part of the optimization procedure. Hyperparameters can be tuned by cross-validation discussed in Subsection 3.1.2. We construct a grid of values of (λ, α) (we only have to consider λ if we are performing ridge regression or lasso), compute the cross-validation error for each pair of values of (λ, α), and choose the pair that gives the lowest cross-validation error. Given the optimal hyperparameters, the penalized regression model is fitted to all of the training data and its prediction accuracy is evaluated on the test data. All of these steps can be readily implemented in R, so there is no need to worry about their computational aspects.

Pros and cons of regularization techniques for feature selection. How do regularization techniques compare to the stepwise feature selection techniques in Subsection 3.2.4? As you expect, none of the feature selection methods are universally superior and there are relative merits.

Merits.

1. The `glmnet()` function that implements penalized regression in R requires the binarization of categorical predictors in advance. As we will see in Subsection 3.4.2, this allows us to assess the significance of individual factor levels, not just the significance of the entire categorical predictor.
2. Regularization is computationally more efficient than stepwise selection algorithms. As stated on page 219 of *Introduction to Statistical Learning* (ISLR), the computations required to fit a penalized regression model "simultaneously for all values of λ are almost identical to those for fitting a model using least squares."

Demerits.

1. Regularization techniques may not produce the most interpretable model, especially for ridge regression, for which all features are retained.
2. During the model fitting process, all numeric features are standardized to ensure that they are on the same scale. This makes the interpretation of the coefficient estimates less intuitive than under linear models fitted by ordinary least squares.
3. The `glmnet()` function is restricted in terms of model forms. As suggested in the June 2019 PA exam, the function can accommodate some, but not all of the distributions for GLMs. The gamma model, for instance, is not covered.

3.3 Case Study 1: Fitting a Linear Model in R

In this section we will illustrate how a linear model can be fitted and used to make predictions in R by means of the `Advertising` dataset that is associated with ISLR. Although this dataset is only a simulated one (i.e., the observations are generated by computer algorithms, but are not from real data) and you may have already come across it when studying for Exam SRM, we have decided to make use of (or revisit) this dataset due to the following reasons:

- Its marketing context demonstrates how linear models can be applied to aid decision-making in practical situations.
- It allows us to illustrate many of the modeling concepts such as polynomial regression and interaction effects.

- Your familiarity with it from Exam SRM may make it easier for you to understand the following data analysis. (We will look at datasets *not* from ISLR in later chapters.)

Our treatment, however, differs from that of ISLR in two respects. First, in addition to using the `Advertising` dataset to illustrate the concept of interaction effects, as is done in ISLR, we will also demonstrate how the addition of nonlinear transformations of individual predictors such as higher-power terms can improve model fit and prediction accuracy. Second, unlike in Exam SRM where one of the focuses is on inference (e.g., using hypothesis tests to understand the data-generating mechanism), in Exam PA our main focus is on *prediction*, that is, we will simply use linear models as a tool to predict and we are primarily interested in those that yield accurate predictions rather than those that lend themselves to simple interpretation. Remember that PA stands for "predictive" analytics!

After completing this case study, you should be able to:

- Fit a multiple linear regression model using the `lm()` function and extract useful information from a fitted model using the `summary()` function.
- Generate additional features such as interaction and polynomial variables in a linear model.
- Partition the data into training and test sets using the `createDataPartition()` from the `caret` package.
- Generate predictions on the training set as well as on the test set using the `predict()` function.
- Compare the predictive performance of different linear models.

Variable	Description
X	A variable of row indices
TV	Amount of budget allocated to TV advertising (in thousands of dollars)
radio	Amount of budget allocated to radio advertising (in thousands of dollars)
newspaper	Amount of budget allocated to newspaper advertising (in thousands of dollars)
sales	Amount of sales of a particular product (in thousands of units)

Table 3.2: Data dictionary for the ad data.

Data description. First and foremost, let's run CHUNK 1 to read the `Advertising` dataset[xiii] into R with the resulting data frame abbreviated to `ad` (to save typing!) and print a few rows of the data.

```
# CHUNK 1
ad <- read.csv("Advertising.csv")
head(ad)

##   X    TV radio newspaper sales
## 1 1 230.1  37.8      69.2  22.1
## 2 2  44.5  39.3      45.1  10.4
## 3 3  17.2  45.9      69.3   9.3
## 4 4 151.5  41.3      58.5  18.5
## 5 5 180.8  10.8      58.4  12.9
## 6 6   8.7  48.9      75.0   7.2
```

The data contains $n = 200$ observations of 5 variables described in Table 3.2.

Suppose that we are statistical consultants hired by the company that offers the product. The company is interested in boosting `sales` of the product, but cannot directly do so (that is determined by market demand). Instead, it has the liberty to control the advertising expenditure in each of the three advertising media: `TV`, `radio`, and `newspaper`. If we can construct a linear model that accurately predicts `sales` (the target variable) on the basis of the budgets spent on the three advertising media (the predictors), then such a model can provide the basis for a valuable marketing plan that specifies how much should be spent on the three media to maximize `sales`, a business issue of great interest to the company.

EXAM NOTE

Projects in Exam PA often have an underlying business problem driven by practical considerations (e.g., to identify the key factors affecting...) to make them more realistic, interesting, and meaningful. Our job is to use our predictive analytic toolkit to provide recommendations and solutions that are statistically sound and practically useful.

Before proceeding, let us delete the `X` variable, which is simply a variable of row indices and contributes no useful information for understanding `sales`, by assigning it to the `NULL` symbol

[xiii] The original CSV file can be downloaded from `http://faculty.marshall.usc.edu/gareth-james/ISL/Advertising.csv`.

(recall how to delete variables as discussed on page 42). Remember to append the name of the data frame, ad in this case, or else the X variable would remain in the data (try to use the command X <- NULL and see what, if anything, happens).

```
# CHUNK 1 (Cont.)
# Remember to append the name of the dataset ("ad")
ad$X <- NULL
```

Data exploration. In any predictive analytic endeavors, the first step of our analysis is to gain a general understanding of the key characteristics of the variables in our data. To this end, we can use the numeric and graphical tools introduced in Section 2.2. For example, let's apply the `summary()` function to the ad data in CHUNK 2. Doing so returns a summary of each of the four numeric variables in the data.

```
# CHUNK 2
summary(ad)
```

```
##        TV              radio          newspaper          sales
##  Min.   :  0.70   Min.   : 0.000   Min.   :  0.30   Min.   : 1.60
##  1st Qu.: 74.38   1st Qu.: 9.975   1st Qu.: 12.75   1st Qu.:10.38
##  Median :149.75   Median :22.900   Median : 25.75   Median :12.90
##  Mean   :147.04   Mean   :23.264   Mean   : 30.55   Mean   :14.02
##  3rd Qu.:218.82   3rd Qu.:36.525   3rd Qu.: 45.10   3rd Qu.:17.40
##  Max.   :296.40   Max.   :49.600   Max.   :114.00   Max.   :27.00
```

For TV, radio, and sales, their mean is pretty close to their median, showing that their distributions are not far from symmetric. The mean of newspaper is higher than its median more remarkably, which indicates a fair amount of right skewness in its distribution.

We can also look at graphical displays of the four variables to gain a more intuitive understanding of their distributions. In CHUNK 3, we use the ggplot2 package to produce histograms of the four variables (box plots can also be used); see Figure 3.3.1. We can see that the distribution of sales, the target variable, is rather symmetric and bell-shaped, which supports the use of a linear model (with normal random errors). The distributions of the three advertising budgets are not as regular, but this will not derail our analysis based on linear models. Also, the histograms of the three media do not suggest any particularly useful transformations.

```
# CHUNK 3
library(ggplot2)
library(gridExtra)
p1 <- ggplot(ad, aes(x = sales)) + geom_histogram()
p2 <- ggplot(ad, aes(x = TV)) + geom_histogram()
p3 <- ggplot(ad, aes(x = radio)) + geom_histogram()
p4 <- ggplot(ad, aes(x = newspaper)) + geom_histogram()
grid.arrange(p1, p2, p3, p4, ncol = 2)
```

