# Assignment 1 DMT

*Hans, Aditya*

*April 6, 2018*

## Load the data.

Here, we load the data in. We remove the first column and sort the , per ID, variable and date.

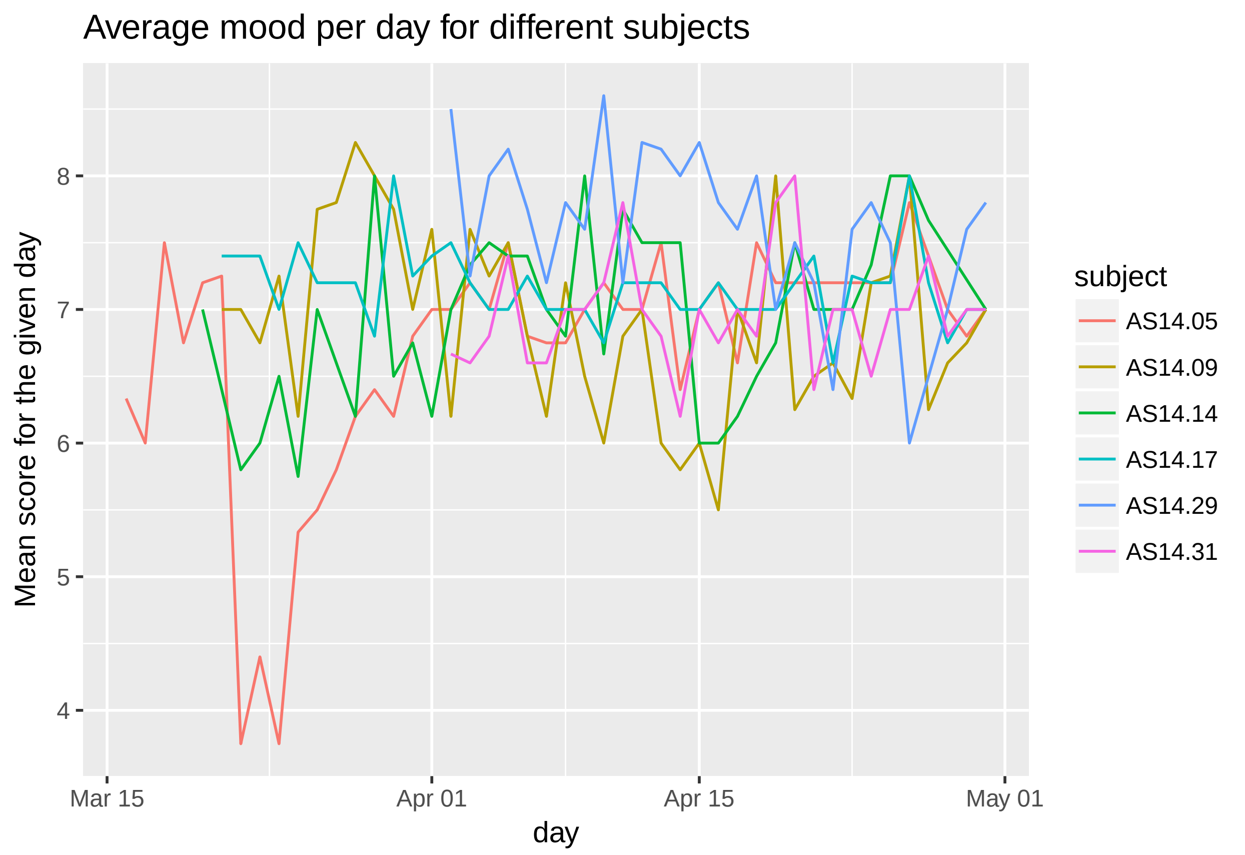
```
# Load the data. Keep value as character for now as the format is quite screwed up. Also, remove the fi
df <- read_csv("dataset_mood_smartphone.csv", col_types = cols(time = col_datetime(), value=col_characte
  mutate(time = as.POSIXct(time), day = as.Date(time), subject = as.factor(id)) %>%
  arrange(variable, subject, time)
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
df$X1 <- NULL
df$id <- NULL

# Derive the average mood per day per person. This will be the basis for our derived dataframe
summary_df <- df %>% filter(variable == "mood") %>% group_by(subject, day) %>% summarize(meanMood = mear
```

Let us first plot the mean mood per day for the subjects.

```
# Plot is hard to read for all subjects. Let us plot it for a subset of the entire period and not all s
set.seed(1410)
nsubs <- 6
random_subset <- sapply(sample(1:32, nsubs), function (number) ifelse(number<10,paste0("AS14.0",number)

summary_df %>% filter(subject %in% random_subset) %>% group_by(subject, day) %>% filter(row_number() ==
  subset(day> "2014-03-15" & day < "2014-05-01") %>%
  ggplot(aes(x=day, y=meanMood, color = subject)) +
  geom_line() +
  ggtitle("Average mood per day for different subjects") +
  ylab("Mean score for the given day")
```



This plot shows that the trajectories of the means scores are not really smooth. Also, what is interesting is that for example subject 5 shows that a bad mood can appear out of nowhere and persist throughout the the next days. Therefore, it may be a good idea to include at least 1-day lagged features including the 1-day lagged mood in addition to features aggregated over some time window.

