Phystables Containment v1 Analysis

Reading in Data

Raw data is created using a `json_to_csv.py` script in the same directory as this file which reads in json data for each individual participant and coalesces it into a column-wise csv where each row is an individual trial by a participant.

The primary columns we're interested in for this analysis are each participant's response time and accuracy across trials. We compute and modify these variables in the processing section.

```
data = suppressMessages(read_csv("phystables_env_raw.csv"))
glimpse(data)
```

```
## Observations: 20,224
## Variables: 17
## $ subjID         <chr> "user_1548179758602", "user_1548179758602", "us...
## $ logts          <dbl> 1.54818e+12, 1.54818e+12, 1.54818e+12, 1.54818e...
## $ trialname      <chr> "contain_sc2_var_l3_complex_l3", "contain_sc3_v...
## $ trialindex     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, ...
## $ trialstructure <chr> "{'Name': 'contain_sc2_var_l3_complex_l3', 'Dim...
## $ goaldistances  <chr> "[{'name': 'red', 'goal': {'top': 130, 'left': ...
## $ numwalls       <int> 6, 3, 5, 9, 3, 4, 3, 13, 7, 7, 13, 9, 5, 10, 3,...
## $ walldistances  <chr> "[{'wall': {'top': 200, 'left': 20, 'bottom': 4...
## $ numbounces     <int> 5, 12, 5, 49, 18, 3, 1, 7, 31, 8, 5, 2, 10, 11,...
## $ ballstartpos   <chr> "{'x': 110, 'y': 500}", "{'x': 400, 'y': 30}", ...
## $ ballwaitpos    <chr> "{'x': 162.5, 'y': 570.19999999998}", "{'x': 31...
## $ trialtarget    <chr> "red", "red", "green", "green", "green", "red",...
## $ targetswitched <chr> "False", "False", "True", "False", "True", "Tru...
## $ usertarget     <chr> "green", "red", "red", "green", "green", "green...
## $ responsetime   <int> 1560, 2753, 1525, 1344, 1205, 1108, 788, 677, 2...
## $ simtime        <int> 1528, 2120, 911, 3653, 4794, 1178, 653, 1270, 6...
## $ score          <int> -10, 32, -10, 60, 64, -10, 78, 82, 101, 106, 92...
```

Total participants:

```
length(levels(as.factor(data$subjID)))
```

```
## [1] 79
```

Data Cleanup

Response Time Cleanup

No Response Trials

For trials in which participants did not make a guess, their responsetime retains its initialization value of -1. We set it to 10s * 1000 ms / s, since the trial could not last longer than 10s. This allows us to be able to compute log10 response time during Data Processing.

```
data.no.resp = data %>%
  filter(responsetime == -1)
table(data.no.resp$subjID)
```

```
##
## user_1547856241568 user_1548187033593
##                  1                  3
```

```
data$responsetime[data$responsetime == -1] = 10 * 1000
```

Immediate Response Trials

There are a handful of trials in which participants had a response time of 0. This will also be problematic when we try to calculate the log10 response time, so we set it to 1.

Is this the right way to handle this?

```
data.immediate.resp = data %>%
  filter(responsetime == 0)
table(data.immediate.resp$subjID)
```

```
##
## user_1547758784529 user_1547852362121 user_1547856183346
##                  1                  1                  1
## user_1548179758602 user_1548183285262 user_1548273717903
##                  1                  1                  5
## user_1548276006558 user_1548363860525 user_1548374111673
##                  1                  2                  2
## user_1548435094148 user_1548435604423 user_1548456718025
##                  1                  2                  1
## user_1548457510128
##                  1
```

```
data$responsetime[data$responsetime == 0] = 1
```

Data Processing

We add columns detailing each trial's scenario (`l1`-`l4`), containment level (`l1`-`l3`), complexity level (`l1`-`l4`), and rotation (can be `NA`, `LEFT`, `RIGHT`, or `distractor` for distractor scenarios).

Additionally, we add columns for log response time and whether the participant was correct on each trial (computed by whether their guess of "red" or "green" matched the correct answer for that trial).

We look at additional variants of response time in the response time analysis below.

```
# Add columns for scenario, containment, complexity, and rotation
scenario.split = with(data, strsplit(trialname, "_"))
data$scenario = unlist(lapply(scenario.split, "[", 2))
data$containment = unlist(lapply(scenario.split, "[", 4))
data$complexity = unlist(lapply(scenario.split, "[", 6))
data$rotation = unlist(lapply(scenario.split, "[", 7))
data$rotation[is.na(data$rotation)] = "NONE" # avoid NAs in this column
# Add column for log response time
data = data %>%
  group_by(scenario, subjID) %>%
  mutate(log.responsetime = log10(responsetime))
# Add column for whether selection was correct
data = data %>%
  mutate(correct = trialtarget == usertarget)

glimpse(data)
```

```
## Observations: 20,224
## Variables: 23
## $ subjID           <chr> "user_1548179758602", "user_1548179758602", "...
## $ logts            <dbl> 1.54818e+12, 1.54818e+12, 1.54818e+12, 1.5481...
## $ trialname        <chr> "contain_sc2_var_l3_complex_l3", "contain_sc3...
## $ trialindex       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
## $ trialstructure   <chr> "{'Name': 'contain_sc2_var_l3_complex_l3', 'D...
## $ goaldistances    <chr> "[{'name': 'red', 'goal': {'top': 130, 'left'...
## $ numwalls         <int> 6, 3, 5, 9, 3, 4, 3, 13, 7, 7, 13, 9, 5, 10, ...
## $ walldistances    <chr> "[{'wall': {'top': 200, 'left': 20, 'bottom':...
## $ numbounces       <int> 5, 12, 5, 49, 18, 3, 1, 7, 31, 8, 5, 2, 10, 1...
## $ ballstartpos     <chr> "{'x': 110, 'y': 500}", "{'x': 400, 'y': 30}"...
## $ ballwaitpos      <chr> "{'x': 162.5, 'y': 570.19999999998}", "{'x': ...
## $ trialtarget      <chr> "red", "red", "green", "green", "green", "red...
## $ targetswitched   <chr> "False", "False", "True", "False", "True", "T...
## $ usertarget       <chr> "green", "red", "red", "green", "green", "gre...
## $ responsetime     <dbl> 1560, 2753, 1525, 1344, 1205, 1108, 788, 677,...
## $ simtime          <int> 1528, 2120, 911, 3653, 4794, 1178, 653, 1270,...
## $ score            <int> -10, 32, -10, 60, 64, -10, 78, 82, 101, 106, ...
## $ scenario         <chr> "sc2", "sc3", "sc2", "sc4", "sc4", "sc2", "sc...
## $ containment      <chr> "l3", "l1", "l1", "l1", "l1", "l3", "l1", "l2...
## $ complexity       <chr> "l3", "l2", "l2", "l4", "l3", "l1", "l3", "l4...
## $ rotation         <chr> "NONE", "distractor", "distractor", "distract...
## $ log.responsetime <dbl> 3.193125, 3.439806, 3.183270, 3.128399, 3.080...
## $ correct          <lgl> FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, TRUE, ...
```

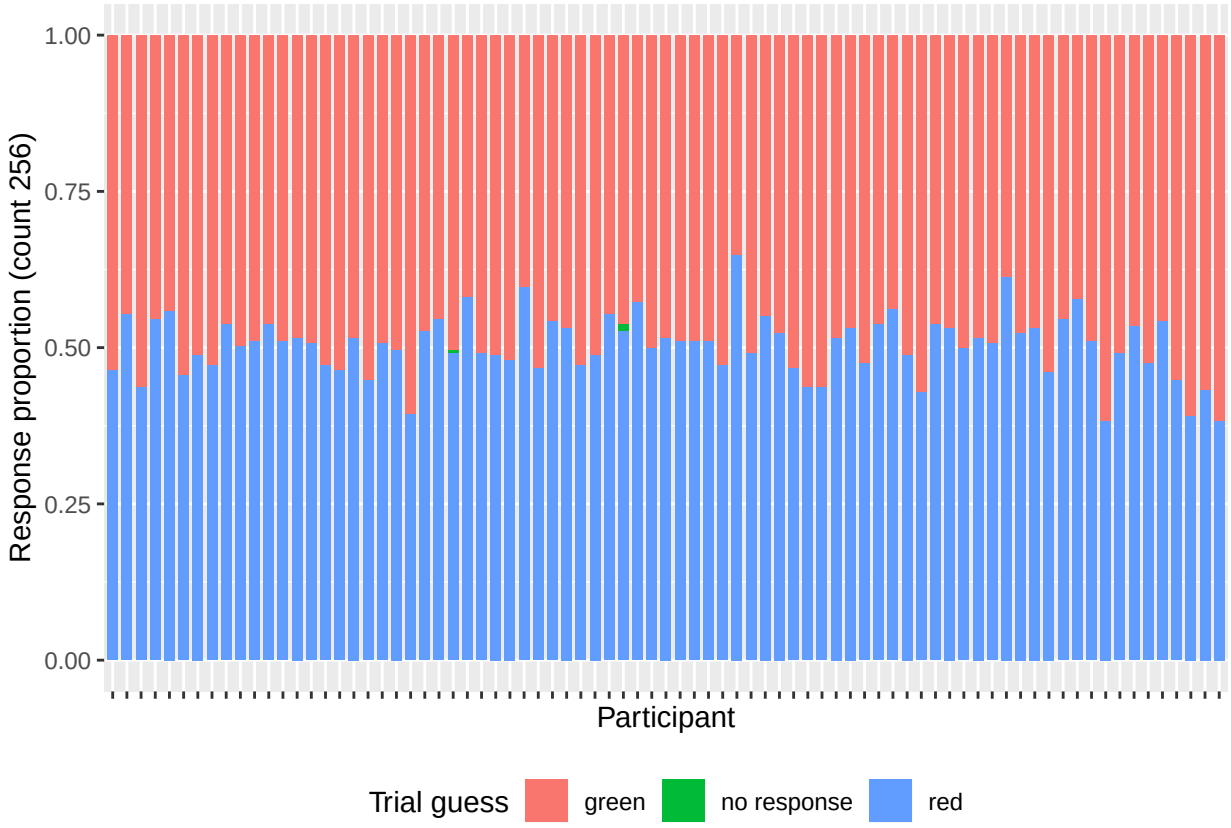
Sanity Checks: Data Quality

In this section, we look at the data for each participant to ensure basic benchmarks for data quality are met. We outline these benchmarks explicitly, such as ensuring that no participants simply guessed "red" on every trial.