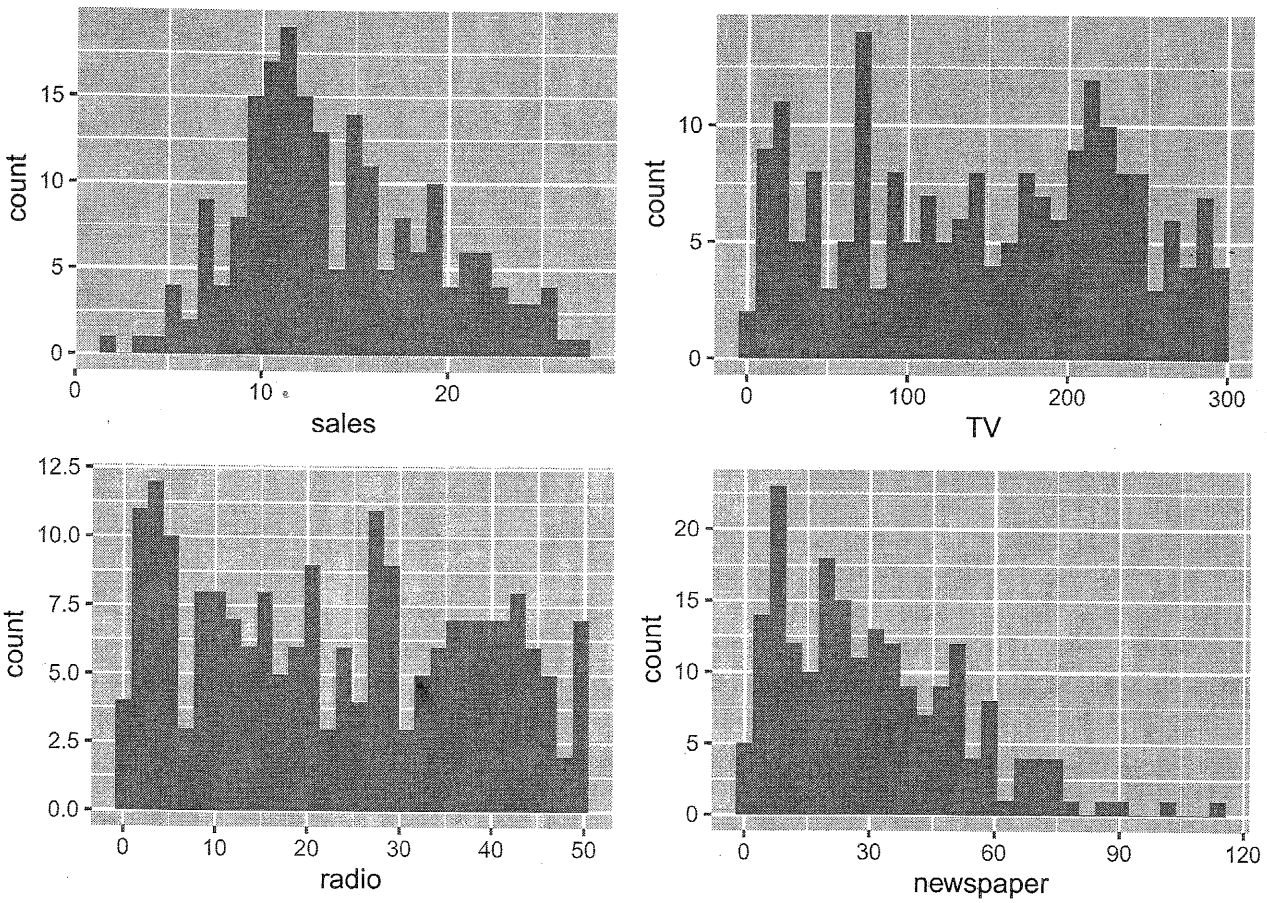


Figure 3.3.1: Histograms of the four variables in the ad data.

Now let's turn to bivariate data exploration. As pointed out in Subsection 2.2.2, very often the pairwise relationship between the target variable and each predictor provides useful input for building a predictive model. Since the target variable and predictors in the ad data are all numeric in nature, scatterplots will serve our purpose well. In CHUNK 4, we use ggplot2 to build the scatterplot for sales against each advertising medium.[xiv] In each plot, the fitted regression line is shown in blue; see Figure 3.3.2.

```
# CHUNK 4
p1 <- ggplot(ad, aes(x = TV, y = sales)) +
  geom_point() + geom_smooth(method = "lm", se = FALSE)
p2 <- ggplot(ad, aes(x = radio, y = sales)) +
  geom_point() + geom_smooth(method = "lm", se = FALSE)
p3 <- ggplot(ad, aes(x = newspaper, y = sales)) +
  geom_point() + geom_smooth(method = "lm", se = FALSE)
grid.arrange(p1, p2, p3, ncol = 2)
```

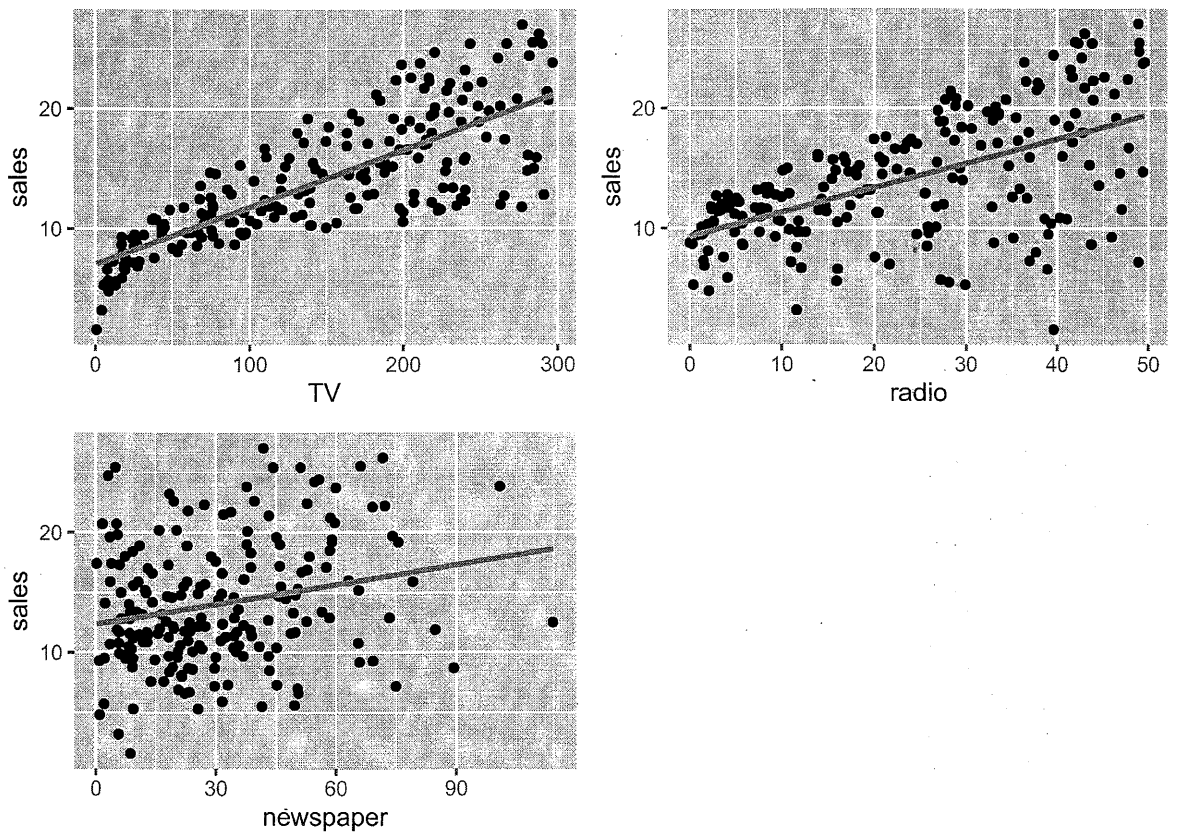


Figure 3.3.2: Scatterplots of sales against TV, radio, and newspaper in the ad data.

[xiv]There are other ways more efficient than using ggplot2 to construct the scatterplot between the target variable and each predictor manually. For example, we can use the pairs() function to create a matrix of scatterplots, called a *scatterplot matrix*, for every pair of numeric variables. Although this function is not covered in the PA e-learning modules, it provides us with a helicopter view of the directional impact of each numeric predictor on the target variable.

The scatterplots show clearly that `sales` have a positive linear relationship with both `TV` and `radio`. This is intuitive as increasing the spending on `TV` and `radio` will naturally lead to an increase in `sales`. The relationship between `sales` and `newspaper` is not as clear. There is a mild increasing trend, but the amount of fluctuation of the data points about the fitted line is substantial. Later in this section, we will use R output to show that `newspaper` is not a real driver of `sales`.

3.3.1 Simple Linear Regression

Let's start with something simple: A "simple" linear regression model, which is a linear model with only one predictor. The generic model equation is $y = \beta_0 + \beta_1 x + \varepsilon$. Since we have just one predictor and one target variable, the bivariate nature of the setting allows us to visualize the results of a linear model using a two-dimensional plot and gain valuable insights. For the purpose of illustration, in this subsection we will study the simple linear model regressing `sales` only on `TV`—among the three advertising media, `TV` seems to have the strongest impact on `sales`. After completing this subsection, you should be familiar with the basic commands in R for fitting and extracting information from linear models.

Workhorse: `lm()` function. The basic function for fitting linear models in R is the `lm()` function (standing for "linear models"). Its syntax in its simplest form is

```
<linear_model> <- lm(<formula>, <data>)
```

where:

- `<formula>` is a symbolic description of the linear model to be fitted using R's formula syntax. The target variable is preceded by a tilde (`~`), which is followed by the predictor(s), so the formula `y ~ x` instructs R to fit a simple linear model regressing the target variable `y` on the single predictor `x`. By default, an intercept is included. Note that `y ~ x` is not the same as `x ~ y`—never mix up the predictor and target variable!
- `<data>` specifies the data frame that hosts the variables used in the linear model.

