Preprocessing

Eva

4/3/2020

Contents

clean WS, set WD

```
rm(list = ls());
```

Set your local working directory. This should be (and is assumed to be in the rest of the code) the highest point in your local folder:

```
localGitDir <- 'C:/Users/eva_v/Documents/GitHub/leverhulmeNDL'
#setwd(localGitDir);
```

```r
fribbleSet <- read.csv(paste(localGitDir, "/exp1/stimuli/stimuli.csv", sep = ""),
                       header = T,
                       colClasses=c("cueID"="factor",
                        "bodyShape"="factor",
                        "label"="factor",
                        "fribbleID"="factor"
                        ));
```

Check stimuli set

It's important to check that every fribble is unique in the way its features are assembled within each category. Feature position and identity are coded into cueID.

I'm going to check whether the combination of cues used to build the fribble is unique by filtering for n > 1:

```r
fribbleSet %>%
  group_by(category, cueID) %>%
  count() %>%
  filter(n > 1);
```

```
## Warning: Factor `cueID` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
## # A tibble: 0 x 3
## # Groups:   category, cueID [1]
## # ... with 3 variables: category <int>, cueID <fct>, n <int>
```

Great, each Fribble is unique!

Load data

List the files present in the folder, and load them.

```r
df <- list.files(paste(localGitDir, "/exp1/data/", sep = ""));
```

We have 4 files.

```r
for (i in 1:length(df)){
  gsub(".csv$", "", df[i]) -> id
  assign(id, data.frame())
  read.csv(paste(localGitDir, "/exp1/data/", df[i], sep = ""),
           na.strings=c("","NA"),
           colClasses=c("presentedLabel"="factor",
                        "presentedImage"="factor",
                        "learningType"="factor",
                        "Trial.Type"="factor",
                        "Test.Part"="factor",
                        "Key.Press"="factor"
                        ))-> temp
```

```
  assign(paste0(id), temp)
};

rm(temp, df, i, id);
```

The dataset name is decided autonomously by Gorilla. Importantly, Gorilla produces a different file per condition, and codes the conditions by the last 4 letters.

- 2yjh is the FL learning
- q8hp is the LF learning

I'm going to rename them for clarity.

```
dataFL<-`data_exp_15519-v13_task-2yjh`
dataFL2<-`data_exp_15519-v14_task-2yjh`

rm(`data_exp_15519-v13_task-2yjh`)
rm(`data_exp_15519-v14_task-2yjh`)

dataLF <- `data_exp_15519-v13_task-q8hp`
dataLF2 <- `data_exp_15519-v14_task-q8hp`

rm(`data_exp_15519-v13_task-q8hp`)
rm(`data_exp_15519-v14_task-q8hp`)
```

```
rbind(dataFL, dataFL2)-> dataFL
rbind(dataLF, dataLF2)-> dataLF

rm(dataFL2, dataLF2)
```

Gorilla's output is extremely messy. Each row is a screen event. However, we want only the events related to 1. the presentations of the fribbles and the labels 2. participants' response and 3. what type of tasks.

I have coded these info in some columns and rows that I'm going to select:

```
raw_dataFL<- dataFL[c('Participant.Private.ID', 'learningType', 'Test.Part' ,
         'presentedImage', 'presentedLabel', 'Reaction.Time', "Key.Press",
          'Trial.Type', 'Trial.Index', 'Correct')]

raw_dataLF<- dataLF[c('Participant.Private.ID', 'learningType', 'Test.Part' ,
         'presentedImage', 'presentedLabel', 'Reaction.Time', "Key.Press",
          'Trial.Type', 'Trial.Index', 'Correct')]
```

Select rows:

```
rowsIwantTokeep <- c("learningBlock1", "learningBlock2", "learningBlock3",
                        "learningBlock4", "generalizationPL", "generalizationLP",
                        "randomDot", "contingencyJudgement")

raw_dataFL <- raw_dataFL %>%
  filter(Test.Part %in% rowsIwantTokeep ) %>%
```

```r
  rename(subjID = Participant.Private.ID,
         learning = learningType,
         task = Test.Part,
         fribbleID = presentedImage,
         label = presentedLabel,
         rt = Reaction.Time,
         resp = Key.Press,
         trialType = Trial.Type,
         trialIndex = Trial.Index,
         acc = Correct)

raw_dataLF <- raw_dataLF %>%
  filter(Test.Part %in% rowsIwantTokeep ) %>%
  rename(subjID = Participant.Private.ID,
         learning = learningType,
         task = Test.Part,
         fribbleID = presentedImage,
         label = presentedLabel,
         rt = Reaction.Time,
         resp = Key.Press,
         trialType = Trial.Type,
         trialIndex = Trial.Index,
         acc = Correct)

rm(rowsIwantTokeep, dataFL, dataLF);
```

I'm going to merge both datasets, FL and LF, because we have anyway a column "learning" that can tell us which one is which.

```r
rbind(raw_dataFL, raw_dataLF)-> raw_data;
rm(raw_dataFL, raw_dataLF);
```

Check learning

Let's filter and check learning trials:

```r
learningBlocks <- c("learningBlock1", "learningBlock2", "learningBlock3", "learningBlock4");

learning <- raw_data %>%
  filter(task %in% learningBlocks)

learning <- droplevels(learning);
rm(learningBlocks)
```

How many trials per participant?

```r
learning %>%
  group_by(subjID, learning) %>%
  count()
```

```
## # A tibble: 80 x 3
## # Groups:   subjID, learning [80]
##     subjID learning     n
##      <int> <fct>    <int>
##  1 1414932 LF         120
##  2 1414933 LF         120
##  3 1414937 FL         120
##  4 1414945 FL         120
##  5 1414957 FL         120
##  6 1415040 FL         120
##  7 1420163 FL         120
##  8 1420165 FL         120
##  9 1420169 LF         120
## 10 1420171 LF         120
## # ... with 70 more rows
```

Great, 120 trials per participant.

Let's check whether the blocks' length varied across participants:

```r
learning %>%
  group_by(subjID, task) %>%
  count()
```

```
## # A tibble: 320 x 3
## # Groups:   subjID, task [320]
##     subjID task               n
##      <int> <fct>          <int>
##  1 1414932 learningBlock1    21
##  2 1414932 learningBlock2    28
##  3 1414932 learningBlock3    47
##  4 1414932 learningBlock4    24
##  5 1414933 learningBlock1    26
##  6 1414933 learningBlock2    22
##  7 1414933 learningBlock3    44
##  8 1414933 learningBlock4    28
##  9 1414937 learningBlock1    27
## 10 1414937 learningBlock2    47
## # ... with 310 more rows
```

Great! Each participant had a different amount of trials distributed across blocks. That's important because our random dot task was presented at the end of each block, and we wanted its presentation to be unpredictable. Anyway, the sum of all the learning trials was always 120.

Did we assign our learning randomly every couple of people?

```r
table(learning$subjID, learning$learning)
```

```
##
##            FL  LF
##   1414932   0 120
##   1414933   0 120
##   1414937 120   0
```

```
##    1414945 120   0
##    1414957 120   0
##    1415040 120   0
##    1420163 120   0
##    1420165 120   0
##    1420169   0 120
##    1420171   0 120
##    1420177 120   0
##    1420180 120   0
##    1420185   0 120
##    1420199 120   0
##    1420204   0 120
##    1420552   0 120
##    1420573   0 120
##    1420577   0 120
##    1420580 120   0
##    1420622 120   0
##    1422463 120   0
##    1422465 120   0
##    1422466 120   0
##    1422467   0 120
##    1422470   0 120
##    1422472 120   0
##    1422473   0 120
##    1422475   0 120
##    1422476   0 120
##    1422477 120   0
##    1422675 120   0
##    1422676   0 120
##    1422677 120   0
##    1422678   0 120
##    1422679 120   0
##    1422680   0 120
##    1422681   0 120
##    1422689 120   0
##    1422715   0 120
##    1422716 120   0
##    1431942   0 120
##    1431944 120   0
##    1431946 120   0
##    1431948   0 120
##    1431949 120   0
##    1431952   0 120
##    1431953 120   0
##    1431954   0 120
##    1431956   0 120
##    1431957 120   0
##    1431958 120   0
##    1431959   0 120
##    1431960   0 120
##    1431961 120   0
##    1431963   0 120
##    1431965 120   0
##    1431966   0 120
```

```
##   1431968   0 120
##   1431969 120   0
##   1431970   0 120
##   1431972 120   0
##   1431974 120   0
##   1431978 120   0
##   1431979 120   0
##   1431981   0 120
##   1431984 120   0
##   1431989   0 120
##   1431992 120   0
##   1431997 120   0
##   1431998   0 120
##   1431999   0 120
##   1432003   0 120
##   1432007   0 120
##   1432009 120   0
##   1432011 120   0
##   1432030   0 120
##   1432052 120   0
##   1432075 120   0
##   1432301   0 120
##   1432323   0 120
```

Kind of. Apparently, if a participant access Gorilla, but it's not allowed to start the experiment (e.g., the browser is not suitable), or leaves the session, this counts anyway for the randomisation.

The rows related to the presentation of fribbles and labels, inherit Gorilla's http address of where they are stored. Nothing I can do to change this in Gorilla, but we can clean the rows by those info like this:

```r
as.factor(gsub("/task/70033/56/asset/|/task/70033/57/asset/", "", learning$fribbleID))-> learning$fribbl
as.factor(gsub(".jpg$", "", learning$fribbleID))-> learning$fribbleID

as.factor(gsub("/task/70033/56/asset/|/task/70033/57/asset/", "", learning$label))-> learning$label
as.factor(gsub(".mp3$", "", learning$label))-> learning$label
learning$resp <- as.factor('NA')
```

This is how the learning dataframe looks like now:

```r
head(learning);
```

```
##    subjID learning           task fribbleID label rt resp
## 1 1414937       FL learningBlock1     20375 FLbim NA   NA
## 2 1414937       FL learningBlock1     31075 FLtob NA   NA
## 3 1414937       FL learningBlock1     32775 FLtob NA   NA
## 4 1414937       FL learningBlock1     32875 FLtob NA   NA
## 5 1414937       FL learningBlock1     22025 FLbim NA   NA
## 6 1414937       FL learningBlock1     10425 FLdep NA   NA
##                 trialType trialIndex acc
## 1 audio-keyboard-response         22  NA
## 2 audio-keyboard-response         25  NA
## 3 audio-keyboard-response         28  NA
## 4 audio-keyboard-response         31  NA
```

```
## 5 audio-keyboard-response         34  NA
## 6 audio-keyboard-response         37  NA
```

```r
summary(learning);
```

```
##      subjID        learning              task        fribbleID      label     
##  Min.   :1414932   FL:4920   learningBlock1:2283   10975   :  86   FLbim:1640  
##  1st Qu.:1422003   LF:4680   learningBlock2:2549   22575   :  84   FLdep:1640  
##  Median :1427329             learningBlock3:2336   31975   :  84   FLtob:1640  
##  Mean   :1426320             learningBlock4:2432   32675   :  84   LFbim:1560  
##  3rd Qu.:1431971                                   21875   :  82   LFdep:1560  
##  Max.   :1432323                                   30375   :  82   LFtob:1560  
##                                                    (Other):9098               
##        rt          resp                        trialType      trialIndex 
##  Min.   : 12.36   NA:9600   audio-keyboard-response:4920   Min.   : 22  
##  1st Qu.: 52.50             image-keyboard-response:4680   1st Qu.:115  
##  Median : 88.00                                            Median :211  
##  Mean   :126.25                                            Mean   :211  
##  3rd Qu.:214.71                                            3rd Qu.:307  
##  Max.   :249.00                                            Max.   :400  
##  NA's   :9593                                                           
##       acc      
##  Min.   : NA   
##  1st Qu.: NA   
##  Median : NA   
##  Mean   :NaN   
##  3rd Qu.: NA   
##  Max.   : NA   
##  NA's   :9600  
```

Our fribbles were presented two times during learning. Let's check fribbles presented > 2 times:

```r
learning %>%
  group_by(subjID, fribbleID) %>%
  count() %>%
  filter(n >2)
```

```
## Warning: Factor `fribbleID` contains implicit NA, consider using
## `forcats::fct_explicit_na`

## # A tibble: 0 x 3
## # Groups:   subjID, fribbleID [1]
## # ... with 3 variables: subjID <int>, fribbleID <fct>, n <int>
```

None, perfect. Let's check whether there are fribbles presented only once:

```r
learning %>%
  group_by(subjID, fribbleID) %>%
  count() %>%
  filter(n < 2)
```

```
## Warning: Factor `fribbleID` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
## # A tibble: 0 x 3
## # Groups:   subjID, fribbleID [1]
## # ... with 3 variables: subjID <int>, fribbleID <fct>, n <int>
```

Perfect.

Check the association between the fribbles and the labels. Fribbles ID are coded in this way: e.g., 10175-> [1] is the category [01] is the number of the fribble [75] is the frequency.

In the column fribbleID we have the fribble presented, in the column label we have the sound played.

Association between fribbles and labels are fixed:

- category 1, regardless of the frequency, has the label: dep
- category 2, regardless of the frequency, has the label: bim
- category 3, regardless of the frequency, has the label: tob

I'm going to add a column for category, fribble number, and frequency, in order to check whether everything is okay:

We should have only 3 categories, presented twice per participant. Each category is made of 20 exemplars.

```r
learning$category <- 0
learning[substr(as.character(learning$fribbleID), 1, 1)==1,]$category <- 1
learning[substr(as.character(learning$fribbleID), 1, 1)==2,]$category <- 2
learning[substr(as.character(learning$fribbleID), 1, 1)==3,]$category <- 3

(nrow(learning[learning$category==1,]) / length(unique(learning$subjID))) / 2
```

```
## [1] 20
```

```r
(nrow(learning[learning$category==2,]) / length(unique(learning$subjID))) / 2
```

```
## [1] 20
```

```r
(nrow(learning[learning$category==3,]) / length(unique(learning$subjID))) / 2
```

```
## [1] 20
```

We have 15 high frequency and 5 low frequency exemplars x category:

```r
learning$frequency <- 25
learning[substr(as.character(learning$fribbleID), 4, 5)==75,]$frequency <- 75

(nrow(learning[learning$frequency==25,]) / length(unique(learning$subjID))) / 2
```

```
## [1] 15
```

```
(nrow(learning[learning$frequency==75,]) / length(unique(learning$subjID))) / 2
```

```
## [1] 45
```

Now let's check the fribble-label association:

```
table(learning$category, learning$label, learning$frequency)
```

```
## , ,  = 25
##
##
##     FLbim FLdep FLtob LFbim LFdep LFtob
##   1     0   410     0     0   390     0
##   2   410     0     0   390     0     0
##   3     0     0   410     0     0   390
##
## , ,  = 75
##
##
##     FLbim FLdep FLtob LFbim LFdep LFtob
##   1     0  1230     0     0  1170     0
##   2  1230     0     0  1170     0     0
##   3     0     0  1230     0     0  1170
```

Okay, each label was associated to its correct fribble (coded here as category).

Check Testing

I'm going to select the tests and clean the rows from Gorilla's http address:

```
tests <- c("generalizationPL", "generalizationLP", "contingencyJudgement", "randomDot");

testing <- raw_data %>%
  filter(task %in% tests)


testing <- droplevels(testing);
rm(tests);

as.factor(gsub("/task/70033/56/asset/|/task/70033/57/asset/", "", testing$fribbleID))-> testing$fribble
as.factor(gsub(".jpg$", "", testing$fribbleID))-> testing$fribbleID

as.factor(gsub("/task/70033/56/asset/|/task/70033/57/asset/", "", testing$label))-> testing$label
as.factor(gsub(".mp3$", "", testing$label))-> testing$label
```

Check test 1: Generalization from picture to labels

We filter the rows for this task, and clean both the resp and fribble columns.

```r
generalizationPL <- testing %>%
  filter(task == 'generalizationPL')
generalizationPL <- droplevels(generalizationPL);

as.factor(gsub("/task/70033/56/asset/|/task/70033/57/asset/", "", generalizationPL$resp))-> generalizat
as.factor(gsub(".mp3$", "", generalizationPL$resp))-> generalizationPL$resp
as.factor(gsub(".jpg", "", generalizationPL$resp))-> generalizationPL$resp

as.factor(gsub('[[:punct:]]|"', "", generalizationPL$label))-> generalizationPL$label
as.factor(gsub('mp3', "_", generalizationPL$label))-> generalizationPL$label
```

Check how many trials participants:

```r
generalizationPL %>%
  group_by(subjID) %>%
  count()
```

```
## # A tibble: 80 x 2
## # Groups:   subjID [80]
##     subjID     n
##      <int> <int>
##  1 1414932    24
##  2 1414933    24
##  3 1414937    24
##  4 1414945    24
##  5 1414957    24
##  6 1415040    24
##  7 1420163    24
##  8 1420165    24
##  9 1420169    24
## 10 1420171    24
## # ... with 70 more rows
```

Great, 24 trials per participant.