Participant Guesses

Here we look at the proportion of "red" and "green" guesses that each participant made over all trials. We are explicitly looking for a reasonable balance of each guess. If any participant selected exclusively "red", "green", or "no response", we would be concerned that they were not making an honest attempt at the task.

```
data %>%
  ggplot(aes(x = subjID, fill = usertarget)) +
  geom_bar(position = "fill", width = 0.75) +
  theme(legend.position = "bottom", axis.text.x = element_blank()) +
  labs(x = "Participant", y = "Response proportion (count 256)", fill = "Trial guess")
```



Looks like all "no response" trials come from two participants: should we remove these trials from subsequent analysis?

```
no.resp.data = data %>%
  filter(usertarget == "no response")
table(no.resp.data$subjID)
```

```
##
## user_1547856241568 user_1548187033593
##                  1                  3
```

Participant Accuracy

Here we look at participant accuracy based on the proportion of their guesses which were correct. This number should not be so low that it reflects a lack of understanding of the task. If it is too high, we should be worried about ceiling effects.

```
data %>%
  ggplot(aes(x = subjID, fill = correct)) +
  geom_bar(position = "fill", width = 0.75) +
  theme(legend.position = "bottom", axis.text.x = element_blank()) +
  labs(x = "Participant", y = "Response proportion (count 256)", fill = "Trial correct")
```

We can see one participant that has significantly lower accuracy overall than the others. **Should we remove this participant's data?**

```
low.accuracy = data %>%
  group_by(subjID) %>%
  summarize(correct.count = sum(correct),
            total.count = length(correct),
            correct.pct = correct.count / total.count) %>%
  filter(correct.pct < 0.5)
kable(low.accuracy)
```

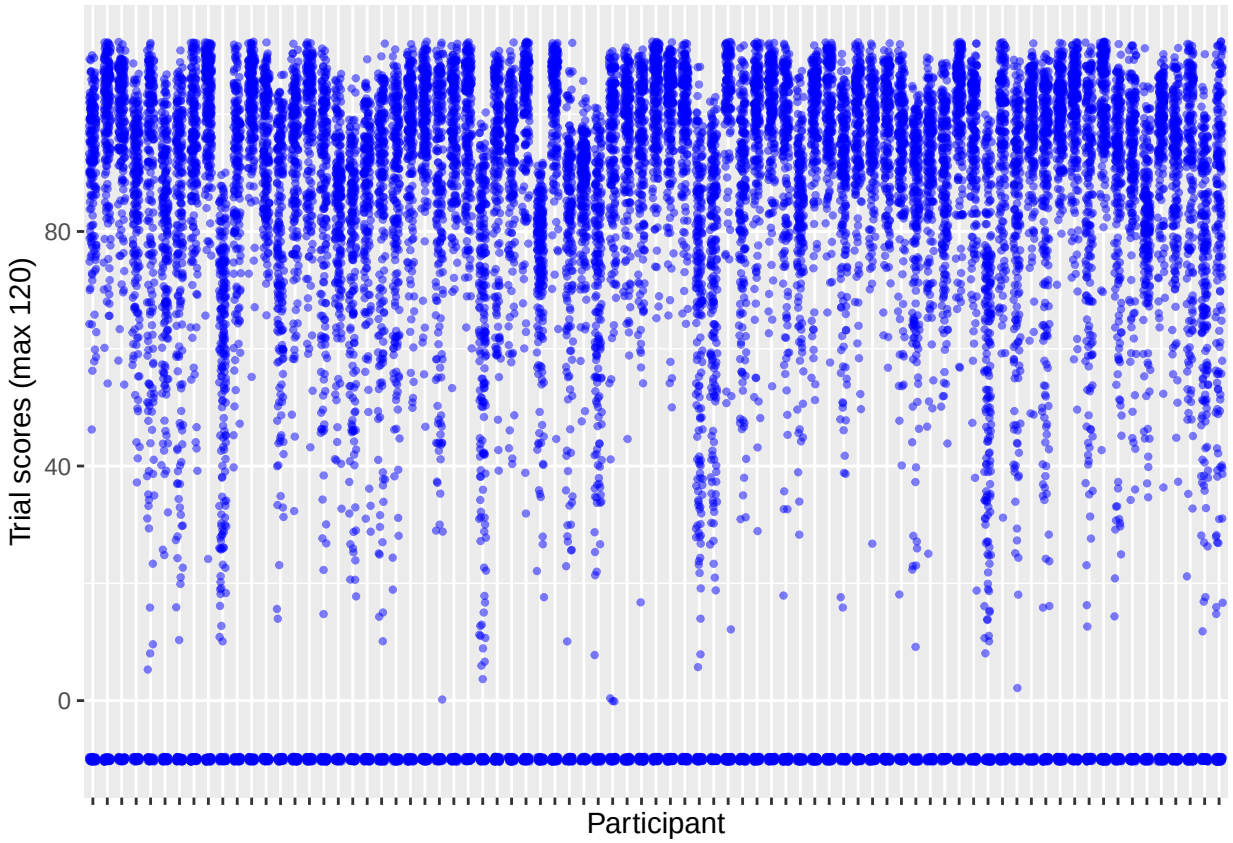
subjID	correct.count	total.count	correct.pct
user_1548273717903	120	256	0.46875

subjID	correct.count	total.count	correct.pct

Participant Scores

Here we look at the distribution of scores across all trials by each participant. These scores will be a combination of -10 for incorrect trials, 0 for no response trials, and a number > 0 for correct guesses that exponentially decays based on reponse time. If any participant has too many scores of 0 or -10 or if their scores > 0 don't show some variance, we should be concerned that they were not making an honest attempt at the task.

```
data %>%
  ggplot(aes(x = subjID, y = score)) +
  theme(axis.text.x = element_blank()) +
  geom_jitter(alpha = 0.5, size = 0.8, width = 0.25, color = "blue") +
  labs(x = "Participant", y = "Trial scores (max 120)")
```



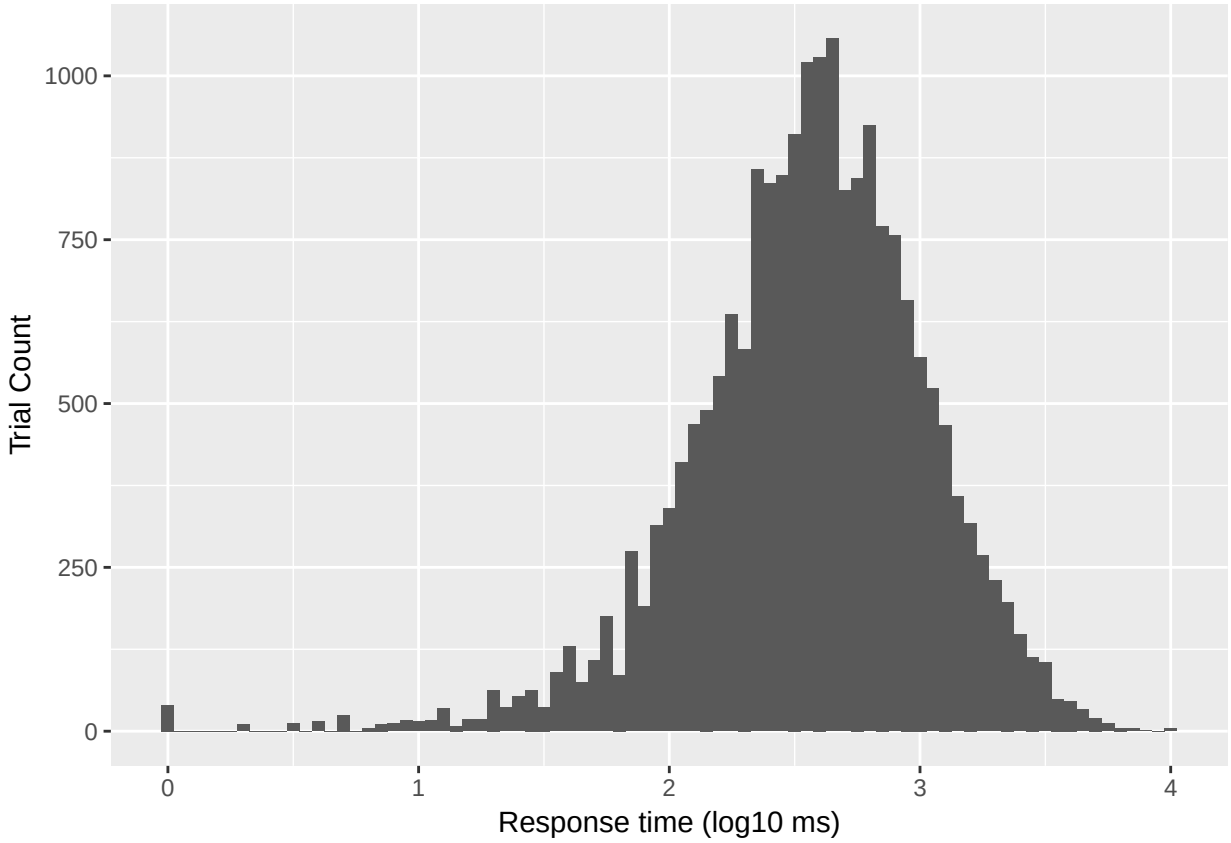
Participant Response Times

Here we look at a number of features of participant log response times to make sure this data (our primary DV) is as expected.

