Stats Homework 2

Environment setup

Same initial steps that we ran in `Stats Demo 1`, just consolidated.

```
knitr::opts_chunk$set(echo = TRUE, tidy.opts = list(width.cutoff = 60), tidy = TRUE)

library(Sleuth3) # example datasets from textbook, "The Statistical Sleuth - A Course in
# Methods of Data Analysis (3rd Edition)"
library(MASS) # example dataset representing the sales of different models of cars in 1993
library(reshape2) # for formatting and aggregation of data frames
library(ggplot2) # for creating graphs
library(dplyr) # for data manipulation and clean-up
library(plotly) # for creating interactive web graphics from ggplot2 graphs
library(knitr) # required for generating PDF output
library(modeest) # required for `mfv()` function
library(nortest) # required for running Anderson-Darling test for normality
library(effsize)
```

ANOVA

As an SE researcher you are evaluating different programming languages. For the next set of questions input the R code and interpret your findings.

a) The results of your first study compares Java, Python, and Ruby code based on the size of the programs in source (i.e. non-blank, non-commented) lines of code. Perform an ANOVA to determine whether there is an effect on size due to programming language. Use `lang-size.csv`.

```
# code goes here.
data_a <- read.csv("lang-size.csv")
data_a
```

```
##      lang sloc
## 1    java  207
## 2    java  296
## 3    java  348
## 4    java  309
## 5    java  231
## 6    java  228
## 7    java  318
## 8    java  212
## 9    java  284
## 10   java  267
## 11   java  354
```

```
## 12   java  262
## 13   java  259
## 14   java  342
## 15   java  252
## 16   java  299
## 17   java  312
## 18   java  285
## 19   java  266
## 20   java  333
## 21   java  280
## 22   java  283
## 23   java  368
## 24   java  407
## 25   java  325
## 26   java  339
## 27   java  255
## 28   java  327
## 29   java  268
## 30   java  315
## 31 python  401
## 32 python  391
## 33 python  331
## 34 python  370
## 35 python  415
## 36 python  419
## 37 python  340
## 38 python  382
## 39 python  319
## 40 python  387
## 41 python  455
## 42 python  438
## 43 python  388
## 44 python  449
## 45 python  437
## 46 python  404
## 47 python  352
## 48 python  390
## 49 python  446
## 50 python  424
## 51 python  370
## 52 python  291
## 53 python  366
## 54 python  294
## 55 python  337
## 56 python  381
## 57 python  366
## 58 python  356
## 59 python  395
## 60 python  387
## 61   ruby  188
## 62   ruby  227
## 63   ruby  208
## 64   ruby  267
## 65   ruby  303
```

```
## 66   ruby  311
## 67   ruby  287
## 68   ruby  226
## 69   ruby  278
## 70   ruby  188
## 71   ruby  269
## 72   ruby  178
## 73   ruby  198
## 74   ruby  239
## 75   ruby  176
## 76   ruby  309
## 77   ruby  176
## 78   ruby  228
## 79   ruby  197
## 80   ruby  326
## 81   ruby  280
## 82   ruby  239
## 83   ruby  286
## 84   ruby  286
## 85   ruby  272
## 86   ruby  258
## 87   ruby  283
## 88   ruby  361
## 89   ruby  191
## 90   ruby  246
```

```
summary(aov(sloc ~ lang, data = data_a))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## lang         2 276056  138028    62.6 <2e-16 ***
## Residuals   87 191836    2205
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Report: We ran Anova($F(2, 87) = 62.6$, $p<0.001$)

The p-value is lower than 0.05, and the differences are statistically significant, which is the language has a real impact.

b) In a subsequent study you measured the programming time (in hours) required to solve a program in Java, Python, and Ruby. This was a within subject study design: each participant solved the problem three times, and all participants solved the problem in the same order (Java, then Python, then Ruby). Perform an ANOVA to determine whether there is an effect due to programming language. Use `lang-time.csv`.

