Physical Pragmatics (symbols_1)

Preprocessing

```
# Preprocess and merge each partition of the participant data.
data_5 = tibble()
num_participants = 0
for (partition_path in list.files(data_path, pattern="data",
                                  full.names=TRUE)) {
  # Read in the current partition of the participant data.
  data_0 = read_csv(file.path(partition_path, "raw_data.csv"))

  # Combine the trial and exclusion columns.
  data_1 = data_0 %>%
      gather(trial_type, num, trial_num, exclusion_num) %>%
      na.omit() %>%
      mutate(trial=gsub("num", "", paste(trial_type, num, sep=""))) %>%
      select(-trial_type, -num) %>%
      mutate(target=gsub("\"", "", target, fixed=TRUE))

  # Read in and merge the participant age information.
  data_2 = read_csv(file.path(partition_path, "subject_information.csv")) %>%
    select(workerid, first_object, second_object, age) %>%
    mutate(age=as.numeric(gsub("\"", "", age))) %>%
    right_join(data_1)

  # Import the setup information to know which side the low-cost door was on.
  setup_0 = read_tsv(file.path(partition_path, "trial_information.tsv")) %>%
    select(workerid, Answer.setup)

  # Remove the quotes, backslashes, and braces.
  setup_1 = setup_0 %>%
    mutate(Answer.setup=gsub("\\", "", Answer.setup, fixed=TRUE)) %>%
    mutate(Answer.setup=gsub("\"", "", Answer.setup, fixed=TRUE)) %>%
    mutate(Answer.setup=gsub("{", "", Answer.setup, fixed=TRUE)) %>%
    mutate(Answer.setup=gsub("}", "", Answer.setup, fixed=TRUE))

  # Extract all relevant trial information.
  setup_2 = setup_1 %>%
    separate(Answer.setup,
             into=c("condition", "first_side", "second_side",
                    "first_object", "second_object", "doors"),
             sep=",") %>%
    mutate(condition=gsub("condition:", "", condition),
           first_side=gsub("first_side:", "", first_side),
           second_side=gsub("second_side:", "", second_side),
```

```r
           first_object=gsub("first_object:", "", first_object),
           second_object=gsub("second_object:", "", second_object),
           doors=gsub("doors:", "", doors)) %>%
    mutate(first_object=gsub("fishbowl", "string", first_object),
           second_object=gsub("fishbowl", "string", second_object))

  # Merge the trial information with the participant data and remake the
  # participant column.
  data_3 = data_2 %>%
    select(workerid, first_object, second_object, age) %>%
    unique() %>%
    rownames_to_column(var="participant") %>%
    mutate(participant=as.numeric(participant)+num_participants) %>%
    right_join(left_join(data_2, setup_2)) %>%
    select(-workerid)

  # Update the current number of participants.
  num_participants = num_participants + length(unique(data_3$participant))

  # Exclude participants that did not say the modified door is harder to walk
  # through on the low-cost trial.
  data_4 = data_3 %>%
    filter(trial=="exclusion_1", target==first_side) %>%
    select(participant) %>%
    left_join(data_3)

  # Write the current partition of the preprocessed data.
  write_csv(data_4, file.path(partition_path, "data.csv"))

  # Merge the current partition of participant data with the rest.
  data_5 = data_5 %>%
    rbind(data_4)
}
```

Compute Door Endorsements

Here we compute the participant door endorsements (N=160, M=36.13 years, SD=10.9 years) for each of our trials.

