

Annotated Code and Summary Statistics

February 27, 2020

The script, `pns-engage.R`, takes raw data from the PNS study, performs a series of data processing tasks, and then produces a dataset to be used in analyses. The functions called in this script implement decisions made during the process of curating data. We provide details of the most important functions, annotated code, and summary statistics on the amount of data retained after specific data processing steps. Code from `pns-engage.R` are highlighted in marked in sky blue (code highlights are visible in html version of this document).

1 Details of the `SetUpPostQuit()` function

The function `SetUpPostQuit(df.raw, df.time.frame){...}` implements data processing tasks performed for all analyses using PNS study data. Here, we annotate code `{...}` within this function, highlighted in sea green. First, let us create the `df.time.frame` and `df.raw` variables, which are inputs to this function.

```
pns.quit.dates <- read.csv(file.path(path.pns.input_data, "pns_quit_dates.csv"))
df.time.frame <- SetTimeFrame(df.quit.dates = pns.quit.dates,
                             study.duration = 21,
                             addtime = 0)
```

```
# Note: When using SetUpPostQuit() for other analyses,
# df.raw could alternatively be one of the other PNS Post Quit data
df.raw <- read.csv(file.path(path.pns.input_data, "Post_Quit_Random.csv"))
```

Time variables prefixed by `delivered.` and `assessment.` pertain to time when an EMA was delivered, and when a participant began completing an EMA, respectively.

```
# Rename variables in raw data and create new time variables
df.out <- df.raw %>%
  rename(id = Part_ID,
         record.id = Record_ID,
         record.status = Record_Status,
         assessment.type = Asse_Name,
         delivered.hrts = Initiated, # Time when EMA was delivered
         assessment.hrts = AssessmentBegin # Time when participant began completing EMA
        ) %>%
  mutate(record.id = as.character(record.id),
         assessment.type = as.character(assessment.type),
         delivered.hrts = as.character(delivered.hrts),
         assessment.hrts = as.character(assessment.hrts)) %>%
  mutate(delivered.unixts = as.POSIXct(strptime(delivered.hrts,
                                              format = "%m/%d/%Y %I:%M:%S %p",
                                              tz="EST5EDT")),
         assessment.unixts = as.POSIXct(strptime(assessment.hrts,
                                              format = "%m/%d/%Y %I:%M:%S %p",
                                              tz="EST5EDT"))) %>%
  mutate(delivered.unixts = as.numeric(delivered.unixts),
```

```

    assessment.unixts = as.numeric(assessment.unixts)) %>%
  mutate(delay = assessment.unixts - delivered.unixts)

# How many EMAs are in df.out now?
nrow(df.out)

## [1] 8358

# Sanity check: Do all EMAs have a timestamp corresponding to time delivered?
# If this is the case, the output should be `TRUE`
nrow(df.out) == sum(!is.na(df.out$delivered.unixts))

## [1] TRUE

```

Each row in `df.raw` contains a time stamp of when an EMA was delivered; rows in `df.raw` corresponding to EMAs delivered outside lower and upper time bounds specified in `df.time.frame` are excluded.

```

# Decision rule: exclude EMAs delivered before start of clock or
# after end of clock
df.out <- df.out %>%
  left_join(x = ., y = df.time.frame, by = "id") %>%
  filter(delivered.unixts >= start.clock & delivered.unixts <= end.clock) %>%
  arrange(id, delivered.unixts)

# How many EMAs are in df.out now?
nrow(df.out)

```

```
## [1] 7577
```

The `Responded` and `Completed` variables in the raw data pertain to timestamps of when participants began responding or completed EMAs, respectively. `Responded` has two levels, `True` and `Missing` while `Completed` has three levels, `True`, `False`, and `Missing`.

```

df.out <- df.out %>%
  mutate(Responded = as.character(Responded),
         Completed = as.character(Completed)) %>%
  mutate(Responded = if_else(Responded=="", "Missing", Responded),
         Completed = if_else(Completed=="", "Missing", Completed))

# Tabulate combination values from these two variables
table(df.out$Responded, df.out$Completed)

```

```

##
##           False Missing True
## Missing  2139      27    0
##  True      255      19 5137

```

Let us investigate the record status of the groups with a missing value.

```

df.out %>%
  filter(Responded=="Missing" & Completed=="Missing") %>%
  group_by(record.status) %>%
  summarise(n())

## # A tibble: 1 x 2
##   record.status `n()`
##   <fct>         <int>
## 1 FRAGMENT RECORD      27

