Machine Learning Essentials

Practical Guide in R

Alboukadel KASSAMBARA

Edition 1
sthda.com/english

Machine Learning Essentials

Alboukadel KASSAMBARA

Published by STHDA (`http://www.sthda.com`), Alboukadel Kassambara

Contact: Alboukadel Kassambara <alboukadel.kassambara@gmail.com>

For general information contact Alboukadel Kassambara <alboukadel.kassambara@gmail.com>.

Contents

Preface

0.1 What you will learn

Large amount of data are recorded every day in different fields, including marketing, bio-medical and security. To discover knowledge from these data, you need machine learning techniques, which are classified into two categories:

1. **Unsupervised machine learning methods**:

These include mainly *clustering* and *principal component analysis* methods. The goal of clustering is to identify pattern or groups of similar objects within a data set of interest. Principal component methods consist of summarizing and visualizing the most important information contained in a multivariate data set.

These methods are "unsupervised" because we are not guided by a priori ideas of which variables or samples belong in which clusters or groups. The machine algorithm "learns" how to cluster or summarize the data.

2. **Supervised machine learning methods**:

Supervised learning consists of building mathematical models for predicting the outcome of future observations. Predictive models can be classified into two main groups:

- *regression analysis* for predicting a continuous variable. For example, you might want to predict life expectancy based on socio-economic indicators.
- *Classification* for predicting the class (or group) of individuals. For example, you might want to predict the probability of being diabetes-positive based on the glucose concentration in the plasma of patients.

These methods are supervised because we build the model based on known outcome values. That is, the machine learns from known observation outcomes in order to predict the outcome of future cases.

In this book, we present a practical guide to machine learning methods for exploring data sets, as well as, for building predictive models.

You'll learn the basic ideas of each method and reproducible R codes for easily computing a large number of machine learning techniques.

0.2 Key features of this book

Our goal was to write a practical guide to machine learning for every one.

The main parts of the book include:

- **Unsupervised learning methods**, to explore and discover knowledge from a large multivariate data set using clustering and principal component methods. You will learn hierarchical clustering, k-means, principal component analysis and correspondence analysis methods.
- **Regression analysis**, to predict a quantitative outcome value using linear regression and non-linear regression strategies.
- **Classification techniques**, to predict a qualitative outcome value using logistic regression, discriminant analysis, naive bayes classifier and support vector machines.
- **Advanced machine learning methods**, to build robust regression and classification models using k-nearest neighbors methods, decision tree models, ensemble methods (bagging, random forest and boosting)
- **Model selection methods**, to select automatically the best combination of predictor variables for building an optimal predictive model. These include, best subsets selection methods, stepwise regression and penalized regression (ridge, lasso and elastic net regression models). We also present principal component-based regression methods, which are useful when the data contain multiple correlated predictor variables.
- **Model validation and evaluation techniques** for measuring the performance of a predictive model.
- **Model diagnostics** for detecting and fixing a potential problems in a predictive model.

The book presents the basic principles of these tasks and provide many examples in R. This book offers solid guidance in data mining for students and researchers.

Key features:

- Covers machine learning algorithm and implementation
- Key mathematical concepts are presented
- Short, self-contained chapters with practical examples. This means that, you don't need to read the different chapters in sequence.

> At the end of each chapter, we present R lab sections in which we systematically work through applications of the various methods discussed in that chapter.

0.3 Book website

`http://www.sthda.com/english`

About the author

Alboukadel Kassambara is a PhD in Bioinformatics and Cancer Biology. He works since many years on genomic data analysis and visualization (read more: `http://www.alboukadel.com/`).

He has work experiences in statistical and computational methods to identify prognostic and predictive biomarker signatures through integrative analysis of large-scale genomic and clinical data sets.

He created a bioinformatics web-tool named GenomicScape (www.genomicscape.com) which is an easy-to-use web tool for gene expression data analysis and visualization.

He developed also a training website on data science, named STHDA (Statistical Tools for High-throughput Data Analysis, www.sthda.com/english), which contains many tutorials on data analysis and visualization using R software and packages.

He is the author of many popular R packages for:

- multivariate data analysis (**factoextra**, `http://www.sthda.com/english/rpkgs/factoextra`),
- survival analysis (**survminer**, `http://www.sthda.com/english/rpkgs/survminer/`),
- correlation analysis (**ggcorrplot**, `http://www.sthda.com/english/wiki/ggcorrplot-visualization-of-a-correlation-matrix-using-ggplot2`),
- creating publication ready plots in R (**ggpubr**, `http://www.sthda.com/english/rpkgs/ggpubr`).

Recently, he published several books on data analysis and visualization:

1. Practical Guide to Cluster Analysis in R (`https://goo.gl/yhhpXh`)
2. Practical Guide To Principal Component Methods in R (`https://goo.gl/d4Doz9`)
3. R Graphics Essentials for Great Data Visualization (`https://goo.gl/oT8Ra6`)
4. Network Analysis and Visulization in R (`https://goo.gl/WBdn4n`)

Part I

Basics

Chapter 1

Introduction to R

R is a free and powerful statistical software for analyzing and visualizing data.

In this chapter, you'll learn how to install R and required packages, as well as, how to import your data into R.

1.1 Install R and RStudio

RStudio is an integrated development environment for R that makes using R easier. R and RStudio can be installed on Windows, MAC OSX and Linux platforms.

1. R can be downloaded and installed from the Comprehensive R Archive Network (CRAN) webpage (`http://cran.r-project.org/`)
2. After installing R software, install also the RStudio software available at: `http://www.rstudio.com/products/RStudio/`.
3. Launch RStudio and start use R inside R studio.

1.2 Install and load required R packages

An R package is a collection of functionalities that extends the capabilities of base R. To use the R code provide in this book, you should install the following R packages:

- `tidyverse` for easy data manipulation and visualization.
- `caret` package for easy machine learning workflow.

1. **Installing packages**:

```
mypkgs <- c("tidyverse", "caret")
install.packages(mypkgs)
```

3. **Load required packages**. After installation, you must first load the package for using the functions in the package. The function `library()` is used for this task. For example, type this:

```
library("tidyverse")
library("caret")
```

Now, we can use R functions available in these package.

If you want to learn more about a given function, say mean(), type this in R console: `?mean`.

1.3 Data format

Your data should be in rectangular format, where columns are variables and rows are observations (individuals or samples).