Overall Response Time Distribution

First, we look at the overall distribution of participant log response times. This should be roughly normal.

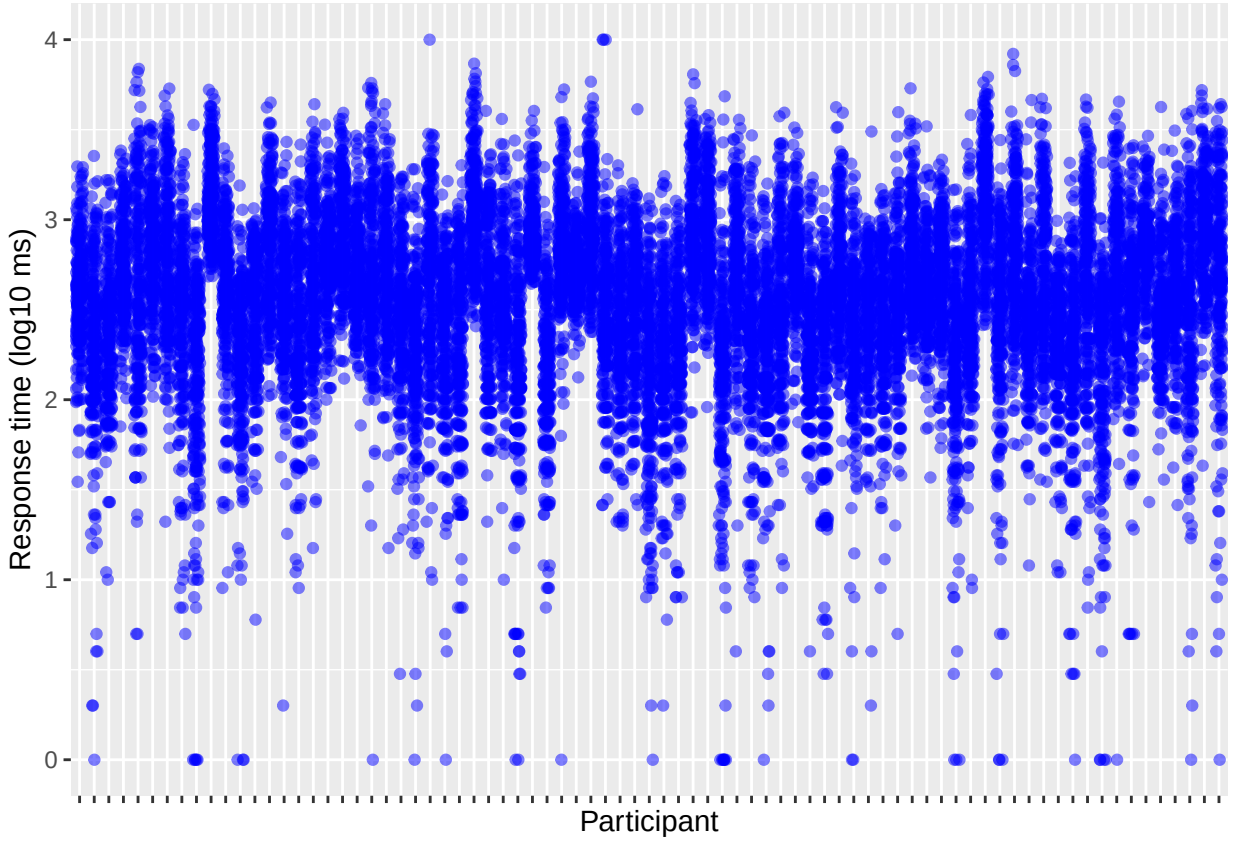
```
data %>%
  ggplot(aes(x = log.responsetime)) +
  geom_histogram(binwidth = 0.05) +
  labs(x = "Response time (log10 ms)", y = "Trial Count")
```



Individual Participant Response Time Distributions

Next, we look at individual participants' distributions of log response times. These should have some variance and not be too close to 0, otherwise we would be concerned that participants were guessing as fast as possible without making an honest attempt (though this ought to be reflected in the accuracy analysis above as well).

```
data %>%
  ggplot(aes(x = subjID, y = log.responsetime)) +
  geom_jitter(width = 0.25, alpha = 0.5, color = "blue") +
  theme(axis.text.x = element_blank()) +
  labs(x = "Participant", y = "Response time (log10 ms)")
```

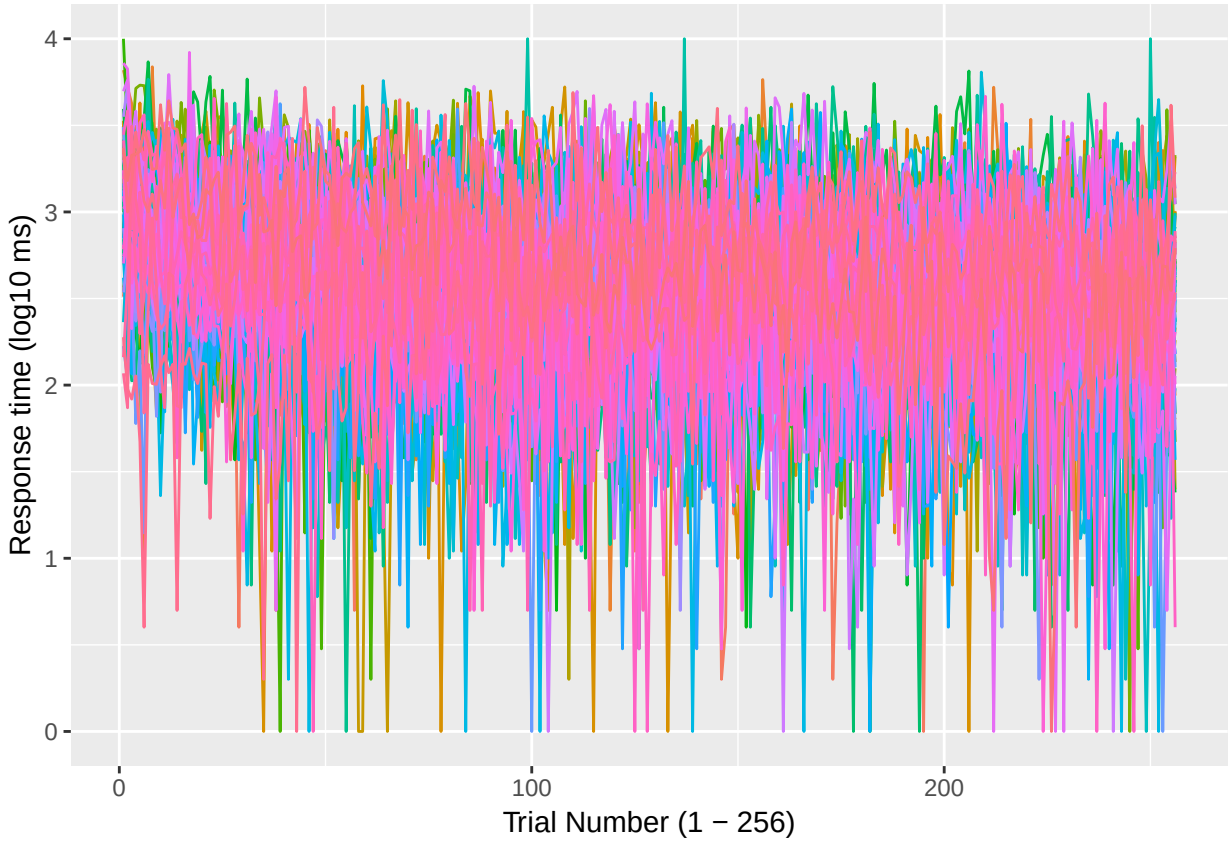


Individual Participant Response Times by Trial Index

Finally, we look at participant log response times by trial index in order to ensure that participants do not show signs of giving up as the task wears on or learning significantly over the course of the task.

In the first graph below, we don't see signs of individual responses dropping to consistently low levels that would signal becoming tired of the task and guessing (this would in theory show up in the accuracy analysis as well but this view is more fine-grained).