```
# code goes here
data_b <- read.csv("lang-time.csv")
data_b
```

```
##      lang participant times
## 1    java          P1  11.3
## 2    java          P2   7.8
## 3    java          P3  12.6
```

```
## 4    java           P4   6.3
## 5    java           P5   8.1
## 6    java           P6  10.1
## 7    java           P7   3.2
## 8    java           P8   7.3
## 9    java           P9   8.1
## 10   java          P10   7.9
## 11   java          P11   9.0
## 12   java          P12   8.7
## 13   java          P13   7.7
## 14   java          P14  11.6
## 15   java          P15   6.0
## 16   java          P16   8.1
## 17   java          P17  10.9
## 18   java          P18   9.1
## 19   java          P19   8.8
## 20   java          P20   9.8
## 21   java          P21   9.5
## 22   java          P22   8.7
## 23   java          P23  10.4
## 24   java          P24   9.7
## 25 python           P1   7.5
## 26 python           P2   5.1
## 27 python           P3   6.8
## 28 python           P4   5.7
## 29 python           P5   3.1
## 30 python           P6   6.8
## 31 python           P7   7.6
## 32 python           P8  11.2
## 33 python           P9  11.1
## 34 python          P10  10.2
## 35 python          P11   5.8
## 36 python          P12   4.3
## 37 python          P13   6.2
## 38 python          P14  10.5
## 39 python          P15   8.2
## 40 python          P16   8.8
## 41 python          P17   6.4
## 42 python          P18   4.9
## 43 python          P19   9.1
## 44 python          P20   7.3
## 45 python          P21   5.9
## 46 python          P22   8.0
## 47 python          P23   9.8
## 48 python          P24   9.8
## 49   ruby           P1   9.1
## 50   ruby           P2   6.2
## 51   ruby           P3   9.2
## 52   ruby           P4   6.9
## 53   ruby           P5   7.4
## 54   ruby           P6   9.7
## 55   ruby           P7   8.5
## 56   ruby           P8   7.1
## 57   ruby           P9   9.7
```

```
## 58   ruby         P10   6.2
## 59   ruby         P11   5.4
## 60   ruby         P12   7.7
## 61   ruby         P13   7.1
## 62   ruby         P14   6.9
## 63   ruby         P15   6.2
## 64   ruby         P16   5.3
## 65   ruby         P17   9.4
## 66   ruby         P18   3.7
## 67   ruby         P19   9.3
## 68   ruby         P20   4.9
## 69   ruby         P21   8.8
## 70   ruby         P22   4.4
## 71   ruby         P23   8.3
## 72   ruby         P24   5.7
```

```
summary(aov(times ~ lang + participant, data = data_b))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## lang         2  33.32  16.661   4.583 0.0153 *
## participant 23 110.62   4.809   1.323 0.2061
## Residuals   46 167.24   3.636
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Report: We ran Anova ($F(2, 46) = 4.583$, $p = 0.0153$) , Anova ($F(23, 46) = 1.323$, $p = 0.2061$)

The p-value of lang is lower than 0.05 showing substantial evidence against the null hypothesis.

c) Your realized you should have counterbalanced, so you replicated the study from (b) which uses a crossover design to control for ordering. Each participant solved the problem in all three languages, but in each participant solved them in a different order. Perform an ANOVA to determine whether there is an effect due to programming language. Use `lang-time-crossover.csv`.

```
# code goes here
data_c <- read.csv("lang-time-crossover.csv")
data_c
```

```
##    participant treatment   lang times
## 1           P1        T1   java   6.4
## 2           P2        T1   java   8.3
## 3           P3        T1 python   7.0
## 4           P4        T1 python  10.5
## 5           P5        T1   ruby  10.6
## 6           P6        T1   ruby   4.0
## 7           P1        T2 python   8.2
## 8           P2        T2   ruby   5.5
## 9           P3        T2   java   7.7
## 10          P4        T2   ruby   7.5
## 11          P5        T2   java   7.0
## 12          P6        T2 python   4.4
## 13          P1        T3   ruby   5.7
## 14          P2        T3 python   7.9
```

```
## 15          P3        T3   ruby   8.8
## 16          P4        T3   java   9.5
## 17          P5        T3 python   8.0
## 18          P6        T3   java   8.0
```

```r
summary(aov(times ~ lang + participant + treatment, data = data_c))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## lang         2  2.170   1.085   0.377  0.698
## participant  5 26.100   5.220   1.812  0.217
## treatment    2  5.623   2.812   0.976  0.418
## Residuals    8 23.047   2.881
```

Report: We ran Anova ($F(2, 8) = 0.377$, $p = 0.698$) , Anova ($F(5, 8) = 1.812$, $p = 0.217$), Anova ($F(2, 8) = 0.976$, $p = 0.418$)

The p-value of lang is higher than 0.05, there's no strong evidence against null hypothesis. We cannot conclude that a significant difference exists.

