UX-Game_SceneData

Jan Kirk

2023-05-17

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.0
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rjson)
library(jsonlite)
```

```
## Warning: pakke 'jsonlite' blev bygget under R version 4.2.3
```

```
##
## Vedhæfter pakke: 'jsonlite'
##
## De følgende objekter er maskerede fra 'package:rjson':
##
##     fromJSON, toJSON
##
## Det følgende objekt er maskeret fra 'package:purrr':
##
##     flatten
```

```
library(viridis);
```

```
## Warning: pakke 'viridis' blev bygget under R version 4.2.3
```

```
## Indlæser krævet pakke: viridisLite
```

```
library(RColorBrewer);
```

Loading data and creating dataframes.

```
sceneData_List <- fromJSON("sceneData.json")
sceneData <- as.data.frame(sceneData_List)
aNN_TrainingData <- fromJSON("C:/Users/Jan Kirk/Documents/GitHub/UX1_Bachelor/Assets/ML_Data/aNN_Traini
aNN_TrainingData_VALIDATE_EVERY_EPOCH <- fromJSON("C:/Users/Jan Kirk/Documents/GitHub/UX1_Bachelor/Asse
```

##Adding scene number to dataframe.

```
sceneData <- sceneData %>%
  group_by(gameId) %>%
  mutate(sceneNumber = row_number())
```

##Selecting the control group.

```
controlGroup <- subset(sceneData, isControlGroup == 1)
scenesInControlGroup <- nrow(controlGroup)
print(sprintf("Scenes in CONTROL group: %d", scenesInControlGroup))
```

```
## [1] "Scenes in CONTROL group: 23"
```

##Selecting the test group.

```
testGroup <- filter(sceneData, isControlGroup == 0)
scenesInTestGroup <- nrow(testGroup)
print(sprintf("Scenes in TEST group: %d", scenesInTestGroup))
```

```
## [1] "Scenes in TEST group: 27"
```

Plots

```
# Plotting control group general experience.
dataFrame <- controlGroup
subT  <- sprintf("Confidence interval 95%%, %d Games ~ %d Scenes.", length(unique(dataFrame$gameId)), n
ggplot(dataFrame, aes(x=sceneNumber, y=generalExp, color=gameId)) +
  geom_point(position = position_jitter(width = 0.15, height = 0.15)) +
  scale_x_continuous(limits = c(0.5, max(5.5))) +
  scale_y_continuous(limits = c(-0.5, max(4.5))) +
  scale_color_gradient(low = "black", high = "red") +
  geom_smooth(method="lm") +
  labs(title = "General experience in CONTROL group, by scene no.",
       subtitle = subT,
       x = "Scene number", y = "General experience")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```


```
# Plotting control group difficulty experience.
dataFrame <- controlGroup
subT  <- sprintf("Confidence interval 95%%, %d Games ~ %d Scenes.", length(unique(dataFrame$gameId)), ni
ggplot(dataFrame, aes(x=sceneNumber, y=diffExp, color=gameId)) +
  geom_point(position = position_jitter(width = 0.15, height = 0.15)) +
  scale_x_continuous(limits = c(0.5, max(5.5))) +
  scale_y_continuous(limits = c(-0.5, max(4.5))) +
  scale_color_gradient(low = "black", high = "red") +
  geom_smooth(method = "lm", formula = y ~ x) +
  labs(title = "Difficulty experience in CONTROL group, by scene no.",
       subtitle = subT,
       x = "Scene number", y = "Difficulty experience")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```


```
# Plotting TEST group general experience.
dataFrame <- testGroup
subT  <- sprintf("Confidence interval 95%%, %d Games ~ %d Scenes.", length(unique(dataFrame$gameId)), n
ggplot(dataFrame, aes(x=sceneNumber, y=generalExp, color=gameId)) +
  geom_point(position = position_jitter(width = 0.15, height = 0.15)) +
  scale_x_continuous(limits = c(0.5, max(5.5))) +
  scale_y_continuous(limits = c(-0.5, max(4.5))) +
  scale_color_gradient(low = "black", high = "green") +
  geom_smooth(method = "lm", formula = y ~ x) +
  labs(title = "General experience in TEST group, by scene no.",
      subtitle = subT,
      x = "Scene number", y = "General experience")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```


```
# Plotting TEST group difficulty experience.
dataFrame <- testGroup
subT  <- sprintf("Confidence interval 95%%, %d Games ~ %d Scenes.", length(unique(dataFrame$gameId)), n
ggplot(dataFrame, aes(x=sceneNumber, y=diffExp, color=gameId)) +
  geom_point(position = position_jitter(width = 0.15, height = 0.15)) +
  scale_x_continuous(limits = c(0.5, max(5.5))) +
  scale_y_continuous(limits = c(-0.5, max(4.5))) +
  scale_color_gradient(low = "black", high = "green") +
  geom_smooth(method = "lm", formula = y ~ x) +
  labs(title = "Difficulty experience in TEST group, by scene no.",
       subtitle = subT,
       x = "Scene number", y = "Difficulty experience")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```

Goodness of fit test:

Checking if distribution of generalExp answers, in CONTROL group, follows the null-distribution (H0) or not (HA).

```
# New data frame with unique experience values, and their counts.
df1 <- controlGroup %>%
  group_by(generalExp) %>% summarise(count = n()) %>%
  mutate(nullCount = sum(count)/5) # Adding nullCount column (must have at least 5 expected cases).

# => Independence is questionable, since one player has several scenes.
# => We only have 4.6 expected cases, minimum is 5.
# => Conditions for using Chi^2 test is not fulfilled.

# Doing it anyway, for the sake of the example.

# Adding z-score column.
df1 <- df1 %>%
mutate(Z_n = (count - `nullCount`) / sqrt(`nullCount`))

# Test statistic
test_statistic <- sum(df1$Z_n ^ 2)

# P-value
p_value <- 1 - pchisq(test_statistic, df = nrow(df1) - 1)
```

```r
print(p_value)
```

```
## [1] 0.0884443
```

```r
# p-value was 0.088 so HA was rejected in favor of H0:
# The distribution of answers is the same as the null-distribution.
# Or => There is not statistical significance that the distribution of answers is different from the nu
```

Goodness of fit test:

Checking if distribution of diffExp answers, in CONTROL group, follows the null-distribution or not.