```

```
df.out %>%
  filter(Responded=="Missing" & Completed=="False") %>%
  group_by(record.status) %>%
  summarise(n())
```

```
## # A tibble: 1 x 2
##   record.status      `n()`
##   <fct>             <int>
## 1 Incomplete/Timed Out 2139
```

```
df.out %>%
  filter(Responded=="True" & Completed=="Missing") %>%
  group_by(record.status) %>%
  summarise(n())
```

```
## # A tibble: 1 x 2
##   record.status      `n()`
##   <fct>             <int>
## 1 FRAGMENT RECORD    19
```

Decision rule: exclude EMAs that are "not valid"

```
df.out <- df.out %>%
  filter((Responded=="True" & Completed=="True") |
         (Responded=="True" & Completed=="False") |
         (Responded=="True" & Completed=="Missing") |
         (Responded=="Missing" & Completed=="False")) %>%
  rename(responded=Responded,
         completed=Completed)
```

How many EMAs are in df.out now?

```
nrow(df.out)
```

```
## [1] 7550
```

Let us investigate the time between when an EMA is delivered to a participant (time variables prefixed by delivered.) and the time when a participant actually begins completing an EMA (time variables prefixed by assessment.).

```
df.out %>%
  summarise(no.record = sum(is.na(assessment.unixts)),
            with.record = sum(!is.na(assessment.unixts)),
            prop.positive = sum(!is.na(delay) & (delay>0))/with.record,
            prop.zero = sum(!is.na(delay) & (delay==0))/with.record,
            prop.negative = sum(!is.na(delay) & (delay<0))/with.record,
            MIN = min(delay, na.rm=TRUE)/(60*60), # in hours
            MAX = max(delay, na.rm=TRUE)/(60*60) # in hours
  )
```

```
##   no.record with.record prop.positive prop.zero prop.negative      MIN
## 1      2198      5352      0.8729447 0.1268685  0.000186846 -387.3797
##      MAX
## 1 17.58111
```

*# Decision rule: exclude EMAs based on difference between
assessment.unixts and delivered.unixts*

```
df.out <- df.out %>% filter(is.na(delay)|(delay >= 0))
```

```
# How many EMAs are in df.out now?
nrow(df.out)
```

```
## [1] 7549
```

Now, let us get a sense of the volume of data remaining for each participant.

```
df.out %>%
  group_by(id) %>%
  summarise(tot.prompts = n()) %>%
  summarise(MEAN = mean(tot.prompts),
            MIN = min(tot.prompts),
            MAX = max(tot.prompts))
```

```
## # A tibble: 1 x 3
##   MEAN  MIN  MAX
##   <dbl> <int> <int>
## 1  45.5     1   67
```

```
df.out %>%
  filter(responded=="True") %>%
  group_by(id) %>%
  summarise(tot.prompts = n()) %>%
  summarise(MEAN = mean(tot.prompts),
            MIN = min(tot.prompts),
            MAX = max(tot.prompts))
```

```
## # A tibble: 1 x 3
##   MEAN  MIN  MAX
##   <dbl> <int> <int>
## 1  33.2     1   63
```

```
df.out %>%
  filter(completed=="True") %>%
  group_by(id) %>%
  summarise(tot.prompts = n()) %>%
  summarise(MEAN = mean(tot.prompts),
            MIN = min(tot.prompts),
            MAX = max(tot.prompts))
```

```
## # A tibble: 1 x 3
##   MEAN  MIN  MAX
##   <dbl> <int> <int>
## 1  31.7     1   63
```

Let us now create the variable `with.any.response`. This variable is an indicator for whether there is any recorded response in each row of `df.out`.

```
# Calling CheckAnyResponse() constructs these variables
df.out <- CheckAnyResponse(df = df.out, drop.cols = these.cols)
```

```
df.out %>%
  mutate(check.condition = (with.any.response==1)) %>%
  filter(check.condition) %>%
  group_by(id) %>%
  summarise(tot.prompts = n()) %>%
  summarise(MEAN = mean(tot.prompts),
            MIN = min(tot.prompts),
```

```

MAX = max(tot.prompts))

## # A tibble: 1 x 3
##   MEAN  MIN  MAX
##   <dbl> <int> <int>
## 1  32.7    1   63

df.out %>%
  group_by(with.any.response, record.status, responded, completed) %>%
  summarise(num.ema=n()) %>%
  arrange(with.any.response, desc(record.status))

## # A tibble: 6 x 5
## # Groups:   with.any.response, record.status, responded [6]
##   with.any.response record.status      responded completed num.ema
##           <dbl> <fct>           <chr>      <chr>      <int>
## 1             0 Incomplete/Timed Out Missing    False      2139
## 2             0 Incomplete/Timed Out True       False       70
## 3             0 FRAGMENT RECORD   True       Missing     7
## 4             1 Incomplete/Timed Out True       False     185
## 5             1 FRAGMENT RECORD   True       Missing     12
## 6             1 Completed         True       True     5136

