MOOD_prediction_MEM_with_time

Rajnish Kumar

2025-02-21

Using the MOOD dataset with entry time of the MOOD score available, along with type == 20 for steps count.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(Matrix)
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(lme4)
library(lme4)
library(lmerTest)
```

```
## Warning: package 'lmerTest' was built under R version 4.4.2
```

```
##
## Attaching package: 'lmerTest'
```

```
## The following object is masked from 'package:lme4':
##
##     lmer
```

```
## The following object is masked from 'package:stats':
##
##     step
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.1
```

```
library(stringr)
```

Read files

```
df_sleep <-read.csv('C:\\Users\\rajnishk\\OneDrive - Michigan Medicine\\Documents\\Student Wellness Data
df_MOOD_time_polar <- read.csv('C:\\Users/rajnishk/OneDrive - Michigan Medicine/Documents/Student Wellne
df_MOOD <- read.csv('C:\\Users\\rajnishk\\OneDrive - Michigan Medicine\\Documents\\Student Wellness Data
df_step_20 <- read.csv('C:\\Users/rajnishk/OneDrive - Michigan Medicine/Documents/Student Wellness Datas
```

```
# df_mood <- read.csv('C:/Users/rajnishk/University of Michigan Dropbox/Rajnish Kumar/K24_NHLBI_Admin S
df_mood <- read.csv('C:/Users/rajnishk/OneDrive - Michigan Medicine/Documents/Student Wellness Dataset/
df_mood$mood_score <- as.integer(df_mood$INT_SRVY_RSPNS)
```

```
## Warning: NAs introduced by coercion
```

```
df_mood <- df_mood %>% filter(!is.na(mood_score))
```

df_MOOD_time_polar$mood_score <- -as.integer(df_MOOD_time_polar$INT_SRVY_RSPNS) sum(df_MOOD_time_polar$I
== "SELECTED", na.rm = TRUE) # There are NAs in the mood_score that come from these values.

df_MOOD_time_polar <- df_MOOD_time_polar %>% filter(!is.na(mood_score))

```
# See all unique time zones in your data
tz_pattern <- "\\b[A-Z]{3,5}\\b"
df_mood$time_zone <- str_extract(df_mood$INT_SRVY_RSPNS_DT, tz_pattern)
df_mood$time_stamp <- do.call(c, lapply(1:nrow(df_mood), function(i) {
  if(is.na(df_mood$time_zone[i])) {
    as.POSIXct(df_mood$INT_SRVY_RSPNS_DT[i], format = "%Y-%m-%dT%H:%M:%S")
  } else {
    as.POSIXct(df_mood$INT_SRVY_RSPNS_DT[i], format = "%Y-%m-%d %H:%M:%S")
  }
}))
```

In this group, the entries have only 6 different time zones, in BMT there were 9 different time zones of time entries.

participants have entered their MOOD scores multiple times a day. Let us confirm it first.

```
df_mood <- df_mood %>%
  mutate(
    mood_entry_date = as_date(time_stamp),
    mood_entry_time = format(time_stamp, "%H:%M:%S")
  )
```

```
detailed_multiple_entries <- df_mood %>%
  group_by(STUDY_PRTCPT_ID, mood_entry_date) %>%
  filter(n() > 1 | n_distinct(mood_score) > 1) %>%
  arrange(STUDY_PRTCPT_ID, mood_entry_date)
print("detailed multiple entries:")
```

```
## [1] "detailed multiple entries:"
```

```
print(detailed_multiple_entries)
```

```
## # A tibble: 40 x 12
## # Groups:   STUDY_PRTCPT_ID, mood_entry_date [20]
##    INT_SRVY_RSPNS_ID STUDY_ID STUDY_PRTCPT_ID INT_SRVY_ID INT_SRVY_QSTN_ID
##                <int>    <int> <chr>                <int>            <int>
##  1            356175      401 8MRVHIRR               421              941
##  2            356176      401 8MRVHIRR               421              941
##  3            384556      401 F1MALNM8               421              941
##  4            384557      401 F1MALNM8               421              941
##  5            385336      401 F1MALNM8               421              941
##  6            385346      401 F1MALNM8               421              941
##  7            378616      401 KL8ZU5XC               421              941
##  8            378617      401 KL8ZU5XC               421              941
##  9            384535      401 KL8ZU5XC               421              941
## 10            384536      401 KL8ZU5XC               421              941
## # i 30 more rows
## # i 7 more variables: INT_SRVY_RSPNS_DT <chr>, INT_SRVY_RSPNS <chr>,
## #   mood_score <int>, time_zone <chr>, time_stamp <dttm>,
## #   mood_entry_date <date>, mood_entry_time <chr>
```

Pick only the highest entries of MOOD from participants for each day. There will be further discussion on what value to pick.

```r
particpant_entry_counts <- table(df_mood$STUDY_PRTCPT_ID)


unique_combinations <- df_mood %>%
  distinct(STUDY_PRTCPT_ID, mood_entry_date) %>%
  nrow()

print(paste("Number of unique combinations of STUDY_PRTCPT_ID and mood_entry_date:", unique_combinations
```

```
## [1] "Number of unique combinations of STUDY_PRTCPT_ID and mood_entry_date: 742"
```

```r
print(paste(unique_combinations,"out of", dim(df_mood)[1]," participant and mood score entry date are u
```

```
## [1] "742 out of 762  participant and mood score entry date are unique, and rest are extra entries wh
```

```r
# Find repeated entries
repeated_entries <- df_mood %>%
  group_by(STUDY_PRTCPT_ID, mood_entry_date) %>%
  filter(n() > 1) %>%
  ungroup()

