IAPS Synthesis Document

Jonathan Trattner, Rachel Jones, and Delaney Teceno

Last compiled on August 03, 2020

Introduction

This document synthesizes different experiments aimed at understanding the neuro-computational basis of subjective feelings. In 2018, forty-eight subjects (original cohort) rated 120 images from the International Affective Picture System database which provides normative ratings on valence, arousal, and dominance. Our goal is to predict these normative ratings based on subjects' responses to the questions "How positive does this image make you feel?" and "How negative does this image make you feel?" We test our model on data collected during the summer of 2020 (new cohort), consisting of 24 healthy volunteers (18 females and 6 males). Subjects were recruited for this study at the Piedmont Triad Research Community Center and are between the ages of 18 and 64. We conducted an a priori power analysis using a power value of no lower than 80% for a linear regression with an α value of 0.05. We used the R package `pwr` to calculate the minimal sample size for general linear model. This suggested 20 participants would ensure 80% statistical power. All subjects were screened 24 hours in advance, and minutes before, scheduled appointments to ensure no COVID-19 symptomology. The average age of participants was "X" years old (SD = x, range = x). The study was approved by the institutional review board of Wake Forest University School of Medicine.

The first part of this document provides reproducible code for each step in the data analysis pipeline, including reading the data, tidying the data, running regression models, and visualizing results for the ratings task described above. The second part of this document provides reproducible code for analyzing performance on an IAPS choice task completed only by new cohort subjects.

Ratings Task

Read in Data

Below is the code to read in the data. The directories chunk is specific to the files' location on our computer and will need to be changed if run on another machine. The data is manipulated as it's read in – please see the code comments for more detail.

```r
dirIAPS <- '~/Desktop/IAPS/IAPS Data/' #set the directory for IAPS data
dirData <- '~/Desktop/IAPS/Subject Data/' #set the directory for subject data
dirImageOrder <- '~/Desktop/IAPS/Subject Image Order/' #set the directory for image order
dirRatings <- '~/Desktop/IAPS/RJT/ratings/' #for new cohort
fileRatings <- dir_ls(dirRatings, glob = "*.txt.txt") #get the paths for the files that contain the str
filesIAPS <- dir_ls(dirIAPS) #list the files in the IAPS directory
filesData <- dir_ls(dirData) #list the files in the subject data directory
filesImageOrder <- dir_ls(dirImageOrder) #list the files in the image order directory
```

```r
#Read in the demographics data
demographics <- read_csv(file = "~/Desktop/IAPS/Demographics/KLRIF_2018-07-09.csv")

#Import the IAPS key data set
```

```r
keyIAPS <- filesIAPS %>%
  map_df(~read_csv(.x, na = c("", ".", NA),
                   col_types = cols(IAPS = col_character())) %>%
           select(desc:dom1sd) %>%
           drop_na(),
         .id = "form") %>%
  mutate(form = case_when(str_detect(form, "1_IAPS") ~ 1,
                          str_detect(form, "2_IAPS") ~ 2,
                          str_detect(form, "3_IAPS") ~ 3,
                          str_detect(form, "4_IAPS") ~ 4),
         form = as.factor(form))

#Read in subject rating
subjectData <- filesData %>%
  map_df(~read_csv(.x) %>%
           rename(time = Timestamp,
                  subject = `Subject ID:`),
         .id = "form") %>%
  mutate(form = case_when(str_detect(form, "1_SFA") ~ 1,
                          str_detect(form, "2_SFA") ~ 2,
                          str_detect(form, "3_SFA") ~ 3,
                          str_detect(form, "4_SFA") ~ 4),
           form = as.factor(form)) %>%
  #pivot the data so all ratings are in a column "rating" instead of in their own rows.
  pivot_longer(cols = -c(form, time, subject), values_to = "rating") %>%
  select(-name) %>%
  #have a running order for each subject
  group_by(subject) %>%
  mutate(order = row_number()) %>%
  ungroup() %>%
  #make this subjectID string match that of imageOrder
  mutate(subject = str_replace(subject, "\\-", "\\_")) %>%
  select(-time) %>%
  #arrange by each subject and the form (so one subject goes through all four
  #forms before moving to next subject)
  arrange(subject, form)


  imageOrder <- filesImageOrder %>%
  map_df(~read_csv(.x),
         .id = "form") %>%
  mutate(form = case_when(str_detect(form, "1_pictureID") ~ 1,
                          str_detect(form, "2_pictureID") ~ 2,
                          str_detect(form, "3_pictureID") ~ 3,
                          str_detect(form, "4_pictureID") ~ 4),
         form = as.factor(form)) %>%
  #pivot the data so all image values (IDs) are in a column "value"
  pivot_longer(cols = -c(X1, form),
               names_to = "subject") %>%
  #remvoe subject prefix
  mutate(subject = str_remove(.$subject, "\\d\\_")) %>%
  #arrange by subject
  arrange(subject) %>%
  #remove NAs that appear because of how the data was read in where each form's
  #image ID were basically in a new column for the subject with the prefix
  #1_SFA_..."
  drop_na() %>%
  #have a running order for each subject, and remove the X1 column.
  group_by(subject) %>%
  mutate(order = row_number()) %>%
  ungroup() %>%
  select(-X1)

#Read in and manipulate the new cohort data.
newCohortRatings <-
  #Read in all the ratings data into a list with each element corresponding to a
  #different file.
  map(fileRatings, ~readRatings(.x)) %>%
  #take each list element (subject file) and call our function
  #processRatingsData, creating an ID column called "subject". This column
  #simply contains the file path string name.
  map_df(~processRatingsData(.x), .id = "subject") %>%
  #extract from the subject column the IDs. For example "IAPS_B_01".
  mutate(subject = str_extract(subject, "[:graph:]{9}(?=rating.txt)"))

#Read in and manipulate the new cohort demographics information. File path is
#dependent on user's computer.
newCohortDems <- read_csv('~/Desktop/IAPS/RJT/demographics/IAPSdems.csv') %>%
  mutate(gender = case_when(gender == "female" ~ 0,
                            gender == "male" ~ 1))
```

