Lab 1: Simple Linear Regression in R
STAT 632, Spring 2020

1 Data Set

For this lab we consider a data set called `fandango` from the R package `fivethirtyeight`. The data set contains a sample of movie ratings from Fandango, Rotten Tomatoes, Metacritic, and IMDb. The data was used for the FiveThirtyEight article Be Suspicious Of Online Movie Ratings, Especially Fandango's. The article argues that the movie ratings on Fandango tend to be higher than other websites.

First, to install the package run the command:

```
install.packages("fivethirtyeight")
```

You only need to install a package on your computer once. Next, to load the contents of the package into RStudio run the command:

```
library(fivethirtyeight)
```

Note that you need to run the `library()` command each time you open RStudio and want to use a particular package. The `fandango` data set should now be available in RStudio. Try previewing the first several rows of this data frame:

```
head(fandango)
```

You can also read about this data set in the help menu by entering the command:

```
help(fandango)
```

In this lab, we will use simple linear regression to model the relationship between the variables `fandango_ratingvalue` and `imdb_norm`. Note that the IMDb ratings have been normalized between 0 and 5 so that they can be compared directly with Fandango ratings.

A good place to start, before actually fitting a model, is to look at some descriptive statistics of the variables.

```
summary(fandango$fandango_ratingvalue)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.700   3.500   3.900   3.845   4.200   4.800
```

```r
summary(fandango$imdb_norm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.150   3.450   3.368   3.700   4.300
```

2 Simple Linear Regression Model

Use the `lm()` function to fit a simple linear regression model.

```r
lm1 <- lm(fandango_ratingvalue ~ imdb_norm, data = fandango)
```

The function uses the formula notation $y \sim x$, where y is the response variable (`fandango_ratingvalue`) and x is the explanatory variable (`imdb_norm`).

Use the `summary()` function to print out important information about the linear regression model we just fit.

```r
summary(lm1)
```

```
##
## Call:
## lm(formula = fandango_ratingvalue ~ imdb_norm, data = fandango)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99653 -0.26402 -0.01228  0.30208  1.07577
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.72393    0.23784   7.248 2.37e-11 ***
## imdb_norm    0.62974    0.06991   9.008 1.16e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4035 on 144 degrees of freedom
## Multiple R-squared:  0.3604,Adjusted R-squared:  0.356
## F-statistic: 81.15 on 1 and 144 DF,  p-value: 1.155e-15
```

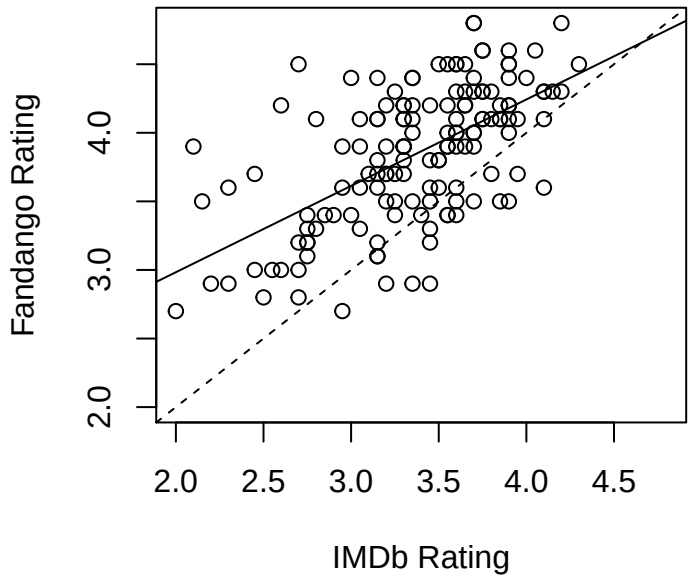
The least squares estimates of the slope and intercept are given in the `Coefficients` table of the summary output. The equation of the least squares regression line can therefore be written as

$$\hat{y} = 1.7239 + 0.62974x$$

The summary output gives an $R^2 = 0.3604$. This means that 36.04% of the variation in fandango ratings can be explained by the IMDb ratings.

Next, we can make a scatterplot of our data and superimpose the least squares regression line. The 1-1 line (i.e., the $y = x$ line) is also plotted for comparison.

```
plot(fandango_ratingvalue ~ imdb_norm, data = fandango,
     ylab = "Fandango Rating", xlab="IMDb Rating",
     xlim=c(2,4.8), ylim = c(2,4.8))
abline(lm1) # add regression line
abline(0, 1, lty=2) # add 1-1 line
```



The scatter plot shows that Fandango movie ratings tend to be higher than IMDb; this agrees with the conclusions from the FiveThirtyEight article. Specifically, the regression line is above the 1-1 line, indicating that Fandango ratings are higher than IMDb, on average, especially for movies that are rated poorly on IMDb.

