Collaborative 3DUI Data Analysis

Jerônimo G. Grandi

August 22, 2016

Data analysis

This is a on going analysis of the data collected in the user experiment performed....

Comparisons between groups

Hypotesis

- Groups with more than one member perform the tasks with less errors.
- Groups with two and three members perform the tasks with less errors than with one and four.
- Groups with more than one member perform the tasks faster.
- Groups with two and three members perform the tasks faster than with one and four.
- Task one and two are easier than three and four.

Data Summary

```
sourceDataGroups <- read.csv("errorTimeAndVarPerTeam.csv",header = TRUE,  sep=";", dec = ",")
sourceDataGroups
```

```
##     Time.T1   Time.T2   Time.T3   Time.T4  Error.T1  Error.T2 Error.T3
## 1   44.6010  19.92100  58.66863  47.19800  4.280380  1.923633 0.867342
## 2   27.7300  23.75830  16.39880  21.40800 16.953010 11.753480 3.243309
## 3   44.5060  86.01300 127.23090 162.96200  9.070060  3.251326 2.972291
## 4   56.6990  90.59442  82.56380  87.06737  8.529819  3.595758 3.701374
## 5   86.2698  13.64980  29.00930  22.00295  3.869670  3.004579 2.359320
## 6   26.6007  15.95330  49.16030  16.25100  6.642847  4.069859 4.060901
## 7  129.5357  87.26190 120.67140  74.45340  7.381541  4.849439 3.978512
## 8   88.6565 113.11968  91.11170  71.80547  5.705525  4.512303 3.053629
## 9   37.2235  16.45030  36.21300  21.09751  1.317529  2.192855 2.259253
## 10  32.5000  14.49990  22.41350  16.95763  9.220284  5.547572 3.317118
## 11 130.7602  69.36490  76.82150  56.40480  6.207668  6.037892 4.150805
## 12  63.4430  58.41970  55.29020  63.90760  5.088869  4.339551 4.628316
## 13  71.4288  17.20950  21.44110  54.20073  4.024921  2.576868 1.300738
## 14  55.8885  29.62760  17.18680  37.12230  8.576299  8.645294 3.357939
## 15 281.5956 139.37750  86.81320 127.17520  3.341505  4.726371 3.179116
## 16 225.2061 111.04040  61.24430  60.76750  3.299659  3.707839 4.247869
## 17  32.6565  59.33450  20.20286  19.35610  4.456124  4.601643 2.329338
## 18  41.4964  16.48410  25.70100  30.75650  9.002918  8.891509 4.309965
## 19 105.6555 109.53340 283.41380 212.14000  6.817599  4.380987 2.652203
## 20 139.6341  83.16500 192.72400 113.40100  8.455383  5.540681 2.953497
## 21       NA  35.49839  24.40880        NA        NA  1.390942 2.187007
## 22       NA  34.55538  42.26010        NA        NA  4.142436 8.903512
## 23       NA  99.48090  41.29660        NA        NA  4.699787 4.778647
```

```
## 24       NA  93.11320  38.11630        NA        NA  4.160006 4.583976
##    Error.T4 Var.T1      Var.T2     Var.T3     Var.T4
## 1  1.554914      0 0.263696129 0.28683555 0.00659468
## 2  3.538198      0 0.022688678 0.23868391 0.06372513
## 3  2.389965      0 0.199389588 0.13345997 0.07470829
## 4  2.110792      0 0.080684134 0.24071698 0.16046311
## 5  1.402203      0 0.225296778 0.09436949 0.11641209
## 6  4.501410      0 0.038367520 0.06848351 0.05595397
## 7  3.337539      0 0.001521254 0.14643913 0.18305597
## 8  2.288988      0 0.041505246 0.09415421 0.11706688
## 9  3.145458      0 0.068950900 0.23868853 0.12520884
## 10 6.065219      0 0.003346766 0.15437672 0.09580485
## 11 3.023767      0 0.182106965 0.04889033 0.05751343
## 12 3.257560      0 0.185827978 0.01119598 0.17610524
## 13 1.623129      0 0.334446219 0.15409347 0.14721198
## 14 2.768127      0 0.044372642 0.11124479 0.23253372
## 15 1.776181      0 0.426703954 0.03485980 0.18255851
## 16 1.903549      0 0.295053083 0.11615541 0.14353863
## 17 2.046559      0          NA 0.41520724 0.50393597
## 18 2.979571      0          NA 0.23777665 0.23708575
## 19 2.376898      0          NA 0.01503553 0.12033979
## 20 1.918039      0          NA 0.01889699 0.06050588
## 21       NA     NA          NA 0.29975586         NA
## 22       NA     NA          NA 0.17296443         NA
## 23       NA     NA          NA 0.11817581         NA
## 24       NA     NA          NA 0.21563057         NA
```

