SCEM Final Coursework

Task I

Rohan Anthony (2704500)

IMPORTANT NOTES:

- **DO NOT** change the code block names. Enter your solutions to each question into the predefined code blocks.
- **DO NOT** add calls to `install.packages()` or load any additional packages in your code blocks. Calls to `install.packages()` may result in zero marks for the activity where it happens.
- **DO NOT** load any additional packages. All packages allowed and required for this coursework are already loaded in the **Setup** code block. Use of other packages may result in execution errors when marking, which can result in zero marks for one or more activities.
- **Don't forget to replace the placeholder value in code block "ID" by your valid student ID number.**
- Instructions for this task are provided in blue text below. Your solution should NOT be added within those text boxes, and should NOT be in blue text. **During marking, all the text within the text blocks below, with names starting with "prompt", will be removed and not visible to the marker.**

```
##
## List of allowed packages (pre-loaded):
##  ggplot2
##  multcomp
##  devtools
##  dplyr
##  tidyr
##  knitr
##  ExpDE
```

In this task you will design, run, and analise an experiment comparing different configurations of the DE algorithm for a class of problems of interest. Here are the required statistical parameters for your experiment:

```
## Desired statistical parameters for your experimental design:
## ============================================================
## Significance level (alpha): alpha = 0.04
## Minimally relevant effect size (MRES): d* = 0.75
## Desired statistical power for MRES (desired power): (1-beta)* = 0.8
## Number of repeated runs for each method on each problem: nruns = 50
## ============================================================
```

Here are the methods you will compare in your experiment. You will be able to pass method names directly to the data-generating function later in this task.

```
## ============================================================
## Your experiment will compare 5 algorithm configurations
```

```
## Your selected methods are named:
## DE.2704500.A
## DE.2704500.B
## DE.2704500.C
## DE.2704500.D
## DE.2704500.E
## Each method can be passed by name as the argument method.name of function ExpDE2()
## If you want to inspect the specificities of each method, check the output list.
```

Question 1.1

1.1.a.

Enter the statistical hypotheses of your initial (omnibus) test for this task in the text block below. Provide a brief (less than 150 words) description of what is being tested.

We are trying to determine whether there is any statistically significant difference in performance between five DE configurations (DE.2704500.A-E). Each configuration is executed 50 times on the same optimization problem, producing a sample of loss values. A one-way ANOVA at alpha = 0.04 will be used to test for differences in mean loss among configurations.

$H_0 : \mu_A = \mu_B = \mu_C = \mu_D = \mu_E$

$H_1 : \exists i \neq j$ such that$\mu_i \neq \mu_j$

1.1.b.

The name of the problem you should use for this item is automatically generated and stored in a variable called `my.problem`.

Below is an example of how to generate a single observation for your first method on the test problem `my.problem`.

```
## DO NOT CHANGE ANYTHING IN THIS CODE BLOCK
x <- ExpDE2(method.name = method.names[1], problem.name = my.problem)
my_obs <- log(x)
my_obs
## [1] 1.979294
```

Adapt this code to generate the necessary number of observations (defined in your experimental parameters) for all of your test methods. Store your observations in a tidy data frame named `myres`, containing one observation per row and the following columns:

- `Method` (containing the method name)
- `Value` (containing the observation)

Your data frame `myres` should have (nruns x number of methods) rows.

```
set.seed(MY_STUDENT_ID) ## <--- DO NOT CHANGE THIS LINE

# ADD YOUR CODE HERE
myres <- data.frame(Method = character(exp_pars$nruns * 5),
                    Value = numeric(exp_pars$nruns * 5))

idx <- 1
```

```r
for (m in method.names) {
  for (i in seq_len(exp_pars$nruns)) {
    x <- ExpDE2(method.name = m, problem.name = my.problem)
    myres[idx, "Method"] <- m
    myres[idx, "Value"] <- as.numeric(log(x))
    idx <- idx + 1
  }
}

myres$Method <- factor(myres$Method)

dim(myres)
## [1] 250   2
head(myres)
##         Method    Value
## 1 DE.2704500.A 2.179876
## 2 DE.2704500.A 2.093849
## 3 DE.2704500.A 2.068901
## 4 DE.2704500.A 2.069979
## 5 DE.2704500.A 1.839927
## 6 DE.2704500.A 2.033670


## (DON'T FORGET THAT YOUR OBSERVATIONS SHOULD BE STORED IN A DATA FRAME CALLED
## myres WITH THE STRUCTURE DESCRIBED ABOVE).
```

