E4-analysis-ROIs

```r
#Define a function for determining if `x,y` value falls in box.
in.box <- function(x, y, left, right, top, bottom, padding){
  is.in.the.box <- x >= left - padding & x <= right + padding & y >= top - padding & y <= bottom + paddi
  return(is.in.the.box)
}
```

```r
data.files <- list.files('data/run-2', full.names = TRUE)
data.tables <- lapply(data.files, function(file){
  data.table <- fromJSON(file)
  return(data.table)
})
all.data <- bind_rows(data.tables)
```

```r
task.data <- all.data %>%
  dplyr::filter(compatibility != 'NA', compatibility != 'filler') %>%
  dplyr::select(subject, trial_index, rt, images, webgazer_data, mouse_events, compatibility, audio, ta

screen.data <- all.data %>%
  dplyr::select(subject, screen_height, screen_width) %>%
  filter(!is.na(screen_height)) %>%
  unique() %>%
  group_by(subject) %>%
  mutate(n()) %>% # one subject has 2 rows because width is different by less than a pixel. I'll just k
  sample_n(size=1)


# Add a column that uniquely identifies the combination of images and audio shown on the screen
trialID.data <- task.data %>%
  group_by(audio, images) %>%
  slice(1) %>%
  select(audio, images) %>%
  ungroup() %>%
  mutate(trialID = 1:n())

task.data <- task.data %>%
  left_join(trialID.data)
```

```
## Joining, by = c("images", "audio")
```

```r
eyetracking.data <- task.data %>%
  tidyr::unpack(webgazer_targets) %>%
  tidyr::unpack(c(`#jspsych-free-sort-draggable-0`, `#jspsych-free-sort-draggable-1`, `#jspsych-free-so
  unnest(webgazer_data)
```

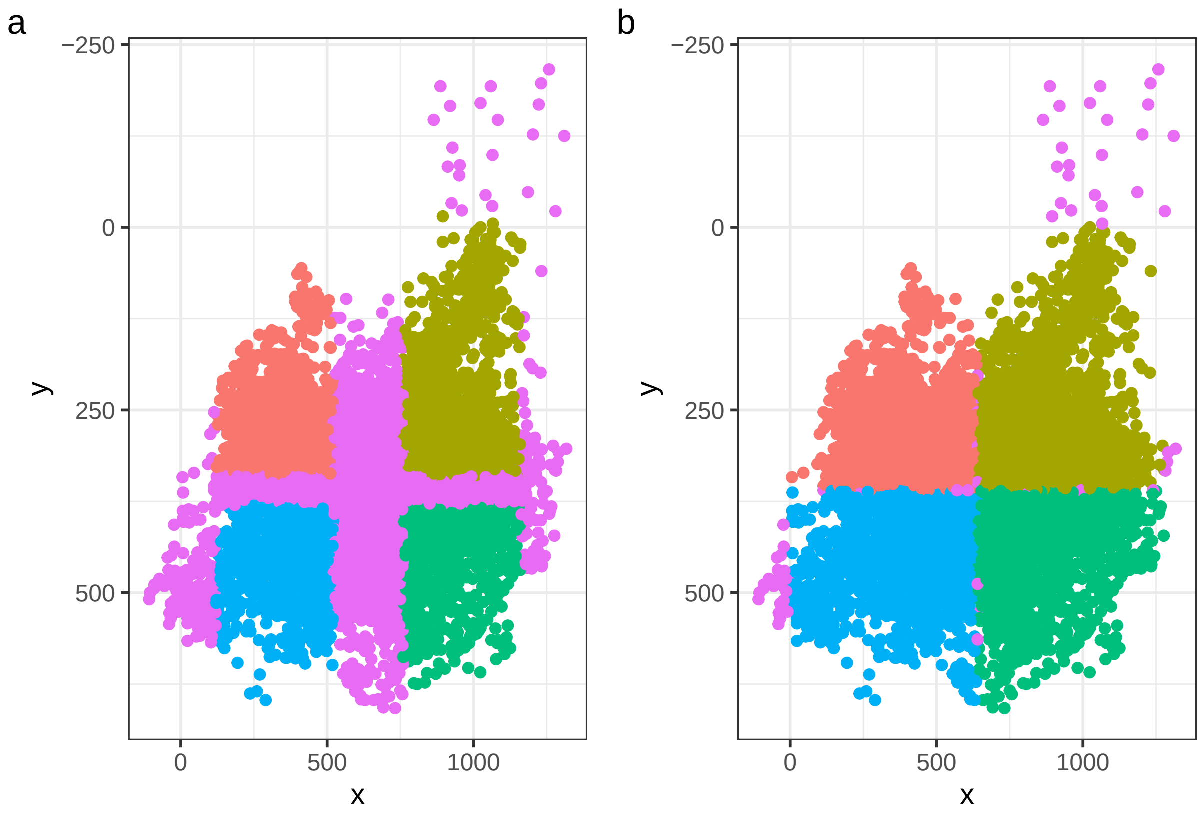
```
mousetracking.data <- task.data %>%
  unnest(mouse_events)
```

```
#First figure out which object they are looking at.
