

FLO replication - Preprocessing + analysis + results summary

Eva

4/3/2020

Contents

clean WS, set WD	2
Load functions from the lab repo	3
Check stimuli set	3
Load data	4
Check learning	6
How many trials per participant?	6
Did we assign our learning randomly every couple of people?	7
Check if fribbles are presented > 2 times:	11
Check whether there are fribbles presented only once:	11
Check the association between the fribbles and the labels (high and low frequency with the correct labels)	12
Check Testing	13
Check test 1: Generalization from picture to labels	13
Check how many trials participants	14
Check whether participants saw a unique fribble:	14
Check trial/stimuli per category, per frequency, per subject	15
Check test 2: Generalization from label to pictures	16
How many trials per participant?	16
Check whether participants saw a unique fribble	17
Check whether fribbles presented were either high or low frequency within trials	22
Check trial distribution per frequency:	23
Check responses distribution by category:	23
Check responses distribution by frequency:	24
Check test 3: Contingency Judgement task	24

How many trials per participant?	24
Did participants see a fribble more than once?	25
Are labels repeated equally?	25
Check category presentation:	27
Check test 4: Random dot task	28
How many trials per participant?	28
How was accuracy distributed across participants?	29
Check the overall accuracy of participants, filtering by timeouts:	30
Data visualization	32
Select the version of the experiment	32
Reaction times	32
accuracy	34
RandomDot	34
Task 1: from picture to labels	39
Task 2: from label to pictures	46
Comparison by frequency by learning by tasks	53
Speed-accuracy trade-off by tasks	57
Final Comparisons	62
Task 3: Contingency judgement	66
plot mean responses	72
Bayes factor calculation with GLMMs	81
Estimates of the betas from the FLO paper	81
GLMMs with all tests separately	82
Combine both generalization tasks in one dataset	85
The model	86
BF for Frequency:	88
BF for learning:	88
BF for the interaction frequency by learning	89
Summary of the results	89

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);
```

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

Load functions from the lab repo

```
urlFolder <- 'https://api.github.com/repos/n400peanuts/languagelearninglab/git/trees/master?recursive=1'
urlRaw <- 'https://raw.githubusercontent.com/n400peanuts/languagelearninglab/master/tools/'
```

```
loadFunctionsGithub <-function(urlFolder, urlRaw){
  if (!require(httr)) {
    stop("httr not installed")
  }
  else if (!require(RCurl)){
    stop("RCurl not installed")
  }
  else {
    print('----loading. Please wait----')
  };
  httr::GET(urlFolder)-> req
  stop_for_status(req)
  filelist <- unlist(lapply(content(req)$tree, "[", "path"), use.names = F)
  urlFunctions <- grep("docs/tools/", filelist, value = TRUE, fixed = TRUE)
  gsub("docs/tools/", "", urlFunctions) -> functions
  for (i in 1:length(functions)){
    RCurl::getURL(paste0(urlRaw, functions[i]), ssl.verifypeer = FALSE)-> temp
    eval(parse(text = temp), envir = .GlobalEnv)
  };
}
```

```
loadFunctionsGithub(urlFolder = urlFolder, urlRaw = urlRaw);
```

```
## [1] "----loading. Please wait----"
```

```
rm(urlFolder, urlRaw)
```

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 $n > 1$:

```
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.

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

We have 6 files.

```
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`
dataFL$Experiment.Version <- c(14)
dataFL2<-`data_exp_15519-v14_task-2yjh`
dataFL3<-`data_exp_15519-v15_task-2yjh`

rm(`data_exp_15519-v13_task-2yjh`)
rm(`data_exp_15519-v14_task-2yjh`)
rm(`data_exp_15519-v15_task-2yjh`)

dataLF <- `data_exp_15519-v13_task-q8hp`
dataLF$Experiment.Version <- c(14)
dataLF2 <- `data_exp_15519-v14_task-q8hp`
dataLF3 <- `data_exp_15519-v15_task-q8hp`

rm(`data_exp_15519-v13_task-q8hp`)
rm(`data_exp_15519-v14_task-q8hp`)
rm(`data_exp_15519-v15_task-q8hp`)

```

```

rbind(dataFL, dataFL2, dataFL3)-> dataFL
rbind(dataLF, dataLF2, dataLF3)-> dataLF

rm(dataFL2, dataFL3, dataLF2, dataLF3)

```

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', 'Experiment.Version')]

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

```

Select rows:

```

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

raw_dataFL <- raw_dataFL %>%
  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)

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.

```

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

```

Check learning

Let's filter and check learning trials:

```

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

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

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

```

How many trials per participant?

```

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

## # A tibble: 120 x 3
## # Groups:   subjID, learning [120]
##   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 110 more rows
```

Great, 120 trials per participant, per learning.

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

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

```
## # A tibble: 480 x 3
## # Groups:   subjID, task [480]
##   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 470 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?

```
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
##	1457883	0	120
##	1458992	120	0
##	1458996	0	120
##	1458997	0	120
##	1458998	0	120
##	1459001	120	0
##	1459002	120	0
##	1459003	120	0
##	1459007	120	0
##	1459009	120	0
##	1459013	120	0
##	1459015	0	120
##	1459018	120	0
##	1459020	0	120
##	1459024	120	0
##	1459025	0	120
##	1459029	120	0
##	1459036	0	120
##	1459039	0	120
##	1459043	0	120
##	1459046	0	120
##	1459047	120	0
##	1459048	120	0
##	1459052	120	0
##	1459057	120	0
##	1459064	120	0
##	1459067	0	120
##	1459078	0	120
##	1459109	0	120
##	1459696	116	0
##	1459697	120	0
##	1459699	0	120
##	1459700	0	120
##	1459701	0	120
##	1459702	120	0
##	1459703	120	0
##	1459706	120	0

```
## 1459708 120 0
## 1459709 0 120
## 1459767 120 0
```

Kind of. After chicking with Gorilla's suppoert: 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:

```
as.factor(gsub("/task/70033/56/asset/||task/70033/57/asset/||task/70033/58/asset/", "", learning$fribbleID))
as.factor(gsub(".jpg$", "", learning$fribbleID))-> learning$fribbleID

as.factor(gsub("/task/70033/56/asset/||task/70033/57/asset/||task/70033/58/asset/", "", learning$label))
as.factor(gsub(".mp3$", "", learning$label))-> learning$label
learning$resp <- as.factor('NA')
```

This is how the learning dataframe looks like now:

```
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 Experiment.Version
## 1 audio-keyboard-response      22  NA              14
## 2 audio-keyboard-response      25  NA              14
## 3 audio-keyboard-response      28  NA              14
## 4 audio-keyboard-response      31  NA              14
## 5 audio-keyboard-response      34  NA              14
## 6 audio-keyboard-response      37  NA              14
```

```
summary(learning);
```

```
##      subjID      learning      task      fribbleID      label
## Min.      :1414932  FL:7556  learningBlock1:3529  10475 : 124  FLbim:2519
## 1st Qu.:1422477  LF:6840  learningBlock2:3773  31675 : 124  FLdep:2517
## Median :1431970              learningBlock3:3595  13375 : 121  FLtob:2520
## Mean      :1437270              learningBlock4:3499  22775 : 120  LFbim:2280
## 3rd Qu.:1459009              30375 : 120  LFdep:2280
## Max.      :1459767              32475 : 120  LFtob:2280
##                                (Other):13667
##      rt      resp      trialType      trialIndex
## Min.      : 12.36  NA:14396  audio-keyboard-response:7556  Min.      : 22.0
## 1st Qu.: 52.50      image-keyboard-response:6840  1st Qu.:115.0
## Median : 88.00                                Median :211.0
## Mean      :126.25                                Mean      :210.8
```

```
## 3rd Qu.:214.71          3rd Qu.:307.0
## Max.    :249.00        Max.    :400.0
## NA's    :14389
##      acc      Experiment.Version
## Min.    : NA      Min.    :14.00
## 1st Qu.: NA      1st Qu.:14.00
## Median : NA      Median :14.00
## Mean    :NaN      Mean    :14.33
## 3rd Qu.: NA      3rd Qu.:15.00
## Max.    : NA      Max.    :15.00
## NA's    :14396
```

Our fribbles were presented two times during learning.

Check if fribbles are presented > 2 times:

```
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.

Check whether there are fribbles presented only once:

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

```
## # A tibble: 4 x 3
## # Groups:   subjID, fribbleID [4]
##   subjID fribbleID     n
##   <int> <fct>      <int>
## 1 1459696 12075         1
## 2 1459696 12675         1
## 3 1459696 13375         1
## 4 1459696 22125         1
```

Perfect.

Check the association between the fribbles and the labels (high and low frequency with the correct 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.

```
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] 19.9875
```

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

```
## [1] 19.99583
```

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

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

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

```
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] 14.99583
```

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

```
## [1] 44.9875
```

Now let's check the fribble-label association:

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

```
## , , = 25
##
##
##      FLbim FLdep FLtob LFbim LFdep LFtob
##  1      0   630    0     0   570    0
##  2   629    0    0   570    0    0
##  3      0    0   630    0    0   570
##
## , , = 75
##
##
##      FLbim FLdep FLtob LFbim LFdep LFtob
##  1      0  1887    0     0  1710    0
##  2  1890    0    0   1710    0    0
##  3      0    0  1890    0    0  1710
```

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/||task/70033/58/asset/", "", testing$fribbleID))
as.factor(gsub(".jpg$", "", testing$fribbleID))-> testing$fribbleID

as.factor(gsub("/task/70033/56/asset/||task/70033/57/asset/||task/70033/58/asset/", "", 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.

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

as.factor(gsub("/task/70033/56/asset/||task/70033/57/asset/||task/70033/58/asset/", "", generalizationPL$resp))
as.factor(gsub(".mp3$", "", generalizationPL$resp))-> generalizationPL$resp
as.factor(gsub(".jpg", "", generalizationPL$resp))-> generalizationPL$resp
```

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

Check how many trials participants

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

```
## # A tibble: 120 x 2
## # Groups:   subjID [120]
##   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 110 more rows
```

Great, 24 trials per participant.

Check whether participants saw a unique fribble:

```
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.

```
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.

```
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: 434 x 3
## # Groups:   subjID, resp [434]
##   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 424 more rows
```

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

So far, so good.

Check trial/stimuli per category, per frequency, per subject

We have 24 trials per participant, but within those trials we *should* have 8 trials per category, 4 low frequency and 4 high frequency trials.

```
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: 120 x 2
## # Groups:   subjID [120]
##   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 110 more rows
```

24 trials, great.

Check whether participants saw a unique fribble

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/|/task/70033/58/asset/", "", generalizationLP$fribbleID))-> generalizationLP$fribbleID

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] 1459007
## [1] 0
## [1] 1459002
## [1] 0
## [1] 1459009
## [1] 0
## [1] 1459001
## [1] 0
## [1] 1459003
## [1] 0
## [1] 1459013
## [1] 0
## [1] 1459029
## [1] 0
## [1] 1458992
## [1] 0
## [1] 1459018
## [1] 0
## [1] 1459024
## [1] 0
## [1] 1459047
## [1] 0
## [1] 1459052
## [1] 0
## [1] 1459064
## [1] 0
## [1] 1459048
## [1] 0
## [1] 1459057
## [1] 0
## [1] 1459697
## [1] 0
## [1] 1459696
## [1] 0
## [1] 1459706
## [1] 0
## [1] 1459702
## [1] 0
## [1] 1459708
## [1] 0
## [1] 1459703
```

```
## [1] 0
## [1] 1459767
## [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
## [1] 1457883
## [1] 0
## [1] 1458997
## [1] 0
## [1] 1459015
## [1] 0
## [1] 1459025
## [1] 0
## [1] 1458998
## [1] 0
## [1] 1458996
## [1] 0
## [1] 1459043
## [1] 0
## [1] 1459036
## [1] 0
## [1] 1459039
## [1] 0
## [1] 1459046
## [1] 0
## [1] 1459067
## [1] 0
## [1] 1459020
## [1] 0
## [1] 1459078
## [1] 0
## [1] 1459109
```

```
## [1] 0
## [1] 1459701
## [1] 0
## [1] 1459700
## [1] 0
## [1] 1459709
## [1] 0
## [1] 1459699
## [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 within trials

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
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:

Check trial distribution per frequency:

```
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:

```
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);
```

Check responses distribution by category:

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

```
## # A tibble: 427 x 3
## # Groups:   subjID, category [427]
##   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 417 more rows
```

Cool.

Check responses distribution by frequency:

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

## # A tibble: 837 x 4
## # Groups:   subjID, label, frequency [837]
##   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 827 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: 120 x 2
## # Groups:   subjID [120]
##   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 110 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
## 1457883 8 8 8
## 1458992 8 8 8
## 1458996 8 8 8
## 1458997 8 8 8
## 1458998 8 8 8
## 1459001 8 8 8
## 1459002 8 8 8
## 1459003 8 8 8
## 1459007 8 8 8
## 1459009 8 8 8
## 1459013 8 8 8
## 1459015 8 8 8
## 1459018 8 8 8
## 1459020 8 8 8
## 1459024 8 8 8
## 1459025 8 8 8
## 1459029 8 8 8
## 1459036 8 8 8
## 1459039 8 8 8
## 1459043 8 8 8
## 1459046 8 8 8
## 1459047 8 8 8
## 1459048 8 8 8
## 1459052 8 8 8
## 1459057 8 8 8
## 1459064 8 8 8
## 1459067 8 8 8
## 1459078 8 8 8
## 1459109 8 8 8
## 1459696 8 8 8
## 1459697 8 8 8
## 1459699 8 8 8
## 1459700 8 8 8
## 1459701 8 8 8
## 1459702 8 8 8
## 1459703 8 8 8
## 1459706 8 8 8
## 1459708 8 8 8
## 1459709 8 8 8
## 1459767 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: 360 x 3
## # Groups:   subjID, category [360]
##   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 350 more rows
```

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

```
##
##   bim dep tob
## 1 312 312 336
## 2 384 288 288
## 3 264 360 336
```

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?

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

```
## # A tibble: 120 x 2
## # Groups:   subjID [120]
##   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 110 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

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

```
## # A tibble: 83 x 3
## # Groups:   subjID, resp [83]
##   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 73 more rows
```

Here we can see that some participant missed some trials.

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

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

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

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

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

```
## [1] 325
```

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

```
## [1] 196
```

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.

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

Check the overall accuracy of participants, filtering by timeouts:

```
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
81 1457883 1.0000000
82 1458992 1.0000000
83 1458996 1.0000000
84 1458997 1.0000000
85 1458998 1.0000000
86 1459001 0.6521739
87 1459002 1.0000000
88 1459003 0.9600000
89 1459007 1.0000000
90 1459009 0.2916667
91 1459013 0.9600000
92 1459015 0.9565217
93 1459018 1.0000000
94 1459020 1.0000000
95 1459024 1.0000000

```
## 96 1459025 1.0000000
## 97 1459029 1.0000000
## 98 1459036 0.5652174
## 99 1459039 1.0000000
## 100 1459043 1.0000000
## 101 1459046 1.0000000
## 102 1459047 1.0000000
## 103 1459048 0.9523810
## 104 1459052 1.0000000
## 105 1459057 0.9166667
## 106 1459064 1.0000000
## 107 1459067 1.0000000
## 108 1459078 0.8800000
## 109 1459109 0.9583333
## 110 1459696 0.8333333
## 111 1459697 1.0000000
## 112 1459699 1.0000000
## 113 1459700 0.9600000
## 114 1459701 0.9615385
## 115 1459702 1.0000000
## 116 1459703 0.6956522
## 117 1459706 0.9565217
## 118 1459708 1.0000000
## 119 1459709 1.0000000
## 120 1459767 1.0000000
```

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

Data visualization

Okay, from the sanity checks done above we can draw two conclusions:

1. Learning and Testing was presented as it was supposed to be and
2. data was stored correctly

Let's see now if data makes sense.

Select the version of the experiment

Select the version of the experiment you want:

- Version 14 has 80 subjects, label picture task has 2500ms as timeout
- Version 15 has 42 subjects, label picture task has 3500ms as timeout

```
ver2 <- c(14)
```

Reaction times


```

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

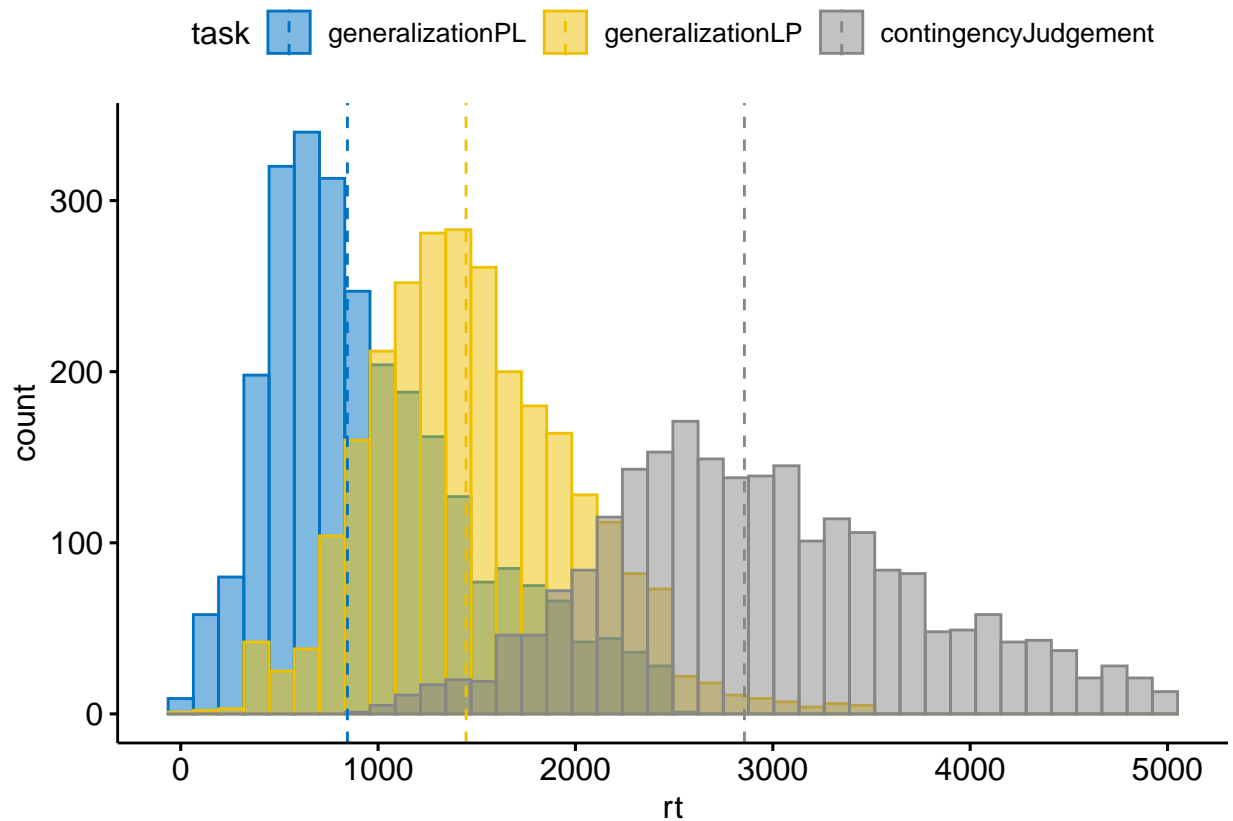
```

```

gghistogram(alltasks,
  x = "rt",
  y = "..count..",
  xlab = "rt",
  color = "task",
  fill = "task",
  bins = 40,
  palette = "jco",
  add = "median"
)

```

Warning: Removed 934 rows containing non-finite values (stat_bin).



