```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------------------

## v ggplot2 3.1.0       v purrr   0.2.5
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.1       v stringr 1.4.0
## v readr   1.1.1       v forcats 0.3.0

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

```r
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------------------------------------------

## v broom     0.5.0     v recipes   0.1.4
## v dials     0.0.2     v rsample   0.0.4
## v infer     0.4.0     v yardstick 0.0.2
## v parsnip   0.0.1

## -- Conflicts -----------------------------------------------------------------------------------
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## x yardstick::tidy() masks rsample::tidy(), recipes::tidy(), broom::tidy()
```

```r
library(ISLR)
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

Initial split

`rsample::initial_split` is an alternative to `modelr::resample_partition`

```
auto_split <- initial_split(Auto, prop = 0.7)
train_data <- training(auto_split)
test_data <- testing(auto_split)
```

A regression problem

```
folds <- vfold_cv(train_data, 5)
```

```
folds %>% mutate(fits = splits %>% map(~lm(mpg ~ poly(horsepower, 3), data = analysis(.)))) %>%
        mutate(mses = map2_dbl(splits, fits, ~ modelr::mse(.y, assessment(.x)))) %>%
        summarize(sum(mses))
```

```
## # A tibble: 1 x 1
##   `sum(mses)`
##         <dbl>
## 1        98.2
```

```
do_cv <- function(folds, d) {
  folds %>% mutate(fits = splits %>% map(~lm(mpg ~ poly(horsepower, d), data = analysis(.)))) %>%
          mutate(mses = map2_dbl(splits, fits, ~ modelr::mse(.y, assessment(.x)))) %>%
          summarize(sum(mses))
}
```

```
map_dfr(1:10, ~do_cv(folds, .), .id = "d")
```

```
## # A tibble: 10 x 2
##    d     `sum(mses)`
##    <chr>       <dbl>
##  1 1           122.
##  2 2            97.2
##  3 3            98.2
##  4 4           106.
##  5 5           104.
##  6 6           121.
##  7 7           103.
##  8 8           101.
##  9 9           782.
## 10 10          245.
```

```
fit <- lm(mpg ~ poly(horsepower, 2), data = train_data)
modelr::mse(fit, test_data)
```

```
## [1] 18.44618
```

I have presented you a very general appaoch to perform cross validation, however, there is a simpler command to obtain the prediction error for linear regression problems.

```
fit <- glm(mpg ~ poly(horsepower, 3), train_data, family = "gaussian")
boot::cv.glm(train_data, fit, K = 5)$delta[1]
```

```
## [1] 19.67176
```

A classification problem

```
head(Smarket)
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction
## 1 2001  0.381 -0.192 -2.624 -1.055  5.010 1.1913  0.959        Up
## 2 2001  0.959  0.381 -0.192 -2.624 -1.055 1.2965  1.032        Up
## 3 2001  1.032  0.959  0.381 -0.192 -2.624 1.4112 -0.623      Down
## 4 2001 -0.623  1.032  0.959  0.381 -0.192 1.2760  0.614        Up
## 5 2001  0.614 -0.623  1.032  0.959  0.381 1.2057  0.213        Up
## 6 2001  0.213  0.614 -0.623  1.032  0.959 1.3491  1.392        Up
```

```
smarket_split <- initial_split(Smarket, prop = 0.7)
train_data <- training(smarket_split)
test_data <- testing(smarket_split)
```

```
folds <- vfold_cv(train_data, 5)
```

```
cutoff <- 0.5
folds %>% mutate(fits = splits %>% map(~glm(Direction ~ Lag1 + Lag2, data = analysis(.), family = "binon
    transmute(acc = map2_dbl(
        splits, fits,
        ~ assessment(.x) %>% modelr::add_predictions(.y) %>% mutate(prob = exp(pred)/ (1 + exp(pred))) %
          mutate(EstDir = factor(ifelse(prob > cutoff, "Up", "Down"), levels = levels(Direction))) %>%
          accuracy(Direction, EstDir) %>% pull(.estimate)
    )) %>%
    summarize(miscls_rate = 1 - mean(acc))
```

```
## # A tibble: 1 x 1
##   miscls_rate
##         <dbl>
## 1       0.517
```

We need to compute the misclassificaiton rate for different `cutoff`. Let's do everything in one shot

```
folds %>% mutate(fits = splits %>% map(~glm(Direction ~ Lag1 + Lag2, data = analysis(.), family = "binon
    transmute(results = map2(
        splits, fits,
        ~ assessment(.x) %>% modelr::add_predictions(.y) %>% mutate(prob = exp(pred)/ (1 + exp(pred)))
    )) %>%
    crossing(cutoff = seq(0.45, 0.54, 0.01)) %>%
```

```r
    mutate(acc = map2_dbl(results, cutoff,
                          ~ .x %>% mutate(EstDir = factor(ifelse(prob > .y, "Up", "Down"), levels = level
                            accuracy(Direction, EstDir) %>% pull(.estimate)
        )) %>%
    group_by(cutoff) %>%
    summarize(miscls_rate = 1 - mean(acc))
```

```
## # A tibble: 10 x 2
##    cutoff miscls_rate
##     <dbl>       <dbl>
##  1   0.45       0.493
##  2   0.46       0.496
##  3   0.47       0.502
##  4   0.48       0.504
##  5   0.49       0.515
##  6   0.5        0.517
##  7   0.51       0.512
##  8   0.52       0.507
##  9   0.53       0.52
## 10   0.54       0.519
```