Let's run the code in CHUNK 5 to fit a simple linear regression model regressing `sales` on `TV`, save the fitted model for ease of further manipulation as an object named `model.slr` (to distinguish it from more complex linear models we will fit later in this section), and show what we can get by typing the name of an `lm` object.

```
# CHUNK 5
model.slr <- lm(sales ~ TV, data = ad)
model.slr

##
## Call:
## lm(formula = sales ~ TV, data = ad)
##
## Coefficients:
## (Intercept)           TV
##     7.03259      0.04754
```

Typing the name of an `lm` object will return only the least squares coefficient estimates of the fitted linear model. In the `ad` data, the estimated coefficients are $\hat{\beta}_0 = 7.03259$ and $\hat{\beta}_1 = 0.04754$, and the fitted regression line is

$$\widehat{\texttt{sales}} = 7.03259 + 0.04754 \times \texttt{TV}.$$

To make sense of the fitted regression equation, we can say that:

- When no amount is spent on TV advertising (i.e., `TV` equals 0), the *expected* amount of the company's sales is estimated to be $1,000\hat{\beta}_0 = 7,032.59$ units (remember that the variable `sales` is in thousands of units).
- When an additional \$1,000 spent on TV advertising (i.e., `TV` increases by 1; remember that `TV` is also in thousands of dollars), the *expected* increase in the company's sales is estimated to be $1,000\hat{\beta}_1 = 47.54$ units.

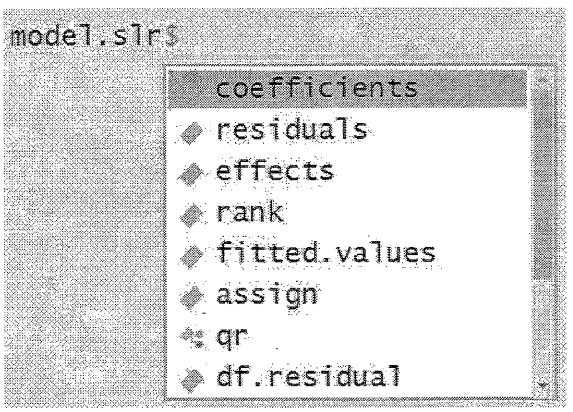
You should note that in some situations the interpretation based on the value of $\hat{\beta}_0$ may not make practical sense. For example, in the `Galton` dataset in Slide 5 of PA Module 6, the predictor represents the height of a parent and the target variable represents the height of his/her child. In this case, the value of $\hat{\beta}_0$ provides an estimate of the expected height of a child when his/her parent has zero height. However, the height of a person cannot be zero!

EXAM NOTE

In Exam PA, you are often asked to interpret the results of your model. When you interpret the coefficient estimates, make sure to be precise and concise. Do not miss seemingly unimportant terms such as "expected."

To learn more about a fitted linear model. More often than not, we are interested in aspects of a fitted linear model beyond the values of the coefficient estimates. Additional information about a linear model can be accessed in two ways: (some information can be accessed in both ways)

1. *Using the fact that an `lm` object is a list:* An object returned by the `lm()` function is technically a list (recall that lists are discussed in Subsection 1.2.4), whose components can be accessed via the dollar sign (`$`) notation. Try to type `model.slr$` and press `Tab` to see what components are stored.



Function	Action
`summary()`	Displays a detailed analysis of the fitted model.
`coefficients()` (or simply `coef()`)	Returns a vector of coefficient estimates. `coef(model.slr)[1]` is the value of $\hat{\beta}_0$ and `coef(model.slr)[2]` is the value of $\hat{\beta}_1$.
`confint()`	Produces confidence intervals for the regression coefficients. The probability level is 95% by default, but can be reset using the `level` option.
`residuals()` (or simply `resid()`)	Returns a vector of raw residuals $e_i = y_i - \hat{y}_i$ for $i = 1, \ldots, n$.
`anova()`	Returns the ANOVA table for the fitted model or the ANOVA table for comparing multiple linear models.
`AIC()`	Returns the AIC of the fitted model.
`plot()`	Produces four diagnostic plots for evaluating the appropriateness of the fitted model.

Table 3.3: A list of R functions accompanying the `lm()` function.

2. *Supporting functions for `lm()`:* In R, there is a collection of subsidiary functions that can be applied to an `lm` object. Some of the most commonly used ones are listed in Table 3.3.

Let's try, for instance, the `summary()` function and see what model statistics are included in the summary.

```
# CHUNK 6
s <- summary(model.slr)
s

##
## Call:
## lm(formula = sales ~ TV, data = ad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3860 -1.9545 -0.1913  2.0671  7.2124
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.032594   0.457843   15.36   <2e-16 ***
## TV          0.047537   0.002691   17.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 198 degrees of freedom
```

```
## Multiple R-squared:  0.6119,Adjusted R-squared:  0.6099
## F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

There is a wealth of useful information in the summary, which has two salient parts:

1. *Coefficient estimates table:* Besides the values of the coefficient estimates, we can also obtain their standard errors (in the `Std. Error` column), the accompanying t-statistics (in the `t value` column) for testing the null hypothesis $H_0 : \beta_j = 0$ for $j = 0, 1$, and the associated p-values (in the `Pr(>|t|)` column) for two-sided alternative hypotheses. The asterisks in the p-value column indicate whether the two regression coefficients are statistically significant based on a two-sided t-test, with more asterisks indicating a higher degree of statistical significance. Since the two p-values are virtually zero, we can say that both the intercept and slope are significantly different from zero (in fact, significantly positive). Using the coefficient estimates and the standard errors, we can calculate the 95% confidence intervals for β_0 and β_1 by hand as

   $$\text{coefficient estimate} \pm \text{t-quantile} \times \text{standard error}.$$

   Alternatively and more easily, we can directly apply the `confint()` function to determine the two confidence intervals, as in CHUNK 7.

```
# CHUNK 7
confint(model.slr)

##                 2.5 %     97.5 %
## (Intercept) 6.12971927 7.93546783
## TV          0.04223072 0.05284256
```

   Since both end-points of the confidence interval for β_1 are positive, we can even assert that:

   TV advertising is a significantly *positive* factor associated with sales.

2. *Overall model quality:* The last three lines of the model summary also contains information about more global aspects of the fitted model, such as the residual standard error $s = \sqrt{s^2}$, the coefficient of determination R^2, and the F-statistic for testing the significance of all of the predictor(s) taken together (since `TV` is the only predictor, the F-statistic is for testing the significance of `TV` only).[xv] We can see that the fitted model has explained 61.19% of the variation of `sales` (around its mean), indicating that only a moderate amount of variability of `sales` is explained by regressing `sales` on `TV`. In the next subsection, we will bring in the two other advertising media `radio` and `newspaper` in an attempt to better explain `sales`.

[IMPORTANT!] Making predictions: The `predict()` function. Given the least squares estimates of the intercept β_0 and slope β_1, we can produce predictions for `sales` on the basis of the budget spent on TV advertising via the prediction equation

$$\widehat{\texttt{sales}} = 7.03259 + 0.04754 \times \texttt{TV}$$

[xv] In the special case of simple linear regression, the F-test is equivalent to the t-test for the slope, with the F-statistic being the square of the t-statistic for the slope. In the `ad` data, $F = 312.1 \overset{\text{(rounding error)}}{=} 17.67^2 = t(\hat{\beta}_1)^2$, and the two tests share the same p-value.

over different values of TV. In R, predicted or fitted values can be obtained using the extremely versatile `predict()` function, which is arguably *the most commonly used R function in Exam PA*, allowing us to obtain predictions for models of different types—remember that predictive analytics is mostly concerned with "predictions!" Depending on the class of the object that is supplied as an argument, the `predict()` function will be able to figure out the type of model used and predict accordingly. When applied to an `lm` object, the `predict()` function will by default generate predictions for the same predictor values in the dataset used to train the model. In other words, the fitted values $\hat{y}_i$'s[xvi] will be produced.

```
# CHUNK 8
head(predict(model.slr))

##        1        2        3        4        5        6
## 17.970775  9.147974  7.850224 14.234395 15.627218  7.446162
```

As a check, we can manually calculate the first fitted value as $\hat{y}_1 = 7.03259 + 0.04754(230.1) = 17.97$, which agrees with the output above (up to rounding error).

We can also make predictions for the target variable at specific values of the predictor, not necessarily those in the dataset, by adding the `newdata` argument to the `predict()` function. This argument expects a data frame (recall that data frames are discussed in Subsection 1.2.3) that shares the same predictor variable(s) as the original dataset and contains the predictor value(s) at which predictions are to be carried out. The data frame can be constructed using the `data.frame()` function on the fly or prior to executing the `predict()` function. For instance, to predict `sales` when `TV` is set to 0, 100, 200, and 300, we can use the R code in CHUNK 9 to make four corresponding predictions.

