Valence color association in attentional capture

Milos, Hermann, Strongway

30 July, 2023

Experiment 1

Association phase

Here is a brief summary:

```
# create a plot function for reuse purpose
myplot <- function(df, x, y, factor) {
  sx = sym(x)
  sy = sym(paste0('m',y))
  se = sym(paste0('se',y))
  sp = sym(factor)
  my_aes = aes(!!sx, !!sy, ymin = !!sy - !!se, max = !!sy + !!se,  shape = !!sp, color = !!sp)

  pd = position_dodge(width = 0.5)

  fig = ggplot(df, my_aes) +
    #geom_col_pattern(position = pd, width = 0.4, fill = 'white', colour = 'black',  pattern_density =
  #geom_bar(stat = 'identity', position = pd, width = 0.4) +
  geom_point(position = pd) +
  geom_errorbar(position = pd, width = 0.2) + theme_classic() +
  theme(legend.position = 'top')

}

# Overall mean RT and accuracy
print("Mean RT:")
```

```
## [1] "Mean RT:"
```

```
print(mean(exp1_train_m$RT))
```

```
## [1] 0.7572989
```

```
print("Mean Accuracy:")
```

```
## [1] "Mean Accuracy:"
```

```
print(mean(exp1_train_m$accuracy))
```

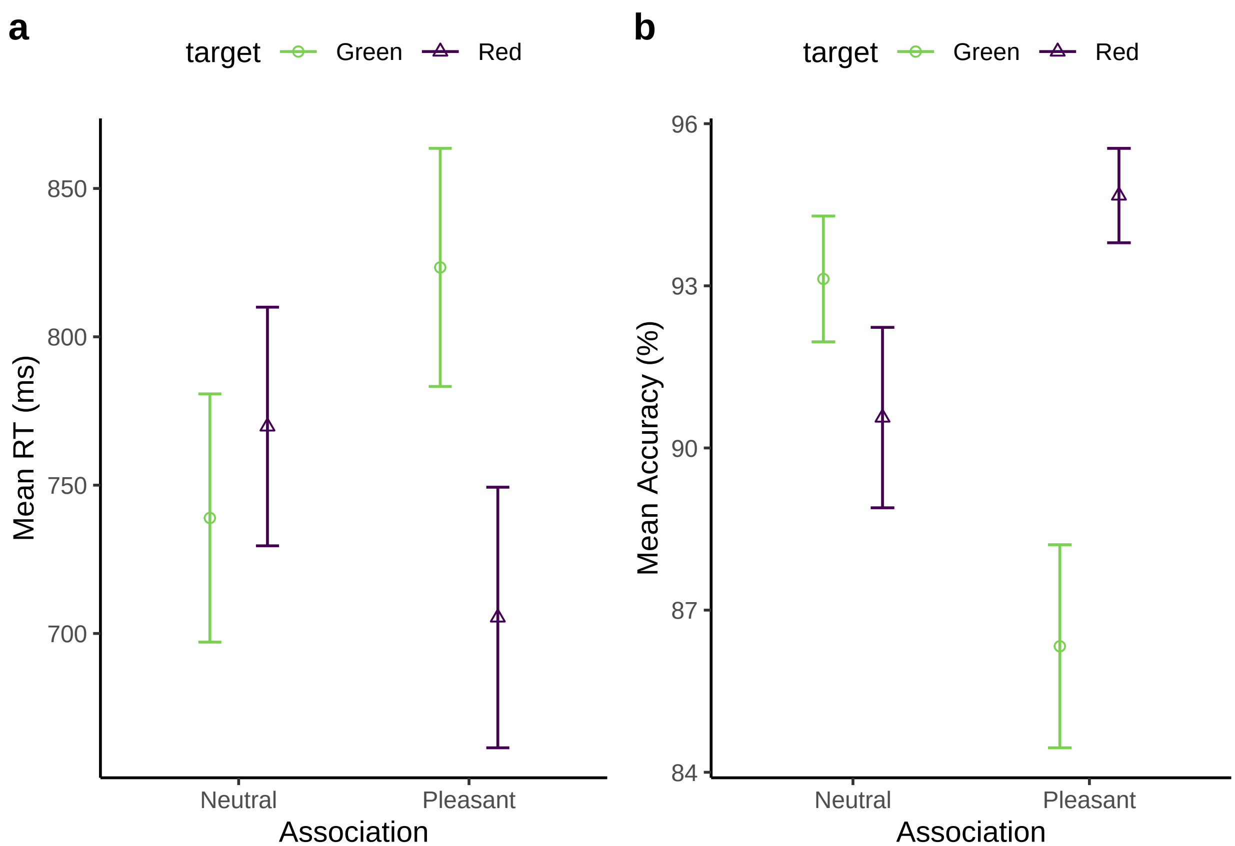
```
## [1] 0.9132414
```

```
# print mean values
exp1_train_m %>% group_by(Association, target, Group) %>%
    summarise(mRT = mean(RT), mAccuracy = mean(accuracy),n=n())
```

```
## # A tibble: 4 x 6
## # Groups:   Association, target [4]
##   Association target Group          mRT mAccuracy     n
##   <fct>       <fct>  <fct>        <dbl>     <dbl> <int>
## 1 Neutral     green  red_Pleasant 0.739     0.931    19
## 2 Neutral     red    red_Neutral  0.770     0.906    17
## 3 Pleasant    green  red_Neutral  0.823     0.863    17
## 4 Pleasant    red    red_Pleasant 0.705     0.947    19
```

Visualize the mean RTs and accuracy in the training session. We combine the two groups together.

```
exp1_mm1 = exp1_train_m %>% group_by(Association, target) %>%
  summarise(n=n(),mRT = mean(RT)*1000, seRT = sd(RT)/sqrt(n)*1000,
            mAcc = mean(accuracy)*100, seAcc = sd(accuracy)*100/sqrt(n))

# mean RT plot
# recommended by reviewers, we use viridis color scheme
fig1_1 = myplot(exp1_mm1, "Association","RT","target") + xlab('Group') + ylab('Mean RT (ms)') +
  scale_color_viridis_d( end = 0.8, labels = c("Green","Red"), direction  = -1) +
#  scale_color_manual(values = c("green","red"), labels = c("Green","Red")) +
  scale_shape_manual(values = c(1,2), labels = c("Green","Red")) +
  xlab("Association")
# mean Accuracy plot
fig1_2 = myplot(exp1_mm1, "Association","Acc","target") + xlab('Group') + ylab('Mean Accuracy (%)') +
  scale_color_viridis_d( end = 0.8, labels = c("Green","Red"), direction  = -1) +
  scale_shape_manual(values = c(1,2), labels = c("Green","Red")) +
  xlab("Association")
fig1 = plot_grid(fig1_1, fig1_2, nrow = 1, labels = c("a","b"))
ggsave(filename = './figures/fig_e1_training.png',fig1, width = 7, height = 3.5)
fig1
```



Having the accuracy and RT in the training session, we can do the ANOVA test. We use the `lmer` function from the `lmerTest` package.

```
# test the accuracy with anova with factors of association and target
#aov(accuracy ~ Association*target + Error(name), data = exp1_train_m) -> aov1
#summary(aov1)

aov1 = tidy(anova(  lmer(accuracy ~ Association*target + (1|name), data = exp1_train_m) ))
aov1
```

```
## # A tibble: 3 x 7
##   term                  sumsq  meansq NumDF DenDF statistic p.value
##   <chr>                 <dbl>   <dbl> <int> <dbl>     <dbl>   <dbl>
## 1 Association        0.00324 0.00324      1  34.0      2.23 0.145
## 2 target             0.0150  0.0150       1  34.0     10.3  0.00293
## 3 Association:target 0.0135  0.0135       1  34.0      9.31 0.00439
```

The results showed the significant main effect of target and the interaction between association and target. We then use the `emmeans` function to do the post-hoc comparison and `F_to_eta2()` to get the effect size.

```
F_to_eta2(f = aov1$statistic, df = aov1$NumDF, df_error=aov1$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.06           | [0.00, 1.00]
## 0.23           | [0.06, 1.00]
## 0.22           | [0.05, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```r
# test the accuracy with anova with factors of association and target
aov2 = tidy(anova(lmer(RT ~ Association*target + (1|name), data = exp1_train_m)))
aov2
```

```
## # A tibble: 3 x 7
##   term                 sumsq  meansq NumDF DenDF statistic    p.value
##   <chr>                <dbl>   <dbl> <int> <dbl>     <dbl>      <dbl>
## 1 Association        0.00181 0.00181     1  34.0      1.67 0.205
## 2 target             0.0341  0.0341      1  34.0     31.5  0.00000277
## 3 Association:target 0.00174 0.00174     1  34.0      1.61 0.213
```

The results showed the significant main effect of target. And its effect size:

```r
F_to_eta2(f = aov2$statistic, df = aov2$NumDF, df_error=aov2$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.05           | [0.00, 1.00]
## 0.48           | [0.27, 1.00]
## 0.05           | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

Test phase

The overall means of the test phase:

```r
exp1_test_m %>% group_by(Duration, Association, distractor) %>% summarise(mRT = mean(RT), mAccuracy = me
```

```
## # A tibble: 10 x 5
## # Groups:   Duration, Association [6]
##    Duration Association distractor   mRT mAccuracy
##    <fct>    <fct>       <chr>      <dbl>     <dbl>
##  1 Long     Neutral     green      0.661     0.935
##  2 Long     Neutral     red        0.651     0.928
##  3 Long     Pleasant    green      0.675     0.943
##  4 Long     Pleasant    red        0.644     0.940
##  5 Long     Absent      absent     0.647     0.933
##  6 Short    Neutral     green      0.630     0.874
##  7 Short    Neutral     red        0.643     0.876
##  8 Short    Pleasant    green      0.644     0.880
##  9 Short    Pleasant    red        0.634     0.880
## 10 Short    Absent      absent     0.639     0.882
```

It turns out Exposure is the main factor that affects the accuracy and mean RTs

```r
exp1_test_m %>% group_by(Duration) %>% summarise(mRT = mean(RT), mAccuracy = mean(Accuracy))
```

```
## # A tibble: 2 x 3
##   Duration   mRT mAccuracy
##   <fct>    <dbl>     <dbl>
## 1 Long     0.654     0.936
## 2 Short    0.638     0.879
```

Now test effects of exposure, association, and distractor color, and their interactions on accuracy and RTs. Note the factors “Color-Valence Association” and “Expsosure Duration” were full factorial design, and the distractor color is linked to valence association. We we assume the distractor color contribute alone to accuracy and RTs. Any interactions come from color-valence association and exposure duration.

```
exp1_test_m$RTms = exp1_test_m$RT*1000

#aov2 = aov(Accuracy ~ Association*Duration*distractor + Error(name), data = exp1_test_m)
#summary(aov2)
aov3 = tidy(anova(lmer(Accuracy ~ Association*Duration+distractor + (1|name), data = exp1_test_m)))
aov3
```

```
## # A tibble: 4 x 7
##   term                    sumsq   meansq NumDF DenDF statistic  p.value
##   <chr>                   <dbl>    <dbl> <int> <dbl>     <dbl>    <dbl>
## 1 Association          0.00199  0.000993     2  174.    0.599  5.50e- 1
## 2 Duration             0.175    0.175        1  174.  105.     1.28e-19
## 3 distractor           0.000152 0.000152     1  174.    0.0916 7.63e- 1
## 4 Association:Duration 0.000947 0.000473     2  174.    0.286  7.52e- 1
```

It shows only the main effect of exposure duration is significant. The effect sizes are:

```
F_to_eta2(f = aov3$statistic, df = aov3$NumDF, df_error=aov3$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 6.84e-03       | [0.00, 1.00]
## 0.38           | [0.29, 1.00]
## 5.26e-04       | [0.00, 1.00]
## 3.27e-03       | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

For mean RTs, again it shows only the main effect of exposure duration is significant.

```
aov4 = tidy(anova(lmer(RT ~ Association*Duration+distractor + (1|name), data = exp1_test_m)))
aov4
```

```
## # A tibble: 4 x 7
##   term                    sumsq   meansq NumDF DenDF statistic p.value
##   <chr>                   <dbl>    <dbl> <int> <dbl>     <dbl>   <dbl>
## 1 Association          0.000255 0.000128     2  174.    0.0730 0.930
## 2 Duration             0.0138   0.0138       1  174.    7.87   0.00559
## 3 distractor           0.00329  0.00329      1  174.    1.88   0.172
## 4 Association:Duration 0.00164  0.000818     2  174.    0.468  0.627
```

