Cross Validation

Philip Waggoner, MACS 30100 University of Chicago

Resampling

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ISLR)
library(broom)
library(rsample)
library(rcfss)
library(yardstick)
```

```
## For binary classification, the first factor level is assumed to be the event.
## Use the argument `event_level = "second"` to alter this as needed.

##
## Attaching package: 'yardstick'

## The following object is masked from 'package:readr':
##
##     spec
```

```
Auto <- as_tibble(Auto)

set.seed(1234)

auto_split <- initial_split(data = Auto,
                            prop = 0.5)

auto_train <- training(auto_split)
auto_test <- testing(auto_split)
```

```
auto_lm <- glm(mpg ~ horsepower,
               data = auto_train); summary(auto_lm)
```

```
##
## Call:
## glm(formula = mpg ~ horsepower, data = auto_train)
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -13.7460   -3.2696   -0.1623    2.6319   13.9996
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 40.800608   1.036885   39.35   <2e-16 ***
## horsepower  -0.167425   0.009566  -17.50   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 23.75524)
##
##     Null deviance: 11885.6  on 195  degrees of freedom
## Residual deviance:  4608.5  on 194  degrees of freedom
## AIC: 1181.1
##
## Number of Fisher Scoring iterations: 2
```

To calculate the error (MSE) for training set:

```
(train_mse <- augment(auto_lm,
                      newdata = auto_train) %>%
  mse(truth = mpg,
      estimate = .fitted))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        23.5
```

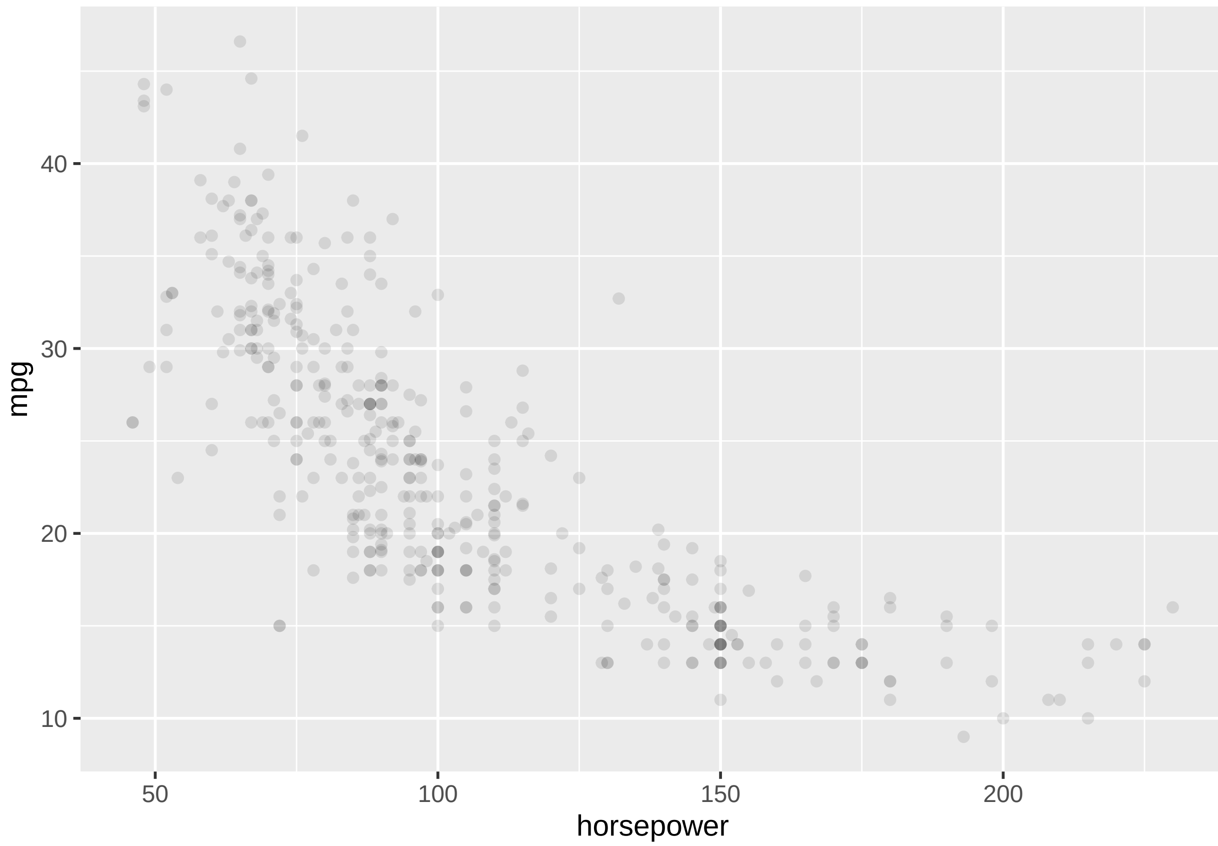
For the validation set:

