Piloting COLA tasks

Adam Parker

24th August 2020

This R Markdown file processes the various new tasks that were piloted for inclusion in a registered report.

This script analyses the following: 1) Feedback 2) Questionnaires - demographics - spelling dictation - spelling recognition - grammar game 3) Rhyme decision task 4) OVP 5) Dichotic listening 6) Colour scales 7) Semntic deicison task

Feedback

First, let's have a look at the feedback on the tasks. This will be useful for identifying any issues. We'll print the output, but it may not be that easy on the eyes.

```
# load
feedback <- read.csv("./data/piloting/data_exp_23909-v5_questionnaire-8xl2.csv",stringsAsFactors = F)

# relabel
feedback$subject <- feedback$Participant.Public.ID
  feedback$subject <- as.factor(feedback$subject)
# count and print subjects
nsub <-length(levels(feedback$subject))

for (i in 1:nsub) { # loop through subjects
# subset
  subname <- levels(feedback$subject)[i] # find subject subject
  myrows <- which(feedback$subject==subname) # select rows for this subject
  tmp <- data.frame(feedback[myrows,])
  # extract
  print(paste0("subject: ", subname, ". Feedback: ", tmp[tmp$Question.Key == "response-1",]$Response))
}
```

```
## [1] "subject: . Feedback: "
## [1] "subject: 5bf0357856a321000106ab8b. Feedback: everything ran fine! "
## [1] "subject: 5c90bb590d21a20017ec455d. Feedback: The study operated as it should, however, despite r
## [1] "subject: 5ce33217f1117f0017c80de1. Feedback: Everything ran ok x"
## [1] "subject: 5d544ce3be713b0001841f1e. Feedback: There were several times throughout the session the
## [1] "subject: 5d6c3abc4ecaee0001e3163d. Feedback: Yes fine"
## [1] "subject: 5de95a309ba3e808f0c16788. Feedback: I started to get a headache from looking at the sc
## [1] "subject: 5e1cf0eb65b6d3071f489de9. Feedback: all went okay, though sometimes some of the instru
## [1] "subject: 5e3eef5b7642cd1506ea4bf3. Feedback: yes"
## [1] "subject: 5e53c72e7a87a8000fba236c. Feedback: fine"
## [1] "subject: 5e696506e570590a12418b2f. Feedback: I found the line ones particulary chellenging but y
```

```
## [1] "subject: 5e9da14442f41809eb155db4. Feedback: Yes, all ok"
## [1] "subject: 5e9dd8f9d9428d1371472f42. Feedback: All good. I found the colours really hard! "
## [1] "subject: 5eb2bbd8ca570703c4136c78. Feedback: Interesting challenge, Despite being sat here fully
## [1] "subject: 5eb2d4b5444b2f06e7257f77. Feedback: none"
## [1] "subject: 5eb554340c06143e826cc8e1. Feedback: Everything ran well!"
## [1] "subject: 5ecb75a89b41896d3d07e48c. Feedback: yes it all went fine."
## [1] "subject: 5ecf166b3314d1326d49dd7a. Feedback: Thanks please send more of the same type of survey
## [1] "subject: 5ed1570aabecc609ed7ebd0f. Feedback: Everything seemed fine"
## [1] "subject: 5ed92350fb714c483e9f477d. Feedback: sometimes there was a delay for the recognising wo
## [1] "subject: 5ed934c19e6b4a496cd6da06. Feedback: For the four-letter typing test (l hand / r hand) i
## [1] "subject: 5eeaa7b09312211d1553c286. Feedback: Everything ran very well, i have really enjoyed thi
## [1] "subject: 5ef660beebde2833d8492e58. Feedback: The colour test was difficult (which is blacker etc
## [1] "subject: 5efdad0bce3c552031b572a8. Feedback: Yes everything ran ok but I absolutely could not te
## [1] "subject: 5f03645306e72126281ebd71. Feedback: Yes"
## [1] "subject: 5f084541958a6502a4f7233d. Feedback: thought it started out interesting but got repetetiv
## [1] "subject: 5f1b30fc4dd4ab11cf1f6503. Feedback: For some of the tasks the instructions were all on
## [1] "subject: 5f21ac0441a50c10f31ddf96. Feedback: Everything ran ok, thank you."
## [1] "subject: 5f26b1901a468541fdc90ecf. Feedback: Everything ran ok"
## [1] "subject: 5f2a8d05a252862c152d180f. Feedback: All good\n"
## [1] "subject: 5f32ce09e1ca2a06721836a3. Feedback: Everything went ok, was actually an interesting stu
```

Generally, it seems that the tasks were fairly long and a few Ps had difficulties with long refresh rates.

The following Ps might be worth removing from OVP: - 5eb2bbd8ca570703c4136c78 - 5ed92350fb714c483e9f477d

This participants reported issues with dichotic: - 5f1b30fc4dd4ab11cf1f6503

Questionnaires

This R Markdown processes the following questionnaires from the Online battery:

- demographics
- spelling measures
- word game

Once processed, a file is made.

Basic demographics

First, the demographics is processed.This gives us a participant subject, age, gender, footedness, and handedness.

```
demo_dat <- read.csv("./data/piloting/data_exp_23909-v5_questionnaire-mjwu.csv",stringsAsFactors = F)

# relabel
demo_dat$subject <- demo_dat$Participant.Public.ID
  demo_dat$subject <- as.factor(demo_dat$subject)
# count and print subjects
nsub <-length(levels(demo_dat$subject))

# create data frame
allsum <- data.frame(matrix(ncol = 8, nrow = nsub))
```

```
  colnames(allsum) <-
    c("subject", "age", "gender", "footedness", "handedness", "education", "bilingual", "strongest.langu
myrow <- 0 # start row counter
for (i in 1:nsub) { # loop through subjects
  subname <- levels(demo_dat$subject)[i] # find subject subject
  myrows <- which(demo_dat$subject==subname) # select rows for this subject
  tmp <- data.frame(demo_dat[myrows,])

    myrow <- i

    allsum$subject[myrow] <- subname
    allsum$age[myrow] <- as.numeric(tmp[tmp$Question.Key == "Age",]$Response)
    allsum$gender[myrow] <- tmp[tmp$Question.Key == "Gender",]$Response
    allsum$footedness[myrow] <- tmp[tmp$Question.Key == "footedness",]$Response
    allsum$handedness[myrow] <- tmp[tmp$Question.Key == "categorical_hand",]$Response
    allsum$bilingual[myrow] <- tmp[tmp$Question.Key == "bilingual",]$Response
    allsum$strongest.language[myrow] <- tolower(tmp[tmp$Question.Key == "bilingual-text",]$Response)
    allsum$education[myrow] <- tolower(tmp[tmp$Question.Key == "education",]$Response)

    allsum$bilingual <- as.character(allsum$bilingual)
    allsum$bilingual <- gsub('. My strongest language is:', '', allsum$bilingual)
}

#create a numerical code for handedness - can be used in plots to distinguish handedness groups (eg if
allsum$handpch <-1  #numeric code for plotting handedness
allsum$handpch[allsum$handedness=='Left'] <-2

#numerical code for gender - as we have only one 'prefer not to say' we will assign to NA

allsum$genderpch <-2
w<-which(allsum$gender=='Male')
allsum$genderpch[w] <-1
w<-which(allsum$gender=='Prefer not to say')
allsum$genderpch[w]<-NA
```

This indicates that 15/30 participants reported being right handed. Let's check the EHI for this.

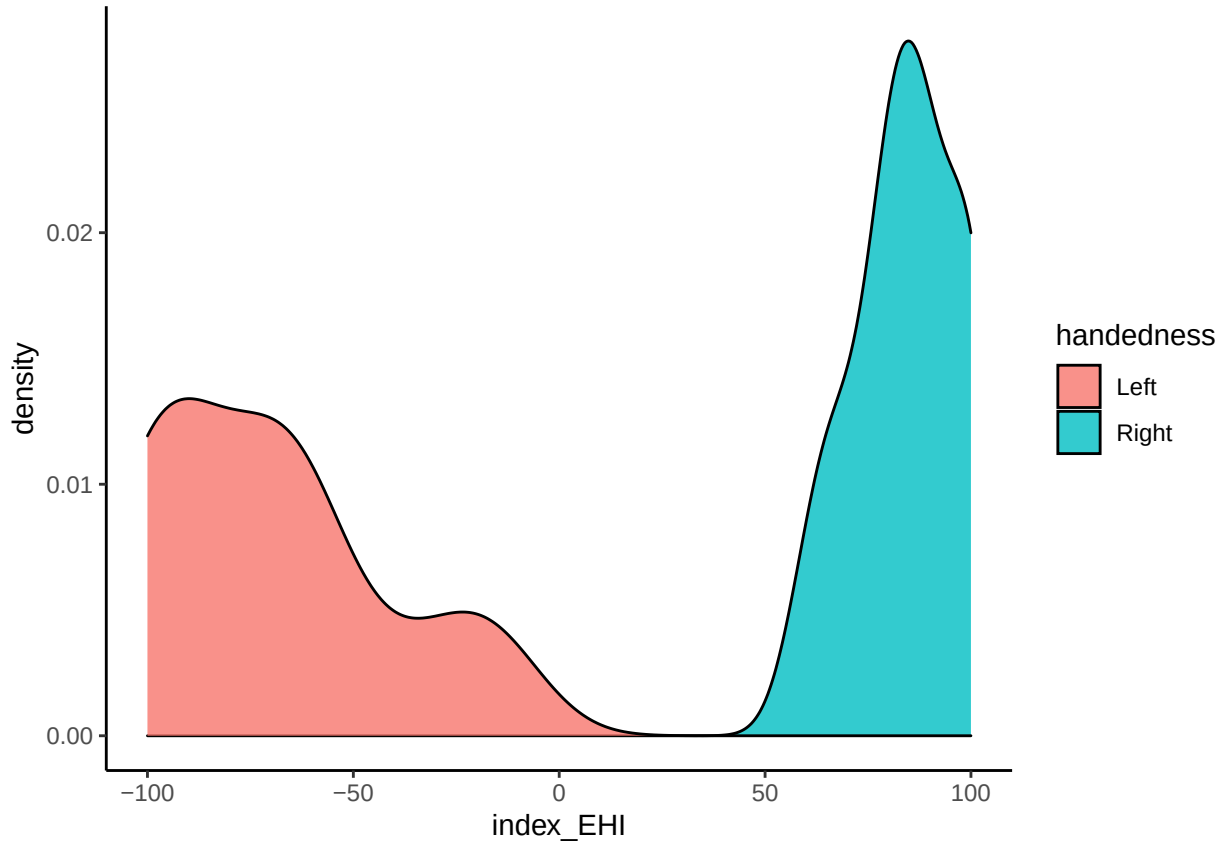
Edinburgh Handedness

Now we add the EHI. Here we score and plot the Edinburgh handedness inventory (Oldfield, 1971) in its short form.

```
EHI <- read.csv("./data/piloting/data_exp_23909-v5_questionnaire-bczg.csv",stringsAsFactors = F)

# select only responses
EHI <- EHI[EHI$Question.Key == "response-2-quantised" | EHI$Question.Key == "response-3-quantised" |
           EHI$Question.Key == "response-4-quantised" | EHI$Question.Key == "response-5-quantised" |
           EHI$Question.Key == "response-6-quantised" | EHI$Question.Key == "response-7-quantised" |
           EHI$Question.Key == "response-8-quantised" | EHI$Question.Key == "response-9-quantised" |
           EHI$Question.Key == "response-10-quantised" | EHI$Question.Key == "response-11-quantised" |
           EHI$Question.Key == "response-12-quantised" | EHI$Question.Key == "response-13-quantised" |
           EHI$Question.Key == "response-14-quantised" | EHI$Question.Key == "response-15-quantised" |
```

```
          EHI$Question.Key == "response-16-quantised" | EHI$Question.Key == "response-17-quantised" |
          EHI$Question.Key == "response-18-quantised" | EHI$Question.Key == "response-19-quantised" |
          EHI$Question.Key == "response-20-quantised",]

# recode variables
EHI$subject <- EHI$Participant.Public.ID
EHI$subject <- as.factor(EHI$subject)

# reduce data
# new data
meaningful <- c("subject", "Question Key", "Response") # select wanted columns
c<-which(names(EHI) %in% meaningful) #find colnumbers of unwanted
EHI <-EHI[,c] #remove unwanted columns
EHI <- na.omit(EHI) # remove NAs

# score responses according to the original scoring method in Oldfield (1971)
EHI$right_hand <- 0
EHI$left_hand <- 0
for (r in 1:nrow(EHI)) {
  if(EHI$Response[r] == 1){
    EHI$right_hand[r]= 2
  } else {
    if(EHI$Response[r] == 2){
      EHI$right_hand[r]= 1
    } else {
      if(EHI$Response[r] == 4){
        EHI$left_hand[r]= 1
      } else {
        if(EHI$Response[r] == 5){
          EHI$left_hand[r]= 2
        } else {
          (EHI$right_hand[r]= 1) & (EHI$left_hand[r]= 1)
        }
      }
    }
  }
}

# re-reduce data
# new data
meaningful <- c("subject", "right_hand", "left_hand") # select wanted columns
c<-which(names(EHI) %in% meaningful) #find colnumbers of unwanted
EHI <-EHI[,c] #remove unwanted columns
EHI <- na.omit(EHI) # remove NAs

# find sum for left and right for each participant
# create hand data
nsub <-length(levels(EHI$subject))
EHI2 <- data.frame(matrix(ncol = 3, nrow = nsub))
  colnames(EHI2) <- c("subject", "right_hand", "left_hand")


for (i in 1:nsub) { # loop through subjects
```

```
  myrow <- i
  subname <- levels(EHI$subject)[i] # find subject subject
  myrows <- which(EHI$subject==subname) # select rows for this subject
  tmp <- data.frame(EHI[myrows,])

    EHI2$subject[myrow] <- subname # add row for subject
    EHI2$right_hand[myrow] <- sum(tmp$right_hand) # sum for R hand
    EHI2$left_hand[myrow] <- sum(tmp$left_hand) # sum for L hand
  }

EHI2$index_EHI <- 100*(EHI2$right_hand-EHI2$left_hand)/(EHI2$right_hand+EHI2$left_hand)

# now create a data frame only for index
# merge with allsum
allsum$index_EHI <- NA #initialise
for (i in 1:nrow(allsum)){
  thissub <- allsum$subject[i]
  w<-which(EHI2$subject==thissub)
 allsum$index_EHI[i] <- EHI2$index_EHI[w]
}


ggplot(allsum, aes(x=index_EHI)) + geom_density(aes(group=handpch,fill=handedness), alpha= .8) + theme_
```



It looks as though all left handers fall less than 0 and all right handers fall above.

Spelling measures

Next, let's process some spelling measures. I wanted to pilot these here to see if I could move some simple measures online. I also had a curiousity of how these short tasks relate to language lateralisation.

