# Class 21: Modeling III

June 19, 2018

# General

# Annoucements

- 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

# Case study: Mario Kart eBay prices dataset

# 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

# Last time...

- Removed outliers

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

# Last time...

- Removed outliers

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

- Split dataset 80/20 into `train` and `test`

```
train <- mariokart2 %>%
  sample_frac(size = 0.80, replace = FALSE)

test  <- mariokart2 %>%
  anti_join(train, by = 'ID')
```

# Last time...

- Removed outliers

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

- Split dataset 80/20 into `train` and `test`

```
train <- mariokart2 %>%
  sample_frac(size = 0.80, replace = FALSE)

test  <- mariokart2 %>%
  anti_join(train, by = 'ID')
```

- Built model of `totalPr` (response) using `cond` (explanatory) variable

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

# Univariate model results

# Univariate model results

Predict training dataset and compute the residuals

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

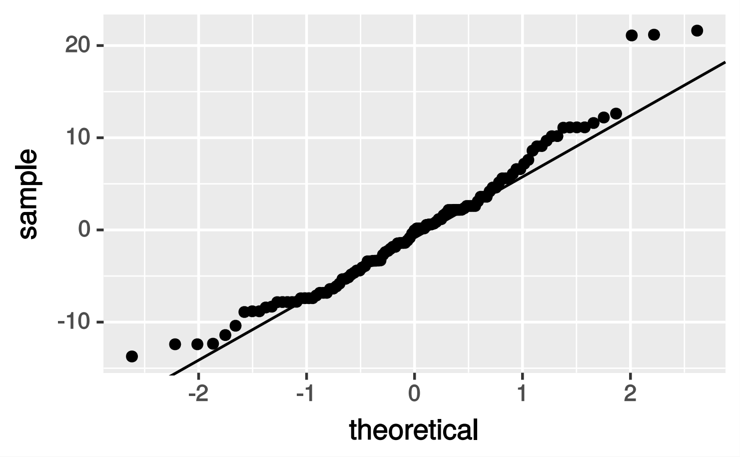
# Univariate model results

Predict training dataset and compute the residuals

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

Check if residuals are nearly normal

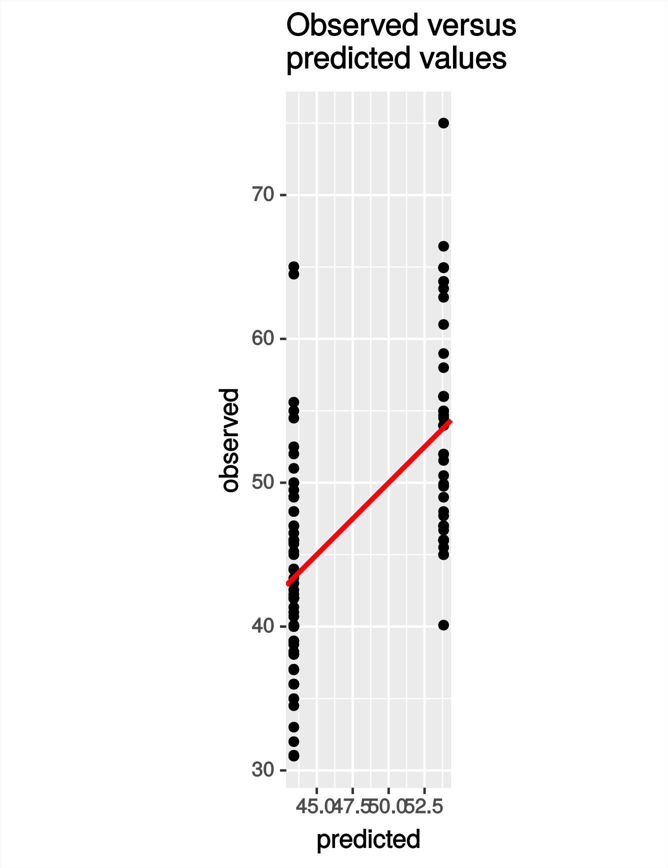
```
ggplot(mariokart_cond_model_df) +
  geom_qq(
    mapping = aes(sample = resid)
  ) +
  geom_qq_ref_line(
    data = mariokart_cond_model_df,
    variable = "resid"
  )
```



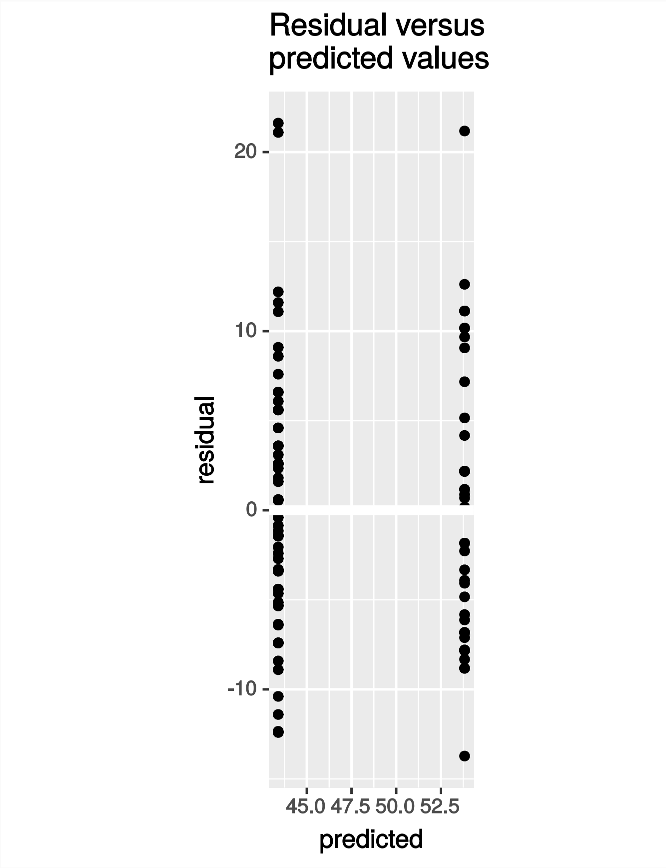

Deviations from normal distribution with long tail on the right

# Plots for model evaluation

```r
ggplot(mariokart_cond_model_df) +
  geom_point(aes(pred, totalPr)) +
  geom_abline(slope = 1, intercept = 0)
```