Check whether participants saw a unique fribble:

```r
generalizationPL %>%
  group_by(subjID, fribbleID) %>%
  count() %>%
  filter(n > 1)
```

```
## Warning: Factor `fribbleID` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
## # A tibble: 0 x 3
## # Groups:   subjID, fribbleID [1]
## # ... with 3 variables: subjID <int>, fribbleID <fct>, n <int>
```

Great!

Integrate stimuli info. In the file "fribbleSet" I have listed all the fribbles ID and their category, along with their cueIDs and body shape. I'm going to add those columns by merging the test file with the fribbleSet by fribbleID. The rest of the file is left untouched.

```r
merge(generalizationPL, fribbleSet, by = 'fribbleID')-> generalizationPL;
generalizationPL$label.y <- NULL;

generalizationPL <- rename(generalizationPL, label = label.x);
```

Let's check the responses they made, just to see if they make sense.

For example, we want the resp column to be one of the labels.

```r
generalizationPL %>%
  group_by(subjID, resp) %>%
  count()
```

```
## Warning: Factor `resp` contains implicit NA, consider using
## `forcats::fct_explicit_na`

## Warning: Factor `resp` contains implicit NA, consider using
## `forcats::fct_explicit_na`

## Warning: Factor `resp` contains implicit NA, consider using
## `forcats::fct_explicit_na`

## # A tibble: 291 x 3
## # Groups:   subjID, resp [291]
##     subjID resp      n
##      <int> <fct> <int>
##  1 1414932 bim       6
##  2 1414932 dep       5
##  3 1414932 tob       9
##  4 1414932 <NA>      4
##  5 1414933 bim       8
##  6 1414933 dep       8
##  7 1414933 tob       8
##  8 1414937 bim       8
##  9 1414937 dep       7
## 10 1414937 tob       8
## # ... with 281 more rows
```

Great, some participant missed some trials (coded as NA), but that's okay.

So far, so good.

We have 24 trials per participant, but within those trials we have 8 trials per category, 4 low frequency and 4 high frequency trials. Let's check:

```r
head(table(generalizationPL$subjID, generalizationPL$category, generalizationPL$frequency))
```

```
## , ,  = 25
##
##
##             1 2 3
##   1414932 4 4 4
##   1414933 4 4 4
```

```
##   1414937 4 4 4
##   1414945 4 4 4
##   1414957 4 4 4
##   1415040 4 4 4
##
## , ,  = 75
##
##
##           1 2 3
##   1414932 4 4 4
##   1414933 4 4 4
##   1414937 4 4 4
##   1414945 4 4 4
##   1414957 4 4 4
##   1415040 4 4 4
```

Let's check the second task.

Check test 2: Generalization from label to pictures

```
generalizationLP <- testing %>%
  filter(task == 'generalizationLP')
generalizationLP <- droplevels(generalizationLP)
```

How many trials per participant?

```
generalizationLP %>%
  group_by(subjID) %>%
  count()
```

```
## # A tibble: 80 x 2
## # Groups:   subjID [80]
##     subjID     n
##      <int> <int>
##  1 1414932    24
##  2 1414933    24
##  3 1414937    24
##  4 1414945    24
##  5 1414957    24
##  6 1415040    24
##  7 1420163    24
##  8 1420165    24
##  9 1420169    24
## 10 1420171    24
## # ... with 70 more rows
```

24 trials, great.

Let's check whether participants saw a unique fribble by checking for duplicates: First let's clean the rows from Gorilla gibberish.

```
as.factor(gsub('[[:punct:]]|"', "", generalizationLP$fribbleID))-> generalizationLP$fribbleID
as.factor(gsub('jpg', "_", generalizationLP$fribbleID))-> generalizationLP$fribbleID

as.factor(gsub("/task/70033/56/asset/|/task/70033/57/asset/", "", generalizationLP$resp))-> generalizat
as.factor(gsub(".jpg", "", generalizationLP$resp))-> generalizationLP$resp
```

Then check for duplicates:

```
substr(as.character(generalizationLP$fribbleID), 1, 5)-> temp
substr(as.character(generalizationLP$fribbleID), 7, 11)-> temp2
substr(as.character(generalizationLP$fribbleID), 13, 17)-> temp3

fribblePresented <- c(temp,temp2,temp3)
unique(generalizationLP$subjID)-> subj

duplicatedFribbles <- NA;
for (i in 1:length(subj)){
  substr(as.character(generalizationLP[generalizationLP$subjID==subj[i],]$fribbleID), 1, 5)-> temp
  substr(as.character(generalizationLP[generalizationLP$subjID==subj[i],]$fribbleID), 7, 11)-> temp2
  substr(as.character(generalizationLP[generalizationLP$subjID==subj[i],]$fribbleID), 13, 17)-> temp3
  fribblePresented <- c(temp,temp2,temp3)
  dup <- fribblePresented[duplicated(fribblePresented)] #extract duplicated elements
  print(subj[i])

  if (length(dup)>0){
    print(dup)
  } else {
    print(length(dup))
  }

};
```

```
## [1] 1414937
## [1] 0
## [1] 1414945
## [1] 0
## [1] 1414957
## [1] 0
## [1] 1415040
## [1] 0
## [1] 1431949
## [1] 0
## [1] 1431944
## [1] 0
## [1] 1431953
## [1] 0
## [1] 1431958
## [1] 0
## [1] 1431965
## [1] 0
## [1] 1431946
## [1] 0
## [1] 1431957
```

```
## [1] 0
## [1] 1431961
## [1] 0
## [1] 1431969
## [1] 0
## [1] 1431978
## [1] 0
## [1] 1431979
## [1] 0
## [1] 1422477
## [1] 0
## [1] 1422675
## [1] 0
## [1] 1422677
## [1] 0
## [1] 1422679
## [1] 0
## [1] 1422689
## [1] 0
## [1] 1422716
## [1] 0
## [1] 1431972
## [1] 0
## [1] 1431974
## [1] 0
## [1] 1431984
## [1] 0
## [1] 1431992
## [1] 0
## [1] 1431997
## [1] 0
## [1] 1432009
## [1] 0
## [1] 1432011
## [1] 0
## [1] 1432052
## [1] 0
## [1] 1432075
## [1] 0
## [1] 1420163
## [1] 0
## [1] 1420165
## [1] 0
## [1] 1420177
## [1] 0
## [1] 1420180
## [1] 0
## [1] 1420199
## [1] 0
## [1] 1420580
## [1] 0
## [1] 1420622
## [1] 0
## [1] 1422463
```

```
## [1] 0
## [1] 1422465
## [1] 0
## [1] 1422466
## [1] 0
## [1] 1422472
## [1] 0
## [1] 1414933
## [1] 0
## [1] 1414932
## [1] 0
## [1] 1420169
## [1] 0
## [1] 1420171
## [1] 0
## [1] 1420577
## [1] 0
## [1] 1422467
## [1] 0
## [1] 1422475
## [1] 0
## [1] 1422678
## [1] 0
## [1] 1422680
## [1] 0
## [1] 1422681
## [1] 0
## [1] 1431942
## [1] 0
## [1] 1431948
## [1] 0
## [1] 1431966
## [1] 0
## [1] 1431968
## [1] 0
## [1] 1431952
## [1] 0
## [1] 1431954
## [1] 0
## [1] 1431956
## [1] 0
## [1] 1431959
## [1] 0
## [1] 1431960
## [1] 0
## [1] 1431963
## [1] 0
## [1] 1431970
## [1] 0
## [1] 1431981
## [1] 0
## [1] 1431989
## [1] 0
## [1] 1431998
```

```
## [1] 0
## [1] 1431999
## [1] 0
## [1] 1432003
## [1] 0
## [1] 1432007
## [1] 0
## [1] 1432030
## [1] 0
## [1] 1420185
## [1] 0
## [1] 1420204
## [1] 0
## [1] 1420552
## [1] 0
## [1] 1420573
## [1] 0
## [1] 1422470
## [1] 0
## [1] 1422473
## [1] 0
## [1] 1422476
## [1] 0
## [1] 1422676
## [1] 0
## [1] 1422715
## [1] 0
## [1] 1432301
## [1] 0
## [1] 1432323
## [1] 0
```

```
rm(subj, temp, temp2, temp3, i, fribblePresented, duplicatedFribbles, dup)
```

Great! participants saw always different fribble.

Check whether fribbles presented were either high or low frequency.

In this task we have three pictures and one label pronounced. This means that the fribbleID column contains 3 images. I'm going to cycle over the dataset, and break the fribbleID column in three, then I'm going to print the fribble that within the same trial has a different frequency. I'm going to print the fribbles that are presented wrongly, e.g., "low high low" etc. If all fribbles are presented correctly: , e.g., "low low low" and "high high high", then the output is empty.

```
unique(generalizationLP$subjID)-> subj;

trials <- NULL;
task <- NULL;

for (i in 1:length(subj)){
  as.integer(substr(as.character(generalizationLP[generalizationLP$subjID==subj[i],]$fribbleID), 4, 5))
  as.integer(substr(as.character(generalizationLP[generalizationLP$subjID==subj[i],]$fribbleID), 10, 11
  as.integer(substr(as.character(generalizationLP[generalizationLP$subjID==subj[i],]$fribbleID), 16, 17
trials <- cbind(temp, temp2, temp3, as.integer(subj[i])) # store it in columns along with subj info
```

```r
task <- rbind(task, trials) #store all subjs
};

for (i in 1:nrow(task)){ #check by rows whether there is a unique number, print the row if wrong
  if ((task[i,1] == task[i,2] & task[i,3])== FALSE) {
    print('wrong frequency fribble:')
    print(task[i,1], task[i,2], task[i,3])
  }
};

frequency <- ifelse(substr(as.character(task[,1]), 1, 1)==2, 'low', 'high')
cbind(task, frequency)->task
as.data.frame(task)-> task
rm(trials, i, subj, temp, temp2, temp3);
```

Great, fribbles presented were either low or high frequency. Check whether participants saw 4 trials with low and 4 trials with high frequency:

Let's see how these are distributed:

```r
head(table(task$V4, task$frequency))
```

```
##
##           high low
##   1414932   12  12
##   1414933   12  12
##   1414937   12  12
##   1414945   12  12
##   1414957   12  12
##   1415040   12  12
```

I'm going to merge the stimuli set now.

When we do it, this time we need to merge by resp and not by fribbleID, because our fribble selected is coded in this column:

```r
fribbleSet$resp <- fribbleSet$fribbleID # column's name needs to be the same in order to merge
merge(generalizationLP, fribbleSet, by = 'resp', all.x = T)-> generalizationLP;
fribbleSet$resp <- NULL;
generalizationLP$fribbleID.y <- NULL;
generalizationLP$label.y <- NULL;
generalizationLP <- rename(generalizationLP, label = label.x);
generalizationLP <- rename(generalizationLP, fribbleID = fribbleID.x);
```

Let's check whether we have responses in all the three categories:

```r
generalizationLP %>%
  group_by(subjID, category) %>%
  count()
```

```
## # A tibble: 295 x 3
## # Groups:   subjID, category [295]
```

```
##     subjID category     n
##      <int>    <int> <int>
##  1 1414932        1     7
##  2 1414932        2    11
##  3 1414932        3     2
##  4 1414932       NA     4
##  5 1414933        1     8
##  6 1414933        2     5
##  7 1414933        3    10
##  8 1414933       NA     1
##  9 1414937        1     7
## 10 1414937        2     7
## # ... with 285 more rows
```

Cool.

Check responses distribution over category:

```
generalizationLP %>%
  group_by(subjID, label, frequency) %>%
  count()
```

```
## # A tibble: 583 x 4
## # Groups:   subjID, label, frequency [583]
##     subjID label frequency     n
##      <int> <fct>     <int> <int>
##  1 1414932 bim          25     3
##  2 1414932 bim          75     4
##  3 1414932 bim          NA     1
##  4 1414932 dep          25     3
##  5 1414932 dep          75     3
##  6 1414932 dep          NA     2
##  7 1414932 tob          25     3
##  8 1414932 tob          75     4
##  9 1414932 tob          NA     1
## 10 1414933 bim          25     4
## # ... with 573 more rows
```

Check test 3: Contingency Judgement task

```
contingencyJudgement <- testing %>%
  filter(task == 'contingencyJudgement')
contingencyJudgement <- droplevels(contingencyJudgement)
```

How many trials per participant?

```
contingencyJudgement %>%
  group_by(subjID) %>%
  count()
```

```
## # A tibble: 80 x 2
```

```
## # Groups:   subjID [80]
##     subjID     n
##      <int> <int>
##  1 1414932    24
##  2 1414933    24
##  3 1414937    24
##  4 1414945    24
##  5 1414957    24
##  6 1415040    24
##  7 1420163    24
##  8 1420165    24
##  9 1420169    24
## 10 1420171    24
## # ... with 70 more rows
```

Very good.

Did participants see a fribble more than once?

```
droplevels(contingencyJudgement) %>%
  group_by(subjID, fribbleID) %>%
  count() %>%
  filter( n > 1)
```

```
## Warning: Factor `fribbleID` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
## # A tibble: 0 x 3
## # Groups:   subjID, fribbleID [1]
## # ... with 3 variables: subjID <int>, fribbleID <fct>, n <int>
```

No! that's great.

Are labels repeated equally?

```
table(contingencyJudgement$subjID, contingencyJudgement$label)
```

```
##
##           bim dep tob
##   1414932   8   8   8
##   1414933   8   8   8
##   1414937   8   8   8
##   1414945   8   8   8
##   1414957   8   8   8
##   1415040   8   8   8
##   1420163   8   8   8
##   1420165   8   8   8
##   1420169   8   8   8
##   1420171   8   8   8
##   1420177   8   8   8
##   1420180   8   8   8
##   1420185   8   8   8
##   1420199   8   8   8
```

```
##    1420204   8   8   8
##    1420552   8   8   8
##    1420573   8   8   8
##    1420577   8   8   8
##    1420580   8   8   8
##    1420622   8   8   8
##    1422463   8   8   8
##    1422465   8   8   8
##    1422466   8   8   8
##    1422467   8   8   8
##    1422470   8   8   8
##    1422472   8   8   8
##    1422473   8   8   8
##    1422475   8   8   8
##    1422476   8   8   8
##    1422477   8   8   8
##    1422675   8   8   8
##    1422676   8   8   8
##    1422677   8   8   8
##    1422678   8   8   8
##    1422679   8   8   8
##    1422680   8   8   8
##    1422681   8   8   8
##    1422689   8   8   8
##    1422715   8   8   8
##    1422716   8   8   8
##    1431942   8   8   8
##    1431944   8   8   8
##    1431946   8   8   8
##    1431948   8   8   8
##    1431949   8   8   8
##    1431952   8   8   8
##    1431953   8   8   8
##    1431954   8   8   8
##    1431956   8   8   8
##    1431957   8   8   8
##    1431958   8   8   8
##    1431959   8   8   8
##    1431960   8   8   8
##    1431961   8   8   8
##    1431963   8   8   8
##    1431965   8   8   8
##    1431966   8   8   8
##    1431968   8   8   8
##    1431969   8   8   8
##    1431970   8   8   8
##    1431972   8   8   8
##    1431974   8   8   8
##    1431978   8   8   8
##    1431979   8   8   8
##    1431981   8   8   8
##    1431984   8   8   8
##    1431989   8   8   8
##    1431992   8   8   8
```

```
##   1431997   8   8   8
##   1431998   8   8   8
##   1431999   8   8   8
##   1432003   8   8   8
##   1432007   8   8   8
##   1432009   8   8   8
##   1432011   8   8   8
##   1432030   8   8   8
##   1432052   8   8   8
##   1432075   8   8   8
##   1432301   8   8   8
##   1432323   8   8   8
```

good

```
merge(contingencyJudgement, fribbleSet, by = 'fribbleID')-> contingencyJudgement
contingencyJudgement$label.y <- NULL;
contingencyJudgement <- rename(contingencyJudgement, label = label.x)
```

Check category presentation:

```
contingencyJudgement %>%
  group_by(subjID, category) %>%
  count()
```

```
## # A tibble: 240 x 3
## # Groups:   subjID, category [240]
##     subjID category     n
##      <int>    <int> <int>
##  1 1414932        1     8
##  2 1414932        2     8
##  3 1414932        3     8
##  4 1414933        1     8
##  5 1414933        2     8
##  6 1414933        3     8
##  7 1414937        1     8
##  8 1414937        2     8
##  9 1414937        3     8
## 10 1414945        1     8
## # ... with 230 more rows
```

Check test 4: Random dot task

Let's check our random dot task. This was inserted randomly during trials 4 times. 5 trials each time, plus 4 practice trials.

```
randomDot <- testing %>%
  filter(task == 'randomDot')
```

How many trials per participant?

```r
randomDot %>%
  group_by(subjID) %>%
  count()
```

```
## # A tibble: 80 x 2
## # Groups:   subjID [80]
##     subjID     n
##      <int> <int>
##  1 1414932    26
##  2 1414933    26
##  3 1414937    26
##  4 1414945    26
##  5 1414957    26
##  6 1415040    26
##  7 1420163    26
##  8 1420165    26
##  9 1420169    26
## 10 1420171    26
## # ... with 70 more rows
```

we have 5 trials repeated during learning four times (20) plus 4 practice trials. How was accuracy distributed across participants?