Spelling dictation

First, we score a dictation task with 20 items.

```
dictation <- read.csv("./data/piloting/data_exp_23909-v5_task-skrc.csv",stringsAsFactors = F)

# recode variables
dictation$subject <- dictation$Participant.Public.ID
  dictation$subject <- as.factor(dictation$subject)

# remove to corrrect rows
dictation <- dictation[dictation$display== "task",]
dictation <- dictation[dictation$Zone.Type== "response_text_entry",]

dict_agg <- aggregate(data= dictation, FUN= mean, Correct~ subject)
names(dict_agg)[2] <- "spelling.dictation"

allsum <- merge(allsum, dict_agg, by.x= "subject", by.y = "subject")
```

Spelling recongition

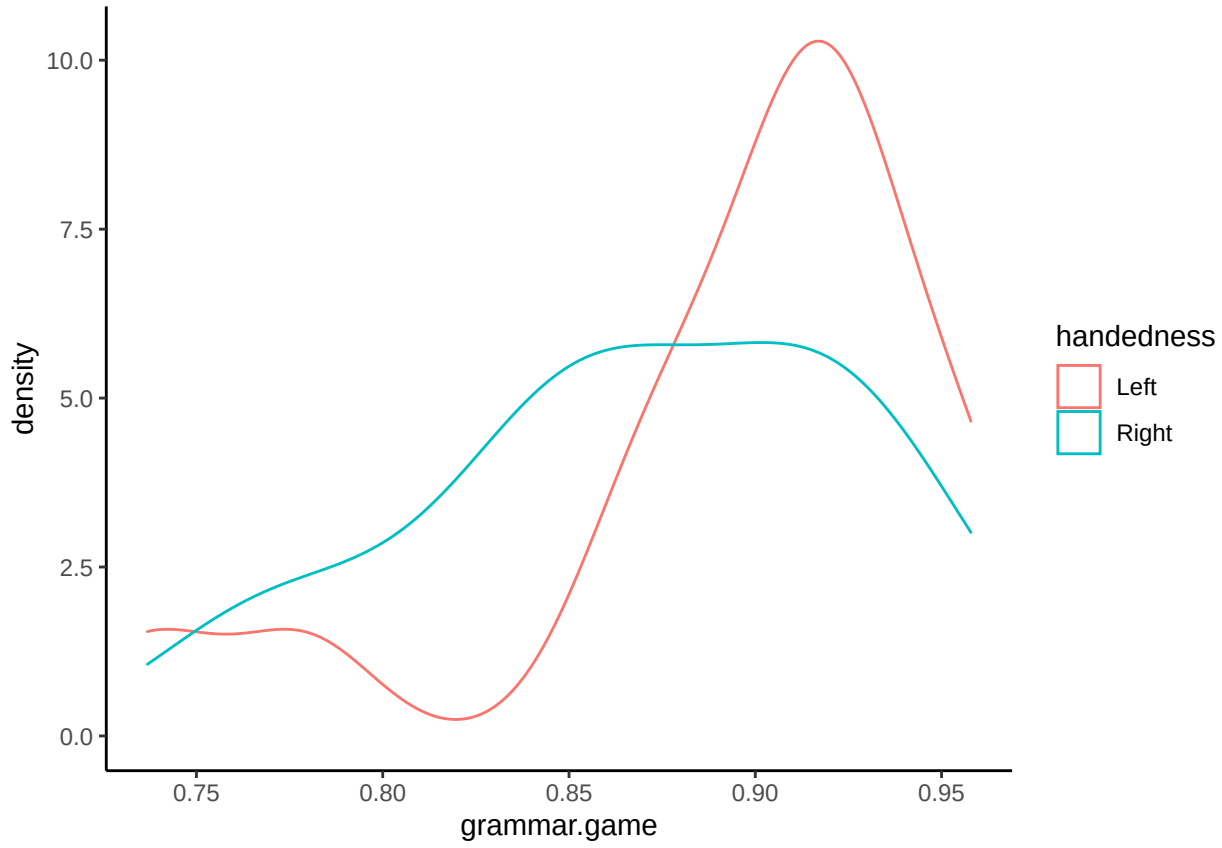
Now let's score a recognition test where participants see 88 words/non-words and indicate their lexical status. We'll then standardise and average together to get a z score fro spelling. This is typical as the two are usually correlated.

```
recognition <- read.csv("./data/piloting/data_exp_23909-v5_task-bod7.csv",stringsAsFactors = F)

# recode variables
recognition$subject <- recognition$Participant.Public.ID
  recognition$subject <- as.factor(recognition$subject)

# remove to corrrect rows
recognition <- recognition[recognition$display== "task",]

recog_agg <- aggregate(data= recognition, FUN= mean, Correct~ subject)
  names(recog_agg)[2] <- "spelling.recognition"

allsum <- merge(allsum, recog_agg, by.x= "subject", by.y = "subject")

# now create z spell
allsum$zrecog <- scale(allsum$spelling.recognition, center = TRUE, scale = TRUE)
allsum$zdict <- scale(allsum$spelling.dictation, center = TRUE, scale = TRUE)
allsum$zSpell <- (allsum$zrecog + allsum$zdict) / 2
```

Grammar Quiz

Here we score the word game that is able to distinguish between natives and non-natives. It's important to note that this was formatted from its original version so that it could be scored easily.

```
grammar <- read.csv("./data/piloting/data_exp_23909-v5_task-duyy.csv",stringsAsFactors = F)

# recode variables
grammar$subject <- grammar$Participant.Public.ID
  grammar$subject <- as.factor(grammar$subject)

# remove
grammar <- grammar[grammar$display == "picture" | grammar$display == "4AFC" |
                     grammar$display == "blanks" | grammar$display == "grammar",]
# means
grammar_agg <- aggregate(FUN= mean, data= grammar, Correct~ subject)
  names(grammar_agg)[2] <- "grammar.game"
allsum <- merge(allsum, grammar_agg, by.x= "subject", by.y = "subject")
# plot
ggplot(allsum, aes(x=grammar.game, color=handedness)) +
  geom_density() + theme_classic()
```




```
# summary
print(paste0("Mean grammar acuuracy: ", round(mean(grammar_agg$grammar.game)*100,2), "%"))
```

```
## [1] "Mean grammar acuuracy: 88.17%"
```

```
print(paste0("Range grammar acuuracy: ", round(min(grammar_agg$grammar.game)*100,2), "%", "-",
             round(max(grammar_agg$grammar.game)*100,2), "%"))
```

```
## [1] "Range grammar acuuracy: 73.68%-95.79%"
```

This is a fairly low score compared to Hartshorne et al (spelling?). I think they said that ~95% was when late learners began to become distingusihable, but our data suggests otherwise. This might because we changed how the game was administered to make it easier for scoring. In hindsight this may have been a mistake.

Tasks

First of all, let's combine the various spreadsheets.

```
#mydir = "./data/piloting/tasks"
#myfiles = list.files(path=mydir, pattern="*.csv", full.names=TRUE)
#dat_csv = plyr::ldply(myfiles, read.csv)

#write.csv(dat_csv, "./data/piloting/all_pilot_dat.csv")
```

We'll now prepare the data accordingly and read in the spreadsheet.

```
all_dat <- read.csv("data/piloting/all_pilot_dat.csv", na.strings = c("NA", ""), stringsAsFactors=F)
# cleaning - do operations that apply for all tasks
# write as factors
all_dat$subject <- as.factor(all_dat$Participant.Public.ID)
all_dat$RT <- as.numeric(as.character(all_dat$Reaction.Time))
```

```
## Warning: NAs introduced by coercion
```

```
all_dat$Trial <-as.numeric(all_dat$Trial.Number) #useful if we want to restrict analysis to subset of t
```

```
## Warning: NAs introduced by coercion
```

```
all_dat$Task.Name <- as.factor(all_dat$Task.Name)
all_dat$word<-substring(all_dat$word,66,(nchar(all_dat$word)-4)) #remove format from written word stimu
```

Functions

This is a function that can apply to all tasks; user specifies the original file to start from, the wanted columns, and the specific task to select.

```
# This function just returns a subfile with just the necessary rows/cols
  procdata <- function(origfile,wanted,task,disp){
  nufile <- origfile[origfile$Task.Name==task,] #restrict consideration to this task
  nufile <- origfile[origfile$display==disp,] #restrict consideration to the correct display

  c<-which(names(nufile) %in% wanted) #find colnumbers of wanted
  #NB beware that the columns may not be in the same order as in the wanted list
```

```
  nufile <- nufile[,c]#select only wanted columns
  nufile <- na.omit(nufile) # remove NAs
  return(nufile)
}
```

The Hoaglin-Iglewicz function is used to remove outliers. We create an 'outlier' column that codes anticipations as 2, and long RT outliers as 1. All else is zero. The original RT is saved as allRT, and the RT column has NA for outliers, so they will be automatically excluded henceforth.

Outlier removal put in separate function, which can be applied to different task RTs.
Origdat is the dataframe with data for outlier removal, mycolname is name of variable
For outlier removal (usually RT), lowcut is a cutoff value below which data excluded (typically v fast responses that are implausible, e.g. 201 ms for RT) and zcut is the constant in Hoaglin-Iglewicz method (usually 2.2). Here the focus is on just one tail of the distribution (slow RTs) so we use 1.65 as zcut.

```
detect.outliers <- function(origdat,lowcut,zcut){

# NB this function returns origdat with outlier rows marked, rather than removed
# This makes it easier to record the N outliers per subject.

# NB FUNCTION ASSUMES THAT THERE IS ARE COLUMNS CALLED subject, Correct and RT IN origfile.
# Outliers are computed by subject - NB contrary to previous version, these are defined
# across all trials, rather than separately for L and R.
origdat$allRT <- origdat$RT #we save a copy of RT, as we will be altering RT so that outliers become NA
origdat$outlier<-0
origdat$outlier[origdat$RT<lowcut] <- 2 #outlier col is coded 2 for anticipations
origdat$RT[origdat$RT<lowcut]<-NA #we exclude anticipations when computing quantiles
#Now extract correct RTs for each subject to compute limits for outliers

  for (i in 1:nsub) { # loop through subjects
  subname <- origdat$subject[i] # find subject
  myrows <- which(origdat$subject==subname) # select rows for this subject
  #NB myrows is NOT a consecutive series, because done in blocks
  firstrow <-min(myrows)
  tmp <- data.frame(origdat[myrows,])

  RTcorr <-tmp$RT[tmp$Correct==1] #correct RTs for this subject as a vector

 # IDentify 25th and 75th quartiles, and the difference between them
          lower_quartile <- quantile(RTcorr, probs=0.25, na.rm="TRUE")
          upper_quartile <- quantile(RTcorr, probs=0.75, na.rm="TRUE")
          quartile_diff <- upper_quartile - lower_quartile
        # Outliers are defined as being below or above 2.2 times the quartile difference
          lower_limit <- lower_quartile - zcut*quartile_diff
          upper_limit <- upper_quartile + zcut*quartile_diff
        # create outlier variable
w1 <- which(origdat$RT[myrows]>upper_limit) #rows with outlier, nb RELATIVE to myrows range
origdat$outlier[myrows[w1]]<-1
}
 #For RT we just remove slow outliers, so ignore lower_limit
origdat$RT[origdat$outlier>0]<-NA #RT variable now has outliers excluded as NA
return(origdat)
}
```

This is a generic chunk of code that can be used for Chimeric faces and Rhyme Detection.

```
make.df <- function(origdat,varlist,latlist,nloop){
nsub<-length(unique(origdat$subject))
# create compact data frame ; one row for each subject, subjects stacked, L above R
temp_dat <- data.frame(matrix(ncol = 4, nrow = nsub*nloop))
colnames(temp_dat) <- varlist
myrow <- 0 # start row counter
for (i in 1:nsub) { # loop through subjects
  subname <- i # find subject
  myrows <- which(origdat$subject==subname) # select rows for this subject
  latcol <- which(colnames(origdat) == varlist[2])
  RTcol <- which(colnames(origdat) == varlist[4])
  tmp <- data.frame(origdat[myrows,]) #all trials from this subject

  for (j in 1:nloop) { #make row for means for each side for this subject
    myrow <- myrow+1
    w<- which(tmp[,latcol]== latlist[j]) # match on laterality
    tmp1 <- tmp[w,]
    tmp2 <- tmp1[tmp1$Correct == 1,]

    temp_dat$subject[myrow] <- subname # add row for subject
    temp_dat$side[myrow] <- latlist[j] # add row for sIDe
    temp_dat$accurate[myrow] <- sum(tmp1$Correct,na.rm=TRUE) # add row for N accurate response
    temp_dat$p.corr[myrow] <- 100*mean(tmp1$Correct,na.rm=TRUE) # add row for %accurate responses - NB

    temp_dat$RT[myrow] <- mean(as.numeric(tmp2$RT),na.rm=TRUE) # add row for mean RT
    #NB for RDT task, also looked at medians, but it did not make much difference, presumably because v
    temp_dat$N[myrow] <- length(tmp1$RT) # add count of rows for this subject
  }
  #If there are no selections/trials for one side, then exclude L and R
  # if(temp_dat$N[myrow==0]||temp_dat$N[(myrow-1)==0]){
  #   temp_dat$RT[(myrow-1):myrow]<-NA
  # }
}
return(temp_dat)
}
```

This plots data and conducts t-tests for each task.

```
dopirate <- function(origdat,task,measure){
# measure for each sIDe
# plot the number of correct responses in each sIDe
  myhead<-paste0(task,": ",measure)
pirateplot(formula = X1 ~ side,
           data = origdat,
           main = myhead,
           theme = 0,
           pal = "southpark", # southpark color palette
           bean.f.o = .0, # Bean fill
           point.o = .3, # Points
           inf.f.o = .7, # Inference fill
           inf.b.o = .8, # Inference border
           avg.line.o = 1, # Average line
```

```
            bar.f.o = .5, # Bar
            point.pch = 21,
            point.cex = .7,
            inf.method= "ci")
# plot L & R to get slope
q <- ggplot(data=origdat, aes(x=side, y=X1, group=as.factor(subject), color=subject)) +
    geom_line() +
    geom_point() +
    ggtitle(paste0(task,": ",measure," by side"))
q + theme_bw() + theme(legend.position = "none")
#  t test
print(paste0(task,measure))
myt1<-t.test(origdat$X1~ origdat$side, paired = TRUE, alternative = "two.sided")
print(myt1)
}
```

