Section A01 CDS 101 *Introduction to Computational and Data Sciences*

Class 20: Modeling II

June 18, 2018

General

Annoucements

- Complete Reading 15 (last one) in advance of class on Tuesday, June 19th
- Homework 4 and extra credit Homework 5 due by **11:59pm on Wednesday, June 20th**
    - Homework 4 must be submitted before you can turn in Homework 5
- **Final project due dates**
    - **Annotations first draft**: 12:00pm noon on Thursday, June 21st
    - **Peer reviews**: 6:00pm on Thursday, June 21st
    - **Annotations and final draft**: 9:00am on Friday, June 22nd
    - **Comparative discussion of simulations**: 10:30am on Friday, June 22nd
- **Final interviews scheduled during final exam period**: Friday, June 22nd between 10:30am and 1:15pm

Linear models in the tidyverse

Last time...

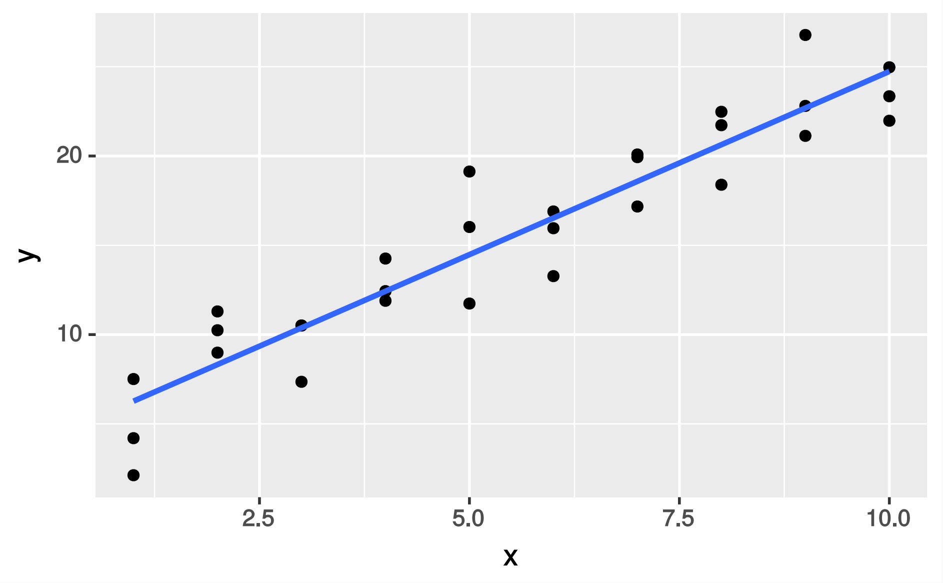
Last time...

- We used the `sim1` dataset loaded via `library(modelr)` and used `geom_smooth()` with `method = "lm"` to show what the linear model will look like

Last time...

- We used the `sim1` dataset loaded via `library(modelr)` and used `geom_smooth()` with `method = "lm"` to show what the linear model will look like

```
ggplot(sim1) +
  geom_point(mapping = aes(x = x, y = y)) +
  geom_smooth(mapping = aes(x = x, y = y), method = "lm", se = FALSE)
```



Using `lm()` to build linear models

```
sim1_mod <- lm(y ~ x, data = sim1)
```

Using `lm()` to build linear models

```
sim1_mod <- lm(y ~ x, data = sim1)
```

- `summary()` gives a general report about the model

```
summary(sim1_mod)
```

```
##
## Call:
## lm(formula = y ~ x, data = sim1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1469 -1.5197  0.1331  1.4670  4.6516
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.2208     0.8688   4.858 4.09e-05 ***
## x             2.0515     0.1400  14.651 1.17e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.203 on 28 degrees of freedom
## Multiple R-squared:  0.8846,    Adjusted R-squared:  0.8805
## F-statistic: 214.7 on 1 and 28 DF,  p-value: 1.173e-14
```

A tidier `lm()` summary

A tidier `lm()` summary

```r
library(broom)  # Installed alongside tidyverse
```

A tidier `lm()` summary

```r
library(broom)  # Installed alongside tidyverse
```

Get model parameters with `tidy()`

```r
sim1_mod <- lm(y ~ x, data = sim1)
```

```r
sim1_mod %>%
  tidy() %>%
  as_data_frame()
```

A tidier `lm()` summary

```r
library(broom)  # Installed alongside tidyverse
```

Get model parameters with `tidy()`

```r
sim1_mod <- lm(y ~ x, data = sim1)
```

```r
sim1_mod %>%
  tidy() %>%
  as_data_frame()
```

term	estimate	std.error	statistic	p.value
(Intercept)	4.220822	0.8688261	4.858074	4.09e-05
x	2.051533	0.1400240	14.651295	0.00e+00

A tidier lm() summary

```r
library(broom)  # Installed alongside tidyverse
```

Get model parameters with `tidy()`

```r
sim1_mod <- lm(y ~ x, data = sim1)
```

```r
sim1_mod %>%
  tidy() %>%
  as_data_frame()
```

term	estimate	std.error	statistic	p.value
(Intercept)	4.220822	0.8688261	4.858074	4.09e-05
x	2.051533	0.1400240	14.651295	0.00e+00

Get additional model details with `glance()`

```r
sim1_mod %>%
  glance() %>%
  as_data_frame()  # broom doesn't output tibbles by default
```

A tidier `lm()` summary

```r
library(broom)  # Installed alongside tidyverse
```

Get model parameters with `tidy()`

```r
sim1_mod <- lm(y ~ x, data = sim1)
```

```r
sim1_mod %>%
  tidy() %>%
  as_data_frame()
```

term	estimate	std.error	statistic	p.value
(Intercept)	4.220822	0.8688261	4.858074	4.09e-05
x	2.051533	0.1400240	14.651295	0.00e+00

Get additional model details with `glance()`

```r
sim1_mod %>%
  glance() %>%
  as_data_frame()  # broom doesn't output tibbles by default
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.8846124	0.8804914	2.202876	214.6604	0	2	-65.22618	136.4524	140.656	135.8746	28

Method for plotting our model

- The following is a basic recipe for visualizing our models

Method for plotting our model

- The following is a basic recipe for visualizing our models
- Create a series of `x` values with `data_grid()`:

```
grid <- data_grid(sim1, x)
```

Method for plotting our model

- The following is a basic recipe for visualizing our models
- Create a series of `x` values with `data_grid()`:

```
grid <- data_grid(sim1, x)
```

x
1
2
3
4
5
6

Extract predictions and residuals

Extract predictions and residuals

- Use `add_predictions()` to import predictions into your tibble

Extract predictions and residuals

- Use `add_predictions()` to import predictions into your tibble

```
grid2 <- add_predictions(grid, sim1_mod)
```

Extract predictions and residuals

- Use `add_predictions()` to import predictions into your tibble

```
grid2 <- add_predictions(grid, sim1_mod)
```

- Use `add_residuals()` to extract the residuals from your fit.

Extract predictions and residuals

- Use `add_predictions()` to import predictions into your tibble

```
grid2 <- add_predictions(grid, sim1_mod)
```

- Use `add_residuals()` to extract the residuals from your fit.

```
sim1_resid <- add_residuals(sim1, sim1_mod)
```

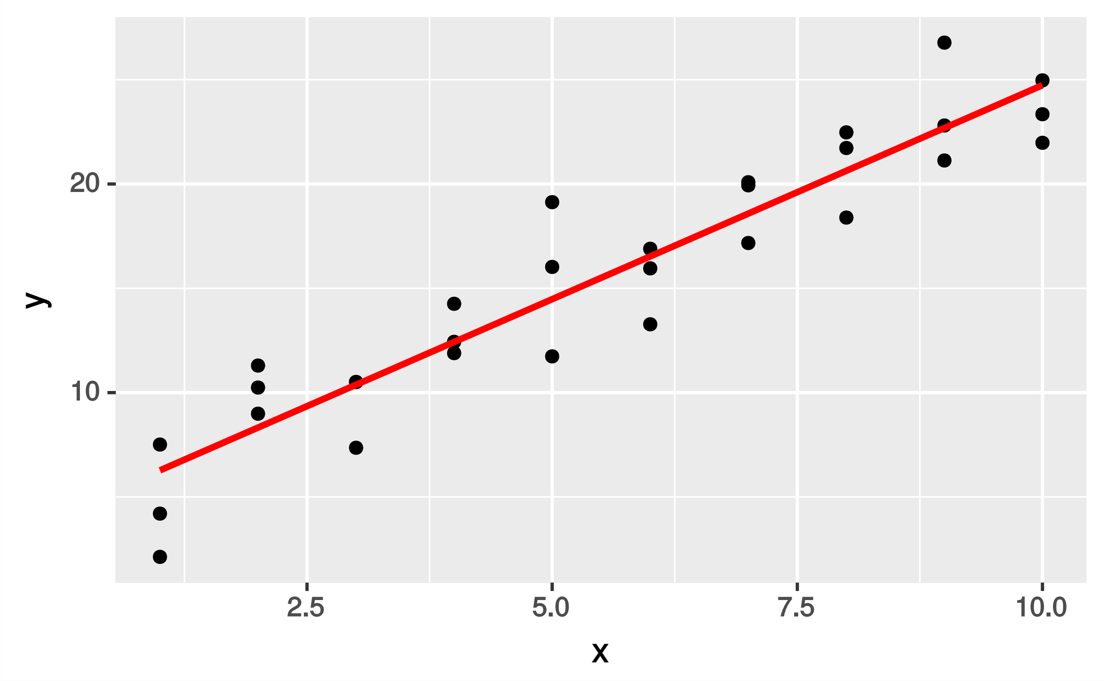
Visualize the full model

- Create a plot:

Visualize the full model

- Create a plot:

```
ggplot(sim1) +
  geom_point(aes(x = x, y = y)) +
  geom_line(aes(x = x, y = pred), data = grid2, color = "red", size = 1)
```



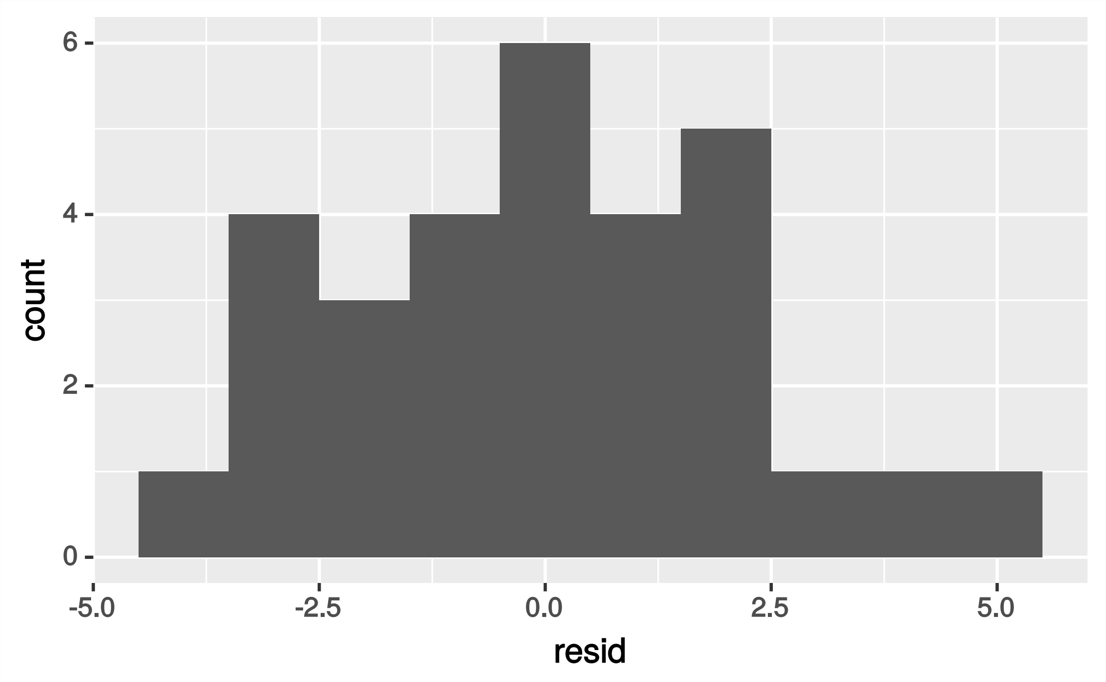
Inspect the residuals

- Use `geom_histogram()` to inspect the absolute residuals.

Inspect the residuals

- Use `geom_histogram()` to inspect the absolute residuals.