#Calculate gaze in ROIs.

eyetracking.data.with.roi <- eyetracking.data %>%
  left_join(screen.data, by = 'subject') %>%
  #filter(!subject %in% bad.eyetracking.data.subjects) %>%
  #filter(x >= 0, y >= 0) %>% # negative values indicate something's incorrect (drops about 12,000 rows
  mutate(in.roi.0 = in.box(x,y,`#jspsych-free-sort-draggable-0.left`, `#jspsych-free-sort-draggable-0.ri
  mutate(in.roi.1 = in.box(x,y,`#jspsych-free-sort-draggable-1.left`, `#jspsych-free-sort-draggable-1.ri
  mutate(in.roi.2 = in.box(x,y,`#jspsych-free-sort-draggable-2.left`, `#jspsych-free-sort-draggable-2.ri
  mutate(in.roi.3 = in.box(x,y,`#jspsych-free-sort-draggable-3.left`, `#jspsych-free-sort-draggable-3.ri
  mutate(in.roi.instrument = case_when(
    target_instrument == '#jspsych-free-sort-draggable-0' | target_instrument == '#jspsych-freesort-drag
    target_instrument == '#jspsych-free-sort-draggable-1' | target_instrument == '#jspsych-freesort-drag
    target_instrument == '#jspsych-free-sort-draggable-2' | target_instrument == '#jspsych-freesort-drag
    target_instrument == '#jspsych-free-sort-draggable-3' | target_instrument == '#jspsych-freesort-drag
  )) %>%
  mutate(in.roi.animal = case_when(
    target_animal == '#jspsych-free-sort-draggable-0' | target_animal == '#jspsych-freesort-draggable-0
    target_animal == '#jspsych-free-sort-draggable-1' | target_animal == '#jspsych-freesort-draggable-1
    target_animal == '#jspsych-free-sort-draggable-2' | target_animal == '#jspsych-freesort-draggable-2
    target_animal == '#jspsych-free-sort-draggable-3' | target_animal == '#jspsych-freesort-draggable-3
  )) %>%
  #sample_n(100) %>%
  rowwise() %>%
  mutate(roi_vec = list(c(in.roi.0, in.roi.1, in.roi.2, in.roi.3)),
         which_roi = ifelse(sum(roi_vec)==1, which(unlist(roi_vec) == TRUE), 99 ) )

padding = 0
eyetracking.data.with.quadrant.roi <- eyetracking.data %>%
  left_join(screen.data, by = 'subject') %>%
  #filter(!subject %in% bad.eyetracking.data.subjects) %>%
  #filter(x >= 0, y >= 0) %>% # negative values indicate something's incorrect (drops about 12,000 rows
  mutate(in.roi.0 = in.box(x,y, left = 0, right = screen_width/2, top = 0, bottom = screen_height/2, pad
  mutate(in.roi.1 = in.box(x,y, left = screen_width/2, right = screen_width, top = 0, bottom = screen_he
  mutate(in.roi.2 = in.box(x,y, left = screen_width/2, right = screen_width, top = screen_height/2 , bot
  mutate(in.roi.3 = in.box(x,y, left = 0, right = screen_width/2, top = screen_height/2 , bottom = scree
  mutate(in.roi.instrument = case_when(
    target_instrument == '#jspsych-free-sort-draggable-0' | target_instrument == '#jspsych-freesort-drag
    target_instrument == '#jspsych-free-sort-draggable-1' | target_instrument == '#jspsych-freesort-drag
    target_instrument == '#jspsych-free-sort-draggable-2' | target_instrument == '#jspsych-freesort-drag
