exp1_stats

Yiming Lu

2025-06-24

setup

This file contains the statistical analysis performed on experiment 1. The data imported here are already sorted. The file sorting experiment data is titled 'exp1_sort_data.Rmd'. We do all the ploting and stats here.

```
pre_test <- read.delim('pre_test_results.txt') %>% rename(prob_p=Response.100)
pre_test_subjects <- unique(pre_test$subject)
post_test <- read.delim('post_test_results.txt') %>% filter(subject %in% pre_test_subjects) %>% rename(
neutral <- read.delim('neutral_results.txt')
```

```
pre_test_participants <- unique(pre_test$subject)

# save figures for the pre-test results
for (part in 1:length(pre_test_participants)) {
  part_id <- pre_test_participants[part]
  df_part <- pre_test %>% filter(subject==part_id)

  get_baseline_plots(df=df_part, par=as.character(part_id), session="pre_test")
}

# get average results
pre_test %>%
  group_by(VOT, f0) %>%
  summarise(prob_p=mean(prob_p), .groups='drop') %>%
  get_baseline_plots(par='average', session="pre_test")

# linear mixed effect model for pre-test

m_pre <- lmer(prob_p ~ VOT * f0 + (1 + VOT * f0 | subject), data=pre_test)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
m_pre_results <- tidy(m_pre) %>% filter(effect=="fixed")
m_pre_results
```

```
## # A tibble: 4 x 8
##   effect group term        estimate std.error statistic    df       p.value
##   <chr>  <chr> <chr>          <dbl>     <dbl>     <dbl> <dbl>         <dbl>
## 1 fixed  <NA>  (Intercept) -26.5       5.66      -4.69   16.9 0.000216
## 2 fixed  <NA>  VOT          15.4       1.31      11.7    16.9 0.00000000156
## 3 fixed  <NA>  f0            2.96      0.437      6.78   27.6 0.000000252
## 4 fixed  <NA>  VOT:f0       -0.0812    0.0985    -0.824  29.2 0.417
```

```r
post_test_participants <- unique(post_test$subject)

# save figures for the post-test results
for (part in 1:length(post_test_participants)) {
  part_id <- post_test_participants[part]
  df_part <- post_test %>% filter(subject==part_id)

  get_baseline_plots(df=df_part, par=as.character(part_id), session="post_test")
}
# get average results
post_test %>%
  group_by(VOT, f0) %>%
  summarise(prob_p=mean(prob_p), .groups='drop') %>%
  get_baseline_plots(par='average', session="post_test")

# linear mixed effect model for post-test

m_post <- lmer(prob_p ~ VOT * f0 + (1 + VOT * f0 | subject), data=post_test)
```

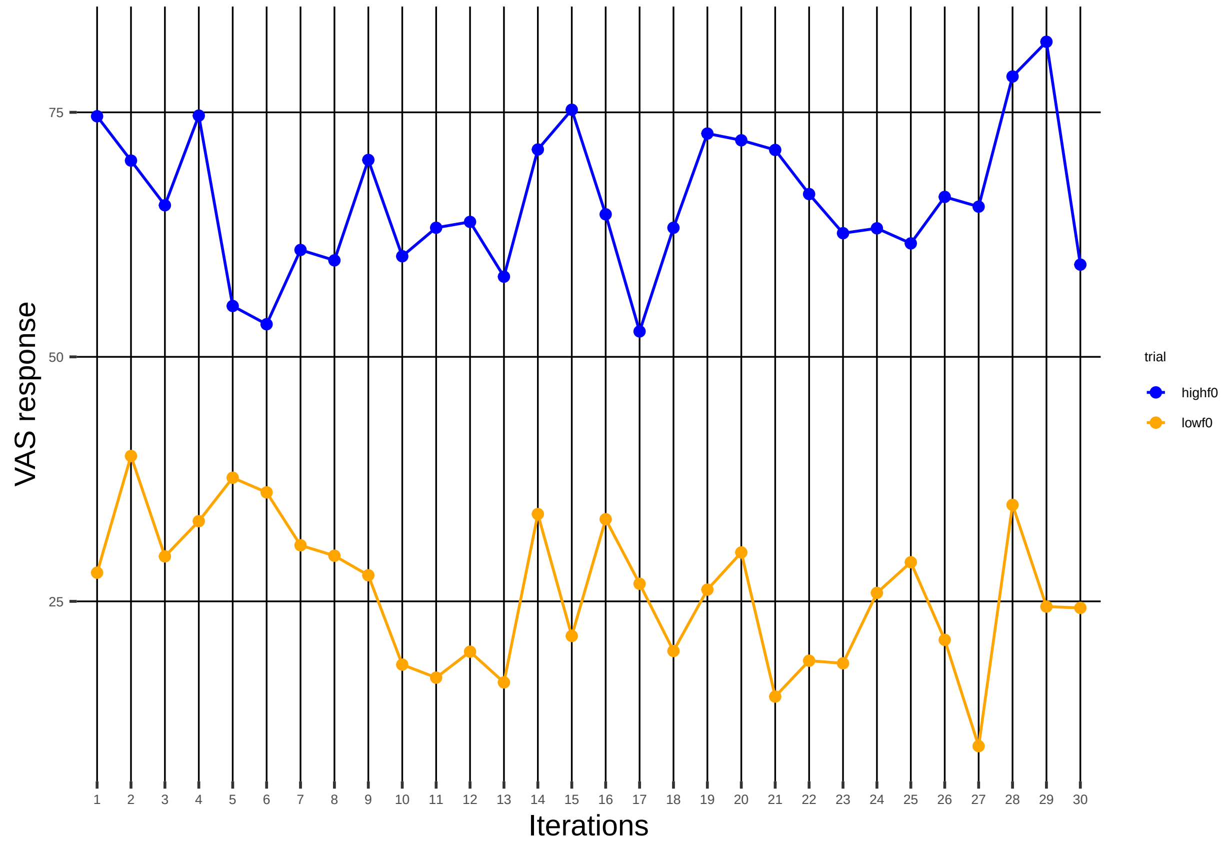
```
## boundary (singular) fit: see help('isSingular')
```

```r
m_post_results <- tidy(m_post) %>% filter(effect=="fixed")
m_post_results
```

```
## # A tibble: 4 x 8
##   effect group term        estimate std.error statistic    df     p.value
##   <chr>  <chr> <chr>          <dbl>     <dbl>     <dbl> <dbl>       <dbl>
## 1 fixed  <NA>  (Intercept) -23.1        7.63     -3.03   10.6 0.0118
## 2 fixed  <NA>  VOT          15.0        1.56      9.65   11.1 0.000000990
## 3 fixed  <NA>  f0            2.66       0.622     4.29   15.1 0.000641
## 4 fixed  <NA>  VOT:f0       -0.0664     0.117    -0.565  21.3 0.578
```

