# PerceptionAndAction_rubes

MeiSanderson

2024-12-30

## Analysis of experiment data

```
overlord <- read_csv("PA_RubeCube_Experiment.csv")
```

```
## New names:
## Rows: 9 Columns: 13
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (6): Tidsstempel, RESEARCHER ONLY: Condition, Consent, Please supply you... dbl
## (7): How old are you?, How confident are you about your ability to succe...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `How difficult do you find this learning process to be for you?` -> `How
##   difficult do you find this learning process to be for you?...10`
## * `How difficult do you find this learning process to be for you?` -> `How
##   difficult do you find this learning process to be for you?...12`
```

```
## re-naming columns
colnames(data)
```

```
## NULL
```

```
colnames(overlord)[1] <- "Timeslot"
colnames(overlord)[2] <- "Condition"
colnames(overlord)[3] <- "Consent"
colnames(overlord)[4] <- "ParticipantID"
colnames(overlord)[5] <- "Gender"
colnames(overlord)[6] <- "Age"
colnames(overlord)[7] <- "Confidence_1"
colnames(overlord)[8] <- "Difficulty_1"
colnames(overlord)[9] <- "Confidence_2"
colnames(overlord)[10] <- "Difficulty_2"
colnames(overlord)[11] <- "Confidence_3"
colnames(overlord)[12] <- "Difficulty_3"
colnames(overlord)[13] <- "SolveTime"
overlord$SolveTime <- as.numeric(overlord$SolveTime)

## renaming the conditions to their more concise name
overlord$Condition[overlord$Condition == "N-E: Novice presenting as Expert"] <- "N_E"
```

```r
overlord$Condition[overlord$Condition == "N-N: Novice presenting as Novice"] <- "N_N"
overlord$Condition[overlord$Condition == "E-N: Expert presenting as Novice"] <- "E_N"
overlord$Condition[overlord$Condition == "E-E: Expert presenting as Expert"] <- "E_E"


## removing the one participant
data <- filter(overlord, ParticipantID != "SR08")
```

```r
# colnames(data)


## OVERALL

cat("DESCRIPTIVE STATS FOR ALL PARTICIPANTS", "\n\n")
```

```
## DESCRIPTIVE STATS FOR ALL PARTICIPANTS
```

```r
## age
age_mean <- mean(data$Age) ### mean
age_sd <- sd(data$Age) ### sd
age_median <- median(sort(data$Age)) ### median

cat("Mean age:", age_mean, "Sd age:", age_sd, "\n", "Median age:", age_median, "\n")