# Count the number of repeated rows
num_repeated_rows <- nrow(repeated_entries)

# Print the number of repeated rows
cat("Number of repeated rows:", num_repeated_rows, "\n")
```

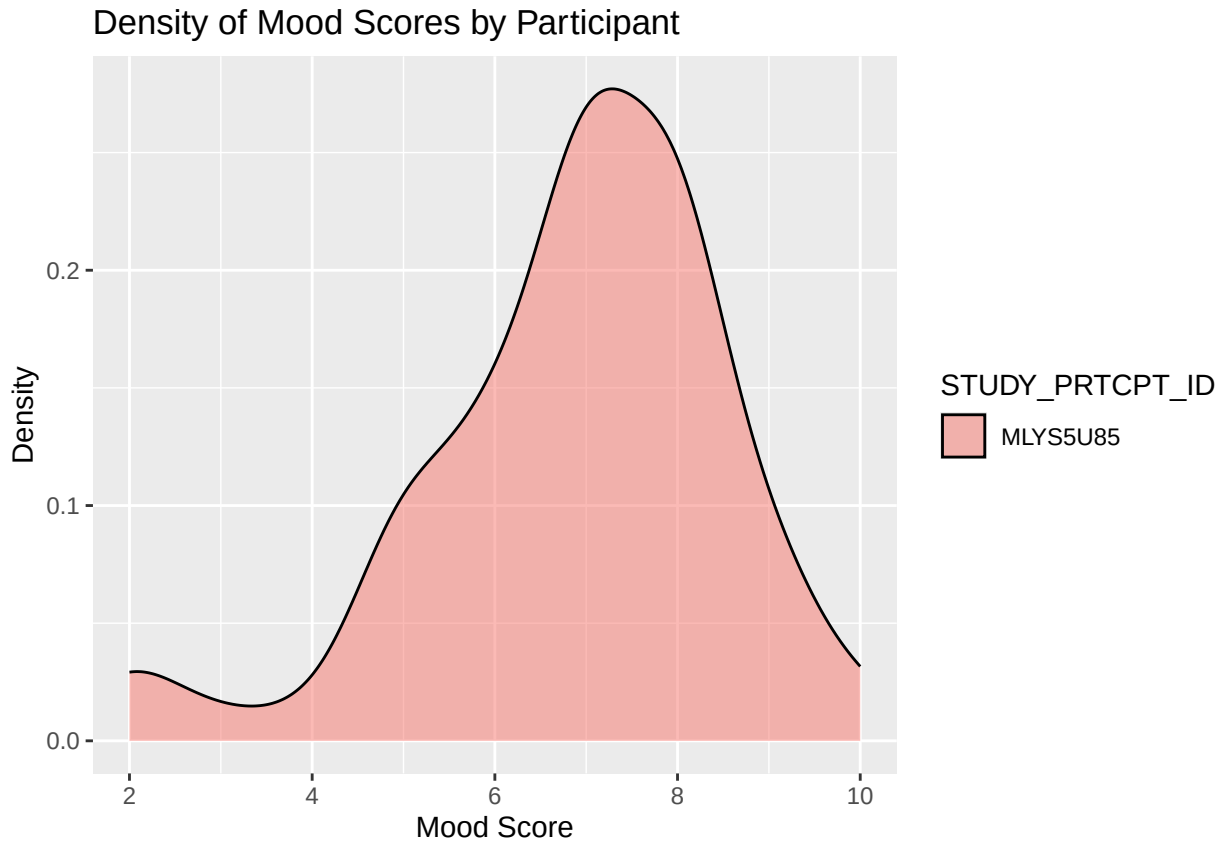
```
## Number of repeated rows: 40
```

```r
# Store repeated entries in a separate data frame
df_repeated <- repeated_entries

# Print the first few rows of the new data frame
print(head(df_repeated))
```

```
## # A tibble: 6 x 12
##   INT_SRVY_RSPNS_ID STUDY_ID STUDY_PRTCPT_ID INT_SRVY_ID INT_SRVY_QSTN_ID
##               <int>    <int> <chr>                <int>            <int>
## 1            364976      401 XGJ8AFS8               421              941
## 2            354875      401 XGJ8AFS8               421              941
## 3            384535      401 KL8ZU5XC               421              941
## 4            384536      401 KL8ZU5XC               421              941
## 5            384556      401 F1MALNM8               421              941
## 6            384557      401 F1MALNM8               421              941
## # i 7 more variables: INT_SRVY_RSPNS_DT <chr>, INT_SRVY_RSPNS <chr>,
## #   mood_score <int>, time_zone <chr>, time_stamp <dttm>,
## #   mood_entry_date <date>, mood_entry_time <chr>
```

```r
unique_combinations_multiple <- detailed_multiple_entries%>% distinct(STUDY_PRTCPT_ID,mood_entry_date)
```

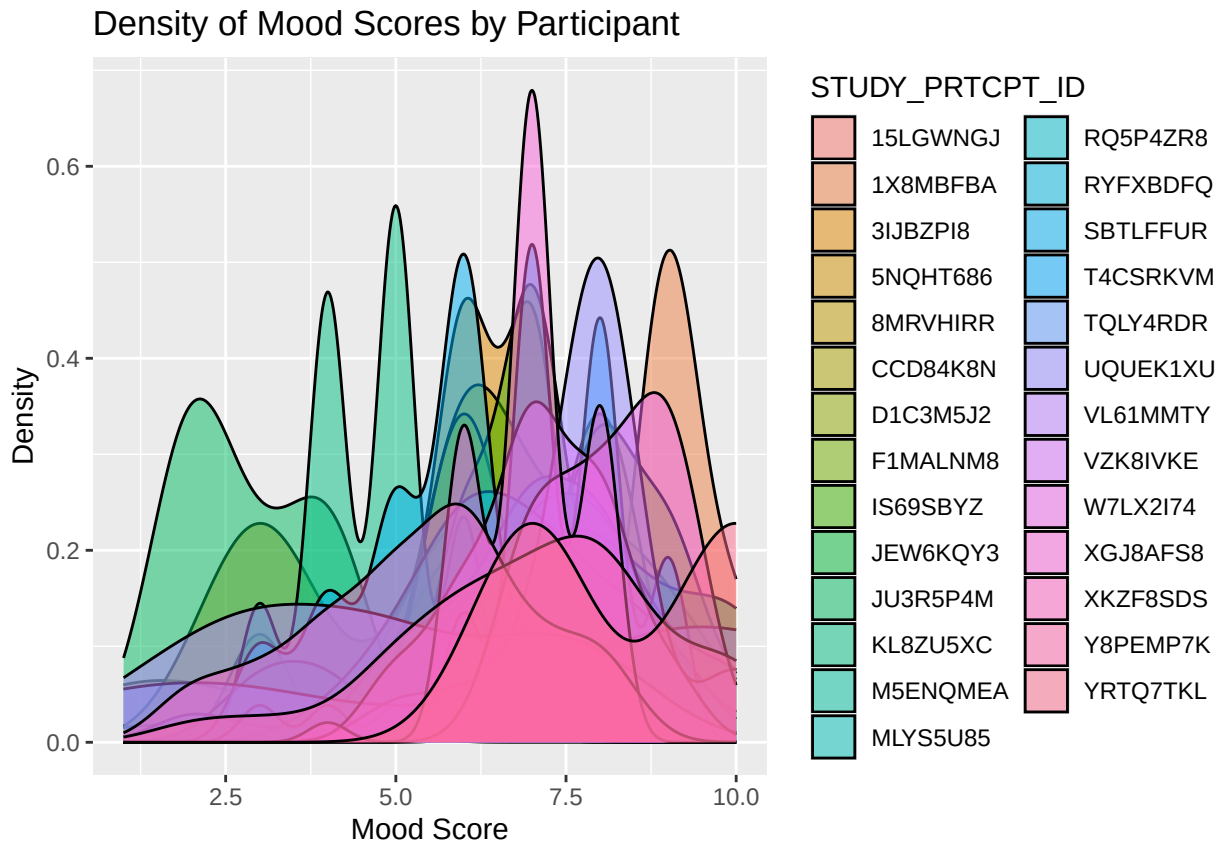
```r
ggplot(df_mood[df_mood$STUDY_PRTCPT_ID=='MLYS5U85',], aes(x = mood_score, fill = STUDY_PRTCPT_ID)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density of Mood Scores by Participant", x = "Mood Score", y = "Density")
```



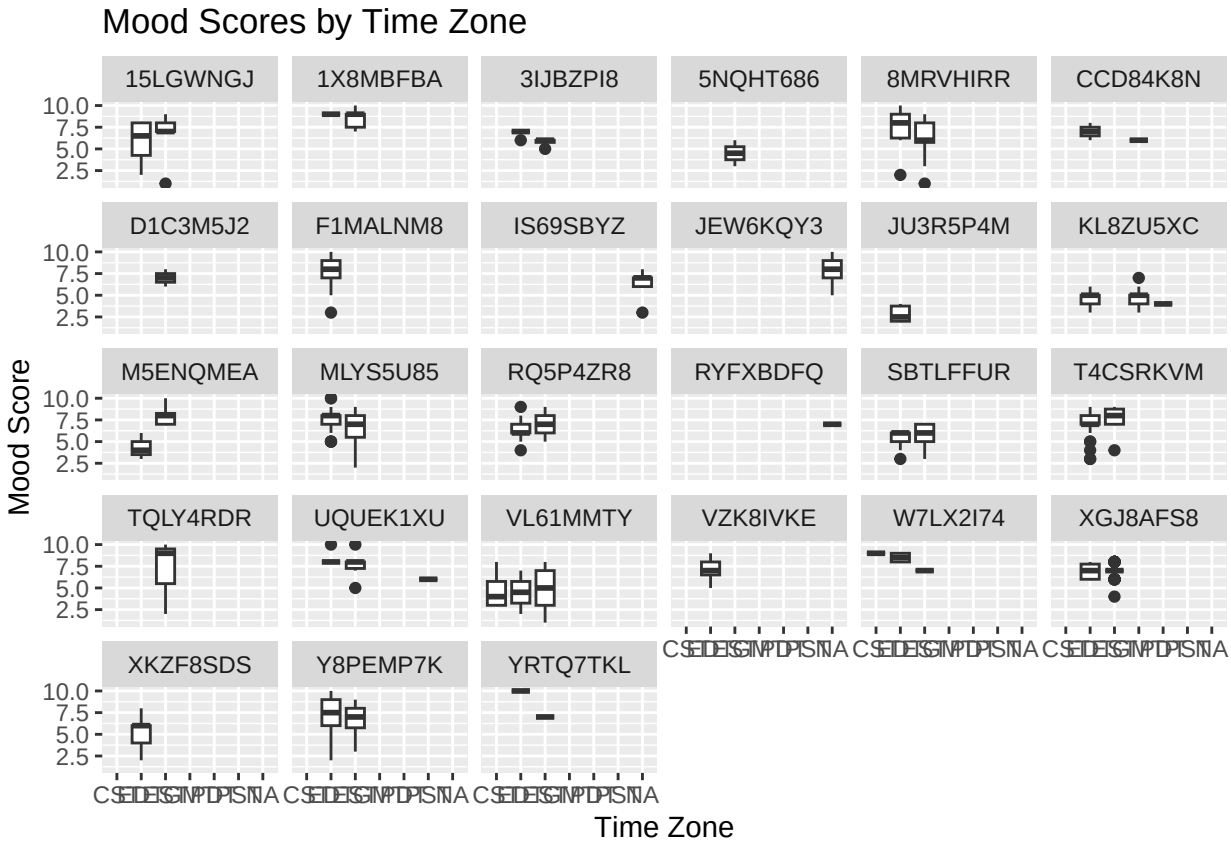