We start by performing summary statistics and linear mixed effect models for pre/post tests. We found that f0 and VOT both had a significant influence on the VAS response. No significant interactions between VOT and f0 was found.

```r
# get data: only test trials, but wouldn't we need to run some stats on the exposure trials as well?

#let us plot the trajectory
neutral_aver <- neutral %>%
  group_by(Display, trial) %>%
  summarise(prob_p=mean(prob_p), .groups='drop') %>%
  mutate(ID='averaged')

# plot the same for each participant
neutral_participants <- unique(neutral$ProlificID)
for (part in 1:length(neutral_participants)) {
  part_id <- neutral_participants[part]
  df_part <- neutral %>% filter(ProlificID==part_id)

  traj_test(df=df_part,block='neutral',par=part_id)
}

traj_test(df=neutral_aver,block='neutral',par='average')
```



```
# treatment coding
df_neutral <- neutral %>%
    mutate(Display_num = str_extract(Display, "\\d+")) %>%
    mutate(Display = as.numeric(Display_num)) %>%
    mutate(trial=factor(trial),
           ID=as.factor(ProlificID))
contrasts(df_neutral$trial) <- 'contr.treatment'
df_neutral <- within(df_neutral, trial <- relevel(trial, ref='lowf0'))
df_neutral$trial <- as.ordered(df_neutral$trial) # this is essential
# we want to set the reference level as low f0.

# Step 1: compare models with different fixed terms (parametric/smooth, impact of f0)
m.neutral1 <- bam(prob_p ~
          trial #non-smooth term
          + s(Display) #reference smooth
          + s(Display, by=trial) # difference smooth
          ,
          data=df_neutral,
          method='ML') # this is the model with f0 as a fixed non-smooth effect and a difference smooth
summary(m.neutral1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## prob_p ~ trial + s(Display) + s(Display, by = trial)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   45.942      1.120   41.02   <2e-16 ***
## trial.L       28.223      1.584   17.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                          edf Ref.df     F p-value
## s(Display)             1.001  1.002 3.949  0.0473 *
## s(Display):trialhighf0 1.000  1.000 3.771  0.0524 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.267   Deviance explained =   27%
## -ML = 4313.2  Scale est. = 1099.9    n = 877
```

```
m.neutral2 <- bam(prob_p ~
          + s(Display)
          ,
          data=df_neutral,
          method='ML') # what happens if f0 is removed from the parametric term and the smooth term?
compareML(m.neutral1, m.neutral2)
```

```
## m.neutral1: prob_p ~ trial + s(Display) + s(Display, by = trial)
##
## m.neutral2: prob_p ~ +s(Display)
##
## Chi-square test of ML scores
## -----
##        Model    Score Edf Difference    Df  p.value Sig.
## 1 m.neutral2 4450.609   3
## 2 m.neutral1 4313.221   6    137.388 3.000  < 2e-16  ***
##
## AIC difference: -270.78, model m.neutral1 has lower AIC.
```

```
# Comparing the model with a non-smooth fixed effect of f0 but not a difference smooth
m.neutral3 <- bam(prob_p ~
          trial
          + s(Display)
          ,
          data=df_neutral,
          method='ML')

compareML(m.neutral1, m.neutral3)
```

```
## m.neutral1: prob_p ~ trial + s(Display) + s(Display, by = trial)
##
## m.neutral3: prob_p ~ trial + s(Display)
##
```

```
## Chi-square test of ML scores
## -----
##        Model    Score Edf Difference    Df p.value Sig.
## 1 m.neutral3 4315.114   4
## 2 m.neutral1 4313.221   6      1.892 2.000   0.151
##
## AIC difference: -1.78, model m.neutral1 has lower AIC.
```

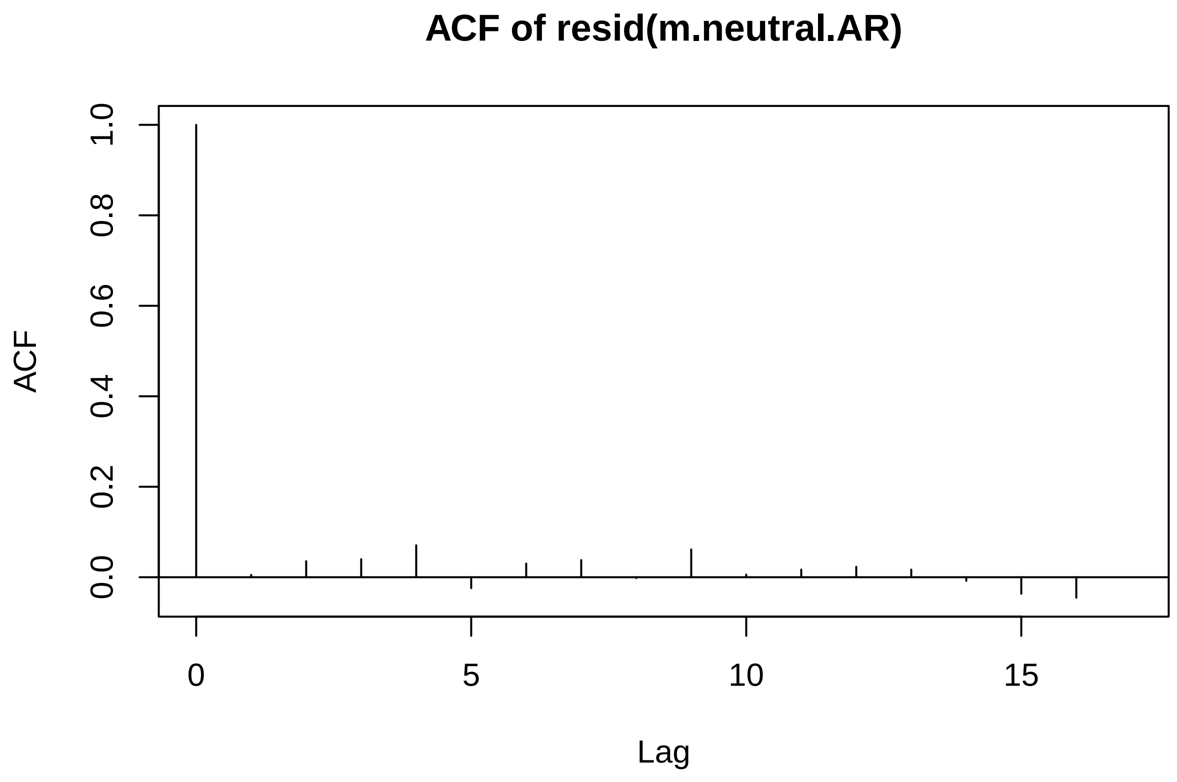
```
# in this case, not a significant difference was found through chi-square tests/AIC
# Conclusion: m.neutral3 (the model including a parametric term of f0 without a difference smooth) is t

# Step 2: add random effect structure
# but still, we need a comparison between different models wity random effects to find the model with t
m.neutral.ran.intercept <- bam(prob_p ~
          trial #non-smooth term
          + s(Display) #reference smooth
          + s(ID, bs='re') # random intercept
          , # but this model might be unnecessarily complex
          data=df_neutral,
          method='ML')

m.neutral.ran.slope <- bam(prob_p ~
          trial #non-smooth term
          + s(Display) #reference smooth
          + s(ID, Display, bs='re') #random slope
          , # but this model might be unnecessarily complex
          data=df_neutral,
          method='ML')

m.neutral.ran.smooth <- bam(prob_p ~
          trial #non-smooth term
          + s(Display) #reference smooth
          + s(Display, ID, bs='fs', xt='cr', m=1, k=10) # random smooths
          , # but this model might be unnecessarily complex, we will do a model comparison
          data=df_neutral,
          method='ML')
# compare through AIC: because these models are not nested.
AIC(m.neutral.ran.intercept, m.neutral.ran.slope, m.neutral.ran.smooth, m.neutral3)
```

```
##                                df      AIC
## m.neutral.ran.intercept 17.438573 8409.230
## m.neutral.ran.slope     17.142619 8492.970
## m.neutral.ran.smooth    39.940184 8390.904
## m.neutral3               4.004446 8638.220
```