```
(test_mse <- augment(auto_lm,
                     newdata = auto_test) %>%
  mse(truth = mpg,
      estimate = .fitted))
```

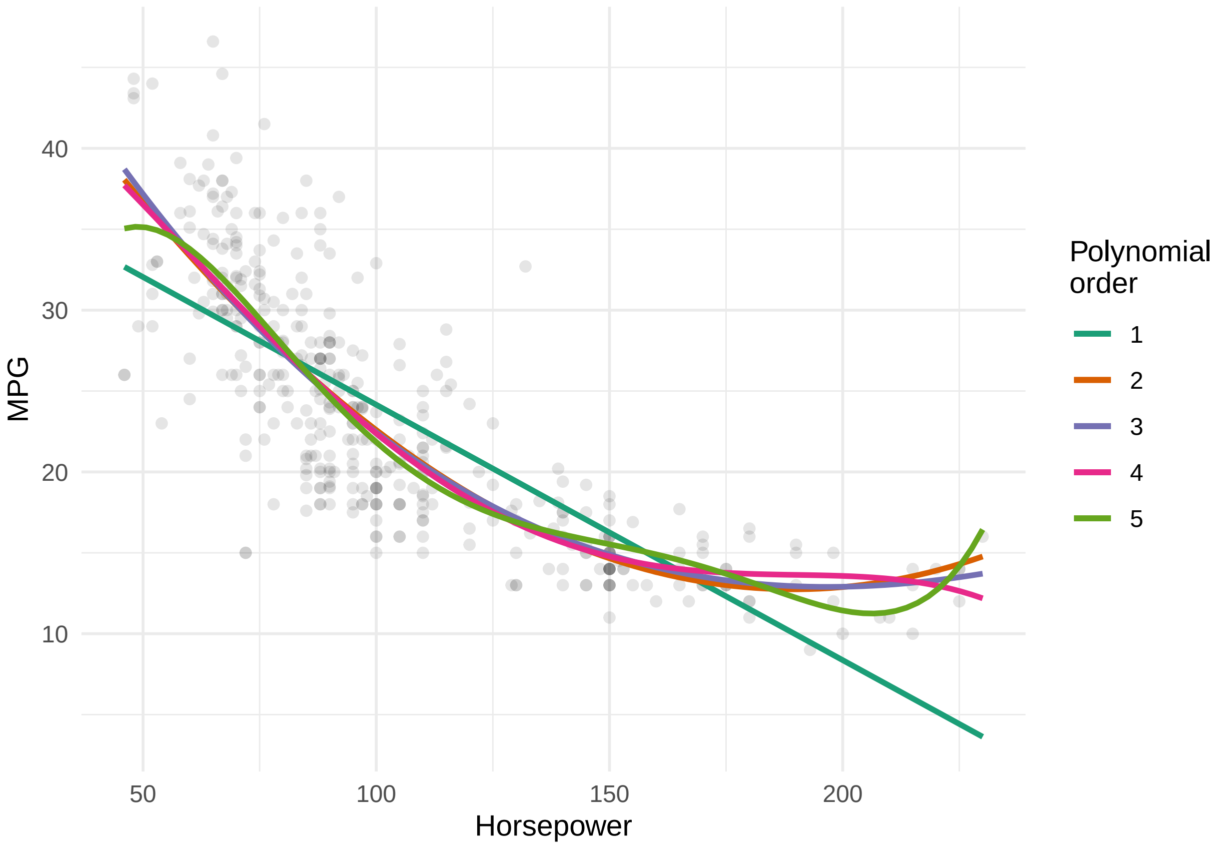
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        24.7
```