```r
ggplot(df_mood, aes(x = mood_score, fill = STUDY_PRTCPT_ID)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density of Mood Scores by Participant", x = "Mood Score", y = "Density")
```

```
## Warning: Groups with fewer than two data points have been dropped.
```

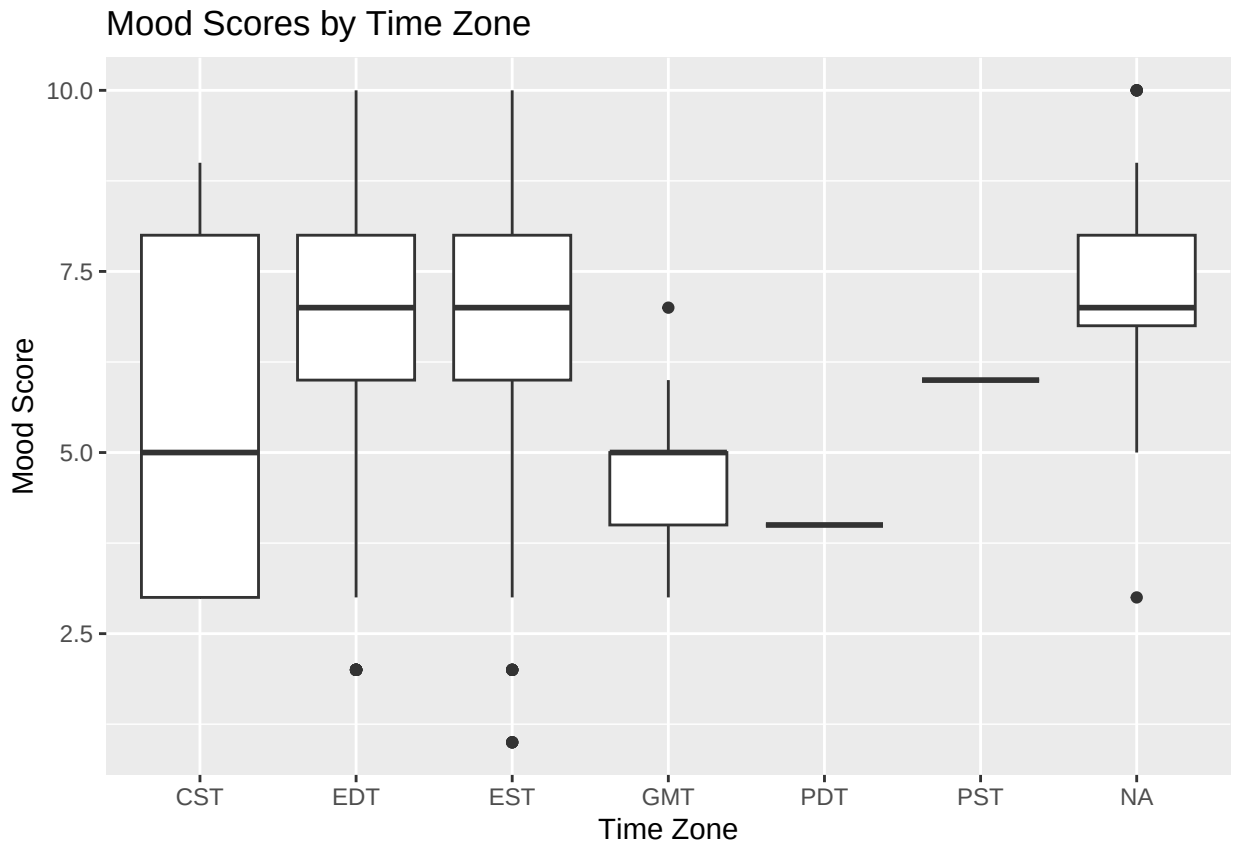
```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```




```
ggplot(df_mood, aes(x=time_zone, y=mood_score)) +
  geom_boxplot() +
  labs(title="Mood Scores by Time Zone", x="Time Zone", y="Mood Score") + facet_wrap(~STUDY_PRTCPT_ID)
```




```r
ggplot(df_mood, aes(x=time_zone, y=mood_score)) +
  geom_boxplot() +
  labs(title="Mood Scores by Time Zone", x="Time Zone", y="Mood Score")
```




```
time_zone_counts <- df_mood %>% count(time_zone)

print(time_zone_counts)
```

