Physical Pragmatics (observer_1)

```
# Read in the participant data.
human_0 = read_csv(file.path(human_path, "raw_data.csv"), quote="~")

# Convert the JSON string into JSON.
human_1 = lapply(na.omit(human_0$data), fromJSON)

# Extract the trial information for each participant and stack them.
human_3 = tibble()
for (p in 1:length(human_1)) {
  # Skip the NA entries (these only store the quiz information).
  if (is.null(human_1[p][[1]]$condition)) {
    next
  }

  # Trim the layout and add the participant ID back in.
  human_2 = human_1[p][[1]]$trials %>%
    as_tibble() %>%
    mutate(unique_id=human_1[p][[1]]$id,
           age=as.integer(human_1[p][[1]]$subject_information$age),
           layout=str_sub(layout, 1)) %>%
    select(unique_id, age, layout, target_0, target_1)

  # Stack the trial information for the current participant.
  human_3 = rbind(human_3, human_2)
}

# Extract the participant data from the mentalistic condition.
human_4 = human_3 %>%
  filter(target_1!=-1) %>%
  select(unique_id) %>%
  unique() %>%
  rownames_to_column(var="participant") %>%
  mutate(participant=as.numeric(participant)-1) %>%
  right_join(filter(human_3, target_1!=-1)) %>%
  select(participant, layout, target_0, target_1) %>%
  rename(reward_0=target_0, cooperation=target_1)

# Extract the participant data from the non-mentalistic condition.
human_5 = human_3 %>%
  filter(target_1==-1) %>%
  select(unique_id) %>%
  unique() %>%
  rownames_to_column(var="participant") %>%
  mutate(participant=as.numeric(participant)-1) %>%
  right_join(filter(human_3, target_1==-1)) %>%
  select(participant, layout, target_0) %>%
```

```
  rename(reward_1=target_0)

# Merge the two data partitions and extract the layout information.
human_6 = human_4 %>%
  left_join(human_5)

# Extract the layout information and re-label the natural costs.
human_7 = human_6 %>%
  separate(layout, into=c("natural_cost", "enforcer_action"), sep="_") %>%
  mutate(natural_cost=factor(natural_cost,
                             levels=c("[5 5]", "[5 7]", "[5 9]",
                                      "[7 5]", "[7 7]", "[7 9]",
                                      "[9 5]", "[9 7]", "[9 9]"),
                             labels=c("[1.25 1.25]", "[1.25 1.75]",
                                      "[1.25 2.25]", "[1.75 1.25]",
                                      "[1.75 1.75]", "[1.75 2.25]",
                                      "[2.25 1.25]", "[2.25 1.75]",
                                      "[2.25 2.25]")))

# Write the preprocessed participant data.
write_csv(human_7, file.path(human_path, "data.csv"))

# Extract the participant age information.
age = human_3 %>%
  select(unique_id, age) %>%
  unique()
```

Compute Mean Participant Judgments

```
# Read in the preprocessed participant data.
human_7 = read_csv(file.path(human_path, "data.csv"))

# Define the bootstrap functions
compute_mean = function(data, indices) {
  return(mean(data[indices]))
}

compute_bootstrap = function(data) {
  simulations = boot(data=data,
                     statistic=compute_mean,
                     R=10000)

  return(boot.ci(simulations, type="bca")$bca)
}

# Compute the bootstrapped 95% CIs for each dependent measure.
set.seed(seed)
ci = data.frame()
for (nc in unique(human_7$natural_cost)) {
  for (ea in unique(human_7$enforcer_action)) {
    # Filter the relevant data using the current natural cost and enforcer
```

```r
    # action.
    human_8 = human_7 %>%
      filter(natural_cost==nc, enforcer_action==ea)

    # Compute the bootstrapped 95% CI for each dependent measure.
    reward_0_bootstrap = compute_bootstrap(human_8$reward_0)
    cooperation_bootstrap = compute_bootstrap(human_8$cooperation)
    reward_1_bootstrap = compute_bootstrap(human_8$reward_1)
    ci = rbind(ci, data.frame(natural_cost=nc,
                              enforcer_action=ea,
                              lower_ci_reward_0=reward_0_bootstrap[4],
                              upper_ci_reward_0=reward_0_bootstrap[5],
                              lower_ci_cooperation=cooperation_bootstrap[4],
                              upper_ci_cooperation=cooperation_bootstrap[5],
                              lower_ci_reward_1=reward_1_bootstrap[4],
                              upper_ci_reward_1=reward_1_bootstrap[5]))
  }
}

# Read in the model data.
model_0 = read_csv(file.path(model_path, "predictions.csv"))

# Compute the mean participant judgments for each dependent measure and merge
# the model data.
data_0 = human_7 %>%
  group_by(natural_cost, enforcer_action) %>%
  summarize(human_reward_0=mean(reward_0), human_cooperation=mean(cooperation),
            human_reward_1=mean(reward_1)) %>%
  left_join(ci) %>%
  left_join(select(model_0, -rationality, -enforcer_reward))
```