d) You have some simulated results from an experiment that compared development time for Java, Python and Ruby, for subjects with low experience and high experience. Perform an ANOVA and identify which factors (language, experience) had a statistically significant effect. Also specify whether the interaction between programming language and experience was statistically significant or not. Use `lang-time-exp.csv`. yes

```r
# code goes here
data_d <- read.csv("lang-time-exp.csv")
data_d
```

```
##       lang  exp times
## 1     java  low  11.0
## 2     java  low  10.6
## 3     java  low   8.3
## 4     java  low   9.8
## 5     java  low  11.6
## 6     java  low  11.8
## 7     java  low   8.6
## 8     java  low  10.3
## 9     java  low   7.7
## 10    java  low  10.5
## 11    java  low  13.2
## 12    java  low  12.5
## 13    java  low  10.5
## 14    java  low  13.0
## 15    java  low  12.5
## 16    java  low  11.2
## 17    java  low   9.1
## 18    java  low  10.6
## 19    java  low  12.9
## 20    java  low  12.0
## 21    java high   5.8
## 22    java high   2.6
## 23    java high   5.7
```

```
## 24    java high   2.8
## 25    java high   4.5
## 26    java high   6.3
## 27    java high   5.6
## 28    java high   5.3
## 29    java high   6.8
## 30    java high   6.5
## 31    java high   3.3
## 32    java high   6.8
## 33    java high   8.9
## 34    java high   7.4
## 35    java high   4.2
## 36    java high   4.1
## 37    java high   7.7
## 38    java high   3.5
## 39    java high   6.4
## 40    java high   5.7
## 41  python  low  12.2
## 42  python  low   8.5
## 43  python  low   8.3
## 44  python  low  11.7
## 45  python  low   8.1
## 46  python  low   9.9
## 47  python  low  10.5
## 48  python  low   9.4
## 49  python  low   8.6
## 50  python  low  11.3
## 51  python  low   9.2
## 52  python  low   9.3
## 53  python  low  12.7
## 54  python  low  14.3
## 55  python  low  11.0
## 56  python  low  11.6
## 57  python  low   8.2
## 58  python  low  11.1
## 59  python  low   8.7
## 60  python  low  10.6
## 61  python high   3.5
## 62  python high   5.1
## 63  python high   4.3
## 64  python high   6.7
## 65  python high   8.1
## 66  python high   8.4
## 67  python high   7.5
## 68  python high   5.0
## 69  python high   7.1
## 70  python high   3.5
## 71  python high   6.8
## 72  python high   3.1
## 73  python high   3.9
## 74  python high   5.6
## 75  python high   3.0
## 76  python high   8.3
## 77  python high   3.0
```

```
## 78  python high   5.1
## 79  python high   3.9
## 80  python high   9.0
## 81    ruby  low  10.2
## 82    ruby  low   8.6
## 83    ruby  low  10.4
## 84    ruby  low  10.4
## 85    ruby  low   9.9
## 86    ruby  low   9.3
## 87    ruby  low  10.3
## 88    ruby  low  13.4
## 89    ruby  low   6.6
## 90    ruby  low   8.9
## 91    ruby  low   8.2
## 92    ruby  low   8.6
## 93    ruby  low   9.1
## 94    ruby  low  11.3
## 95    ruby  low  10.2
## 96    ruby  low   6.2
## 97    ruby  low   5.8
## 98    ruby  low  10.8
## 99    ruby  low   9.3
## 100   ruby  low  11.5
## 101   ruby high   3.5
## 102   ruby high   5.8
## 103   ruby high   2.9
## 104   ruby high   6.4
## 105   ruby high   3.7
## 106   ruby high   6.1
## 107   ruby high   6.3
## 108   ruby high   1.7
## 109   ruby high   7.1
## 110   ruby high   7.3
## 111   ruby high   2.4
## 112   ruby high   7.3
## 113   ruby high   4.1
## 114   ruby high   4.4
## 115   ruby high   6.8
## 116   ruby high   5.2
## 117   ruby high   7.1
## 118   ruby high   6.5
## 119   ruby high   7.5
## 120   ruby high   6.9
```