First, let's consider that when we have a timeout, the output is -1

```r
randomDot %>%
  group_by(subjID, resp) %>%
  filter(rt == -1) %>%
  count()
```

```
## # A tibble: 57 x 3
## # Groups:   subjID, resp [57]
##     subjID resp      n
##      <int> <fct> <int>
##  1 1414932 -1       10
##  2 1414933 -1        1
##  3 1414945 -1        3
##  4 1415040 -1        1
##  5 1420163 -1        2
##  6 1420165 -1        1
##  7 1420180 -1        2
##  8 1420185 -1        1
##  9 1420204 -1        1
## 10 1420552 -1        3
## # ... with 47 more rows
```

Here we can see that some participant missed some trials.

Let's see how accuracy is coded when response is -1:

```r
head(randomDot[randomDot$rt == -1,]$acc)
```

```
## [1] NA NA NA NA NA NA
```

So it is coded as "NA", great. However:

```r
nrow(randomDot[is.na(randomDot$acc),]) #total of NA
```

```
## [1] 198
```

```r
nrow(randomDot[randomDot$resp == -1,]) # total of timeouts
```

```
## [1] 127
```

There are more NA's in acc than can be explained by timeouts. This means that also wrong responses are coded as NA. We need to recode those.

```r
randomDot[is.na(randomDot$acc),]$acc <- 0 #recode everything that is wrong or timeout as 0
```

So, now we can check the overall accuracy of participants, filtering by timeouts:

```r
aggregate(acc ~ subjID, data = randomDot[!(randomDot$resp == -1),], FUN = mean)# without timeouts
```

```
##     subjID       acc
## 1  1414932 0.6875000
## 2  1414933 1.0000000
## 3  1414937 1.0000000
## 4  1414945 1.0000000
## 5  1414957 1.0000000
## 6  1415040 1.0000000
## 7  1420163 0.9583333
## 8  1420165 0.9600000
## 9  1420169 1.0000000
## 10 1420171 1.0000000
## 11 1420177 1.0000000
## 12 1420180 0.9583333
## 13 1420185 1.0000000
## 14 1420199 1.0000000
## 15 1420204 1.0000000
## 16 1420552 1.0000000
## 17 1420573 1.0000000
## 18 1420577 0.9583333
## 19 1420580 1.0000000
## 20 1420622 1.0000000
## 21 1422463 1.0000000
## 22 1422465 1.0000000
## 23 1422466 0.9565217
## 24 1422467 1.0000000
## 25 1422470 0.7600000
## 26 1422472 1.0000000
## 27 1422473 1.0000000
## 28 1422475 0.5200000
## 29 1422476 0.9600000
## 30 1422477 1.0000000
## 31 1422675 1.0000000
```

```
## 32 1422676 0.9615385
## 33 1422677 0.9047619
## 34 1422678 0.9600000
## 35 1422679 0.9565217
## 36 1422680 1.0000000
## 37 1422681 1.0000000
## 38 1422689 0.6000000
## 39 1422715 1.0000000
## 40 1422716 1.0000000
## 41 1431942 0.8461538
## 42 1431944 0.7619048
## 43 1431946 1.0000000
## 44 1431948 0.9600000
## 45 1431949 1.0000000
## 46 1431952 0.9565217
## 47 1431953 0.9615385
## 48 1431954 1.0000000
## 49 1431956 0.9166667
## 50 1431957 1.0000000
## 51 1431958 0.9615385
## 52 1431959 1.0000000
## 53 1431960 1.0000000
## 54 1431961 1.0000000
## 55 1431963 1.0000000
## 56 1431965 1.0000000
## 57 1431966 0.9600000
## 58 1431968 1.0000000
## 59 1431969 1.0000000
## 60 1431970 0.9565217
## 61 1431972 0.9600000
## 62 1431974 1.0000000
## 63 1431978 1.0000000
## 64 1431979 1.0000000
## 65 1431981 1.0000000
## 66 1431984 0.9600000
## 67 1431989 1.0000000
## 68 1431992 1.0000000
## 69 1431997 1.0000000
## 70 1431998 1.0000000
## 71 1431999 1.0000000
## 72 1432003 0.9130435
## 73 1432007 1.0000000
## 74 1432009 0.9600000
## 75 1432011 0.9090909
## 76 1432030 1.0000000
## 77 1432052 0.9166667
## 78 1432075 0.9600000
## 79 1432301 1.0000000
## 80 1432323 1.0000000
```

Now that we have all tests separated, better to remove this file:

Data visualization

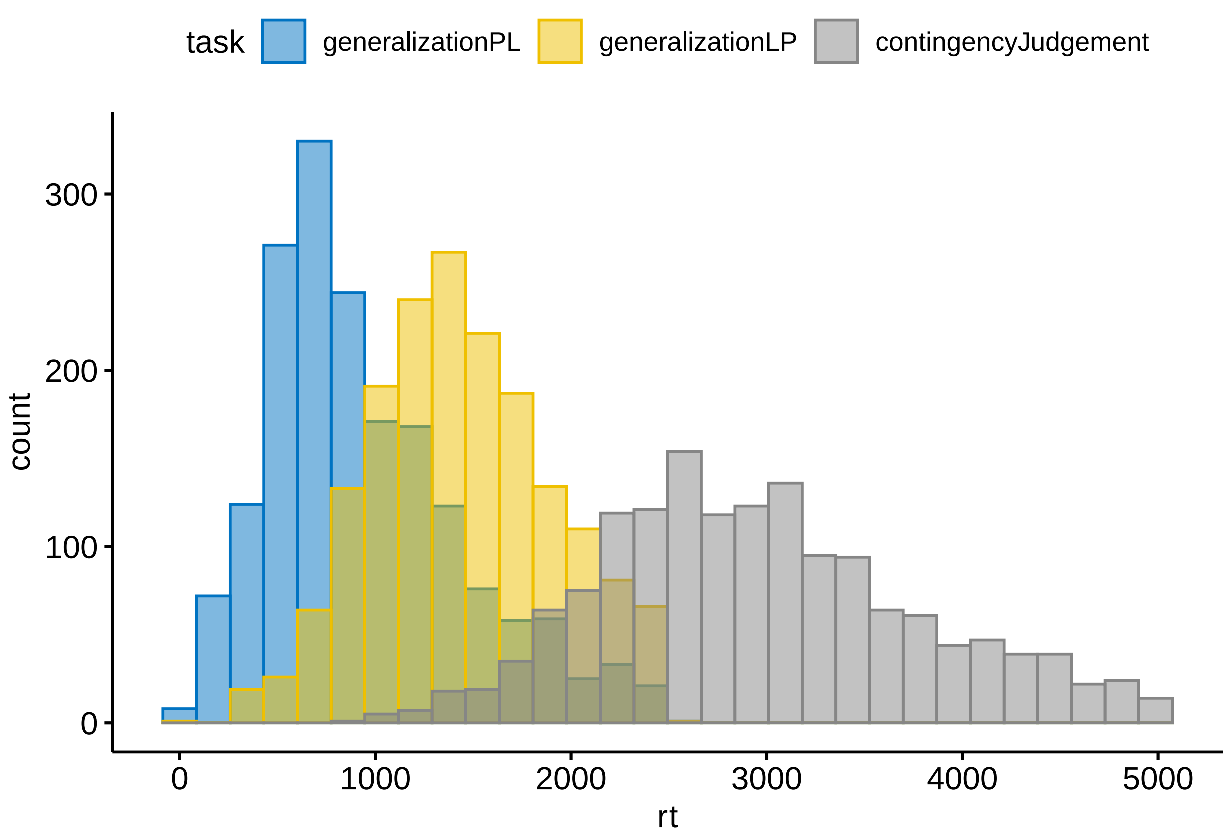
rt

```r
rbind(generalizationPL, generalizationLP, contingencyJudgement)-> alltasks
alltasks <- droplevels(alltasks)
```

```r
gghistogram(alltasks,
       x = "rt",
       y = "..count..",
       xlab = "rt",
       color = "task",
       fill = "task",
       palette = "jco"
)
```

```
## Warning: Using `bins = 30` by default. Pick better value with the argument
## `bins`.
```

```
## Warning: Removed 697 rows containing non-finite values (stat_bin).
```



```
rm(alltasks)
```

accuracy

RandomDot

```
unique(randomDot$subjID)-> subj;
randomDot-> randomTask

trials <- c(rep('0', 6), rep('1', 5),
              rep('2', 5), rep('3', 5),
              rep('4', 5))

trialstot <- as.factor(rep(trials, length(subj)))

randomTask$blocks <- trialstot
```

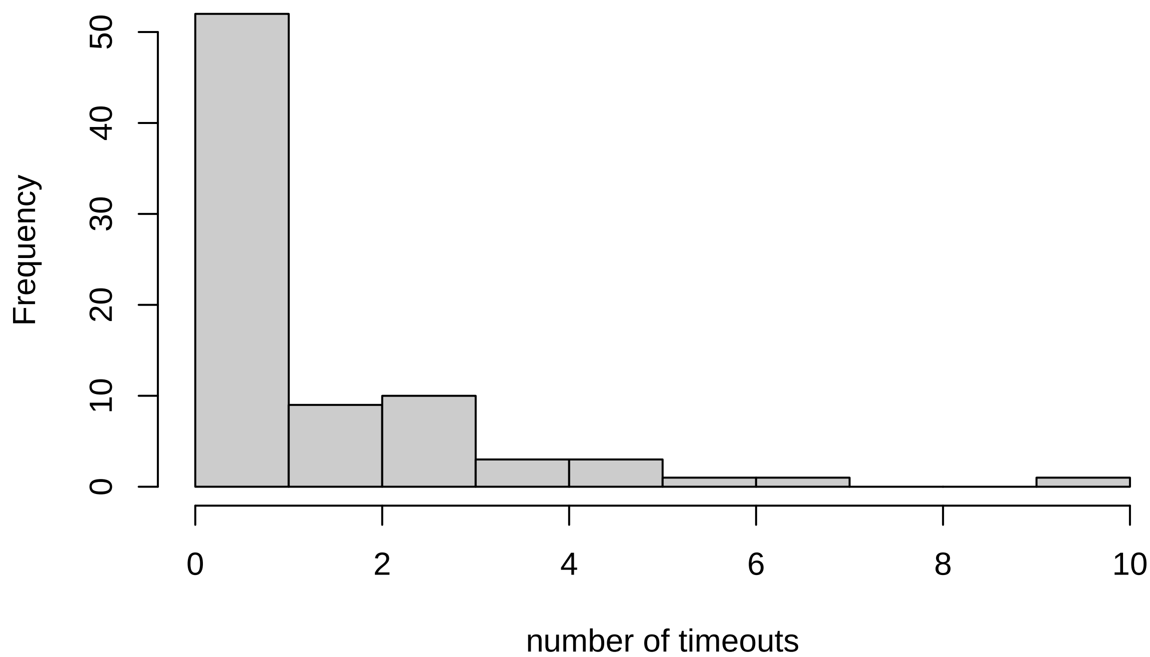
How many timeouts?

```
randomTask$timeout <- ifelse(randomTask$resp== -1, 1, 0)
```

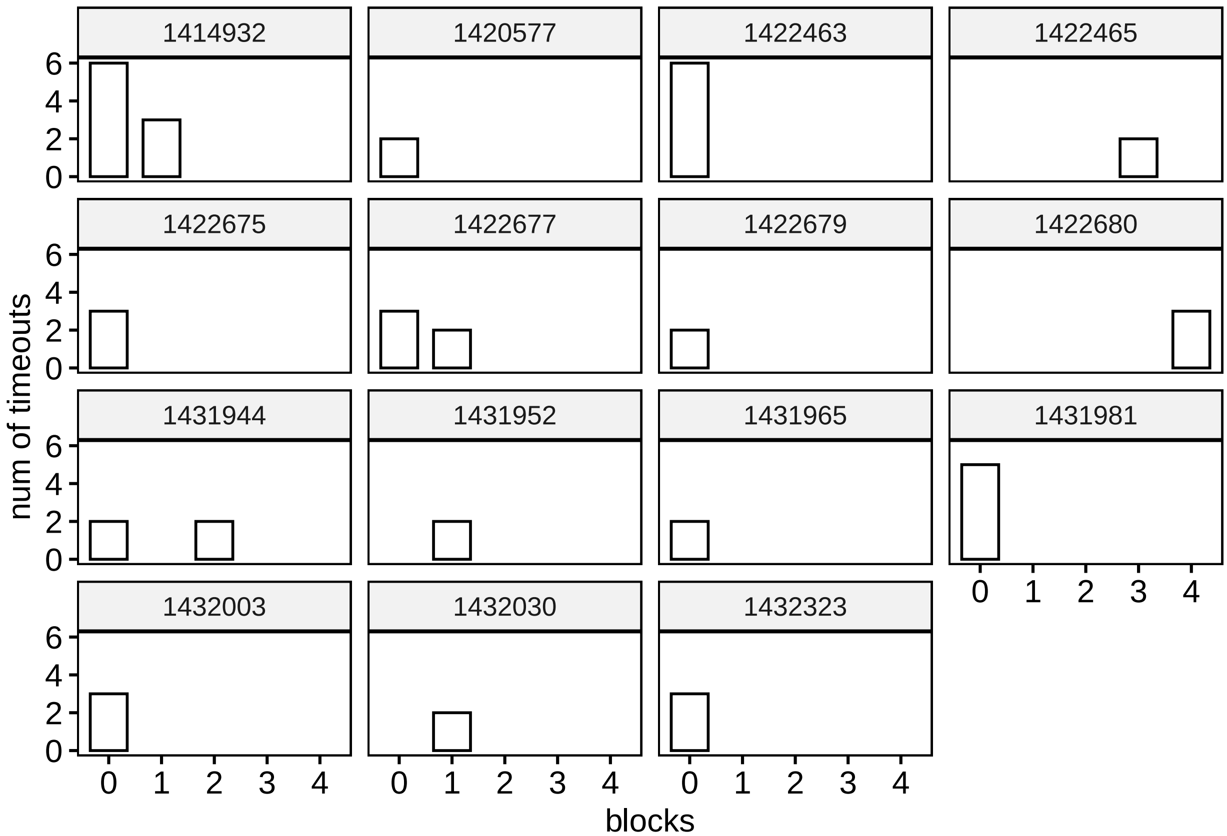
```
temp<-randomTask %>%
  count(timeout, subjID) %>%
  filter(timeout == 1)

unique(temp$subjID)-> subjs

temp2<-randomTask[!(randomTask$subjID %in% subjs),] %>%
  count(timeout, subjID) %>%
  filter(timeout == 0)

temp2[temp2$timeout==0,]$n <- 0

rbind(temp,temp2)-> timeout
```

Histogram

```
hist(timeout$n, xlab = 'number of timeouts',
     main = '',
     col=grey(.80),
     border=grey(0),
     breaks = seq(0,max(timeout$n),1))
```