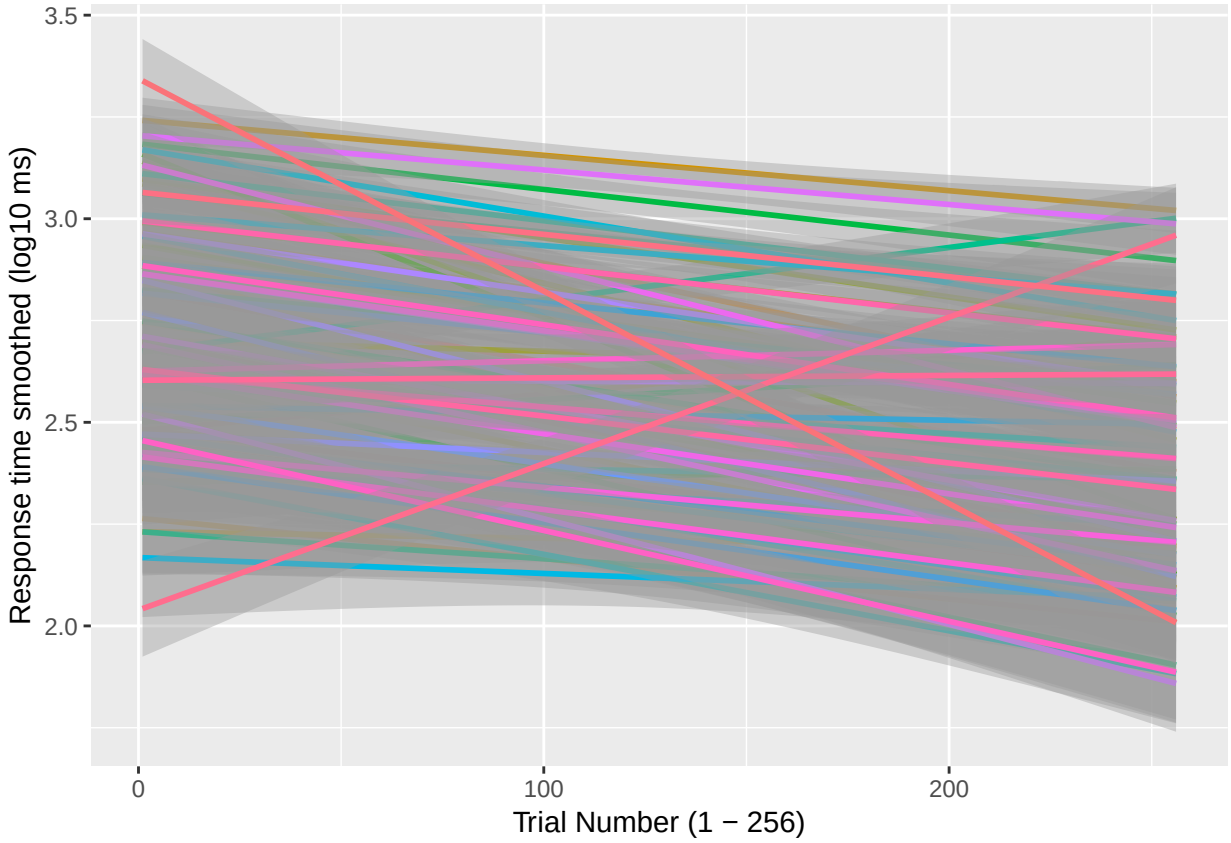
```
data %>%
  ggplot(aes(x = trialindex, y = log.responsetime, color = subjID)) +
  geom_line() +
  theme(legend.position = "none") +
  labs(x = "Trial Number (1 - 256)",
       y = "Response time (log10 ms)",
       color = "Participant")
```



In the graph below, a smoothed view of the previous one, we see some signs of participants learning over the course of the task.

What do we need to do to confirm this?

```
data %>%
  ggplot(aes(x = trialindex, y = log.responsetime, color = subjID)) +
  geom_smooth(method = 'lm') +
  theme(legend.position = "none") +
  labs(x = "Trial Number (1 - 256)",
       y = "Response time smoothed (log10 ms)",
       color = "Participant")
```



Data Analysis

Analysis Functions

The supporting labels and functions below help with the analysis in the Response Time Analysis section.

```
containment_labels = c(
  l1 = "low containment",
  l2 = "medium containment",
  l3 = "high containment"
)

complexity_labels = c(
  l1 = "none",
  l2 = "low",
  l3 = "medium",
  l4 = "high"
)

scenario_labels = c(
  sc1 = "scenario 1",
  sc2 = "scenario 2",
  sc3 = "scenario 3",
  sc4 = "scenario 4"
)
```

```
make.canonical.bargraph = function(df.means, title, xlab, ylab) {
  df.means %>%
    ggplot(aes(x = complexity, y = means)) +
    geom_bar(stat = "identity", width = 0.5) +
    scale_x_discrete(labels = complexity_labels) +
    geom_errorbar(mapping = aes(ymin = se.lower, ymax = se.upper), width = 0.2) +
    facet_wrap(. ~ containment,
               scales = "free_x",
               labeller = labeller(containment = containment_labels)) +
    labs(x = xlab, y = ylab) +
    ggtitle(title)
}

make.canonical.bargraph.scenario = function(df.means, title, xlab, ylab) {
  df.means %>%
    ggplot(aes(x = complexity, y = means, color = scenario)) +
    geom_bar(stat = "identity", width = 0.5) +
    scale_x_discrete(labels = complexity_labels) +
    guides(color = FALSE) +
    geom_errorbar(mapping = aes(ymin = se.lower, ymax = se.upper), width = 0.2) +
    facet_grid(scenario ~ containment,
               scales = "free_x",
               labeller = labeller(containment = containment_labels,
                                   scenario = scenario_labels)) +
    labs(x = xlab, y = ylab) +
    ggtitle(title)
}
```

Analysis Data Processing

To support a more nuanced analysis of response time, we add columns for trial response time scaled by mean response time for each each participant, scenario. We also take the log of these response time modifications.

```
data = data %>%
  filter(rotation %in% c("NONE", "LEFT", "RIGHT", "TWICE")) %>%
  group_by(scenario, subjID) %>%
  mutate(trials = n(),
         mean.scenario.subj.responsetime = mean(responsetime),
         log.mean.scenario.subj.responsetime = mean(log.responsetime),
         scaled.responsetime = responsetime - mean.scenario.subj.responsetime,
         log.scaled.responsetime = log.responsetime - log.mean.scenario.subj.responsetime)

glimpse(data)
```

```
## Observations: 15,168
## Variables: 28
## $ subjID                              <chr> "user_1548179758602", "use...
## $ logts                               <dbl> 1.54818e+12, 1.54818e+12, ...
## $ trialname                           <chr> "contain_sc2_var_l3_comple...
## $ trialindex                          <int> 1, 6, 7, 8, 9, 10, 14, 16,...
## $ trialstructure                      <chr> "{'Name': 'contain_sc2_var...
## $ goaldistances                       <chr> "[{'name': 'red', 'goal': ...
## $ numwalls                            <int> 6, 4, 3, 13, 7, 7, 10, 3, ...
## $ walldistances                       <chr> "[{'wall': {'top': 200, 'l...
## $ numbounces                          <int> 5, 3, 1, 7, 31, 8, 11, 8, ...
## $ ballstartpos                        <chr> "{'x': 110, 'y': 500}", "{...
## $ ballwaitpos                         <chr> "{'x': 162.5, 'y': 570.199...
## $ trialtarget                         <chr> "red", "red", "red", "gree...
## $ targetswitched                      <chr> "False", "True", "False", ...
## $ usertarget                          <chr> "green", "green", "red", "...
## $ responsetime                        <dbl> 1560, 1108, 788, 677, 221,...
## $ simtime                             <int> 1528, 1178, 653, 1270, 644...
## $ score                               <int> -10, -10, 78, 82, 101, 106...
## $ scenario                            <chr> "sc2", "sc2", "sc4", "sc1"...
## $ containment                         <chr> "l3", "l3", "l1", "l2", "l...
## $ complexity                          <chr> "l3", "l1", "l3", "l4", "l...
## $ rotation                            <chr> "NONE", "RIGHT", "TWICE", ...
## $ log.responsetime                    <dbl> 3.193125, 3.044540, 2.8965...
## $ correct                             <lgl> FALSE, FALSE, TRUE, TRUE, ...
## $ trials                              <int> 48, 48, 48, 48, 48, 48, 48...
## $ mean.scenario.subj.responsetime     <dbl> 254.3958, 254.3958, 270.47...
## $ log.mean.scenario.subj.responsetime <dbl> 2.068401, 2.068401, 2.1066...
## $ scaled.responsetime                 <dbl> 1305.60417, 853.60417, 517...
## $ log.scaled.responsetime             <dbl> 1.1247240799, 0.9761392419...
```

```
# sanity check the scaled responsetime: these should be 0
data %>%
  group_by(scenario, subjID) %>%
  summarize(sum.scaled.responsetime = sum(scaled.responsetime),
            sum.log.scaled.responsetime = sum(log.scaled.responsetime)) %>%
  summarize(scaled.responsetime.check = max(sum.scaled.responsetime),
            log.scaled.responsetime.check = max(sum.scaled.responsetime))
```