Apply Min-Max Scaling

```r
# Extract the reward inferences from the mentalistic condition.
mentalistic_rewards = data_0 %>%
  select(natural_cost, enforcer_action, model_reward_0, human_reward_0,
         lower_ci_reward_0, upper_ci_reward_0) %>%
  rename(model=model_reward_0, participants=human_reward_0,
         lower=lower_ci_reward_0, upper=upper_ci_reward_0) %>%
  mutate(type="mentalistic_desires")

# Extract the reward inferences from the non-mentalistic condition.
nonmentalistic_rewards = data_0 %>%
  select(natural_cost, enforcer_action, model_reward_1, human_reward_1,
         lower_ci_reward_1, upper_ci_reward_1) %>%
  rename(model=model_reward_1, participants=human_reward_1,
         lower=lower_ci_reward_1, upper=upper_ci_reward_1) %>%
  mutate(type="non-mentalistic_desires")

# Merge the reward inferences from both conditions.
reward_inferences = mentalistic_rewards %>%
```

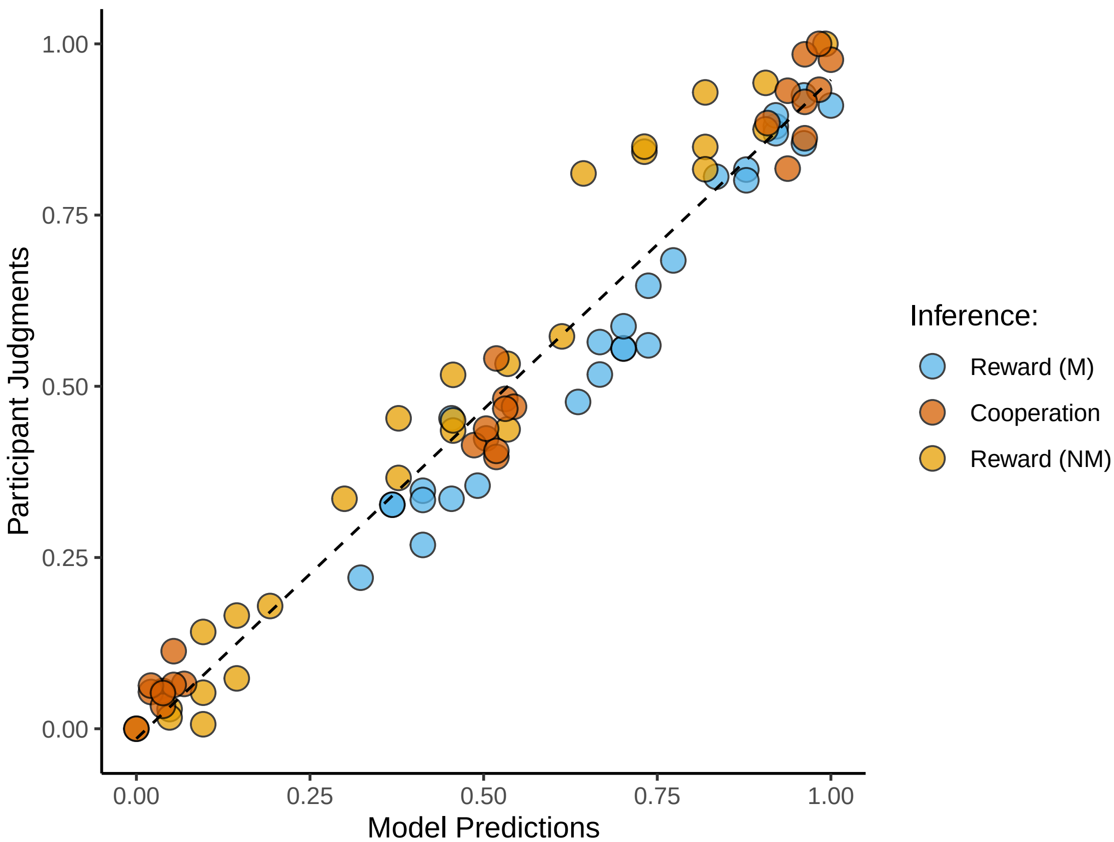
```r
  rbind(nonmentalistic_rewards)

# Apply the min-max scaling to the reward inferences.
model_min = min(reward_inferences$model)
model_max = max(reward_inferences$model) - model_min
human_min = min(reward_inferences$participants)
human_max <- max(reward_inferences$participants) - human_min
data_1 = reward_inferences %>%
  mutate(model=(model-model_min)/model_max,
         participants=(participants-human_min)/human_max,
         lower=(lower-human_min)/human_max,
         upper=(upper-human_min)/human_max)

# Extract the cooperation inferences from the mentalistic condition.
cooperation_inferences = data_0 %>%
  select(natural_cost, enforcer_action, model_cooperation, human_cooperation,
         lower_ci_cooperation, upper_ci_cooperation) %>%
  rename(model=model_cooperation, participants=human_cooperation,
         lower=lower_ci_cooperation, upper=upper_ci_cooperation) %>%
  mutate(type="cooperation")

# Apply the min-max scaling to the cooperation inferences.
model_min = min(cooperation_inferences$model)
model_max = max(cooperation_inferences$model) - model_min
human_min = min(cooperation_inferences$participants)
