t-test

Sebastian Heucke

9/27/2019

Contents

Multiple Regression

This tutorial is a markdown version of the video https://www.youtube.com/watch?v=hokALdIst8k&list=PLblh5JKOoLUJJpBNfk8_YadPwDTO2SCbx&index=6&t=0s. It starts with a simple regression in R and then shows how multiple can be used to determine which parameters are the most valuable.

The raw data for mouse **size, weight** and **tail** length.

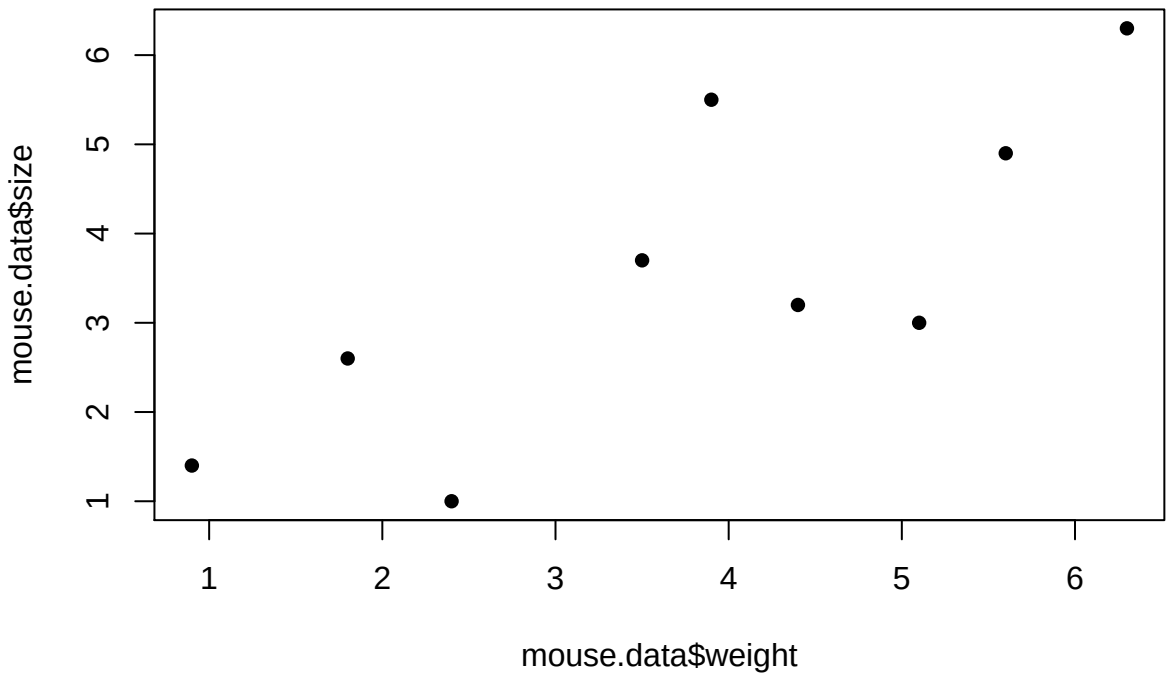
```
mouse.data <- data.frame(
  size = c(1.4, 2.6, 1.0, 3.7, 5.5, 3.2, 3.0, 4.9, 6.3),
  weight = c(0.9, 1.8, 2.4, 3.5, 3.9, 4.4, 5.1, 5.6, 6.3),
  tail = c(0.7, 1.3, 0.7, 2.0, 3.6, 3.0, 2.9, 3.9, 4.0))

mouse.data
```

```
##   size weight tail
## 1  1.4    0.9  0.7
## 2  2.6    1.8  1.3
## 3  1.0    2.4  0.7
## 4  3.7    3.5  2.0
## 5  5.5    3.9  3.6
## 6  3.2    4.4  3.0
## 7  3.0    5.1  2.9
## 8  4.9    5.6  3.9
## 9  6.3    6.3  4.0
```

For simple regression, we will focus on how well **weight** predicts **size**.

Step 1: always plot your data.

```
# STEP 1: Draw a graph of the data to make sure the relationship make sense
plot(mouse.data$weight, mouse.data$size, pch=16)
```



We specified **weight** for the x-axis and **size** for the y-axis.