The two generalization tasks looks quite different. I'm going to plot it separately for a better inspection:

```

p<- gghistogram(alltasks, #will throw warnings related to non responses but that's okay, ggplot simply
  x = "rt",
  y = "..count..",
  xlab = "rt",
  facet.by = "task",
  add = "median",

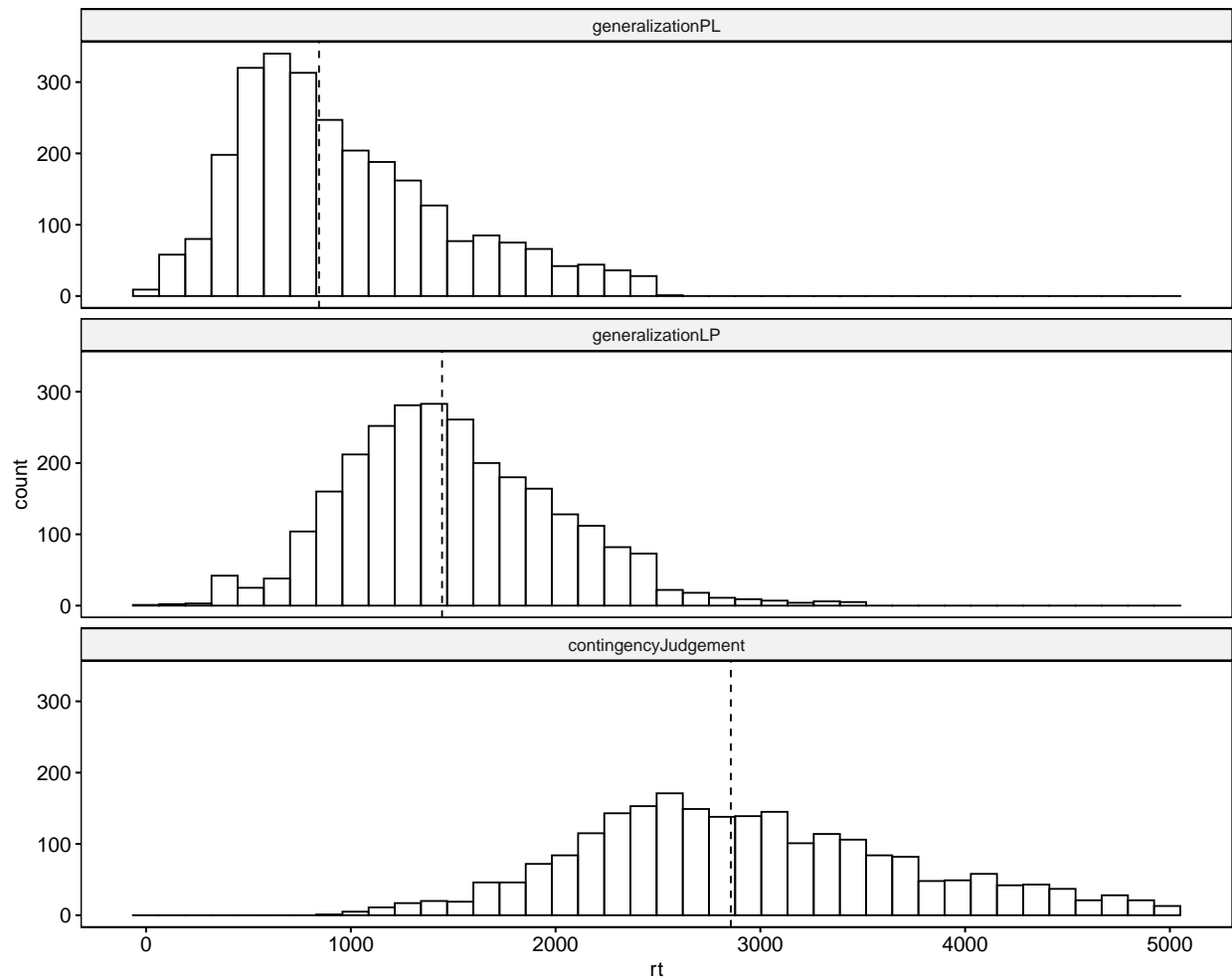
```

```

    bins = 40
  )
  facet(p, facet.by = "task",
        nrow = 3,
        ncol = 1)

```

```
## Warning: Removed 934 rows containing non-finite values (stat_bin).
```



The tails of the first two tasks don't end smoothly, especially in task 2.

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

```

```

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

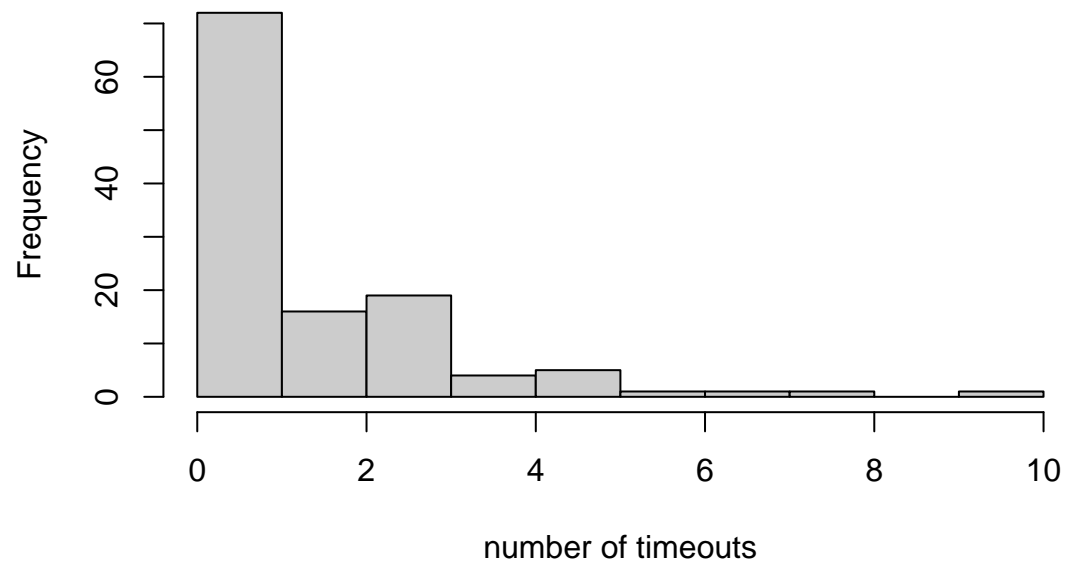
```

How many timeouts by participant? Histogram by participant:

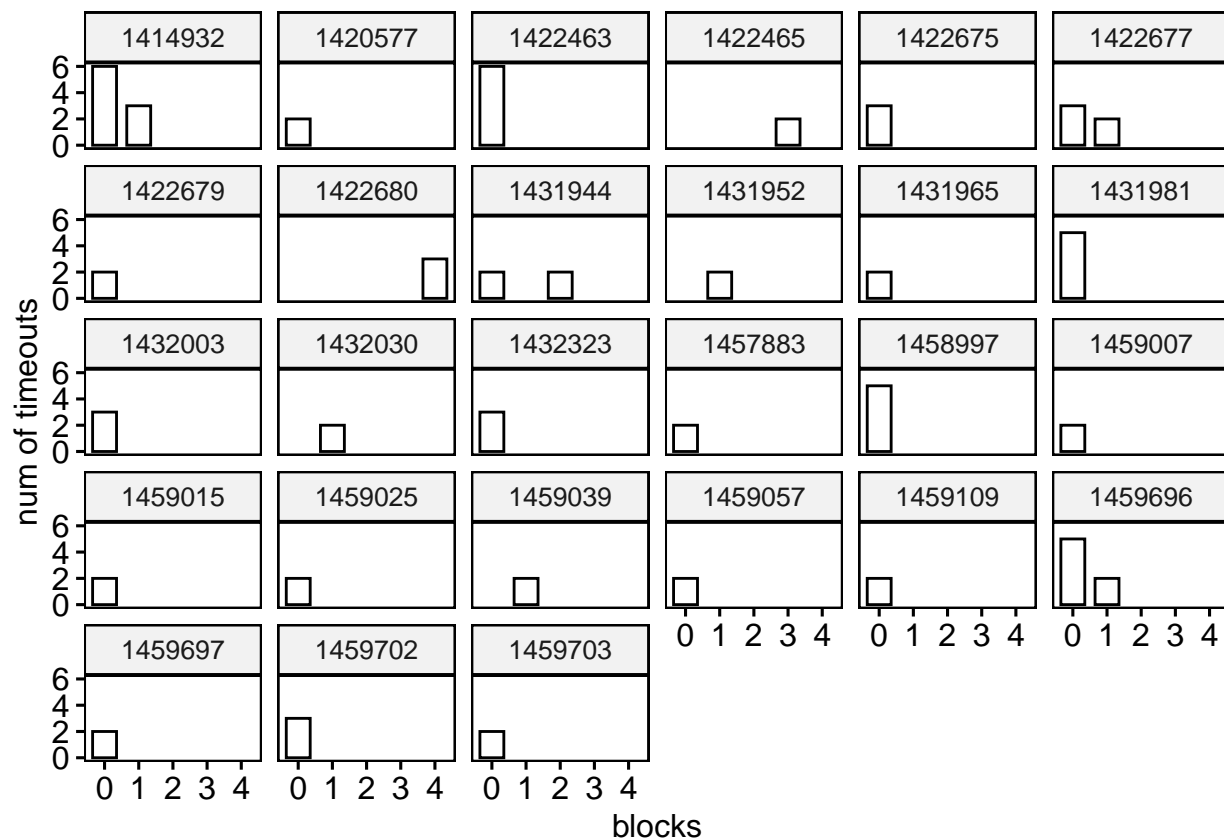
```

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")
```

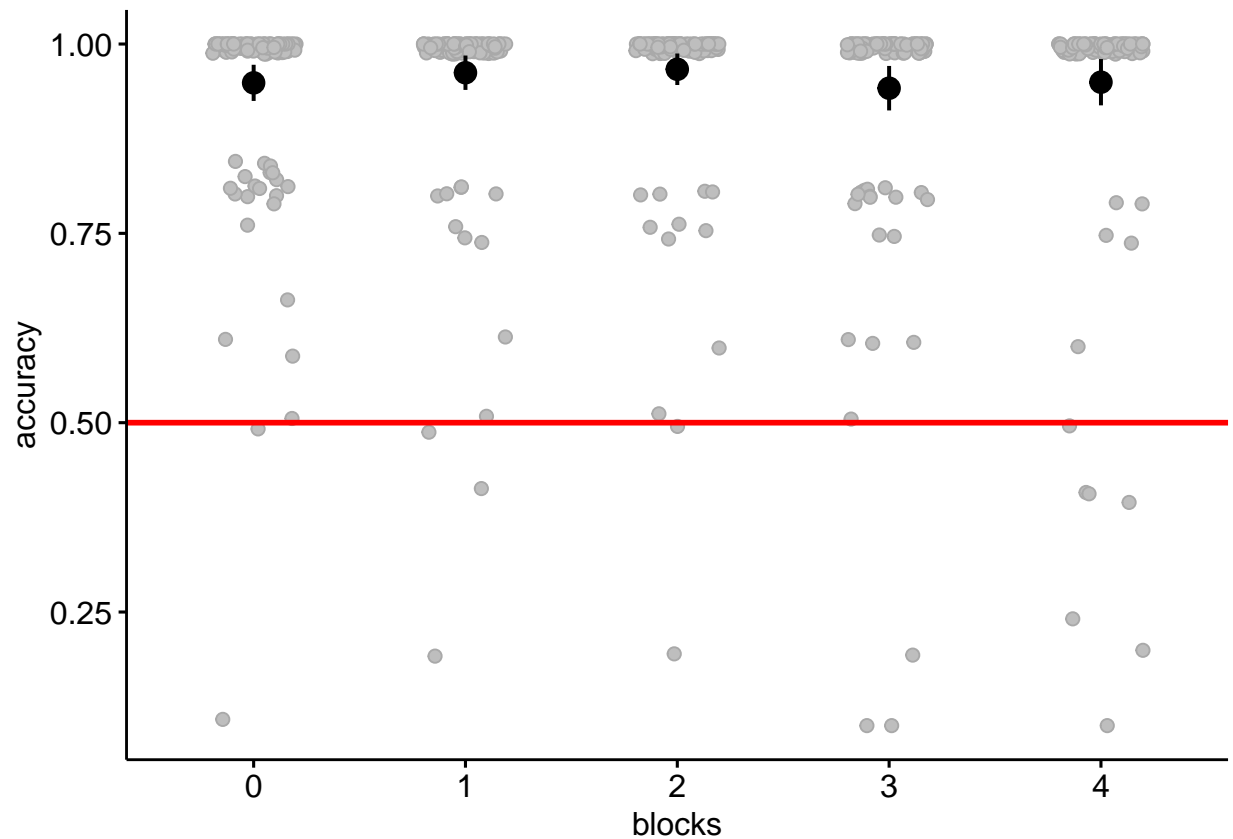


Subjects that made more than 3 timeouts

```
unique(timeout[timeout>3,]$subjID) -> problematicPeople
```

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

accdistr[accdistr$m <= .5,]
```

```
## # A tibble: 21 x 3
## # Groups:   subjID [11]
##   subjID blocks     m
##   <int> <fct> <dbl>
## 1 1414932 4      0.25
## 2 1422470 1      0.4
## 3 1422475 2      0.5
## 4 1422475 3      0.2
## 5 1422475 4      0
## 6 1422689 3      0
## 7 1422689 4      0.4
## 8 1431942 4      0.4
## 9 1459001 3      0.5
## 10 1459001 4      0.2
## # ... with 11 more rows
```

```
unique(accdistr[accdistr$m < .7,]$subjID) -> dumbPeople
```

```
setdiff(dumbPeople, problematicPeople)-> dumbPeople
```

People that scored less than 70%: Let's consider them as bad subjects.

```
c(problematicPeople, dumbPeople)->badsubjs
```

```
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.

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

How many participants do we have per learning?

```
## [1] 120
```

```
f1<- length(unique(generalizationPL[generalizationPL$learning=='FL' &
                                !(generalizationPL$subjID %in% badsubjs),]$subjID))
lf<- length(unique(generalizationPL[generalizationPL$learning=='LF' &
                                !(generalizationPL$subjID %in% badsubjs),]$subjID))
f1
```

```
## [1] 55
```

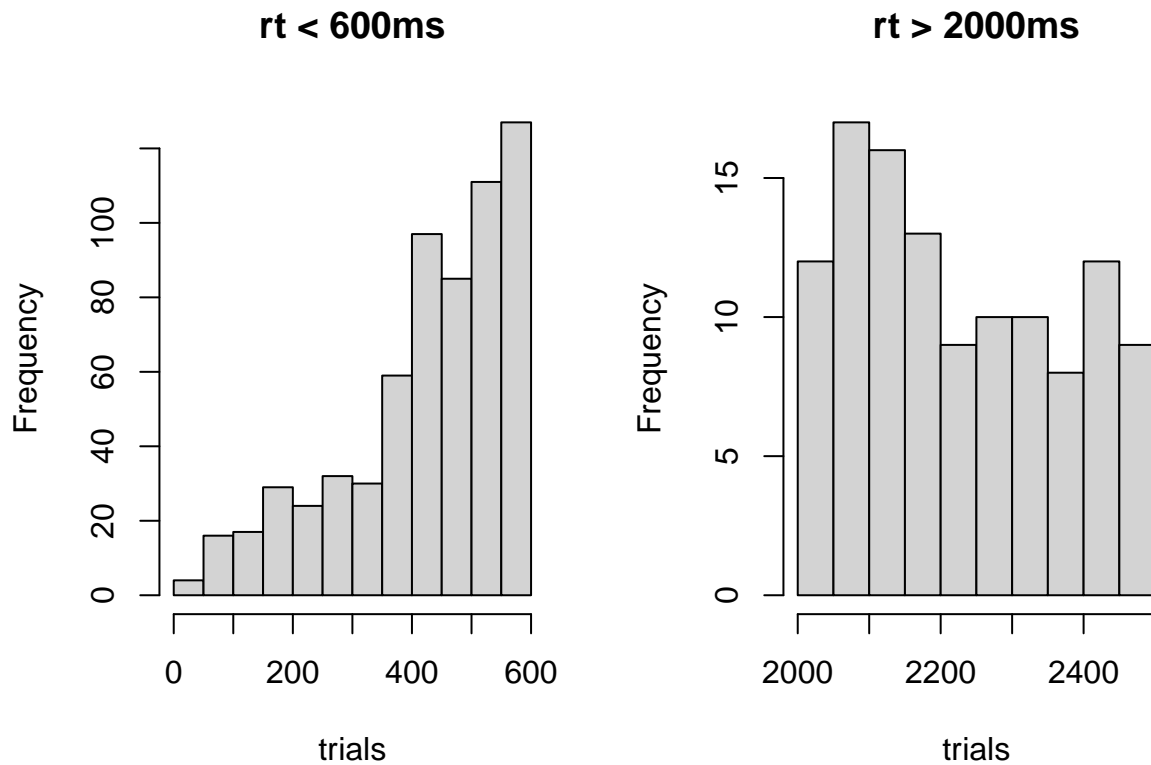
```
lf
```

```
## [1] 47
```

We have 55 for feature-label learning, and 47 for label-feature learning.

Check tails of the rt distribution The point is that we can't rely on responses made very early, because these might be simply mistakes or technical errors.

```
par(mfrow=c(1,2))
hist(generalizationPL[generalizationPL$rt<600 &
                    !(generalizationPL$subjID %in% badsubjs),]$rt, main = 'rt < 600ms', xlab = 'tr
hist(generalizationPL[generalizationPL$rt>2000 &
                    !(generalizationPL$subjID %in% badsubjs),]$rt, main = 'rt > 2000ms', xlab = 'tr
```



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

I would remove $rt < 100ms$ for all tasks.

```
round(nrow(generalizationPL[generalizationPL$rt<100 &
!(generalizationPL$subjID %in% badsubjs),]) / nrow(generalizationPL[!(gen
```

How many, what type of trials do we have?

```
## [1] 6.74
```

```
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;
```



```
n <- length(unique(pictureLabel[!(pictureLabel$subjID %in% badsubjs),]$subjID))

nrows <- (nrow(generalizationPL[!(generalizationPL$subjID %in% badsubjs),])) - (nrow(pictureLabel[!(pic

sort(unique(pictureLabel[!(pictureLabel$subjID %in% badsubjs),]$subjID))-> subjs;
sort(unique(generalizationPL[!(generalizationPL$subjID %in% badsubjs),]$subjID)) ->totsubjs;

subjmisses<- setdiff(totsubjs, subjs);

rm(subjs, totsubjs);
```

We have 101 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 145 responses, that is 5.0347222 over the total: 2880.

How many trials per participant do we have now?

```
pictureLabel %>%
  group_by(subjID) %>%
  count() %>% filter(n<=18)
```

```
## # A tibble: 2 x 2
## # Groups:   subjID [2]
##   subjID      n
##   <int> <int>
## 1 1422475     18
## 2 1432075     18
```

No one had less than 18 trials, over the total (24). That's fine!

Barplot accuracy by category + frequency + learning picture label

```
c(badsubjs, subjmisses) -> badsubjs

rm(n, subjmisses, nrows)

ss_prop<-aggregate(acc ~ frequency+category+subjID+learning,
                    data = pictureLabel[pictureLabel$rt > 100 &
                                          !(pictureLabel$subjID %in% badsubjs),], FUN = mean)
```

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

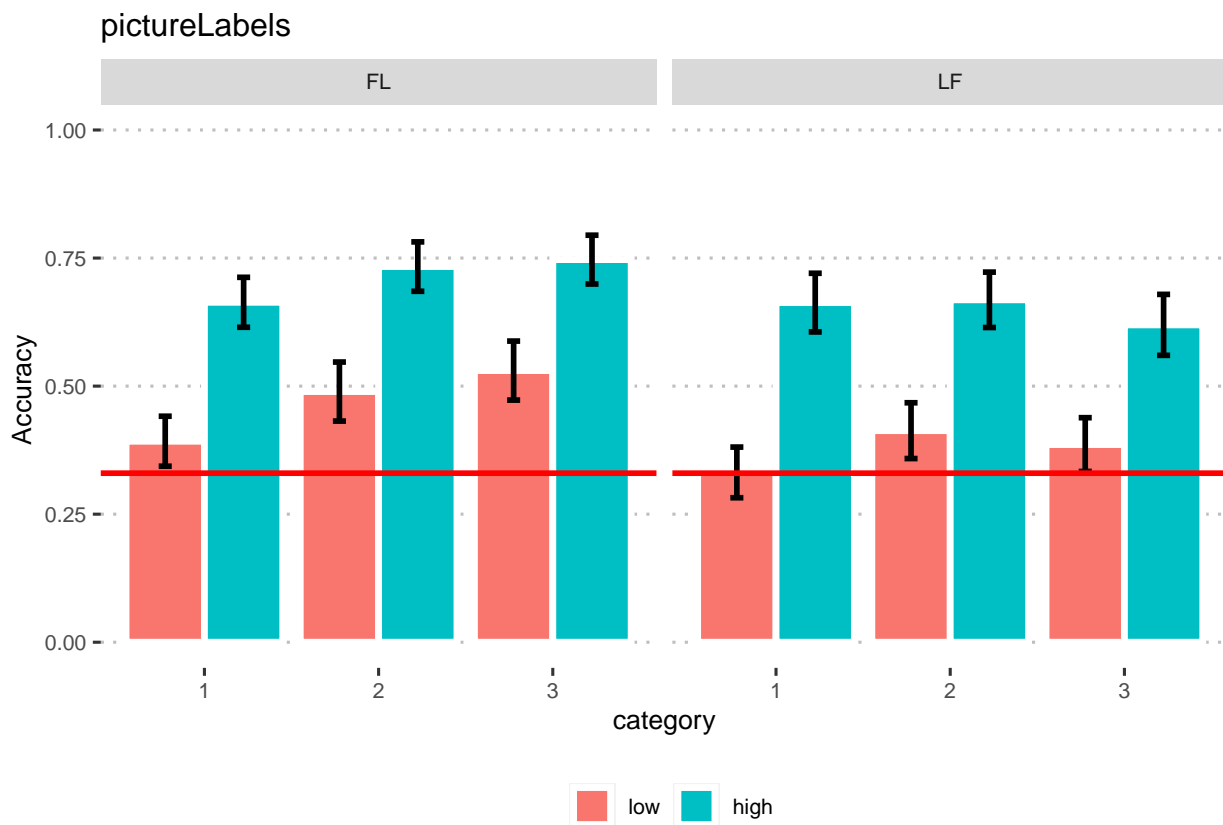
```
ms <- ss_prop %>%
  group_by( category, frequency, learning) %>%
  summarise(n=n(),
            mean=mean(acc),
            sd=sd(acc)
  ) %>%
  mutate( se=sd/sqrt(n)) %>%
  mutate( ci=se * qt((1-0.05)/2 + .5, n-1))
```