- Column names should be compatible with R naming conventions. Avoid column with blank space and special characters. Good column names: `long_jump` or `long.jump`. Bad column name: `long jump`.
- Avoid beginning column names with a number. Use letter instead. Good column names: `sport_100m` or `x100m`. Bad column name: `100m`.
- Replace missing values by `NA` (for not available)

For example, your data should look like this:

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
44           5.0         3.5          1.6         0.6     setosa
118          7.7         3.8          6.7         2.2  virginica
61           5.0         2.0          3.5         1.0 versicolor
130          7.2         3.0          5.8         1.6  virginica
```

Read more at: Best Practices in Preparing Data Files for Importing into R[1]

1.4 Import your data in R

First, save your data into txt or csv file formats and import it as follow (you will be asked to choose the file):

```
# Reads tab delimited files (.txt tab)
my_data <- read.delim(file.choose())

# Reads comma (,) delimited files (.csv)
my_data <- read.csv(file.choose())

# Reads semicolon(;) separated files(.csv)
my_data <- read.csv2(file.choose())
```

Read more about how to import data into R at this link: `http://www.sthda.com/english/wiki/importing-data-into-r`

1.5 Demo data sets

R comes with several demo data sets for playing with R functions. The most used R demo data sets include **iris**. To load a demo data set, use the function **data**() as follow. The function `head()` is used to inspect the data.

[1] `http://www.sthda.com/english/wiki/best-practices-in-preparing-data-files-for-importing-into-r`

```
data("iris")   # Loading
head(iris, n = 3)  # Print the first n = 3 rows
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

To learn more about iris data sets, type this:

```
?iris
```

After typing the above R code, you will see the description of `iris` data set.

1.6 Data manipulation

After importing your data in R, you can easily manipulate it using the `tidyverse` packages[2].

After loading tidyverse, you can use the following R functions:

- `filter()`: Pick rows (observations/samples) based on their values.
- `distinct()`: Remove duplicate rows.
- `arrange()`: Reorder the rows.
- `select()`: Select columns (variables) by their names.
- `rename()`: Rename columns.
- `mutate()`: Add/create new variables.
- `summarise()`: Compute statistical summaries (e.g., computing the mean or the sum)
- `group_by()`: Operate on subsets of the data set.

Note that, tidyverse packages allows to use the forward-pipe chaining operator (%>%) for combining multiple operations. For example, x %>% f is equivalent to f(x). Using the pipe (%>%), the output of each operation is passed to the next operation. This makes R programming easy.

Read more at: `http://r4ds.had.co.nz/transform.html`.

1.7 Data visualization

There are different graphic packages available in R for visualizing your data. In this book, we'll create graphics using mainly the ggplot2 system which belongs to the tidyverse packages.

If you are beginer in ggplot2, read our tutorial on R Graphics Essantials[3].

1.8 Close your R/RStudio session

Each time you close R/RStudio, you will be asked whether you want to save the data from your R session. If you decide to save, the data will be available in future R sessions.

[2] `http://r4ds.had.co.nz/transform.html`
[3] `http://www.sthda.com/english/articles/32-r-graphics-essentials/`

Part II

Regression Analysis

Chapter 2

Introduction

Regression analysis (or **regression model**) consists of a set of *machine learning* methods that allow us to predict a continuous outcome variable (y) based on the value of one or multiple predictor variables (x).

Briefly, the goal of regression model is to build a mathematical equation that defines y as a function of the x variables. Next, this equation can be used to predict the outcome (y) on the basis of new values of the predictor variables (x).

Linear regression is the most simple and popular technique for predicting a continuous variable. It assumes a linear relationship between the outcome and the predictor variables. See Chapter 3.

The linear regression equation can be written as `y = b0 + b*x`, where:

- b0 is the intercept,
- b is the regression weight or coefficient associated with the predictor variable x.

Technically, the linear regression coefficients are detetermined so that the error in predicting the outcome value is minimized. This method of computing the beta coefficients is called the **Ordinary Least Squares** method.

When you have multiple predictor variables, say x1 and x2, the regression equation can be written as `y = b0 + b1*x1 + b2*x2`. In some situations, there might be an **interaction effect** between some predictors, that is for example, increasing the value of a predictor variable x1 may increase the effectiveness of the predictor x2 in explaining the variation in the outcome variable. See Chapter 4.

Note also that, linear regression models can incorporate both continuous and **categorical predictor variables**. See Chapter 5.

When you build the linear regression model, you need to **diagnostic** whether linear model is suitable for your data. See Chapter 8.

In some cases, the relationship between the outcome and the predictor variables is not linear. In these situations, you need to build a **non-linear regression**, such as *polynomial and spline regression*. See Chapter 6.

When you have multiple predictors in the regression model, you might want to select the best combination of predictor variables to build an optimal predictive model. This process called **model selection**, consists of comparing multiple models containing different sets of predictors in order to select the best performing model that minimize the prediction error. Linear model

selection approaches include **best subsets regression** (Chapter 16) and **stepwise regression** (Chapter 17)

In some situations, such as in genomic fields, you might have a large multivariate data set containing some correlated predictors. In this case, the information, in the original data set, can be summarized into few new variables (called principal components) that are a linear combination of the original variables. This few principal components can be used to build a linear model, which might be more performant for your data. This approach is know as **principal component-based methods** (Chapter 19), which include: **principal component regression** and **partial least squares regression**.

An alternative method to simplify a large multivariate model is to use **penalized regression** (Chapter 18), which penalizes the model for having too many variables. The most well known penalized regression include **ridge regression** and the **lasso regression**.

You can apply all these different regression models on your data, compare the models and finally select the best approach that explains well your data. To do so, you need some statistical metrics to compare the performance of the different models in explaining your data and in predicting the outcome of new test data.

The best model is defined as the model that has the lowest prediction error. The most popular metrics for comparing regression models, include:

- **Root Mean Squared Error**, which measures the model prediction error. It corresponds to the average difference between the observed known values of the outcome and the predicted value by the model. RMSE is computed as `RMSE = mean((observeds - predicteds)^2) %>% sqrt()`. The lower the RMSE, the better the model.
- **Adjusted R-square**, representing the proportion of variation (i.e., information), in your data, explained by the model. This corresponds to the overall quality of the model. The higher the adjusted R2, the better the model

Note that, the above mentioned metrics should be computed on a new test data that has not been used to train (i.e. build) the model. If you have a large data set, with many records, you can randomly split the data into training set (80% for building the predictive model) and test set or validation set (20% for evaluating the model performance).

