Physical Pragmatics (observer_0)

Preprocessing

Re-Label Participant Data

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

# Extract the participant data from the mentalistic condition.
human_1 = human_0 %>%
  filter(target_1!=-1) %>%
  select(workerid, layout, target_0, target_1) %>%
  rename(reward_0=target_0, cooperation=target_1)

# Extract the participant data from the non-mentalistic condition.
human_2 = human_0 %>%
  filter(target_1==-1) %>%
  select(workerid, layout, target_0) %>%
  rename(reward_1=target_0) %>%
  mutate(workerid=workerid-length(unique(workerid)))

# Merge the two data partitions and extract the layout information.
human_3 = human_1 %>%
  left_join(human_2)

# Extract the layout information and re-label the natural costs.
human_4 = human_3 %>%
  mutate(layout=substr(layout, 2, 12)) %>%
  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_4, file.path(human_path, "data.csv"))

# Read in the participant age information.
age = read_csv(file.path(human_path, "subject_information.csv")) %>%
  select(workerid, age) %>%
  mutate(age=as.numeric(substr(age, 2, 3)))
```

Compute Mean Participant Judgments

```r
# Read in the preprocessed participant data.
human_4 = 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_4$natural_cost)) {
  for (ea in unique(human_4$enforcer_action)) {
    # Filter the relevant data using the current natural cost and enforcer
    # action.
    human_5 = human_4 %>%
      filter(natural_cost==nc, enforcer_action==ea)

    # Compute the bootstrapped 95% CI for each dependent measure.
    reward_0_bootstrap = compute_bootstrap(human_5$reward_0)
    cooperation_bootstrap = compute_bootstrap(human_5$cooperation)
    reward_1_bootstrap = compute_bootstrap(human_5$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_4 %>%
  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

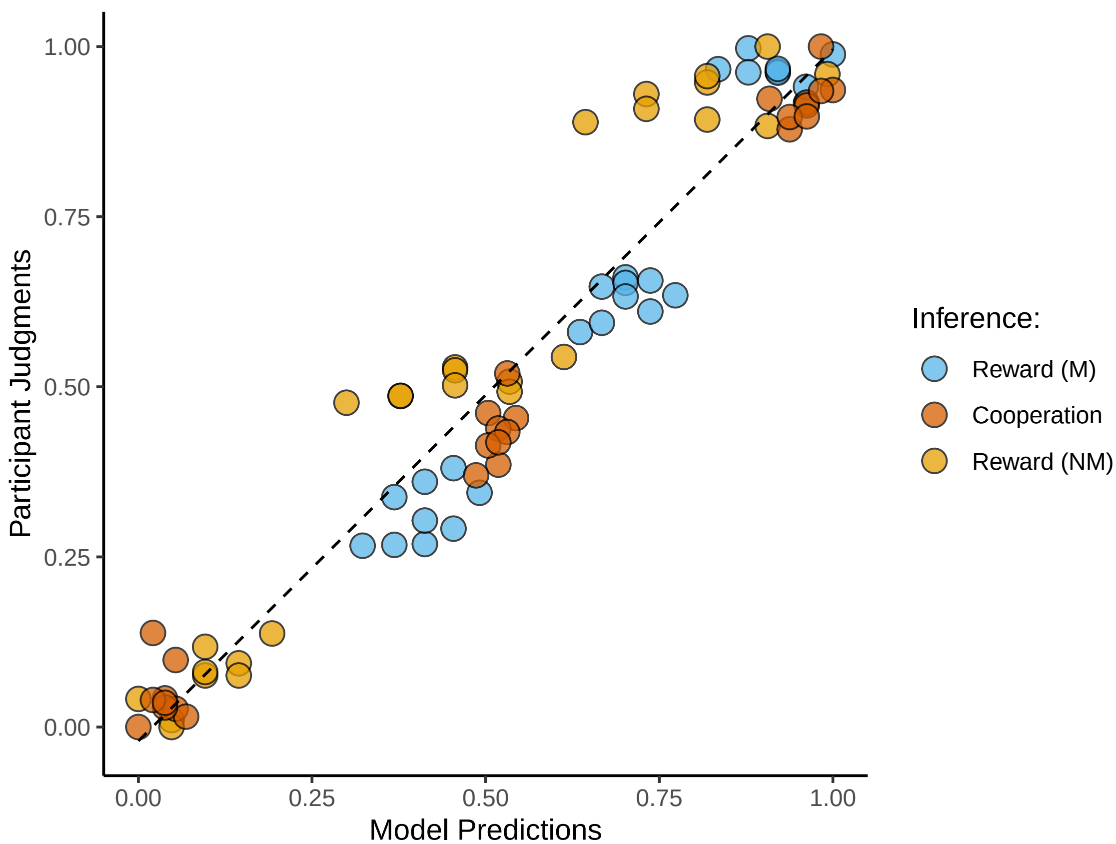
```
# Extract the reward inferences from the mentalistic condition.
mentalistic_rewards = data_0 %>%
  select(natural_cost, enforcer_action, model_reward_0, human_reward_0,
         lower_ci_reward_0, upper_ci_reward_0) %>%
  rename(model=model_reward_0, participants=human_reward_0,
         lower=lower_ci_reward_0, 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 %>%
  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 Main Results

Here we plot mean participant judgments (N=80, M=34.81 years, SD=10.31 years) against our model's predictions in both conditions jointly.