```

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('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);

```



```

df <- aggregate(acc ~ subjID+frequency+learning+category,
  data = pictureLabel[pictureLabel$rt > 100 &
    !(pictureLabel$subjID %in% badsubjs)],, mean)
df$frequency <- as.factor(df$frequency)
plyr::revalue(df$frequency, c("25"="low"))-> df$frequency;

```

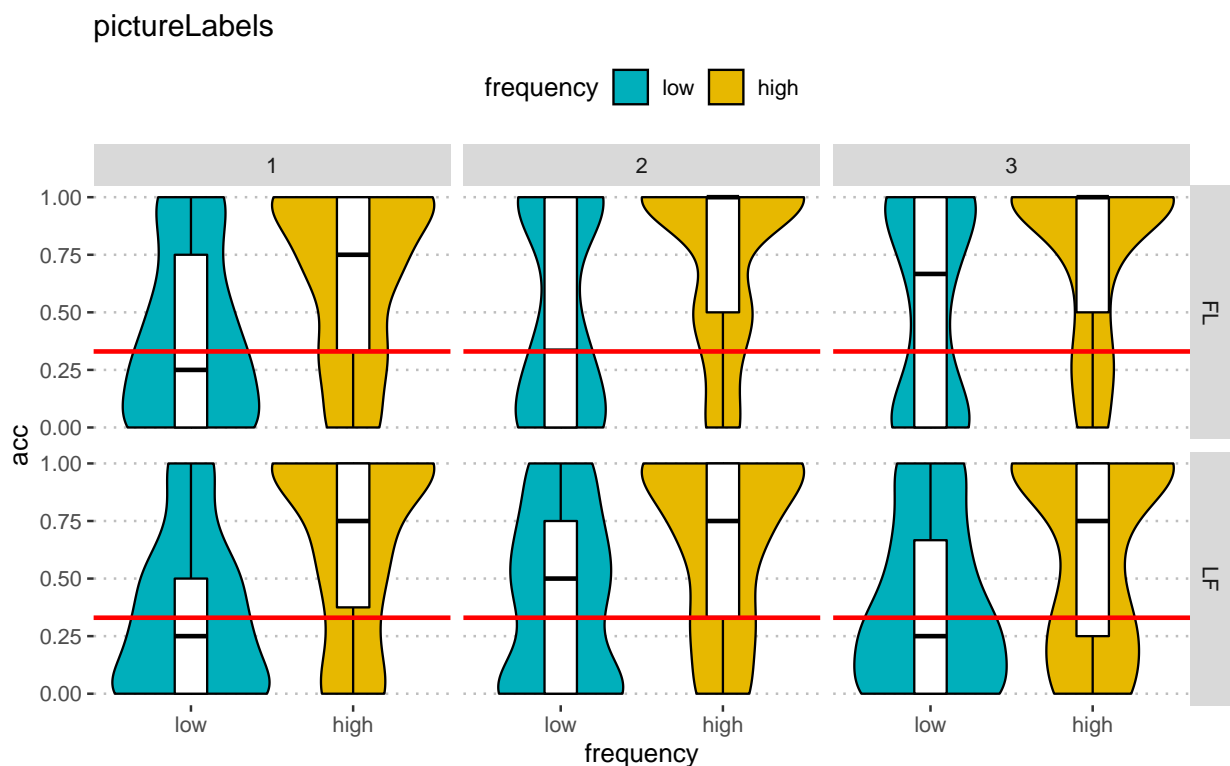
```

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);

```

Violin plot accuracy by category + frequency + learning



Violin plot accuracy by frequency + learning Let's see how participants scored for the high/low frequency:

```

df <- aggregate(acc ~ subjID+frequency+learning,
  data = pictureLabel[pictureLabel$rt > 100 &
    !(pictureLabel$subjID %in% badsubjs)], 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",

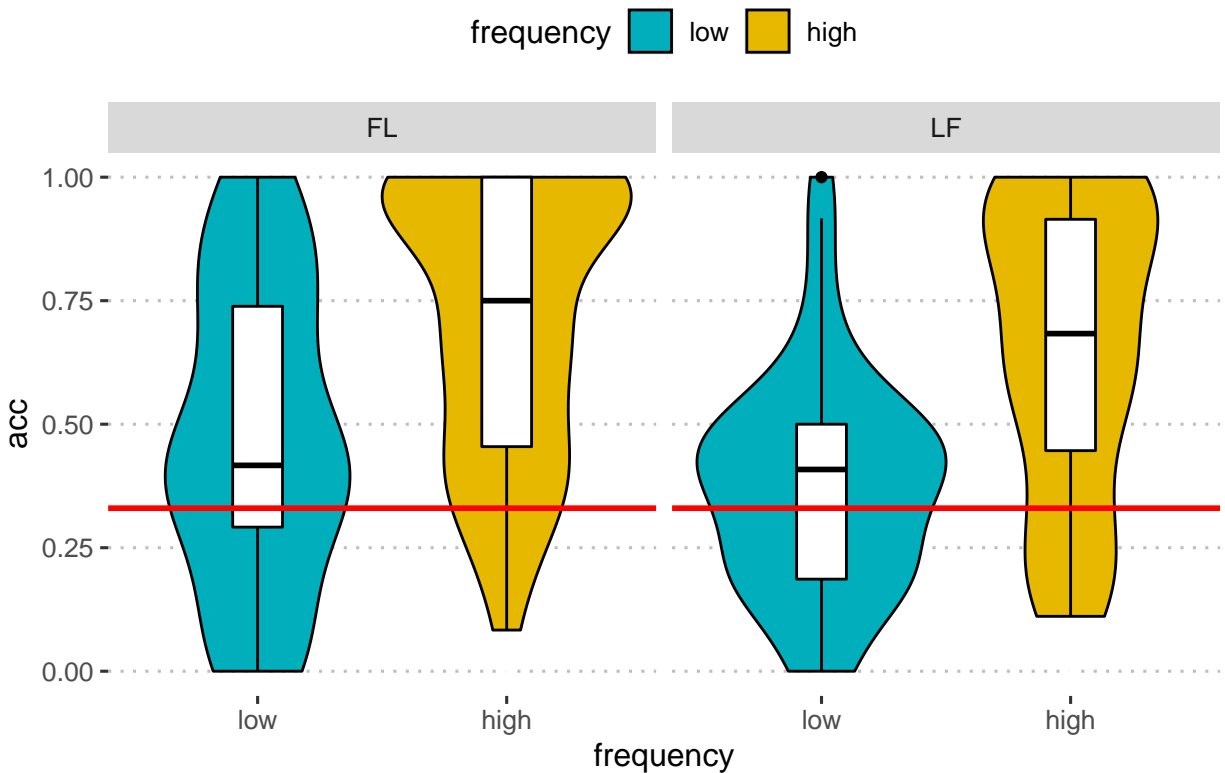
```

```

add.params = list(fill = "white"),
trim=TRUE) +
ggtitle('pictureLabels') +
facet_grid( . ~ learning) +
theme_pubclean()+
geom_hline(yintercept = .33, col='red', lwd=1);

```

pictureLabels



```

df %>%
  group_by(learning, frequency) %>%
  summarise(mean(acc))

```

```

## # A tibble: 4 x 3
## # Groups:   learning [2]
##   learning frequency `mean(acc)`
##   <fct>      <fct>      <dbl>
## 1 FL        low         0.474
## 2 FL        high        0.721
## 3 LF        low         0.376
## 4 LF        high        0.656

```

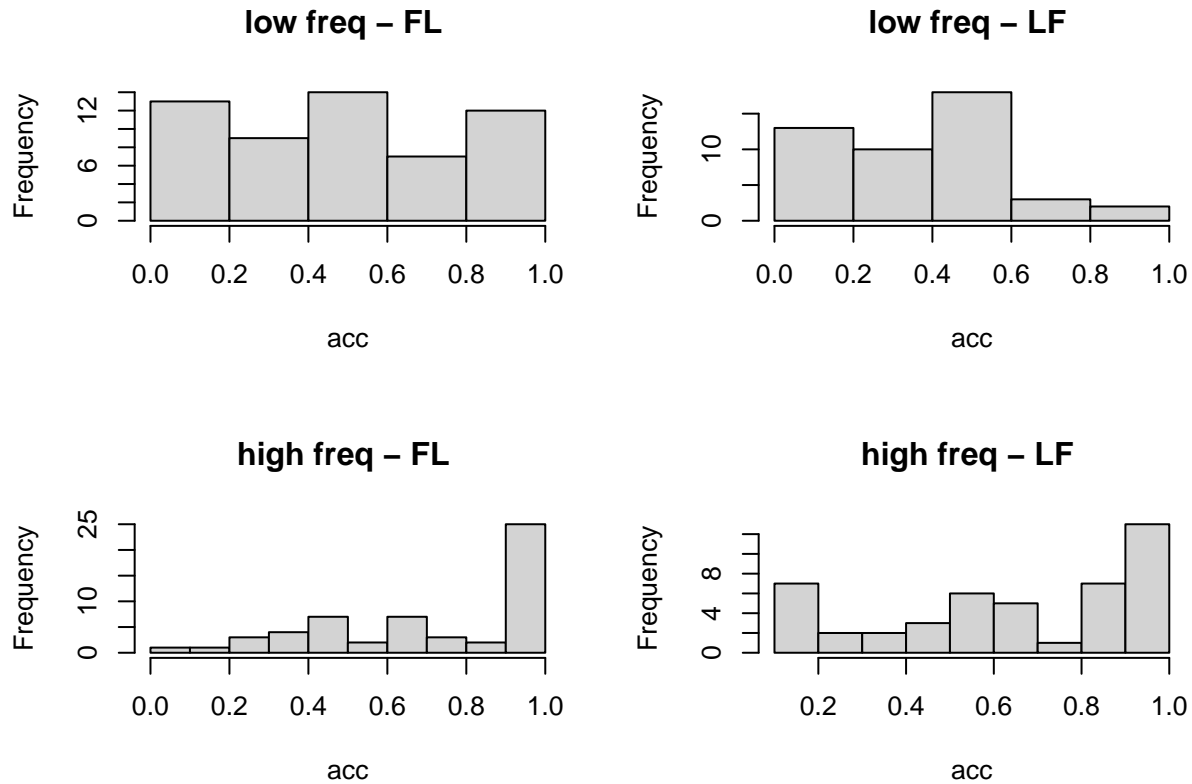
Closer inspection:

```

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$rt > 100 &
                                     !(pictureLabel$subjID %in% badsubjs) ], 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 ") +
  xlab("frequency") +
  ggtitle('pictureLabels') +
  coord_cartesian(ylim = c(0, 1))+
  ggpubr::theme_pubclean() +
  theme(legend.position = "none") +
  theme(text = element_text(size=10)) +
  geom_hline(yintercept = .33, col='red', lwd=1);

```

Barplot accuracy by frequency + learning

Task 2: from label to pictures

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

```

length(unique(generalizationLP$subjID))

## [1] 120

fl<- length(unique(generalizationLP[generalizationLP$learning=='FL' &
  !(generalizationLP$subjID %in% badsubjs),]$subjID))

lf<- length(unique(generalizationLP[generalizationLP$learning=='LF' &
  !(generalizationLP$subjID %in% badsubjs),]$subjID))
fl

## [1] 55

lf

## [1] 46

```

How many participants do we have per learning? We have 55 for feature-label learning, and 46 for label-feature learning.

```

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;

subjmisses<- setdiff(totsubjs, subjs);

```

Great, we have 120 participants in this task, so -0, and we have missed 195 over the total 2880, that is 6.7708333. The subject(s) that missed completely the task is: .

How many, what type of trials do we have? How many datapoints did we lose for no-responses?

```
round(nrow(generalizationLP[(is.na(generalizationLP$resp)) &
                             !(generalizationLP$subjID %in% badsubjs),]) / nrow(generalizationLP[!(gene
```

```
## [1] 6.77
```

How many trials were $rt < 100$?

```
round(nrow(generalizationLP[generalizationLP$rt<100 &
                             !(generalizationLP$subjID %in% badsubjs),]) / nrow(generalizationLP[!(gene
```

```
## [1] 6.81
```

Once trimmed, how many trials per participant do we have in this task?

```
labelPicture %>%
  group_by(subjID) %>%
  count() %>%
  filter(n<=18)
```

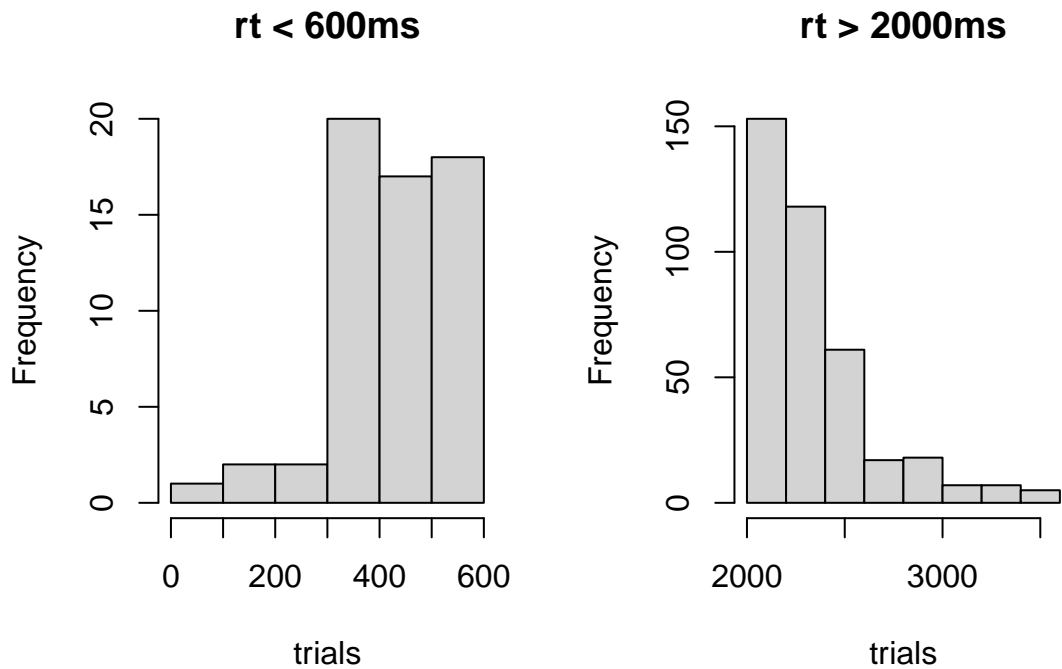
```
## # A tibble: 8 x 2
## # Groups:   subjID [8]
##   subjID     n
##   <int> <int>
## 1 1420577    18
## 2 1422475    18
## 3 1422477    17
## 4 1422677    17
## 5 1422680     9
## 6 1422689    17
## 7 1432009     8
## 8 1432075    17
```

Here we have less datapoints. For sure, 1422680 needs to be added to the black list because has few correct trials.

```
c(badsubjs, 1422680, 1432009) -> badsubjs
```

```
par(mfrow=c(1,2))
hist(generalizationLP[generalizationLP$rt<600 &
                      !(generalizationLP$subjID %in% badsubjs),]$rt, main = 'rt < 600ms', xlab = 'tri
hist(generalizationLP[generalizationLP$rt>2000 &
                      !(generalizationLP$subjID %in% badsubjs),]$rt, main = 'rt > 2000ms', xlab = 'tr
```

Check tails of the rt distribution



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

```
rm(n, nrow, 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;
```

Barplot accuracy by category+learning+frequency Calculate the proportion of correct in each condition

```
rm(submitted)
ss_prop<-aggregate(acc ~ frequency+category+subjID+learning,
                    data = labelPicture[labelPicture$rt > 100 &
                                          labelPicture$rt <= 2500 &
                                          !(labelPicture$subjID %in% badsubjs)],, 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)) %>%
```



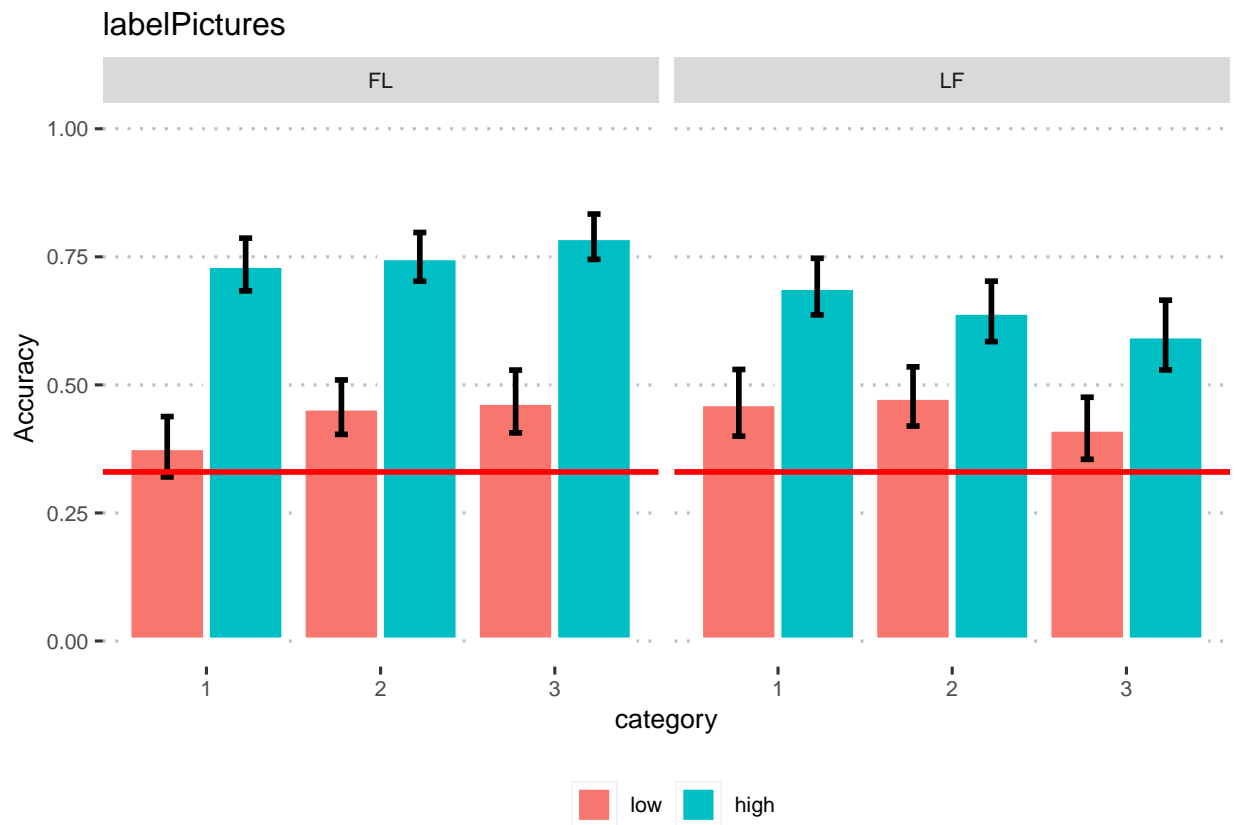
```

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$rt > 100 &
    labelPicture$rt <=2500 &

```

```

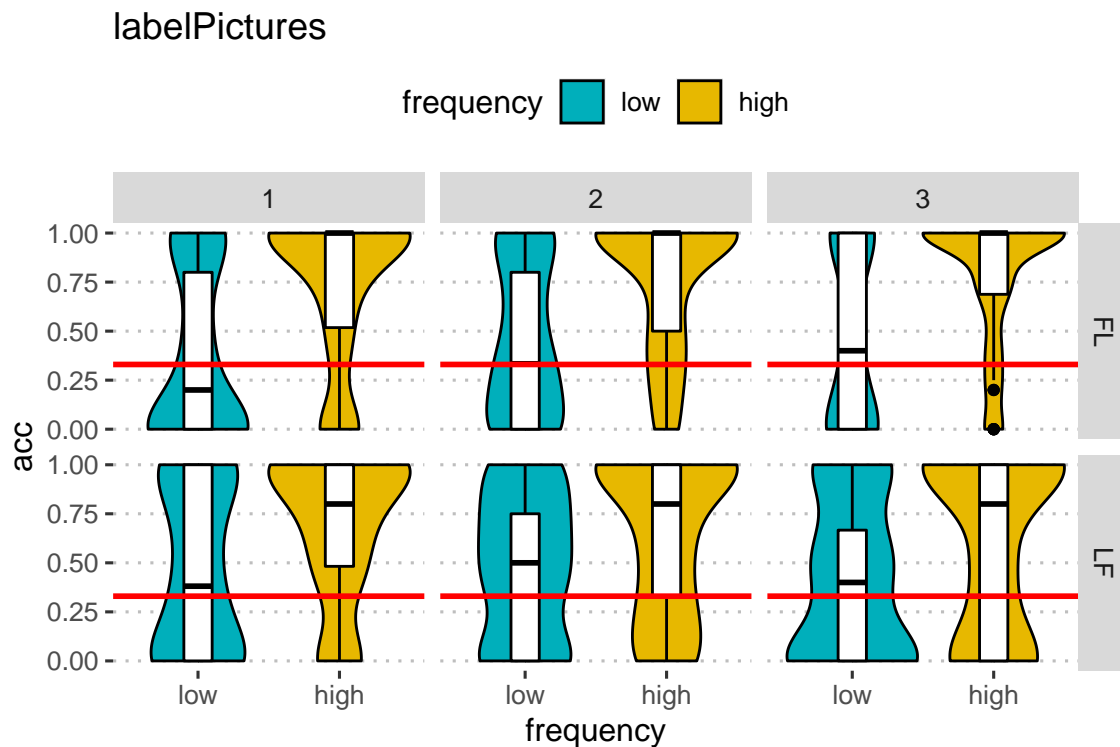
!(labelPicture$subjID %in% badsubjs),], mean)