```r
# New data frame with unique experience values, and their counts.
df2 <- controlGroup %>%
  group_by(diffExp) %>% summarise(count = n()) %>%
  mutate(nullCount = sum(count)/5) # Adding nullCount column (must have at least 5 expected cases).

# => Independence is questionable, since one player has several scenes.
# => We only have 4.6 expected cases, minimum is 5.
# => Conditions for using Chi^2 test is not fulfilled.

# Doing it anyway, for the sake of the example.

# Adding z-score column.
df2 <- df2 %>%
mutate(Z_n = (count - `nullCount`) / sqrt(`nullCount`))

# Test statistic
test_statistic <- sum(df2$Z_n ^ 2)

# P-value
p_value <- 1 - pchisq(test_statistic, df = nrow(df2) - 1)
print(p_value)
```

```
## [1] 0.4544488
```

```r
# p-value was 0.454 so HA was rejected in favor of H0:
# The distribution of answers is the same as the null-distribution.
# Or => There is not statistical significance that the distribution of answers is different from the nu
```

Goodness of fit test:

Checking if distribution of generalExp answers, in TEST group, follows the null-distribution or not.

```r
# New data frame with unique experience values, and their counts.
df3 <- testGroup %>%
  group_by(generalExp) %>% summarise(count = n()) %>%
  mutate(nullCount = sum(count)/5) # Adding nullCount column (must have at least 5 expected cases).

# => Independence is questionable, since one player has several scenes.
# => We have 5.4 expected cases, minimum is 5 (fulfilled).
```

```r
# => Conditions for using Chi^2 goodness of fit test might be fulfilled.

# Adding z-score column.
df3 <- df3 %>%
mutate(Z_n = (count - `nullCount`) / sqrt(`nullCount`))

# Test statistic
test_statistic <- sum(df3$Z_n ^ 2)

# P-value
p_value <- 1 - pchisq(test_statistic, df = nrow(df3) - 1)
print(p_value)
```

```
## [1] 0.03154517
```

```r
# p-value was 0.032 so H0 was rejected in favor of HA:
# The distribution of answers is NOT the same as the null-distribution.
# Or => There IS statistical significance that the distribution of answers is different from the null-d
```

Goodness of fit test:

Checking if distribution of diffExp answers, in TEST group, follows the null-distribution or not.

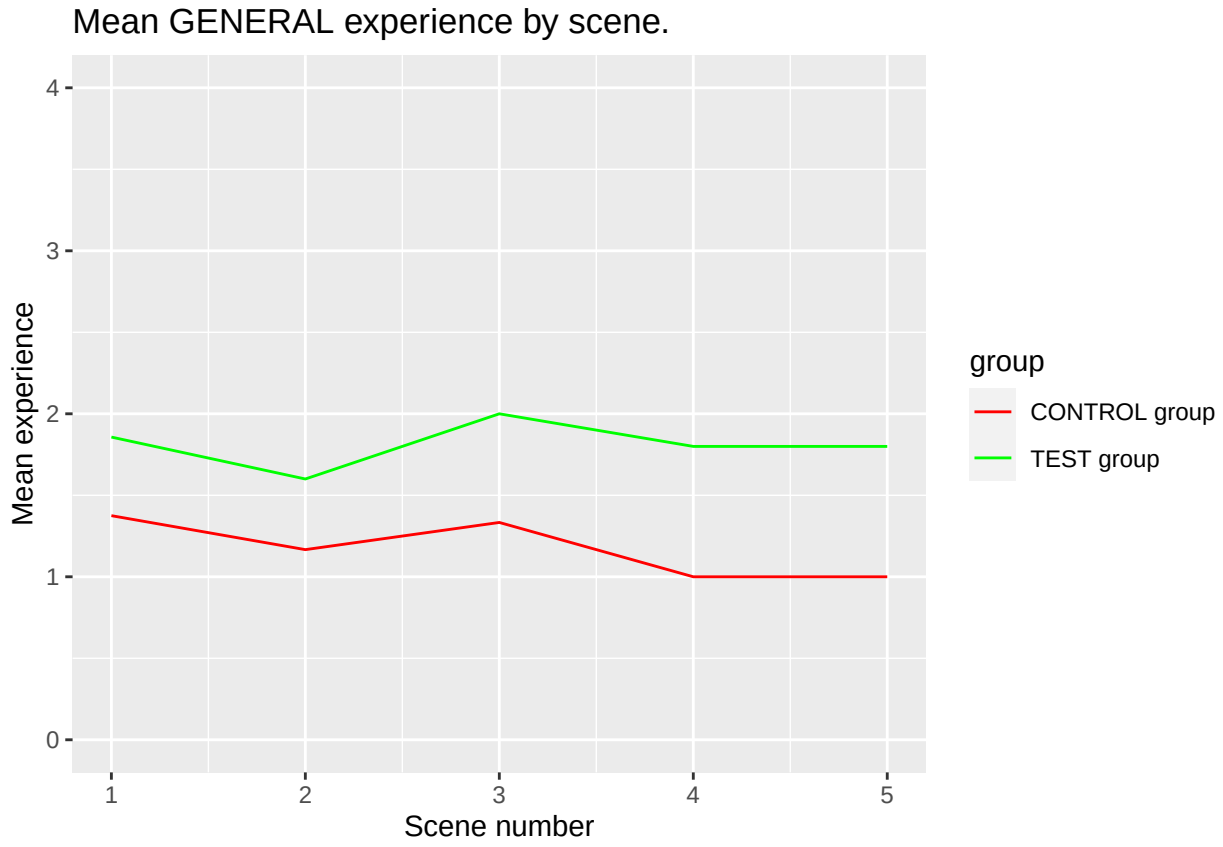
```r
# New data frame with unique experience values, and their counts.
df4 <- testGroup %>%
  group_by(diffExp) %>% summarise(count = n()) %>%
  mutate(nullCount = sum(count)/5) # Adding nullCount column (must have at least 5 expected cases).