```r
summary(aov(times ~ lang + exp + lang:exp, data = data_d))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## lang          2   11.1     5.6   1.730  0.182
## exp           1  663.2   663.2 206.137 <2e-16 ***
## lang:exp      2    9.7     4.8   1.502  0.227
## Residuals   114  366.8     3.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Report:We ran Anova ($F(2, 114) = 1.730$, $p = 0.182$) , Anova ($F(1, 114) = 206.137$, $p < 0.001$), Anova ($F(2, 114) = 1.502$, $p = 0.227$)

The exp p-value is lower than 0.05, which means the differences are statistically significant.

The p-value between language and experience is 0.227; we cannot conclude that a significant difference exists.

Part 3: Data analysis of an experiment

In this question, you'll analyze the raw data from an experiment and write up the results (similar to a publication).

The data is from a experiment to test whether statically typed languages (e.g. Java) or dynamically typed languages (e.g. Python) require more programming effort. The study evaluates the languages on two problems, a "small" problem and a "large" problem, to see if the results change based on the size of the problem. The study is a factorial design. The raw data from the experiment is available in this file: `lang-time-size.csv`.

Analyze the data and write up a short "results" section (as if it were a part of a paper) with your analysis of the data. This section should contain: * Box plots to show the raw data * Analysis of variance tables to determine if there are any interactions * Results of pairwise t-tests to test if the factors that you may see significance truly have an effect * Interaction plot between the 2 factors * Effect sizes for programming language for the "small" problem and for the "large" problem. * I am not looking for a specific format, use your judgement about the best way to present this data to convey the results to a reader.

NEW

Part 3: Data analysis of an experiment

In this question, you'll analyze the raw data from an experiment and write up the results (similar to a publication).

The data is from a experiment to test whether statically typed languages (e.g. Java) or dynamically typed languages (e.g. Python) require more programming effort. The study evaluates the languages on two problems, a "small" problem and a "large" problem, to see if the results change based on the size of the problem. The study is a factorial design. The raw data from the experiment is available in this file: `lang-time-size.csv`.

Analyze the data and write up a short "results" section (as if it were a part of a paper) with your analysis of the data. This section should contain: * Analysis of variance tables to determine if there are any interactions * Interaction plot between the 2 factors * Effect sizes for programming language for the "small" problem and for the "large" problem. * I am not looking for a specific format, use your judgement about the best way to present this data to convey the results to a reader.