```
# 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
```



```r
# 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=0.97 (95% CI: 0.95-0.98) with participant judgments.

Closer Look at Example Trials

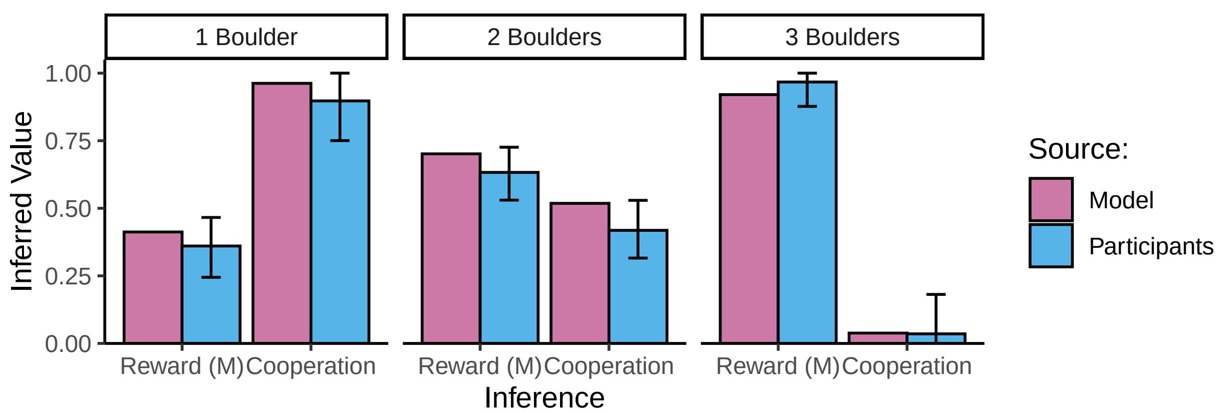
Here we plot the data from a subset of our trials in greater detail.

Mentalistic Condition Examples

We first begin by plotting the data from the *mentalistic* condition.

```r
# Extract the data for a given natural cost and clamp the bootstrapped 95% CIs.
data_4 = data_3 %>%
  filter(natural_cost=="[2.25 2.25]") %>%
  gather(source, value, model, participants) %>%
  mutate(lower=ifelse(source=="model", NA, lower),
         lower=ifelse(lower<0, 0, lower),
         upper=ifelse(source=="model", NA, upper),
         upper=ifelse(upper>1, 1, upper))

# Plot the data from the example trials in the mentalistic condition.
plot_1 = data_4 %>%
  filter(type %in% c("mentalistic_desires", "cooperation")) %>%
  ggplot(aes(x=type, y=value, fill=source)) +
  geom_bar(stat="identity", position=position_dodge(), color="black") +
  geom_errorbar(aes(ymin=lower, ymax=upper), position=position_dodge(0.9),
                width=0.3) +
  facet_wrap(~factor(enforcer_action,
                     levels=c("[1 0]", "[2 0]", "[3 0]"),
                     labels=c("1 Boulder", "2 Boulders", "3 Boulders"))) +
```

```
  theme_classic() +
  theme(aspect.ratio=1.0) +
  scale_x_discrete(name="Inference",
                   limits=c("mentalistic_desires", "cooperation"),
                   labels=c("Reward (M)", "Cooperation")) +
  scale_y_continuous(name="Inferred Value",
                     expand=c(0, 0),
                     limits=c(0, 1.05)) +
  scale_fill_manual(name="Source:",
                    limits=c("model", "participants"),
                    labels=c("Model", "Participants"),
                    values=c(color_palette[8], color_palette[3]))
plot_1
```