```

2 How do we operationalize engagement in completion of EMAs?

Let us count the number of EMAs corresponding to a TRUE value for the variables `with.any.response`, or `responded`, or `completed`.

```

df.out %>%
  summarise(tot.ema = n(),
            tot.with.any.response = sum(with.any.response),
            tot.responded = sum(responded=="True"),
            tot.completed = sum(completed=="True"),
            prop.with.any.response = tot.with.any.response/tot.ema,
            prop.responded = tot.responded/tot.ema,
            prop.completed = tot.completed/tot.ema
  )

##   tot.ema tot.with.any.response tot.responded tot.completed
## 1    7549             5333           5410           5136
##   prop.with.any.response prop.responded prop.completed
## 1           0.7064512           0.7166512           0.680355

```

Let us determine the type of timestamps these variables have.

```

df.out %>%
  mutate(is.missing.ass = is.na(assessment.unixts),
         is.missing.del = is.na(delivered.unixts)) %>%
  group_by(with.any.response,
            responded,
            completed,
            is.missing.ass,
            is.missing.del,
            record.status) %>%
  summarise(num.ema = n()) %>%
  print(width=Inf)

```

```
## # A tibble: 8 x 7
## # Groups:   with.any.response, responded, completed, is.missing.ass,
## #   is.missing.del [8]
##   with.any.response responded completed is.missing.ass is.missing.del
##           <dbl> <chr>      <chr>      <lgl>      <lgl>
## 1             0 Missing   False     FALSE     FALSE
## 2             0 Missing   False     TRUE      FALSE
## 3             0 True      False     FALSE     FALSE
## 4             0 True      False     TRUE      FALSE
## 5             0 True      Missing   TRUE      FALSE
## 6             1 True      False     FALSE     FALSE
## 7             1 True      Missing   FALSE     FALSE
## 8             1 True      True      FALSE     FALSE
##   record.status      num.ema
##   <fct>             <int>
## 1 Incomplete/Timed Out      16
## 2 Incomplete/Timed Out    2123
## 3 Incomplete/Timed Out       2
## 4 Incomplete/Timed Out     68
## 5 FRAGMENT RECORD          7
## 6 Incomplete/Timed Out    185
## 7 FRAGMENT RECORD         12
## 8 Completed                5136
```

The table shows that all EMAs in `df.out` have the timestamp `delivered.unixts`. However, the EMAs having `responded==Missing` and `is.missing.assessment.unixts==FALSE` are unexpected. Also, observe that when `with.any.response==1`, the associated EMA always has an `assessment.unixts` timestamp and an `delivered.unixts` timestamp. On the other hand, observe that when `with.any.response==0`, the associated EMA always has an `delivered.unixts` timestamp.

```
df.post.quit.random <- df.out
```

The variable `engage.yes` is used to operationalize engagement with EMA completion; it is a binary variable equal to 1 if `with.any.response` is equal to 1 and equal to 0 otherwise.

```
# Implement decision rules for outcome variable engaged.yes
df.post.quit.random <- df.post.quit.random %>%
  mutate(engaged.yes = with.any.response)
```

Let us count the number of EMAs corresponding to a TRUE value for the variables `engaged.yes`.

```
df.post.quit.random %>%
  summarise(tot.ema = n(),
            tot.engaged = sum(engaged.yes),
            prop.engaged = tot.engaged/tot.ema)
```

```
##   tot.ema tot.engaged prop.engaged
## 1    7549         5333    0.7064512
```

3 Time associated with each EMA in data analyses

Let the variable `time.unixts.scaled` be the time elapsed since quit (in seconds). How is this variable calculated for EMAs when a participant engaged in completion and for EMAs when participants did not engage in completion?

We see that there can be about a 5-minute to 20-minute time gap between `assessment.unixts` and `delivered.unixts` among 5% of EMAs with a `assessment.unixts` timestamp.

```
quantile((df.post.quit.random$delay)/60, c(0, 0.50, 0.95, 1), na.rm = TRUE) # in minutes
```

```
##           0%           50%           95%          100%
##    0.0000000    0.3166667    5.3583333 1054.8666667
```

How many EMAs have `delay` greater than 20 minutes?

```
df.post.quit.random %>% filter(delay > 20*60) %>% summarise(num.ema=n())
```

```
##   num.ema
## 1       16
```