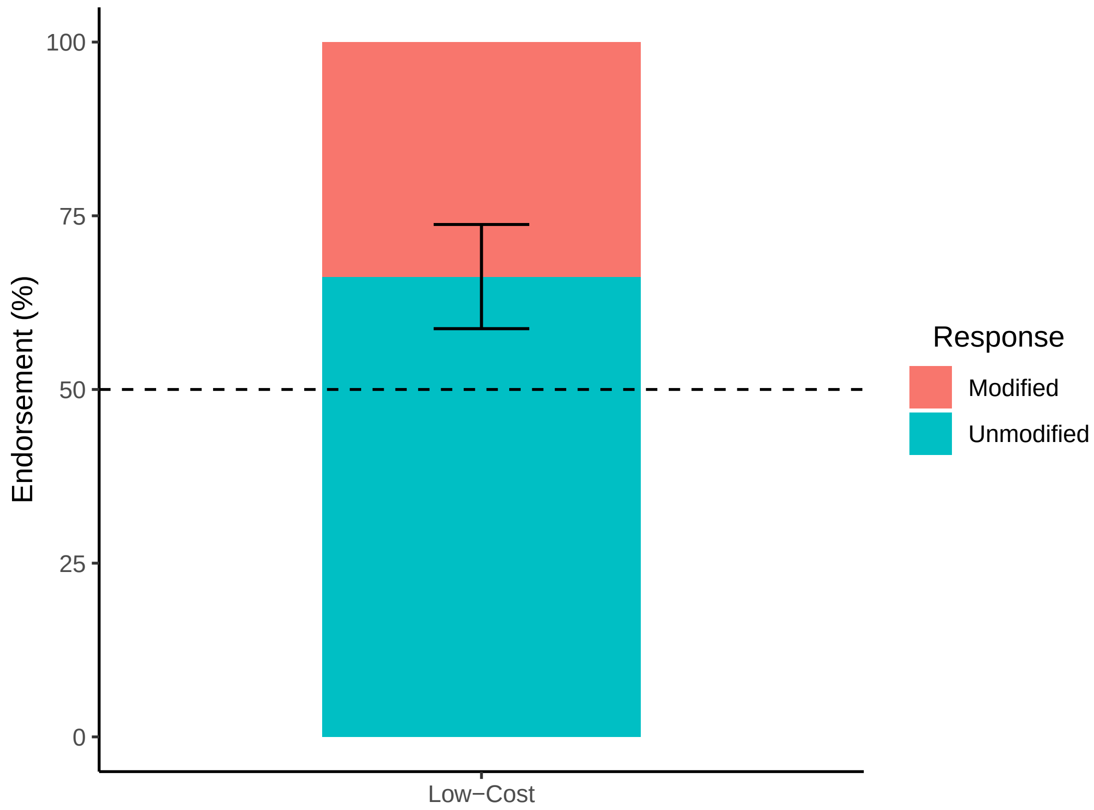
`decider_0` Replication

First, we'll analyze the low-cost trial, which serves as a replication of `decider_0`. Since the low-cost trials are identical across conditions, we analyze them jointly.

```r
# Read in and merge the preprocessed participant data.
data_5 = tibble()
for (partition_path in list.files(data_path, pattern="data",
                                  full.names=TRUE)) {
  # Read in the current partition of the preprocessed participant data.
  data_5 = data_5 %>%
    rbind(read_csv(file.path(partition_path, "data.csv")))
```

```r
}

# Filter the data from the low-cost trial.
data_6 = data_5 %>%
  filter(trial=="trial_1") %>%
  mutate(target=as.numeric(target))

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

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

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

# Compute the bootstrapped 95% CIs for the participant endorsement of the
# unmodified door in the low-cost trial.
set.seed(seed)
bootstrap_data = compute_bootstrap(data_6)
ci = data.frame(trial="trial_1",
                lower_ci=bootstrap_data[4]*100,
                upper_ci=bootstrap_data[5]*100)

# Compute the participant endorsement for the unmodified door in the low-cost
# trial.
data_7 = data_6 %>%
  group_by(trial) %>%
  summarize(unmodified=sum(target)/n()*100) %>%
  mutate(modified=100-unmodified) %>%
  left_join(ci) %>%
  gather(door, endorsement, unmodified, modified)

