

```
## Observations: 143
## Variables: 12
## $ ID          <dbl> 150377422259, 260483376854, 320432342985, 280405224677, ...
## $ duration    <int> 3, 7, 3, 3, 1, 3, 1, 1, 3, 7, 1, 1, 1, 1, 7, 7, 3, 3, 1,...
## $ nBids       <int> 20, 13, 16, 18, 20, 19, 13, 15, 29, 8, 15, 15, 13, 16, 6...
## $ cond        <fctr> new, used, new, new, new, new, used, new, used, used, n...
## $ startPr     <dbl> 0.99, 0.99, 0.99, 0.99, 0.01, 0.99, 0.01, 1.00, 0.99, 19...
## $ shipPr      <dbl> 4.00, 3.99, 3.50, 0.00, 0.00, 4.00, 0.00, 2.99, 4.00, 4....
## $ totalPr     <dbl> 51.55, 37.04, 45.50, 44.00, 71.00, 45.00, 37.02, 53.99, ...
## $ shipSp      <fctr> standard, firstClass, firstClass, standard, media, stan...
## $ sellerRate <int> 1580, 365, 998, 7, 820, 270144, 7284, 4858, 27, 201, 485...
## $ stockPhoto <fctr> yes, yes, no, yes, yes, yes, yes, yes, yes, no, yes, ye...
## $ wheels      <int> 1, 1, 1, 1, 2, 0, 0, 2, 1, 1, 2, 2, 2, 2, 1, 0, 1, 1, 2,...
## $ title       <fctr> ~~ Wii MARIO KART &amp; WHEEL ~ NINTENDO Wii ~ BRAND NE...
```

Check for outliers:

```
ggplot(mariokart) +
  geom_histogram(
    mapping = aes(x = totalPr, fill = cond, color = cond),
    position = "identity", alpha = 0.3, binwidth = 5,
    center = 0)
```



What are these outliers?

```
mariokart %>%
  filter(totalPr > 100) %>%
  glimpse(width = 80)
```

```
## Observations: 2
## Variables: 12
## $ ID          <dbl> 110439174663, 130335427560
## $ duration    <int> 7, 3
## $ nBids       <int> 22, 27
## $ cond        <fctr> used, used
## $ startPr     <dbl> 1.00, 6.95
## $ shipPr      <dbl> 25.51, 4.00
## $ totalPr     <dbl> 326.51, 118.50
## $ shipSp      <fctr> parcel, parcel
## $ sellerRate <int> 115, 41
## $ stockPhoto <fctr> no, no
## $ wheels      <int> 2, 0
## $ title       <fctr> Nintedo Wii Console Bundle Guitar Hero 5 Mario Kart , 1...
```

Look at the titles:

```
mariokart %>%
  filter(totalPr > 100) %>%
  select(title) %>%
  head()
```

```
## # A tibble: 2 x 1
##                                                       title
##                                                      <fctr>
## 1    Nintedo Wii Console Bundle Guitar Hero 5 Mario Kart
## 2 10 Nintendo Wii Games - MarioKart Wii, SpiderMan 3, etc
```

These are bundled items, not like the rest of the items in the dataset. Let's remove them:

```
mariokart_no_outliers <- mariokart %>% filter(totalPr <= 100)
```

