installing required libraries and loading data:

recoding conditions:

Before we begin our analysis we must recode the conditions into factors of interest:

```
## 'data.frame':    10684 obs. of  11 variables:
##  $ X        : int  4006 4007 4008 4009 4010 4011 4012 4013 4014 4015 ...
##  $ subj     : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ item     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ pos      : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ word     : Factor w/ 318 levels "Agent","Agentin",..: 66 55 59 229 65 76 270 287 120 81 ...
##  $ RT       : int  852 1022 794 1625 1025 667 942 684 2518 887 ...
##  $ newCond  : Factor w/ 6 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ newSubj  : Factor w/ 144 levels "S101","S1010",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amb      : chr  "amb" "amb" "amb" "amb" ...
##  $ firstGen : chr  "f" "f" "f" "f" ...
##  $ secondGen: chr  "f" "f" "f" "f" ...
```

```
## 'data.frame':    4320 obs. of  11 variables:
##  $ X        : int  4010 4011 4012 4013 4014 4280 4281 4282 4283 4284 ...
##  $ subj     : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ item     : int  34 34 34 34 34 33 33 33 33 33 ...
##  $ pos      : int  4 5 6 7 8 4 5 6 7 8 ...
##  $ word     : Factor w/ 318 levels "Agent","Agentin",..: 65 76 270 287 120 65 318 140 136 120 ...
##  $ RT       : int  1025 667 942 684 2518 728 475 397 364 1153 ...
##  $ newCond  : Factor w/ 6 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ newSubj  : Factor w/ 144 levels "S101","S1010",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amb      : chr  "amb" "amb" "amb" "amb" ...