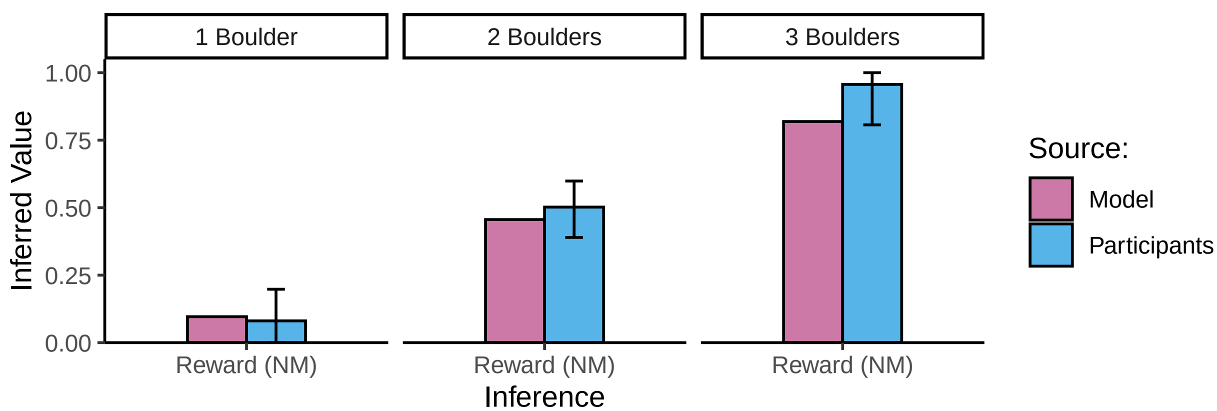


Non-Mentalistic Condition Examples

Now we plot the data from the *non-mentalistic* condition.

```
# Plot the data from the example trials in the non-mentalistic condition.
plot_2 = data_4 %>%
  filter(type %in% c("non-mentalistic_desires")) %>%
  ggplot(aes(x=type, y=value, fill=source)) +
  geom_bar(stat="identity", position=position_dodge(), width=0.5,
           color="black") +
  geom_errorbar(aes(ymin=lower, ymax=upper), position=position_dodge(0.5),
                width=0.15) +
```

```
  facet_wrap(~factor(enforcer_action,
                     levels=c("[1 0]", "[2 0]", "[3 0]"),
                     labels=c("1 Boulder", "2 Boulders", "3 Boulders"))) +
  theme_classic() +
  theme(aspect.ratio=1.0) +
  scale_x_discrete(name="Inference",
                   limits=c("non-mentalistic_desires"),
                   labels=c("Reward (NM)")) +
  scale_y_continuous(name="Inferred Value",
                     expand=c(0, 0),
                     limits=c(0, 1.05)) +
  scale_fill_manual(name="Source:",
                    limits=c("model", "participants"),
                    labels=c("Model", "Participants"),
                    values=c(color_palette[8], color_palette[3]))
plot_2
```



Decider Cost Lesion

Here we evaluate mean participant judgments (N=80, M=34.81 years, SD=10.31 years) against our model's performance when deciders no longer have an understanding of costs–that is, they navigate the world with their rewards as their only consideration.

Preprocessing

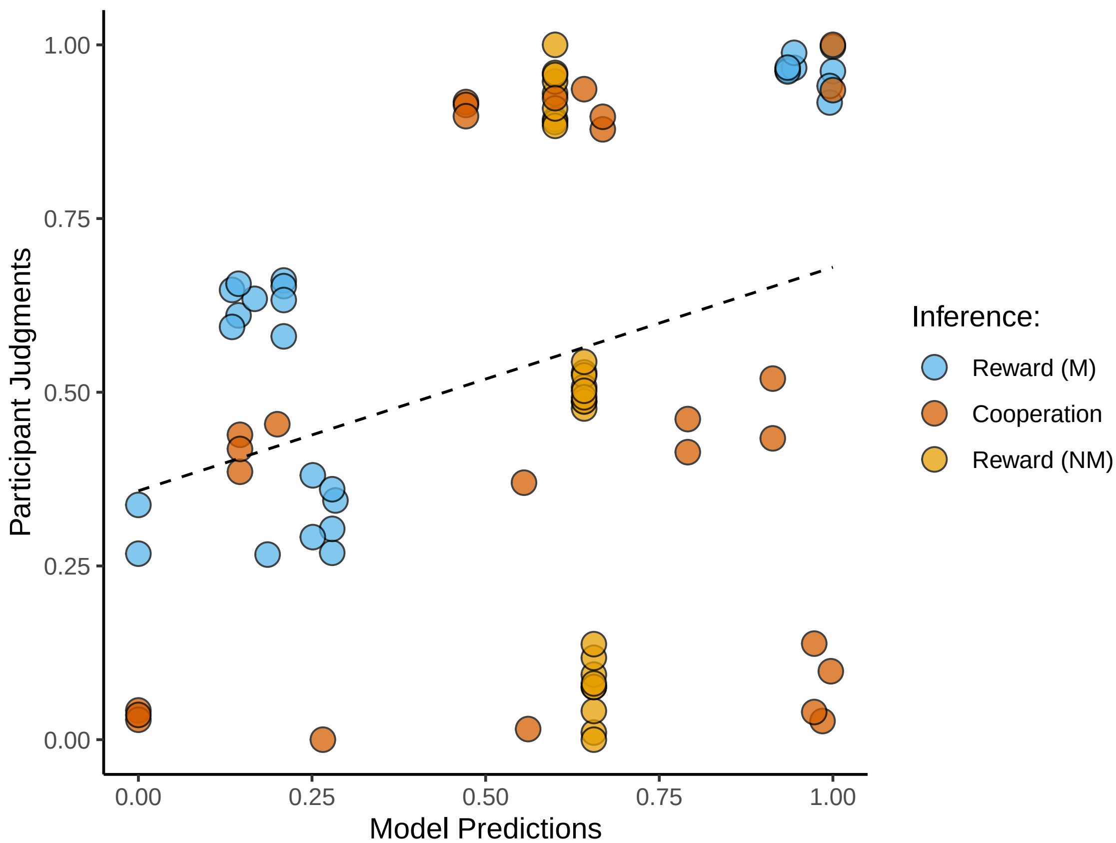
Apply Min-Max Scaling

```
# Read in the model predictions of the "decider cost lesion" model.
model_1 = read_csv(file.path(model_path,
                             "decider_cost_lesion/predictions.csv"))

# Compute the mean participant judgments for each dependent measure and merge
# the model data.
data_5 = human_4 %>%
  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_1, -rationality, -enforcer_reward))

# Extract the reward inferences from the mentalistic condition.
mentalistic_rewards = data_5 %>%
  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_5 %>%
  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 %>%
  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_6 = 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_5 %>%
  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_7 = 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_8 = data_6 %>%
  rbind(data_7)
```