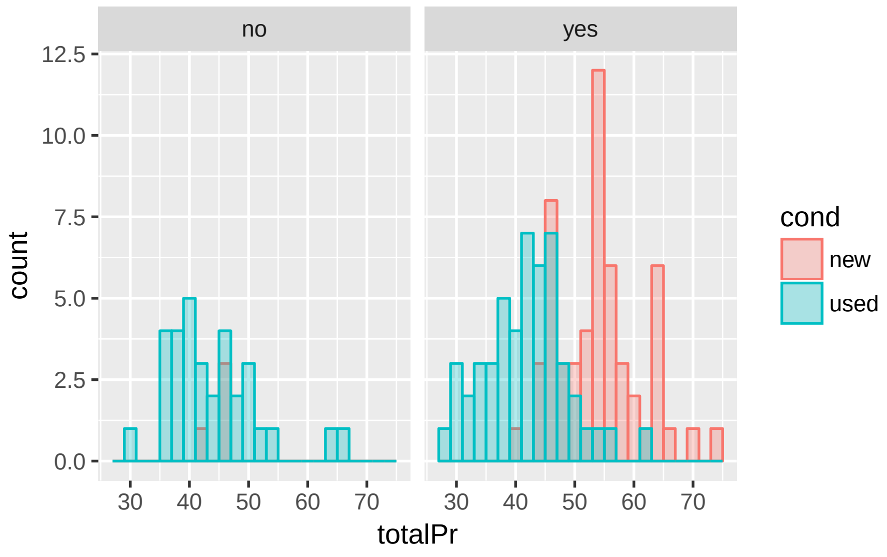
Pare down the dataset and stick with a subset of variables:

```r
mariokart <- select(
  mariokart_no_outliers, totalPr, cond, stockPhoto, duration, wheels)
glimpse(mariokart, width = 80)
```

```
## Observations: 141
## Variables: 5
## $ totalPr    <dbl> 51.55, 37.04, 45.50, 44.00, 71.00, 45.00, 37.02, 53.99, ...
## $ cond       <fctr> new, used, new, new, new, new, used, new, used, used, n...
## $ stockPhoto <fctr> yes, yes, no, yes, yes, yes, yes, yes, yes, no, yes, ye...
## $ duration   <int> 3, 7, 3, 3, 1, 3, 1, 1, 3, 7, 1, 1, 1, 1, 7, 7, 3, 3, 1,...
## $ wheels     <int> 1, 1, 1, 1, 2, 0, 0, 2, 1, 1, 2, 2, 2, 2, 1, 0, 1, 1, 2,...
```

Visualize how much the game condition and whether a stock photo was part of the listing affected the total price:

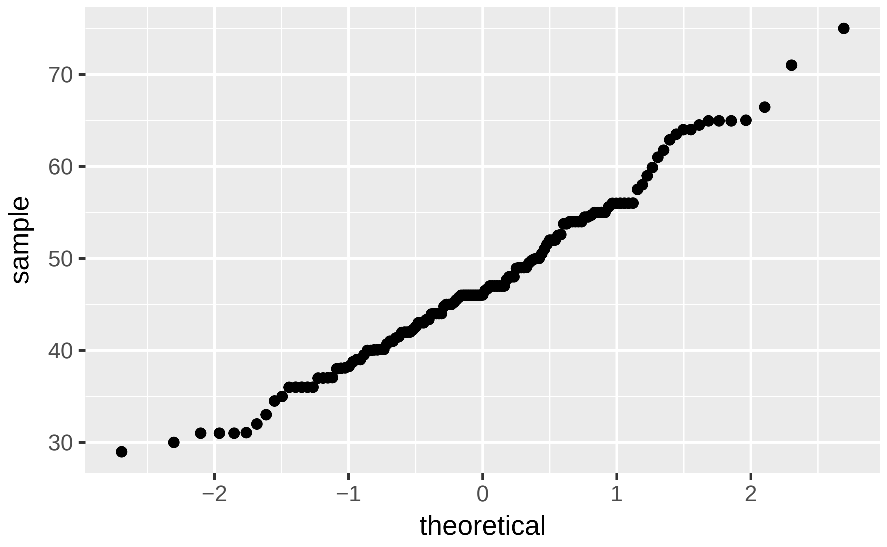
```r
ggplot(mariokart) +
  geom_histogram(
    mapping = aes(totalPr, fill = cond, color = cond), position = "identity",
    alpha = 0.3, center = 0, binwidth = 2) +
  facet_wrap(~stockPhoto)
```



Is `totalPr` nearly normal? How does the distribution shape change if we split the dataset by categories?