But for now, add complexity to the model to attempt to better explain the data.

```
# visualize each model

ggplot(Auto, aes(horsepower, mpg)) +
  geom_point(alpha = .1)
```




```
ggplot(Auto, aes(horsepower, mpg)) +
  geom_point(alpha = .1) +
  geom_smooth(aes(color = "1"),
              method = "glm",
              formula = y ~ poly(x, i = 1, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "2"),
              method = "glm",
              formula = y ~ poly(x, i = 2, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "3"),
              method = "glm",
              formula = y ~ poly(x, i = 3, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "4"),
              method = "glm",
              formula = y ~ poly(x, i = 4, raw = TRUE),
              se = FALSE) +
  geom_smooth(aes(color = "5"),
              method = "glm",
              formula = y ~ poly(x, i = 5, raw = TRUE),
              se = FALSE) +
  scale_color_brewer(type = "qual", palette = "Dark2") +
  labs(x = "Horsepower",
       y = "MPG",
       color = "Polynomial\norder") +
  theme_minimal()
```

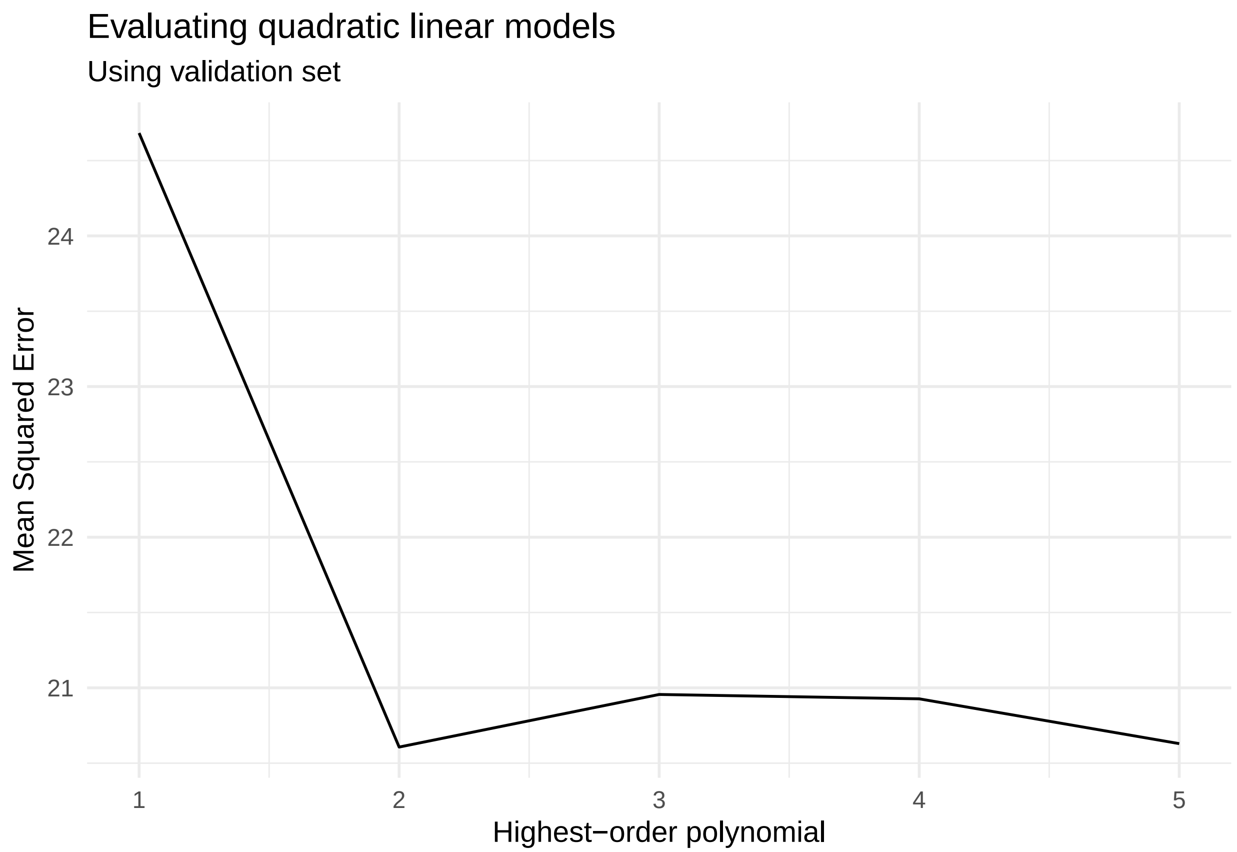



```
# function to train the model (training set) and evaluate the model (validation set), across each polyn
poly_results <- function(train, test, i) {

  mod <- glm(mpg ~ poly(horsepower, i, raw = TRUE), data = train)

  res <- augment(mod,
                 newdata = test) %>%
    mse(truth = mpg,
        estimate = .fitted)
  res
}

# function to return MSE for a unique order polynomial term
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
poly_mse <- function(i, train, test){
  poly_results(train, test, i) %$%
    mean(.estimate)
```

```
}

cv_mse <- tibble(terms = seq(from = 1, to = 5),
                 mse_test = map_dbl(terms, poly_mse, auto_train, auto_test))

ggplot(cv_mse, aes(terms, mse_test)) +
  geom_line() +
  labs(title = "Evaluating quadratic linear models",
       subtitle = "Using validation set",
       x = "Highest-order polynomial",
       y = "Mean Squared Error") +
  theme_minimal()
```