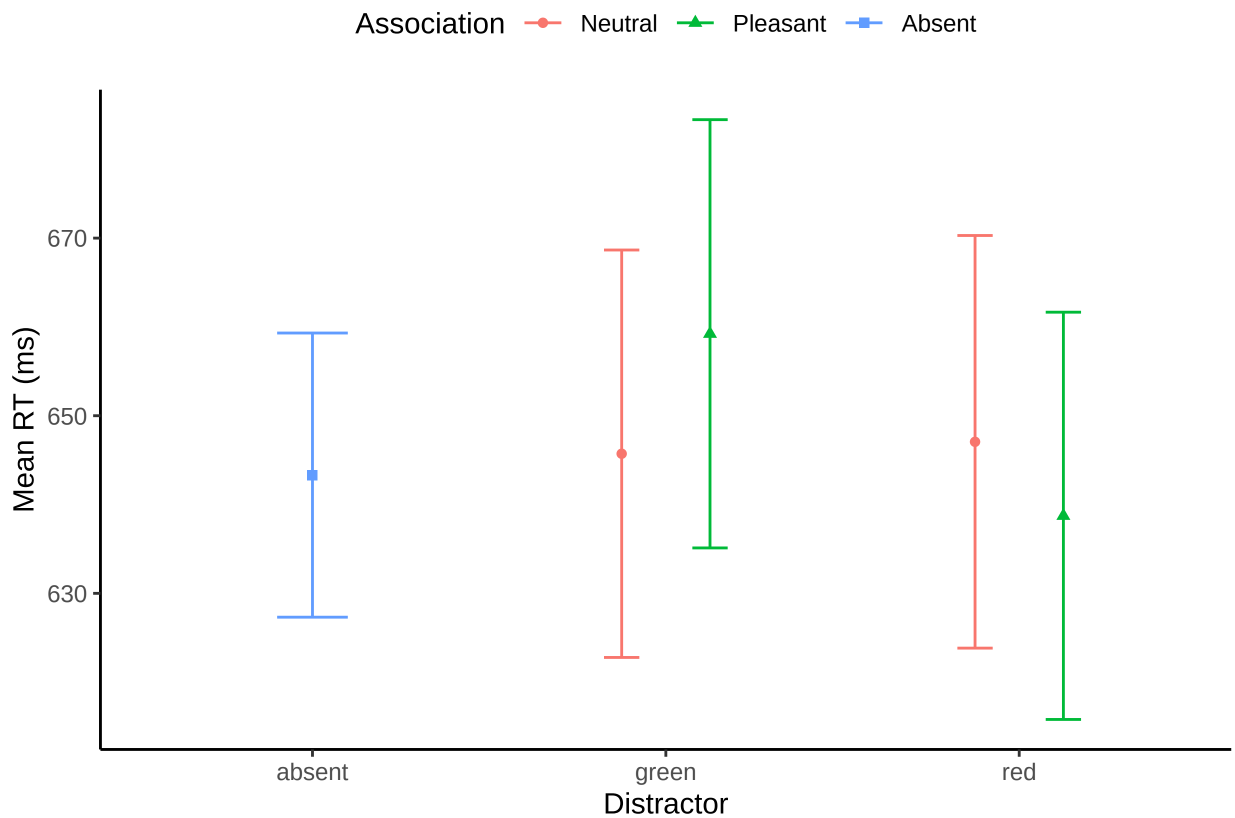
And its effect sizes:

```
F_to_eta2(f = aov4$statistic, df = aov4$NumDF, df_error=aov4$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 8.38e-04       | [0.00, 1.00]
## 0.04           | [0.01, 1.00]
## 0.01           | [0.00, 1.00]
## 5.35e-03       | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

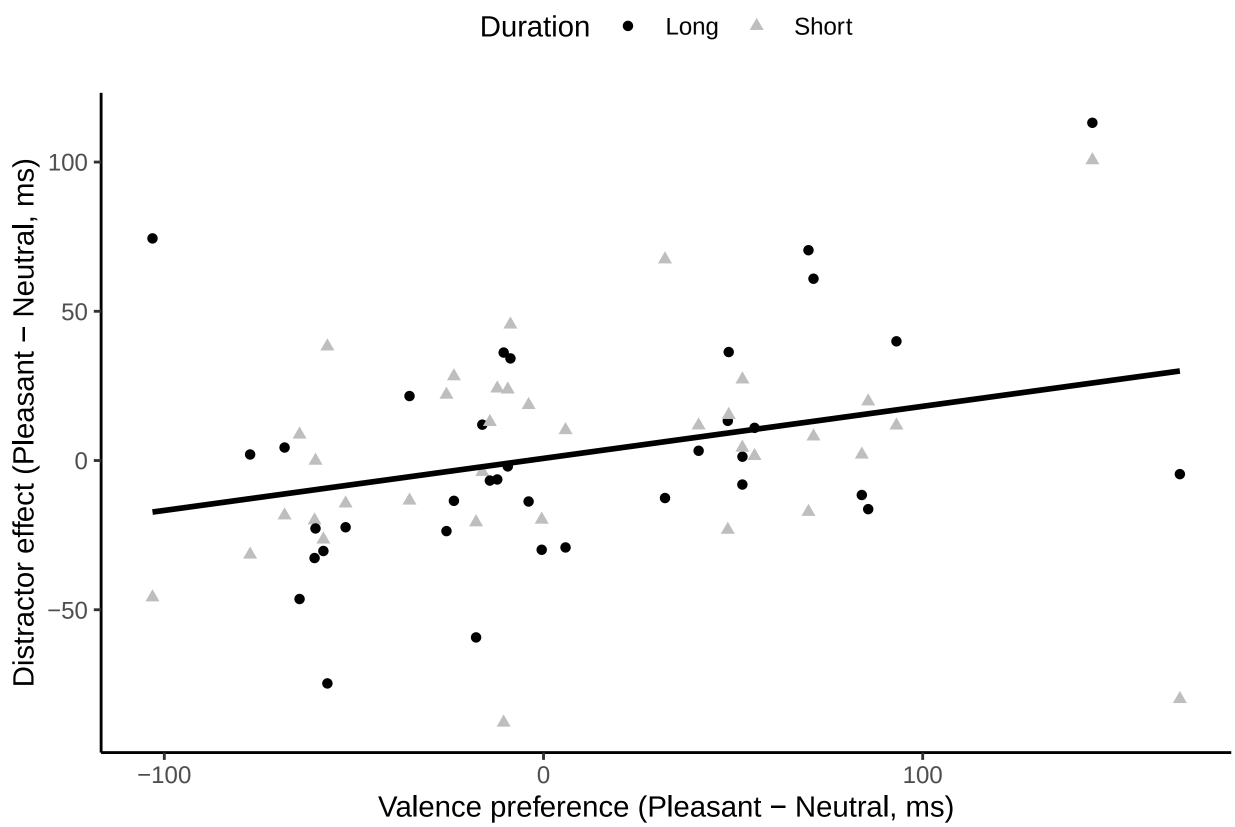
Visualize the mean RT and association-test correlation.

```r
exp1_test_mm = exp1_test_m %>% group_by(Association, distractor) %>%
  summarise(mRT = mean(RTms), seRT = sd(RTms)/sqrt(n()))

fig2_1 = myplot(exp1_test_mm, "distractor","RT","Association") + xlab('Distractor') + ylab('Mean RT (ms
fig2_1
```




```r
# correlation

exp1_train_m %>% select(name, target, RT) %>% # calculate preference of color
  pivot_wider(names_from = target, values_from = RT) %>%
  mutate(colorDiff = sign(red - green), colorRG = (red - green)*1000) %>%
  mutate(Preference = factor(colorDiff, label = c("Red","Green"))) -> exp1_color_preference

exp1_train_m %>% select(name, Group, Association, RT) %>%
  pivot_wider(names_from = Association, values_from = RT) %>%
  mutate(Learning = (Pleasant - Neutral)*1000) %>% select (-Neutral, -Pleasant) -> exp1_learning
# for emotion difference
exp1_test_m %>% filter(Association != 'Absent') %>% group_by(name, Group, Association, Duration) %>%
  summarise(RT = mean(RT)) %>% pivot_wider(names_from = Association, values_from = RT) %>%
  mutate(Interference = (Pleasant - Neutral)*1000) %>% select(-Neutral, -Pleasant) -> exp1_interference
# for color difference
exp1_test_m %>% filter(Association != 'Absent') %>% group_by(name, Group, distractor) %>%
  summarise(RT = mean(RT)) %>% pivot_wider(names_from = distractor, values_from = RT) %>%
  mutate(distractorRG = red - green) %>% select(-red, -green) -> exp1_test_distractorRG
exp1_correlation = left_join(exp1_learning, exp1_interference, by = c('name', 'Group')) %>%
  left_join(., exp1_color_preference, by = c('name')) %>%
  left_join(., exp1_test_distractorRG, by = c('name','Group'))
```

```r
# fig of correlation
fig2_2 = ggplot(exp1_correlation, aes(Learning, Interference)) +
  geom_point(aes(color = Duration, shape = Duration)) +
  scale_color_manual(values = c("black","grey")) +
  geom_smooth(method = 'lm', se = F, color = 'black') + theme_classic() +
  xlab('Valence preference (Pleasant - Neutral, ms)') +
  ylab('Distractor effect (Pleasant - Neutral, ms)') + #facet_wrap(~Group, ncol = 1) +
  theme(legend.position = 'top')
# save figure
ggsave(filename = './figures/fig_e1_corr.png',fig2_2, width = 3.5, height = 3.5)

fig2_2
```



Now we analyze the correlation between the association effect and the distractor effect.

```r
mod_cor1 = lm(Interference ~ Learning, data = exp1_correlation)
print(summary(mod_cor1))
```

```
##
## Call:
## lm(formula = Interference ~ Learning, data = exp1_correlation)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -109.898  -18.764   -5.017   18.736   91.692
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  0.72466    4.15682   0.174  0.86211
## Learning     0.17445    0.06575   2.653  0.00986 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35.01 on 70 degrees of freedom
## Multiple R-squared:  0.09138,    Adjusted R-squared:  0.07839
## F-statistic: 7.039 on 1 and 70 DF,  p-value: 0.009859
```

```
print(cor_test(data = exp1_correlation, vars = Learning, vars2 = Interference ))
```

```
## # A tibble: 1 x 8
##   var1     var2           cor statistic       p conf.low conf.high method
##   <chr>    <chr>        <dbl>     <dbl>   <dbl>    <dbl>     <dbl> <chr>
## 1 Learning Interference   0.3      2.65 0.00986   0.0759     0.499 Pearson
```

Experiment 2

Training session

Overall summary

```
exp2_train_m %>% group_by(Group, Arousal, Valence) %>% summarise(mRT = mean(RT), mAccuracy = mean(Accura
```

```
## # A tibble: 8 x 6
## # Groups:   Group, Arousal [4]
##   Group              Arousal Valence    mRT mAccuracy     n
##   <chr>              <chr>   <fct>    <dbl>     <dbl> <int>
## 1 high_Pleasant_green high    Neutral  0.628     0.943    20
## 2 high_Pleasant_green high    Pleasant 0.652     0.935    20
## 3 high_Pleasant_green low     Neutral  0.702     0.937    20
## 4 high_Pleasant_green low     Pleasant 0.666     0.948    20
## 5 high_Pleasant_red   high    Neutral  0.706     0.903    18
## 6 high_Pleasant_red   high    Pleasant 0.663     0.938    18
## 7 high_Pleasant_red   low     Neutral  0.678     0.939    18
## 8 high_Pleasant_red   low     Pleasant 0.735     0.909    18
```

```
# Overall mean RT and accuracy
print("Mean RT:")
```

```
## [1] "Mean RT:"
```

```
print(mean(exp2_train_m$RT))
```

```
## [1] 0.6778527
```

```
print("Mean Accuracy:")
```

```
## [1] "Mean Accuracy:"
```

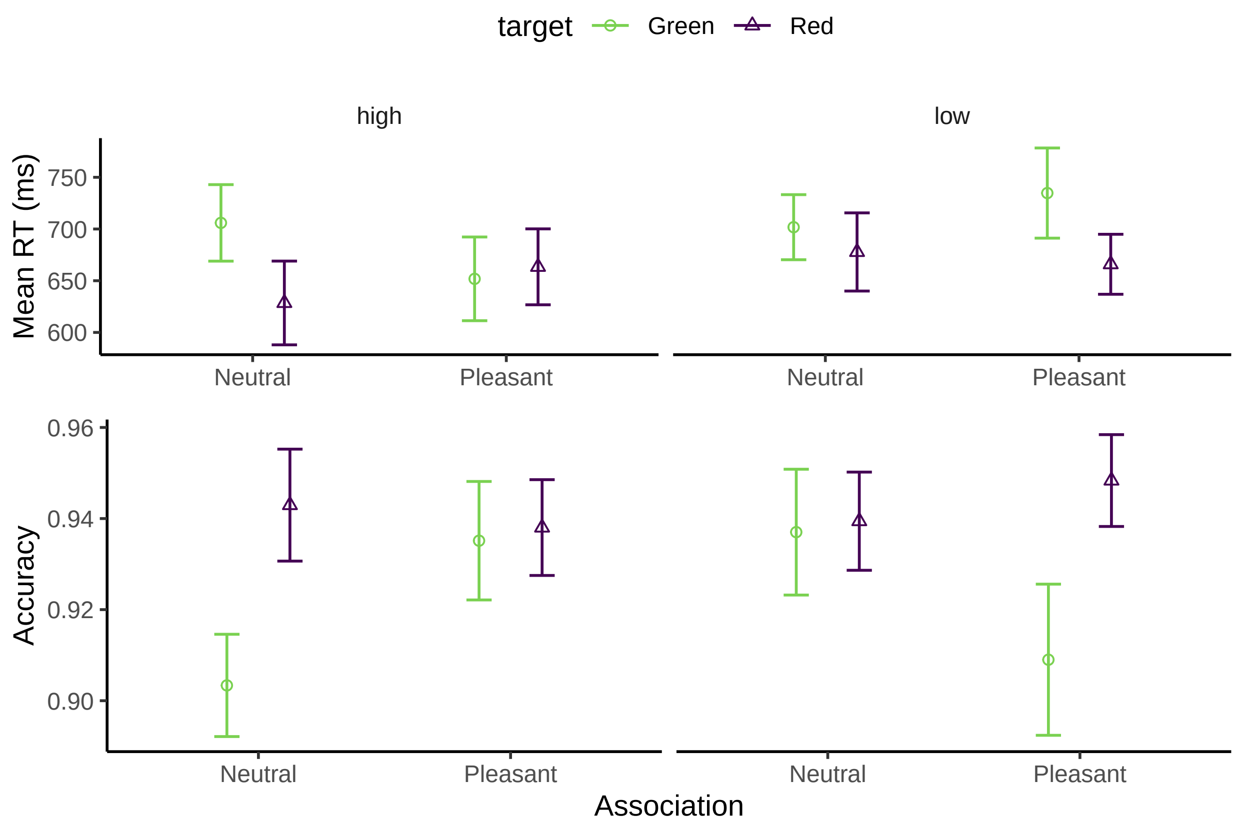
```
print(mean(exp2_train_m$Accuracy))
```

```
## [1] 0.9321403
```

Figure for training session

```
exp2_mm1 = exp2_train_m %>% group_by(Arousal, Valence, target) %>%
  summarise(n=n(),mRT = mean(RT)*1000,  seRT = sd(RT)/sqrt(n)*1000,
```