# Plot the participant endorsement for the unmodified door in the low-cost
# trial, which is a replication of `decider_0`.
plot_0 = data_7 %>%
  ggplot(aes(x=trial, y=endorsement, fill=door)) +
  geom_bar(stat="identity", width=0.5) +
  geom_errorbar(aes(ymin=lower_ci, ymax=upper_ci), width=0.15) +
  geom_hline(yintercept=50, linetype="dashed", color="black") +
  theme_classic() +
  theme(aspect.ratio=1.0,
        legend.title=element_text(hjust=0.5)) +
  scale_x_discrete(name="",
                   limits=c("trial_1"),
                   labels=c("Low-Cost")) +
  ylab("Endorsement (%)") +
  scale_fill_discrete(name="Response",
                      limits=c("modified", "unmodified"),
```

```
                              labels=c("Modified", "Unmodified"))
plot_0
```



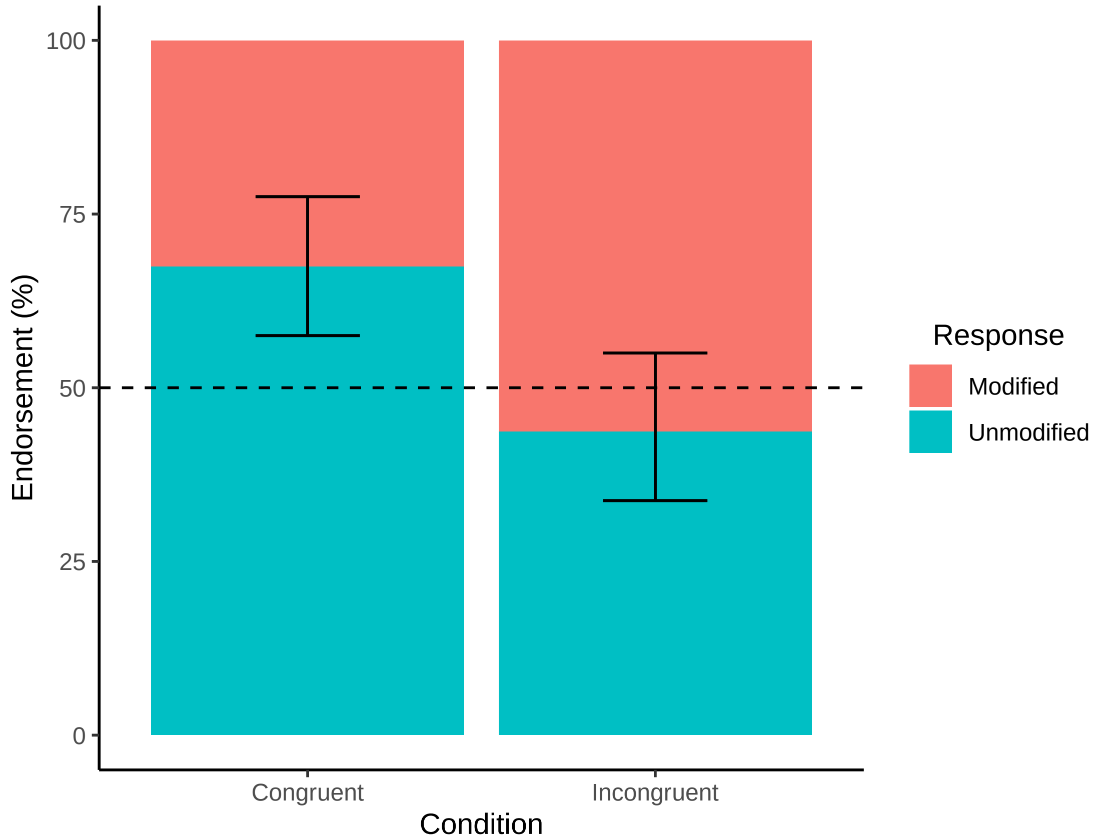

```
# Compute a binomial test over the joint data.
# NOTE: Computing a two-tailed binomial test for "conservativeness".
binom.test(x=sum(data_6$target),
           n=length(data_6$target),
           p=0.5, alternative="two.sided")
```

```
##
##  Exact binomial test
##
## data:  sum(data_6$target) and length(data_6$target)
## number of successes = 106, number of trials = 160, p-value = 4.8e-05
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.5836060 0.7352473
## sample estimates:
## probability of success
##                 0.6625
```

Main Results

Now, we'll analyze our main results, comparing the participant door endorsements per condition in the symbol trial.