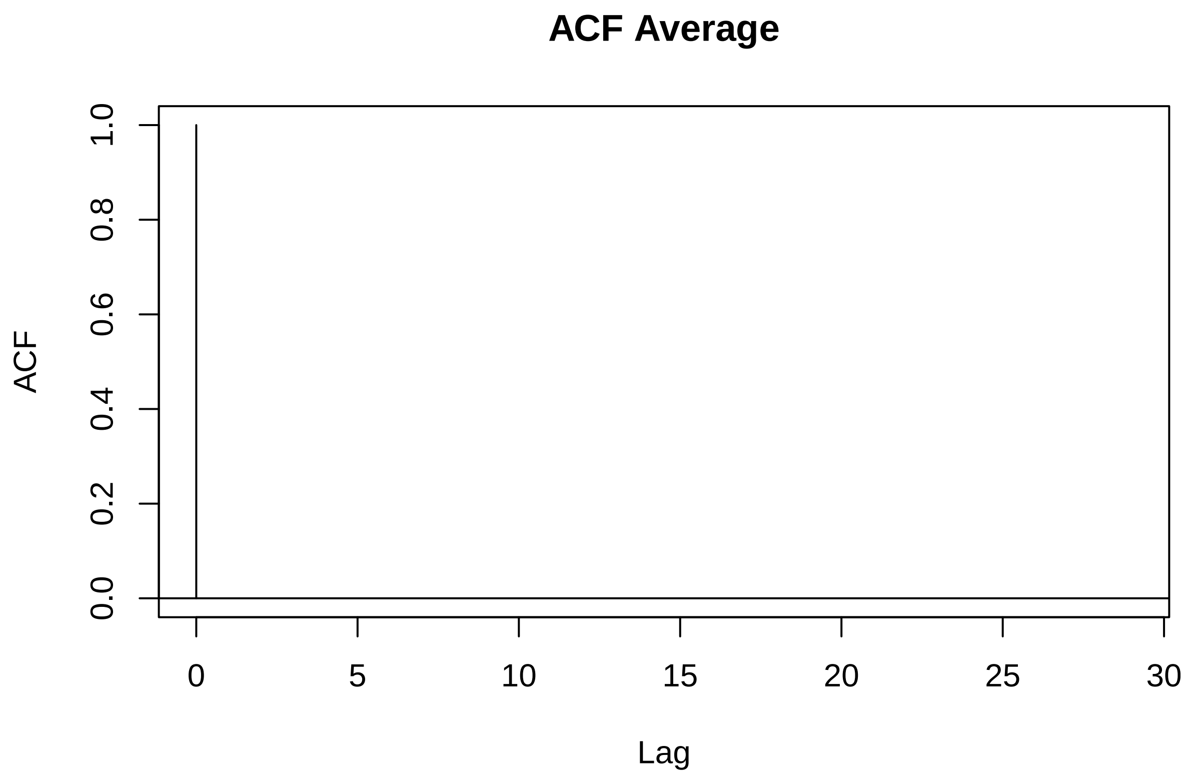
```
# Conclusion: random smooth model has the lowest AIC

# create an AR1 model
# 1. mark the starting point
df_neutral$start.event <- df_neutral$Display == 1
# 2. estimate roughly the degree of autocorrelation
r1 <- start_value_rho(m.neutral3)
# 3. create an AR model
m.neutral.AR <- bam(prob_p ~
          trial #non-smooth term
          + s(Display) #reference smooth
          + s(Display, by=trial) # difference smooth
          ,
          data=df_neutral,
          method='fREML',
          rho=r1,
          AR.start=df_neutral$start.event) # the same as m.neutral1, but with residue error autocorrela

summary(m.neutral.AR)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## prob_p ~ trial + s(Display) + s(Display, by = trial)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   45.965      1.423    32.3   <2e-16 ***
## trial.L       28.084      1.357    20.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                          edf Ref.df     F p-value
## s(Display)             1.494  1.835 2.943  0.0972 .
## s(Display):trialhighf0 1.000  1.000 5.517  0.0191 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.268   Deviance explained = 27.1%
## fREML = 4277.4  Scale est. = 1090.7    n = 877
```

```
acf_plot(resid(m.neutral.AR), split_by=list(df_neutral$ID))
```



```r
acf_resid(m.neutral.AR, split_pred = "(AR.start)")
```



```r
# it seems that the AR1 model reduced residue autocorrelation; why is the residue 0? It can't possibly

# however, the random smooth model seems to capture a higher degree of variance, which one should we us
```

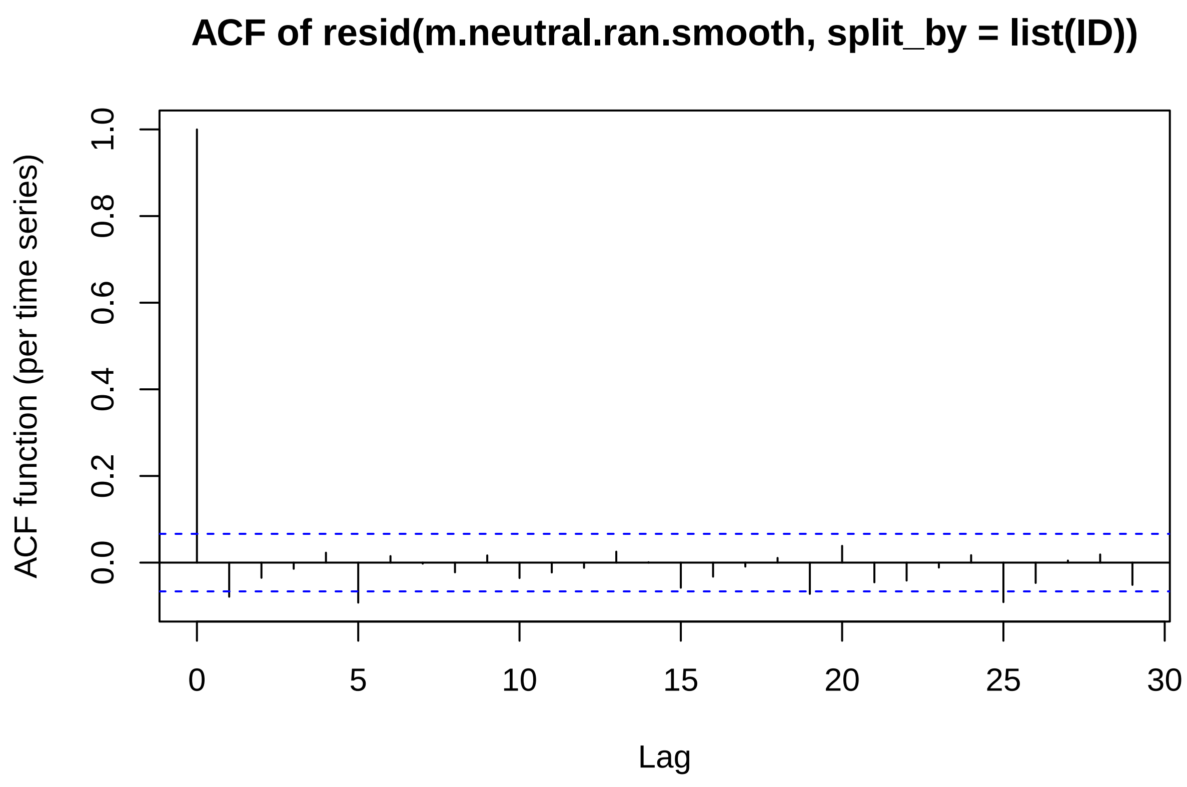
```
AIC(m.neutral.AR, m.neutral.ran.smooth)
```

```
##                             df      AIC
## m.neutral.AR          5.835391 8575.991
## m.neutral.ran.smooth 39.940184 8390.904
```

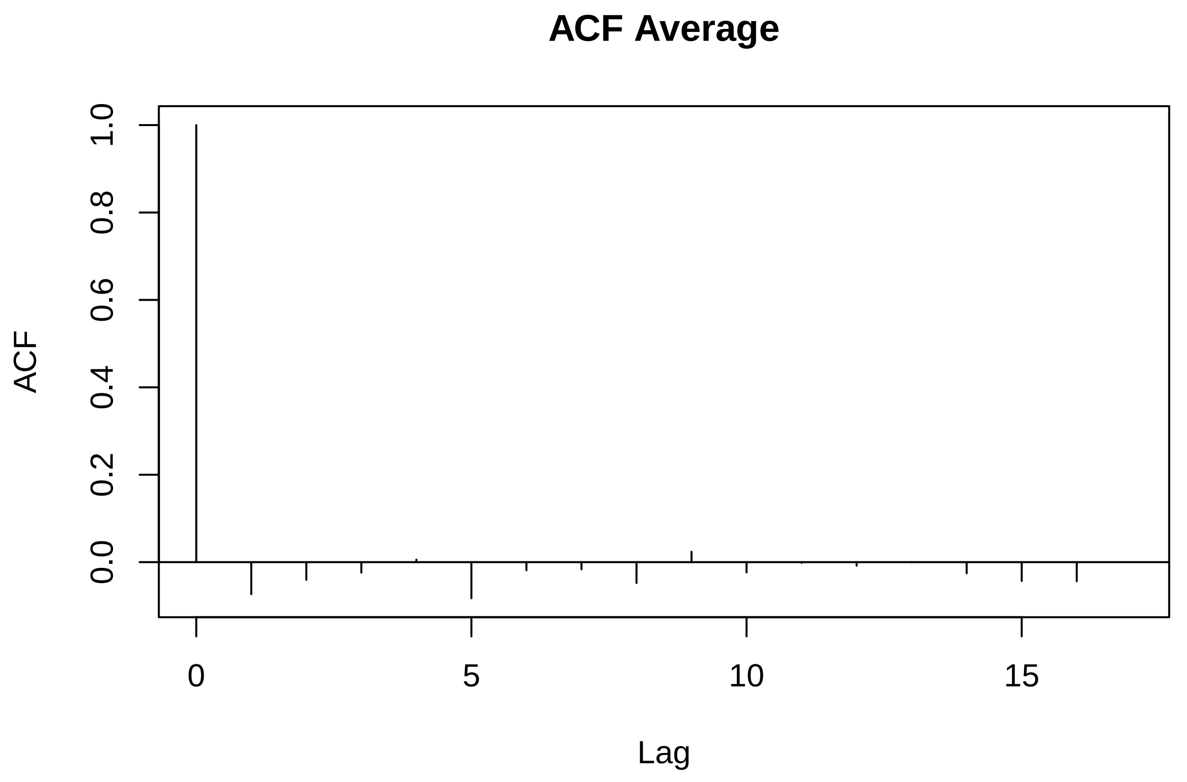
```
# Conclusion: the random smooth model is the best; it has lower AIC and explains more variance than the


# get the acf plots for the best model
summary(m.neutral.ran.smooth)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## prob_p ~ trial + s(Display) + s(Display, ID, bs = "fs", xt = "cr",
##     m = 1, k = 10)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   46.018      3.421   13.45   <2e-16 ***
## trial.L       28.170      1.348   20.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                  edf  Ref.df     F p-value
## s(Display)      1.56   1.879 0.669   0.555
## s(Display,ID) 30.24 148.000 2.267  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.471   Deviance explained =   49%
## -ML = 4212.5  Scale est. = 794.79    n = 877
```

```
acf_plot(resid(m.neutral.ran.smooth, split_by=list(ID)))
```



```
acf_resid(m.neutral.ran.smooth, split_pred=c('ID'))
```