```
            mAccuracy = mean(Accuracy), seAccuracy = sd(Accuracy)/sqrt(n))

fig4_1 = myplot(exp2_mm1, "Valence","RT","target") + facet_wrap(~Arousal) +
   ylab('Mean RT (ms)') +
  scale_color_viridis_d( end = 0.8, labels = c("Green","Red"), direction  = -1) +
  scale_shape_manual(values = c(1,2), labels = c("Green","Red")) +
  theme(axis.title.x = element_blank(),
                              panel.border = element_blank(), panel.grid.major = element_blank(),
               panel.grid.minor = element_blank(), strip.background = element_blank())
fig4_2 =  myplot(exp2_mm1, "Valence","Accuracy","target") + facet_wrap(~Arousal) +
  xlab('Association') + ylab('Accuracy') +
  scale_color_viridis_d( end = 0.8, labels = c("Green","Red"), direction  = -1) +
  scale_shape_manual(values = c(1,2), labels = c("Green","Red")) +
  theme(legend.position = 'none',
                                                strip.text = element_blank())
fig4 = plot_grid(fig4_1, fig4_2, nrow = 2 )
# save figure
ggsave(filename = './figures/fig_e2_training.png',fig4, width = 7, height = 5)
fig4
```



Linear mixed model for the training accuracy and RTs

```
aov5 = tidy(anova(lmer(Accuracy ~ Valence*Arousal*target + (1|name), data = exp2_train_m)))
aov5
```

```
## # A tibble: 7 x 7
##   term                         sumsq      meansq NumDF DenDF  statistic p.value
##   <chr>                        <dbl>       <dbl> <int> <dbl>      <dbl>   <dbl>
## 1 Valence                   0.000142   0.000142      1  108.     0.0967   0.756
```

```
## 2 Arousal                0.000485    0.000485        1 108.   0.330     0.567
## 3 target                 0.0168      0.0168          1 108.  11.4       0.00100
## 4 Valence:Arousal        0.00500     0.00500         1 108.   3.40      0.0678
## 5 Valence:target         0.000000122 0.000000122     1 108.   0.0000828 0.993
## 6 Arousal:target         0.00000126  0.00000126      1 108.   0.000860  0.977
## 7 Valence:Arousal:target 0.00255     0.00255         1  36.0  1.73      0.196
```

It shows the main effect of target, marginal effect of Valence x Arousal, but not others. Here are the effect sizes:

```
F_to_eta2(f = aov5$statistic, df = aov5$NumDF, df_error=aov5$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 8.95e-04       | [0.00, 1.00]
## 3.05e-03       | [0.00, 1.00]
## 0.10           | [0.03, 1.00]
## 0.03           | [0.00, 1.00]
## 7.67e-07       | [0.00, 1.00]
## 7.96e-06       | [0.00, 1.00]
## 0.05           | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```
mod6 = lmer(RT ~ Valence*Arousal*target + (1|name), data = exp2_train_m)
aov6 = tidy(anova(mod6))
aov6
```

```
## # A tibble: 7 x 7
##   term                        sumsq     meansq NumDF DenDF statistic p.value
##   <chr>                       <dbl>      <dbl> <int> <dbl>     <dbl>   <dbl>
## 1 Valence                0.00000883 0.00000883     1 108.    0.00115 0.973
## 2 Arousal                0.0404     0.0404         1 108.    5.27    0.0236
## 3 target                 0.0596     0.0596         1 108.    7.78    0.00625
## 4 Valence:Arousal        0.00384    0.00384        1 108.    0.501   0.480
## 5 Valence:target         0.00462    0.00462        1 108.    0.603   0.439
## 6 Arousal:target         0.00172    0.00172        1 108.    0.225   0.636
## 7 Valence:Arousal:target 0.00396    0.00396        1  36.0   0.516   0.477
```

It shows the main effect of Arousal, target, but not Valence. Here are the effect sizes:

```
F_to_eta2(f = aov6$statistic, df = aov6$NumDF, df_error=aov6$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 1.07e-05       | [0.00, 1.00]
## 0.05           | [0.00, 1.00]
## 0.07           | [0.01, 1.00]
## 4.62e-03       | [0.00, 1.00]
## 5.55e-03       | [0.00, 1.00]
## 2.08e-03       | [0.00, 1.00]
## 0.01           | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

compare the meanRTs of high vs. low arousal, and target color (red vs. green)

```r
emmeans(mod6, pairwise ~ Arousal, adjust = 'BY')
```

```
## $emmeans
##  Arousal emmean     SE   df lower.CL upper.CL
##  high     0.662 0.0244 42.9    0.613    0.712
##  low      0.695 0.0244 42.9    0.646    0.744
##
## Results are averaged over the levels of: Valence, target
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast   estimate     SE  df t.ratio p.value
##  high - low  -0.0327 0.0142 108  -2.296  0.0236
##
## Results are averaged over the levels of: Valence, target
## Degrees-of-freedom method: kenward-roger
```

```r
emmeans(mod6, pairwise ~ target, adjust = 'BY')
```

```
## $emmeans
##  target emmean     SE   df lower.CL upper.CL
##  green   0.699 0.0244 42.9    0.649    0.748
##  red     0.659 0.0244 42.9    0.610    0.708
##
## Results are averaged over the levels of: Valence, Arousal
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast    estimate     SE  df t.ratio p.value
##  green - red   0.0397 0.0142 108   2.789  0.0063
##
## Results are averaged over the levels of: Valence, Arousal
## Degrees-of-freedom method: kenward-roger
```

Test session

A general mean RT and accuracy in the test session:

```r
# Overall mean RT and accuracy
print("Mean RT:")
```

```
## [1] "Mean RT:"
```

```r
print(mean(exp2_test_m$RT))
```

```
## [1] 0.5715276
```

```r
print("Mean Accuracy:")
```

```
## [1] "Mean Accuracy:"
```

```r
print(mean(exp2_test_m$accuracy))
```

```
## [1] 0.9346595
```

First, let's check effects in the training session for accuracy and RTs.

```
# accuracy
mod7 = lmer(accuracy ~ Valence*Arousal*Duration + distractor + (1|name), data = exp2_test_m)
aov7 = tidy(anova(mod7))
aov7
```

```
## # A tibble: 8 x 7
##   term                        sumsq   meansq NumDF DenDF statistic  p.value
##   <chr>                       <dbl>    <dbl> <int> <dbl>     <dbl>    <dbl>
## 1 Valence                  0.00163  0.000817     2  406.     0.528 5.90e- 1
## 2 Arousal                  0.00558  0.00558      1  406.     3.61  5.83e- 2
## 3 Duration                 0.232    0.232        1  406.   150.    1.65e-29
## 4 distractor               0.00126  0.00126      1  406.     0.811 3.68e- 1
## 5 Valence:Arousal          0.000318 0.000159     2  406.     0.103 9.02e- 1
## 6 Valence:Duration         0.00580  0.00290      2  406.     1.87  1.55e- 1
## 7 Arousal:Duration         0.000826 0.000826     1  406.     0.533 4.66e- 1
## 8 Valence:Arousal:Duration 0.000782 0.000391     2  406.     0.252 7.77e- 1
```

The main effect of Arousal, Exposure Duration, and Distractor Color were significant, but not Valence! Here are the effect sizes:

```
F_to_eta2(f = aov7$statistic, df = aov7$NumDF, df_error=aov7$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 2.59e-03       | [0.00, 1.00]
## 8.80e-03       | [0.00, 1.00]
## 0.27           | [0.21, 1.00]
## 1.99e-03       | [0.00, 1.00]
## 5.06e-04       | [0.00, 1.00]
## 9.14e-03       | [0.00, 1.00]
## 1.31e-03       | [0.00, 1.00]
## 1.24e-03       | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```
exp2_test_m %>% group_by(Duration) %>% summarise(mAcc = mean(accuracy))
```

```
## # A tibble: 2 x 2
##   Duration  mAcc
##   <chr>    <dbl>
## 1 Long     0.957
## 2 Short    0.912
```

```
exp2_test_m %>% group_by(Arousal) %>% summarise(mAcc = mean(accuracy))
```

```
## # A tibble: 2 x 2
##   Arousal  mAcc
##   <chr>   <dbl>
## 1 high    0.938
## 2 low     0.931
```

Linear mixed model and ANOVA for RTs

```
mod8 = lmer(RT ~ Valence*Arousal*Duration + distractor + (1|name), data = exp2_test_m)
aov8 = tidy(anova(mod8))
aov8
```

```
## # A tibble: 8 x 7
##   term                         sumsq    meansq NumDF DenDF statistic    p.value
##   <chr>                        <dbl>     <dbl> <int> <dbl>     <dbl>      <dbl>
## 1 Valence                   0.000425 0.000213      2  406.    0.0521 0.949
## 2 Arousal                   0.0963   0.0963        1  406.   23.6     0.00000169
## 3 Duration                  0.0529   0.0529        1  406.   13.0     0.000354
## 4 distractor                0.00366  0.00366       1  406.    0.898   0.344
## 5 Valence:Arousal           0.000228 0.000114      2  406.    0.0279 0.972
## 6 Valence:Duration          0.00114  0.000568      2  406.    0.139   0.870
## 7 Arousal:Duration          0.00163  0.00163       1  406.    0.399   0.528
## 8 Valence:Arousal:Duration 0.000124 0.0000620     2  406.    0.0152 0.985
```

The main effect of Arousal, Exposure Duration, were significant, but not Valence and distractor color! Here are the effect sizes:

```
F_to_eta2(f = aov8$statistic, df = aov8$NumDF, df_error=aov8$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 2.57e-04       | [0.00, 1.00]
## 0.05           | [0.02, 1.00]
## 0.03           | [0.01, 1.00]
## 2.21e-03       | [0.00, 1.00]
## 1.37e-04       | [0.00, 1.00]
## 6.86e-04       | [0.00, 1.00]
## 9.82e-04       | [0.00, 1.00]
## 7.50e-05       | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```
exp2_test_m %>% group_by(Arousal) %>% summarise(mRT = mean(RT))
```

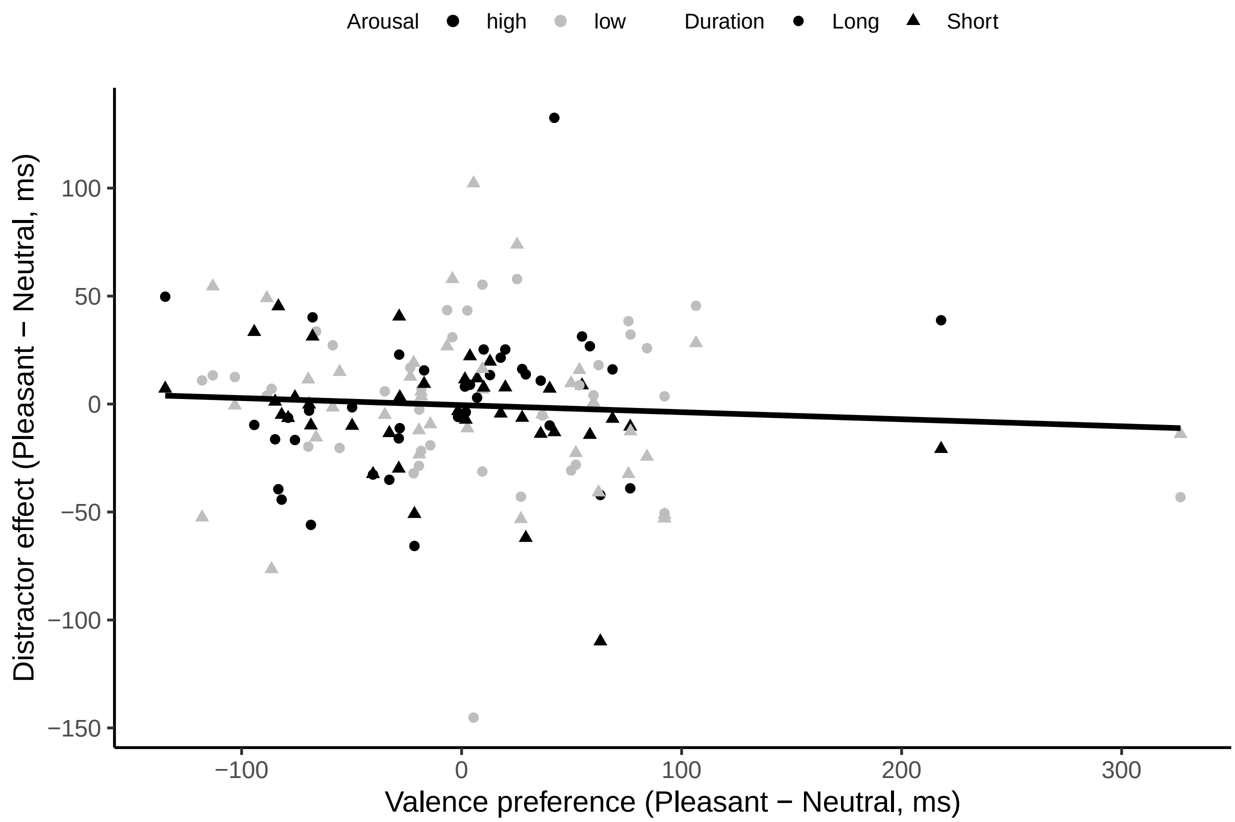
```
## # A tibble: 2 x 2
##   Arousal   mRT
##   <chr>   <dbl>
## 1 high    0.557
## 2 low     0.586
```

```
exp2_test_m %>% group_by(Duration) %>% summarise(mRT = mean(RT))
```

```
## # A tibble: 2 x 2
##   Duration   mRT
##   <chr>    <dbl>
## 1 Long     0.582
## 2 Short    0.561
```

```
# build correlation data set
ungroup(exp2_train_m) %>% group_by(name,Arousal, Valence) %>% summarize(RT = mean(RT)) %>%
  pivot_wider(names_from = Valence, values_from = RT) %>%
  mutate(Learning = (Pleasant - Neutral)*1000) %>% select (-Neutral, -Pleasant) -> exp2_learning

ungroup(exp2_test_m) %>% filter(Valence != 'Absent') %>% group_by(name,Arousal,Duration, Valence) %>%
  summarise(RT = mean(RT)) -> exp2_inter
# association-based interference
exp2_inter %>% pivot_wider(names_from = Valence, values_from = RT) %>%
  mutate(Interference = (Pleasant - Neutral)*1000) %>% select(-Neutral, -Pleasant) -> exp2_interference
# color-based interference
```

```
#exp2_inter %>% pivot_wider(names_from = Valence, values_from = RT) -> exp2_inter_color

exp2_correlation = left_join(exp2_learning, exp2_interference, by = c('name','Arousal'))


# correlation figure
fig5_1 = ggplot(exp2_correlation, aes(Learning, Interference)) +
  geom_point(aes( color = Arousal, shape = Duration)) +
  geom_smooth(method = 'lm', se = F, color = 'black') + theme_classic() +
  scale_color_manual(values = c("black","grey")) +
  xlab('Valence preference (Pleasant - Neutral, ms)') +
  ylab('Distractor effect (Pleasant - Neutral, ms)') +
  theme(legend.position = 'top', legend.text = element_text(size=8),
        legend.title = element_text(size = 8))

# reduce font size of the legend
# save figure
#ggsave(filename = './figures/fig_e2_corr.png',fig5_1, width = 3.5, height = 3.5)
fig5_1
```




```
cor_test(data = ungroup(exp2_correlation), vars = Learning, vars2 = Interference )
```

```
## # A tibble: 1 x 8
##   var1     var2            cor statistic     p conf.low conf.high method
##   <chr>    <chr>         <dbl>     <dbl> <dbl>    <dbl>     <dbl> <chr>
## 1 Learning Interference -0.069    -0.850 0.397   -0.226    0.0910 Pearson
```