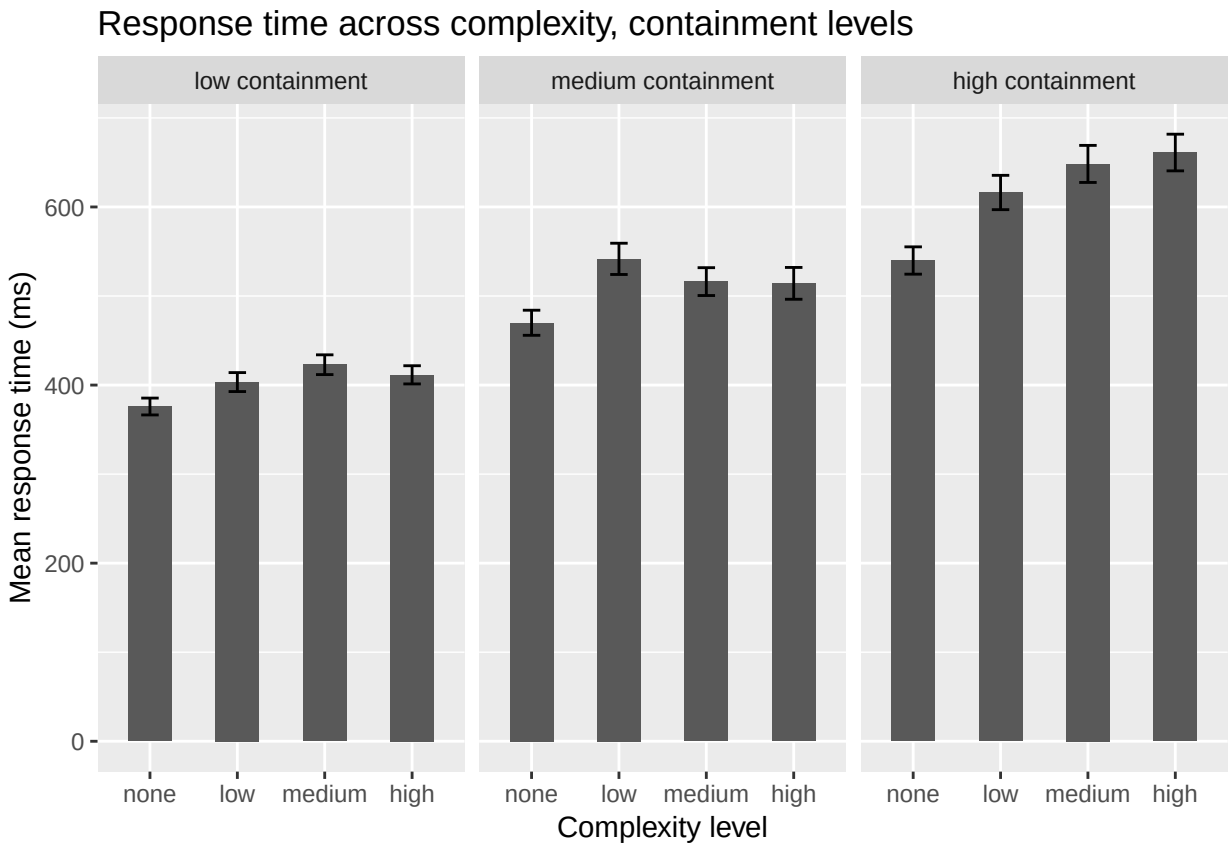
```
## # A tibble: 4 x 3
##   scenario scaled.responsetime.check log.scaled.responsetime.check
##   <chr>                        <dbl>                         <dbl>
## 1 sc1              0.00000000000409              0.00000000000409
## 2 sc2              0.00000000000455              0.00000000000455
## 3 sc3              0.00000000000387              0.00000000000387
## 4 sc4              0.00000000000432              0.00000000000432
```

Response Time Analysis

In this section, we look at response times across varying complexity and containment levels.

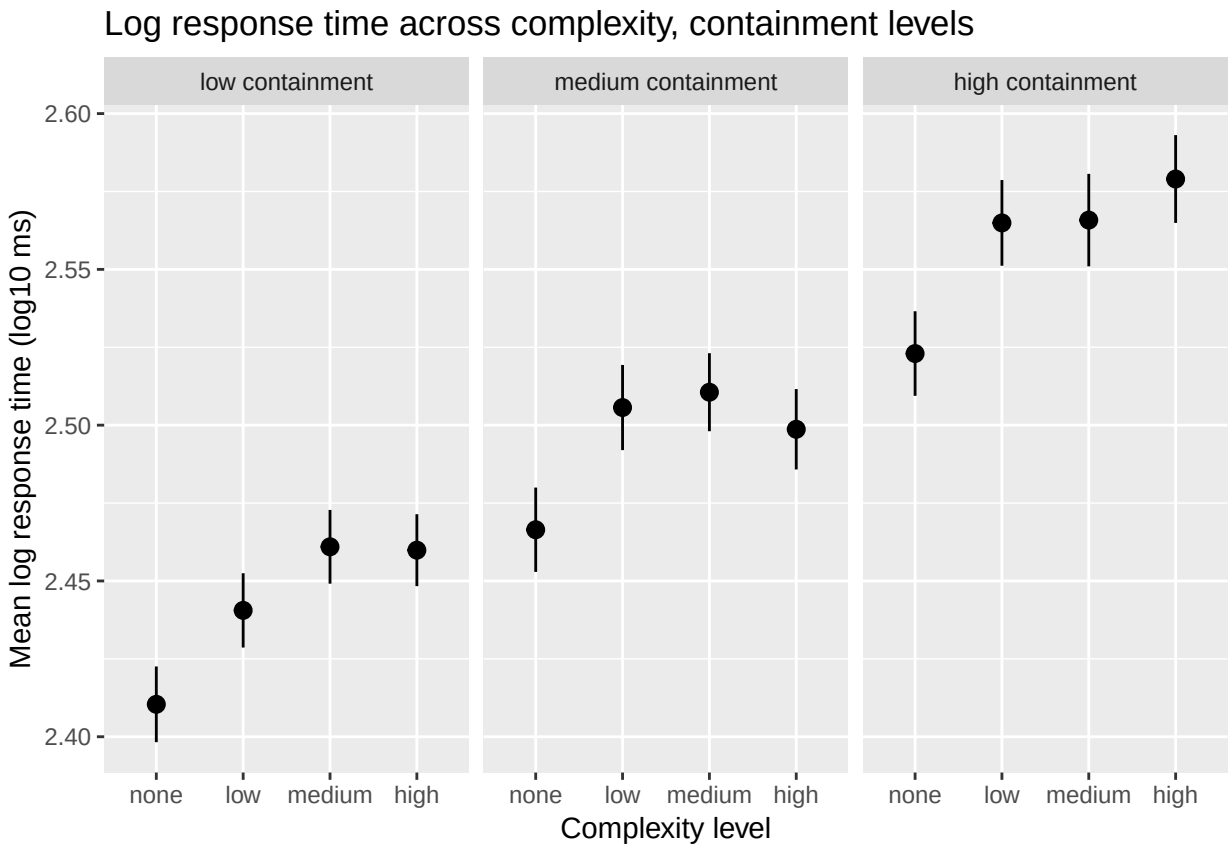
Raw Response Time

```
title = "Response time across complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean response time (ms)"

# Calculate means, CIs
responsetime.means = data %>%
  group_by(containment, complexity) %>%
  summarize(means = mean(responsetime),
            trials = n(),
            se.lower = means - sqrt(var(responsetime) / length(responsetime)),
            se.upper = means + sqrt(var(responsetime) / length(responsetime))) %>%
  select(containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph(responsetime.means, title, xlab, ylab)
```



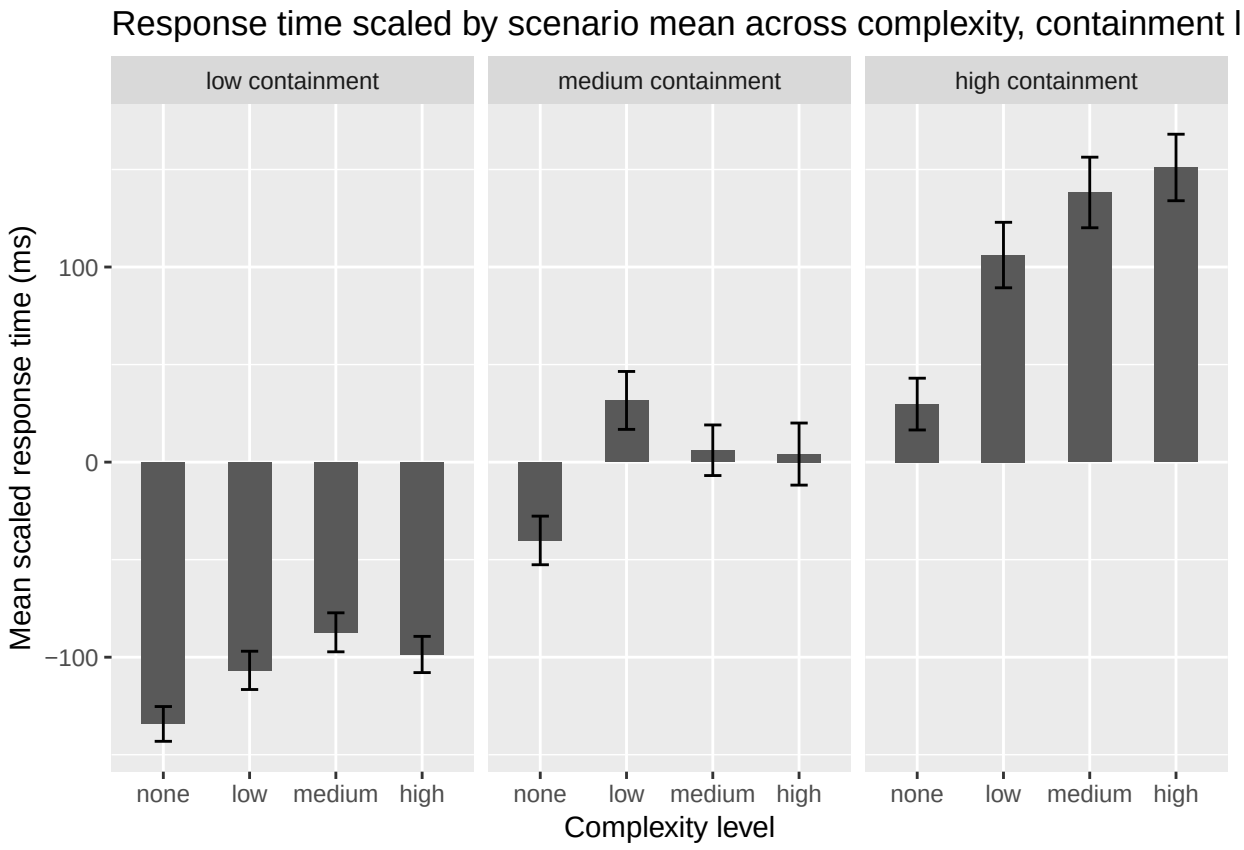
Log Response Time

```
title = "Log response time across complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean log response time (log10 ms)"

# Calculate means, CIs
log.responsetime.means = data %>%
  group_by(containment, complexity) %>%
  summarize(means = mean(log.responsetime),
            trials = n(),
            se.lower = means - sqrt(var(log.responsetime) / length(log.responsetime)),
            se.upper = means + sqrt(var(log.responsetime) / length(log.responsetime))) %>%
  select(containment, complexity, means, trials, se.lower, se.upper)
# Graph data
log.responsetime.means %>%
  ggplot(aes(x = complexity, y = means, ymin = se.lower, ymax = se.upper)) +
  geom_pointrange() +
  scale_x_discrete(labels = complexity_labels) +
  facet_wrap(. ~ containment,
             scales = "free_x",
             labeller = labeller(containment = containment_labels)) +
  labs(x = xlab, y = ylab) +
  ggtitle(title)
```