```
# CHUNK 9
df <- data.frame(TV = seq(0, 300, by = 100))
predict(model.slr, newdata = df)

##        1        2        3        4
##  7.032594 11.786258 16.539922 21.293586

# OR simply...
# predict(model.slr, newdata = data.frame(TV = seq(0, 300, by = 100)))
```

Again, you can check manually that these predictions are correct. For example, at `TV` = 100, we have $\widehat{\texttt{sales}} = 7.03259 + 0.04754(100) = 11.79$, consistent with the output above.

Besides point predictions, the `predict()` function can also generate *interval* estimates (for the mean of the target variable) or *interval* predictions (for individual target variables) at particular predictor values of interest, when its `interval` argument is specified to `"confidence"` and `"prediction"`, respectively. By default, 95% intervals are produced, though the probability level can be changed via the use of the additional `level` argument.

In CHUNK 10, we generate the prediction intervals for `sales` at the same values of `TV` as in the `ad` data (`lwr` and `upr` give the lower and upper bounds of the intervals, respectively), augment the

[xvi] The `fitted()` function applied to an `lm` object can also be used to produce fitted values.

data with these intervals, and plot the prediction bands[xvii] on the scatterplot of `sales` against `TV`; see Figure 3.3.3.

```
# CHUNK 10
pred.int <- predict(model.slr, interval = "prediction")
head(pred.int)

##        fit       lwr      upr
## 1 17.970775 11.5135459 24.42800
## 2  9.147974  2.6828666 15.61308
## 3  7.850224  1.3713181 14.32913
## 4 14.234395  7.7921786 20.67661
## 5 15.627218  9.1825560 22.07188
## 6  7.446162  0.9623058 13.93002
```

Finally, to learn more about the `predict()` function applied to `lm` objects, type `?predict.lm`. The "`.lm`" extension is needed as `predict()` is a general-purpose function whose output varies with the object that is specified as an input.

Example 3.3.1. (Value of TV giving rise to shortest prediction interval) Write R code to identify the value of `TV` that leads to the shortest 95% prediction interval for `sales`. Comment on this value.

Solution. The following code from CHUNK 11 uses the `which.min()` function (introduced on page 44) to identify the row index that corresponds to the shortest 95% prediction interval for `sales` and subsets the `ad.pred` data over this row index.

```
# CHUNK 11
ad.pred[which.min(ad.pred$upr - ad.pred$lwr), "TV"]

## [1] 147.3
```

Since the standard error of the prediction error is minimized when $x_* = \bar{x}$ (you may have learned this from Exam SRM), this value of `TV` should be very close to the sample mean of `TV`, as we can confirm with the following command:

```
mean(ad.pred$TV)

## [1] 147.0425
```

□

[xvii] The two bands appear to be parallel straight lines because the standard error of the prediction error can be shown to be

$$\text{SE}(\texttt{sales}_* - \widehat{\texttt{sales}}_*) = \sqrt{s^2\left[1 + \underbrace{\frac{1}{n} + \frac{(\texttt{TV}_* - \overline{\texttt{TV}})^2}{\sum_{i=1}^{200}(\texttt{TV}_i - \overline{\texttt{TV}})^2}}_{\approx 0}\right]} \approx \sqrt{s^2} = 3.259,$$

which is independent of $\texttt{TV}_*$, the value of `TV` at which prediction is performed. In other words, the irreducible error dominates in this case.

```
# CHUNK 10 (Cont.)
ad.pred <- cbind(ad, pred.int)
ggplot(ad.pred, aes(x = TV, y = sales)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_line(aes(y = lwr), color = "red", linetype = "dashed") +
  geom_line(aes(y = upr), color = "red", linetype = "dashed")
```

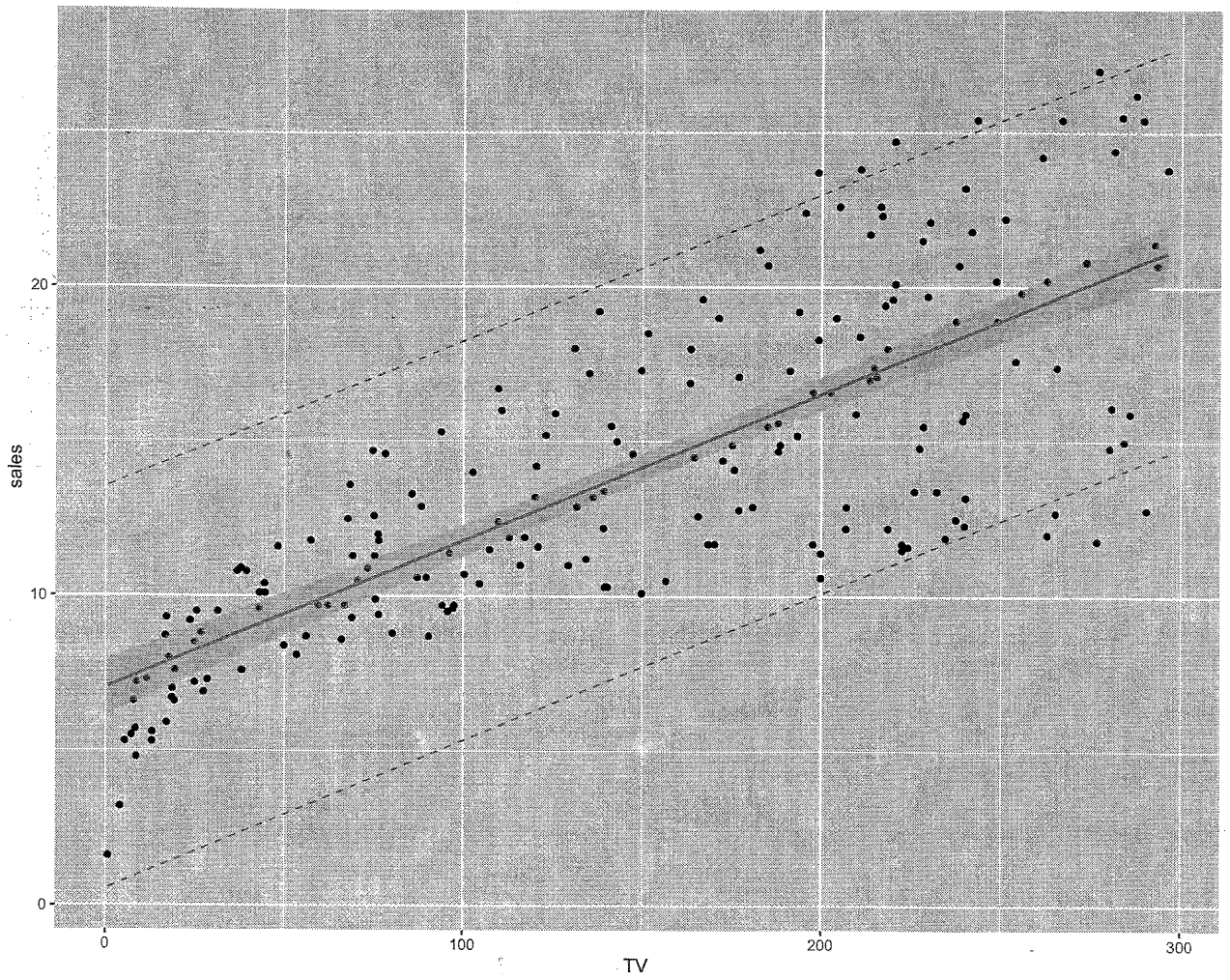


Figure 3.3.3: Upper and lower prediction bands for `sales` based on `TV` in the `ad` data.

Example 3.3.2. (Two more simple linear regression models) Fit separate simple linear regression models of `sales` on `radio` and `newspaper`. If one and only one of the three advertising media can be included as a predictor, which medium would you choose?

Solution. In CHUNK 12, we fit the simple linear regression models of `sales` on `radio` and `newspaper`, and print a summary for each model.