Another generic function that can be used for different tasks to make a laterality index and write it to the allsum file

```
# This function modifies a copy of the allsum file and returns it

# User selects whether accuracy or RT by specifying col X1 in origdat when calling the function

# Polarity is -1 or 1 and can be set to keep all LIs positive if in a given direction
# Cutoff will create NA if level of performance is outside predetermined limits

# 19 Jul 2020:settled on setting polarity so that L hemisphere advantage is positive

makeLI <- function(allsum,origdat,task,mycolname,polarity,mycutoff){
# we do this for accuracy first
# reformat data frame so left and right are side by side
keeps <- c("subject", "side", "X1") #X1 holds either accuracy or RT
Lsubjectf <- origdat[keeps]
Lsubjectf <- spread(Lsubjectf, side, X1)
# calculate each participants' lat index

Lsubjectf$thisLI <- 100*(Lsubjectf$Right - Lsubjectf$Left)/(Lsubjectf$Right + Lsubjectf$Left)


Lsubjectf$LI <- Lsubjectf$thisLI*polarity #polarity can be set to reverse sign - eg for RT a large scor

# check if accuracy or RT level is outside acceptable cutoff.
# Do this based on average for left and right
# LI will be set to NA for those outside limits
Lsubjectf$avg <- (Lsubjectf$Left+Lsubjectf$Right)/2
w<-which(Lsubjectf$avg<mycutoff)
if(length(w)>0){
Lsubjectf$LI[w] <- NA}

# now create a data frame for lat index
Lsubjectf <- select(Lsubjectf, "subject", "LI")
# now merge with allsum
# First check if this column already exists and delete it if so
w<-which(colnames(allsum)==mycolname)
```

```
if(length(w)>0){
allsum<-allsum[,-w]}

#now add this LI
allsum <- merge(allsum, Lsubjectf, by= "subject",all.x=T)
#Need all.x = T to retain all original subjects from allsum, even if no match
lastcol<-length(colnames(allsum))
colnames(allsum)[lastcol]<-mycolname

return(allsum)
}
```

Make Z score LI. NB this is NOT a conventional z-score based on whole sample, but rather is an individual-based z-score that indicates whether the person's L-R difference is significantly different from chance. So you either compare their L-sided RTs and their R-sided RTs, or the proportions correct on L vs proportion correct on R, to see if there is bias away from zero in that individual. Note that this in effect adjusts for any overall differences in accuracy and RT, as the person is compared with themselves.

```
# This function modifies a copy of the allsum file and returns it

# User selects the measure by specifying col X1 in origdat when calling the function

# Polarity is -1 or 1 and can be set to keep all LIs positive if in a given direction
# Cutoff will create NA if level of performance is outside predetermined limits

makeZ <- function(allsum,origdat,task,mycolname,polarity,mycutoff){
# we do this for accuracy first
# reformat data frame
keeps <- c("subject", "side", "X1") #X1 holds either accuracy or RT
Zdf <- origdat[keeps]
Zdf <- spread(Zdf, side, X1)
# calculate each participants' z score for laterality
Zdf <-
  Zdf %>%
  group_by(subject) %>%
  mutate(n= Left + Right,
         pR= Right/n,
         pL= Left/n,
         Z= (pR-.5)/sqrt(pR*pL/n))
# remove infinite results
is.na(Zdf) <- do.call(cbind,lapply(Zdf, is.infinite))
Zdf <- na.omit(Zdf)


Zdf$Z <- Zdf$Z*polarity #polarity can be set to reverse sign - eg for RT a large score is bad, whereas

# now create a data frame for lat index
Zdf <- select(Zdf, "subject", "Z")
# now merge with allsum
# First check if this column already exists and delete it if so
w<-which(colnames(allsum)==mycolname)
if(length(w)>0){
allsum<-allsum[,-w]} #delete cols that already exist with that name
```

```
allsum <- merge(allsum, Zdf, by= "subject",all.x=T)
lastcol<-length(colnames(allsum))
colnames(allsum)[lastcol]<-mycolname

return(allsum)
}
```

Create a simple numerical ID to make life easier when screening data if we need to identify subjects whose data is odd in some way. Ensure this is consistent across all_dat1 and all_dat2 and allsum. We first rename the current subject variable so we can reuse this name with the numbered ID for all future computations. Again, this is rather slow because done in a loop, which is probably not very efficient!

```
#Use allsum as the basis for the check
#First rename all subject to origID
allsum <- allsum %>% rename(origID = subject)
all_dat <- all_dat %>% rename(origID = subject)
all_dat$subject <- NA
for (n in 1:nrow(allsum)){
  allsum$subject[n] <- n
  w<-which(all_dat$origID == allsum$origID[n])
  all_dat$subject[w] <-n
}
```

Rhyme Detection Task

This portion of the notebook prcoesses the "Rhyme Detection Task' implemented online via Gorilla.sc, which aims to assess the lateralisation of language processing. This is a new task devised for this project. In the task, participants are shown a word in the central visual field and two images in the left and right visual field. The task is to judge which of the two images has a name that rhymes with the central word. The logic of the task is that to perform one must generate the phonology of a pictured word's name, which is a classic left-hemisphere task - it is assumed this will be easier for pictures the are projected to the left hemisphere (ie RVF).

We start by reading the data and cutting out unwanted information. Here we are including item-specific information as it may be used for later analysis.

```
origfile <- all_dat
task <- "Rhyme Detection Task"
disp <- "Task"

# nb when creating new file, we default to name 'RDT_dat', but this is reassigned to 'RDT_dat' at end o

# new data
wanted <- c("Trial","subject", "Participant.Public.ID", "Correct", "RT", "ANSWER","Response","word","Le
RDT_dat <- procdata(origfile,wanted,task,disp) #Here we reuse the procdata function that we developed f

# do some wrangling on the ANSWER variable to make it compatible with functions
RDT_dat<-filter(RDT_dat,ANSWER %in% c('left','right'))
  c <- which(colnames(RDT_dat)=="ANSWER")
  colnames(RDT_dat)[c]<-'side'
RDT_dat$side<-as.factor(RDT_dat$side)
  levels(RDT_dat$side)<-c("left","right")
```

```
#get rid of .png extensions
RDT_dat$Left_image<-tools::file_path_sans_ext(RDT_dat$Left_image)
RDT_dat$Right_image<-tools::file_path_sans_ext(RDT_dat$Right_image)

RDT_dat$picpair <- paste0(RDT_dat$Left_image,'_',RDT_dat$Right_image)
w<-which(RDT_dat$side=='right')
RDT_dat$picpair[w]<- paste0(RDT_dat$Right_image[w],'_',RDT_dat$Left_image[w])

RDT_dat$item <- paste0(RDT_dat$word,'_',RDT_dat$picpair)

#Also reveals occasional trials where Response shows 'Could not set screen ...' These were coded as err
s <- "Could not set image to requested size.  Either the participants screen or the containing zone is
w<-which (RDT_dat$Response==s)
RDT_dat<-RDT_dat[-w,]

# count subjects
nsub <-length(unique(RDT_dat$subject))
print(paste0("subjects: ", nsub))
```

```
## [1] "subjects: 30"
```

Now use generic function to remove outliers.

```
RDT_dat$RT <- log(RDT_dat$RT)
RDT_dat<-detect.outliers(RDT_dat,log(200),1.65) #numbers specify min RT and zscore for Hoaglin-Iglewicz

outliertable<-table(RDT_dat$subject,RDT_dat$outlier)
```

Scrutiny of outliertable indicates a few problematic cases with fewer than 260 trials in total. Seems related to glitches in Gorilla. Based on this, we will remove subject 8 as they have less than

Now the data is ready, create an empty data frame and populate this. It includes columns for the participant identifier, rhyme, acuracy, RT (RT only for correct answers).

```
#We create RDT files for day1 and day 2
varlist <-   c("subject", "side", "accurate", "RT") #need to be in this order, ie sub, side, acc and RR
latlist <- c("left","right") #names of factor levels
nloop<-length(latlist)
RDT_df <- make.df(RDT_dat,varlist,latlist,nloop)


# remove NaN
is.nan.data.frame <- function(x)
  do.call(cbind, lapply(x, is.nan))
RDT_df[is.nan(RDT_df)] <- NA

#Also record % correct for each category in allsum
myfile<-RDT_df
pcorL<-myfile[myfile$side=='left',]
pcorR<- myfile[myfile$side=='right',]
pcorL<-select(pcorL,subject,p.corr)
pcorR<-select(pcorR,subject,p.corr)
```

```
colnames(pcorL)[2]<-paste0('RDT','.pcorrL')
colnames(pcorR)[2]<-paste0('RDT','.pcorrR')
allsum <- merge(allsum,pcorL,by="subject")
allsum <- merge(allsum,pcorR,by="subject")
```

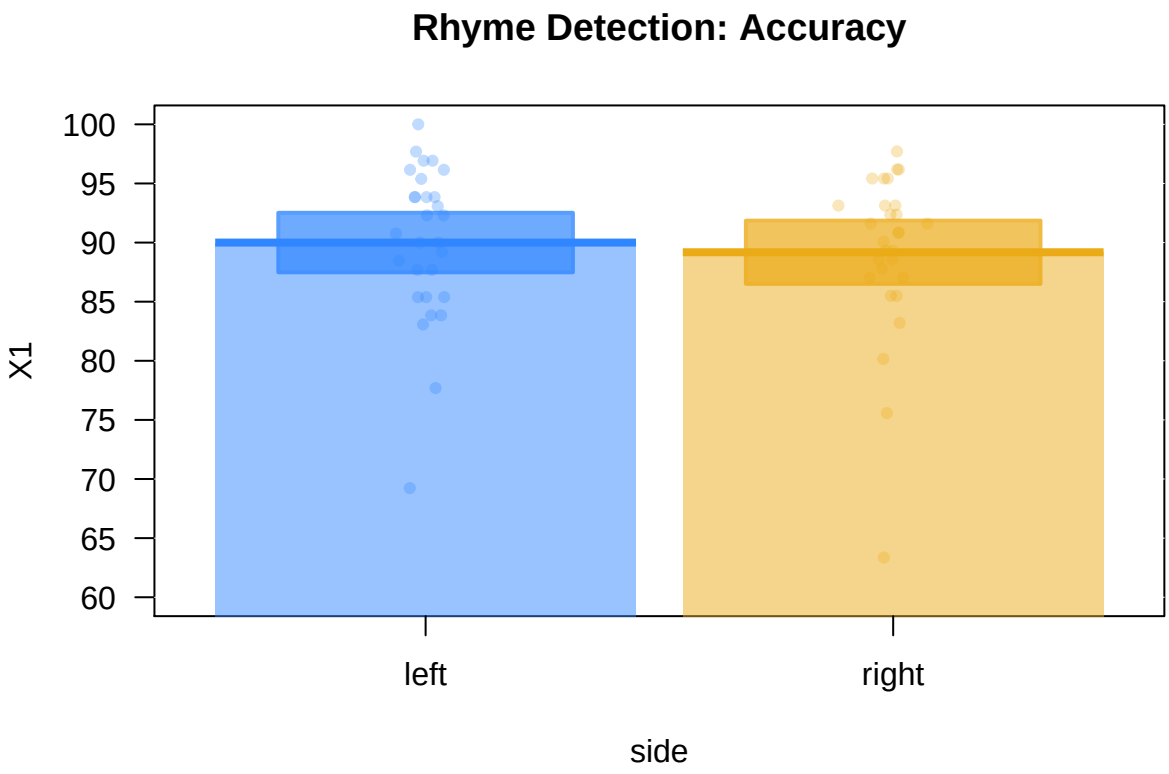
Next exclude those who score less than 50% correct responses on either side

```
wn <-unique(which(RDT_df$accurate ==0)) #this catches those who were always wrong on 'no rhyme' trials,

w1<-which(RDT_df$p.corr < 50)

exRDT<-unique(c(RDT_df$subject[w1],RDT_df$subject[wn], 8))

allsum$exRDT <- 0
allsum$exRDT[allsum$subject %in% exRDT]<-1 #decided best to have separate column for exclusion for each
w<-which(allsum$exRDT==0)
incRDT <- allsum$subject[w]
ninc <- length(incRDT)
print(paste0('RDT: number remaining after exclusions = ',ninc))
```

```
## [1] "RDT: number remaining after exclusions = 29"
```

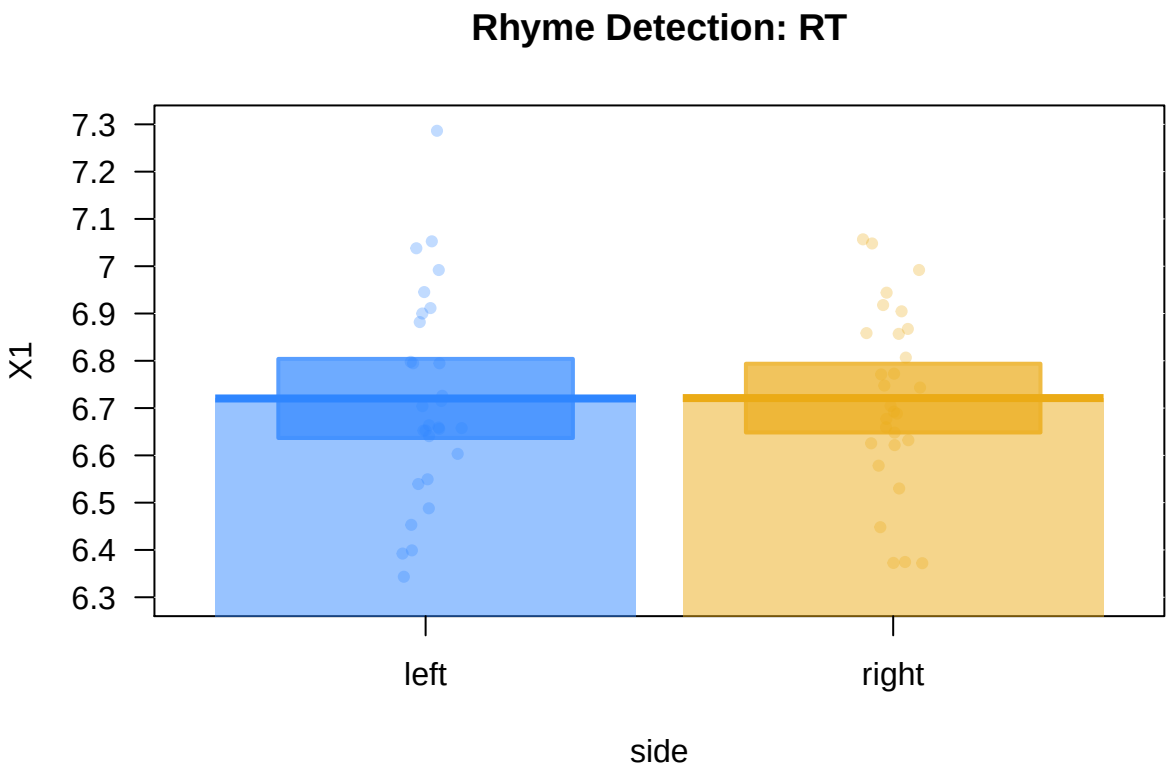
Now that we are ready to look at means in each visual field.

```
origdat <-RDT_df[RDT_df$subject %in% incRDT,]

origdat <-origdat[origdat$side %in% c('left','right'),]

# accuracy
measure <- 'Accuracy'
origdat$X1 <- origdat$p.corr
task <- 'Rhyme Detection'
dopirate(origdat,task,measure)
```




```
## [1] "Rhyme DetectionAccuracy"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = 1.0542, df = 28, p-value = 0.3008
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.7721127  2.4093857
## sample estimates:
## mean of the differences
##               0.8186365
```

```
# RT
origdat$X1 <- origdat$RT
measure <- 'RT'
dopirate(origdat,task,measure)
```