```
sourceDataTasks <- read.csv("errorAndTimePerTask.csv",header = TRUE,  sep=";")
sourceDataTasks
```

```
##    Members.Task Time.task.1 Time.task.2 Time.task.3 Time.task.4
## 1             1    44.60100    27.73000     44.5060    56.69900
## 2             1    86.26980    26.60070    129.5357    88.65650
## 3             1    37.22350    32.50000    130.7602    63.44300
## 4             1    71.42880    55.88850    281.5956   225.20610
## 5             1    32.65650    41.49640    105.6555   139.63410
## 6             2    19.92100    23.75830     86.0130    90.59442
## 7             2    13.64980    15.95330     87.2619   113.11968
## 8             2    16.45030    14.49990     69.3649    58.41970
## 9             2    17.20950    29.62760    139.3775   111.04040
## 10            2    59.33450    16.48410    109.5334    83.16500
## 11            2    35.49839    34.55538     99.4809    93.11320
## 12            3    58.66863    16.39880    127.2309    82.56380
## 13            3    29.00930    49.16030    120.6714    91.11170
## 14            3    36.21300    22.41350     76.8215    55.29020
## 15            3    21.44110    17.18680     86.8132    61.24430
## 16            3    20.20286    25.70100    283.4138   192.72400
## 17            3    24.40880    42.26010     41.2966    38.11630
## 18            4    47.19800    21.40800    162.9620    87.06737
## 19            4    22.00295    16.25100     74.4534    71.80547
## 20            4    21.09751    16.95763     56.4048    63.90760
## 21            4    54.20073    37.12230    127.1752    60.76750
## 22            4    19.35610    30.75650    212.1400   113.40100
##    Error.task.1 Error.task.2 Error.task.3 Error.task.4
## 1      4.280380    16.953010     9.070060     8.529819
```

```
## 2      3.869670     6.642847     7.381541     5.705525
## 3      1.317529     9.220284     6.207668     5.088869
## 4      4.024921     8.576299     3.341505     3.299659
## 5      4.456124     9.002918     6.817599     8.455383
## 6      1.923633    11.753480     3.251326     3.595758
## 7      3.004579     4.069859     4.849439     4.512303
## 8      2.192855     5.547572     6.037892     4.339551
## 9      2.576868     8.645294     4.726371     3.707839
## 10     4.601643     8.891509     4.380987     5.540681
## 11     1.390942     4.142436     4.699787     4.160006
## 12     0.867342     3.243309     2.972291     3.701374
## 13     2.359320     4.060901     3.978512     3.053629
## 14     2.259253     3.317118     4.150805     4.628316
## 15     1.300738     3.357939     3.179116     4.247869
## 16     2.329338     4.309965     2.652203     2.953497
## 17     2.187007     8.903512     4.778647     4.583976
## 18     1.554914     3.538198     2.389965     2.110792
## 19     1.402203     4.501410     3.337539     2.288988
## 20     3.145458     6.065219     3.023767     3.257560
## 21     1.623129     2.768127     1.776181     1.903549
## 22     2.046559     2.979571     2.376898     1.918039
```

```
#errorXGroups <- read.csv("ErrorXGroups.csv",header = FALSE)
#errorXGroups
#timeXGroups <- read.csv("TimeXGroups.csv", header = FALSE)
#timeXGroups
```

Below the data is summarized:

```
#describe(errorXGroups)
#describe(timeXGroups)
describe(sourceDataGroups)
```