```
ggplot(sim1_resid) +
  geom_histogram(aes(x = resid), binwidth = 1)
```



Are the residuals normal?

- The residuals should be nearly normal.

Are the residuals normal?

- The residuals should be nearly normal.
- A good test for normal residuals is a Q-Q plot:

Are the residuals normal?

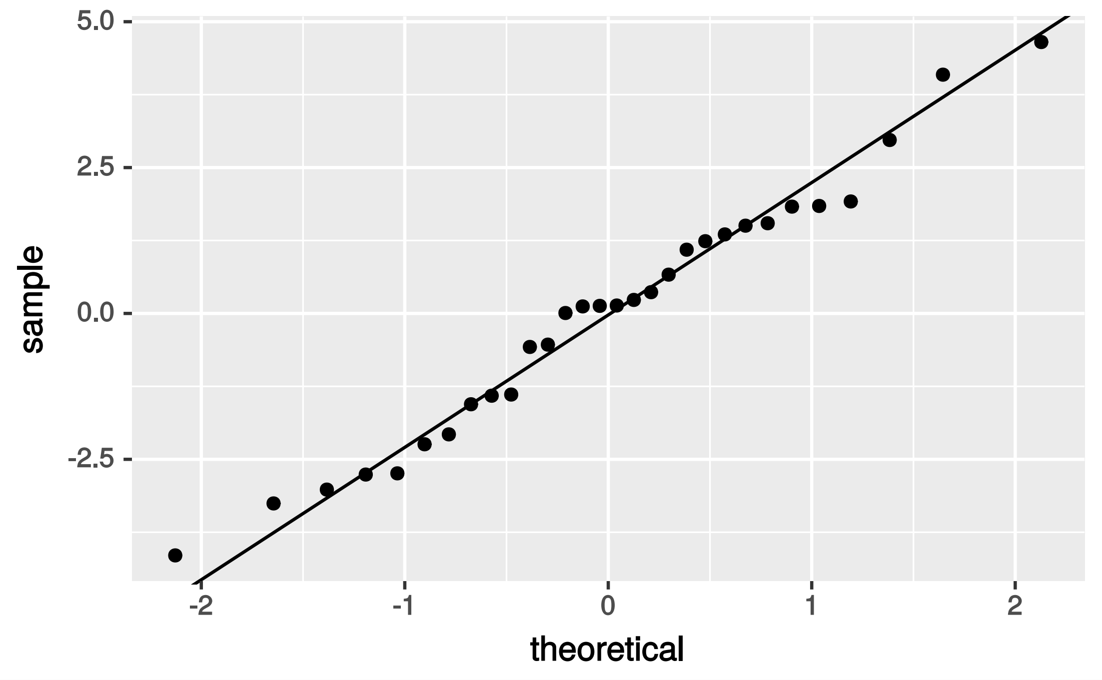
- The residuals should be nearly normal.

- A good test for normal residuals is a Q-Q plot:

```
qq_x <- qnorm(p = c(0.25, 0.75))
qq_y <- quantile(x = pull(sim1_resid, resid), probs = c(0.25, 0.75), type = 1)
qq_slope <- diff(qq_y) / diff(qq_x)
qq_int <- pluck(qq_y, 1) - qq_slope * pluck(qq_x, 1)
ggplot(sim1_resid) +
  geom_qq(aes(sample = resid)) +
  geom_abline(intercept = qq_int, slope = qq_slope)
```

Are the residuals normal?

- The residuals should be nearly normal.
- A good test for normal residuals is a Q-Q plot:



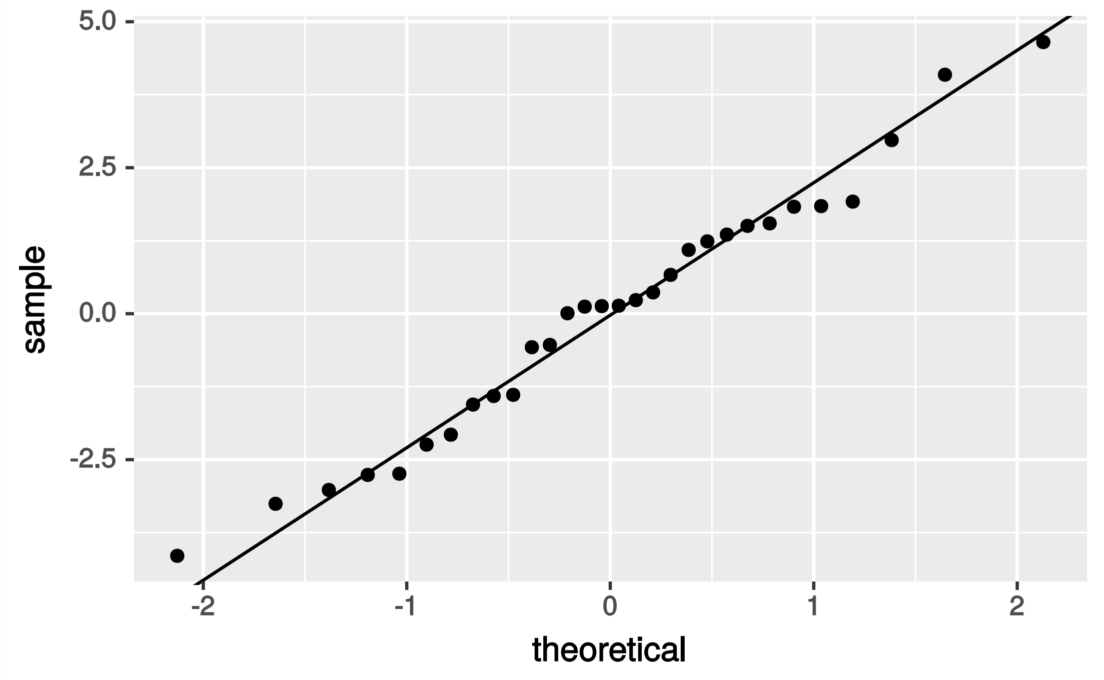
Aside: Create function for plotting reference line

```
geom_qq_ref_line <- function(data, variable) {
  qq_x <- qnorm(p = c(0.25, 0.75))
  qq_y <- quantile(
    x = pull(data, variable),
    probs = c(0.25, 0.75),
    type = 1
  )
  qq_slope <- diff(qq_y) / diff(qq_x)
  qq_int <- pluck(qq_y, 1) - qq_slope * pluck(qq_x, 1)

  geom_abline(intercept = qq_int, slope = qq_slope)
}
```

Aside: Create function for plotting reference line

```
ggplot(sim1_resid) +
  geom_qq(aes(sample = resid)) +
  geom_qq_ref_line(data = sim1_resid, variable = "resid")
```



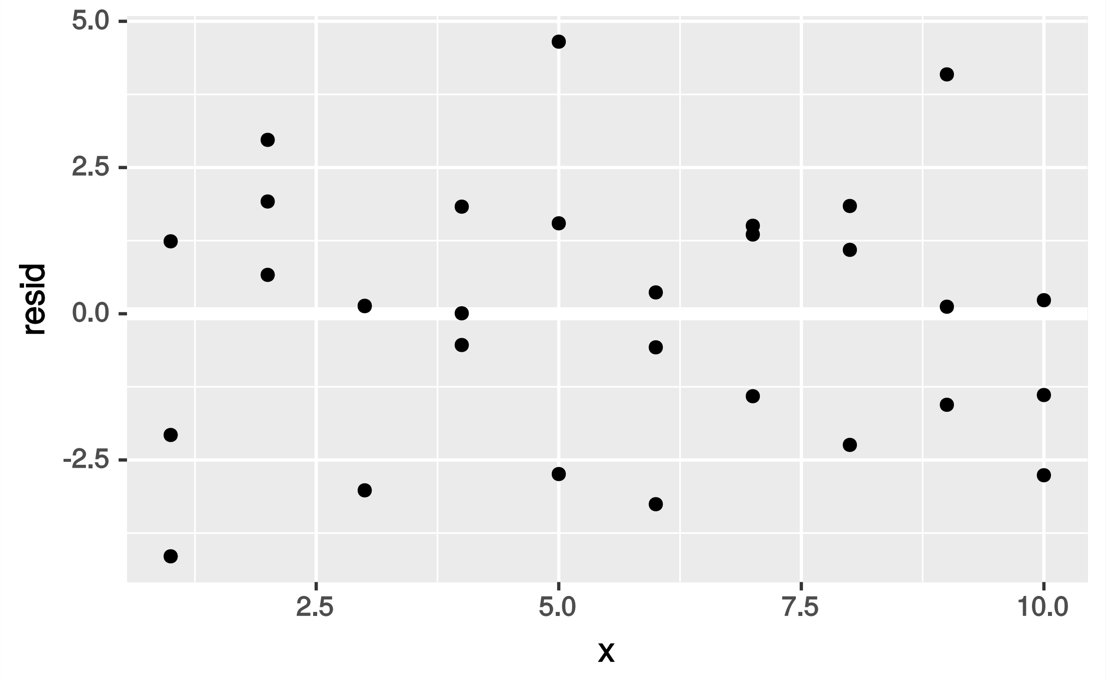
Residual spread

- Inspect the residual spread as a function of `x` to check whether the variability is constant or not:

Residual spread

- Inspect the residual spread as a function of `x` to check whether the variability is constant or not:

```
ggplot(sim1_resid) +
  geom_ref_line(h = 0) +
  geom_point(aes(x = x, y = resid))
```



Case study: Mario Kart eBay prices dataset

Machine Learning and prediction

Machine Learning and prediction

- Machine Learning models are built with the purpose of making predictions

Machine Learning and prediction

- Machine Learning models are built with the purpose of making predictions
- The model is "trained" on a dataset and "learns" how to reproduce the general structure and features in that dataset

Machine Learning and prediction

- Machine Learning models are built with the purpose of making predictions
- The model is "trained" on a dataset and "learns" how to reproduce the general structure and features in that dataset
- In the best case scenario, you get a model with strong predictive powers that can take a series of inputs and generate a highly accurate output

Machine Learning and prediction

- Machine Learning models are built with the purpose of making predictions
- The model is "trained" on a dataset and "learns" how to reproduce the general structure and features in that dataset
- In the best case scenario, you get a model with strong predictive powers that can take a series of inputs and generate a highly accurate output
- Generally only interested in accuracy, not understanding, making **prediction** distinct from **inference**

Machine Learning and prediction

- Machine Learning models are built with the purpose of making predictions
- The model is "trained" on a dataset and "learns" how to reproduce the general structure and features in that dataset
- In the best case scenario, you get a model with strong predictive powers that can take a series of inputs and generate a highly accurate output
- Generally only interested in accuracy, not understanding, making **prediction** distinct from **inference**
- This accuracy comes at a price, as the most accurate prediction models are frequently the most complicated

Machine Learning and prediction

- Machine Learning models are built with the purpose of making predictions
- The model is "trained" on a dataset and "learns" how to reproduce the general structure and features in that dataset
- In the best case scenario, you get a model with strong predictive powers that can take a series of inputs and generate a highly accurate output
- Generally only interested in accuracy, not understanding, making **prediction** distinct from **inference**
- This accuracy comes at a price, as the most accurate prediction models are frequently the most complicated
- This is what people mean when they say that Machine Learning algorithms are like a "black box"

Can we predict accurately eBay prices?

- Data scraped from eBay listings for the video game *Mario Kart Wii*

Image: *Mario Kart Wii* cover art, ©Nintendo, downloaded from Wikipedia, https://en.wikipedia.org/wiki/File:Mario_Kart_Wii.png

Can we predict accurately eBay prices?

- Data scraped from eBay listings for the video game *Mario Kart Wii*
- Can we predict each game's final selling price using other information on a eBay listing page?

Image: *Mario Kart Wii* cover art, ©Nintendo, downloaded from Wikipedia, https://en.wikipedia.org/wiki/File:Mario_Kart_Wii.png

Can we predict accurately eBay prices?

- Data scraped from eBay listings for the video game *Mario Kart Wii*
- Can we predict each game's final selling price using other information on a eBay listing page?

Goal

Build a model that predicts the dataset variable `totalPr` using the other columns

Image: *Mario Kart Wii* cover art, ©Nintendo, downloaded from Wikipedia, https://en.wikipedia.org/wiki/File:Mario_Kart_Wii.png

Data exploration

What's in this dataset?

What's in this dataset?

- What are the first several entries of the *Mario Kart* dataset?

What's in this dataset?

- What are the first several entries of the *Mario Kart* dataset?

```
mariokart %>%
  glimpse()
```

What's in this dataset?

- What are the first several entries of the *Mario Kart* dataset?

```
mariokart %>%
  glimpse()
```

```
## Observations: 143
## Variables: 12
## $ ID         <dbl> 150377422259, 260483376854, 320432342985, 280405224...
## $ duration   <int> 3, 7, 3, 3, 1, 3, 1, 1, 3, 7, 1, 1, 1, 1, 7, 7, 3, ...
## $ nBids      <int> 20, 13, 16, 18, 20, 19, 13, 15, 29, 8, 15, 15, 13, ...
## $ cond       <fct> new, used, new, new, new, new, used, new, used, use...
## $ startPr    <dbl> 0.99, 0.99, 0.99, 0.99, 0.01, 0.99, 0.01, 1.00, 0.9...
## $ shipPr     <dbl> 4.00, 3.99, 3.50, 0.00, 0.00, 4.00, 0.00, 2.99, 4.0...
## $ totalPr    <dbl> 51.55, 37.04, 45.50, 44.00, 71.00, 45.00, 37.02, 53...
## $ shipSp     <fct> standard, firstClass, firstClass, standard, media, ...
## $ sellerRate <int> 1580, 365, 998, 7, 820, 270144, 7284, 4858, 27, 201...
## $ stockPhoto <fct> yes, yes, no, yes, yes, yes, yes, yes, yes, no, yes...
## $ wheels     <int> 1, 1, 1, 1, 2, 0, 0, 2, 1, 1, 2, 2, 2, 2, 1, 0, 1, ...
## $ title      <fct> ~~ Wii MARIO KART &amp; WHEEL ~ NINTENDO Wii ~ BRAN...
```

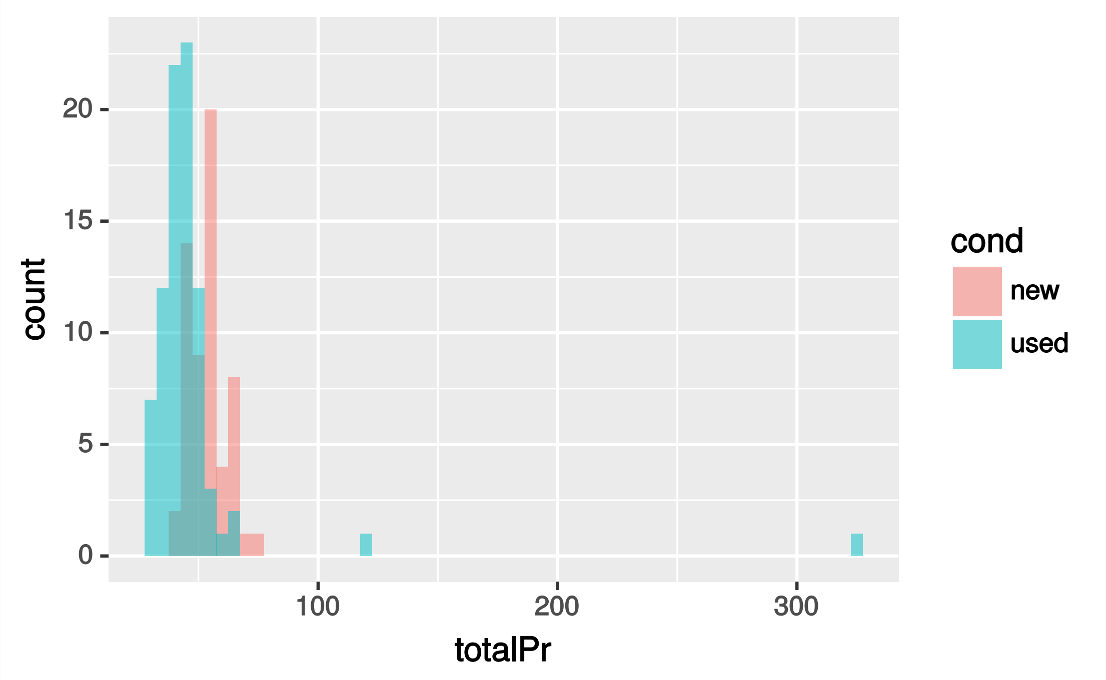
Exploring the response variable

- What is the shape and center of the response variable `totalPr`?

Exploring the response variable

- What is the shape and center of the response variable `totalPr`?

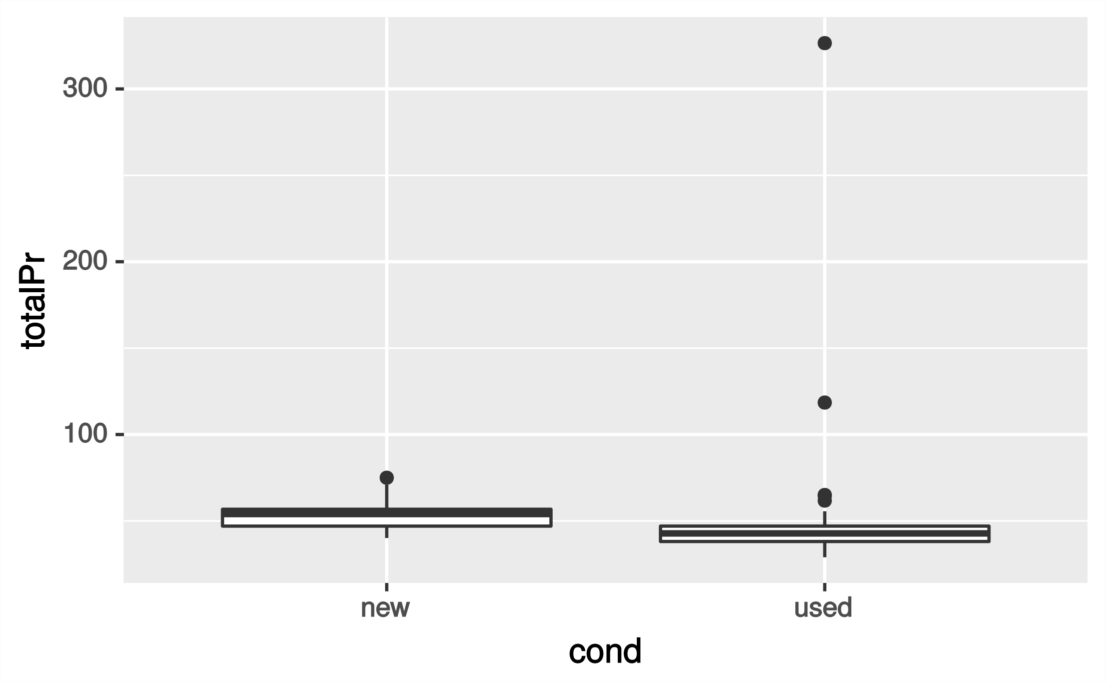
```
ggplot(mariokart) +
  geom_histogram(
    mapping = aes(x = totalPr, fill = cond),
    position = "identity", alpha = 0.5, binwidth = 5, center = 0
  )
```



Exploring the response variable

- A box plot is nice to use for exploration as well

```
ggplot(mariokart) +
  geom_boxplot(mapping = aes(x = cond, y = totalPr))
```



Find the outliers

Find the outliers

- What are the outliers?

Find the outliers

- What are the outliers?
- Filter the dataset to isolate them

Find the outliers

- What are the outliers?
- Filter the dataset to isolate them

```
mariokart %>%
  filter(totalPr > 100) %>%
  glimpse()
```

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

Inspect outlier characteristics

Inspect outlier characteristics

- Look at the listing titles

Inspect outlier characteristics

- Look at the listing titles

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

title
Nintedo Wii Console Bundle Guitar Hero 5 Mario Kart
10 Nintendo Wii Games - MarioKart Wii, SpiderMan 3, etc

Inspect outlier characteristics

- Look at the listing titles

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

title
Nintedo Wii Console Bundle Guitar Hero 5 Mario Kart
10 Nintendo Wii Games - MarioKart Wii, SpiderMan 3, etc

- These are bundled items, not like the rest of the items in the dataset.

Inspect outlier characteristics

- Look at the listing titles

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

title
Nintedo Wii Console Bundle Guitar Hero 5 Mario Kart
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 the outliers

Inspect outlier characteristics

- Look at the listing titles

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

title
Nintedo Wii Console Bundle Guitar Hero 5 Mario Kart
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 the outliers
- For simplicity, we will also restrict ourselves to a subset of variables: `cond`, `stockPhoto`, `duration`, and `wheels`

Removing outliers