Tidy the Data

This section describes how we tidied the data in order to get it in a format to perform the elastic net regression. Please see code comments for more details.

```
#tidy the demographics data
tidyDems <- demographics %>%
  #rename variables
  rename(subject = "Subject ID",
         age = "Age (years)",
         gender = "What is your gender?") %>%
  #make subject characters line up with others and recode gender as binary for
  #regression purposes
  mutate(subject = str_replace(subject, "\\-", "\\_"),
         gender = recode(gender, "Male" = 1, "Female" = 0)) %>%
  #select only relevant variables
  select(subject, age, gender)


#join the imageOrder and subject data by subject, form, and order.
orderedSubjectData <- full_join(imageOrder, subjectData, by = c("form", "subject", "order")) %>%
  rename(picID = value)

#Label the orderedSubjectData with a column that has the type of question asked
#based on the picID's P or N. Then rename the picID column as IAPS, removing any
```

```
#"P" or "N" at the end, so the names are the same as in the keyIAPS dataframe.
labeledOrderedSubjectData <- orderedSubjectData %>%
  mutate(question = case_when(str_detect(picID, "P") ~ "positive",
                              str_detect(picID, "N") ~ "negative",
                              TRUE ~ "Oops"),
         picID = str_remove_all(picID, "P|N")) %>%
  rename(IAPS = picID)

#create positiveRatings as a dataframe that has IAPS data from the rounds where
#it was asked how positive the image made them feel. Then combine the rating and
#question data into one column that has the ratings for when the question was
#about positive feelings. I could've done this using the following code instead.
#Same thing.

#labeledOrderedSubjectData %>%
#  filter(question == "positive") %>%
#  select(-question) %>%
#  rename(positive = rating))

positiveRatings <- labeledOrderedSubjectData %>%
  filter(question == "positive") %>%
  pivot_wider(names_from = question, values_from = rating)

#create negativeRatings as a dataframe that has IAPS data from the rounds where
#it was asked how negative the image made them feel. Then combine the
#rating and question data into one column that has the ratings for when the
#question was about negative feelings.
negativeRatings <- labeledOrderedSubjectData %>%
  filter(question == "negative") %>%
  pivot_wider(names_from = question, values_from = rating)

#Combine the positive and negative ratings data by IAPS, subject, and form.
#Could've just used IAPS but just more specificity. Then remove miscellaneous
#order column and join that with the IAPS key. Then join that with the
#demographics info. Then remove NAs since there are some images participants saw
#that are not standardized in the IAPS key (and one participant who has no
#gender data and we exclude). The final data has 14 columns and 5640 rows and
#consists of 120 positive and negative ratings from 47 subjects.
finalData <- full_join(positiveRatings, negativeRatings, by = c("IAPS", "subject", "form")) %>%
  select(-c(order.x, order.y)) %>%
  full_join(keyIAPS, by = c("IAPS", "form")) %>%
  full_join(tidyDems, by = "subject") %>%
  drop_na() %>%
  select(subject, age, gender,
         form, IAPS, desc,
         positive, negative, everything())

# Create the final data to perform regressions by setting up the new cohort
# ratings data using the iapsr regSetup function, joining that with the new
# cohort demographics information by subject, and joining that by the IAPS
# ratings key.
newCohortFinalData <- newCohortRatings %>%
  #remove a picture not in the IAPS key
  filter(picture != "7037") %>%
  regSetup() %>%
  left_join(newCohortDems, by = "subject") %>%
  left_join(keyIAPS, by = "IAPS")
```

Below, we can see a preview of the data on which we will fit regression models. The first table shows the first 10 rows for subject SFA_1025 from the original cohort. The second table shows the first 10 rows for subject IAPS_B_01. The age column is in years, gender is binary (0 for females 1 for males for use in regression analysis), IAPS column is the picture ID. The desc column is a description of the IAPS images. The positive and negative columns show the subject's ratings when asked "How positive does this image make you feel?" and "How negative does this image make you feel?". The remaining columns describe the mean and standard deviation for the image's standardized valence, arousal, and dominance, per the IAPS dataset.

```
## # A tibble: 10 x 14
##    subject   age gender form  IAPS  desc  positive negative valmn valsd aromn
##    <chr>   <dbl>  <dbl> <fct> <chr> <chr>    <dbl>    <dbl> <dbl> <dbl> <dbl>
##  1 SFA_10~    29      0 1     5210  Seas~        7        1  8.03  1.09  4.6
##  2 SFA_10~    29      0 1     8179  Bung~        6        2  6.04  2.56  7.1
##  3 SFA_10~    29      0 1     7461  Fren~        6        1  6.14  2.42  5.38
##  4 SFA_10~    29      0 1     9480  Skull        1        9  3.12  2.39  5.87
##  5 SFA_10~    29      0 1     3120  Dead~        1       10  1.56  1.09  6.84
##  6 SFA_10~    29      0 1     8190  Skier        8        1  8.1   1.39  6.28
##  7 SFA_10~    29      0 1     9410  Sold~        7        7  1.51  1.15  7.07
##  8 SFA_10~    29      0 1     1440  Seal        10        0  8.19  1.53  4.61
##  9 SFA_10~    29      0 1     6910  Bomb~        7        3  5.31  2.28  5.62
## 10 SFA_10~    29      0 1     7476  Ramen        8        2  5.18  2.22  4.73
## # ... with 3 more variables: arosd <dbl>, dom1mn <dbl>, dom1sd <dbl>
```

```
## # A tibble: 10 x 14
##    subject IAPS  positive negative   age gender form  desc  valmn valsd aromn
##    <chr>   <chr>    <dbl>    <dbl> <dbl>  <dbl> <fct> <chr> <dbl> <dbl> <dbl>
##  1 IAPS_B~ 1640         6        1    31      0 3     Coyo~  6.27  2.22  5.13
##  2 IAPS_B~ 4302         0        7    31      0 3     Erot~  4.99  2.64  5.68
##  3 IAPS_B~ 4085         0        3    31      0 3     Erot~  5.71  2.37  5.77
##  4 IAPS_B~ 2512         3        0    31      0 1     Man    4.86  0.84  3.46
##  5 IAPS_B~ 4770         3        1    31      0 2     Fema~  4.91  2.61  5.85
##  6 IAPS_B~ 9480         2        3    31      0 1     Skull  3.12  2.39  5.87
##  7 IAPS_B~ 4232         1        7    31      0 1     Erot~  5.95  2.53  6.28
##  8 IAPS_B~ 3000         0        1    31      0 1     Muti~  1.59  1.35  7.34
##  9 IAPS_B~ 5626        10        0    31      0 4     Hang~  6.71  2.06  6.1
## 10 IAPS_B~ 1080         3        0    31      0 2     Snake  4.24  2.08  5.69
## # ... with 3 more variables: arosd <dbl>, dom1mn <dbl>, dom1sd <dbl>
```

Modeling the Data

Methods and Code

Using both the `tidymodels` framework in R and the `caret` package in R, we tuned an elastic net regression with a wide array of alpha (0 to 1) and lambda (0 to 200) values using six fold and ten fold cross validation. Our response variable was the mean rating of valence, arousal, or dominance, per the IAPS dataset. Our predictors were subjects' positive and negative ratings (and the interaction between them), age, and gender. We looked at the root mean squared error (rmse), mean absolute error (mae), and the R^2 metrics to evaluate model fit on a dataset of 6 subjects whose ratings were collected during the summer of 2020 (new cohort). These indicated that a regular linear regression performed best. We used both the `caret` package to perform

