B.L. Experiment Result Analysis

Contents

1 Overview

1.1 Summaries of the columns

```
##                                          stimulus          lvl
##  clothing_n_01|dresser_n_05                  :  50   hypernym:3300
##  hand_tool_n_01|highboy_n_01                 :  50   bl      :3300
##  musical_instrument_n_01|armchair_n_01       :  50   hyponym :3300
##  acoustic_guitar_n_01|acoustic_guitar_n_01:  25
##  acoustic_guitar_n_01|damson_n_01            :  25
##  apple_n_01|granny_smith_n_01                :  25
##  (Other)                                     :9675
##                       branch          p_id         accuracy        react_time
##  edible_fruit.n.01       :1650   1      : 396   Min.   :0.0000   Min.   :  29
##  musical_instrument.n.01:2100   2      : 396   1st Qu.:1.0000   1st Qu.: 457
##  clothing.n.01           :2250   3      : 396   Median :1.0000   Median : 563
##  hand_tool.n.01          :1950   4      : 396   Mean   :0.8851   Mean   : 648
##  furniture.n.01          :1950   5      : 396   3rd Qu.:1.0000   3rd Qu.: 736
##                                  6      : 396   Max.   :1.0000   Max.   :8148
##                                  (Other):7524
```

The counts of all stimuli, levels and participant IDs should be the same within their group. If some are higher than others, you might have duplicates and need to take another look at the input data before the evaluation. Here, you can also already see the overall mean accuracy of the participant answers, as well as the mean and median reaction times.

1.2 Reaction time and Mistake Overview

The following box plots in Fig.1 give a short overview of the reaction time means and quantiles with regards to the category levels and also to the stimulus type. A closer look into reaction times can be found in section 4.

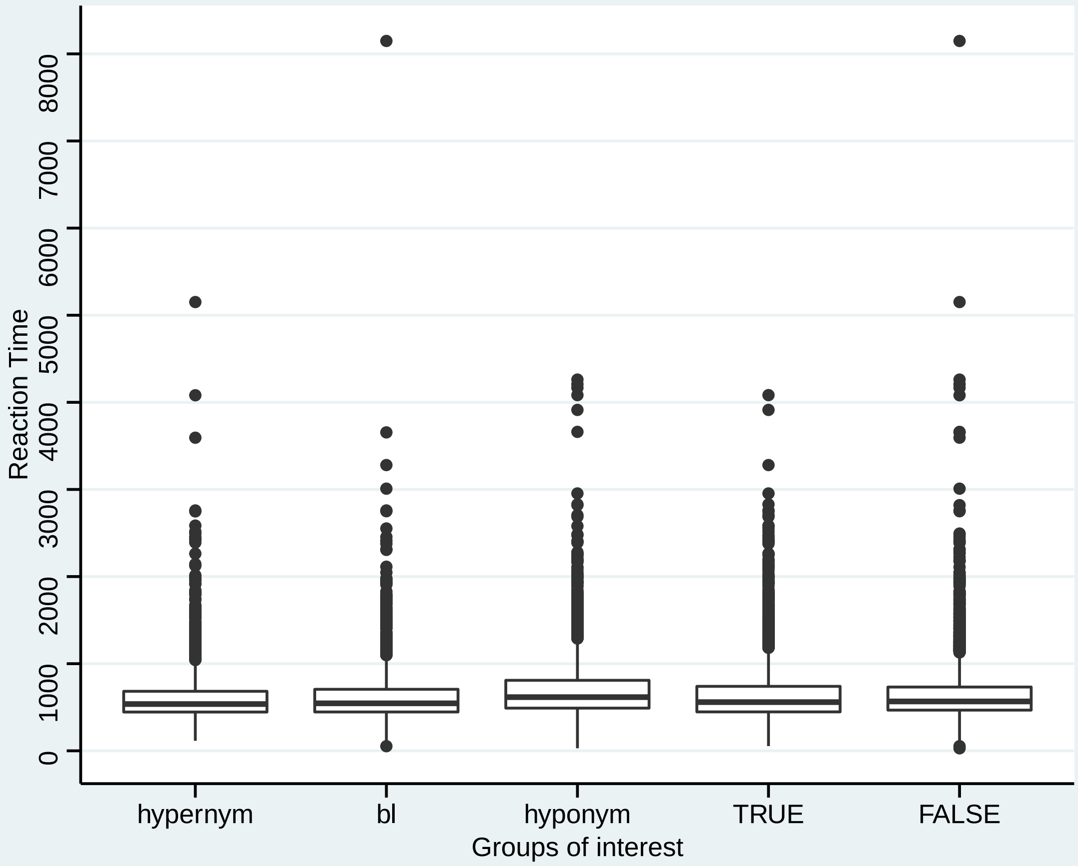


Fig.1: Basic reaction times overview.

Figure 2 is meant to give an overview of the amount of mistakes made by participants with regard to the main category branches. If the branches are balanced (same amount of categories in each group), then the columns in the plot should have the same height.

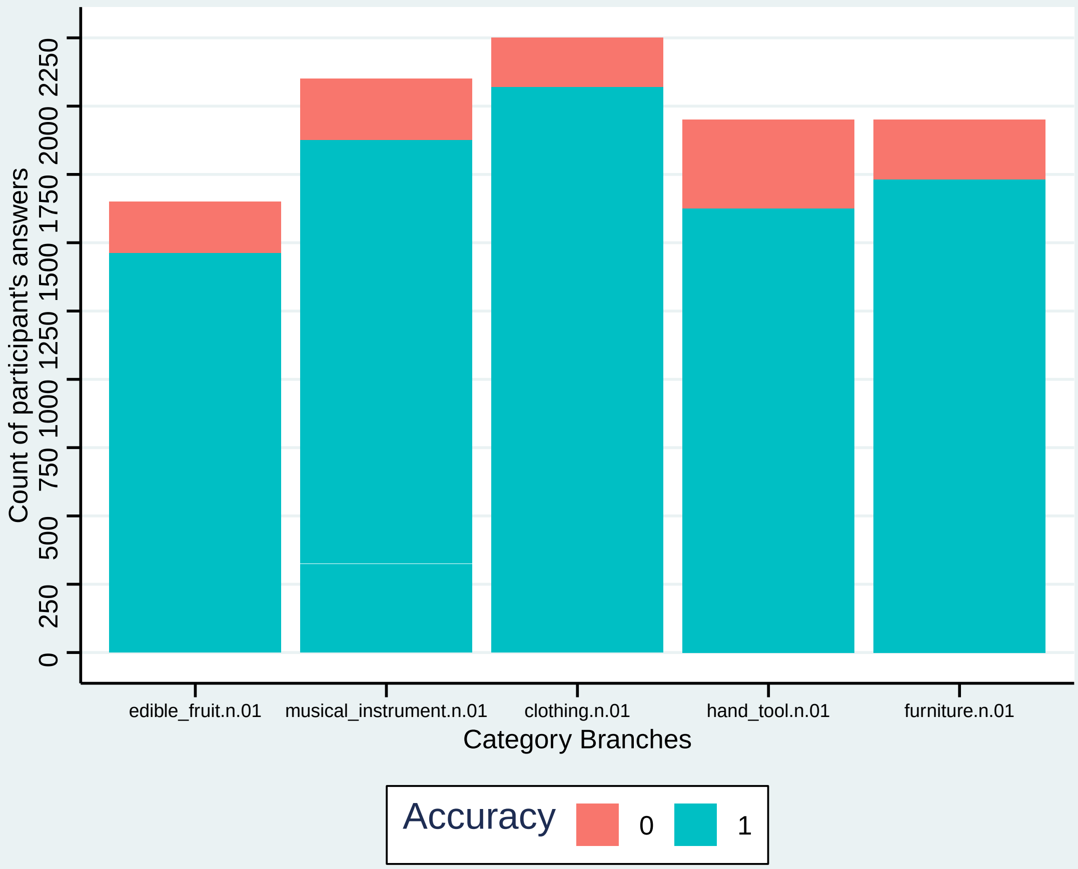
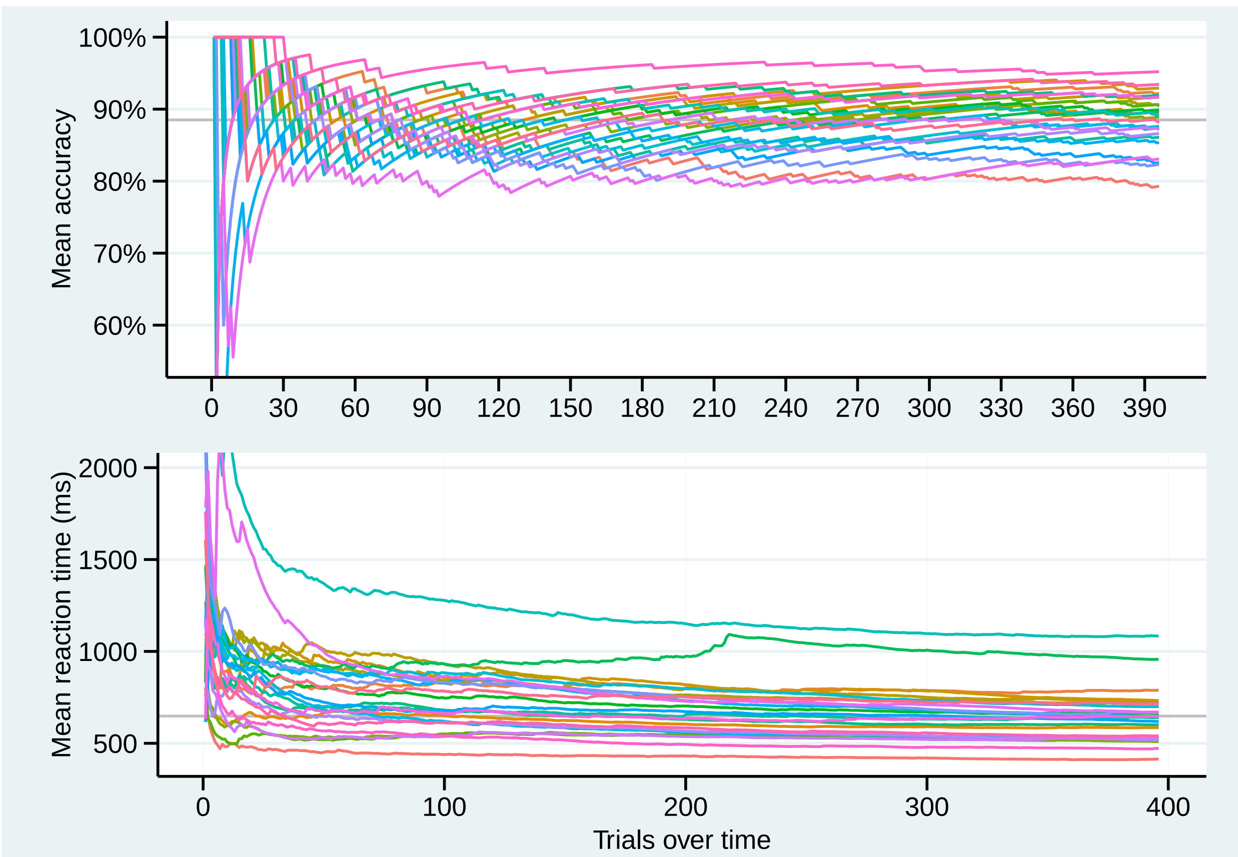