# => Independence is questionable, since one player has several scenes.
# => We have 5.4 expected cases, minimum is 5 (fulfilled).
# => Conditions for using Chi^2 goodness of fit test might be fulfilled.

# Adding z-score column.
df4 <- df4 %>%
mutate(Z_n = (count - `nullCount`) / sqrt(`nullCount`))

# Test statistic
test_statistic <- sum(df4$Z_n ^ 2)

# P-value
p_value <- 1 - pchisq(test_statistic, df = nrow(df4) - 1)
print(p_value)
```

```
## [1] 0.04290389
```

```r
# p-value was 0.043 so H0 was rejected in favor of HA:
# The distribution of answers is NOT the same as the null-distribution.
# Or => There IS statistical significance that the distribution of answers is different from the null-d
```

Plotting means of General/Difficulty experiences per scene.

```
meansGenExpCONT <- controlGroup %>%
  group_by(sceneNumber) %>% summarise_at(vars(generalExp), list(exp_mean = mean))

meansGenExpTEST <- testGroup %>%
  group_by(sceneNumber) %>% summarise_at(vars(generalExp), list(exp_mean = mean))


dfComb <- rbind(meansGenExpCONT, meansGenExpTEST)
dfComb$group <- rep(c("CONTROL group", "TEST group"), each = 5)

# Plotting General experience means for control group and test group.
ggplot(data = dfComb, aes(x = sceneNumber, y = exp_mean, group = group, color = group)) +
  geom_line() +
  scale_y_continuous(limits = c(0, max(4))) +
  labs(title = "Mean GENERAL experience by scene.",
       x = "Scene number",
       y = "Mean experience") +
  scale_color_manual(values = c("red", "green"))
```




```
meansDiffExpCONT <- controlGroup %>%
  group_by(sceneNumber) %>% summarise_at(vars(diffExp), list(exp_mean = mean))

meansDiffExpTEST <- testGroup %>%
  group_by(sceneNumber) %>% summarise_at(vars(diffExp), list(exp_mean = mean))
```

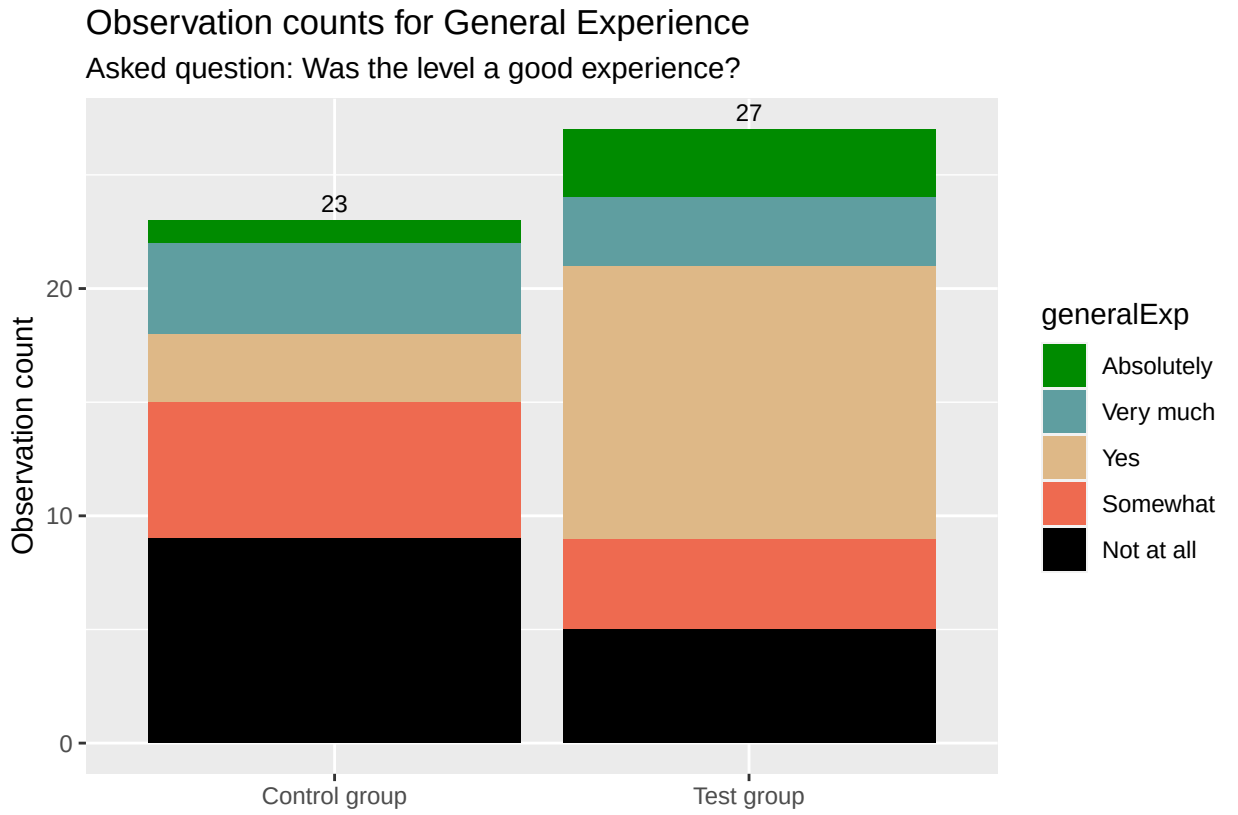
```r
dfComb <- rbind(meansDiffExpCONT, meansDiffExpTEST)
dfComb$group <- rep(c("CONTROL group",
                      "TEST group"),
                    each = 5)


# Plotting Difficulty experience means for control group and test group.
ggplot(data = dfComb, aes(x = sceneNumber, y = exp_mean, group = group, color = group)) +
  geom_line() +
  scale_y_continuous(limits = c(0, max(4))) +
  labs(title = "Mean DIFFICULTY experience by scene.",
       x = "Scene number",
       y = "Mean experience") +
  scale_color_manual(values = c("red", "green"))
```

Comparison of experiences between CONTROL group and TEST group.