10 fold cross validation on the data, and the `lm` function in R fit on the entire dataset. The linear models performed equally well independent of cross validation. It should also be noted that we tested the linear model without the interaction between positive and negative ratings and it performed worse on all metrics. Included are code snippets for performing linear regressions using `lm` and `caret`. Elastic net regression code is included but model coefficients are not explicitly reported.

```r
#create a recipe object -- simply describing the formula to fit for predicting
#the mean valence from the positive and negative ratings. THEN add the
#interaction between positive and negative.
interactRec <- recipe(valmn ~ positive + negative + age + gender, data = train) %>%
  step_interact(terms = ~ positive:negative)


#create a workflow object with the recipe and model specification
wfInteract <- workflow() %>%
  add_recipe(interactRec) %>%
  add_model(netSpec)
```

```r
#tune the interact model
tic("tune interact grid")
netTunedInteract <- tune_grid(wfInteract,
                      resamples = CVtraining,
                      grid = netGrid)
toc()
```

```r
#preview penalty and mean RMSE/RSQ
netTunedInteract %>% collect_metrics() %>%
ggplot(aes(penalty, mean, color = mixture)) +
  geom_point(size = 1.5) +
  facet_wrap(~.metric, scales = "free", nrow = 2) +
  labs(y = "Mean", x = "Penalty", color = "Mixture", title = "Average Metric and Penalty") +
  myGGTheme
```

```r
#preview the lowest RMSE -- best metrics that yield it
print("Penalty and mixture and their relation to RMSE - Interact model:")
netTunedInteract %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  arrange(mean) %>%
  select(-c(.estimator, n)) %>%
  rename(Penalty = penalty,
         Mixture = mixture,
         Metric = .metric,
         Mean = mean,
         `Std Error` = std_err) %>%
  head(10)
```

```r
#select best interact metrics
bestMetricsInteract <- netTunedInteract %>% select_best(metric = "rmse")

#finalize the workflow with the best interact metrics
finalWFInteract <- finalize_workflow(wf, parameters = bestMetricsInteract)

#save last fit
testFitInteract <- last_fit(finalWFInteract, split = split)
```

```r
#preview testFit metrics
testFitInteract %>%
  collect_metrics() %>%
  select(-.estimator) %>%
  rename(Metric = .metric,
         Estimate = .estimate)

#Save interact model's predictions
InteractPreds <- testFitInteract %>%
  select(.predictions) %>%
  unnest(.predictions) %>%
  rename(interactPrediction = .pred,
         Actual = valmn)

#preview interact model's predictions
print("Preview of the predictions of best fitting interact model on mean valence per IAPS data:")
InteractPreds %>%
  select(interactPrediction, Actual) %>%
  head(20)


#preview model metrics
print("Model Beta Estimates for Elastic Net Regression - Interact:")
finalWFInteract %>%
  fit(test) %>%
  pull_workflow_fit() %>%
  vi(lambda = bestMetrics$penalty) %>%
  select(-Sign)

print("Interact Model Fit Metrics:")
metrics(InteractPreds, truth = Actual, estimate = interactPrediction) %>%
  select(-.estimator) %>%
  rename(Metric = .metric,
         Estimate = .estimate)
```

```r
#### Create a list of formulas for the outcome
outcomeFormulasCaret <- list("arousal" = aromn ~ positive + negative + (positive * negative) + age + ger
                        "dominance" = dom1mn ~ positive + negative + (positive * negative) + age + gende
                        "valence" = valmn ~ positive + negative + (positive * negative) + age + gender)

#Create strings vectors/lists that have the outcome strings and truths to supply in a map2 function for
outcomeStrings <- list("arousal" = "arousal", "dominance" = "dominance", "valence" = "valence")
outcomeTruths <- c("aromn", "dom1mn", "valmn")

#create a train control object
careTrain <- trainControl(method = "cv", number = 10)

#set the seed and predict the data for each model using 10 fold cross validation
set.seed(18)
caretNewPredicted <- map(outcomeFormulasCaret, ~ train(.x, data = finalData, trControl = careTrain, metl
  map_df(~predict(.x, newCohortFinalData) %>% bind_cols(newCohortFinalData), .id = "outcome") %>%
  rename(predicted = `...1`)

#get the metrics for the caret model
```

```r
caretNewPredMetrics <- map2_df(outcomeStrings, outcomeTruths, ~ caretNewPredicted %>% filter(outcome ==
                                metrics(truth = .y, estimate = predicted), .id = "outcome")
```

```r
#### Create a list of formulas for the outcome
outcomeFormulasCaretNoInteraction <- list("arousal" = aromn ~ positive + negative + age + gender,
                        "dominance" = dom1mn ~ positive + negative + age + gender,
                        "valence" = valmn ~ positive + negative + age + gender)

#Create strings vectors/lists that have the outcome strings and truths to supply in a map2 function for
outcomeStrings <- list("arousal" = "arousal", "dominance" = "dominance", "valence" = "valence")
outcomeTruths <- c("aromn", "dom1mn", "valmn")

#create a train control object
careTrain <- trainControl(method = "cv", number = 10)

#set the seed and predict the data for each model using 10 fold cross validation
set.seed(18)
caretNewPredictedNoInteraction <- map(outcomeFormulasCaretNoInteraction, ~ train(.x, data = finalData,
  map_df(~predict(.x, newCohortFinalData) %>% bind_cols(newCohortFinalData), .id = "outcome") %>%
  rename(predicted = `...1`)

#get the metrics for the caret model
caretNewPredMetricsNoInteraction <- map2_df(outcomeStrings, outcomeTruths, ~ caretNewPredictedNoInteract
                                metrics(truth = .y, estimate = predicted), .id = "outcome")
```

```r
compareCarets <- bind_rows(caretNewPredMetricsNoInteraction %>% mutate(type = "no interaction"),
                        caretNewPredMetrics %>% mutate(type = "interaction"))
```

```r
outcomeFormulasNoInteraction <- list("arousal" = lm(formula = aromn ~ positive + negative + age + gende
                                     "dominance" = lm(formula = dom1mn ~ positive + negative + age + ge
                                     "valence" = lm(formula = valmn ~ positive + negative + age + gende


#Create strings vectors/lists that have the outcome strings and truths to supply in a map2 function for
outcomeStrings <- list("arousal" = "arousal", "dominance" = "dominance", "valence" = "valence")
outcomeTruths <- c("aromn", "dom1mn", "valmn")


set.seed(18)
# Create a dataframe with the predictions for each based on the outcome
newCohortPredictedNoInteraction <- map_df(outcomeFormulasNoInteraction, ~ predict(.x, newCohortFinalDat
  rename(predicted = `...1`)

#get metrics for each model's fit on the new data
newCohortPredictedMetricsNoInteraction <- map2_df(outcomeStrings, outcomeTruths, ~ newCohortPredictedNo
```

```r
#### Create a list of formulas for the outcome
outcomeFormulas <- list("arousal" = lm(formula = aromn ~ positive + negative + (positive * negative) + a
                        "dominance" = lm(formula = dom1mn ~ positive + negative + (positive * negative)
                        "valence" = lm(formula = valmn ~ positive + negative + (positive * negative) + a


set.seed(18)
# Create a dataframe with the predictions for each based on the outcome
```

```r
newCohortPredicted <- map_df(outcomeFormulas, ~ predict(.x, newCohortFinalData) %>% bind_cols(newCohortF
  rename(predicted = `...1`)

#get metrics for each model's fit on the new data
newCohortPredictedMetrics <- map2_df(outcomeStrings, outcomeTruths, ~ newCohortPredicted %>% filter(outc
                                     metrics(truth = .y, estimate = predicted), .id = "outcome")
```

```r
compareLMs <- bind_rows(newCohortPredictedMetricsNoInteraction %>% mutate(type = "no interaction"),
                        newCohortPredictedMetrics %>% mutate(type = "interaction"))
```

```r
#### Create a list of formulas for the outcome
outcomeFormulasCaret <- list("arousal" = aromn ~ positive + negative + (positive * negative) + age + ge
                             "dominance" = dom1mn ~ positive + negative + (positive * negative) + age +
                             "valence" = valmn ~ positive + negative + (positive * negative) + age + ge

#Create strings vectors/lists that have the outcome strings and truths to supply in a map2 function fo
outcomeStrings <- list("arousal" = "arousal", "dominance" = "dominance", "valence" = "valence")
outcomeTruths <- c("aromn", "dom1mn", "valmn")

#create a train control object
careTrain <- trainControl(method = "cv", number = 10)
caretGrid <- expand.grid(alpha = seq(0, 1, by = 0.1),
  lambda = c(seq(0, 1, length = 20), seq(2,10, by = 1), seq(15,200, by = 5)))

#set the seed and predict the data for each model using 10 fold cross validation
set.seed(18)
caretGlmnetNewPredicted <- map(outcomeFormulasCaret, ~ train(.x, data = finalData, method = "glmnet", t
  map_df(~predict(.x, newCohortFinalData) %>% bind_cols(newCohortFinalData), .id = "outcome") %>%
  rename(predicted = `...1`)
```

```r
#get the metrics for the caret model
caretGlmnetNewPredMetrics <- map2_df(outcomeStrings, outcomeTruths, ~ caretGlmnetNewPredicted %>% filte
                                     metrics(truth = .y, estimate = predicted), .id = "outcome")
```

```r
#### Create a list of formulas for the outcome
outcomeFormulasCaretNoInteraction <- list("arousal" = aromn ~ positive + negative + age + gender,
                             "dominance" = dom1mn ~ positive + negative + age + gender,
                             "valence" = valmn ~ positive + negative + age + gender)

#Create strings vectors/lists that have the outcome strings and truths to supply in a map2 function fo
outcomeStrings <- list("arousal" = "arousal", "dominance" = "dominance", "valence" = "valence")
outcomeTruths <- c("aromn", "dom1mn", "valmn")

#create a train control object
careTrain <- trainControl(method = "cv", number = 10)
caretGrid <- expand.grid(alpha = seq(0, 1, by = 0.1),
                         lambda = c(seq(0, 1, length = 20), seq(2,10, by = 1), seq(15,200, by = 5)))

#set the seed and predict the data for each model using 10 fold cross validation
set.seed(18)
caretGlmnetNewPredNoInteraction <- map(outcomeFormulasCaretNoInteraction, ~ train(.x, data = finalData
  map_df(~predict(.x, newCohortFinalData) %>% bind_cols(newCohortFinalData), .id = "outcome") %>%
  rename(predicted = `...1`)
```

```
#get the metrics for the caret model
caretGlmnetNewPredMetricsNoInteraction <- map2_df(outcomeStrings, outcomeTruths, ~ caretGlmnetNewPredN
                                  metrics(truth = .y, estimate = predicted), .id = "outcome")
```

```
#### compare caret glmnets
compareCaretsGlmnet <- bind_rows(caretGlmnetNewPredMetricsNoInteraction %>% mutate(type = "no interact
                           caretGlmnetNewPredMetrics %>% mutate(type = "interaction"))
```