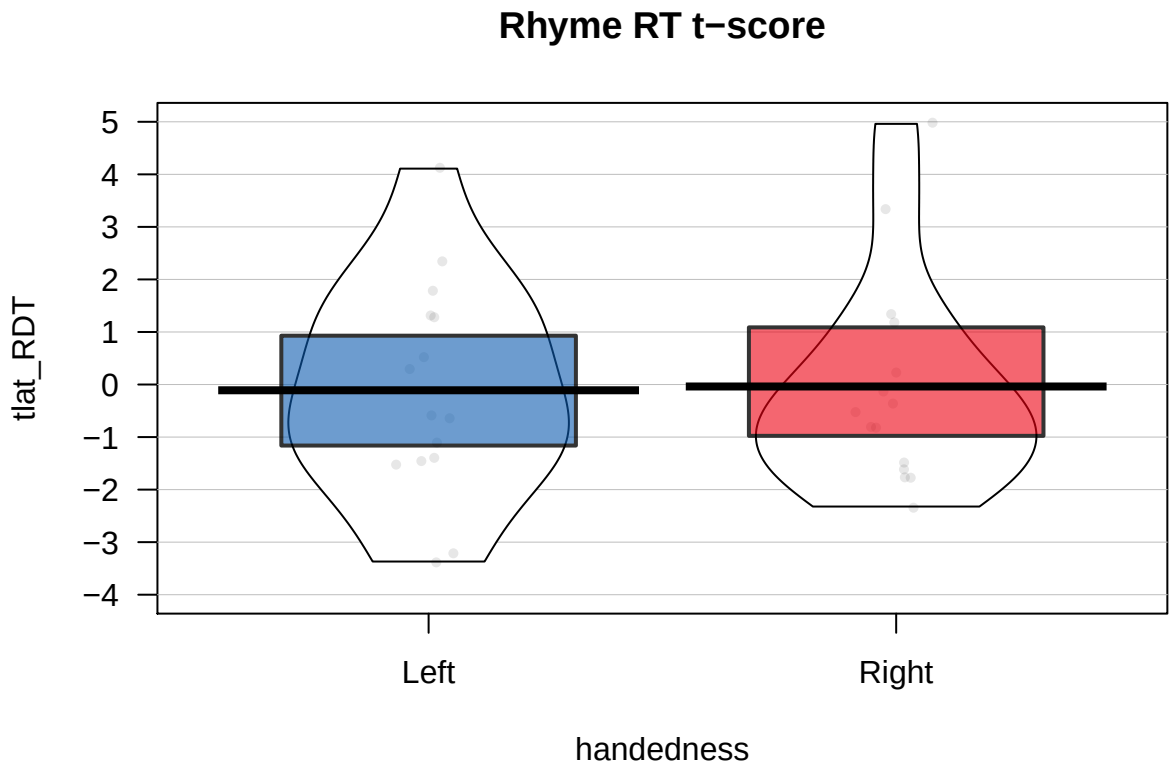

```
## [1] "Rhyme DetectionRT"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = -0.044221, df = 28, p-value = 0.965
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.03817054  0.03655732
## sample estimates:
## mean of the differences
##           -0.0008066143
```

Accuracy looks pretty good, but the difference in response time looks small. Now that we have visualised the data, let's create an LI for the data.

As there is less than 30, we'll use a t-score rather than a z-score here.

```
myfile<-RDT_dat

nsubs <- nrow(allsum)
allsum$tempt<-NA #initialise a new column

for (i in 1:nsubs) {
  mysub<-filter(myfile,side %in% c('left','right'),subject==allsum$subject[i])
    if(nrow(mysub)>10)
```

```
      {
      myt<-t.test(mysub$RT~mysub$side) #this is based on logRT
      allsum$tempt[i]<-myt$statistic*-1
    }
  }
wc<-which(colnames(allsum)=='tempt')
colnames(allsum)[wc]<-paste0('tlat_RDT')

pirateplot(formula = tlat_RDT ~ handedness,
                  data = allsum,
                  theme = 2,
                  main = "Rhyme RT t-score")
```




```
print(paste0("Testing whether LI is different from zero, p= ", t.test(allsum$tlat_RDT)[3]))
```

```
## [1] "Testing whether LI is different from zero, p= 0.842177079423411"
```

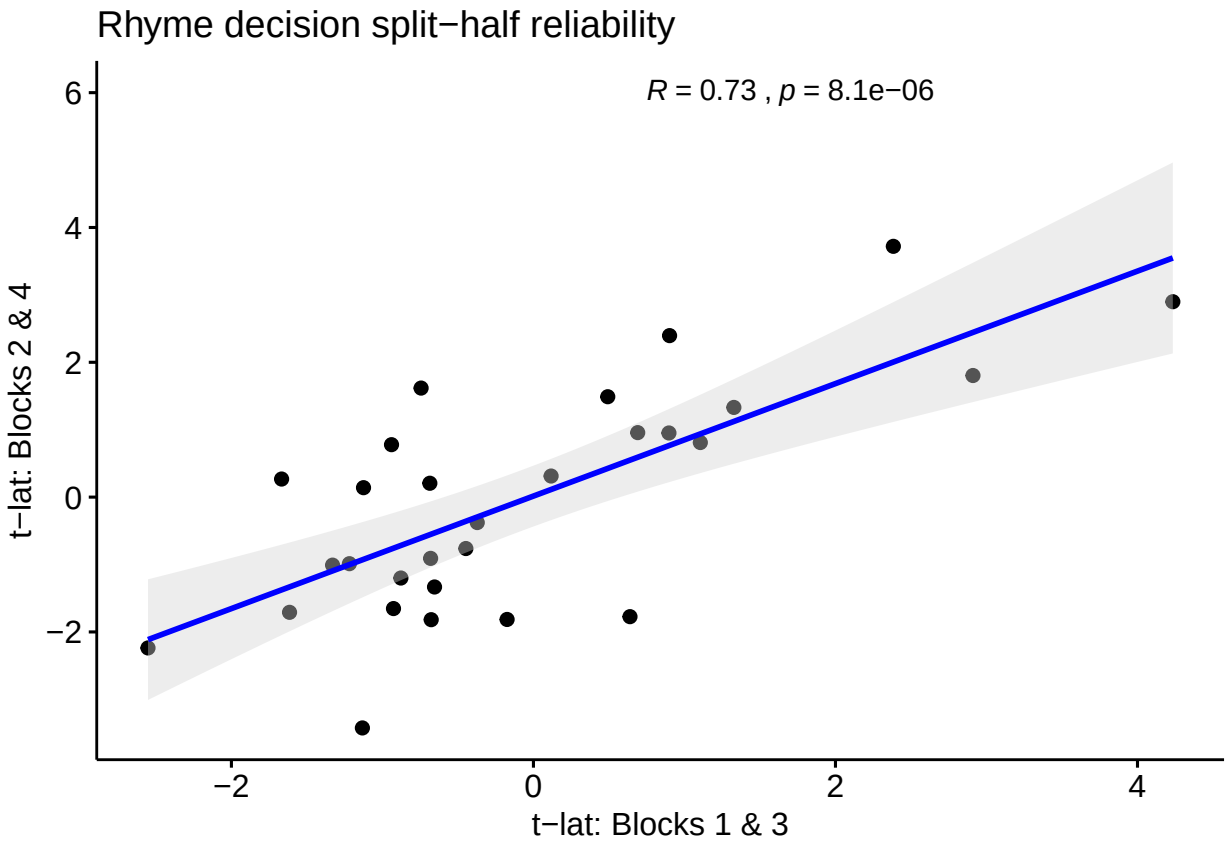
The LIs look pretty much clustered around zero and definitely do not differ between handedness... which is a worry.

Split-half reliability

Let's check the split-half for the rhyme task.

```r
RDT_split = data.frame(matrix(nrow= nsub, ncol= 1))
  colnames(RDT_split) <- c("subject")
# code to two level
RDT_data <- RDT_dat %>% mutate(Spreadsheet.Name= if_else(Spreadsheet.Name== "block 1" |
                                                           Spreadsheet.Name == "block 3", "block1", "blo
# turn block to factor
RDT_data$Spreadsheet.Name <- as.factor(RDT_data$Spreadsheet.Name)
# count blocks
block <- length(unique(RDT_data$Spreadsheet.Name))
# initiate a column
RDT_split$tempt<-NA
# start row
myrow <- 0
# look at blocks
for (b in 1:block) {

  blockname <- levels(RDT_data$Spreadsheet.Name)[b] # find subject subject
  myrows <- which(RDT_data$Spreadsheet.Name==blockname) # select rows for this subject
  tmp <- data.frame(RDT_data[myrows,])

  for (i in 1:nsubs) {

    mysub<-filter(tmp,side %in% c('left','right'),subject== allsum$subject[i])
      if(nrow(mysub)>10)
      {
        myt<-t.test(mysub$RT~mysub$side) #this is based on logRT
        RDT_split$tempt[i]<-myt$statistic*-1
        RDT_split$subject[i]<-i
    }
  }
  wc<-which(colnames(RDT_split)=='tempt')
  colnames(RDT_split)[wc]<-paste0('tlat_RDT',b)
}
# plot
ggscatter(subset(RDT_split, subject != 8), x = "tlat_RDT1", y = "tlat_RDT2",
          add = "reg.line", conf.int = TRUE, add.params = list(color = "blue",fill = "lightgray"),
          xlab= "t-lat: Blocks 1 & 3", ylab= "t-lat: Blocks 2 & 4", title = "Rhyme decision split-half
          stat_cor(method = "pearson", label.x = .75, label.y = 6)
```

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



The split half relibaility seems good for the rhyme task when using the t-lat. It's odd how it isn't lateralised though...

Optimal viewing position task

This portion of the notebook prcoesses the "Optimal Viewing Task' implemented online via Gorilla.sc, which aims to assess the lateralisation of language processing. This is requires participants to make a lexical decision when viewing the first or last letter of a 6-letter word. When viewing at the first letter, the word is presented to the rigth visual field (i.e. left hemipshere). A word presented at the sixth letter will be presented to the left visual field (i.e. right hemispehre). Thus, we would expect a processing advantage when fixating the first letter.

```
origfile <- all_dat
task <- "OVP"
disp <- "Task"

# new data
wanted <- c("Trial","subject", "Participant.Public.ID", "Correct", "RT", "ANSWER","Response","string",
OVP_dat <- procdata(origfile,wanted,task,disp)

# do some wrangling on the ANSWER variable to make it compatible with functions
OVP_dat$side<-as.factor(OVP_dat$VF)
  levels(OVP_dat$side)<-c("left","right")

# remove non-words
```

```
OVP_dat <- OVP_dat[OVP_dat$ANSWER == "word",]

nsub <-length(unique(OVP_dat$subject))
```

Now use generic function to remove outliers.

```
OVP_dat$RT <- log(OVP_dat$RT)
OVP_dat<-detect.outliers(OVP_dat,log(200),1.65) #numbers specify min RT and zscore for Hoaglin-Iglewicz

outliertable<-table(OVP_dat$subject, OVP_dat$outlier)
```

Scrutiny of outliertable indicates a few problematic cases with fewer than 160 word trials in total. Again, participants 8 seems to be an issue.

Now the data is ready, create an empty data frame and populate this. It includes columns for the participant identifier, rhyme, acuracy, RT (RT only for correct answers).

```
#We create RDT files for day1 and day 2
varlist <-   c("subject", "side", "accurate", "RT") #need to be in this order, ie sub, side, acc and RR
latlist <- c("left","right") #names of factor levels
nloop<-length(latlist)
OVP_df <- make.df(OVP_dat,varlist,latlist,nloop)


# remove NaN
is.nan.data.frame <- function(x)
  do.call(cbind, lapply(x, is.nan))
OVP_df[is.nan(OVP_df)] <- NA
```

Next exclude those who have less than 50% correct responses on either side.

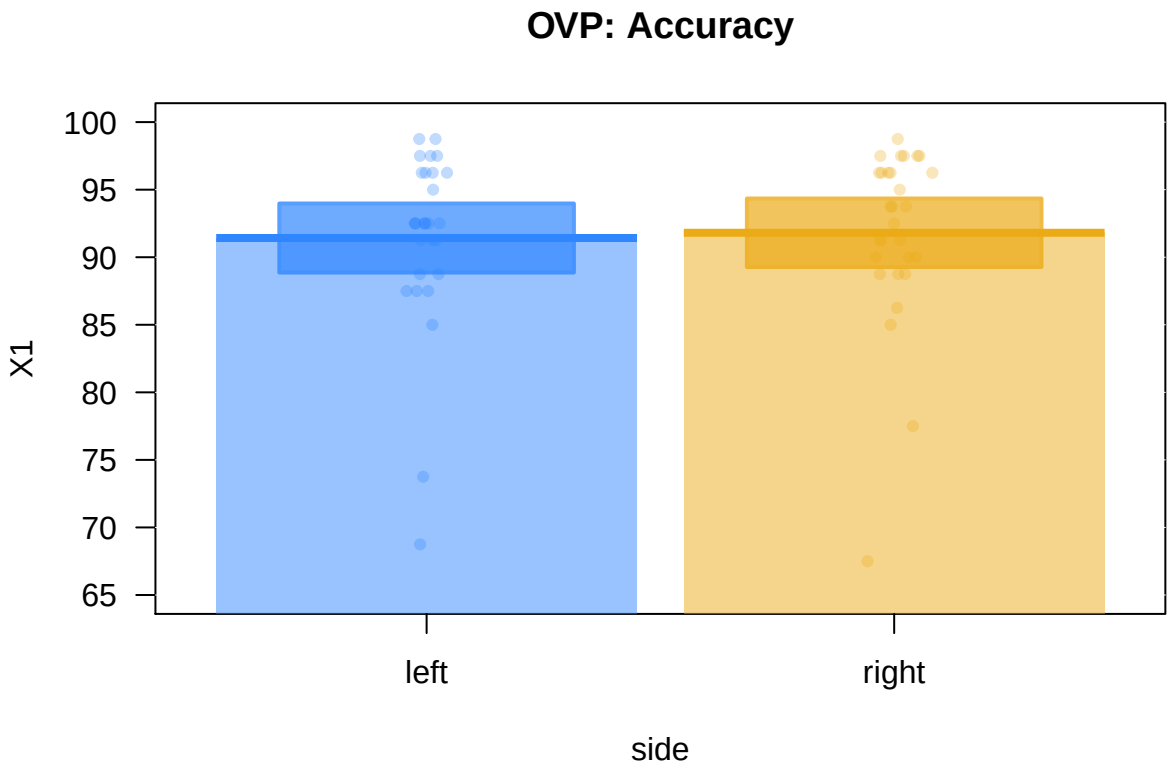
```
wn <-unique(which(OVP_df$accurate ==0)) #this catches those who were always wrong on 'no rhyme' trials,

w1<-which(OVP_df$p.corr < 50)

exOVP<-unique(c(OVP_df$subject[w1],OVP_df$subject[wn], 8))

allsum$exOVP <- 0
allsum$exOVP[allsum$subject %in% exOVP]<-1 #decided best to have separate column for exclusion for each
w<-which(allsum$exOVP==0)
incOVP <- allsum$subject[w]
ninc <- length(incOVP)
print(paste0('OVP: number remaining after exclusions = ',ninc))
```

```
## [1] "OVP: number remaining after exclusions = 29"
```

Ready to visualise.