```
# Code for analysis goes here.

data_part3 <- read.csv("lang-time-size.csv")
data_part3
```

```
##     times   lang  size
## 1    14.0   java small
## 2    20.3   java small
## 3    11.6   java small
## 4    16.6   java small
## 5    15.0   java small
```

```
## 6     8.0   java small
## 7    13.1   java small
## 8    17.4   java small
## 9    12.5   java small
## 10    8.7   java small
## 11   16.4   java small
## 12   11.5   java small
## 13   13.7   java small
## 14    8.0   java small
## 15   18.8   java small
## 16   13.0   java small
## 17   12.9   java small
## 18   13.1   java small
## 19    8.3   java small
## 20   10.9   java small
## 21   18.5   java small
## 22   18.6   java small
## 23   11.4   java small
## 24   11.2   java small
## 25   17.0   java small
## 26   14.7   java small
## 27   15.9   java small
## 28   10.3   java small
## 29   14.3   java small
## 30   16.6   java small
## 31   15.3   java large
## 32   19.0   java large
## 33   30.6   java large
## 34   25.0   java large
## 35   26.7   java large
## 36   22.7   java large
## 37   17.1   java large
## 38   27.3   java large
## 39   13.9   java large
## 40    8.5   java large
## 41   21.9   java large
## 42   24.8   java large
## 43   38.9   java large
## 44   11.1   java large
## 45   16.3   java large
## 46   29.0   java large
## 47   10.0   java large
## 48   31.5   java large
## 49   24.7   java large
## 50   26.5   java large
## 51   24.5   java large
## 52   27.2   java large
## 53   17.1   java large
## 54   32.4   java large
## 55   22.3   java large
## 56   11.9   java large
## 57   13.4   java large
## 58   23.2   java large
## 59   28.3   java large
```

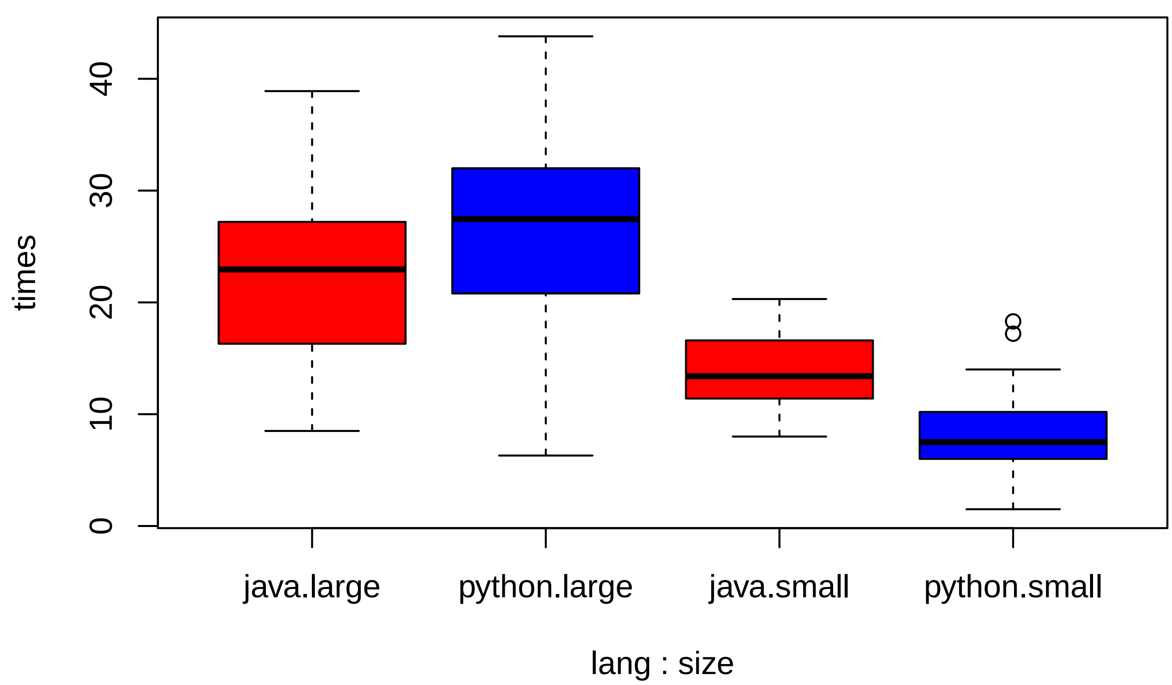
```
## 60   18.1   java large
## 61   10.2 python small
## 62    7.3 python small
## 63    6.6 python small
## 64    3.8 python small
## 65    3.0 python small
## 66    4.4 python small
## 67    7.7 python small
## 68   11.2 python small
## 69    6.0 python small
## 70   14.0 python small
## 71    2.9 python small
## 72    1.5 python small
## 73    3.7 python small
## 74    7.0 python small
## 75    9.2 python small
## 76    6.8 python small
## 77   10.4 python small
## 78   13.0 python small
## 79    5.1 python small
## 80   17.2 python small
## 81    9.0 python small
## 82   18.3 python small
## 83    9.6 python small
## 84    7.1 python small
## 85    7.1 python small
## 86    7.0 python small
## 87    7.7 python small
## 88   11.5 python small
## 89    9.5 python small
## 90    8.4 python small
## 91   20.8 python large
## 92   16.5 python large
## 93   43.8 python large
## 94   30.0 python large
## 95   25.2 python large
## 96   39.4 python large
## 97   25.6 python large
## 98   37.3 python large
## 99   19.0 python large
## 100   7.5 python large
## 101  26.7 python large
## 102  26.8 python large
## 103  17.3 python large
## 104  38.7 python large
## 105  35.6 python large
## 106  24.3 python large
## 107  28.6 python large
## 108  34.4 python large
## 109   6.3 python large
## 110  30.3 python large
## 111  20.0 python large
## 112  32.0 python large
## 113  28.1 python large
```

```
## 114  30.2 python large
## 115  29.7 python large
## 116  31.9 python large
## 117  14.9 python large
## 118  23.3 python large
## 119  35.4 python large
## 120  25.4 python large
```