One of the most robust and popular approach for estimating a model performance is **k-fold cross-validation**. It can be applied even on a small data set. k-fold cross-validation works as follow:

1. Randomly split the data set into k-subsets (or k-fold) (for example 5 subsets)
2. Reserve one subset and train the model on all other subsets
3. Test the model on the reserved subset and record the prediction error
4. Repeat this process until each of the k subsets has served as the test set.
5. Compute the average of the k recorded errors. This is called the cross-validation error serving as the performance metric for the model.

Taken together, the best model is the model that has the lowest cross-validation error, RMSE.

In this Part, you will learn different methods for regression analysis and we'll provide practical example in **R**. The following tehniques are described:

- Ordinary least squares (Chapter 3)
  - Simple linear regression
  - Multiple linear regression
- Model selection methods:
  - Best subsets regression (Chapter 16)

  - Stepwise regression (Chapter 17)
- Principal component-based methods (Chapter 19):
  - Principal component regression (PCR)
  - Partial least squares regression (PLS)
- Penalized regression (Chapter 18):
  - Ridge regression
  - Lasso regression

2.1 Examples of data set

We'll use three different data sets: `marketing` [datarium package], the built-in R `swiss` data set, and the `Boston` data set available in the `MASS` R package.

2.1.1 marketing data

The `marketing` data set [datarium package] contains the impact of three advertising medias (youtube, facebook and newspaper) on sales. It will be used for predicting sales units on the basis of the amount of money spent in the three advertising medias.

Data are the advertising budget in thousands of dollars along with the sales. The advertising experiment has been repeated 200 times with different budgets and the observed sales have been recorded.

First install the `datarium` package:

```
if(!require(devtools)) install.packages("devtools")
devtools::install_github("kassambara/datarium")
```

Then, load `marketing` data set as follow:

```
data("marketing", package = "datarium")
head(marketing, 3)
```

```
##   youtube facebook newspaper sales
## 1   276.1     45.4      83.0  26.5
## 2    53.4     47.2      54.1  12.5
## 3    20.6     55.1      83.2  11.2
```

2.1.2 swiss data

The `swiss` describes 5 socio-economic indicators observed around 1888 used to predict the fertility score of 47 swiss French-speaking provinces.

Load and inspect the data:

```
data("swiss")
head(swiss, 3)
```

```
##              Fertility Agriculture Examination Education Catholic
## Courtelary        80.2        17.0          15        12     9.96
## Delemont          83.1        45.1           6         9    84.84
## Franches-Mnt      92.5        39.7           5         5    93.40
```

```
##              Infant.Mortality
## Courtelary               22.2
## Delemont                 22.2
## Franches-Mnt             20.2
```

The data contain the following variables:

- Fertility Ig: common standardized fertility measure
- Agriculture: % of males involved in agriculture as occupation
- Examination: % draftees receiving highest mark on army examination
- Education: % education beyond primary school for draftees.
- Catholic: % 'catholic' (as opposed to 'protestant').
- Infant.Mortality: live births who live less than 1 year.

2.1.3 Boston data

`Boston` [in `MASS` package] will be used for predicting the median house value (`mdev`), in Boston Suburbs, using different predictor variables:

- `crim`, per capita crime rate by town
- `zn`, proportion of residential land zoned for lots over 25,000 sq.ft
- `indus`, proportion of non-retail business acres per town
- `chas`, Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- `nox`, nitric oxides concentration (parts per 10 million)
- `rm`, average number of rooms per dwelling
- `age`, proportion of owner-occupied units built prior to 1940
- `dis`, weighted distances to five Boston employment centres
- `rad`, index of accessibility to radial highways
- `tax`, full-value property-tax rate per USD 10,000
- `ptratio`, pupil-teacher ratio by town
- `black`, 1000(B - 0.63)^2 where B is the proportion of blacks by town
- `lstat`, percentage of lower status of the population
- `medv`, median value of owner-occupied homes in USD 1000's

Load and inspect the data:

```
data("Boston", package = "MASS")
head(Boston, 3)
```

```
##      crim zn indus chas   nox   rm  age  dis rad tax ptratio black lstat
## 1 0.00632 18  2.31    0 0.538 6.58 65.2 4.09   1 296    15.3   397  4.98
## 2 0.02731  0  7.07    0 0.469 6.42 78.9 4.97   2 242    17.8   397  9.14
## 3 0.02729  0  7.07    0 0.469 7.18 61.1 4.97   2 242    17.8   393  4.03
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
```

Chapter 3

Linear Regression

3.1 Introduction

Linear regression (or **linear model**) is used to predict a quantitative outcome variable (y) on the basis of one or multiple predictor variables (x) (James et al., 2014, Bruce and Bruce (2017)).

The goal is to build a mathematical formula that defines y as a function of the x variable. Once, we built a statistically significant model, it's possible to use it for predicting future outcome on the basis of new x values.

When you build a regression model, you need to assess the performance of the predictive model. In other words, you need to evaluate how well the model is in predicting the outcome of a new test data that have not been used to build the model.

Two important metrics are commonly used to assess the performance of the predictive regression model:

- **Root Mean Squared Error**, which measures the model prediction error. It corresponds to the average difference between the observed known values of the outcome and the predicted value by the model. RMSE is computed as `RMSE = mean((observeds - predicteds)^2) %>% sqrt()`. The lower the RMSE, the better the model.
- **R-square**, representing the squared correlation between the observed known outcome values and the predicted values by the model. The higher the R2, the better the model.

A simple workflow to build to build a predictive regression model is as follow:

1. Randomly split your data into training set (80%) and test set (20%)
2. Build the regression model using the training set
3. Make predictions using the test set and compute the model accuracy metrics

In this chapter, you will learn:

- the basics and the formula of linear regression,
- how to compute simple and multiple regression models in R,
- how to make predictions of the outcome of new data,
- how to assess the performance of the model

3.2 Formula

The mathematical formula of the linear regression can be written as follow:

```
y = b0 + b1*x + e
```

We read this as "y is modeled as beta1 (`b1`) times `x`, plus a constant beta0 (`b0`), plus an error term `e`."

When you have multiple predictor variables, the equation can be written as `y = b0 + b1*x1 + b2*x2 + ... + bn*xn`, where:

- b0 is the intercept,
- b1, b2, ..., bn are the regression weights or coefficients associated with the predictors x1, x2, ..., xn.
- `e` is the *error term* (also known as the *residual errors*), the part of y that can be explained by the regression model

> Note that, b0, b1, b2, ... and bn are known as the regression beta coefficients or parameters.