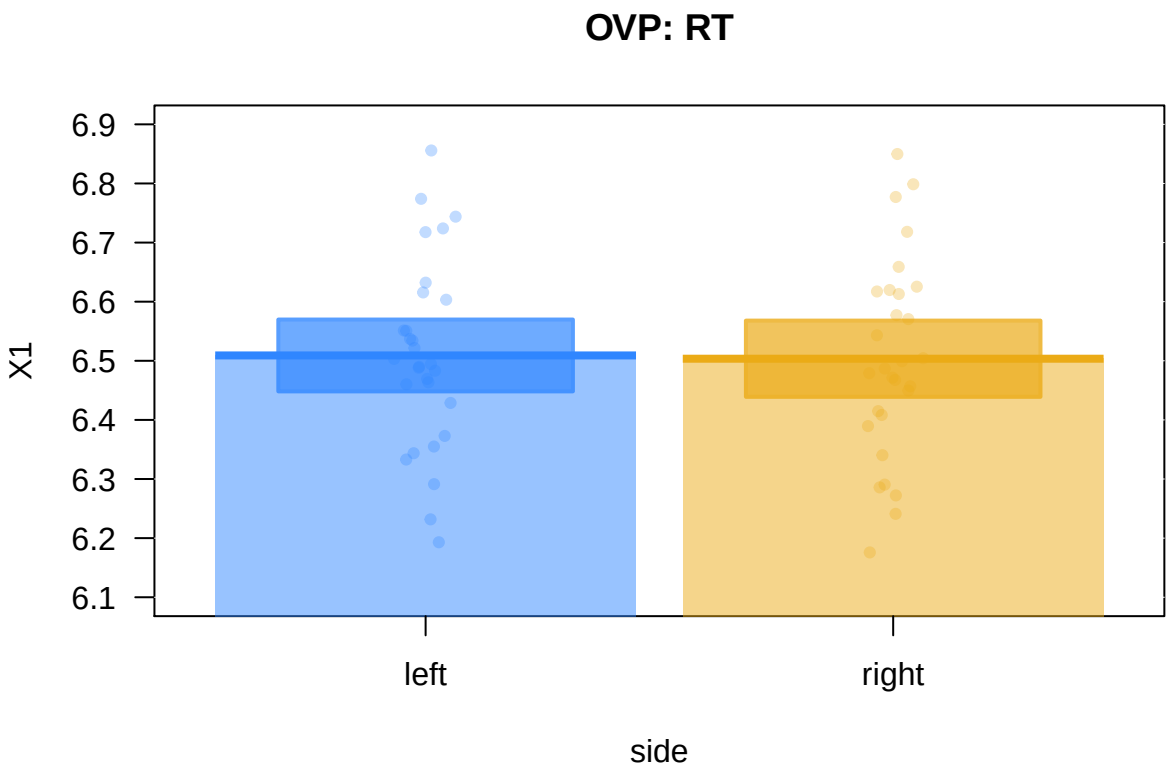
```
origdat <-OVP_df[OVP_df$subject %in% incOVP,]

origdat <-origdat[origdat$side %in% c('left','right'),]
```

```
# accuracy
measure <- 'Accuracy'
origdat$X1 <- origdat$p.corr
task <- 'OVP'
dopirate(origdat,task,measure)
```




```
## [1] "OVPAccuracy"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = -0.35683, df = 28, p-value = 0.7239
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.614856  1.838994
## sample estimates:
## mean of the differences
##               -0.387931
```

```
# RT
origdat$X1 <- origdat$RT
measure <- 'RT'
dopirate(origdat,task,measure)
```

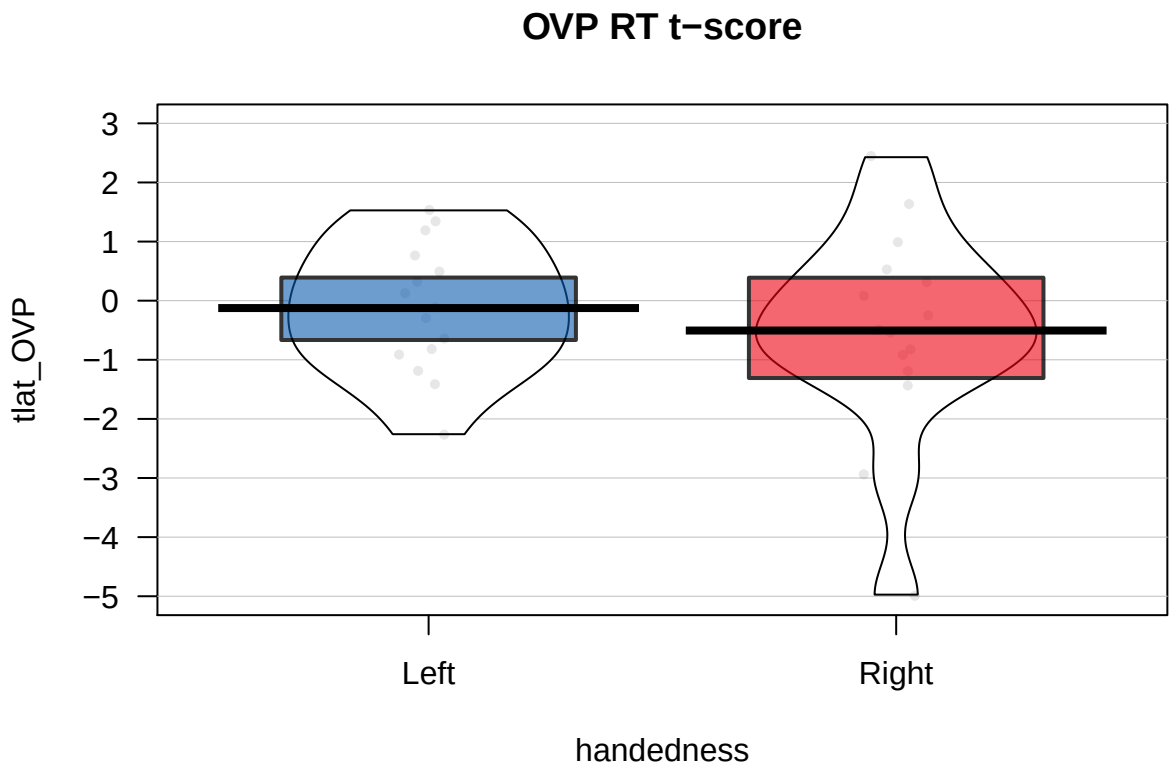



```
## [1] "OVPRT"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = 0.48664, df = 28, p-value = 0.6303
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.01769693  0.02872546
## sample estimates:
## mean of the differences
##             0.005514264
```

Accuracy is close to ceiling and does not show side difference, nor does RT (again).

```
myfile<-OVP_dat

nsubs <- nrow(allsum)
allsum$tempt<-NA #initialise a new column

for (i in 1:nsubs) {
  mysub<-filter(myfile,side %in% c('left','right'),subject==allsum$subject[i])
    if(nrow(mysub)>10)
      {
      myt<-t.test(mysub$RT~mysub$side) #this is based on logRT
```

```
      allsum$tempt[i]<-myt$statistic*-1
    }
  }
wc<-which(colnames(allsum)=='tempt')
colnames(allsum)[wc]<-paste0('tlat_OVP')

pirateplot(formula = tlat_OVP ~ handedness,
                  data = allsum,
                  theme = 2,
                  main = "OVP RT t-score")
```




```
print(paste0("Testing whether LI is different from zero, p= ", t.test(allsum$tlat_OVP)[3]))
```

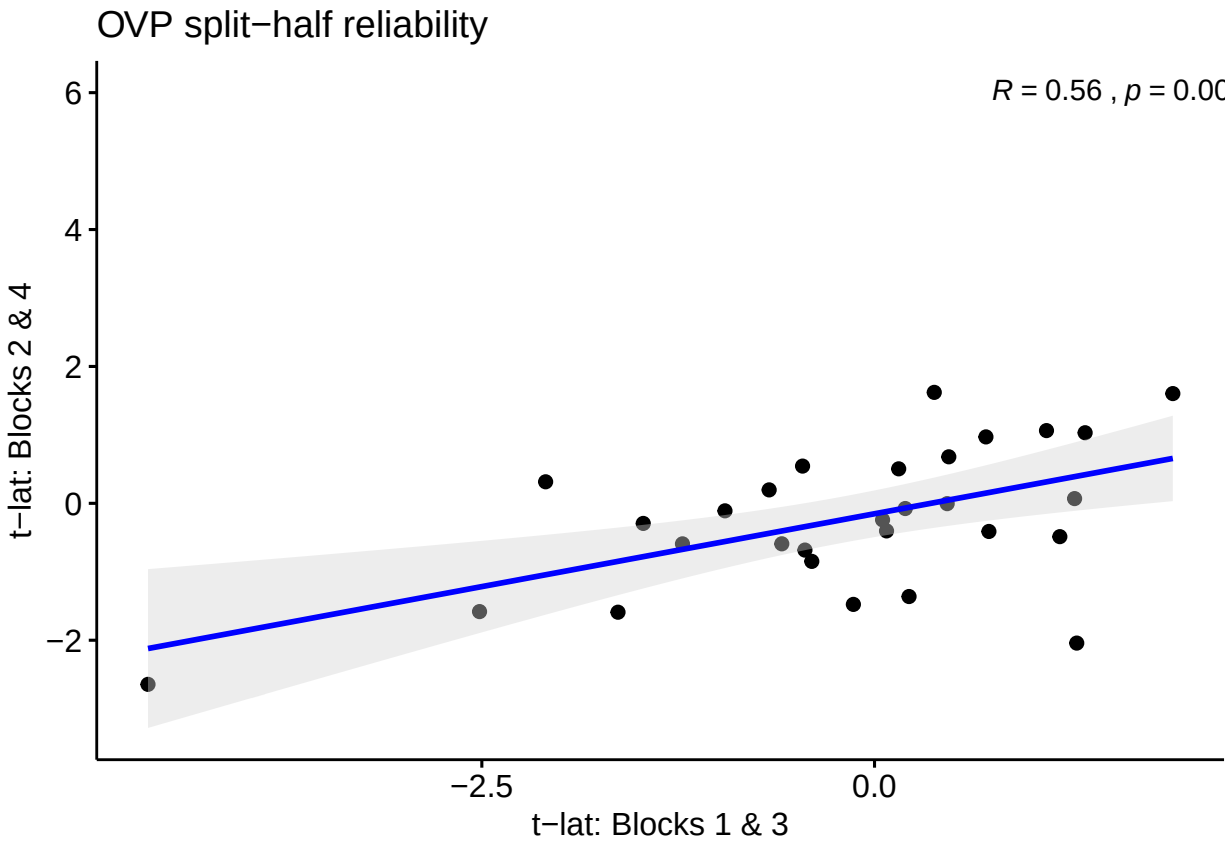
```
## [1] "Testing whether LI is different from zero, p= 0.25217780547858"
```

The OVP doesn't seem hugely lateralised in left handers but slightly better in right handers.

```
OVP_split = data.frame(matrix(nrow= nsub, ncol= 1))
  colnames(OVP_split) <- c("subject")
# code to two level
OVP_data <- OVP_dat %>% mutate(Spreadsheet.Name= if_else(Spreadsheet.Name== "list 1" |
                                                           Spreadsheet.Name == "list 2", "block1", "bloc
# turn block to factor
OVP_data$Spreadsheet.Name <- as.factor(OVP_data$Spreadsheet.Name)
```

```r
# count blocks
block <- length(unique(OVP_data$Spreadsheet.Name))
# initiate a column
OVP_split$tempt<-NA
# start row
myrow <- 0
# look at blocks
for (b in 1:block) {

  blockname <- levels(OVP_data$Spreadsheet.Name)[b] # find subject subject
  myrows <- which(OVP_data$Spreadsheet.Name==blockname) # select rows for this subject
  tmp <- data.frame(OVP_data[myrows,])

  for (i in 1:nsubs) {

    mysub<-filter(tmp,side %in% c('left','right'),subject== allsum$subject[i])
      if(nrow(mysub)>10)
      {
        myt<-t.test(mysub$RT~mysub$side) #this is based on logRT
        OVP_split$tempt[i]<-myt$statistic*-1
        OVP_split$subject[i]<-i
    }
  }
  wc<-which(colnames(OVP_split)=='tempt')
  colnames(OVP_split)[wc]<-paste0('tlat_OVP',b)
}
# plot
ggscatter(subset(OVP_split, subject != 8), x = "tlat_OVP1", y = "tlat_OVP2",
          add = "reg.line", conf.int = TRUE, add.params = list(color = "blue",fill = "lightgray"),
          xlab= "t-lat: Blocks 1 & 3", ylab= "t-lat: Blocks 2 & 4", title = "OVP split-half reliability
          stat_cor(method = "pearson", label.x = .75, label.y = 6)
```

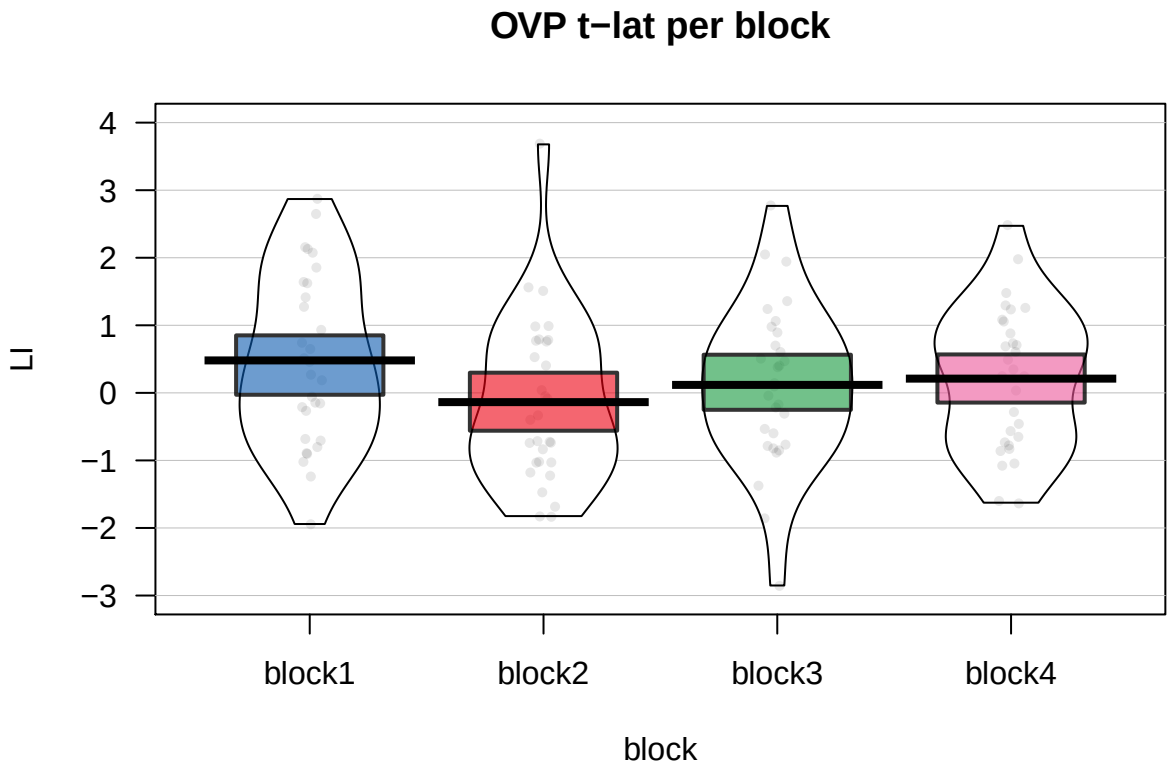
```
## `geom_smooth()` using formula 'y ~ x'
```