```
# CHUNK 12
slr.radio <- lm(sales ~ radio, data = ad)
slr.newspaper <- lm(sales ~ newspaper, data = ad)

summary(slr.radio)

##
## Call:
## lm(formula = sales ~ radio, data = ad)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.7305  -2.1324   0.7707   2.7775   8.1810
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.31164    0.56290  16.542   <2e-16 ***
## radio        0.20250    0.02041   9.921   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.275 on 198 degrees of freedom
## Multiple R-squared:  0.332,Adjusted R-squared:  0.3287
## F-statistic: 98.42 on 1 and 198 DF,  p-value: < 2.2e-16
```

```
summary(slr.newspaper)

##
## Call:
## lm(formula = sales ~ newspaper, data = ad)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.2272  -3.3873  -0.8392   3.5059  12.7751
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.35141    0.62142   19.88  < 2e-16 ***
## newspaper    0.05469    0.01658    3.30  0.00115 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.092 on 198 degrees of freedom
## Multiple R-squared:  0.05212,Adjusted R-squared:  0.04733
## F-statistic: 10.89 on 1 and 198 DF,  p-value: 0.001148
```

Even though `radio` and `newspaper` are significant for `sales` on their own, the quality of these two models is not as favorable as that of the model of `sales` on `TV`. Their coefficients of determination R^2 are only 33.2% and 5.212%, much lower than the 61.19% for the model of `sales` on `TV`. If one and only one of the three advertising media can serve as a predictor for `sales`, it makes full sense to use `TV` because its association with `sales` is the strongest. □

Remark. It is not a good idea to look at which simple linear regression model has the highest estimated coefficient for the predictor (`radio` in this case). The estimated coefficient itself does not take into account the variability (standard error) of the estimate, but the t-statistics do. In fact, ranking the three models on the basis of R^2 is the same as ranking them on the basis of the t-statistics of the slope coefficients.

3.3.2 Multiple Linear Regression

We now extend the simple linear regression model in the preceding subsection and incorporate all of the three advertising media, `TV`, `radio`, and `newspaper`, in a multiple linear regression model. By accommodating the three media in the same model, not only can we capitalize on a larger amount of available information to better understand the sales-generating mechanism, we can also experiment with a wide variety of model forms that capture the interplay among the three media in affecting `sales` and potentially improve prediction accuracy. In the current context, we have two important tasks:

1. To develop linear models that can be used to predict `sales` on the basis of the three advertising media and identify the most important media affecting `sales`.

2. To quantify the prediction accuracy of the linear models developed and make a recommendation as to which model to use.

Model 1: Multiple linear regression model with all three media. As our first multiple linear regression model, let's regress `sales` on `TV`, `radio`, and `newspaper`, with a separate slope coefficient attached to each of the three advertising media. The model equation is

$$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{newspaper} + \varepsilon.$$

Fitting this model with multiple predictors can be accomplished again by the `lm()` function, with the model formula specified in the long way

```
# CHUNK 13
# Long way
model.1 <- lm(sales ~ TV + radio + newspaper, data = ad)
```

or in the short way

```
# OR the shorthand...
model.1 <- lm(sales ~ ., data = ad)
```

In the first formula, we put the target variable `sales` on the left and separate it from the set of predictors by a tilde. The predictors are specified one by one after the tilde sign and separated by plus (+) signs. In general, the formula `Y ~ X1 + X2 + ··· + Xp` means that we fit the linear model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$. In the second formula, the dot (`.`) is a placeholder for all other variables in the dataset, so the formula `sales ~ .` means that we regress `sales` on every variable in the dataset other than `sales`. This shortcut is especially useful when the dataset contains a large number of variables (almost always the case in Exam PA), making it cumbersome to type out all of them. Either way allows us to fit the linear model regressing `sales` on `TV`, `radio`, `newspaper` and to save it as an `lm` object called `model.1` or Model 1 for convenience. Now let's print out a summary for the model to learn about its characteristics.

```
# CHUNK 13 (Cont.)
s <- summary(model.1)
s

##
## Call:
## lm(formula = sales ~ ., data = ad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
```

```
## TV           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972,Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

The least squares coefficient estimates, computed by the matrix formula $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{Y})$, are[xviii]

$$\hat{\beta}_0 = 2.938889, \qquad \hat{\beta}_1 = 0.045765, \qquad \hat{\beta}_2 = 0.188530, \qquad \hat{\beta}_3 = -0.001037.$$

We can interpret these estimates as follows:

- When no money is spent on any of the three advertising media, the expected amount of `sales` is estimated to be $\hat{\beta}_0 = 2.938889$.
- In the multiple linear regression framework, the interpretations of the slope coefficients differ slightly due to the presence of other predictors. For instance, when an additional \$1,000 is spent on TV advertising, the expected increase in `sales` is estimated to be $1,000\hat{\beta}_1 = 45.765$ units, *provided that the advertising budgets in radio and newspaper are held fixed* (don't miss the "holding everything else constant" assumption!). This expected increase is comparable to that in the simple linear regression model of `sales` on `TV` (see page 152). Similar interpretations can be made for $\hat{\beta}_2 = 0.188530$ and $\hat{\beta}_3 = -0.001037$.

We can observe from the p-value column that the *key* advertising media affecting `sales` are `TV` and `radio`, with their p-values being practically zero. As their coefficient estimates are positive, we can conclude that `TV` and `radio` are significant advertising media contributing favorably to `sales`, as we would have expected. `Newspaper`, however, turns out to be highly insignificant. More precisely, there is minimal evidence that `newspaper` is related to `sales` *in the presence of* `TV` *and* `radio`. This is at odds with the simple linear regression model of `sales` on `newspaper`, where `newspaper` is highly significant (recall Example 3.3.2). How can we reconcile this apparent contradiction?

Example 3.3.3. [TECHNICAL!] (Partial correlation between `sales` and `newspaper`) You may recall from Exam SRM that the *partial correlation* between a target variable y and a predictor x_j is the correlation between the two variables *after controlling for the effects of other predictors*. It can be calculated in two ways:

1. *(By definition)* Regress y on all predictors other than x_j and denote the residuals by e_1. Then regress x_j on all predictors other than x_j and denote the residuals by e_2. The partial correlation between y and x_j is defined as the correlation between the two sets of residuals.

[xviii] Slide 12 of Module 6 of the PA e-learning modules suggests that we manually compute the ordinary least squares estimates using the code `solve(t(X) %*% X) %*% (t(X) %*% Y)` (see Example 1.2.2). The point of doing so is to check whether you can reproduce the R output you see. Of course, such a kind of manual computation is not required in Exam PA as you can directly read off the R output.

2. *(A convenient computing formula)* Regress y on all predictors, including x_j, and obtain the t-statistic $t(\hat{\beta}_j)$ associated with x_j. Then the partial correlation between y and x_j can be computed as

$$\frac{t(\hat{\beta}_j)}{\sqrt{t(\hat{\beta}_j)^2 + (n-p-1)}}.$$

Write R code to calculate the partial correlation between `sales` and `newspaper`. Compare it with the ordinary correlation between the two variables. What can you conclude about the association between the two variables?

Solution. To calculate the partial correlation between `sales` and `newspaper` by definition, we have to fit two models, one regressing `sales` on `TV` and `radio` and one regressing `newspaper` on `TV` and `radio`. Then we use the `cor()` function to compute the correlation between the two series of residuals.

```
# CHUNK 14 (Example 3.3.3)
# Calculating partial correlation by definition
m1 <- lm(sales ~ TV + radio, data = ad)
m2 <- lm(newspaper ~ TV + radio, data = ad)

cor(m1$residuals, m2$residuals)

## [1] -0.01262147
```

Instead of fitting two regression models, we can also extract the t-statistic for `newspaper` from Model 1 (which regresses sales on `TV`, `radio`, and `newspaper`) and calculate the partial correlation using the computing formula above.

```
# CHUNK 14 (Cont.)
# Alternative formula to calculate partial correlation
t <- s$coefficients["newspaper", "t value"]
p <- length(coef(model.1)) - 1  # number of predictors in full model
t / sqrt(t^2 + nrow(ad) - p - 1)

## [1] -0.01262147
```

The partial correlation between `sales` and `newspaper` is essentially zero. In contrast, the ordinary correlation between the two variables is 0.228299. Much of this correlation stems from the positive correlation between `radio` and `newspaper`, which is 0.3541038.

```
# CHUNK 14 (Cont.)
cor(ad$sales, ad$newspaper)

## [1] 0.228299

cor(ad$radio, ad$newspaper)

## [1] 0.3541038
```

□

The last three lines of the summary output contain information about the global quality of the fitted linear regression model. We can see that the RSE has dropped from 3.259 to 1.686 and the R^2 has increased noticeably from 0.6119 to 0.8972 (i.e., 89.72% of the variation of `sales` is explained by the three advertising media together) when switching from the simple linear regression model of `sales` on `TV` to the current multiple linear regression model. The F-statistic takes the large value of 570.3. The p-value of the F-test borders on zero, so we have extremely strong evidence that *at least one* of the three advertising media is an important predictor of `sales`. As we have seen above, however, only `TV` and `radio` appear to be truly related to `sales`.

