Comprehension Check: Linear Regression

Q1

2/2 points (graded)

Create a data set using the following code:

```
set.seed(1) # set.seed(1, sample.kind="Rounding") if using R 3.6 or later
n <- 100
Sigma <- 9*matrix(c(1.0, 0.5, 0.5, 1.0), 2, 2)
dat <- MASS::mvrnorm(n = 100, c(69, 69), Sigma) %>%
      data.frame() %>% setNames(c("x", "y"))
```

We will build 100 linear models using the data above and calculate the mean and standard deviation of the combined models. First, set the seed to 1 again (make sure to use `sample.kind="Rounding"` if your R is version 3.6 or later). Then, within a `replicate` loop, (1) partition the dataset into test and training sets with `p=0.5` and using `dat$y` to generate your indices, (2) train a linear model predicting `y` from `x`, (3) generate predictions on the test set, and (4) calculate the RMSE of that model. Then, report the mean and standard deviation (SD) of the RMSEs from all 100 models.

Report all answers to at least 3 significant digits.

Mean:

2.488661 ✔ **Answer:** 2.49

2.488661

Standard deviation (SD):

0.1243952 ✔ **Answer:** 0.124

0.1243952

Explanation

The following code can be used:

```
set.seed(1)    # if R 3.6 or later, set.seed(1, sample.kind="Rounding")
rmse <- replicate(100, {
        test_index <- createDataPartition(dat$y, times = 1, p = 0.5, list = FALSE)
        train_set <- dat %>% slice(-test_index)
        test_set <- dat %>% slice(test_index)
        fit <- lm(y ~ x, data = train_set)
        y_hat <- predict(fit, newdata = test_set)
        sqrt(mean((y_hat-test_set$y)^2))
})

mean(rmse)
sd(rmse)
```

Submit You have used 2 of 10 attempts

Answers are displayed within the problem

Q2

2.0/2.0 points (graded)

Now we will repeat the exercise above but using larger datasets. Write a function that takes a size `n`, then (1) builds a dataset using the code provided in Q1 but with `n` observations instead of 100 and without the `set.seed(1)`, (2) runs the `replicate` loop that you wrote to answer Q1, which builds 100 linear models and returns a vector of RMSEs, and (3) calculates the mean and standard deviation.

Set the seed to 1 (if using R 3.6 or later, use the argument `sample.kind="Rounding"`) and then use `sapply` or `map` to apply your new function to `n <- c(100, 500, 1000, 5000, 10000)`.

Hint: You only need to set the seed once before running your function; do not set a seed within your function. Also be sure to use `sapply` or `map` as you will get different answers running the simulations individually due to setting the seed.

Mean, 100:

2.4977540 ✔ **Answer:** 2.498

2.4977540

SD, 100:

0.1180821 ✔ **Answer:** 0.118

0.1180821

Mean, 500:

2.72095125 ✔ **Answer:** 2.72

2.72095125

SD, 500:

0.08002108 ✔ **Answer:** 0.08

0.08002108

Mean, 1000:

2.55554451 ✔ **Answer:** 2.5555

2.55554451

SD, 1000:

0.04560258 ✔ **Answer:** 0.0456

0.04560258

Mean, 5000:

2.62482800 ✔ **Answer:** 2.6248

2.62482800

SD, 5000:

0.02309673 ✔ **Answer:** 0.0231

0.02309673

Mean, 10000:

2.61844227 ✔ **Answer:** 2.6184

2.61844227

SD, 10000:

0.01689205 ✔ **Answer:** 0.0169

0.01689205

Explanation

The code below can be used to do this calculation:

```
set.seed(1)    # if R 3.6 or later, set.seed(1, sample.kind="Rounding")
n <- c(100, 500, 1000, 5000, 10000)
res <- sapply(n, function(n){
        Sigma <- 9*matrix(c(1.0, 0.5, 0.5, 1.0), 2, 2)
        dat <- MASS::mvrnorm(n, c(69, 69), Sigma) %>%
                data.frame() %>% setNames(c("x", "y"))
        rmse <- replicate(100, {
                test_index <- createDataPartition(dat$y, times = 1, p = 0.5, list =
FALSE)
                train_set <- dat %>% slice(-test_index)
                test_set <- dat %>% slice(test_index)
                fit <- lm(y ~ x, data = train_set)
                y_hat <- predict(fit, newdata = test_set)
                sqrt(mean((y_hat-test_set$y)^2))
        })
        c(avg = mean(rmse), sd = sd(rmse))
})

res
```

Submit You have used 1 of 10 attempts

Answers are displayed within the problem

Q3

1/1 point (graded)

What happens to the RMSE as the size of the dataset becomes larger?

- (●) On average, the RMSE does not change much as `n` gets larger, but the variability of the RMSE decreases.
- ( ) Because of the law of large numbers the RMSE decreases; more data means more precise estimates.

- `n = 10000` is not sufficiently large. To see a decrease in the RMSE we would need to make it larger.
- The RMSE is not a random variable.

✔

Submit You have used 1 of 2 attempts

Answers are displayed within the problem

Q4

2/2 points (graded)

Now repeat the exercise from Q1, this time making the correlation between `x` and `y` larger, as in the following code:

```
set.seed(1)
n <- 100
Sigma <- 9*matrix(c(1.0, 0.95, 0.95, 1.0), 2, 2)
dat <- MASS::mvrnorm(n = 100, c(69, 69), Sigma) %>%
        data.frame() %>% setNames(c("x", "y"))
```

Note what happens to RMSE - set the seed to 1 as before.

Mean:

0.9099808 ✔ **Answer:** 0.91

0.9099808

SD:

0.06244347 ✔ **Answer:** 0.0624

0.06244347

Explanation

The same code as in Q1 can be used:

```
set.seed(1)
rmse <- replicate(100, {
        test_index <- createDataPartition(dat$y, times = 1, p = 0.5, list = FALSE)
        train_set <- dat %>% slice(-test_index)
        test_set <- dat %>% slice(test_index)
        fit <- lm(y ~ x, data = train_set)
        y_hat <- predict(fit, newdata = test_set)
        sqrt(mean((y_hat-test_set$y)^2))
})

mean(rmse)
sd(rmse)
```

Submit You have used 1 of 10 attempts

Answers are displayed within the problem

Q5

1/1 point (graded)

Which of the following best explains why the RMSE in question 4 is so much lower than the RMSE in question 1?

- It is just luck. If we do it again, it will be larger.
- The central limit theorem tells us that the RMSE is normal.
- When we increase the correlation between `x` and `y`, `x` has more predictive power and thus provides a better estimate of `y`.
- These are both examples of regression so the RMSE has to be the same.

✔

Explanation

The correlation between `x` and `y` has a much bigger effect on RMSE than `n`. Large `n` simply provides us with more precise estimates of the linear model coefficients.

Submit You have used 1 of 2 attempts

Answers are displayed within the problem

Q6

1/1 point (graded)

Create a data set using the following code.

```
set.seed(1)
Sigma <- matrix(c(1.0, 0.75, 0.75, 0.75, 1.0, 0.25, 0.75, 0.25, 1.0), 3, 3)
dat <- MASS::mvrnorm(n = 100, c(0, 0, 0), Sigma) %>%
        data.frame() %>% setNames(c("y", "x_1", "x_2"))
```

Note that `y` is correlated with both `x_1` and `x_2` but the two predictors are independent of each other, as seen by `cor(dat)`.

Set the seed to 1, then use the **caret** package to partition into a test and training set of equal size. Compare the RMSE when using just `x_1`, just `x_2` and both `x_1` and `x_2`. Train a linear model for each.

Which of the three models performs the best (has the lowest RMSE)?