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);

```

Violin plot accuracy by category+learning+frequency



```

#rm(ms, ss_prop)

```

```

ms <- aggregate(acc ~ subjID+frequency+learning,
  data = labelPicture[labelPicture$rt > 100 &
    labelPicture$rt <= 2500 &
    !(labelPicture$subjID %in% badsubjs)],, mean)

```

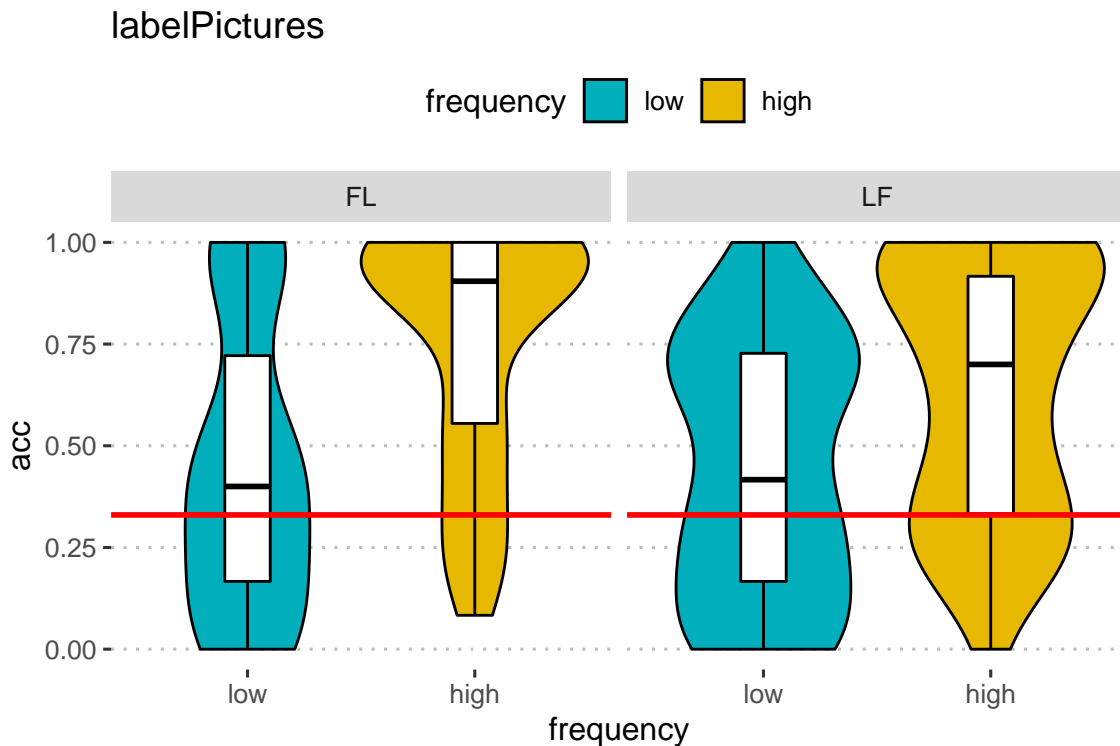
```

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

```

Violinplot accuracy by learning+frequency



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

```

ms %>%
  group_by(learning, frequency) %>%
  summarise(mean(acc))

```

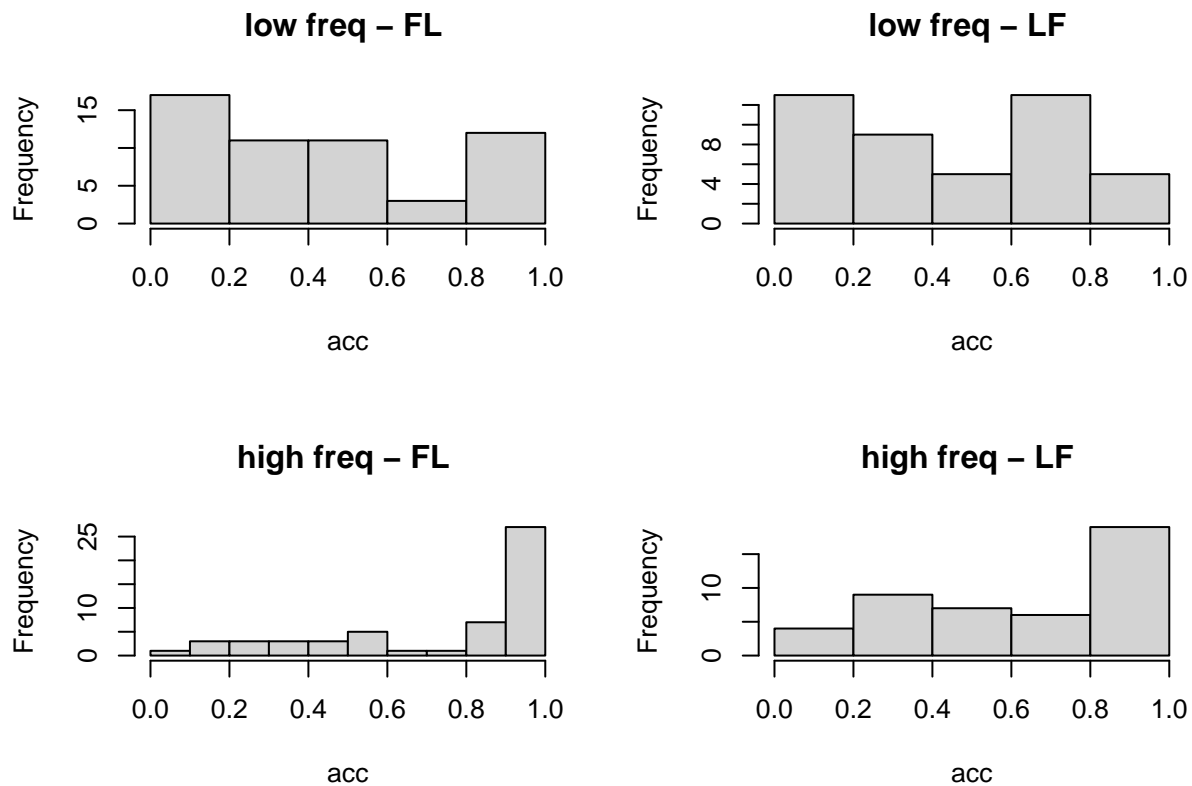
```

## # A tibble: 4 x 3
## # Groups:   learning [2]
##   learning frequency `mean(acc)`
##   <fct>    <fct>         <dbl>
## 1 FL      low          0.439

```

```
## 2 FL      high      0.750
## 3 LF      low       0.439
## 4 LF      high      0.644
```

```
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))
```

```
#barPlot aggregated over categories:

ms <- aggregate(acc ~ subjID+frequency+learning,
                 data=labelPicture[labelPicture$rt > 100 &
                                   labelPicture$rt <= 2500 &
                                   !(labelPicture$subjID %in% badsubjs)], 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);

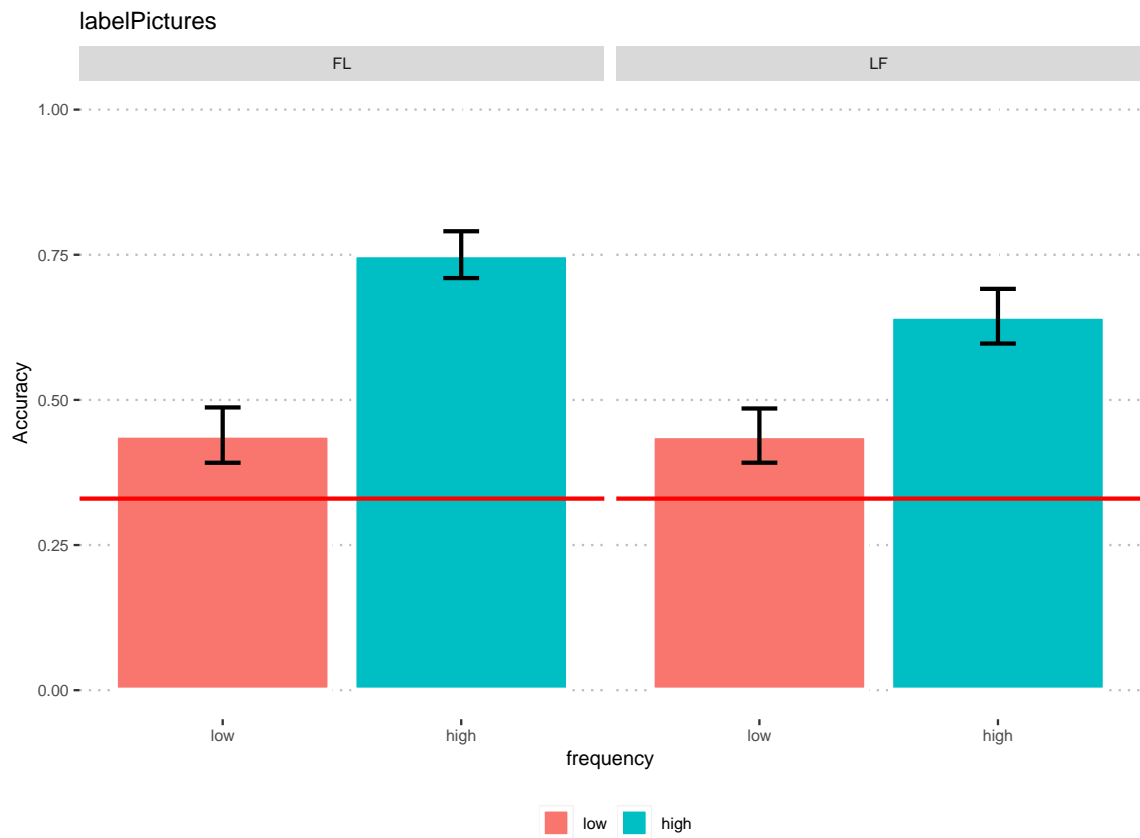
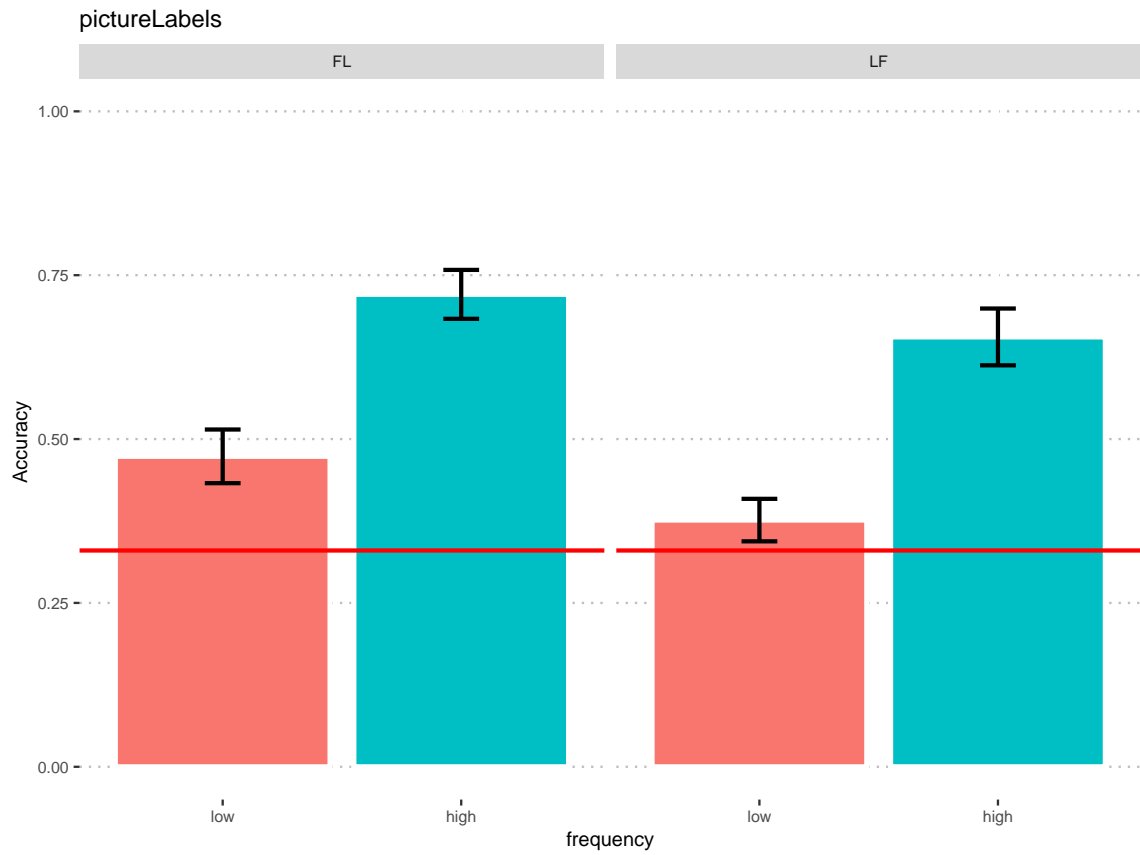
```

Barplot accuracy by frequency + learning

Comparison by frequency by learning by tasks

Quick summary of what we have so far:

```
grid.arrange(p1,lp)
```



What's going on in the low frequency condition? One way to see whether they simply learned another association is to check that wrong choices are distributed equally (50%) to the other two categories. If they are, then they didn't learn anything, but if they are not distributed equally, they have learned another association.

Label picture:

```
#select only inaccurate trials
temp <- labelPicture[labelPicture$acc==0,]

round(nrow(temp)/nrow(labelPicture)*100,2)
```

```
## [1] 46.07
```

How many of those are low frequency trials?

```
round(nrow(temp[temp$frequency==25,])/nrow(labelPicture)*100,2)
```

```
## [1] 29.5
```

How many of those are low frequency trials and how are they distributed across learnings?

```
round(nrow(temp[temp$frequency==25 & temp$learning=="FL",])/nrow(labelPicture)*100,2)
```

```
## [1] 15.79
```

```
round(nrow(temp[temp$frequency==25 & temp$learning=="LF",])/nrow(labelPicture)*100,2)
```

```
## [1] 13.71
```

FL people make more errors in the low freq condition

How many of those are high frequency trials and how are they distributed across learnings?

```
round(nrow(temp[temp$frequency==75 & temp$learning=="FL",])/nrow(labelPicture)*100,2)
```

```
## [1] 7.19
```

```
round(nrow(temp[temp$frequency==75 & temp$learning=="LF",])/nrow(labelPicture)*100,2)
```

```
## [1] 9.39
```

While they are pretty much the same in the high frequency

Label picture task:

correct choice is listed in "label", that is, label presented. Participant's choice is listed in "category", that is, the fribble's category.

```
temp %>%
  filter(frequency=="25") %>%
  group_by(learning, label, category) %>%
  count()
```

```
## # A tibble: 12 x 4
## # Groups:   learning, label, category [12]
##   learning label category     n
##   <fct>    <fct>    <int> <int>
## 1 FL      bim        1     36
## 2 FL      bim        3     92
## 3 FL      dep        2    107
## 4 FL      dep        3     43
## 5 FL      tob        1    107
## 6 FL      tob        2     39
## 7 LF      bim        1     22
## 8 LF      bim        3    100
## 9 LF      dep        2     74
## 10 LF     dep        3     44
## 11 LF     tob        1     78
## 12 LF     tob        2     50
```

Nope, they definitely learned another association. The association they have learned is based on the high saliency feature, rather than on the low saliency one. Let's see if that is the case also for the other task:

Picture label task:

```
#select only inaccurate trials
temp <- pictureLabel[pictureLabel$acc==0,]

round(nrow(temp)/nrow(pictureLabel)*100,2)
```

```
## [1] 46.26
```

How many of those are low frequency trials?

```
round(nrow(temp[temp$frequency==25,])/nrow(pictureLabel)*100,2)
```

```
## [1] 29.93
```

How many of those are low frequency trials and how are they distributed across learnings?

```
round(nrow(temp[temp$frequency==25 & temp$learning=="FL",])/nrow(pictureLabel)*100,2)
```

```
## [1] 15.44
```

```
round(nrow(temp[temp$frequency==25 & temp$learning=="LF",])/nrow(pictureLabel)*100,2)
```

```
## [1] 14.48
```

How many of those are high frequency trials and how are they distributed across learnings?


```
round(nrow(temp[temp$frequency==75 & temp$learning=="FL",])/nrow(pictureLabel)*100,2)
```

```
## [1] 7.93
```

```
round(nrow(temp[temp$frequency==75 & temp$learning=="LF",])/nrow(pictureLabel)*100,2)
```

```
## [1] 8.41
```

Picture label task:

correct choice is listed in “category”, that is, the category of the fribble presented. Participant’s choice is listed in “resp” column, that is, the label chosen.

```
temp %>%
  filter(frequency=="25") %>%
  group_by(learning, category, resp) %>%
  count()
```

```
## # A tibble: 12 x 4
## # Groups:   learning, category, resp [12]
##   learning category resp      n
##   <fct>      <int> <fct> <int>
## 1 FL          1 bim      44
## 2 FL          1 tob     110
## 3 FL          2 dep      78
## 4 FL          2 tob      60
## 5 FL          3 bim      83
## 6 FL          3 dep      42
## 7 LF          1 bim      53
## 8 LF          1 tob      93
## 9 LF          2 dep      61
## 10 LF         2 tob      55
## 11 LF         3 bim      91
## 12 LF         3 dep      38
```

In both tasks participants were driven by the high salient feature in making errors, they simply learned only one association between the label and the high salient feature, and made decisions based on this.

Speed-accuracy trade-off by tasks

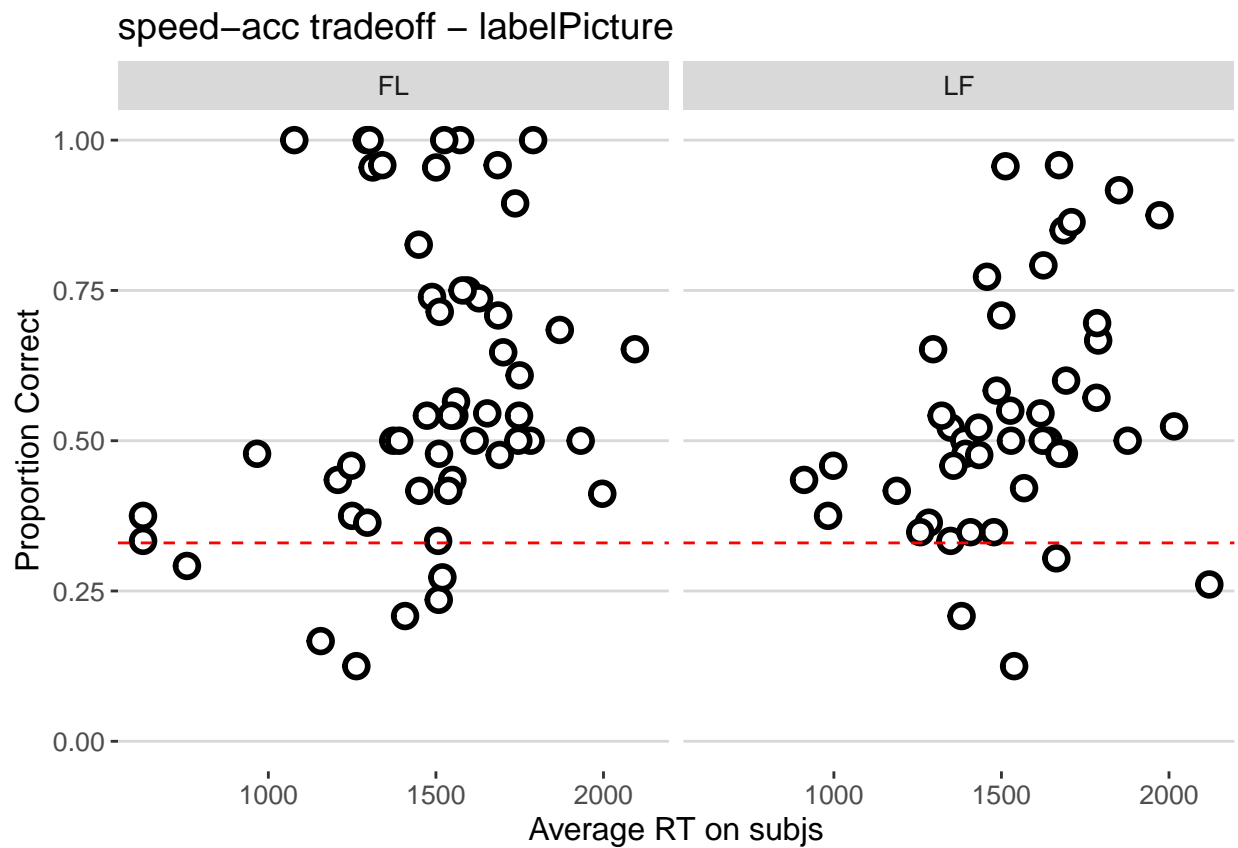
Inspection of the speed-accuracy trade-off:

Label Picture