```r
# Filter the data from the symbol trial.
data_8 = data_5 %>%
  filter(trial=="trial_3") %>%
  mutate(target=as.numeric(target))

# Compute the bootstrapped 95% CIs for the participant endorsement of the
# unmodified door in the symbol trial.
set.seed(seed)
ci = data.frame()
for (c in unique(data_8$condition)) {
  # Filter the relevant data for the current condition.
  data_9 = data_8 %>%
    filter(condition==c)

  # Compute the bootstrapped 95% CIs.
  bootstrap_data = compute_bootstrap(data_9)
  ci = rbind(ci, data.frame(trial="trial_3",
                            condition=c,
                            lower_ci=bootstrap_data[4]*100,
                            upper_ci=bootstrap_data[5]*100))
}

# Compute the participant endorsement for the unmodified door in the symbol
# trial.
data_10 = data_8 %>%
  group_by(trial, condition) %>%
  summarize(unmodified=sum(target)/n()*100) %>%
  mutate(modified=100-unmodified) %>%
  left_join(ci) %>%
  gather(door, endorsement, unmodified, modified)

# Plot the participant endorsement for the unmodified door in the symbol trial.
plot_1 = data_10 %>%
  ggplot(aes(x=condition, y=endorsement, fill=door)) +
  geom_histogram(stat="identity") +
  geom_errorbar(aes(ymin=lower_ci, ymax=upper_ci), width=0.3) +
  geom_hline(yintercept=50, linetype="dashed", color="black") +
  theme_classic() +
  theme(aspect.ratio=1.0,
        legend.title=element_text(hjust=0.5)) +
  scale_x_discrete(name="Condition",
                   limits=c("congruent", "incongruent"),
                   labels=c("Congruent", "Incongruent")) +
  ylab("Endorsement (%)") +
  scale_fill_discrete(name="Response",
                      limits=c("modified", "unmodified"),
                      labels=c("Modified", "Unmodified"))
plot_1
```




```r
# Compute a mixed-effects logistic regression with objects as random
# intercepts.
mem_0 = glmer(target~condition+(1|first_object)+(1|second_object),
              data=data_8, family="binomial")
summary(mem_0)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: target ~ condition + (1 | first_object) + (1 | second_object)
##    Data: data_8
##
##      AIC      BIC   logLik deviance df.resid
##    212.1    224.4   -102.0    204.1      156
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.1686 -0.8703  0.5005  0.7870  1.9779
##
## Random effects:
##  Groups        Name        Variance Std.Dev.
##  first_object  (Intercept) 0.0000   0.0000
##  second_object (Intercept) 0.4605   0.6786
## Number of obs: 160, groups:  first_object, 8; second_object, 8
##
```

```
## Fixed effects:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)             0.8025     0.3489   2.300  0.02143 *
## conditionincongruent  -1.0862     0.3494  -3.109  0.00188 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr)
## cndtnncngrn -0.532
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## boundary (singular) fit: see ?isSingular
```