```

```
## Mean age: 20.875 Sd age: 1.246423
##  Median age: 21
```

```r
## gender
gender_count <- table(data$Gender)
cat("Count of genders: female = ", gender_count[1], "& male = ", gender_count[2], "\n\n\n ")
```

```
## Count of genders: female =  6 & male =  2
##
##
##
```

```r
## PER CONDITION

cat("DESCRIPTIVE STATS FOR THE CONDITIONS", "\n\n")
```

```
## DESCRIPTIVE STATS FOR THE CONDITIONS
```

```r
### - pro x pro

## age
age_mean_ee <- mean(filter(data, Condition == "E_E")$Age) ### mean (aka the median since we have only t
age_sd_ee <- sd(filter(data, Condition == "E_E")$Age) ### sd

cat("Mean age_ee:", age_mean_ee, "Sd age_ee:", age_sd_ee, "\n")
```

```
## Mean age_ee: 20.5 Sd age_ee: 0.7071068
```

```r
## gender
gender_count_ee <- table(filter(data, Condition == "E_E")$Gender)
cat("Count of genders_ee: female = ", gender_count_ee[1], "& male = ", gender_count_ee[2], "\n\n")
```

```
## Count of genders_ee: female =  2 & male =  NA
```

```r
### pro x novice

## age
age_mean_en <- mean(filter(data, Condition == "E_N")$Age) ### mean
age_sd_en <- sd(filter(data, Condition == "E_N")$Age) ### sd

cat("Mean age_en:", age_mean_en, "Sd age_en:", age_sd_en, "\n")
```

```
## Mean age_en: 22 Sd age_en: 1.414214
```

```r
## gender
gender_count_en <- table(filter(data, Condition == "E_N")$Gender)
cat("Count of genders_en: female = ", gender_count_en[1], "& male = ", gender_count_en[2], "\n\n")
```

```
## Count of genders_en: female =  1 & male =  1
```

```r
### novice x novice

## age
age_mean_nn <- mean(filter(data, Condition == "N_N")$Age) ### mean
age_sd_nn <- sd(filter(data, Condition == "N_N")$Age) ### sd

cat("Mean age_nn:", age_mean_nn, "Sd age_nn:", age_sd_nn, "\n")
```

```
## Mean age_nn: 21 Sd age_nn: 1.414214
```

```r
## gender
gender_count_nn <- table(filter(data, Condition == "N_N")$Gender)
cat("Count of genders_nn: female = ", gender_count_nn[1], "& male = ", gender_count_nn[2], "\n\n")
```

```
## Count of genders_nn: female =  1 & male =  1
```

```r
### novice x pro

## age
age_mean_ne <- mean(filter(data, Condition == "N_E")$Age) ### mean
age_sd_ne <- sd(filter(data, Condition == "N_E")$Age) ### sd