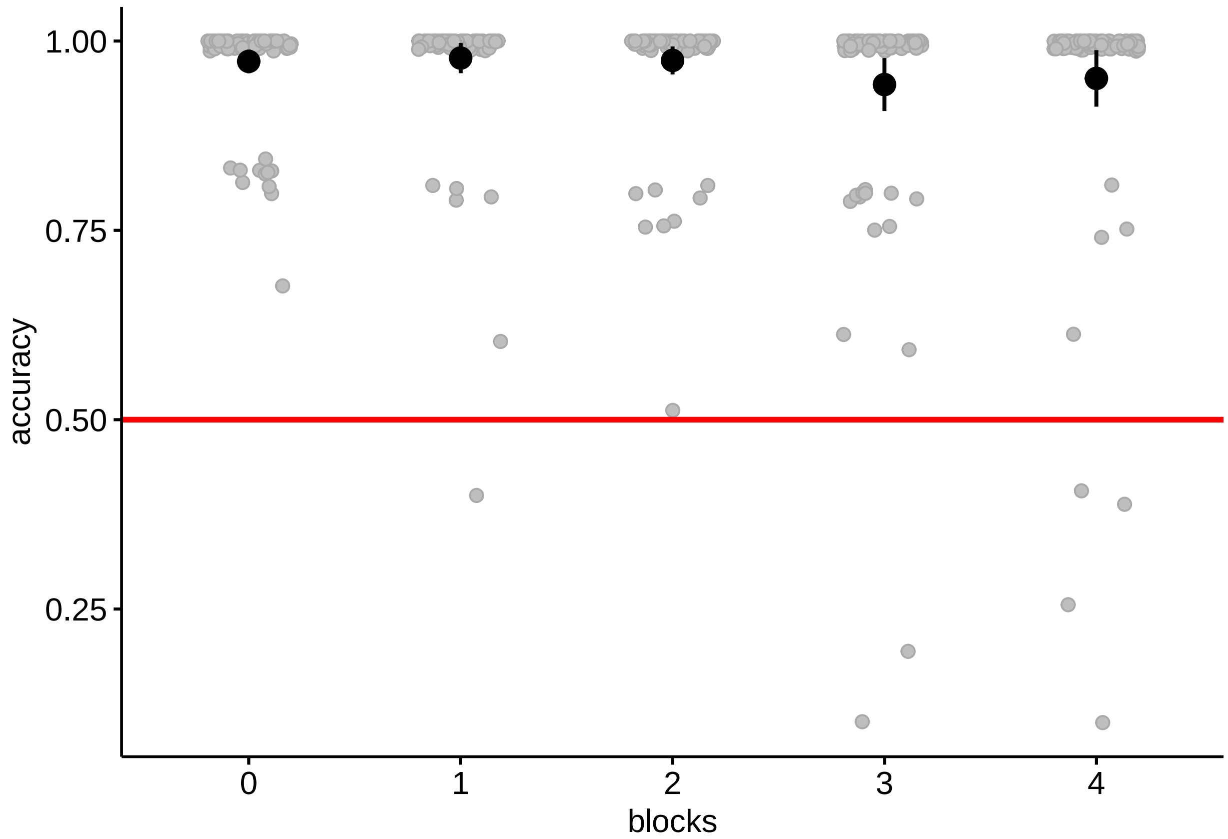



```
timeout <- randomTask %>%
  group_by(subjID, blocks) %>%
  filter(resp == -1) %>%
  count()


ggbarplot(timeout[timeout$n>1,], x = "blocks", y = "n",
          facet.by = "subjID",
          sort.by.groups = TRUE,      # Sort inside each group
          ylab = "num of timeouts")
```




```
accdistr <- randomTask[!(randomTask$resp == -1),] %>%
  group_by(subjID, blocks) %>%
  summarise(m = mean(acc))
```

```
ggstripchart(accdistr, x = "blocks", y = "m",
             xlab = "blocks",
             ylab = "accuracy",
             add = "mean_ci",
             size = 2,
             color = "darkgray",
             shape = 21,
             fill = "gray",
             error.plot = "pointrange",
             add.params = list(color = "black",
                               size = 0.7)) +
  scale_y_continuous(limits = c(0.1, 1), oob = scales::squish) + #to prevent jitter to move above 100%
  geom_hline(yintercept = .50, col='red', lwd=1);
```




```
accdistr[accdistr$m<.7,]
```

```
## # A tibble: 13 x 3
## # Groups:   subjID [8]
##     subjID blocks     m
##      <int> <fct>  <dbl>
##  1 1414932 3      0.6
##  2 1414932 4      0.25
##  3 1422470 1      0.4
##  4 1422475 2      0.5
##  5 1422475 3      0.2
##  6 1422475 4      0
##  7 1422689 3      0
##  8 1422689 4      0.4
##  9 1431942 4      0.4
## 10 1431944 1      0.6
## 11 1431944 3      0.6
## 12 1431956 4      0.6
## 13 1432003 0      0.667
```

```r
rm(temp, temp2, timeout, subj, subjs, trials, trialstot, accdistr)
```

Task 1: from picture to labels

The column fribbleID stores the fribble presented, while the column label stores the labels presented. Resp column in this task refers to the label selected. Category and frequency refers to the fribbleID column.

I'm going to add 1 in the accuracy column for every instance where response matches the category column, i.e., the participant correctly associated the fribble to its label.

I remove the no-response, and compute accuracy based on category and response.

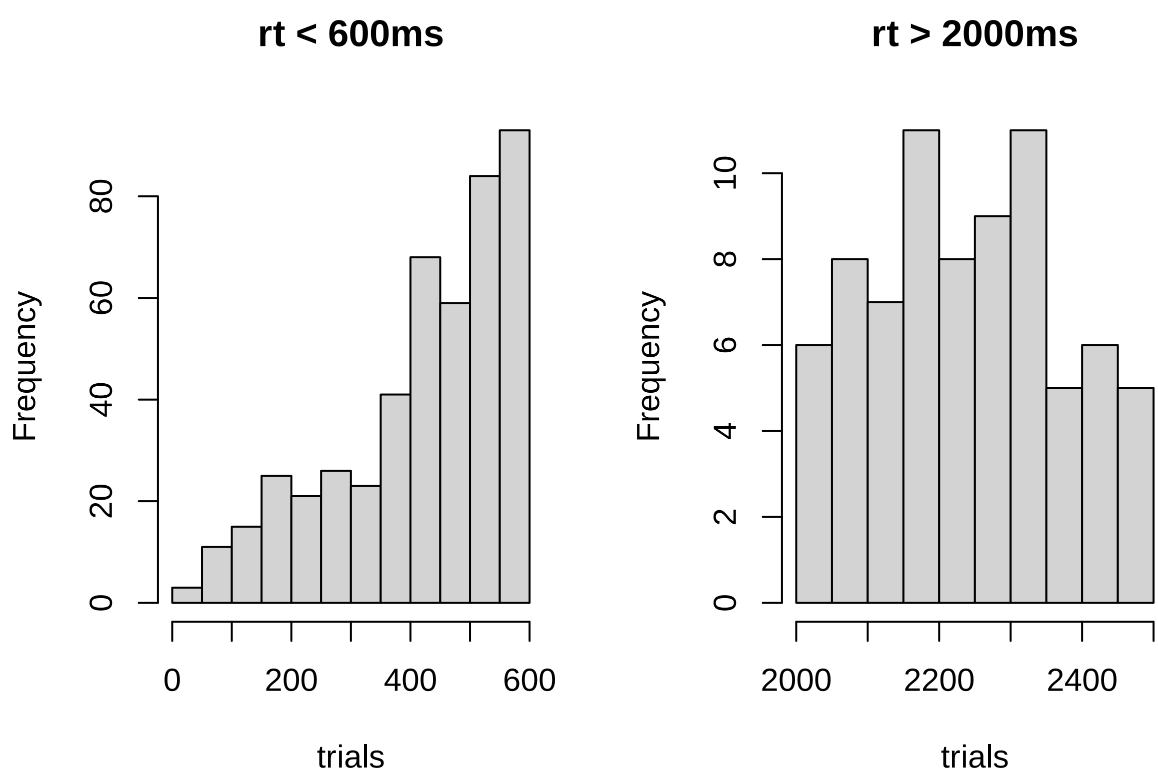
```r
length(unique(generalizationPL$subjID))
```

```
## [1] 80
```

```r
fl<- length(unique(generalizationPL[generalizationPL$learning=='FL',]$subjID))
lf<- length(unique(generalizationPL[generalizationPL$learning=='LF',]$subjID))
```

We have 41 for feature-label learning, and 39 for label-feature learning.

```r
par(mfrow=c(1,2))
hist(generalizationPL[generalizationPL$rt<600,]$rt, main = 'rt < 600ms', xlab = 'trials');
hist(generalizationPL[generalizationPL$rt>2000,]$rt, main = 'rt > 2000ms', xlab = 'trials');
```



```r
par(mfrow=c(1,1))
```

```r
rm(fl,lf)
pictureLabel <- generalizationPL[!(is.na(generalizationPL$resp)),]

pictureLabel$acc <- 0;
pictureLabel[pictureLabel$category==1 & pictureLabel$resp=='dep',]$acc <- 1;

pictureLabel[pictureLabel$category==2 & pictureLabel$resp=='bim',]$acc <- 1;

pictureLabel[pictureLabel$category==3 & pictureLabel$resp=='tob',]$acc <- 1;
```

```r
n <- length(unique(pictureLabel$subjID))
nrows <- (nrow(generalizationPL)) - (nrow(pictureLabel))
```

```r
sort(unique(pictureLabel$subjID))-> subjs;
sort(unique(generalizationPL$subjID)) ->totsubjs;

subjmissed<- setdiff(totsubjs, subjs);

rm(subjs, totsubjs);
```

We have 79 participants in this task, this is -1 compared to our total number of participants. The subject(s) that didn't answer at all the task is: 1420171. We have lost also 136 responses, that is 7.0833333 over the total: 1920.

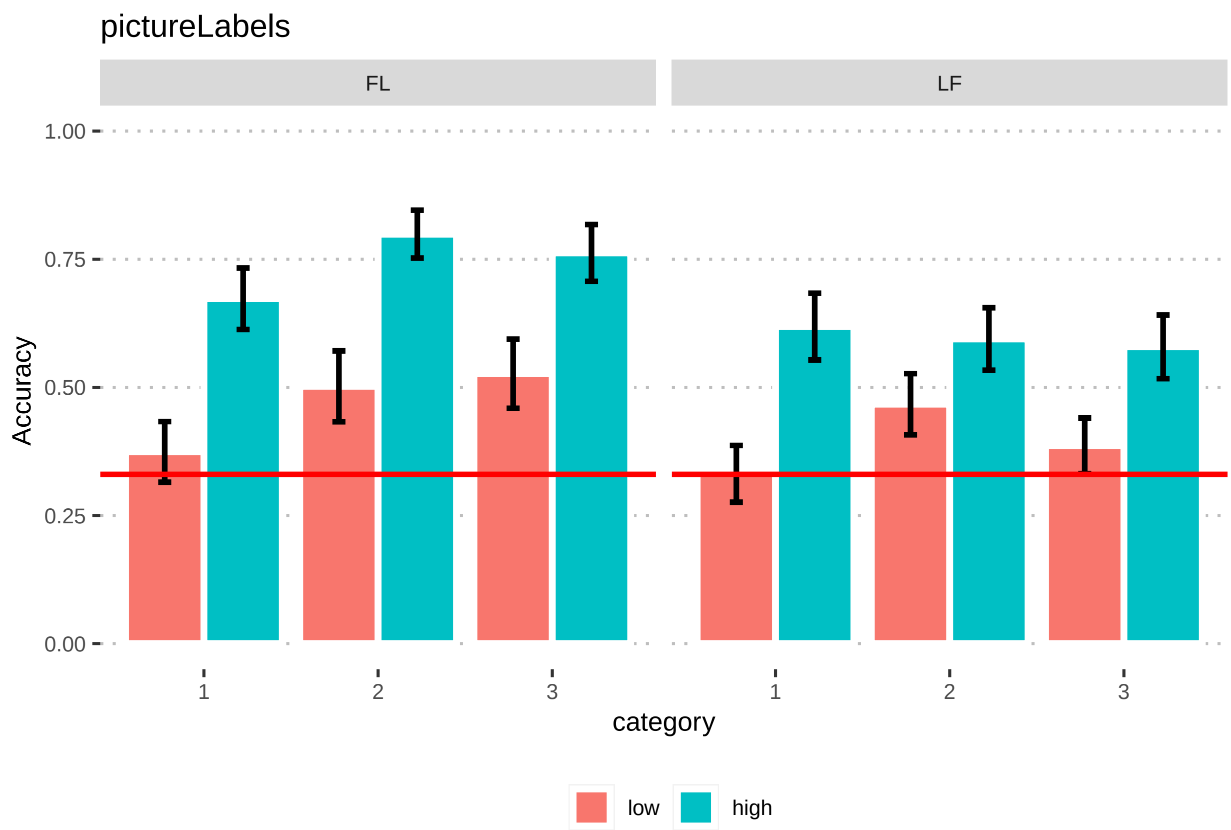
Calculate the proportion of correct in each condition:

```r
rm(n, subjmissed, nrows)

ss_prop<-aggregate(acc ~ frequency+category+subjID+learning,
                   data = pictureLabel[pictureLabel$rt > 200,], FUN = mean)
```

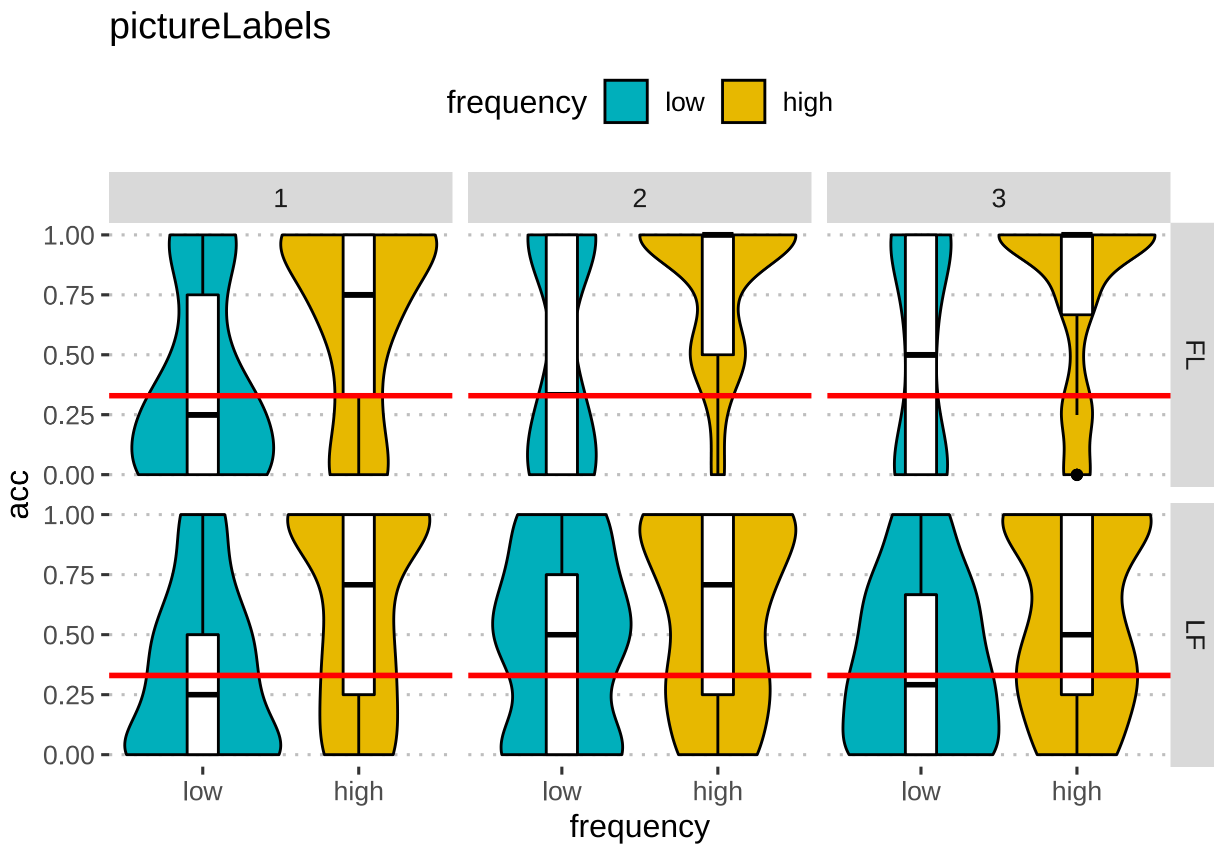
Plot aggregated over subjs. To see accuracy distributed over categories.

```r
ms <- ss_prop %>%
  group_by( category, frequency, learning) %>%
  summarise(n=n(),
    mean=mean(acc),
    sd=sd(acc)
  ) %>%
  mutate( se=sd/sqrt(n))  %>%
  mutate( ci=se * qt((1-0.05)/2 + .5, n-1))

ms$frequency <- as.factor(ms$frequency)
plyr::revalue(ms$frequency, c("25"="low"))-> ms$frequency;
plyr::revalue(ms$frequency, c("75"="high"))-> ms$frequency;

ggplot(aes(x = category, y = mean, fill = frequency), data = ms) +
  facet_grid( . ~ learning) +
  geom_bar(stat = "identity", color='white', position=position_dodge(), size=1.2) +
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.15, size=1,position=position_dodge(.9)) +
```

```r
  ylab("Accuracy ") +
  xlab("category") +
  ggtitle('pictureLabels') +
  coord_cartesian(ylim = c(0, 1))+
  ggpubr::theme_pubclean() +
  theme(legend.position="bottom", legend.title = element_blank()) +
  theme(text = element_text(size=10)) +
  geom_hline(yintercept = .33, col='red', lwd=1);
```