When participant engages EMA completion (i.e., `engaged.yes==1`), then time when s/he began the assessment (the variable `assessment.unixts`) is associated with the EMA in the variable `time.unixts`. On the other hand when a participant does not engage in EMA completion, then the time when EMA was delivered (the variable `delivered.unixts`) is associated with the EMA in the variable `time.unixts`. Hence, time elapsed since quit is then the difference between `time.unixts` and Quit Time (the variable `start.clock`, which represents 4am on Quit Day). This is implemented using the code below.

```
# Implement decision rules for:
# (1) Timestamp when engaged.yes=1
# (2) Timestamp when engaged.yes=0
df.post.quit.random <- df.post.quit.random %>%
  mutate(time.unixts = if_else(engaged.yes == 1, assessment.unixts, delivered.unixts)) %>%
  mutate(time.unixts.scaled = time.unixts - start.clock) %>%
  mutate(delivered.unixts.scaled = delivered.unixts - start.clock) %>%
  mutate(assessment.unixts.scaled = assessment.unixts - start.clock)
```

How many individuals at least 1 row during the 21 day post-quit period?

```
length(unique(df.post.quit.random$id))
```

```
## [1] 166
```

4 Plots of the data

```
library(ggplot2)
df.plot <- df.post.quit.random %>%
  mutate(t = time.unixts.scaled/(60*60*24) + 4/24) %>% # t is 12AM on Quit Day
  mutate(engaged.yes = as.factor(engaged.yes))
gg.all <- ggplot(df.plot)

all.inc <- seq(1,21,1)
for(i in 1:length(all.inc)){
  inc <- all.inc[i]
  gg.all <- gg.all + annotate("rect", xmin=-2/24 + inc, xmax=8/24 + inc, ymin=3000, ymax=Inf, alpha=0.1)
}

gg.all <- gg.all + geom_point(aes(t, id, color=engaged.yes), alpha=0.5)
gg.all <- gg.all + labs(x = "No. of Days since 12AM on Quit Day")
gg.all <- gg.all + labs(y="Each row is one participant's data")
gg.all <- gg.all + labs(title = "Time of EMA delivery \nAll Random EMAs within 21-Day Post Quit Period")
gg.all <- gg.all + labs(subtitle = "Shaded area denotes time between 10PM - 8AM \nEach point denotes one EMA")
gg.all <- gg.all + theme(axis.text.y = element_blank(), axis.ticks.y = element_blank())
```

```
gg.all <- gg.all + theme(legend.position = "bottom")
```

```
# Zoom to first few days
```