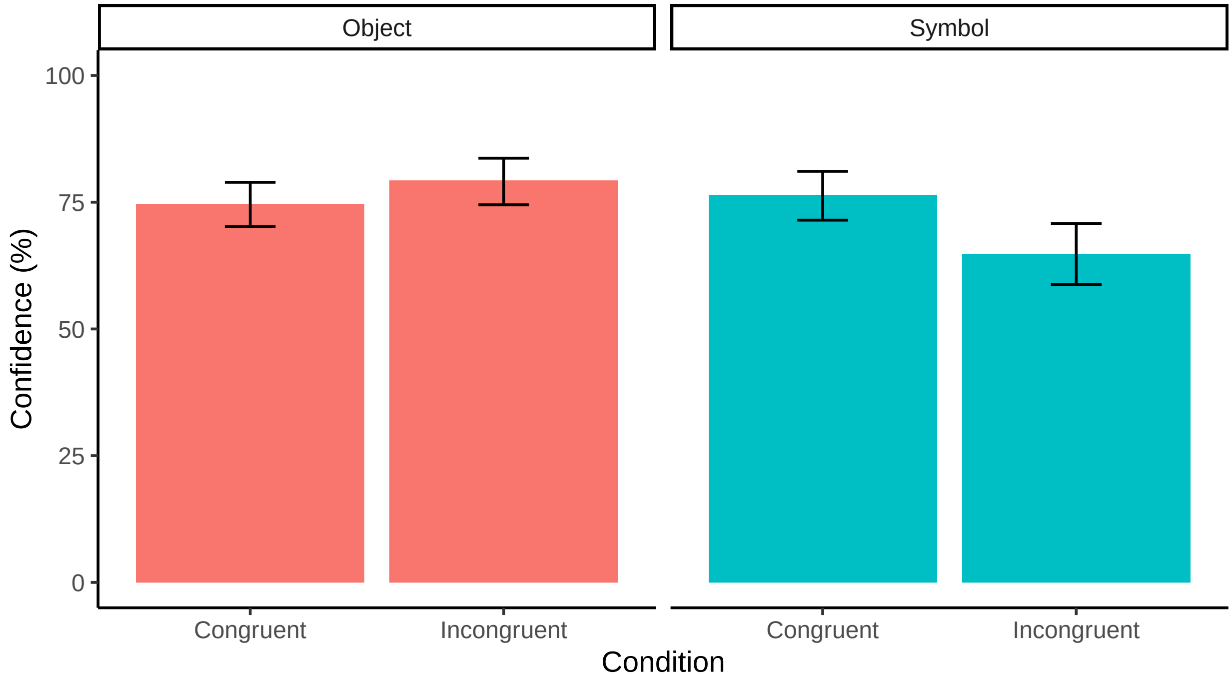
```r
# Compute a binomial test on the alternative hypothesis that the participant
# endorsement for the symbol door is above chance for an initial
# low-cost door.
# NOTE: Computing a two-sided binomial test for "conservativeness".
data_11 = data_8 %>%
  filter(condition=="congruent")
binom.test(x=sum(data_11$target), n=length(data_11$target), p=0.5,
           alternative="two.sided")
```

```
##
##  Exact binomial test
##
## data:  sum(data_11$target) and length(data_11$target)
## number of successes = 54, number of trials = 80, p-value = 0.002325
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.5610630 0.7755404
## sample estimates:
## probability of success
##                  0.675
```

```r
data_12 = data_8 %>%
  filter(condition=="incongruent")
binom.test(x=sum(data_12$target), n=length(data_12$target), p=0.5,
           alternative="two.sided")
```

```
##
##  Exact binomial test
##
## data:  sum(data_12$target) and length(data_12$target)
## number of successes = 35, number of trials = 80, p-value = 0.3143
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.3267640 0.5530024
## sample estimates:
## probability of success
##                 0.4375
```

```r
# Compute the bootstrapped 95% CIs for participant confidence in both trials
# across both conditions.
set.seed(seed)
ci = data.frame()
for (c in unique(data_5$condition)) {
  for (t in c("trial_2", "trial_4")) {
    # Filter the relevant data for the current condition and trial.
    data_13 = data_5 %>%
      filter(condition==c, trial==t) %>%
      mutate(target=as.numeric(target))

    # Compute the bootstrapped 95% CIs.
    bootstrap_data = compute_bootstrap(data_13)
    ci = rbind(ci, data.frame(condition=c,
                              trial=t,
                              lower_ci=bootstrap_data[4],
                              upper_ci=bootstrap_data[5]))
  }
}

# Compute the participant confidence in both trials across both conditions.
data_14 = data_5 %>%
  filter(trial %in% c("trial_2", "trial_4")) %>%
  mutate(target=as.numeric(target)) %>%
  group_by(condition, trial) %>%
  summarize(confidence=mean(target)*100) %>%
  ungroup() %>%
  left_join(ci) %>%
  mutate(lower_ci=lower_ci*100, upper_ci=upper_ci*100)

# Plot the participant confidence in both trials across both conditions.
plot_2 = data_14 %>%
  ggplot(aes(x=condition, y=confidence, fill=trial)) +
  geom_bar(stat="identity") +
  theme_classic() +
  theme(aspect.ratio=1.0,
        legend.position="none") +
  facet_wrap(~factor(trial,
                     levels=c("trial_2", "trial_4"),
                     labels=c("Object", "Symbol"))) +
  geom_errorbar(aes(ymin=lower_ci, ymax=upper_ci), width=0.2) +
  ylim(c(0, 100)) +
  scale_x_discrete(name="Condition",
                   limits=c("congruent", "incongruent"),
                   labels=c("Congruent", "Incongruent")) +
  ylab("Confidence (%)")
plot_2
```