Model 2: Model with only `TV` and `radio`. As Model 1 suggests that `newspaper` is not associated with `sales` in the presence of `TV` and `radio`, let us try to refine the model by dropping `newspaper` and regressing `sales` only on `TV` and `radio`. Doing so leads to Model 2, whose summary is displayed in CHUNK 15.

```
# CHUNK 15
model.2 <- lm(sales ~ TV + radio, data = ad)

# OR regress sales on all media except newspaper
# model.2 <- lm(sales ~ . - newspaper, data = ad)

summary(model.2)

##
## Call:
## lm(formula = sales ~ TV + radio, data = ad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7977 -0.8752  0.2422  1.1708  2.8328
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.92110    0.29449   9.919   <2e-16 ***
## TV           0.04575    0.00139  32.909   <2e-16 ***
## radio        0.18799    0.00804  23.382   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.681 on 197 degrees of freedom
## Multiple R-squared:  0.8972,Adjusted R-squared:  0.8962
## F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

The deletion of `newspaper` causes minimal effects on the model, speaking to the statistical insignificance of `newspaper` in the presence of `TV` and `radio`. Not only does the R^2 stay approximately the same at 89.72% (we know that the R^2 of Model 2 cannot be higher than that of Model 1, but the drop is negligible), the F-statistic and adjusted R^2 of the model also increase when `newspaper` is omitted. These results suggest that regressing `sales` only on `TV` and `radio` improves the performance of the linear model and is likely to enhance its prediction accuracy.

Besides looking at the RSE and (adjusted) R^2, it is also a good idea to plot the data points $\{(\texttt{TV}_i, \texttt{radio}_i, \texttt{sales}_i)\}_{i=1}^{200}$, as graphical summaries can sometimes reveal problems that cannot be reflected by numerical statistics. Due to the presence of two predictors, the data points now exist in a three-dimensional space and the fitted regression function is now a three-dimensional plane instead of a line. In CHUNK 16, we use the `scatter3d()` function in the `car` and `rgl` packages to plot the three-dimensional data points and the fitted regression plane (see Figure 3.3.4). These two packages will not be available on the PA exam and you are not responsible for generating three-dimensional plots. Here just pay attention to what you can observe from the plot.

When CHUNK 16 is run, R will open a separate window carrying a three-dimensional plot of `sales` against `TV` and `radio`. You can rotate the plot to look at the behavior of the data points and the fitted regression plane from different perspectives. As we expect, the regression plane is upward tilting with respect to both `TV` and `radio`—the more we spend on `TV` and `radio`, the larger the value of `sales` on average. Response observations that lie above the fitted plane have their residuals (which are positive) colored in cyan whereas those that lie below the plane have their (negative) residuals colored in purple. There is a clear pattern of positive residuals on the 45-degree line and negative residuals when most of the advertising expenditure is spent on either `TV` or `radio`. In other words, the fitted model systematically underestimates `sales` when the advertising budget is split between `TV` and `radio`, but overestimates `sales` when the company relies mainly on either `TV` or `radio` as the advertising medium. These deficiencies provide the motivation for us to refine our linear model with the goal of effectively capturing the systematic patterns in the observations.

```
# CHUNK 16 (not required for Exam PA)
library(car)
library(rgl)
scatter3d(sales ~ TV + radio, data = ad)
```

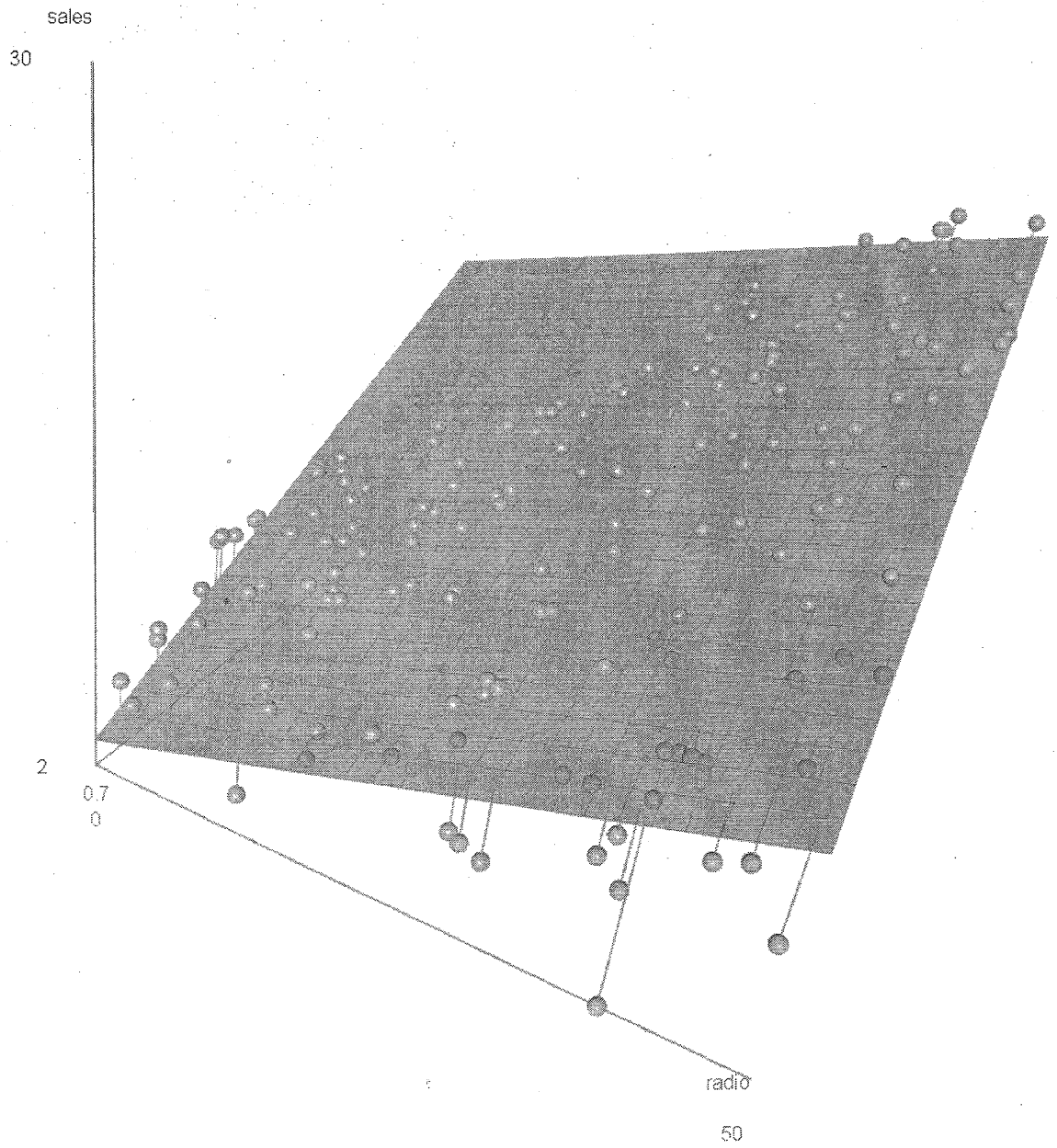


Figure 3.3.4: A three-dimensional plot for the linear model regressing `sales` on `TV` and `radio` in the `ad` data.

Model 3: Model with TV and radio with interaction. Figure 3.3.4 suggests the presence of a prominent interaction effect between TV and radio which is not taken into account by Model 2. In other words, the rate of increase in sales with respect to TV (resp. radio) appears to increase with the value of radio (resp. TV), which is at odds with Model 2's implicit assumption that the average effect on sales of each medium is independent of the amount spent on the other medium. To take care of the synergy or interaction effect between TV and radio, we employ the following model (called Model 3):

$$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \boxed{\texttt{TV} \times \texttt{radio}} + \varepsilon,$$

which has the product of TV and radio added as a new predictor. Since

$$\frac{\partial}{\partial \texttt{TV}} \mathbb{E}[\texttt{sales}] = \beta_1 + \beta_3 \times \texttt{radio}, \qquad \frac{\partial}{\partial \texttt{radio}} \mathbb{E}[\texttt{sales}] = \beta_2 + \beta_3 \times \texttt{TV},$$

the average effect of TV (resp. radio) on sales now depends on the amount spent on radio (resp. TV). From Figure 3.3.4, we expect the value of β_3 to be positive.

In R, the colon (:) operator is used to specify a single interaction term, so X1:X2 is the interaction term for the predictors X1 and X2. If you use the asterisk (*) and type X1 * X2, then this is identical to the long form X1 + X2 + X1:X2, with both main effects terms and the interaction term inserted. Notice that some algebraic operators are interpreted differently in an R formula. For instance, the asterisk *, when used in an R formula, does not mean multiplication.

Let's run CHUNK 17 to fit the above model with interaction effects and look at its summary output.

```
# CHUNK 17
model.3 <- lm(sales ~ TV * radio, data = ad)

# OR
# model.3 <- lm(sales ~ TV + radio + TV:radio, data = ad)

summary(model.3)

##
## Call:
## lm(formula = sales ~ TV * radio, data = ad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3366 -0.4028  0.1831  0.5948  1.5246
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.750e+00  2.479e-01  27.233   <2e-16 ***
## TV          1.910e-02  1.504e-03  12.699   <2e-16 ***
## radio       2.886e-02  8.905e-03   3.241   0.0014 **
## TV:radio    1.086e-03  5.242e-05  20.727   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.9435 on 196 degrees of freedom
## Multiple R-squared:  0.9678,Adjusted R-squared:  0.9673
## F-statistic:  1963 on 3 and 196 DF,  p-value: < 2.2e-16
```

The estimated coefficient for the interaction term, as we expect, is positive, with an extremely low p-value, indicating that we have very strong evidence that positive interaction effects exist between `TV` and `radio`. Further evidence of the existence of the interaction effect is furnished by the notable jump of R^2 from 89.72% for Model 2 to 96.78% for Model 3 and the sharp decline of the RSE from 1.681 for Model 2 to 0.9435 for Model 3. By any standard, Model 3 is superior to the main-effects-only Model 2.

Example 3.3.4. (Evaluating three marketing plans) Consider the `Advertising` data again. Suppose that as the manager of the company, you are given a total of \$100,000 to spend on advertising. You have in mind three ways to allocate the \$100,000 budget among TV advertising and radio advertising:

1. Spending the entire amount on radio advertising.
2. Splitting the budget evenly between TV and radio advertising.
3. Spending \$70,000 on radio advertising and \$30,000 on TV advertising.

Write R code to evaluate these three marketing plans. Which one would you recommend?
(Note: Keep in mind that the `TV` and `radio` variables are measured in thousands of dollars.)

Solution. Let's first build a data frame called `df` to host the three pairs of predictor values of interest.