```
aggregate(acc ~ subjID+learning, labelPicture[labelPicture$rt > 100 &
  !(labelPicture$subjID %in% badsubjs) ],, mean)-> speedacc

aggregate(rt ~ subjID+learning, labelPicture[labelPicture$rt > 100 &
  !(labelPicture$subjID %in% badsubjs)],, 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 - labelPicture")
```



```
aggregate(acc ~ subjID+learning+frequency, labelPicture[labelPicture$rt > 100 &
                                                         labelPicture$rt <= 2500 &
                                                         !(labelPicture$subjID %in% badsubjs)], mean)-> speedacc

aggregate(rt ~ subjID+learning+frequency, labelPicture[labelPicture$rt > 100 &
                                                         labelPicture$rt <= 2500 &
                                                         !(labelPicture$subjID %in% badsubjs)], mean)-> speedacc2

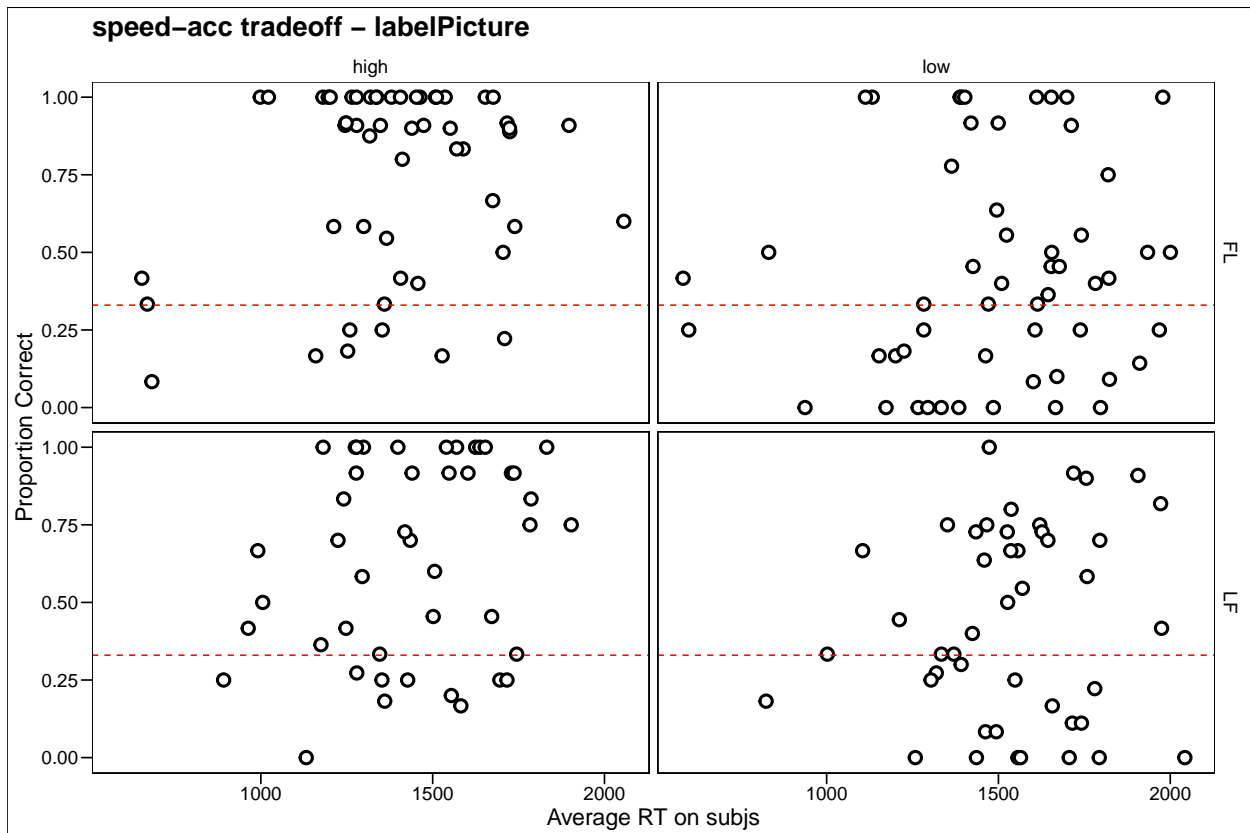
merge(speedacc, speedacc2, by = c("subjID", "learning", "frequency"))-> speedacc
rm(speedacc2)
dplyr::recode(speedacc$frequency, "25"="low", "75"="high")-> speedacc$frequency;

ggplot(aes(x=rt, y=acc),
       data = speedacc) +
```

```

facet_grid( learning ~ frequency) +
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_base()+
xlab("Average RT on subjs") +
ylab("Proportion Correct") +
ggtitle("speed-acc tradeoff - labelPicture")

```



```

speedacc %>%
  group_by(frequency, learning) %>%
  summarise(mean(rt), median(rt))

```

```

## # A tibble: 4 x 4
## # Groups:   frequency [2]
##   frequency learning `mean(rt)` `median(rt)`
##   <chr>      <fct>      <dbl>      <dbl>
## 1 high      FL          1390.      1373.
## 2 high      LF          1441.      1435.
## 3 low       FL          1489.      1504.
## 4 low       LF          1539.      1537.

```

PictureLabel

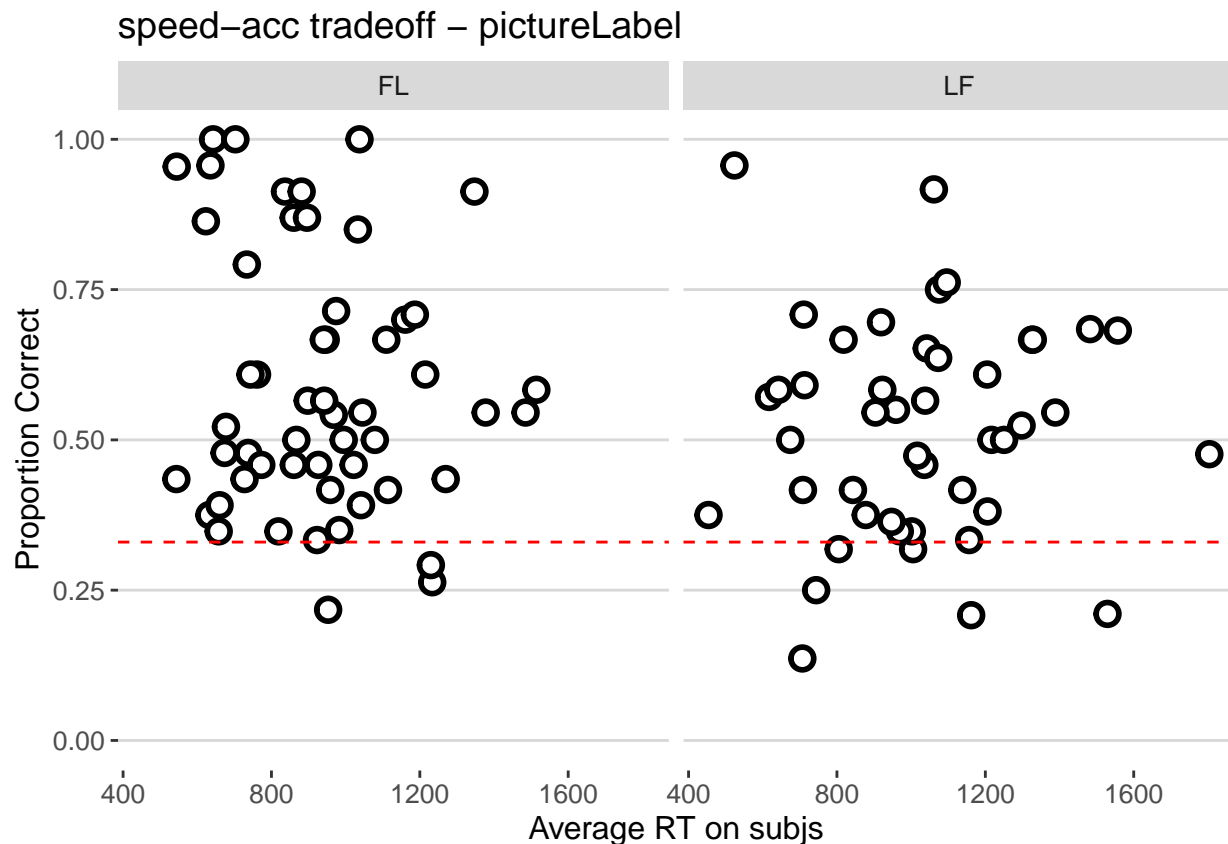
```

aggregate(acc ~ subjID+learning, pictureLabel[pictureLabel$rt > 100 &
!(pictureLabel$subjID %in% badsubjs),], mean)-> speedacc

aggregate(rt ~ subjID+learning, pictureLabel[pictureLabel$rt > 100 &
!(pictureLabel$subjID %in% badsubjs),], 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 - pictureLabel")

```



```

aggregate(acc ~ subjID+learning+frequency, pictureLabel[pictureLabel$rt > 100 &
!(pictureLabel$subjID %in% badsubjs),], mean)-> speedacc
aggregate(rt ~ subjID+learning+frequency, pictureLabel[pictureLabel$rt > 100 &
!(pictureLabel$subjID %in% badsubjs),], mean)-> speedacc2
merge(speedacc, speedacc2, by = c("subjID", "learning", "frequency"))-> speedacc
rm(speedacc2)

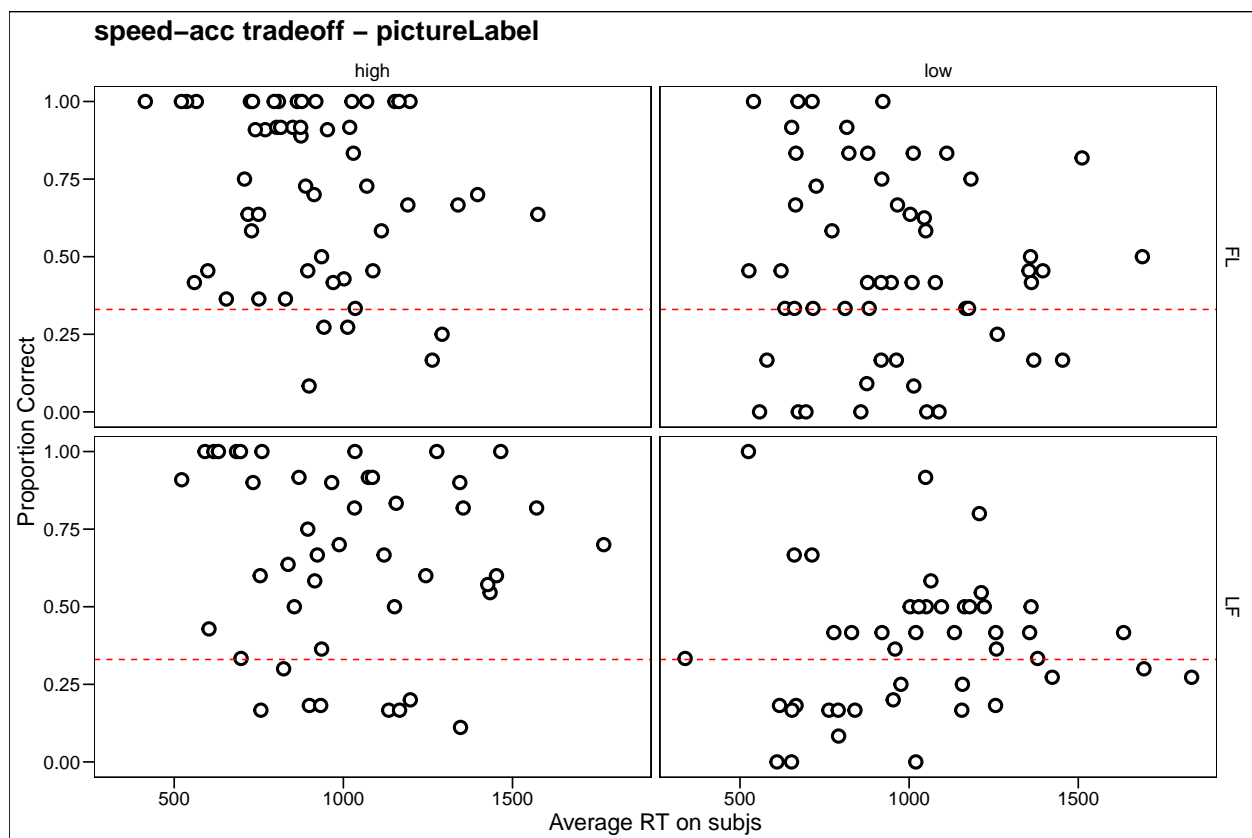
```

```

dplyr::recode(speedacc$frequency, "25"="low", "75"="high")-> speedacc$frequency;

ggplot(aes(x=rt, y=acc),
       data = speedacc) +
  facet_grid( learning ~ frequency) +
  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_base()+
  xlab("Average RT on subjs") +
  ylab("Proportion Correct") +
  ggtitle("speed-acc tradeoff - pictureLabel")

```



```

speedacc %>%
  group_by(frequency, learning) %>%
  summarise(mean(rt), median(rt))

```

```

## # A tibble: 4 x 4
## # Groups:   frequency [2]
##   frequency learning `mean(rt)` `median(rt)`
##   <chr>      <fct>      <dbl>      <dbl>
## 1 high      FL           911.       892.
## 2 high      LF          1016.      966.

```

```
## 3 low      FL      948.      918.
## 4 low      LF     1028.     1029.
```

Final Comparisons

Barplot labelPicture

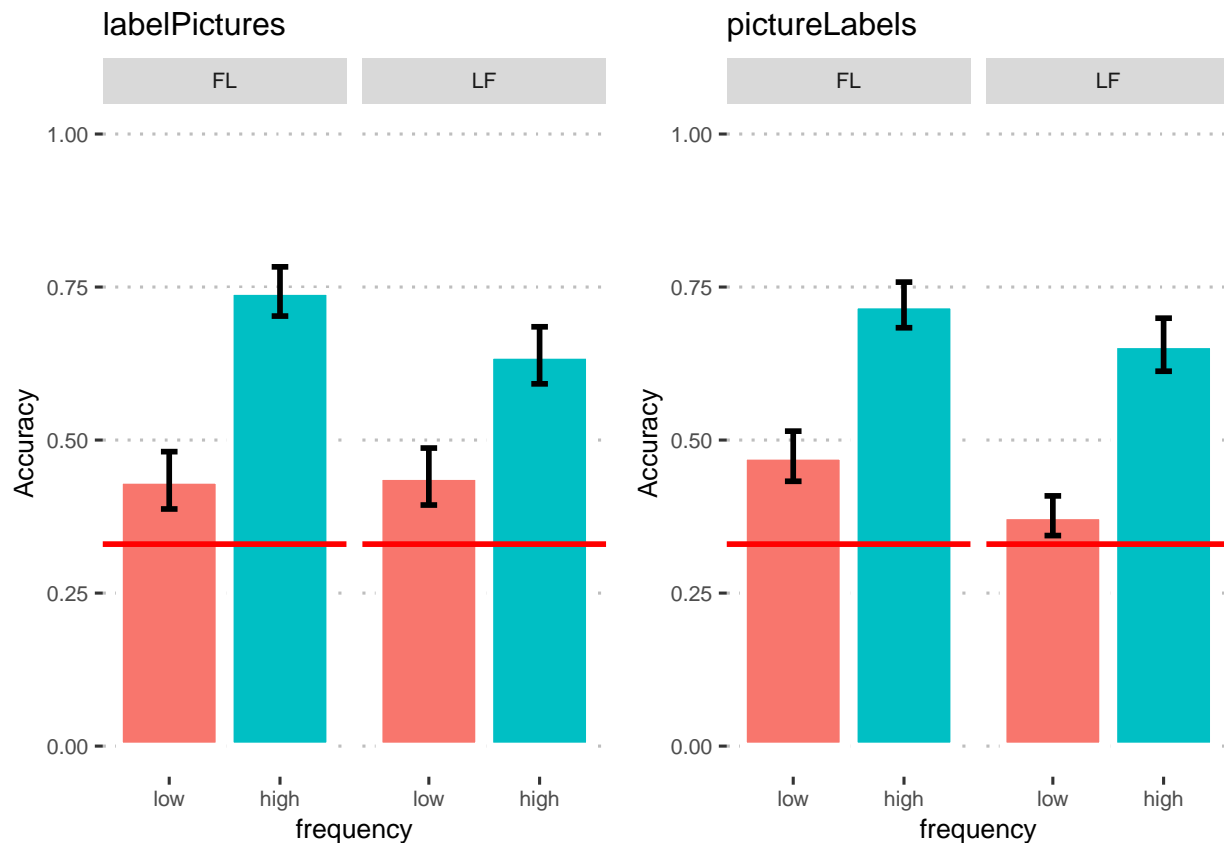
```
ms <- aggregate(acc ~ subjID+frequency+learning,
                 data=labelPicture[labelPicture$rt > 100 &
                                   !(labelPicture$subjID %in% badsubjs)],, FUN= mean)

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 = "none") +
  theme(text = element_text(size=10)) +
  geom_hline(yintercept = .33, col='red', lwd=1);

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



```
ms <- aggregate(acc ~ subjID+frequency+learning,
                 data=labelPicture[labelPicture$rt > 100 &
                                   labelPicture$rt <=2500 &
                                   !(labelPicture$subjID %in% badsubjs)],, FUN= 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;

lp_violin<- 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) +
  theme_pubclean()+
  theme(legend.position = "none") +
  geom_hline(yintercept = .33, col='red', lwd=1);

ms <- aggregate(acc ~ subjID+frequency+learning,
                 data=pictureLabel[pictureLabel$rt > 100 &
                                   !(pictureLabel$subjID %in% badsubjs)],, FUN= mean)

ms$frequency <- as.factor(ms$frequency)
```

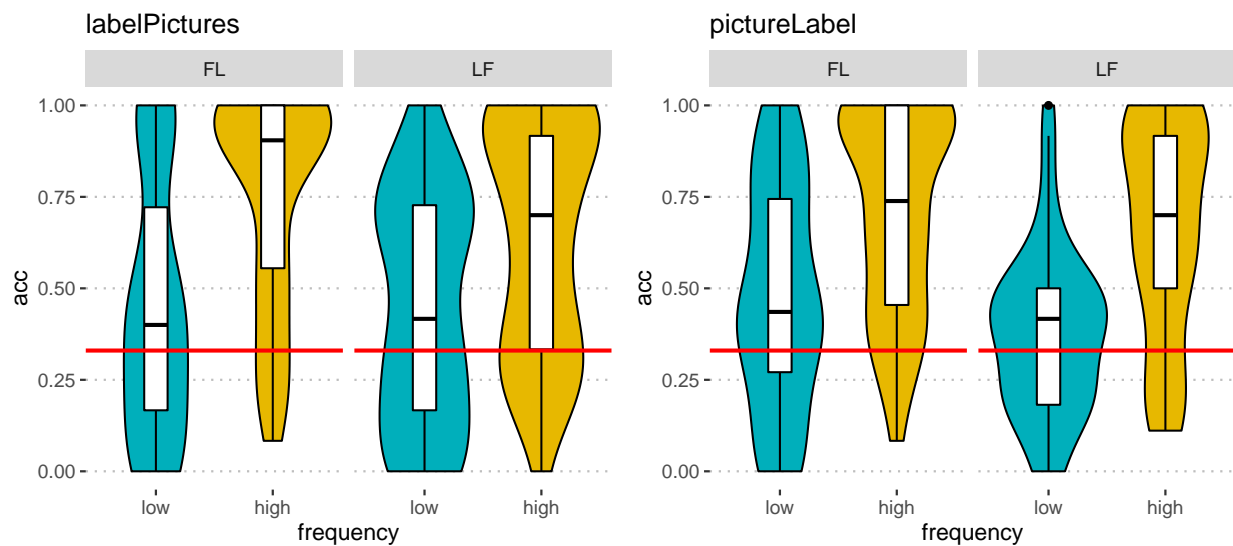
```

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

pl_violin<- ggviolin(ms, x = "frequency", y = "acc", fill = "frequency",
  palette = c("#00AFBB", "#E7B800"),
  add = "boxplot",
  add.params = list(fill = "white"),
  trim=TRUE) +
  ggtitle('pictureLabel') +
  facet_grid( . ~ learning) +
  theme_pubclean()+
  theme(legend.position = "none") +
  geom_hline(yintercept = .33, col='red', lwd=1);

grid.arrange(lp_violin, pl_violin, ncol=2)