## Summarization of features per day.

Here, we aggregate some statistics per day. Note we cannot use these as features as they depend on the day themselves but it is a neat intermediate step for feature construction in a next step.

```
# Add a Boolean to indicate whether a day is in the weekend or not.
summary_df <- summary_df %>% mutate(weekend = ifelse(as.POSIXlt(day)$wday==0 | as.POSIXlt(day)$wday==6,

# Total screen time per day per person.
summary_df <- df %>% filter(variable == "screen") %>% group_by(day, subject) %>% summarize(total_screen_
  merge(summary_df, all.y = TRUE)

# Features based on total app use per category during day. This yields many missing values as for some
# (we checked this, not an artefact on our part).
summary_df <- df %>% filter(variable %in% c("appCat.communication","appCat.game","appCat.social","call"
  summarize(total = sum(as.numeric(value))) %>% mutate(variable = paste0("total_",variable)) %>%
  spread(variable, total) %>%
  mutate(total_call = ifelse(is.na(total_call), 0, total_call), total_sms = ifelse(is.na(total_sms), 0,
  merge(summary_df, all.y = TRUE, by=c("day","subject"))

# Total smartphone app usage
```

```r
allAppCats <- unique(df$variable)[grep("appCat",unique(df$variable))]
summary_df <- df %>% filter(variable %in% allAppCats) %>% group_by(subject,day) %>%
  summarize(total_app_usage = sum(as.numeric(value))) %>% merge(summary_df, all.y = TRUE)

# Feature based off-time during the night. Night use is difficult as we have missing observations for s
morning_use <- df %>% filter(as.POSIXlt(df$time)$hour < 9) %>%
  filter(variable == "activity") %>% arrange(subject, time) %>%
  group_by(subject,day) %>% mutate(missing_obs = 9 - n()) %>%
  filter(value == 0) %>% mutate(total_0_obs = n()) %>% ungroup() %>%
  group_by(subject,day) %>% filter(row_number() == 1) %>%
  mutate(total_off_hours = total_0_obs + missing_obs) %>%
  subset(select=c("subject","day","total_off_hours"))

night_use <- df %>% filter(as.POSIXlt(df$time)$hour >= 21) %>%
  filter(variable == "activity") %>% arrange(subject, time) %>%
  group_by(subject,day) %>% mutate(missing_obs = 3-n()) %>%
  filter(value == 0) %>% mutate(total_0_obs = n()) %>% ungroup() %>%
  group_by(day) %>% filter(row_number() == 1) %>%
  mutate(total_off_hours = total_0_obs + missing_obs) %>%
  subset(select=c("subject","day","total_off_hours"))

morning_use$night_off_time <- morning_use$total_off_hours + lag(night_use$total_off_hours)
```

```
## Warning in morning_use$total_off_hours + lag(night_use$total_off_hours):
## longer object length is not a multiple of shorter object length
```

```r
summary_df <- morning_use %>% subset(select=c("subject","day","night_off_time")) %>% merge(summary_df, a
```

## Other ideas for features:

- Maybe last recorded mood of this person night before.
- Something with the arousal and valence scores. Not sure what these mean but apparently its some neuropsychological measure.
- Is time slept during the day not allowed? I guess it would be a nice feature.