##  $ firstGen : chr  "f" "f" "f" "f" ...
##  $ secondGen: chr  "f" "f" "f" "f" ...
```

plotting reading time:

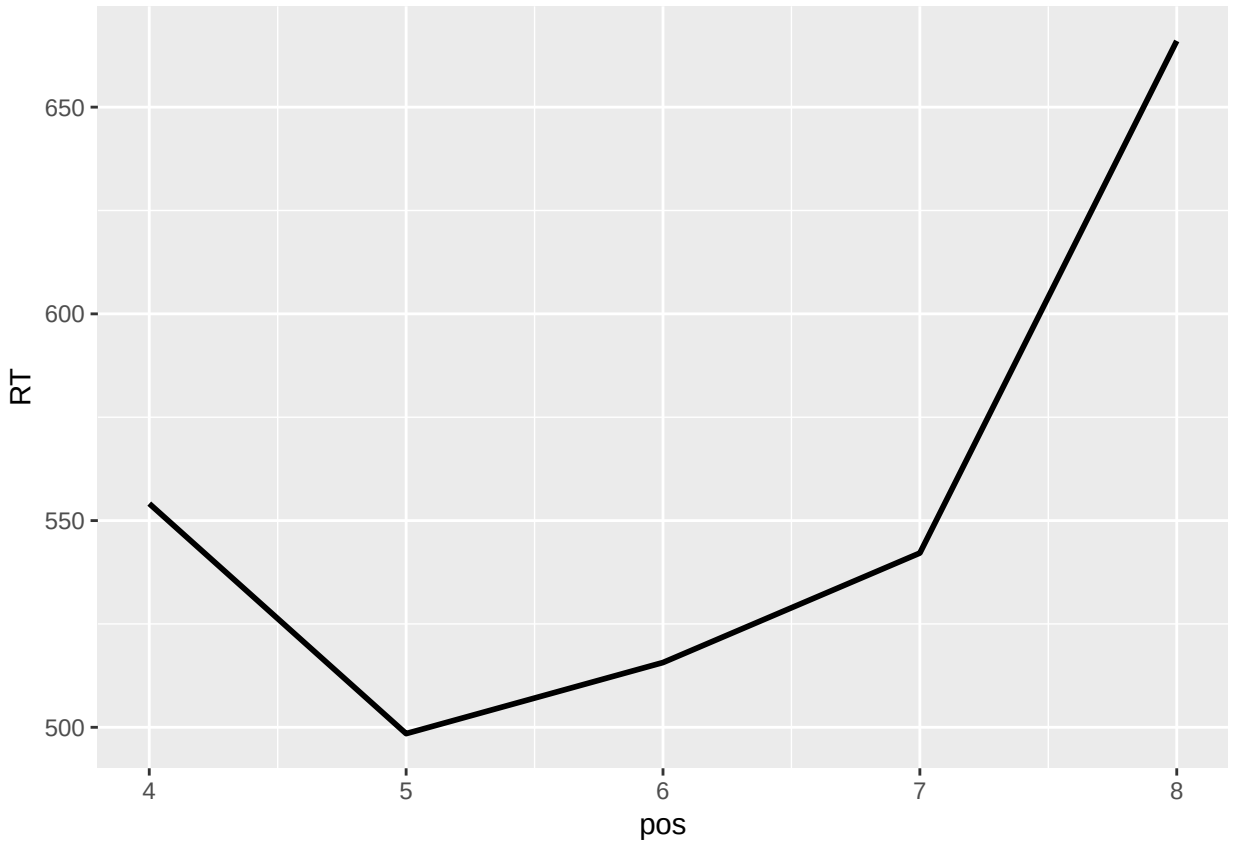
Now in order to get a sense of the distribution of the data, we make the following plots:

```
# reading times across all conditions:

lvP0 <- ggplot(lvRel,
               aes(pos, RT))

lvP1 <- lvP0 + stat_summary(fun.y = mean, geom = "line", size = 1)

lvP1
```

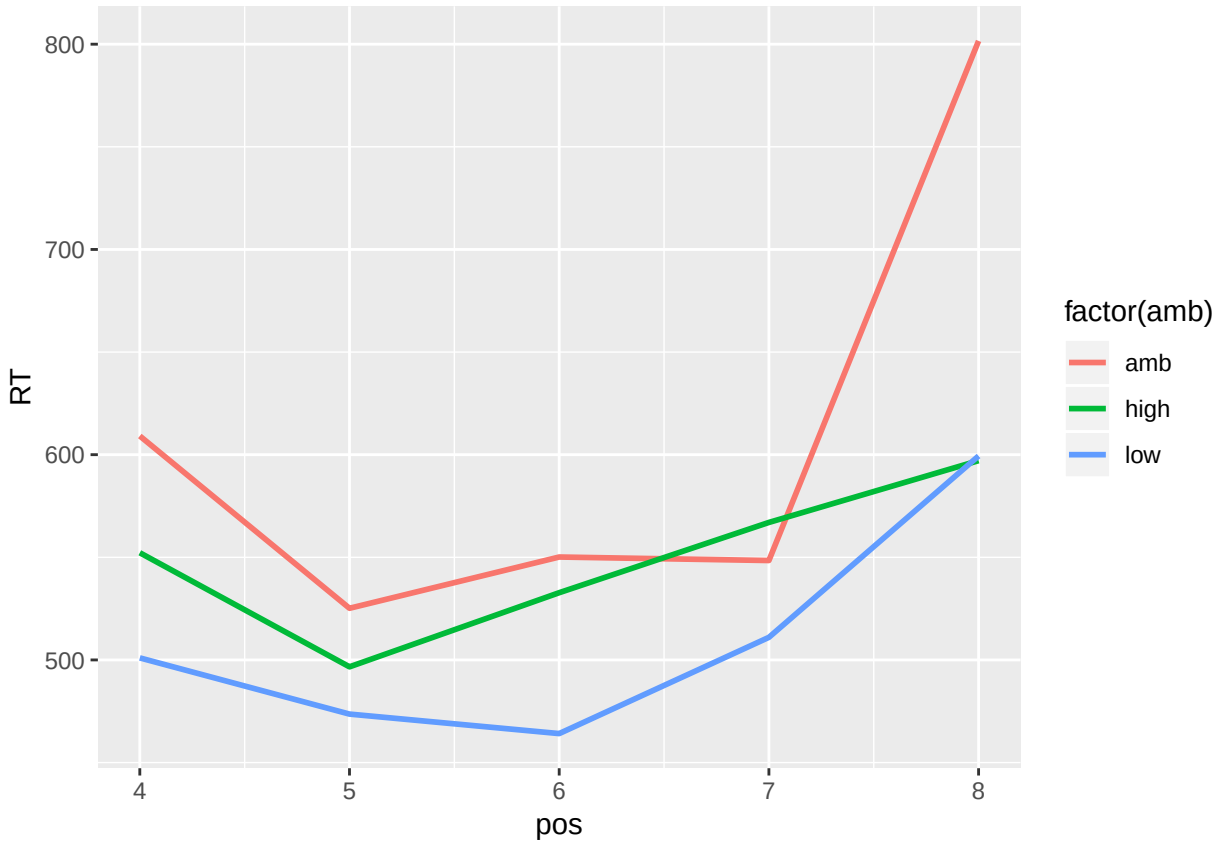



```
# separately for the three ambiguity values:

lvP0 <- ggplot(lvRel,
               aes(pos, RT,
                   colour = factor(amb)))

lvP1 <- lvP0 + stat_summary(fun.y = mean, geom = "line", size = 1)

lvP1
```

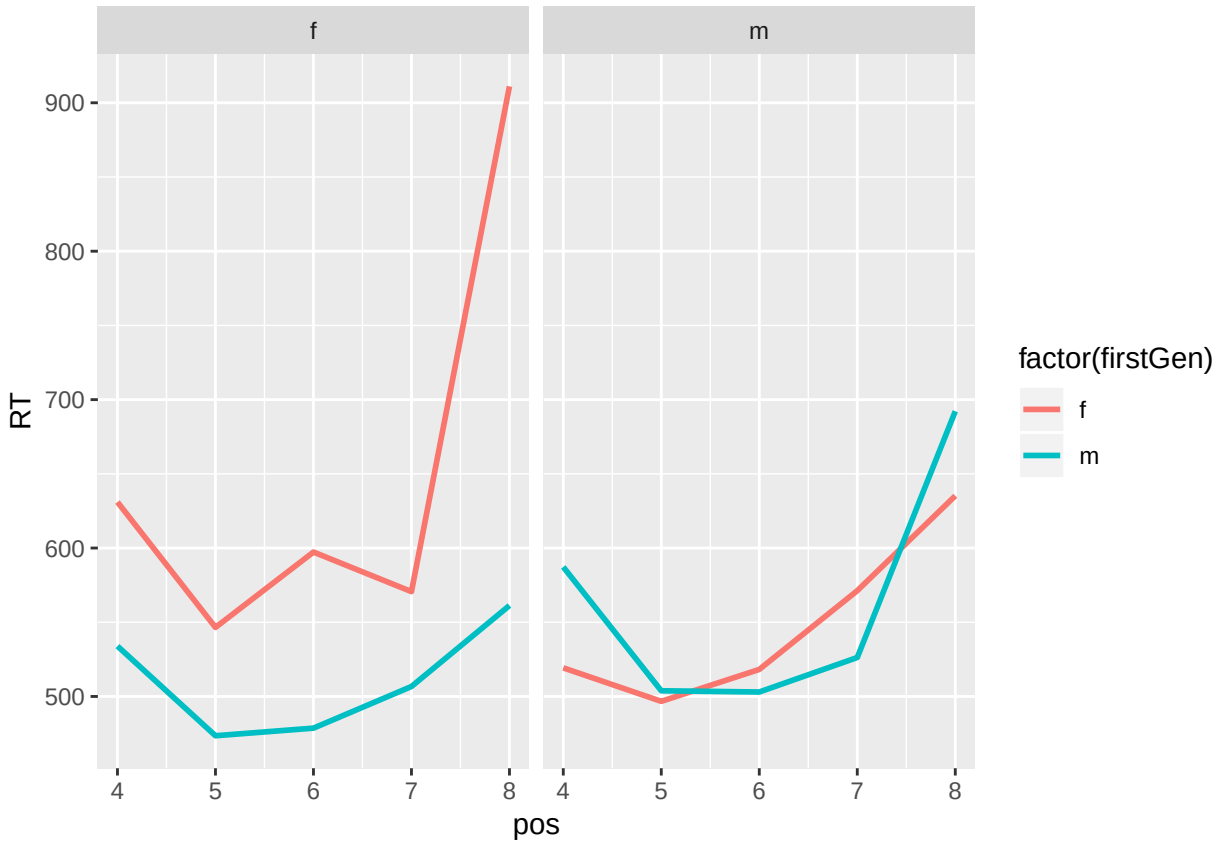