Fig.2. Proportions of correct to incorrect participant answers.

1.3 Do reaction times and accuracies improve over trials?

```
participant_paths <- exp_results[c("p_id", "accuracy", "react_time")]

acceptable_acc <- 0.85

#creation of df per participant
p_dfs <- hash()
for(i in 1:p_num){
  p_dfs[[paste("p",i, sep="")]] <- filter(participant_paths, p_id==i)
}

cumul_acc <- c(1:stim_num*0)
cumul_acc_mean <- c(1:stim_num*0)
cumul_rt <- c(1:stim_num*0)
cumul_rt_mean <- c(1:stim_num*0)
trials <- c(1:stim_num)

participant_paths$trials <- sort(c((1:rows)%%stim_num)+1)

for(i in 1:p_num){
  part = paste("p",i, sep="")
  part_df <- p_dfs[[part]]
  for(j in 1:stim_num){
    if(j>1){
      cumul_acc[j] <- cumul_acc[j-1] + part_df$accuracy[j]
      cumul_rt[j] <- cumul_rt[j-1] + part_df$react_time[j]
    } else {
      cumul_acc[j] <- part_df$accuracy[j]
      cumul_rt[j] <- part_df$react_time[j]
    }
    cumul_acc_mean[j] <- cumul_acc[j]/j
    cumul_rt_mean[j] <- cumul_rt[j]/j
  }
  p_dfs[[part]]$cumul_acc_mean <- cumul_acc_mean
  p_dfs[[part]]$cumul_rt_mean <- cumul_rt_mean

}

participant_acc_over_time <- ggplot(participant_paths, aes(x=trials, y=cumul_acc_mean))+
  geom_hline(yintercept = acc_mean, color="#c0c0c0")
participant_rt_over_time <- ggplot()+
  geom_hline(yintercept = rt_mean, color="#c0c0c0")

for(p in keys(p_dfs)){
  pp <- p_dfs[[p]]
  participant_acc_over_time <- participant_acc_over_time +
    geom_line(pp, mapping=aes(x=trials, y=cumul_acc_mean, color=p_id))
  participant_rt_over_time <- participant_rt_over_time +
    geom_line(pp, mapping=aes(x=trials, y=cumul_rt_mean, color=p_id))
}

participant_acc_over_time <- participant_acc_over_time +
```

```
  scale_y_continuous(name = "Mean accuracy", n.breaks = 5, labels = scales::percent) +
  scale_x_continuous(name="Number of trials",n.breaks=15)+
  theme(legend.position = 0, axis.title.x = element_blank(), axis.text.y = element_text(angle=0) ) +
  coord_cartesian(ylim=c(0.55,1))

participant_rt_over_time <- participant_rt_over_time +
  scale_y_continuous(name = "Mean reaction time (ms)", n.breaks = 5) +
  xlab("Trials over time") +
  theme(panel.grid.major.x = element_line(size = 0.001),legend.position = 0, axis.text.y = element_text
  coord_cartesian(ylim=c(400,2000))


grid.arrange(participant_acc_over_time, participant_rt_over_time)
```



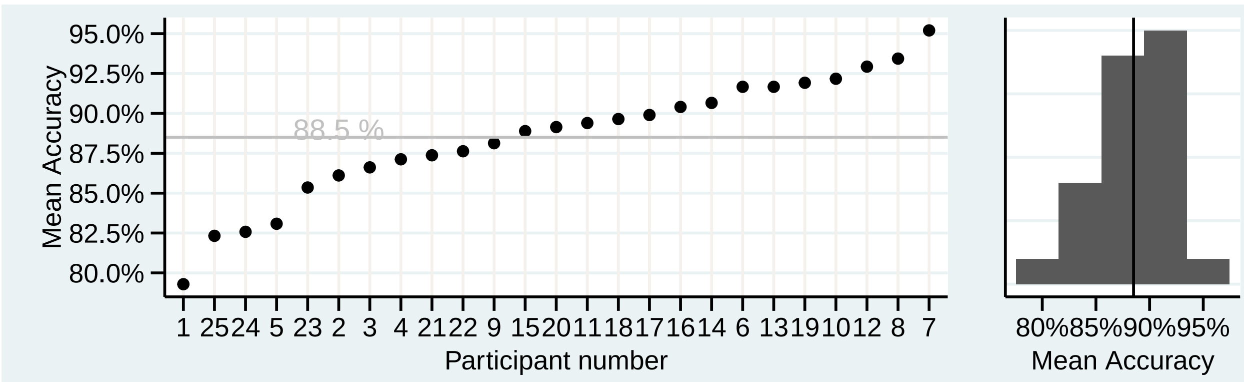
2 Mistake Analysis w.r.t. Participant

This section is concerned with mistakes in the experiment from the participant point of view. It helps to determine the performance of participants. It can be used to find participants that should not be rewarded for not doing the job seriously (if it was a condition in the job description) or to reward bonuses (if there are any) to high-performing participants. The per-participant mean reaction times and accuracies are shown.

```
##        p_id       mean_rt          mean_acc
##   1      : 1   Min.   : 414.1   Min.   :0.7929
```

```
## 2       : 1   1st Qu.: 534.2   1st Qu.:0.8662
## 3       : 1   Median : 644.3   Median :0.8914
## 4       : 1   Mean   : 648.0   Mean   :0.8851
## 5       : 1   3rd Qu.: 715.5   3rd Qu.:0.9167
## 6       : 1   Max.   :1083.5   Max.   :0.9520
## (Other):19
```

Figure 3 compares the mean accuracies (% on y axis) and reaction times (values in columns) of all participants. It only shows the number of the participant, their actual ID from the job must be looked up manually. Research shows that an accuracy of up to 85% is to be expected [1], anything below could be considered unserious. Look at the following evaluations to verify such a suspicion.



The following plot (Fig.4) shows box plots for participant's reaction times on stimuli. It is restricted to those participants that show a sub-standard accuracy (<85%). If reaction times are all extremely short, then the participant did not do the task appropriately. Generally, the subordinate reaction times should be slower than the rest. If the range of reaction times is great, then the participant might have struggled with the category names. Outliers above the 3rd quantile indicate that the participant got stuck on those stimuli / thought longer about them. Outliers below the 1st quantile are answers given too fast to have even registered the stimulus.

