Making predictions using real data in R

Consider again our marketing data:

The following dataset containts measurements related to the impact of three advertising medias on sales of a product, P. The variables are:

- `youtube` : the advertising budget allocated to YouTube. Measured in thousands of dollars;
- `facebook` : the advertising budget allocated to Facebook. Measured in thousands of dollars; and
- `newspaper` : the advertising budget allocated to a local newspaper. Measured in thousands of dollars.
- `sales` : the value in the i^{th} row of the sales column is a measurement of the sales (in thousands of units) for product P for company i.

The advertising data treat "a company selling product P" as the statistical unit, and "all companies selling product P" as the population. We assume that the $n = 200$ companies in the dataset were chosen at random from the population (a strong assumption!).

First, we'll read in the data and split it into a training set and a testing set.

In [9]:

```
library(RCurl) #a package that includes the function getURL(), which allows for
library(ggplot2)
url = getURL("https://raw.githubusercontent.com/bzaharatos/-Statistical-Modeling
marketing = read.csv(text = url, sep = "")
```

In [10]:

```
set.seed(17711) #set the random number generator seed.
n = floor(0.8 * nrow(marketing)) #find the number corresponding to 80% of the da
index = sample(seq_len(nrow(marketing)), size = n) #randomly sample indicies to

train = marketing[index, ] #set the training set to be the randomly sampled rows
test = marketing[-index, ] #set the testing set to be the remaining rows
dim(test) #check the dimensions
dim(train) #check the dimensions
```

40 · 4

160 · 4

Next, we'll fit our linear model to the training data, with `sales` as the response, and `facebook`, `youtube` and `newspaper` as predictors.

In [11]:

```
lm_marketing = lm(sales ~ youtube + facebook + newspaper, data = train)
summary(lm_marketing)
```

```
Call:
```

```
lm(formula = sales ~ youtube + facebook + newspaper, data = train)

Residuals:
   Min     1Q Median     3Q    Max 
-6.217 -1.173  0.236  1.516  3.397 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.563341   0.385414   9.245   <2e-16 ***
youtube      0.046087   0.001467  31.408   <2e-16 ***
facebook     0.192086   0.008809  21.805   <2e-16 ***
newspaper   -0.005099   0.006027  -0.846    0.399    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.889 on 156 degrees of freedom
Multiple R-squared:  0.9115,	Adjusted R-squared:  0.9098 
F-statistic: 535.6 on 3 and 156 DF,  p-value: < 2.2e-16
```

Now, let's extract some values from the testing set, and use them to make predictions.

In [12]:
```
set.seed(1101)
index = sample(seq_len(nrow(test)), size = 5)
test[index,]
star = test[index,]
round(predict(lm_marketing, new = star, interval = "prediction"),2)
```

A data.frame: 5 × 4

	youtube	facebook	newspaper	sales
	<dbl>	<dbl>	<dbl>	<dbl>
63	287.16	18.60	32.76	18.84
115	93.84	56.16	41.40	17.52
169	258.48	28.32	69.12	20.52
121	169.56	32.16	55.44	18.60
21	262.08	33.24	64.08	21.60

A matrix: 5 × 3 of type dbl

	fit	lwr	upr
63	20.20	16.44	23.96
115	18.46	14.68	22.25
169	20.56	16.80	24.33
121	17.27	13.52	21.02
21	21.70	17.94	25.46

We can also make these predictions "by hand".