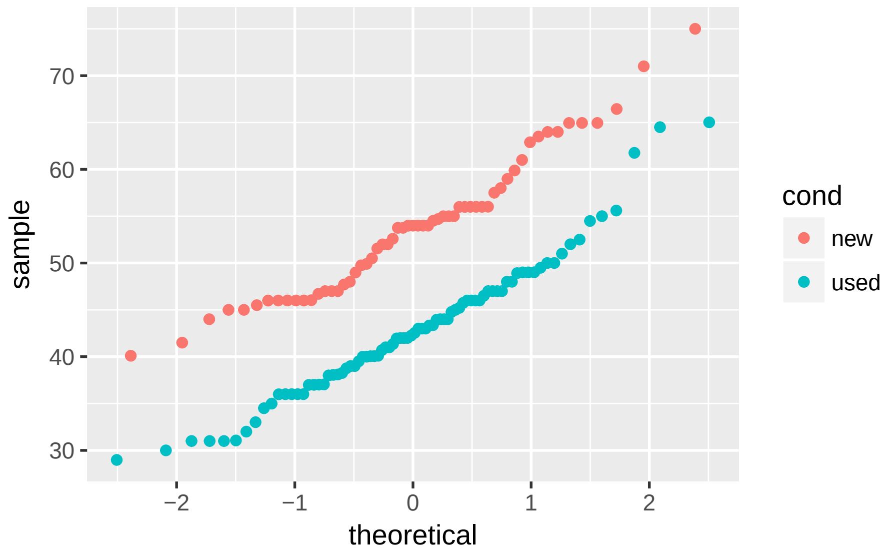
First, make a Q-Q plot to check `totalPr` by itself:

```r
ggplot(mariokart) +
  geom_qq(mapping = aes(sample = totalPr))
```



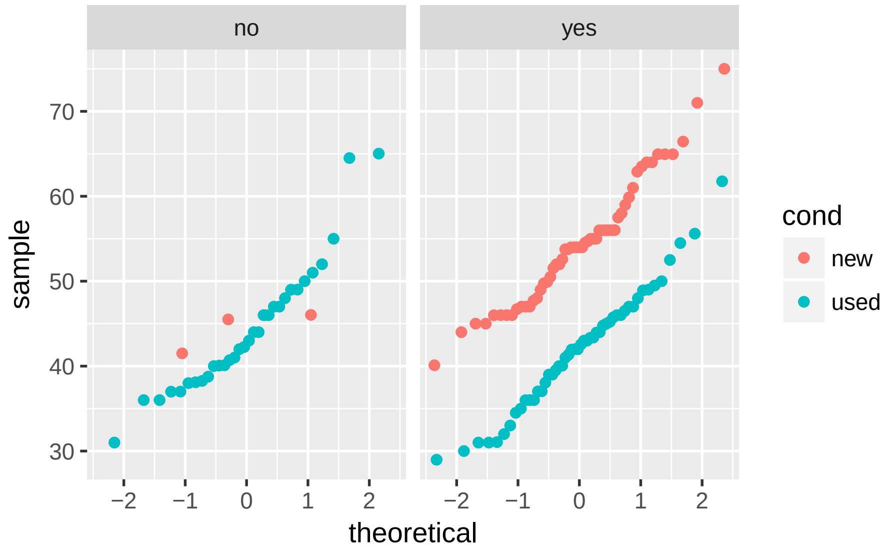
Next, make a Q-Q plot with `totalPr` split by game condition:

```r
ggplot(mariokart) +
  geom_qq(mapping = aes(sample = totalPr, color = cond))
```



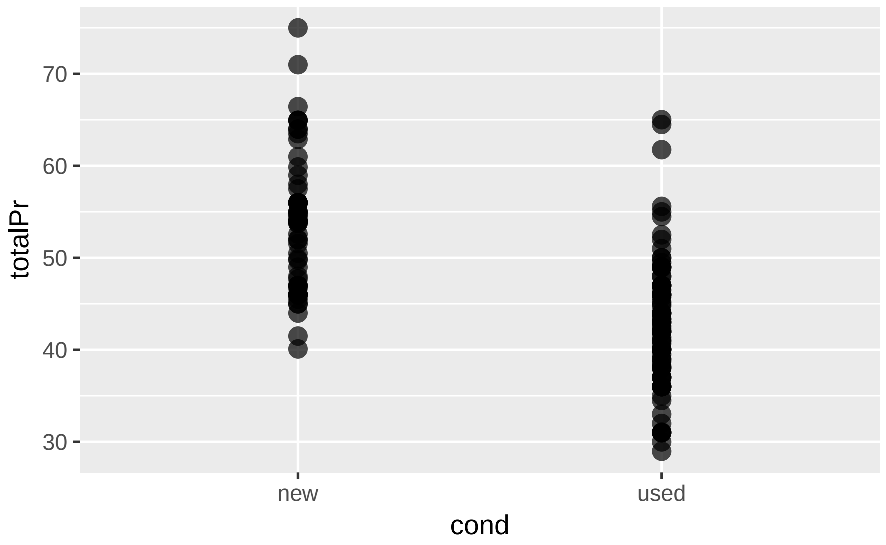
Finally, make a Q-Q plot with `totalPr` split by game condition and faceted by `stockPhoto`:

```r
ggplot(mariokart) +
  geom_qq(mapping = aes(sample = totalPr, color = cond)) +
  facet_wrap( ~ stockPhoto)
```