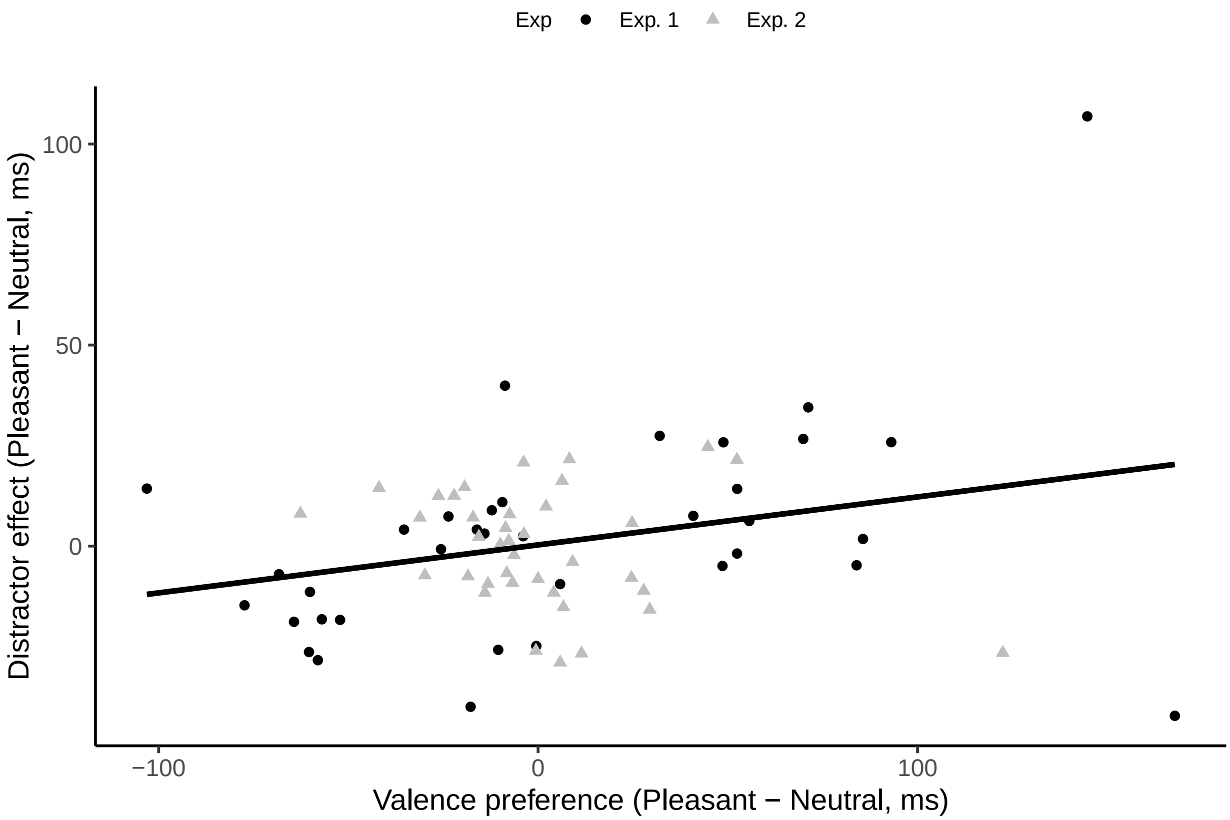
```r
comb_corr = rbind(exp1_correlation %>% group_by(name) %>%
                    summarise(Learning = mean(Learning), Interference = mean(Interference)) %>%
                    mutate(Exp = 'Exp. 1'),
                      exp2_correlation %>% ungroup() %>%group_by(name) %>%
                    summarise(Learning = mean(Learning), Interference = mean(Interference)) %>%
                    mutate(Exp = 'Exp. 2'))
# combine experiment 1 and 2 for correlation analysis
cor_test(data = comb_corr,
         vars = Learning, vars2 = Interference )
```

```
## # A tibble: 1 x 8
##   var1     var2           cor statistic      p conf.low conf.high method
##   <chr>    <chr>        <dbl>     <dbl>  <dbl>    <dbl>     <dbl> <chr>
## 1 Learning Interference  0.28      2.43 0.0175   0.0502     0.474 Pearson
```

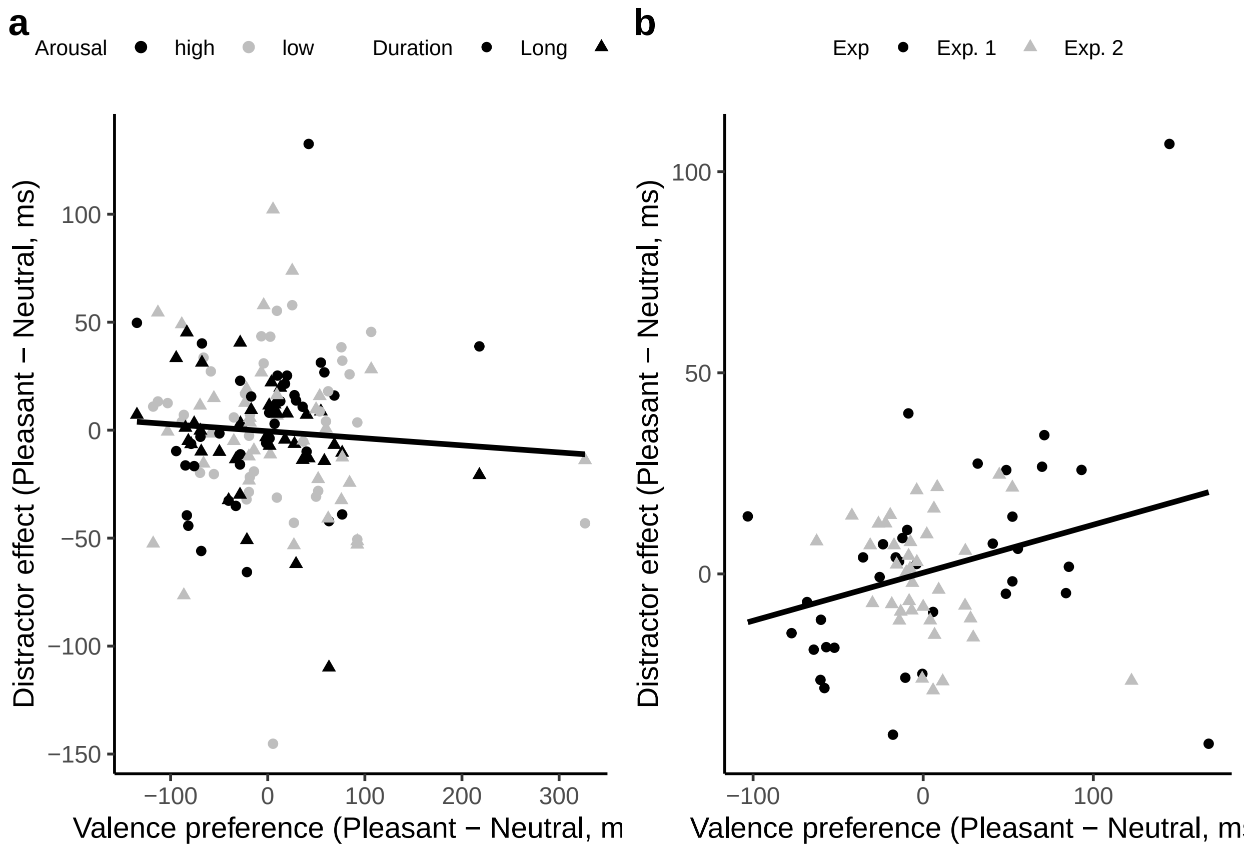
```r
anova(lm(Interference ~ Learning, data = comb_corr))
```

```
## Analysis of Variance Table
##
## Response: Interference
##           Df Sum Sq Mean Sq F value  Pr(>F)
## Learning   1   2517 2516.96  5.9169 0.01748 *
## Residuals 72  30628  425.38
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
fig5_2 = ggplot(comb_corr, aes(Learning, Interference)) +
  geom_point(aes(shape = Exp, color = Exp)) +
  geom_smooth(method = 'lm', se = F, color = 'black') + theme_classic() +
  scale_color_manual(values = c("black","grey")) +
  xlab('Valence preference (Pleasant - Neutral, ms)') +
  ylab('Distractor effect (Pleasant - Neutral, ms)') +
  theme(legend.position = 'top',, legend.text = element_text(size=8),
        legend.title = element_text(size = 8))
fig5_2
```




```r
fig5 = plot_grid(fig5_1,fig5_2, labels = c("a","b"))
# save fig5
ggsave(filename = './figures/fig5_corr.png',fig5, width = 9, height = 4.5)
fig5
```



Experiment 3

Association phase

Here is a brief summary:

```
# print mean values
exp3_train_m %>% group_by(target, valence) %>% summarise(mRT = mean(RT), mAccuracy = mean(Accuracy),n=n
```

```
## # A tibble: 4 x 5
## # Groups:   target [2]
##   target valence    mRT mAccuracy     n
##   <chr>  <chr>    <dbl>     <dbl> <int>
## 1 green  neutral  0.895     0.865    12
## 2 green  pleasant 0.777     0.913    13
## 3 red    neutral  0.732     0.935    13
## 4 red    pleasant 0.848     0.907    12
```

```
exp3_train_m %>% group_by(valence) %>% summarise(mRT = mean(RT), mAccuracy = mean(Accuracy),n=n())
```

```
## # A tibble: 2 x 4
##   valence    mRT mAccuracy     n
##   <chr>    <dbl>     <dbl> <int>
## 1 neutral  0.810     0.901    25
## 2 pleasant 0.811     0.910    25
```

```r
# Overall mean RT and accuracy
print("Mean RT:")
```

```
## [1] "Mean RT:"
```

```r
print(mean(exp3_train_m$RT))
```

```
## [1] 0.8105898
```

```r
print("Mean Accuracy:")
```

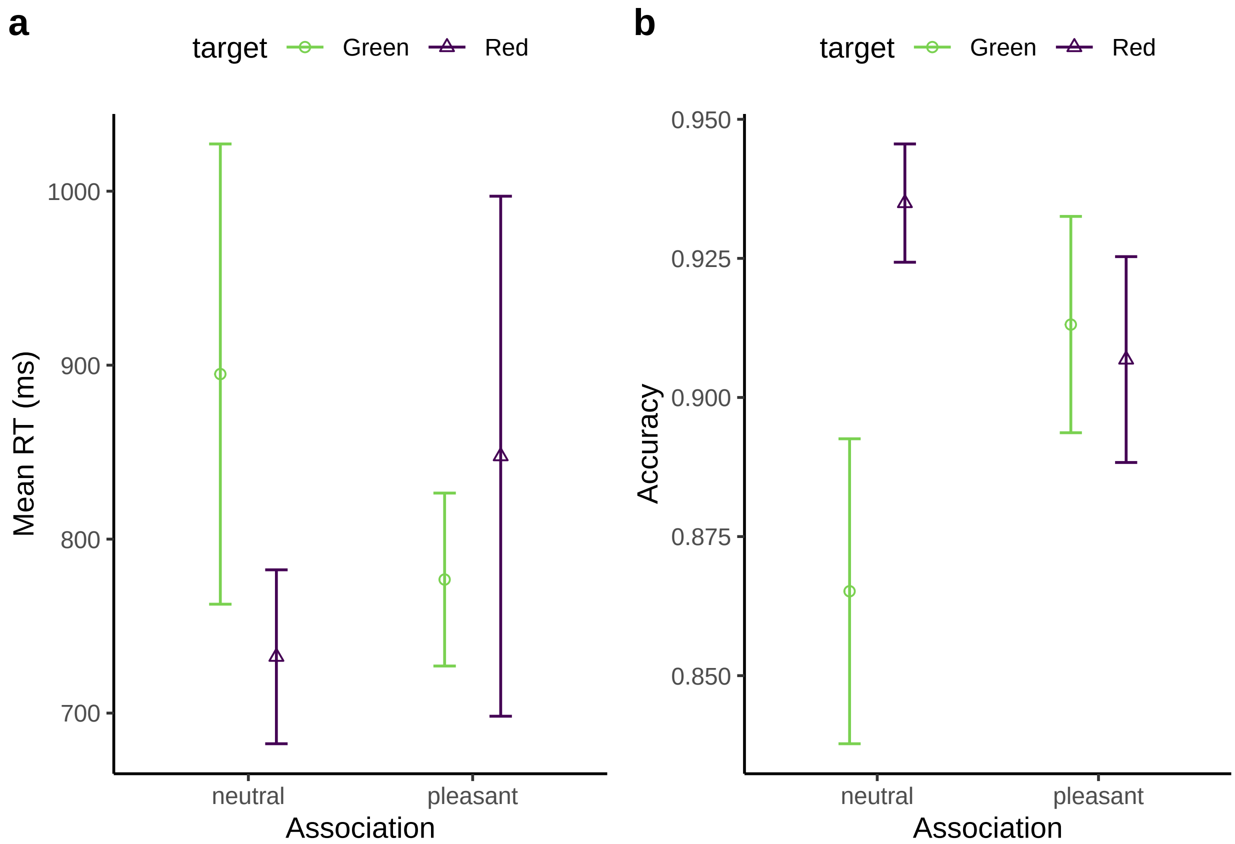
```
## [1] "Mean Accuracy:"
```

```r
print(mean(exp3_train_m$Accuracy))
```

```
## [1] 0.9057661
```