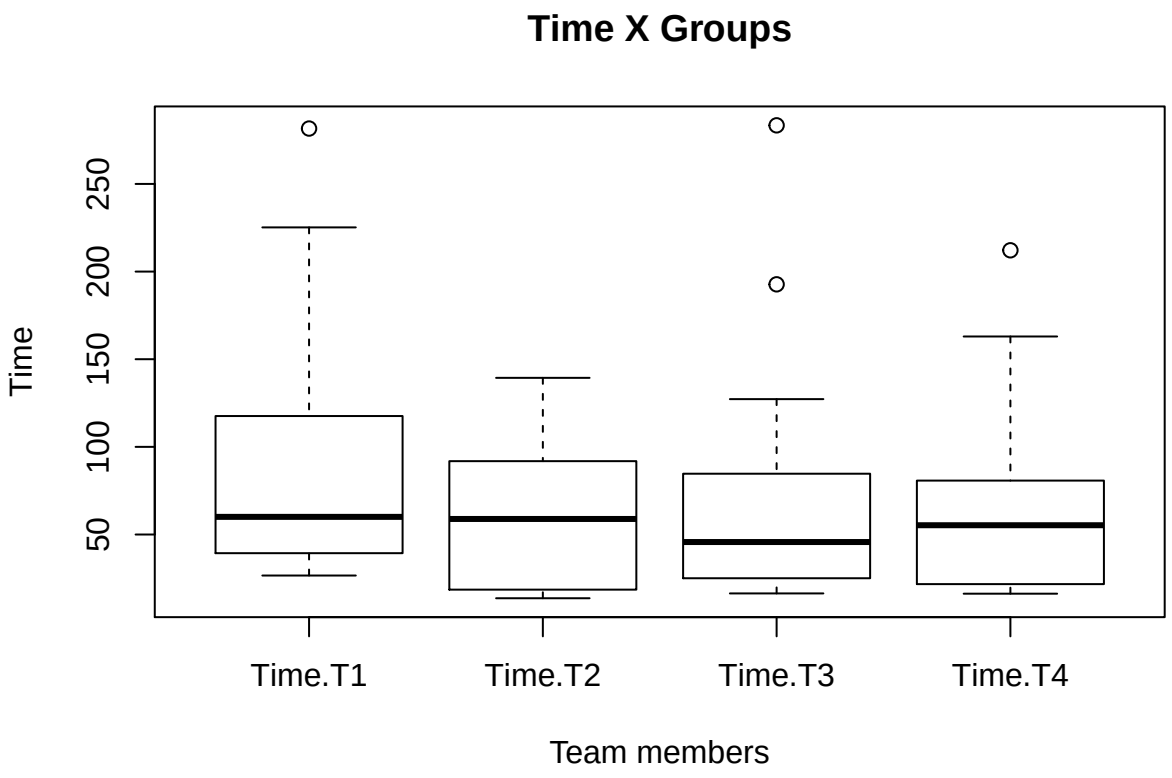
```
## sourceDataGroups
##
##  12  Variables      24  Observations
## ---------------------------------------------------------------------------
## Time.T1
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      20       4      20       1    86.1   27.67   32.02   40.43   60.07
##     .75     .90     .95
##  111.63  148.19  228.03
##
## lowest :  26.60  27.73  32.50  32.66  37.22
## highest: 129.54 130.76 139.63 225.21 281.60
## ---------------------------------------------------------------------------
## Time.T2
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      24       0      24       1   59.89   14.72   16.10   19.24   58.88
##     .75     .90     .95
##   91.22  110.59  112.81
##
## lowest :  13.65  14.50  15.95  16.45  16.48
## highest:  99.48 109.53 111.04 113.12 139.38
## ---------------------------------------------------------------------------
```

```
## Time.T3
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      24       0      24       1   67.52   17.64   20.57   25.38   45.71
##     .75     .90     .95
##   83.63  125.26  182.90
##
## lowest :  16.40  17.19  20.20  21.44  22.41
## highest:  91.11 120.67 127.23 192.72 283.41
## ------------------------------------------------------------------------
## Time.T4
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      20       4      20       1   65.82   16.92   19.12   21.85   55.30
##     .75     .90     .95
##   77.61  130.75  165.42
##
## lowest :  16.25  16.96  19.36  21.10  21.41
## highest:  87.07 113.40 127.18 162.96 212.14
## ------------------------------------------------------------------------
## Error.T1
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      20       4      20       1   6.612   3.201   3.337   4.217   6.425
##     .75     .90     .95
##   8.541   9.085   9.607
##
## lowest :  1.318  3.300  3.342  3.870  4.025
## highest:  8.576  9.003  9.070  9.220 16.953
## ------------------------------------------------------------------------
## Error.T2
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      24       0      24       1   4.689   1.964   2.308   3.510   4.360
##     .75     .90     .95
##   5.022   7.863   8.855
##
## lowest :  1.391  1.924  2.193  2.577  3.005
## highest:  5.548  6.038  8.645  8.892 11.753
## ------------------------------------------------------------------------
## Error.T3
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      24       0      24       1   3.474   1.434   2.209   2.579   3.280
##     .75     .90     .95
##   4.175   4.615   4.756
##
## lowest : 0.8673 1.3007 2.1870 2.2593 2.3293
## highest: 4.3100 4.5840 4.6283 4.7786 8.9035
## ------------------------------------------------------------------------
## Error.T4
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      20       4      20       1     2.7   1.547   1.616   1.914   2.383
##     .75     .90     .95
##   3.173   3.635   4.580
##
## lowest : 1.402 1.555 1.623 1.776 1.904
## highest: 3.258 3.338 3.538 4.501 6.065
## ------------------------------------------------------------------------
```