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



# Multivariate linear regression models

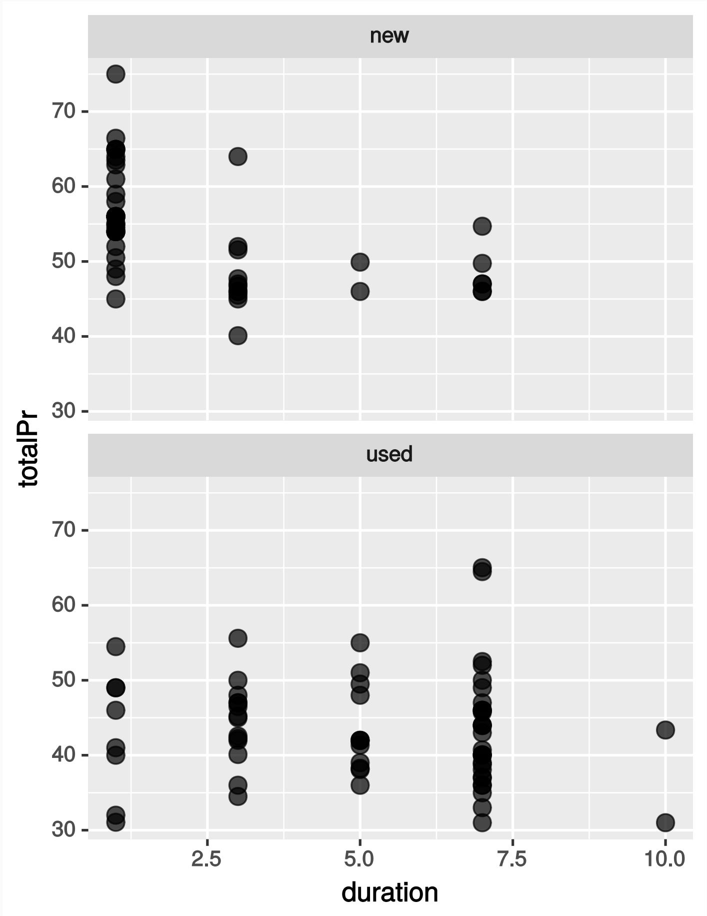
# When many variables matter

Let's see how `cond` and `duration` affect `totalPr`:

# When many variables matter

Let's see how `cond` and `duration` affect `totalPr`:

```
ggplot(train) +
  geom_point(aes(duration, totalPr)) +
  facet_wrap(~cond, ncol = 1)
```

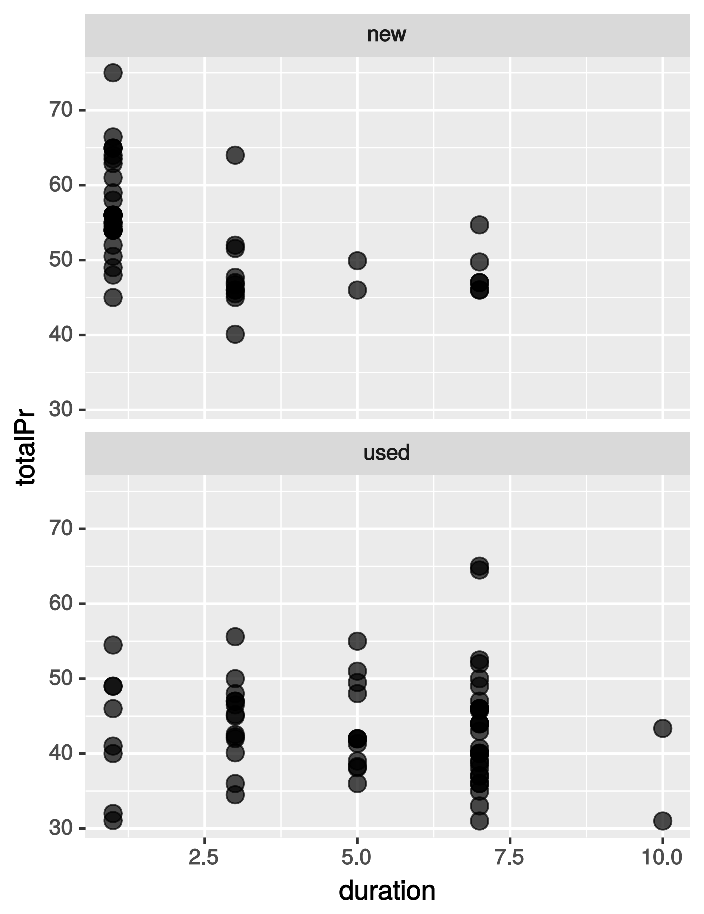


# When many variables matter

Let's see how `cond` and `duration` affect `totalPr`:

```
ggplot(train) +
  geom_point(aes(duration, totalPr)) +
  facet_wrap(~cond, ncol = 1)
```

- There's a modest dependence of `duration` on `cond`, especially with new games of short duration



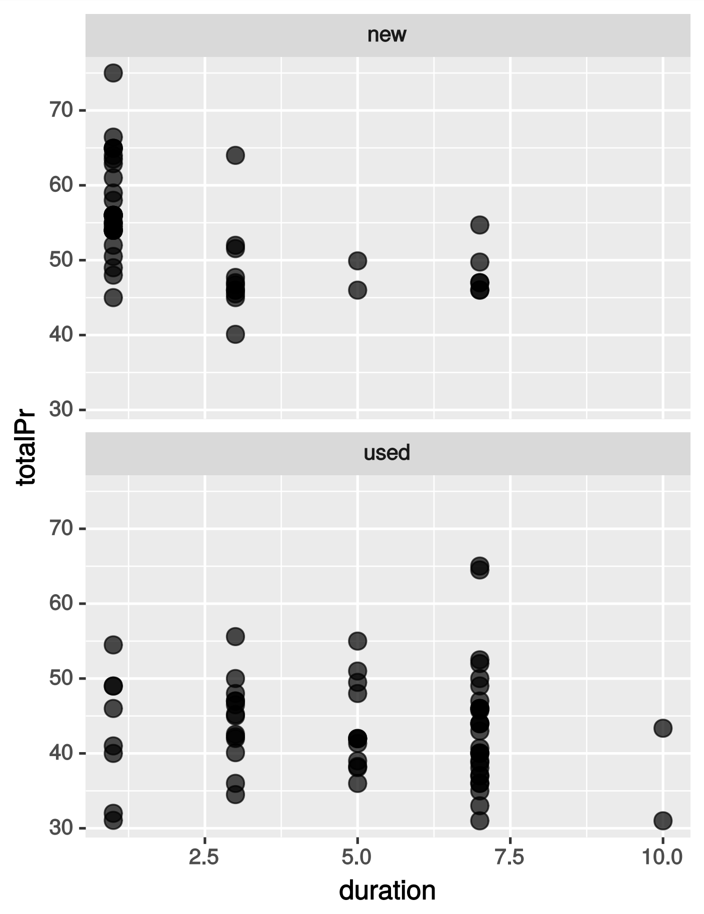
# When many variables matter

Let's see how `cond` and `duration` affect `totalPr`:

```
ggplot(train) +
  geom_point(aes(duration, totalPr)) +
  facet_wrap(~cond, ncol = 1)
```

- There's a modest dependence of `duration` on `cond`, especially with new games of short duration