Plot the mean RTs and relative mean RTs in the training session.

```r
exp3_mm1 = exp3_train_m %>% group_by(valence, target) %>%
  summarise(n=n(),mRT = mean(RT)*1000,  seRT = sd(RT)/sqrt(n)*1000,
            mAccuracy = mean(Accuracy), seAccuracy = sd(Accuracy)/sqrt(n))

fig6_1 = myplot(exp3_mm1, "valence","RT","target") +
  scale_color_viridis_d( end = 0.8, labels = c("Green","Red"), direction  = -1) +
  scale_shape_manual(values = c(1,2), labels = c("Green","Red")) +
  xlab('Association') + ylab('Mean RT (ms)')
fig6_2 = myplot(exp3_mm1, "valence","Accuracy","target") +
  scale_color_viridis_d( end = 0.8, labels = c("Green","Red"), direction  = -1) +
  scale_shape_manual(values = c(1,2), labels = c("Green","Red")) +
  xlab('Association') + ylab('Accuracy')
fig6 = plot_grid(fig6_1, fig6_2, nrow = 1, labels = c("a","b"))
#save fig
ggsave(filename = './figures/fig_e3_training.png',fig6, width = 7, height = 3.5)
fig6
```



ANOVA tests:

```
mod9 = lmer(Accuracy ~ target*valence + (1 | participants), data=exp3_train_m)
aov9 = tidy(anova(mod9))
aov9
```

```
## # A tibble: 3 x 7
##   term              sumsq  meansq NumDF DenDF statistic p.value
##   <chr>             <dbl>   <dbl> <int> <dbl>     <dbl>   <dbl>
## 1 target          0.0126  0.0126      1  23.0     6.36   0.0191
## 2 valence         0.00122 0.00122     1  23.0     0.619  0.439
## 3 target:valence 0.00468 0.00468     1  23.0     2.37   0.138
```

it's effect sizes:

```
F_to_eta2(f = aov9$statistic, df = aov9$NumDF, df_error=aov9$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.22           | [0.02, 1.00]
## 0.03           | [0.00, 1.00]
## 0.09           | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

```
exp3_train_m %>% group_by(target) %>% summarise(mAcc = mean(Accuracy))
```

```
## # A tibble: 2 x 2
##   target  mAcc
##   <chr>  <dbl>
```

```
## 1 green  0.890
## 2 red    0.921
```

Statistical tests for RTs

```
mod10 = lmer(RT ~ target*valence + (1 | participants), data=exp3_train_m)
aov10 = tidy(anova(mod10))
aov10
```

```
## # A tibble: 3 x 7
##   term                sumsq    meansq NumDF DenDF statistic p.value
##   <chr>               <dbl>     <dbl> <int> <dbl>     <dbl>   <dbl>
## 1 target         0.0262    0.0262        1  23.0   6.19      0.0205
## 2 valence        0.0000246 0.0000246     1  23.0   0.00581   0.940
## 3 target:valence 0.00278   0.00278       1  23.0   0.659     0.425
```

and the effect sizes:

```
F_to_eta2(f = aov10$statistic, df = aov10$NumDF, df_error=aov10$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.21           | [0.02, 1.00]
## 2.53e-04       | [0.00, 1.00]
## 0.03           | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

Test phase

The overall means of the test phase. Get a mean table:

```
exp3_test_m %>% group_by( target, exposure, target_status) %>% summarise(mRT = mean(RT), mAccuracy = mea
  pivot_wider(names_from = target_status, values_from = c('mRT','mAccuracy'))
```

```
## # A tibble: 6 x 8
## # Groups:   target, exposure [6]
##   target exposure mRT_none mRT_neutral mRT_pleasant mAccuracy_none
##   <chr>  <chr>       <dbl>       <dbl>        <dbl>          <dbl>
## 1 absent long        0.808      NA           NA              0.937
## 2 absent short       0.842      NA           NA              0.876
## 3 green  long       NA           0.816        0.685         NA
## 4 green  short      NA           0.779        0.747         NA
## 5 red    long       NA           0.651        0.779         NA
## 6 red    short      NA           0.700        0.781         NA
## # ... with 2 more variables: mAccuracy_neutral <dbl>, mAccuracy_pleasant <dbl>
```

```
print(mean(exp3_test_m$Accuracy))
```

```
## [1] 0.9258052
```

Statistical tests on accuracy and RTs in the test phase.

```
mod11 = lmer(Accuracy ~ target_status*exposure + target + (1|participants), data = exp3_test_m)
aov11 = tidy(anova(mod11))
aov11
```

```
## # A tibble: 4 x 7
##   term                      sumsq   meansq NumDF DenDF statistic    p.value
```

```
##   <chr>                      <dbl>    <dbl> <int> <dbl>     <dbl>      <dbl>
## 1 target_status           0.0241   0.0120      2  119.      6.34  0.00243
## 2 exposure                0.0457   0.0457      1  119.     24.1   0.00000294
## 3 target                  0.000443 0.000443    1  119.      0.234 0.630
## 4 target_status:exposure  0.0137   0.00683     2  119.      3.60  0.0304
```

and the effect sizes:

```
F_to_eta2(f = aov11$statistic, df = aov11$NumDF, df_error=aov11$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.10           | [0.02, 1.00]
## 0.17           | [0.08, 1.00]
## 1.96e-03       | [0.00, 1.00]
## 0.06           | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

Pairwise comparison on Target Association:

```
emmeans(lmer(Accuracy ~ target_status + (1|participants), data = exp3_test_m), pairwise ~ target_status
```

```
## $emmeans
##  target_status emmean      SE   df lower.CL upper.CL
##  neutral        0.935 0.00975 50.5    0.915    0.955
##  none           0.907 0.00975 50.5    0.887    0.926
##  pleasant       0.936 0.00975 50.5    0.916    0.956
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast            estimate      SE  df t.ratio p.value
##  neutral - none      0.028423 0.00964 123   2.949  0.0105
##  neutral - pleasant -0.000977 0.00964 123  -0.101  1.0000
##  none - pleasant    -0.029400 0.00964 123  -3.050  0.0105
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: BY method for 3 tests
```

Now statistics for RTs:

```
mod12 = lmer(RT ~ target_status*exposure + target + (1|participants), data = exp3_test_m)
aov12 = tidy(anova(mod12))
aov12
```

```
## # A tibble: 4 x 7
##   term                     sumsq  meansq NumDF DenDF statistic     p.value
##   <chr>                    <dbl>   <dbl> <int> <dbl>     <dbl>       <dbl>
## 1 target_status           0.251   0.126      2  119.     18.1   0.000000140
## 2 exposure                0.0231  0.0231     1  119.      3.32  0.0709
## 3 target                  0.0210  0.0210     1  119.      3.03  0.0844
## 4 target_status:exposure  0.00552 0.00276    2  119.      0.398 0.673
```

Only the main effect of target status is significant. Here are the effect sizes:

```
F_to_eta2(f = aov12$statistic, df = aov12$NumDF, df_error=aov12$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.23            | [0.13, 1.00]
## 0.03            | [0.00, 1.00]
## 0.02            | [0.00, 1.00]
## 6.64e-03        | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

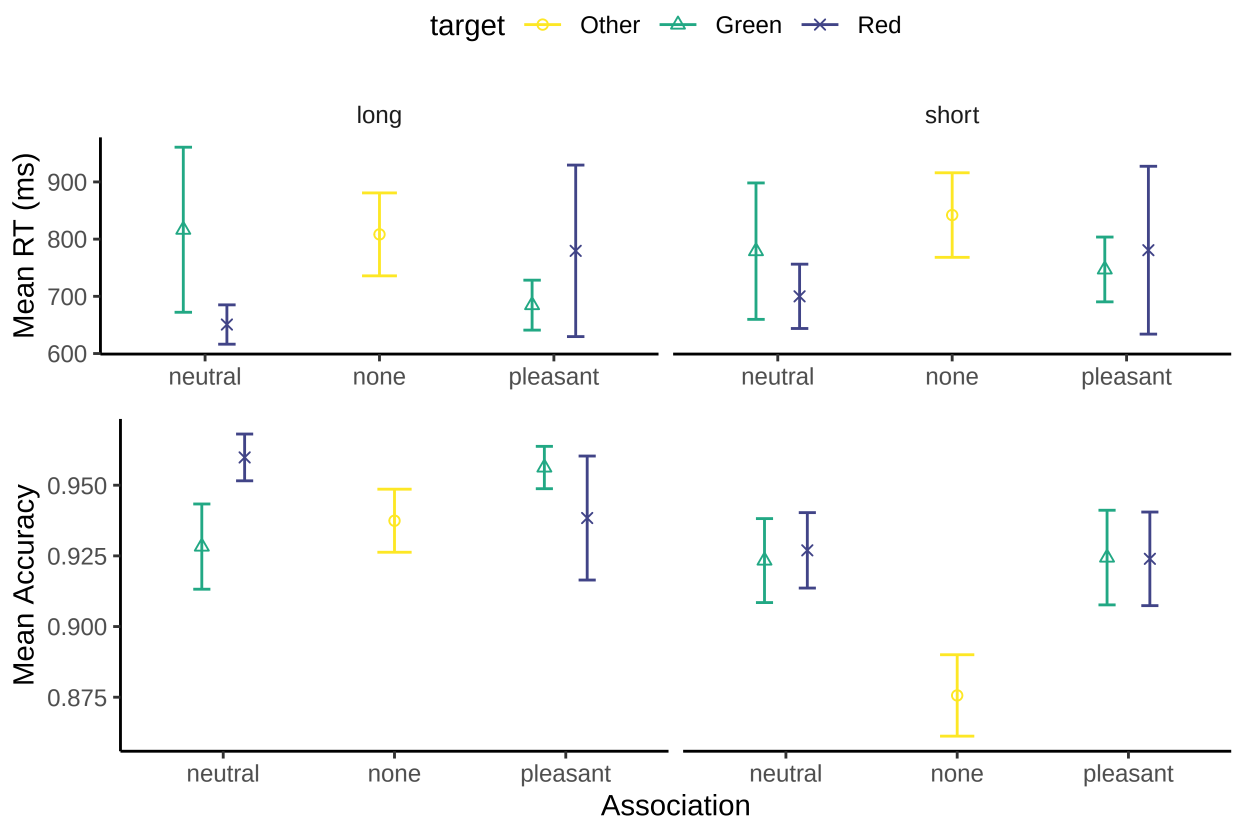
```
emmeans(lmer(RT ~ target_status + (1|participants), data = exp3_test_m), pairwise ~ target_status,
        adjust = "BY")
```

```
## $emmeans
##  target_status emmean     SE   df lower.CL upper.CL
##  neutral        0.734 0.0709 24.9    0.588    0.880
##  none           0.825 0.0709 24.9    0.679    0.971
##  pleasant       0.747 0.0709 24.9    0.601    0.893
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
##  contrast           estimate     SE  df t.ratio p.value
##  neutral - none      -0.0911 0.0169 123  -5.398  <.0001
##  neutral - pleasant  -0.0126 0.0169 123  -0.745  0.8387
##  none - pleasant      0.0785 0.0169 123   4.652  <.0001
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: BY method for 3 tests
```