```
summary(aov(times ~ lang + size + lang:size, data = data_part3))
```

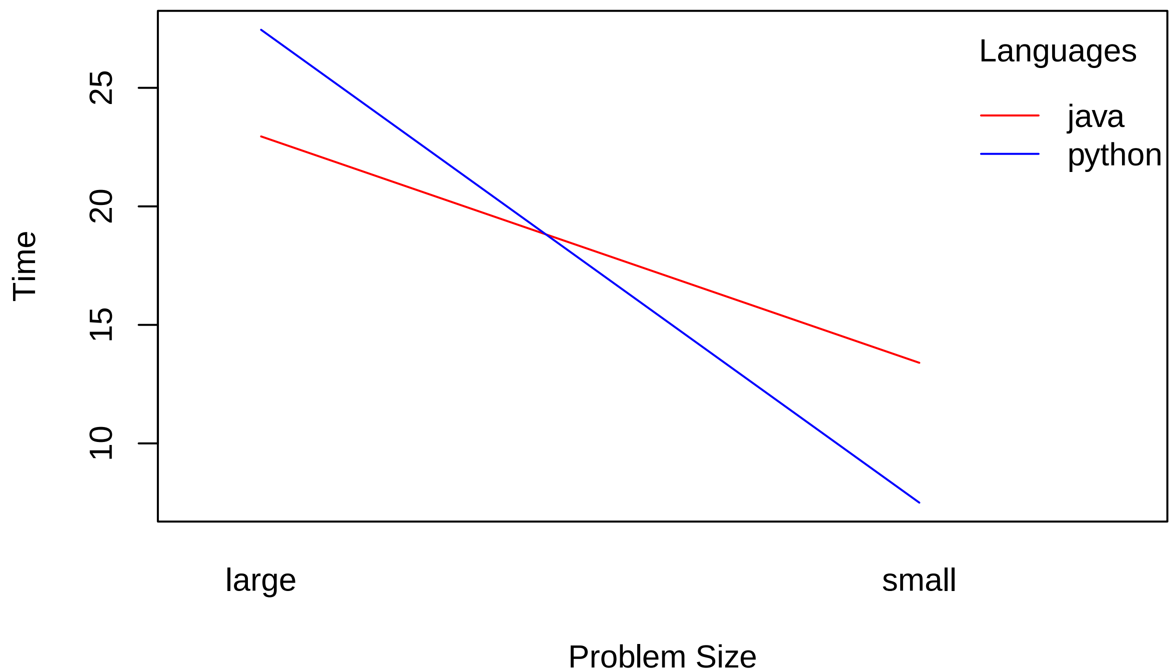
```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## lang          1      3       3   0.085    0.772
## size          1   5410    5410 133.246  < 2e-16 ***
## lang:size     1    811     811  19.968 1.84e-05 ***
## Residuals   116   4709      41
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
boxplot(times ~ lang + size,data = data_part3, col=(c("red","blue")))
```




```
interaction.plot(x.factor = data_part3$size, #x-axis variable
                 trace.factor = data_part3$lang, #variable for lines
                 response = data_part3$times, #y-axis variable
                 fun = median, #metric to plot
```

```
                 ylab = "Time",
                 xlab = "Problem Size",
                 col = c("red", "blue"),
                 lty = 1, #line type
                 lwd = 1, #line width
                 trace.label = "Languages")
```




```
# Effect sizes

small_java <- data_part3[which(data_part3$size == "small" &data_part3$lang == "java" ),]
small_python <- data_part3[which(data_part3$size == "small" &data_part3$lang == "python" ),]

large_java <- data_part3[which(data_part3$size == "large" &data_part3$lang == "java" ),]
large_python <- data_part3[which(data_part3$size == "large" &data_part3$lang == "python" ),]

cat("small problem effect sizes:\n")
```

```
## small problem effect sizes:
```

```
cohen.d(small_java$times, small_python$times)
```

```
##
## Cohen's d
##
```

```
## d estimate: 1.500909 (large)
## 95 percent confidence interval:
##     lower     upper
## 0.9158068 2.0860117
```

```
cat("larg problem effect sizes:\n")
```

```
## larg problem effect sizes:
```

```
cohen.d(large_java$times, large_python$times)
```

```
##
## Cohen's d
##
## d estimate: -0.5911475 (medium)
## 95 percent confidence interval:
##       lower       upper
## -1.11915639 -0.06313861
```

```
#cohen.d(size$java, size$python)
```

part results: Analysis from the part3

Report: We ran Anova ($F(1, 116) = 0.085$, $p = 0.772$) , Anova ($F(1, 114) = 133.246$, $p < 0.001$), Anova ($F(1, 116) = 19.968$, $p < 0.001$)

The p-value of size and the interactions between language and size are lower than 0.001. Therefore the differences are statistically significant. The size and the interaction between language and size have impacts on time.

By using the Cohen's function to evaluate effect size. The Cohen's d in small problem shows it has a large effect size, and the large problem doesn't.