```
# CHUNK 18
df <- data.frame(TV = c(0, 50, 30), radio = c(100, 50, 70))
```

There is no need to create a separate variable for the product of `TV` and `radio` as there is no such variable in the original `ad` data frame. Now let's apply the `predict()` function to perform predictions at the values contained in `df`.

```
# CHUNK 18 (Cont.)
predict(model.3, newdata = df)

##        1        2        3
##  9.636254 11.864528 11.625115
```

We can manually check that these predictions are correct (and so our code works properly!). For example, when `TV` = `radio` = 50, we have

$$\begin{aligned}\widehat{\texttt{sales}} &= 6.750220203 + 0.019101074(50) + 0.028860340(50) + 0.001086495(50)^2 \\ &= 11.864528;\end{aligned}$$

the more accurate coefficient estimates can be obtained from `model.3$coefficients` or `coef(model.3)`.

```
coef(model.3)

##  (Intercept)           TV        radio     TV:radio
## 6.750220203 0.019101074 0.028860340 0.001086495
```

Of the three marketing plans, the second one, which splits the budget evenly among the two advertising media and maximizes the interaction term $\texttt{TV} \times \texttt{radio}$, leads to the highest predicted `sales` and so is our recommended plan.

□

Remark. The optimal combination of `TV` and `radio` can be found by maximizing the fitted regression function $\widehat{\texttt{sales}} = \hat{\beta}_0 + \hat{\beta}_1 \times \texttt{TV} + \hat{\beta}_2 \times \texttt{radio} + \hat{\beta}_3 \times \texttt{TV} \times \texttt{radio}$ subject to the constraint $\texttt{TV} + \texttt{radio} = 100$. To this end, we can apply the method of Lagrange multipliers in multi-variable calculus.

In Model 3, both the main effects and interaction effect are statistically significant as evidenced by their extremely small p-value. In other situations, it can happen that the interaction effect is significant, but not the main effects. If you recall the *hierarchical principle*,[xix] which you may have seen when studying for Exam SRM, it is a common practice that whenever a higher-order variable is retained in a model (due to statistical significance), so are all lower-order variables, even if they are insignificant. As a consequence, if `X1:X2` is kept, then so are the main effects. This explains why the shorthand `X1 * X2` is much more commonly used than `X1:X2` alone.

Model 4: Model with `TV` and `radio` with interaction and polynomial terms. Besides the use of interaction terms, another way to accommodate nonlinear relationships between the target variable and individual predictors is polynomial regression. Via the use of higher-power terms, we are able to take care of a wide variety of curved relationships which are substantially more complex than linear ones.

In Figure 3.3.2, we observed a positive relationship between `sales` and `TV`, but upon closer examination, the relationship appears to be curved rather than linear.[xx] The expected rate of increase of `sales` with respect to `TV` seems to be increasing for small values of `TV`, but eventually starts to decline when the value of `TV` is sufficiently large, conforming to the *law of diminishing marginal returns* in economics—the marginal increase in `sales` due to an additional unit increase in `TV` eventually declines beyond a certain point of saturation. This motivates the introduction of higher powers of `TV`, say the square of `TV`, as an additional predictor to be added to Model 3. The equation of the resulting model, called Model 4, becomes

$$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \boxed{\texttt{TV}^2} + \beta_4 \times \texttt{TV} \times \texttt{radio} + \varepsilon.$$

When $\texttt{TV}^2$ is inserted, the expected rate of change of `sales` with respect to `TV` is

$$\frac{\partial}{\partial \texttt{TV}} \mathbb{E}[\texttt{sales}] = \beta_1 + 2\beta_3 \times \texttt{TV} + \beta_4 \times \texttt{radio},$$

which depends not only on the value of `radio` (due to the interaction term), but also on the value of `TV` (due to the square term).

[xix] See page 89 of ISLR.

[xx] The curved relationship between `sales` and `radio` is less pronounced.

In R, higher powers of a predictor can be created before applying the `lm()` function, e.g., the square of a predictor `X` can be set up using the command

```
X2 <- X^2
```

or they can be created on the fly and entered into the model formula using the `I()` function,[xxi] where "I" stands for "insulation." For example, the formula `Y ~ X + I(X^2)` fits the quadratic model $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \varepsilon$. The use of the `I()` function is necessary because the `^` operator, when used in an R formula, has a different meaning than exponentiation and the `I()` function ensures that the elements within the parentheses are processed arithmetically. If you are interested, the precise meaning of `"^k"` in a formula, not discussed in the PA e-learning modules, is to include terms up to a k-way interaction, so, for example, the R formula `(X1 + X2 + X3)^2` is interpreted by R as `X1 + X2 + X3 + X1:X2 + X1:X3 + X2:X3`. There are no square terms for any of the `X`'s!

Run CHUNK 19 to fit Model 4 and generate its summary.

```
# CHUNK 19
model.4 <- lm(sales ~ TV * radio + I(TV^2), data = ad)
summary(model.4)

##
## Call:
## lm(formula = sales ~ TV * radio + I(TV^2), data = ad)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9949 -0.2969 -0.0066  0.3798  1.1686
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.137e+00  1.927e-01  26.663  < 2e-16 ***
## TV           5.092e-02  2.232e-03  22.810  < 2e-16 ***
## radio        3.516e-02  5.901e-03   5.959 1.17e-08 ***
## I(TV^2)     -1.097e-04  6.893e-06 -15.920  < 2e-16 ***
## TV:radio     1.077e-03  3.466e-05  31.061  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6238 on 195 degrees of freedom
## Multiple R-squared:  0.986,Adjusted R-squared:  0.9857
## F-statistic:  3432 on 4 and 195 DF,  p-value: < 2.2e-16
```

Just as we expect, the coefficient estimate of the square term is negative and the p-value is extremely small, showing that the square term is highly significant. Compared to Model 3, the adjusted R^2

[xxi] Pages 116 and 117 of ISLR suggest using the `poly()` function when fitting a polynomial regression function of a high degree to avoid having to type out so many `I()`'s. If you use this function, which is not covered in the PA e-learning modules, in your future work, you should keep in mind that this function uses the so-called orthogonal polynomials instead of the "usual" polynomials. Although the predicted values are independent of the basis generating the polynomials, the parameter estimates and interpretation will differ. Type `?poly` to learn more.

and RSE of Model 4 have experienced a noticeable amount of improvement, from 0.9673 to 0.9857 and from 0.9435 to 0.6238, respectively. These metrics are pointers that Model 4 outperforms even Model 3 and is by far the best model among the four linear models we have constructed.

3.3.3 Evaluation of Linear Models

While the adjusted R^2 and RSE are indirect estimates of the test error of a linear model, in Exam PA it is most desirable to evaluate the predictive performance of a linear model directly on an independent test set. In this subsection, we split the `ad` dataset into two parts:

- A training set, where we train our candidate linear models and get the coefficient estimates.
- A test set, where we use the linear models fitted to the training set to make predictions.

We will then evaluate the prediction accuracy of the four linear models defined in Subsection 3.3.2 according to a certain criterion (e.g., test RMSE).

Creation of training and test sets: `createDataPartition()`. There are a number of ways to split the data into the training and test sets in R (e.g., using the `sample()` function is one of the easiest). In Exam PA, we will use the `createDataPartition()`[xxii] function in the `caret` package (available on the exam). Used in all of the past and sample PA projects, this function is extremely important as each PA project involves model validation in one way or another. CHUNK 20 performs the training/test split for the `ad` dataset using this function.