```
mariokart2 <- mariokart %>%
  filter(totalPr <= 100) %>%
  select(totalPr, cond, stockPhoto, duration, wheels)
```

Removing outliers

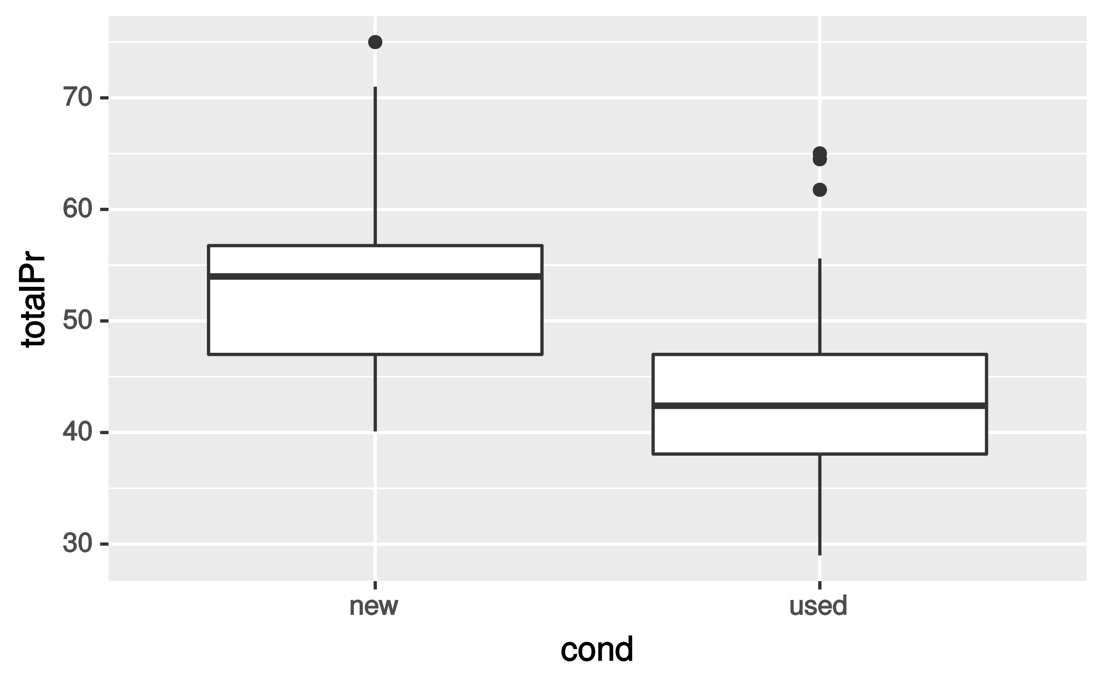
```
mariokart2 <- mariokart %>%
  filter(totalPr <= 100) %>%
  select(totalPr, cond, stockPhoto, duration, wheels)
```

- Let's check the box plot again, this time with no outliers

Removing outliers

```
mariokart2 <- mariokart %>%
  filter(totalPr <= 100) %>%
  select(totalPr, cond, stockPhoto, duration, wheels)
```

- Let's check the box plot again, this time with no outliers



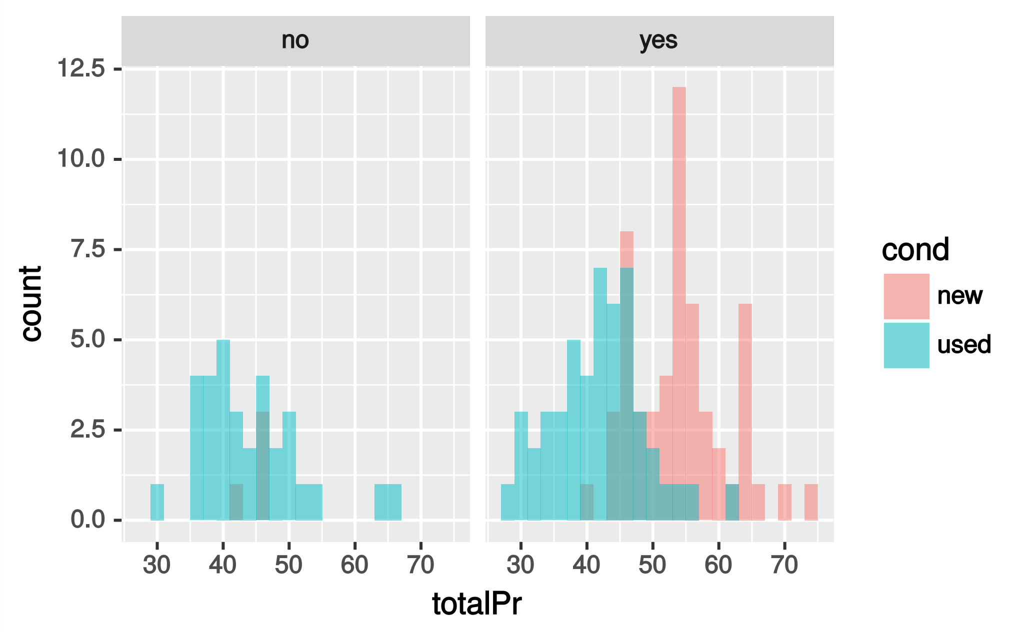
Looking for trends

- Continue exploring the dataset to find trends: does game condition and using a stock photo affect the total price?

Looking for trends

- Continue exploring the dataset to find trends: does game condition and using a stock photo affect the total price?

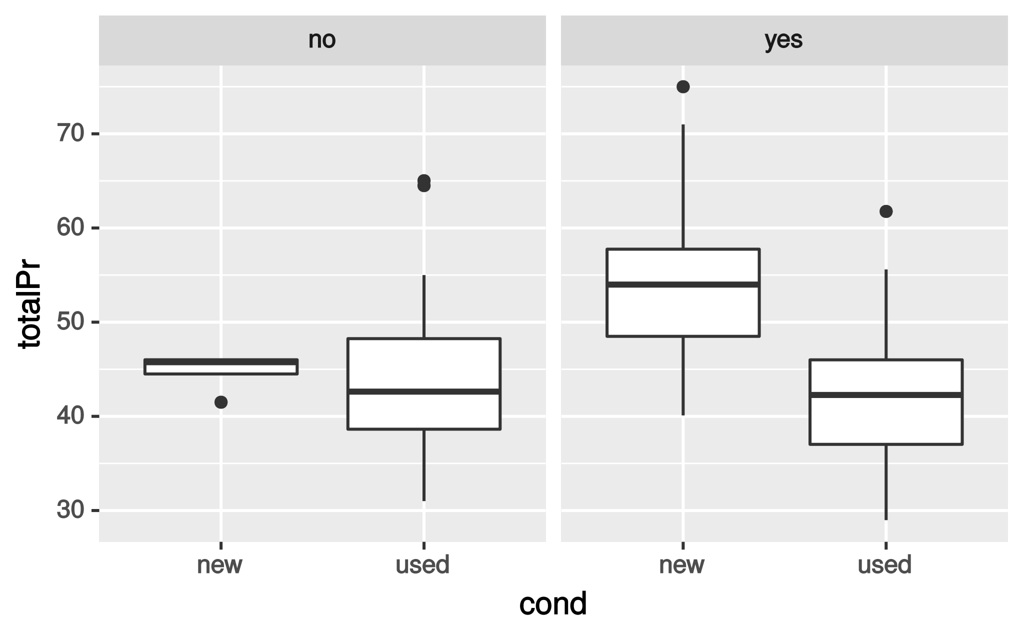
```
ggplot(mariokart2) +
  geom_histogram(
    mapping = aes(totalPr, fill = cond), position = "identity",
    alpha = 0.5, center = 0, binwidth = 2
  ) +
  facet_wrap(~stockPhoto)
```



Looking for trends

- A box plot would also be an appropriate way to show this data:

```
ggplot(mariokart2) +
  geom_boxplot(mapping = aes(x = cond, y = totalPr)) +
  facet_wrap(~stockPhoto)
```



Data distribution of `totalPr`

Data distribution of `totalPr`

- Is `totalPr` nearly normal?

Data distribution of `totalPr`

- Is `totalPr` nearly normal?
- How does the distribution shape change within categories?

Data distribution of `totalPr`

- Is `totalPr` nearly normal?
- How does the distribution shape change within categories?
- Use Q-Q plot to check `totalPr` by itself:

Data distribution of `totalPr`

- Is `totalPr` nearly normal?
- How does the distribution shape change within categories?
- Use Q-Q plot to check `totalPr` by itself:

```
ggplot(mariokart2) +
  geom_qq(mapping = aes(sample = totalPr)) +
  geom_qq_ref_line(data = mariokart2, variable = "totalPr")
```

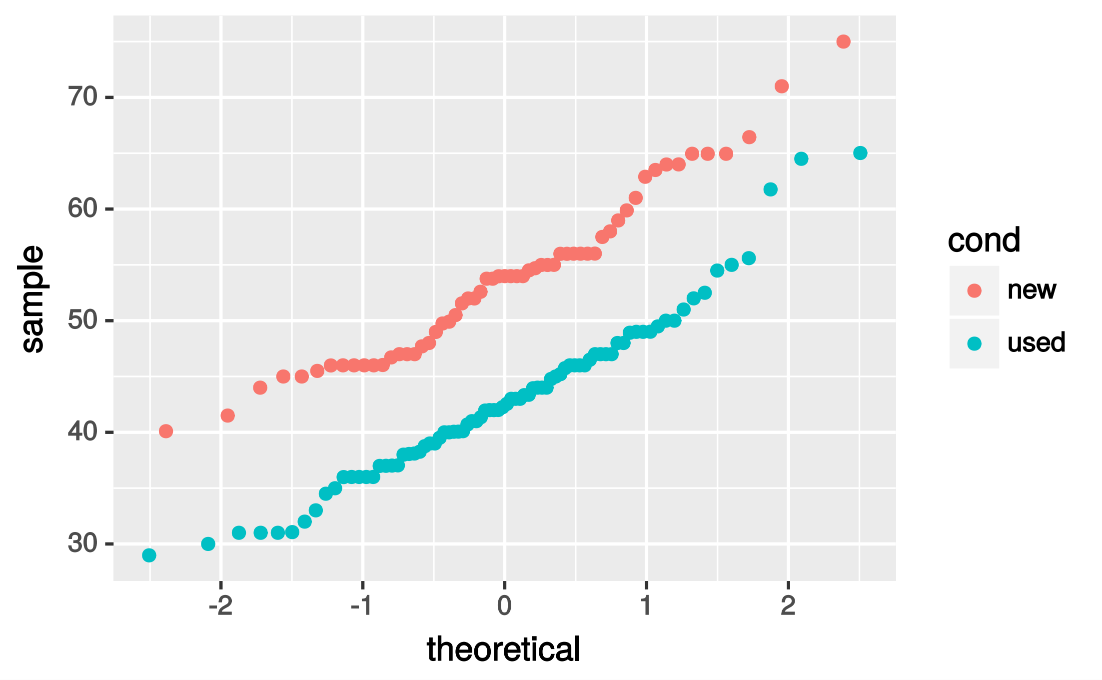
`totalPr` distribution within groups

- Q-Q plot with `totalPr` split by game condition:

`totalPr` distribution within groups

- Q-Q plot with `totalPr` split by game condition:

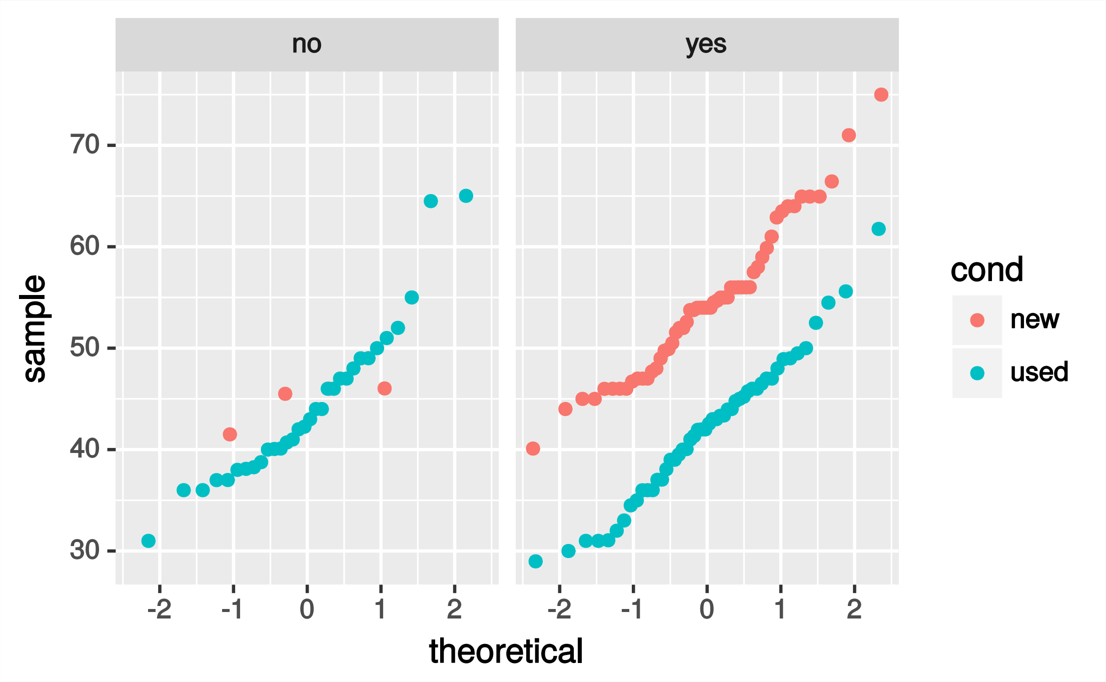
```
ggplot(mariokart2) +
  geom_qq(mapping = aes(sample = totalPr, color = cond))
```



totalPr distribution within groups

- Q-Q plot with `totalPr` split by game condition and faceted by `stockPhoto`:

```
ggplot(mariokart2) +
  geom_qq(mapping = aes(sample = totalPr, color = cond)) +
  facet_wrap( ~ stockPhoto)
```



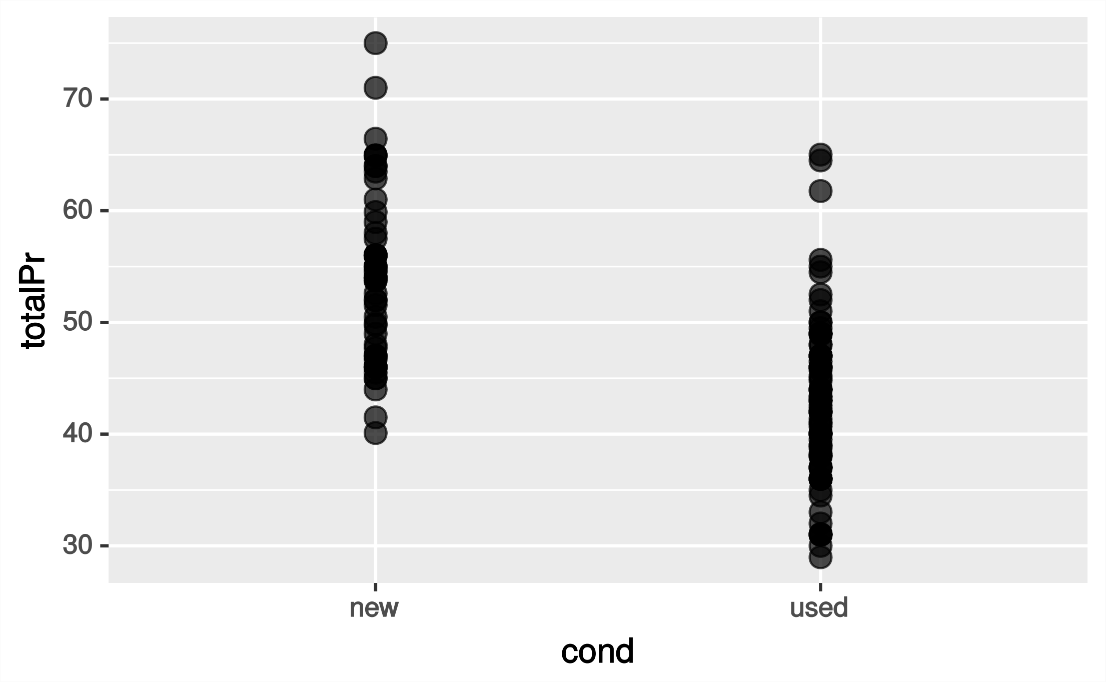
Categorical variables in scatterplots

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

Categorical variables in scatterplots

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

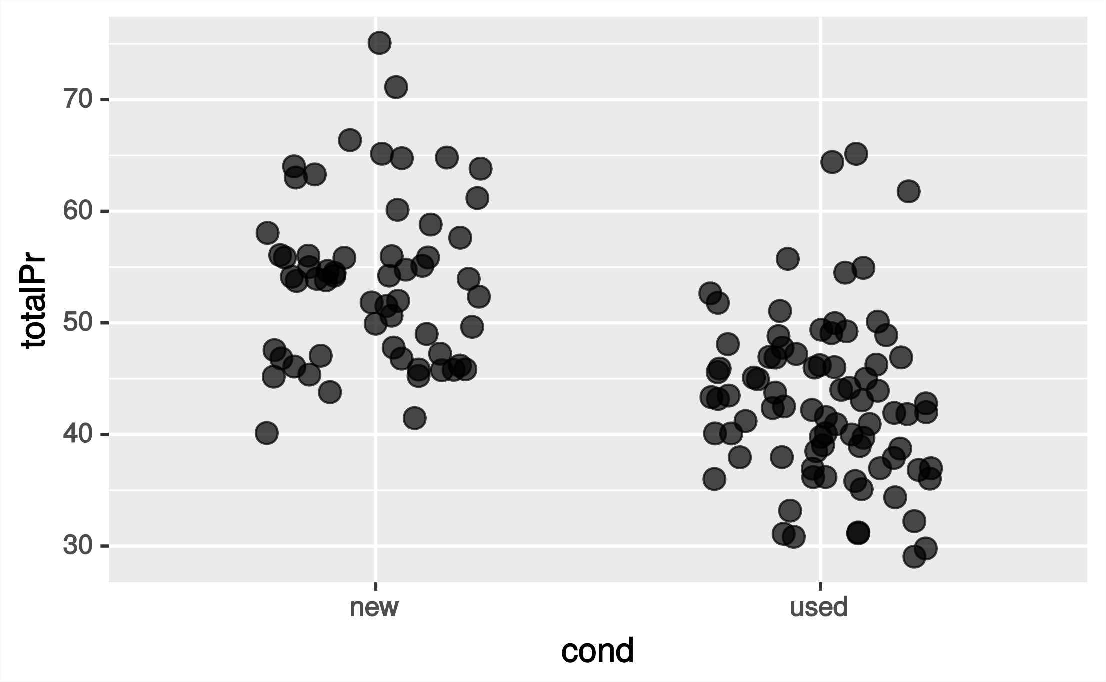
```
ggplot(mariokart2) +
  geom_point(mapping = aes(cond, totalPr), size = 3, alpha = 0.7)
```



Categorical variables in scatterplots

- It's easier to see the points if we jitter them

```
ggplot(mariokart2) +
  geom_jitter(
    mapping = aes(cond, totalPr), size = 3, alpha = 0.7, width = 0.25,
    height = 0.25)
```



Training and testing datasets

Split dataset 80/20

Split dataset 80/20

- Frequently, it's good practice to split a dataset prior to testing a model.

Split dataset 80/20

- Frequently, it's good practice to split a dataset prior to testing a model.

- The following code splits the data into two partitions

Split dataset 80/20

- Frequently, it's good practice to split a dataset prior to testing a model.
- The following code splits the data into two partitions

```
mariokart_with_ids <- mariokart2 %>%
 bind_cols(id = 1:nrow(mariokart2))

train <- mariokart_with_ids %>%
  sample_frac(size = 0.80, replace = FALSE)

test  <- mariokart_with_ids %>%
  anti_join(train, by = 'id')
```

Split dataset 80/20

- Frequently, it's good practice to split a dataset prior to testing a model.
- The following code splits the data into two partitions

```
mariokart_with_ids <- mariokart2 %>%
 bind_cols(id = 1:nrow(mariokart2))

train <- mariokart_with_ids %>%
  sample_frac(size = 0.80, replace = FALSE)

test  <- mariokart_with_ids %>%
  anti_join(train, by = 'id')
```

- 80% is randomly selected and placed in the training dataset

Split dataset 80/20

- Frequently, it's good practice to split a dataset prior to testing a model.
- The following code splits the data into two partitions

```
mariokart_with_ids <- mariokart2 %>%
 bind_cols(id = 1:nrow(mariokart2))

train <- mariokart_with_ids %>%
  sample_frac(size = 0.80, replace = FALSE)

test  <- mariokart_with_ids %>%
  anti_join(train, by = 'id')
```

- 80% is randomly selected and placed in the training dataset
- Remaining 20% is used for the testing dataset

Split dataset 80/20

- Frequently, it's good practice to split a dataset prior to testing a model.
- The following code splits the data into two partitions