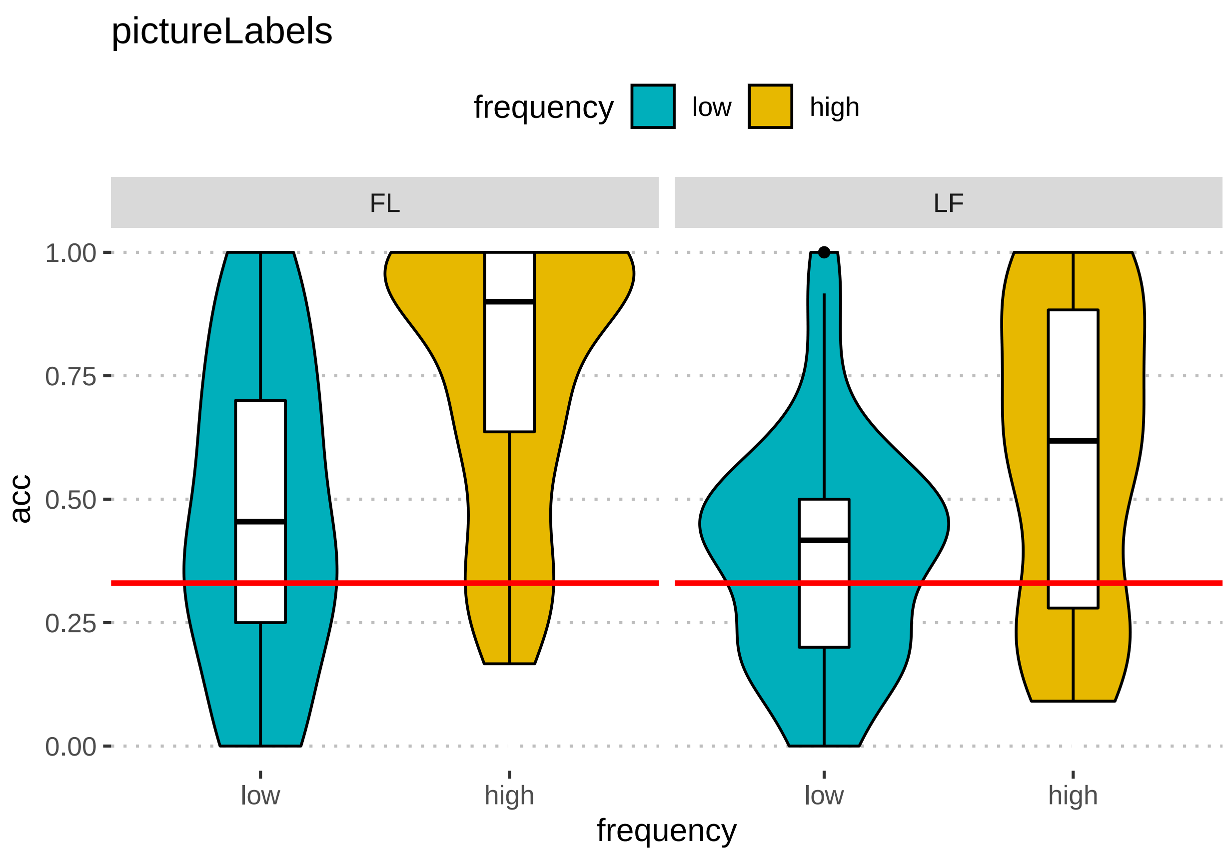



```r
df <- aggregate(acc ~ subjID+frequency+learning+category,
               data = pictureLabel[pictureLabel$rt > 200,], mean)
df$frequency <- as.factor(df$frequency)
plyr::revalue(df$frequency, c("25"="low"))-> df$frequency;
plyr::revalue(df$frequency, c("75"="high"))-> df$frequency;

ggviolin(df, x = "frequency", y = "acc", fill = "frequency",
         palette = c("#00AFBB", "#E7B800"),
         add = "boxplot",
         add.params = list(fill = "white"),
         trim=TRUE) +
        ggtitle('pictureLabels') +
        facet_grid( learning ~ category) +
        theme_pubclean()+
  geom_hline(yintercept = .33, col='red', lwd=1);
```

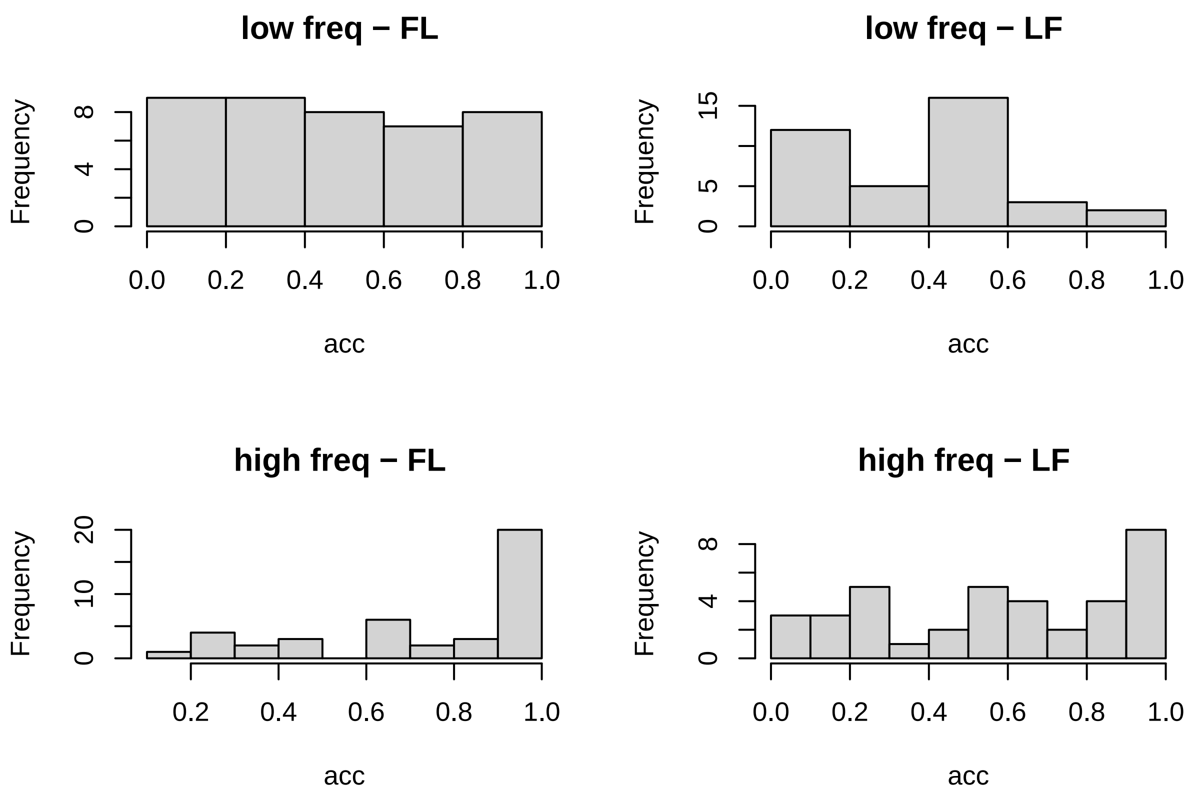


Let's see how participants scored for the high/low frequency:

```
df <- aggregate(acc ~ subjID+frequency+learning,
                data = pictureLabel[pictureLabel$rt > 200,], mean)
df$frequency <- as.factor(df$frequency)
plyr::revalue(df$frequency, c("25"="low"))-> df$frequency;
plyr::revalue(df$frequency, c("75"="high"))-> df$frequency;

ggviolin(df, x = "frequency", y = "acc", fill = "frequency",
         palette = c("#00AFBB", "#E7B800"),
         add = "boxplot",
         add.params = list(fill = "white"),
         trim=TRUE) +
        ggtitle('pictureLabels') +
        facet_grid( . ~ learning) +
        theme_pubclean()+
  geom_hline(yintercept = .33, col='red', lwd=1);
```




```r
par(mfrow=c(2,2))
hist(df[df$frequency=='low' & df$learning=='FL',]$acc, xlab = 'acc', main = 'low freq - FL ')
hist(df[df$frequency=='low' & df$learning=='LF',]$acc, xlab = 'acc', main = 'low freq - LF ')
hist(df[df$frequency=='high' & df$learning=='FL',]$acc, xlab = 'acc', main = 'high freq - FL ')
hist(df[df$frequency=='high' & df$learning=='LF',]$acc, xlab = 'acc', main = 'high freq - LF ')
```




```
par(mfrow=c(1,1))
```

```
#barPlot aggregated over categories:

ms <- aggregate(acc ~ subjID+frequency+learning,
                data=pictureLabel[pictureLabel$rt > 200,], FUN= mean)

df<- ms %>%
  group_by(frequency, learning)%>%
  summarise(
    mean = mean(acc),
    sd = sd(acc),
    n = n()) %>%
  mutate( se=sd/sqrt(n))  %>%
  mutate( ci=se * qt((1-0.05)/2 + .5, n-1))

df$frequency <- as.factor(df$frequency)
plyr::revalue(df$frequency, c("25"="low"))-> df$frequency;
plyr::revalue(df$frequency, c("75"="high"))-> df$frequency;


pl<-ggplot(aes(x = frequency, y = mean, fill = frequency), data = df) +
  facet_grid( . ~ learning) +
  geom_bar(stat = "identity", color='white', position=position_dodge(), size=1.2) +
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.15, size=1,position=position_dodge(.9)) +
  ylab("Accuracy ") +
```

```r
  xlab("frequency") +
  ggtitle('pictureLabels') +
  coord_cartesian(ylim = c(0, 1))+
  ggpubr::theme_pubclean() +
  theme(legend.position="bottom", legend.title = element_blank()) +
  theme(text = element_text(size=10)) +
  geom_hline(yintercept = .33, col='red', lwd=1);
```

Task 2: from label to pictures

Let's check now the generalizaton from label to pictures:

```r
length(unique(generalizationLP$subjID))
```

```
## [1] 80
```

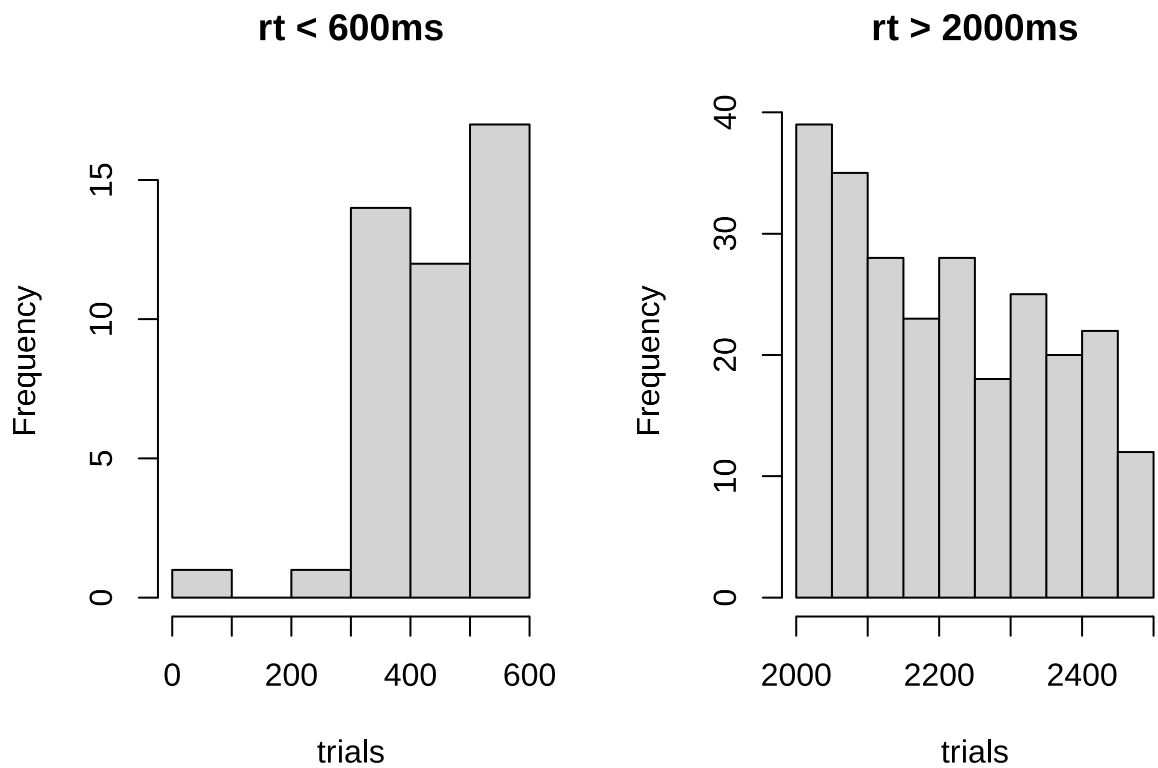
```r
fl<- length(unique(generalizationLP[generalizationLP$learning=='FL',]$subjID))
lf<- length(unique(generalizationLP[generalizationLP$learning=='LF',]$subjID))
```

We have 41 for feature-label learning, and 39 for label-feature learning.

```r
rm(fl,lf)
labelPicture <- generalizationLP[!(is.na(generalizationLP$resp)),]
n<- length(unique(labelPicture$subjID))
nrows <- (nrow(generalizationLP)) - (nrow(labelPicture))

sort(unique(labelPicture$subjID))-> subjs;
sort(unique(generalizationLP$subjID)) ->totsubjs;

subjmissed<- setdiff(totsubjs, subjs);
```

Great, we have 80 participants in this task, so -0, and we have missed 179 over the total 1920, that is 9.3229167. The subject(s) that missed completely the task is: .

```r
par(mfrow=c(1,2))
hist(generalizationLP[generalizationLP$rt<600,]$rt, main = 'rt < 600ms', xlab = 'trials');
hist(generalizationLP[generalizationLP$rt>2000,]$rt, main = 'rt > 2000ms', xlab = 'trials');
```




```
par(mfrow=c(1,1))
```

```
rm(n, nrows, subjs, totsubjs);
labelPicture$acc <- 0;
labelPicture[labelPicture$category==1 & labelPicture$label=='dep',]$acc <- 1;
labelPicture[labelPicture$category==2 & labelPicture$label=='bim',]$acc <- 1;
labelPicture[labelPicture$category==3 & labelPicture$label=='tob',]$acc <- 1;
```

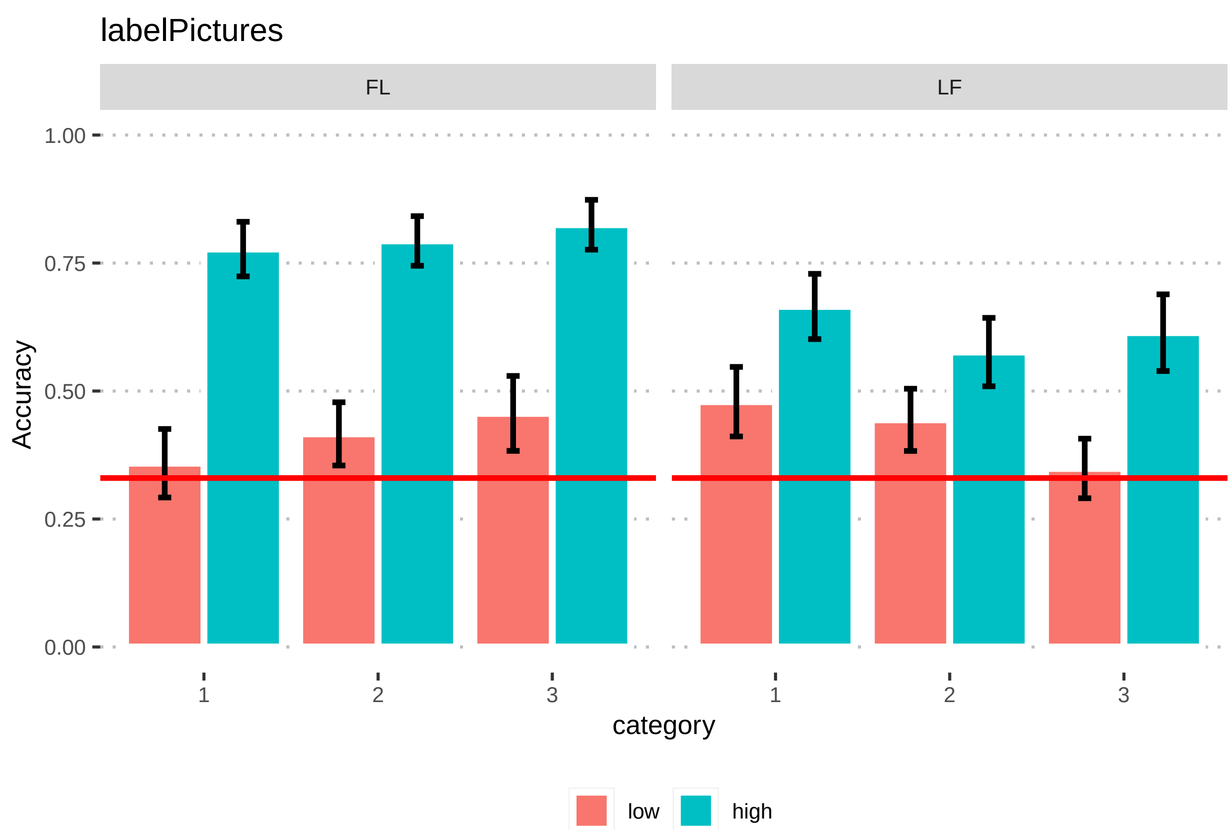
Calculate the proportion of correct in each condition

```
rm(subjmissed)
ss_prop<-aggregate(acc ~ frequency+category+subjID+learning,
                   data = labelPicture[labelPicture$rt > 200 ,], FUN = mean)
```

Plot aggregated over subjs. To see accuracy distributed over categories.