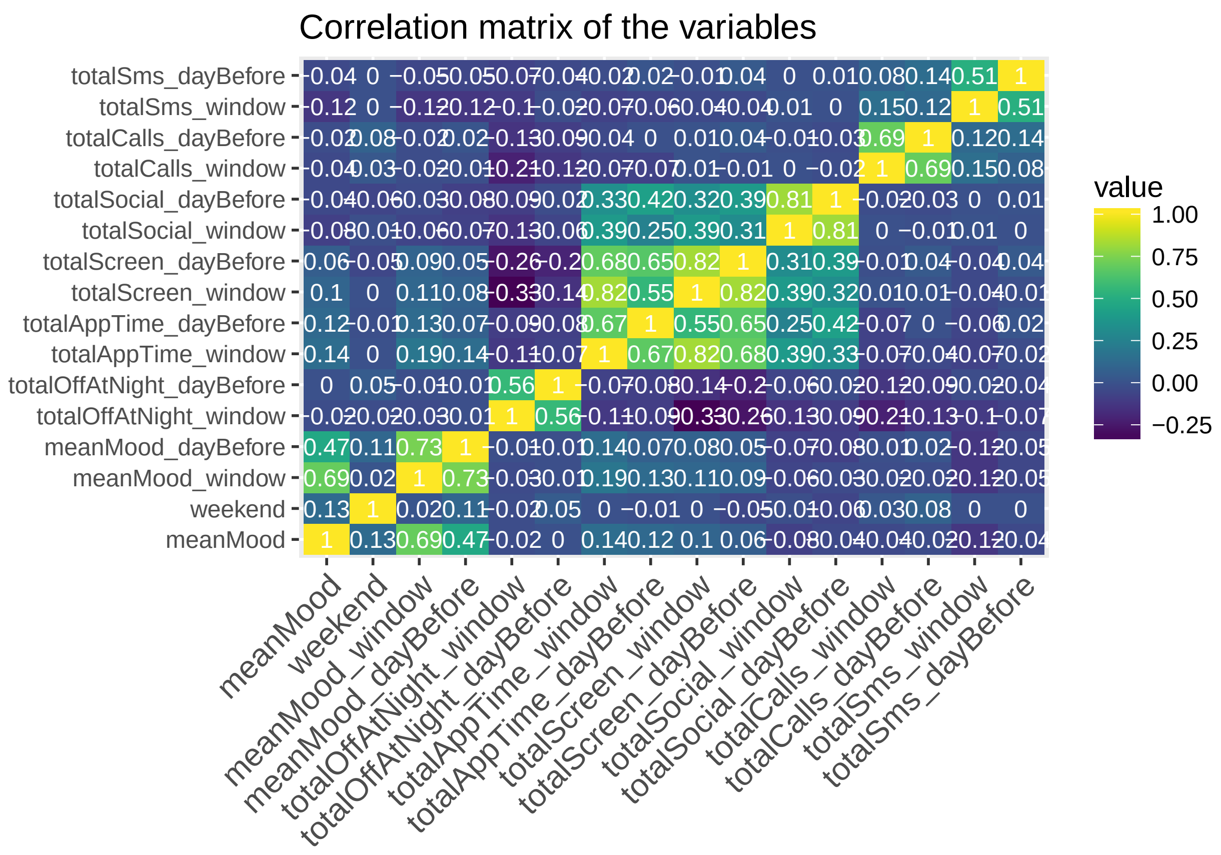
## Feature engineering from summaries.

Note, to predict the mood of the next day, we should not use observations of activity during the day itself but rather of activity in preceding days. We will construct those features here.

```r
'%ni%' <- function(x,y)!('%in%'(x,y))   # Not in operator.
allowedFeatures = c("subject","day","weekend","meanMood")
notAllowedFeatures = setdiff(names(summary_df),allowedFeatures)

derived_df <- summary_df %>% subset(select=c("subject","day","weekend","meanMood"))
window_size <- 5

# Some features based on preceding periods.
derived_df <- summary_df %>% group_by(subject) %>%
  mutate(meanMood_window = roll_mean(meanMood, window_size, align="right", fill = NA)) %>%
  mutate(meanMood_dayBefore = lag(meanMood)) %>%
  mutate(totalOffAtNight_window = roll_sum(night_off_time, window_size, align="right", fill = NA)) %>%
  mutate(totalOffAtNight_dayBefore = lag(night_off_time)) %>%
  mutate(totalAppTime_window = roll_sum(total_app_usage, window_size, align="right", fill=NA)) %>%
  mutate(totalAppTime_dayBefore = lag(total_app_usage)) %>%
  mutate(totalScreen_window = roll_sum(total_screen_time, window_size, align="right", fill=NA)) %>%
  mutate(totalScreen_dayBefore = lag(total_screen_time)) %>%
  mutate(totalSocial_window = roll_sum(total_appCat.social, window_size, align="right", fill=NA)) %>%
  mutate(totalSocial_dayBefore = lag(total_appCat.social)) %>%
  mutate(totalCalls_window = roll_sum(total_call, window_size, align="right", fill=NA)) %>%
  mutate(totalCalls_dayBefore = lag(total_call)) %>%
  mutate(totalSms_window = roll_sum(total_sms, window_size, align="right", fill=NA)) %>%
  mutate(totalSms_dayBefore = lag(total_sms)) %>%
  merge(derived_df, all.x = TRUE, all.y= TRUE) %>%
  arrange(subject, day) %>%
  group_by(subject) %>%
  slice(window_size:n())    # Remove first (window-size-1) features for each subject.

# Remove non-allowed features
derived_df <- derived_df %>% subset(select = names(derived_df) %ni% notAllowedFeatures)
```

## Correlations

```
derived_df[-c(1,2)] %>% cor(use="pairwise.complete.obs") %>% round(2) %>% melt() %>%
  ggplot(aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  geom_text(aes(label = round(value, 2)),size=3,colour='white') +
  scale_fill_viridis() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1),
        axis.title = element_blank()) +
  ggtitle("Correlation matrix of the variables")
```



OLDDDDDDDDDDDDDD STUFF HERE.

## Next days

Prediction error with mean alone.