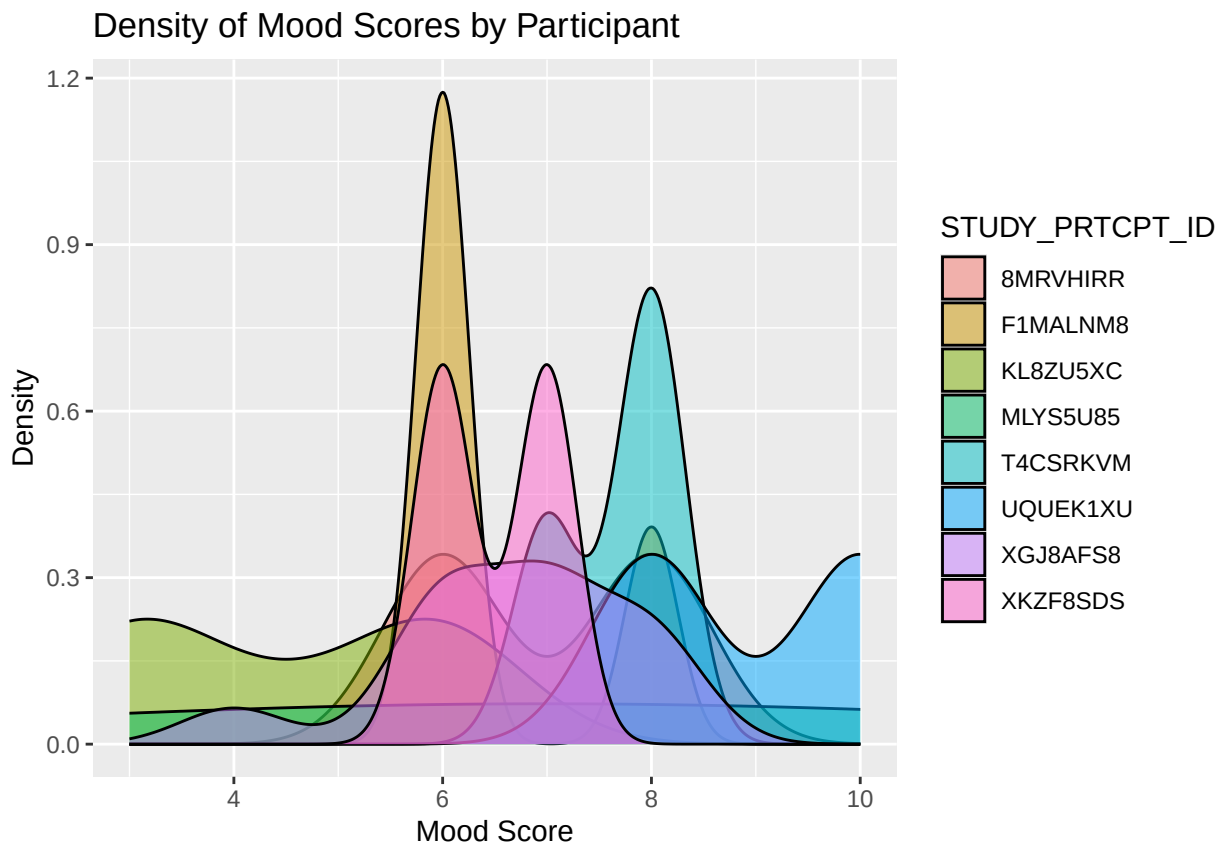
```
##   time_zone   n
## 1       CST   5
## 2       EDT 442
## 3       EST 238
## 4       GMT  47
## 5       PDT   1
## 6       PST   1
## 7      <NA>  28
```

```
df_mood$time_numeric <- as.numeric(df_mood$time_stamp)
cor.test(df_mood$time_numeric, df_mood$mood_score)
```

```
##
##  Pearson's product-moment correlation
##
## data:  df_mood$time_numeric and df_mood$mood_score
## t = -4.5704, df = 760, p-value = 5.681e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.23188100 -0.09361713
## sample estimates:
```

```
##        cor
## -0.1635521
```

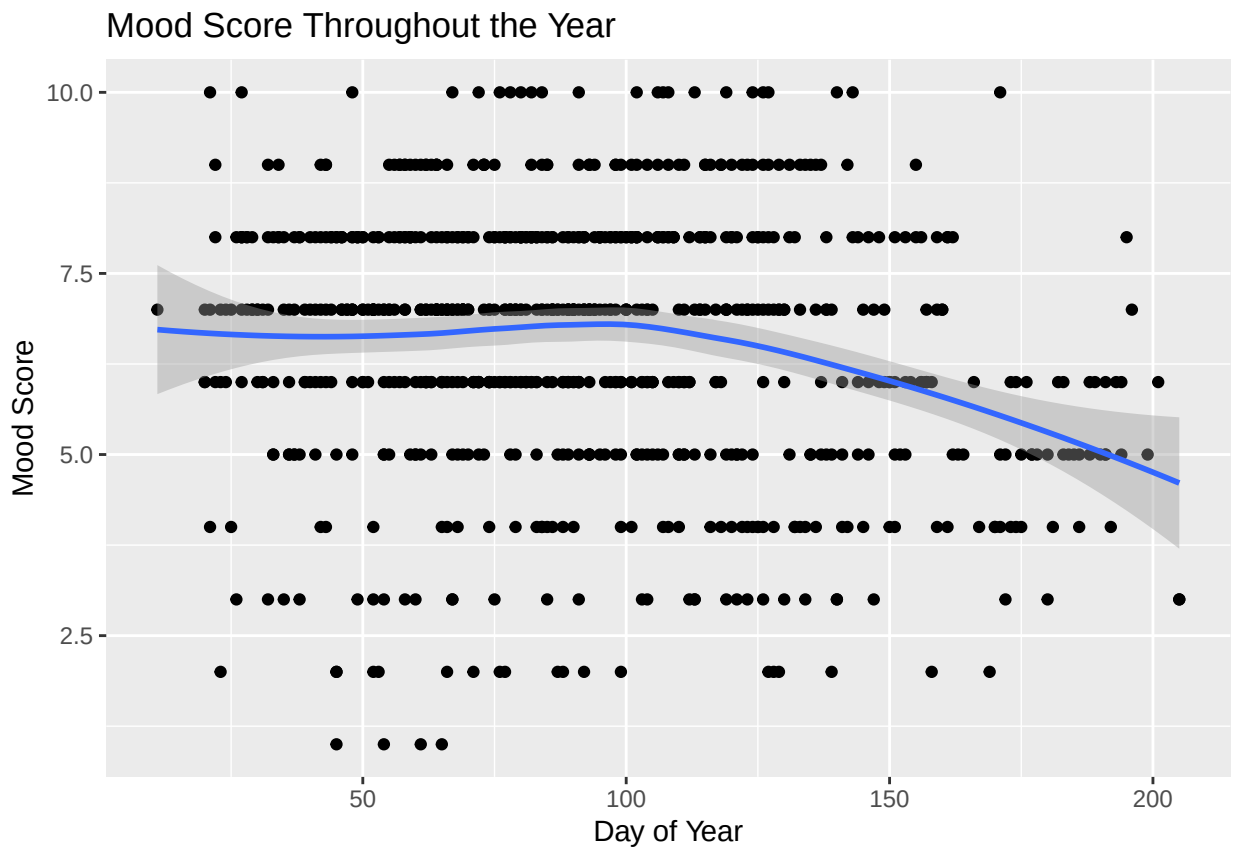
```
ggplot(detailed_multiple_entries, aes(x = mood_score, fill = STUDY_PRTCPT_ID)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density of Mood Scores by Participant", x = "Mood Score", y = "Density")
```




```
# library(lubridate)
# library(dplyr)

df_mood <- df_mood %>%
  mutate(day_of_year = yday(mood_entry_date))
```