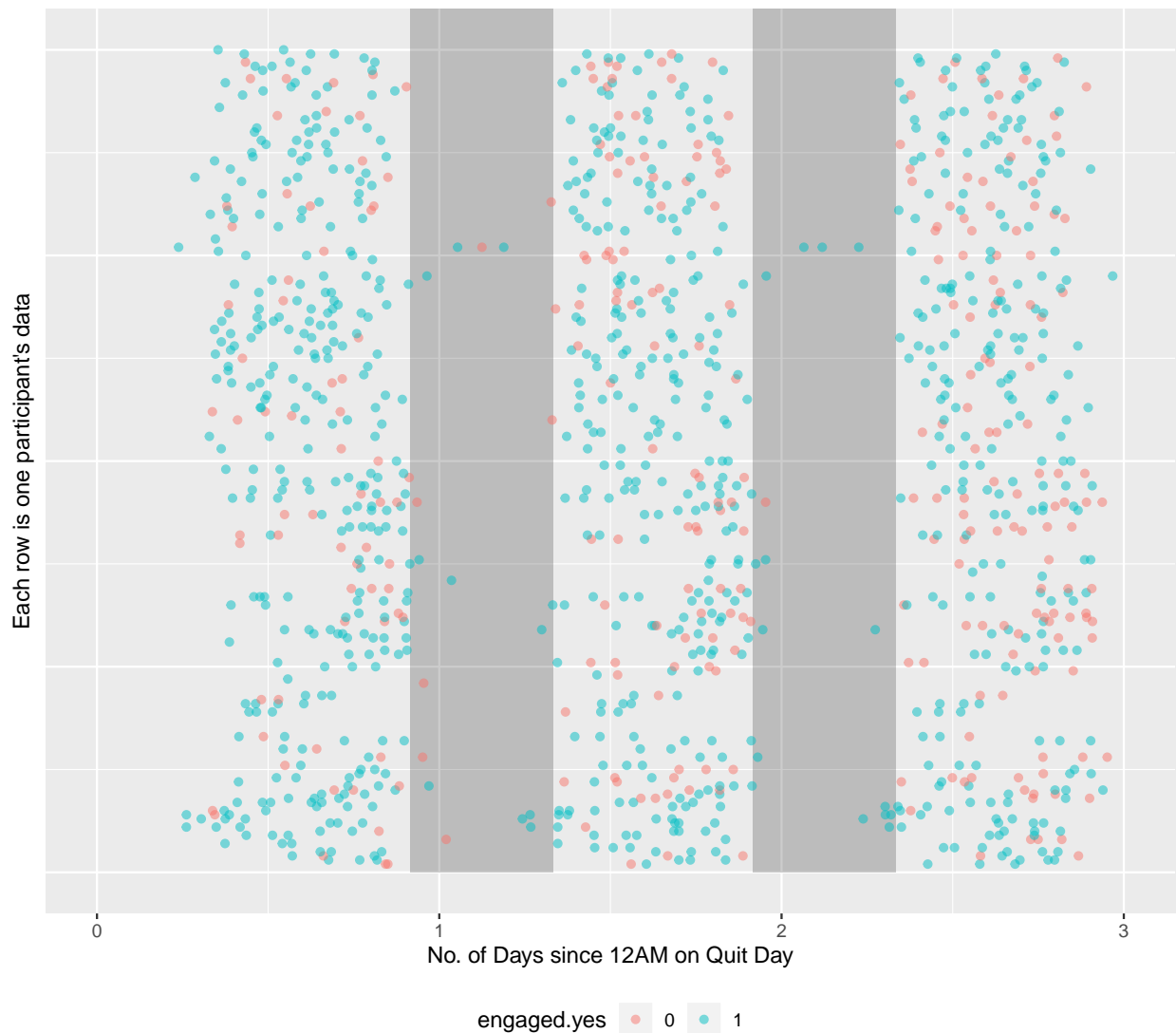
```
gg.all + xlim(0,3)
```

Time of EMA delivery

All Random EMAs within 21-Day Post Quit Period