```



```

#rm(ms, ss_prop)

```

Barplots + violinPlots with data from both tasks:

```

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

```

```

ms <- aggregate(acc ~ subjID+frequency+learning, data = genTask[genTask$rt>100 &
  genTask$Experiment.Version==ver2 &
  !(genTask$subjID %in% badsubjs),], 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",

```

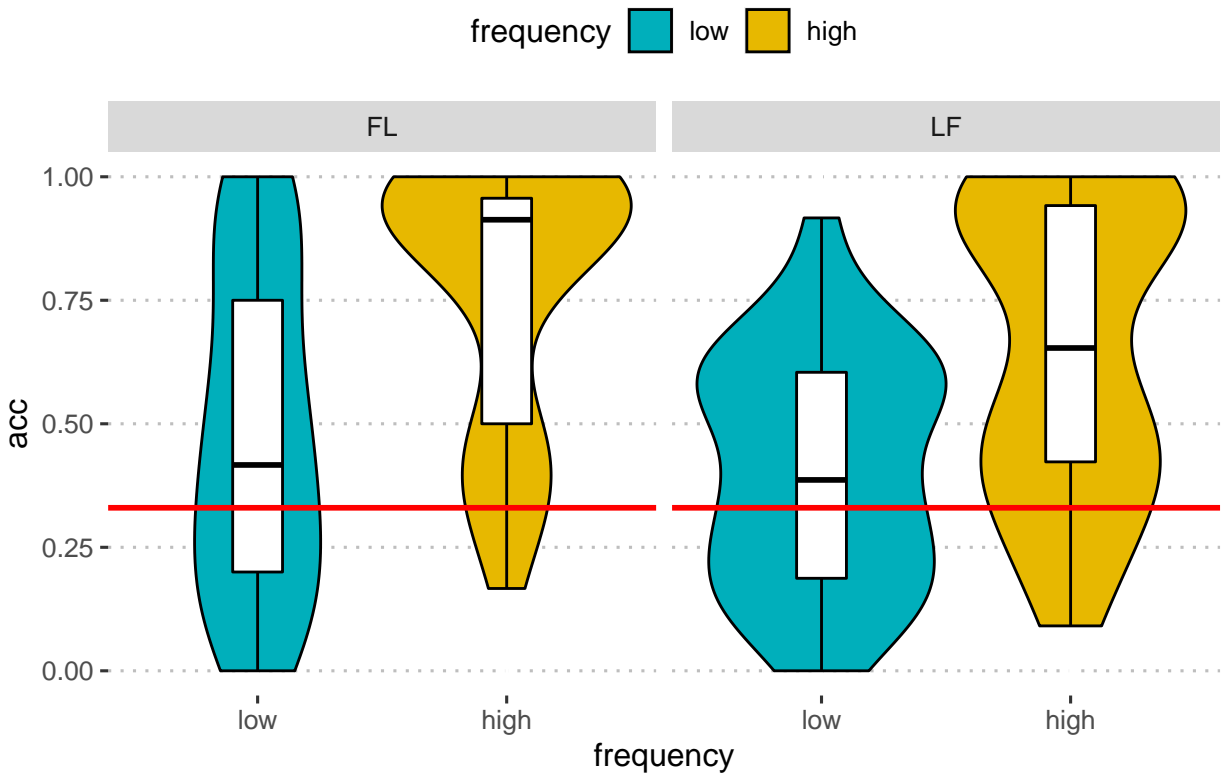


```

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);

```

labelPictures + pictureLabels



```

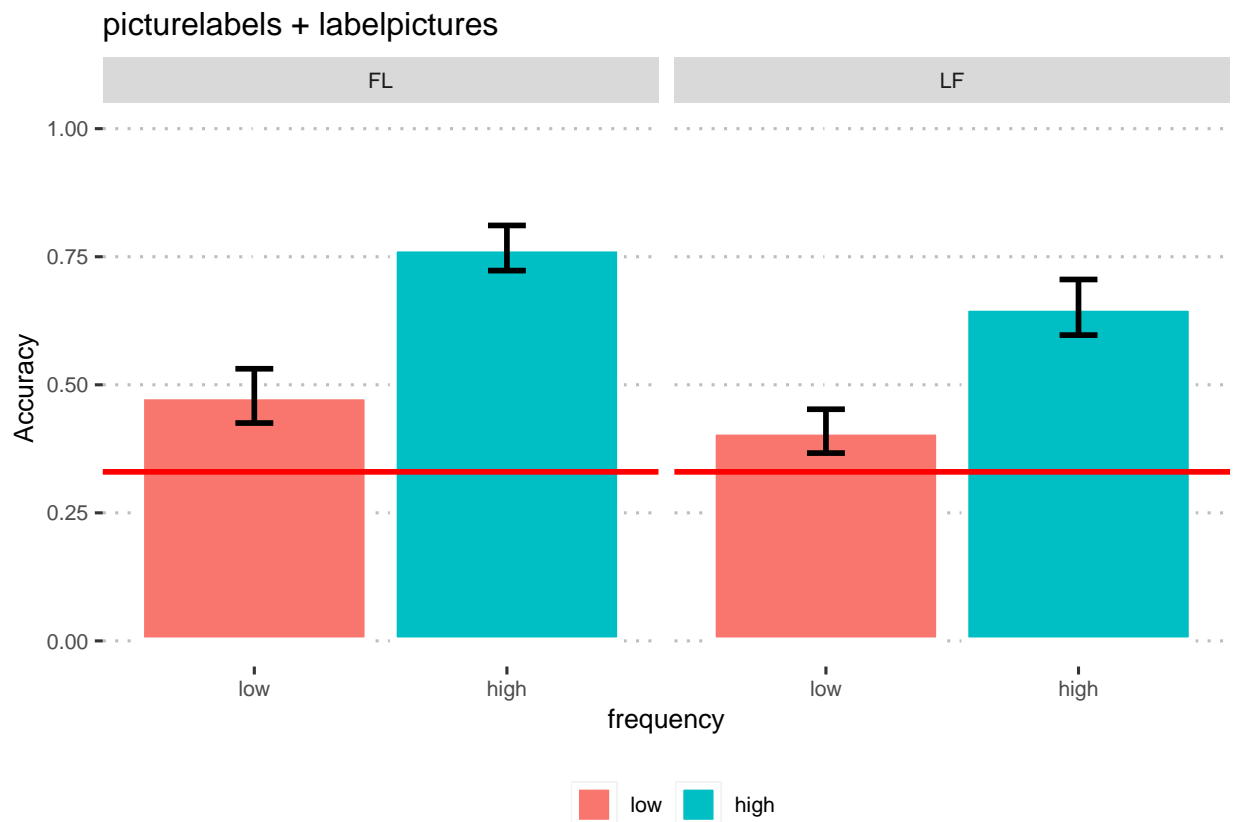
ms <- aggregate(acc ~ subjID+frequency+learning, data=genTask[genTask$rt>100 &
                                                                    genTask$Experiment.Version==ver2 &
                                                                    !(genTask$subjID %in% badsubjs)], FUN= mean)

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] 120
```

```
fl<- length(unique(contingencyJudgement[contingencyJudgement$learning=='FL' & contingencyJudgement$Expe
lf<- length(unique(contingencyJudgement[contingencyJudgement$learning=='LF' & contingencyJudgement$Expe
fl
```

```
## [1] 41
```

```
lf
```

```
## [1] 39
```

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

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

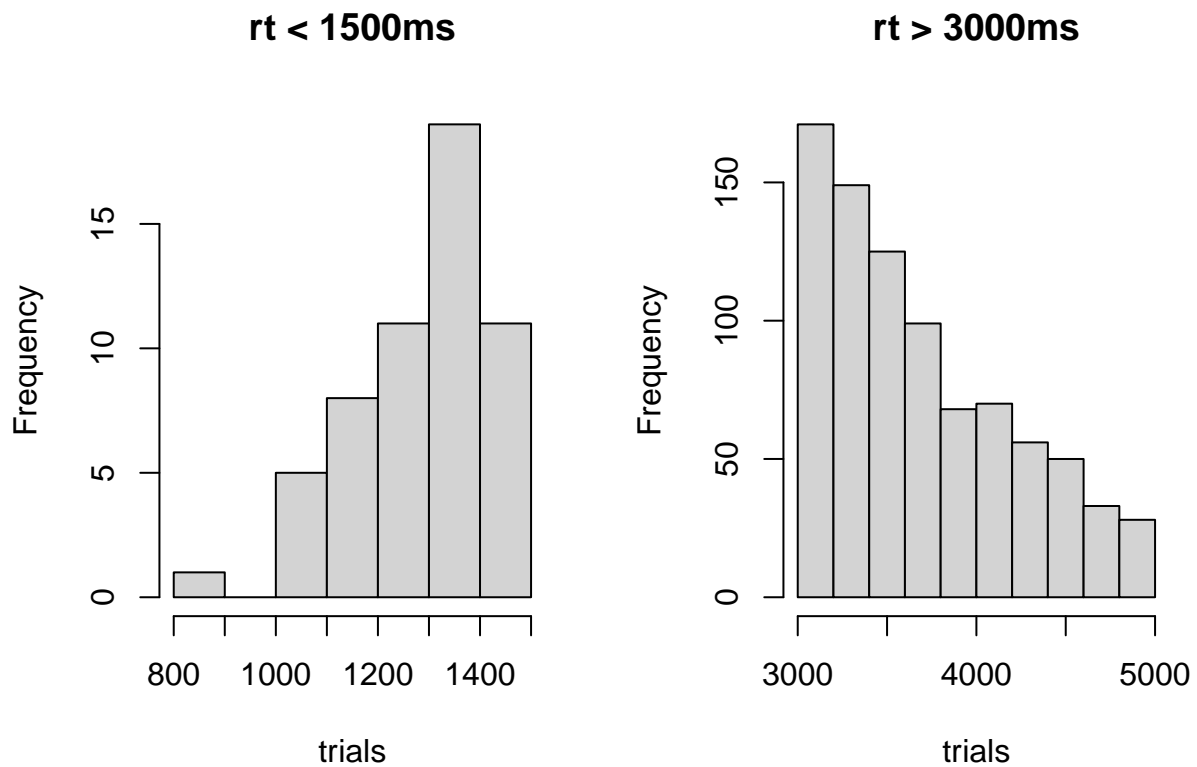
sort(unique(conjjudge$subjID))-> subjs;
sort(unique(contingencyJudgement$subjID)) ->totsubjs;

subjmissesd<- setdiff(totsubjs, subjs);

badsubjs <- c(badsubjs, subjmissesd)
badsubjs <- unique(badsubjs)
```

We have 111 participants in this task, so -9, and we have missed 559 over the total 2880, that is 19.4097222. The subject(s) that missed completely the task is/are: 1414932, 1420171, 1420199, 1422475, 1431960, 1431997, 1459020, 1459057, 1459078.

```
par(mfrow=c(1,2))
hist(conjjudge[conjjudge$rt<1500 & !(conjjudge$subjID %in% badsubjs),]$rt, main = 'rt < 1500ms', xlab = 't
hist(conjjudge[conjjudge$rt>3000 & !(conjjudge$subjID %in% badsubjs),]$rt, main = 'rt > 3000ms', xlab = 't
```



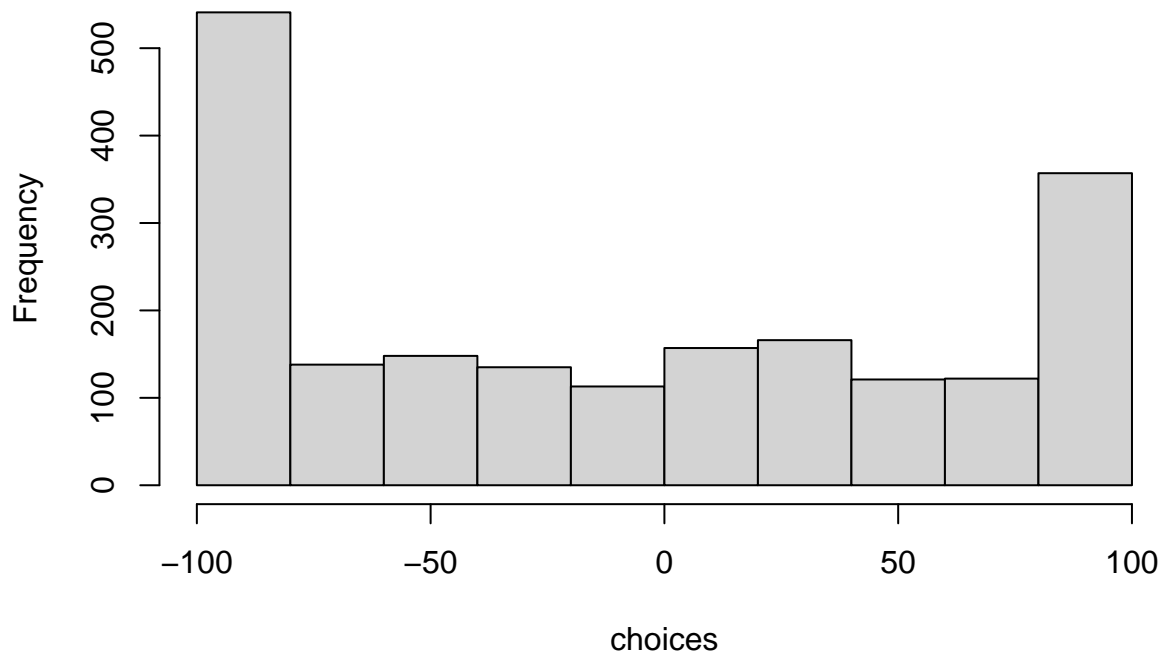
```
par(mfrow=c(1,1))
```

Resp is coded as factor, need to correct this:

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

```
hist(conjjudge[!(conjjudge$subjID %in% badsubjs),]$resp, main = 'resp distribution', xlab = 'choices')
```

resp distribution



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[!(conjudge$subjID %in% badsubjs)], FUN = mean)
```

```
##   category    resp
## 1         1 -12.119084
## 2         2  -6.564636
## 3         3 -10.635821
```

Okay, in this task one fribble was presented along with a label. The association between the fribble presented and the label could have been correct, or wrong. In this case then accuracy column does **not** refer to the participants' accuracy, but rather to the fribble-label pair presented. This should be therefore necessarily equal to the chance level, i.e, around 33%, of course this number is dependent by the number of datapoints left without no-responses because we filtered out those.

```
conjudge$acc <- 0;
conjudge[conjudge$category==1 & conjudge$label=='dep',]$acc <- 1;
conjudge[conjudge$category==2 & conjudge$label=='bim',]$acc <- 1;
conjudge[conjudge$category==3 & conjudge$label=='tob',]$acc <- 1;
```

```
mean(conjudge[!(conjudge$subjID %in% badsubjs),]$acc)
```

```
## [1] 0.3523524
```

Quite there, everything good.

```
respDistr<-summarySEwithin(data = conjudge[!(conjudge$subjID %in% badsubjs),], measurevar = "resp", bet
```

```
## Automatically converting the following non-factors to factors: frequency, category
```

```
## Loading required package: plyr
```

```
## -----
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
```

```
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
```

```
## library(plyr); library(dplyr)
```

```
## -----
```

```
##
```

```
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:ggpubr':
```

```
##
```

```
##      mutate
```

```
## The following objects are masked from 'package:dplyr':
```

```
##
```

```
##      arrange, count, desc, failwith, id, mutate, rename, summarise,
```

```
##      summarize
```

```
## The following object is masked from 'package:purrr':
```

```
##
```

```
##      compact
```

```
respDistr
```

```
##      learning frequency category label  N      resp  resp_norm      sd
## 1         FL        25          1  bim 67 -31.3582090 -14.1087315 51.86845
## 2         FL        25          1  dep 45  2.3333333 -28.6281805 56.62847
## 3         FL        25          1  tob 59 -0.6271186  16.7903079 59.40639
## 4         FL        25          2  bim 67  28.2537313   0.3038255 71.42933
## 5         FL        25          2  dep 60 -13.3166667   3.6104924 54.78085
## 6         FL        25          2  tob 53 -31.2075472  -2.6381954 43.77948
## 7         FL        25          3  bim 46 -33.7608696  -0.5991529 56.19839
## 8         FL        25          3  dep 71 -48.0281690 -31.5804323 48.20588
## 9         FL        25          3  tob 66  14.9848485  -8.1380523 69.60998
## 10        FL        75          1  bim 72 -29.7916667 -13.2997975 66.07554
## 11        FL        75          1  dep 47  24.8085106  -6.0097113 59.45004
## 12        FL        75          1  tob 66 -56.7272727 -39.7207891 50.47919
## 13        FL        75          2  bim 70  66.3142857  37.7407473 61.56664
## 14        FL        75          2  dep 65 -60.9230769 -39.6869158 47.20973
## 15        FL        75          2  tob 54 -53.1481481 -25.3652798 36.29847
## 16        FL        75          3  bim 47 -68.7872340 -34.7896631 43.58048
```

## 17	FL	75	3	dep 71	-49.6338028	-37.7716413	44.79396
## 18	FL	75	3	tob 70	63.1857143	39.3122969	60.83277
## 19	LF	25	1	bim 52	-24.0192308	-13.4298651	46.36911
## 20	LF	25	1	dep 56	5.2857143	-21.8581395	69.28297
## 21	LF	25	1	tob 40	-11.2750000	-6.3878238	60.86102
## 22	LF	25	2	bim 56	10.0714286	-23.3775019	51.38502
## 23	LF	25	2	dep 39	-6.8205128	4.7026189	57.82144
## 24	LF	25	2	tob 57	-42.3684211	-26.3358630	47.08686
## 25	LF	25	3	bim 38	11.1578947	26.3661505	63.27118
## 26	LF	25	3	dep 51	-38.2352941	-32.4228652	53.64881
## 27	LF	25	3	tob 54	5.3518519	-23.9193328	54.61213
## 28	LF	75	1	bim 54	-39.1851852	-28.4874570	36.67076
## 29	LF	75	1	dep 56	69.3035714	34.8370718	56.31310
## 30	LF	75	1	tob 41	-37.6341463	-31.9129596	61.39834
## 31	LF	75	2	bim 57	66.9473684	30.2636915	64.75691
## 32	LF	75	2	dep 37	-47.3783784	-36.2311056	61.97356
## 33	LF	75	2	tob 58	-27.8620690	-12.5594870	59.68045
## 34	LF	75	3	bim 39	-56.4615385	-41.2231689	46.00065
## 35	LF	75	3	dep 57	-14.8421053	-13.1330737	57.65721
## 36	LF	75	3	tob 60	57.7833333	25.8176773	65.50872
##	se	ci					
## 1	6.336740	12.651710					
## 2	8.441674	17.013075					
## 3	7.734053	15.481389					
## 4	8.726481	17.422982					
## 5	7.072178	14.151395					
## 6	6.013574	12.067118					
## 7	8.286001	16.688863					
## 8	5.720986	11.410147					
## 9	8.568396	17.112268					
## 10	7.787078	15.526992					
## 11	8.671680	17.455186					
## 12	6.213559	12.409334					
## 13	7.358621	14.680047					
## 14	5.855647	11.697999					
## 15	4.939596	9.907574					
## 16	6.356867	12.795709					
## 17	5.316065	10.602558					
## 18	7.270906	14.505061					
## 19	6.430238	12.909242					
## 20	9.258327	18.554101					
## 21	9.622972	19.464299					
## 22	6.866612	13.760999					
## 23	9.258840	18.743542					
## 24	6.236806	12.493825					
## 25	10.263941	20.796721					
## 26	7.512336	15.088970					
## 27	7.431769	14.906241					
## 28	4.990258	10.009189					
## 29	7.525155	15.080748					
## 30	9.588809	19.379707					
## 31	8.577262	17.182321					
## 32	10.188391	20.663014					
## 33	7.836426	15.692173					