```
# separately for the gender of the first noun as a function of the gender of the second noun:

lvP0 <- ggplot(lvRel,
               aes(pos, RT,
                   colour = factor(firstGen)))

lvP1 <- lvP0 + stat_summary(fun.y = mean, geom = "line", size = 1) +
  facet_wrap(~ secondGen)

lvP1
```

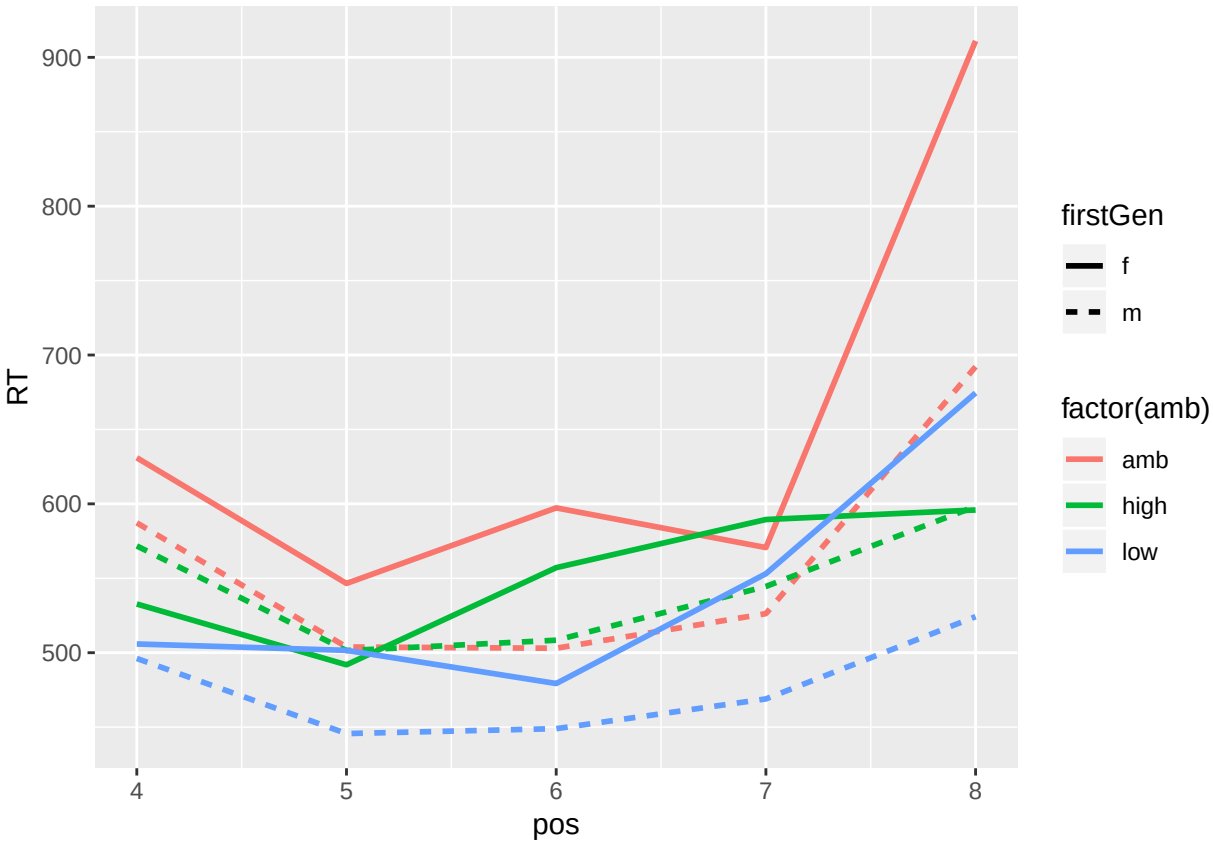



```
# and finally putting it all into a single plot, by ambiguity and gender of the first noun:

lvP0 <- ggplot(lvRel,
               aes(pos, RT,
                   colour = factor(amb),
                   linetype = firstGen))

lvP1 <- lvP0 + stat_summary(fun.y = mean, geom = "line", size = 1)

lvP1
```



anova analysis:

We are now ready to perform the ANOVA that tests the influence of the factor values on reading times at a given position.