Shaded area denotes time between 10PM – 8AM

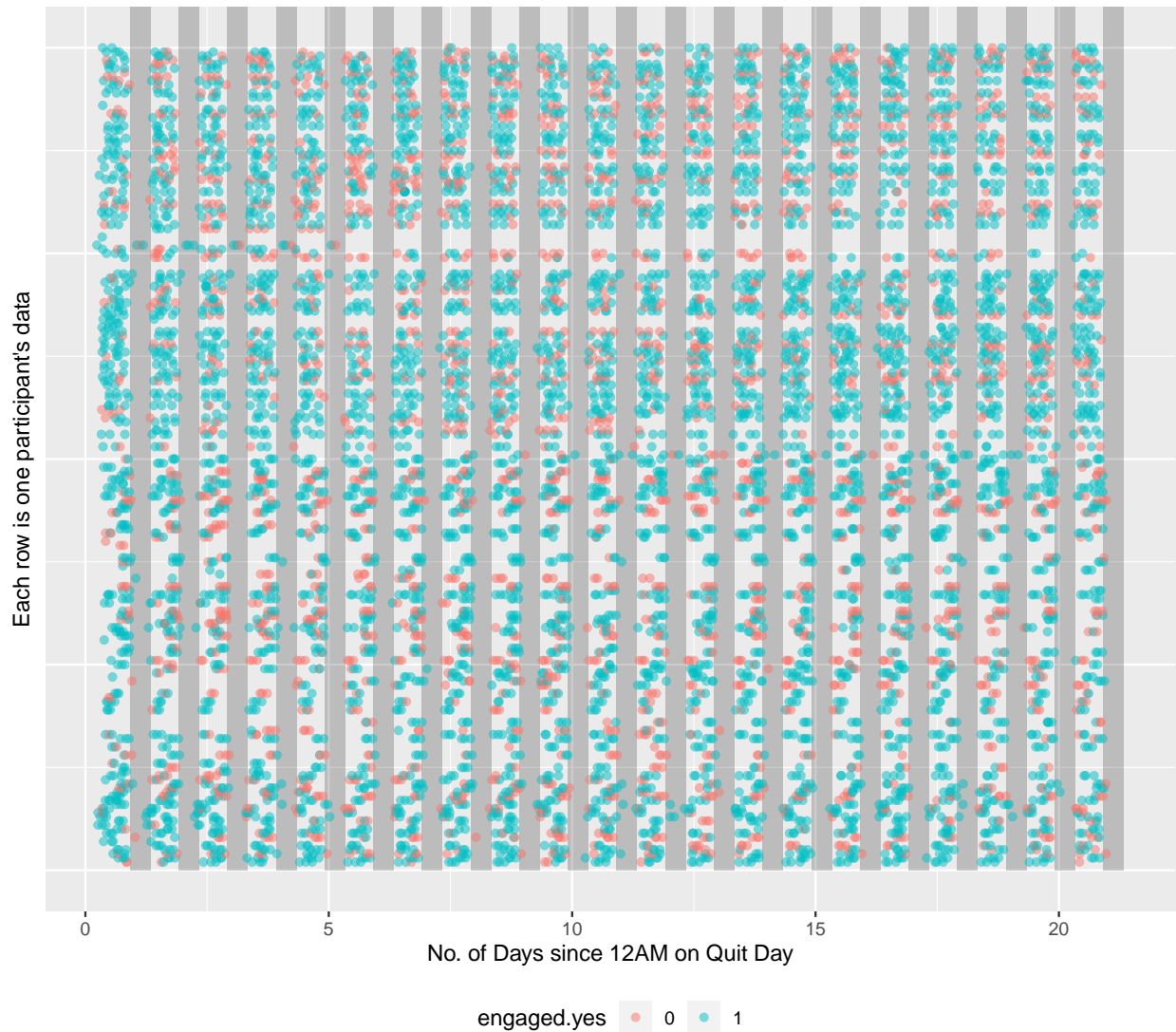
Each point denotes one random EMA



```
# Plot all days
```

```
gg.all
```