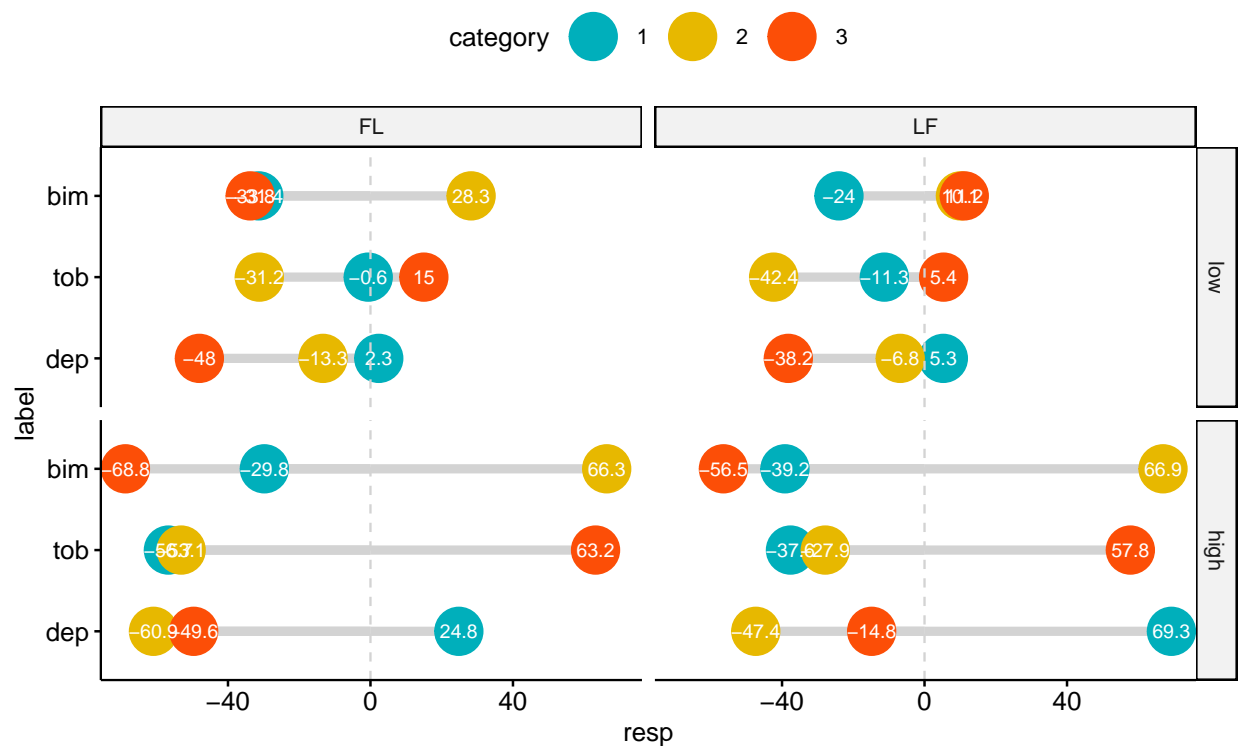
```
## 34 7.365999 14.911685
## 35 7.636884 15.298517
## 36 8.457139 16.922696
```

plot mean responses

```
plyr::revalue(respDistr$frequency, c("25"="low"))-> respDistr$frequency;
plyr::revalue(respDistr$frequency, c("75"="high"))-> respDistr$frequency;

lollipop<-ggdotchart(respDistr, x = "label", y = "resp",
  color = "category", # Color by groups
  palette = c("#00AFBB", "#E7B800", "#FC4E07"), # Custom color palette
  add = "segments", # Add segments from y = 0 to dots
  rotate = T,
  add.params = list(color = "lightgray", size = 2), # Change segment color and size
  group = "category", # Order by groups
  dot.size = 10, # Large dot size
  label = round(respDistr$resp,1), # Add mpg values as dot labels
  font.label = list(color = "white", size = 9, # Adjust label parameters
    vjust = 0.5), # ggplot2 theme
  ggtheme = theme_pubr()
)+ facet_grid( frequency ~ learning) +
  geom_hline(yintercept = 0, linetype = 2, color = "lightgray")

lollipop
```



Plot to compare with RW weights:


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

For each learning condition, look at their average score for each of the 6 combinations of frequency and type:

High frequency:

```
highFreqFL<-data.frame(
  learning = rep("FL",9),
  frequency = rep("high",9),
  type = c(rep("match",3),
            rep("mismatch-type1",3),
            rep("mismatch-type2",3)),
  label = c("dep_cat1", "bim_cat2", "tob_cat3"),
  fribble = c(1.1,2.1,3.1,
              3.1,1.1,2.1,
              2.1,3.1,1.1),
  fribbleCategory = c("cat1", "cat2", "cat3", #match
                     "cat3", "cat1", "cat2", #mis-type1
                     "cat2", "cat3", "cat1")) #mis-type2

highFreqLF<-data.frame(
  learning = rep("LF",9),
  frequency = rep("high",9),
  type = c(rep("match",3),
            rep("mismatch-type1",3),
            rep("mismatch-type2",3)),
  label = c("dep_cat1", "bim_cat2", "tob_cat3"),
  fribble = c(1.1,2.1,3.1,
              3.1,1.1,2.1,
              2.1,3.1,1.1),
  fribbleCategory = c("cat1", "cat2", "cat3", #match
                     "cat3", "cat1", "cat2", #mis-type1
                     "cat2", "cat3", "cat1")) #mis-type2

rbind(highFreqFL, highFreqLF)-> highFreq
rm(highFreqFL, highFreqLF)
```

Okay, let's fill each row:

```
resp <- c(
#-----FL LEARNING
# ROW 1                                     #MATCH
mean(conjudge[conjudge$learning=="FL" &
              conjudge$frequency=="high" &
              conjudge$label=="dep" &
              conjudge$category==1 &
              !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 2
mean(conjudge[conjudge$learning=="FL" &
              conjudge$frequency=="high" &
              conjudge$label=="bim" &
              conjudge$category==2 &
```

```

!(conjudge$subjID %in% badsubjs),]$resp),
# ROW 3
mean(conjudge[conjudge$learning=="FL" &
  conjudge$frequency=="high" &
  conjudge$label=="tob" &
  conjudge$category==3 &
  !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 4
mean(conjudge[conjudge$learning=="FL" &
  conjudge$frequency=="high" &
  conjudge$label=="dep" &
  conjudge$category==3 &
  !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 5
mean(conjudge[conjudge$learning=="FL" &
  conjudge$frequency=="high" &
  conjudge$label=="bim" &
  conjudge$category==1 &
  !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 6
mean(conjudge[conjudge$learning=="FL" &
  conjudge$frequency=="high" &
  conjudge$label=="tob" &
  conjudge$category==2 &
  !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 7
mean(conjudge[conjudge$learning=="FL" &
  conjudge$frequency=="high" &
  conjudge$label=="dep" &
  conjudge$category==2 &
  !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 8
mean(conjudge[conjudge$learning=="FL" &
  conjudge$frequency=="high" &
  conjudge$label=="bim" &
  conjudge$category==3 &
  !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 9
mean(conjudge[conjudge$learning=="FL" &
  conjudge$frequency=="high" &
  conjudge$label=="tob" &
  conjudge$category==1 &
  !(conjudge$subjID %in% badsubjs),]$resp),
#----- LF LEARNING
# ROW 1
mean(conjudge[conjudge$learning=="LF" &
  conjudge$frequency=="high" &
  conjudge$label=="dep" &
  conjudge$category==1 &
  !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 2
mean(conjudge[conjudge$learning=="LF" &
  conjudge$frequency=="high" &

```

```

        conjudge$label=="bim" &
        conjudge$category==2 &
        !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 3
mean(conjuge[conjuge$learning=="LF" &
        conjuge$frequency=="high" &
        conjuge$label=="tob" &
        conjuge$category==3 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 4                                     #MISMATCH -TYPE1
mean(conjuge[conjuge$learning=="LF" &
        conjuge$frequency=="high" &
        conjuge$label=="dep" &
        conjuge$category==3 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 5
mean(conjuge[conjuge$learning=="LF" &
        conjuge$frequency=="high" &
        conjuge$label=="bim" &
        conjuge$category==1 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 6
mean(conjuge[conjuge$learning=="LF" &
        conjuge$frequency=="high" &
        conjuge$label=="tob" &
        conjuge$category==2 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 7                                     #MISMATCH -TYPE2
mean(conjuge[conjuge$learning=="LF" &
        conjuge$frequency=="high" &
        conjuge$label=="dep" &
        conjuge$category==2 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 8
mean(conjuge[conjuge$learning=="LF" &
        conjuge$frequency=="high" &
        conjuge$label=="bim" &
        conjuge$category==3 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 9
mean(conjuge[conjuge$learning=="LF" &
        conjuge$frequency=="high" &
        conjuge$label=="tob" &
        conjuge$category==1 &
        !(conjuge$subjID %in% badsubjs),]$resp)
)
highFreq$resp <- resp

```

```
highFreq
```

##	learning	frequency	type	label	fribble	fribbleCategory	resp
## 1	FL	high	match	dep_cat1	1.1	cat1	24.80851
## 2	FL	high	match	bim_cat2	2.1	cat2	66.31429

## 3	FL	high	match	tob_cat3	3.1	cat3	63.18571
## 4	FL	high	mismatch-type1	dep_cat1	3.1	cat3	-49.63380
## 5	FL	high	mismatch-type1	bim_cat2	1.1	cat1	-29.79167
## 6	FL	high	mismatch-type1	tob_cat3	2.1	cat2	-53.14815
## 7	FL	high	mismatch-type2	dep_cat1	2.1	cat2	-60.92308
## 8	FL	high	mismatch-type2	bim_cat2	3.1	cat3	-68.78723
## 9	FL	high	mismatch-type2	tob_cat3	1.1	cat1	-56.72727
## 10	LF	high	match	dep_cat1	1.1	cat1	69.30357
## 11	LF	high	match	bim_cat2	2.1	cat2	66.94737
## 12	LF	high	match	tob_cat3	3.1	cat3	57.78333
## 13	LF	high	mismatch-type1	dep_cat1	3.1	cat3	-14.84211
## 14	LF	high	mismatch-type1	bim_cat2	1.1	cat1	-39.18519
## 15	LF	high	mismatch-type1	tob_cat3	2.1	cat2	-27.86207
## 16	LF	high	mismatch-type2	dep_cat1	2.1	cat2	-47.37838
## 17	LF	high	mismatch-type2	bim_cat2	3.1	cat3	-56.46154
## 18	LF	high	mismatch-type2	tob_cat3	1.1	cat1	-37.63415

Low frequency:

```
lowFreqFL<-data.frame(
  learning = rep("FL",9),
  frequency = rep("low",9),
  type = c(rep("match",3),
            rep("mismatch-type1",3),
            rep("mismatch-type2",3)),
  label = c("dep_cat1", "bim_cat2", "tob_cat3"),
  fribble = c(1.2,2.2,3.2,
              2.2,3.2,1.2,
              3.2,1.2,2.2),
  fribbleCategory = c("cat1", "cat2", "cat3", #match
                     "cat2", "cat3", "cat1", #mis-type1
                     "cat3", "cat1", "cat2")) #mis-type2

lowFreqLF<-data.frame(
  learning = rep("LF",9),
  frequency = rep("low",9),
  type = c(rep("match",3),
            rep("mismatch-type1",3),
            rep("mismatch-type2",3)),
  label = c("dep_cat1", "bim_cat2", "tob_cat3"),
  fribble = c(1.2,2.2,3.2,
              2.2,3.2,1.2,
              3.2,1.2,2.2),
  fribbleCategory = c("cat1", "cat2", "cat3", #match
                     "cat2", "cat3", "cat1", #mis-type1
                     "cat3", "cat1", "cat2")) #mis-type2

lowFreq<- rbind(lowFreqFL, lowFreqLF)
rm(lowFreqFL, lowFreqLF)
```

```
resp <- c(
#-----FL LEARNING
# ROW 1                                     #MATCH
mean(conjuge[conjuge$learning=="FL" &
```

```

        conjudge$frequency=="low" &
        conjudge$label=="dep" &
        conjudge$category==1 &
        !(conjudge$subjID %in% badsubjs),]$resp),
# ROW 2
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="bim" &
        conjuge$category==2 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 3
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="tob" &
        conjuge$category==3 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 4
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="dep" &
        conjuge$category==2 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 5
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="bim" &
        conjuge$category==3 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 6
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="tob" &
        conjuge$category==1 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 7
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="dep" &
        conjuge$category==3 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 8
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="bim" &
        conjuge$category==1 &
        !(conjuge$subjID %in% badsubjs),]$resp),
# ROW 9
mean(conjuge[conjuge$learning=="FL" &
        conjuge$frequency=="low" &
        conjuge$label=="tob" &
        conjuge$category==2 &
        !(conjuge$subjID %in% badsubjs),]$resp),

```

#MISMATCH -TYPE1

#MISMATCH -TYPE2

```

#-----LF LEARNING
# ROW 1
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="dep" &
  conjjudge$category==1 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 2
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="bim" &
  conjjudge$category==2 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 3
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="tob" &
  conjjudge$category==3 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 4
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="dep" &
  conjjudge$category==2 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 5
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="bim" &
  conjjudge$category==3 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 6
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="tob" &
  conjjudge$category==1 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 7
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="dep" &
  conjjudge$category==3 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 8
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="bim" &
  conjjudge$category==1 &
  !(conjjudge$subjID %in% badsubjs)],)$resp),
# ROW 9
mean(conjjudge[conjjudge$learning=="LF" &
  conjjudge$frequency=="low" &
  conjjudge$label=="tob" &

```

```

      conjudge$category==2 &
      !(conjudge$subjID %in% badsubjs),]$resp)
)
lowFreq$resp <- resp

```

```

rbind(highFreq, lowFreq)-> humansWeights
humansWeights$learning <- as.factor(humansWeights$learning); humansWeights$frequency <- as.factor(humansWeights$frequency)
rm(highFreq, lowFreq)
summary(humansWeights)

```

```

## learning frequency      type      label      fribble
## FL:18   high:18   match      :12   bim_cat2:12   Min.    :1.10
## LF:18   low :18   mismatch-type1:12   dep_cat1:12   1st Qu.:1.20
##                                mismatch-type2:12   tob_cat3:12   Median :2.15
##                                :12                                Mean    :2.15
##                                :12                                3rd Qu.:3.10
##                                :12                                Max.    :3.20
## fribbleCategory      resp
## cat1:12              Min.    :-68.79
## cat2:12              1st Qu.: -39.98
## cat3:12              Median :-19.43
##                      Mean    :-11.04
##                      3rd Qu.: 10.34
##                      Max.    : 69.30

```

```

dataWeight <- aggregate(resp ~ learning + frequency + type, data = humansWeights, FUN = mean)
dataWeight

```

```

## learning frequency      type      resp
## 1      FL      high      match   51.436170
## 2      LF      high      match   64.678091
## 3      FL      low       match   15.190638
## 4      LF      low       match   6.902998
## 5      FL      high mismatch-type1 -44.191206
## 6      LF      high mismatch-type1 -27.296453
## 7      FL      low mismatch-type1 -15.901552
## 8      LF      low mismatch-type1 -2.312539
## 9      FL      high mismatch-type2 -62.145861
## 10     LF      high mismatch-type2 -47.158021
## 11     FL      low mismatch-type2 -36.864642
## 12     LF      low mismatch-type2 -34.874315

```

```

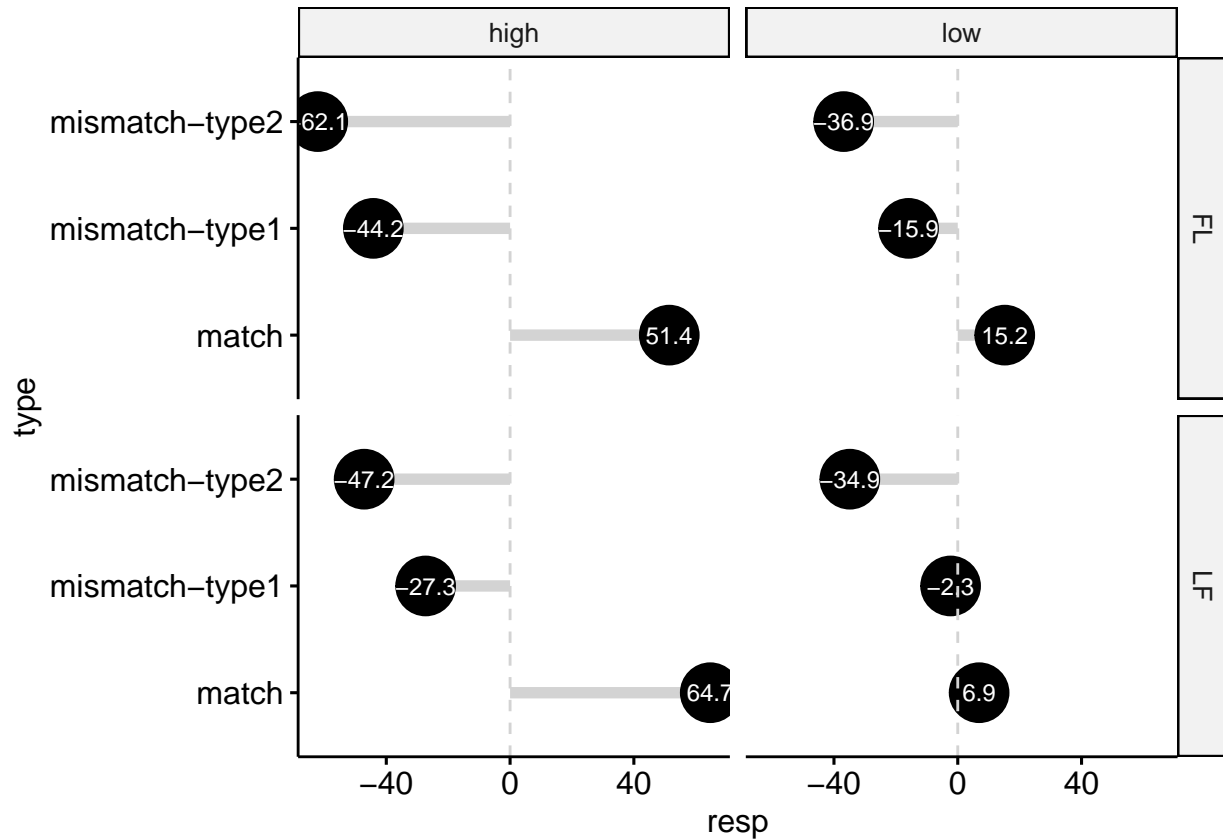
lollipopWeight<-ggdotchart(dataWeight, x = "type", y = "resp",
  #color = "learning", # Color by groups
  palette = c("#00AFBB", "#E7B800", "#FC4E07"), # Custom color palette
  add = "segments", # Add segments from y = 0 to dots
  rotate = T,
  add.params = list(color = "lightgray", size = 2), # Change segment color and size
  #group = "learning", # Order by groups
  dot.size = 10, # Large dot size
  label = round(dataWeight$resp,1), # Add mpg values as dot labels
  font.label = list(color = "white", size = 9,

```

```

                                vjust = 0.5),                # Adjust label parameters
                                ggtheme = theme_pubr()        # ggplot2 theme
                                )+ facet_grid( learning ~ frequency) +
                                geom_hline(yintercept = 0, linetype = 2, color = "lightgray")
lollipopWeight

```



Raw means

```

respShape<- summarySEwithin(data = conjudge[!(conjudge$subjID %in% badsubjs) & conjudge$acc==1,], measu
respShape

```

##	learning	frequency	bodyShape	N	resp	resp_norm	sd	se
## 1	FL	high	barrel	70	63.185714	67.863381	51.73751	6.183815
## 2	FL	high	jar	47	24.808511	32.653574	58.90257	8.591823
## 3	FL	high	trunk	70	66.314286	56.459166	55.08613	6.584052
## 4	FL	low	barrel	45	2.333333	9.495814	51.64421	7.698665
## 5	FL	low	jar	67	28.253731	19.320010	57.25256	6.994513
## 6	FL	low	trunk	66	14.984848	19.362189	57.67604	7.099430
## 7	LF	high	barrel	60	57.783333	60.207209	58.92314	7.606945
## 8	LF	high	jar	56	69.303571	66.621438	55.93900	7.475163
## 9	LF	high	trunk	57	66.947368	62.365961	56.85924	7.531191
## 10	LF	low	barrel	56	5.285714	5.962281	63.37117	8.468328
## 11	LF	low	jar	56	10.071429	8.621142	50.85262	6.795468
## 12	LF	low	trunk	54	5.351852	10.727451	63.93947	8.701059


```
##          ci
## 1  12.33637
## 2  17.29444
## 3  13.13482
## 4  15.51564
## 5  13.96500
## 6  14.17854
## 7  15.22146
## 8  14.98056
## 9  15.08679
## 10 16.97091
## 11 13.61842
## 12 17.45211
```

Let's clean the global environment:

```
rm(n_bins, rt_range, problematicPeople, frequency, dumbPeople, break_seq, task, temp, timeslice_range, )
```

```
## Warning in rm(n_bins, rt_range, problematicPeople, frequency, dumbPeople, :
## object 'n_bins' not found
```

```
## Warning in rm(n_bins, rt_range, problematicPeople, frequency, dumbPeople, :
## object 'rt_range' not found
```

```
## Warning in rm(n_bins, rt_range, problematicPeople, frequency, dumbPeople, :
## object 'break_seq' not found
```

```
## Warning in rm(n_bins, rt_range, problematicPeople, frequency, dumbPeople, :
## object 'timeslice_range' not found
```

```
## Warning in rm(n_bins, rt_range, problematicPeople, frequency, dumbPeople, :
## object 'count_range' not found
```

Bayes factor calculation with GLMMs

Estimates of the betas from the FLO paper

```
#means
highfreq_mean<- mean(88, 98)
lowfreq_mean <- mean(38, 78)

n <- c(32)

#sd
highfreq_sd <- c(5*sqrt(n)) #Paper has standard errors represented (I guess),
                        #I'm going to transform it back to standard deviations
lowfreq_sd <- c(5*sqrt(n)) #also, they look the same to me from the picture
                        #but low frequency should lead more variability.
```

Main effect of frequency:

```
frequency_beta<- logodds(highfreq_mean) - logodds(lowfreq_mean)
```