```r
#Observations for general experience.
dataFrame <- sceneData
dataFrame %>%
  mutate(
    generalExp = factor(generalExp, levels = c(4, 3, 2, 1, 0)),
    isControlGroup = factor(isControlGroup, levels = c(1, 0),labels = c("Control group", "Test group"))
  ) %>%
```

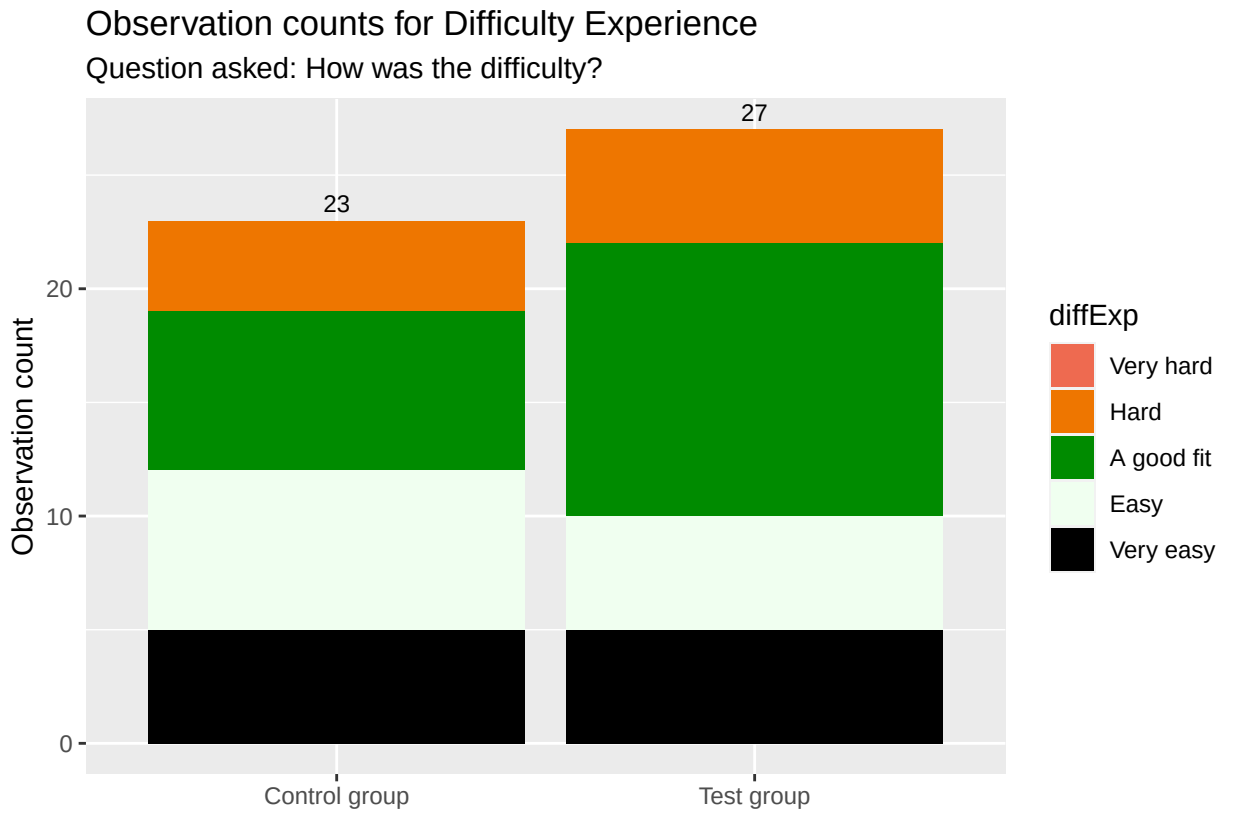
```
  ggplot() +
  geom_bar(aes(x=isControlGroup, fill=generalExp), stat = "count") +

  geom_text(aes(x = isControlGroup, y = ..count.., label = after_stat(count)),
            stat = "count", vjust = -0.5, color = "black", size = 3) +

  scale_fill_manual(values = c("#008B00", "#5F9EA0", "#DEB887", "#EE6A50", "black"),
                    labels = c("Absolutely", "Very much",  "Yes",  "Somewhat", "Not at all"),
                    drop = FALSE
                    ) +
  labs(title = "Observation counts for General Experience",
       subtitle = "Asked question: Was the level a good experience?",
       y = "Observation count", x="")
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
```




```
#Observations for difficulty experience.
dataFrame <- sceneData
dataFrame %>%
  mutate(
    diffExp = factor(diffExp, levels = c(4, 3, 2, 1, 0)),
    isControlGroup = factor(isControlGroup, levels = c(1, 0),labels = c("Control group", "Test group"))
  ) %>%
  ggplot() +
  geom_bar(aes(x=isControlGroup, fill=diffExp), stat = "count") +

  geom_text(aes(x = isControlGroup, y = ..count.., label = after_stat(count)),
```

```
            stat = "count", vjust = -0.5, color = "black", size = 3) +

  #scale_fill_manual(values = c( "black", "#EE6A50", "#008B00",  "#EE7600", "#F0FFF0")) +
  scale_fill_manual(values = c( "#EE6A50", "#EE7600", "#008B00", "#F0FFF0", "black"),
                    labels = c("Very hard", "Hard", "A good fit", "Easy", "Very easy"),
                    drop = FALSE
                    ) +
  labs(title = "Observation counts for Difficulty Experience",
       subtitle = "Question asked: How was the difficulty?",
       y = "Observation count", x="")
```



Difference of means T-test, comparing generalExp for control and test groups.