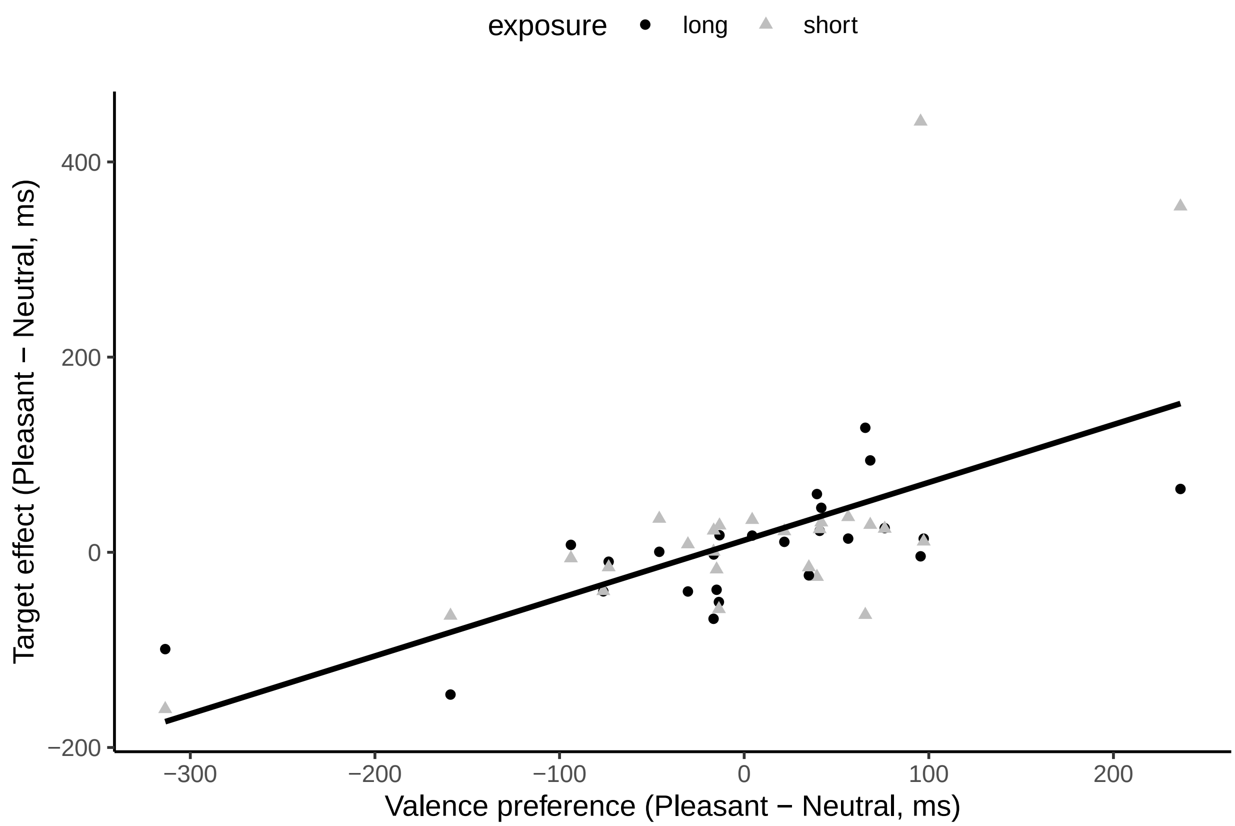
Visualize the mean RT etc.

```
# grand mean of the test
exp3_mm2 = exp3_test_m %>% group_by(target, exposure, target_status) %>%
  summarise(n=n(),mRT = mean(RT)*1000, seRT = sd(RT)/sqrt(n)*1000,
            mAccuracy = mean(Accuracy), seAccuracy = sd(Accuracy)/sqrt(n))

# correlation
# find out association group first
sub_group = exp3_train_m %>% select(participants, target, valence) %>%
  filter(valence == 'pleasant') %>% distinct() %>% #select unique
  unite("Group", target, valence)

ungroup(exp3_train_m) %>% select(participants, target, RT) %>% # calculate preference of color
  pivot_wider(names_from = target, values_from = RT) %>%
  mutate(colorDiff = sign(red - green)) %>%
  mutate(Preference = factor(colorDiff, labels = c("Red","Green"))) -> exp3_color_preference

exp3_train_m %>% select(participants, valence, RT) %>%
  pivot_wider(names_from = valence, values_from = RT) %>%
  mutate(Learning = (pleasant - neutral)*1000) %>% select (-neutral, -pleasant) -> exp3_learning
exp3_test_m %>% filter(target_status != 'none') %>% group_by(participants, exposure, target_status) %>%
  summarise(RT = mean(RT)) %>% pivot_wider(names_from = target_status, values_from = RT) %>%
  mutate(Interference = (pleasant - neutral)*1000) %>% select(-neutral, -pleasant) -> exp3_interference
exp3_correlation = left_join(exp3_learning, exp3_interference, by = c('participants')) %>%
  left_join(., exp3_color_preference, by = c('participants')) %>%
```

```r
  left_join(., sub_group, by = c('participants'))

# mean RT
fig7_1 = myplot(exp3_mm2, "target_status","RT","target") +
  xlab('Association') + ylab('Mean RT (ms)') +
  scale_color_viridis_d(begin = 0.2, labels = c("Other","Green","Red"), direction = -1) +
#  scale_color_manual(values = c("grey","green","red"), labels = c("Other","Green","Red")) +
  scale_shape_manual(values = c(1,2,4), labels = c("Other","Green","Red")) +
  facet_wrap(~exposure) +
  theme(axis.title.x = element_blank(),
                                panel.border = element_blank(), panel.grid.major = element_blank(),
             panel.grid.minor = element_blank(), strip.background = element_blank())

# mean ACC
fig7_2 = myplot(exp3_mm2, "target_status","Accuracy","target") +
  xlab('Association') + ylab('Mean Accuracy') +
  scale_color_viridis_d(begin = 0.2, labels = c("Other","Green","Red"), direction = -1) +
#  scale_color_manual(values = c("grey","green","red"), labels = c("Other","Green","Red")) +
  scale_shape_manual(values = c(1,2,4), labels = c("Other","Green","Red")) +
  facet_wrap(~exposure) +
  theme(legend.position = 'none',
                                             strip.text = element_blank())

fig7 = plot_grid(fig7_1, fig7_2, nrow = 2)
#save fig
ggsave(filename = './figures/fig_e3_test.png',fig7, width = 7, height = 5)

fig7
```




```
# fig of correlation
fig8 = ggplot(exp3_correlation, aes(Learning, Interference)) +
  geom_point(aes(color = exposure, shape = exposure)) +
  geom_smooth(method = 'lm', se = F, color = 'black') + theme_classic() +
  xlab('Valence preference (Pleasant - Neutral, ms)') +
  ylab('Target effect (Pleasant - Neutral, ms)') +
  scale_color_manual(values = c("black","grey")) +
  theme(legend.position = 'top')
#save fig
ggsave(filename = './figures/fig_e3_corr.png',fig8, width = 3.5, height = 3.5)
fig8
```




```r
# correlation

cor_test(data = ungroup(exp3_correlation), vars = Learning, vars2 = Interference )
```

```
## # A tibble: 1 x 8
##   var1     var2           cor statistic          p conf.low conf.high method
##   <chr>    <chr>        <dbl>     <dbl>      <dbl>    <dbl>     <dbl> <chr>
## 1 Learning Interference  0.63      5.57 0.00000115    0.421     0.770 Pearson
```

```r
summary(lm(Interference ~ Learning, data = exp3_correlation))
```

```
##
## Call:
## lm(formula = Interference ~ Learning, data = exp3_correlation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -115.30  -40.01   -6.59   18.03  372.42
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.3291    10.5841   1.165     0.25
## Learning      0.5928     0.1065   5.566 1.15e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 74.84 on 48 degrees of freedom
## Multiple R-squared:  0.3923, Adjusted R-squared:  0.3796
## F-statistic: 30.98 on 1 and 48 DF,  p-value: 1.145e-06
```

Omnibus analysis

1. Training phase

```
tr1 = exp1_train_m %>% select(name, Association, target, RT) %>% rename(Valence = Association) %>%
  mutate(Experiment = 'Exp. 1')
tr2 = ungroup(exp2_train_m) %>% select(name, Valence, target, RT) %>%
  mutate(Experiment = 'Exp. 2', name = paste0('e2_',name) )
tr3 = exp3_train_m %>% select(participants, valence, target, RT) %>%
  rename(Valence = valence, name = participants) %>% mutate(Experiment = 'Exp. 3', name = paste0('e3_',
tr3$Valence = factor(tr3$Valence, labels = c("Neutral","Pleasant"))
trs = rbind(tr1, tr2, tr3)

mod13 = lmer(RT ~ target*Valence + (1| name), data=trs)
aov13 = tidy(anova(mod13))
aov13
```

```
## # A tibble: 3 x 7
##   term                sumsq   meansq NumDF DenDF statistic   p.value
##   <chr>               <dbl>    <dbl> <int> <dbl>     <dbl>     <dbl>
## 1 target         0.117    0.117        1  171.   19.9     0.0000146
## 2 Valence        0.000412 0.000412     1  171.    0.0700 0.792
## 3 target:Valence 0.00604  0.00604      1  186.    1.03   0.312
```

It shows only the target color was significant. Here are the effect sizes:

```
F_to_eta2(f = aov13$statistic, df = aov13$NumDF, df_error=aov13$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.10           | [0.04, 1.00]
## 4.09e-04       | [0.00, 1.00]
## 5.49e-03       | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

now calculate Bayes factors for the main effects of target and valence

```
b1 = brm(RT ~ target + (1|name), data = trs, save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.39 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.490389 seconds (Warm-up)
## Chain 1:                0.213918 seconds (Sampling)
## Chain 1:                0.704307 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.540398 seconds (Warm-up)
## Chain 2:                0.281292 seconds (Sampling)
## Chain 2:                0.82169 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.5e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
```

```
## Chain 3:  Elapsed Time: 0.512799 seconds (Warm-up)
## Chain 3:                0.291715 seconds (Sampling)
## Chain 3:                0.804514 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.2e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.551359 seconds (Warm-up)
## Chain 4:                0.246789 seconds (Sampling)
## Chain 4:                0.798148 seconds (Total)
## Chain 4:
```

```
b0 = brm(RT ~ 1 + (1|name), data = trs, save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.41 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.588862 seconds (Warm-up)
## Chain 1:                0.246644 seconds (Sampling)
```

```
## Chain 1:                0.835506 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.4e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.565406 seconds (Warm-up)
## Chain 2:                0.344148 seconds (Sampling)
## Chain 2:                0.909554 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.515763 seconds (Warm-up)
## Chain 3:                0.281914 seconds (Sampling)
## Chain 3:                0.797677 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 4).
```

```
## Chain 4:
## Chain 4: Gradient evaluation took 1.5e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.551134 seconds (Warm-up)
## Chain 4:                0.260495 seconds (Sampling)
## Chain 4:                0.811629 seconds (Total)
## Chain 4:
```

```
bayes_factor(b1, b0)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 8
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
```

```
## Estimated Bayes factor in favor of b1 over b0: 275.20817
```

Valence:

```
b1 = brm(RT ~ Valence + (1|name), data = trs, save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.6e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.46 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
```

```
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.536052 seconds (Warm-up)
## Chain 1:                0.216063 seconds (Sampling)
## Chain 1:                0.752115 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.4e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.507422 seconds (Warm-up)
## Chain 2:                0.243187 seconds (Sampling)
## Chain 2:                0.750609 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.5e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.519351 seconds (Warm-up)
## Chain 3:                0.243299 seconds (Sampling)
## Chain 3:                0.76265 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '367c3127f25e3014b416c6e61efe826e' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.522959 seconds (Warm-up)
## Chain 4:                0.25941 seconds (Sampling)
## Chain 4:                0.782369 seconds (Total)
## Chain 4:
```

```
b0 = brm(RT ~ 1 + (1|name), data = trs, save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.39 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.535174 seconds (Warm-up)
## Chain 1:                0.220793 seconds (Sampling)
## Chain 1:                0.755967 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.4e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.561545 seconds (Warm-up)
## Chain 2:                0.217235 seconds (Sampling)
## Chain 2:                0.77878 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
## Chain 3:
## Chain 3:  Elapsed Time: 0.552953 seconds (Warm-up)
## Chain 3:                0.228171 seconds (Sampling)
## Chain 3:                0.781124 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'baf327a57cc58d11412fc9de48d6c39a' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.549916 seconds (Warm-up)
## Chain 4:                0.255022 seconds (Sampling)
## Chain 4:                0.804938 seconds (Total)
## Chain 4:
```

```
bayes_factor(b1, b0)
```

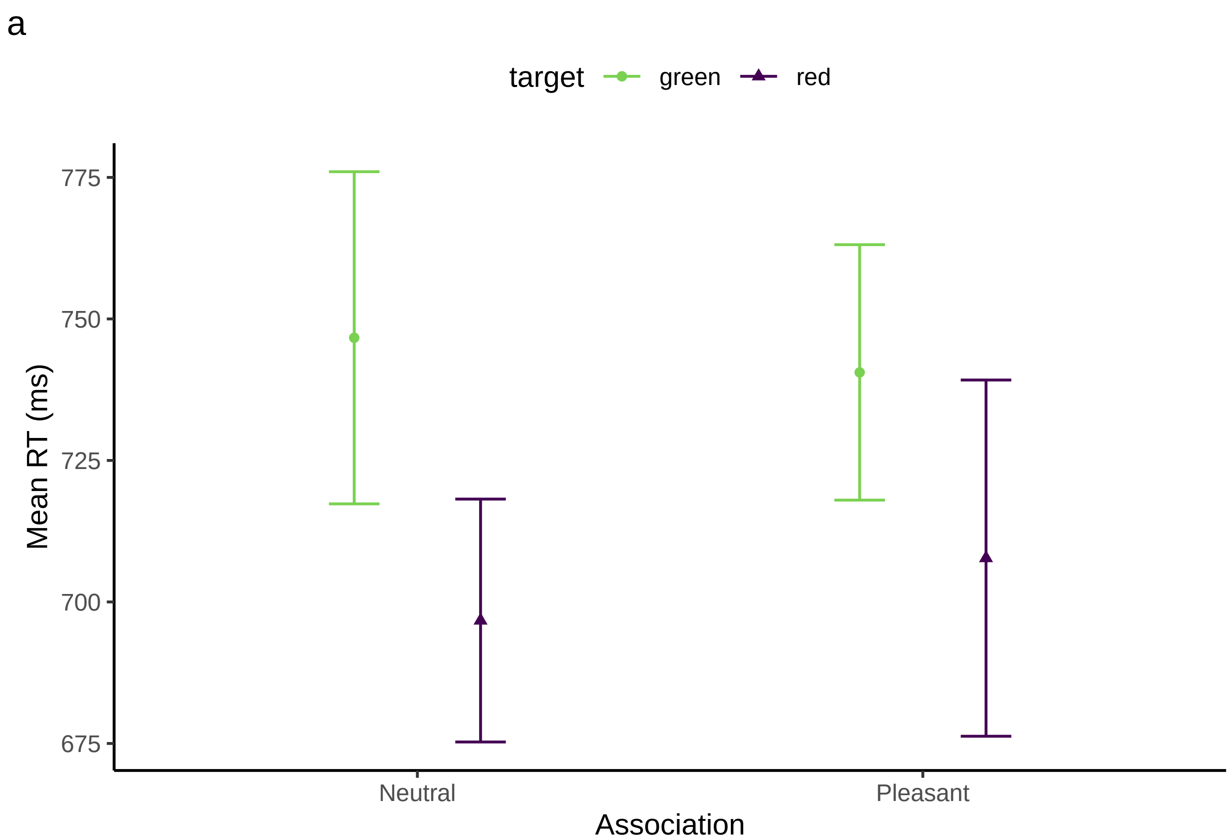
```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 8
```

```
## Estimated Bayes factor in favor of b1 over b0: 0.02603
```

Visualize the mean RTs in the training phase.