```
mariokart_with_ids <- mariokart2 %>%
 bind_cols(id = 1:nrow(mariokart2))

train <- mariokart_with_ids %>%
  sample_frac(size = 0.80, replace = FALSE)

test  <- mariokart_with_ids %>%
  anti_join(train, by = 'id')
```

- 80% is randomly selected and placed in the training dataset
- Remaining 20% is used for the testing dataset
- All subsequent model building will be done using the `train` dataset

Univariate linear regression models

Predict using game condition

- Let's start with a refresher on creating a univariate linear model using `lm()`

Predict using game condition

- Let's start with a refresher on creating a univariate linear model using `lm()`
- Build a model that uses the `cond` categorical variable to predict the total price `totalPr`

Predict using game condition

- Let's start with a refresher on creating a univariate linear model using `lm()`
- Build a model that uses the `cond` categorical variable to predict the total price `totalPr`

```
mariokart_cond_model_lm <- lm(totalPr ~ cond, data = train)
```

Predict using game condition

- Let's start with a refresher on creating a univariate linear model using `lm()`
- Build a model that uses the `cond` categorical variable to predict the total price `totalPr`

```
mariokart_cond_model_lm <- lm(totalPr ~ cond, data = train)
```

- Predict training dataset and compute the residuals

Predict using game condition

- Let's start with a refresher on creating a univariate linear model using `lm()`
- Build a model that uses the `cond` categorical variable to predict the total price `totalPr`

```
mariokart_cond_model_lm <- lm(totalPr ~ cond, data = train)
```

- Predict training dataset and compute the residuals

```
mariokart_cond_model_df <- train %>%
  add_predictions(mariokart_cond_model_lm) %>%
  add_residuals(mariokart_cond_model_lm)
```

Summary of our fit

Print out some basic details about the linear fit:

Summary of our fit

Print out some basic details about the linear fit:

```
summary(mariokart_cond_model_lm)
```

```
##
## Call:
## lm(formula = totalPr ~ cond, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2187  -5.7760  -0.1987   3.8013  21.8213
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   53.476      1.061  50.423  < 2e-16 ***
## condused     -10.277      1.420  -7.236 6.37e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.499 on 111 degrees of freedom
## Multiple R-squared:  0.3205,    Adjusted R-squared:  0.3144
## F-statistic: 52.36 on 1 and 111 DF,  p-value: 6.369e-11
```

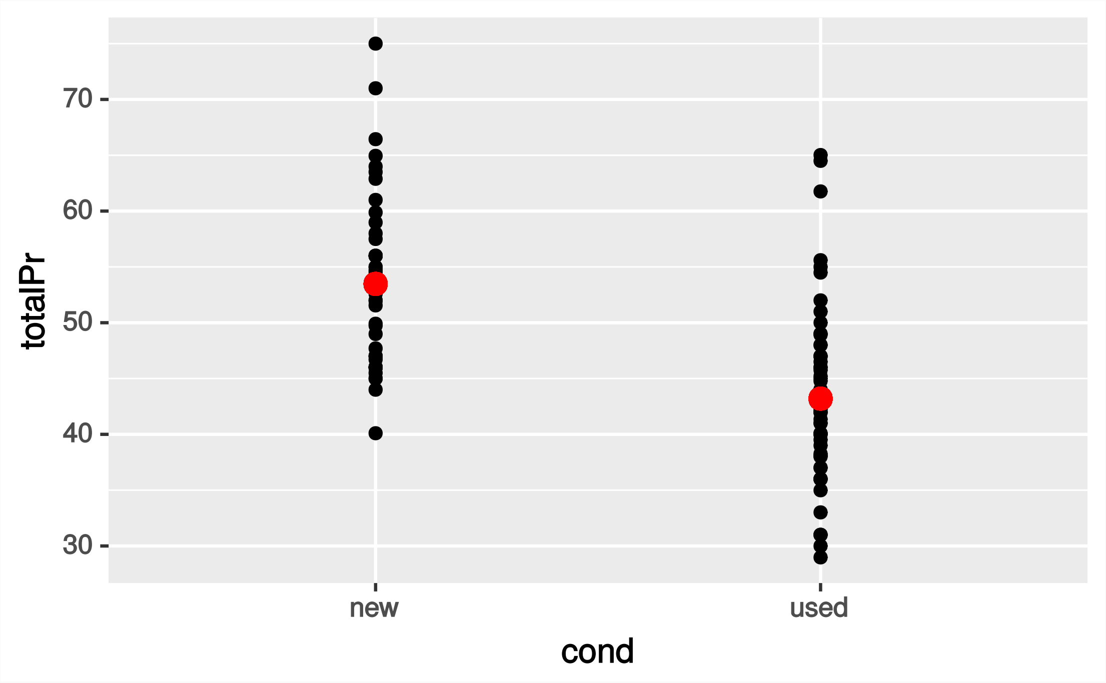
Visualize the model

- Since `cond` is categorical, what will it look like when we overlay our models' predictions on the data?

Visualize the model

- Since `cond` is categorical, what will it look like when we overlay our models' predictions on the data?

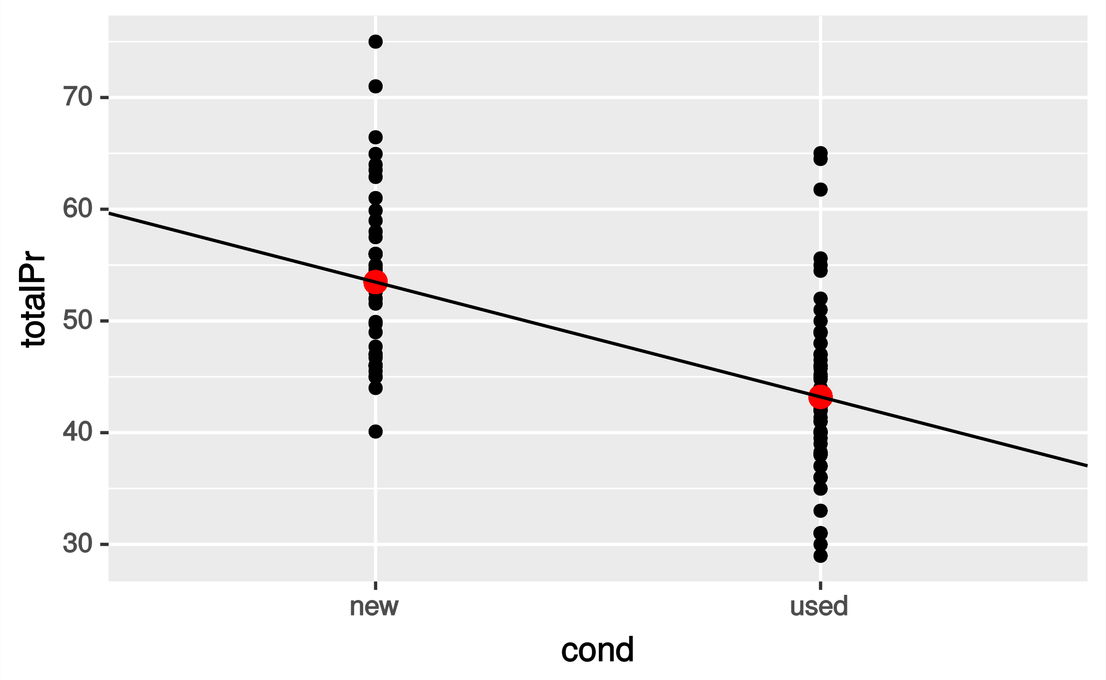
```
ggplot(mariokart_cond_model_df) +
  geom_point(mapping = aes(x = cond, y = totalPr)) +
  geom_point(mapping = aes(x = cond, y = pred), color = "red", size = 3)
```



Visualize the model

- Since `cond` is categorical, what will it look like when we overlay our models' predictions on the data?