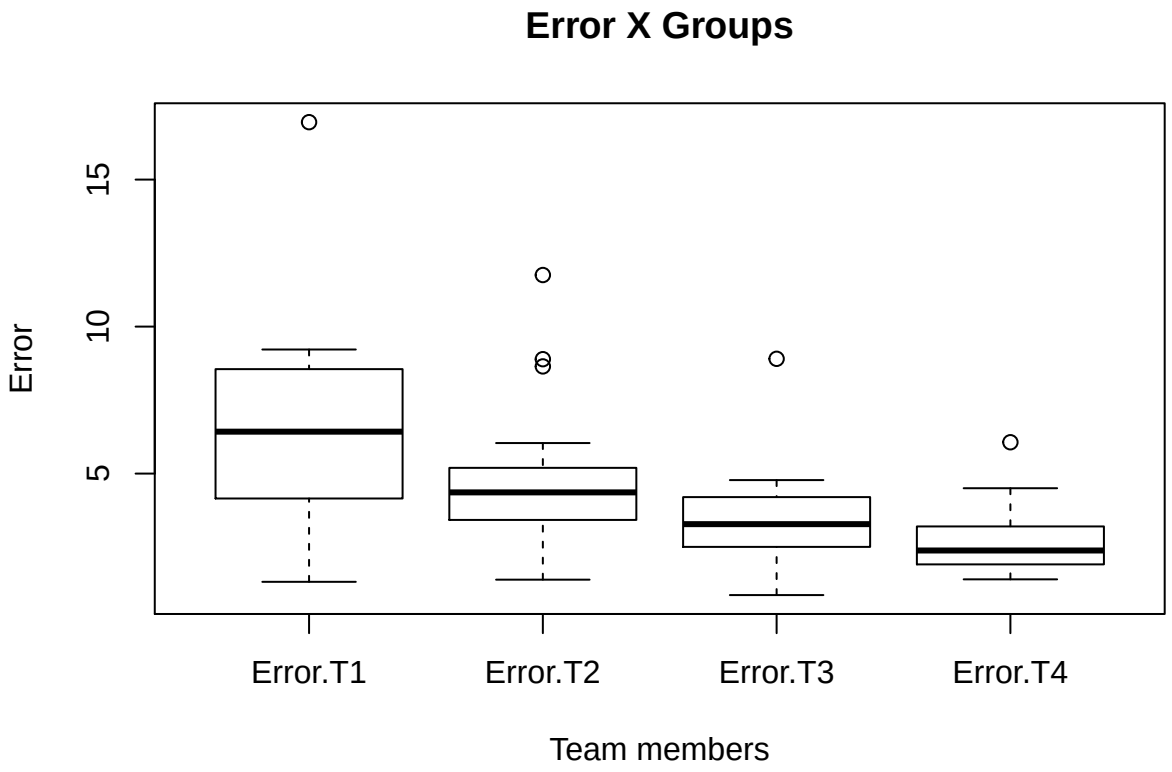
```
## Var.T1
##       n missing  unique    Info    Mean
##      20       4       1       0       0
## ------------------------------------------------------------------------------
## Var.T2
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      16       8      16       1  0.1509 0.00289 0.01302 0.04072 0.13140
##     .75     .90     .95
## 0.23490 0.31475 0.35751
##
## 0.001521254 (1, 6%), 0.003346766 (1, 6%)
## 0.022688678 (1, 6%), 0.03836752 (1, 6%)
## 0.041505246 (1, 6%), 0.044372642 (1, 6%)
## 0.0689509 (1, 6%), 0.080684134 (1, 6%)
## 0.182106965 (1, 6%), 0.185827978 (1, 6%)
## 0.199389588 (1, 6%), 0.225296778 (1, 6%)
## 0.263696129 (1, 6%), 0.295053083 (1, 6%)
## 0.334446219 (1, 6%), 0.426703954 (1, 6%)
## ------------------------------------------------------------------------------
## Var.T3
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      24       0      24       1  0.1528 0.01561 0.02369 0.08774 0.13995
##     .75     .90     .95
## 0.23800 0.27300 0.29782
##
## lowest : 0.01120 0.01504 0.01890 0.03486 0.04889
## highest: 0.23869 0.24072 0.28684 0.29976 0.41521
## ------------------------------------------------------------------------------
## Var.T4
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      20       4      20       1   0.143 0.05349 0.05736 0.07196 0.12277
##     .75     .90     .95
## 0.17772 0.23299 0.25043
##
## lowest : 0.006595 0.055954 0.057513 0.060506 0.063725
## highest: 0.182559 0.183056 0.232534 0.237086 0.503936
## ------------------------------------------------------------------------------
```