1.1.c

Estimate the mean loss and associated parametric confidence interval (at the significance level provided in your experimental parameters) for all methods. Store your results in a single dataframe called `myCIs01`, with the following columns:

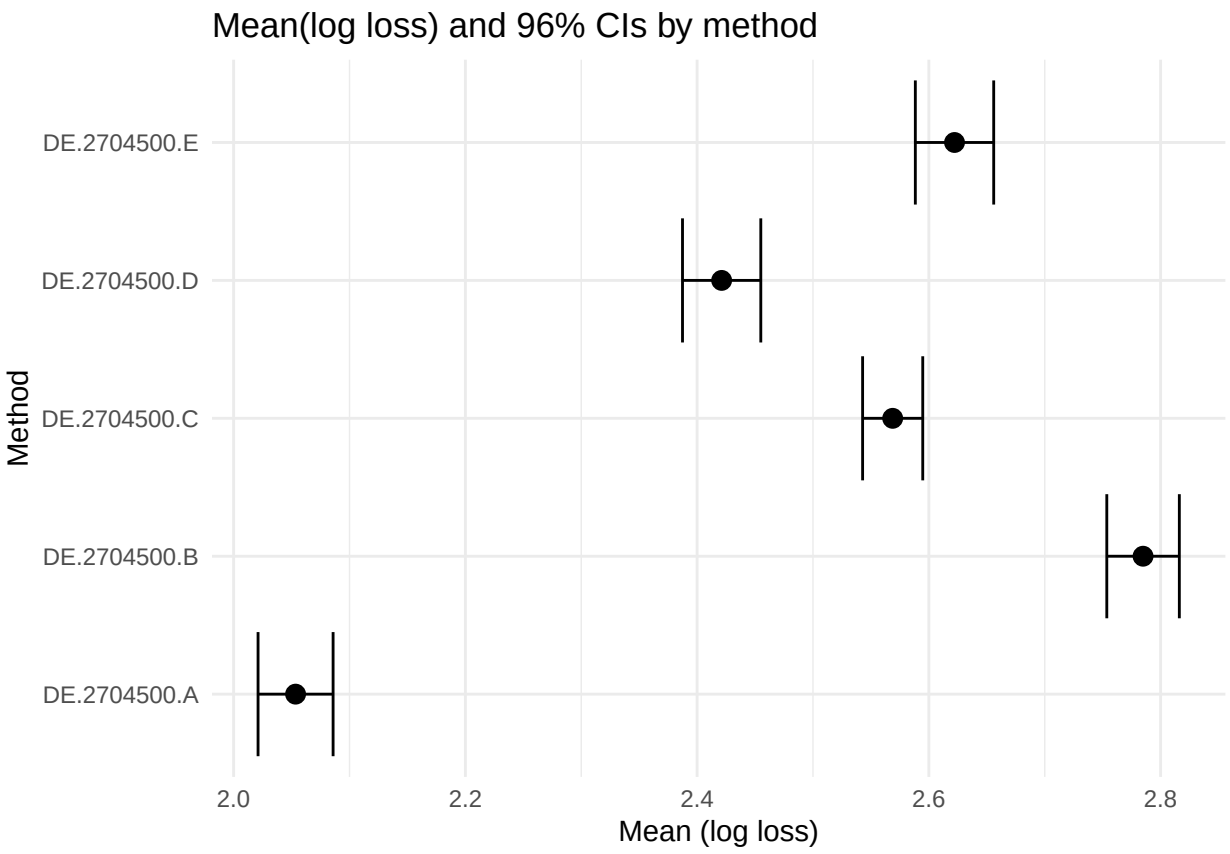
- `Method` (containing the method name)
- `Mean` (containing the sample estimate of the mean)
- `CI.lower` (containing the estimated lower bound of the confidence interval)
- `CI.upper` (containing the estimated upper bound of the confidence interval)

```r
# ADD YOUR CODE HERE
alpha <- exp_pars$alpha
methods_unique <- unique(myres$Method)

myCIs01 <- myres %>%
  group_by(Method) %>%
  summarise(
    Mean = mean(Value),
    n = n(),
    sd = sd(Value),
    se = sd(Value) / sqrt(n),
    t_crit = qt(1 - alpha / 2, df = n - 1),
    CI.lower = Mean - t_crit * se,
    CI.upper = Mean + t_crit * se
  ) %>%
  select(Method, Mean, CI.lower, CI.upper)
```

```
myCIs01
## # A tibble: 5 x 4
##   Method        Mean CI.lower CI.upper
##   <fct>        <dbl>    <dbl>    <dbl>
## 1 DE.2704500.A  2.05     2.02     2.09
## 2 DE.2704500.B  2.78     2.75     2.82
## 3 DE.2704500.C  2.57     2.54     2.59
## 4 DE.2704500.D  2.42     2.39     2.45
## 5 DE.2704500.E  2.62     2.59     2.66

## (DON'T FORGET THAT YOUR OUTPUT OBJECT SHOULD BE STORED IN A VARIABLE CALLED
## myCIs01).
```