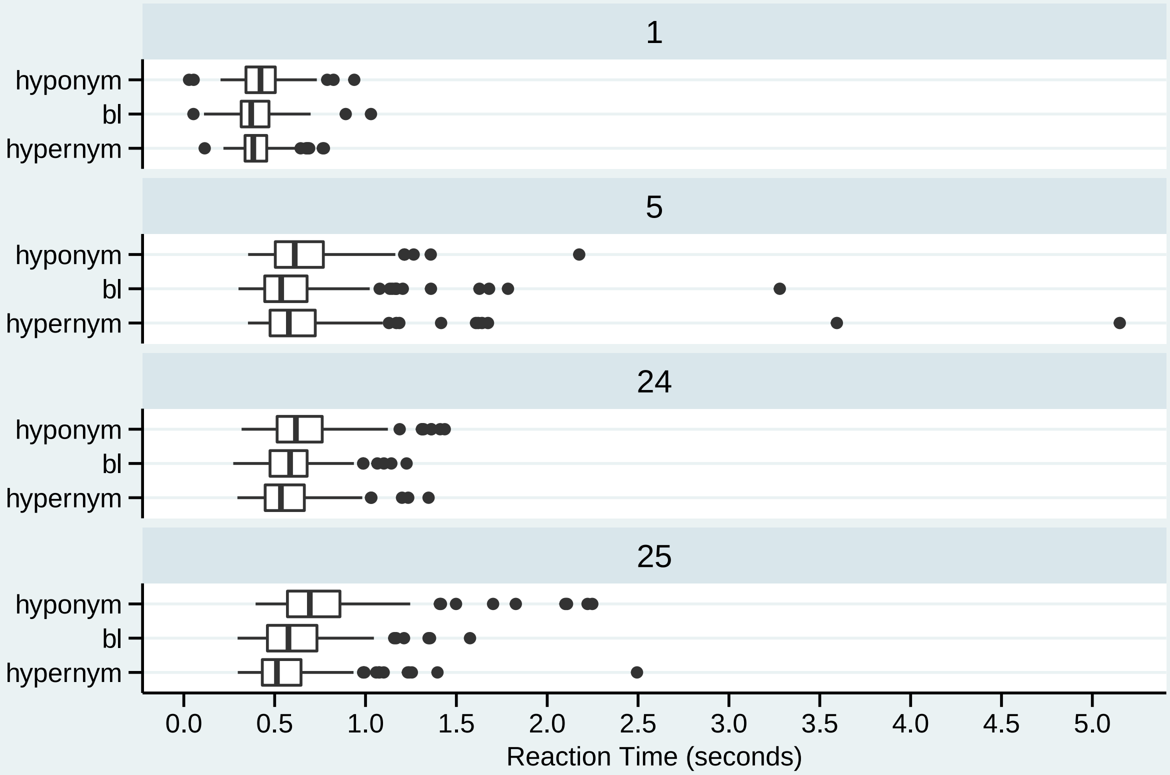
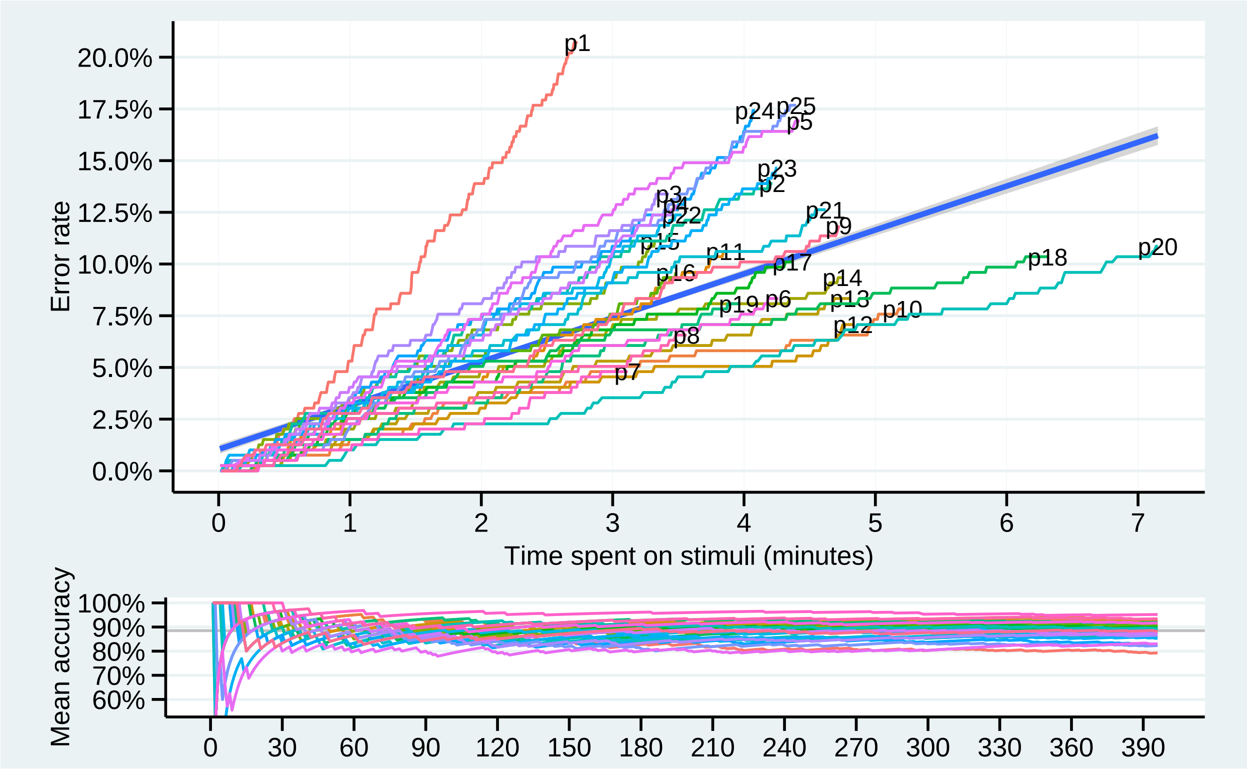


Fig.4. Box plots for reaction times of low accuracy participants.

The final plot (Fig.5) of this section highlights how participants individually experienced the experiment over time. Each path represents one participant's mistake count as time moves on. Time that passed equally fast for each participant (e.g. showing the stimulus, moving on to the next) is not considered. Only time that passed while the participant was able to answer is considered. Thus, this plot does not show moments where participants took a break during the experiment's break-time screens.

```
## 'geom_smooth()' using formula 'y ~ x'
```



These information should be enough to get a good idea why some participants did better or worse than others. It could also already indicate a possible issue with the experiment procedure or content. The next section will take a closer look at mistakes with respect to the stimuli and their inherent qualities.

3 Mistake Analysis w.r.t. Stimulus

In the following overview (Fig.6), you can see the percentages of mistakes to correct answers of all stimuli. If all participants have seen the stimuli in the same order, then the columns will be in order of how stimuli have been originally ordered.

The following sections try to shine light on what caused mistakes with rights to one specific component or aspect of the stimulus.

3.1 Stimuli

Figure 7 shows the stimuli (image + label) that received the most erroneous answers. The cutoff is made at 50%, when half the participants had a mistake.

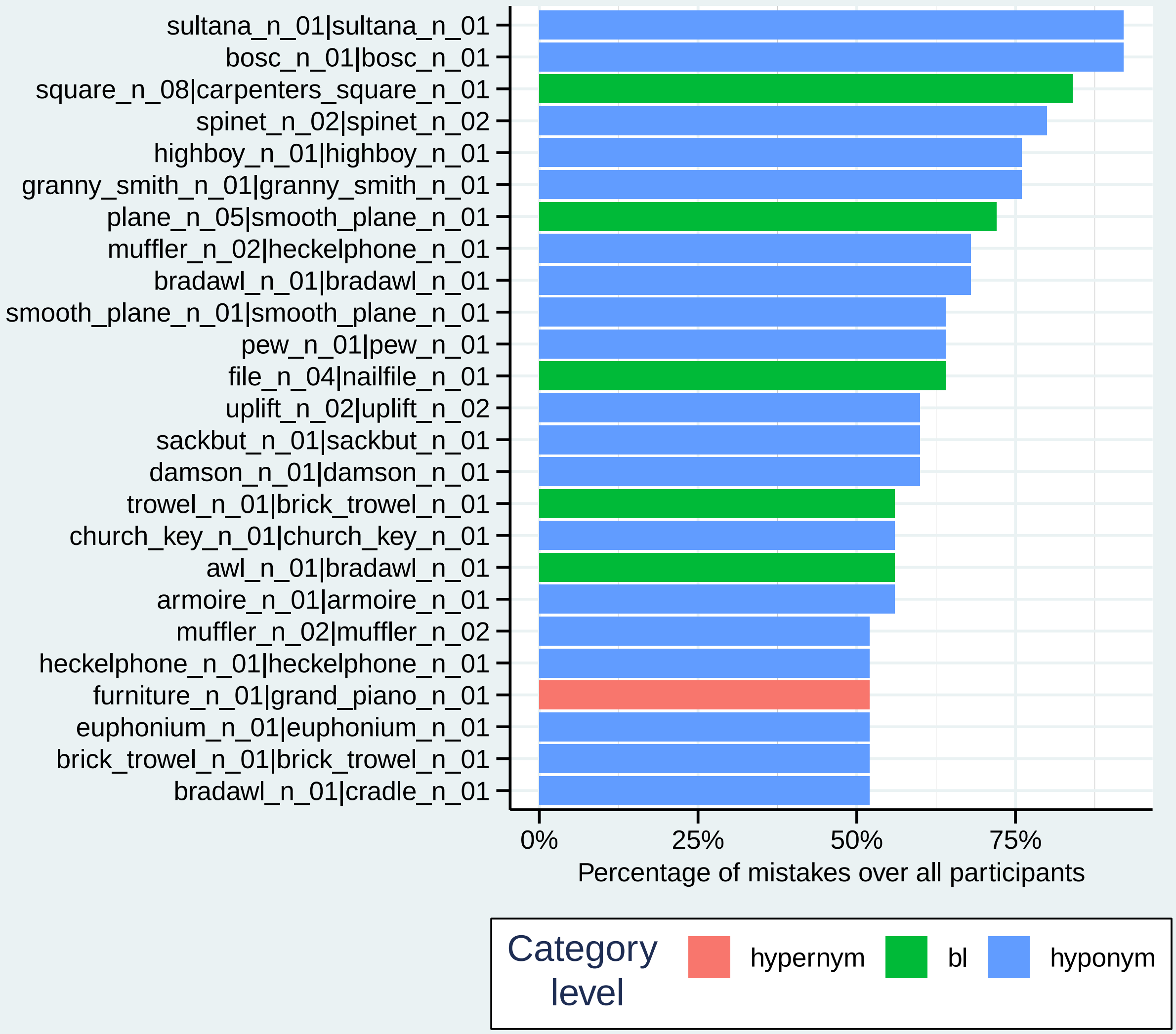


Fig.7. High error Stimuli.

3.2 Image

Figure 8 shows the images that received the most erroneous answers. The cutoff is made at 15%.