Analysis of "Decider Cost Lesion" Model Results

```
# Plot the model predictions and mean participant judgments in both conditions
# for the "decider cost lesion" model.
plot_3 = data_8 %>%
  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_3
```




```
# 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_1 = cor(data_8$model, data_8$participants)

# Compute the bootstrapped 95% CI for the correlation.
set.seed(seed)
cor_1_bootstrap = compute_bootstrap(data_8)
cor_1_ci = data.frame(
  lower=cor_1_bootstrap[4],
  upper=cor_1_bootstrap[5]
)
```

Our “decider cost lesion” model predictions yield a correlation of r=0.29 (95% CI: 0.08-0.47) with participant judgments.

Enforcer Cost Lesion

Now we evaluate mean participant judgments (N=80, M=34.81 years, SD=10.31 years) against our model's performance when enforcers no longer have an understanding of costs–that is, they manipulate the environment with their rewards as their only consideration.

Preprocessing

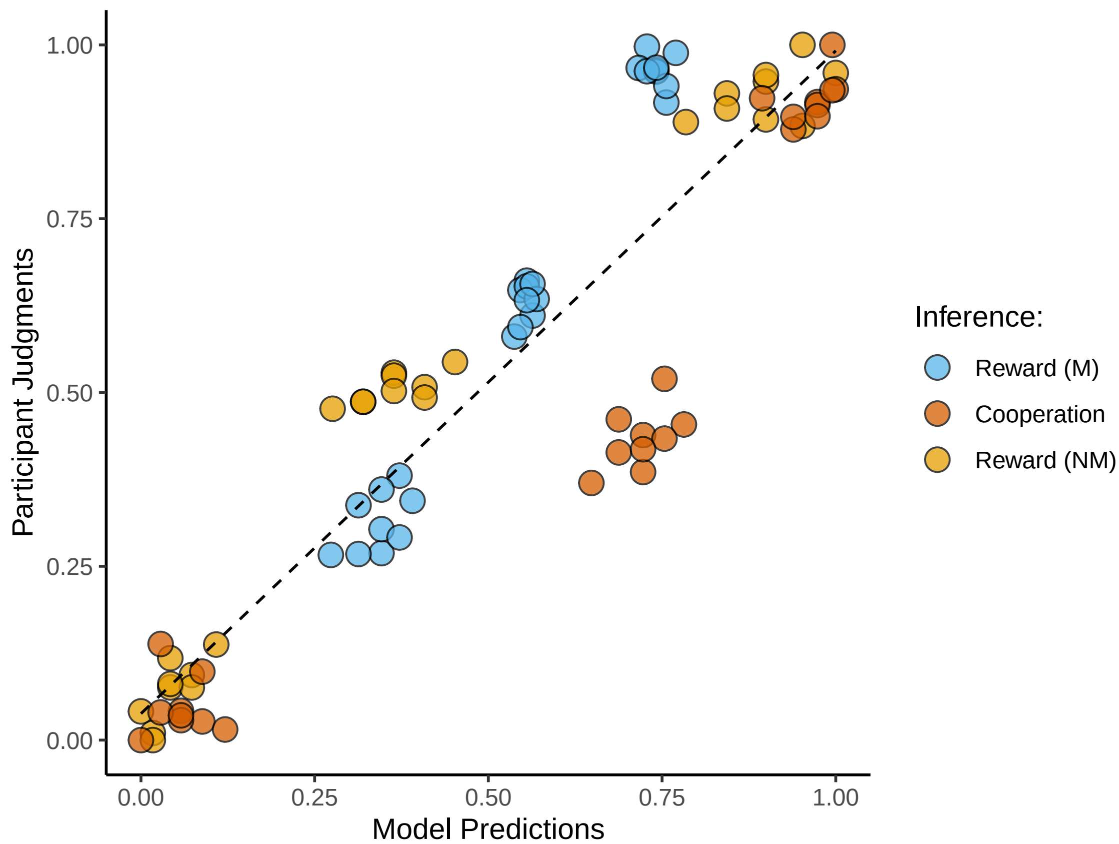
Apply Min-Max Scaling

```
# Read in the model predictions of the "enforcer cost lesion" model.
model_2 = read_csv(file.path(model_path,
                             "enforcer_cost_lesion/predictions.csv"))

# Compute the mean participant judgments for each dependent measure and merge
# the model data.
data_9 = human_4 %>%
  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_2, -rationality, -enforcer_reward))

# Extract the reward inferences from the mentalistic condition.
mentalistic_rewards = data_9 %>%
  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_9 %>%
  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 %>%
  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_10 = 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_9 %>%
  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_11 = 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_12 = data_10 %>%
  rbind(data_11)
```