Performance Metrics

To reiterate, our best model is a simple linear regression that predicts the mean arousal, dominance, and valence level using subjects' positive and negative ratings (and the interaction between them), age, and gender. The model statistics can be seen in the tables below.

```
print("Linear Regression Coefficients for Predicting Arousal:")
```

```
## [1] "Linear Regression Coefficients for Predicting Arousal:"
```

```
outcomeFormulas$arousal %>% tidy()
```

```
## # A tibble: 6 x 5
##   term              estimate std.error statistic   p.value
##   <chr>                <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)        4.47      0.0593      75.5  0.
## 2 positive           0.0430    0.00545      7.89 3.55e- 15
## 3 negative           0.182     0.00529     34.4  2.57e-235
## 4 age                0.00331   0.00152      2.18 2.93e-  2
## 5 gender             0.0608    0.0307       1.98 4.78e-  2
## 6 positive:negative -0.0171    0.00133    -12.9  2.55e- 37
```

```
print("Linear Regression Coefficients for Predicting Dominance:")
```

```
## [1] "Linear Regression Coefficients for Predicting Dominance:"
```

```
outcomeFormulas$dominance %>% tidy()
```

```
## # A tibble: 6 x 5
##   term              estimate std.error statistic  p.value
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)        5.49      0.0462     119.   0.
## 2 positive           0.0737    0.00425     17.3  1.24e-65
## 3 negative          -0.206     0.00413    -49.9  0.
## 4 age               -0.00425   0.00118     -3.59 3.31e- 4
## 5 gender            -0.0872    0.0239      -3.64 2.71e- 4
## 6 positive:negative  0.0111    0.00104     10.7  2.21e-26
```

```
print("Linear Regression Coefficients for Predicting Valence:")
```

```
## [1] "Linear Regression Coefficients for Predicting Valence:"
```

```
outcomeFormulas$valence %>% tidy()
```

```
## # A tibble: 6 x 5
##   term              estimate std.error statistic   p.value
##   <chr>                <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)        5.38      0.0698      77.0  0.
## 2 positive           0.205     0.00642     32.0  3.17e-206
## 3 negative          -0.278     0.00624    -44.6  0.
## 4 age                -0.00637   0.00179     -3.56 3.68e-  4
## 5 gender             -0.132     0.0362      -3.64 2.71e-  4
## 6 positive:negative   0.00872   0.00157      5.56 2.75e-  8
```

```
print("Linear Regression Prediction Metrics:")
```

```
## [1] "Linear Regression Prediction Metrics:"
```

```
newCohortPredictedMetrics
```

```
## # A tibble: 9 x 4
##   outcome   .metric .estimator .estimate
##   <chr>     <chr>   <chr>          <dbl>
## 1 arousal   rmse    standard       1.11
## 2 arousal   rsq     standard       0.225
## 3 arousal   mae     standard       0.854
## 4 dominance rmse    standard       0.875
## 5 dominance rsq     standard       0.554
## 6 dominance mae     standard       0.679
## 7 valence   rmse    standard       1.28
## 8 valence   rsq     standard       0.630
## 9 valence   mae     standard       1.01
```

