Modality effects in a signalling game

Intro

This script uses data compiled by *analyseData.R*.

Load libraries

```
library(lme4)
library(sjPlot)
library(ggplot2)
library(lattice)
library(influence.ME)
```

Load data

```
d = read.csv("../../data/FinalSignalData.csv")
```

Work out number of turns in each trial.

```
# Number of turns in each trial
numTurns = tapply(d$trialString, d$trialString, length)
d$numberOfTurns = numTurns[d$trialString]
```

We don't need info on every signal in each turn, just the trial time. Keep only 1st signal in each trial.

```
d = d[!duplicated(d$trialString),]
```

Descriptive stats

Here is a graph showing the distribution of trial lengths by conditions:

The distribution of trial times is very skewed:

```
hist(d$trialLength)
```

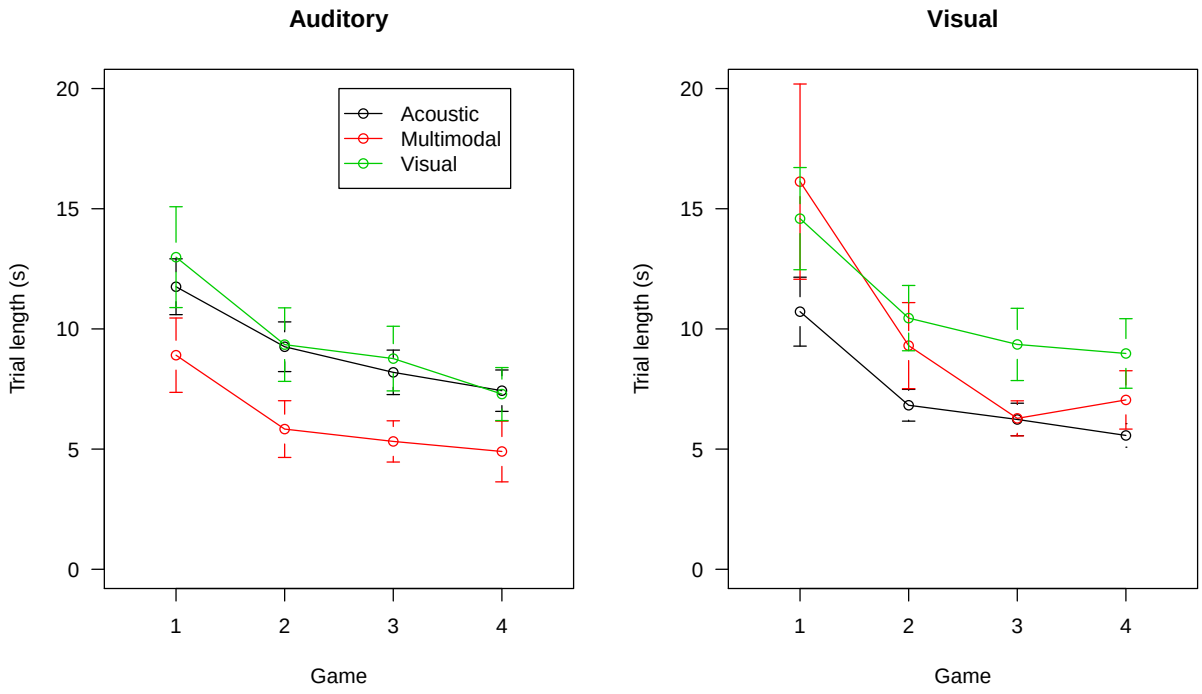
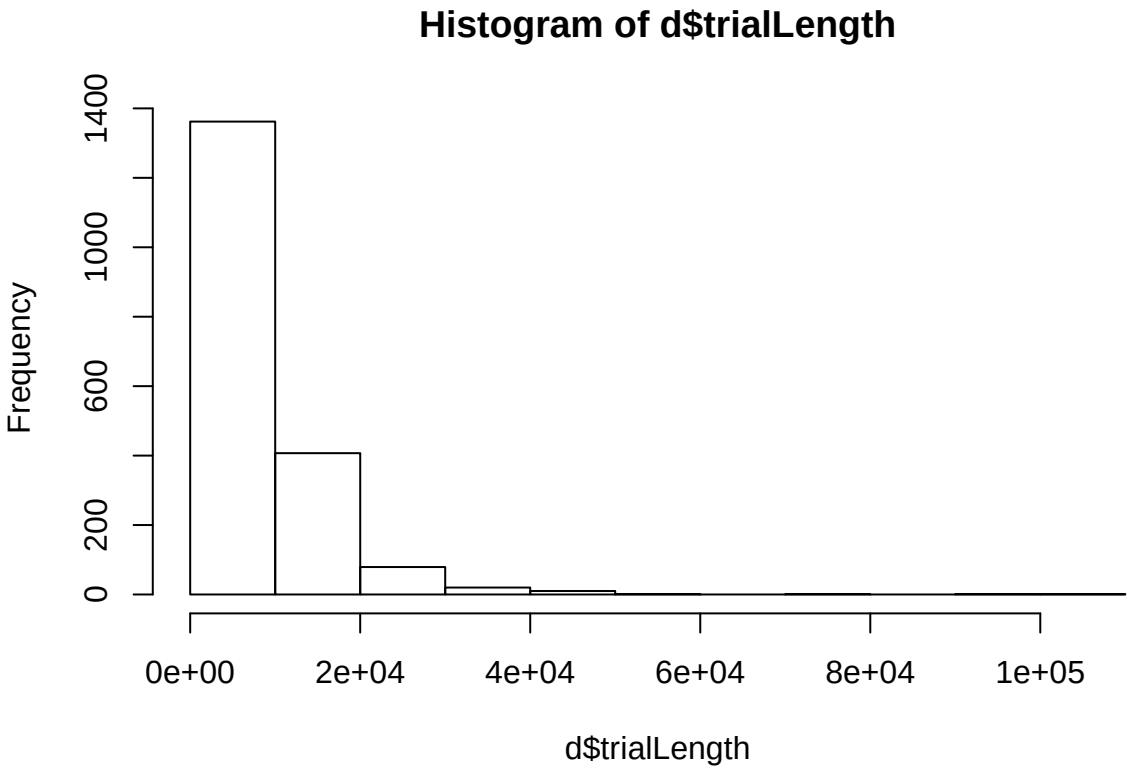
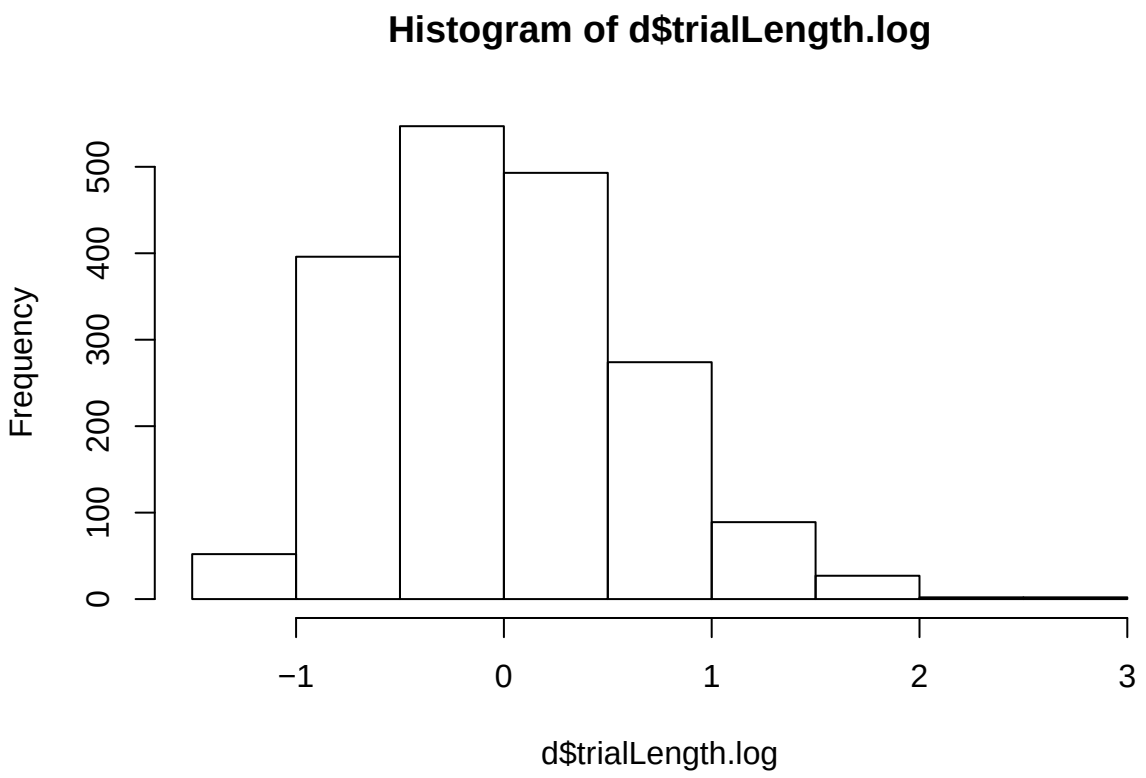