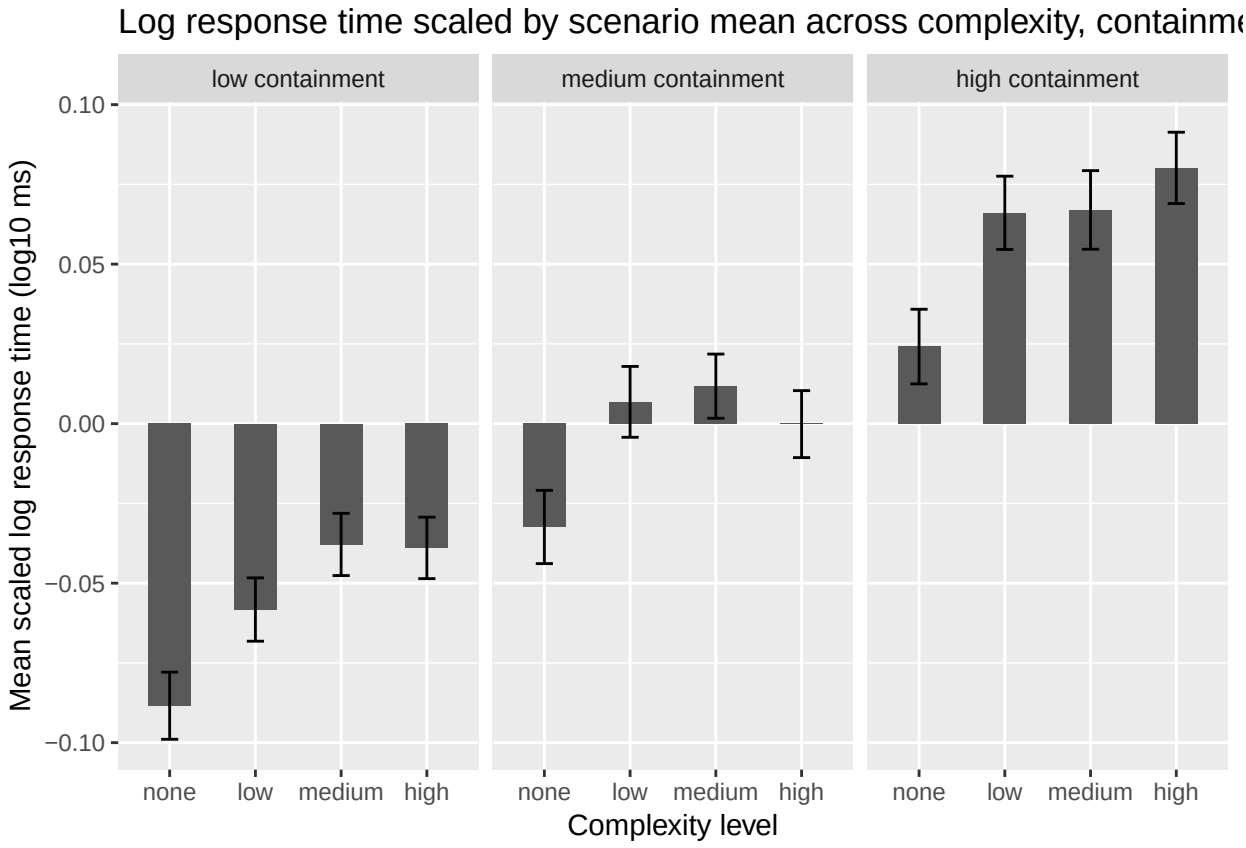
Raw Response Time Scaled by Participant, Scenario Mean

```
title = "Response time scaled by scenario mean across complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean scaled response time (ms)"

# Calculate means, CIs
scaled.responsetime.means = data %>%
  group_by(containment, complexity) %>%
  summarize(means = mean(scaled.responsetime),
            trials = n(),
            se.lower = means - sqrt(var(scaled.responsetime) / length(scaled.responsetime)),
            se.upper = means + sqrt(var(scaled.responsetime) / length(scaled.responsetime))) %>%
  select(containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph(scaled.responsetime.means, title, xlab, ylab)
```



Log Response Time Scaled by Participant, Scenario Mean

```
title = "Log response time scaled by scenario mean across complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean scaled log response time (log10 ms)"

# Calculate means, CIs
log.scaled.responsetime.means = data %>%
  group_by(containment, complexity) %>%
  summarize(means = mean(log.scaled.responsetime),
            trials = n(),
            se.lower = means - sqrt(var(log.scaled.responsetime) / length(log.scaled.responsetime)),
            se.upper = means + sqrt(var(log.scaled.responsetime) / length(log.scaled.responsetime))) %>%
  select(containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph(log.scaled.responsetime.means, title, xlab, ylab)
```

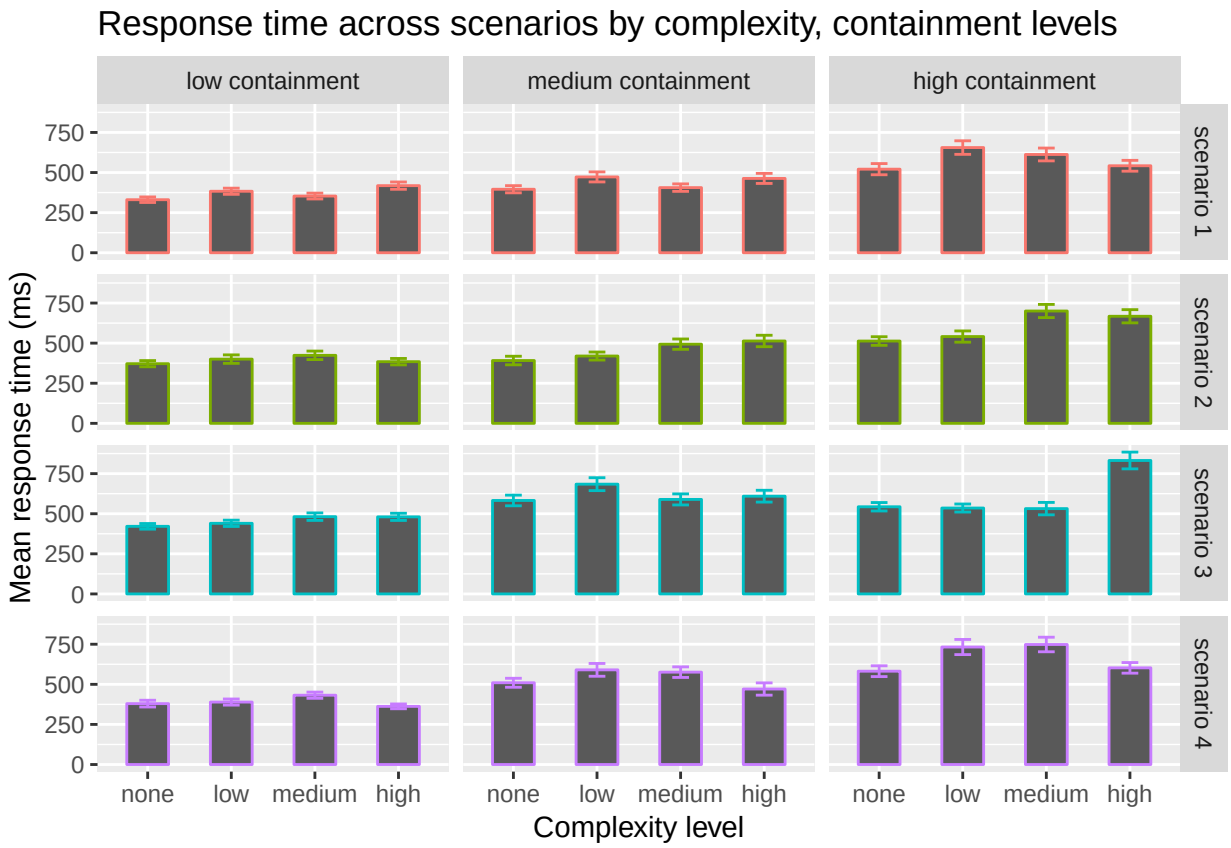


Response Time Analysis by Scenario

In this section, we look at response times across varying complexity and containment levels *for each scenario*.