And ploted:

```
boxplot(sourceDataGroups[1:4],xlab="Team members",ylab="Time",main="Time X Groups")
```




```
boxplot(sourceDataGroups[5:8],xlab="Team members",ylab="Error",main="Error X Groups")
```




```
#boxplot(timeXGroups,xlab="Team members",ylab="Time",main="Time X Groups")
#boxplot(errorXGroups,xlab="Team members",ylab="Error",main="Error X Groups")
```

Correlation Analysis

The Pearson product-moment correlation coefficient is a measure of the linear correlation between two variables X and Y, giving a value between +1 and -1 inclusive, where 1 is total positive correlation, 0 is no correlation, and -1 is total negative correlation. It is widely used in the sciences as a measure of the degree of linear dependence between two variables.

First we need to take the average of time and error of each team:

```
sourceDataTasks$MeanTime <- rowMeans(subset(sourceDataTasks, select = c(2,3,4,5)), na.rm = TRUE)
sourceDataTasks$MeanError <- rowMeans(subset(sourceDataTasks, select = c(6,7,8,9)), na.rm = TRUE)
```

Now we can make the pearson correlation analysis. It will tell us if there is a correlation between team members and time to solve the tasks, and team members and errors performed during the tasks.

First, the team members vs. time:

```
teamTimeCorr <- sourceDataTasks[,c(1,10)]
teamTimeCorr <- rcorr(as.matrix(teamTimeCorr))
teamTimeCorr
```

```
##              Members.Task MeanTime
## Members.Task         1.00    -0.19
## MeanTime            -0.19     1.00
##
## n= 22
##
##
## P
##              Members.Task MeanTime
## Members.Task              0.397
## MeanTime     0.397
```

Team members vs. error:

```
teamErrorCorr <- sourceDataTasks[,c(1,11)]
teamErrorCorr <- rcorr(as.matrix(teamErrorCorr))
teamErrorCorr
```

```
##              Members.Task MeanError
## Members.Task         1.00     -0.79
## MeanError           -0.79      1.00
##
## n= 22
##
##
## P
##              Members.Task MeanError
## Members.Task               0
## MeanError     0
```

As the is greater than 0.05, we assume that there is no correlation between team members and time to complete the tasks. In the other hand, is < 0.0001, so there is a strong correlation between team members and errors performed. At this point we don't know if the errors grow with more team members or in the other way around.