Generate a plot of the confidence intervals using ggplot2. You plot should have the methods on the y-axis and the values on the x-axis - something similar (not identical) to the figure in this link (without the vertical dashed line):

```
# ADD YOUR CODE HERE
ggplot(myCIs01, aes(x = Mean, y = Method)) +
  geom_point(size = 3) +
  geom_errorbarh(aes(xmin = CI.lower, xmax = CI.upper)) +
  labs(
    title = paste0("Mean(log loss) and ", round((1 - exp_pars$alpha) * 100), "% CIs by method"),
    x = "Mean (log loss)",
    y = "Method"
  ) +
theme_minimal()
## Warning: `geom_errobarh()` was deprecated in ggplot2 4.0.0.
## i Please use the `orientation` argument of `geom_errorbar()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



1.1.d Statistical testing

Use the code block below to perform the adequate statistical test, according to the instructions of the coursework descriptor. Store the statistical test object in a variable called `mytest01`. Echo the results of your test (using the `summary()` function, if applicable) to print the results.

```
# ADD YOUR CODE HERE
mytest01 <- aov(Value ~ Method, data = myres)
summary(mytest01)
##              Df Sum Sq Mean Sq F value Pr(>F)
## Method        4 15.295   3.824   341.3 <2e-16 ***
## Residuals   245  2.745   0.011
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