```
ggplot(mariokart_cond_model_df) +
  geom_point(mapping = aes(x = cond, y = totalPr)) +
  geom_point(mapping = aes(x = cond, y = pred), color = "red", size = 3)
```



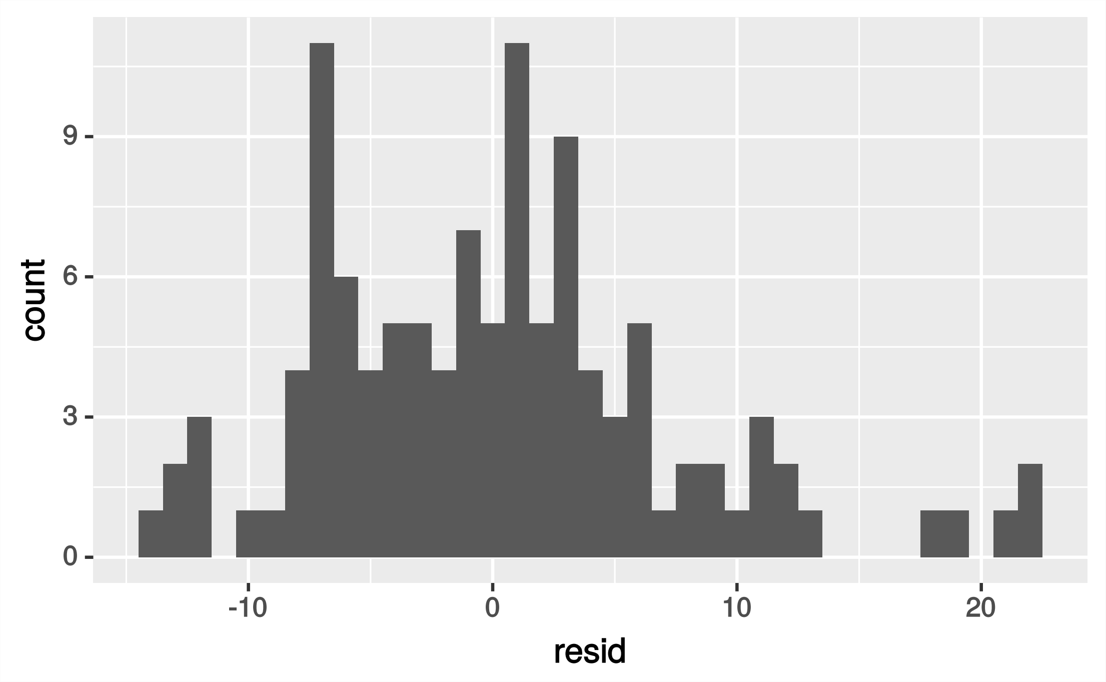
Inspect residuals

- Let's inspect the residuals:

Inspect residuals

- Let's inspect the residuals:

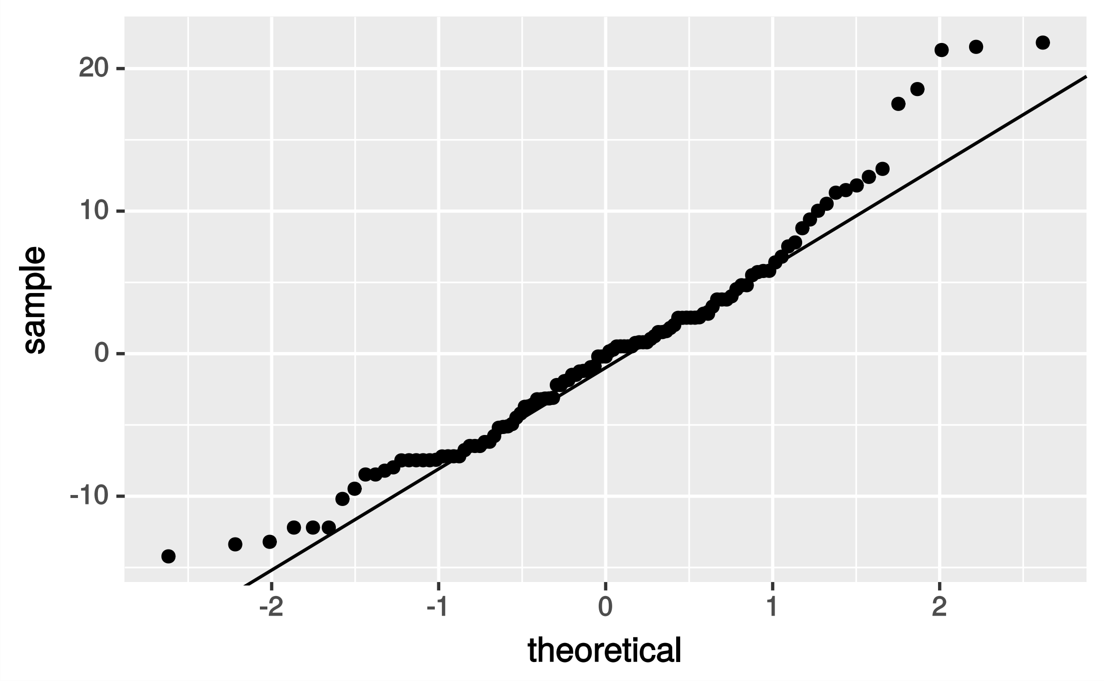
```
ggplot(mariokart_cond_model_df) +
  geom_histogram(mapping = aes(x = resid), binwidth = 1, center = 0)
```



Inspect residuals

- Let's inspect the residuals:

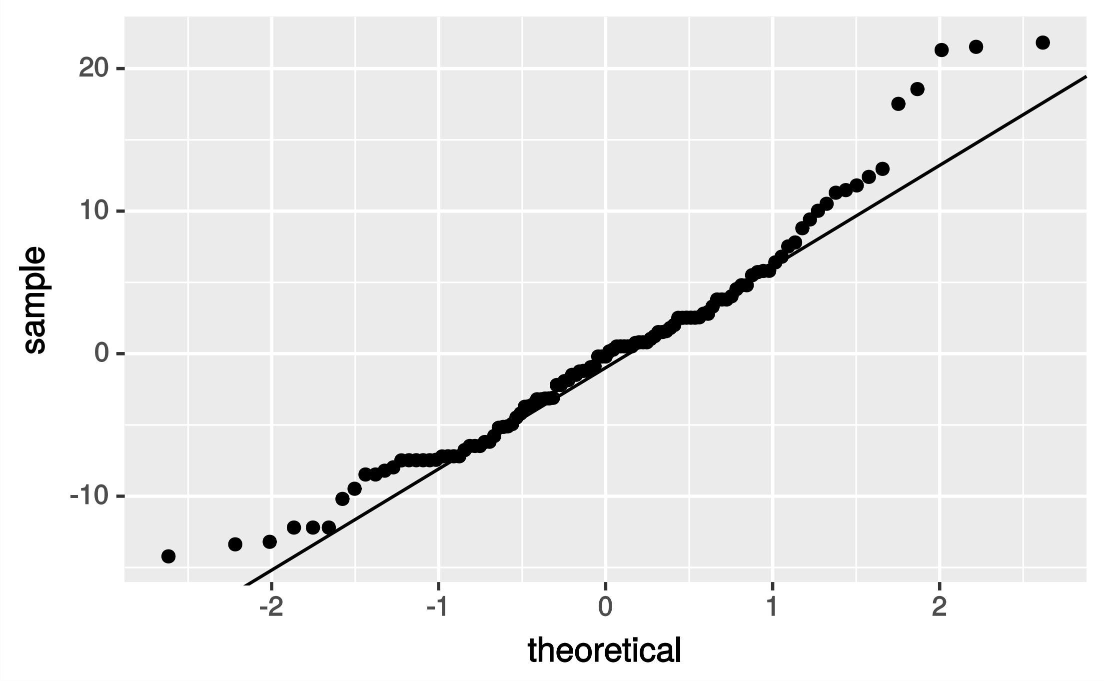
```
ggplot(mariokart_cond_model_df) +
  geom_qq(mapping = aes(sample = resid)) +
  geom_qq_ref_line(data = mariokart_cond_model_df, variable = "resid")
```



Inspect residuals

- Let's inspect the residuals:

```
ggplot(mariokart_cond_model_df) +
  geom_qq(mapping = aes(sample = resid)) +
  geom_qq_ref_line(data = mariokart_cond_model_df, variable = "resid")
```



- Deviations from normal distribution with long tail on the right

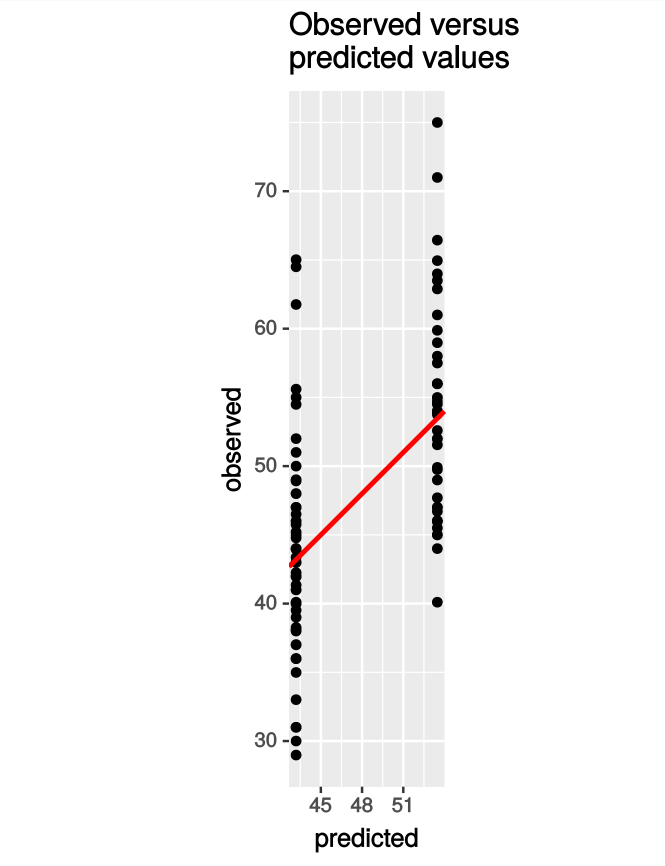
Observed values vs. predicted values

- Accurate prediction is our goal, so we should visualize how well the predictions match with the actual values

Observed values vs. predicted values

- Accurate prediction is our goal, so we should visualize how well the predictions match with the actual values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(totalPr, pred)) +
  geom_abline(
    slope = 1, intercept = 0,
    color = "red", size = 1)
```

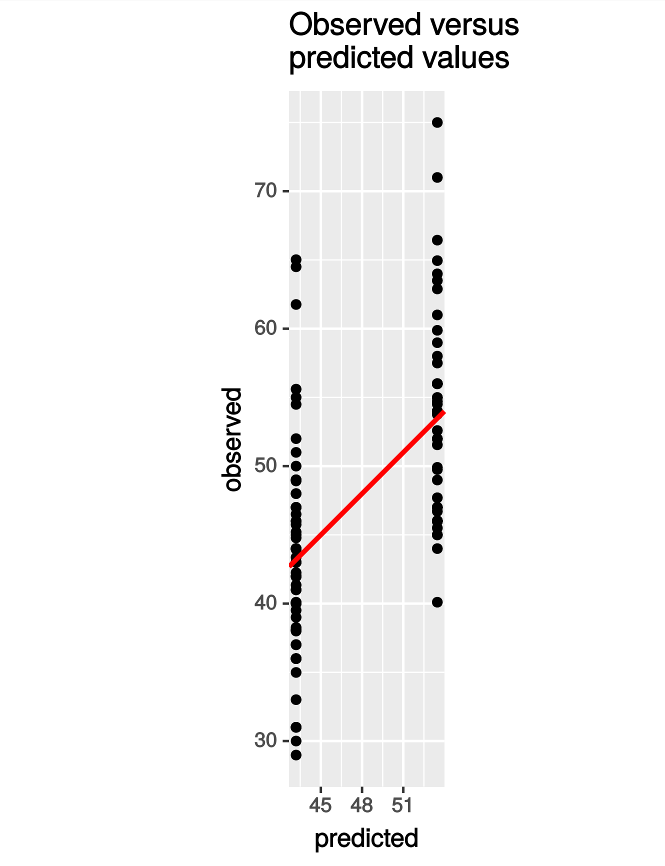


Observed values vs. predicted values

- Accurate prediction is our goal, so we should visualize how well the predictions match with the actual values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(totalPr, pred)) +
  geom_abline(
    slope = 1, intercept = 0,
    color = "red", size = 1)
```

- This is called an "observed versus predicted" plot†



† There isn't a precise name for this type of plot, so you may see this called an "actual versus predicted" plot or an "actual versus fitted" plot, or something else.

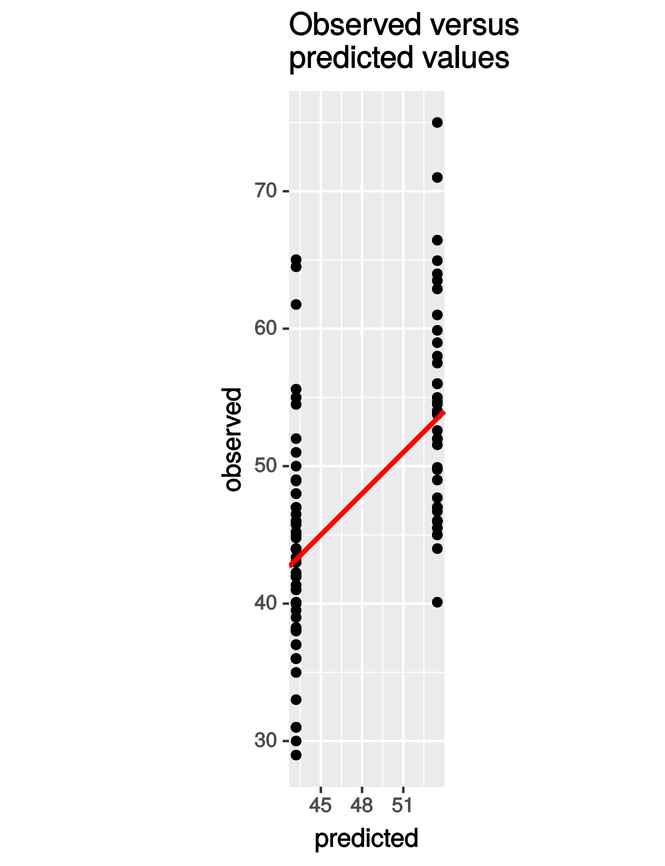
Observed values vs. predicted values

- Accurate prediction is our goal, so we should visualize how well the predictions match with the actual values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(totalPr, pred)) +
  geom_abline(
    slope = 1, intercept = 0,
    color = "red", size = 1)
```