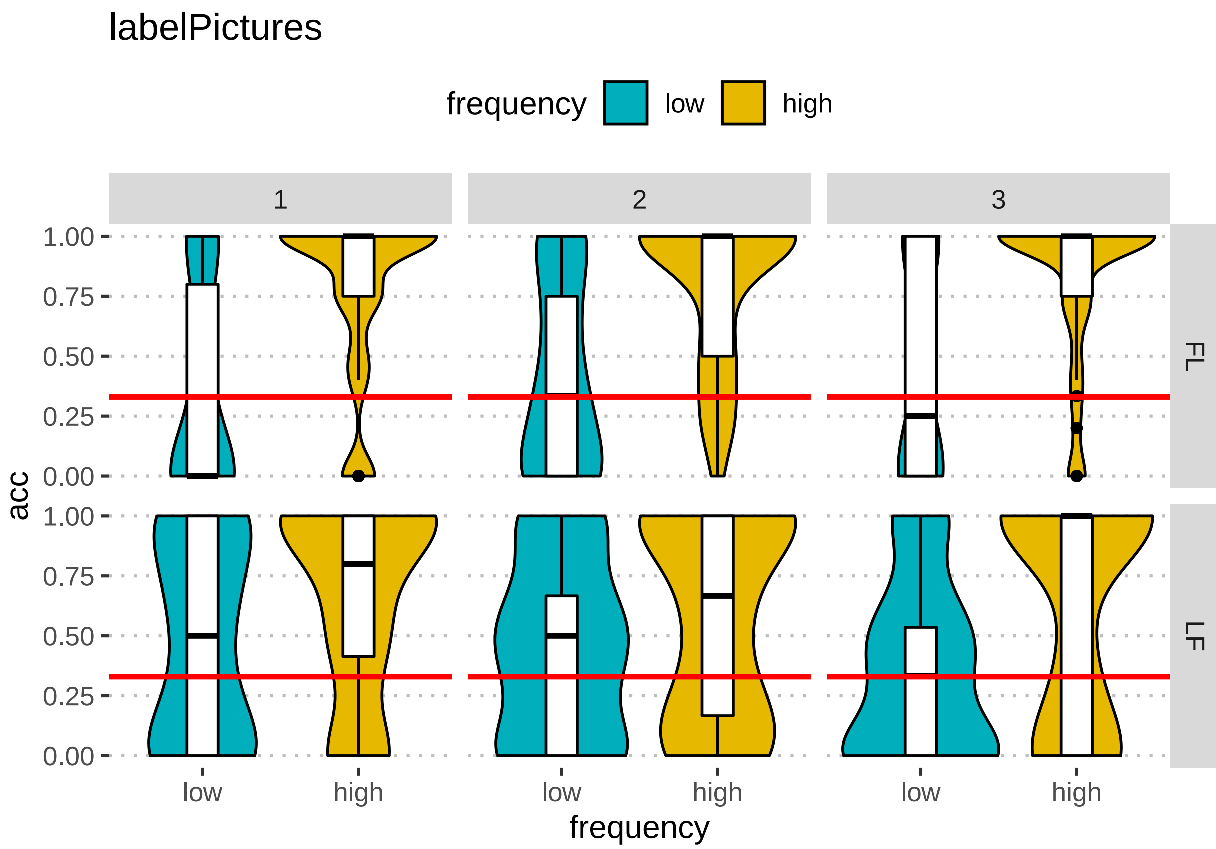
```
ms <- ss_prop %>%
  group_by(category, frequency, learning) %>%
  summarise(
    n=n(),
    mean=mean(acc),
    sd=sd(acc)
  ) %>%
  mutate( se=sd/sqrt(n))  %>%
```

```
  mutate( ci=se * qt((1-0.05)/2 + .5, n-1))

ms$frequency <- as.factor(ms$frequency)
plyr::revalue(ms$frequency, c("25"="low"))-> ms$frequency;
plyr::revalue(ms$frequency, c("75"="high"))-> ms$frequency;

ggplot(aes(x = category, y = mean, fill = frequency), data = ms) +
  facet_grid( . ~ learning) +
  geom_bar(stat = "identity", color='white', position=position_dodge(), size=1.2) +
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.15, size=1,position=position_dodge(.9)) +
  ylab("Accuracy ") +
  xlab("category") +
  ggtitle('labelPictures') +
  coord_cartesian(ylim = c(0, 1))+
  ggpubr::theme_pubclean() +
  theme(legend.position="bottom", legend.title = element_blank()) +
  theme(text = element_text(size=10)) +
  geom_hline(yintercept = .33, col='red', lwd=1);
```




```
ms <- aggregate(acc ~ subjID+frequency+learning+category,
                data = labelPicture[labelPicture$rt > 200,], mean)

ms$frequency <- as.factor(ms$frequency)
plyr::revalue(ms$frequency, c("25"="low"))-> ms$frequency;
plyr::revalue(ms$frequency, c("75"="high"))-> ms$frequency;
```

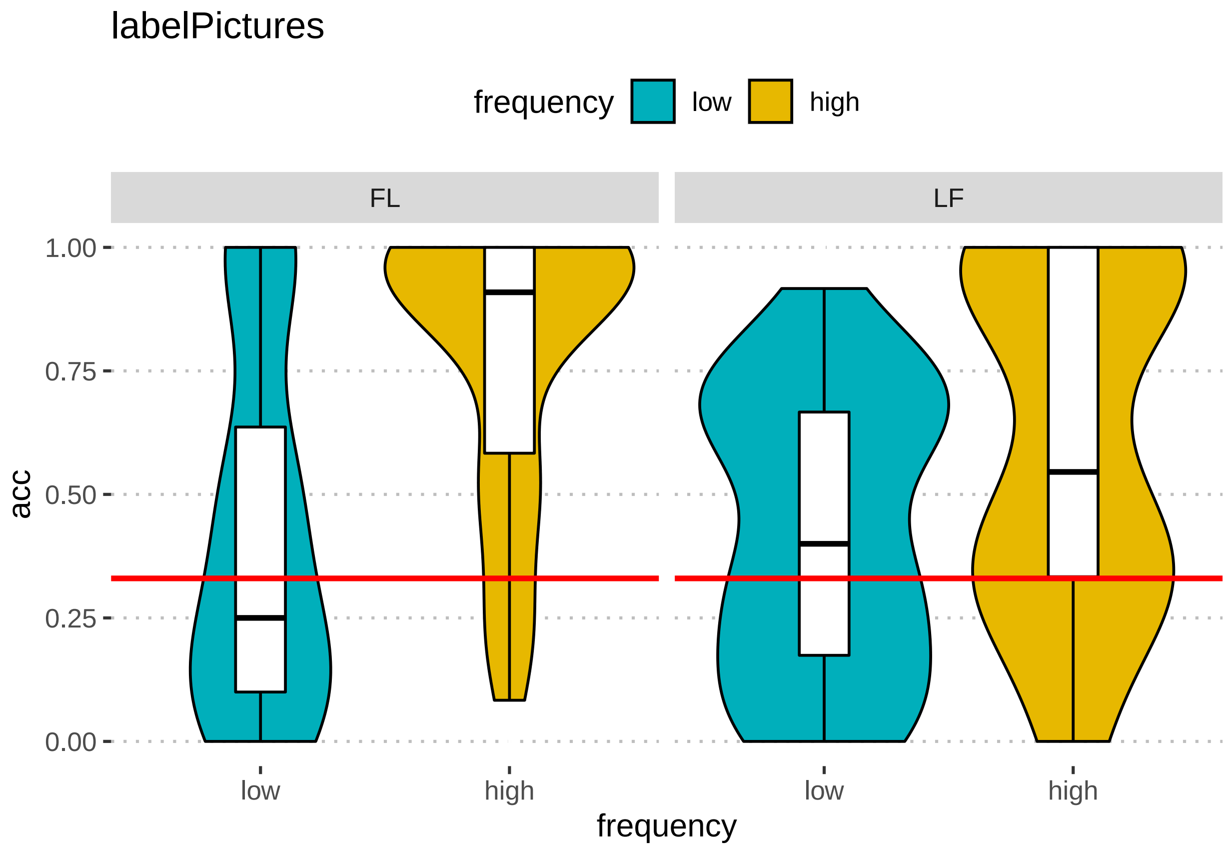
```
ggviolin(ms, x = "frequency", y = "acc", fill = "frequency",
         palette = c("#00AFBB", "#E7B800"),
         add = "boxplot",
         add.params = list(fill = "white"),
         trim=TRUE) +
         ggtitle('labelPictures') +
        facet_grid( learning ~ category) +
        theme_pubclean()+
  geom_hline(yintercept = .33, col='red', lwd=1);
```




```
#rm(ms, ss_prop)
```

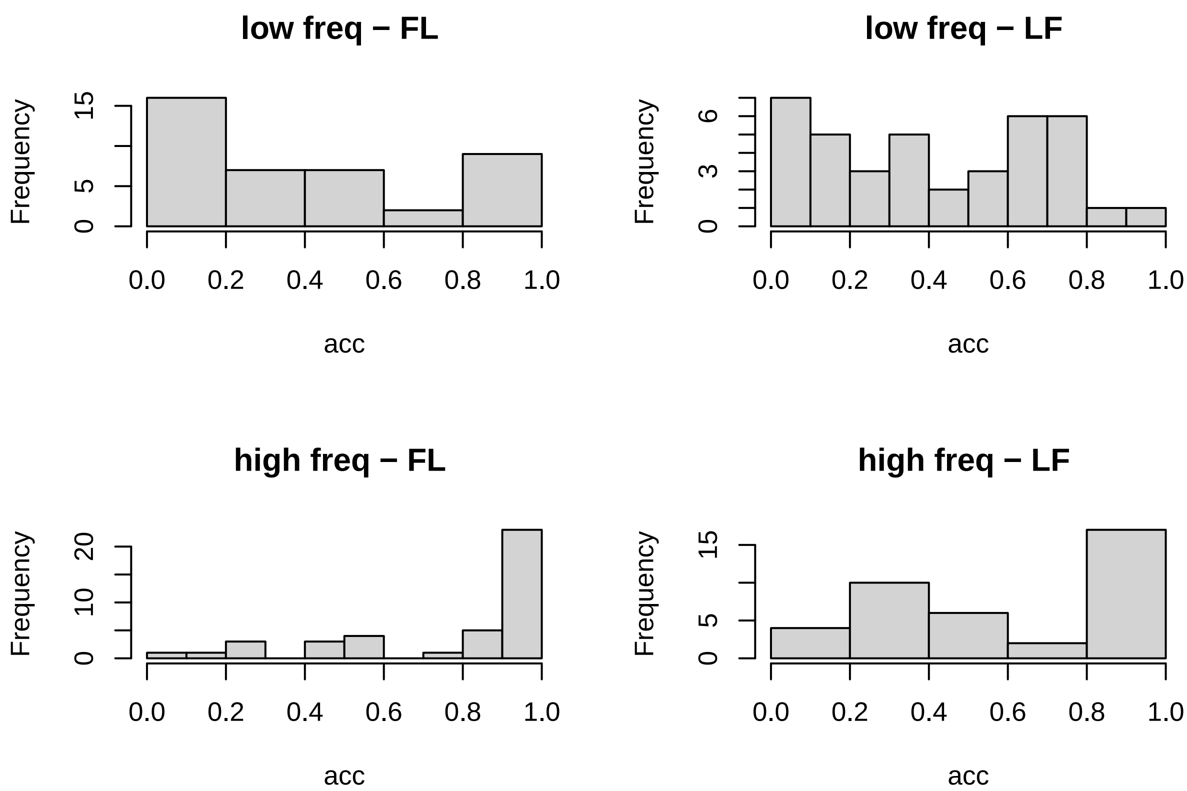
```
ms <- aggregate(acc ~ subjID+frequency+learning,
                data = labelPicture[labelPicture$rt > 200,], mean)

ms$frequency <- as.factor(ms$frequency)
plyr::revalue(ms$frequency, c("25"="low"))-> ms$frequency;
plyr::revalue(ms$frequency, c("75"="high"))-> ms$frequency;

ggviolin(ms, x = "frequency", y = "acc", fill = "frequency",
         palette = c("#00AFBB", "#E7B800"),
         add = "boxplot",
         add.params = list(fill = "white"),
         trim=TRUE) +
```

```r
        ggtitle('labelPictures') +
       facet_grid( . ~ learning) +
       theme_pubclean()+
  geom_hline(yintercept = .33, col='red', lwd=1);
```




```r
#rm(ms, ss_prop)
```

```r
par(mfrow=c(2,2))
hist(ms[ms$frequency=='low' & ms$learning=='FL',]$acc, xlab = 'acc', main = 'low freq - FL ')
hist(ms[ms$frequency=='low' & ms$learning=='LF',]$acc, xlab = 'acc', main = 'low freq - LF ')
hist(ms[ms$frequency=='high' & ms$learning=='FL',]$acc, xlab = 'acc', main = 'high freq - FL ')
hist(ms[ms$frequency=='high' & ms$learning=='LF',]$acc, xlab = 'acc', main = 'high freq - LF ')
```



