Stat 502 Project Report

Arman Bilge, Cheng Wang, and Zexuan Zhou

December 2, 2016

1. Introduction

2. Experimental Design

2.1 Motivation

2.2 Experimental Settings

2.2.1 Experimental Units and Randomization

2.2.2 Measurement

3. Data and Analysis

3.1 Data Quality

After invesgating an addtive model and finding that we have an unequal variance for the residual, we used the Box-Cox procedure to perform a transformation. The result was that we did a log transformation for our running time and the residual tends to stablize but still there are some diverge in the lower range. These data points mainly come from the running result from the Windows OS and they have very small value because the Windows computer we used has the best hardware setting and thus for some of the programs they run faster. The number of these data points is relatively small compared to the total number of all data points so we consider that we have achieved the goal to stablize the variance among residuals.

3.1.1 Summary Statistics

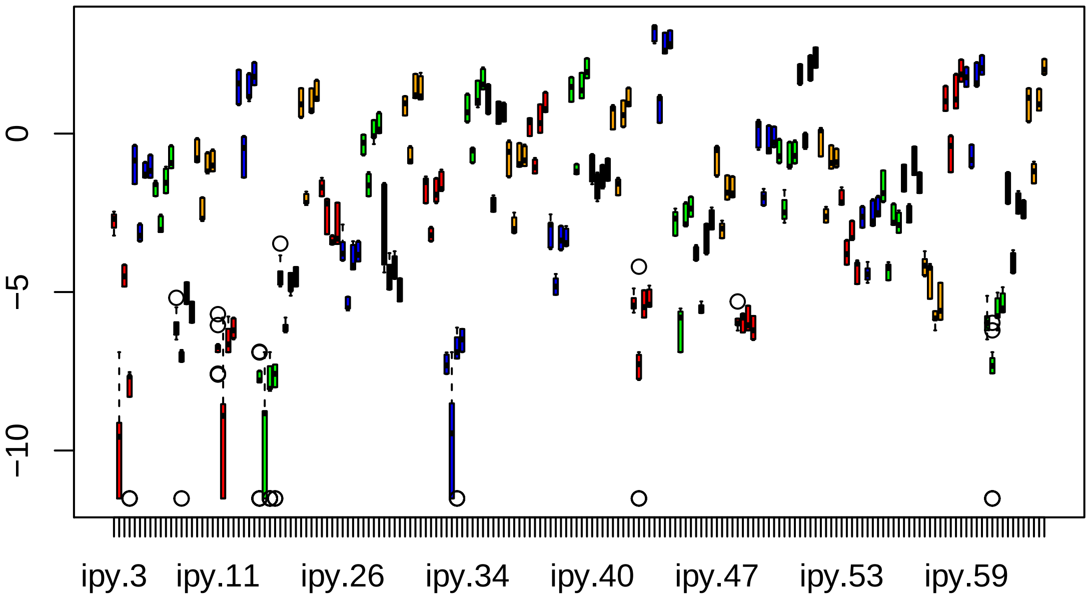
Interpreter	Median	Mean	SD
iPy	-1.41	-1.73	2.77
PyPy	-3	-3.69	2.82
Python	-1.55	-1.92	2.81
Python3	-1.39	-1.83	3.03

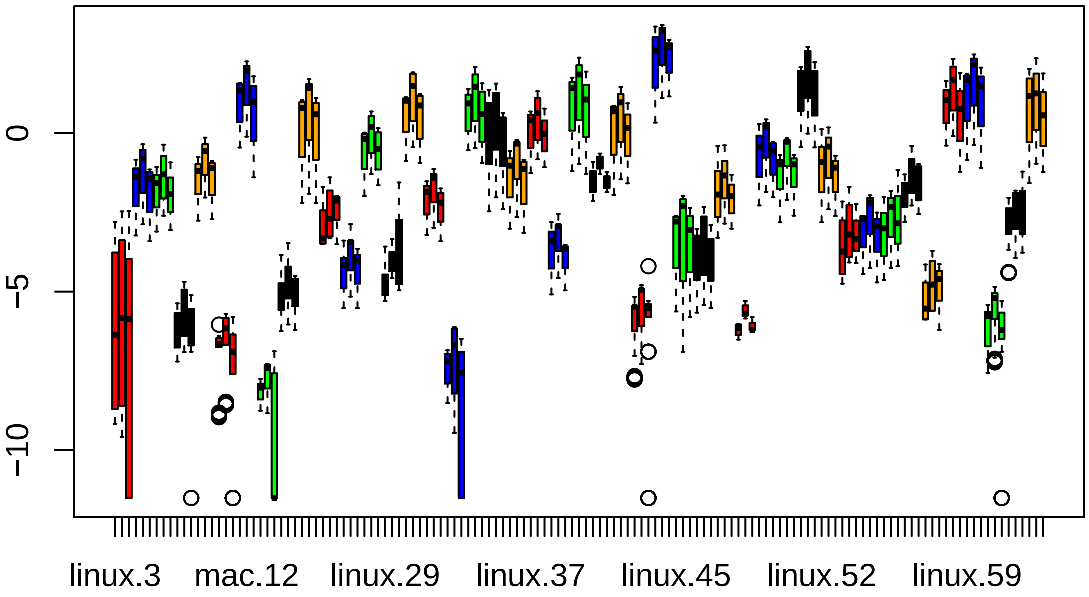
Comparing different interpreters, we found that on average the interpreter PyPy is faster than the rest and interpreters with JIT (iPy and Pypy) are faster than the interpreters without JIT (Python and Python3).

