LAB 3

Michael Kummer

Simple Linear Regression

Setup

1) Open a new Rscript and save it.

2) Install the necessary packages. If you have already installed the packages before, you don't need to repeat this step. The packages we are going to use today are:

- data.table
- ggplot2
- stargazer
- Hmisc (**NEW - requires survival and lattice**)

3) Activate the packages:

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.1.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
library(stargazer)
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
##library(Hmisc)
```

4) Set your working directory.

```
setwd("C:/TopicsDig/Labs")
# change the file's path to your own
```

Load the data

Download the data for today's lab from the course's moodle page and then copy the file from your downloads folder into your working directory.

Attention: You should not open your csv file in excel and then save it, as it may corrupt the file. You should simply save the csv file directly in your working directory or copy-paste it from your downloads folder to your working directory.

You can use the list files command to check if the data file is stored in your working directory.

```
list.files()
```

```
## [1] "ECO-R002-lab03-hypothesis-testing-examples.pdf"
## [2] "ECO-R002-lab03-hypothesis-testing-examples.Rmd"
## [3] "ECO-R002-lab03.Rmd"
## [4] "ECO-R002-lab03_SimpleRegression.pdf"
## [5] "Lab Home Assignment 3or4.pdf"
## [6] "Lab_03.R"
## [7] "Lab_03.Rmd"
## [8] "Lab_03_files"
## [9] "Lab03.Rmd"
```

This is a ".csv" file. In order to load a csv file into R we need to use the function "read.csv()". When using this function we must name the data while loading it. For now, I will name the data "sales".

```
sales <- read.csv("C:/TopicsDig/Labs/datasets/sales-data.csv")
```

The next step is to convert the data into the data.table format. We do this using the function data.table and, for convenience, we will add "dt" to the file name so that we know it is in the data.table format:

```
dt.sales <- data.table(sales)
```

Now we have two copies of the same data set in our work environment, so we can remove the object "sales".

```
rm(sales)
```

Alternatively, we can load the data and convert it to the data.table format all in one step:

```
dt.sales <- data.table(read.csv("C:/TopicsDig/Labs/datasets/sales-data.csv"))
```

Explore the data

There are multiple ways we can learn about the contents of this data set. For instance:

```
ncol(dt.sales)
```

```
## [1] 2
```

```
nrow(dt.sales)
```

```
## [1] 22
```

```
colnames(dt.sales)
```

```
## [1] "sales"       "advertising"
```

```
head(dt.sales)
```

```
##    sales advertising
## 1:   999          48
## 2:  1169          50
## 3:  1036          68
## 4:   643          52
## 5:   988          76
## 6:  1076          74
```

```
stargazer(dt.sales, type = "text")
```

```
##
## ==========================================
## Statistic   N    Mean    St. Dev. Min  Max
```

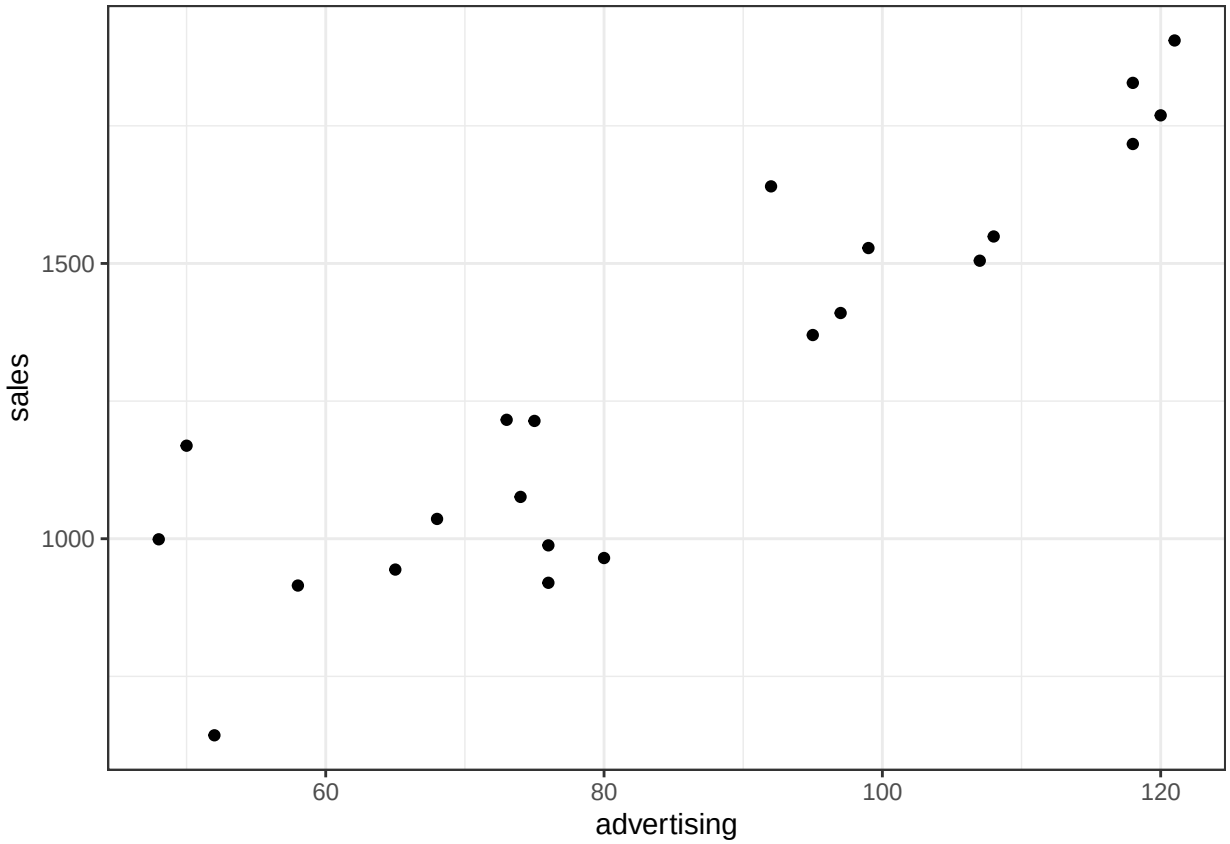
```
## ------------------------------------------
## sales       22 1,286.636 353.621  643 1,905
## advertising 22  85.000    23.759  48   121
## ------------------------------------------
```