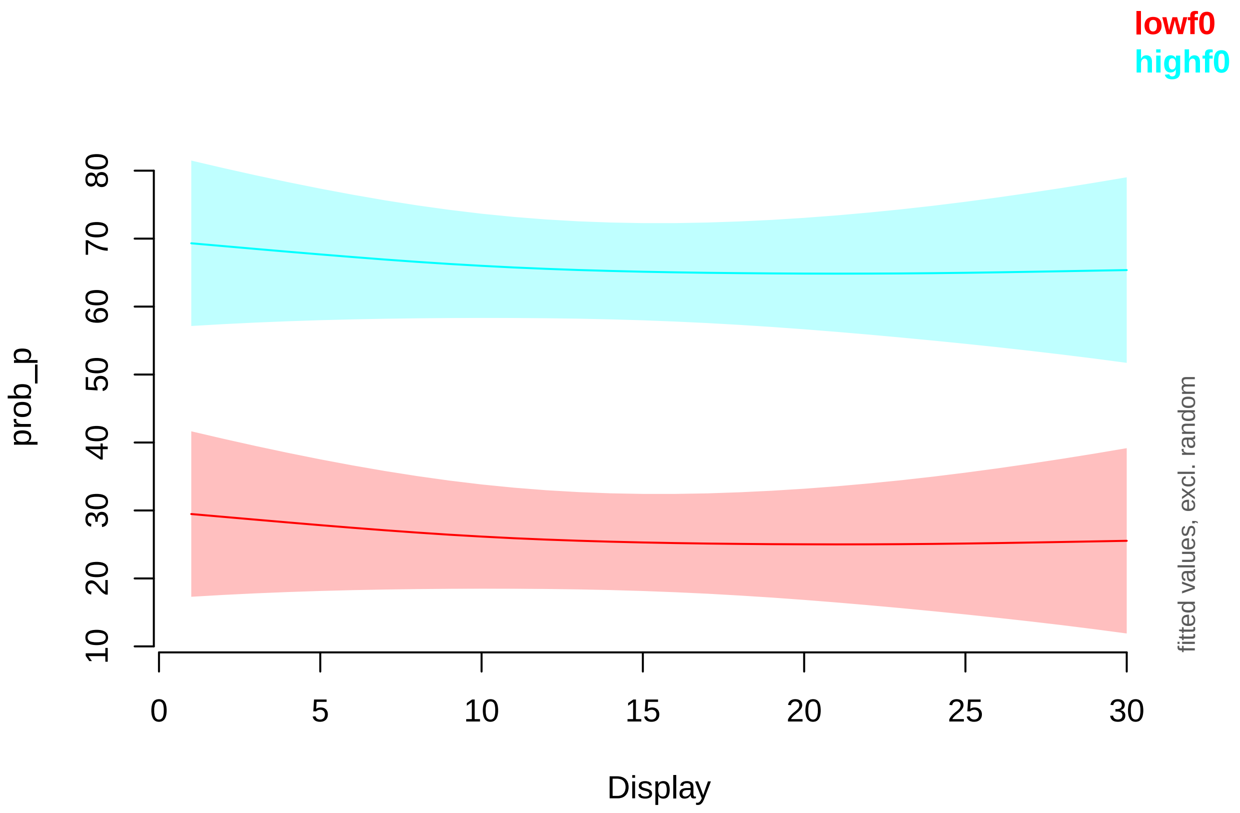


```
# get the diff between two f0 levels in the best model
# we want to plot the differences between these two levels
plot_smooth(m.neutral.ran.smooth,
            view='Display',
            plot_all='trial',
            rug=F,
            rm.ranef=T
            )
```

```
## Summary:
##  * trial : factor; set to the value(s): highf0, lowf0.
##  * Display : numeric predictor; with 30 values ranging from 1.000000 to 30.000000.
##  * ID : factor; set to the value(s): 5ba288c63fd80800016308c4. (Might be canceled as random effect, c
##  * NOTE : The following random effects columns are canceled: s(Display,ID)
##
```

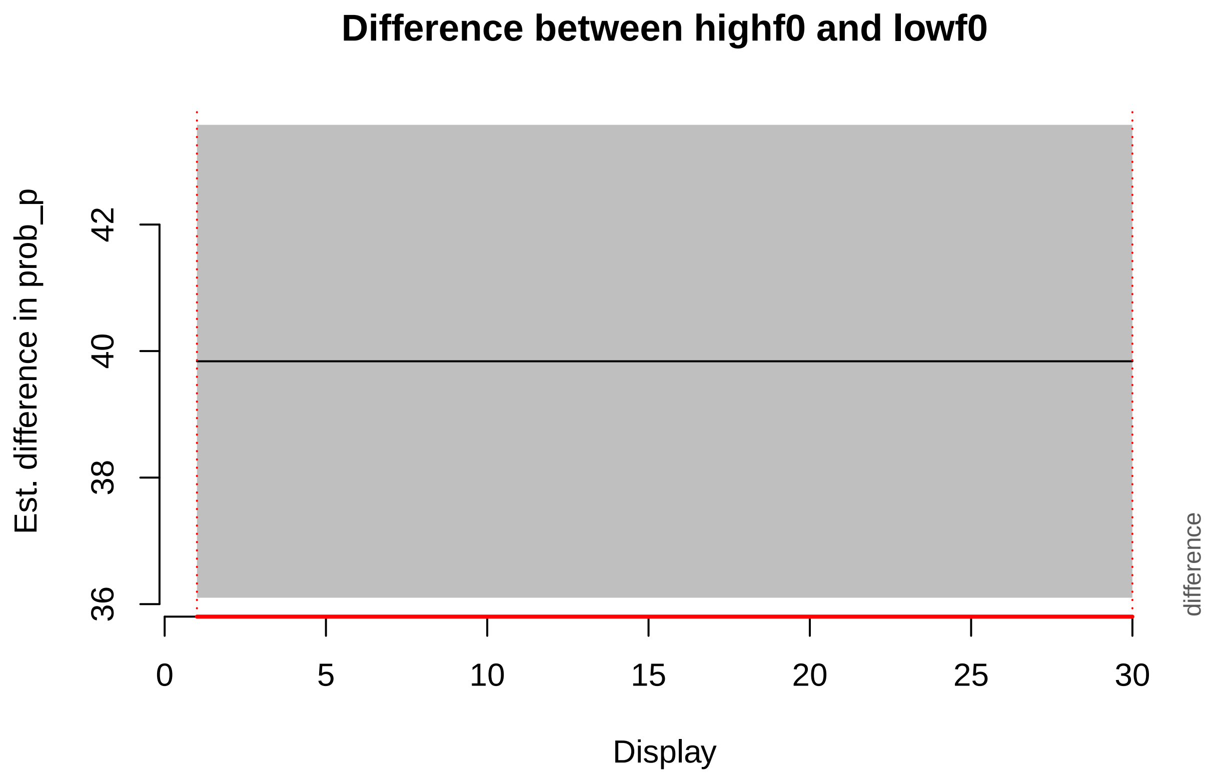
```
get_pred <- plot_smooth(m.neutral.ran.smooth,
             view='Display',
             plot_all='trial',
             rug=F,
             rm.ranef=T
             )$fv
```

```
## Summary:
##  * trial : factor; set to the value(s): highf0, lowf0.
##  * Display : numeric predictor; with 30 values ranging from 1.000000 to 30.000000.
##  * ID : factor; set to the value(s): 5ba288c63fd80800016308c4. (Might be canceled as random effect, c
##  * NOTE : The following random effects columns are canceled: s(Display,ID)
##
```



```
plot_diff(m.neutral.ran.smooth,
          view='Display',
          comp=list(trial=c("highf0","lowf0"))
)
```