The split half relibaility doesn't look as good as we'd hoped. But, participants did report that they found the task mind numbing. I'm wondering if performance changed much over block and if there are ways that we modify– maybe shorter wait times between trials (current 1.5 seconds which is long).

```
OVP_4blocks = data.frame(matrix(nrow= nsub, ncol= 1))
  colnames(OVP_4blocks) <- c("subject")
# turn block to factor
OVP_dat$Spreadsheet.Name <- as.factor(OVP_dat$Spreadsheet.Name)
# count blocks
block <- length(unique(OVP_dat$Spreadsheet.Name))
# initiate a column
OVP_4blocks$tempt<-NA
# start row
myrow <- 0
# look at blocks
for (b in 1:block) {

  blockname <- levels(OVP_dat$Spreadsheet.Name)[b] # find subject subject
  myrows <- which(OVP_dat$Spreadsheet.Name==blockname) # select rows for this subject
  tmp <- data.frame(OVP_dat[myrows,])

  for (i in 1:nsubs) {

    mysub<-filter(tmp,side %in% c('left','right'),subject== allsum$subject[i])
      if(nrow(mysub)>10)
      {
```

```
        myt<-t.test(mysub$RT~mysub$side) #this is based on logRT
        OVP_4blocks$tempt[i]<-myt$statistic
        OVP_4blocks$subject[i]<-i
    }
  }
  wc<-which(colnames(OVP_4blocks)=='tempt')
  colnames(OVP_4blocks)[wc]<-paste0('block',b)
}
OVP_long <- gather(OVP_4blocks, block, LI, block1:block4, factor_key=TRUE)
# plot
pirateplot(formula = LI~ block,
           data = OVP_long,
           main = "OVP t-lat per block",
           theme = 2,
           ylab = "LI",
           xlab= "block")
```



There doesn't seem to be too much difference in LIs.

Dichotic listening

This is included here as we know it's reliable and shows differences between right and left. If it doesn't then maybe the previous measures didn't isn't such an issue.

This part of the notebook prcoesses the "Dichotic listening task' implmented online via Gorilla.sc which

assesses the lateralisation of receptive language. In the task, participants hear a different CV syllable in each ear. Participants report the syllable they heard clearest.

This task is near identical to that reported in Karlsson, Johnstone, and Carey (2019) and was original described in Hugdahl, Westerhausen, Alho, Medvedev, Laine, & Hämäläinen (2009).

```
origfile <- all_dat
# lowercase for responses
origfile$left_channel <- tolower(origfile$left_channel)
origfile$right_channel <- tolower(origfile$right_channel)
# make meaningful stimuli name
origfile$stimuli <- paste0(origfile$left_channel, "-", origfile$right_channel)
#As with earlier reading in, we default to saving dichotic_dat2 (day2), but at end of the chunk we reas
task <- "Dichotic Listening"
wanted <- c("subject", "stimuli", "Response", "RT", "left_channel", "right_channel") # select wanted co
disp <- "e_task"
dichotic_all <- procdata(origfile,wanted,task,disp) #use generic function (see above) to read in releva
dichotic_all <- dichotic_all[dichotic_all$Response != "AUDIO STARTED",]

dichotic_all$Correct <- 0 #default is error
w<-union(which(dichotic_all$Response==dichotic_all$left_channel),which(dichotic_all$Response==dichotic_a
dichotic_all$Correct[w] <-1
print(paste0('total Error (0) and Correct(1)'))
```

```
## [1] "total Error (0) and Correct(1)"
```

```
print(table(dichotic_all$Correct))
```

```
##
##   0   1
## 156 924
```

```
# count subjects
nsub <-length(unique(dichotic_all$subject))
print(paste(task,": subjects: ", nsub))
```

```
## [1] "Dichotic Listening : subjects:  30"
```

```
dichotic_same <- dichotic_all[dichotic_all$left_channel == dichotic_all$right_channel,] # create data f
dichotic_dat <- dichotic_all[dichotic_all$left_channel != dichotic_all$right_channel,] # remove items w
```

Now remove outliers, using same function as for CF.

```
dichotic_dat$RT <- log(dichotic_dat$RT)
dichotic_dat <- detect.outliers(dichotic_dat,log(200),1.65) #use generic outlier removal function
  outliertable<-table(dichotic_dat$subject,dichotic_dat$outlier)
```

Now, we judge participants' ability to correctly identify sounds using the "dichotic_same" dataframe. Participants are marked as excluded if their performance is below 75%.

```
dichotic_same <- dichotic_same
dichotic_diff <- dichotic_dat
# calculate participants average
sound_mean <- aggregate(FUN= mean, data= dichotic_same, Correct~ subject)
dichotic_acc <- aggregate(FUN= mean, data= dichotic_diff, Correct~ subject)
# add this to allsum

# Check if sound.recog columns already exist (ie if this chunk was run previously)
w <- which(colnames(allsum) %in% c('sound.recog','dich_acc'))
  if (length(w)>0){
  allsum<-allsum[,-w]
}

allsum <- merge(allsum, sound_mean, by= "subject",all=TRUE)
allsum <- merge(allsum, dichotic_acc, by= "subject",all.x=TRUE)

mycol <- length(colnames(allsum))
colnames(allsum)[(mycol-1):mycol]<-c('sound.recog','dichotic.acc')

# once again, mark rather than remove those with poor performance on identical stimuli
w<-which(allsum$sound.recog < .75)
allsum$exclude<- 0
ww<-intersect(which(allsum$exclude==1),w)
```

In this section, the correct sequences are marked as 1, incorrect asnwers are marked as 0. We then mark whether the correct answers were in the left or right ear.

```
# this will mark the side responded to. NB to allow for generic functions for all tasks, we use the lab
dichotic_dat <-
  dichotic_dat %>%
  mutate(side= ifelse(as.character(Response) == as.character(left_channel) & Correct == 1, "Left",
                      ifelse(as.character(Response) == as.character(right_channel) & Correct == 1, "Ri
# code side as factor
dichotic_dat$side <- as.factor(dichotic_dat$side)
```

Now the data is ready, create an empty data frame and populate this. It includes columns for the participant identifier, side, and count of accurate responses for each side.

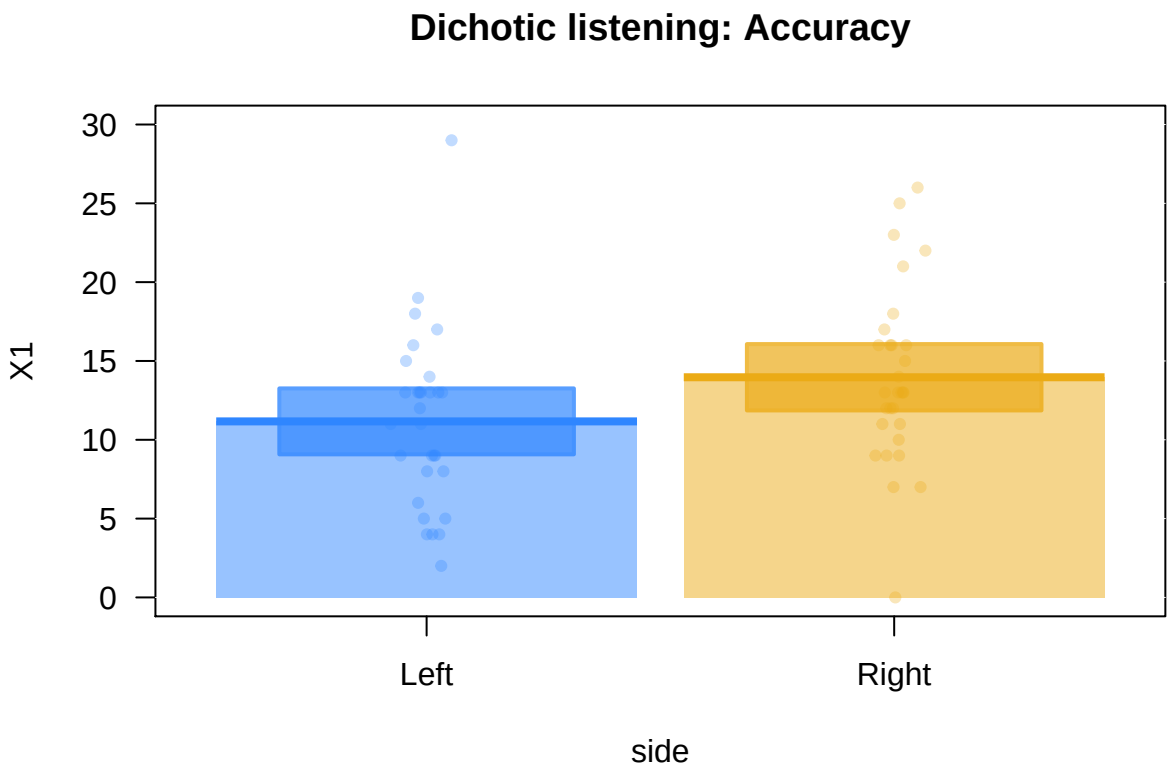
```
#We create DL_dat files for day1 and day 2
varlist <-   c("subject", "side", "accurate", "RT") #need to be in this order, ie sub, side, acc and RR
origdat <- dichotic_dat
latlist <- c("Left","Right") #names of factor levels
nloop=2 #same as levels of latlist
DL_dat <- make.df(dichotic_dat,varlist,latlist,nloop)

w<-which(is.na(DL_dat$subject))
if (length(w)>0) {DL_dat<-DL_dat[-w,]}
DL_dat[is.na(DL_dat)] <- 0
```

We have to remove cases where v few trials because of technical problems. Specify minimum of 80 (out of 120) trials. (Most detected by this proces have less than 10 trials!). NB This is only run for session 2 because there were no technical problems for session 1.

```
techerr<-table(dichotic_dat$subject)
te<-which(techerr<45)
techerr<-techerr[te] #subject numbers for those with few trials.
techerr<-as.numeric(names(techerr))
```

The data is visualised using the pirate plot function and t.test conducted.

```
includesubs <- which(allsum$exclude<1) #only plot/t-test included cases
origdat <- DL_dat[DL_dat$subject %in% includesubs,]
measure <- 'Accuracy'
origdat$X1 <- origdat$accurate #we used a dummy column name so we can vary the data that are used in th
task <- 'Dichotic listening'
dopirate(origdat,task,measure)
```




```
## [1] "Dichotic listeningAccuracy"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = -1.4133, df = 29, p-value = 0.1682
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.851959  1.251959
## sample estimates:
## mean of the differences
```

```
##                   -2.8
```

Here we use the Z score LI iscomputed. This is an individualised score that directly indicates whether or not the person is reliably lateralised on one test occasion.

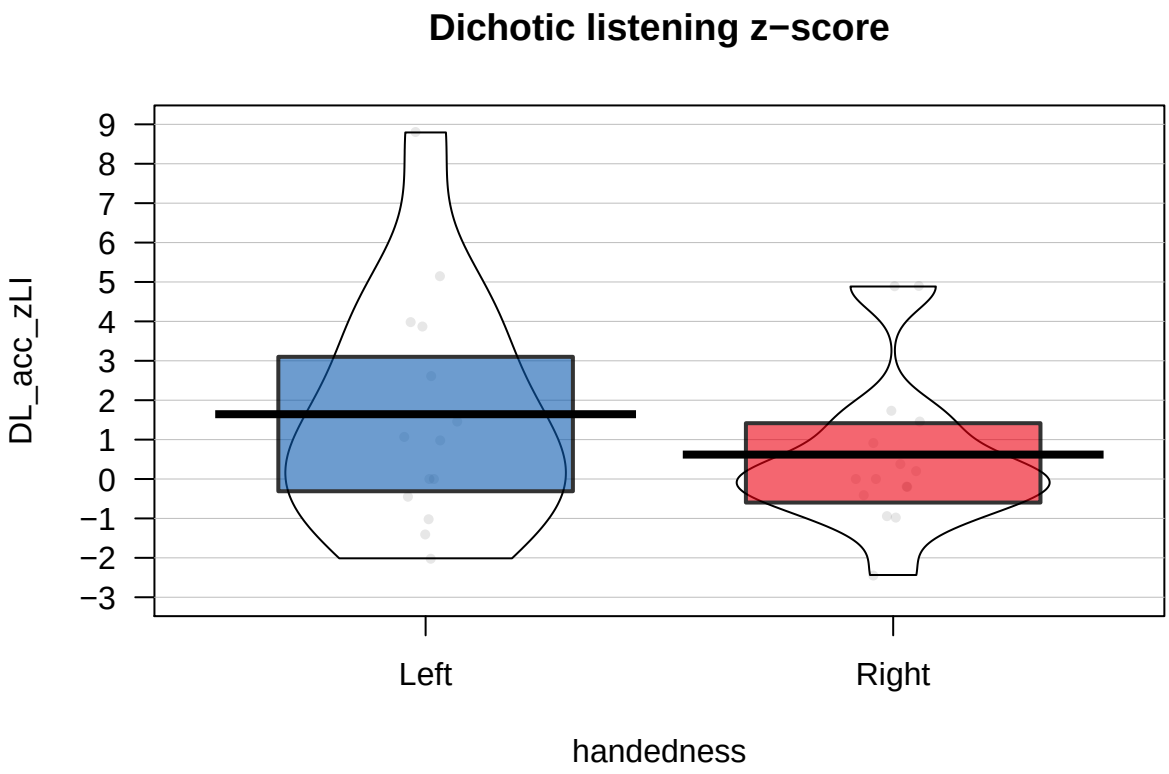
```
#We add dichotic LIs to allsum for day 1 and then day2
task<-'Dichotic listening'
origdat<-DL_dat
origdat$X1 <- origdat$accurate
mycolname<-"DL_acc_zLI"
mycutoff <- 10 #NB! need to check against prereg. I think this is N correct, rather than percentage. Pe
polarity<- (1)#so LI is positive in predicted direction
allsum <- makeZ(allsum,origdat,task,mycolname,polarity,mycutoff)
```

Is dichotic linked to handedness?

```
partsum <- allsum[allsum$subject %in% includesubs,]

t.test(partsum$DL_acc_zLI~partsum$handedness)
```

```
##
##  Welch Two Sample t-test
##
## data:  partsum$DL_acc_zLI by partsum$handedness
## t = 1.0783, df = 22.586, p-value = 0.2923
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.9427962  2.9914644
## sample estimates:
##  mean in group Left mean in group Right
##           1.6442843           0.6199502
```

```
pirateplot(formula = DL_acc_zLI ~ handedness,
                  data = partsum,
                  theme = 2,
                  main = "Dichotic listening z-score")
```



Not a lateralised as I thought… maybe we have odd Ps?