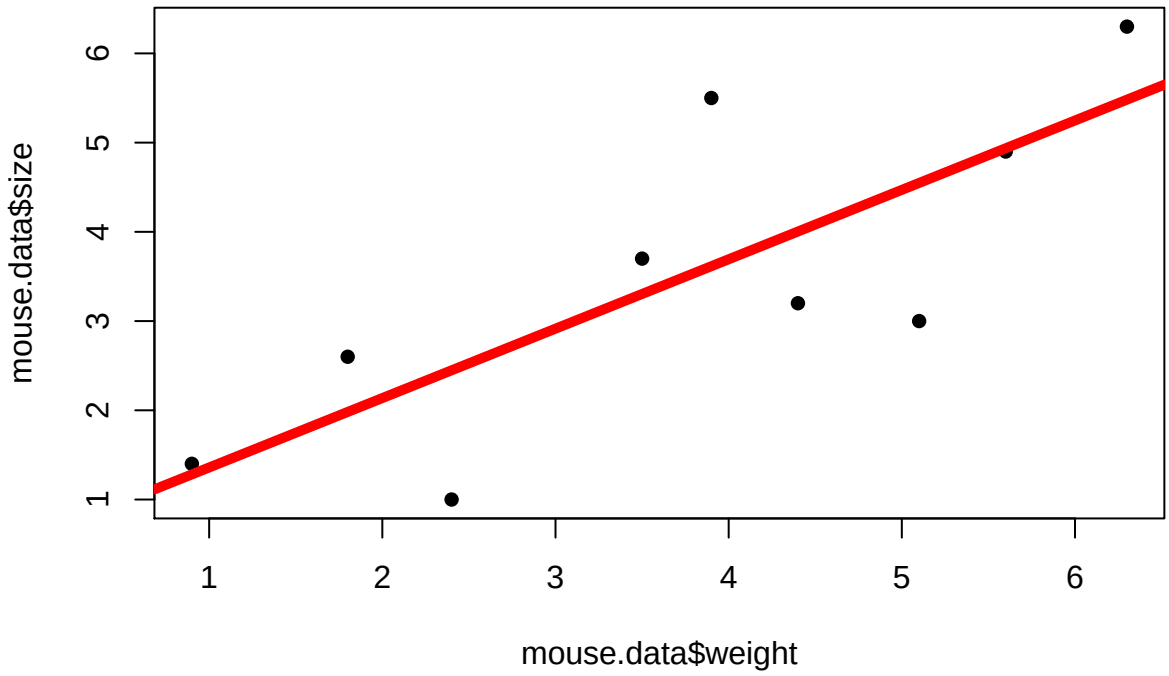
Step 2: use the **lm()** (linear model) function to fit a line to the data.

```
plot(mouse.data$weight, mouse.data$size, pch=16)
# STEP 2: Do the regression
simple.regression <- lm(size ~ weight, data=mouse.data)

# STEP 3: Look at the R^2, F-value and p-value
summary(simple.regression)
```

```
##
## Call:
## lm(formula = size ~ weight, data = mouse.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5482 -0.8037  0.1186  0.6186  1.8852
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5813     0.9647   0.603   0.5658
## weight        0.7778     0.2334   3.332   0.0126 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.19 on 7 degrees of freedom
```

```
## Multiple R-squared:  0.6133, Adjusted R-squared:  0.558
## F-statistic:  11.1 on 1 and 7 DF,  p-value: 0.01256
```

```
abline(simple.regression, lwd=5, col="red")
```



In R, "size ~ weight" is how you specify the following equation:

$size = Y_{intercept} + slope \times weight$

We spefify **size** is predicted by **weight** and by default, R adds the terms for the **Y-intercept** and the **slope**. R then uses least-squares to find the values for **y-intercept** and **slope** that minimize the squared residuals from the line.