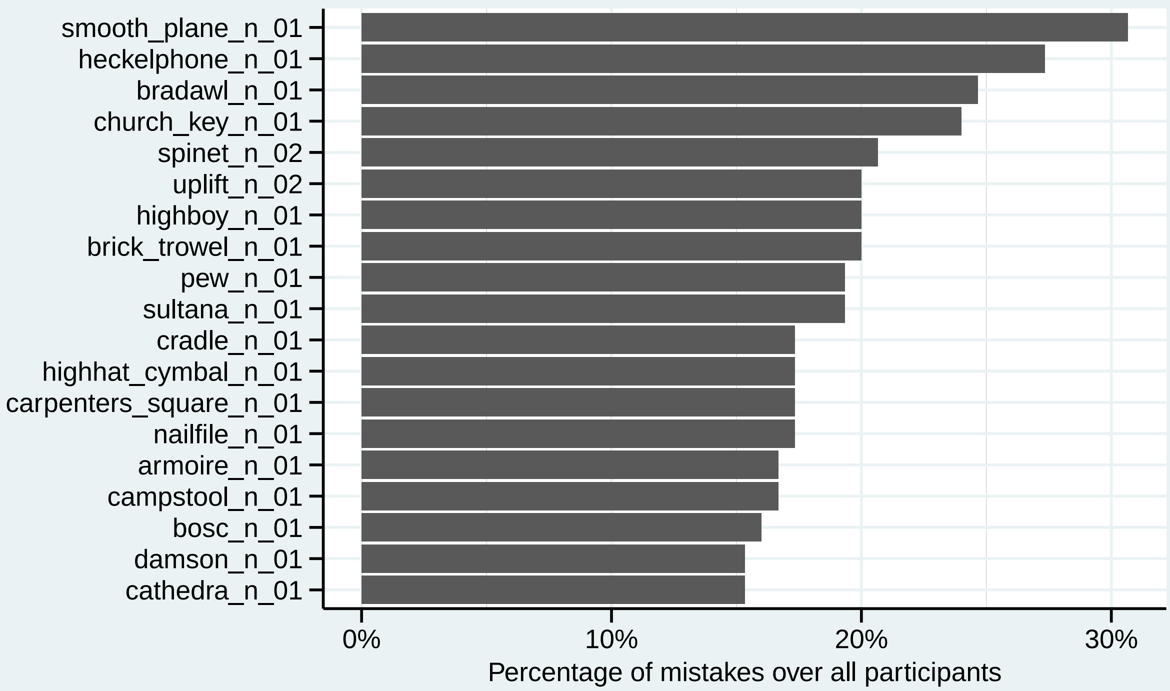


Fig.8. High error Images.

3.3 Category Name

Figure 9 shows the category names that received the most erroneous answers. The cutoff is made at 15%.

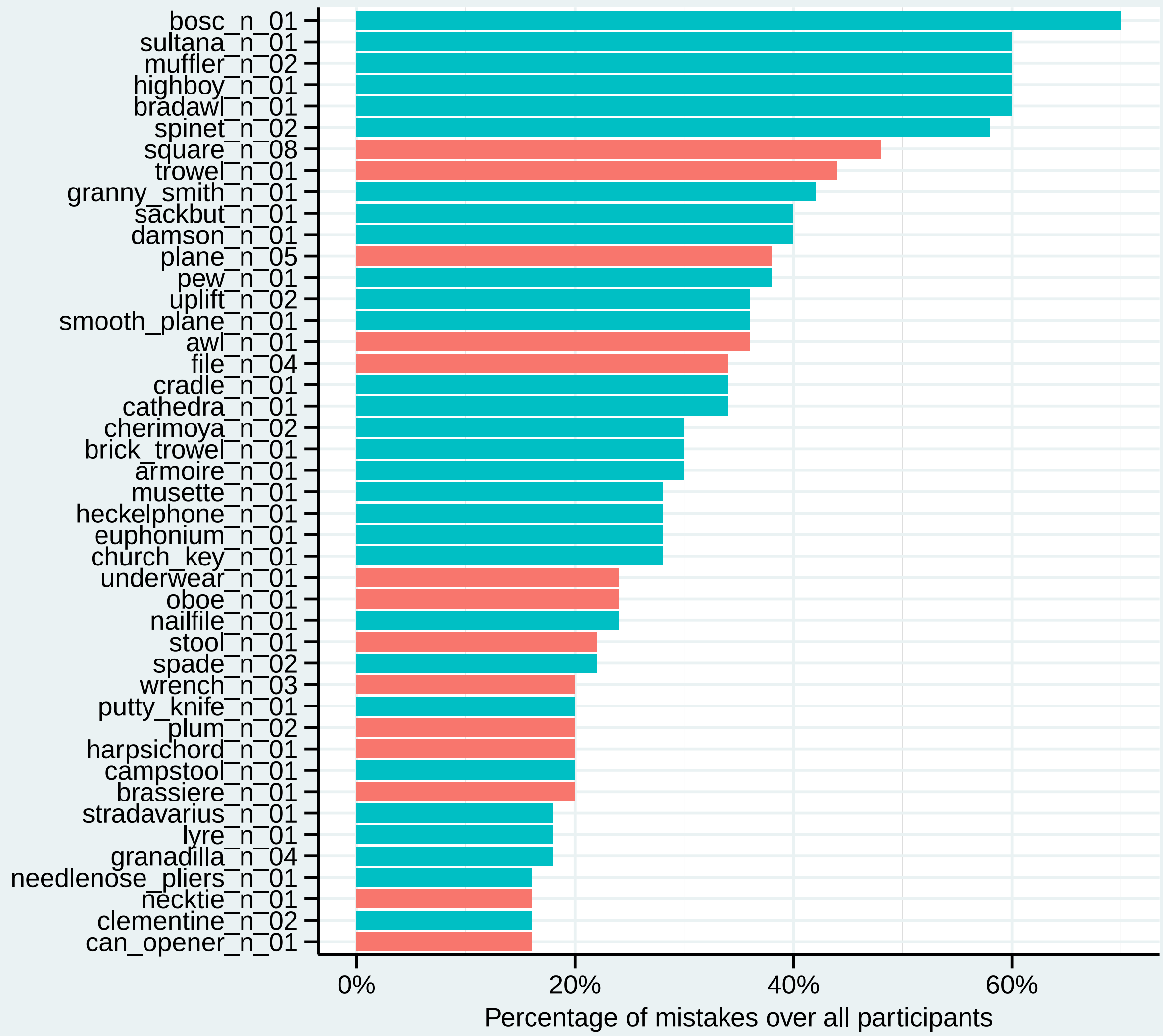


Fig.9. High error Category names. Colors highlight the category level.

3.4 Branch

Figure 10 shows the mistake percentage per branch. The columns are colored to highlight the share of each category level within the respective branches.

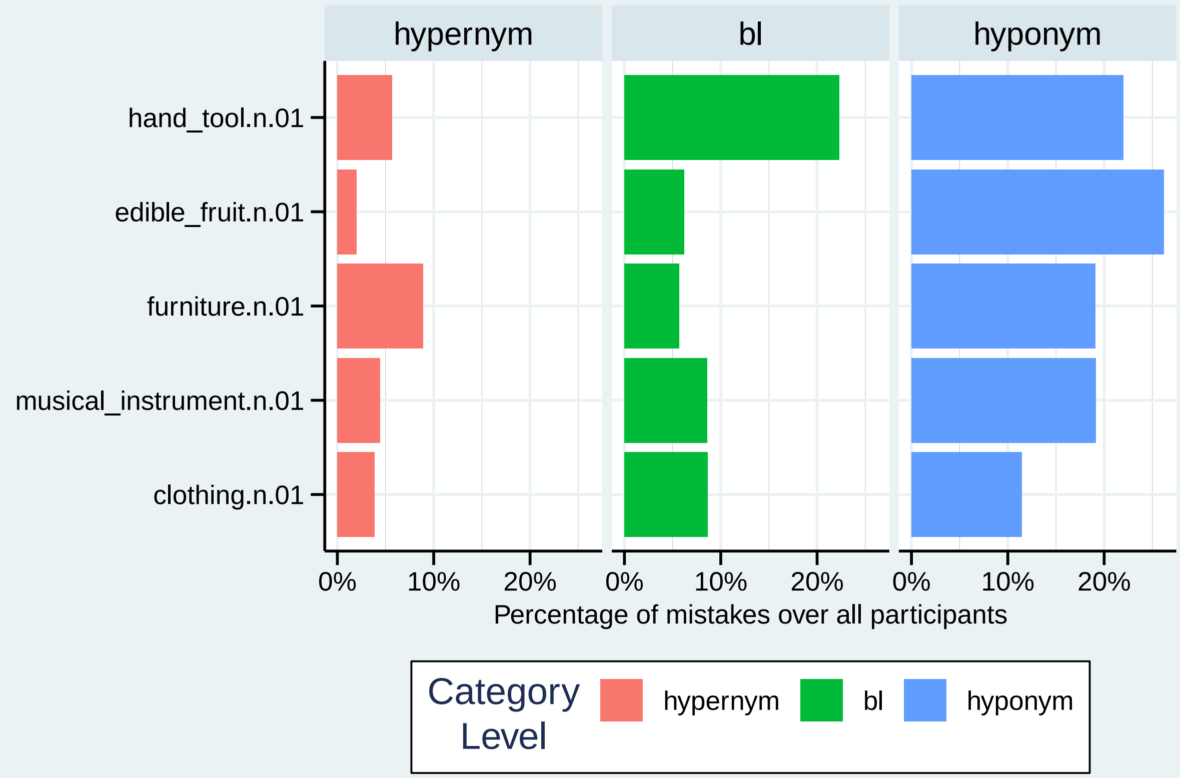


Fig.10. Error rates of individual branches.

3.5 Level

Figure 11 shows the mistake percentage per category level. The columns are colored to highlight the share of both stimulus types.

Fig.11. Error rates of category levels.

3.6 Stimulus Type

Figure 12 shows a pie chart of True/False stimulus type's correct and incorrect answer rates.