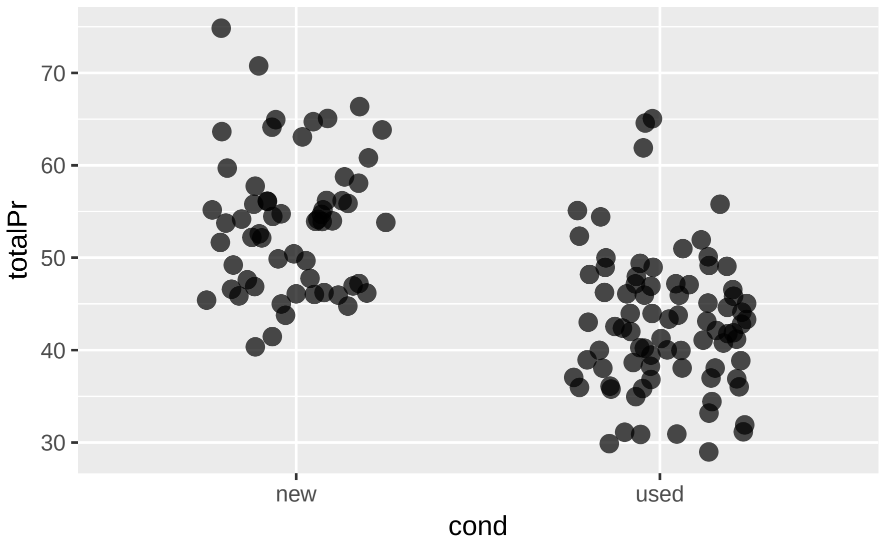
What happens if we plot `totalPr` as a function of `cond`, a categorical variable?

```r
ggplot(mariokart) +
  geom_point(mapping = aes(cond, totalPr), size = 3, alpha = 0.7)
```



It's a little easier to see the points if we jitter them:

```r
ggplot(mariokart) +
  geom_jitter(mapping = aes(cond, totalPr), size = 3, alpha = 0.7,
              width = 0.25, height = 0.25)
```



Univariate linear regression model of Mario Kart's total price

First, let's try and model the game price by the cond categorical variable:

```
mariokart_linear_model <- lm(totalPr~cond, data = mariokart)
grid <- data_grid(mariokart, cond)
grid <- add_predictions(grid, mariokart_linear_model)
mariokart_linear_compare <- select(mariokart, cond, totalPr)
mariokart_linear_compare <- add_residuals(
  data = mariokart_linear_compare, model = mariokart_linear_model)
```

Remember what grid looks like:

```
head(grid, n = 10)
```

```
## # A tibble: 2 x 2
##     cond     pred
##   <fctr>    <dbl>
## 1    new 53.77068
## 2   used 42.87110
```

Here's what the residuals look like relative to the data points:

```
head(mariokart_linear_compare, n = 10)
```

```
## # A tibble: 10 x 3
##      cond totalPr     resid
##    <fctr>   <dbl>     <dbl>
##  1    new   51.55 -2.220678
##  2   used   37.04 -5.831098
##  3    new   45.50 -8.270678
##  4    new   44.00 -9.770678
##  5    new   71.00 17.229322
##  6    new   45.00 -8.770678
##  7   used   37.02 -5.851098
##  8    new   53.99  0.219322
##  9   used   47.00  4.128902
## 10   used   50.00  7.128902
```