```
# Calculating means:

subjectMeans <- lvRel %>%
  filter(pos == 6) %>%
  group_by(newSubj, amb, firstGen) %>%
  summarise(meanRT = mean(RT)) %>%
  data.frame()

colnames(subjectMeans) <- c("subj",
                            "amb",
                            "firstGen",
                            "meanRT")


grandMean <- mean(subjectMeans$meanRT)
g_means <- tapply(subjectMeans$meanRT, subjectMeans$firstGen, mean)
pos_means <- tapply(subjectMeans$meanRT,subjectMeans$amb, mean)
g_pos_means <- tapply(subjectMeans$meanRT, list(subjectMeans$firstGen,subjectMeans$amb), mean)

grandMean;g_means;pos_means;g_pos_means
```

```
## [1] 515.6725
```

```
##        f        m
## 544.5718 486.7731
```

```
##      amb     high      low
## 550.1458 532.7569 464.1146
```

```
##        amb     high      low
## f 597.2986 557.1319 479.2847
## m 502.9931 508.3819 448.9444
```

```
# Calculating sums of squares:

sos_total <- sum((subjectMeans$meanRT - grandMean)^2)
sos_total
```

```
## [1] 4006206
```

```
sos_gen <- sum((g_means - grandMean)^2)
sos_gen <- sos_gen*24*3

sos_amb <- sum((pos_means-grandMean)^2)
sos_amb <- sos_amb*24*2

acc <- 0
for(i in 1:2){
  for(j in 1:3){
    acc = acc + (g_pos_means[i,j] - g_means[i] - pos_means[j] + grandMean)^2
  }
}
sos_g_amb <- 24*sum(acc)
sos_g_amb
```

```
## [1] 26023.13
```

```
sos_err <- sos_total-sos_gen-sos_amb-sos_g_amb
sos_err
```

```
## [1] 3661270
```

```
# Calculating mean square errors:

ms_total <- sos_total/(144-1)
ms_gen <- sos_gen/(2-1)
ms_amb <- sos_amb/(3-1)