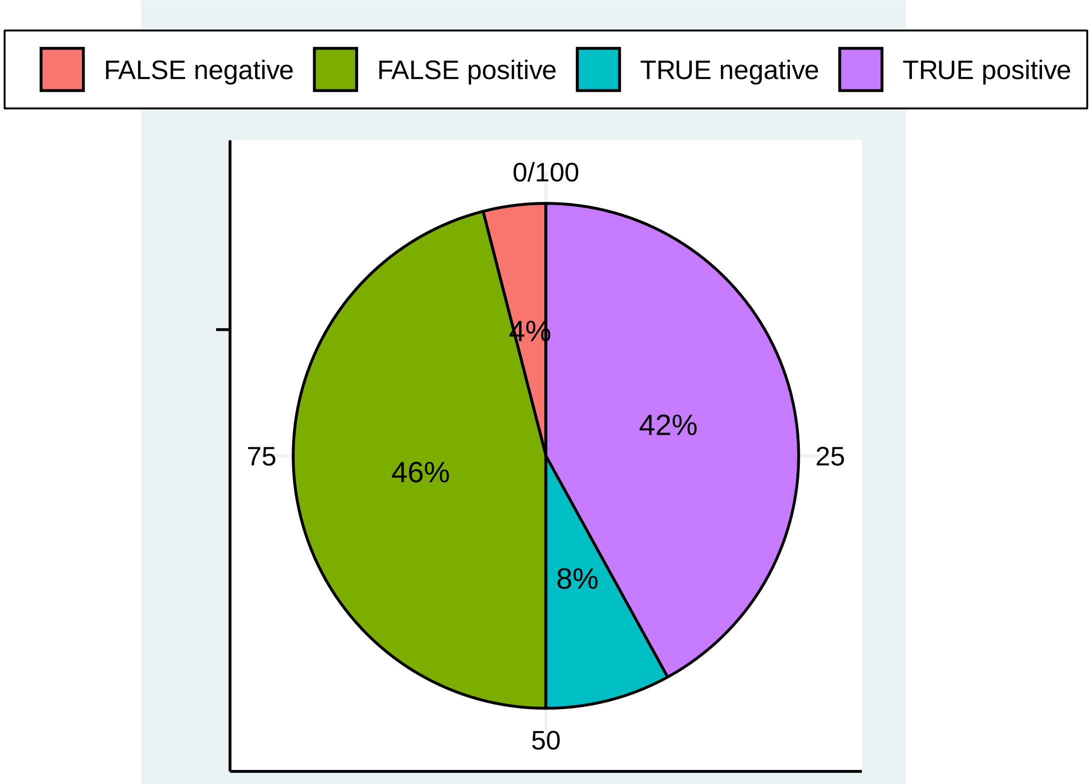


Fig.12. Error rates of stimulus types.

4 Component significance analysis in terms of error rates

This section calculates the one-way ANOVA and two-way ANOVA of each factor (stimulus components) to determine significant differences between groups. Then, they are pitted against each other in an AIC table to determine the best-fitting statistical model among the ANOVA models.

4.1 One-Way

```
aov_branch <- aov(err ~ branch, data=aov_data)
aov_lvl <- aov(err ~ lvl, data=aov_data)
aov_c_name <- aov(err ~ c_name, data=aov_data)
aov_img <- aov(err ~ img, data=aov_data)
aov_stim <- aov(err ~ stimulus, data=aov_data)
aov_stype <- aov(err ~ stim_type, data=aov_data)
```

4.1.1 Branch

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## branch        4  0.326 0.08162   2.904 0.0217 *
## Residuals   388 10.907 0.02811
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.1.2 Level

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## lvl           2  1.323  0.6616   26.04 2.44e-11 ***
## Residuals   390  9.910  0.0254
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.1.3 Category name

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## c_name      136  7.068 0.05197   3.194 6.17e-16 ***
## Residuals   256  4.165 0.01627
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.1.4 Image

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## img          65  1.589 0.02445   0.829  0.819
## Residuals   327  9.644 0.02949
```

4.1.5 Stimulus

```
##              Df Sum Sq Mean Sq
## stimulus    392  11.23 0.02866
```

4.1.6 Stimulus type

```
##              Df Sum Sq Mean Sq F value  Pr(>F)
## stim_type     1  0.701  0.7007   26.01 5.3e-07 ***
## Residuals   391 10.532  0.0269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.1.7 AIC

```
## Warning: package 'AICcmodavg' was built under R version 4.2.1
```

```
##
## Model selection based on AICc:
##
##              K    AICc Delta_AICc AICcWt Cum.Wt     LL
## Branch       6 -281.18        Inf    NaN    NaN 146.70
## Level        4 -322.95        Inf    NaN    NaN 165.53
## Cat. Name  138 -244.67        Inf    NaN    NaN 335.85
## Image       67 -179.72        Inf    NaN    NaN 170.88
## Stim. Type   3 -301.06        Inf    NaN    NaN 153.56
## Stimulus   394    -Inf        NaN    NaN    NaN    Inf
```

4.2 Two-way

In the first iteration of this evaluation, the models with the most significant differences within their groups were those for stimulus type, category level, category name and branch (lower significance). The AIC table determined, in this order, that the ANOVA models for category level, stimulus type and branch showed the best fit. The following subsections compare ANOVA of pairs and triples of these components in another AIC table.

```
aov_lvl_type <- aov(err ~ lvl + stim_type, data=aov_data)
aov_lvl_branch <- aov(err ~ lvl + branch, data=aov_data)
aov_type_branch <- aov(err ~ stim_type + branch, data=aov_data)
aov_lvl_type_branch <- aov(err ~ lvl + stim_type + branch, data=aov_data)
```

4.2.1 Level + Type

```
##                Df Sum Sq Mean Sq F value   Pr(>F)
## lvl             2  1.323  0.6616   28.00 4.38e-12 ***
## stim_type       1  0.717  0.7173   30.36 6.56e-08 ***
## Residuals     389  9.193  0.0236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.2 Level + Branch

```
##                Df Sum Sq Mean Sq F value   Pr(>F)
## lvl             2  1.323  0.6616  26.642 1.45e-11 ***
## branch          4  0.325  0.0813   3.275   0.0117 *
## Residuals     386  9.585  0.0248
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.3 Type + Branch

```
##                Df Sum Sq Mean Sq F value   Pr(>F)
## stim_type       1  0.701  0.7007  26.567 4.07e-07 ***
## branch          4  0.326  0.0814   3.087    0.016 *
## Residuals     387 10.207  0.0264
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.4 Level + Type + Branch

```
##                Df Sum Sq Mean Sq F value   Pr(>F)
## lvl             2  1.323  0.6616  28.721 2.37e-12 ***
## stim_type       1  0.717  0.7173  31.142 4.53e-08 ***
## branch          4  0.324  0.0811   3.522  0.00772 **
## Residuals     385  8.868  0.0230
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.5 AIC

```
##
## Model selection based on AICc:
##
##                        K    AICc Delta_AICc AICcWt Cum.Wt     LL
## Level & Type & Branch 9 -356.24       0.00   0.95   0.95 187.35
## Level & Type          5 -350.43       5.81   0.05   1.00 180.29
## Level & Branch        8 -327.80      28.44   0.00   1.00 172.09
## Type & Branch         7 -305.17      51.07   0.00   1.00 159.73
```

A Tukey test can be used to measure the differences between group-member pairings.

```
tukey <- TukeyHSD(aov_lvl_branch)

print(tukey)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = err ~ lvl + branch, data = aov_data)
##
## $lvl
##                         diff         lwr        upr     p adj
## bl-hypernym       0.05294574 0.007045087 0.09884639 0.0189998
## hyponym-hypernym 0.14082452 0.094923875 0.18672517 0.0000000
## hyponym-bl       0.08787879 0.042242698 0.13351488 0.0000234
##
## $branch
##                                                  diff          lwr        upr
## musical_instrument.n.01-edible_fruit.n.01 -0.007612805 -0.078839619 0.06361401
## clothing.n.01-edible_fruit.n.01           -0.034597029 -0.104752295 0.03555824
## hand_tool.n.01-edible_fruit.n.01           0.052667661 -0.019777951 0.12511327
## furniture.n.01-edible_fruit.n.01          -0.002237762 -0.074468720 0.06999320
## clothing.n.01-musical_instrument.n.01     -0.026984224 -0.092885158 0.03891671
## hand_tool.n.01-musical_instrument.n.01     0.060280466 -0.008053555 0.12861449
## furniture.n.01-musical_instrument.n.01     0.005375043 -0.062731367 0.07348145
## hand_tool.n.01-clothing.n.01               0.087264690  0.020048316 0.15448106
## furniture.n.01-clothing.n.01               0.032359266 -0.034625698 0.09934423
## furniture.n.01-hand_tool.n.01             -0.054905423 -0.124285469 0.01447462
##                                               p adj
## musical_instrument.n.01-edible_fruit.n.01 0.9983894
## clothing.n.01-edible_fruit.n.01           0.6590137
## hand_tool.n.01-edible_fruit.n.01          0.2715710
## furniture.n.01-edible_fruit.n.01          0.9999882
## clothing.n.01-musical_instrument.n.01     0.7947523
## hand_tool.n.01-musical_instrument.n.01    0.1127500
## furniture.n.01-musical_instrument.n.01    0.9995124
## hand_tool.n.01-clothing.n.01              0.0038262
## furniture.n.01-clothing.n.01              0.6764229
## furniture.n.01-hand_tool.n.01             0.1937813
```

```r
plot(tukey, sub="Fig.13. Tukey Confidence intervals for level and branch.")
```