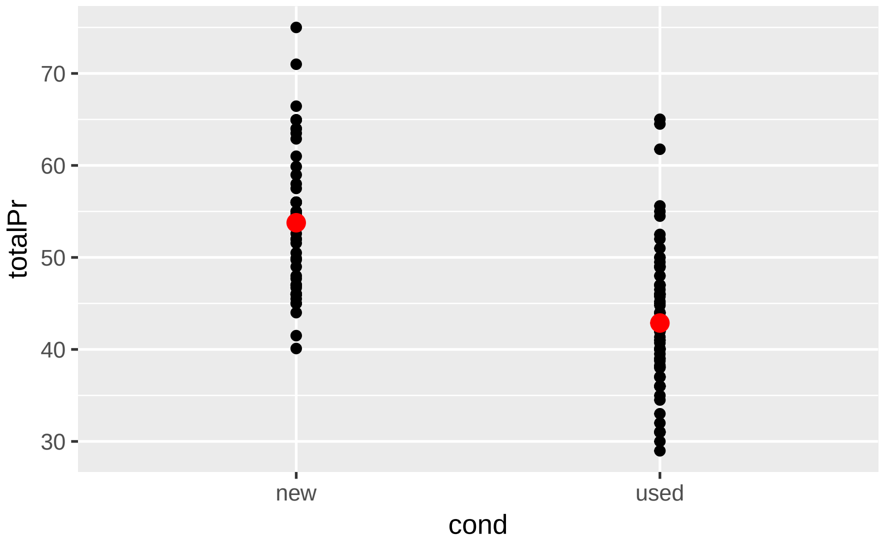
Print out some basic details about the linear fit:

```r
summary(mariokart_linear_model)
```

```
##
## Call:
## lm(formula = totalPr ~ cond, data = mariokart)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.8911  -5.8311   0.1289   4.1289  22.1489
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  53.7707     0.9596  56.034  < 2e-16 ***
## condused    -10.8996     1.2583  -8.662 1.06e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.371 on 139 degrees of freedom
## Multiple R-squared:  0.3506, Adjusted R-squared:  0.3459
## F-statistic: 75.03 on 1 and 139 DF,  p-value: 1.056e-14
```

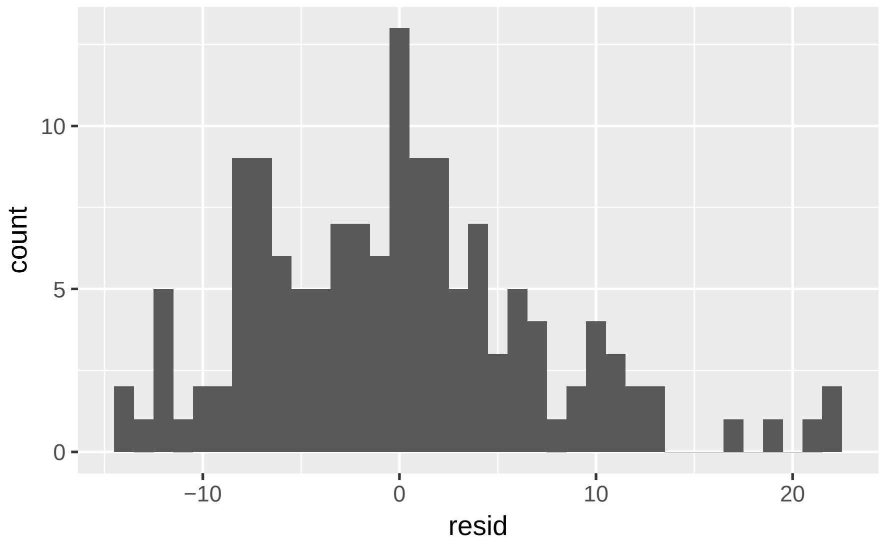
Since cond is categorical, what will it look like when we overlay our models' predictions on the data? We make the plot:

```r
ggplot(mariokart) +
  geom_point(aes(cond, totalPr)) +
  geom_point(aes(cond, pred), data=grid, color="red", size=3)
```



Next, let's inspect the residuals:

```r
ggplot(mariokart_linear_compare) +
  geom_histogram(aes(resid), binwidth = 1, center = 0)
```