Analysis of Enforcer Cost Lesion Results

```
# Plot the model predictions and mean participant judgments in both conditions
# for the "enforcer cost lesion" model.
plot_4 = data_12 %>%
  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_4
```




```r
# 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_2 = cor(data_12$model, data_12$participants)

# Compute the bootstrapped 95% CI for the correlation.
set.seed(seed)
cor_2_bootstrap = compute_bootstrap(data_12)
cor_2_ci = data.frame(
  lower=cor_2_bootstrap[4],
  upper=cor_2_bootstrap[5]
)
```

Our “enforcer cost lesion” model predictions yield a correlation of r=0.91 (95% CI: 0.86-0.94) with participant judgments.

Full Model vs. Lesioned Models

```
# Define the boostrap functions.
compute_cor_diff = function(data, indices) {
  cor_full_model = cor(data$full_model[indices],
                       data$participants[indices],
                       method="pearson")
  cor_lesioned_model = cor(data$lesioned_model[indices],
                           data$participants[indices],
                           method="pearson")
  return(cor_full_model-cor_lesioned_model)
}

# Define the bootstrap function to simulate the data.
compute_bootstrap = function(data) {
  simulations = boot(data=data,
                     statistic=compute_cor_diff,
                     R=10000)

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

# Merge the full model data with the data from the "decider cost lesion" model.
data_13 = data_3 %>%
  rename(full_model=model) %>%
  left_join(rename(data_8, lesioned_model=model))

# Compute the correlation difference between the full model and the "decider
# cost lesion" model.
cor_diff_0 = cor_0 - cor_1

# Compute the bootstrapped 95% CI for the correlation difference.
set.seed(seed)
cor_diff_0_bootstrap = compute_bootstrap(data_13)
cor_diff_0_ci = data.frame(
  lower=cor_diff_0_bootstrap[4],
  upper=cor_diff_0_bootstrap[5]
)

# Merge the full model data with the data from the "enforcer cost lesion" model.
data_14 = data_3 %>%
  rename(full_model=model) %>%
  left_join(rename(data_12, lesioned_model=model))

# Compute the correlation difference between the full model and the "enforcer
# cost lesion" model.
cor_diff_1 = cor_0 - cor_2

# Compute the bootstrapped 95% CI for the correlation difference.
set.seed(seed)
cor_diff_1_bootstrap = compute_bootstrap(data_14)
cor_diff_1_ci = data.frame(
  lower=cor_diff_1_bootstrap[4],
  upper=cor_diff_1_bootstrap[5]
```