Shapiro-Wilk test

Before we conduce a variance test on the data to evaluate if more or less members cause more errors, we need to check if the data is normally distributed. For this test, we use the Shapiro-wilk test for each group of team members.

```
sapply(lapply(sourceDataGroups[1:8], shapiro.test), `[`, c("statistic","p.value"))
```

```
##           Time.T1      Time.T2   Time.T3      Time.T4     Error.T1
## statistic 0.7970137    0.8924259 0.7497053    0.8422909   0.888081
## p.value   0.0007800236 0.0149233 4.931358e-05 0.003969711 0.02479839
##           Error.T2    Error.T3    Error.T4
## statistic 0.8642574   0.8581019   0.8685364
## p.value   0.004068348 0.003102281 0.01107456
```

As the $P < 0.05$ of most tests, we know that our data are not normally distributed. For this case, and because we are comparing distinct groups, the variance test will be performed by a Kruskal-Wallis.

Kruskal-Wallis

We perform the Kruskal test with Dunn posthoc with the time data:

```
library(dunn.test)
timeToDunn <- gather(sourceDataGroups, "group", "errors", 1:4)
dunn.test(timeToDunn$error,timeToDunn$group, kw=TRUE, method="bonferroni")
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 0.4131, df = 3, p-value = 0.94
##
##
##                             Comparison of x by group
##                                   (Bonferroni)
## Col Mean-|
## Row Mean |    Time.T1    Time.T2    Time.T3
## ---------+---------------------------------
##  Time.T2 |   0.458962
##          |     1.0000
##          |
##  Time.T3 |   0.011851  -0.468934
##          |     1.0000     1.0000
##          |
##  Time.T4 |   0.439422   0.000000   0.447111
##          |     1.0000     1.0000     1.0000
```

... and with the error data:

```
errorToDunn <- gather(sourceDataGroups, "group", "errors", 5:8)
dunn.test(errorToDunn$error,errorToDunn$group, kw=TRUE, method="bonferroni")
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 23.2985, df = 3, p-value = 0
##
```

```
##
##                            Comparison of x by group
##                                  (Bonferroni)
## Col Mean-|
## Row Mean |   Error.T1   Error.T2   Error.T3
## ---------+---------------------------------
## Error.T2 |   1.671010
##          |     0.2842
##          |
## Error.T3 |   3.141628   1.542397
##          |     0.0050     0.3689
##          |
## Error.T4 |   4.573711   3.106075   1.635456
##          |     0.0000     0.0057     0.3059
```

Comparison between tasks

Hypotesis

- H1. Task 1 and 2 are easier than 3 and 4;
- H2. Task 4 is performed with less errors than task 3;
- H3. Task 4 is performed in less time than task 3;

If H2 or/and H3 is confirmed, we can say that the teams have improved their performance with only one trainning.

Summary

The data is arranged by Task vs. Team members. Collumns are organized as follows:

- Members: Number of users in the team;
- T1, T2, T3, T4: Time to complete task 1 to 4;
- E1, E2, E3, E4: Errors in task 1 to 4.