Classification

Predict passenger survival (yes or no) during the sinking of the Titanic.

```
library(titanic)

titanic <- as_tibble(titanic_train) %>%
  mutate(Survived = factor(Survived))

titanic %>%
  head(n = 5)
```

```
## # A tibble: 5 x 12
##   PassengerId Survived Pclass Name  Sex     Age SibSp Parch Ticket  Fare Cabin
```

```
##         <int> <fct>     <int> <chr> <chr> <dbl> <int> <int> <chr>   <dbl> <chr>
## 1           1 0             3 Brau~ male     22     1     0 A/5 2~  7.25 ""
## 2           2 1             1 Cumi~ fema~    38     1     0 PC 17~ 71.3  "C85"
## 3           3 1             3 Heik~ fema~    26     0     0 STON/~  7.92 ""
## 4           4 1             1 Futr~ fema~    35     1     0 113803 53.1  "C12~
## 5           5 0             3 Alle~ male     35     0     0 373450  8.05 ""
## # ... with 1 more variable: Embarked <chr>
```

```
survive_age_woman_x <- glm(Survived ~ Age * Sex, data = titanic,
                           family = binomial)
summary(survive_age_woman_x)
```

```
##
## Call:
## glm(formula = Survived ~ Age * Sex, family = binomial, data = titanic)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9401  -0.7136  -0.5883   0.7626   2.2455
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.59380    0.31032   1.913  0.05569 .
## Age          0.01970    0.01057   1.863  0.06240 .
## Sexmale     -1.31775    0.40842  -3.226  0.00125 **
## Age:Sexmale -0.04112    0.01355  -3.034  0.00241 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 964.52  on 713  degrees of freedom
## Residual deviance: 740.40  on 710  degrees of freedom
##   (177 observations deleted due to missingness)
## AIC: 748.4
##
## Number of Fisher Scoring iterations: 4
```

```
# helper function to convert log-odds to probabilities
logit2prob <- function(x){
  exp(x) / (1 + exp(x))
}
```

```
# split the data into training and validation sets
titanic_split <- initial_split(data = titanic,
                               prop = 0.5)

# fit model to training data
train_model <- glm(Survived ~ Age * Sex,
                   data = training(titanic_split),
                   family = binomial); broom::tidy(train_model)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    0.720     0.451      1.59  0.111
```

```
## 2 Age           0.0191    0.0155      1.23  0.218
## 3 Sexmale      -1.39      0.586      -2.37  0.0179
## 4 Age:Sexmale  -0.0389    0.0195     -1.99  0.0467
```

```
# calculate predictions using validation set
x_test_accuracy <- augment(train_model,
                           newdata = testing(titanic_split)) %>%
  as_tibble() %>%
  mutate(.prob = logit2prob(.fitted),
         .pred = factor(round(.prob)))

# calculate test accuracy rate from yardstick
accuracy(x_test_accuracy,
         truth = Survived,
         estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.784
```

Let's jump into CV a bit more, starting with LOOCV.