## (DON'T FORGET THAT THE OBJECT RETURNED BY THE STATISTICAL TEST FUNCTION
## SHOULD BE STORED IN A VARIABLE CALLED mytest01).
```

Add a comment in the text block below (up to 100 words) indicating if your results were statistically significant, based on the observed p-value and your predefined significance level.

The ANOVA test indicates a statistically significant difference in mean log-loss across the five DE configurations. The p-value obtained is $<$ 2e-16, which is far below the predefined significance level alpha = 0.04. Therefore, we reject the null hypothesis that all methods have equal mean loss. We can conclude that at least one method's mean performance is significantly different from the others.

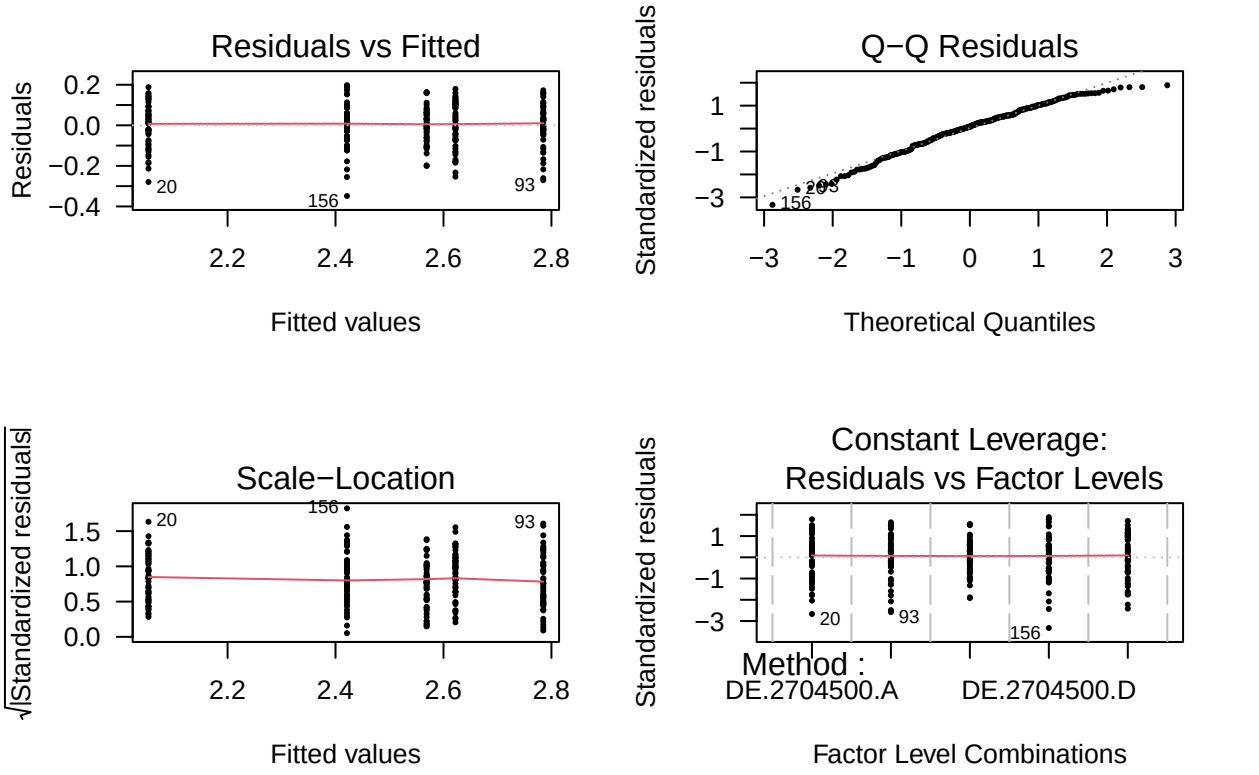
1.1.e Checking assumptions

Provide a brief (less than 150 words) description and explanation of the statistical assumptions of the test used, as a bullet list:

- Independence of Observations: The 50 samples for each method must be independent of one another.
- Normality of Residuals: The residuals from the ANOVA must be approximately normally distributed.
- Homogeneity of Variances: All methods should have similar variance in their log-loss values.
- Scale of Measurement: Dependent variable is measured on a continuous scale.
- Model Additivity - The effects of the Method configurations on the mean loss must combine additively.

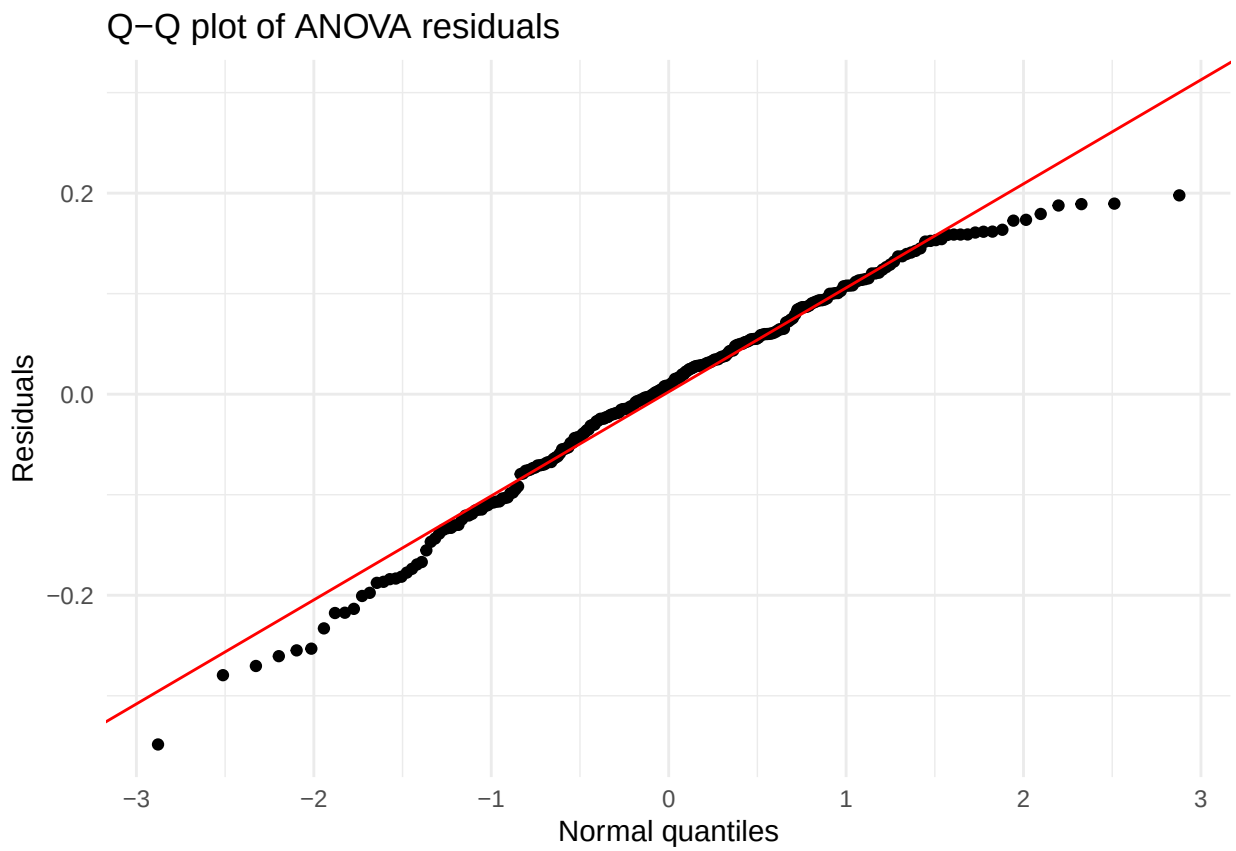
Generate residual plots which could be used to verify the remaining assumptions of your test (you can either use the default plotting methods of the statistical test, or generate the appropriate residual plots using, e.g., base R plots or ggplot2).