- ○ `x_1`
- ○ `x_2`
- ◉ `x_1` and `x_2`

✔

Explanation

The linear model with both predictors performs the best, as seen using the following code:

```
set.seed(1)
test_index <- createDataPartition(dat$y, times = 1, p = 0.5, list = FALSE)
train_set <- dat %>% slice(-test_index)
test_set <- dat %>% slice(test_index)

fit <- lm(y ~ x_1, data = train_set)
y_hat <- predict(fit, newdata = test_set)
sqrt(mean((y_hat-test_set$y)^2))

fit <- lm(y ~ x_2, data = train_set)
y_hat <- predict(fit, newdata = test_set)
sqrt(mean((y_hat-test_set$y)^2))

fit <- lm(y ~ x_1 + x_2, data = train_set)
y_hat <- predict(fit, newdata = test_set)
sqrt(mean((y_hat-test_set$y)^2))
```

Submit You have used 1 of 1 attempt

Answers are displayed within the problem

Q7

1/1 point (graded)
Report the lowest RMSE of the three models tested in Q6.

0.3070962 ✔ **Answer:** 0.307

0.3070962

Explanation
The lowest RMSE is for the model that includes `x_1` and `x_2` as predictors.

Submit You have used 1 of 10 attempts

Answers are displayed within the problem

Q8

1/1 point (graded)
Repeat the exercise from Q6 but now create an example in which `x_1` and `x_2` are highly correlated.

```
set.seed(1)
Sigma <- matrix(c(1.0, 0.75, 0.75, 0.75, 1.0, 0.95, 0.75, 0.95, 1.0), 3, 3)
dat <- MASS::mvrnorm(n = 100, c(0, 0, 0), Sigma) %>%
        data.frame() %>% setNames(c("y", "x_1", "x_2"))
```

Set the seed to 1, then use the **caret** package to partition into a test and training set of equal size. Compare the RMSE when using just `x_1`, just `x_2`, and both `x_1` and `x_2`.

Compare the results from Q6 and Q8. What can you conclude?

- Unless we include all predictors we have no predictive power.
- Adding extra predictors improves RMSE regardless of whether the added predictors are correlated with other predictors or not.

- Adding extra predictors results in over fitting.

- Adding extra predictors can improve RMSE substantially, but not when the added predictors are highly correlated with other predictors.

✔

Explanation

The following code will allow you to repeat the exercise in Q6 with predictors that are highly correlated:

```
set.seed(1)
test_index <- createDataPartition(dat$y, times = 1, p = 0.5, list = FALSE)
train_set <- dat %>% slice(-test_index)
test_set <- dat %>% slice(test_index)

fit <- lm(y ~ x_1, data = train_set)
y_hat <- predict(fit, newdata = test_set)
sqrt(mean((y_hat-test_set$y)^2))

fit <- lm(y ~ x_2, data = train_set)
y_hat <- predict(fit, newdata = test_set)
sqrt(mean((y_hat-test_set$y)^2))

fit <- lm(y ~ x_1 + x_2, data = train_set)
y_hat <- predict(fit, newdata = test_set)
sqrt(mean((y_hat-test_set$y)^2))
```

When the predictors are highly correlated with each other, adding addtional predictors does not improve the model substantially, thus RMSE stays roughly the same.

Submit

You have used 1 of 2 attempts

ℹ Answers are displayed within the problem

Ask your questions or make your comments about Linear Regression here! **Remember, one of the best ways to reinforce your own learning is by explaining something to someone else, so we encourage you to answer each other's questions (without giving away the answers, of course).**

Some reminders:

- Search the discussion board before posting to see if someone else has asked the same thing before asking a new question

- Please be specific in the title and body of your post regarding which question you're asking about to facilitate answering your question.
- Posting snippets of code is okay, but posting full code solutions is not.
- If you do post snippets of code, please format it as code for readability. If you're not sure how to do this, there are instructions in a pinned post in the "general" discussion forum.

Discussion: Linear Regression

Show Discussion

Topic: Section 3: Linear Regression for Prediction, Smoothing, and Working with Matrices / 3.1.1: Linear Regression