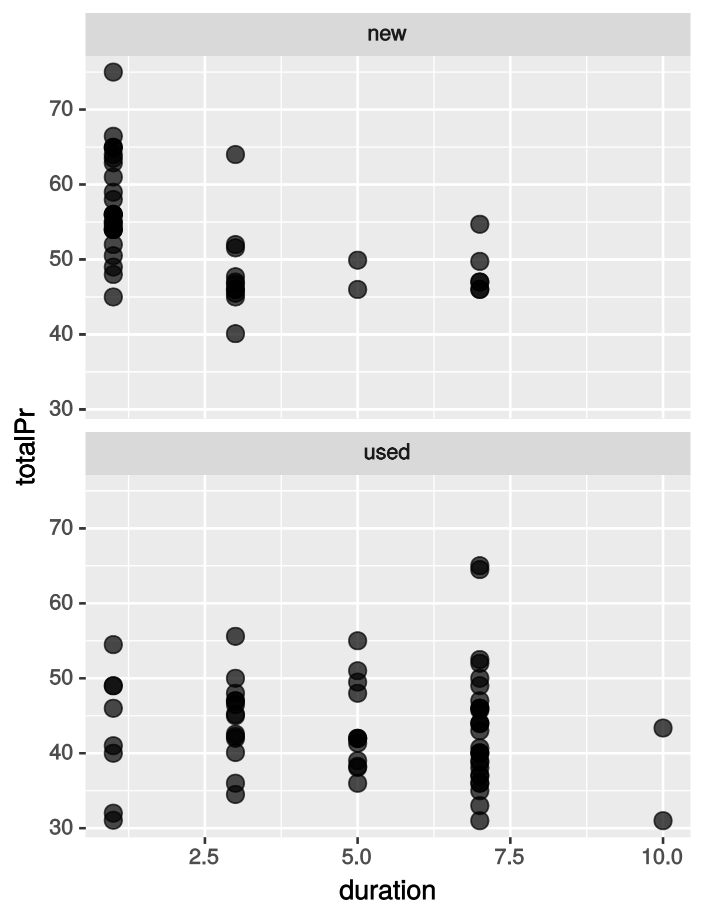
- **If independent:** you'd see same trend in both boxes, just shifted by a constant amount



# When many variables matter

Let's see how `cond` and `duration` affect `totalPr`:

```
ggplot(train) +
  geom_point(aes(duration, totalPr)) +
  facet_wrap(~cond, ncol = 1)
```



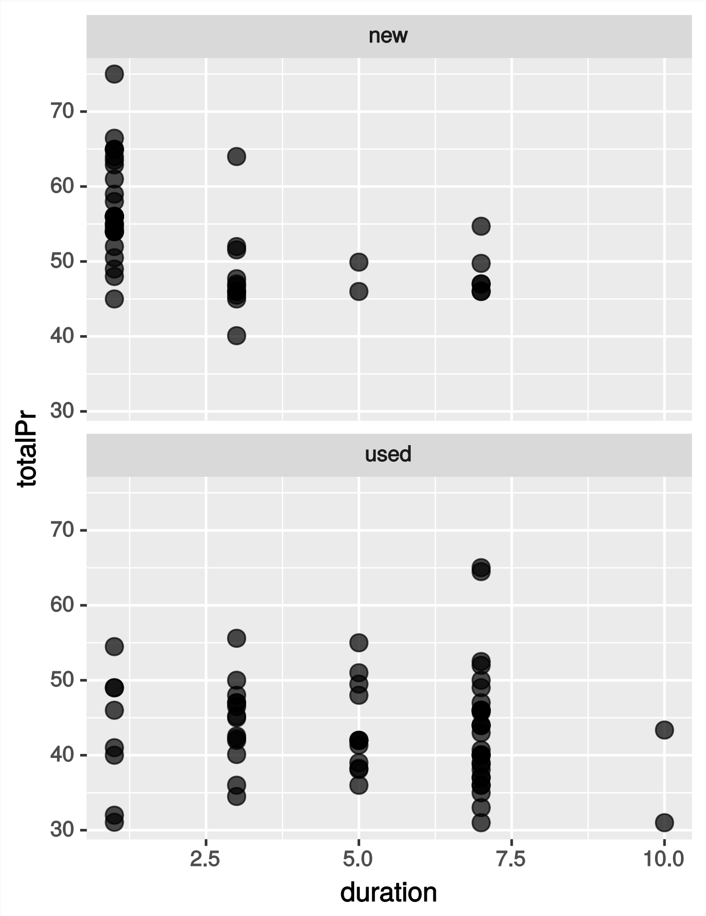

- There's a modest dependence of `duration` on `cond`, especially with new games of short duration
- **If independent:** you'd see same trend in both boxes, just shifted by a constant amount
- **If interacting:** different trends in both boxes, not just a constant shift

# When many variables matter

Let's see how `cond` and `duration` affect `totalPr`:

```
ggplot(train) +
  geom_point(aes(duration, totalPr)) +
  facet_wrap(~cond, ncol = 1)
```

- There's a modest dependence of `duration` on `cond`, especially with new games of short duration
- **If independent:** you'd see same trend in both boxes, just shifted by a constant amount
- **If interacting:** different trends in both boxes, not just a constant shift
- Modest interaction between `cond` and `duration`, keep that in mind



# Predicting price using four variables

# Predicting price using four variables

- Build a linear model using the variables `cond`, `stockPhoto`, `duration`, and `wheels`

# Predicting price using four variables

- Build a linear model using the variables `cond`, `stockPhoto`, `duration`, and `wheels`
- Variables are independent in this model and we do not consider interaction terms like `cond * duration`.

# Predicting price using four variables

- Build a linear model using the variables `cond`, `stockPhoto`, `duration`, and `wheels`
- Variables are independent in this model and we do not consider interaction terms like `cond * duration`.

```
mariokart_multivar_model_lm <- lm(
  formula = totalPr ~ cond + stockPhoto + duration + wheels,
  data = train
)
```

# Predicting price using four variables

- Build a linear model using the variables `cond`, `stockPhoto`, `duration`, and `wheels`
- Variables are independent in this model and we do not consider interaction terms like `cond * duration`.

```
mariokart_multivar_model_lm <- lm(
  formula = totalPr ~ cond + stockPhoto + duration + wheels,
  data = train
)
```

- Predict training dataset and compute the residuals

# Predicting price using four variables

- Build a linear model using the variables `cond`, `stockPhoto`, `duration`, and `wheels`
- Variables are independent in this model and we do not consider interaction terms like `cond * duration`.

```
mariokart_multivar_model_lm <- lm(
  formula = totalPr ~ cond + stockPhoto + duration + wheels,
  data = train
)
```

- Predict training dataset and compute the residuals

```
mariokart_multivar_model_df <- train %>%
  add_predictions(mariokart_multivar_model_lm) %>%
  add_residuals(mariokart_multivar_model_lm)
```

# Visualize the model...

- We do this just like last time, right? We can plot the model on top of a plot of the predictor variables.

# Visualize the model...

- We do this just like last time, right? We can plot the model on top of a plot of the predictor variables.
- This would be possible...