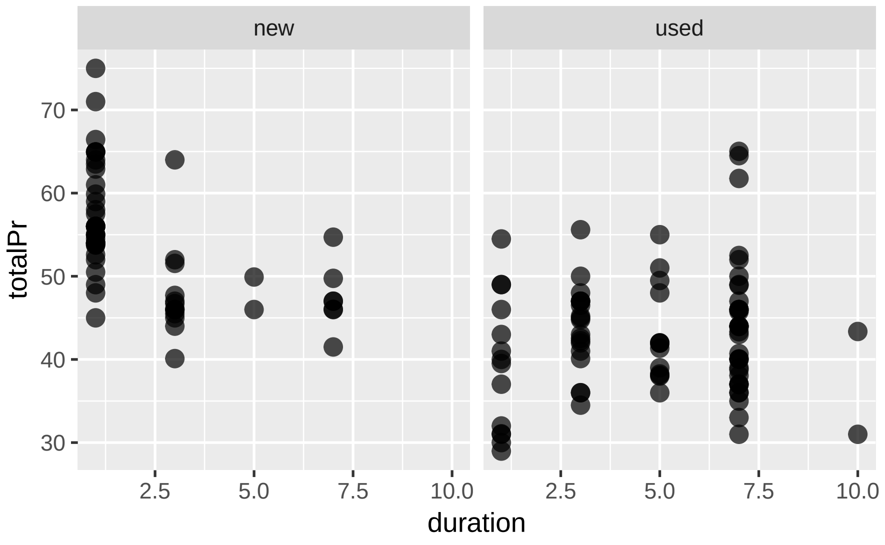


It seems like we might need more than one variable to model this dataset.

Multivariate linear regression model of Mario Kart's total price

Can other variables have an effect? Let's see how `cond` is affected by `duration`:

```r
ggplot(mariokart) +
  geom_point(aes(duration, totalPr), size=3, alpha=0.7) +
  facet_wrap(~cond)
```



There's a slight dependence for new games, but overall the dependence is weak and the two variables move independently of one another.

Let's build a linear regression model taking the variables `cond`, `stockPhoto`, `duration`, and `wheels` all into account. We assume the variables are independent, i.e. we do not consider interaction terms such as `cond * duration`. The model is built in a way that's similar to the univariate case:

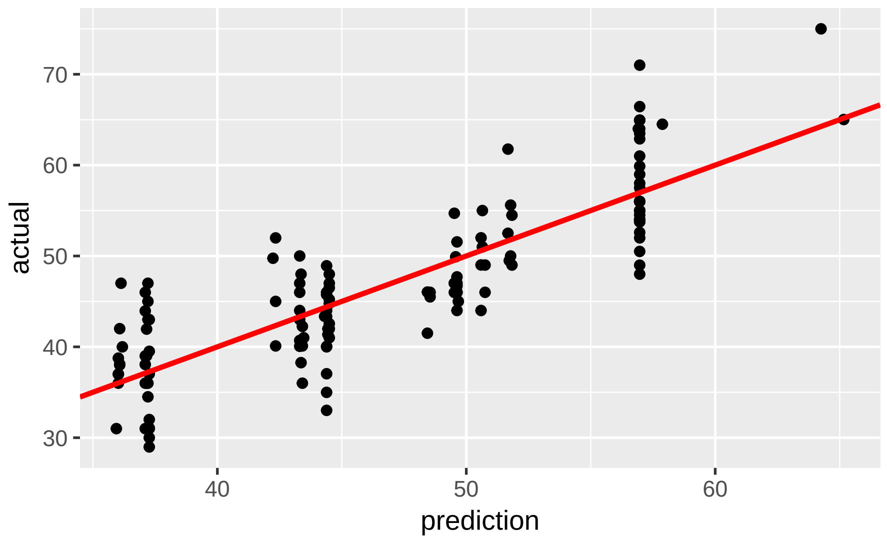
```r
mariokart_model2 <- lm(totalPr ~ cond + stockPhoto + duration + wheels,
                       data=mariokart)
grid <- data_grid(mariokart, cond, stockPhoto, duration, wheels)
grid <- add_predictions(grid, mariokart_model2)
mariokart_model2_compare <- select(mariokart, everything())
mariokart_model2_compare <- add_residuals(
  mariokart_model2_compare, mariokart_model2)
```

Because we have a multivariate model, we cannot create a single visualization that shows how the model responds to all variables at the same time. However, there are some other methods for visualizing a multivariate model's performance. One of them is to plot the model's predicted values for each data point as a function of the actual values. To help create this plot, we can use `predict()`, which takes the `lm()` model saved in a variable as its input, and then gives you a list of predictions as output. To create this plot, first we create a `tibble` containing the predicted and actual values of `totalPr`:

```
mariokart_model2_pred_vs_actual <- tibble(
  prediction = predict(mariokart_model2), actual = mariokart$totalPr)
```