```
# Set up the bootstrap functions.
compute_difference = function(data, indices) {
  congruent_data = data[indices,] %>%
    filter(condition=="congruent", trial=="trial_4") %>%
    mutate(target=as.numeric(target))
  incongruent_data = data[indices,] %>%
    filter(condition=="incongruent", trial=="trial_4") %>%
    mutate(target=as.numeric(target))
  return(mean(congruent_data$target)-mean(incongruent_data$target))
}

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

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

# Compute the boostrapped 95% CI for the difference between the mean
# participant confidence judgments in the symbol trial.
set.seed(seed)
bootstrap_data = compute_bootstrap(data_5)
difference_lower_ci = bootstrap_data[4] * 100
difference_upper_ci = bootstrap_data[5] * 100
```

```r
# Compute the difference between the mean participant confidence judgments in
# the symbol trial.
difference =
  filter(data_14, condition=="congruent", trial=="trial_4")$confidence -
  filter(data_14, condition=="incongruent", trial=="trial_4")$confidence
print(paste("Difference CI (lower):", round(difference_lower_ci, 2), sep=" "))
```

```
## [1] "Difference CI (lower): 3.8"
```

```r
print(paste("Difference CI (upper):", round(difference_upper_ci, 2), sep=" "))
```

```
## [1] "Difference CI (upper): 19.23"
```

```r
print(paste("Difference:", difference, sep=" "))
```

```
## [1] "Difference: 11.625"
```

```r
# Compute a U test between participant confidence in the symbol trial of both
# conditions.
data_15 = data_5 %>%
  filter(condition=="congruent", trial=="trial_4") %>%
  mutate(target=as.numeric(target))
data_16 = data_5 %>%
  filter(condition=="incongruent", trial=="trial_4") %>%
  mutate(target=as.numeric(target))
wilcox.test(data_15$target, data_16$target)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data_15$target and data_16$target
## W = 3978, p-value = 0.007916
## alternative hypothesis: true location shift is not equal to 0
```

```r
# Set up the bootstrap functions.
compute_mean = function(data, indices) {
  return(mean(data[indices]))
}

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

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

# Filter the confidence ratings for the object trial in both conditions.
data_17 = data_5 %>%
  filter(trial=="trial_2") %>%
```

```r
  mutate(target=as.numeric(target))

# Compute the bootstrapped 95% CIs for the average participant confidence in
# the object trial across both conditions.
set.seed(seed)
bootstrap_data = compute_bootstrap(data_17)
confidence_lower_ci = bootstrap_data[4] * 100
confidence_upper_ci = bootstrap_data[5] * 100

# Print the average participant confidence and 95% CIs in the object trial
# across both conditions.
confidence = mean(data_17$target) * 100
print(paste("Average Confidence CI (lower):", round(confidence_lower_ci, 2),
            sep=" "))
```

```
## [1] "Average Confidence CI (lower): 73.71"
```

```r
print(paste("Average Confidence CI (upper):", round(confidence_upper_ci, 2),
            sep=" "))
```

```
## [1] "Average Confidence CI (upper): 80.03"
```

```r
print(paste("Average Confidence:", confidence, sep=" "))
```

```
## [1] "Average Confidence: 76.95625"
```

```r
# Compute the agreement between participant responses across trials in each
# condition.
data_18 = data_5 %>%
  filter(trial %in% c("trial_1", "trial_3")) %>%
  mutate(target=as.numeric(target)) %>%
  spread(trial, target) %>%
  group_by(condition) %>%
  summarize(agreement=sum(trial_1==trial_3)/n(), total=n()) %>%
  ungroup() %>%
  select(-total)
data_18
```

```
## # A tibble: 2 x 2
##   condition   agreement
##   <chr>           <dbl>
## 1 congruent       0.912
## 2 incongruent     0.675
```