# Visualize the model...

- We do this just like last time, right? We can plot the model on top of a plot of the predictor variables.
- This would be possible... if we could create 5-dimensional images

# Visualize the model...

- We do this just like last time, right? We can plot the model on top of a plot of the predictor variables.
- This would be possible... if we could create 5-dimensional images
- Use observed versus predicted and residual versus predicted plots like we created for the `totalPr ~ cond` model

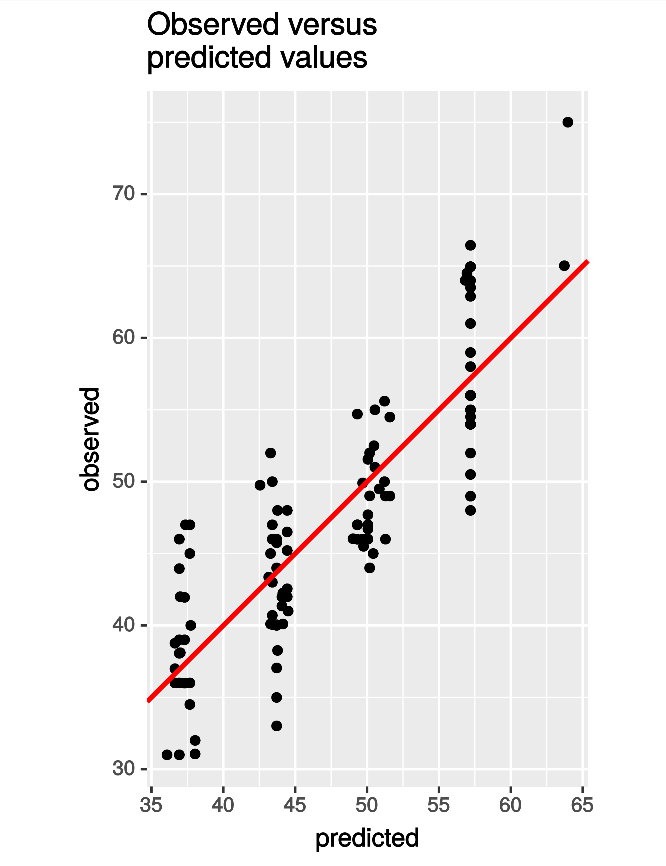
# Visualize the model...

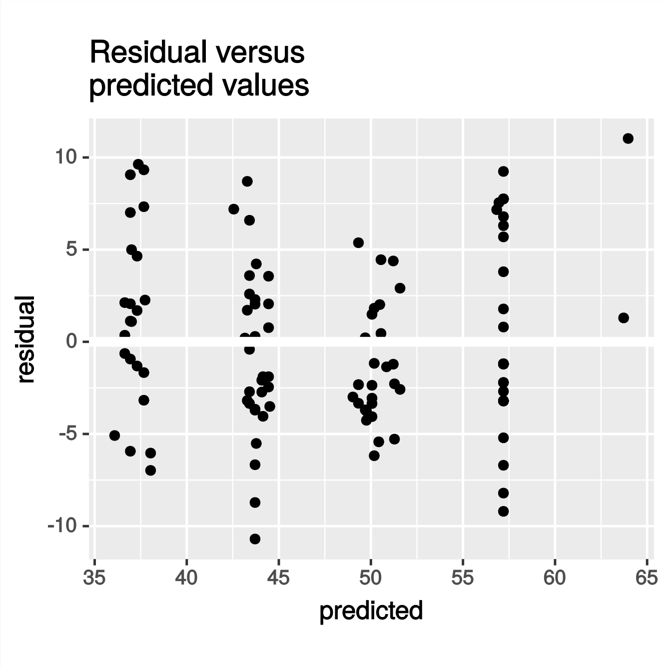
- We do this just like last time, right? We can plot the model on top of a plot of the predictor variables.
- This would be possible... if we could create 5-dimensional images
- Use observed versus predicted and residual versus predicted plots like we created for the `totalPr ~ cond` model

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

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

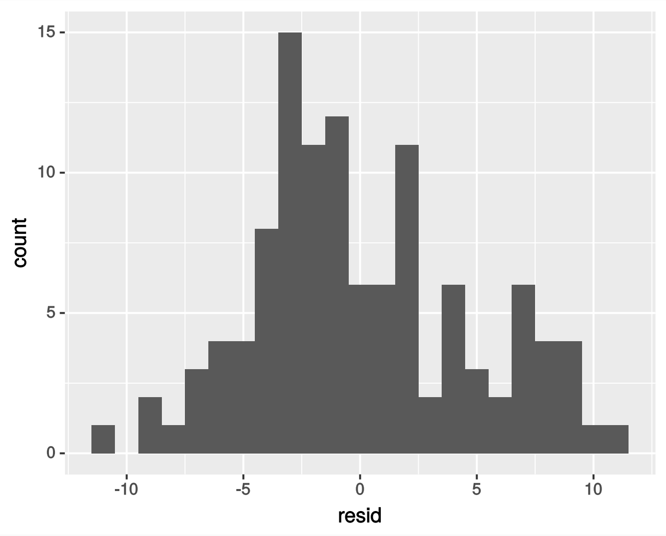
# Multivariate model performance



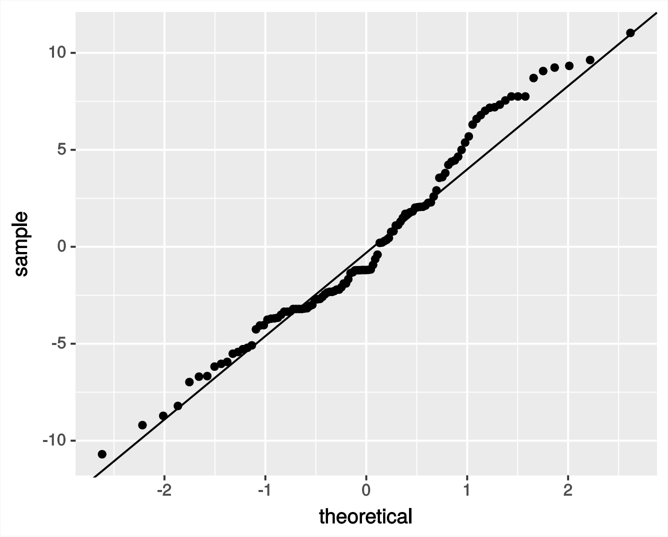



# Inspect multivariate model residuals

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

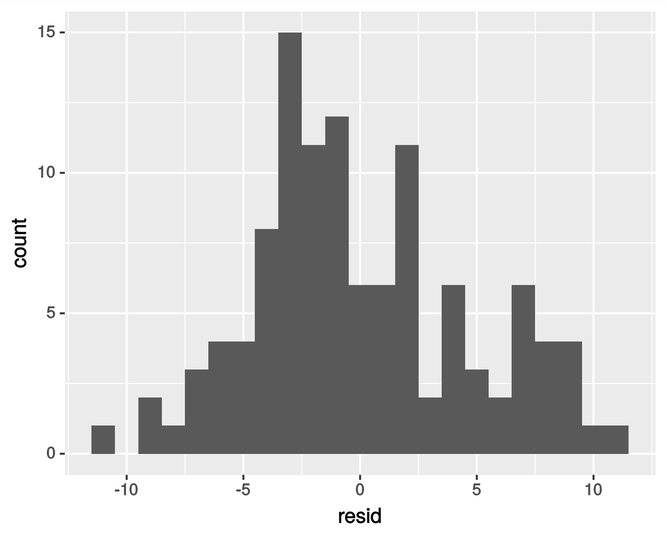



```
ggplot(mariokart_multivar_model_df) +
  geom_qq(mapping = aes(sample = resid)) +
  geom_qq_ref_line(data = mariokart_multiva
```