Main effect of learning:

```
#mean
LF_mean <- mean(38, 88)
FL_mean <- mean(78, 98)

n <- c(16)

#sd
LF_sd <- c(5*sqrt(n)) #how can be possible that learnings have the same se?
FL_sd <- c(5*sqrt(n))
```

```
learning_beta <- logodds(FL_mean) - logodds(LF_mean)
#positive > higher in the FL
```

Interaction between freq and learning:

Frequency effect (high-low) is greater in the LF than in FL:

```
#(logodds(highfreq_FL)-logodds(lowfreq_FL))- (logodds(highfreq_LF)-logodds(lowfreq_LF))
freqBylearning_beta <- (logodds(98)-logodds(78))- (logodds(88)-logodds(38))*-1
```

GLMMs with all tests separately

Picture label

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

```
pictureLabel$learning = relevel(pictureLabel$learning, ref = "LF")
pictureLabel$frequency = relevel(pictureLabel$frequency, ref = "low")
pictureLabel <- lizCenter(pictureLabel, list("learning" , "frequency", "task"))
```

```
summarySEwithin(data = pictureLabel[pictureLabel$rt > 100 & !(pictureLabel$subjID %in% badsubjs),], mea
```

```
##   learning frequency   N      acc acc_norm      sd      se      ci
## 1      FL      high 582 0.7079038 0.6729505 0.5831509 0.02417238 0.04747590
## 2      FL      low 611 0.4877250 0.4550235 0.6269022 0.02536175 0.04980694
## 3      LF      high 465 0.6688172 0.7084257 0.5947813 0.02758232 0.05420174
## 4      LF      low 494 0.3785425 0.4228856 0.6642473 0.02988590 0.05871944
```

```
piclab_model <- glmer(acc ~ frequency*learning + (frequency|subjID),
  data = pictureLabel[pictureLabel$rt > 100 & !(pictureLabel$subjID %in% badsubjs),],
  family="binomial",
  control=glmerControl(optimizer = "bobyqa"))
```

```
adjusted.piclab_model = adjust_intercept_model(piclab_model, chance = log(0.33/(1-0.33)))
round(adjusted.piclab_model,5)
```

```
##               Estimate Std. Error  z value Pr(>|z|)
## (Intercept)      0.09817    0.21988  0.44649  0.65524
## frequencyhigh     1.64364    0.35383  4.64523  0.00000
## learningFL        0.55500    0.29760  1.86492  0.06219
## frequencyhigh:learningFL -0.18451    0.47891 -0.38527  0.70004

piclab_model.emm <- emmeans(piclab_model , ~ frequency* learning )
contrast(piclab_model.emm, "consec", simple = "each", combine = F, adjust = "bonferroni")

## $`simple contrasts for frequency`
## learning = LF:
## contrast estimate SE df z.ratio p.value
## high - low      1.64 0.354 Inf 4.645 <.0001
##
## learning = FL:
## contrast estimate SE df z.ratio p.value
## high - low      1.46 0.328 Inf 4.448 <.0001
##
## Results are given on the log odds ratio (not the response) scale.
##
## $`simple contrasts for learning`
## frequency = low:
## contrast estimate SE df z.ratio p.value
## FL - LF         0.555 0.298 Inf 1.865 0.0622
##
## frequency = high:
## contrast estimate SE df z.ratio p.value
## FL - LF         0.370 0.387 Inf 0.957 0.3386
##
## Results are given on the log odds ratio (not the response) scale.
```

Label picture

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

labelPicture$learning = relevel(labelPicture$learning, ref = "LF")
labelPicture$frequency = relevel(labelPicture$frequency, ref = "low")
labelPicture <- lizCenter(labelPicture, list("learning" , "frequency", "task"))
```

```
summarySEwithin(data = labelPicture[labelPicture$rt > 100 & labelPicture$rt <=2500 & !(labelPicture$subjID %in%
```

```
## learning frequency N acc acc_norm sd se ci
## 1 FL high 586 0.7440273 0.7151810 0.5322962 0.02198895 0.04318691
## 2 FL low 554 0.4458484 0.4244508 0.5849702 0.02485300 0.04881783
## 3 LF high 484 0.6508264 0.6784148 0.6177757 0.02808071 0.05517544
## 4 LF low 460 0.4304348 0.4639248 0.6508264 0.03034494 0.05963222
```

```
labpic_model <- glmer(acc ~ frequency.ct*learning.ct + (frequency.ct|subjID),
  data = labelPicture[labelPicture$rt > 100 & labelPicture$rt <=2500 & !(labelPicture$subjID %in%
  family="binomial",
```

```
control=glmerControl(optimizer = "bobyqa"))

adjusted.labpic_model = adjust_intercept_model(labpic_model, chance = log(0.33/(1-0.33)))
round(adjusted.labpic_model,5)
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.32254    0.17272  7.65720  0.00000
## frequency.ct      1.94468    0.35298  5.50936  0.00000
## learning.ct       0.51646    0.33501  1.54163  0.12316
## frequency.ct:learning.ct  0.59238    0.68714  0.86208  0.38864
```

```
labpic_model.emm <- emmeans(labpic_model, ~ frequency.ct* learning.ct )
contrast(labpic_model.emm, "consec", simple = "each", combine = F, adjust = "bonferroni")
```

```
## $`simple contrasts for frequency.ct`
## learning.ct = -0.525:
## contrast              estimate      SE df z.ratio p.value
## 0.491247672253259 - -0.508752327746741      1.63 0.505 Inf 3.233  0.0012
##
## learning.ct = 0.475:
## contrast              estimate      SE df z.ratio p.value
## 0.491247672253259 - -0.508752327746741      2.23 0.480 Inf 4.637  <.0001
##
## Results are given on the log odds ratio (not the response) scale.
##
## $`simple contrasts for learning.ct`
## frequency.ct = -0.509:
## contrast              estimate      SE df z.ratio p.value
## 0.475232774674115 - -0.524767225325885      0.215 0.468 Inf 0.460  0.6455
##
## frequency.ct = 0.491:
## contrast              estimate      SE df z.ratio p.value
## 0.475232774674115 - -0.524767225325885      0.807 0.491 Inf 1.643  0.1003
##
## Results are given on the log odds ratio (not the response) scale.
```

Contingency judgement

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

```
## The following `from` values were not present in `x`: 25
```

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

```
## The following `from` values were not present in `x`: 75
```

```
conjjudge$learning = relevel(conjjudge$learning, ref = "FL")
conjjudge$frequency = relevel(conjjudge$frequency, ref = "low")
conjjudge <- lizCenter(conjjudge, list("learning" , "frequency"))
```

```
conjudge_model <- lmer(resp ~ learning * frequency +(frequency|subjID),
  data = conjudge[!(conjudge$subjID %in% badsubjs) & conjudge$acc==0,])
```

```
car::Anova(conjudge_model)
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: resp
##               Chisq Df Pr(>Chisq)
## learning      1.3493  1  0.2454058
## frequency     11.7018  1  0.0006244 ***
## learning:frequency 0.7078  1  0.4001847
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
conjudge_model.emm <- emmeans(conjudge_model , ~ learning* frequency )
contrast(conjudge_model.emm, "consec", simple = "each", combine = F, adjust = "bonferroni")
```

```
## $`simple contrasts for learning`
## frequency = low:
## contrast estimate SE df t.ratio p.value
## LF - FL      5.48 11.0 70.9 0.500  0.6187
##
## frequency = high:
## contrast estimate SE df t.ratio p.value
## LF - FL     16.13 11.2 71.8 1.434  0.1559
##
## Degrees-of-freedom method: kenward-roger
##
## $`simple contrasts for frequency`
## learning = FL:
## contrast estimate SE df t.ratio p.value
## high - low    -26.3 8.44 70 -3.111  0.0027
##
## learning = LF:
## contrast estimate SE df t.ratio p.value
## high - low    -15.6 9.45 72 -1.652  0.1030
##
## Degrees-of-freedom method: kenward-roger
```

Combine both generalization tasks in one dataset

I'm going to combine both generalization tasks in one single dataset called genTask

```
genTask <- rbind(labelPicture[labelPicture$rt > 100 & !(labelPicture$subjID %in% badsubjs),],
  pictureLabel[pictureLabel$rt > 100 & !(pictureLabel$subjID %in% badsubjs),])

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

```
## The following `from` values were not present in `x`: 25
```

```
plyr::revalue(genTask$frequency, c("75"="high"))-> genTask$frequency;
```

The following `from` values were not present in `x`: 75

Relevel the variables:

```
genTask$learning = relevel(genTask$learning, ref = "LF")
genTask$frequency = relevel(genTask$frequency, ref = "low")
genTask <- lizCenter(genTask, list("learning", "frequency", "task"))
```

The model

```
genTask_model <- glmer(acc ~ frequency.ct*learning.ct + task.ct + (frequency.ct|subjID) ,
  data = genTask,
  family="binomial",
  control=glmerControl(optimizer = "bobyqa"))

adjusted.genTask_model = adjust_intercept_model(genTask_model, chance = log(0.33/(1-0.33)))
round(adjusted.genTask_model,5)
```

	Estimate	Std. Error	z value	Pr(> z)
## (Intercept)	1.22227	0.13587	8.99560	0.00000
## frequency.ct	1.68180	0.26120	6.43863	0.00000
## learning.ct	0.47846	0.27079	1.76693	0.07724
## task.ct	-0.03490	0.07604	-0.45891	0.64630
## frequency.ct:learning.ct	0.28263	0.52044	0.54305	0.58710

Further inspection:

```
genTask_model.emm <- emmeans(genTask_model , ~ frequency.ct * learning.ct )
contrast(genTask_model.emm, "consec", simple = "each", combine = F, adjust = "bonferroni")
```

```
## $`simple contrasts for frequency.ct`
## learning.ct = -0.55:
## contrast                estimate    SE df z.ratio p.value
## 0.502322340919647 - -0.497677659080353    1.53 0.382 Inf 3.994  0.0001
##
## learning.ct =  0.45:
## contrast                estimate    SE df z.ratio p.value
## 0.502322340919647 - -0.497677659080353    1.81 0.356 Inf 5.087  <.0001
##
## Results are averaged over the levels of: task.ct
## Results are given on the log odds ratio (not the response) scale.
##
## $`simple contrasts for learning.ct`
## frequency.ct = -0.498:
## contrast                estimate    SE df z.ratio p.value
## 0.44960520204366 - -0.55039479795634    0.338 0.349 Inf 0.967  0.3333
##
```

```
## frequency.ct = 0.502:
## contrast estimate SE df z.ratio p.value
## 0.44960520204366 - -0.55039479795634 0.620 0.400 Inf 1.549 0.1213
##
## Results are averaged over the levels of: task.ct
## Results are given on the log odds ratio (not the response) scale.
```

Okay, with both tasks together the take home message is the following:

- Main effect of frequency, with high frequency having higher accuracy than low frequency in both learnings.
- Main effect of learning, with FL learning having higher accuracy in the high frequency condition.
- No difference between learnings in the low frequency condition.
- No difference between tasks

```
genTask %>%
  group_by(frequency, learning) %>%
  summarise(mean = mean(acc))
```

```
##      mean
## 1 0.5652578
```

```
summarySEwithin(data = genTask, measurevar = "acc", betweenvars = "learning", withinvars = "frequency",
```

```
## learning frequency N acc acc_norm sd se ci
## 1 FL high 1182 0.7225042 0.6930796 0.5647076 0.01642536 0.03222614
## 2 FL low 1188 0.4638047 0.4380815 0.6156339 0.01786135 0.03504334
## 3 LF high 961 0.6566077 0.6885886 0.6131101 0.01977774 0.03881260
## 4 LF low 975 0.4082051 0.4436979 0.6707224 0.02148031 0.04215301
```

I'm going to create a table with the estimates:

```
genTask_bf = data.frame(
  condition = c(
    "frequency by learning",
    "learning",
    "frequency",
    "task"
  ),

  meandiff = c(
    round(summary(genTask_model)$coefficients["frequency.ct:learning.ct", "Estimate"],3),
    round(summary(genTask_model)$coefficients["learning.ct", "Estimate"],3),
    round(summary(genTask_model)$coefficients["frequency.ct", "Estimate"],3),
    round(summary(genTask_model)$coefficients["task.ct", "Estimate"],3)
  ),

  se = c(
    round(summary(genTask_model)$coefficients["frequency.ct:learning.ct", "Std. Error"],3),
```

```

round(summary(genTask_model)$coefficients["learning.ct", "Std. Error"],3),
round(summary(genTask_model)$coefficients["frequency.ct", "Std. Error"],3),
round(summary(genTask_model)$coefficients["task.ct", "Std. Error"],3)
)
)

genTask_bf

```

```

##           condition meandiff    se
## 1 frequency by learning    0.283 0.520
## 2           learning    0.478 0.271
## 3           frequency    1.682 0.261
## 4              task   -0.035 0.076

```

BF for Frequency:

```

Bf(sd = genTask_bf[genTask_bf$condition=='frequency'],$se,
  obtained = genTask_bf[genTask_bf$condition=='frequency'],$meandiff,
  uniform = 0,
  sdtheory = highfreq_sd,
  meanoftheory = frequency_beta,
  tail = 1)

```

```

## $LikelihoodTheory
## [1] 0.028197
##
## $Likelihoodnull
## [1] 1.465403e-09
##
## $BayesFactor
## [1] 19241809

```

BF for learning:

```

Bf(sd = genTask_bf[genTask_bf$condition=='learning'],$se,
  obtained = genTask_bf[genTask_bf$condition=='learning'],$meandiff,
  uniform = 0,
  sdtheory = LF_sd,
  meanoftheory = learning_beta,
  tail = 1)

```

```

## $LikelihoodTheory
## [1] 0.03823377
##
## $Likelihoodnull
## [1] 0.3107198
##
## $BayesFactor
## [1] 0.123049

```


BF for the interaction frequency by learning

```
Bf(sd = genTask_bf[genTask_bf$condition=='frequency by learning'],$se,
  obtained = genTask_bf[genTask_bf$condition=='frequency by learning'],$meandiff,
  uniform = 0,
  sdtheory = LF_sd, #don't know how to compute sd of the interaction
  meanoftheory = freqBylearning_beta,
  tail = 1)
```

```
## $LikelihoodTheory
## [1] 0.02852534
##
## $Likelihoodnull
## [1] 0.6615924
##
## $BayesFactor
## [1] 0.04311618
```

```
rm(speedacc, n, lowfreq_mean, highfreq_mean, lowfreq_sd, highfreq_sd, LF_mean, FL_mean, LF_sd, FL_sd)
```

Summary of the results

We have collected 120 participants. Among these, 63 FL learning and 57 LF learning.

We had four tasks:

- Picture label task
- Label picture task
- Contingency judgement task
- Random dot task (attention check)

Participants that scored $\leq .5$ accuracy and had >3 timeouts in the attention check (random dot task) were removed from the analysis. Participants that skipped completely one of the tasks were removed. Participants that had very few datapoints, i.e., less than 1/2 also removed. In total for picture label task we had 52 for FL learning, and 43 for LF learning.

Raw means/sd for the effects.

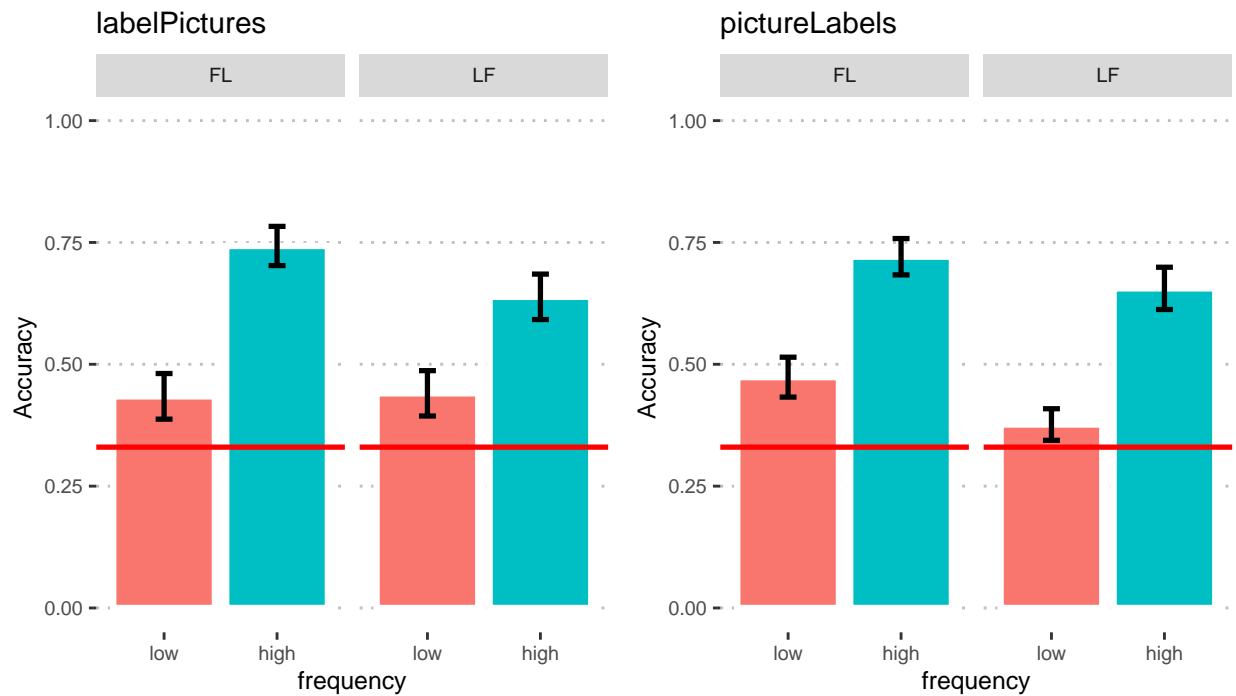
Label Picture:

##	learning	frequency	N	acc	acc_norm	sd	se	ci
## 1	FL	high	586	0.7440273	0.7151810	0.5322962	0.02198895	0.04318691
## 2	FL	low	554	0.4458484	0.4244508	0.5849702	0.02485300	0.04881783
## 3	LF	high	484	0.6508264	0.6784148	0.6177757	0.02808071	0.05517544
## 4	LF	low	460	0.4304348	0.4639248	0.6508264	0.03034494	0.05963222

Picture Label:

##	learning	frequency	N	acc	acc_norm	sd	se	ci
## 1	FL	high	582	0.7079038	0.6729505	0.5831509	0.02417238	0.04747590
## 2	FL	low	611	0.4877250	0.4550235	0.6269022	0.02536175	0.04980694
## 3	LF	high	465	0.6688172	0.7084257	0.5947813	0.02758232	0.05420174
## 4	LF	low	494	0.3785425	0.4228856	0.6642473	0.02988590	0.05871944

Data Visualization:



GLMMs models:

Picture label

##		Estimate	Std. Error	z value	Pr(> z)
##	(Intercept)	0.09817	0.21988	0.44649	0.65524
##	frequencyhigh	1.64364	0.35383	4.64523	0.00000
##	learningFL	0.55500	0.29760	1.86492	0.06219
##	frequencyhigh:learningFL	-0.18451	0.47891	-0.38527	0.70004

Label picture

##		Estimate	Std. Error	z value	Pr(> z)
##	(Intercept)	1.32254	0.17272	7.65720	0.00000
##	frequency.ct	1.94468	0.35298	5.50936	0.00000
##	learning.ct	0.51646	0.33501	1.54163	0.12316
##	frequency.ct:learning.ct	0.59238	0.68714	0.86208	0.38864

What we have learned from these data:

- Main effect of frequency, with high frequency having higher accuracy than low frequency in both learnings.
- Marginal effect of learning in the pictureLabel task, with FL learning having higher accuracy in the low frequency condition.
- No difference between learnings in the high frequency condition, although there is a trend for FL being higher than LF in the pictureLabel task
- What’s important here is that the two tasks seems to behave completely differently.

This means that the effect of frequency (high vs low) was super robust, and this is the only thing that we have replicated 100%. The difference between learnings unfortunately wasn’t there, although we see a trend in this direction in one task, but not the other. Why is this the case?

How do we explain these results: We don’t know for sure, however, throughout this experiment we have realised several important details that are not identical to the FLO paper and therefore could have affected the results:

- Learning: stimuli were pseudo-randomised with exemplars belonging to high and low frequency category of one category never displayed consequentially.
- The whole FLO experiment was visual, not audio, therefore this might cause less ambiguity, i.e., higher accuracy, and perfect balance in the test tasks for the trial duration. Also, this would remove the confound due to the addition of the sentence, in fact, we speculated that the two learnings varies in the contiguity between stimulus and label. I.e., FL: [fribble]+“This was a . . . X” Versus LF: “This is a . . .X”+[fribble] introduces two different types of lags between the presentation of the label and the stimulus. We speculated that this might cause differences, we don’t know how.
- Michael suggested that participants in his original experiment did only one of the two tasks, and not both. Exposition to both tasks might cause greater noise, especially in the labelPicture task where participants see 72 different fribbles. This might cause super confusion in the participants. Indeed, I found that from the folder Mike has shared with me (later on during this experiment) the number of stimuli didn’t match with the number of test trials reported in the paper.

What we’re going to do next: We’re going to re-do the replication! This time for real: by checking for the right amount of test trials, same fribbles used by Michael and same modality.