```
m_trs = trs %>% group_by(Valence, target) %>%
  summarise(n=n(),mRT = mean(RT)*1000, seRT = sd(RT)/sqrt(n)*1000)
```

```
fig_om_train = myplot(m_trs, "Valence","RT","target") +
  xlab('Association') + ylab('Mean RT (ms)') +
  scale_color_viridis_d( end = 0.8, direction  = -1) +
  labs(tag = "a")
fig_om_train
```



2. Test phase

```
te1 = exp1_test_m %>% group_by(name, Association, distractor) %>% summarise(RT = mean(RT)) %>%
  rename(Valence = Association) %>%   mutate(Experiment = 'Exp. 1') %>%
  rename(Color = distractor)
te2 = exp2_test_m %>% group_by(name, Valence, distractor) %>% summarise(RT = mean(RT)) %>%
  mutate(Experiment = 'Exp. 2', name = paste0('e2_',name) ) %>% rename(Color = distractor)
te3 = exp3_test_m %>% ungroup() %>% group_by(participants, target_status, target) %>%
  summarise(RT = mean(RT)) %>%
  rename(Valence = target_status, name = participants) %>%
  mutate(Experiment = 'Exp. 3', name = paste0('e3_', name)) %>% rename(Color = target)
te3$Valence = factor(te3$Valence, labels = c( "Neutral", "Other","Pleasant"))
tests = rbind(te1, te2, te3)

# here we only focus on the pleasant and neutral association in the test phases (ignore the absent asso
tests_np = tests %>% filter(Valence %in% c('Neutral', 'Pleasant'))
mod14 = lmer(RT ~ Color*Valence + (1| name),
  data=tests %>% filter(Valence %in% c('Neutral', 'Pleasant')))
aov14 = tidy(anova(mod14))
aov14
```

```
## # A tibble: 3 x 7
```

```
##   term             sumsq   meansq NumDF DenDF statistic p.value
##   <chr>            <dbl>    <dbl> <int> <dbl>     <dbl>   <dbl>
## 1 Color         0.00938  0.00938      1  171.      3.70  0.0560
## 2 Valence       0.000477 0.000477     1  171.      0.188 0.665
## 3 Color:Valence 0.0228   0.0228       1  179.      8.98  0.00312
```

and related effect sizes:

```
F_to_eta2(f = aov14$statistic, df = aov14$NumDF, df_error=aov14$DenDF)
```

```
## Eta2 (partial) |       95% CI
## -----------------------------
## 0.02           | [0.00, 1.00]
## 1.10e-03       | [0.00, 1.00]
## 0.05           | [0.01, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

Now calculate Bayes factors for the main effects of color and valence

```
b1 = brm(RT ~ Color + (1|name), data = tests_np,
        save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.34 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.631304 seconds (Warm-up)
## Chain 1:                0.443062 seconds (Sampling)
## Chain 1:                1.07437 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.5e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
```

```
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.641046 seconds (Warm-up)
## Chain 2:                0.437868 seconds (Sampling)
## Chain 2:                1.07891 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.615011 seconds (Warm-up)
## Chain 3:                0.436495 seconds (Sampling)
## Chain 3:                1.05151 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
```

```
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.704945 seconds (Warm-up)
## Chain 4:                0.433336 seconds (Sampling)
## Chain 4:                1.13828 seconds (Total)
## Chain 4:
```

```r
bayes_factor(b1, b0)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 8
```

```
## Estimated Bayes factor in favor of b1 over b0: 422889938941907690207581352715354112.00000
```

and Valence

```r
b1 = brm(RT ~ Valence + (1|name), data = tests_np,
         save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.53 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
```

```
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.644134 seconds (Warm-up)
## Chain 1:                0.439036 seconds (Sampling)
## Chain 1:                1.08317 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.7e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.17 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.617183 seconds (Warm-up)
## Chain 2:                0.306583 seconds (Sampling)
## Chain 2:                0.923766 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.619544 seconds (Warm-up)
## Chain 3:                0.443889 seconds (Sampling)
```

```
## Chain 3:                1.06343 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.6e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.651612 seconds (Warm-up)
## Chain 4:                0.428373 seconds (Sampling)
## Chain 4:                1.07998 seconds (Total)
## Chain 4:
```

```r
b0 = brm(RT ~ 1 + (1|name), data = tests_np,
          save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL 'b9108b952394403d0a7db815c0f478b1' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.4 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.686481 seconds (Warm-up)
## Chain 1:                0.449926 seconds (Sampling)
## Chain 1:                1.13641 seconds (Total)
```

```
## Chain 1:
##
## SAMPLING FOR MODEL 'b9108b952394403d0a7db815c0f478b1' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.708461 seconds (Warm-up)
## Chain 2:                0.380744 seconds (Sampling)
## Chain 2:                1.0892 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'b9108b952394403d0a7db815c0f478b1' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.24 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.701646 seconds (Warm-up)
## Chain 3:                0.449848 seconds (Sampling)
## Chain 3:                1.15149 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'b9108b952394403d0a7db815c0f478b1' NOW (CHAIN 4).
## Chain 4:
```

```
## Chain 4: Gradient evaluation took 1.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.719794 seconds (Warm-up)
## Chain 4:                0.460131 seconds (Sampling)
## Chain 4:                1.17993 seconds (Total)
## Chain 4:
```

```
bayes_factor(b1, b0)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 8
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
```

```
## Estimated Bayes factor in favor of b1 over b0: 0.01656
```

and their interaction:

```
b1 = brm(RT ~ Valence*Color + (1|name), data = tests_np,
         save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.43 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
```

```
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.706587 seconds (Warm-up)
## Chain 1:                0.456162 seconds (Sampling)
## Chain 1:                1.16275 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.682231 seconds (Warm-up)
## Chain 2:                0.398462 seconds (Sampling)
## Chain 2:                1.08069 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
```

```
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.720143 seconds (Warm-up)
## Chain 3:                0.456619 seconds (Sampling)
## Chain 3:                1.17676 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.697938 seconds (Warm-up)
## Chain 4:                0.424627 seconds (Sampling)
## Chain 4:                1.12256 seconds (Total)
## Chain 4:
```

```r
b0 = brm(RT ~ Valence + Color + (1|name), data = tests_np,
         save_all_pars = TRUE, family = gaussian())
```

```
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.33 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.648651 seconds (Warm-up)
## Chain 1:                0.4374 seconds (Sampling)
## Chain 1:                1.08605 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.664242 seconds (Warm-up)
## Chain 2:                0.441851 seconds (Sampling)
## Chain 2:                1.10609 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.628002 seconds (Warm-up)
## Chain 3:                0.451768 seconds (Sampling)
## Chain 3:                1.07977 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '3a430c42e2e77ebdce691cf3b8938668' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.8e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.667053 seconds (Warm-up)
## Chain 4:                0.4424 seconds (Sampling)
## Chain 4:                1.10945 seconds (Total)
## Chain 4:
```

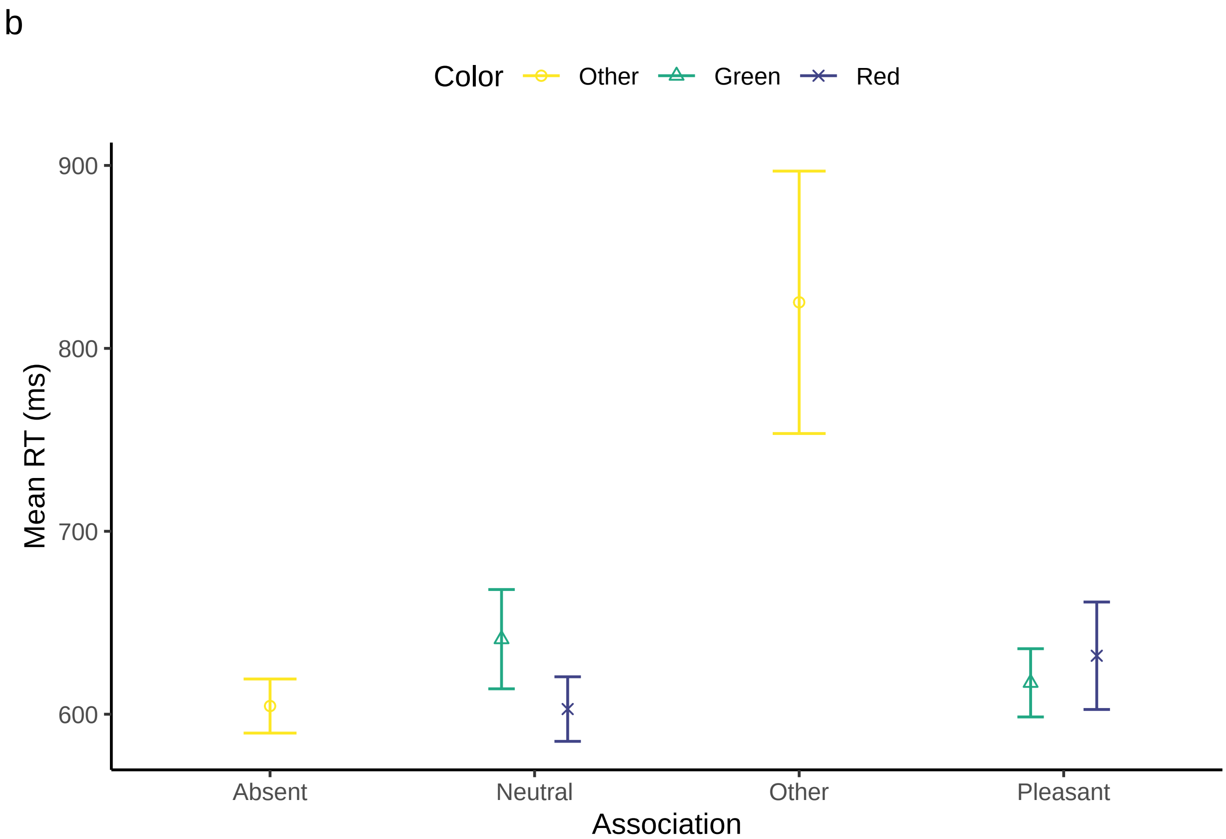
```r
bayes_factor(b1, b0)
```

```
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
## Iteration: 8
## Iteration: 9
## Iteration: 1
## Iteration: 2
## Iteration: 3
## Iteration: 4
## Iteration: 5
## Iteration: 6
## Iteration: 7
```

```
## Estimated Bayes factor in favor of b1 over b0: 3.78363
```

Visualize the mean RTs in the test phase.

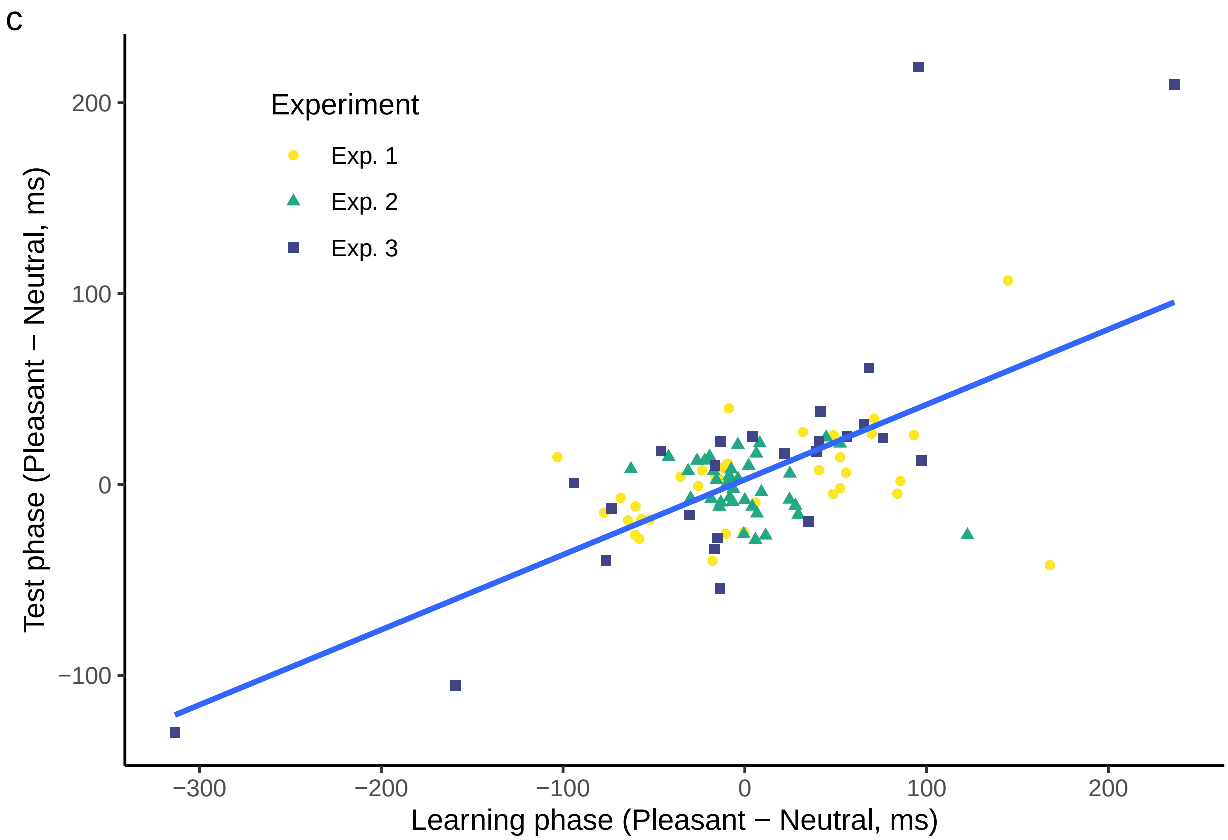
```
m_tests = tests %>% group_by(Valence, Color) %>%
  summarise(n=n(),mRT = mean(RT)*1000, seRT = sd(RT)/sqrt(n)*1000)

fig_om_test = myplot(m_tests, "Valence","RT","Color") + xlab('Association') +
  ylab('Mean RT (ms)') +
  scale_color_viridis_d(begin = 0.2, labels = c("Other","Green","Red"), direction = -1) +
#  scale_color_manual(values = c("grey","green","red"), labels = c("Other","Green","Red")) +
  scale_shape_manual(values = c(1,2,4), labels = c("Other","Green","Red")) + labs(tag = "b")

fig_om_test
```