- This is called an "observed versus predicted" plot†

- There's a residuals version of this, the "residual versus predicted" plot



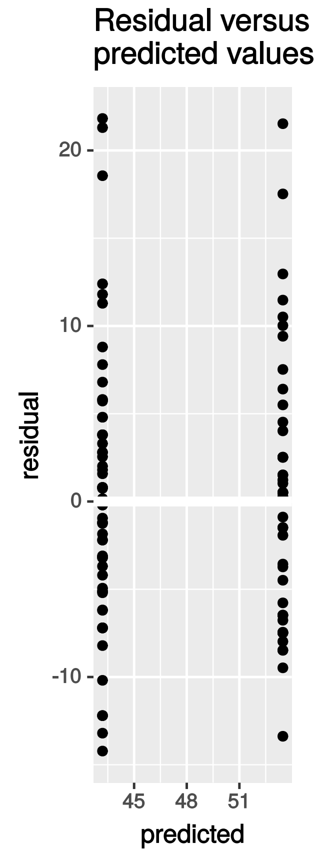
† There isn't a precise name for this type of plot, so you may see this called an "actual versus predicted" plot or an "actual versus fitted" plot, or something else.

Residual vs. predicted values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(pred, resid)) +
  geom_ref_line(h = 0)
```

Residual vs. predicted values

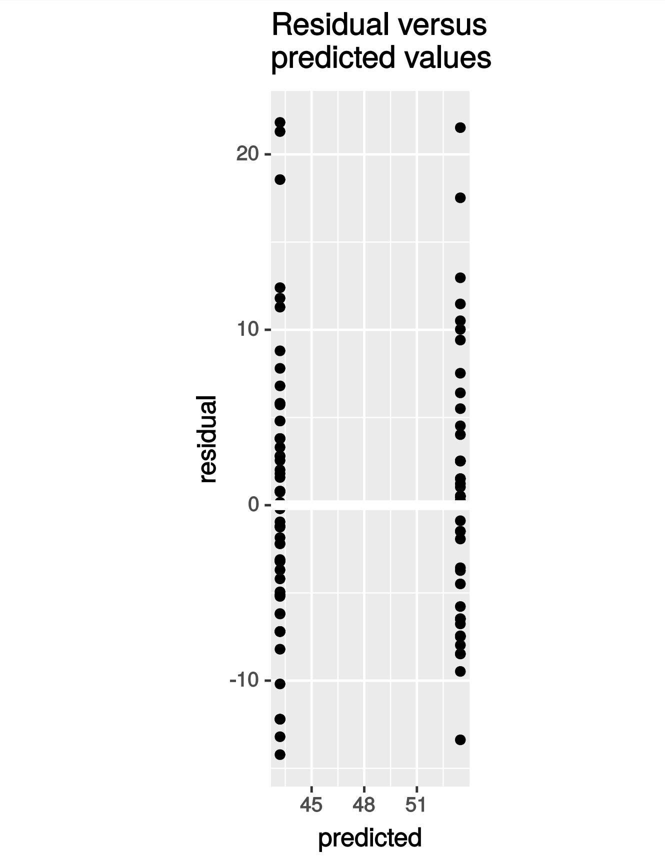
```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(pred, resid)) +
  geom_ref_line(h = 0)
```



Residual vs. predicted values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(pred, resid)) +
  geom_ref_line(h = 0)
```

- The residual spread stays consistent, so that's good

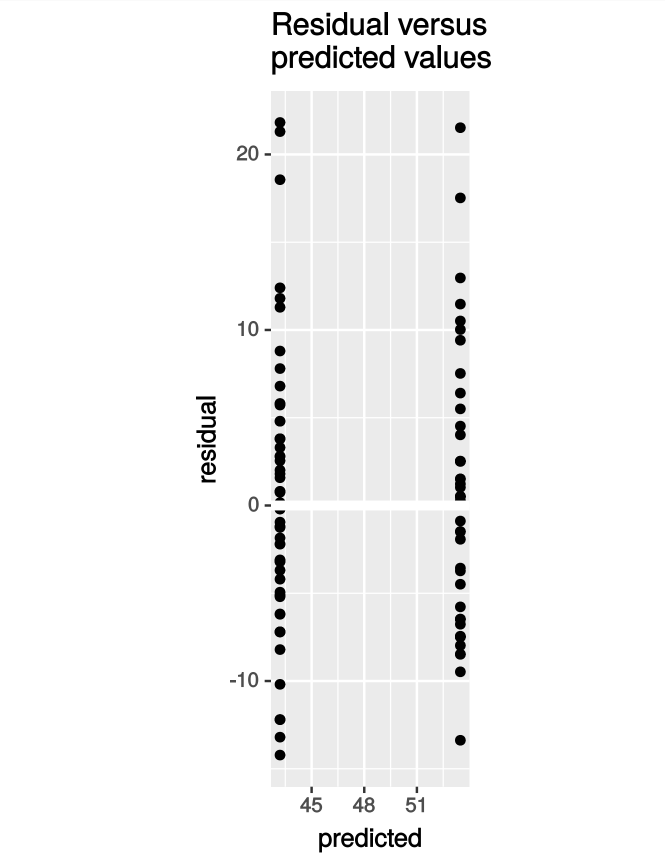


Residual vs. predicted values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(pred, resid)) +
  geom_ref_line(h = 0)
```

- The residual spread stays consistent, so that's good

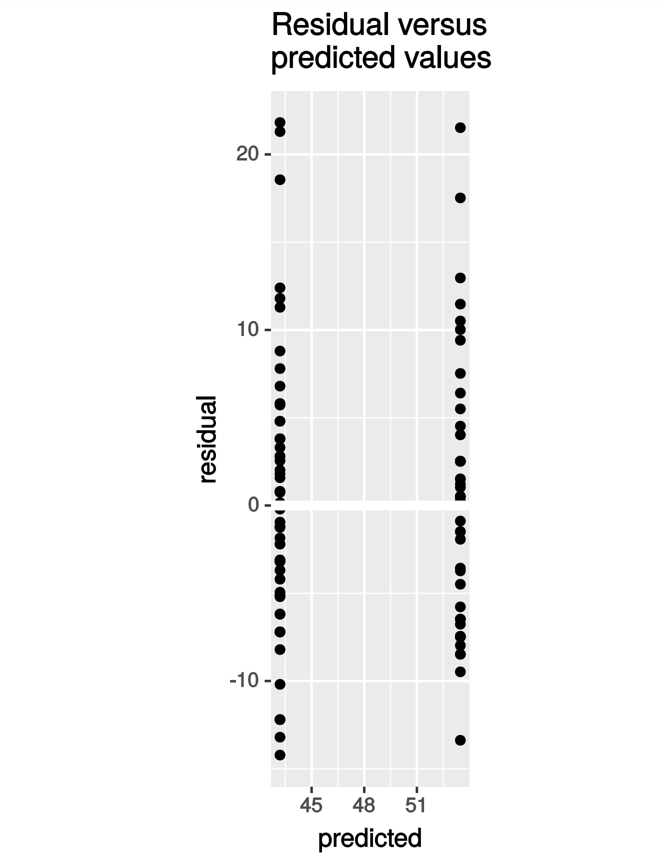
- However, the long tails and this model's poor prediction ability are good enough reason to try and build a better model



Residual vs. predicted values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(pred, resid)) +
  geom_ref_line(h = 0)
```

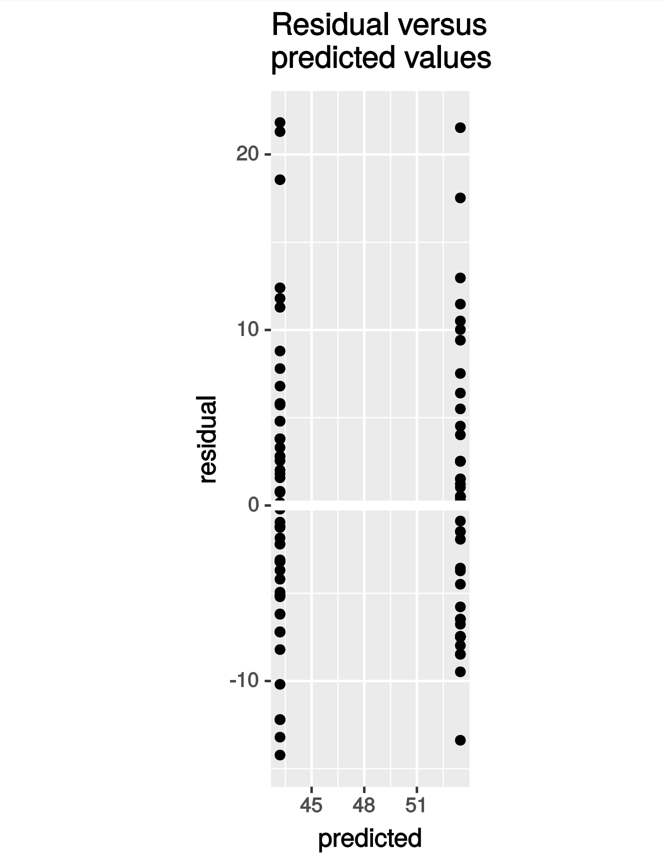
- The residual spread stays consistent, so that's good
- However, the long tails and this model's poor prediction ability are good enough reason to try and build a better model
- We can try building other univariate models with the other columns



Residual vs. predicted values

```
ggplot(mariokart_cond_model_df) +
  geom_point(aes(pred, resid)) +
  geom_ref_line(h = 0)
```

- The residual spread stays consistent, so that's good
- However, the long tails and this model's poor prediction ability are good enough reason to try and build a better model
- We can try building other univariate models with the other columns
- However, as we'll find out, it's better to train **multivariate** models on this dataset



Credits

`modelr` package examples using `sim` data set adapted from content in chapters 23.2 and 23.3 of *R for Data Science* by Hadley Wickham and Garrett Grolemund and made available under the CC BY-NC-ND 3.0 license.

Mario Kart data set source: David M Diez, Christopher D Barr, and Mine Çetinkaya-Rundel. 2012. *openintro*: OpenIntro data sets and supplemental functions. http://cran.r-project.org/web/packages/openintro

Mario Kart example loosely adapted from content in chapters 6.1, 6.2, and 6.3 of the *Introductory Statistics with Randomization and Simulation* textbook by David M Diez, Christopher D Barr, and Mine Çetinkaya-Rundel and made available under the CC BY-NC-SA 3.0 Unported license.