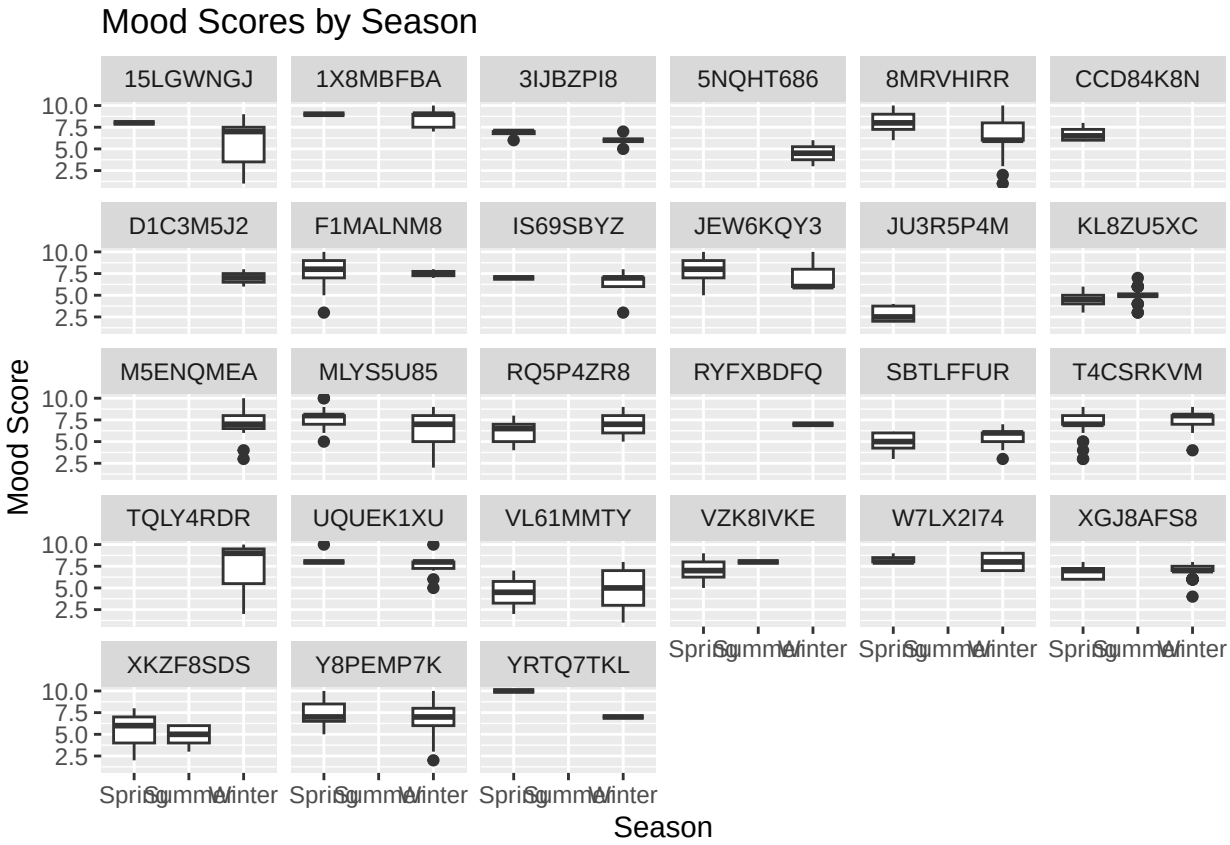
```
ggplot(df_mood, aes(x = day_of_year, y = mood_score)) +
  geom_point() +
  geom_smooth(method = "loess") +
  labs(title = "Mood Score Throughout the Year", x = "Day of Year", y = "Mood Score")
```

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



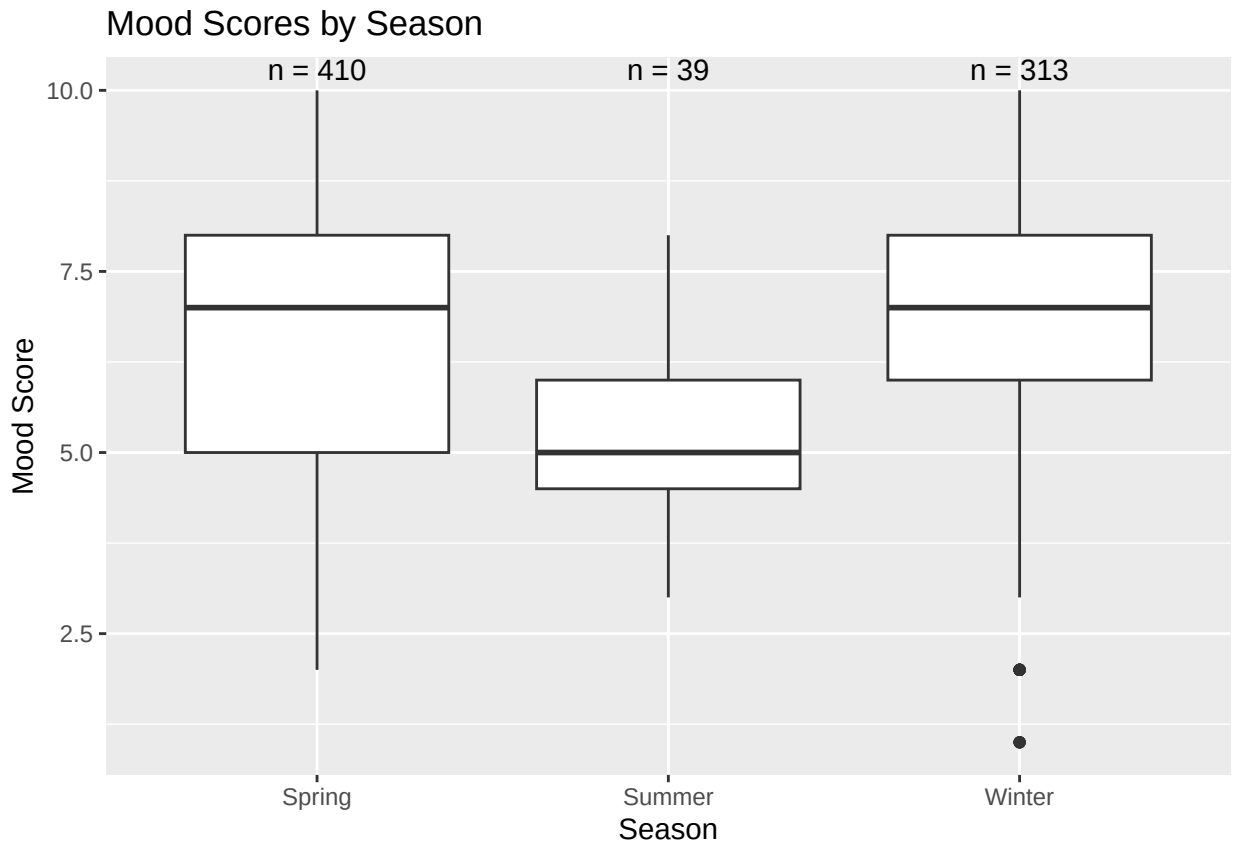

```
  # labs(title = "Mood Score Throughout the Year", x = "Day of Year", y = "Mood Score")+ facet_wrap(~ST
```

```
df_mood <- df_mood %>%
  mutate(season = case_when(
    day_of_year >= 80 & day_of_year < 172 ~ "Spring",
    day_of_year >= 172 & day_of_year < 264 ~ "Summer",
    day_of_year >= 264 & day_of_year < 355 ~ "Fall",
    TRUE ~ "Winter"
  ))

ggplot(df_mood, aes(x = season, y = mood_score)) +
  geom_boxplot() +
  labs(title = "Mood Scores by Season", x = "Season", y = "Mood Score")+ facet_wrap(~STUDY_PRTCPT_ID)
```