```
# ADD YOUR CODE HERE
par(mfrow = c(2, 2))
plot(mytest01, pch = 16, cex = 0.5, las = 1)
```

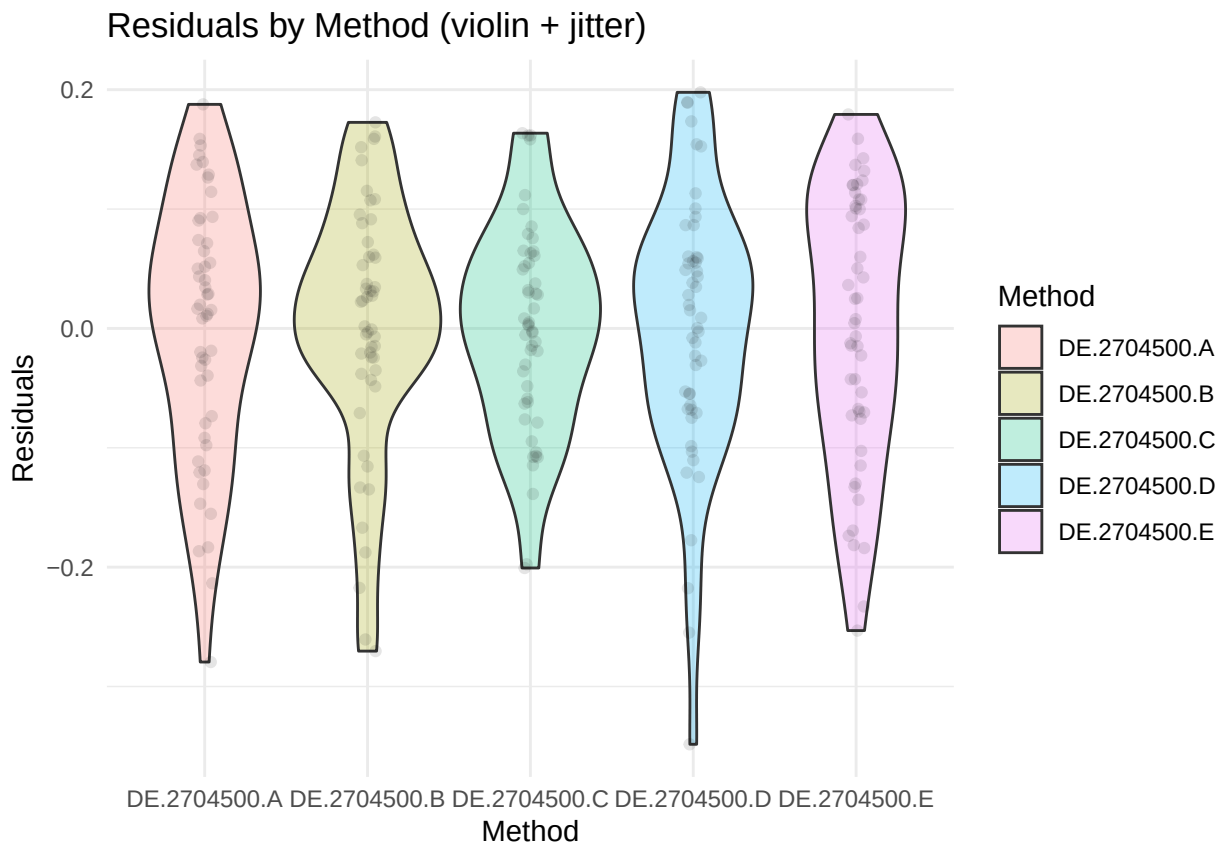