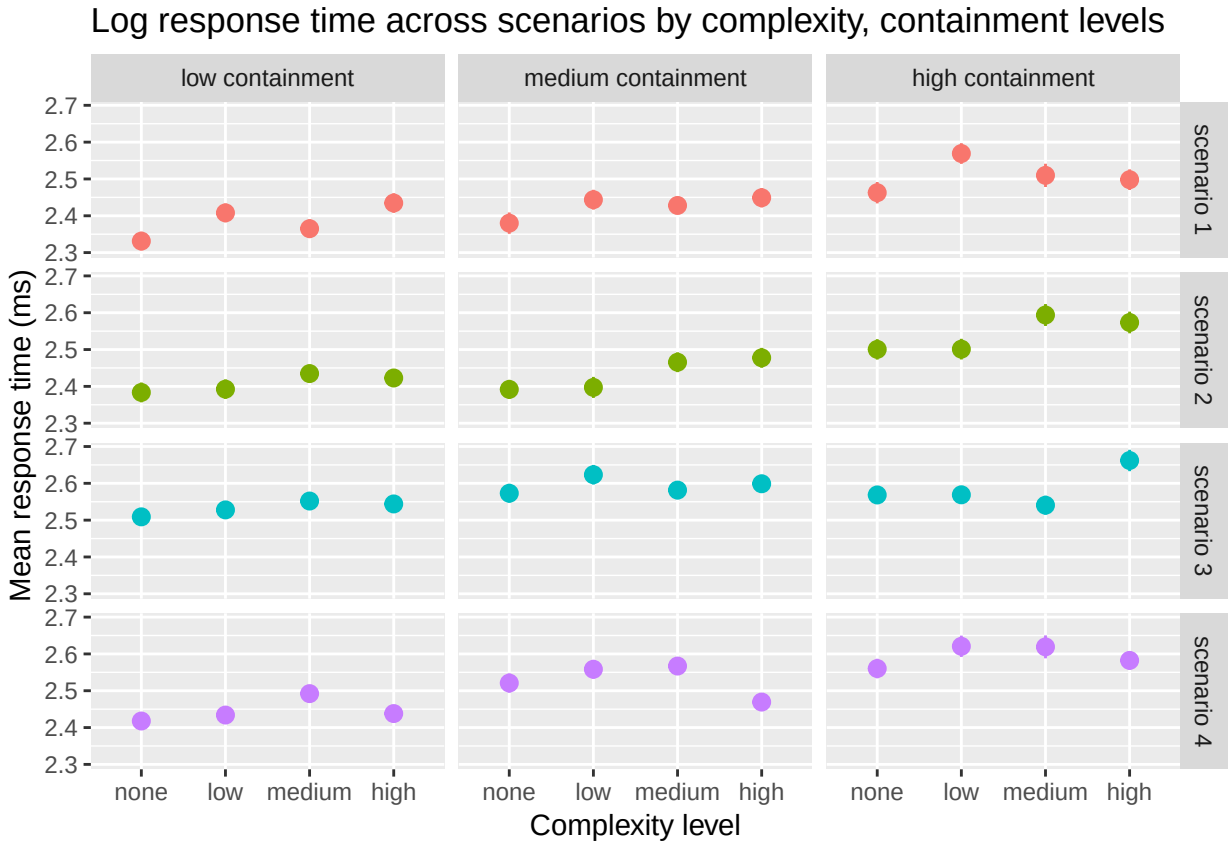
Raw Response Time by Scenario

```
title = "Response time across scenarios by complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean response time (ms)"

# Calculate means, CIs
responsetime.scenario.means = data %>%
  group_by(scenario, containment, complexity) %>%
  summarize(means = mean(responsetime),
            trials = n(),
            se.lower = means - sqrt(var(responsetime) / length(responsetime)),
            se.upper = means + sqrt(var(responsetime) / length(responsetime))) %>%
  select(scenario, containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph.scenario(responsetime.scenario.means, title, xlab, ylab)
```



Log Response Time by Scenario

```
title = "Log response time across scenarios by complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean response time (ms)"

# Calculate means, CIs
log.responsetime.scenario.means = data %>%
  group_by(scenario, containment, complexity) %>%
  summarize(means = mean(log.responsetime),
            trials = n(),
            se.lower = means - sqrt(var(log.responsetime) / length(log.responsetime)),
            se.upper = means + sqrt(var(log.responsetime) / length(log.responsetime))) %>%
  select(scenario, containment, complexity, means, trials, se.lower, se.upper)
# Graph data
log.responsetime.scenario.means %>%
    ggplot(aes(x = complexity, y = means, ymin = se.lower, ymax = se.upper, color = scenario)) +
    geom_pointrange() +
    scale_x_discrete(labels = complexity_labels) +
    guides(color = FALSE) +
    facet_grid(scenario ~ containment,
               scales = "free_x",
               labeller = labeller(containment = containment_labels,
                                   scenario = scenario_labels)) +
    labs(x = xlab, y = ylab) +
    ggtitle(title)
```

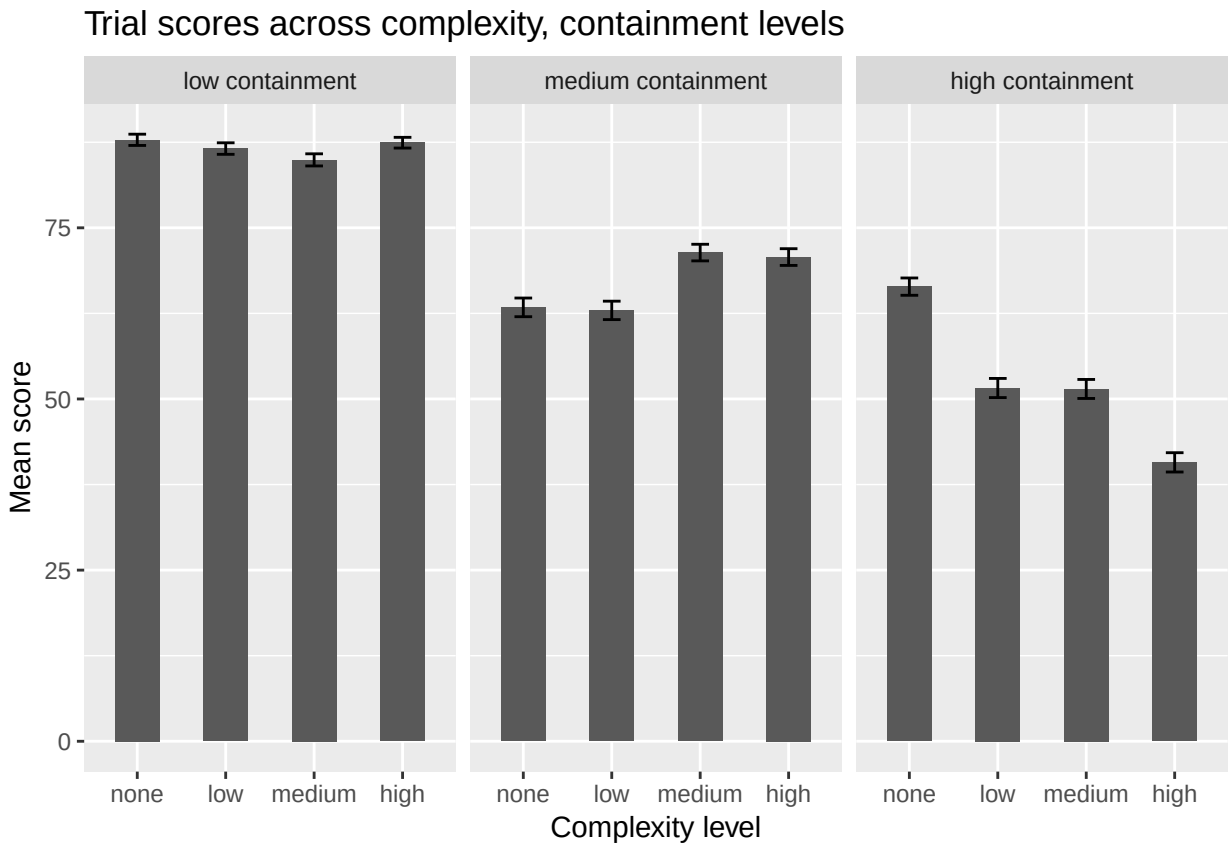


Score Analysis

In this section, we look at scores across varying complexity and containment levels.