The figure below illustrates a simple linear regression model, where:

- the best-fit regression line is in blue
- the intercept (b0) and the slope (b1) are shown in green
- the error terms (e) are represented by vertical red lines

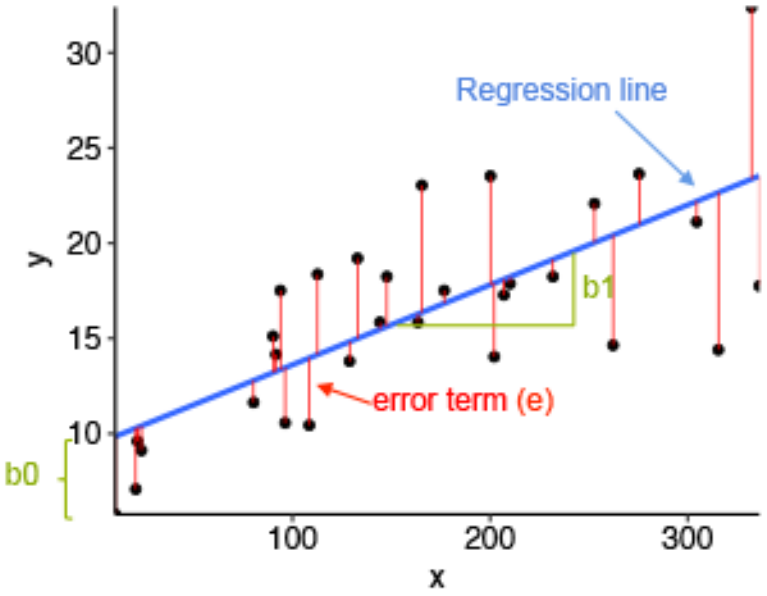


Figure 3.1: Linear regression

From the scatter plot above, it can be seen that not all the data points fall exactly on the fitted regression line. Some of the points are above the blue curve and some are below it; overall, the residual errors (e) have approximately mean zero.

The sum of the squares of the residual errors are called the **Residual Sum of Squares** or **RSS**.

The average variation of points around the fitted regression line is called the **Residual Standard Error** (**RSE**). This is one the metrics used to evaluate the overall quality of the fitted

regression model. The lower the RSE, the better it is.

Since the mean error term is zero, the outcome variable y can be approximately estimated as follow:

```
y ~ b0 + b1*x
```

Mathematically, the beta coefficients (b0 and b1) are determined so that the RSS is as minimal as possible. This method of determining the beta coefficients is technically called **least squares** regression or **ordinary least squares** (OLS) regression.

Once, the beta coefficients are calculated, a t-test is performed to check whether or not these coefficients are significantly different from zero. A non-zero beta coefficients means that there is a significant relationship between the predictors (x) and the outcome variable (y).

3.3 Loading Required R packages

- `tidyverse` for easy data manipulation and visualization
- `caret` for easy machine learning workflow

```
library(tidyverse)
library(caret)
theme_set(theme_bw())
```

3.4 Preparing the data

We'll use the `marketing` data set, introduced in the Chapter 2, for predicting sales units on the basis of the amount of money spent in the three advertising medias (youtube, facebook and newspaper)

We'll randomly split the data into training set (80% for building a predictive model) and test set (20% for evaluating the model). Make sure to set seed for reproducibility.

```
# Load the data
data("marketing", package = "datarium")
# Inspect the data
sample_n(marketing, 3)
```

```
##     youtube facebook newspaper sales
## 158     180     1.56      29.2  12.1
## 82      288     4.92      44.3  14.8
## 175     267     4.08      15.7  13.8
```

```
# Split the data into training and test set
set.seed(123)
training.samples <- marketing$sales %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- marketing[training.samples, ]
test.data <- marketing[-training.samples, ]
```

3.5 Computing linear regression

The R function `lm()` is used to compute linear regression model.

3.5.1 Quick start R code

```
# Build the model
model <- lm(sales ~., data = train.data)
# Summarize the model
summary(model)
# Make predictions
predictions <- model %>% predict(test.data)
# Model performance
# (a) Prediction error, RMSE
RMSE(predictions, test.data$sales)
# (b) R-square
R2(predictions, test.data$sales)
```

3.5.2 Simple linear regression

The **simple linear regression** is used to predict a continuous outcome variable (y) based on one single predictor variable (x).

In the following example, we'll build a simple linear model to predict sales units based on the advertising budget spent on youtube. The regression equation can be written as `sales = b0 + b1*youtube`.

The R function `lm()` can be used to determine the beta coefficients of the linear model, as follow:

```
model <- lm(sales ~ youtube, data = train.data)
summary(model)$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.3839    0.62442    13.4 5.22e-28
## youtube       0.0468    0.00301    15.6 7.84e-34
```

The output above shows the estimate of the regression beta coefficients (column `Estimate`) and their significance levels (column `Pr(>|t|)`. The intercept (`b0`) is 8.38 and the coefficient of youtube variable is 0.046.

The estimated regression equation can be written as follow: `sales = 8.38 + 0.046*youtube`. Using this formula, for each new youtube advertising budget, you can predict the number of sale units.

For example:

- For a youtube advertising budget equal zero, we can expect a sale of 8.38 units.
- For a youtube advertising budget equal 1000, we can expect a sale of 8.38 + 0.046*1000 = 55 units.

Predictions can be easily made using the R function `predict()`. In the following example, we predict sales units for two youtube advertising budget: 0 and 1000.

```
newdata <- data.frame(youtube = c(0,  1000))
model %>% predict(newdata)
```

```
##     1     2
##  8.38 55.19
```

3.5.3 Multiple linear regression

Multiple linear regression is an extension of simple linear regression for predicting an outcome variable (y) on the basis of multiple distinct predictor variables (x).

For example, with three predictor variables (x), the prediction of y is expressed by the following equation: `y = b0 + b1*x1 + b2*x2 + b3*x3`

The regression beta coefficients measure the association between each predictor variable and the outcome. "b_j" can be interpreted as the average effect on y of a one unit increase in "x_j", holding all other predictors fixed.