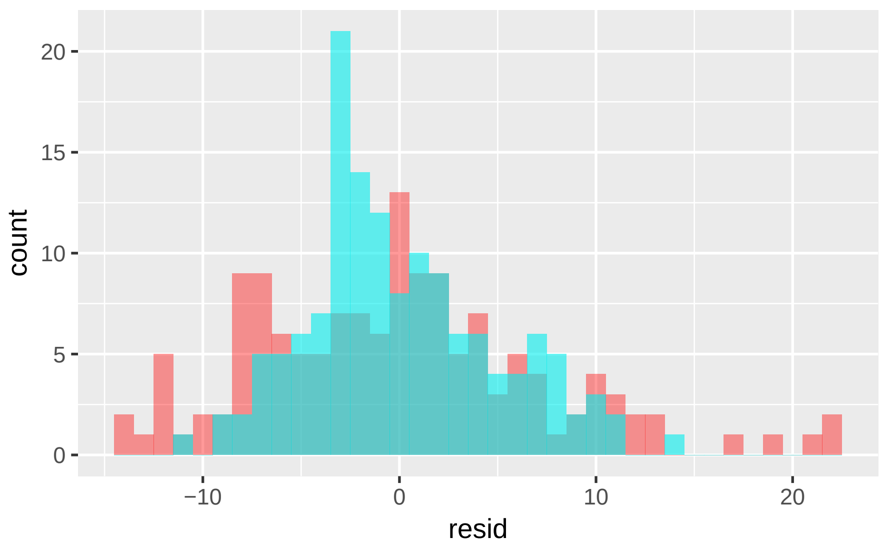
Then we create a scatterplot of the table. The straight line shows what a perfect model would correspond to, so deviations from the line show where the model fails to be quantitatively accurate (note that this is just another way to look at residuals):

```
ggplot(mariokart_model2_pred_vs_actual) +
  geom_point(aes(prediction, actual)) +
  geom_abline(slope = 1, intercept = 0, color = "red", size = 1)
```



We can also look at the residuals the traditional way. Let's compute the residuals and compare them against the univariate model with just cond:

```
ggplot(mariokart_model2_compare) +
  geom_histogram(data = mariokart_linear_compare, mapping = aes(resid),
                 alpha = 0.4, binwidth = 1, fill = "red",
                 position = "identity", center = 0) +
  geom_histogram(mapping = aes(resid), binwidth = 1, alpha = 0.6,
                 fill = "cyan2", position = "identity", center = 0)
```



The overall spread in the residuals has decreased, so it appears that the multivariate model is better at prediction.

Model comparison

The above comparison of residuals indirectly raises the general question of how to compare models against one another. For example, how should we compare models with different numbers of variables? One way is to compute a parameter like the Akaike information criterion (AIC), which is described below:

Akaike information criterion (AIC)

The Akaike information criterion is defined as follows:

> The Akaike information criterion (AIC) is a measure of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Hence, AIC provides a means for model selection. AIC is founded on information theory: it offers a relative estimate of the information lost when a given model is used to represent the process that generates the data. In doing so, it deals with the trade-off between the goodness of fit of the model and the complexity of the model. AIC does not provide a test of a model in the sense of testing a null hypothesis, so it can tell nothing about the quality of the model in an absolute sense. If all the candidate models fit poorly, AIC will not give any warning of that. Wikipedia

In terms of using the AIC in practice:

> To apply AIC in practice, we start with a set of candidate models, and then find the models' corresponding AIC values. There will almost always be information lost due to using a candidate model to represent the "true" model (i.e. the process that generates the data). We wish to select, from among the candidate models, the model that minimizes the information loss. We cannot choose with certainty, but we can minimize the estimated information loss.
>
> As an example, suppose that there are three candidate models, whose AIC values are 100, 102, and 110. Then the second model is $\exp((100 - 102)/2) = 0.368$ times as probable as the first model to minimize the information loss. Similarly, the third model is $\exp((100 - 110)/2) = 0.007$ times as probable as the first model to minimize the information loss.
>
> In this example, we would omit the third model from further consideration. We then have three options: (1) gather more data, in the hope that this will allow clearly distinguishing between the first two models; (2) simply conclude that the data is insufficient to support selecting one model from among the first two;