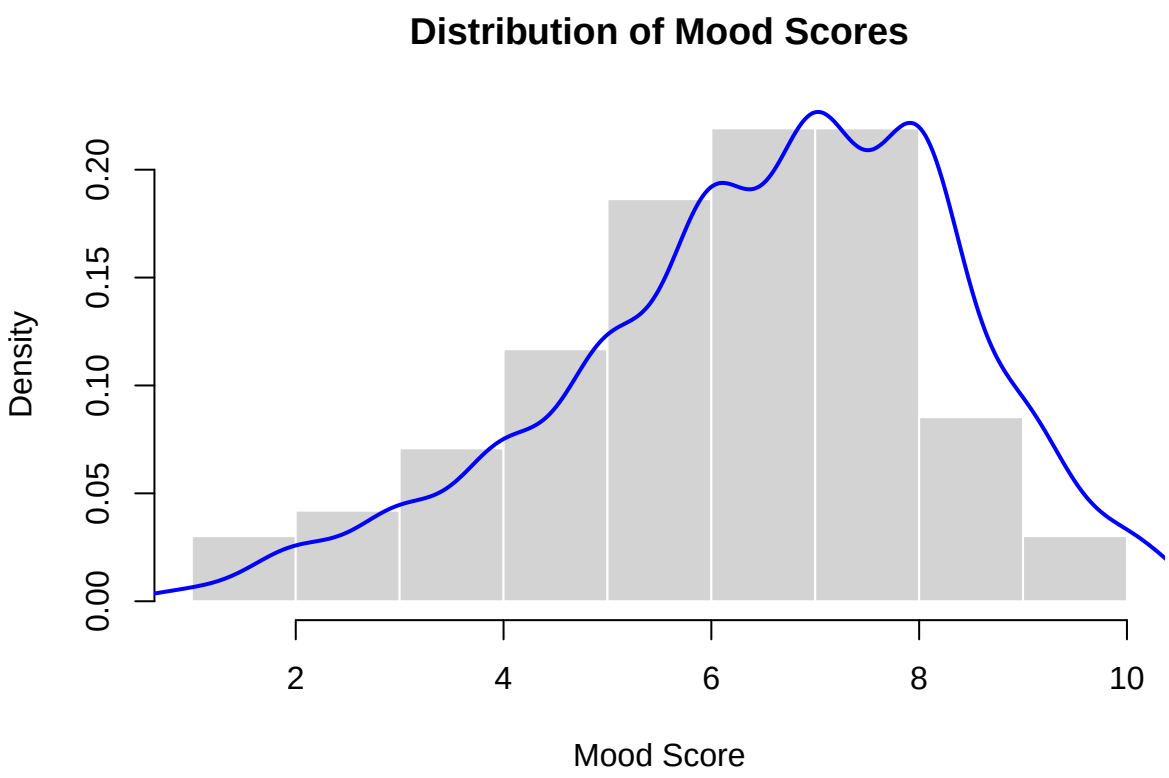

```
  # labs(title = "Mood Scores by Season", x = "Season", y = "Mood Score")
```

```
# library(ggplot2)
# library(dplyr)

# First, calculate the count for each season
season_counts <- df_mood %>%
  group_by(season) %>%
  summarise(count = n(), .groups = 'drop')

# Create the plot
ggplot(df_mood, aes(x = season, y = mood_score)) +
  geom_boxplot() +
  geom_text(data = season_counts,
            aes(y = max(df_mood$mood_score), label = paste("n =", count)),
            vjust = -0.5) +
  labs(title = "Mood Scores by Season", x = "Season", y = "Mood Score")
```




```
hist(df_mood$mood_score, main="Distribution of Mood Scores", xlab="Mood Score")
```


```r
# Create histogram
hist(df_mood$mood_score, main="Distribution of Mood Scores", xlab="Mood Score",
     probability=TRUE, col="lightgray", border="white")

# Add density plot
lines(density(df_mood$mood_score), col="blue", lwd=2)
```




```
df_device_ID <- read.csv('C:\\Users/rajnishk/OneDrive - Michigan Medicine/Documents/Student Wellness Da

df_step_20 <- df_step_20 %>%
  left_join(df_device_ID %>% select(PRTCPT_DVC_ID, PRTCPT_DVC_TYP_ID, STUDY_PRTCPT_ID),
            by = c("PRTCPT_DVC_ID", "PRTCPT_DVC_TYP_ID"))
```

```
df_step_20$STUDY_METRIC_MSR_START_DT <- as.Date(df_step_20$STUDY_METRIC_MSR_START_DT)
df_step_20$STUDY_METRIC_MSR_END_DT <- as.Date(df_step_20$STUDY_METRIC_MSR_END_DT)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
df_MOOD_time_polar$mood_score <- as.integer(df_MOOD_time_polar$INT_SRVY_RSPNS)
```

```
## Warning: NAs introduced by coercion
```

```
sum(df_MOOD_time_polar$INT_SRVY_RSPNS == "SELECTED", na.rm = TRUE) # There are NAs in the mood_score th
```

```
## [1] 119
```

```
print("The NAs in the mood_score come from sum(is.na(df_MOOD_time_polar$mood_score))")
```

```
## [1] "The NAs in the mood_score come from sum(is.na(df_MOOD_time_polar$mood_score))"
```

```r
sum(is.na(df_MOOD_time_polar$mood_score))
```

```
## [1] 119
```

```r
df_MOOD_time_polar <- df_MOOD_time_polar %>%
  filter(!is.na(mood_score))
```

```r
df_MOOD_time_polar$Date <- as.Date(df_MOOD_time_polar$Date, format = "%Y-%m-%d")
df_sleep$SLEEP_DATE <- as.Date(df_sleep$SLEEP_DATE, format = "%Y-%m-%d")
```

prediction of MOOD using mixed effect model as well as a regression model, just from time of entry.

```r
library(lme4)
library(lmerTest)  # For p-values in the output
mixed_model <- lmer(mood_score ~ time_normalized_SIN + time_normalized_COS + (1|STUDY_PRTCPT_ID), data =

print("model summary from summary(mixed_model)")
```

```
## [1] "model summary from summary(mixed_model)"
```

```r
summary(mixed_model)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: mood_score ~ time_normalized_SIN + time_normalized_COS + (1 |
##     STUDY_PRTCPT_ID)
##    Data: df_MOOD_time_polar
##
## REML criterion at convergence: 2693.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.8757 -0.5834  0.0791  0.6596  2.1546
##
## Random effects:
##  Groups          Name        Variance Std.Dev.
##  STUDY_PRTCPT_ID (Intercept) 1.424    1.193
##  Residual                    2.312    1.520
## Number of obs: 715, groups:  STUDY_PRTCPT_ID, 27
##
## Fixed effects:
##                     Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)           6.5602     0.2618  28.4557  25.057   <2e-16 ***
## time_normalized_SIN  -0.1774     0.1549 705.0097  -1.146   0.2524
## time_normalized_COS  -0.1910     0.1088 705.3057  -1.756   0.0796 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) t__SIN
## tm_nrml_SIN 0.264
## tm_nrml_COS 0.009  0.348
```

```r
print("more details from anova(mixed_model)")
```

```
## [1] "more details from anova(mixed_model)"
```

```r
anova(mixed_model)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##                     Sum Sq Mean Sq NumDF  DenDF F value  Pr(>F)
## time_normalized_SIN 3.0339  3.0339     1 705.01  1.3123 0.25236
## time_normalized_COS 7.1260  7.1260     1 705.31  3.0824 0.07958 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
print("confidence intervals from confint(mixed_model).....")
```

```
## [1] "confidence intervals from confint(mixed_model)....."
```

```r
confint(mixed_model)
```