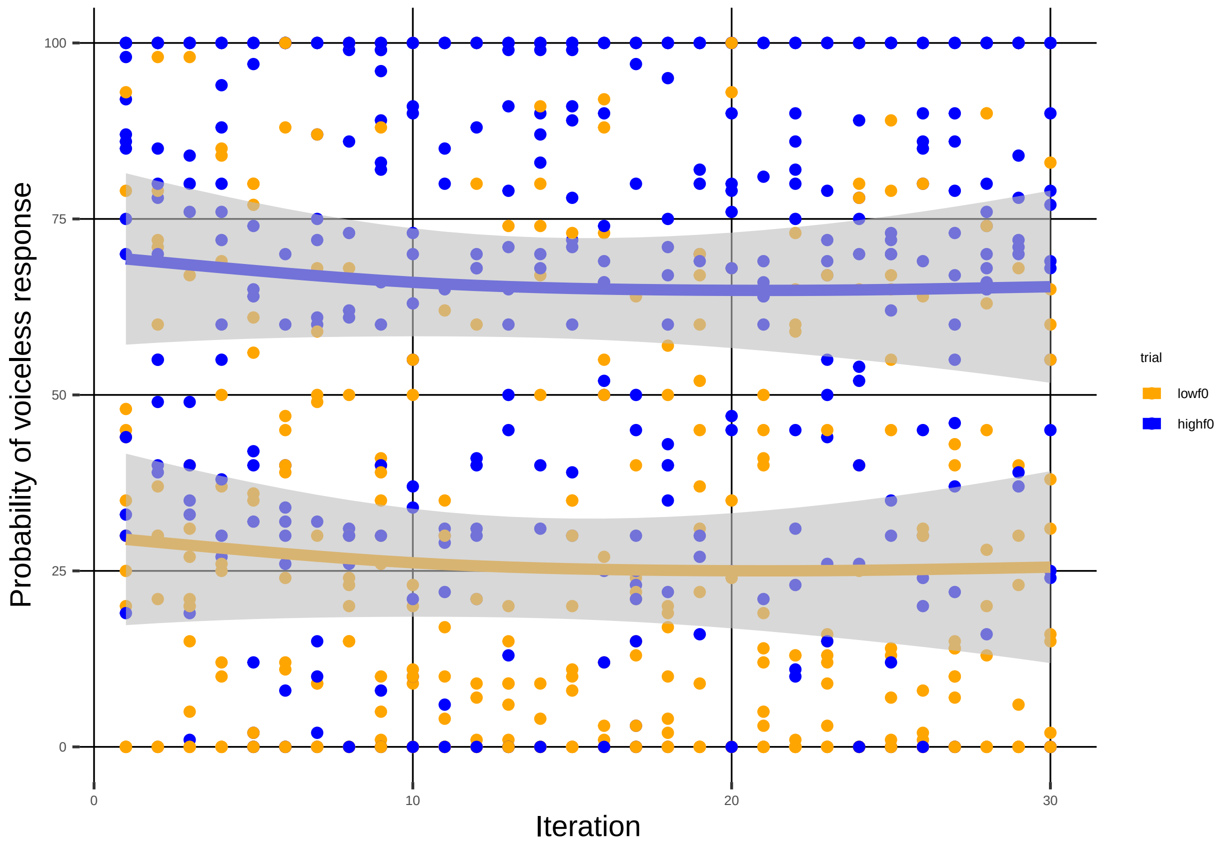
```
## Summary:
##  * Display : numeric predictor; with 100 values ranging from 1.000000 to 30.000000.
##  * ID : factor; set to the value(s): 5ba288c63fd80800016308c4.
```



```
##
## Display window(s) of significant difference(s):
##  1.000000 - 30.000000
```

```
p_traj_overlay <- ggplot(data=df_neutral, aes(x=Display, y=prob_p, group = trial, color = trial), alpha
    scale_color_manual(values = c("highf0"='blue', "lowf0"='orange')) +
    geom_point() +
    geom_line(data=get_pred, aes(x=Display, y=fit, group = trial, color = trial), size=2, inherit.aes=F
    geom_ribbon(data=get_pred, aes(x=Display, ymin = ll, ymax = ul, group = trial), alpha=.6, fill='gre
    scale_y_continuous(breaks=seq(0,100,by=25)) +
    xlab('Iteration') +
    ylab('Probability of voiceless response') +
    theme(
      panel.background = element_rect(fill='white'),
      panel.grid.major = element_line(color='black', linewidth=0.3),
      legend.text = element_text(size=5),
      legend.title = element_text(size=5),
      legend.key.height = unit(0.4, "cm"),
      legend.key.width = unit(0.3, "cm"),
      axis.text.x = element_text(size=5),
      axis.text.y = element_text(size=5)
    )
```

```
p_traj_overlay
```



```
ggsave(glue("./neutral_figures/trajectory_m_pred.png"), plot=p_traj_overlay, width=1500, height=1000, dp
```

We conclude from the generalized additive mixed effect model that f0 exerts an effect on the overall VAS response, but this effect does not change with exposure. The difference smooth model does not explain more variance than the model with the non-smooth parameter of f0 only. That is, the f0 cue weight does not change over 3 neutral blocks. Regarding random effect, the random smooth model outperform the random intercept and slope models. The line in the above plot shows model prediction, whereas the shade shows the confidence interval.

change of secondary cue before and after exposure

```
pre_theta <- read_delim("./pre_all.txt") %>% mutate(time='before')
```

```
## Rows: 18 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (2): subject, comment
## dbl (13): B1, B2, Slope, X1, X2, Theta, PointVar, exitflag, R, LS, AR, N, st...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
post_theta <- read_delim("./post_all.txt") %>% mutate(time='after')
```

```
## Rows: 15 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (2): subject, comment
## dbl (13): B1, B2, Slope, X1, X2, Theta, PointVar, exitflag, R, LS, AR, N, st...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pre_theta <- pre_theta %>% filter(subject %in% post_theta$subject)
theta <- rbind(pre_theta, post_theta) %>% mutate(time=factor(time))

contrasts(theta$time) <- 'contr.treatment'
theta <- within(theta, time <- relevel(time, ref='before'))
theta$time <- as.ordered(theta$time)


m_theta <- lmer(Theta ~ time + (1|subject), data=theta) # run this when we actually get the data back
tidy(m_theta)
```

```
## # A tibble: 4 x 8
##   effect   group    term            estimate std.error statistic    df   p.value
##   <chr>    <chr>    <chr>              <dbl>     <dbl>     <dbl> <dbl>     <dbl>
## 1 fixed    <NA>     (Intercept)       247.        9.01    27.5    14.0  1.41e-13
## 2 fixed    <NA>     time.L             -9.83     11.8     -0.834  14.0  4.18e- 1
## 3 ran_pars subject  sd__(Intercept)    13.4      NA       NA      NA   NA
## 4 ran_pars Residual sd__Observation    45.6      NA       NA      NA   NA
```