LOOCV

LOOCV in regression

```
library(tidyverse)
library(ISLR)
library(broom)
library(rsample)
library(rcfss)
library(yardstick)

loocv_data <- loo_cv(Auto)

loocv_data %>%
  names()
```

```
## [1] "splits" "id"
```

```
first_resample <- loocv_data$splits[[1]]
first_resample
```

```
## <Analysis/Assess/Total>
## <391/1/392>
```

```
# but first, note training() and analysis() from resample contain same information; just different form
first_resample %>%
  analysis() %>% # data frame version
  head(n = 5)
```

```
## # A tibble: 5 x 9
##     mpg cylinders displacement horsepower weight acceleration  year origin name
##   <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <fct>
## 1    18         8          307        130   3504           12    70      1 chev~
```

```
## 2    15          8          350         165   3693          11.5     70       1 buic~
## 3    18          8          318         150   3436          11       70       1 plym~
## 4    16          8          304         150   3433          12       70       1 amc ~
## 5    17          8          302         140   3449          10.5     70       1 ford~
```

```
first_resample %>%
  training() %>%
  head(n = 5)
```

```
## # A tibble: 5 x 9
##      mpg cylinders displacement horsepower weight acceleration  year origin name
##    <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <fct>
## 1    18          8          307        130   3504          12       70       1 chev~
## 2    15          8          350        165   3693          11.5     70       1 buic~
## 3    18          8          318        150   3436          11       70       1 plym~
## 4    16          8          304        150   3433          12       70       1 amc ~
## 5    17          8          302        140   3449          10.5     70       1 ford~
```

```
# same with assessment() and testing()
first_resample %>%
  assessment() %>%
  head(n = 5)
```

```
## # A tibble: 1 x 9
##      mpg cylinders displacement horsepower weight acceleration  year origin name
##    <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <fct>
## 1  18.5          8          360        150   3940           13    79       1 chry~
```

```
first_resample %>%
  testing() %>%
  head(n = 5)
```

```
## # A tibble: 1 x 9
##      mpg cylinders displacement horsepower weight acceleration  year origin name
##    <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl> <fct>
## 1  18.5          8          360        150   3940           13    79       1 chry~
```

```
holdout_results <- function(splits) {
  mod <- glm(mpg ~ horsepower, data = analysis(splits))

  res <- augment(mod, newdata = assessment(splits)) %>%
    mse(truth = mpg, estimate = .fitted)

  res
}
```

This function works also for a single resample:

```
holdout_results(loocv_data$splits[[1]])
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        5.08
```

```
loocv_data_poly1 <- loocv_data %>%
  mutate(results = map(splits, holdout_results)) %>%
  unnest(results) %>%
```

```
  spread(.metric, .estimate)

loocv_data_poly1 %>%
  head(n = 5)
```

```
## # A tibble: 5 x 4
##   splits          id        .estimator    mse
##   <list>          <chr>     <chr>       <dbl>
## 1 <split [391/1]> Resample1 standard    5.08
## 2 <split [391/1]> Resample2 standard   49.9
## 3 <split [391/1]> Resample3 standard   44.1
## 4 <split [391/1]> Resample4 standard   25.3
## 5 <split [391/1]> Resample5 standard    0.538
```

```
loocv_data_poly1 %>%
  summarize(mse = mean(mse))
```

```
## # A tibble: 1 x 1
##     mse
##   <dbl>
## 1  24.2
```

```
# modified function to estimate model with varying polynomial order
holdout_results <- function(splits, i) {
  mod <- glm(mpg ~ poly(horsepower, i, raw = TRUE),
             data = analysis(splits))

  res <- augment(mod, newdata = assessment(splits)) %>%
    mse(truth = mpg, estimate = .fitted)
  res
}

# function to return MSE for a specific polynomial term
poly_mse <- function(i, loocv_data){
  loocv_mod <- loocv_data %>%
    mutate(results = map(splits, holdout_results, i)) %>%
    unnest(results) %>%
    spread(.metric, .estimate)

  mean(loocv_mod$mse)
}

library(tictoc)

{ # wrap and time
tic()
cv_mse <- tibble(terms = seq(from = 1, to = 5),
                 mse_loocv = map_dbl(terms, poly_mse, loocv_data))
toc()
}
```