```
result <- t.test(x=controlGroup$generalExp, y=testGroup$generalExp)
print(result)
```

```
##
##  Welch Two Sample t-test
##
## data:  controlGroup$generalExp and testGroup$generalExp
## t = -1.6883, df = 45.827, p-value = 0.09814
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.3097625  0.1149155
## sample estimates:
```

```
## mean of x mean of y
##  1.217391  1.814815
```

```
# stat_function(fun = dnorm, args = list(mean=mean(dataFrame$diffExp),
#                                        sd=sd(dataFrame$diffExp)), color="blue")
```

Difference of means T-test, comparing diffExp for control and test groups.

```
result <- t.test(x=controlGroup$diffExp, y=testGroup$diffExp)
print(result)
```

```
##
##  Welch Two Sample t-test
##
## data:  controlGroup$diffExp and testGroup$diffExp
## t = -0.67148, df = 46.254, p-value = 0.5053
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.7788558  0.3891618
## sample estimates:
## mean of x mean of y
##  1.434783  1.629630
```

Chi^2 independence tests

Testing to see if there is a dependence between being in CONTROL/TEST group and reported GENERAL experience category.

```
obs <- tribble(
~group, ~generalExp0, ~generalExp1, ~generalExp2, ~generalExp3, ~generalExp4,
  "controlGroup",
  nrow(filter(sceneData, isControlGroup == 1 & generalExp == 0)),
  nrow(filter(sceneData, isControlGroup == 1 & generalExp == 1)),
  nrow(filter(sceneData, isControlGroup == 1 & generalExp == 2)),
  nrow(filter(sceneData, isControlGroup == 1 & generalExp == 3)),
  nrow(filter(sceneData, isControlGroup == 1 & generalExp == 4)),
  "testGroup",
  nrow(filter(sceneData, isControlGroup == 0 & generalExp == 0)),
  nrow(filter(sceneData, isControlGroup == 0 & generalExp == 1)),
  nrow(filter(sceneData, isControlGroup == 0 & generalExp == 2)),
  nrow(filter(sceneData, isControlGroup == 0 & generalExp == 3)),
  nrow(filter(sceneData, isControlGroup == 0 & generalExp == 4))
)

totalObs <- sum(select(obs, 2:6))
answer0 <- sum(obs$generalExp0)
answer1 <- sum(obs$generalExp1)
answer2 <- sum(obs$generalExp2)
answer3 <- sum(obs$generalExp3)
answer4 <- sum(obs$generalExp4)
```

```
controlCnt <- nrow(filter(sceneData, isControlGroup == 1))
testCnt <- nrow(filter(sceneData, isControlGroup == 0))

exp <- tribble(
~group, ~generalExp0, ~generalExp1, ~generalExp2, ~generalExp3, ~generalExp4,
  "controlGroup",
  answer0 * controlCnt / totalObs,
  answer1 * controlCnt / totalObs,
  answer2 * controlCnt / totalObs,
  answer3 * controlCnt / totalObs,
  answer4 * controlCnt / totalObs,
  "testGroup",
  answer0 * testCnt / totalObs,
  answer1 * testCnt / totalObs,
  answer2 * testCnt / totalObs,
  answer3 * testCnt / totalObs,
  answer4 * testCnt / totalObs,
)

obs %>%
select(-group) %>%
chisq.test(correct = FALSE)
```

```
## Warning in chisq.test(., correct = FALSE): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  .
## X-squared = 7.8157, df = 4, p-value = 0.09857
```

Testing to see if there is a dependence between being in CONTROL/TEST group and reported DIFFICULTY experience category.

```
obs1 <- tribble(
~group, ~diffExp0, ~diffExp1, ~diffExp2, ~diffExp3, ~diffExp4,
  "controlGroup",
  nrow(filter(sceneData, isControlGroup == 1 & diffExp == 0)),
  nrow(filter(sceneData, isControlGroup == 1 & diffExp == 1)),
  nrow(filter(sceneData, isControlGroup == 1 & diffExp == 2)),
  nrow(filter(sceneData, isControlGroup == 1 & diffExp == 3)),
  nrow(filter(sceneData, isControlGroup == 1 & diffExp == 4)),
  "testGroup",
  nrow(filter(sceneData, isControlGroup == 0 & diffExp == 0)),
  nrow(filter(sceneData, isControlGroup == 0 & diffExp == 1)),
  nrow(filter(sceneData, isControlGroup == 0 & diffExp == 2)),
  nrow(filter(sceneData, isControlGroup == 0 & diffExp == 3)),
  nrow(filter(sceneData, isControlGroup == 0 & diffExp == 4))
)

totalObs <- sum(select(obs1, 2:6))
answer0 <- sum(obs1$diffExp0)
answer1 <- sum(obs1$diffExp1)
```

```
answer2 <- sum(obs1$diffExp2)
answer3 <- sum(obs1$diffExp3)
answer4 <- sum(obs1$diffExp4)

controlCnt <- nrow(filter(sceneData, isControlGroup == 1))
testCnt <- nrow(filter(sceneData, isControlGroup == 0))

exp1 <- tribble(
~group, ~diffExp0, ~diffExp1, ~diffExp2, ~diffExp3, ~diffExp4,
  "controlGroup",
  answer0 * controlCnt / totalObs,
  answer1 * controlCnt / totalObs,
  answer2 * controlCnt / totalObs,
  answer3 * controlCnt / totalObs,
  answer4 * controlCnt / totalObs,
  "testGroup",
  answer0 * testCnt / totalObs,
  answer1 * testCnt / totalObs,
  answer2 * testCnt / totalObs,
  answer3 * testCnt / totalObs,
  answer4 * testCnt / totalObs,
)