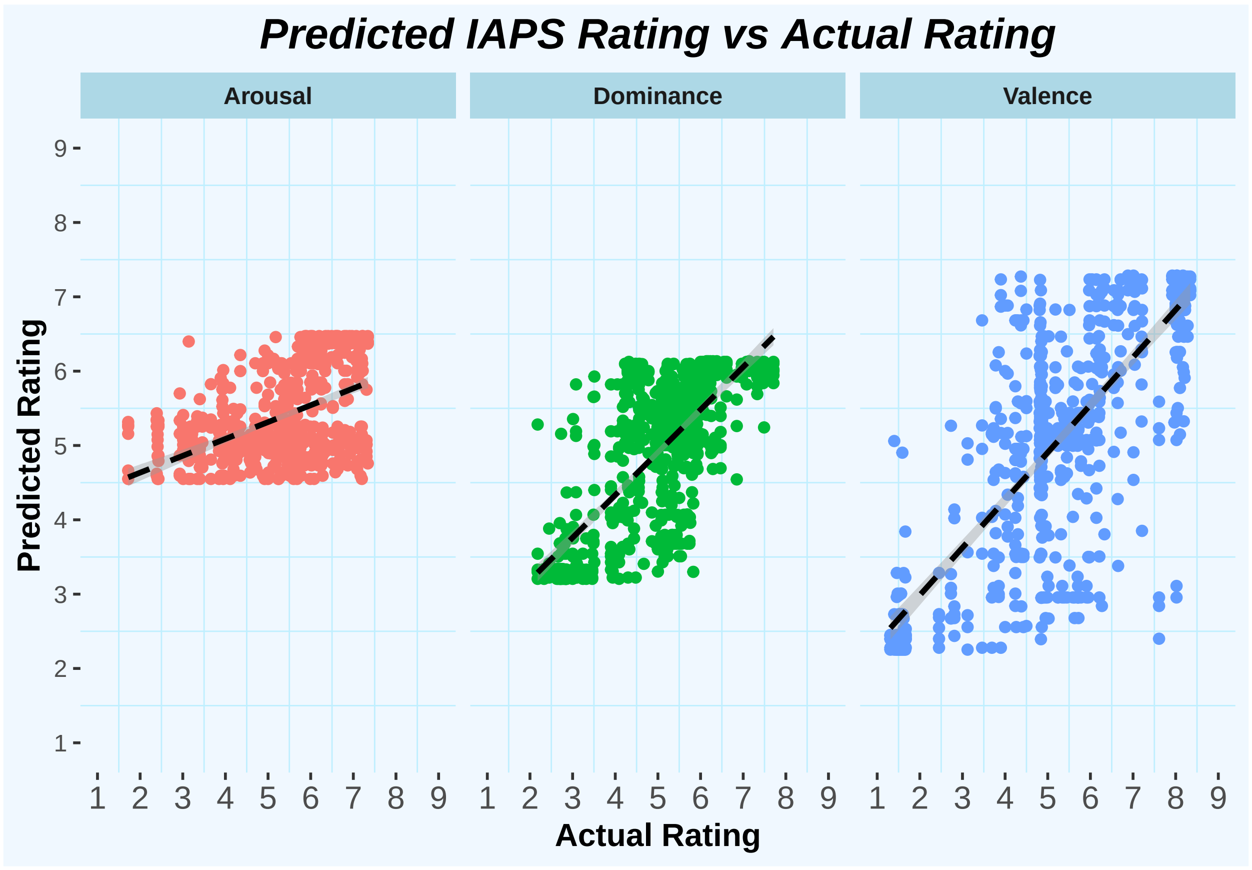
How do the predictions correlate?

```
#The actual column is "rating_mean" and the predicted column is "predicted"
actual_predicted <- newCohortPredicted %>%
  pivot_longer(cols = contains("mn"), names_to = "mean_type", values_to = "rating_mean") %>%
  pivot_longer(cols = contains("sd"), names_to = "sd_type", values_to = "rating_sd") %>%
  filter((outcome == "arousal" & mean_type == "aromn" & sd_type == "arosd") |
         (outcome == "dominance" & mean_type == "dom1mn" & sd_type == "dom1sd") |
         (outcome == "valence" & mean_type == "valmn" & sd_type == "valsd")) %>%
  mutate(outcome = str_to_sentence(outcome))
```

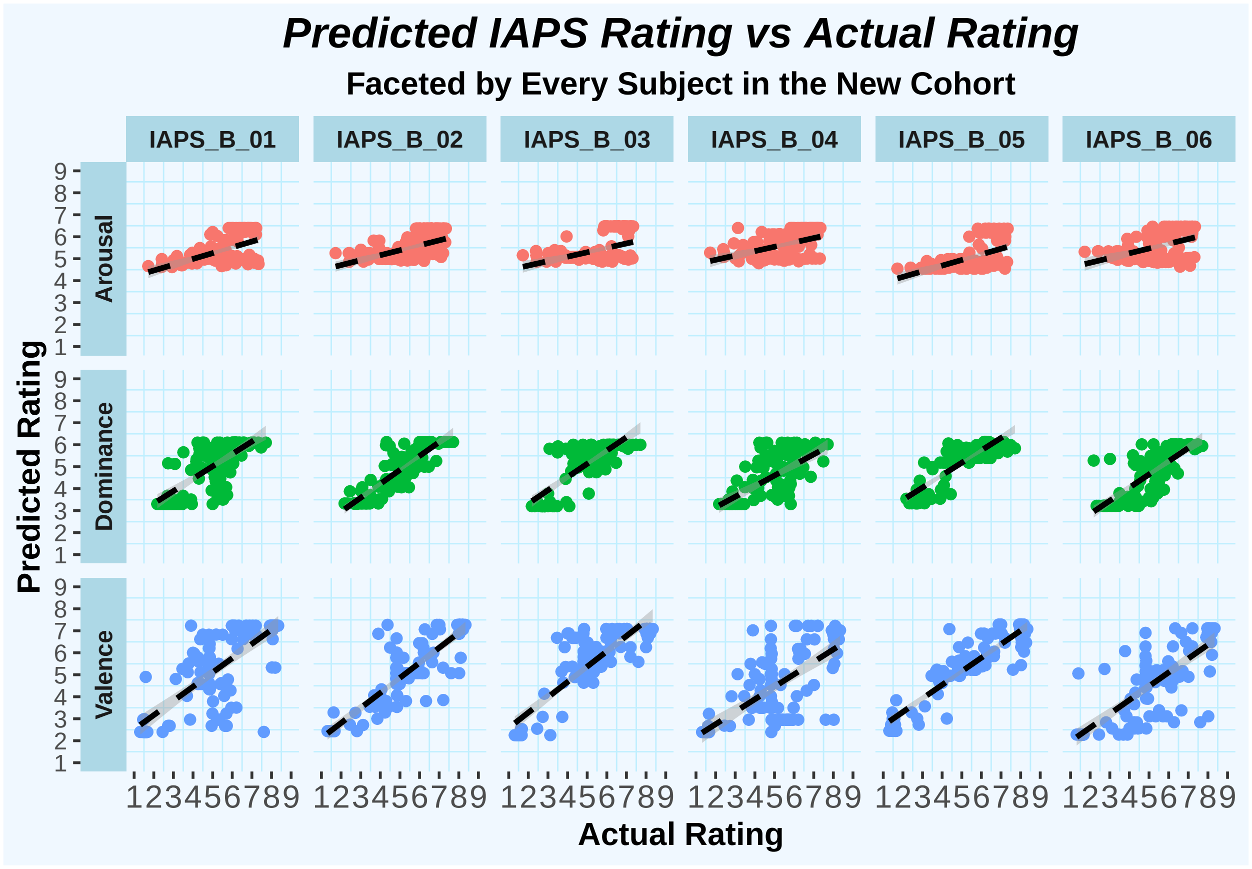
```
AroCor <- actual_predicted %>% filter(outcome == "Arousal")
DomCor <- actual_predicted %>% filter(outcome == "Dominance")
ValCor <- actual_predicted %>% filter(outcome == "Valence")
```