Interpreter	Median	Mean	SD
Linux	-2.16	-2.41	2.87
Mac	-1.78	-2.94	2.93
Windows	-2.02	-2.52	3.09

Comparing different OS we found that on average Linux is faster, however the means of all three interpreters are close to each other so we need to conduct further inspection.

3.1.2 Overall Observation





3.2 Model Proposal

Recall that we are interested in the following questions:
1. Which interpreters perform best on average?
2. Does OS have influence on the running time?
3. Is there any interaction effect on running time between the OS and the interpreters?
Based on these two questions we proposed two models. One addtive model and one interaction model. We use the 45 programs as our blocking factor.
### 3.2.1 Addtive Model

$$y = \mu_i + \alpha_{ij} + \beta_{ijk} + \epsilon_{ijkl},\ \epsilon_{ijkl} \sim N(0, \sigma)$$

$$\mu - blocks/programs,\ i = 1, ..., 45$$

$$\alpha - OS,\ j = 1, 2, 3$$

$$\beta - interpreter,\ k = 1, 2, 3, 4$$

$$\epsilon - error, l = 1, ..., 5400$$

3.2.2 Interaction Model

$$y = \mu_i + \alpha_{ij} + \beta_{ijk} + \alpha\beta_{ijk} + \epsilon_{ijkl},\ \epsilon_{ijkl} \sim N(0, \sigma)$$

$$\mu - blocks/programs,\ i = 1, ..., 45$$

$$\alpha - OS,\ j = 1, 2, 3$$

$$\beta - interpreter,\ k = 1, 2, 3, 4$$

$$\alpha\beta - interaction\ effect\ of\ OS\ and\ interpreter$$

$$\epsilon - error,\ l = 1, ..., 5400$$

3.3 ANOVA Summaries

3.3.1 Summary for the additive model

```
## Analysis of Variance Table
##
## Response: time
##                   Df Sum Sq Mean Sq F value Pr(>F)
## factor(program)   44  41024     932    1796 <2e-16 ***
## os                 2    336     168     323 <2e-16 ***
## interpreter        3   3573    1191    2295 <2e-16 ***
## Residuals       5350   2777       1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table for the additive model suggests that the OS does have an influence on the running time of programs. However, the portion of variance that the OS factor accounts for is pretty small because we observe a relatively small value for MST of the OS factor.

3.3.2 Summary for the interaction model

```
## Analysis of Variance Table
##
## Response: time
##                   Df Sum Sq Mean Sq F value  Pr(>F)
## factor(program)   44  41024     932  1820.2 < 2e-16 ***
## os                 2    336     168   327.8 < 2e-16 ***
## interpreter        3   3573    1191  2325.1 < 2e-16 ***
## os:interpreter     6     39       7    12.8 2.1e-14 ***
## Residuals       5344   2737       1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table for the interaction model suggests that there exists an interaction effect of OS and interpreter. However, the portion of variance that the interaction term accounts for is pretty small because we observe a relatively small value for MST of the OS factor. As for which combination of OS and interpreter perform better we will do a contrast test latter.