cat("Mean age_ne:", age_mean_ne, "Sd age_ne:", age_sd_ne, "\n")
```

```
## Mean age_ne: 20 Sd age_ne: 1.414214
```

```r
## gender
gender_count_ne <- table(filter(data, Condition == "N_E")$Gender)
cat("Count of genders_ne: female = ", gender_count_ne[1], "& male = ", gender_count_ne[2], "\n\n")
```

```
## Count of genders_ne: female =  2 & male =  NA
```

### Main hypothesis (RQ, H1) testing

```r
## data frame with the conditions and mean solve time for each - not used for t-tests but could have bee
data_dummy <- as.data.frame(data %>% group_by(Condition = as.factor(Condition))
                             %>% dplyr::summarise(SolveTime = mean(as.numeric(SolveTime))))


# - - - - -


# BODILY CUES
cat("BODILY CUES", "\n\n")
```

```
## BODILY CUES
```

```r
## nxn and nxe
currentConditions <- filter(data, Condition == "N_N" | Condition == "N_E")

currentDataFrame <- as.data.frame(currentConditions %>% group_by(Condition)
                             %>% dplyr::summarise(Performance = mean(as.numeric(SolveTime))))


### adding the dummy coded conditions
data_dummy_nnAndne <- ifelse(currentDataFrame$Condition == "N_N", 1, 0) ### nn = 1, ne = 0

currentDataFrame$Dummy <- data_dummy_nnAndne


### testing for normality - not done because of the lack of data points would make any result extremely

### the non-parametric unpaired two-sample t-test substitute, Mann-Whitney U test, is used instead
cat("non-parametric unpaired two-sample t-test for nxn and nxe:", "\n")
```

```
## non-parametric unpaired two-sample t-test for nxn and nxe:
```

```r
wilcox.test(Performance ~ Dummy, currentDataFrame, exact = TRUE) ### the small sample size means no rel
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  Performance by Dummy
## W = 0, p-value = 1
## alternative hypothesis: true location shift is not equal to 0
```

```r
## exe and exn
currentConditions <- filter(data, Condition == "E_E" | Condition == "E_N")

currentDataFrame <- as.data.frame(currentConditions %>% group_by(Condition)
                              %>% dplyr::summarise(Performance = mean(as.numeric(SolveTime))))

### adding the dummy coded conditions
data_dummy_eeAnden <- ifelse(currentDataFrame$Condition == "E_E", 1, 0) ### ee = 1, en = 0