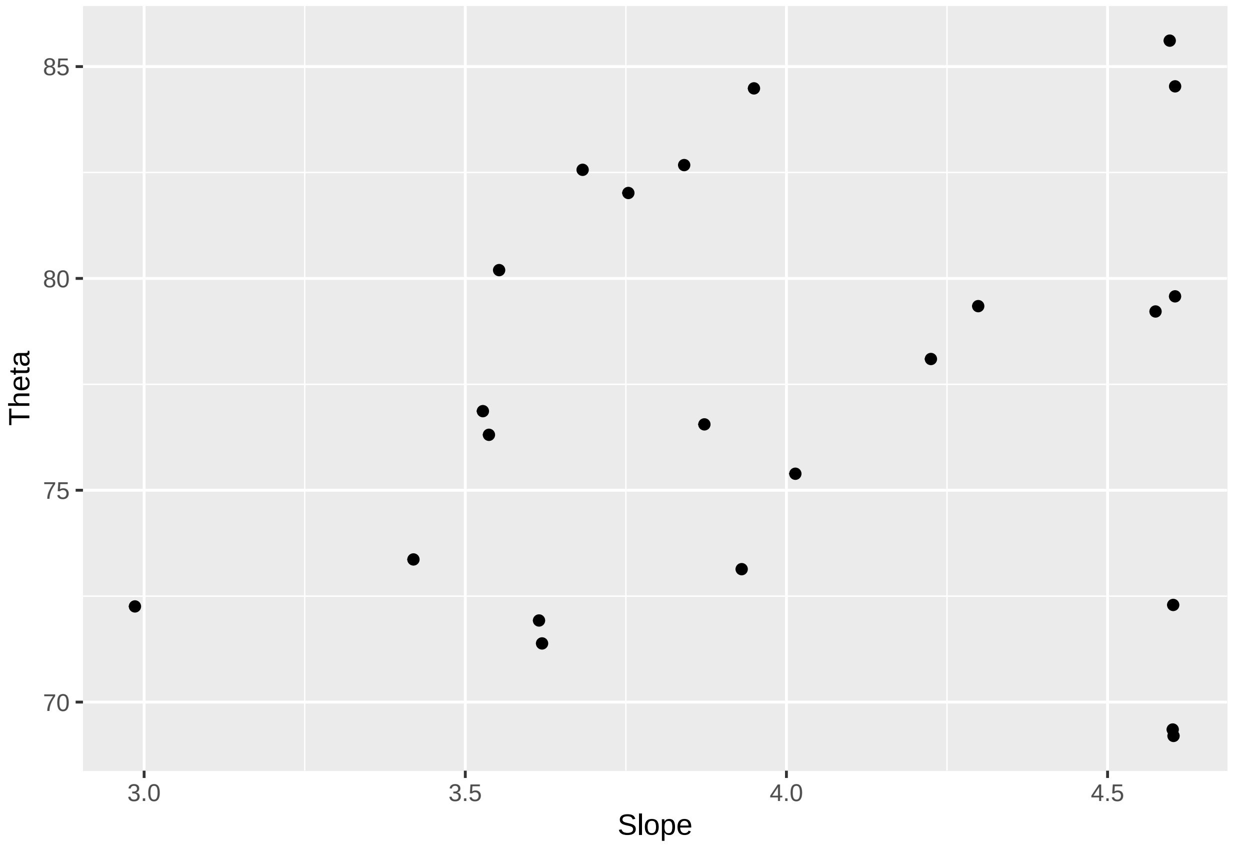
We found no significant differences in theta between pre-test and post-test. That is, no re-weighting can be detected after exposure.

relationship between gradience and theta (secondary cue)

```
# use data from pre-test only
theta <- theta %>% mutate(Slope=log(Slope), Theta=90-(270-Theta)) %>% filter(R > 0.9)
m_theta <- lm(Slope ~ Theta, data=theta) # but is this right?
m_theta_null <- lm(Slope ~ 1, data=theta)
anova(m_theta_null, m_theta)
```

```
## Analysis of Variance Table
##
## Model 1: Slope ~ 1
## Model 2: Slope ~ Theta
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     22 5.1537
## 2     21 5.0569  1  0.096786 0.4019 0.5329
```

```r
# No relationship between slope and theta?
ggplot(data=theta, aes(x=Slope, y=Theta)) +
  geom_point()
```



Surprisingly, we also did not find a relationship between gradience (slope) and secondary cue use (theta). But this could be due to our smaller number of participants.

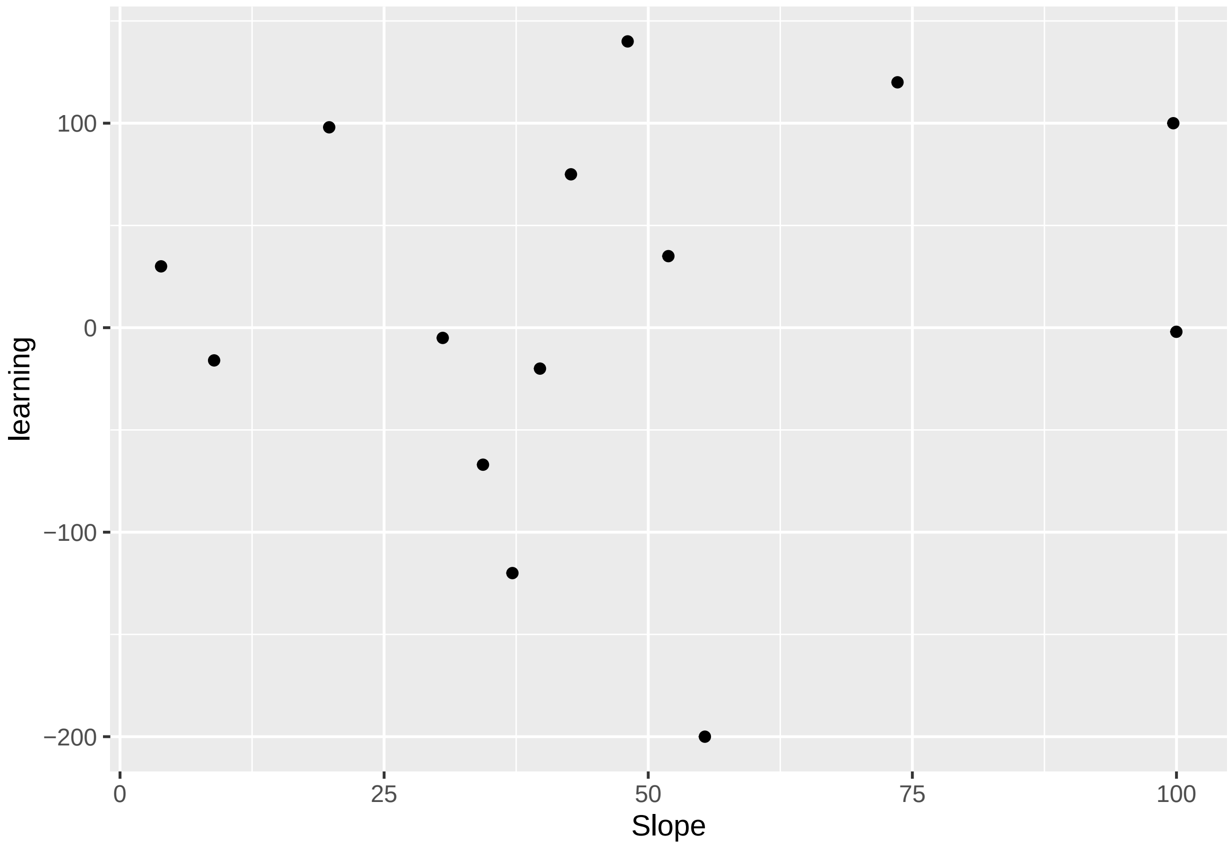
relationship between gradience and adaptation

Below we perform some exploratory analysis for the relationship between gradiency and adaptation # 1. adaptation within exposure block

```r
# get slope from pre-test and get adaptation between 1st and 30th iterations
neutral_compa <-  neutral %>%
  filter(Display %in% c("Neutral 1", "Neutral 30")) %>%
  group_by(Display, ProlificID) %>%
  mutate(diff_prob_p=prob_p[1]-prob_p[2]) %>%
  ungroup() %>%
  dplyr::select(ProlificID, Display, diff_prob_p) %>%
  distinct() %>%
  group_by(ProlificID) %>%
  mutate(learning=diff_prob_p[2]-diff_prob_p[1]) %>% #compare the diff in high-low f0 responses before
  dplyr::select(ProlificID, learning) %>%
  distinct() %>%
  filter(!is.na(learning)) %>%
  rename(subject=ProlificID)

df_adaptation <- pre_theta %>%
  left_join(neutral_compa, by='subject')

m_adaptation <- lm(learning ~ Slope,data=df_adaptation) # but does this make sense? think harder!

tidy(m_adaptation)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)  -12.0      50.5      -0.237   0.817
## 2 Slope          0.520     0.936     0.556   0.589
```

```
ggplot(data=df_adaptation, aes(x=Slope, y=learning)) +
  geom_point()
```