ms_gen_amb <- sos_g_amb/((2-1)*(3-1))
ms_err <- sos_err/(144-2*3)

# Calculating F stats:

f_gen <- ms_gen / ms_err
f_amb <- ms_amb / ms_err
f_g_amb <- ms_gen_amb/ ms_err

# Comparing results to output of aov():

f_gen;f_amb;f_g_amb
```

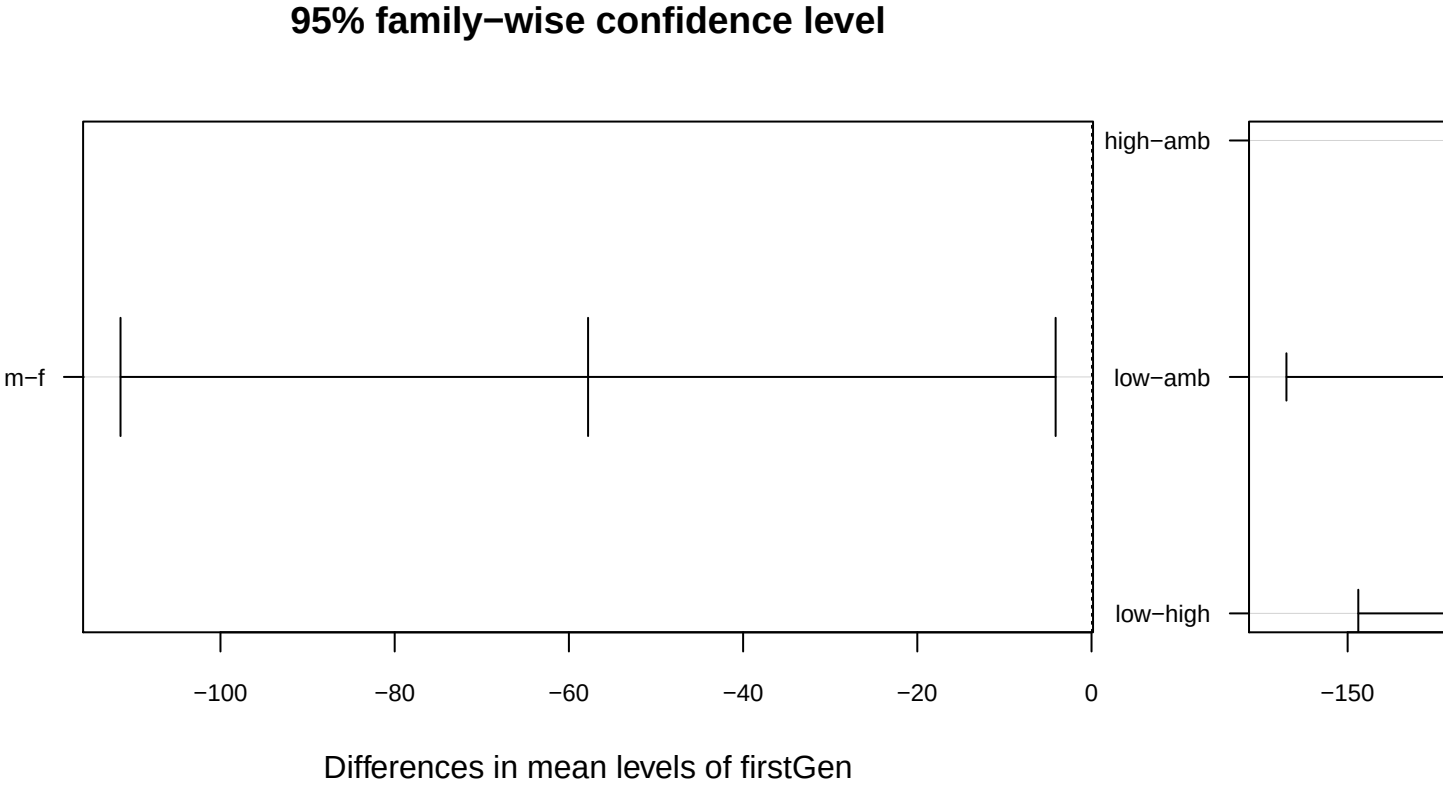
```
## [1] 4.532988
## [1] 3.743711
## [1] 0.4904298
ms_total;ms_gen;ms_amb;ms_gen_amb;ms_err
## [1] 28015.43
## [1] 120264.5
## [1] 99324.19
## [1] 13011.57
## [1] 26530.94
sos_total;sos_gen;sos_amb;sos_g_amb;sos_err
## [1] 4006206
## [1] 120264.5
## [1] 198648.4
## [1] 26023.13
## [1] 3661270
summary(aov(meanRT ~ firstGen + amb + firstGen:amb,
            data = subjectMeans))
##               Df  Sum Sq Mean Sq F value Pr(>F)
## firstGen       1  120264  120264   4.533 0.0350 *
## amb            2  198648   99324   3.744 0.0261 *
## firstGen:amb   2   26023   13012   0.490 0.6134
## Residuals    138 3661270   26531
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

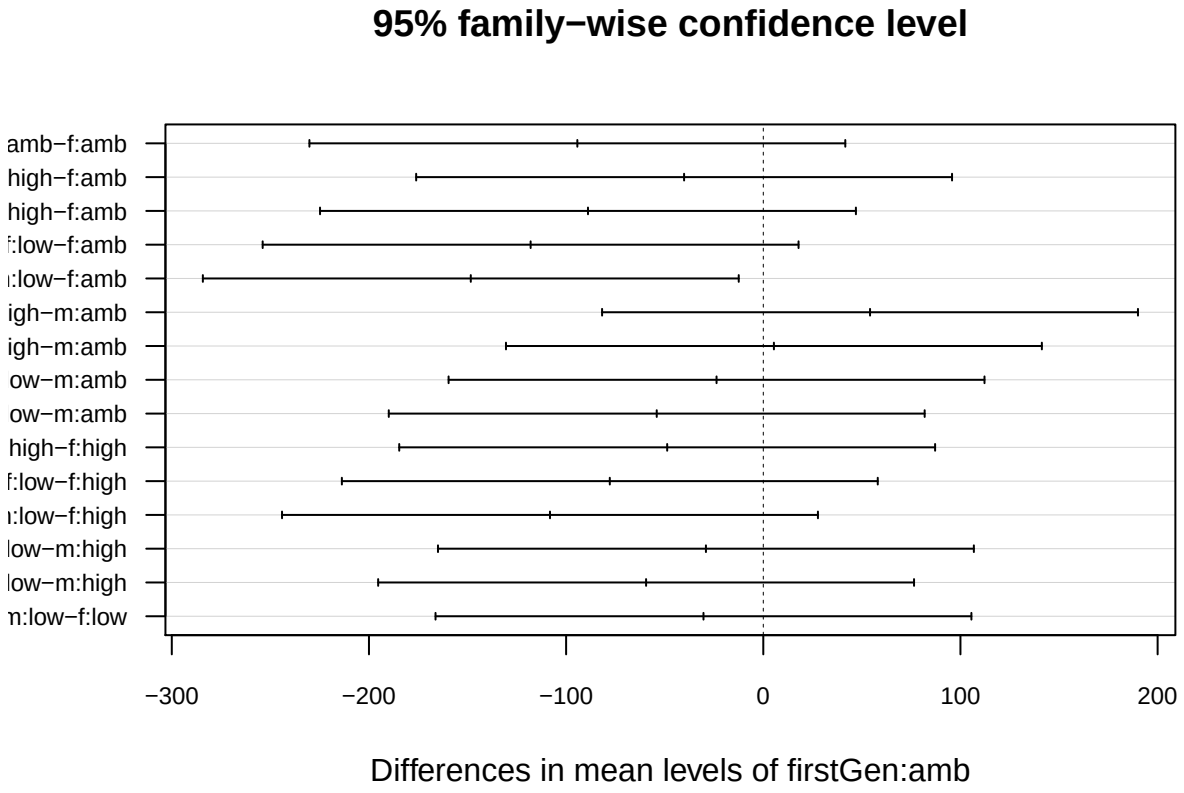
We see that our handmade ANOVA produces the same results as the aov() function. From the p values it seems like the gender of the first noun and the attachment position have some effect on the reading times as the p values are < 0.05. The interaction factor does not have a significant effect on reading times.

tukey's HSD:

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = meanRT ~ firstGen + amb + firstGen:amb, data = subjectMeans)