currentDataFrame$Dummy <- data_dummy_eeAnden


### testing for normality - not done because of the lack of data points would make any result extremely

### the non-parametric unpaired two-sample t-test substitute, Mann-Whitney U test, is used instead
cat("non-parametric unpaired two-sample t-test for exe and exn:", "\n")
```

```
## non-parametric unpaired two-sample t-test for exe and exn:
```

```r
wilcox.test(Performance ~ Dummy, currentDataFrame, exact = TRUE) ### the small sample size means no rel
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  Performance by Dummy
## W = 0, p-value = 1
## alternative hypothesis: true location shift is not equal to 0
```

```r
# - - - - -


# LINGUISTIC CUES
cat("LINGUISTIC CUES", "\n\n")
```

```
## LINGUISTIC CUES
```

```r
## nxn and exn
currentConditions <- filter(data, Condition == "N_N" | Condition == "E_N")

currentDataFrame <- as.data.frame(currentConditions %>% group_by(Condition)
                              %>% dplyr::summarise(Performance = mean(as.numeric(SolveTime))))

### adding the dummy coded conditions
data_dummy_nnAnden <- ifelse(currentDataFrame$Condition == "N_N", 1, 0) ### nn = 1, en = 0

currentDataFrame$Dummy <- data_dummy_nnAnden


### testing for normality - not done because of the lack of data points would make any result extremely

### the non-parametric unpaired two-sample t-test substitute, Mann-Whitney U test, is used instead
cat("non-parametric unpaired two-sample t-test for nxn and exn:", "\n")
```

```
## non-parametric unpaired two-sample t-test for nxn and exn:
```

```r
wilcox.test(Performance ~ Dummy, currentDataFrame, exact = TRUE) ### the small sample size means no rel
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  Performance by Dummy
## W = 0, p-value = 1
## alternative hypothesis: true location shift is not equal to 0
```

```r
# - - - - - -


# H2
cat("H2", "\n\n")
```

```
## H2
```

```r
## exe and nxe
currentConditions <- filter(data, Condition == "E_E" | Condition == "N_E")

currentDataFrame <- as.data.frame(currentConditions %>% group_by(Condition)
                              %>% dplyr::summarise(Performance = mean(as.numeric(SolveTime))))