    target_instrument == '#jspsych-free-sort-draggable-3' | target_instrument == '#jspsych-freesort-drag
  )) %>%
  mutate(in.roi.animal = case_when(
    target_animal == '#jspsych-free-sort-draggable-0' | target_animal == '#jspsych-freesort-draggable-0
    target_animal == '#jspsych-free-sort-draggable-1' | target_animal == '#jspsych-freesort-draggable-1
    target_animal == '#jspsych-free-sort-draggable-2' | target_animal == '#jspsych-freesort-draggable-2
    target_animal == '#jspsych-free-sort-draggable-3' | target_animal == '#jspsych-freesort-draggable-3
  )) %>%
  #sample_n(100) %>%
  rowwise() %>%
  mutate(roi_vec = list(c(in.roi.0, in.roi.1, in.roi.2, in.roi.3)),
         which_roi = ifelse(sum(roi_vec)==1, which(unlist(roi_vec) == TRUE), 99 ) )
```

```
ggplot(eyetracking.data.with.roi %>% filter(subject == '5564e577fdf99b6c901a75a6'))+
  geom_text(aes(x = `#jspsych-free-sort-draggable-0.x`, y = `#jspsych-free-sort-draggable-0.y`), color
  geom_text(aes(x = `#jspsych-free-sort-draggable-1.x`, y = `#jspsych-free-sort-draggable-1.y`), color
  geom_text(aes(x = `#jspsych-free-sort-draggable-2.x`, y = `#jspsych-free-sort-draggable-2.y`), color
  geom_text(aes(x = `#jspsych-free-sort-draggable-3.x`, y = `#jspsych-free-sort-draggable-3.y`), color
  scale_y_reverse()
```

Eye-tracking on the web differs critically from in-lab eye-tracking in that the size of the display differs across participants. Thus the size of the ROIs differs across participants. The current version of the web experiment used a bounding box around each image to determine the ROI. This approach is flexible and accomodates variability in image size, but may exclude looks that are directed at the image but fall outside of the image (due to participant or eye-tracker noise) as show in Figure @ref(fig:E4-example-subj-looks-ROI-quadrant) - left . Alternatively, The display can be split into 4 quadrants which jointly cover the entire screen (see Figure @ref(fig:E4-example-subj-looks-ROI-quadrant) - right).

```
p_ROI<-ggplot(eyetracking.data.with.roi %>% filter(subject == '5564e577fdf99b6c901a75a6'))+
  geom_point(aes(x = x, y = y, color = as.factor(which_roi)))+
  scale_y_reverse()+
  theme_bw()+
  theme(legend.position = "none")


p_quad<- ggplot(eyetracking.data.with.quadrant.roi %>% filter(subject == '5564e577fdf99b6c901a75a6'))+
  geom_point(aes(x = x, y = y, color = as.factor(which_roi)))+
  scale_y_reverse()+
  theme_bw()+
  theme(legend.position = "none")
#p_quad
p_patch<-p_ROI + p_quad + plot_annotation(tag_levels = 'a')
p_patch
```



```
ggplot(eyetracking.data.with.roi %>% filter(subject == '5564e577fdf99b6c901a75a6'))+
  geom_point(aes(x = x, y = y, color = as.factor(which_roi)), alpha = 0.5)+
  facet_grid(compatibility~target_animal)+
  scale_y_reverse()
```

```
# load audio data
audio.info <- read_csv('info/audio_timing.csv')