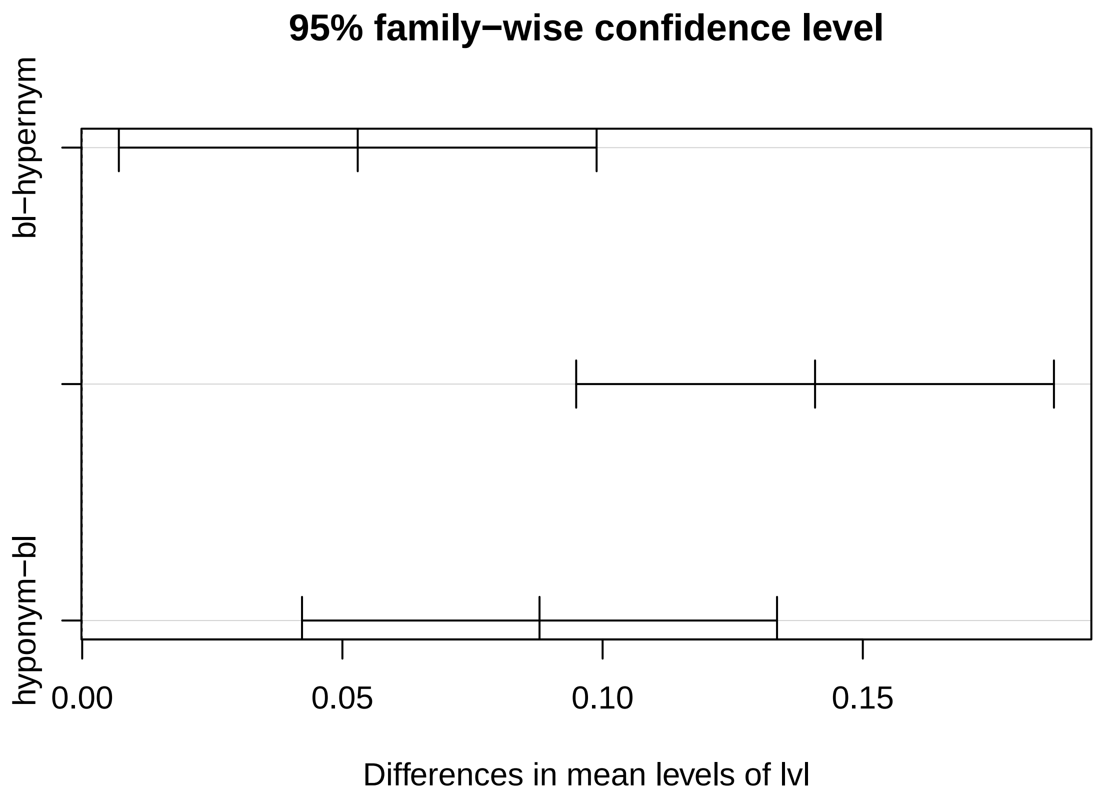


Fig.13. Tukey Confidence intervals for level and branch.

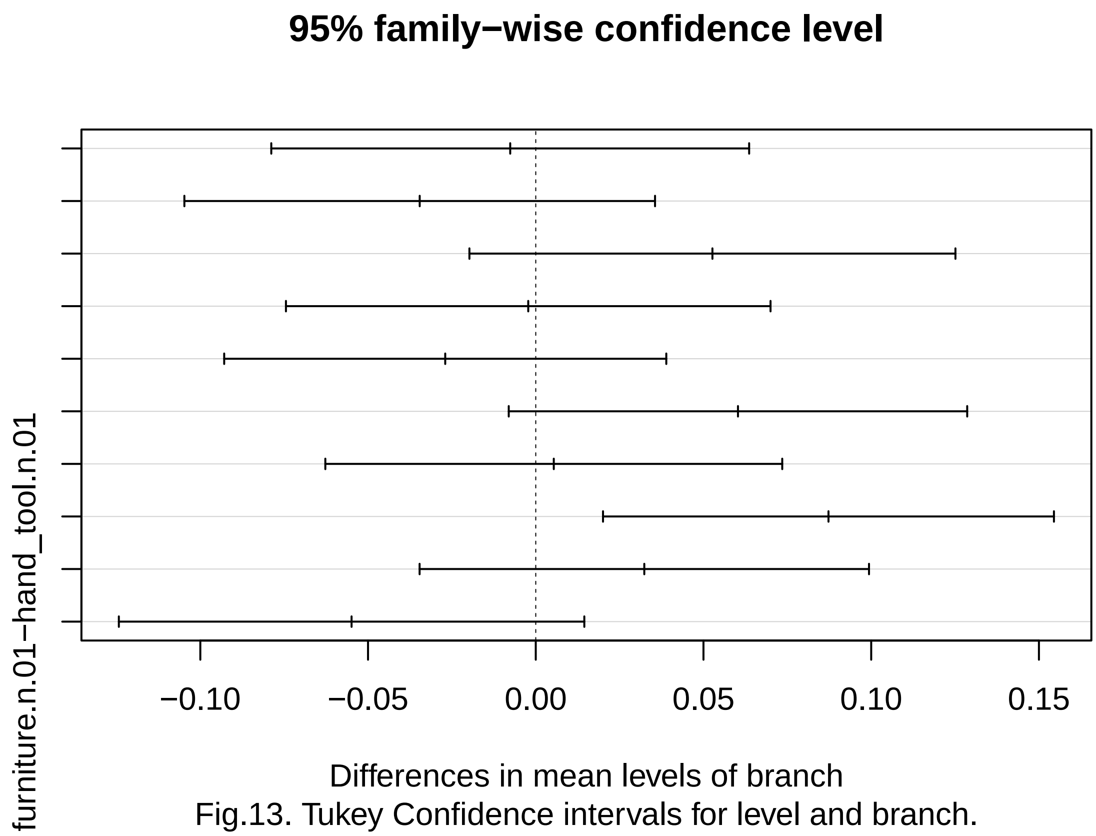


Fig.13. Tukey Confidence intervals for level and branch.

5 Analysis of reaction times

This section is concerned with the reaction times recorded during the experiment. The per-participant reaction times have already been illustrated in Fig.3.

5.1 Overview

To give an overview, Fig.14 illustrates the distribution of stimuli over the range of reaction times. The columns for the three distinct category levels sit next to each other to facilitate the identification of possible differences. Fig.15 shows box plots for reaction times within the three category levels, one plot per stimulus type. Fig.15 is also a visual representation of table 1, which is the same table Rosch[2] created.

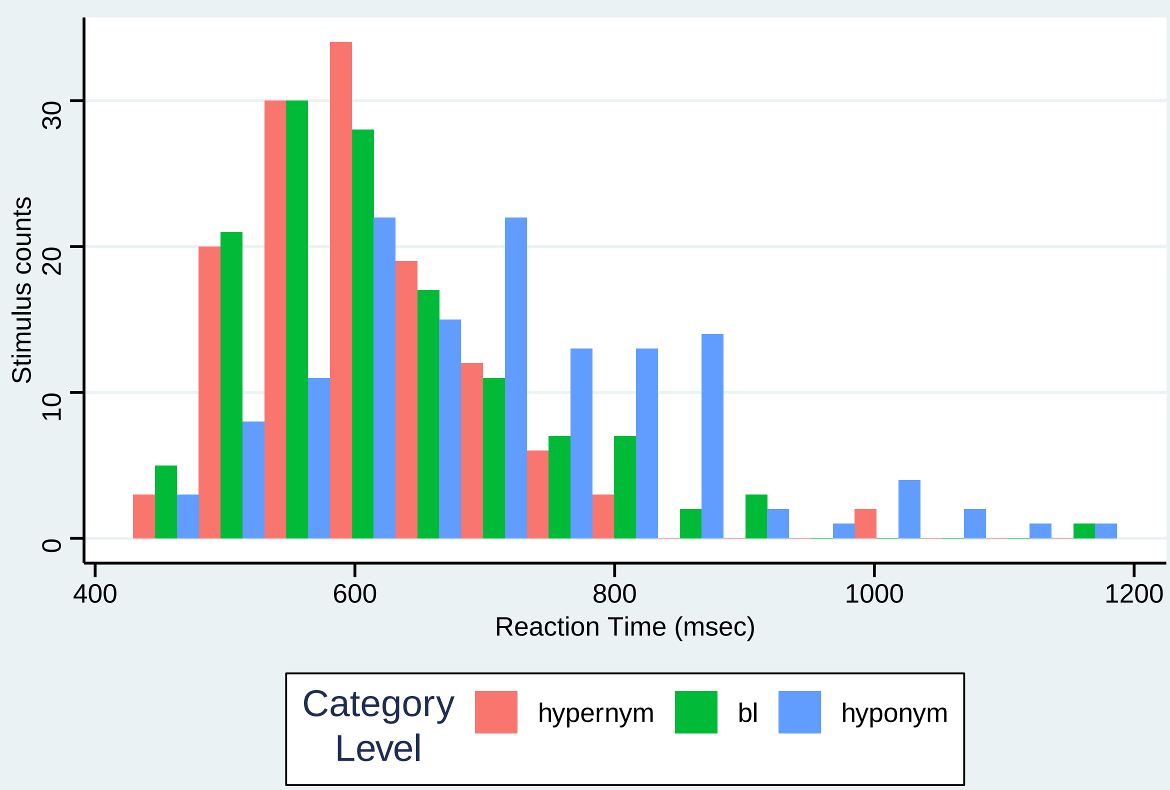


Fig.14. Distribution of stimuli frequencies.