In this section, we'll build a multiple regression model to predict sales based on the budget invested in three advertising medias: youtube, facebook and newspaper. The formula is as follow: `sales = b0 + b1*youtube + b2*facebook + b3*newspaper`

You can compute the multiple regression model coefficients in R as follow:

```
model <- lm(sales ~ youtube + facebook + newspaper,
            data = train.data)
summary(model)$coef
```

Note that, if you have many predictor variables in your data, you can simply include all the available variables in the model using `~.`:

```
model <- lm(sales ~., data = train.data)
summary(model)$coef
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.39188    0.44062   7.698 1.41e-12
## youtube      0.04557    0.00159  28.630 2.03e-64
## facebook     0.18694    0.00989  18.905 2.07e-42
## newspaper    0.00179    0.00677   0.264 7.92e-01
```

From the output above, the coefficients table shows the beta coefficient estimates and their significance levels. Columns are:

- `Estimate`: the intercept (b0) and the beta coefficient estimates associated to each predictor variable
- `Std.Error`: the standard error of the coefficient estimates. This represents the accuracy of the coefficients. The larger the standard error, the less confident we are about the estimate.
- `t value`: the t-statistic, which is the coefficient estimate (column 2) divided by the standard error of the estimate (column 3)
- `Pr(>|t|)`: The p-value corresponding to the t-statistic. The smaller the p-value, the more significant the estimate is.

As previously described, you can easily make predictions using the R function `predict()`:

```
# New advertising budgets
newdata <- data.frame(
  youtube = 2000, facebook = 1000,
  newspaper = 1000
)
# Predict sales values
model %>% predict(newdata)
```

```
##   1
## 283
```

3.6 Interpretation

Before using a model for predictions, you need to assess the statistical significance of the model. This can be easily checked by displaying the statistical summary of the model.

3.6.1 Model summary

Display the statistical summary of the model as follow:

```
summary(model)
```

```
##
## Call:
## lm(formula = sales ~ ., data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.412  -1.110   0.348   1.422   3.499
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.39188    0.44062    7.70  1.4e-12 ***
## youtube      0.04557    0.00159   28.63  < 2e-16 ***
## facebook     0.18694    0.00989   18.90  < 2e-16 ***
## newspaper    0.00179    0.00677    0.26     0.79
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.12 on 158 degrees of freedom
## Multiple R-squared:  0.89,   Adjusted R-squared:  0.888
## F-statistic:  427 on 3 and 158 DF,  p-value: <2e-16
```

The summary outputs shows 6 components, including:

- **Call**. Shows the function call used to compute the regression model.
- **Residuals**. Provide a quick view of the distribution of the residuals, which by definition have a mean zero. Therefore, the median should not be far from zero, and the minimum and maximum should be roughly equal in absolute value.

- **Coefficients**. Shows the regression beta coefficients and their statistical significance. Predictor variables, that are significantly associated to the outcome variable, are marked by stars.
- **Residual standard error** (RSE), **R-squared** (R2) and the **F-statistic** are metrics that are used to check how well the model fits to our data.

The first step in interpreting the multiple regression analysis is to examine the F-statistic and the associated p-value, at the bottom of model summary.

In our example, it can be seen that p-value of the F-statistic is $< 2.2e\text{-}16$, which is highly significant. This means that, at least, one of the predictor variables is significantly related to the outcome variable.

3.6.2 Coefficients significance

To see which predictor variables are significant, you can examine the coefficients table, which shows the estimate of regression beta coefficients and the associated t-statistic p-values.

```
summary(model)$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.39188    0.44062   7.698 1.41e-12
## youtube      0.04557    0.00159  28.630 2.03e-64
## facebook     0.18694    0.00989  18.905 2.07e-42
## newspaper    0.00179    0.00677   0.264 7.92e-01
```

For a given the predictor, the t-statistic evaluates whether or not there is significant association between the predictor and the outcome variable, that is whether the beta coefficient of the predictor is significantly different from zero.

It can be seen that, changing in youtube and facebook advertising budget are significantly associated to changes in sales while changes in newspaper budget is not significantly associated with sales.

For a given predictor variable, the coefficient (b) can be interpreted as the average effect on y of a one unit increase in predictor, holding all other predictors fixed.

For example, for a fixed amount of youtube and newspaper advertising budget, spending an additional 1 000 dollars on facebook advertising leads to an increase in sales by approximately 0.1885*1000 = 189 sale units, on average.

The youtube coefficient suggests that for every 1 000 dollars increase in youtube advertising budget, holding all other predictors constant, we can expect an increase of 0.045*1000 = 45 sales units, on average.

We found that newspaper is not significant in the multiple regression model. This means that, for a fixed amount of youtube and newspaper advertising budget, changes in the newspaper advertising budget will not significantly affect sales units.

As the newspaper variable is not significant, it is possible to remove it from the model:

```
model <- lm(sales ~ youtube + facebook, data = train.data)
summary(model)
```

```
##
```

```
## Call:
## lm(formula = sales ~ youtube + facebook, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.481  -1.104   0.349   1.423   3.486
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.43446    0.40877     8.4  2.3e-14 ***
## youtube      0.04558    0.00159    28.7  < 2e-16 ***
## facebook     0.18788    0.00920    20.4  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.11 on 159 degrees of freedom
## Multiple R-squared:  0.89,   Adjusted R-squared:  0.889
## F-statistic:  644 on 2 and 159 DF,  p-value: <2e-16
```

Finally, our model equation can be written as follow: `sales = 3.43+ 0.045*youtube + 0.187*facebook`.

3.6.3 Model accuracy

Once you identified that, at least, one predictor variable is significantly associated to the outcome, you should continue the diagnostic by checking how well the model fits the data. This process is also referred to as the *goodness-of-fit*

The overall quality of the linear regression fit can be assessed using the following three quantities, displayed in the model summary:

1. Residual Standard Error (RSE),
2. R-squared (R2) and adjusted R2,
3. F-statistic, which has been already described in the previous section

```
##    rse r.squared f.statistic  p.value
## 1 2.11      0.89         644 5.64e-77
```

1. **Residual standard error** (RSE).

The RSE (or model *sigma*), corresponding to the prediction error, represents roughly the average difference between the observed outcome values and the predicted values by the model. The lower the RSE the best the model fits to our data.

Dividing the RSE by the average value of the outcome variable will give you the prediction error rate, which should be as small as possible.

In our example, using only youtube and facebook predictor variables, the RSE = 2.11, meaning that the observed sales values deviate from the predicted values by approximately 2.11 units in average.

This corresponds to an error rate of 2.11/mean(train.data$sales) = 2.11/16.77 = 13%, which is low.

2. **R-squared and Adjusted R-squared**:

The R-squared (R2) ranges from 0 to 1 and represents the proportion of variation in the outcome variable that can be explained by the model predictor variables.

For a simple linear regression, R2 is the square of the Pearson correlation coefficient between the outcome and the predictor variables. In multiple linear regression, the R2 represents the correlation coefficient between the observed outcome values and the predicted values.