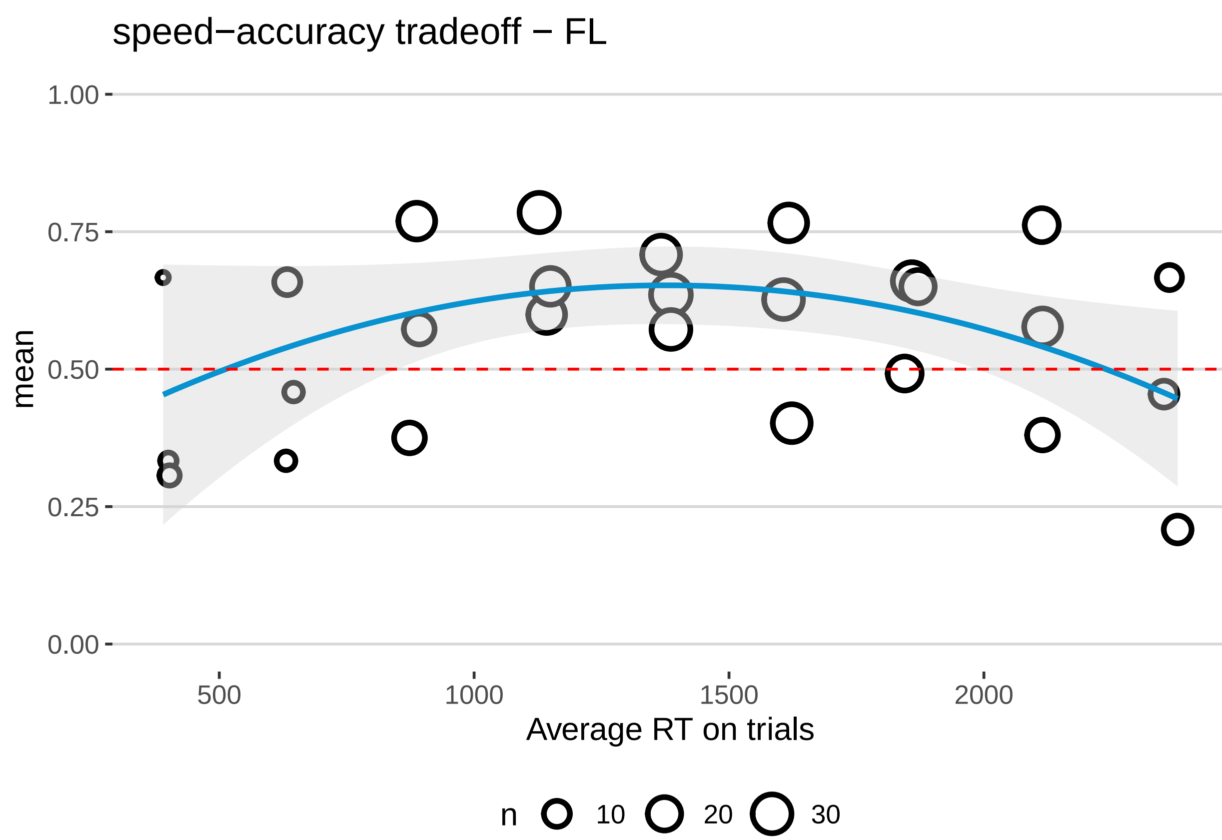

```
par(mfrow=c(1,1))
```

Comparison with both task 1 and 2

Inspection of the speed-accuracy trade-off:

```
rt_range <- 2500
n_bins <- 10
break_seq <- seq(0, rt_range, rt_range/n_bins)

timeslice_range <- labelPicture[labelPicture$rt > 200 ,] %>%
  filter(learning == "FL") %>%
  dplyr::mutate(RT_bin = cut(rt, breaks = break_seq)) %>%
  dplyr::group_by(RT_bin, category) %>%
  dplyr::mutate(RT_bin_avg = mean(rt, na.rm = T))

count_range <- timeslice_range %>%
  group_by(RT_bin, category) %>%
  summarise(subjcount = n_distinct(subjID), totalcount = n())

timeslice_range <- timeslice_range %>%
  dplyr::group_by(RT_bin_avg, category, subjID) %>%
  dplyr::summarise(ss_acc = mean(acc, na.rm=T)) %>%
  dplyr::group_by(RT_bin_avg, category) %>%
  dplyr::summarise(mean = mean(ss_acc),
```

```
            n = n())

ggplot(aes(x=RT_bin_avg, y=mean, weight = n),
           data = timeslice_range) +
  geom_point(aes(size = n), shape = 21, fill = "white", stroke = 1.5) +
  geom_smooth(method = "lm", formula = y ~ poly(x,2), se = TRUE, color = "#0892d0", fill = "lightgray")
  geom_hline(yintercept = 0.5, lty = "dashed", color = 'red') +
  coord_cartesian(ylim = c(0, 1))+
  ggthemes::theme_hc()+
  xlab("Average RT on trials") +
  ggtitle('speed-accuracy tradeoff - FL')
```

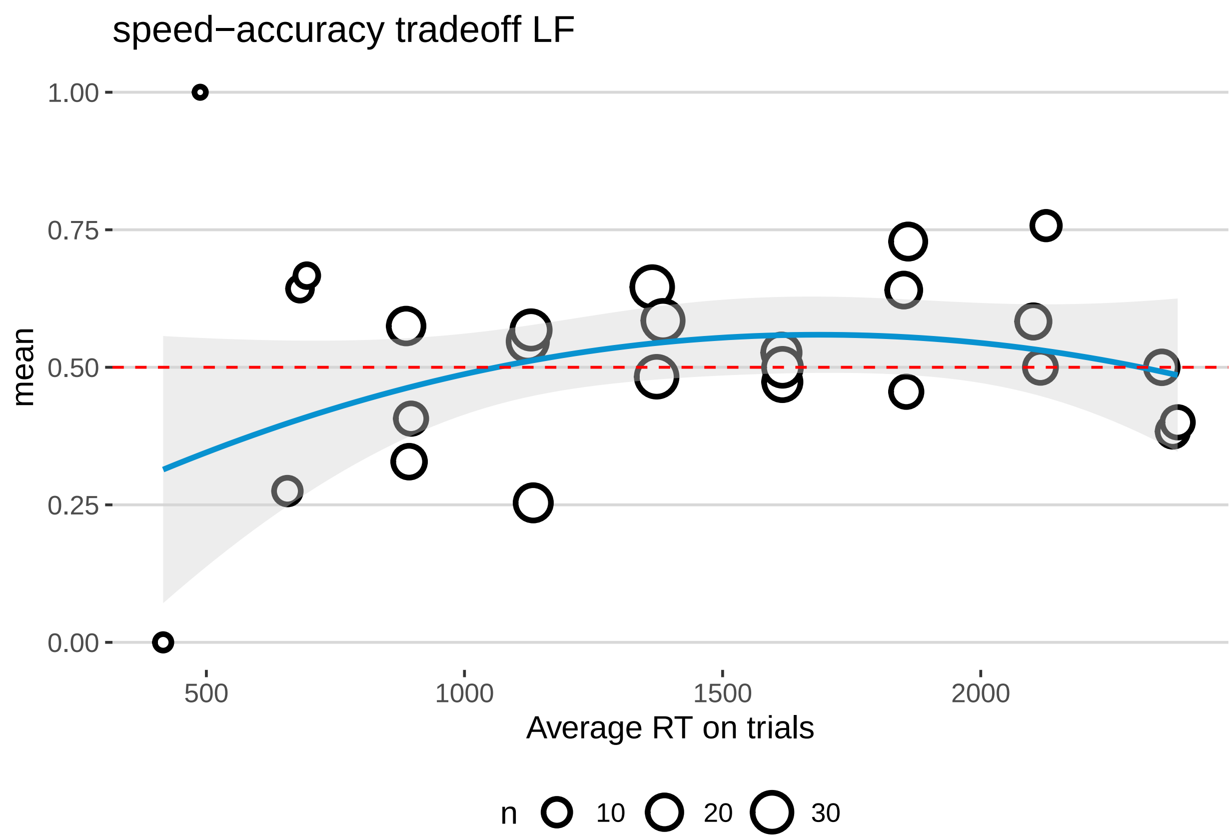



```
  ylab("Proportion Correct")
```

```
## $y
## [1] "Proportion Correct"
##
## attr(,"class")
## [1] "labels"
```

```
rt_range <- 2500
n_bins <- 10
break_seq <- seq(0, rt_range, rt_range/n_bins)
```

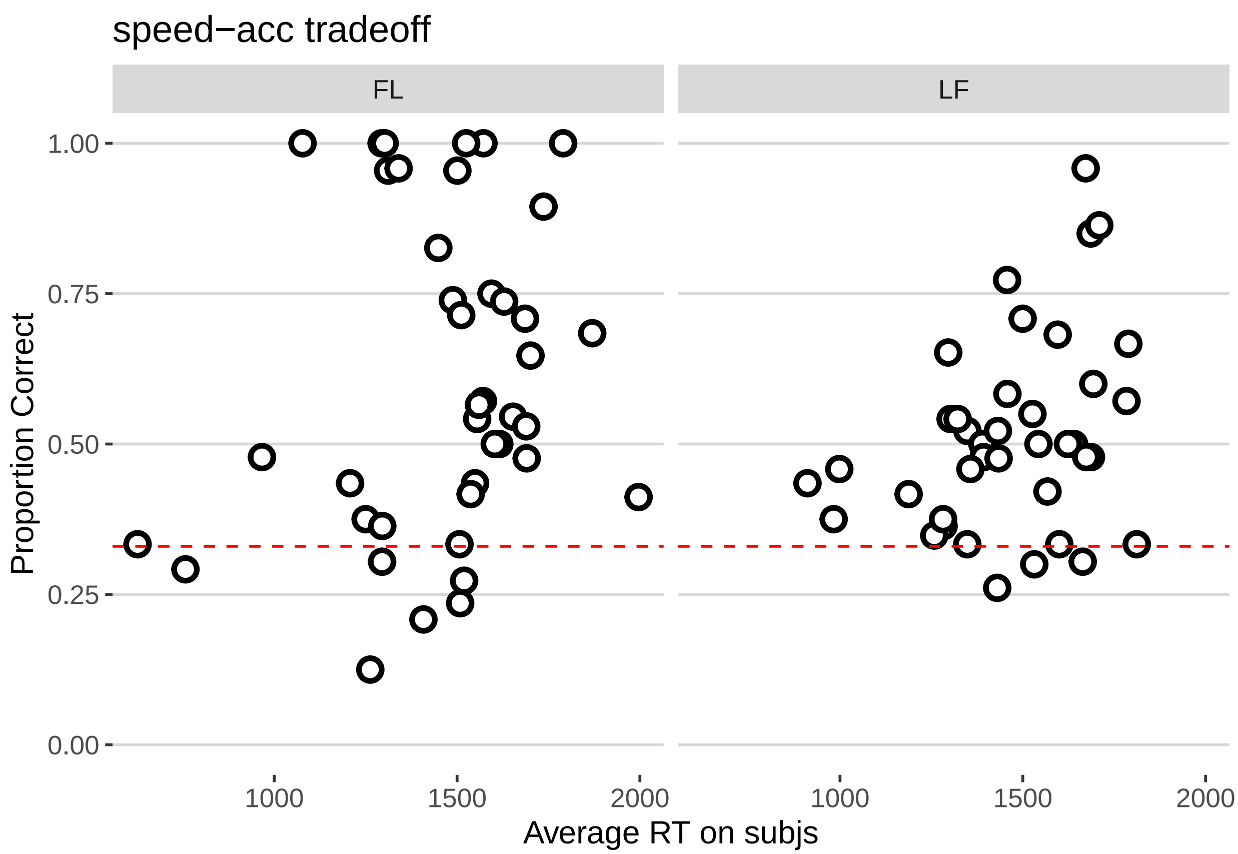
```
timeslice_range <- labelPicture[labelPicture$rt > 200 ,] %>%
  filter(learning == "LF") %>%
  dplyr::mutate(RT_bin = cut(rt, breaks = break_seq)) %>%
  dplyr::group_by(RT_bin, category) %>%
  dplyr::mutate(RT_bin_avg = mean(rt, na.rm = T))

count_range <- timeslice_range %>%
  group_by(RT_bin, category) %>%
  summarise(subjcount = n_distinct(subjID), totalcount = n())

timeslice_range <- timeslice_range %>%
  dplyr::group_by(RT_bin_avg, category, subjID) %>%
  dplyr::summarise(ss_acc = mean(acc, na.rm=T)) %>%
  dplyr::group_by(RT_bin_avg, category) %>%
  dplyr::summarise(mean = mean(ss_acc),
           n = n())

ggplot(aes(x=RT_bin_avg, y=mean, weight = n),
          data = timeslice_range) +
  geom_point(aes(size = n), shape = 21, fill = "white", stroke = 1.5) +
  geom_smooth(method = "lm", formula = y ~ poly(x,2), se = TRUE, color = "#0892d0", fill = "lightgray")
  geom_hline(yintercept = 0.5, lty = "dashed", color = 'red') +
  coord_cartesian(ylim = c(0, 1))+
  ggthemes::theme_hc()+
  xlab("Average RT on trials") +
  ggtitle('speed-accuracy tradeoff LF')
```




```
  ylab("Proportion Correct")
```

```
## $y
## [1] "Proportion Correct"
##
## attr(,"class")
## [1] "labels"
```

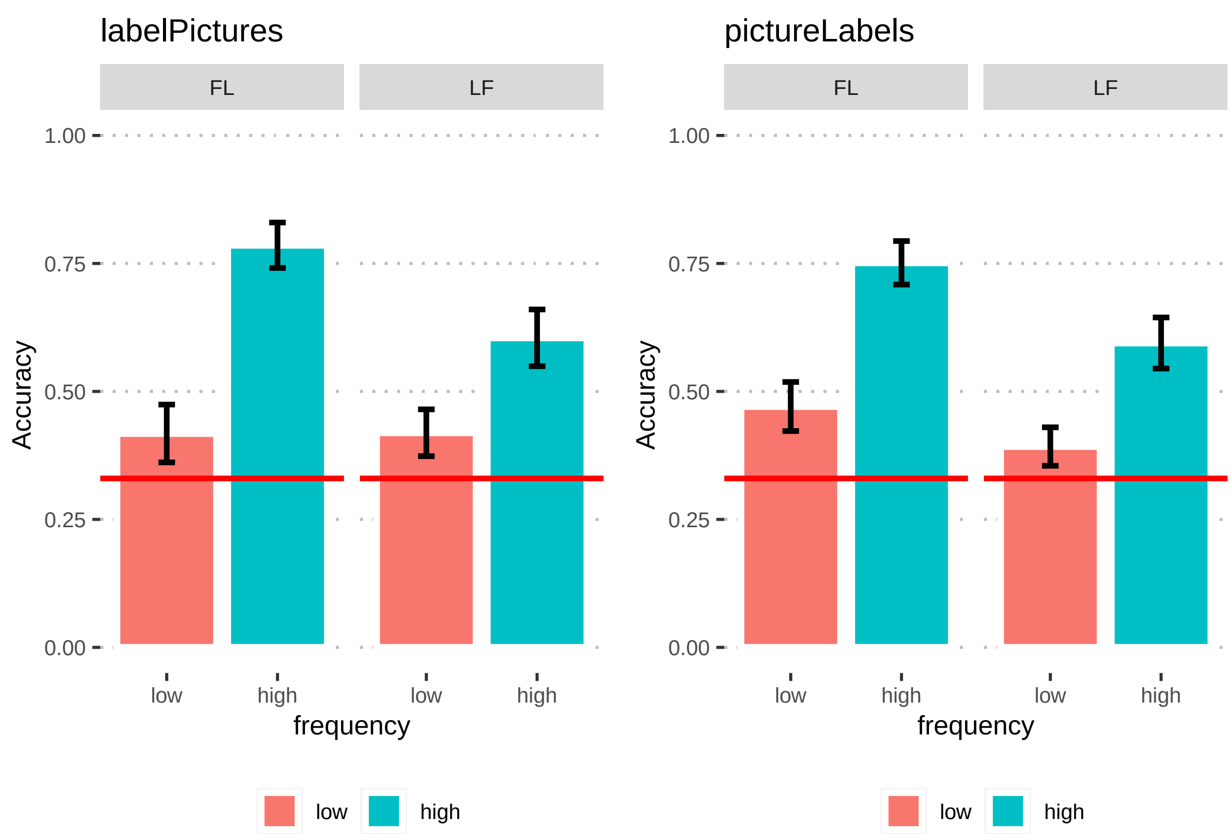
```
aggregate(acc ~ subjID+learning, labelPicture[labelPicture$rt > 200 ,], mean)-> speedacc
aggregate(rt ~ subjID+learning, labelPicture[labelPicture$rt > 200,], mean)-> speedacc2
merge(speedacc, speedacc2, by =  c("subjID", "learning"))-> speedacc

ggplot(aes(x=rt, y=acc),
           data = speedacc) +
  facet_grid( . ~ learning) +
  geom_point( shape = 21, fill = "white", size = 3, stroke = 1.5) +
  #geom_smooth(method = "lm", formula = y ~ poly(x,2), se = TRUE, color = "#0892d0", fill = "lightgray"
  geom_hline(yintercept = 0.33, lty = "dashed", color = 'red') +
  coord_cartesian(ylim = c(0, 1))+
  ggthemes::theme_hc()+
  xlab("Average RT on subjs") +
  ylab("Proportion Correct") +
  ggtitle("speed-acc tradeoff")
```




```
ms <- aggregate(acc ~ subjID+frequency+learning,
                data=labelPicture[labelPicture$rt > 200,], FUN= mean)

df<- ms %>%
  group_by(frequency, learning)%>%
  summarise(
    mean = mean(acc),
    sd = sd(acc),
    n = n()) %>%
  mutate( se=sd/sqrt(n))  %>%
  mutate( ci=se * qt((1-0.05)/2 + .5, n-1))

df$frequency <- as.factor(df$frequency)
plyr::revalue(df$frequency, c("25"="low"))-> df$frequency;
plyr::revalue(df$frequency, c("75"="high"))-> df$frequency;


lp<-ggplot(aes(x = frequency, y = mean, fill = frequency), data = df) +
  facet_grid( . ~ learning) +
  geom_bar(stat = "identity", color='white', position=position_dodge(), size=1.2) +
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.15, size=1,position=position_dodge(.9)) +
  ylab("Accuracy ") +
  xlab("frequency") +
  ggtitle("labelPictures") +
  coord_cartesian(ylim = c(0, 1))+
  ggpubr::theme_pubclean() +
```

```
  theme(legend.position="bottom", legend.title = element_blank()) +
  theme(text = element_text(size=10)) +
  geom_hline(yintercept = .33, col='red', lwd=1);
```

```
grid.arrange(lp, pl, ncol=2)
```