In order to check how accurate our model is, we can plot the actual vs. predicted outcomes and observe the trends. We do this only for the best fitting model as outlined in the above section. The first plot is an average over all 6 subjects in the new cohort. The predicted and actual ratings have an overall correlation of 0.474 when the outcome is arousal, 0.744 when the outcome is dominance, and 0.794, when the outcome is valence. The second plot shows the individual variations in these relationships.

```
#On the whole
actual_predicted %>%
  ggplot(aes(x = rating_mean, y = predicted, color = outcome)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~outcome) +
  scale_y_continuous(limits = c(1,9), breaks = c(seq(1,9))) +
  scale_x_continuous(limits = c(1,9), breaks = c(seq(1,9))) +
  geom_smooth(method = "lm", show.legend = FALSE, color = "black", linetype = "dashed") +
  labs(title = "Predicted IAPS Rating vs Actual Rating",
       y = "Predicted Rating",
       x = "Actual Rating") +
  myGGTheme +
  theme(plot.background = element_rect(fill = "aliceblue"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_line(color = "lightblue1"),
        panel.background = element_rect(fill = "aliceblue"),
        strip.background = element_rect(fill = "lightblue"),
        strip.text = element_text(face = "bold"))
```



```
#On the whole
actual_predicted %>%
  ggplot(aes(x = rating_mean, y = predicted, color = outcome)) +
  geom_point(show.legend = FALSE) +
  facet_grid(outcome ~ subject, switch = "y") +
  scale_y_continuous(limits = c(1,9), breaks = c(seq(1,9))) +
  scale_x_continuous(limits = c(1,9), breaks = c(seq(1,9))) +
  geom_smooth(method = "lm", show.legend = FALSE, color = "black", linetype = "dashed") +
  labs(title = "Predicted IAPS Rating vs Actual Rating",
       subtitle = "Faceted by Every Subject in the New Cohort",
       y = "Predicted Rating",
       x = "Actual Rating") +
  myGGTheme +
  theme(plot.background = element_rect(fill = "aliceblue"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_line(color = "lightblue1"),
        panel.background = element_rect(fill = "aliceblue"),
        strip.background = element_rect(fill = "lightblue"),
        strip.text = element_text(face = "bold"))
```



Do the prediction errors vary with rating uncertainty?

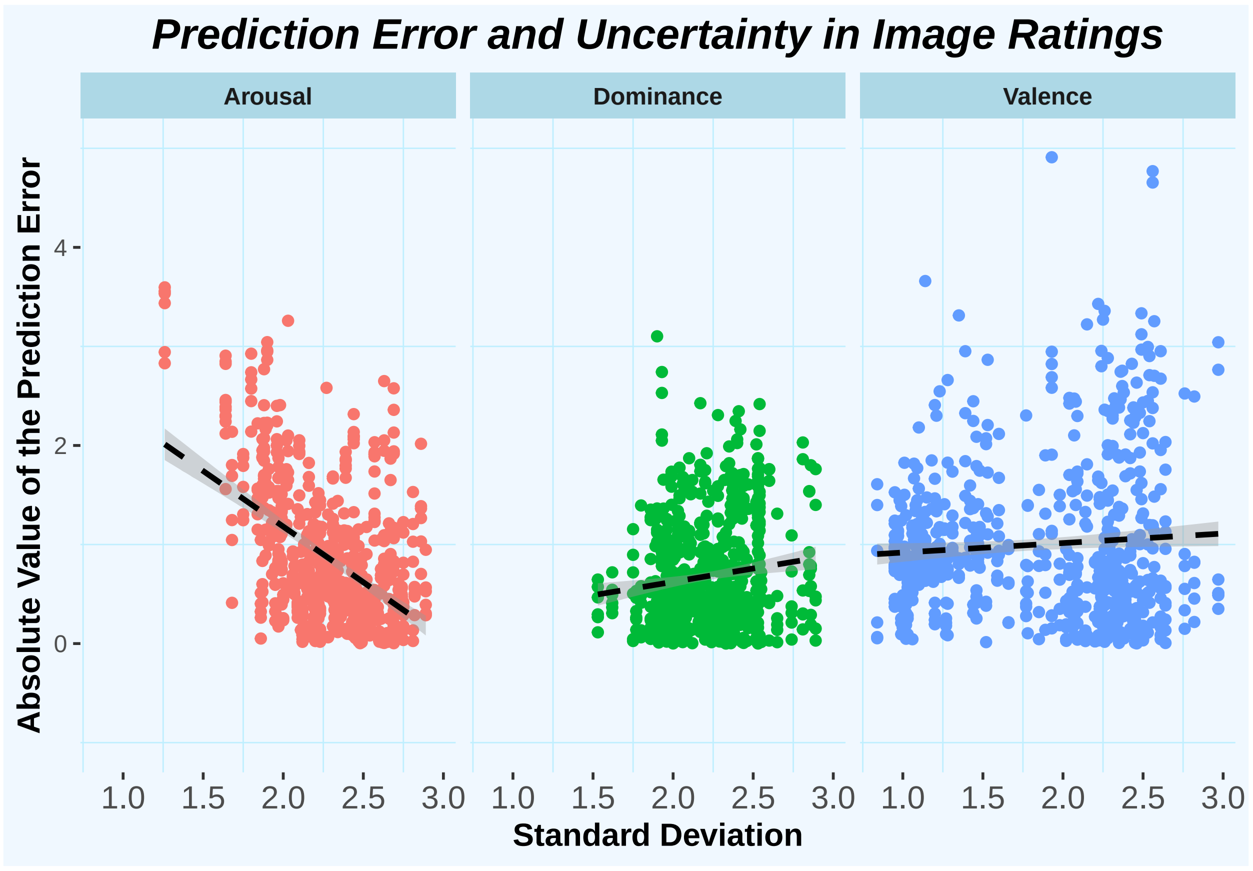
Another question to ask is whether or not the prediction errors (the difference between actual and predicted rating) varies with the uncertainty, as measured by the standard deviation, of each rating from the IAPS dataset.

```
#create a prediction error data frame that has the difference between the
#predicted and actual outcomes.
PE <- actual_predicted %>%
  mutate(PE = predicted - rating_mean,
         PE = abs(PE)) %>%
  select(PE, rating_sd, outcome)
```

```
AroCorPE <- PE %>% filter(outcome == "Arousal")
DomCorPE <- PE %>% filter(outcome == "Dominance")
ValCorPE <- PE %>% filter(outcome == "Valence")
```

As seen in the graph below, prediction errors and image rating uncertainty (standard deviation) have correlations of -0.487 when the outcome is arousal, 0.127 when the outcome is dominance, and 0.083, when the outcome is valence.

```
#Plot!
PE %>%
  ggplot(aes(x = rating_sd, y = PE, color = outcome)) +
  geom_point(show.legend = FALSE) +
  geom_smooth(method = "lm", show.legend = FALSE, color = "black", linetype = "dashed") +
  labs(y = "Absolute Value of the Prediction Error",
       x = "Standard Deviation",
       title = "Prediction Error and Uncertainty in Image Ratings") +
  facet_wrap(~outcome) +
  myGGTheme +
  scale_y_continuous(limits = c(-1,5)) +
  theme(plot.background = element_rect(fill = "aliceblue"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_line(color = "lightblue1"),
        panel.background = element_rect(fill = "aliceblue"),
        strip.background = element_rect(fill = "lightblue"),
        strip.text = element_text(face = "bold"))
```



Choice Task

Participants and Procedure

Task Description

The choice game was a 150-trial two-choice probabilistic reward task with emotionally-stimulating images from the IAPS database used as reinforcers. Each trial began with presentation of two icons on a computer screen, and participants were required to choose between them using a Logitech gaming controller. A total of 58 icons were randomly selected to be in one of six groups, with six unique icons per group (Table 1). Within each group, icons held fixed probabilities of displaying positive, neutral, or negative images as outcomes. The game was divided into three phases that the icon groups corresponded to, with 25 trails in phase one, 50 trails in phase 2, and 75 trials in phase 3 (Table 2).