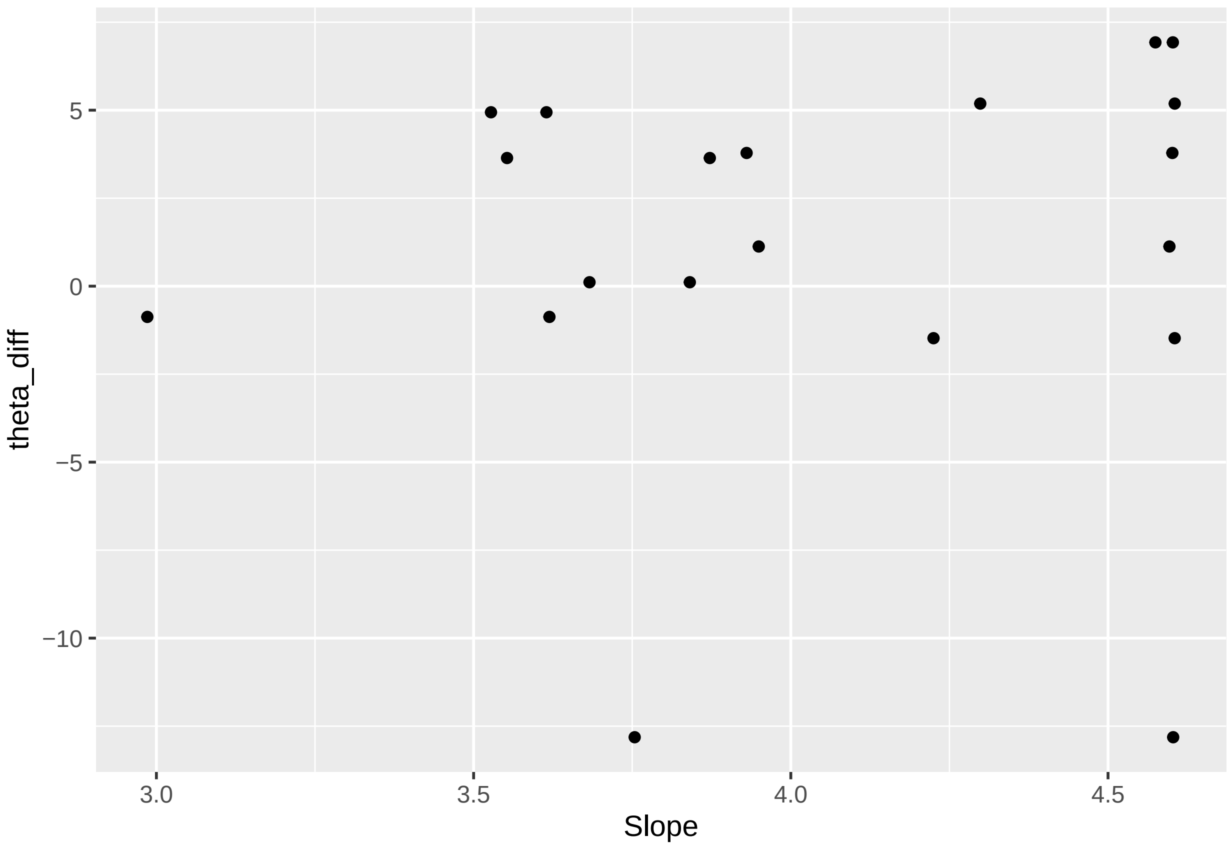
We did not find a significant relationship between adaptation (exemplified by the changes in the difference between high/low f0 test trial VAS response) and gradience (slope)

2. adaptation for change in theta

```
theta_compa <-
  theta %>%
  group_by(subject) %>%
  mutate(theta_diff = Theta[2]-Theta[1])

m_adap_theta <- lm(theta_diff ~ Slope, data=theta_compa)

tidy(m_adap_theta)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   -1.98      10.7     -0.184   0.856
## 2 Slope          0.749      2.63     0.285   0.779
```

```
ggplot(data=theta_compa, aes(x=Slope, y=theta_diff)) +
  geom_point()
```



# We also did not find a significant relationship between adaptation (change in theta before and after exposure) and gradience (slope).

relationship between variability and theta (secondary cue)

```
trial_pre <- read_delim('trial_pre_all.txt')
```

```
## Rows: 3382 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): ProlificID
## dbl (4): f0, VOT, Response_100, pred
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

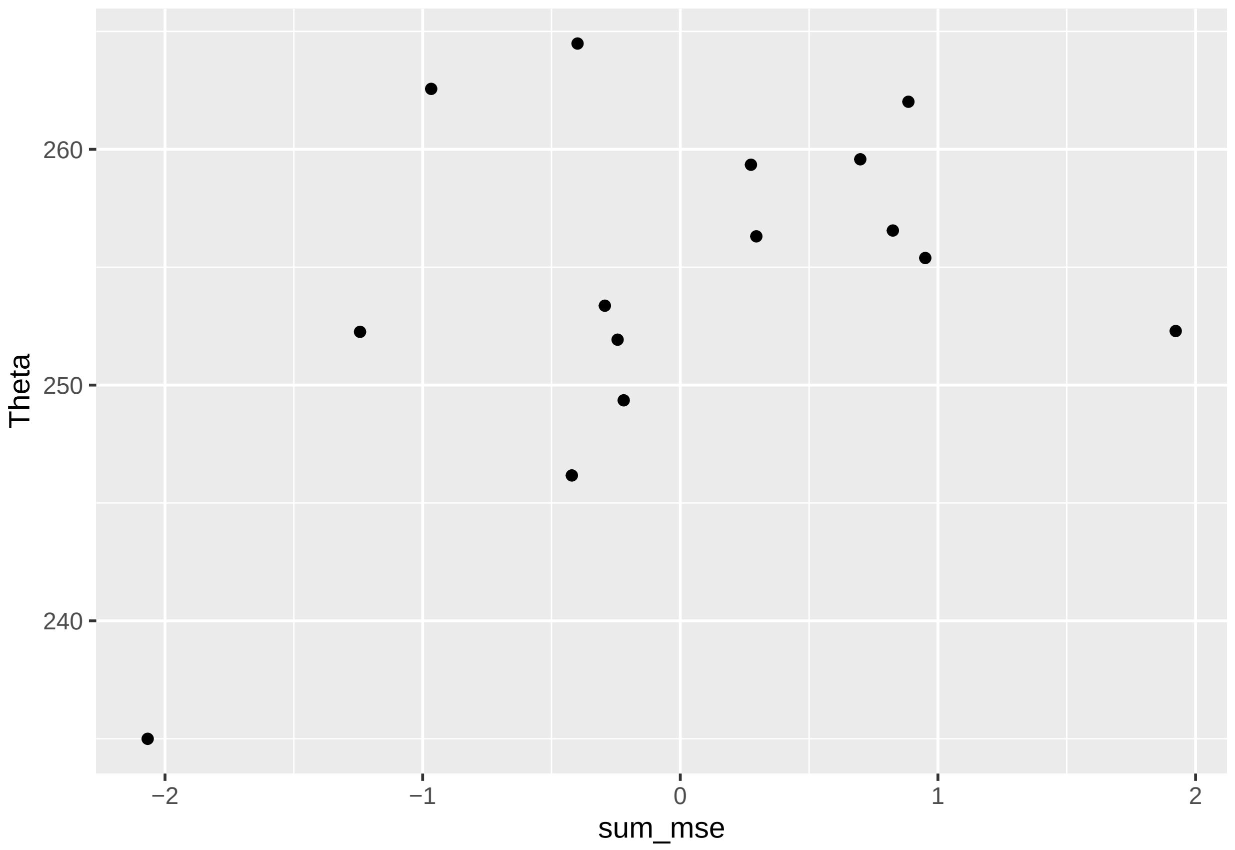
```
trial_post <- read_delim('trial_post_all.txt')
```

```
## Rows: 2783 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): ProlificID
## dbl (4): f0, VOT, Response_100, pred
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
trial_pre <- trial_pre %>% filter(ProlificID %in% trial_post$ProlificID)

var_pre <- get_var(trial_pre) %>% left_join(pre_theta, by='subject')
var_post <- get_var(trial_post) %>% left_join(post_theta, by='subject')

m_var_theta <- lm(sum_mse ~ Theta, data=var_pre)
summary(m_var_theta)
```

```
##
## Call:
## lm(formula = sum_mse ~ Theta, data = var_pre)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.46388 -0.55236  0.08523  0.40312  2.05437
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -15.56812    8.46354  -1.839   0.0888 .
## Theta         0.06119    0.03325   1.840   0.0887 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9243 on 13 degrees of freedom
## Multiple R-squared:  0.2067, Adjusted R-squared:  0.1456
## F-statistic: 3.386 on 1 and 13 DF,  p-value: 0.08868
```