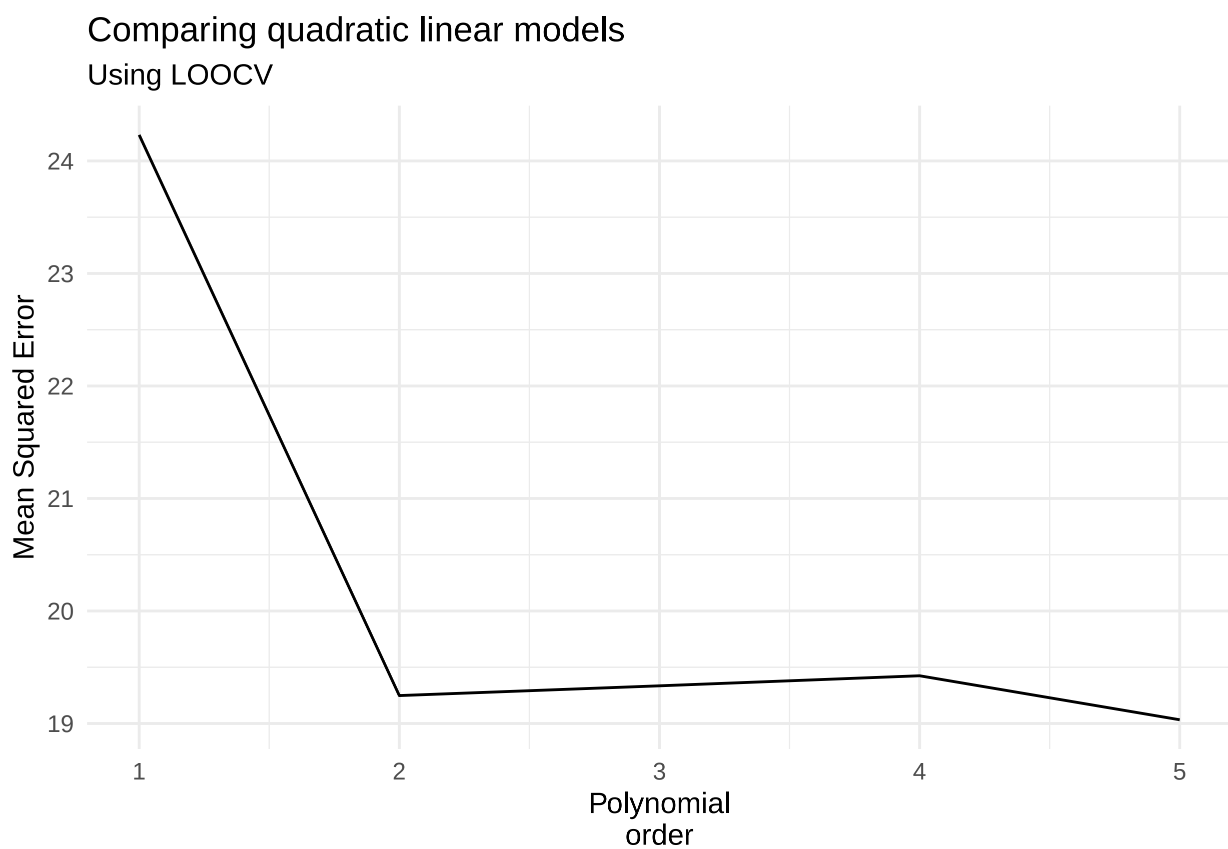
```
## 36.76 sec elapsed
```

```
cv_mse
```

```
## # A tibble: 5 x 2
##   terms mse_loocv
```

```
##    <int>      <dbl>
## 1      1       24.2
## 2      2       19.2
## 3      3       19.3
## 4      4       19.4
## 5      5       19.0
```

```
ggplot(cv_mse, aes(terms, mse_loocv)) +
  geom_line() +
  labs(title = "Comparing quadratic linear models",
       subtitle = "Using LOOCV",
       x = "Polynomial\norder",
       y = "Mean Squared Error") +
  theme_minimal()
```



LOOCV in classification

Let's verify the error rate of our interactive terms model for the Titanic data set:

```
#library(titanic) # re-load if a new session

titanic <- as_tibble(titanic_train) %>%
  mutate(Survived = factor(Survived))

# Now, write a function to generate assessment statistics for titanic model
holdout_results <- function(splits) {
  mod <- glm(Survived ~ Age * Sex, data = analysis(splits),
             family = binomial)
```

```r
  res <- augment(mod, newdata = assessment(splits)) %>%
    as_tibble() %>%
    mutate(.prob = logit2prob(.fitted),
           .pred = round(.prob))
  res
}

titanic_loocv <- loo_cv(titanic) %>%
  mutate(results = map(splits, holdout_results)) %>%
  unnest(results) %>%
  mutate(.pred = factor(.pred)) %>%
  group_by(id) %>%
  accuracy(truth = Survived, estimate = .pred)

1 - mean(titanic_loocv$.estimate, na.rm = TRUE)
```

```
## [1] 0.219888
```