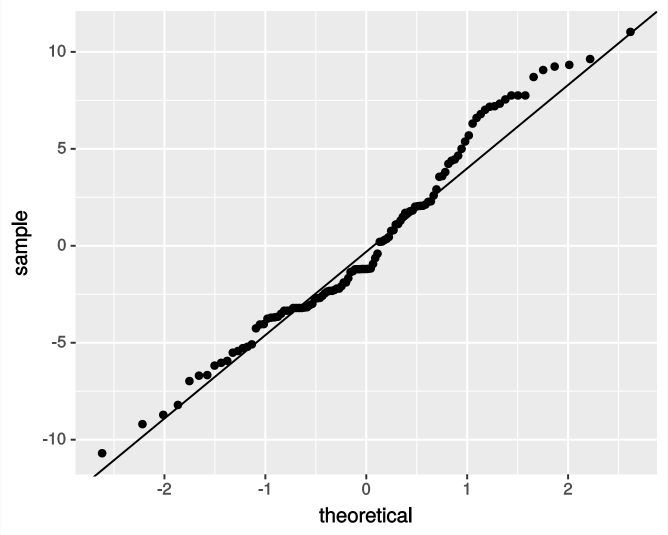


# Inspect multivariate model residuals

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




```
ggplot(mariokart_multivar_model_df) +
  geom_qq(mapping = aes(sample = resid)) +
  geom_qq_ref_line(data = mariokart_multiva
```




- Residuals still show deviations from the normal distribution on the right-side tail, but they're smaller overall

# Comparing the two models

- Compare the residual histograms of the two models

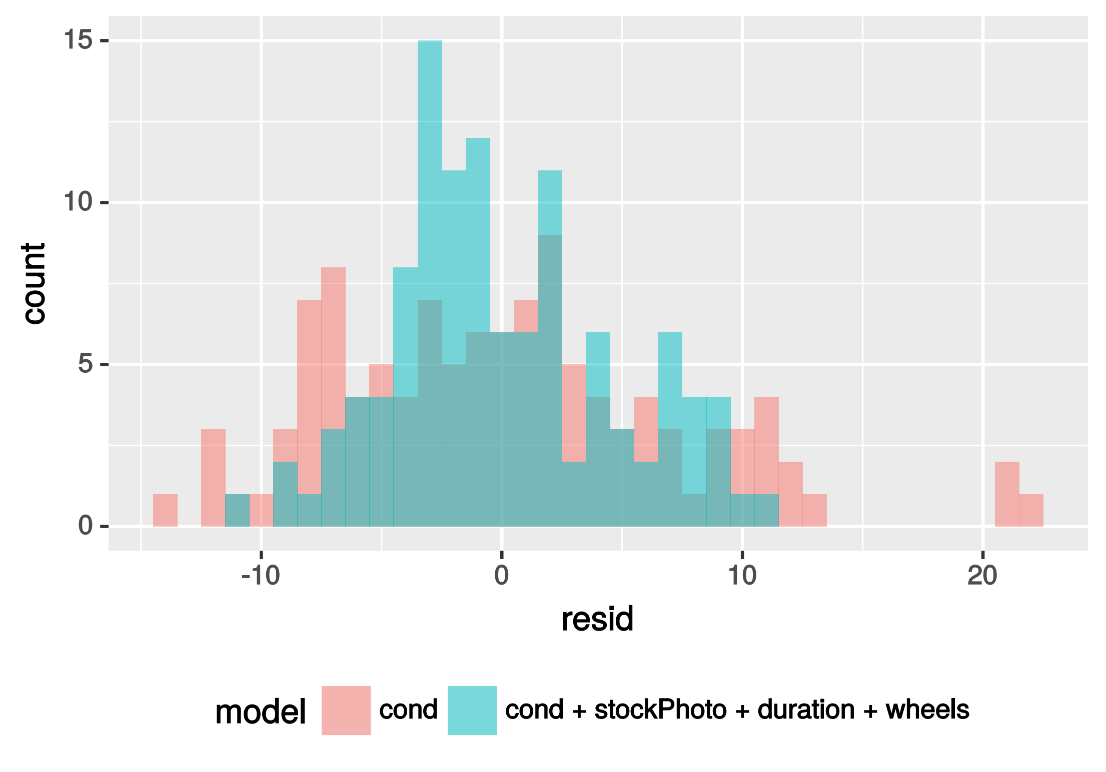
# Comparing the two models

- Compare the residual histograms of the two models

```
data_frame(
  model = c(
    rep("cond", nrow(mariokart_cond_model_df)),
    rep(
      "cond + stockPhoto + duration + wheels",
      nrow(mariokart_multivar_model_df)
    )
  ),
  resid = c(
    pull(mariokart_cond_model_df, "resid"),
    pull(mariokart_multivar_model_df, "resid")
  )
) %>%
  ggplot() +
  geom_histogram(
    mapping = aes(x = resid, fill = model), alpha = 0.5, binwidth = 1,
    position = "identity", center = 0
  ) +
  theme(legend.position = "bottom")
```