```
ggplot(data=var_pre, aes(x=sum_mse, y=Theta)) +
  geom_point()
```



# We found a marginally significant relationship between variability and secondary cue use (theta). It seems that larger variability corresponds with a stronger primary cue.

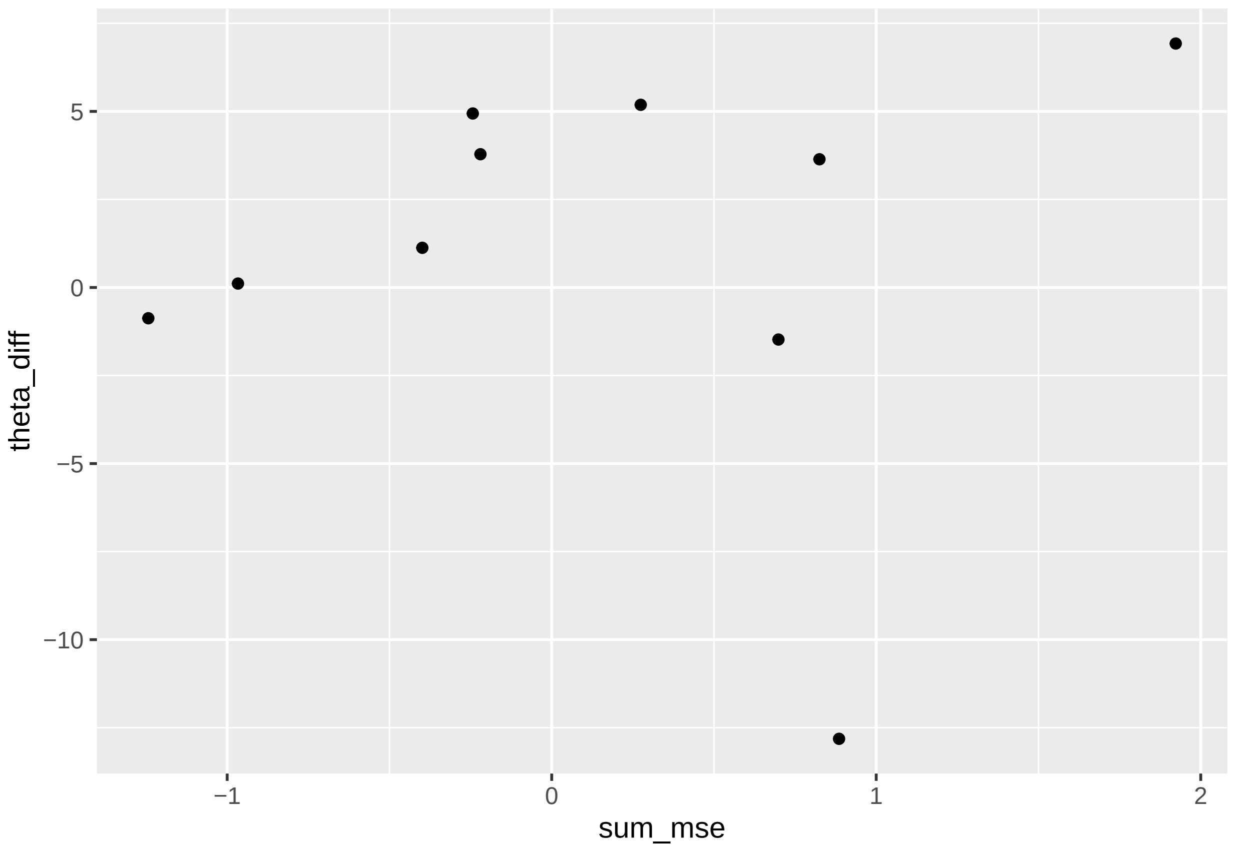
relationship between variability and adaptation

theta diff

```
theta_var <- theta_compa %>%
  ungroup() %>%
  dplyr::select(subject, theta_diff) %>%
  distinct() %>%
  left_join(var_pre, by='subject')

m_adap_var <- lm(theta_diff ~ sum_mse, data=theta_var)
tidy(m_adap_var)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)    1.01       1.91     0.531   0.610
## 2 sum_mse        0.272      2.07     0.131   0.899
```

```
ggplot(data=theta_var, aes(x=sum_mse, y=theta_diff)) +
  geom_point()
```

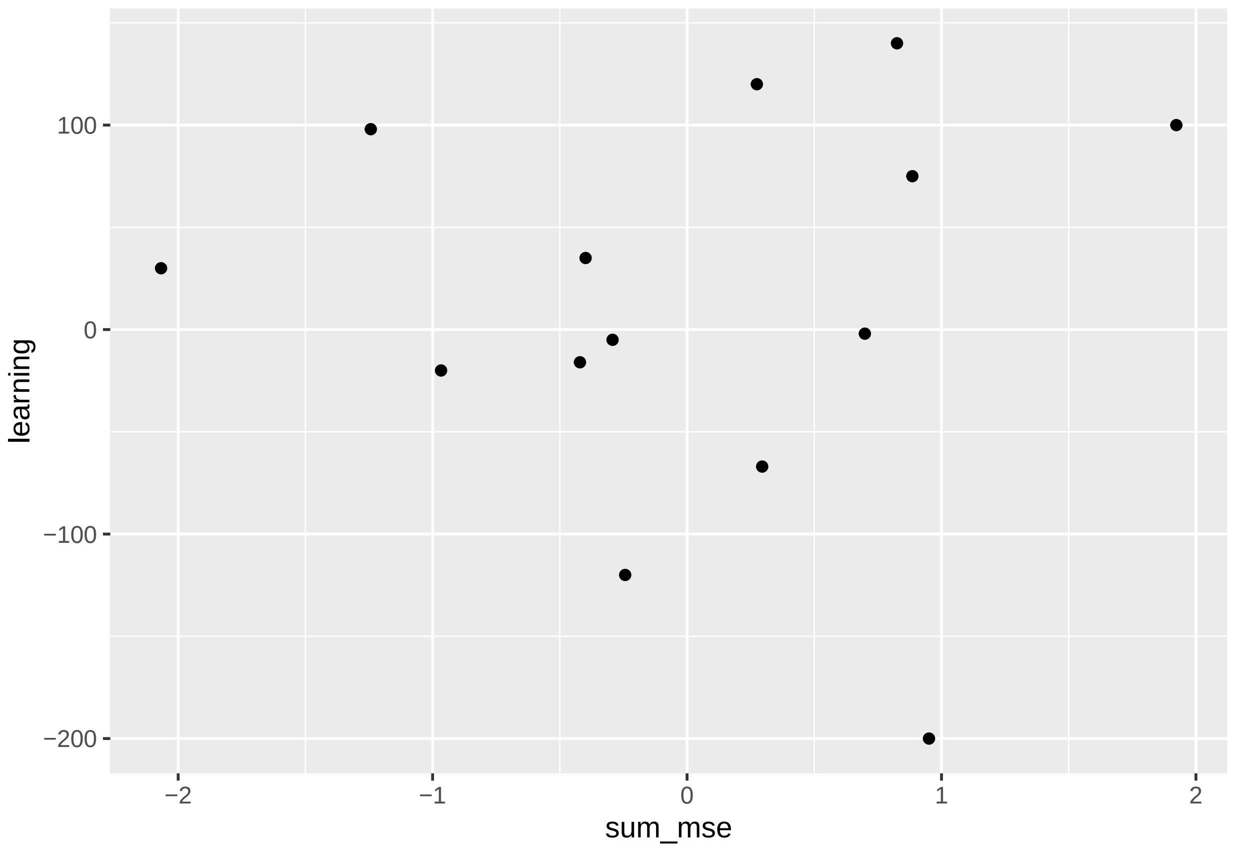


# Variability does not explain adaptation, as measured by changes in theta.

```
neutral_diff_var <- neutral_compa %>%
  ungroup() %>%
  left_join(var_pre, by='subject')

m_neutral_adap_var <- lm(learning ~ sum_mse, data=neutral_diff_var)

summary(m_neutral_adap_var)
```

```
##
## Call:
## lm(formula = learning ~ sum_mse, data = neutral_diff_var)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -215.49  -27.84    4.35   75.60  124.98
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   11.941     26.586   0.449    0.661
## sum_mse        3.734     26.632   0.140    0.891
##
## Residual standard error: 99.46 on 12 degrees of freedom
## Multiple R-squared:  0.001635,   Adjusted R-squared:  -0.08156
## F-statistic: 0.01965 on 1 and 12 DF,  p-value: 0.8908
```

```r
ggplot(data=neutral_diff_var, aes(x=sum_mse, y=learning)) +
  geom_point()
```



Also, variability does not explain adaptation, as measured by changes in the difference between high/low f0 test trial VAS responses.