```

```
#Calculate average animal onset
animal.onset <- audio.info %>% pull(onset_noun) %>% mean()
instrument.onset <- audio.info %>% pull(onset_instrument) %>% mean()
```

```
# Merge in audio timing information
eyetracking.data.with.roi <- eyetracking.data.with.roi %>%
  mutate(sound = str_split(audio, pattern="/", simplify = T)[,4])

eyetracking.data.with.roi <- eyetracking.data.with.roi %>%
  left_join(audio.info, by="sound")

# Add time window information
eyetracking.data.with.time.windows <- eyetracking.data.with.roi %>%
  ungroup() %>%
  #dplyr::slice_sample(n=100) %>%
  mutate(time.window = case_when(
    t < onset_verb + 200 ~ "pre-verb-onset",
    t <= onset_noun + 200 ~ "post-verb-onset-pre-animal-onset",
    t <= onset_instrument + 200 ~ "post-animal-onset-pre-instrument-onset",
    t <= onset_instrument + 1500 + 200 ~ "post-instrument-onset",
    TRUE ~ "end"
  ),
  time.from.verb = t - onset_verb,
  time.window = factor(time.window, levels = c("pre-verb-onset", "post-verb-onset-pre-animal-onset","po
```

```
# Merge in audio timing information
eyetracking.data.with.quadrant.roi <- eyetracking.data.with.quadrant.roi %>%
  mutate(sound = str_split(audio, pattern="/", simplify = T)[,4]) %>%
  left_join(audio.info, by="sound")

# Add time window information
eyetracking.data.with.time.windows.quadrant <- eyetracking.data.with.quadrant.roi %>%
  ungroup() %>%
  #dplyr::slice_sample(n=100) %>%
  mutate(time.window = case_when(
    t < onset_verb + 200 ~ "pre-verb-onset",
    t <= onset_noun + 200 ~ "post-verb-onset-pre-animal-onset",
    t <= onset_instrument + 200 ~ "post-animal-onset-pre-instrument-onset",
    t <= onset_instrument + 1500 + 200 ~ "post-instrument-onset",
    TRUE ~ "end"
  ),
  time.from.verb = t - onset_verb,
  time.window = factor(time.window, levels = c("pre-verb-onset", "post-verb-onset-pre-animal-onset","po
```

```
ggplot(eyetracking.data.with.time.windows.quadrant %>%
         # filter(subject %in% calib.by.subj[calib.by.subj$mean_percent_in_roi>75,]$subject) %>%
         filter(in.roi.animal)) +
                # calib_by_subj comes from the calibration analysis rmd
  geom_vline(aes(xintercept=screen_width/2))+
  geom_hline(aes(yintercept=screen_height/2))+
  geom_point(aes(x = x, y = y), alpha=0.2)+
  facet_grid(time.window~target_animal)+
  scale_y_reverse()
```

```
#Add time window
eyetracking.data.with.time.windows <- eyetracking.data.with.time.windows %>%
  mutate(t.window = floor(time.from.verb/50)*50)
```

```
#Add time window
eyetracking.data.with.time.windows.quadrant <- eyetracking.data.with.time.windows.quadrant %>%
  mutate(t.window = floor(time.from.verb/50)*50)
```

```
#Summarize data for plotting
eyetracking.figure.2.data <- eyetracking.data.with.time.windows %>%
              # filter(subject %in% calib.by.subj[calib.by.subj$mean_percent_in_roi>50,]$subject) %>% #