```
par(mfrow = c(1,1))

res <- residuals(mytest01)

ggplot(data.frame(x = res), aes(sample = x)) +
  geom_qq() +
  geom_qq_line(col = "red") +
```

```
  theme_minimal() +
  xlab("Normal quantiles") +
  ylab("Residuals") +
  ggtitle("Q-Q plot of ANOVA residuals")
```




```
res_df <- myres %>% mutate(Residual = res)

ggplot(res_df, aes(x = Method, y = Residual)) +
  geom_violin(aes(fill = Method), alpha = 0.25) +
  geom_jitter(width = .05, alpha = .1) +
  theme_minimal() +
  xlab("Method") +
  ylab("Residuals") +
  ggtitle("Residuals by Method (violin + jitter)")
```



1.1.f

Perform a post-hoc Tukey test, using the multcomp package. Store the object returned by the function that executes the Tukey test in variable `myMHT1`. Use the `summary()` function to inspect the results of these comparisons.

```
# ADD YOUR CODE HERE
library(multcomp)

myMHT1 <- glht(mytest01, linfct = mcp(Method = "Tukey"))
summary(myMHT1)
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: aov(formula = Value ~ Method, data = myres)
##
## Linear Hypotheses:
##                                    Estimate Std. Error t value Pr(>|t|)
## DE.2704500.B - DE.2704500.A == 0  0.73139    0.02117  34.551   <0.001 ***
## DE.2704500.C - DE.2704500.A == 0  0.51531    0.02117  24.343   <0.001 ***
## DE.2704500.D - DE.2704500.A == 0  0.36771    0.02117  17.370   <0.001 ***
## DE.2704500.E - DE.2704500.A == 0  0.56868    0.02117  26.864   <0.001 ***
```

```
## DE.2704500.C - DE.2704500.B == 0 -0.21609    0.02117 -10.208   <0.001 ***
## DE.2704500.D - DE.2704500.B == 0 -0.36369    0.02117 -17.180   <0.001 ***
## DE.2704500.E - DE.2704500.B == 0 -0.16271    0.02117  -7.686   <0.001 ***
## DE.2704500.D - DE.2704500.C == 0 -0.14760    0.02117  -6.973   <0.001 ***
## DE.2704500.E - DE.2704500.C == 0  0.05338    0.02117   2.521   0.0892 .
## DE.2704500.E - DE.2704500.D == 0  0.20097    0.02117   9.494   <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)

## (DON'T FORGET THAT YOUR OUTPUT OBJECT SHOULD BE STORED IN A VARIABLE CALLED
## myMHT1).
```

Extract the simultaneous confidence intervals of your Tukey test comparisons using function `confint()`, using the confidence level indicated by your specific experimental parameters. Store the object returned by `confint()` into a variable called myCIs02. Echo this object and inspect the table of confidence intervals.