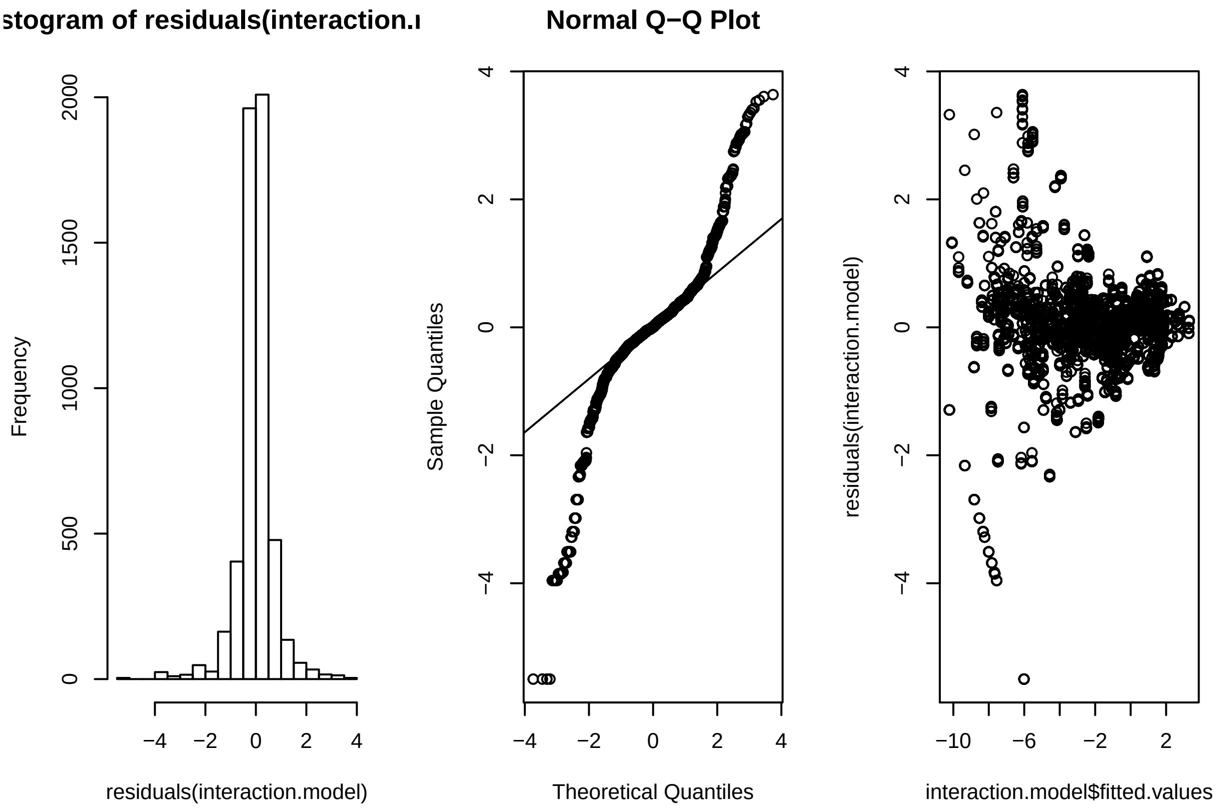
3.3.3 Full Model vs. Reduced Model

We conducted a model selection test, to decide whether we should use the full model (interaction model) or the reduced model (additive model)

```
## Analysis of Variance Table
##
## Model 1: time ~ factor(program) + os + interpreter
## Model 2: time ~ factor(program) + os * interpreter
##   Res.Df  RSS Df Sum of Sq    F  Pr(>F)
## 1   5350 2777
## 2   5344 2737  6      39.4 12.8 2.1e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA table tells us that there is a difference between two models. Based on our quesions of interest, we decide to choose the interaction model, which also will help us conduct contrast test to determine what conbination of OS and interpreter performs best.

3.4 Assumption Check



The normality assumption is violated. And the equal variance assumption seems also violated. However these do not affect the robustness of ANOVA. There are some skewed points in two sides. We think that these violations might be because we run 100 times for warmup but only run each program 10 times for measurements.

3.5 Contrast Tests

3.5.1 Contrasts

We proposed following contrasts:
$C_1 : \beta_1 + \beta_2 - \beta_3 - \beta_4 = 0$ (JIT vs Non-JIT)
$C_2 : \beta_1 - \beta_2 = 0$ (which one is the best within JIT, iPy or PyPy)
$C_3 : \alpha_1 - \alpha_2 = 0$ (Mac vs Win)
$C_4 : \alpha_2 - \alpha_3 = 0$ (Mac vs Linx)
$C_5 : (\alpha\beta_{23} - \alpha\beta_{33}) - (\alpha\beta_{24} - \alpha\beta_{34})$ (OS*Py vs OS*Py3)

The reason we only do one contrast for the interaction effect is because that we found that the other interaction terms are not significatn in our interaction model, which means that we can ignore those terms.

3.5.2 CIs for contrasts

Contrast	Confidence	Interval
C1	-1.37	-1.26
C2	-2.11	-2.03
C3	0.534	0.614
C4	0.22	0.299
C5	0.256	0.369

We see that all of the 95% confidence intervals for our contrasts do not contain zero so we condclude the following:
C_1: JIT is faster than Non-JIT.
C_2: Within JIT, PyPy is faster. This also implies that iPy performs best among all four interpreters.
C_3: Windows OS is faster than Mac OS.
C_4: Windows OS is faster than Linx OS. Combine this with C_3 we conclude that Windows OS performs best.
C_5: Python3 has a better perfromance than Python when the OS changes from Mac to Windows.