k-fold CV

Back to the `Auto` data set and comparing across polynomial orders.

```r
# our helper function to estimate model with varying highest order polynomial
holdout_results <- function(splits, i) {
  mod <- glm(mpg ~ poly(horsepower, i, raw = TRUE), data = analysis(splits))

  holdout <- assessment(splits)

  res <- augment(mod, newdata = holdout) %>%
    mse(truth = mpg, estimate = .fitted)

  res
}

# function to return MSE for a specific fit
poly_mse <- function(i, vfold_data){
  vfold_mod <- vfold_data %>%
    mutate(results = map(splits, holdout_results, i)) %>%
    unnest(results) %>%
    spread(.metric, .estimate)

  mean(vfold_mod$mse)
}

# split Auto into 10 folds
auto_cv10 <- vfold_cv(data = Auto,
                      v = 10)

# as before...
auto_cv10 %>%
  names()
```

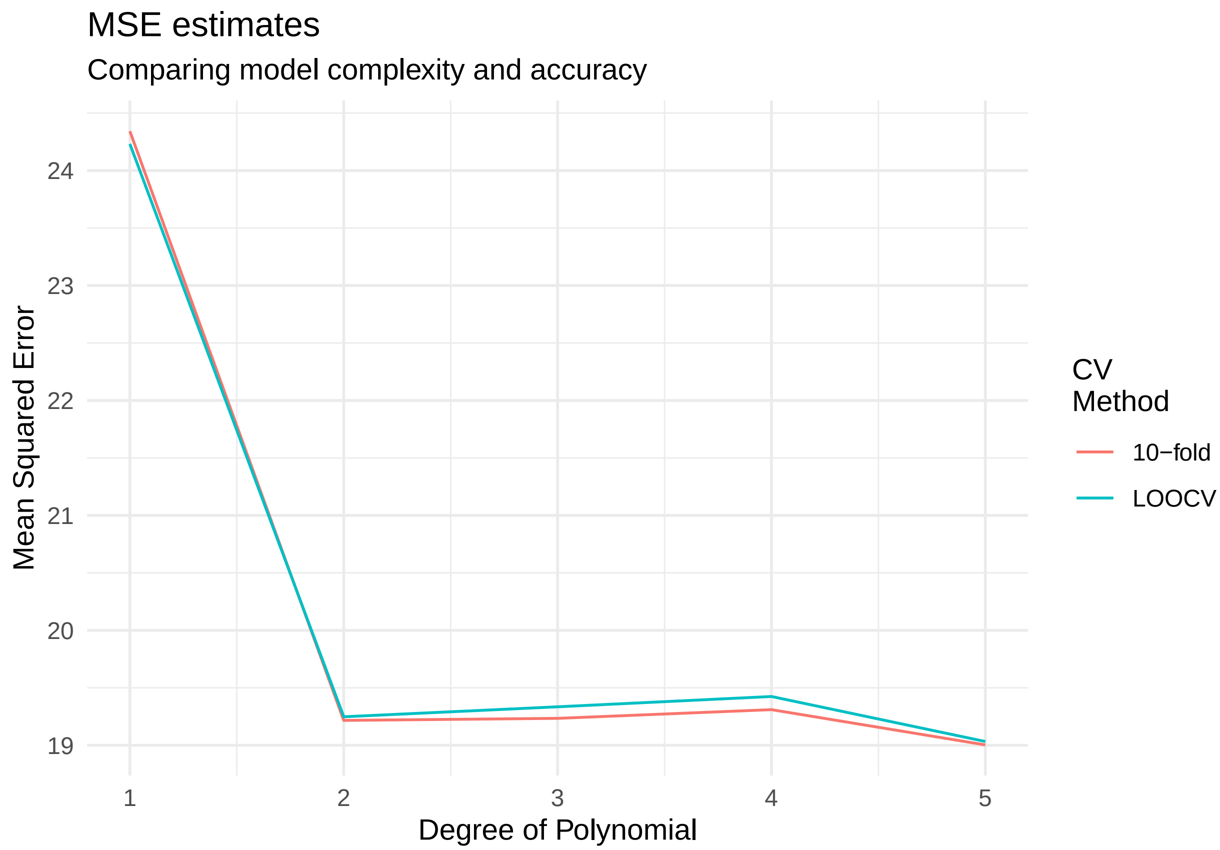
```
## [1] "splits" "id"
```

```r
cv_mse <- tibble(terms = seq(from = 1,
                             to = 5),
                 mse_vfold = map_dbl(terms, poly_mse, auto_cv10))
cv_mse
```

```
## # A tibble: 5 x 2
##   terms mse_vfold
##   <int>     <dbl>
## 1     1      24.3
## 2     2      19.2
## 3     3      19.2
## 4     4      19.3
## 5     5      19.0
```