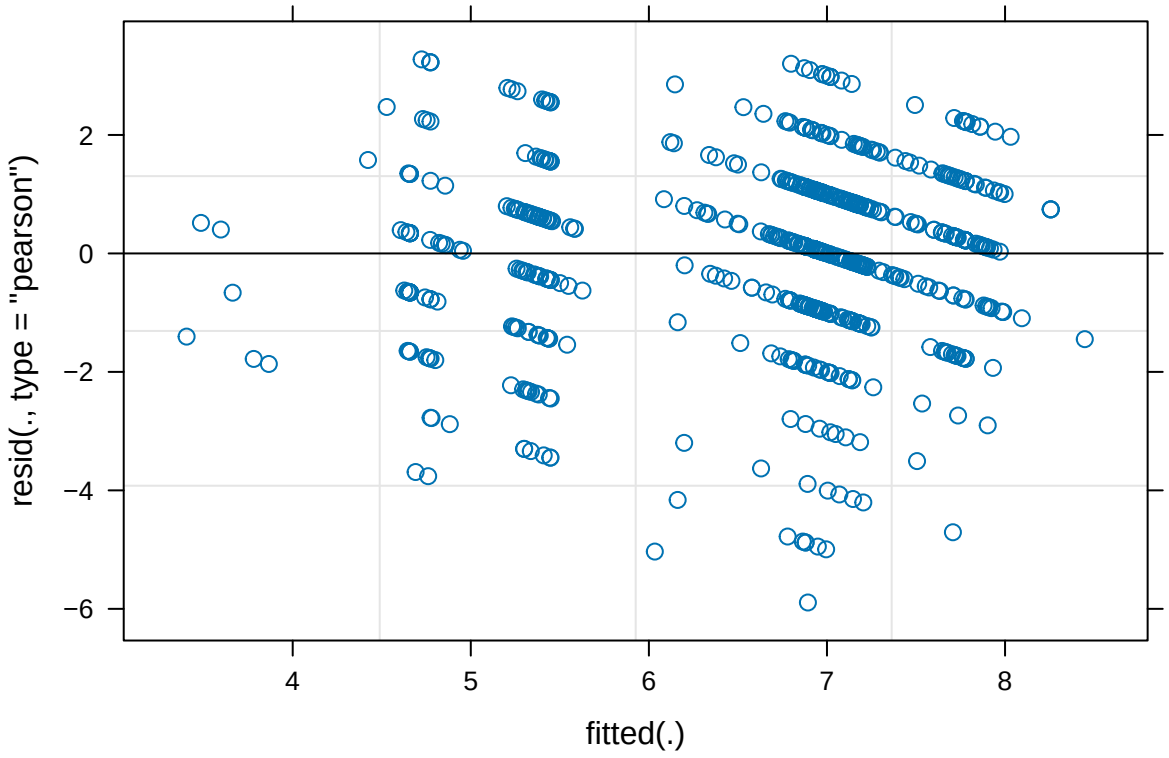
```
## Computing profile confidence intervals ...
```

```
##                          2.5 %     97.5 %
## .sig01               0.8560331 1.63479086
## .sigma               1.4415282 1.60223707
## (Intercept)          6.0395773 7.07987384
## time_normalized_SIN -0.4802749 0.12709408
## time_normalized_COS -0.4046346 0.02179465
```

```r
print("more diagnostics from plot(mixed_model).......")
```

```
## [1] "more diagnostics from plot(mixed_model)......."
```

```r
plot(mixed_model)
```




```
# print("random effects from ranef(mixed_model)...")
# ranef(mixed_model)
```

```
mixed_model_interaction <- lmer(mood_score ~ time_normalized_SIN * time_normalized_COS + (1|STUDY_PRTCP
                                data = df_MOOD_time_polar)
print("model summary from summary(mixed_model_interaction)")
```

```
## [1] "model summary from summary(mixed_model_interaction)"
```

```
summary(mixed_model_interaction)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: mood_score ~ time_normalized_SIN * time_normalized_COS + (1 |
##     STUDY_PRTCPT_ID)
##    Data: df_MOOD_time_polar
##
## REML criterion at convergence: 2694.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.8602 -0.5740  0.0833  0.6488  2.1911
##
## Random effects:
```

```
##  Groups          Name        Variance Std.Dev.
##  STUDY_PRTCPT_ID (Intercept) 1.418    1.191
##  Residual                    2.314    1.521
## Number of obs: 715, groups:  STUDY_PRTCPT_ID, 27
##
## Fixed effects:
##                                       Estimate Std. Error       df t value
## (Intercept)                             6.5805     0.2637  29.4946  24.958
## time_normalized_SIN                    -0.1834     0.1553 703.4682  -1.181
## time_normalized_COS                    -0.1593     0.1217 704.4242  -1.310
## time_normalized_SIN:time_normalized_COS  0.1451     0.2488 702.6294   0.583
##                                       Pr(>|t|)
## (Intercept)                            <2e-16 ***
## time_normalized_SIN                     0.238
## time_normalized_COS                     0.191
## time_normalized_SIN:time_normalized_COS   0.560
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) tm__SIN t__COS
## tm_nrml_SIN  0.253
## tm_nrml_COS  0.067  0.280
## t__SIN:__CO  0.132 -0.067   0.447
```

```
print("more details from anova(mixed_model_interaction)")
```

```
## [1] "more details from anova(mixed_model_interaction)"
```

```
anova(mixed_model_interaction)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##                                       Sum Sq Mean Sq NumDF  DenDF F value
## time_normalized_SIN                   3.2272  3.2272     1 703.47  1.3944
## time_normalized_COS                   3.9700  3.9700     1 704.42  1.7154
## time_normalized_SIN:time_normalized_COS 0.7870  0.7870     1 702.63  0.3400
##                                       Pr(>F)
## time_normalized_SIN                   0.2381
## time_normalized_COS                   0.1907
## time_normalized_SIN:time_normalized_COS 0.5600
```

```
print("confidence intervals from confint(mixed_model_interaction).....")
```

```
## [1] "confidence intervals from confint(mixed_model_interaction)....."
```

```
confint(mixed_model_interaction)
```

```
## Computing profile confidence intervals ...
```