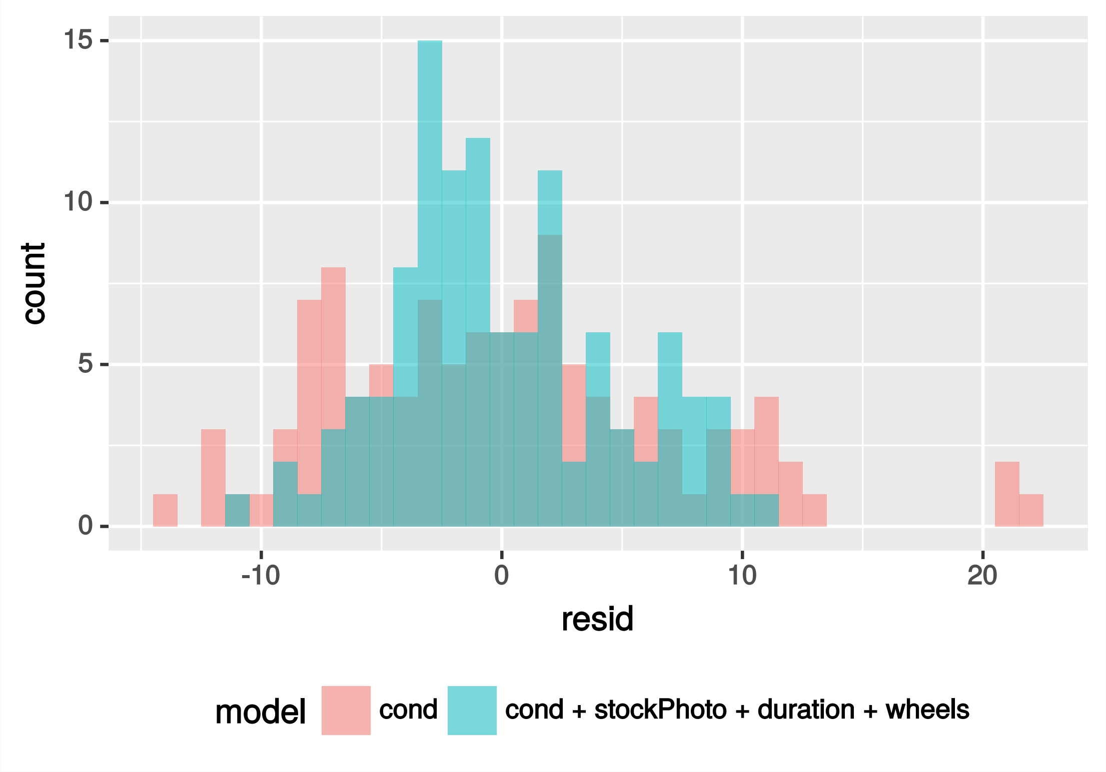
# Comparing the two models

- Compare the residual histograms of the two models



# Comparing the two models

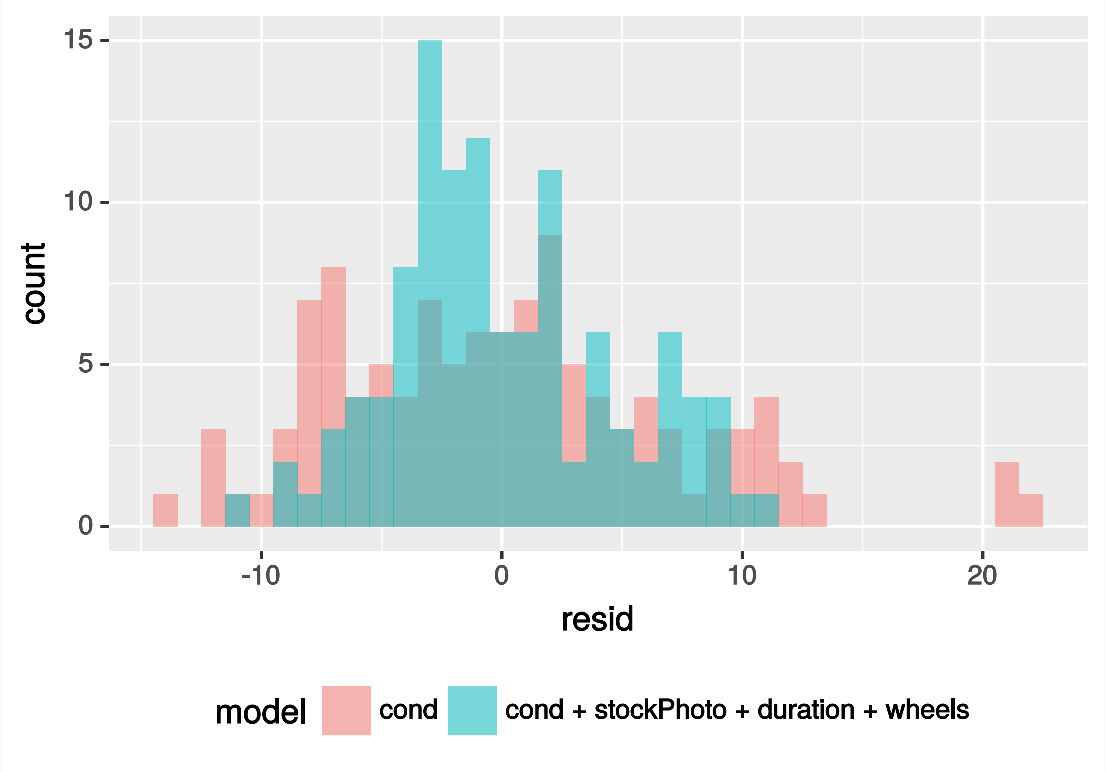
- Compare the residual histograms of the two models




- Multivariate model *seems* better

# Comparing the two models

- Compare the residual histograms of the two models




- Multivariate model *seems* better, but it'd be better if we had an objective measure of model quality

# Model selection

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models, but it's just a qualitative measure

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models, but it's just a qualitative measure
- How should we compare and rank models?

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models, but it's just a qualitative measure
- How should we compare and rank models?
- This is what model selection is about, computing scores and measures of model performance for different models, and selecting the best choice.

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models, but it's just a qualitative measure
- How should we compare and rank models?
- This is what model selection is about, computing scores and measures of model performance for different models, and selecting the best choice.
- Bootstrapping is one option

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models, but it's just a qualitative measure
- How should we compare and rank models?
- This is what model selection is about, computing scores and measures of model performance for different models, and selecting the best choice.
- Bootstrapping is one option
- Cross-validation is another method that can compare relative model performance using only training data

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models, but it's just a qualitative measure
- How should we compare and rank models?
- This is what model selection is about, computing scores and measures of model performance for different models, and selecting the best choice.
- Bootstrapping is one option
- Cross-validation is another method that can compare relative model performance using only training data
- A popular flavor of cross-validation (especially among data scientists) is called **k-fold cross-validation**

# Question, what kind of model is best?

- Comparing residuals can help us understand the relative performance of models, but it's just a qualitative measure
- How should we compare and rank models?
- This is what model selection is about, computing scores and measures of model performance for different models, and selecting the best choice.
- Bootstrapping is one option
- Cross-validation is another method that can compare relative model performance using only training data
- A popular flavor of cross-validation (especially among data scientists) is called **k-fold cross-validation**
- **Basic idea:** Estimate how robust your model is by systematically removing different chunks (the "folds") of the dataset, repeating the fitting process, then testing its predictive power on the folds