```
# ADD YOUR CODE HERE

ci_level <- 1 - exp_pars$alpha
myCIs02 <- confint(myMHT1, level = ci_level)
print(myCIs02)
##
##   Simultaneous Confidence Intervals
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: aov(formula = Value ~ Method, data = myres)
##
## Quantile = 2.8311
## 96% family-wise confidence level
##
##
## Linear Hypotheses:
##                                   Estimate  lwr       upr
## DE.2704500.B - DE.2704500.A == 0  0.731393  0.671463  0.791324
## DE.2704500.C - DE.2704500.A == 0  0.515306  0.455376  0.575237
## DE.2704500.D - DE.2704500.A == 0  0.367707  0.307777  0.427637
## DE.2704500.E - DE.2704500.A == 0  0.568682  0.508751  0.628612
## DE.2704500.C - DE.2704500.B == 0 -0.216087 -0.276017 -0.156157
## DE.2704500.D - DE.2704500.B == 0 -0.363686 -0.423617 -0.303756
## DE.2704500.E - DE.2704500.B == 0 -0.162712 -0.222642 -0.102781
## DE.2704500.D - DE.2704500.C == 0 -0.147599 -0.207530 -0.087669
## DE.2704500.E - DE.2704500.C == 0  0.053375 -0.006555  0.113306
## DE.2704500.E - DE.2704500.D == 0  0.200975  0.141044  0.260905

## (DON'T FORGET THAT YOUR OUTPUT OBJECT SHOULD BE STORED IN A VARIABLE CALLED
## myCIs02).
```

Tukey post-hoc test shows that almost all method pairs differ significantly at alpha = 0.04. All comparisons have adjusted p-values < 0.04 except E - C, whose confidence interval includes zero. This tells us that all methods differ significantly from each other except DE.2704500.E vs DE.2704500.C

1.1.g

Based on your point estimates obtained in item 1.1.c, what is the method that presents the best (i.e., the smallest) mean loss? Set the variable `best_method` below with the name of that method:

```
# ADD YOUR CODE HERE


## (DON'T FORGET THAT THE *NAME* OF THE BEST METHOD BASED ON THE POINT ESTIMATE OF THE MEAN SHOULD BE S
```

Based on the CIs you calculated after the Tukey tests, what is the confidence interval on the difference of the means between the best method above and the second-best one? Store your confidence interval in a single-row dataframe called `topComparison`, with the following columns:

- `Method1` (containing the name of the best method that you identified above)
- `Method2` (containing the name of the second-best method)
- `DiffMeans` (containing the estimated difference of means)
- `CI.lower` (containing the estimated lower bound of the confidence interval)
- `CI.upper` (containing the estimated upper bound of the confidence interval)

```
# ADD YOUR CODE HERE


## (DON'T FORGET TO STORE YOUR RESULTS AS A SINGLE-ROW DATAFRAME CALLED topComparison ).
```

Using the pooled estimate of the residual variance provided by the omnibus method (item 1.1.d) and the estimated difference of means of the top two methods, calculate the observed standardised effect size (Cohen's d coefficient) between the best and second-best methods. Store this value as a variable called `CohenD`

```
# ADD YOUR CODE HERE


## (DON'T FORGET TO STORE YOUR RESULTS AS A NUMERIC VARIABLE CALLED CohenD ).
```

Provide a brief comment (up to 150 words) on whether the best method (in terms of point estimate of mean loss) was significantly better than the second-best one at your predefined significance level, and comparing the observed value of the Cohen's D coefficient with the miminally relevant effect size indicated in your experimental parameters, $d*$

Some text here...

Question 1.2

1.2.a

Enter the statistical hypotheses of your initial (omnibus) test in for this task the text block below. Provide a brief (less than 150 words) description of what is being tested.

Some text here...

H_0 :

H_1 :

1.2.b

Based on the instructions provided in the coursework specs, calculate the required number of test problems necessary for your experiment to have the desired statistical properties. As a reminder, your experimental parameters are given by:

```
## Desired statistical parameters for your experimental design:
## ============================================================
## Significance level (alpha): alpha = 0.04
## Minimally relevant effect size (MRES): d* = 0.75
## Desired statistical power for MRES (desired power): (1-beta)* = 0.8
## Number of repeated runs for each method on each problem: nruns = 50
## ============================================================
```

Use the code block below to calculate your required number of problems, based on the experimental parameters above and the explicit instructions provided in the CW specs. Store your number of problems as a variable called `nprobs`.