```
summary(dt.sales)
```

```
##      sales         advertising
##  Min.   : 643.0   Min.   : 48.00
##  1st Qu.: 990.8   1st Qu.: 69.25
##  Median :1215.0   Median : 78.00
##  Mean   :1286.6   Mean   : 85.00
##  3rd Qu.:1543.8   3rd Qu.:105.00
##  Max.   :1905.0   Max.   :121.00
```

And we can use plots to explore the data. In this case, we are dealing with two continuous variables so it makes sense to use a scatter plot.

```
qplot( data = dt.sales
       , x = advertising
       , y = sales
       , geom = "point") +
  theme_bw()
```



What relationship do we observe? In this case, we can see that the two variables are positively correlated. We can use the function "cor" to get the exact correlation coefficient:

```
dt.sales[, cor(sales, advertising)]
```

```
## [1] 0.9003409
```

In order to know whether the correlation coefficient is statistically significant we can use:

```
#dt.sales[, rcorr(sales, advertising)]
```

This command line results in two tables. The first is the correlation matrix. It gives us the correlation coefficients between all the listed variables. The second table gives us the p-values corresponding to the correlation coefficients in the corresponding positions in the correlation matrix. If the p-value is small enough, we reject the null hypothesis that the correlation coefficient is equal to 0: $H0 : \rho = 0$ vs. $H1 : \rho \neq 0$. If the two variables are normally distributed, the test statistic is: $t_{n-2} = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$ (with a t-distribution with n-2 degrees of freedom).

Simple Regression Analysis

In order to know what is the change in sales that we can expect from increasing our advertising investment by one dollar we can create a simple regression model. In R, we use the "lm" function for creating linear models. Inside this function we will write our equation: y ~ x.

```
lm.sales <- lm(sales ~ advertising, data=dt.sales)
```

In order to get the coefficient estimates, significance levels, and the measures for the quality of our model, we use the "summary" function.

```
summary(lm.sales)
```

```
##
## Call:
## lm(formula = sales ~ advertising, data = dt.sales)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -254.63  -71.78  -17.34   82.97  351.38
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  147.590    127.618   1.157    0.261
## advertising   13.401      1.448   9.252 1.15e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 157.7 on 20 degrees of freedom
## Multiple R-squared:  0.8106, Adjusted R-squared:  0.8011
## F-statistic:  85.6 on 1 and 20 DF,  p-value: 1.15e-08
```

Alternatively:

```
stargazer(lm.sales, type = "text")
```

```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                                sales
## -----------------------------------------------
## advertising                  13.401***
##                               (1.448)
##
```

```
## Constant                       147.590
##                               (127.618)
##
## ------------------------------------------------
## Observations                     22
## R2                              0.811
## Adjusted R2                     0.801
## Residual Std. Error       157.691 (df = 20)
## F Statistic            85.604*** (df = 1; 20)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

We can also extract the parameters of the estimated regression equation using the coefficients function.

```
coeffs = coefficients(lm.sales)
coeffs
```

```
## (Intercept) advertising
##   147.59047    13.40054
```

Interpretation

$\beta_0 = 147.6$ gives us the average sales level when the advertising investment is zero.

$\beta_1 = 13.4$ gives us the increase in sales that results from a 1 unit (dollar) increase in advertising investment.