Phase one held positive-only icons, where fixed probabilities of showing positive images were pre-determined per icon (i.e. 25%, 50%, and 75% vs neutral). Phase two introduced negative-only icons with similar fixed probabilities of showing negative images per icon. In these phases, each round showed pairs of positive icons or pairs of negative icons exclusively. However, in phase 3, all icons were mixed so that any round may have consisted of two positive icons, two negative icons, or one positive and one negative icon. In this phase, outcome magnitudes changed giving rise to higher-complexity positive and negative images. That is, our selected positive and negative images were not equidistant from zero, but rather varied in complexity according to phase number. The probabilities associated with each icon remained the same throughout the game, meaning the expected value of each option is altered by phase 3. We employed this phase transition to understand temporal learning effects from images that vary in reward and complexity. The IAPS images used are presented in Appendix A according to valence, dominance, and arousal.

Participants were instructed to select their preferred icon and told that different types of images would be presented dependent upon their choice (Figure 1). They were not given instructions on the relationship between icons and images or the goal of the experiment. Before and after the game, participants were asked to select one of two icons they prefer with no image outcome. This was important to determine if certain characteristics of icons (i.e. color, laterality) influenced participants' choices throughout the game or if this changed because of image outcome.

Read in the Data

Below, we read in the data and preview the raw text file output from the choice task.

```
choiceFiles <- dir_ls('~/Desktop/IAPS/RJT/choice/', glob = "*.txt.txt") #get the paths for the files th

rawChoiceData <- map(choiceFiles, ~readChoices(.x), .id = "subject")

#preview raw choice data
head(as.data.frame(rawChoiceData[[1]]), 20)
```

```
##              time
## 1  0000000000000
## 2  1594912717127
## 3  1594912724302
## 4  1594912724302
## 5  1594912724302
## 6  1594912724302
## 7  1594912728305
## 8  1594912728308
## 9  1594912733925
## 10 1594912733926
## 11 1594912734126
## 12 1594912734128
## 13 1594912736829
## 14 1594912736830
## 15 1594912736830
## 16 1594912737032
## 17 1594912737032
## 18 1594912737634
## 19 1594912740125
## 20 1594912740125
##                                                                      data
## 1                                                      Subject: IAPS_B_01
## 2                                       KEYPRESS: n, leaving intro screen
## 3                                   Game Start by TTL - leaving ready screen
## 4                                                            HOLD ... 4 secs ( for scanning )
## 5                                                                                        TR 1
## 6                                                                               starting game
## 7  ICON TO GROUP MAPPING: {"icon33":2,"icon49":6,"icon02":4,"icon07":5,"icon28":1,"icon03":3}
## 8                                                         SHOW: FORCED CHOICE INSTRUCTION SCREEN
## 9                                                         KEYPRESS: 3, Continuing to next screen
## 10                                  HOLD ... 0.2 secs - fixed transition duration, ICON ROUND 0
## 11                                                                     CLEAR TRANSITION SCREEN
## 12        SHOW: ICON SELECTION, GROUP ORDER: {"1":"icon07","2":"icon33"}, ICON ROUND 1 of 36
## 13                                KEYPRESS: 2, OPTION: 2, CHOICE: icon33, GROUP: 2, ICON ROUND 1
## 14                           HIGHLIGHT CHOICE: OPTION: 2, CHOICE: icon33, GROUP: 2, ICON ROUND 1
## 15                                      HOLD ... 0.2 secs - fixed choice duration, ICON ROUND 1
## 16                                                                 CLEAR SCREEN , ICON ROUND 1
## 17                                    HOLD ... 0.6 secs - fixed inter-trial delay, ICON ROUND 1
## 18        SHOW: ICON SELECTION, GROUP ORDER: {"1":"icon07","2":"icon28"}, ICON ROUND 2 of 36
## 19                           HIGHLIGHT CHOICE: OPTION: 2, CHOICE: icon28, GROUP: 1, ICON ROUND 2
## 20                                      HOLD ... 0.2 secs - fixed choice duration, ICON ROUND 2
```

Tidy the Data

Below, we tidy the data using the `iapsr` function `processChoiceData` and preview it. For information on how the processing was done, call `?iapsr::processChoiceData()` in the console. In summary, for each subject this function returns a dataframe 150 rows and 13 columns:

- phase: The phase the choices were made in.
- round: The round number these choices were made in, specific to the phase.
- runningRound: The round number these choices were made in (1-150), independent of phase. This is useful for plotting the percent of optimal choices over time.
- icon1: The icon presented in the first order position.
- rank1: The rank of icon1. The higher this is, the higher the expected utility of choosing icon1.
- icon2: The icon presented in the second order position.
- rank2: The rank of icon2. The higher this is, the higher the expected utility of choosing icon2.
- option: The option the subject chose. This should correspond to the icon order as presented in the task.
- choice: The icon the subject chose.
- group: The (payment weighting) group the chosen icon corresponds to.
- image: The image the subject was shown after making a choice.
- optimal: A binary variable tracking whether or not the choice was optimal. This is 1 if the rank of the icon chosen is higher than the rank of the one not chosen. It is 0 otherwise. The ranks are based on the expected utility for each icon. See getGroupInfo for that information.
- percentOptimal: The cumulative sum of optimal choices divided by the total round (1-150).

```r
finalChoiceData <- map_df(rawChoiceData, ~processChoiceData(.x), .id = "subject") %>%
  mutate(subject = str_extract(subject, "[:graph:]{9}(?=choice.txt)"))

finalChoiceData %>%
  head(10)
```

```
## # A tibble: 10 x 14
##    subject phase round runningRound icon1 rank1 icon2 rank2 option choice group
##    <chr>    <dbl> <int>        <int> <chr> <int> <chr> <int> <fct>  <fct>  <fct>
##  1 IAPS_B~     1     1            1 icon~     1 icon~     3 1      icon28 1
##  2 IAPS_B~     1     2            2 icon~     3 icon~     1 2      icon28 1
##  3 IAPS_B~     1     3            3 icon~     3 icon~     2 2      icon33 2
##  4 IAPS_B~     1     4            4 icon~     2 icon~     1 2      icon28 1
##  5 IAPS_B~     1     5            5 icon~     1 icon~     3 1      icon28 1
##  6 IAPS_B~     1     6            6 icon~     3 icon~     2 2      icon33 2
##  7 IAPS_B~     1     7            7 icon~     3 icon~     2 2      icon33 2
##  8 IAPS_B~     1     8            8 icon~     3 icon~     1 2      icon28 1
##  9 IAPS_B~     1     9            9 icon~     3 icon~     1 2      icon28 1
## 10 IAPS_B~     1    10           10 icon~     3 icon~     2 2      icon33 2
## # ... with 3 more variables: image <chr>, optimal <dbl>, percentOptimal <dbl>
```