```
# ADD YOUR CODE HERE


## (DON'T FORGET TO STORE YOUR RESULT AS A VARIABLE CALLED nprobs).
```

1.2.c

The names of the problems you should use for this item are automatically generated and stored in a variable called `problem.names`. The dimensions of each test problem are stored in a variable called `problem.dimensions`.

```
## Warning: ATTENTION. VARIABLE nprobs NOT DEFINED PLEASE COMPLETE ITEM 1.2.b.
```

Here's an example of how to generate a single observation for your first method on the first test problem `problem.names[1]`:

```
## DO NOT CHANGE ANYTHING IN THIS CODE BLOCK
x <- ExpDE2(method.name = method.names[1], problem.name = problem.names[1])
my_obs <- log(x)
my_obs
## [1] 4.321683
```

Adapt the code above to generate your experimental observations. The number of problems is defined by your `nprobs` and the number of repeated runs of each method on each problem is defined in your experimental parameters (nruns). Store your observations in a tidy data frame named `myres2`, containing one observation per row and the following columns:

- `Method` (containing the method name)
- `Problem` (containing the problem name)
- `Problem_type` (containing the problem type)
- `Problem_dimension` (containing the problem dimension)
- `Value` (containing the observation)

Set the first three columns as **factor** variables. For column `Method`, set your reference method (the one defined as the best one in the preliminary tests, question 1.1.g) as the **reference** for that factor. The information on problem types and problem dimensions is available in vectors `problem.type` and `problem.dimension`, respectively.

Your data frame `myres2` should have (nprobs x nruns x number of methods) rows

```r
set.seed(MY_STUDENT_ID) ## <--- DO NOT CHANGE THIS LINE

# ADD YOUR CODE HERE



## (DON'T FORGET THAT YOUR OBSERVATIONS SHOULD BE STORED IN A DATA FRAME CALLED
## myres2 WITH THE STRUCTURE DESCRIBED ABOVE).
```

1.2.d Statistical testing

Use the code block below to perform the adequate statistical test, according to the instructions of the coursework descriptor. Store the statistical test object in a variable called `mytest02`, and use the `summary()` function to print the results.

```r
# ADD YOUR CODE HERE



## (DON'T FORGET THAT THE OBJECT RETURNED BY THE STATISTICAL TEST FUNCTION
## SHOULD BE STORED IN A VARIABLE CALLED mytest02).
```

Add a comment in the text block below (up to 100 words) indicating if your results were statistically significant, based on the observed p-value and your predefined significance level.

Some text here...

1.2.e

Perform a post-hoc Dunnett test, using the multcomp package. Store the object returned by the function that executes the Tukey test in variable `myMHT2`. Use the `summary()` function to inspect the results of these comparisons.

```r
# ADD YOUR CODE HERE


## (DON'T FORGET THAT YOUR OUTPUT OBJECT SHOULD BE STORED IN A VARIABLE CALLED
## myMHT2).
```

Extract the simultaneous confidence intervals of your Dunnett test comparisons using function `confint()`, using the confidence level indicated by your specific experimental parameters. Store the object returned by `confint()` into a variable called myCIs03. Echo this object and inspect the table of confidence intervals.

```r
# ADD YOUR CODE HERE


## (DON'T FORGET THAT YOUR OUTPUT OBJECT SHOULD BE STORED IN A VARIABLE CALLED
## myCIs03).
```

1.2.f

Based on your results, which of the methods would recommend as the selected strategy for your company's needs? Is it the same one as the preliminary test indicated? Is there a second-best that is close enough to be worth mentioning? Use the text block below and add your comments on the results of this experiment (up to 300 words)

Some text here...

Enter the name of your final method recommendation ("best method") as variable `best_final` below

```
# ADD YOUR CODE HERE


## (DON'T FORGET THAT THE NAME OF THE BEST METHOD SHOULD BE STORED AS VARIABLE
## best_final).
```

========== **End of Task I** ==========