### adding the dummy coded conditions
data_dummy_eeAndne <- ifelse(currentDataFrame$Condition == "E_E", 1, 0) ### ee = 1, ne = 0

currentDataFrame$Dummy <- data_dummy_eeAndne


### testing for normality - not done because of the lack of data points would make any result extremely

### the non-parametric unpaired two-sample t-test substitute, Mann-Whitney U test, is used instead
cat("non-parametric unpaired two-sample t-test for exe and nxe:", "\n")
```

```
## non-parametric unpaired two-sample t-test for exe and nxe:
```

```r
wilcox.test(Performance ~ Dummy, currentDataFrame, exact = TRUE) ### the small sample size means no rel
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  Performance by Dummy
## W = 0, p-value = 1
## alternative hypothesis: true location shift is not equal to 0
```

## H2 testing

```r
## nxn and exe
currentConditions <- filter(data, Condition == "N_N" | Condition == "E_E")

currentDataFrame <- as.data.frame(currentConditions %>% group_by(Condition)
                               %>% dplyr::summarise(Performance = mean(as.numeric(SolveTime))))

### adding the dummy coded conditions
data_dummy_nnAndee <- ifelse(currentDataFrame$Condition == "N_N", 1, 0) ### nn = 1, ee = 0

currentDataFrame$Dummy <- data_dummy_nnAndee


### testing for normality - not done because of the lack of data points would make any result extremely

### the non-parametric unpaired two-sample t-test substitute, Mann-Whitney U test, is used instead
cat("non-parametric unpaired two-sample t-test for nxn and exe:", "\n")
```

```
## non-parametric unpaired two-sample t-test for nxn and exe:
```

```r
wilcox.test(Performance ~ Dummy, currentDataFrame, exact = TRUE) ### the small sample size means no rel
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  Performance by Dummy
## W = 0, p-value = 1
## alternative hypothesis: true location shift is not equal to 0
```

```r
## exn and nxe
currentConditions <- filter(data, Condition == "E_N" | Condition == "N_E")

currentDataFrame <- as.data.frame(currentConditions %>% group_by(Condition)
                               %>% dplyr::summarise(Performance = mean(as.numeric(SolveTime))))