human_max <- max(cooperation_inferences$participants) - human_min
data_2 = cooperation_inferences %>%
  mutate(model=(model-model_min)/model_max,
         participants=(participants-human_min)/human_max,
         lower=(lower-human_min)/human_max,
         upper=(upper-human_min)/human_max)

# Merge the min-max-scaled data.
data_3 = data_1 %>%
  rbind(data_2)
```

Analysis of Replication Results

Here we plot mean participant judgments (N=80, M=34.01 years, SD=13.29 years) against our model's predictions in both conditions jointly (a replication of `observer_0`).

```r
# Plot the model predictions and mean participant judgments in both conditions.
plot_0 = data_3 %>%
  ggplot(aes(model, participants)) +
  geom_point(aes(fill=type), alpha=0.75, pch=21, size=4) +
  geom_smooth(method="lm", se=FALSE, color="black", linetype="dashed",
              size=0.5) +
  theme_classic() +
  theme(aspect.ratio=1.0) +
  xlab("Model Predictions") +
  ylab("Participant Judgments") +
  scale_fill_manual(name="Inference:",
```

```
                        limits=c("mentalistic_desires", "cooperation",
                                 "non-mentalistic_desires"),
                        labels=c("Reward (M)", "Cooperation", "Reward (NM)"),
                        values=c(color_palette[3], color_palette[7],
                                 color_palette[2]))
plot_0
```




```
# Define the bootstrap functions.
compute_cor = function(data, indices) {
  return(cor(data$model[indices], data$participants[indices],
             method="pearson"))
}

compute_bootstrap = function(data) {
  simulations = boot(data=data,
                     statistic=compute_cor,
                     R=10000)

  return(boot.ci(simulations, type="bca")$bca)
}

# Compute the correlation.
cor_0 = cor(data_3$model, data_3$participants)

# Compute the bootstrapped 95% CI of the correlation.
```

```
set.seed(seed)
cor_0_bootstrap = compute_bootstrap(data_3)
cor_0_ci = data.frame(
  lower=cor_0_bootstrap[4],
  upper=cor_0_bootstrap[5]
)
```

Our model predictions yield a correlation of r=0.98 (95% CI: 0.96-0.98) with participant judgments.