3 Attributes from `lm` Objects

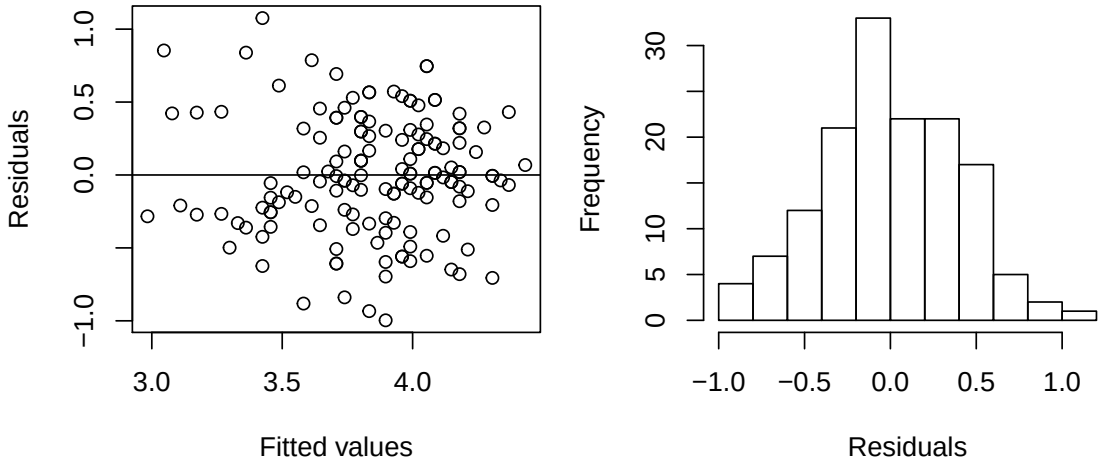
The linear model object we created has various attributes that we can extract by name.

```
attributes(lm1)

## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
##
## $class
## [1] "lm"

# extract coefficients (parameter estimates)
coef(lm1)

## (Intercept)   imdb_norm
##    1.723934    0.629739

# extract residuals
head(resid(lm1))

##           1           2           3           4           5
##  0.320083951  0.540492586  0.320083951  1.075770701 -0.329768456
##           6
##  0.009005639

# extract fitted (predicted) values
head(predict(lm1))

##        1        2        3        4        5        6
## 4.179916 3.959507 4.179916 3.424229 3.329768 3.990994

# 95% confidence intervals for intercept and slope
confint(lm1)

##                 2.5 %    97.5 %
## (Intercept) 1.2538337 2.1940345
## imdb_norm   0.4915635 0.7679144
```

3.1 Model Diagnostics

A useful diagnostic is a plot of the residuals versus the fitted values. Use the `predict()` function to extract the fitted values ($\hat{y}_i$); and use the `resid()` function to extract the residuals ($\hat{e}_i = y_i - \hat{y}_i$).

```
par(mar=c(4,4,1,1), mfrow=c(1,2)) # adjust margins, and split graphics pane
# residual plot
plot(predict(lm1), resid(lm1), xlab="Fitted values", ylab="Residuals")
abline(h=0)
# histogram of residuals
hist(resid(lm1), main="", xlab="Residuals")
```



The points in the residual plot appear randomly scattered around 0. There are no outliers, or indications of nonlinearity or nonconstant variance. The histogram of the residuals also appears normally distributed. Therefore, the conditions for simple linear regression appear to be well satisfied.

4 Making Predictions

We can use the `predict()` function to make predictions at new values of the explanatory variable. Recall, that the least squares regression line for predicting a movie's Fandango rating from the IMDb normalized rating is

$$\hat{y} = 1.7239 + 0.62974x$$

For example, the predicted Fandango rating for a movie with an IMDb normalized rating of $x = 4$ is given by

$$\hat{y} = 1.7239 + 0.62974(4) = 4.24286$$

We can do this calculation in R with the following command:

```
new_x <- data.frame(imdb_norm = 4)
predict(lm1, newdata = new_x)

##       1
## 4.24289
```

which gives the same result as the manual calculation.

We can also use this approach to make several predictions at once. For example, the following command gives the predicted Fandango ratings for movies with IMDb normalized ratings of $x = 2.5, 3.5, 4.5$.

```
new_x <- data.frame(imdb_norm = c(2.5,3.5,4.5))
predict(lm1, newdata = new_x)

##        1        2        3
## 3.298282 3.928020 4.557759
```

The `predict()` function can also be used to compute a 95% prediction interval (more on this next class).

```
new_x <- data.frame(imdb_norm = 4)
predict(lm1, newdata = new_x, interval = "prediction")

##       fit      lwr      upr
## 1 4.24289 3.437823 5.047957
```

Here's how to make the scatterplot with the least squares line using `ggplot2`.

```
library(ggplot2)
ggplot(fandango, aes(imdb_norm, fandango_ratingvalue)) +
  geom_point() + geom_smooth(method = "lm", se = FALSE) +
  xlab("IMDb Rating") + ylab("Fandango Rating")
```