Looks similar on first glance, but how do these results compare to the LOOCV procedure?

```r
auto_loocv <- loo_cv(Auto)

tibble(terms = seq(from = 1, to = 5), # takes a few seconds, given the mapping
       `10-fold` = map_dbl(terms, poly_mse, auto_cv10),
       LOOCV = map_dbl(terms, poly_mse, auto_loocv)
) %>%
  gather(method, MSE, -terms) %>%
  ggplot(aes(terms, MSE, color = method)) +
  geom_line() +
  labs(title = "MSE estimates",
       subtitle = "Comparing model complexity and accuracy",
       x = "Degree of Polynomial",
       y = "Mean Squared Error",
       color = "CV\nMethod") +
  theme_minimal()
```



On your own

For this section, you will work in small groups of 4-5. *I will create these groups at random.*

IMPORTANT: *Don't forget that this code you're working on here is due at the appropriate Canvas module (in the form of an attachment to a "Discussion" post) prior to 5:00 pm CDT today. You need only submit a **single** file/script to be considered for credit (i.e., this .Rmd with your code inserted below each question). Recall, I don't care whether you got things right. I only care that attempts to each question have been made.*

Return to the Titanic data, and apply 10-fold cross validation to a classification task. As noted before, though we haven't covered classification yet, you should have seen enough in today's session to complete (or at least attempt) the following. I will get you started with the packages and data needed to respond to each prompt below.

```
library(tidyverse)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --

## v dials     0.0.9      v recipes   0.1.15
## v infer     0.5.4      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard()     masks purrr::discard()
## x magrittr::extract()   masks tidyr::extract()
## x dplyr::filter()       masks stats::filter()
## x recipes::fixed()      masks stringr::fixed()
```

```
## x dplyr::lag()          masks stats::lag()
## x magrittr::set_names() masks purrr::set_names()
## x yardstick::spec()     masks readr::spec()
## x recipes::step()       masks stats::step()
```

```
library(titanic)

titanic <- as_tibble(titanic_train) %>%
  mutate(Survived = factor(Survived))
```

1. Use 10-fold cross validation to build and evaluate a logistic regression predicting `Survived` as a function of interacting `Age` and `Sex`. To answer this, you will need to:

a. Build the classifer on the training set(s) across folds
b. Evaluate each classifer using the test set(s) across folds

```
holdout_results <- function(splits) {
  mod <- glm(Survived ~ Age * Sex, data = analysis(splits),
             family = binomial)
  res <- augment(mod, newdata = assessment(splits)) %>%
    as_tibble() %>%
    mutate(.prob = logit2prob(.fitted),
           .pred = round(.prob))
  res
}

titanic_cv10 <- vfold_cv(data = titanic, v = 10) %>%
  mutate(results = map(splits, holdout_results)) %>%
  unnest(results) %>%
  mutate(.pred = factor(.pred)) %>%
  group_by(id) %>%
  accuracy(truth = Survived,
           estimate = .pred)
titanic_cv10
```

```
## # A tibble: 10 x 4
##    id     .metric  .estimator .estimate
##    <chr>  <chr>    <chr>          <dbl>
##  1 Fold01 accuracy binary         0.8
##  2 Fold02 accuracy binary         0.761
##  3 Fold03 accuracy binary         0.790
##  4 Fold04 accuracy binary         0.724
##  5 Fold05 accuracy binary         0.859
##  6 Fold06 accuracy binary         0.824
##  7 Fold07 accuracy binary         0.762
##  8 Fold08 accuracy binary         0.690
##  9 Fold09 accuracy binary         0.824
## 10 Fold10 accuracy binary         0.768
```

2. Calculate the cross-validation *error rate* (not the accuracy rate) from your solution and report it.

```
1 - mean(titanic_cv10$.estimate, na.rm = TRUE)
```

```
## [1] 0.2196239
```

3. Is this similar to the error from using LOOCV earlier in the session? Why or why not, do you think? (offer just a couple thoughts on the patterns – differences and similarities in error, computational speed, etc. – from each approach to resampling in this classification setting).

ANSWER: Not a huge difference from the LOOCV approach, but it takes much less time to compute.