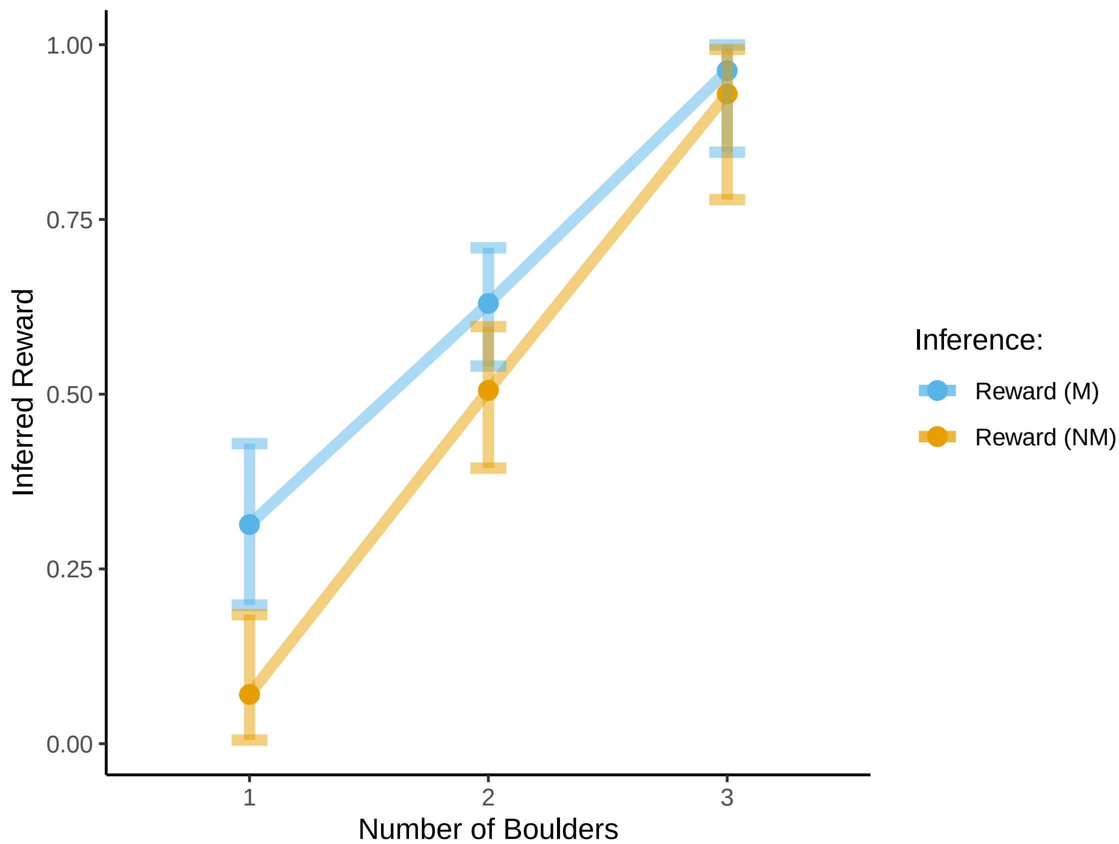
```
)
```

When compared against the "decider cost lesion" model, our model has a correlation difference of Δr=0.68 (95% CI: 0.49-0.89). When compared against the "enforcer cost lesion" model, our model as a correlation difference of Δr=0.06 (95% CI: 0.03-0.1).

Comparing Reward Inferences

Here we conduct an additional analyses comparing the difference in mean participant reward inferences (N=80, M=34.81 years, SD=10.31 years) across conditions.

```
# Filter out the cooperation data, clamp the bootstrapped 95% CIs, and
# compute average rewards per condition and enforcer action.
data_15 = data_3 %>%
  filter(type!="cooperation") %>%
  mutate(lower=ifelse(lower<0, 0, lower),
         upper=ifelse(upper>1, 1, upper)) %>%
  group_by(type, enforcer_action) %>%
  summarize(mean_model=mean(model),
            mean_participants=mean(participants),
            mean_lower=mean(lower),
            mean_upper=mean(upper))

# Plot the reward comparison averaged per condition and enforcer action.
plot_5 = data_15 %>%
  ggplot(aes(x=enforcer_action, y=mean_participants, group=type, color=type)) +
  geom_point(size=3) +
  geom_line(size=2, alpha=0.5) +
  geom_errorbar(aes(ymin=mean_lower, ymax=mean_upper), width=0.15, size=2,
                alpha=0.5) +
  theme_classic() +
  theme(aspect.ratio=1.0) +
  scale_x_discrete(name="Number of Boulders",
                   limits=c("[1 0]", "[2 0]", "[3 0]"),
                   labels=c(1, 2, 3)) +
  ylab("Inferred Reward") +
  scale_colour_manual(name="Inference:",
                      limits=c("mentalistic_desires",
                               "non-mentalistic_desires"),
                      labels=c("Reward (M)", "Reward (NM)"),
                      values=c(color_palette[3], color_palette[2]))
plot_5
```