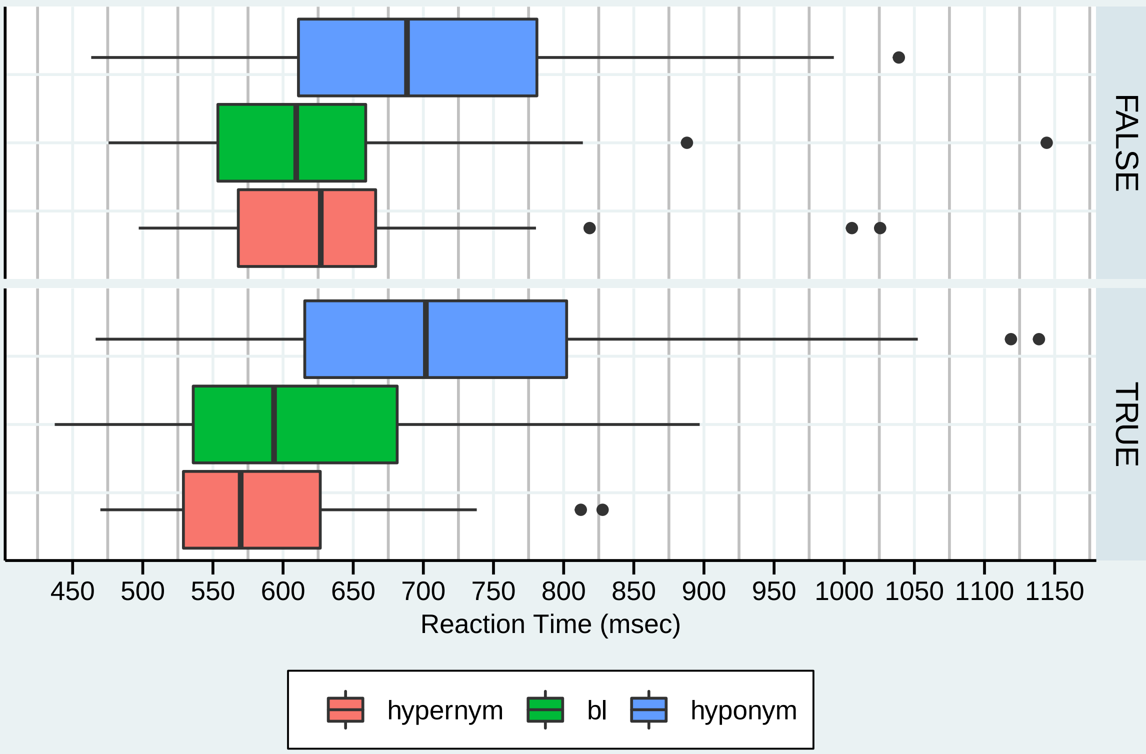


Fig.15 Boxplot representation of reaction time means over all stimuli

```
##   Superordinate Basic Level Subordinate
## F      635.6127    623.7824    696.9606
## T      585.9824    621.8145    723.8018
```

Table 1.: Matrix showing the mean reaction times at different category levels and stimuli types.

5.2 Analysis

This section applies the two-way ANOVA model on the measurements. The category level (between-subject fixed effect) and the category names (random variable) are used as independent variables.

5.2.1 True type stimuli

Anova summary:

```
summary.aov(roschAOV_true)
```

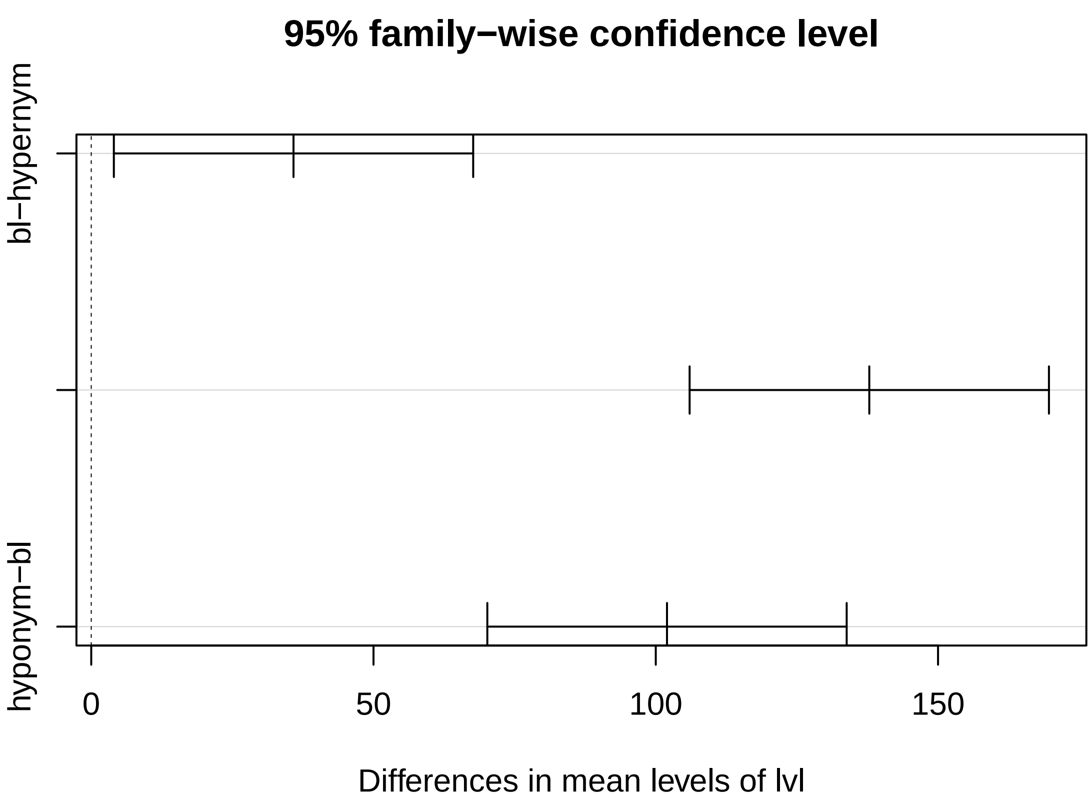
```
##              Df  Sum Sq Mean Sq F value   Pr(>F)    
## lvl           2  674950  337475  58.261 7.09e-15 ***
## c_name      134 2554301   19062   3.291 3.98e-07 ***
## Residuals    61  353341    5792                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tukey test:

```
tukey_r_true
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = rt_mean ~ lvl + c_name, data = r_true)
##
## $lvl
##                          diff       lwr       upr     p adj
## bl-hypernym          35.83212   4.00569  67.65855 0.0237145
## hyponym-hypernym 137.81939 105.99296 169.64583 0.0000000
## hyponym-bl         101.98727  70.16084 133.81370 0.0000000
```

```
plot(tukey_r_true)
```



5.2.2 False type stimuli

Anova summary:

```
summary.aov(roschAOV_false)
```

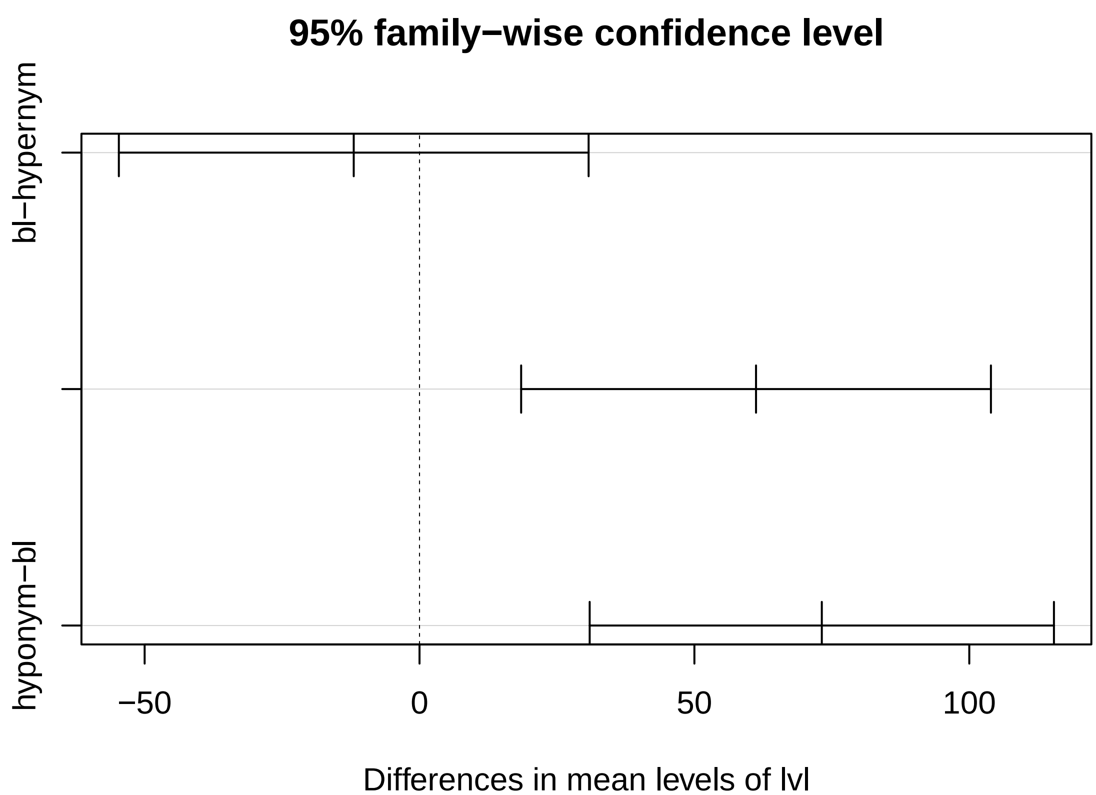
```
##              Df  Sum Sq Mean Sq F value   Pr(>F)
## lvl           2  202575  101288   9.959 0.000191 ***
## c_name      134 1777139   13262   1.304 0.127242
## Residuals    58  589892   10171
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tukey test:

```
tukey_r_false
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = rt_mean ~ lvl + c_name, data = r_false)
##
## $lvl
##                       diff       lwr       upr     p adj
## bl-hypernym      -11.96488 -54.69129  30.76153 0.7797189
## hyponym-hypernym  61.21330  18.48689 103.93972 0.0030132
## hyponym-bl        73.17818  30.95151 115.40485 0.0003015
```

```
plot(tukey_r_false)
```



6 B.L. Determination

Ultimately, the experiment is conducted to determine the basic level category name from a selection of three candidates. These triples form a path through one of the WordNet branches. We only consider a single element in this path basic level. Basic levelness can only be infered from reaction times of the true type stimuli. The false type stimuli reactions say more about the image than about the category names.