Similar to the regression problem, there is a simpler command to obtain the prediction error for classificaiton problems. Note: we need the `cost` function to measure misclassification rate.

```r
fit <- glm(Direction ~ Lag1 + Lag2, train_data, family = "binomial")
cost <- function(r, pi) mean(r != (pi > 0.5))
boot::cv.glm(train_data, fit, cost = cost, K = 5)$delta[1]
```

```
## [1] 0.4834286
```

Bootstrap

```r
Auto %>% summarize(r = cor(mpg, horsepower)) %>% pull(r)
```

```
## [1] -0.7784268
```

To get the "classical" confidence interval

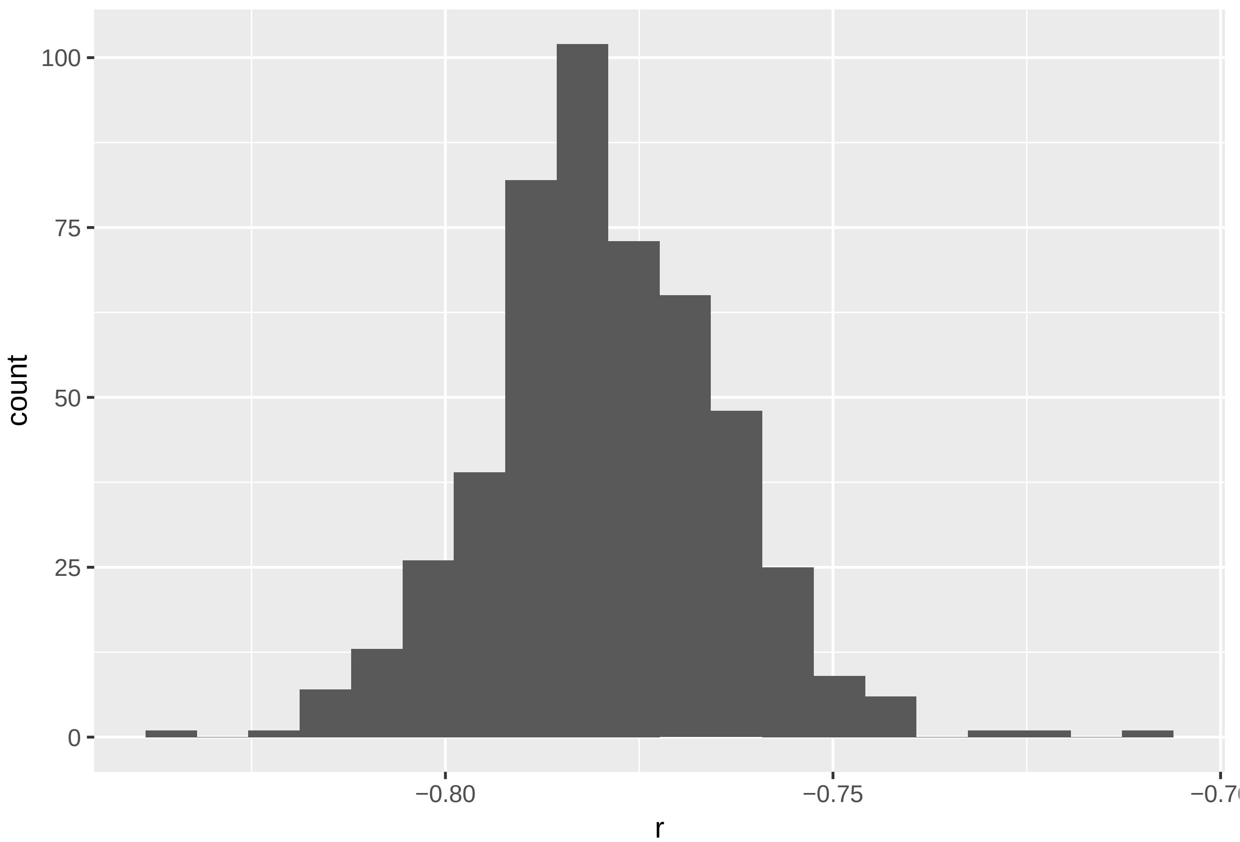
```r
with(Auto, cor.test(mpg, horsepower))
```

```
##
##  Pearson's product-moment correlation
##
## data:  mpg and horsepower
## t = -24.489, df = 390, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.8146631 -0.7361359
## sample estimates:
##        cor
## -0.7784268
```

Use bootstrap to obtain a confidence interval

```
boots <- bootstraps(Auto, times = 500)
boot_sample <- boots %>% transmute(r = map_dbl(splits, ~ with(analysis(.), cor(mpg, horsepower))))
```

```
ggplot(boot_sample) + geom_histogram(aes(x = r), bins = 20)
```