obs1 %>%
select(-group) %>%
chisq.test(correct = FALSE)
```

```
## Warning in chisq.test(., correct = FALSE): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  .
## X-squared = NaN, df = 4, p-value = NA
```

Convoluting data, since there is not enough expected observations.

==========

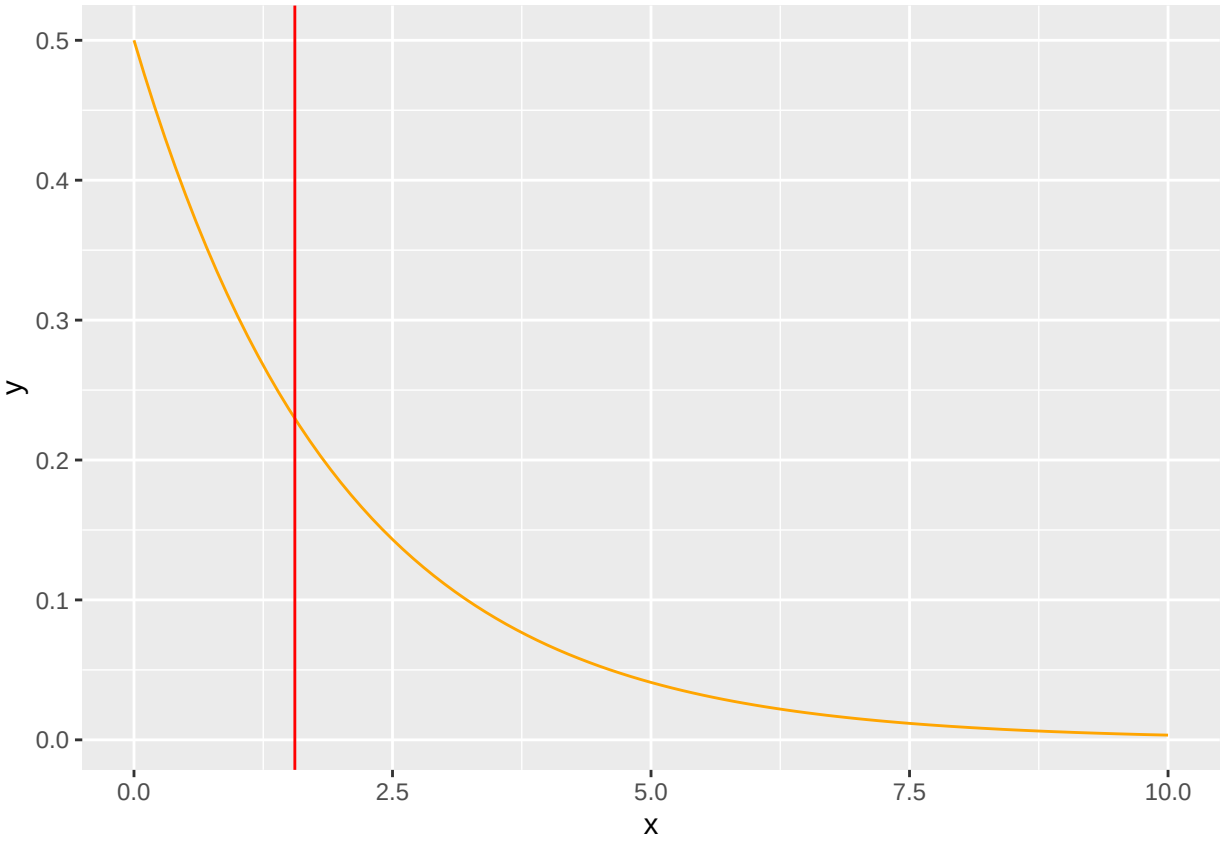
Testing to see if there is a dependence between being in CONTROL/TEST group and CONVOLUTED reported GENERAL experience category.

```
obs2 <- tribble(
~group, ~generalExp0_2, ~generalExp1_3, ~generalExp2_4,
  "controlGroup",
  nrow(filter(sceneData, isControlGroup == 1 & generalExp < 3)),
  nrow(filter(sceneData, isControlGroup == 1 & generalExp > 0 & generalExp < 5)),
  nrow(filter(sceneData, isControlGroup == 1 & generalExp > 1)),
  "testGroup",
  nrow(filter(sceneData, isControlGroup == 0 & generalExp < 3)),
  nrow(filter(sceneData, isControlGroup == 0 & generalExp > 0 & generalExp < 5)),
  nrow(filter(sceneData, isControlGroup == 0 & generalExp > 1))
```

```
)

totalObs <- sum(select(obs2, 2:4))
answer0_2 <- sum(obs2$generalExp0_2)
answer1_3 <- sum(obs2$generalExp1_3)
answer2_4 <- sum(obs2$generalExp2_4)


controlCnt <- as.numeric(obs2[1,2] + obs2[1,3] + obs2[1,4])
testCnt <- as.numeric(obs2[2,2] + obs2[2,3] + obs2[2,4])

exp2 <- tribble(
~group, ~generalExp0_2, ~generalExp1_3, ~generalExp2_4,
  "controlGroup",
  answer0_2 * controlCnt / totalObs,
  answer1_3 * controlCnt / totalObs,
  answer2_4 * controlCnt / totalObs,
  "testGroup",
  answer0_2 * testCnt / totalObs,
  answer1_3 * testCnt / totalObs,
  answer2_4 * testCnt / totalObs
)

# Finding differences
# Observed counts
obsCnt1 <-  as.array(obs2$generalExp0_2)
obsCnt2 <-  as.array(obs2$generalExp1_3)
obsCnt3 <-  as.array(obs2$generalExp2_4)
obsCounts <- c(obsCnt1, obsCnt2, obsCnt3)
# Expected counts
expCnt1 <-  as.array(exp2$generalExp0_2)
expCnt2 <-  as.array(exp2$generalExp1_3)
expCnt3 <-  as.array(exp2$generalExp2_4)
expCounts <- c(expCnt1, expCnt2, expCnt3)
# Differences
differences <- obsCounts - expCounts

# ALL THIS WAS NOT NECESSARY => COULD JUST USE THE IN BUILT METHOD
chi_squared_value <- sum(differences^2 / expCounts)
p_value <- 1 - pchisq(chi_squared_value, df = 2)
# ================================================================


obs2 %>%
select(-group) %>%
chisq.test(correct = FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  .
## X-squared = 1.5556, df = 2, p-value = 0.4594
```

```
ggplot(data.frame(x = seq(0, 10, length = 100)), aes(x = x)) +
stat_function(fun = dchisq, args = list(df = 2), color = 'orange') +
geom_vline(xintercept = chi_squared_value, color = 'red')
```



Testing to see if there is a dependence between being in CONTROL/TEST group and CONVOLUTED reported DIFFICULTY experience category.