(3) take a weighted average of the first two models, with weights proportional to 1 and 0.368, respectively, and then do statistical inference based on the weighted multimodel Wikipedia

Comparing linear regression models of the Mario Kart dataset using R

In R, apply the `AIC()` function to a variable holding the results of a `lm()` calculation. We want the AIC score to be as small as possible, and the model with the lowest score is selected (if scores are close, then those are the plausible models). For the multivariate model with variables `cond`, `stockPhoto`, `duration`, and `wheels`:

```
AIC(mariokart_model2)
```

```
## [1] 855.2946
```

For the univariate model with just `cond`:

```
AIC(mariokart_linear_model)
```

```
## [1] 967.4329
```

The multivariate model has a lower AIC score, so we would select that model over the univariate one. This agrees with our intuition gained from visually comparing the residual distribution of each model.

Split dataset 80/20

Frequently, it's good practice to split a dataset prior to testing a model. Here we'll demonstrate how to create an 80%/20% split, with 80% being the training set and 20% being the testing set.

```
mariokart_split <- mariokart %>%
  crossv_mc(n = 1, test = 0.20)
```

The splitting mechanism in the above code is pretty basic. The outputs tell us the row numbers to use for training, and the row numbers to use for testing.

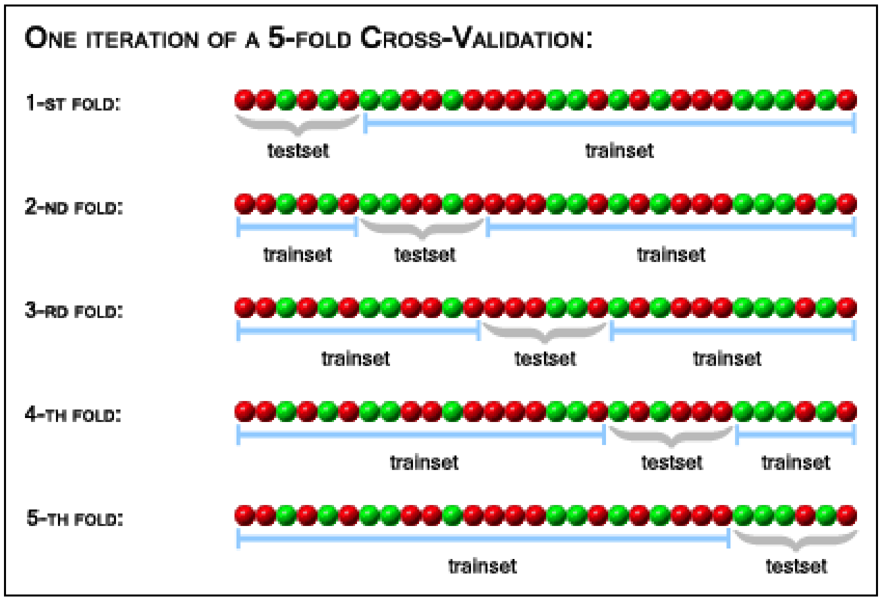
```
training_set_row_ids <- mariokart_split %>%
  pull(train) %>%
  map("idx") %>%
  flatten_int()

testing_set_row_ids <- mariokart_split %>%
  pull(test) %>%
  map("idx") %>%
  flatten_int()
```

Now we can slice our original data to get the two sets:

```
train <- mariokart %>%
  slice(training_set_row_ids)
test <- mariokart %>%
  slice(testing_set_row_ids)
```

k-fold cross-validation

K-fold cross-validation is one kind of cross-validation procedure that's available. These methods allow us to estimate how robust the model is by systematically removing different chunks of the dataset and repeating the fitting process. The picture below illustrates what it means:


ONE ITERATION OF A 5-FOLD CROSS-VALIDATION:
1-ST FOLD:
testset
trainset
2-ND FOLD:
trainset
testset
trainset
3-RD FOLD:
trainset
testset
trainset
4-TH FOLD:
trainset
testset
trainset
5-TH FOLD:
trainset
testset