6.1 Simple rating

In this section, we will see a simple implementation to declare which category name from the triple is the basic level (super- and subordinates can be inferred). For each triple, the category name with the fastest mean reaction time is chosen as basic level. Below, you can see the top nine results of the evaluation. Thereafter follows a confusion matrix of expected and predicted levels.

```
## # A tibble: 9 x 3
## # Groups:   lvl, c_name [7]
##   c_name          lvl   proj_lvl
##   <chr>           <fct> <fct>
## 1 clothing_n_01   super bl
## 2 sweater_n_01    bl    sub
## 3 turtleneck_n_01 sub   sub
## 4 clothing_n_01   super bl
## 5 shirt_n_01      bl    sub
## 6 tank_top_n_01   sub   sub
## 7 clothing_n_01   super super
## 8 necktie_n_01    bl    bl
## 9 bow_tie_n_01    sub   sub
```

```
## Warning: package 'caret' was built under R version 4.2.1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bl sub super
##      bl    28   4    34
##      sub   34  62     0
##      super  4   0    32
##
## Overall Statistics
##
##                Accuracy : 0.6162
##                  95% CI : (0.5446, 0.6842)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 3.927e-16
##
##                   Kappa : 0.4242
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: bl Class: sub Class: super
```

```
## Sensitivity            0.4242     0.9394       0.4848
## Specificity            0.7121     0.7424       0.9697
## Pos Pred Value         0.4242     0.6458       0.8889
## Neg Pred Value         0.7121     0.9608       0.7901
## Prevalence             0.3333     0.3333       0.3333
## Detection Rate         0.1414     0.3131       0.1616
## Detection Prevalence   0.3333     0.4848       0.1818
## Balanced Accuracy      0.5682     0.8409       0.7273
```

6.2 Advanced rating

The basic level is not an inherent quality of a word itself, it is a quality experienced by humans using the words. Everyone experiences language slightly differently, wherefore basic-levelness should rather be expressed as a probability. Then it can be said, this word has a higher probability of being at the basic level than its hypernym or hyponym.

We create a training and testing dataset for sklearn in this section.

```
exp_results_T <- exp_results %>%
  filter(stim_type == TRUE) %>%
  select(c('branch', 'lvl', "c_name", 'img', 'accuracy', 'react_time'))

supV <- data.frame(super = c(1), bl = c(0), sub =c(0))
blV <- data.frame(super = c(0), bl = c(1), sub =c(0))
subV <- data.frame(super = c(0), bl = c(0), sub =c(1))

df_for_adv <- data.frame()

for (imag in all_img){
  triple <- exp_results_T %>%
    filter(img == imag) %>%
    mutate(lvl = ifelse(lvl=='hyponym', 'sub', ifelse(lvl=='hypernym', 'super', 'bl')))

  superlvl <- triple[which(triple$lvl == "super"),]
  superlvl <- cbind(superlvl, supV)
  blvl <- triple[which(triple$lvl == "bl"),]
  blvl <- cbind(blvl, blV)
  sublvl <- triple[which(triple$lvl == "sub"),]
  sublvl <- cbind(sublvl, subV)


  df_for_adv <- rbind(df_for_adv,superlvl)
  df_for_adv <- rbind(df_for_adv,blvl)
  df_for_adv <- rbind(df_for_adv,sublvl)
}

write_csv(df_for_adv, "training_ds.csv")

df_for_test <- exp_results %>%
  filter(stim_type == TRUE) %>%
  select(c('branch', 'lvl', "c_name", 'img', 'accuracy', 'react_time')) %>%
  group_by(branch, lvl, c_name, img) %>%
  summarize(mean_acc=mean(accuracy), mean_rt = mean(react_time))
```

```
## 'summarise()' has grouped output by 'branch', 'lvl', 'c_name'. You can override
## using the '.groups' argument.
```

```
write_csv(df_for_test, "testing_ds.csv")
```

This section uses a linear regression model to predict the basic level. (With little success.) The idea is to collect all instances that belong to one category (total = n. participants * levels), train the model, then predict what the level must be, using the mean reaction time of each category level. Each category name is given a value before training. A 0 for superordinate, a negative value for the subordinate and positive for basic level. After the prediction, the category name with the highest value must be basic level, the remaining levels can be inferred from the branch hierarchy.

```
new_exp_results <- exp_results %>%
  filter(stim_type == TRUE) %>%
  mutate(lvl = ifelse(lvl=='hyponym', 'sub', ifelse(lvl=='hypernym', 'super', 'bl')))

# Seeding the algorithm
new_exp_results$num_lvl <- sapply(new_exp_results$lvl, FUN = function(x){ifelse(x == "sub", 0, ifelse(x


pred_exp_results <- data.frame()



for(imag in all_img){
  triple <- new_exp_results %>%
    filter(img == imag)

  bl_model <- lm(num_lvl ~ react_time, data = triple)

  sum_triple <- triple %>%
  group_by(c_name, img, lvl) %>%
  summarize(react_time = mean(react_time), accuracy = mean(accuracy))

  bl_prediction <- predict(bl_model, newdata = sum_triple, interval = 'prediction')
  sum_triple <- cbind(sum_triple, bl_prediction)

  pred_exp_results <- rbind(pred_exp_results, sum_triple)

}
```

```
df_rated_adv <- data.frame()

for (imag in all_img){
  triple <- pred_exp_results %>%
    filter(img == imag)

  blvl <- triple[which(triple$fit == max(triple$fit)),]
  triple <- triple[which(triple$fit != max(triple$fit)),]
  blvl$pred_lvl <- 'bl'

  if(blvl$lvl == 'super'){
    triple$pred_lvl <- 'sub'
```

```r
    df_rated_adv <- rbind(df_rated_adv, blvl)
    df_rated_adv <- rbind(df_rated_adv, triple)

  } else if(blvl$lvl == 'bl'){
    sublvl <- triple[which(triple$lvl == "sub"),]
    sublvl$pred_lvl <- "sub"
    suplvl <- triple[which(triple$lvl == "super"),]
    suplvl$pred_lvl <- "super"
    df_rated_adv <- rbind(df_rated_adv, suplvl)
    df_rated_adv <- rbind(df_rated_adv, blvl)
    df_rated_adv <- rbind(df_rated_adv, sublvl)
  } else{
    triple$pred_lvl <- 'super'
    df_rated_adv <- rbind(df_rated_adv, blvl)
    df_rated_adv <- rbind(df_rated_adv, triple)
  }

}

df_rated_adv$lvl <- as.factor(df_rated_adv$lvl)
df_rated_adv$pred_lvl <- as.factor(df_rated_adv$pred_lvl)

head(df_rated_adv[c("c_name", "lvl", "pred_lvl")], n = 9)
```

```
## # A tibble: 9 x 3
## # Groups:   c_name [7]
##   c_name          lvl   pred_lvl
##   <chr>           <fct> <fct>
## 1 clothing_n_01   super bl
## 2 sweater_n_01    bl    sub
## 3 turtleneck_n_01 sub   sub
## 4 clothing_n_01   super bl
## 5 shirt_n_01      bl    sub
## 6 tank_top_n_01   sub   sub
## 7 clothing_n_01   super super
## 8 necktie_n_01    bl    bl
## 9 bow_tie_n_01    sub   sub
```

```r
con_mat_bl_oldRating <- confusionMatrix(data=df_rated_adv$pred_lvl,
                                        reference = df_rated_adv$lvl)
con_mat_bl_oldRating
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bl sub super
##      bl    43   0    23
##      sub   23  66     0
##      super  0   0    43
##
## Overall Statistics
##
```

```
##                Accuracy : 0.7677
##                  95% CI : (0.7025, 0.8246)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6515
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: bl Class: sub Class: super
## Sensitivity             0.6515     1.0000       0.6515
## Specificity             0.8258     0.8258       1.0000
## Pos Pred Value          0.6515     0.7416       1.0000
## Neg Pred Value          0.8258     1.0000       0.8516
## Prevalence              0.3333     0.3333       0.3333
## Detection Rate          0.2172     0.3333       0.2172
## Detection Prevalence    0.3333     0.4495       0.2172
## Balanced Accuracy       0.7386     0.9129       0.8258
```

```
# compare with simple projection
rating_comp_df1 <- df_rated_simple %>%
  mutate(stim = paste(c_name,img, sep=":")) %>%
  group_by(stim, lvl, proj_lvl) %>%
  select(stim, lvl, proj_lvl)

rating_comp_df2 <- df_rated_adv %>%
  mutate(stim = paste(c_name,img, sep=":")) %>%
  group_by(stim, lvl, pred_lvl) %>%
  select(stim, lvl, pred_lvl)

rating_comp_df <- merge(rating_comp_df1, rating_comp_df2)

con_mat_bl_simRating <- confusionMatrix(data=rating_comp_df$pred_lvl, reference = rating_comp_df$proj_l
con_mat_bl_simRating
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bl sub super
##      bl    49  13     4
##      sub    4  83     2
##      super 13   0    30
##
## Overall Statistics
##
##                Accuracy : 0.8182
##                  95% CI : (0.7573, 0.8693)
##     No Information Rate : 0.4848
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7121
##
```

```
##  Mcnemar's Test P-Value : 0.009182
##
## Statistics by Class:
##
##                      Class: bl Class: sub Class: super
## Sensitivity             0.7424     0.8646       0.8333
## Specificity             0.8712     0.9412       0.9198
## Pos Pred Value          0.7424     0.9326       0.6977
## Neg Pred Value          0.8712     0.8807       0.9613
## Prevalence              0.3333     0.4848       0.1818
## Detection Rate          0.2475     0.4192       0.1515
## Detection Prevalence    0.3333     0.4495       0.2172
## Balanced Accuracy       0.8068     0.9029       0.8765
```

6.3 Probabilistic rating with error

Stimuli that gathered many True negative responses, but few False positives, might give insight into basic levelness. The participant knew the object, which is why they could correctly determine a label as wrong, but they made many mistakes when confronted with the lower level category name. It could mean that they simply did not know the word, which would considerably lower the probability of it being basic level.

7 References

[1] Gagné, N., & Franzen, L., 2021, https://doi.org/10.31234/osf.io/nt67j

[2] E. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-Braem, “Basic objects in natural categories,” Cognitive Psychology, vol. 8, no. 3, pp. 382–439, 1976.