Colour Scales

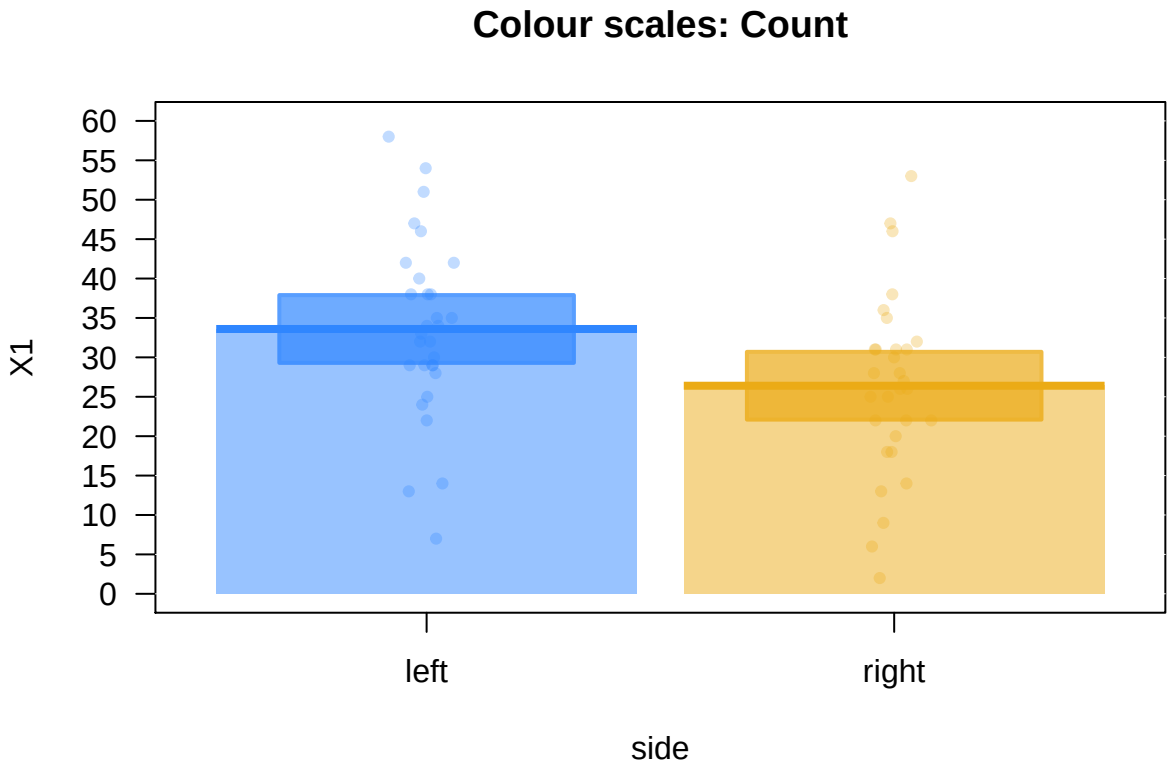
Here we score and assess the relibility of the colour scales task. First thing to do is read the data and mark the right or left bias associated with the response.

Once the data is ready, just get a count for LI caclulation.

```
origfile <- all_dat
task <- "Colour Scales"
wanted <- c("subject", "stimuli", "Response", "RT", "top_bias", "bottom_bias", "colour", "Spreadsheet.Na
disp <- "Task"
scales_all <- procdata(origfile,wanted,task,disp) #use generic function (see above) to read in relevant

# now mark response as a left or right bias
scales_all <- scales_all %>%
  mutate(side = if_else(Response== "Top", top_bias, bottom_bias))

# generate smaller df
scales_df <- scales_all %>% count(subject, side)
```

The data is visualised using the pirate plot function and t.test conducted.

```
origdat <- scales_df
measure <- 'Count'
origdat$X1 <- origdat$n #we used a dummy column name so we can vary the data that are used in the pirat
task <- 'Colour scales'
dopirate(origdat,task,measure)
```




```
## [1] "Colour scalesCount"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = 1.7128, df = 29, p-value = 0.09742
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.397238 15.797238
## sample estimates:
## mean of the differences
##                     7.2
```

Here we use the Z score version. This is an individualised score that directly indicates whether or not the person is reliably lateralised on one test occasion.

```
# code as left or right usings caps for later function
scales_df <- scales_df %>%
  mutate(side = if_else(side== "left", "Left", "Right"))
```

```
#We add dichotic LIs to allsum for day 1 and then day2
origdat<-scales_df
origdat$X1 <- origdat$n
mycolname<-"scales_Z"
mycutoff <- 10 #NB! need to check against prereg. I think this is N correct, rather than percentage. Pe
polarity<- (1)#so LI is positive in predicted direction
allsum <- makeZ(allsum,origdat,task,mycolname,polarity,mycutoff)
```

Is scales linked to handedness?

```
partsum <- allsum[allsum$subject %in% includesubs,]

t.test(partsum$scales_Z~partsum$handedness)
```

```
##
##  Welch Two Sample t-test
##
## data:  partsum$scales_Z by partsum$handedness
## t = -0.059108, df = 22.536, p-value = 0.9534
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.034831  3.810910
## sample estimates:
##  mean in group Left mean in group Right
##           -1.544974           -1.433013
```

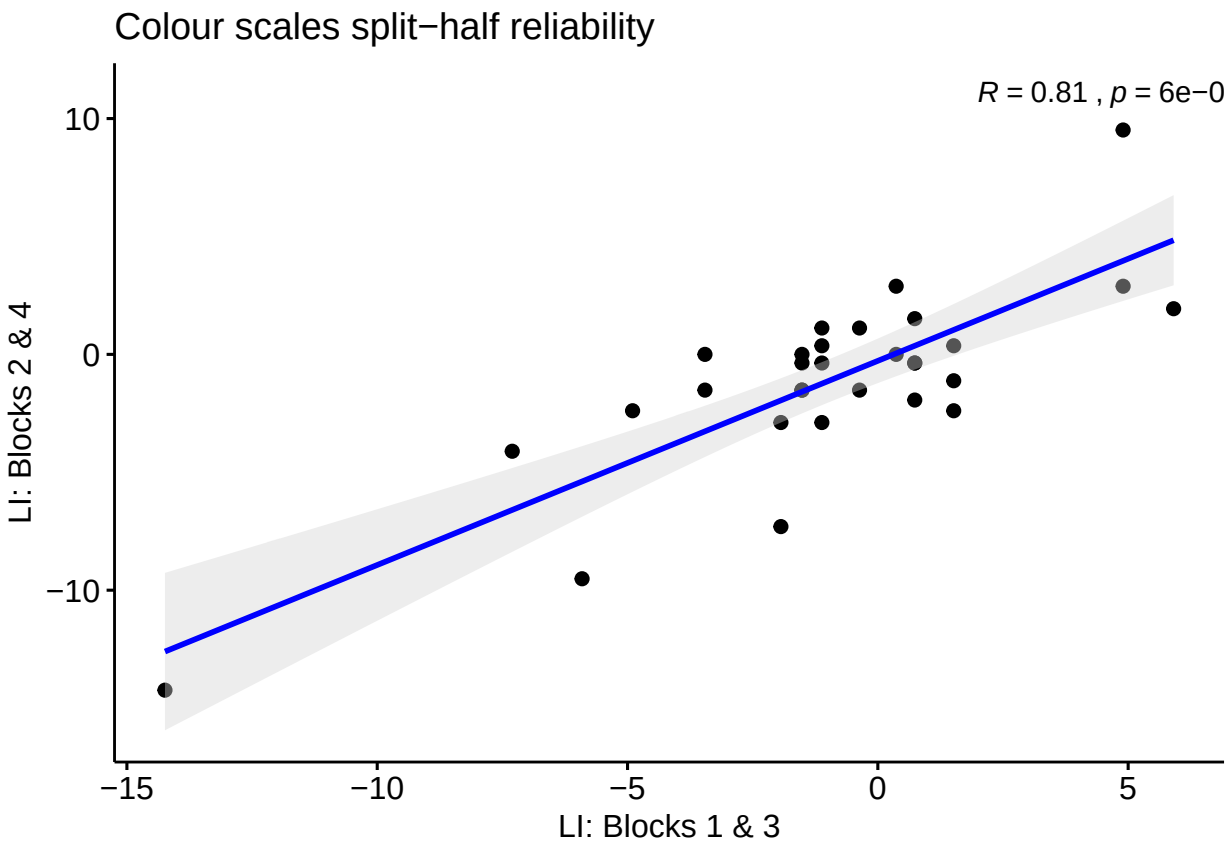
```
pirateplot(formula = scales_Z ~ handedness,
                  data = partsum,
                  theme = 2,
                  main = "Colour scales z-LI")
```

Split-half reliability

```
scales_split = data.frame()
# generate smaller df
scales_df2 <- scales_all %>% count(subject, side, Spreadsheet.Name)
# code to two level
scales_df2 <- scales_df2 %>% mutate(Spreadsheet.Name=
                                if_else(Spreadsheet.Name== "list 1" | Spreadsheet.Name== "list 3", "bl
# convert
scales_df2 <- aggregate(FUN= sum, n~ subject + side + Spreadsheet.Name, data= scales_df2)
# turn block to factor
scales_df2$Spreadsheet.Name <- as.factor(scales_df2$Spreadsheet.Name)
# count blocks
block <- length(unique(scales_df2$Spreadsheet.Name))
# look at blocks
for (b in 1:block) {

  blockname <- levels(scales_df2$Spreadsheet.Name)[b] # find subject subject
  myrows <- which(scales_df2$Spreadsheet.Name==blockname) # select rows for this subject
  tmp <- data.frame(scales_df2[myrows,])

  # code as left or right usings caps for later function
  tmp <- tmp %>%
    mutate(side = if_else(side== "left", "Left", "Right"))
```

```
  #We add dichotic LIs to allsum for day 1 and then day2
  origdat<-tmp
  origdat$X1 <- origdat$n
  mycolname<-"scales_Z"
  mycutoff <- 0 #NB! need to check against prereg. I think this is N correct, rather than percentage. P
  polarity<- (1)#so LI is positive in predicted direction

  newscales <- dplyr::select(allsum, subject)
  scales_block_LI <- makeZ(newscales,origdat,task,mycolname,polarity,mycutoff)
  scales_block_LI$block <- blockname

  df <- data.frame(scales_block_LI)
  scales_split <- rbind(scales_split, df)

}
# wide data
scales_split <- spread(scales_split, block, scales_Z)
# plot
ggscatter(scales_split, x = "block1", y = "block2",
          add = "reg.line", conf.int = TRUE, add.params = list(color = "blue",fill = "lightgray"),
          xlab= "LI: Blocks 1 & 3", ylab= "LI: Blocks 2 & 4", title = "Colour scales split-half reliabil
          stat_cor(method = "pearson", label.x = 2, label.y = 11)
```

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



Semantic task

Lastly, let's see if the semantic decision paradigm works as as visual half field paradigm.

```
origfile <- all_dat
task <- "Semantic Decision"
wanted <- c(c("Trial","subject", "Participant.Public.ID", "Correct", "RT",
              "ANSWER","Response","sound","image1","image2"))
disp <- "Trial"
semDec_all <- procdata(origfile,wanted,task,disp) #use generic function (see above) to read in relevant
# remove audio timestamps
semDec_all <- semDec_all[semDec_all$Response != "AUDIO STARTED",]
# recode images for consistency
colnames(semDec_all)[4] <- "side"
colnames(semDec_all)[6] <- "Left_image"
colnames(semDec_all)[7] <- "Right_image"
```

Now use generic function to remove outliers.

```
semDec_all$RT <- log(semDec_all$RT)
SD_dat<-detect.outliers(semDec_all,log(200),1.65) #numbers specify min RT and zscore for Hoaglin-Iglewi

outliertable<-table(SD_dat$subject,SD_dat$outlier)
```

A quick check of the outlier table shows that all seems to be ok!- this is suprising. Let's have a go at making the df.

```
varlist <-   c("subject", "side", "accurate", "RT") #need to be in this order, ie sub, side, acc and RR
latlist <- c("left","right") #names of factor levels
nloop<-length(latlist)
SD_df <- make.df(SD_dat,varlist,latlist,nloop)
# remove NaN
is.nan.data.frame <- function(x)
  do.call(cbind, lapply(x, is.nan))
RDT_df[is.nan(RDT_df)] <- NA
```