Time of EMA delivery
 All Random EMAs within 21-Day Post Quit Period
 Shaded area denotes time between 10PM – 8AM
 Each point denotes one random EMA



```
library(ggplot2)
library(grid)
library(gridExtra)
set.seed(3798)

df.plot <- df.post.quit.random %>%
  mutate(t = time.unixts.scaled/(60*60*24) + 4/24) %>% # t is 12AM on Quit Day
  mutate(Affect6 = if_else(engaged.yes==0, as.integer(0), Affect6)) %>% # For plotting
  mutate(Affect8 = if_else(engaged.yes==0, as.integer(0), Affect8)) %>%
  mutate(engaged.yes = as.factor(engaged.yes))

cols <- c("0" = "red", "1" = "blue")

ids <- unique(df.plot$id)
```

```

ids <- sample(ids, size=32)

collect.plots <- list()
for(i in 1:length(ids)){
  use.this.id <- ids[i]
  df.plot.this.participant <- df.plot %>% filter(id==use.this.id)
  gg.all <- ggplot(df.plot.this.participant)

  all.inc <- seq(1,21,1)
  for(i in 1:length(all.inc)){
    inc <- all.inc[i]
    gg.all <- gg.all + annotate("rect", xmin= -2/24 + inc, xmax=8/24 + inc, ymin=0, ymax=5.2, alpha=0.2)
  }

  gg.all <- gg.all + geom_point(aes(t, Affect6, color=engaged.yes), alpha=0.5)
  gg.all <- gg.all + scale_colour_manual(values = cols)
  #gg.all <- gg.all + labs(x = "No. of Days since 12AM on Quit Day")
  #gg.all <- gg.all + labs(y="Response to EMA item")
  gg.all <- gg.all + labs(x="", y="")
  gg.all <- gg.all + scale_y_continuous(breaks = c(1,2,3,4,5), labels = c("1","2","3","4","5"))
  gg.all <- gg.all + theme(legend.position = "None")

  # Plot all days
  collect.plots <- append(collect.plots, list(gg.all))
}

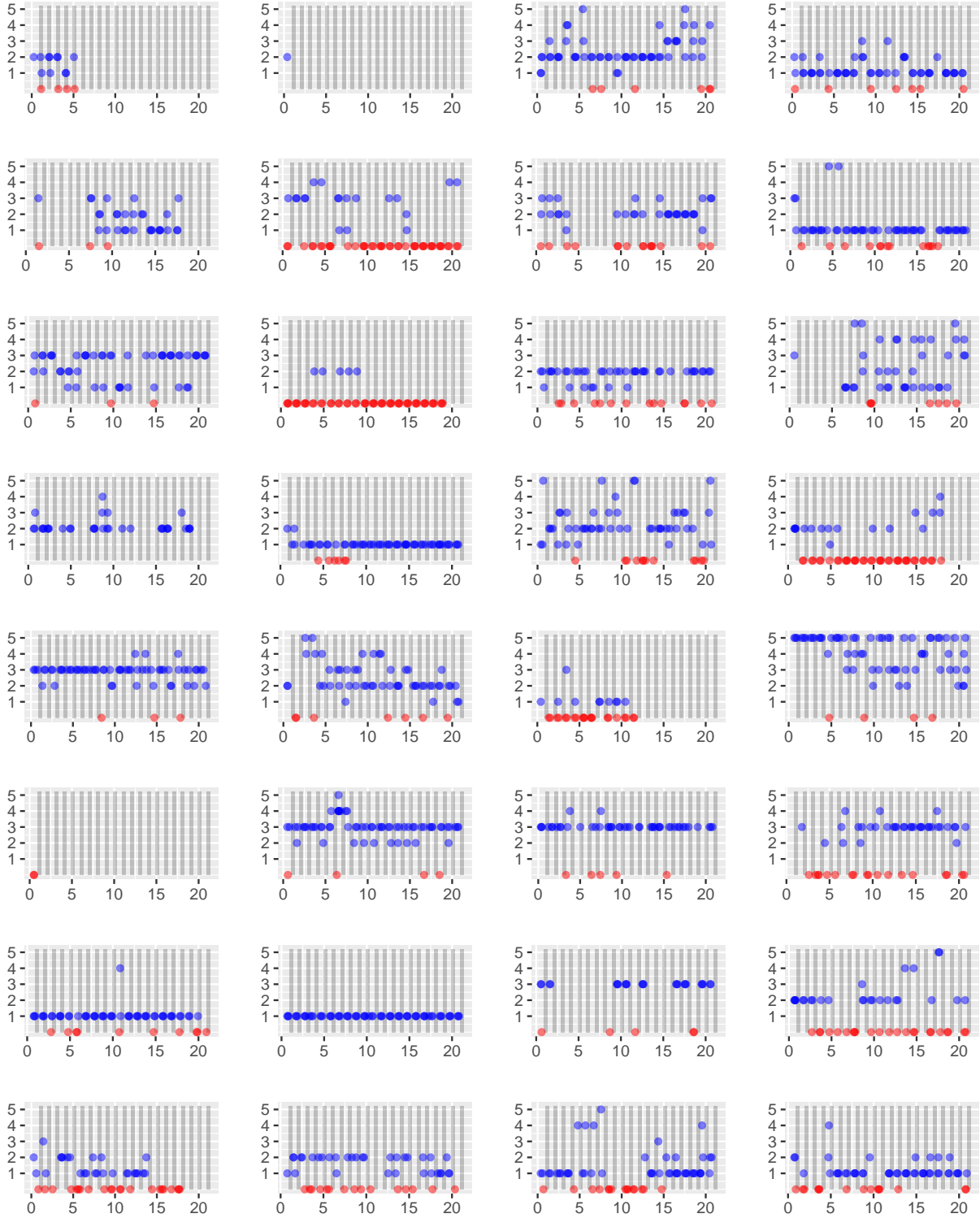
plot.grid <- marrangeGrob(grobs = collect.plots, ncol=4, nrow = 8,
  top = textGrob("Time of EMA delivery versus response to Affect6 ('I feel angry")
  gp=gpar(fontsize=11,font=3)
  ),
  bottom = textGrob("Shaded area denotes time between 10PM - 8AM \nEach point d
  gp=gpar(fontsize=11,font=3)
  )
)

plot.grid