R^2 gives us the percentage of the variation in sales that is explained by the variation in the advertising investment.

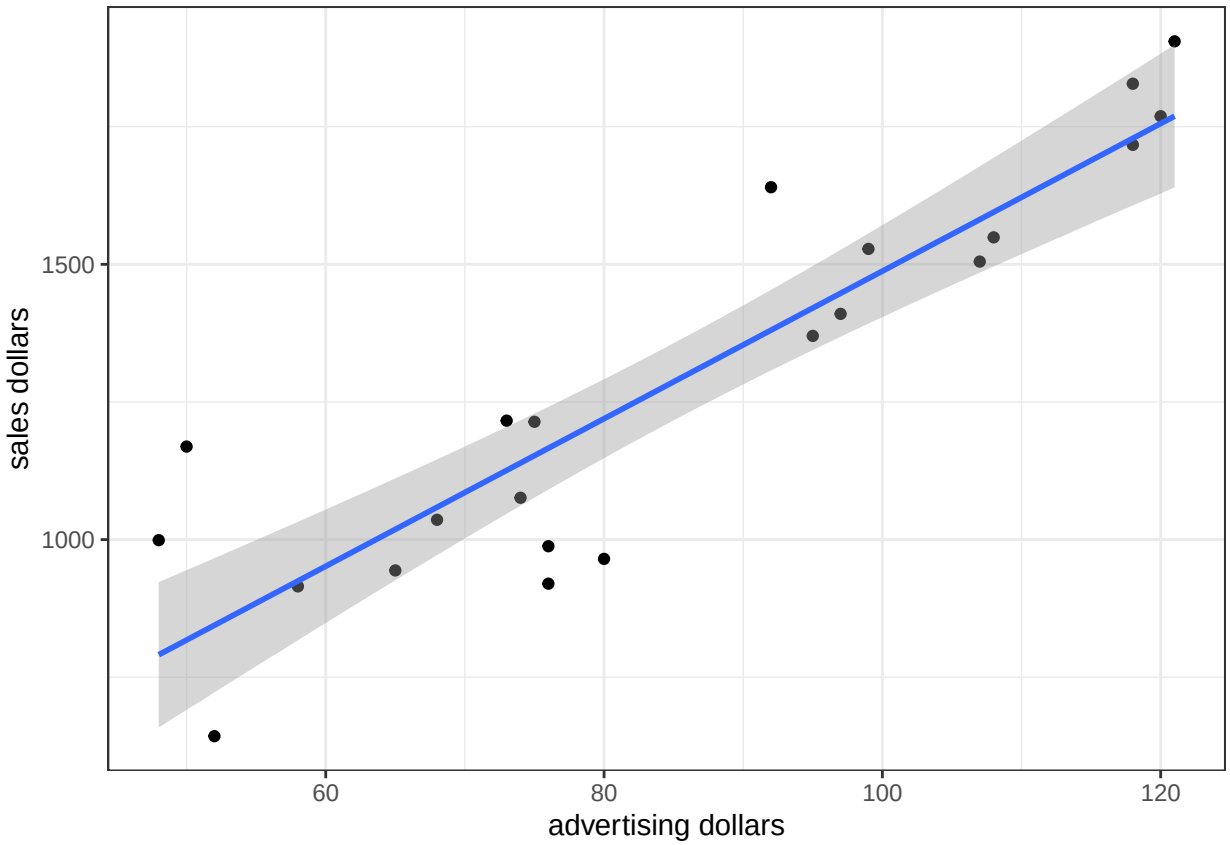
Plot

Plot the relationship between advertising and sales now also plotting the regression line.

```
qplot( data = dt.sales
       , x = advertising
       , y = sales
       , geom = c("point", "smooth")
       , method = lm) +
  theme_bw() +
  labs( x = "advertising dollars", y = "sales dollars")
```

```
## Warning: Ignoring unknown parameters: method
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Predicted values

Obtain the predicted sales for an advertising investment of 100.

```
advertising = 100
sales = coeffs[1] + coeffs[2]*advertising
sales
```

```
## (Intercept)
##    1487.644
```

Alternatively, you can use the "predict" function to do this automatically. We first wrap the parameters inside a new data table variable called newdata.

```
my.budget = data.table(advertising=100)
```

We then apply the predict function and set the predictor variable in the newdata argument. We also set the interval type as "predict", and use the default 0.95 confidence level.

```
predict(lm.sales, my.budget)
```

```
##        1
## 1487.644
```

```
predict(lm.sales, my.budget, interval="predict")
```

```
##        fit      lwr      upr
## 1 1487.644 1148.274 1827.014
```

Acknowledgements and Thanks:

This lab is based on material by M Godinho de Matos, R Belo and F Reis. Gratefully acknowledged!