### adding the dummy coded conditions
data_dummy_enAndne <- ifelse(currentDataFrame$Condition == "E_N", 1, 0) ### en = 1, ne = 0

currentDataFrame$Dummy <- data_dummy_enAndne


### testing for normality - not done because of the lack of data points would make any result extremely

### the non-parametric unpaired two-sample t-test substitute, Mann-Whitney U test, is used instead
cat("non-parametric unpaired two-sample t-test for exn and nxe:", "\n")
```

```
## non-parametric unpaired two-sample t-test for exn and nxe:
```

```r
wilcox.test(Performance ~ Dummy, currentDataFrame, exact = TRUE) ### the small sample size means no rel
```

```
##
```

```
##  Wilcoxon rank sum exact test
##
## data:  Performance by Dummy
## W = 0, p-value = 1
## alternative hypothesis: true location shift is not equal to 0
```

## ANOVA

```r
## data frame with the conditions and mean solve time for each - not used for t-tests but could have bee
data_dummy <- as.data.frame(data %>% group_by(Condition = as.factor(Condition))
                              %>% dplyr::summarise(Performance = mean(as.numeric(SolveTime))))



### set contrasts to effects coding
contrasts(data_dummy$Condition) <- contr.sum(levels(data_dummy$Condition))


### the non-parametric anova substitute, kruskal.test, is used instead
cat("the non-parametric anova substitute for comparing all conditions:", "\n")
```

```
## the non-parametric anova substitute for comparing all conditions:
```

```r
kruskal.test(Performance ~ Condition, data_dummy)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Performance by Condition
## Kruskal-Wallis chi-squared = 3, df = 3, p-value = 0.3916
```

```r
### post-hoc testing - Dunn post-hoc testing
dunn.test(data_dummy$Performance, data_dummy$Condition, method = "bonferroni")
```

```
##   Kruskal-Wallis rank sum test
##
## data: x and group
## Kruskal-Wallis chi-squared = 3, df = 3, p-value = 0.39
##
##
##                            Comparison of x by group
##                                  (Bonferroni)
## Col Mean-|
## Row Mean |        E_E        E_N        N_E
## ---------+---------------------------------
##      E_N |   0.547722
##          |     1.0000
##          |
##      N_E |   1.095445   0.547722
##          |     0.8200     1.0000
```