We first quantify the mean participant reward differences across conditions by computing a t-test over each boulder count.

```
# Filter the reward inferences across in the mentalistic condition, averaged
# over the natural cost and enforcer action.
data_16 = data_3 %>%
  filter(type=="mentalistic_desires")

# Filter the reward inferences across in the non-mentalistic condition,
# averaged over the natural cost and enforcer action.
data_17 = data_3 %>%
  filter(type=="non-mentalistic_desires")

# Compute a t-test across conditions for one boulder.
t.test(filter(data_16, enforcer_action=="[1 0]")$participants,
       filter(data_17, enforcer_action=="[1 0]")$participants,
       alternative="two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  filter(data_16, enforcer_action == "[1 0]")$participants and filter(data_17, enforcer_action =
## t = 11.533, df = 15.951, p-value = 3.774e-09
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1984210 0.2878201
## sample estimates:
```

```
##  mean of x  mean of y
## 0.31342790 0.07030733
```

```
# Compute a t-test across conditions for two boulders.
t.test(filter(data_16, enforcer_action=="[2 0]")$participants,
       filter(data_17, enforcer_action=="[2 0]")$participants,
       alternative="two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  filter(data_16, enforcer_action == "[2 0]")$participants and filter(data_17, enforcer_action =
## t = 10.22, df = 15.153, p-value = 3.408e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.09855046 0.15043299
## sample estimates:
## mean of x mean of y
## 0.6299291 0.5054374
```

```
# Compute a t-test across conditions for three boulders.
t.test(filter(data_16, enforcer_action=="[3 0]")$participants,
       filter(data_17, enforcer_action=="[3 0]")$participants,
       alternative="two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  filter(data_16, enforcer_action == "[3 0]")$participants and filter(data_17, enforcer_action =
## t = 2.1595, df = 13.084, p-value = 0.04995
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  7.938303e-06 6.628048e-02
## sample estimates:
## mean of x mean of y
## 0.9628369 0.9296927
```

Next, we quantify the observed reward differences by computing a linear mixed-effects regression (predicting individual participant judgments as a function of enforcer action and condition).

```
# Filter and preprocess the relevant individual participant judgments.
data_18 = human_4 %>%
  select(-cooperation) %>%
  rename(mentalistic_desires=reward_0, nonmentalistic_desires=reward_1) %>%
  gather(condition, reward, mentalistic_desires, nonmentalistic_desires) %>%
  mutate(workerid=ifelse(condition=="nonmentalistic_desires", workerid+40,
                         workerid),
         enforcer_action=as.numeric(str_sub(enforcer_action, 2, 2)))

# Compute a linear mixed-effects regression predicting individual participant
# judgments as a function of enforcer action and condition.
model_3 = lmer(formula=reward ~ enforcer_action*condition +
               (1+condition|workerid), data=data_18)
summary(model_3)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
```

```
## Formula: reward ~ enforcer_action * condition + (1 + condition | workerid)
##    Data: data_18
##
## REML criterion at convergence: -1965.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.9730 -0.5614  0.0625  0.5856  3.9789
##
## Random effects:
##  Groups   Name                         Variance Std.Dev. Corr
##  workerid (Intercept)                  0.01264  0.1124
##           conditionnonmentalistic_desires 0.03543  0.1882   -0.67
##  Residual                              0.02079  0.1442
## Number of obs: 2160, groups:  workerid, 80
##
## Fixed effects:
##                                                  Estimate Std. Error
## (Intercept)                                     3.155e-01  2.123e-02
## enforcer_action                                 1.908e-01  5.373e-03
## conditionnonmentalistic_desires                -2.018e-01  3.292e-02
## enforcer_action:conditionnonmentalistic_desires  6.168e-02  7.599e-03
##                                                        df t value Pr(>|t|)
## (Intercept)                                     7.034e+01  14.861  < 2e-16 ***
## enforcer_action                                 2.078e+03  35.504  < 2e-16 ***
## conditionnonmentalistic_desires                 1.202e+02  -6.132 1.14e-08 ***
## enforcer_action:conditionnonmentalistic_desires  2.078e+03   8.117 8.06e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) enfrc_ cndtn_
## enforcr_ctn -0.506
## cndtnnnmnt_ -0.645  0.326
## enfrcr_ct:_  0.358 -0.707 -0.462
```

```
# Print the coefficients separately for readability.
summary(model_3)$coefficients
```

```
##                                                    Estimate  Std. Error
## (Intercept)                                      0.31551852 0.021230612
## enforcer_action                                  0.19076389 0.005372955
## conditionnonmentalistic_desires                 -0.20184259 0.032916559
## enforcer_action:conditionnonmentalistic_desires  0.06168056 0.007598506
##                                                         df   t value
## (Intercept)                                       70.33871 14.861489
## enforcer_action                                 2077.99967 35.504461
## conditionnonmentalistic_desires                  120.17362 -6.131947
## enforcer_action:conditionnonmentalistic_desires 2077.99967  8.117458
##                                                     Pr(>|t|)
## (Intercept)                                      2.164824e-23
## enforcer_action                                 3.228756e-216
## conditionnonmentalistic_desires                  1.142711e-08
## enforcer_action:conditionnonmentalistic_desires  8.064194e-16
```