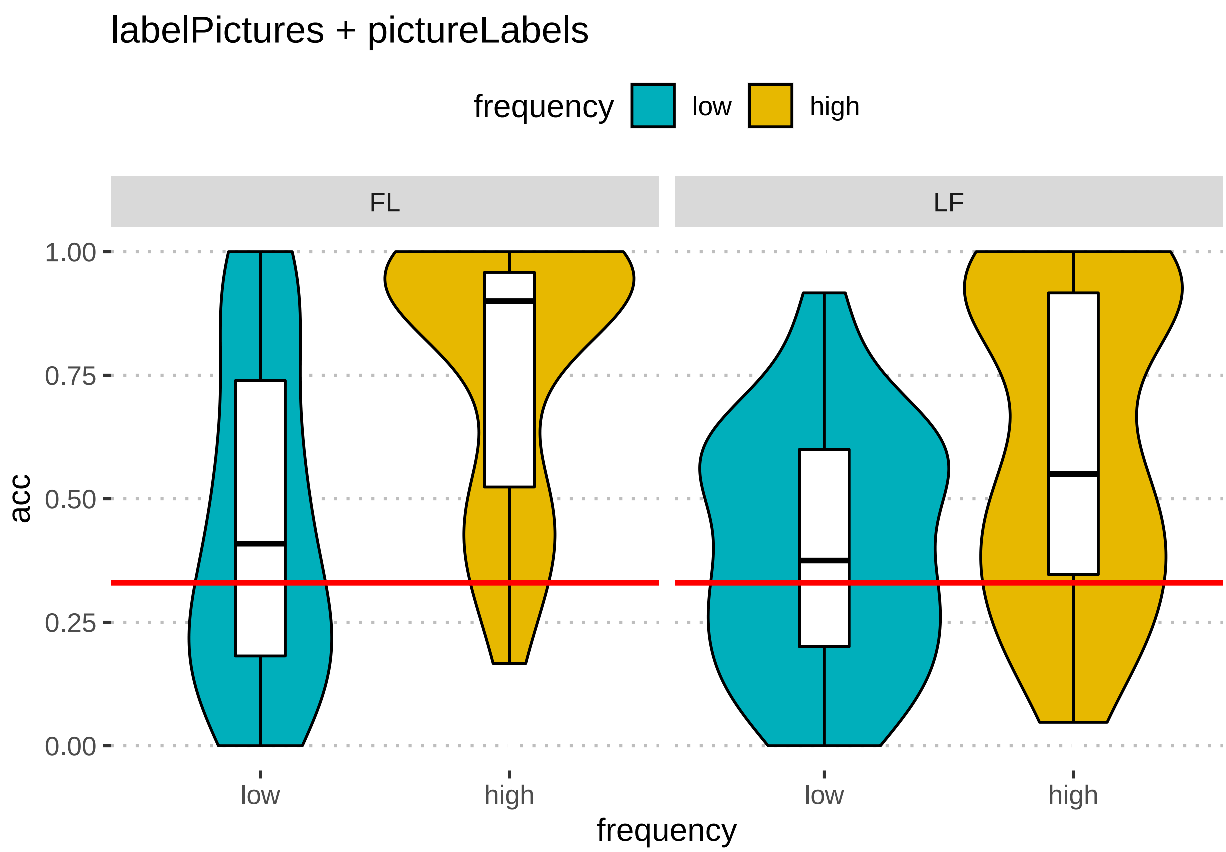


Barplots + violinPlots with data from both tasks:

```
rm(ms, lp, pl, df, ss_prop)
genTask <- rbind(labelPicture, pictureLabel)
```

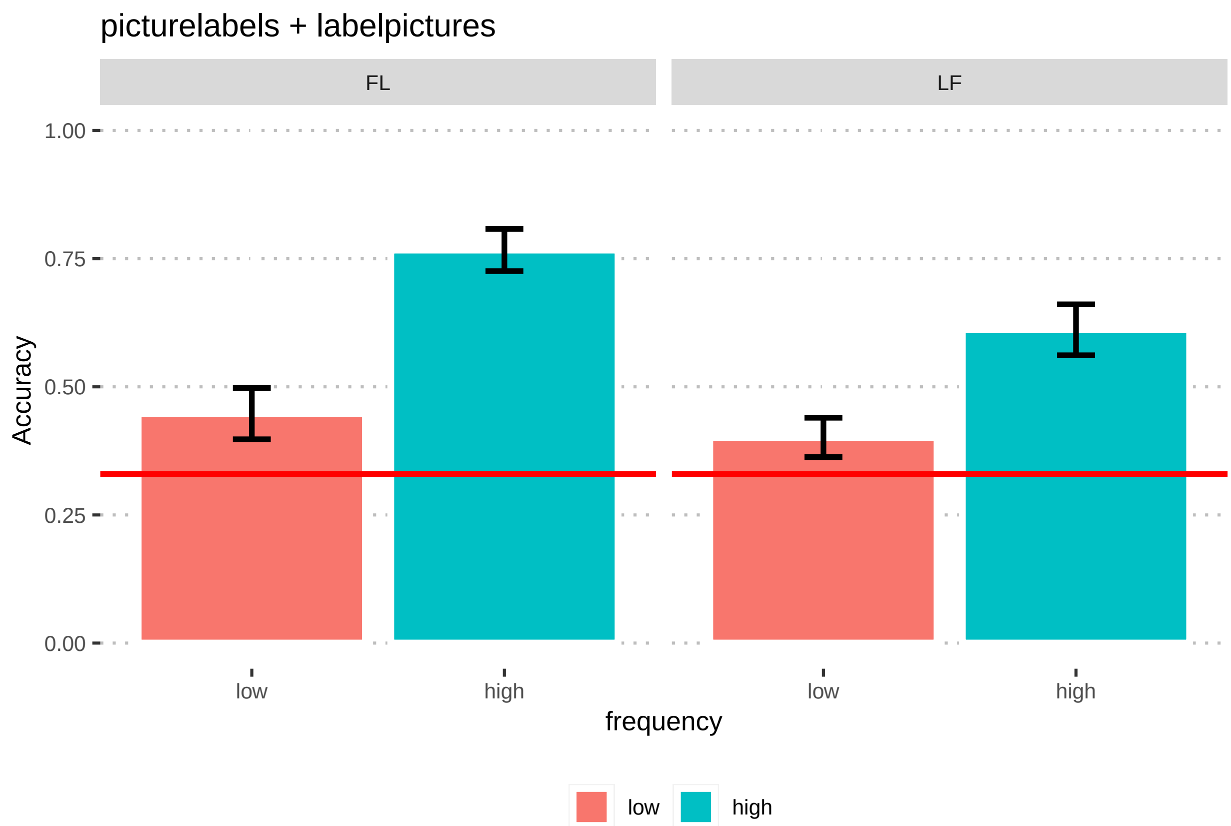
```
ms <- aggregate(acc ~ subjID+frequency+learning, data = genTask, mean)

ms$frequency <- as.factor(ms$frequency)
plyr::revalue(ms$frequency, c("25"="low"))-> ms$frequency;
plyr::revalue(ms$frequency, c("75"="high"))-> ms$frequency;

ggviolin(ms, x = "frequency", y = "acc", fill = "frequency",
         palette = c("#00AFBB", "#E7B800"),
         add = "boxplot",
         add.params = list(fill = "white"),
         trim=TRUE) +
        ggtitle('labelPictures + pictureLabels') +
        facet_grid( . ~ learning) +
```

```
      theme_pubclean()+
  geom_hline(yintercept = .33, col='red', lwd=1);
```




```
ms <- aggregate(acc ~ subjID+frequency+learning, data=genTask, FUN= mean)

df<- ms %>%
  group_by(frequency, learning)%>%
  summarise(
    mean = mean(acc),
    sd = sd(acc),
    n = n()) %>%
  mutate( se=sd/sqrt(n))  %>%
  mutate( ci=se * qt((1-0.05)/2 + .5, n-1))

df$frequency <- as.factor(df$frequency)
plyr::revalue(df$frequency, c("25"="low"))-> df$frequency;
plyr::revalue(df$frequency, c("75"="high"))-> df$frequency;


ggplot(aes(x = frequency, y = mean, fill = frequency), data = df) +
  facet_grid( . ~ learning) +
  geom_bar(stat = "identity", color='white', position=position_dodge(), size=1.2) +
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.15, size=1,position=position_dodge(.9)) +
  ylab("Accuracy ") +
  xlab("frequency") +
```

```
  ggtitle("labelPicture") +
  ggtitle('picturelabels + labelpictures') +
  coord_cartesian(ylim = c(0, 1))+
  ggpubr::theme_pubclean() +
  theme(legend.position="bottom", legend.title = element_blank()) +
  theme(text = element_text(size=10)) +
  geom_hline(yintercept = .33, col='red', lwd=1);
```



Task 3: Contingency judgement

```
length(unique(contingencyJudgement$subjID))
```

```
## [1] 80
```

```
fl<- length(unique(contingencyJudgement[contingencyJudgement$learning=='FL',]$subjID))
lf<- length(unique(contingencyJudgement[contingencyJudgement$learning=='LF',]$subjID))
```

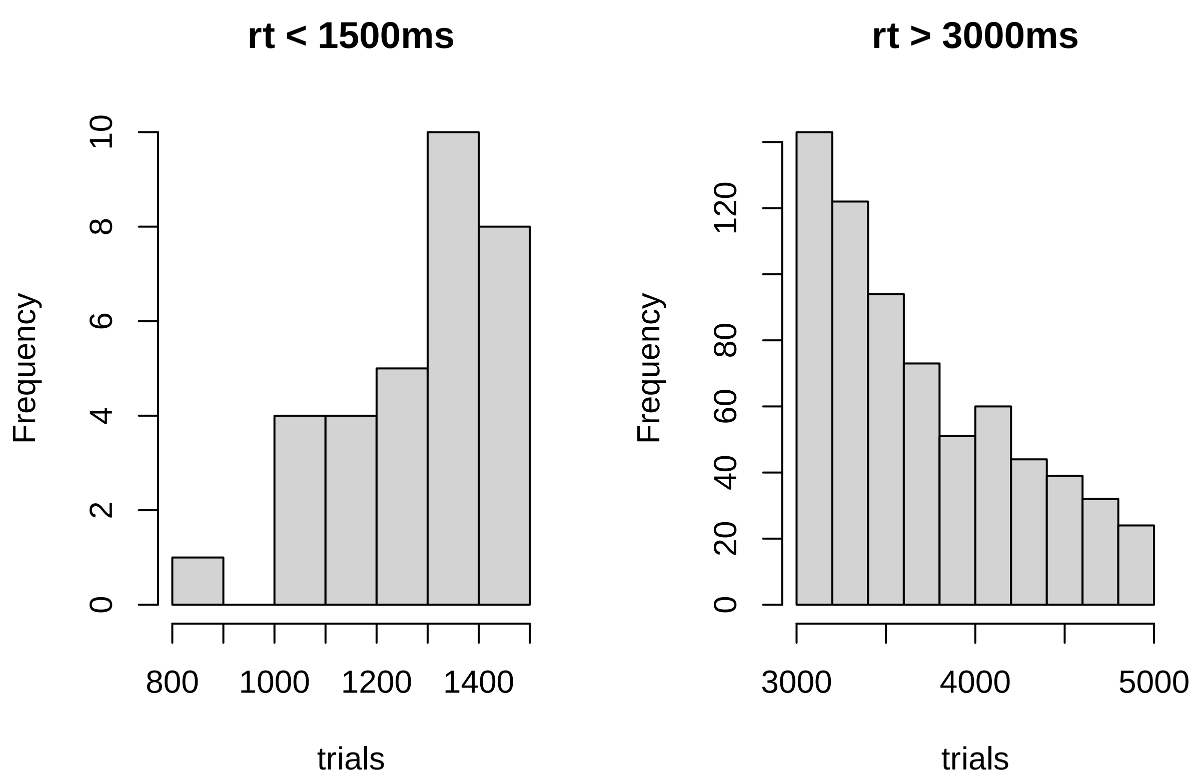
We have 41 for feature-label learning, and 39 for label-feature learning.

```
rm(fl,lf)
conjudge <- contingencyJudgement[!(is.na(contingencyJudgement$resp)),]
n<- length(unique(conjudge$subjID))
```

```r
nrows <- (nrow(contingencyJudgement)) - (nrow(conjudge))

sort(unique(conjudge$subjID))-> subjs;
sort(unique(contingencyJudgement$subjID)) ->totsubjs;

subjmissed<- setdiff(totsubjs, subjs);
```

We have 74 participants in this task, so -6, and we have missed 382 over the total 1920, that is 19.8958333. The subject(s) that missed completely the task is/are: 1414932, 1420171, 1420199, 1422475, 1431960, 1431997.

```r
par(mfrow=c(1,2))
hist(conjudge[conjudge$rt<1500,]$rt, main = 'rt < 1500ms', xlab = 'trials');
hist(conjudge[conjudge$rt>3000,]$rt, main = 'rt > 3000ms', xlab = 'trials');
```

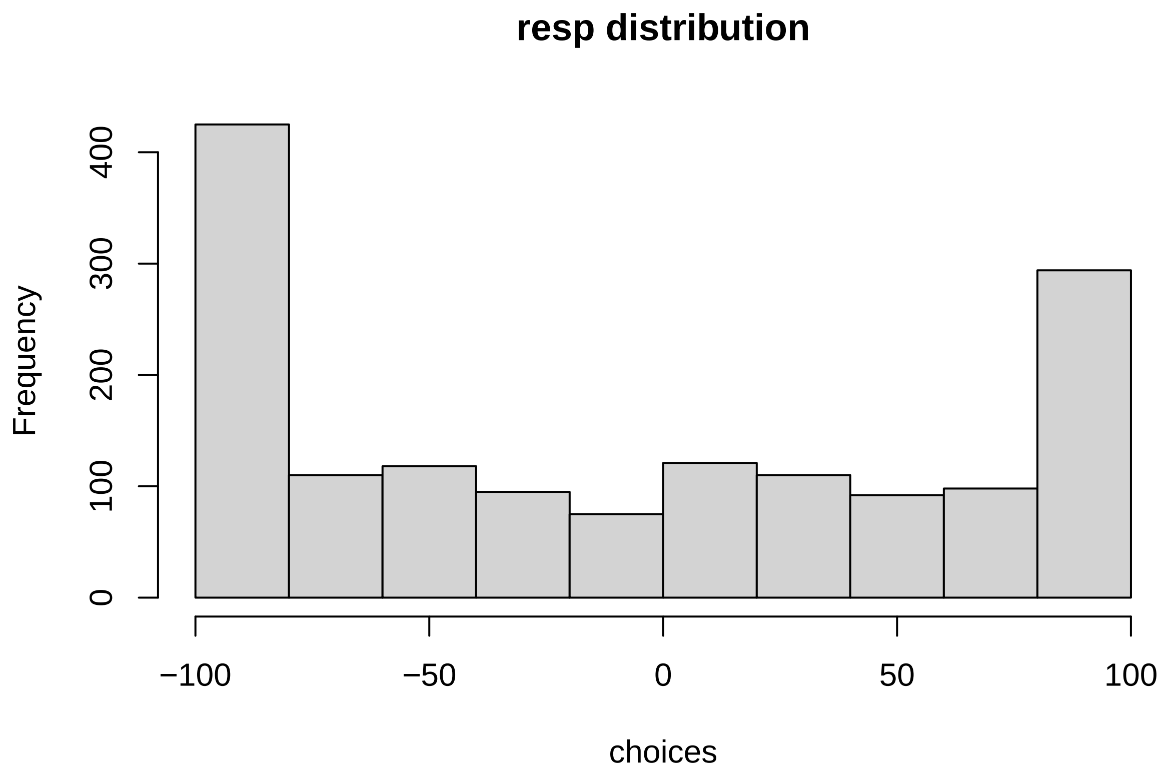



```r
par(mfrow=c(1,1))
```

Resp is coded as factor, need to correct this:

```r
as.numeric(levels(conjudge$resp))[conjudge$resp]-> conjudge$resp
```

```r
hist(conjudge$resp, main = 'resp distribution', xlab = 'choices')
```



Ok, here we don't have right or wrong answers, but we are more interested in take a look how the participants rated the fribble label association:

```
aggregate(resp ~ category, data = conjudge, FUN = mean)
```

```
##   category      resp
## 1        1 -12.52183
## 2        2  -5.50000
## 3        3 -10.09728
```