```
# CHUNK 20
#install.packages(caret)  # uncomment this line the first time you use caret
library(caret)
set.seed(1)  # set the random seed so that the results are reproducible
partition <- createDataPartition(ad$sales, p = 0.7, list = FALSE)
train <- ad[partition, ]
test <- ad[-partition, ]
```

Although all of the sample and past PA projects provide code for doing the training/test split using the `createDataPartition()` function, the syntax of the function is simple enough to understand (recall what you did in Problem 1.5.1). The first argument of the function, set to the target variable in this case, is the variable with respect to which sampling is performed, and the second argument specifies the proportion of data that will be allocated to the training set. In our current context, approximately $200(0.7) \approx 142$ and $200(0.3) \approx 58$ observations constitute the training set and test set, respectively. With the option `list = FALSE`, a vector (instead of a list) of row indices, called `partition` that belong to the training set will be returned (you can use other names such as `training.indices` as in the PA e-learning modules). We then subset the `ad` data frame using `partition` as the row index vector and the complement of `partition` to get the training and test sets, saved in R as `train` and `test`, respectively.

The reason why the SOA strongly encourages PA candidates to use the `createDataPartition()` function is that it automatically performs *stratified sampling*[xxiii] based on the distribution of the

[xxii] The precise output of the `createDataPartition()` function may differ slightly depending on which version of the `caret` package and/or R you are using.

[xxiii] If you took Exam MFE, the predecessor of Exam IFM, then you probably have seen stratified sampling in the simulation chapter. (Remember? ☺)

target variable instead of random sampling. In brief, stratified sampling divides the support of the distribution of the target variable into several "strata" (or ranges) and ensures that every "stratum" is represented in the training and test sets. This way, the training/test split will try to have the training and test sets produce comparable and representative distributions for the target variable. In the case of the `ad` dataset, you can verify the similarity of the values of `sales` in the training and test sets by running the following code:

```
# CHUNK 21
print("TRAIN")
summary(train$sales)
print("TEST")
summary(test$sales)

## [1] "TRAIN"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.60   10.43   12.90   13.96   17.30   26.20
## [1] "TEST"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.20   10.32   12.60   14.19   17.40   27.00
```

A NOTE ON SIMULATION

If you are on version 3.5.x (or an earlier version) of R, you will likely get results different from those in this study manual (which are based on version 3.6.2 of R) whenever `caret`'s data partition and other cross-validation functions are invoked, due to a change in R's random number generator for version 3.6.0 or later. To match the results in this manual while using version 3.5.x of R, please executive the following command at the start of your R session:

```
RNGkind(sample.kind = "Rejection")
```

The two sets of summary statistics are indeed very close to each other, bearing testimony to the stratified sampling that `createDataPartition()` implements.

Note that it suffices to (and we should!) do the training/test split only once as we will use the same training and test sets to evaluate all of our candidate linear models. This is to make sure that the comparisons are fair.

EXAM NOTE

The December 2018 PA exam solution says that "[f]ew candidates explicitly recognized or checked the importance of the use of stratified sampling to improve the effectiveness of the model validation" when using the `createDataPartition()` function. When you do model validation, be sure to:

- Describe the design/results of the training/test split, i.e., the proportion (or number) of observations that go to the training set and test set.
- Point out that you are using stratified sampling to ensure that both the training and test sets contain similar and representative values of the response variable.
- Explicitly check that the two sets of target variable values are comparable by looking at some summary statistics, e.g., the mean.

Fitting the linear models on the training set and testing them out on the test set. Now that the training/test split is done, we first fit the four linear models defined in the preceding subsection to the training set. For convenience, we first restate the definitions of these four models as follows: (the last model is provided for comparison)

Model	Equation
1	$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{newspaper} + \varepsilon$
2	$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \varepsilon$
3	$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{TV} \times \texttt{radio} + \varepsilon$
4	$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{TV}^2 + \beta_4 \times \texttt{TV} \times \texttt{radio} + \varepsilon$
5	$\texttt{sales} = \beta_0 + \beta_1 \times \texttt{TV} + \beta_2 \times \texttt{radio} + \beta_3 \times \texttt{TV}^2 + \beta_4 \times \texttt{radio}^2 + \beta_5 \times \texttt{TV} \times \texttt{radio} + \varepsilon$

Run CHUNK 22 to fit these five models to the training set (the `tr` suffix stands for "training"). Do note that the `data` argument of the `lm()` function is set to `train` rather than `ad`.

```
# CHUNK 22
model.1.tr <- lm(sales ~ TV + radio + newspaper, data = train)
model.2.tr <- lm(sales ~ TV + radio, data = train)
model.3.tr <- lm(sales ~ TV * radio, data = train)
model.4.tr <- lm(sales ~ TV * radio + I(TV^2), data = train)
model.5.tr <- lm(sales ~ TV * radio + I(TV^2) + I(radio^2), data = train)
```

Then for each trained linear model, we generate predictions on the test set and compute the test RMSE, a natural performance measure for linear models given by

$$\text{test RMSE} = \sqrt{\frac{1}{n_{\text{test}}} \sum_{i \in \text{test set}} (y_i - \hat{y}_i)^2}.$$

As the same procedure will be carried out for all of the five models, it is desirable (though not absolutely necessary) to define a function that will help us reduce repetition. In CHUNK 23, we

first define a function called `rmse()` for calculating the RMSE from a vector of observed target variable values (`observed`) and a vector of predicted target variable values (`predicted`), in the training set or test set. Then the function is applied to compute the training RMSE for the five models.

```
# CHUNK 23
rmse <- function(observed, predicted) {
  sqrt(mean((observed - predicted)^2))
}

print("TRAIN")
rmse(train$sales, predict(model.1.tr))
rmse(train$sales, predict(model.2.tr))
rmse(train$sales, predict(model.3.tr))
rmse(train$sales, predict(model.4.tr))
rmse(train$sales, predict(model.5.tr))

## [1] "TRAIN"
## [1] 1.701546
## [1] 1.704158
## [1] 0.9645623
## [1] 0.6433654
## [1] 0.6394058
```

From Models 2 to 5, the linear models have increasing levels of complexity (i.e., they are nested) and thus decreasing training RMSE. Note that we specify as the first argument of the `rmse()` function the `sales` variable in the training set and the second argument is the predicted (or fitted) values generated by the linear model in question. Remember that when the `predict()` function is applied to an `lm` object without additional arguments, the fitted values in the training set are formed by default.

We now apply the linear models fitted to the *training* set to generate the predicted values on the *test* set, which is passed to the `newdata` argument.

```
# CHUNK 23 (Cont.)
print("TEST")
rmse(test$sales, predict(model.1.tr, newdata = test))
rmse(test$sales, predict(model.2.tr, newdata = test))
rmse(test$sales, predict(model.3.tr, newdata = test))
rmse(test$sales, predict(model.4.tr, newdata = test))
rmse(test$sales, predict(model.5.tr, newdata = test))

## [1] "TEST"
## [1] 1.602705
## [1] 1.581734
## [1] 0.8645022
## [1] 0.5473428
## [1] 0.5589234
```

Unlike the training RMSE, the test RMSE first decreases from Models 1 to 4, then eventually rises when going from Models 4 to 5. As we make our linear model more sophisticated, its predictive performance initially improves due to its increased ability to capture complicated patterns. When it becomes unnecessarily complicated, it appears to overfit the data and its predictive performance worsens. This creates the characteristic U-shape that characterizes the test error as a function of model complexity discussed in Subsection 3.1.3. Among the five linear models, Model 4 gives rise to the *smallest* test RMSE and is our recommended linear model on the basis of prediction accuracy. Granted, there may be (and very likely there are!) other specifications of `TV` and `radio` that result in an even more predictive model for `sales`, but Model 4 is arguably the best linear model in terms of prediction accuracy among the five models above.

3.4 Case Study 2: Feature Selection and Regularization

Having looked at a warm-up case study of linear models, we now turn to a substantially more extended case study whose sophistication closely resembles that of a typical PA exam project. In addition to the modeling concepts discussed in the advertising case study in Section 3.3, this section will also illustrate how to deal with categorical predictors in a linear model, how to select useful features from a large set of predictors, and how to perform regularization, all of which are popular test items in Exam PA. Specifically, after completing this case study, you should be able to:

- Fit a multiple linear regression model with both quantitative and qualitative predictors.
- Detect and accommodate interactions between predictors which can be quantitative or qualitative.
- Binarize categorical predictors using the `dummyVars()` function from the `caret` package and recognize the need for explicit binarization.
- Select useful features using the `stepAIC()` function from the `MASS` package and be familiar with the different options allowed by this function.
- Generate and interpret diagnostic plots for a linear model.
- Implement ridge regression and lasso using the `glmnet()` and `cv.glmnet()` functions from the `MASS` package.

To mimic the exam environment and make our learning more systematic, this case study (and the case studies in subsequent chapters) will be organized around a series of well-defined tasks, which are displayed in boxes and worded in a way that parallels released PA exams.

3.4.1 Preparatory Work

Data description. In this section, we will examine the `Credit` dataset that accompanies Chapters 3 and 6 of ISLR. Compared to the advertising dataset in Section 3.3, the `Credit` dataset has a lot more predictors, some of which are categorical, and is a perfect illustration of feature selection. Again, you may have already seen the `Credit` data when studying for Exam SRM, but our treatment here will have a much heavier predictive analytic flavor and will be structured in a way that is tailored for Exam PA (e.g., we will only make use of R functions covered by the PA e-learning modules and exam projects, which are different from the R functions used in ISLR).