Finally, we compute two linear regressions (each predicting mean participant judgments as a function of enforcer action and the natural costs) to examine the predictive power of the enforcer action versus the natural cost in each condition.

```
# Compute a linear regression predicting mean participant judgments as a
# function of enforcer action and the natural costs for the mentalistic
# condition.
data_19 = data_3 %>%
  filter(type=="mentalistic_desires") %>%
  mutate(enforcer_action=as.numeric(substr(enforcer_action, 2, 2))) %>%
  separate(natural_cost, into=c("natural_cost_0", "natural_cost_1"),
           sep=" ") %>%
  mutate(natural_cost_0=as.numeric(gsub("[", "", natural_cost_0, fixed=TRUE)),
         natural_cost_1=as.numeric(gsub("]", "", natural_cost_1, fixed=TRUE)))
model_4 = glm(participants~enforcer_action+natural_cost_0+natural_cost_1,
              family="gaussian", data=data_19)
summary(model_4)
```

```
##
## Call:
## glm(formula = participants ~ enforcer_action + natural_cost_0 +
##     natural_cost_1, family = "gaussian", data = data_19)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.048684  -0.030150   0.001812   0.020063   0.064129
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.032628   0.041024  -0.795    0.435
## enforcer_action  0.324704   0.007596  42.746   <2e-16 ***
## natural_cost_0   0.021844   0.015192   1.438    0.164
## natural_cost_1  -0.011206   0.015192  -0.738    0.468
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.001038625)
##
##     Null deviance: 1.924395  on 26  degrees of freedom
## Residual deviance: 0.023888  on 23  degrees of freedom
## AIC: -103.19
##
## Number of Fisher Scoring iterations: 2
```

```
# Print the coefficients separately for readability.
summary(model_4)$coefficients
```

```
##                    Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)     -0.03262805 0.041023856 -0.7953434 4.345455e-01
## enforcer_action  0.32470449 0.007596142 42.7459739 2.021440e-23
## natural_cost_0   0.02184397 0.015192284  1.4378333 1.639491e-01
## natural_cost_1  -0.01120567 0.015192284 -0.7375898 4.682227e-01
```

```
# Compute a linear regression predicting mean participant judgments as a
# function of enforcer action and the natural costs for the non-mentalistic
# condition.
```

```r
data_20 = data_3 %>%
  filter(type=="non-mentalistic_desires") %>%
  mutate(enforcer_action=as.numeric(substr(enforcer_action, 2, 2))) %>%
  separate(natural_cost, into=c("natural_cost_0", "natural_cost_1"),
           sep=" ") %>%
  mutate(natural_cost_0=as.numeric(gsub("[", "", natural_cost_0, fixed=TRUE)),
         natural_cost_1=as.numeric(gsub("]", "", natural_cost_1, fixed=TRUE)))
model_5 = glm(participants~enforcer_action+natural_cost_0+natural_cost_1,
              family="gaussian", data=data_20)
summary(model_5)
```

```
##
## Call:
## glm(formula = participants ~ enforcer_action + natural_cost_0 +
##     natural_cost_1, family = "gaussian", data = data_20)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.063704  -0.010264   0.003176   0.016592   0.045753
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.384299   0.037082 -10.363 3.88e-10 ***
## enforcer_action  0.429693   0.006866  62.580  < 2e-16 ***
## natural_cost_0   0.046478   0.013733   3.384  0.00255 **
## natural_cost_1  -0.031206   0.013733  -2.272  0.03273 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.0008486199)
##
##     Null deviance: 3.357065  on 26  degrees of freedom
## Residual deviance: 0.019518  on 23  degrees of freedom
## AIC: -108.65
##
## Number of Fisher Scoring iterations: 2
```

```r
# Print the coefficients separately for readability.
summary(model_5)$coefficients
```

```
##                    Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)     -0.38429866 0.037082051 -10.363468 3.881185e-10
## enforcer_action  0.42969267 0.006866262  62.580294 3.389114e-27
## natural_cost_0   0.04647754 0.013732523   3.384487 2.552658e-03
## natural_cost_1  -0.03120567 0.013732523  -2.272392 3.272659e-02
```