Analyze Optimal Choices

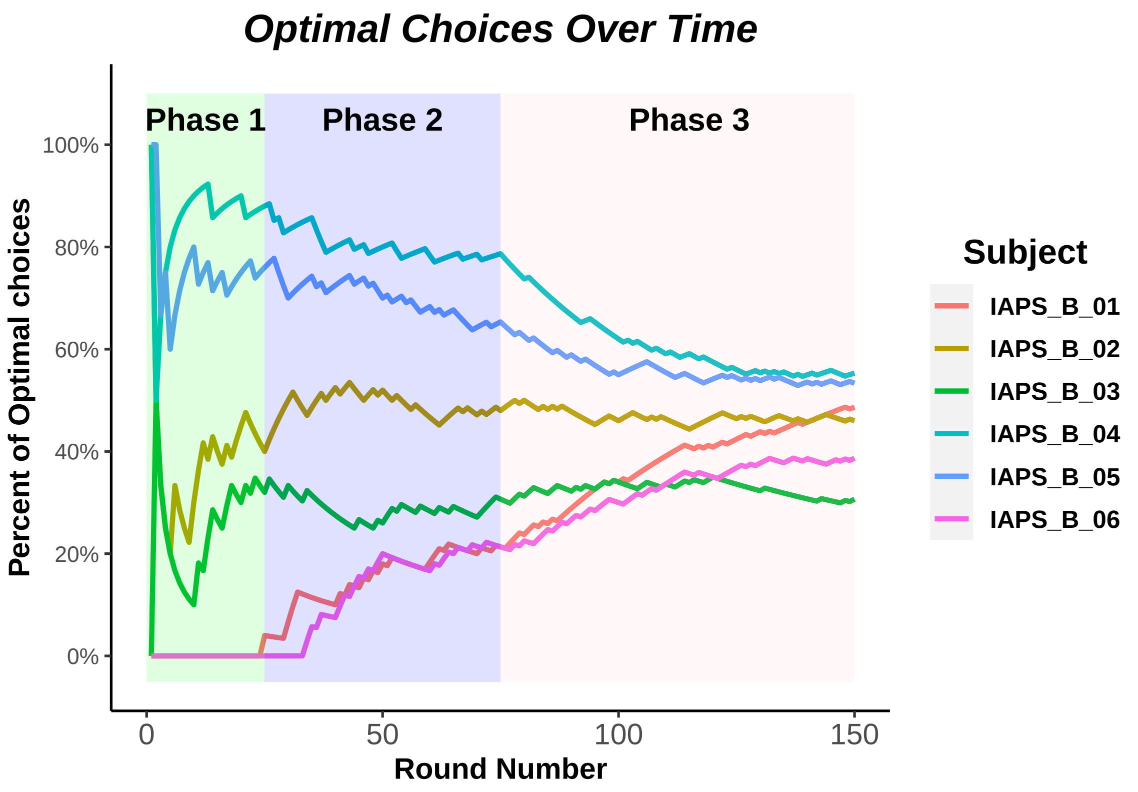
Each subject is presented with a choice between two icons. In phase 1 there are three icons all leading to positively valenced images with varying probabilities. In phase 2 and 3 there are six icons; three lead to positively valenced images and three lead to negatively valenced images with differing probabilities. For instance, icon 1 might offer a reward equivalent to \$1.00 in valence with probability 0.25 and icon 2 might offer a reward equivalent to \$1.00 in valence with probability 0.5. Thus, the expected utility of icon 2 is greater than the expected utility of icon 1. This means that whenever an individual is presented with these two icons, the optimal choice is to choose icon 2.

Using the `getGroupInfo` function from the `iapsr` package, we get the reward-probability mappings for each icon per subject and then plot the cumulative sum of optimal choices.

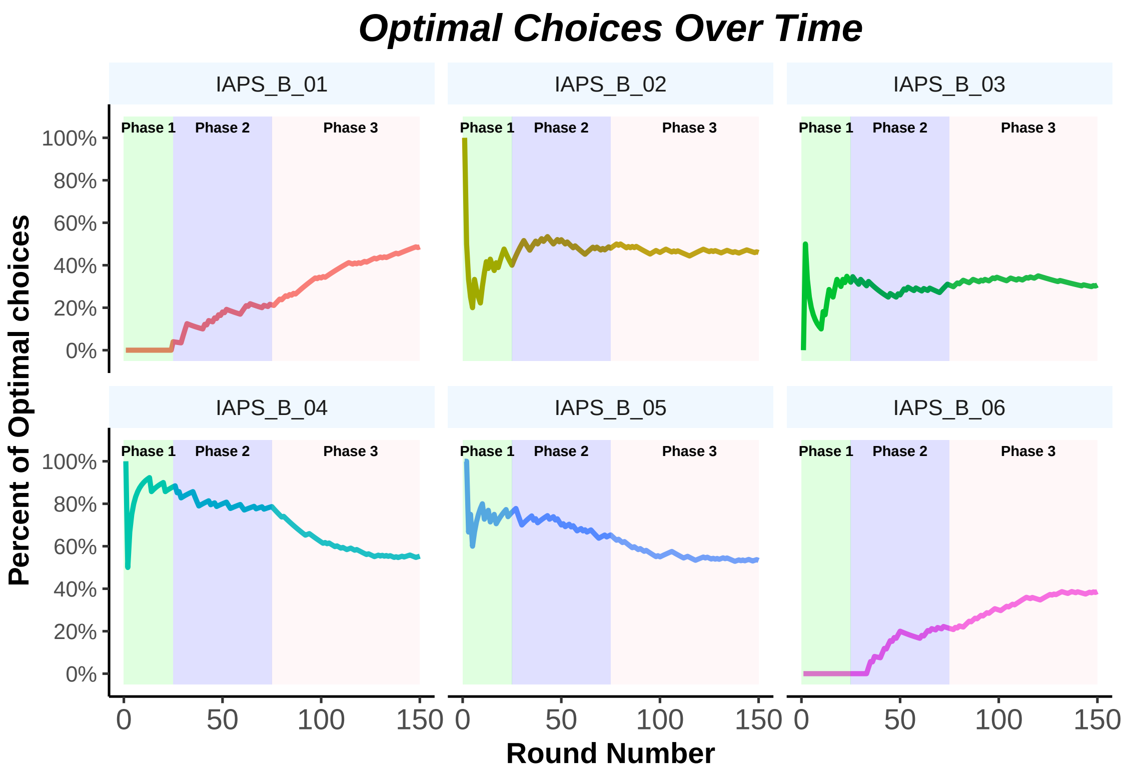
Percent Optimal Plot

Below are two plots created using the `iapsr` function `plotPercentOptimal`. They show the percent of optimal choices taken by each of the 6 subjects over the entire task. Phases are indicated. The second plot is faceted by subject.

```r
finalChoiceData %>% plotPercentOptimal(facet = FALSE)
```



```r
finalChoiceData %>% plotPercentOptimal(facet = TRUE)
```


Optimal Choices Over Time
IAPS_B_01
IAPS_B_02
IAPS_B_03
IAPS_B_04
IAPS_B_05
IAPS_B_06
Phase 1
Phase 2
Phase 3
Percent of Optimal choices
Round Number