  filter(between(t.window , -200, 4000)) %>%
  group_by(subject, compatibility, t.window) %>%
  summarize(p.animal = mean(in.roi.animal), p.instrument = mean(in.roi.instrument)) %>%
  pivot_longer(c('p.animal', 'p.instrument'), names_to="object_type", values_to="prop_fixations") %>%
  #mutate(prop_fixations = if_else(is.na(prop_fixations), 0, prop_fixations)) %>%
  group_by(compatibility, t.window, object_type) %>%
  summarize(M=mean(prop_fixations), SE=sd(prop_fixations)/sqrt(n()))
```

```
## ‘summarise()‘ has grouped output by ’subject’, ’compatibility’. You can override using the ‘.groups‘

## ‘summarise()‘ has grouped output by ’compatibility’, ’t.window’. You can override using the ‘.groups
```

```
fig2<-ggplot(eyetracking.figure.2.data %>%
  mutate(compatibility = factor(compatibility,
                                levels = c("modifier", "equibiased", "instrument" ),
                                labels = c("Modifier", "Equi-biased", "Instrument" ))),
  aes(x=t.window, y=M, ymin=M-SE, ymax=M+SE, color=compatibility, fill=compatibility, linetype=object_t
  geom_ribbon(color=NA, alpha=0.3)+
  geom_line(size=1)+
  scale_color_brewer(palette = "Set1")+
  scale_fill_brewer(palette = "Set1")+
  scale_linetype(labels = c("Animal", "Instrument") )+
  theme_classic() +
  geom_vline(xintercept = animal.onset + 200) +
  geom_vline(xintercept = instrument.onset + 200)+
  labs(y = "Proportion of looks", x = "Time relative to verb onset (ms)")+
  guides(color = guide_legend("Verb bias"),fill = guide_legend("Verb bias"), linetype = guide_legend("G
fig2
```

```
#Summarize data for plotting
eyetracking.figure.3.data <- eyetracking.data.with.time.windows.quadrant %>%
             #  filter(subject %in% calib.by.subj[calib.by.subj$mean_percent_in_roi>50,]$subject) %>% #
  filter(between(t.window , -200, 4000)) %>%
  group_by(subject, compatibility, t.window) %>%
  summarize(p.animal = mean(in.roi.animal), p.instrument = mean(in.roi.instrument)) %>%
  pivot_longer(c('p.animal', 'p.instrument'), names_to="object_type", values_to="prop_fixations") %>%
  #mutate(prop_fixations = if_else(is.na(prop_fixations), 0, prop_fixations)) %>%
  group_by(compatibility, t.window, object_type) %>%
  summarize(M=mean(prop_fixations), SE=sd(prop_fixations)/sqrt(n()))
```

```
## ‘summarise()‘ has grouped output by ’subject’, ’compatibility’. You can override using the ‘.groups‘

## ‘summarise()‘ has grouped output by ’compatibility’, ’t.window’. You can override using the ‘.groups
```

```
fig3<-ggplot(eyetracking.figure.3.data %>%
  mutate(compatibility = factor(compatibility,
                                levels = c("modifier", "equibiased", "instrument" ),
                                labels = c("Modifier", "Equi-biased", "Instrument" ))),