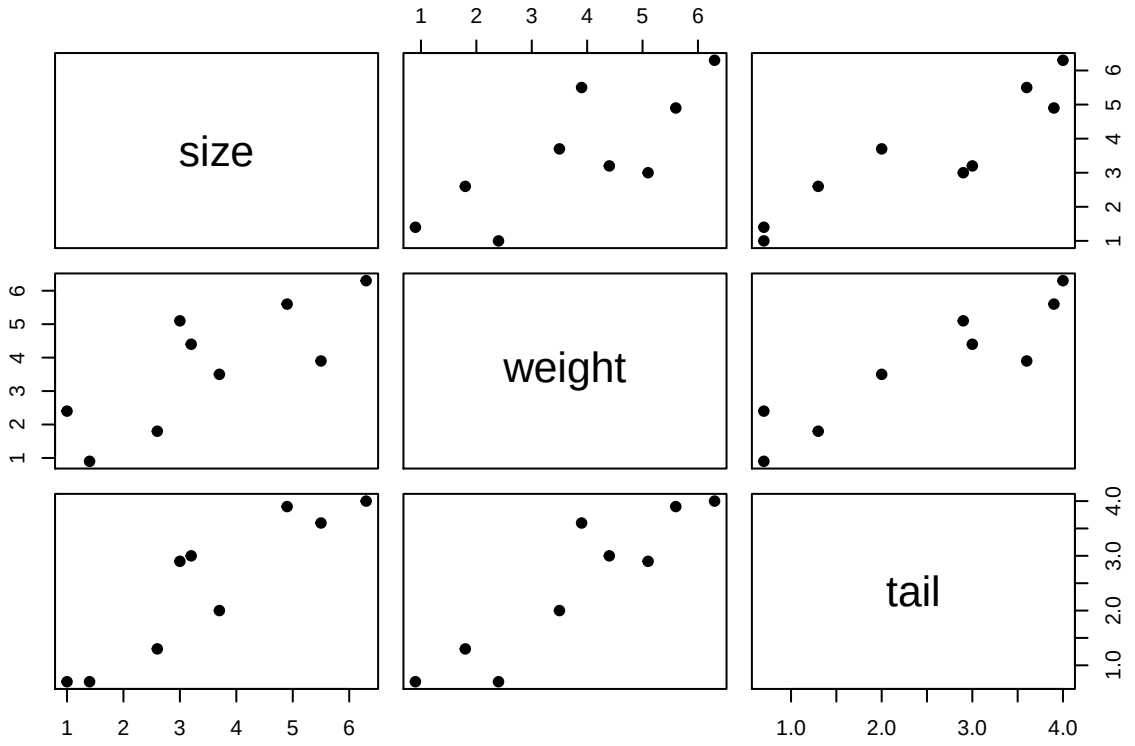
Calling the **summary()** function show you the outcome. Together, the R^2 (0.613) and the p-value (0.012) say that weight does a pretty good job predicting size. With **abline()** you can add the least-squares fit line to the graph.

Multiple regression

For multiple regression, we will use **weight** and **tail** to predict **size**.

Step 1: Always plot the data.

```
plot(mouse.data)
```



Since we didn't specify the x and y-axes, R plots all the data columns (**size, weight** and **tail**) against each other.

Step 2: Use the lm() (linear model) function to fit a plane to the data.

```
multiple.regression <- lm(size ~ weight + tail, data=mouse.data)
summary(multiple.regression)
```

```
##
## Call:
## lm(formula = size ~ weight + tail, data = mouse.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99928 -0.38648 -0.06967  0.34454  1.07932
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.7070     0.6510   1.086   0.3192
## weight       -0.3293     0.3933  -0.837   0.4345
## tail          1.6470     0.5363   3.071   0.0219 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8017 on 6 degrees of freedom
## Multiple R-squared:  0.8496, Adjusted R-squared:  0.7995
## F-statistic: 16.95 on 2 and 6 DF,  p-value: 0.003399
```

In R, "size ~ weight + tail" this is how you specify the following equation:

$size = Y_{intercept} + slope_1 \times weight + slope_2 \times tail$

We specify that **size** is predicted by **weight** and **tail**.

In the summary the R^2 and adjusted R^2 and the p-value look good. Using **weight** and **tail** to predict **size** is good (p-value < 0.05).

Session information

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.23     devtools_2.1.0 usethis_1.5.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1        magrittr_1.5      pkgload_1.0.2
##  [4] R6_2.4.0          rlang_0.4.0       stringr_1.4.0
##  [7] tools_3.6.1       pkgbuild_1.0.5    xfun_0.8
## [10] sessioninfo_1.1.1 cli_1.1.0         withr_2.1.2
## [13] remotes_2.1.0     htmltools_0.3.6   rprojroot_1.3-2
## [16] yaml_2.2.0        digest_0.6.20     assertthat_0.2.1
## [19] crayon_1.3.4      processx_3.4.1    callr_3.3.1
## [22] fs_1.3.1          ps_1.3.0          testthat_2.1.1
## [25] memoise_1.1.0     glue_1.3.1        evaluate_0.14
## [28] rmarkdown_1.14    stringi_1.4.3     compiler_3.6.1
## [31] backports_1.1.4   desc_1.2.0        prettyunits_1.0.2
```

The document was processed on 2019-10-01.