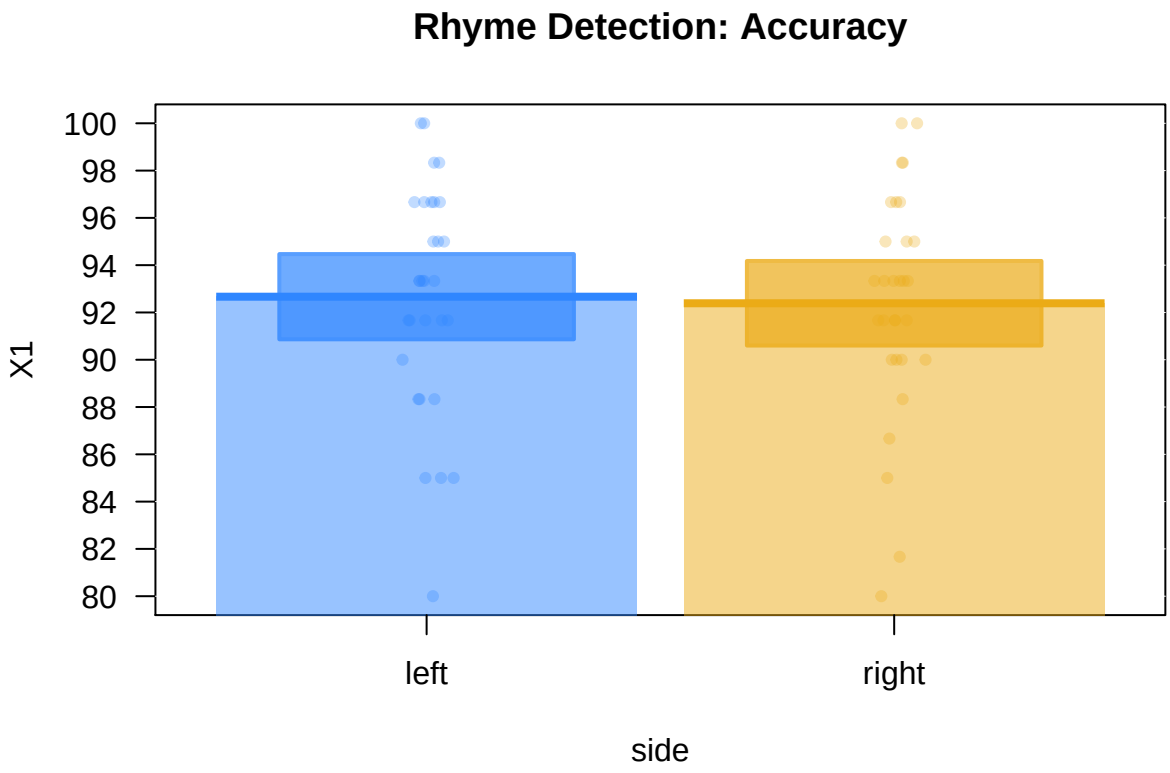
Next exclude those who score less than 50% correct responses on either side

```
wn <-unique(which(SD_df$accurate ==0)) #this catches those who were always wrong on 'no rhyme' trials,

w1<-which(SD_df$p.corr < 50)

exSD<-unique(c(SD_df$subject[w1],SD_df$subject[wn]))

allsum$exSD <- 0
allsum$exSD[allsum$subject %in% exSD]<-1 #decided best to have separate column for exclusion for each t
w<-which(allsum$exSD==0)
incSD <- allsum$subject[w]
ninc <- length(incSD)
print(paste0('SD: number remaining after exclusions = ',ninc))
```

```
## [1] "SD: number remaining after exclusions = 30"
```

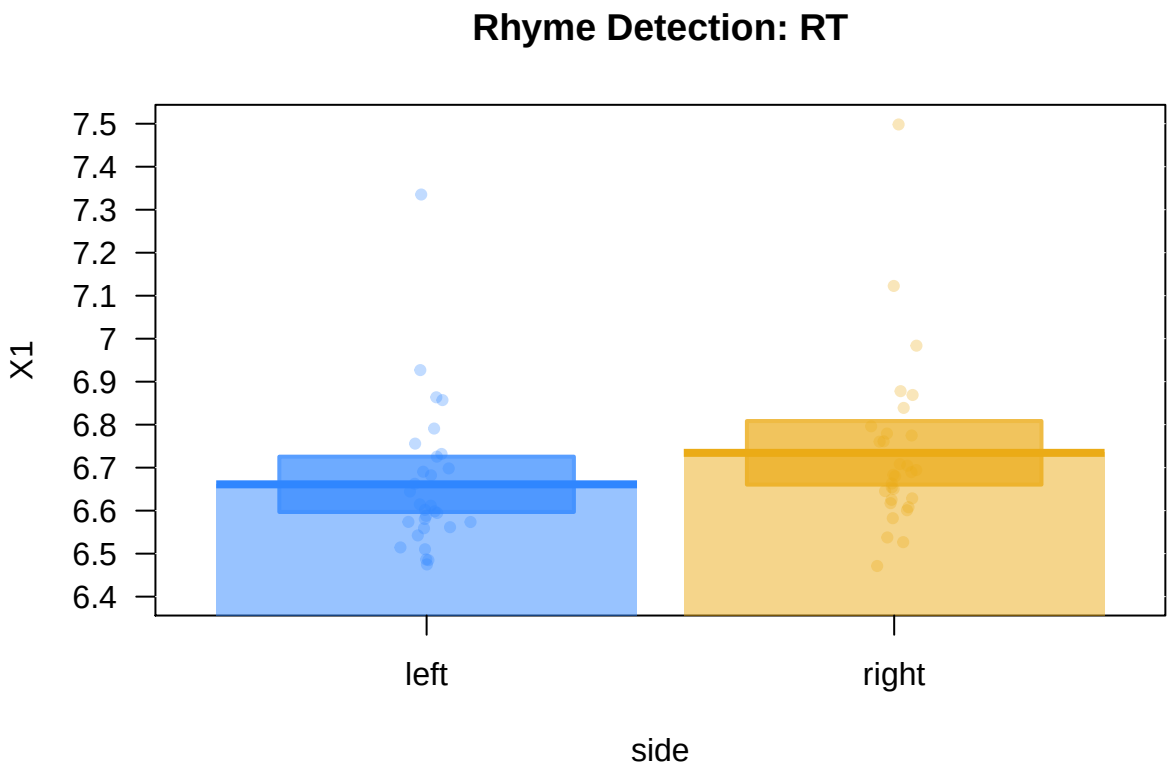
Now that we are ready to look at means in each visual field.

```
origdat <-SD_df[SD_df$subject %in% incSD,]

origdat <-origdat[origdat$side %in% c('left','right'),]

# accuracy
measure <- 'Accuracy'
origdat$X1 <- origdat$p.corr
task <- 'Rhyme Detection'
dopirate(origdat,task,measure)
```




```
## [1] "Rhyme DetectionAccuracy"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = 0.3725, df = 29, p-value = 0.7122
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.247376  1.802932
## sample estimates:
## mean of the differences
##               0.2777778
```

```
# RT
origdat$X1 <- origdat$RT
measure <- 'RT'
dopirate(origdat,task,measure)
```



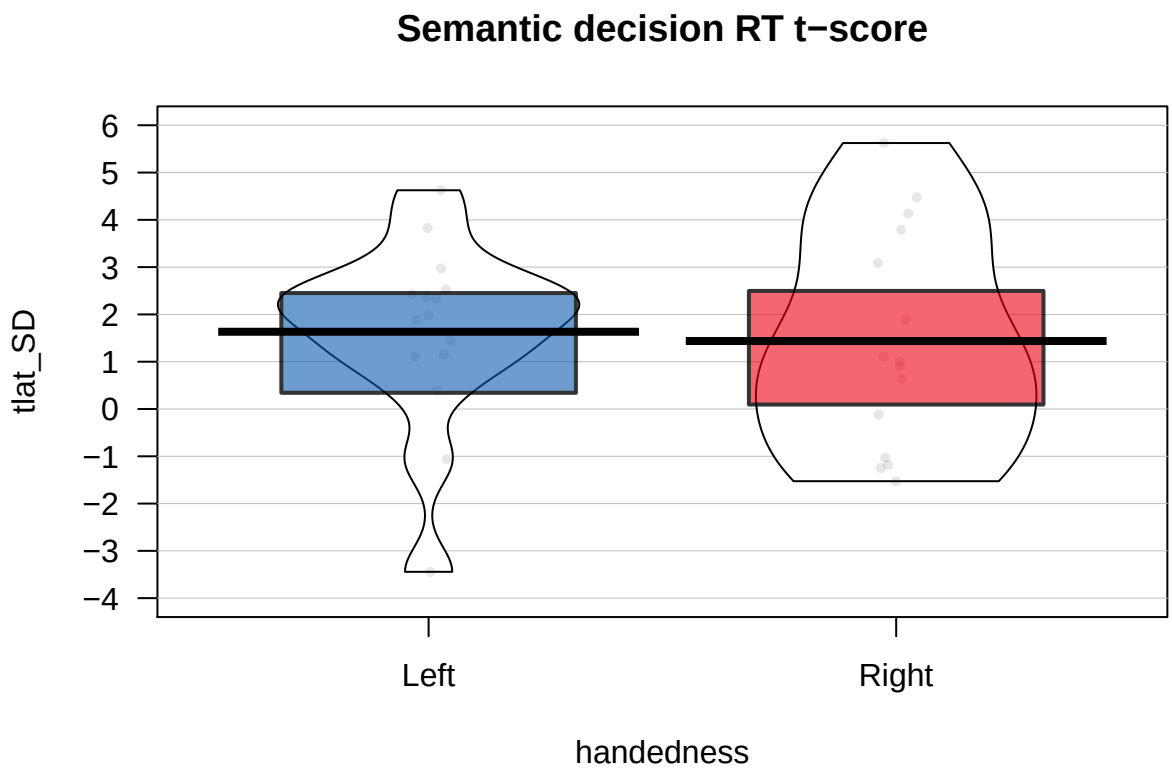

```
## [1] "Rhyme DetectionRT"
##
##  Paired t-test
##
## data:  origdat$X1 by origdat$side
## t = -4.185, df = 29, p-value = 0.0002416
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.10909191 -0.03746718
## sample estimates:
## mean of the differences
##             -0.07327954
```

Accuracy looks pretty good, and there is an odd LVF advanatge.

For conistency, we'll use a t-score rather than a z-score here.

```
myfile<-SD_dat

nsubs <- nrow(allsum)
```

```
allsum$tempt<-NA #initialise a new column

for (i in 1:nsubs) {
  mysub<-filter(myfile,side %in% c('left','right'),subject==allsum$subject[i])
    if(nrow(mysub)>10)
      {
      myt<-t.test(mysub$RT~mysub$side) #this is based on logRT
      allsum$tempt[i]<-myt$statistic*-1
    }
  }
wc<-which(colnames(allsum)=='tempt')
colnames(allsum)[wc]<-paste0('tlat_SD')

pirateplot(formula = tlat_SD ~ handedness,
                  data = allsum,
                  theme = 2,
                  main = "Semantic decision RT t-score")
```




```
print(paste0("Testing whether LI is different from zero, p= ", t.test(allsum$tlat_RDT)[3]))
```

```
## [1] "Testing whether LI is different from zero, p= 0.842177079423411"
```

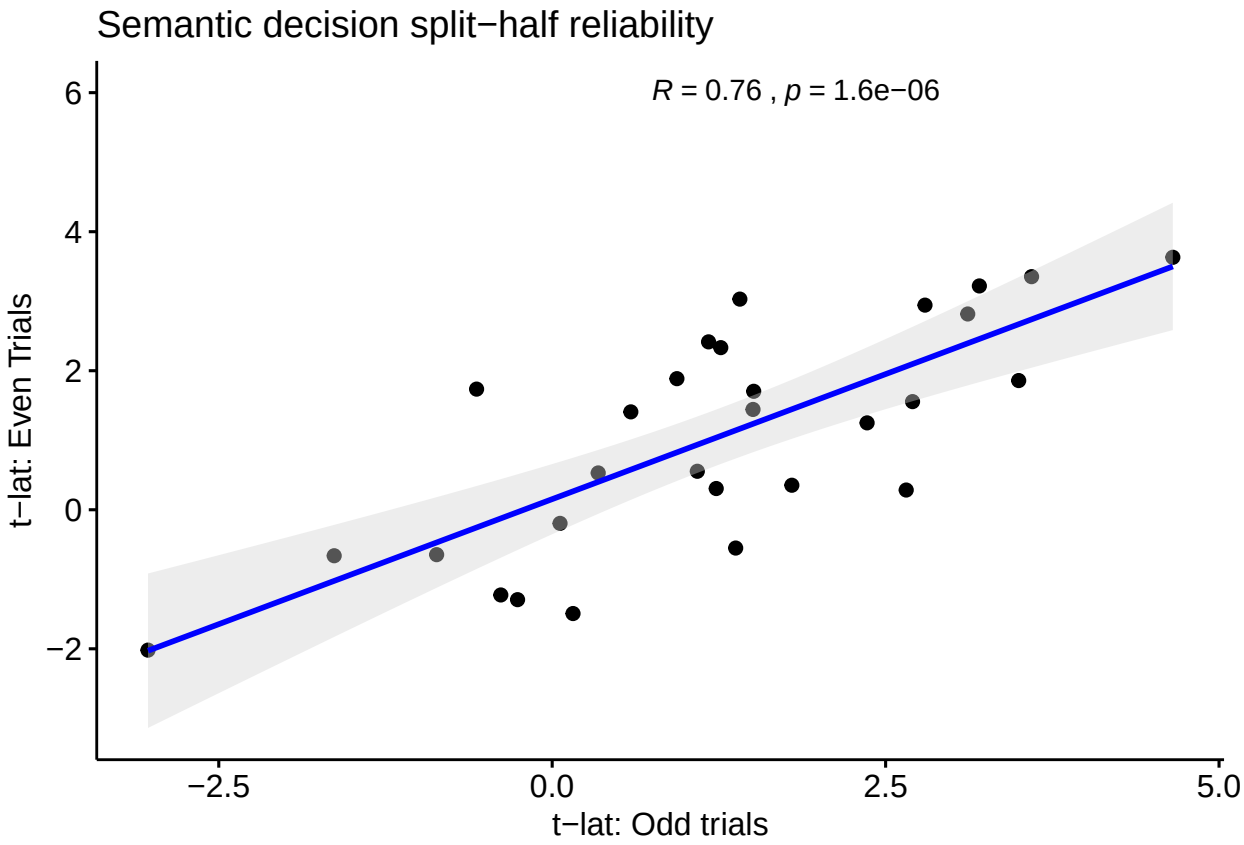
This starts to look lateralised, i.e. away from zero. I'd be keen to know how this looks with Zoe's new items.

Split-half reliability

Let's check the split-half for the semantic decision. As there was only one block, we'll do this in the truest sense (odds vs evens).

```
SD_split = data.frame(matrix(nrow= nsub, ncol= 1))
  colnames(SD_split) <- c("subject")
# code to two level
SD_data <- SD_dat
# label odds and evens
SD_data$group <- c("odds", "evens")
  SD_data$group <- as.factor(SD_data$group)
# count blocks
block <- length(unique(SD_data$group))
# initiate a column
SD_split$tempt<-NA
# start row
myrow <- 0
# look at blocks
for (b in 1:block) {

  blockname <- levels(SD_data$group)[b] # find subject subject
  myrows <- which(SD_data$group==blockname) # select rows for this subject
  tmp <- data.frame(SD_data[myrows,])

  for (i in 1:nsubs) {

    mysub<-filter(tmp,side %in% c('left','right'),subject== allsum$subject[i])
      if(nrow(mysub)>10)
      {
        myt<-t.test(mysub$RT~mysub$side) #this is based on logRT
        SD_split$tempt[i]<-myt$statistic*-1
        SD_split$subject[i]<-i
    }
  }
  wc<-which(colnames(SD_split)=='tempt')
  colnames(SD_split)[wc]<-paste0('tlat_SD',b)
}
# plot
ggscatter(subset(SD_split, subject != 8), x = "tlat_SD1", y = "tlat_SD2",
          add = "reg.line", conf.int = TRUE, add.params = list(color = "blue",fill = "lightgray"),
          xlab= "t-lat: Odd trials", ylab= "t-lat: Even Trials", title = "Semantic decision split-half
          stat_cor(method = "pearson", label.x = .75, label.y = 6)
```

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



The split half relibaility seems good for the semantic decision task when using the t-lat.