```
obs3 <- tribble(
~group, ~diffExp0_2, ~diffExp1_3, ~diffExp2_4,
  "controlGroup",
  nrow(filter(sceneData, isControlGroup == 1 & diffExp < 3)),
  nrow(filter(sceneData, isControlGroup == 1 & diffExp > 0 & diffExp < 5)),
  nrow(filter(sceneData, isControlGroup == 1 & generalExp > 1)),
  "testGroup",
  nrow(filter(sceneData, isControlGroup == 0 & diffExp < 3)),
  nrow(filter(sceneData, isControlGroup == 0 & diffExp > 0 & diffExp < 5)),
  nrow(filter(sceneData, isControlGroup == 0 & diffExp > 1))
)



obs3 %>%
select(-group) %>%
chisq.test(correct = FALSE)
```

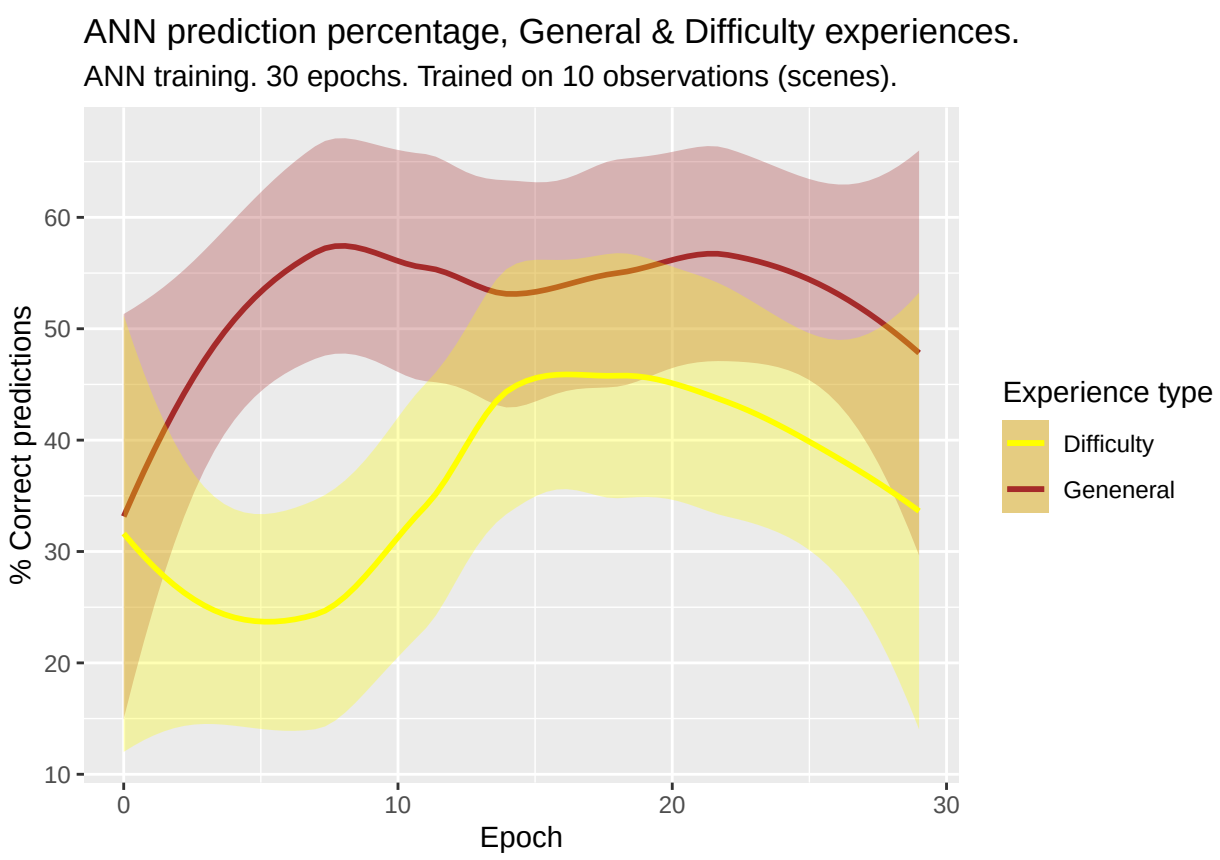
```
##
##  Pearson's Chi-squared test
##
## data:  .
## X-squared = 1.4781, df = 2, p-value = 0.4776
```

```
# ggplot(data.frame(x = seq(0, 10, length = 100)), aes(x = x)) +
# stat_function(fun = dchisq, args = list(df = 2), color = 'orange') +
# geom_vline(xintercept = chi_squared_value, color = 'red')
```

Evaluating ANN prediction for General experience and Difficulty experience over epocs of training.