The R2 measures, how well the model fits the data. The higher the R2, the better the model. However, a problem with the R2, is that, it will always increase when more variables are added to the model, even if those variables are only weakly associated with the outcome (James et al., 2014). A solution is to adjust the R2 by taking into account the number of predictor variables.

The adjustment in the "Adjusted R Square" value in the summary output is a correction for the number of x variables included in the predictive model.

So, you should mainly consider the adjusted R-squared, which is a penalized R2 for a higher number of predictors.

- An (adjusted) R2 that is close to 1 indicates that a large proportion of the variability in the outcome has been explained by the regression model.
- A number near 0 indicates that the regression model did not explain much of the variability in the outcome.

In our example, the adjusted R2 is 0.88, which is good.

3. **F-Statistic**:

Recall that, the F-statistic gives the overall significance of the model. It assess whether at least one predictor variable has a non-zero coefficient.

In a simple linear regression, this test is not really interesting since it just duplicates the information given by the t-test, available in the coefficient table.

The F-statistic becomes more important once we start using multiple predictors as in multiple linear regression.

A large F-statistic will corresponds to a statistically significant p-value ($p < 0.05$). In our example, the F-statistic equal 644 producing a p-value of 1.46e-42, which is highly significant.

3.7 Making predictions

We'll make predictions using the test data in order to evaluate the performance of our regression model.

The procedure is as follow:

1. Predict the sales values based on new advertising budgets in the test data
2. Assess the model performance by computing:
    - The prediction error RMSE (Root Mean Squared Error), representing the average difference between the observed known outcome values in the test data and the predicted outcome values by the model. The lower the RMSE, the better the model.
    - The R-square (R2), representing the correlation between the observed outcome values and the predicted outcome values. The higher the R2, the better the model.

```
# Make predictions
predictions <- model %>% predict(test.data)
# Model performance
# (a) Compute the prediction error, RMSE
RMSE(predictions, test.data$sales)
```

```
## [1] 1.58
```

```
# (b) Compute R-square
R2(predictions, test.data$sales)
```

```
## [1] 0.938
```

From the output above, the R2 is 0.93, meaning that the observed and the predicted outcome values are highly correlated, which is very good.

The prediction error RMSE is 1.58, representing an error rate of 1.58/mean(test.data$sales) = 1.58/17 = 9.2%, which is good.

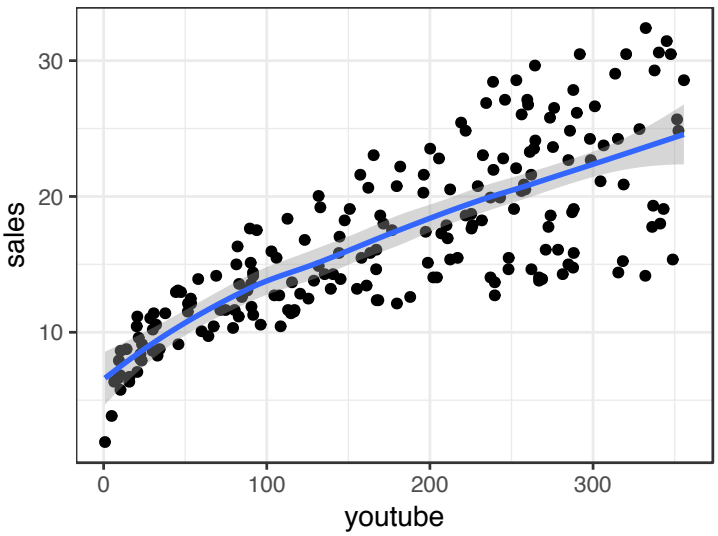
3.8 Discussion

This chapter describes the basics of linear regression and provides practical examples in R for computing simple and multiple linear regression models. We also described how to assess the performance of the model for predictions.

Note that, linear regression assumes a linear relationship between the outcome and the predictor variables. This can be easily checked by creating a scatter plot of the outcome variable vs the predictor variable.

For example, the following R code displays sales units versus youtube advertising budget. We'll also add a smoothed line:

```
ggplot(marketing, aes(x = youtube, y = sales)) +
  geom_point() +
  stat_smooth()
```



The graph above shows a linearly increasing relationship between the `sales` and the `youtube`

variables, which is a good thing.

In addition to the linearity assumptions, the linear regression method makes many other assumptions about your data (see Chapter 8). You should make sure that these assumptions hold true for your data.

Potential problems, include: a) the presence of influential observations in the data (Chapter 8), non-linearity between the outcome and some predictor variables (6) and the presence of strong correlation between predictor variables (Chapter 9).

Chapter 4

Interaction Effects in Multiple Regression

4.1 Introduction

This chapter describes how to compute multiple linear regression with **interaction effects**.

Previously, we have described how to build a multiple linear regression model (Chapter 3) for predicting a continuous outcome variable (y) based on multiple predictor variables (x).

For example, to predict sales, based on advertising budgets spent on youtube and facebook, the model equation is `sales = b0 + b1*youtube + b2*facebook`, where, b0 is the intercept; b1 and b2 are the regression coefficients associated respectively with the predictor variables youtube and facebook.

The above equation, also known as *additive model*, investigates only the main effects of predictors. It assumes that the relationship between a given predictor variable and the outcome is independent of the other predictor variables (James et al., 2014, Bruce and Bruce (2017)).

Considering our example, the additive model assumes that, the effect on sales of youtube advertising is independent of the effect of facebook advertising.

This assumption might not be true. For example, spending money on facebook advertising may increase the effectiveness of youtube advertising on sales. In marketing, this is known as a synergy effect, and in statistics it is referred to as an interaction effect (James et al., 2014).

In this chapter, you'll learn:

- the equation of multiple linear regression with interaction
- R codes for computing the regression coefficients associated with the main effects and the interaction effects
- how to interpret the interaction effect

4.2 Equation

The multiple linear regression equation, with interaction effects between two predictors (x1 and x2), can be written as follow:

```
y = b0 + b1*x1 + b2*x2 + b3*(x1*x2)
```

Considering our example, it becomes:

```
sales = b0 + b1*youtube + b2*facebook + b3*(youtube*facebook)
```

This can be also written as:

```
sales = b0 + (b1 + b3*facebook)*youtube + b2*facebook
```

or as:

```
sales = b0 + b1*youtube + (b2 +b3*youtube)*facebook
```

`b3` can be interpreted as the increase in the effectiveness of youtube advertising for a one unit increase in facebook advertising (or vice-versa).

In the following sections, you will learn how to compute the regression coefficients in R.