# k-fold cross-validation

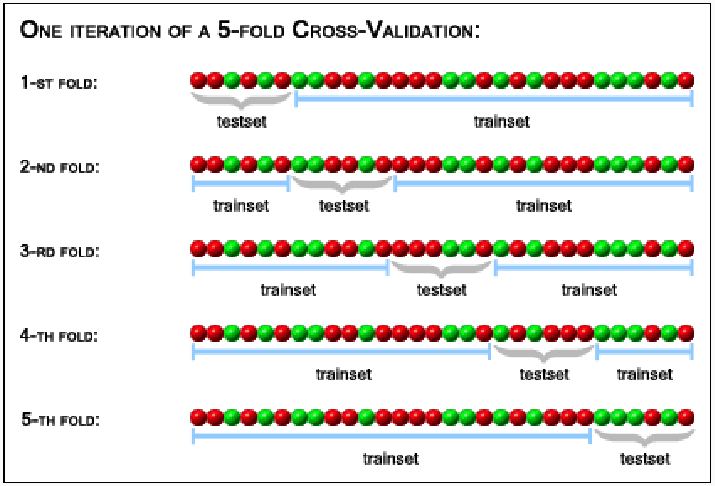
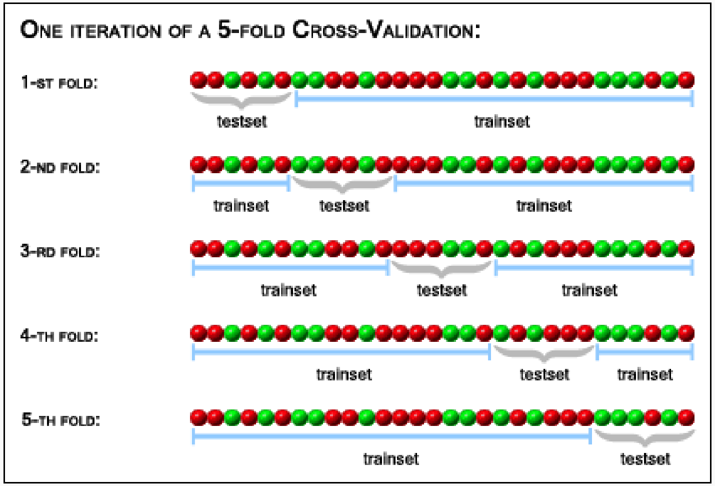



Image: "Cross-Validation Explained", *ProClassify User's Guide*, http://genome.tugraz.at/proclassify/help/pages/images/xv_folds.gif

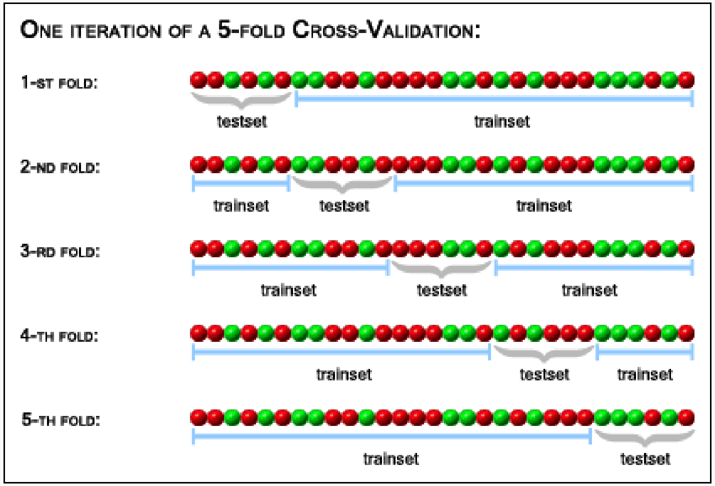
# k-fold cross-validation




- The above example illustrates a *5-fold*, or $k = 5$, cross-validation.

Image: "Cross-Validation Explained", *ProClassify User's Guide*, http://genome.tugraz.at/proclassify/help/pages/images/xv_folds.gif

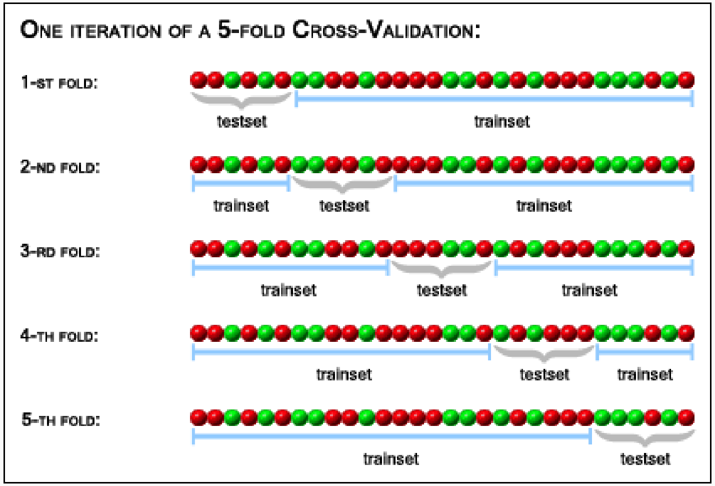
# k-fold cross-validation




- The above example illustrates a *5-fold*, or $k = 5$, cross-validation.
- Each fold will act as a testing set, with the remaining $k - 1$ folds used to train the model.

Image: "Cross-Validation Explained", *ProClassify User's Guide*, http://genome.tugraz.at/proclassify/help/pages/images/xv_folds.gif

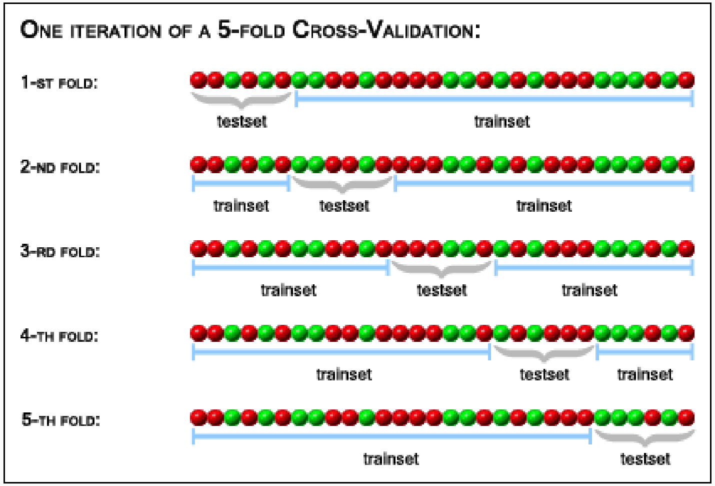
# k-fold cross-validation




- The above example illustrates a *5-fold*, or $\boxed{k = 5}$, cross-validation.
- Each fold will act as a testing set, with the remaining $\boxed{k - 1}$ folds used to train the model.
- Fit model, predict values in testing set, then calculate the mean-squared prediction error (MSE)

Image: "Cross-Validation Explained", *ProClassify User's Guide*, http://genome.tugraz.at/proclassify/help/pages/images/xv_folds.gif

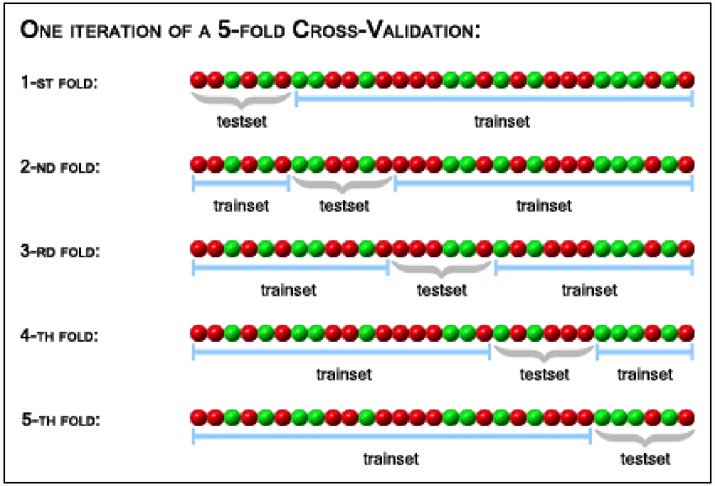
# k-fold cross-validation




- The above example illustrates a *5-fold*, or $\boxed{k = 5}$, cross-validation.
- Each fold will act as a testing set, with the remaining $\boxed{k - 1}$ folds used to train the model.
- Fit model, predict values in testing set, then calculate the mean-squared prediction error (MSE)
- MSE gives an estimate of how well the model works as a predictor