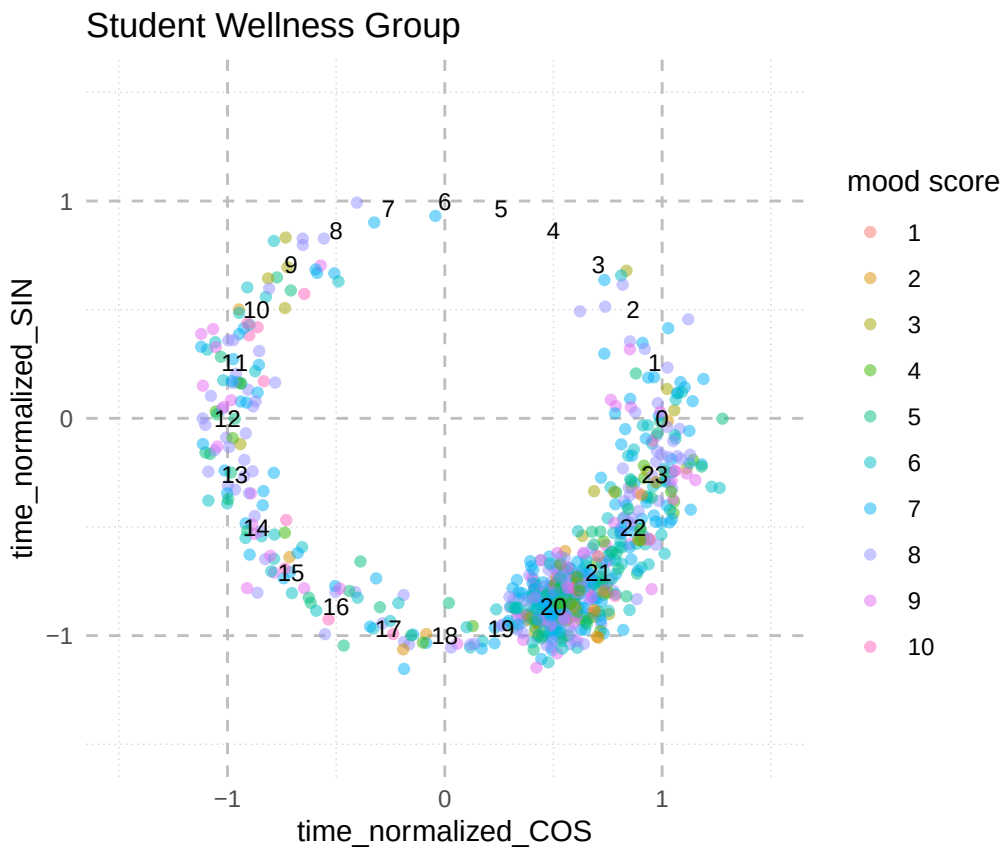
```
##                                           2.5 %     97.5 %
## .sig01                                0.8534160 1.63103389
```

```
## .sigma                                    1.4412880 1.60197304
## (Intercept)                               6.0564745 7.10298119
## time_normalized_SIN                      -0.4869133 0.12165074
## time_normalized_COS                      -0.3977739 0.07865082
## time_normalized_SIN:time_normalized_COS -0.3412954 0.63334489
```

```r
set.seed(123)  # for reproducibility

df_MOOD_time_polar <- df_MOOD_time_polar %>%
  mutate(
    time_normalized_SIN_jitter = time_normalized_SIN + 0.1 * rnorm(n()),
    time_normalized_COS_jitter = time_normalized_COS + 0.1 * rnorm(n())
  )
# ggplot(df_MOOD_time_polar, aes(x = time_normalized_COS_jitter,
#                                y = time_normalized_SIN_jitter,
#                                color = factor(mood_score))) +
#   geom_point() +
#   geom_text(aes(label = 1:nrow(df_MOOD_time_polar)), hjust = 1.5, vjust = 1.5) +
#   scale_color_discrete(name = "self reported mood score") +
#   labs(title = "Caregiver (n=5/year)",
#        x = "time_normalized_COS",
#        y = "time_normalized_SIN") +
#   theme_minimal() +
#   coord_fixed(ratio = 1) +
#   xlim(-1.5, 1.5) +
#   ylim(-1.5, 1.5)
```

```r
# Create a dataframe for hour labels
hour_labels <- data.frame(
  hour = 0:23,
  x = cos(2 * pi * (0:23) / 24),
  y = sin(2 * pi * (0:23) / 24)
)

ggplot(df_MOOD_time_polar, aes(x = time_normalized_COS_jitter,
                               y = time_normalized_SIN_jitter,
                               color = factor(mood_score))) +
  geom_point(alpha = 0.5) +  # Added alpha for transparency
  scale_color_discrete(name = "mood score") +
  geom_text(data = hour_labels, aes(x = x, y = y, label = hour),
            color = "black", size = 3, inherit.aes = FALSE) +
  labs(title = "Student Wellness Group",
       x = "time_normalized_COS",
       y = "time_normalized_SIN") +
  theme_minimal() +
  coord_fixed(ratio = 1) +
  xlim(-1.5, 1.5) +
  ylim(-1.5, 1.5) +
  theme(panel.grid.major = element_line(color = "gray", linetype = "dashed"),
        panel.grid.minor = element_line(color = "lightgray", linetype = "dotted"))
```

```
## Warning: Removed 47 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
        plot.title = element_text(hjust = 0.5)  # This centers the title
```

```
df_mood_step_sleep_SWG <- df_MOOD_time_polar %>%
  inner_join(df_sleep, by = c("STUDY_PRTCPT_ID", "Date" = "SLEEP_DATE")) %>%
  inner_join(df_step_20, by = c("STUDY_PRTCPT_ID", "Date" = "STUDY_METRIC_MSR_START_DT"))
```

```
## Warning in inner_join(., df_sleep, by = c("STUDY_PRTCPT_ID", Date = "SLEEP_DATE")): Detected an unexp
## i Row 8 of `x` matches multiple rows in `y`.
## i Row 193 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## Warning in inner_join(., df_step_20, by = c("STUDY_PRTCPT_ID", Date = "STUDY_METRIC_MSR_START_DT")):
## i Row 129 of `x` matches multiple rows in `y`.
## i Row 1577 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Checking the duplicates in the mood entry data set.
duplicate_check <- df_MOOD_time_polar %>%
  group_by(STUDY_PRTCPT_ID, Date) %>%
  summarise(count = n(), .groups = 'drop') %>%
  filter(count > 1)
```

```r
# View the result
print(duplicate_check)
```

```
## # A tibble: 17 x 3
##    STUDY_PRTCPT_ID Date       count
##    <chr>           <date>     <int>
##  1 8MRVHIRR        2023-01-27     2
##  2 F1MALNM8        2023-05-30     2
##  3 F1MALNM8        2023-06-05     2
##  4 KL8ZU5XC        2023-04-23     2
##  5 KL8ZU5XC        NA            46
##  6 MLYS5U85        2023-03-10     2
##  7 T4CSRKVM        2023-03-18     2
##  8 T4CSRKVM        2023-04-24     2
##  9 T4CSRKVM        2023-04-30     2
## 10 UQUEK1XU        2023-01-27     2
## 11 XGJ8AFS8        2023-01-20     2
## 12 XGJ8AFS8        2023-01-21     2
## 13 XGJ8AFS8        2023-02-05     2
## 14 XGJ8AFS8        2023-02-07     2
## 15 XGJ8AFS8        2023-02-27     2
## 16 XGJ8AFS8        2023-03-03     2
## 17 XKZF8SDS        2023-03-30     2
```

```r
# Get the total number of duplicates
total_duplicates <- nrow(duplicate_check)
print(paste("Total number of STUDY_PRTCPT_ID and Date combinations with duplicates:", total_duplicates))
```

```
## [1] "Total number of STUDY_PRTCPT_ID and Date combinations with duplicates: 17"
```