4.3 Loading Required R packages

- `tidyverse` for easy data manipulation and visualization
- `caret` for easy machine learning workflow

```
library(tidyverse)
library(caret)
```

4.4 Preparing the data

We'll use the `marketing` data set, introduced in the Chapter 2, for predicting sales units on the basis of the amount of money spent in the three advertising medias (youtube, facebook and newspaper)

We'll randomly split the data into training set (80% for building a predictive model) and test set (20% for evaluating the model).

```
# Load the data
data("marketing", package = "datarium")
# Inspect the data
sample_n(marketing, 3)
```

```
##     youtube facebook newspaper sales
## 158     180     1.56      29.2  12.1
## 82      288     4.92      44.3  14.8
## 175     267     4.08      15.7  13.8
```

```
# Split the data into training and test set
set.seed(123)
training.samples <- marketing$sales %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- marketing[training.samples, ]
test.data <- marketing[-training.samples, ]
```

4.5 Computation

4.5.1 Additive model

The standard linear regression model can be computed as follow:

```
# Build the model
model1 <- lm(sales ~ youtube + facebook, data = train.data)
# Summarize the model
summary(model1)
```

```
##
## Call:
## lm(formula = sales ~ youtube + facebook, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.481  -1.104   0.349   1.423   3.486
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.43446    0.40877     8.4  2.3e-14 ***
## youtube      0.04558    0.00159    28.7  < 2e-16 ***
## facebook     0.18788    0.00920    20.4  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.11 on 159 degrees of freedom
## Multiple R-squared:  0.89,   Adjusted R-squared:  0.889
## F-statistic:  644 on 2 and 159 DF,  p-value: <2e-16
```

```
# Make predictions
predictions <- model1 %>% predict(test.data)
# Model performance
# (a) Prediction error, RMSE
RMSE(predictions, test.data$sales)
```

```
## [1] 1.58
```

```
# (b) R-square
R2(predictions, test.data$sales)
```

```
## [1] 0.938
```

4.5.2 Interaction effects

In R, you include interactions between variables using the * operator:

```
# Build the model
# Use this:
model2 <- lm(sales ~ youtube + facebook + youtube:facebook,
             data = marketing)
# Or simply, use this:
```

```
model2 <- lm(sales ~ youtube*facebook, data = train.data)

# Summarize the model
summary(model2)
```

```
##
## Call:
## lm(formula = sales ~ youtube * facebook, data = train.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.438 -0.482  0.231  0.748  1.860
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.90e+00   3.28e-01   24.06   <2e-16 ***
## youtube          1.95e-02   1.64e-03   11.90   <2e-16 ***
## facebook         2.96e-02   9.83e-03    3.01    0.003 **
## youtube:facebook 9.12e-04   4.84e-05   18.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.18 on 158 degrees of freedom
## Multiple R-squared:  0.966,  Adjusted R-squared:  0.966
## F-statistic: 1.51e+03 on 3 and 158 DF,  p-value: <2e-16
```

```
# Make predictions
predictions <- model2 %>% predict(test.data)
# Model performance
# (a) Prediction error, RMSE
RMSE(predictions, test.data$sales)
```

```
## [1] 0.963
```

```
# (b) R-square
R2(predictions, test.data$sales)
```

```
## [1] 0.982
```

4.6 Interpretation

It can be seen that all the coefficients, including the interaction term coefficient, are statistically significant, suggesting that there is an interaction relationship between the two predictor variables (youtube and facebook advertising).

Our model equation looks like this:

```
sales = 7.89 + 0.019*youtube + 0.029*facebook + 0.0009*youtube*facebook
```

We can interpret this as an increase in youtube advertising of 1000 dollars is associated with increased sales of `(b1 + b3*facebook)*1000 = 19 + 0.9*facebook units`. And an increase in facebook advertising of 1000 dollars will be associated with an increase in sales of `(b2 + b3*youtube)*1000 = 28 + 0.9*youtube units`.

Note that, sometimes, it is the case that the interaction term is significant but not the main effects. The hierarchical principle states that, if we include an interaction in a model, we should also include the main effects, even if the p-values associated with their coefficients are not significant [@james2014].

4.7 Comparing the additive and the interaction models

The prediction error RMSE of the interaction model is 0.963, which is lower than the prediction error of the additive model (1.58).

Additionally, the R-square (R2) value of the interaction model is 98% compared to only 93% for the additive model.

These results suggest that the model with the interaction term is better than the model that contains only main effects. So, for this specific data, we should go for the model with the interaction model.

4.8 Discussion

This chapter describes how to compute multiple linear regression with interaction effects. Interaction terms should be included in the model if they are significantly.

Chapter 5

Regression with Categorical Variables

5.1 Introduction

This chapter describes how to compute **regression with categorical variables**.

Categorical variables (also known as *factor* or *qualitative variables*) are variables that classify observations into groups. They have a limited number of different values, called levels. For example the gender of individuals are a categorical variable that can take two levels: Male or Female.

Regression analysis requires numerical variables. So, when a researcher wishes to include a categorical variable in a regression model, supplementary steps are required to make the results interpretable.

In these steps, the categorical variables are recoded into a set of separate binary variables. This recoding is called "dummy coding" and leads to the creation of a table called *contrast matrix*. This is done automatically by statistical software, such as R.

Here, you'll learn how to build and interpret a linear regression model with categorical predictor variables. We'll also provide practical examples in R.

5.2 Loading Required R packages

- `tidyverse` for easy data manipulation and visualization

```
library(tidyverse)
```

5.3 Example of data set

We'll use the `Salaries` data set [`car` package], which contains 2008-09 nine-month academic salary for Assistant Professors, Associate Professors and Professors in a college in the U.S.

The data were collected as part of the on-going effort of the college's administration to monitor salary differences between male and female faculty members.

```
# Load the data
data("Salaries", package = "car")
# Inspect the data
sample_n(Salaries, 3)
```

```
##         rank discipline yrs.since.phd yrs.service  sex salary
## 313     Prof          A            29          19 Male  94350
## 162     Prof          B            26          19 Male 176500
## 349 AsstProf          B             4           3 Male  80139
```

5.4 Categorical variables with two levels

Recall that, the regression equation, for predicting an outcome variable (y) on the basis of a predictor variable (x), can be simply written as `y = b0 + b1*x`. `b0` and 'b1 are the regression beta coefficients, representing the intercept and the slope, respectively.

Suppose that, we wish to investigate differences in salaries between males and females.