##
## $firstGen
##          diff       lwr       upr     p adj
## m-f -57.79861 -111.4769 -4.120348 0.0350232
##
## $amb
##               diff       lwr       upr     p adj
## high-amb -17.38889  -96.1634 61.385623 0.8602332
## low-amb  -86.03125 -164.8058 -7.256738 0.0286618
## low-high -68.64236 -147.4169 10.132151 0.1010856
```

```
##
## $`firstGen:amb`
##                       diff        lwr       upr     p adj
## m:amb-f:amb     -94.305556 -230.20237  41.59126 0.3444868
## f:high-f:amb    -40.166667 -176.06348  95.73015 0.9564812
## m:high-f:amb    -88.916667 -224.81348  46.98015 0.4120587
## f:low-f:amb    -118.013889 -253.91070  17.88293 0.1283709
## m:low-f:amb    -148.354167 -284.25098 -12.45735 0.0236764
## f:high-m:amb     54.138889  -81.75793 190.03570 0.8587519
## m:high-m:amb      5.388889 -130.50793 141.28570 0.9999972
## f:low-m:amb     -23.708333 -159.60515 112.18848 0.9959444
## m:low-m:amb     -54.048611 -189.94543  81.84820 0.8596060
## m:high-f:high   -48.750000 -184.64681  87.14681 0.9047012
## f:low-f:high    -77.847222 -213.74404  58.04959 0.5633667
## m:low-f:high   -108.187500 -244.08431  27.70931 0.2007950
## f:low-m:high    -29.097222 -164.99404 106.79959 0.9894936
## m:low-m:high    -59.437500 -195.33431  76.45931 0.8039069
## m:low-f:low     -30.340278 -166.23709 105.55654 0.9872882
```





The largest and most significant pair-wise comparisons (p val < 0.05) are between 'm' and 'f' for the gender of the first noun, between the low and ambiguous attachment positions and the interaction of these two which would be the 'm:low-f:amb' comparison.