```
dataFrame <- aNN_TrainingData
subT  <- sprintf("ANN training. %d epochs. Trained on %d observations (scenes).", nrow(dataFrame), as.nu
ggplot(dataFrame, aes(x=epoch)) +
  geom_smooth(aes(y = (correctGenExp/trainSize * 100), color = 'Geneneral'), fill = "brown", alpha = 0.3
  geom_smooth(aes(y = (correctDiffExp/trainSize * 100), color = 'Difficulty'), fill = "yellow", alpha =
  scale_color_manual('Experience type', values=c('yellow', 'brown')) +
  labs(title = "ANN prediction percentage, General & Difficulty experiences.",
       subtitle = subT,
       x = "Epoch", y = "% Correct predictions")
```

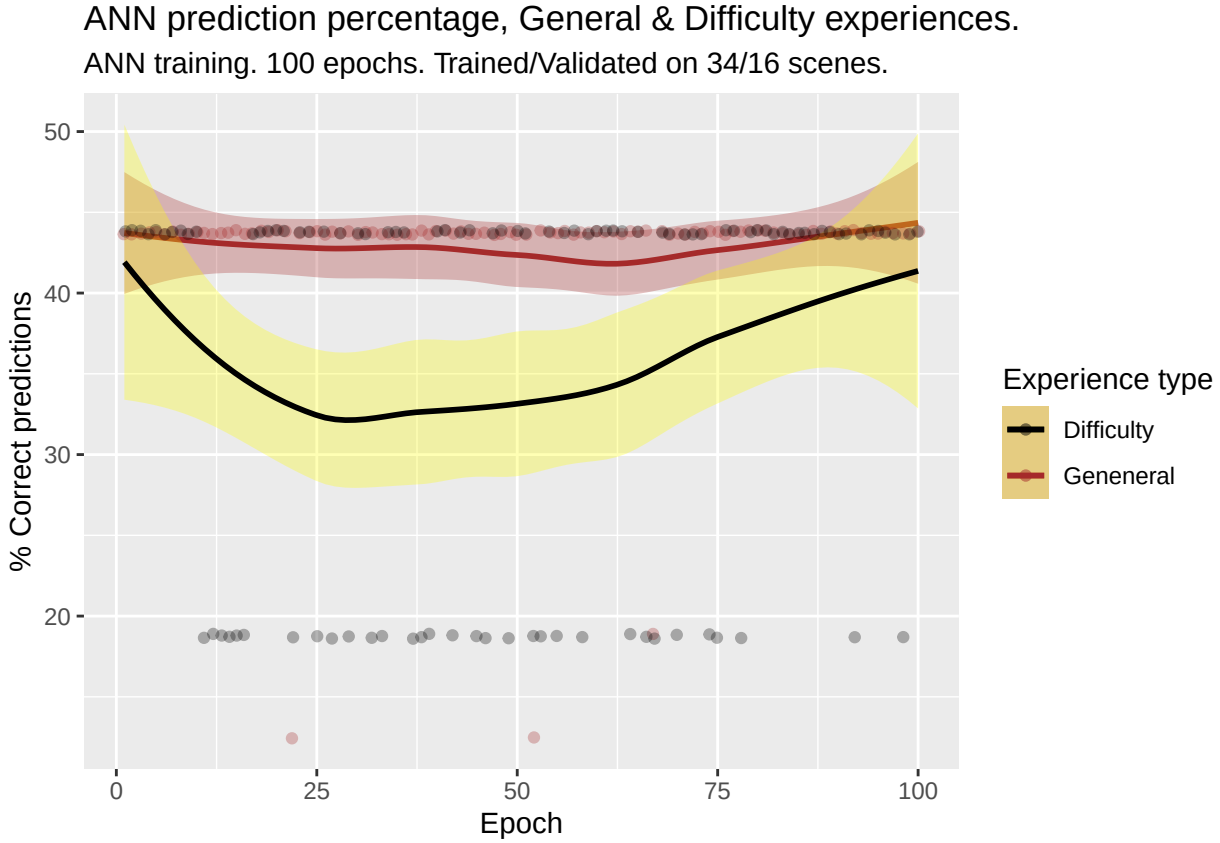
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Evaluating ANN prediction for General experience and Difficulty experience over epocs of training.

```
dataFrame <- aNN_TrainingData_VALIDATE_EVERY_EPOCH
subT  <- sprintf("ANN training. %d epochs. Trained/Validated on %d/%d scenes.", nrow(dataFrame), as.num
ggplot(dataFrame, aes(x=epoch)) +
  geom_smooth(aes(y = (correctGenExp/validateSize * 100), color = 'Geneneral'), fill = "brown", alpha =
  geom_smooth(aes(y = (correctDiffExp/validateSize * 100), color = 'Difficulty'), fill = "yellow", alpha
  geom_point(aes(y = (correctGenExp/validateSize * 100), color = 'Geneneral'), fill = "brown", alpha = 
  geom_point(aes(y = (correctDiffExp/validateSize * 100), color = 'Difficulty'), fill = "yellow", alpha
  scale_color_manual('Experience type', values=c('black', 'brown')) +
  labs(title = "ANN prediction percentage, General & Difficulty experiences.",
       subtitle = subT,
       x = "Epoch", y = "% Correct predictions")
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Statistical significance of ANN General experience prediction.

```
binom.test(7, 16, 0.2, alternative = "greater")
```

```
##
##  Exact binomial test
##
## data:  7 and 16
```

```
## number of successes = 7, number of trials = 16, p-value = 0.02666
## alternative hypothesis: true probability of success is greater than 0.2
## 95 percent confidence interval:
##  0.2266916 1.0000000
## sample estimates:
## probability of success
##                 0.4375
```

Statistical significance of ANN Difficulty experience prediction

```
binom.test(7, 16, 0.2, alternative = "greater")
```

```
##
##  Exact binomial test
##
## data:  7 and 16
## number of successes = 7, number of trials = 16, p-value = 0.02666
## alternative hypothesis: true probability of success is greater than 0.2
## 95 percent confidence interval:
##  0.2266916 1.0000000
## sample estimates:
## probability of success
##                 0.4375
```

Evaluating explicit feedback regarding adaptation

```
dataFrame <- sceneData
  dataFrame %>%
  mutate(
    otherExp = factor(otherExp, levels = c(4, 3, 2, 1, 0)),
    isControlGroup = factor(isControlGroup, levels = c(1, 0),labels = c("Control group", "Test group"))
  ) %>%
  ggplot() +
  geom_bar(aes(x=isControlGroup, fill=otherExp), stat = "count") +

  geom_text(aes(x = isControlGroup, y = after_stat(count), label = after_stat(count)),
            stat = "count", vjust = -0.5, color = "black", size = 3) +

  scale_fill_manual(values = c("#5F9EA0", "#008B00", "#DEB887", "#EE6A50", "black"),
                    labels = c("I saw no adaptation", "I saw adaptation", "None of these", "I didn't wa
                    drop = FALSE
                    ) +
  labs(title = "Observation counts for Adaptation experience categories.",
       subtitle = "Question asked: What did you notice about the game adapting?",
       y = "Observation count", x="")
```

Observation counts for Adaptation experience categories.

Question asked: What did you notice about the game adapting?