```
##           |
##       N_N |  -0.547722  -1.095445  -1.643167
##           |     1.0000     0.8200     0.3010
##
## alpha = 0.05
## Reject Ho if p <= alpha/2
```

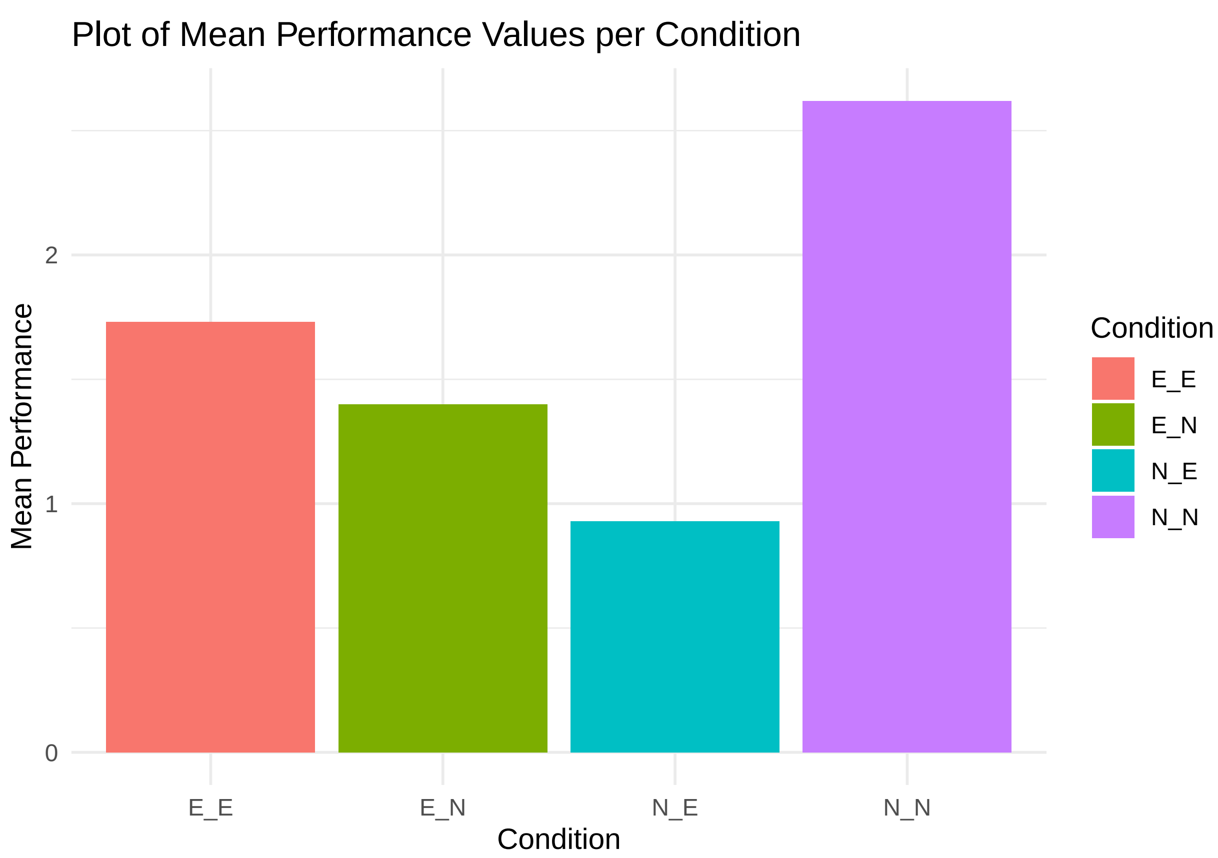
**SPOILER ALERT: no significant results at all**

### The Descriptive Stats

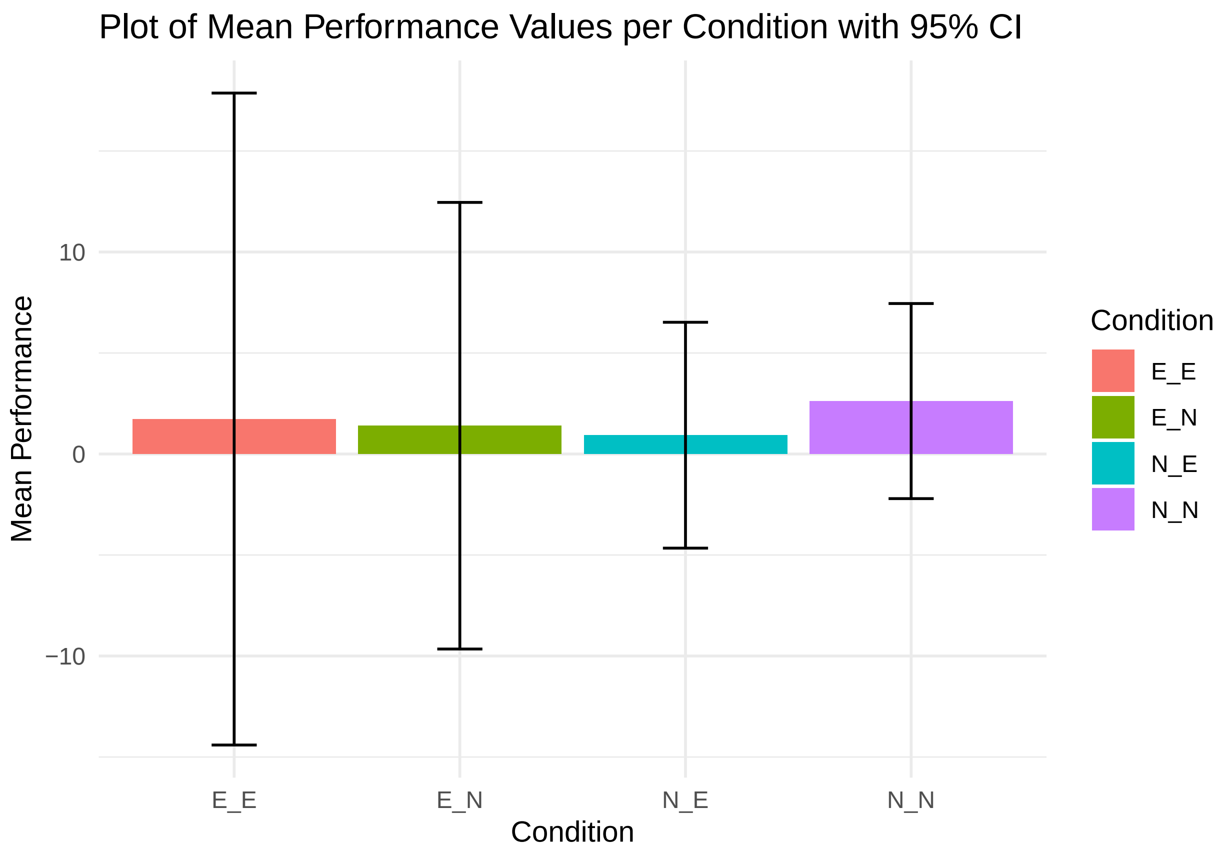
```
## DESCRIPTIVE STATS: calculated earlier - not much else to calculate :D
```

### The Visualisations we make instead

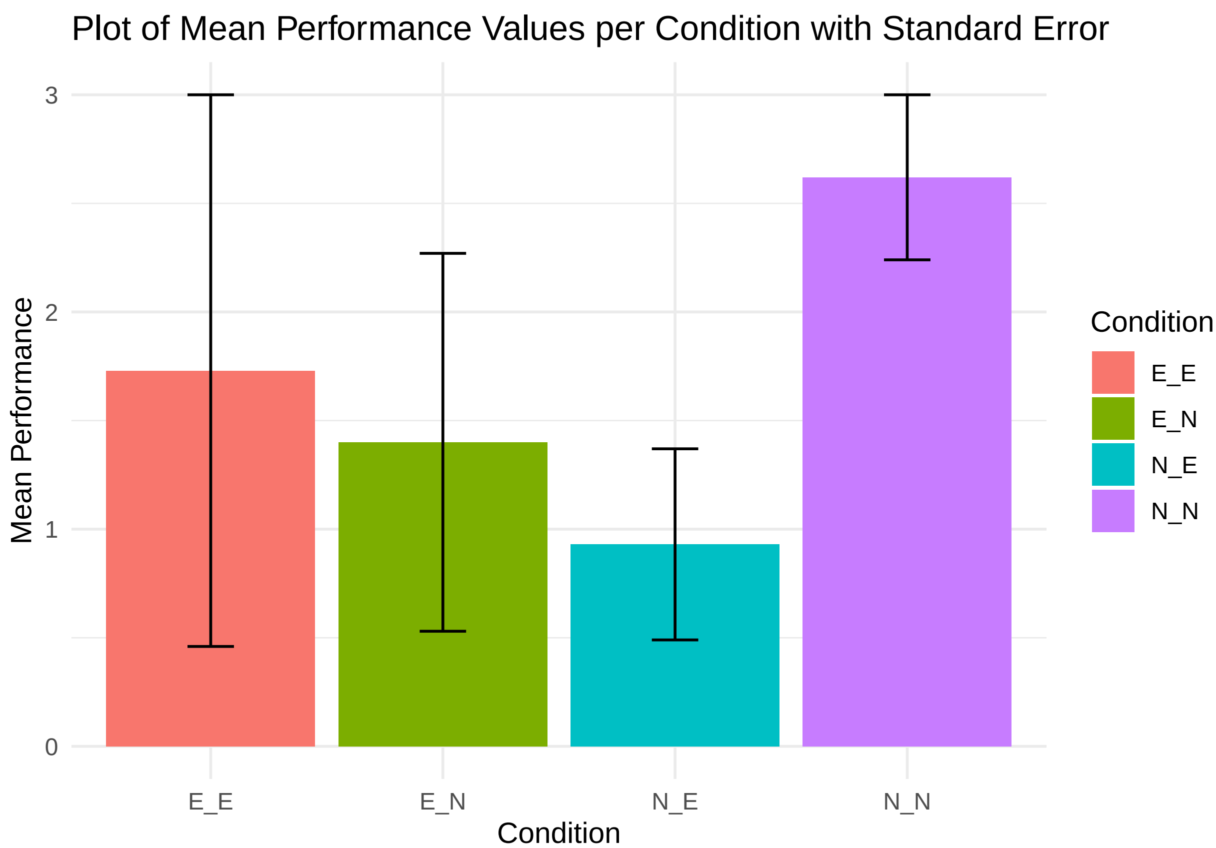
```
# plotting the means of each condition
ggplot(data_dummy, aes(x = Condition, y = Performance, fill = Condition)) +
  geom_col() +
  labs(title = "Plot of Mean Performance Values per Condition",
       x = "Condition",
       y = "Mean Performance") +
  theme_minimal()
```



```r
## same plot with confidence intervals
ggplot(data, aes(x = Condition, y = as.numeric(SolveTime), fill = Condition)) +
  stat_summary(fun = mean, geom = "bar") +
  stat_summary(fun.data = mean_cl_normal, geom = "errorbar", width = 0.2) +
  labs(title = "Plot of Mean Performance Values per Condition with 95% CI",
       x = "Condition",
       y = "Mean Performance") +
  theme_minimal()
```



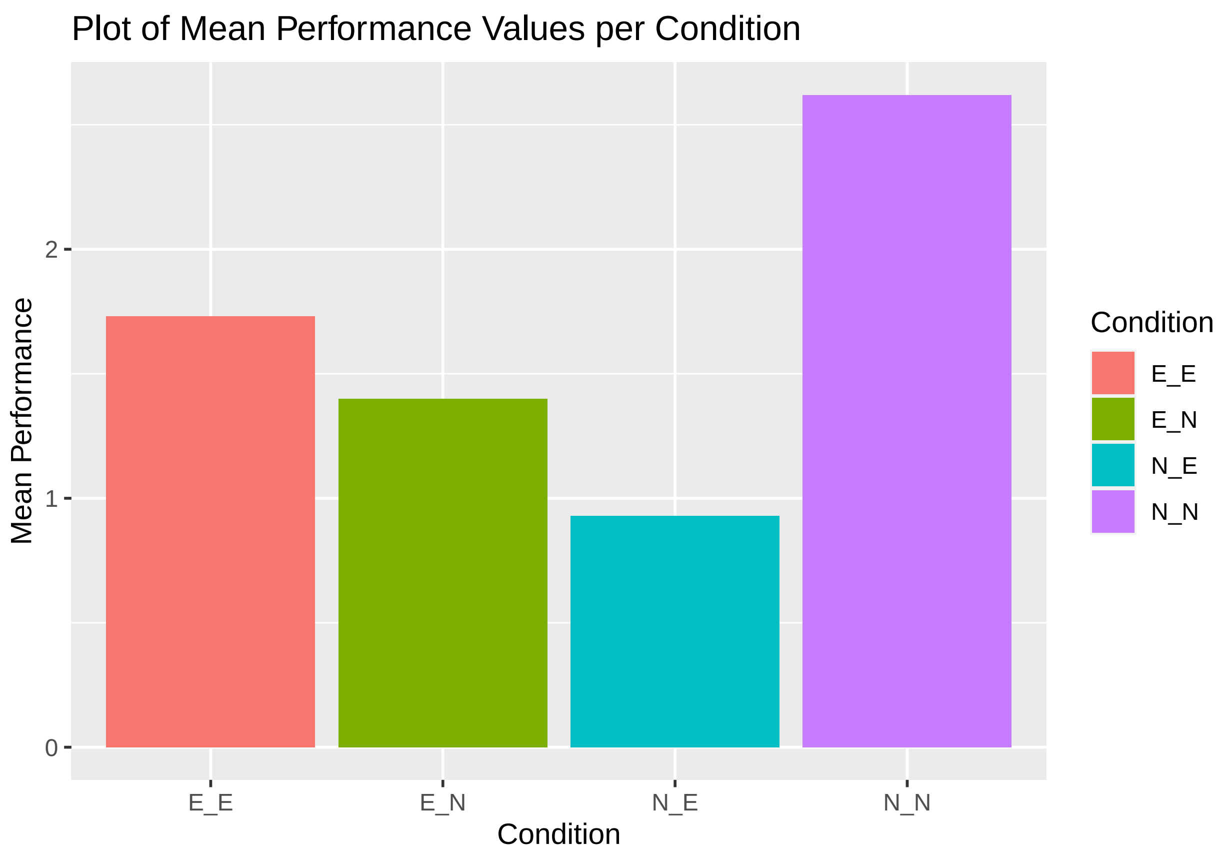

```r
## same plot with standard error
ggplot(data, aes(x = Condition, y = as.numeric(SolveTime), fill = Condition)) +
  stat_summary(fun = mean, geom = "bar") +
  stat_summary(fun.data = mean_se, geom = "errorbar", width = 0.2) +
  labs(title = "Plot of Mean Performance Values per Condition with Standard Error",
       x = "Condition",
       y = "Mean Performance") +
  theme_minimal()
```




```
# - - - - - - -

# different themes

ggplot(data, aes(x = Condition, y = as.numeric(SolveTime), fill = Condition)) +
  stat_summary(fun = mean, geom = "bar") +
  labs(title = "Plot of Mean Performance Values per Condition",
       x = "Condition",
       y = "Mean Performance") +
  theme_gray()
```




```
## VISUALISATIONS: boxplot or histogram, or perhaps scatter plot with the actual datapoints and not mea

# plotting the means of each condition in a more bar chart kinda way
```