A confidence interval for correlation (bootstrap percentile)

```
boot_sample$r %>% quantile(prob = c(0.025, 0.975))
```

```
##       2.5%      97.5%
## -0.8084046 -0.7503145
```

Another alternative of bootstrap CI (bootstrap t CI)

```
r <- with(Auto, cor(mpg, horsepower))
se <- sd(boot_sample$r)
c(r - 2 * se, r + 2 * se)
```

```
## [1] -0.8092284 -0.7476252
```

Using Bootstrap to calcuate prediction error

```
boots %>% mutate(fits = splits %>% map(~lm(mpg ~ poly(horsepower, 3), data = analysis(.)))) %>%
        mutate(mses = map2_dbl(splits, fits, ~ modelr::mse(.y, assessment(.x)))) %>%
        summarize(sum(mses))
```

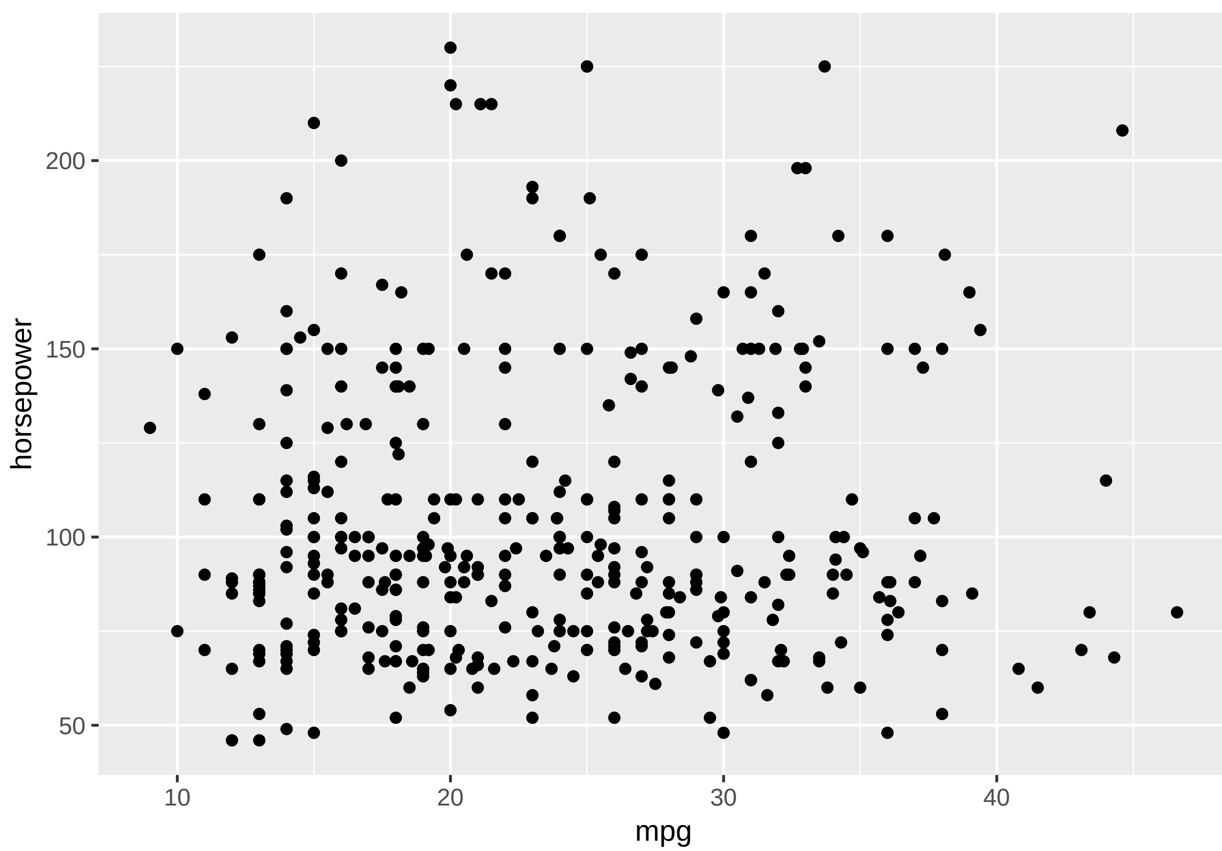
```
## # A tibble: 1 x 1
##   `sum(mses)`
##         <dbl>
## 1       9731.
```

Permutation tests

Recall that the sample correlation between `mpg` and `horsepower` is around -0.78.

```
perms <- modelr::permute(Auto, 1000, mpg)
perm_sample <- perms %>% transmute(r = map_dbl(perm, ~with(as_tibble(.), cor(mpg, horsepower))))
```

```
# ggplot(Auto) + geom_point(aes(mpg, horsepower))
ggplot(as_tibble(perms$perm[[2]])) + geom_point(aes(mpg, horsepower))
```



```
ggplot(perm_sample) + geom_histogram(aes(x = r), bins = 20)
```