In [13]:
```
alpha = 0.05
b = coef(lm_marketing) #parameter estimates
dof = length(train$sales) - length(b) #degrees of freedom
rss = sum(resid(lm_marketing)^2) #used to estimate sigma^2
```

```
sig2hat = rss/dof
X = model.matrix(lm_marketing) #used in the standard error calculation
xstar_matrix = data.matrix(star[1:3]) #predictors from subset of test set
y_star_hat = cbind(1, xstar_matrix)%*%b #predicted values

l = y_star_hat[1] - qt(1-alpha/2, dof)* sqrt(sig2hat*
    (cbind(1,t(xstar_matrix[1,]))%*%solve(t(X)%*%X)%*%t(cbind(1,t(xstar_matrix[1
u = y_star_hat[1] + qt(1-alpha/2, dof)* sqrt(sig2hat*
    (cbind(1,t(xstar_matrix[1,]))%*%solve(t(X)%*%X)%*%t(cbind(1,t(xstar_matrix[1
pi = round(c(l,u), 2);
cat("The prediction interval is", pi, ".")
```

```
The prediction interval is 16.44 23.96 .
```

We can interpret this prediction interval as follows: we are 95% confident that, if a new company selling product P entered the market with a YouTube marketing budget of $\$287,160$, a Facebook marketing budget of $\$18,600$, and a newspaper marketing budget of $\$32,760$, they would sell between $16,440$ and $23,960$ units of product P. Let's unpack what we mean by "confidence". A proper interpretation of confidence requires that we imagine the following procedure:

1. fix the predictors in the training data, and resample the response many times;

1. fit the model to each resample of the training data;

1. compute the prediction interval at the same values of the response, namely, `youtube` $= 287.16$, `facebook` $= 18.60$, and `newspaper` $= 32.76$ for each model from 2.

Among these prediction intervals, 95% would cover the true value of the response.

In this case, since we're predicting a value of the response that was recorded and put in the testing set, we know the true values of `sales` to be 18.84, or $18,840$ units. So, our interval covers the true value.

Let's contrast that with a confidence interval for the mean value of sales of product P, given a set of predictors:

In [14]:
```
round(predict(lm_marketing, new = star, interval = "confidence"),2)
```

A matrix: 5 × 3 of type dbl

	fit	lwr	upr
63	20.20	19.74	20.67
115	18.46	17.84	19.09
169	20.56	20.04	21.09
121	17.27	16.91	17.64
21	21.70	21.22	22.18

We can interpret this confidence interval as follows: we are 95% confident that a new company selling product P with a YouTube marketing budget of $\$287,160$, a Facebook marketing budget of $\$18,600$, and a newspaper marketing budget of $\$32,760$, can expect to sell between $19,740$ and $20,670$ units of product P, *on average*.

Notice how much smaller the confidence interval for the mean is when compared to the prediction interval for a new value of the response!

Mean squared prediction error

Now let's move on to the mean squared prediction error as a way of comparing predictive models. First, we'll fit a new, reduced model, without `newspaper`.

In [15]:

```
lm_marketing2 = lm(sales ~ youtube + facebook, data = train)
summary(lm_marketing2)
```

```
Call:
lm(formula = sales ~ youtube + facebook, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.4413 -1.1186  0.2483  1.4107  3.4251 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 3.458149   0.364475   9.488   <2e-16 ***
youtube     0.045969   0.001459  31.498   <2e-16 ***
facebook    0.189898   0.008413  22.571   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.887 on 157 degrees of freedom
Multiple R-squared:  0.9111,	Adjusted R-squared:   0.91 
F-statistic: 804.4 on 2 and 157 DF,  p-value: < 2.2e-16
```

Now let's compare our two models using the MSPE (using the training set).

In [16]:

```
#the full model
#mseTrain = mean(lm_marketing$resid^2); mseTrain
#with(train, sum((sales - predict(lm_marketing))^2))/n
pred = predict(lm_marketing, test);
mseTest = mean((test$sales - pred)^2); mseTest

#the reduced model
pred2 = predict(lm_marketing2, test);
mseTest2 = mean((test$sales - pred2)^2); mseTest2
mseTest < mseTest2
```

6.20283983769489

6.09375874553239

FALSE

The reduced model does better, in terms of the MSPE, than the full model. This is consistent with our t-test result, providing more evidence that the reduced model would be better than the full model.