Image: "Cross-Validation Explained", *ProClassify User's Guide*, http://genome.tugraz.at/proclassify/help/pages/images/xv_folds.gif

# k-fold cross-validation




- The above example illustrates a *5-fold*, or $k = 5$, cross-validation.
- Each fold will act as a testing set, with the remaining $k - 1$ folds used to train the model.
- Fit model, predict values in testing set, then calculate the mean-squared prediction error (MSE)
- MSE gives an estimate of how well the model works as a predictor
- MSE is general-purpose and allows you to compare models of many types

Image: "Cross-Validation Explained", *ProClassify User's Guide*, http://genome.tugraz.at/proclassify/help/pages/images/xv_folds.gif

# Cross-validating our models

# Cross-validating our models

- The code for doing k-fold cross-validation, even with the `tidyverse` tools, is *just* complicated enough that it's beyond the scope of the course

# Cross-validating our models

- The code for doing k-fold cross-validation, even with the `tidyverse` tools, is *just* complicated enough that it's beyond the scope of the course

- To let you practice model selection, run the following code to load in the function `rep_kfold_cv()`

```
load(url("http://spring18.cds101.com/files/R/repeated_kfold_cross_validation.RData"))
```

# Cross-validating our models

- The code for doing k-fold cross-validation, even with the `tidyverse` tools, is *just* complicated enough that it's beyond the scope of the course

- To let you practice model selection, run the following code to load in the function `rep_kfold_cv()`

```
load(url("http://spring18.cds101.com/files/R/repeated_kfold_cross_validation.RData"))
```

- This function takes a linear regression model and cross-validates it automatically for you, you just supply the following inputs:

| Input | Description |
|---|---|
| data | The training dataset |
| k | Number of folds to use |
| model | Model to cross-validate written in `lm()` syntax |
| cv_reps | Number of times to repeat cross-validation sequence to improve statistics |

# Applying cross-validation to our models

- Cross-validate the univariate model `totalPr ~ cond`

# Applying cross-validation to our models

- Cross-validate the univariate model `totalPr ~ cond`

```
rep_kfold_cv(data = train, k = 10, model = totalPr ~ cond, cv_reps = 3)
```

# Applying cross-validation to our models

- Cross-validate the univariate model `totalPr ~ cond`

```
rep_kfold_cv(data = train, k = 10, model = totalPr ~ cond, cv_reps = 3)
```

| r_squared | mse | adjusted_mse |
|---:|---:|---:|
| 0.2332096 | 52.85823 | 52.78021 |

# Applying cross-validation to our models

- Cross-validate the univariate model `totalPr ~ cond`

```
rep_kfold_cv(data = train, k = 10, model = totalPr ~ cond, cv_reps = 3)
```

| r_squared | mse | adjusted_mse |
|---|---|---|
| 0.2332096 | 52.85823 | 52.78021 |

- Cross-validate the multivariate model `totalPr ~ cond + stockPhoto + duration + wheels`

# Applying cross-validation to our models

- Cross-validate the univariate model `totalPr ~ cond`

```
rep_kfold_cv(data = train, k = 10, model = totalPr ~ cond, cv_reps = 3)
```

| r_squared | mse | adjusted_mse |
|---|---|---|
| 0.2332096 | 52.85823 | 52.78021 |

- Cross-validate the multivariate model `totalPr ~ cond + stockPhoto + duration + wheels`

```
rep_kfold_cv(
  data = train, k = 10,
  model = totalPr ~ cond + stockPhoto + duration + wheels, cv_reps = 3
)
```

# Applying cross-validation to our models

- Cross-validate the univariate model `totalPr ~ cond`

```
rep_kfold_cv(data = train, k = 10, model = totalPr ~ cond, cv_reps = 3)
```

| r_squared | mse | adjusted_mse |
|---|---|---|
| 0.2332096 | 52.85823 | 52.78021 |

- Cross-validate the multivariate model `totalPr ~ cond + stockPhoto + duration + wheels`

```
rep_kfold_cv(
  data = train, k = 10,
  model = totalPr ~ cond + stockPhoto + duration + wheels, cv_reps = 3
)
```

| r_squared | mse | adjusted_mse |
|---|---|---|
| 0.6334545 | 23.59525 | 23.49117 |

# Applying cross-validation to our models

- Cross-validate the univariate model `totalPr ~ cond`

```
rep_kfold_cv(data = train, k = 10, model = totalPr ~ cond, cv_reps = 3)
```

| r_squared | mse | adjusted_mse |
|---|---|---|
| 0.2332096 | 52.85823 | 52.78021 |

- Cross-validate the multivariate model `totalPr ~ cond + stockPhoto + duration + wheels`

```
rep_kfold_cv(
  data = train, k = 10,
  model = totalPr ~ cond + stockPhoto + duration + wheels, cv_reps = 3
)
```

| r_squared | mse | adjusted_mse |
|---|---|---|
| 0.6334545 | 23.59525 | 23.49117 |

*Scores indicate the multivariate model performs better than the univariate model*

# Model selection activity

# Find the best model!

- Use `rep_kfold_cv()` to test the different additive models we can build using the `cond`, `duration`, `stockPhoto`, and `wheels` columns.
- 15 permutations in all
- According to the adjusted mean-squared error (mse), which model should we select?
- Once we've selected a model, fit it to the full training dataset, then check the mean-squared error for its predictions on the `test` data:

```
lm(totalPr ~ cond, data = train) %>%
  mse(test)
```

| model | cond | duration | stockPhoto | wheels | adj_mse |
|---|---|---|---|---|---|
| 1 | ✗ | | | | 52.8 |
| 2 | ✗ | | | ✗ | |
| 3 | ✗ | | ✗ | | |
| 4 | ✗ | | ✗ | ✗ | |
| 5 | ✗ | ✗ | | | |
| 6 | ✗ | ✗ | | ✗ | |
| 7 | ✗ | ✗ | ✗ | | |
| 8 | ✗ | ✗ | ✗ | ✗ | 23.5 |
| 9 | | | | ✗ | |
| 10 | | | ✗ | | |
| 11 | | | ✗ | ✗ | |
| 12 | | ✗ | | | |
| 13 | | ✗ | | ✗ | |
| 14 | | ✗ | ✗ | | |
| 15 | | ✗ | ✗ | ✗ | |

# Credits

**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.