Below the data is summarized:

```
describe(sourceDataTasks[2:5])
```

```
## sourceDataTasks[2:5]
##
##  4  Variables      22  Observations
## ---------------------------------------------------------------------------
## Time.task.1
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   35.82   16.49   17.42   20.43   30.83
##     .75     .90     .95
##   46.55   59.27   70.82
##
## lowest : 13.65 16.45 17.21 19.36 19.92
## highest: 54.20 58.67 59.33 71.43 86.27
## ---------------------------------------------------------------------------
## Time.task.2
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   27.94   15.97   16.27   17.01   26.15
```

```
##     .75     .90     .95
##   34.04   42.18   48.82
##
## lowest : 14.50 15.95 16.25 16.40 16.48
## highest: 37.12 41.50 42.26 49.16 55.89
## ---------------------------------------------------------------------------
## Time.task.3
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   120.6   45.10   57.70   79.12  107.59
##     .75     .90     .95
##  130.45  207.22  278.12
##
## lowest :  41.30  44.51  56.40  69.36  74.45
## highest: 139.38 162.96 212.14 281.60 283.41
## ---------------------------------------------------------------------------
## Time.task.4
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   92.78   55.36   56.87   61.79   85.12
##     .75     .90     .95
##  106.56  137.01  190.07
##
## lowest :  38.12  55.29  56.70  58.42  60.77
## highest: 113.12 113.40 139.63 192.72 225.21
## ---------------------------------------------------------------------------
```

```
describe(sourceDataTasks[6:9])
```

```
## sourceDataTasks[6:9]
##
##  4  Variables      22  Observations
## ---------------------------------------------------------------------------
## Error.task.1
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   2.487   1.302   1.325   1.572   2.226
##     .75     .90     .95
##   3.110   4.255   4.447
##
## lowest : 0.8673 1.3007 1.3175 1.3909 1.4022
## highest: 3.8697 4.0249 4.2804 4.4561 4.6016
## ---------------------------------------------------------------------------
## Error.task.2
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   6.386   2.993   3.251   3.669   5.024
##     .75     .90     .95
##   8.830   9.199  11.627
##
## lowest :  2.768  2.980  3.243  3.317  3.358
## highest:  8.904  9.003  9.220 11.753 16.953
## ---------------------------------------------------------------------------
## Error.task.3
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   4.335   2.378   2.416   3.063   4.065
##     .75     .90     .95
##   4.832   6.757   7.353
##
```

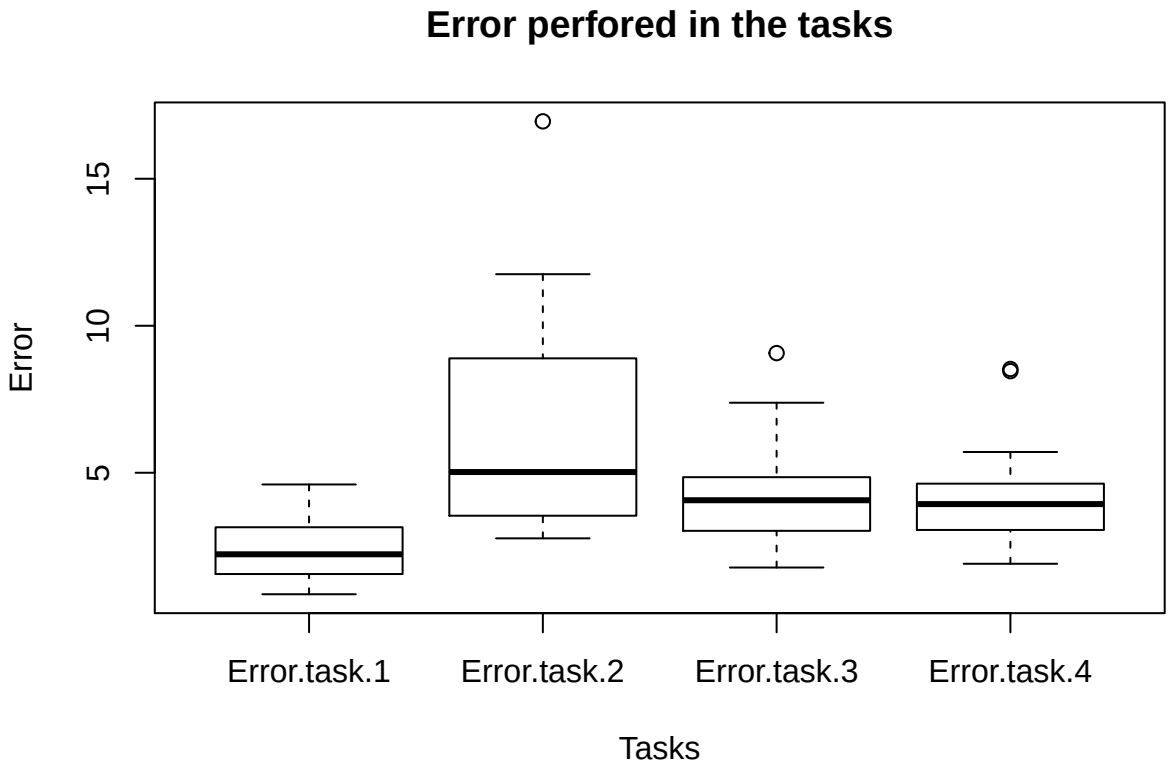
```
## lowest : 1.776 2.377 2.390 2.652 2.972
## highest: 6.038 6.208 6.818 7.382 9.070
## ------------------------------------------------------------------------
## Error.task.4
##       n missing  unique    Info    Mean     .05     .10     .25     .50
##      22       0      22       1   4.163   1.928   2.129   3.105   3.934
##     .75     .90     .95
##   4.617   5.689   8.318
##
## lowest : 1.904 1.918 2.111 2.289 2.953
## highest: 5.089 5.541 5.706 8.455 8.530
## ------------------------------------------------------------------------
```