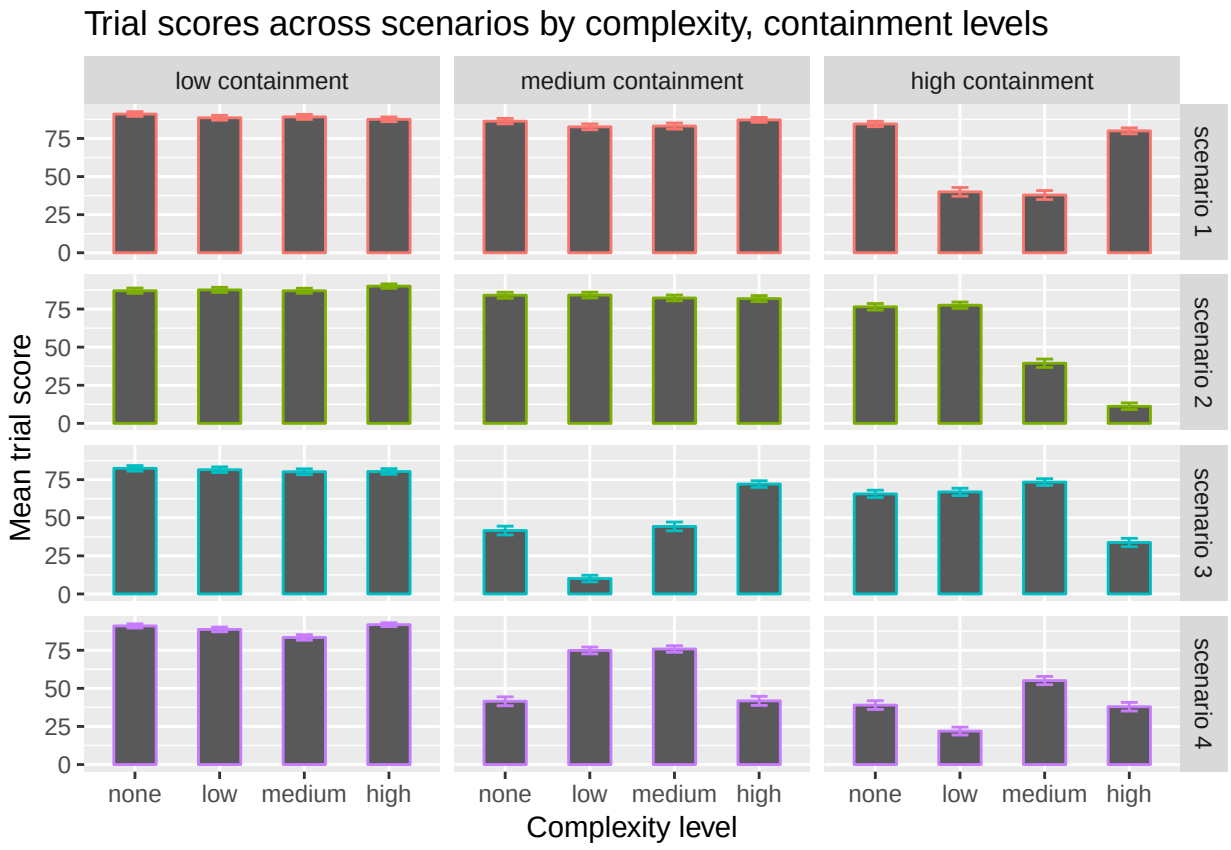
Trial Score

```
title = "Trial scores across complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean score"

# Calculate means, CIs
score.means = data %>%
  group_by(containment, complexity) %>%
  summarize(means = mean(score),
            trials = n(),
            se.lower = means - sqrt(var(score) / length(score)),
            se.upper = means + sqrt(var(score) / length(score))) %>%
  select(containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph(score.means, title, xlab, ylab)
```



Trial Score by Scenario

```
title = "Trial scores across scenarios by complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean trial score"

# Calculate means, CIs
score.scenario.means = data %>%
  group_by(scenario, containment, complexity) %>%
  summarize(means = mean(score),
            trials = n(),
            se.lower = means - sqrt(var(score) / length(score)),
            se.upper = means + sqrt(var(score) / length(score))) %>%
  select(scenario, containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph.scenario(score.scenario.means, title, xlab, ylab)
```



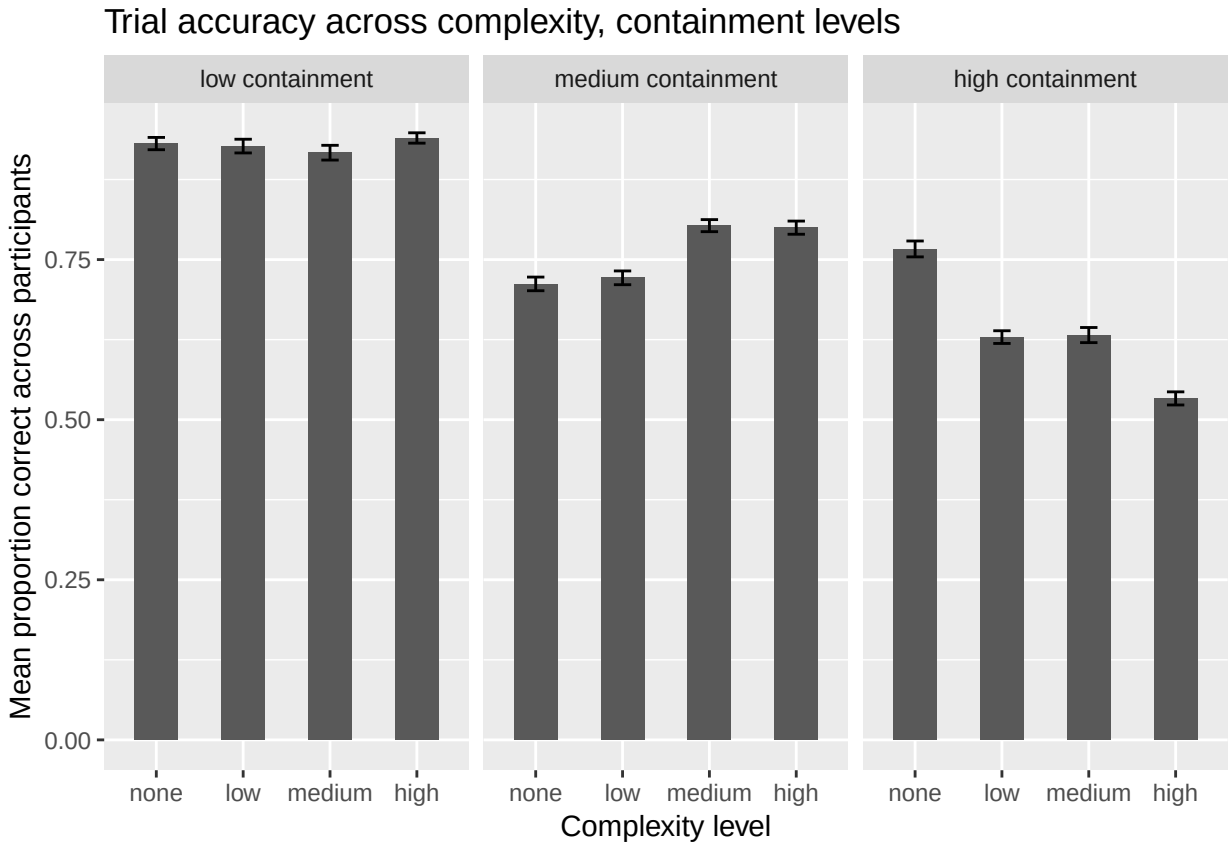
Accuracy Analysis

In this section, we look at participant accuracy across varying complexity and containment levels.

Trial Accuracy

Calculating each participant's proportion of correct trials over each complexity and containment level (16 obs. per participant), below is mean accuracy across participants in each complexity and containment level.

```
title = "Trial accuracy across complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean proportion correct across participants"

# Calculate participant accuracy for each containment/complexity level
participant.accuracy.means = data %>%
  group_by(subjID, containment, complexity) %>%
  summarize(right.answers = sum(correct),
            total.answers = n(),
            accuracy = right.answers / total.answers)

# Calculate means, CIs
accuracy.means = participant.accuracy.means %>%
  group_by(containment, complexity) %>%
  summarize(means = mean(accuracy),
            trials = n(),
            se.lower = means - sqrt(var(accuracy) / length(accuracy)),
            se.upper = means + sqrt(var(accuracy) / length(accuracy))) %>%
  select(containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph(accuracy.means, title, xlab, ylab)
```


Trial accuracy across complexity, containment levels
low containment
medium containment
high containment
Mean proportion correct across participants
0.00
0.25
0.50
0.75
none
low
medium
high
Complexity level

Trial Accuracy by Scenario

Calculating each participant's proportion of correct trials across scenarios for each complexity and containment level (4 obs. per participant), below is mean accuracy for all participants.

```
title = "Trial accuracy across scenarios by complexity, containment levels"
xlab = "Complexity level"
ylab = "Mean proportion correct across participants"

# Calculate participant accuracy in each scenario by containment, complexity level
participant.scenario.accuracy.means = data %>%
  group_by(subjID, scenario, containment, complexity) %>%
  summarize(right.answers = sum(correct),
            total.answers = n(),
            accuracy = right.answers / total.answers)

# Calculate means, CIs
scenario.accuracy.means = participant.scenario.accuracy.means %>%
  group_by(scenario, containment, complexity) %>%
  summarize(means = mean(accuracy),
            trials = n(),
            se.lower = means - sqrt(var(accuracy) / length(accuracy)),
            se.upper = means + sqrt(var(accuracy) / length(accuracy))) %>%
  select(scenario, containment, complexity, means, trials, se.lower, se.upper)
# Graph data
make.canonical.bargraph.scenario(scenario.accuracy.means, title, xlab, ylab)
```