```

*Time of EMA delivery versus response to Affect6 ('I feel angry.') on a 5-point Likert scale
for a sample of individuals from the PNS study*

All Random EMAs within 21-Day Post Quit Period



Shaded area denotes time between 10PM – 8AM

Each point denotes one random EMA (red dots: engaged.yes=0, blue dots: engaged.yes=1)

```

collect.plots <- list()
for(i in 1:length(ids)){
  use.this.id <- ids[i]
  df.plot.this.participant <- df.plot %>% filter(id==use.this.id)
  gg.all <- ggplot(df.plot.this.participant)

  all.inc <- seq(1,21,1)
  for(i in 1:length(all.inc)){
    inc <- all.inc[i]
    gg.all <- gg.all + annotate("rect", xmin= -2/24 + inc, xmax=8/24 + inc, ymin=0, ymax=5.2, alpha=0.2)
  }

  gg.all <- gg.all + geom_point(aes(t, Affect8, color=engaged.yes), alpha=0.5)
  gg.all <- gg.all + scale_colour_manual(values = cols)
  #gg.all <- gg.all + labs(x = "No. of Days since 12AM on Quit Day")
  #gg.all <- gg.all + labs(y="Response to EMA item")
  gg.all <- gg.all + labs(x="", y="")
  gg.all <- gg.all + scale_y_continuous(breaks = c(1,2,3,4,5), labels = c("1","2","3","4","5"))
  gg.all <- gg.all + theme(legend.position = "None")

  # Plot all days
  collect.plots <- append(collect.plots, list(gg.all))
}

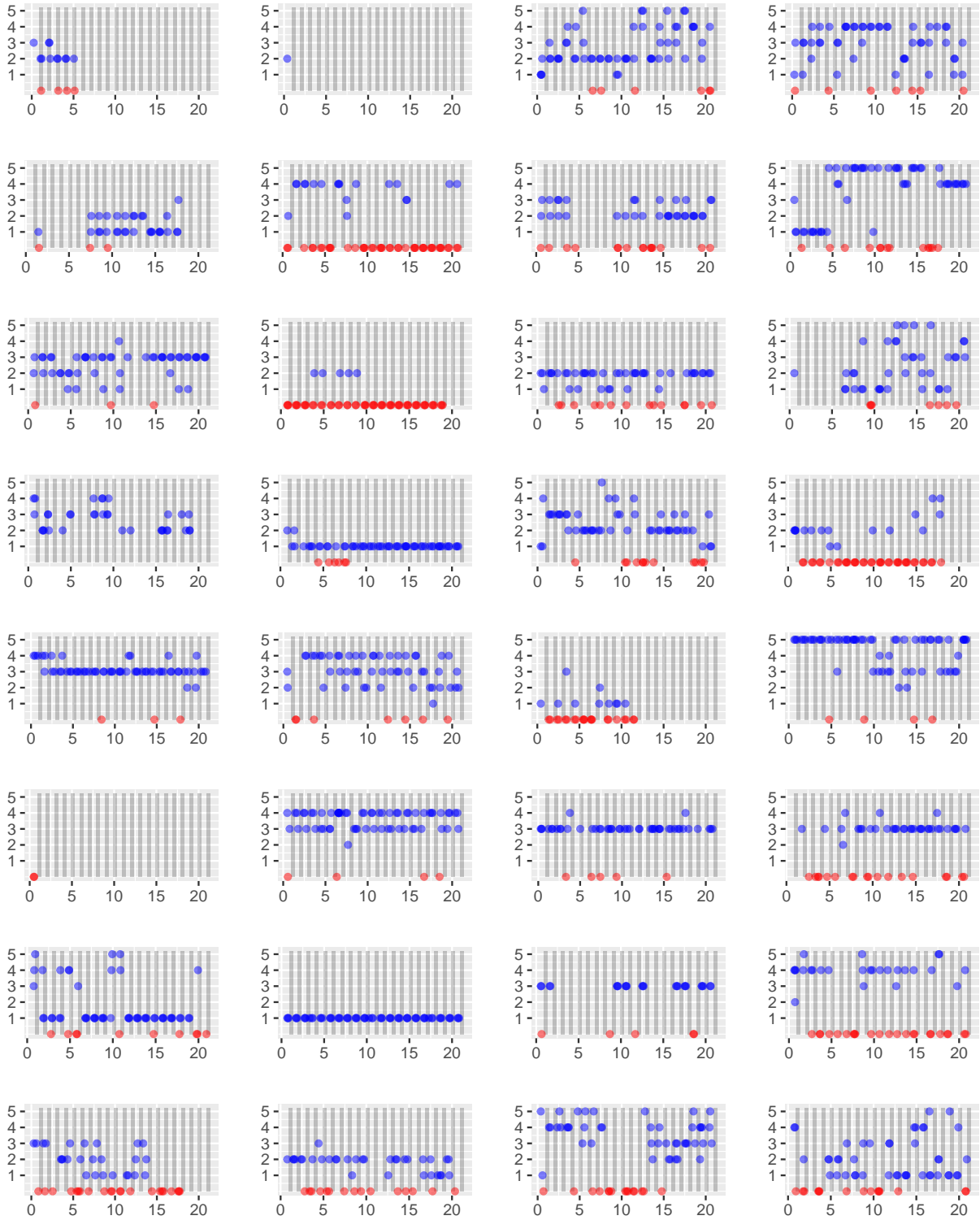
plot.grid <- marrangeGrob(grobs = collect.plots, ncol=4, nrow = 8,
  top = textGrob("Time of EMA delivery versus response to Affect8 ('I feel rested')",
    gp=gpar(fontsize=11,font=3)
  ),
  bottom = textGrob("Shaded area denotes time between 10PM - 8AM \nEach point denotes a response to EMA",
    gp=gpar(fontsize=11,font=3)
  )
)

plot.grid

```

*Time of EMA delivery versus response to Affect8 ('I feel restless.') on a 5-point Likert scale
for a sample of individuals from the PNS study*

All Random EMAs within 21-Day Post Quit Period



Shaded area denotes time between 10PM – 8AM

Each point denotes one random EMA (red dots: engaged.yes=0, blue dots: engaged.yes=1)