And visualize the correlation between the training and test phase.

```
cor1 = exp1_correlation %>% group_by(name) %>%
  summarize(Learning = mean(Learning), Interference = mean(Interference)) %>%
            mutate(Experiment = 'Exp. 1')
cor2 = exp2_correlation %>% ungroup() %>%group_by(name) %>%
  summarize(Learning = mean(Learning), Interference = mean(Interference)) %>%
            mutate(Experiment = 'Exp. 2')
cor3 = exp3_correlation %>% group_by(participants) %>%
  summarize(Learning = mean(Learning), Interference = mean(Interference)) %>%
  rename(name = participants) %>% mutate(Experiment = 'Exp. 3')
corrs = rbind(cor1, cor2, cor3)

fig_omnibus = ggplot(corrs, aes(Learning, Interference)) +
  geom_point(aes(color = Experiment, shape = Experiment)) +
  geom_smooth(method = 'lm', se = FALSE) + theme_classic() +
```

```
  xlab('Learning phase (Pleasant - Neutral, ms)') +
  ylab('Test phase (Pleasant - Neutral, ms)') +
  scale_color_viridis_d(begin = 0.2, direction = -1) +
  theme(legend.position = c(0.2,0.8))+ labs(tag = "c")
fig_omnibus
```



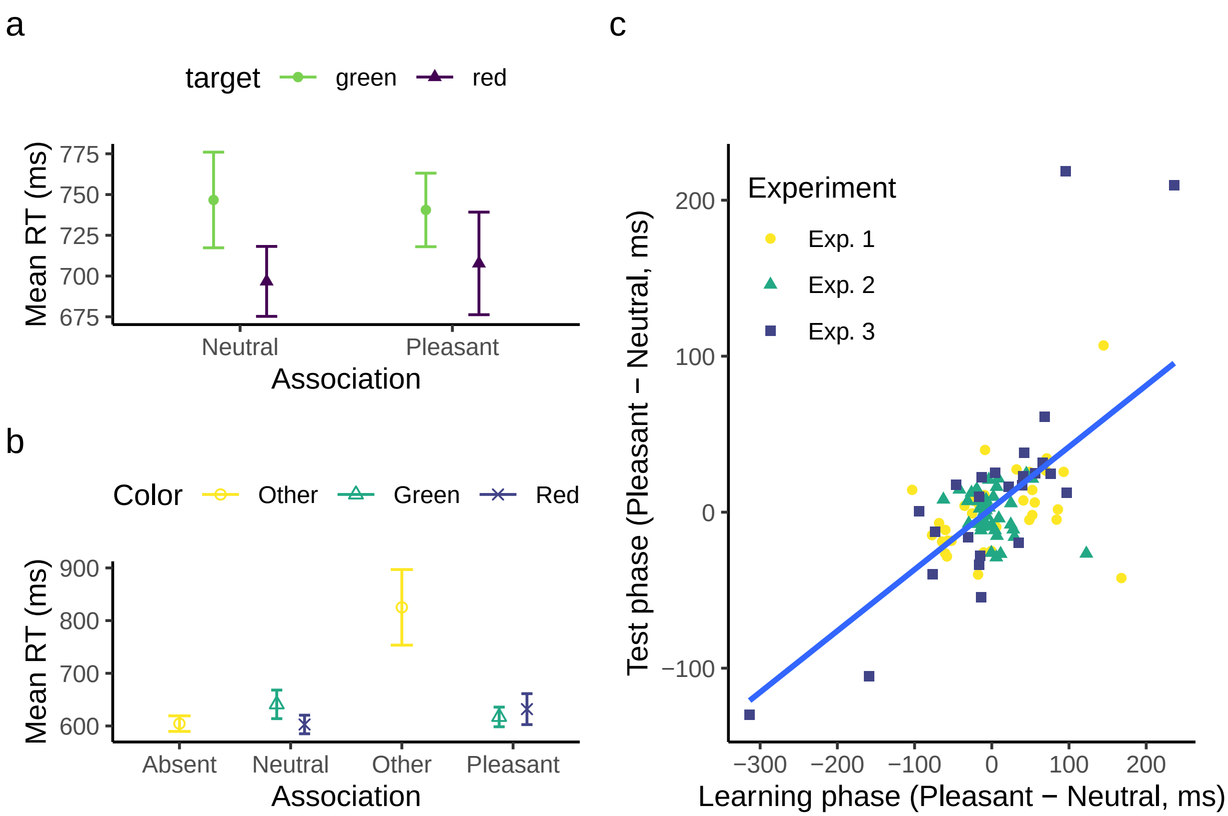
The correlation test showed a significant correlation between the learning and test phase.

```
cor_test(data = ungroup(corrs), vars = Learning, vars2 = Interference )
```

```
## # A tibble: 1 x 8
##   var1     var2           cor statistic        p conf.low conf.high method
##   <chr>    <chr>        <dbl>     <dbl>    <dbl>    <dbl>     <dbl> <chr>
## 1 Learning Interference  0.62      7.84 5.82e-12    0.485     0.730 Pearson
```

Now combine all figures together into one and save it.

```
fig_om = (fig_om_train  / fig_om_test + labs(tag = "b")) | fig_omnibus
#save fig
ggsave(filename = './figures/fig_omn.png',fig_om, width = 7, height = 5)
fig_om
```

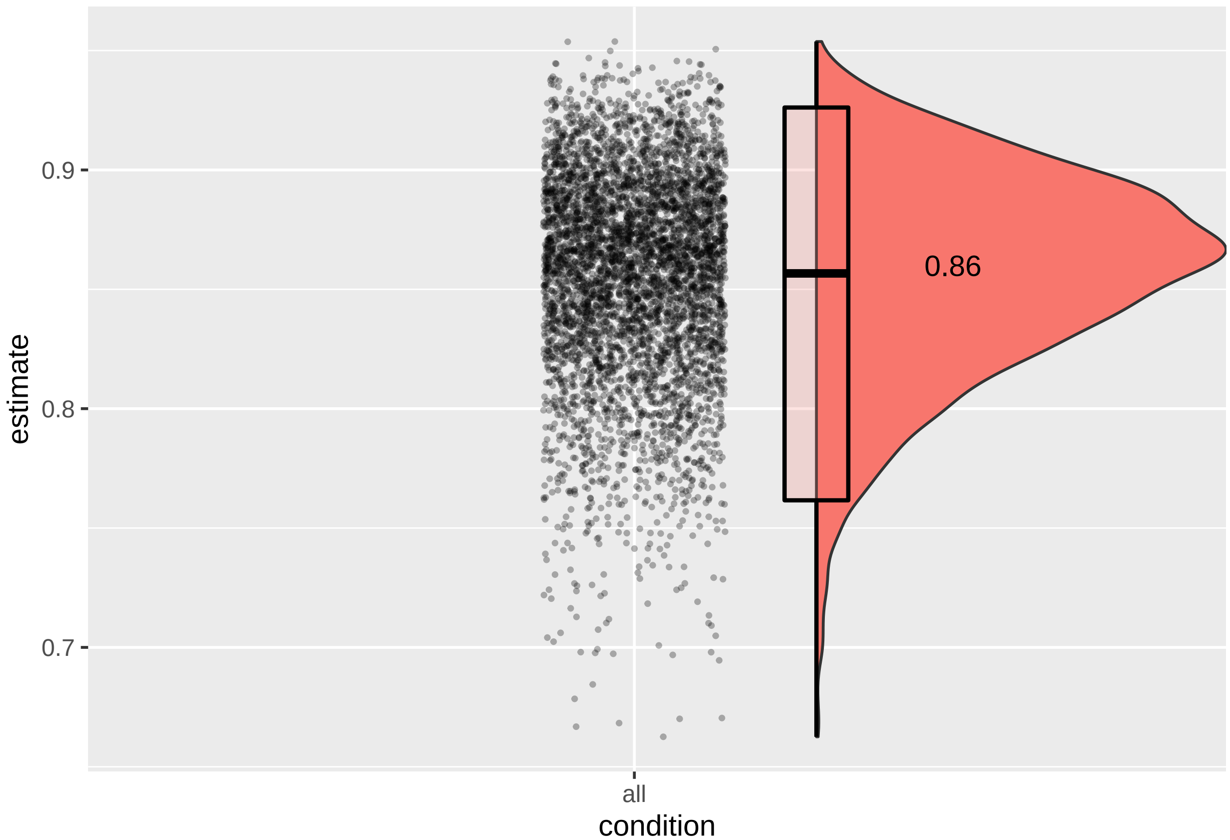


Splithalf reliability

1. Experiment 1

```
# split half reliability
# prepare clean data from raw_exp1$training
droplevels(raw_exp1$training) %>% select(name, trlNr, Association, RT) -> exp1_training
difference1 = splithalf(data = exp1_training,
      outcome = "RT",
      score = "difference",
      halftype = "random",
      permutations = 5000,
      var.RT = "RT",
      var.participant = "name",
      var.compare = "Association",
      compare1 = "Neutral",
      compare2 = "Pleasant",
      average = "mean",
      plot = TRUE)
```

```
##   |                                                                                |
## [1] "Calculating split half estimates"
## [1] "split half estimates for 5000 random splits"
##   condition  n spearmanbrown SB_low SB_high
## 1       all 36          0.86   0.76    0.93
## [1] "this could be reported as: using 5000 random splits, the spearman-brown corrected reliability
```




```
raw_exp1$testing %>% select(name, trlNr, Duration, Association, RT) %>%
    filter(Association != "Absent") %>% droplevels() -> exp1_testing
difference2 = splithalf(data = exp1_testing,
      outcome = "RT",
      score = "difference",
      conditionlist = c("Short","Long"),
      halftype = "random",
      permutations = 5000,
      var.RT = "RT",
      var.condition = "Duration",
      var.participant = "name",
      var.compare = "Association",
      compare1 = "Neutral",
      compare2 = "Pleasant",
      average = "mean")
```

```
##   |                                                                              |
##   |                                                                              |
## [1] "Calculating split half estimates"
## [1] "split half estimates for 5000 random splits"
##   condition  n spearmanbrown SB_low SB_high
## 1      Long 36          0.33  -0.09    0.64
## 2     Short 36          0.17  -0.35    0.59
## [1] "this could be reported as: using 5000 random splits, the spearman-brown corrected reliability  e
```

2. Experiment 2

```r
# split half reliability
# prepare clean data from raw_exp1$training
droplevels(raw_exp2$training) %>% select(name, trlNr, Arousal, Valence, RT) -> exp2_training
difference3 = splithalf(data = exp2_training,
      outcome = "RT",
      score = "difference",
      halftype = "random",
      conditionlist = c("high","low"),
      permutations = 5000,
      var.RT = "RT",
      var.participant = "name",
      var.compare = "Valence",
      var.condition = "Arousal",
      compare1 = "Neutral",
      compare2 = "Pleasant",
      average = "mean")
```

```
##   |                                                                      |
##   |                                                                      |
## [1] "Calculating split half estimates"
## [1] "split half estimates for 5000 random splits"
##   condition  n spearmanbrown SB_low SB_high
## 1      high 38          0.41   0.21    0.58
## 2       low 38          0.34  -0.07    0.62
## [1] "this could be reported as: using 5000 random splits, the spearman-brown corrected reliability  e
```

```r
raw_exp2$testing %>% select(name, trlNr, Duration, Arousal, Valence, RT) %>%
    filter(Valence != "Absent") %>%
    # combine columns Arousal and Duration into one column
    mutate(Condition = paste0(Arousal, "_",Duration)) %>% droplevels() -> exp2_testing
difference4 = splithalf(data = exp2_testing,
      outcome = "RT",
      score = "difference",
      conditionlist = c("high_Short","low_Short","high_Long","low_Long"),
      halftype = "random",
      permutations = 5000,
      var.RT = "RT",
      var.condition = "Condition",
      var.participant = "name",
      var.compare = "Valence",
      compare1 = "Neutral",
      compare2 = "Pleasant",
      average = "mean")
```

```
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
## [1] "Calculating split half estimates"
## [1] "split half estimates for 5000 random splits"
##    condition  n spearmanbrown SB_low SB_high
## 1  high_Long 38          0.23  -0.06    0.47
## 2 high_Short 38          0.06  -0.33    0.44
## 3   low_Long 38         -0.13  -0.48    0.31
```

```
## 4  low_Short 38          0.32  -0.15    0.63
## [1] "this could be reported as: using 5000 random splits, the spearman-brown corrected reliability
```

2. Experiment 3

```
# split half reliability
# prepare clean data from raw_exp1$training
droplevels(raw_exp3$training) %>% select(participants, valence, rt) -> exp3_training
difference5 = splithalf(data = exp3_training,
      outcome = "RT",
      score = "difference",
      halftype = "random",
      permutations = 5000,
      var.RT = "rt",
      var.participant = "participants",
      var.compare = "valence",
      compare1 = "neutral",
      compare2 = "pleasant",
      average = "mean")
```

```
##   |                                                                              |
## [1] "Calculating split half estimates"
## [1] "split half estimates for 5000 random splits"
##   condition  n spearmanbrown SB_low SB_high
## 1       all 25          0.71   0.48    0.85
## [1] "this could be reported as: using 5000 random splits, the spearman-brown corrected reliability
```

```
raw_exp3$testing %>% select(participants, exposure, target_status, rt) %>%
    filter(target_status != "none") %>% droplevels() -> exp3_testing
difference6 = splithalf(data = exp3_testing,
      outcome = "RT",
      score = "difference",
      conditionlist = c("short","long"),
      halftype = "random",
      permutations = 5000,
      var.RT = "rt",
      var.condition = "exposure",
      var.participant = "participants",
      var.compare = "target_status",
      compare1 = "neutral",
      compare2 = "pleasant",
      average = "mean")
```

```
##   |                                                                              |
##   |                                                                              |
## [1] "Calculating split half estimates"
## [1] "split half estimates for 5000 random splits"
##   condition  n spearmanbrown SB_low SB_high
## 1      long 25          0.36  -0.06    0.63
## 2     short 25          0.57   0.24    0.75
## [1] "this could be reported as: using 5000 random splits, the spearman-brown corrected reliability
```

End of the document.