Based on the gender variable, we can create a new dummy variable that takes the value:

- `1` if a person is male
- `0` if a person is female

and use this variable as a predictor in the regression equation, leading to the following the model:

- `b0 + b1` if person is male
- `bo` if person is female

The coefficients can be interpreted as follow:

1. `b0` is the average salary among females,
2. `b0 + b1` is the average salary among males,
3. and `b1` is the average difference in salary between males and females.

For simple demonstration purpose, the following example models the salary difference between males and females by computing a simple linear regression model on the `Salaries` data set [`car` package]. R creates dummy variables automatically:

```
# Compute the model
model <- lm(salary ~ sex, data = Salaries)
summary(model)$coef
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   101002       4809   21.00 2.68e-66
## sexMale        14088       5065    2.78 5.67e-03
```

From the output above, the average salary for female is estimated to be 101002, whereas males are estimated a total of $101002 + 14088 = 115090$. The p-value for the dummy variable `sexMale` is very significant, suggesting that there is a statistical evidence of a difference in average salary between the genders.

The `contrasts()` function returns the coding that R have used to create the dummy variables:

```
contrasts(Salaries$sex)
```

```
##        Male
## Female    0
## Male      1
```

R has created a sexMale dummy variable that takes on a value of 1 if the sex is Male, and 0 otherwise. The decision to code males as 1 and females as 0 (baseline) is arbitrary, and has no effect on the regression computation, but does alter the interpretation of the coefficients.

You can use the function `relevel()` to set the baseline category to males as follow:

```
Salaries <- Salaries %>%
  mutate(sex = relevel(sex, ref = "Male"))
```

The output of the regression fit becomes:

```
model <- lm(salary ~ sex, data = Salaries)
summary(model)$coef
```

```
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   115090       1587   72.50 2.46e-230
## sexFemale     -14088       5065   -2.78  5.67e-03
```

The fact that the coefficient for `sexFemale` in the regression output is negative indicates that being a Female is associated with decrease in salary (relative to Males).

Now the estimates for `bo` and `b1` are 115090 and -14088, respectively, leading once again to a prediction of average salary of 115090 for males and a prediction of 115090 - 14088 = 101002 for females.

Alternatively, instead of a 0/1 coding scheme, we could create a dummy variable -1 (male) / 1 (female) . This results in the model:

- `b0 - b1` if person is male
- `b0 + b1` if person is female

So, if the categorical variable is coded as -1 and 1, then if the regression coefficient is positive, it is subtracted from the group coded as -1 and added to the group coded as 1. If the regression coefficient is negative, then addition and subtraction is reversed.

5.5 Categorical variables with more than two levels

Generally, a categorical variable with n levels will be transformed into n-1 variables each with two levels. These n-1 new variables contain the same information than the single variable. This recoding creates a table called **contrast matrix**.

For example `rank` in the `Salaries` data has three levels: “AsstProf”, “AssocProf” and “Prof”. This variable could be dummy coded into two variables, one called AssocProf and one Prof:

- If rank = AssocProf, then the column AssocProf would be coded with a 1 and Prof with a 0.
- If rank = Prof, then the column AssocProf would be coded with a 0 and Prof would be coded with a 1.
- If rank = AsstProf, then both columns “AssocProf” and “Prof” would be coded with a 0.

This dummy coding is automatically performed by R. For demonstration purpose, you can use the function `model.matrix()` to create a contrast matrix for a factor variable:

```
res <- model.matrix(~rank, data = Salaries)
head(res[, -1])
```

```
##   rankAssocProf rankProf
## 1             0        1
## 2             0        1
## 3             0        0
## 4             0        1
## 5             0        1
## 6             1        0
```

When building linear model, there are different ways to encode categorical variables, known as contrast coding systems. The default option in R is to use the first level of the factor as a reference and interpret the remaining levels relative to this level.

Note that, ANOVA (analyse of variance) is just a special case of linear model where the predictors are categorical variables. And, because R understands the fact that ANOVA and regression are both examples of linear models, it lets you extract the classic ANOVA table from your regression model using the R base `anova()` function or the `Anova()` function [in `car` package]. We generally recommend the `Anova()` function because it automatically takes care of unbalanced designs.

The results of predicting salary from using a multiple regression procedure are presented below.

```
library(car)
model2 <- lm(salary ~ yrs.service + rank + discipline + sex,
             data = Salaries)
Anova(model2)
```

```
## Anova Table (Type II tests)
##
## Response: salary
##               Sum Sq  Df F value  Pr(>F)
## yrs.service 3.24e+08   1    0.63    0.43
## rank        1.03e+11   2  100.26 < 2e-16 ***
## discipline  1.74e+10   1   33.86 1.2e-08 ***
## sex         7.77e+08   1    1.51    0.22
## Residuals   2.01e+11 391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Taking other variables (yrs.service, rank and discipline) into account, it can be seen that the categorical variable sex is no longer significantly associated with the variation in salary between individuals. Significant variables are rank and discipline.

If you want to interpret the contrasts of the categorical variable, type this:

```
summary(model2)
```

```
##
## Call:
## lm(formula = salary ~ yrs.service + rank + discipline + sex,
##     data = Salaries)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -64202 -14255  -1533  10571  99163
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    73122.9     3245.3   22.53  < 2e-16 ***
## yrs.service      -88.8      111.6   -0.80  0.42696
## rankAssocProf  14560.4     4098.3    3.55  0.00043 ***
## rankProf       49159.6     3834.5   12.82  < 2e-16 ***
## disciplineB    13473.4     2315.5    5.82  1.2e-08 ***
## sexFemale      -4771.2     3878.0   -1.23  0.21931
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22700 on 391 degrees of freedom
## Multiple R-squared:  0.448,  Adjusted R-squared:  0.441
## F-statistic: 63.4 on 5 and 391 DF,  p-value: <2e-16
```

For example, it can be seen that being from discipline B (applied departments) is significantly associated with an average increase of 13473.38 in salary compared to discipline A (theoretical departments).

5.6 Discussion

In this chapter we described how categorical variables are included in linear regression model. As regression requires numerical inputs, categorical variables need to be recoded into a set of binary variables.

We provide practical examples for the situations where you have categorical variables containing two or more levels.

Note that, for categorical variables with a large number of levels it might be useful to group together some of the levels.

Some categorical variables have levels that are ordered. They can be converted to numerical values and used as is. For example, if the professor grades ("AsstProf", "AssocProf" and "Prof") have a special meaning, you can convert them into numerical values, ordered from low to high, corresponding to higher-grade professors.