  aes(x=t.window, y=M, ymin=M-SE, ymax=M+SE, color=compatibility, fill=compatibility, linetype=object_t
  geom_ribbon(color=NA, alpha=0.3)+
  geom_line(size=1)+
  scale_color_brewer(palette = "Set1")+
  scale_fill_brewer(palette = "Set1")+
  scale_linetype(labels = c("Animal", "Instrument") )+
  theme_classic() +
  geom_vline(xintercept = animal.onset + 200) +
  geom_vline(xintercept = instrument.onset + 200)+
  labs(y = "Proportion of looks", x = "Time relative to verb onset (ms)")+
  guides(color = guide_legend("Verb bias"),fill = guide_legend("Verb bias"), linetype = guide_legend("Ga
fig3
```

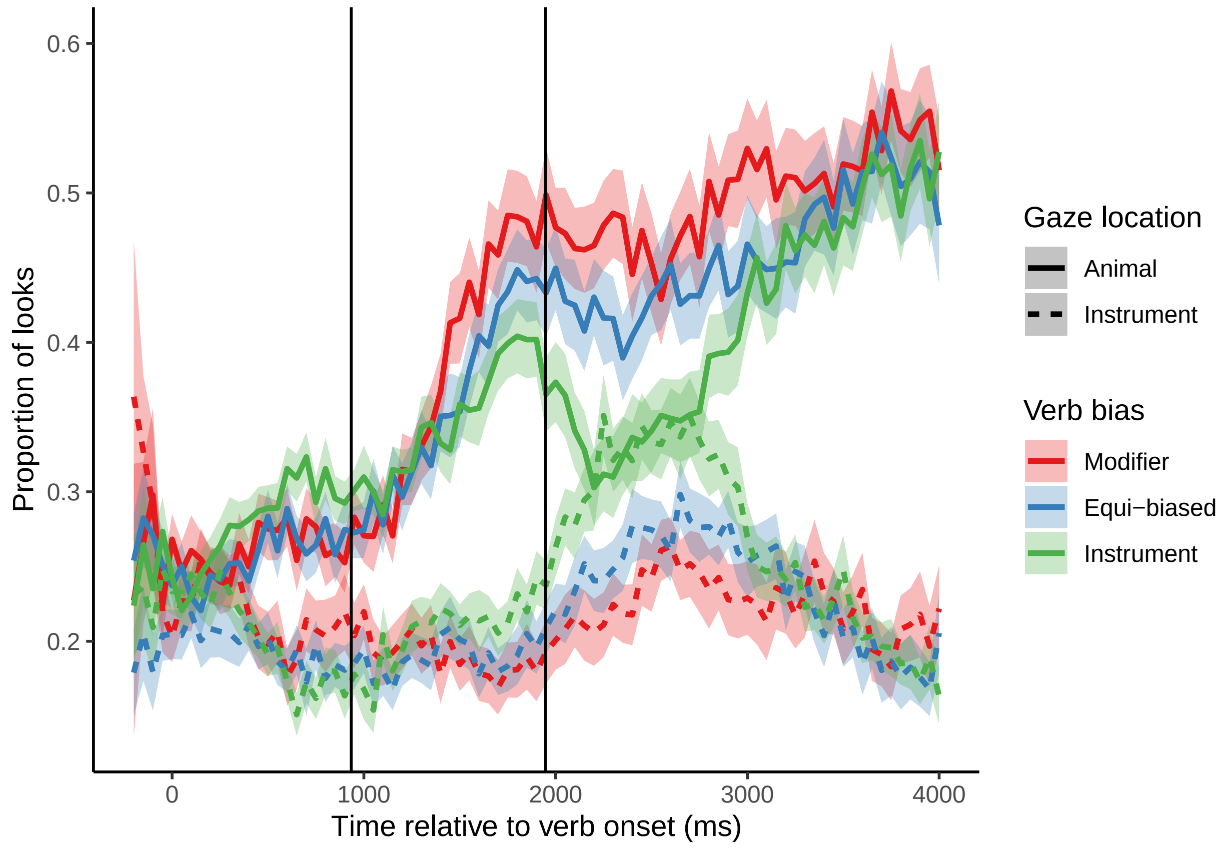


Figure 1: Timecourse of eye-gaze to target animal and target instrument by verb bias condition with gaze categorized based on which quadrant of the screen the coordinates fall in (as opposed to a bounding box around the image). Vertical lines indicate average onsets of animal and instrument offset by 200ms.

```
#Summarize fixations on target and instrument
eyetracking.window.summary.by.trial <- eyetracking.data.with.time.windows.quadrant %>%
  group_by(subject, trialID, sound, compatibility, time.window) %>%
  summarize(prop.fixations.animal = sum(in.roi.animal) / n(),
            prop.fixations.instrument = sum(in.roi.instrument) / n()) %>%
  mutate(compatibility = factor(compatibility))
```

```
## `summarise()` has grouped output by 'subject', 'trialID', 'sound', 'compatibility'. You can override
```

```
# Add orthogonal contrasts to model
contrasts(eyetracking.window.summary.by.trial$compatibility) <- cbind(c(-2/3, 1/3, 1/3), c(0, -1/2, 1/2
data.time.window.3 <- eyetracking.window.summary.by.trial %>% filter(time.window == "post-instrument-ons

model.time.window.3 <- lmer(prop.fixations.animal ~ compatibility + (1 | subject) + (1 | trialID), data=

#summary(model.time.window.3)
```

```
#E4_ET_model1_tab = broom.mixed::tidy(model.time.window.1)
#E4_ET_model2_tab = broom.mixed::tidy(model.time.window.2)
E4_ET_model3_tab = broom.mixed::tidy(model.time.window.3)

E4_ET_model3_c1 = E4_ET_model3_tab %>% filter(term == "compatibility1")
E4_ET_model3_c2 = E4_ET_model3_tab %>% filter(term == "compatibility2")
```

Categorizing gaze location based on which of the four quadrants of the screen the coordinates fell in, increases the overall proportions of fixations (see Figure @ref(fig:E4-gaze-timecourse-fig-quadrants)). In the *post-instrument* window, participants looked more at the target animal in the modifier-biased condition and the equi-biased conditions relative to the instrument-biased condition ($b = 0.08$, $SE = 0.02$, $p < 0.01$) and marginally so in the modifier biased condition relative to the equi-biased condition ($b = 0.04$, $SE = 0.02$, $p = 0.05$). Effect size estimates appeared somewhat larger and noise was somewhat reduced when using the quadrant categorization relative to the bounding box-based ROIs.