Plots

Time of task completion vs. Task for all combinations of teams: The code used to generate the charts is:

```
boxplot(sourceDataTasks[2:5],xlab="Tasks",ylab="Time",main="Time to complete the tasks")
```


```
boxplot(sourceDataTasks[6:9],xlab="Tasks",ylab="Error",main="Error perfored in the tasks")
```



Analysis of time of completion per task

Shapiro

First, we perform the Shapiro normality test. This test determine if the data is normally distributed. It is important to determine if the data is normally distributed to conduce posterior tests.

```
sapply(lapply(sourceDataTasks[1:8], shapiro.test), `[`, c("statistic","p.value"))
```

```
##           Members.Task Time.task.1 Time.task.2 Time.task.3 Time.task.4
## statistic 0.8727459    0.8843174   0.9113139   0.8505394   0.8207751
## p.value   0.008809021  0.01461996  0.05035917  0.003470118 0.001077677
##           Error.task.1 Error.task.2 Error.task.3
## statistic 0.912041     0.8457819    0.9259
## p.value   0.05211448   0.002861565  0.1007961
```

As we can see, the $p - value$ of most Shapiro tests reveled that the data are not normally distributed. Since in this test the comparisons are made with the same subjects and we are varying the tasks, the next step is to perform a Friedman analysis.

Friedman

Friedman test is a non-parametric randomized block analysis of variance. Which is to say it is a non-parametric version of a one way ANOVA with repeated measures. That means that while a simple ANOVA test requires the assumptions of a normal distribution and equal variances (of the residuals), the Friedman test is free from those restriction. The price of this parametric freedom is the loss of power (of Friedman's test compared to the parametric ANOVa versions).

The hypotheses for the comparison across repeated measures are:

- H0: The distributions (whatever they are) are the same across repeated measures
- H1: The distributions across repeated measures are different

The test statistic for the Friedman's test is a Chi-square with [(number of repeated measures)-1] degrees of freedom. A detailed explanation of the method for computing the Friedman test is available on Wikipedia.

Kruskal-Wallis

A collection of data samples are independent if they come from unrelated populations and the samples do not affect each other. Using the Kruskal-Wallis Test, we can decide whether the population distributions are identical without assuming them to follow the normal distribution.

```
{r fig.width=100,echo=FALSE} # library(png) # library(grid) #
img <- readPNG("../../Results/ErroXQntPessoas.png") #  grid.raster(img)
#
```

```
{r fig.width=100,echo=FALSE} # library(png) # library(grid) #
img <- readPNG("../../Results/ErroPerCheckpointXGrupos.png") #
grid.raster(img) #
```

```
{r fig.width=100,echo=FALSE} # library(png) # library(grid) #
img <- readPNG("../../Results/kruskal_posHoc_Student.png") #
grid.raster(img) #
```

Comparisons between tasks

Hypotesis

*** Task 1 and 2 are easier than 3 and 4;**