Figure 1: The efficiency of trials in different conditions



So we transform it using a log transform, then center the data.

```
d$trialLength.log = log(d$trialLength)
meanLogTrialLength = mean(d$trialLength.log)
d$trialLength.log = d$trialLength.log - meanLogTrialLength
hist(d$trialLength.log)
```



Make a variable to represent proportion of games played:

```
# Make a variable that represents the number of trials played
d$trialTotal = d$trial + (d$game * (max(d$trial)+1))
# Convert to proportion of games played, so that estimates reflect change per game.
d$trialTotal = d$trialTotal / 16
# Center the trialTotal variable so intercept reflects after the first game
d$trialTotal = d$trialTotal - 1
```

Make a variable for which stimuli the players experienced first.

```
firstBlock = tapply(as.character(d$condition),d$dyadNumber,head,n=1)
d$firstBlock = as.factor(firstBlock[match(d$dyadNumber,names(firstBlock))])
```

Reorder some levels so that the intercept reflects the most frequent condition.

```
d$incorrect = !d$correct
```

```
turnD = read.csv("../../data/Final_Turn_data.csv")
turnD = turnD[turnD$turnType=="T1",]
turnD = turnD[turnD$role == "Director",]
d$multimodal = turnD[match(d$trialString, turnD$trialString),]$turnModalityType == "multi"
d$multimodal[is.na(d$multimodal)] = F
```

Mixed models

Make a series of models with random effects for dyad, director (nested within dyad) and item.

Not all random slopes are appropriate. For example, items are used in only one stimulus condition, so a random slope for condition by item is not appropriate. Similarly, each dyad only plays in one modality condition.

It is reasonable to have a random slope for trial by dyad, but this caused unreliable model convergence, so is not included.

The final random slopes were for condition and incorrectness by dyad/player, and modality condition by item.

```
# No fixed effects
m0 =  lmer(trialLength.log ~ 1 +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
           data=d, REML = FALSE)
# Add modality condition
modality = lmer(trialLength.log ~ 1 + modalityCondition +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
           data=d, REML = FALSE)
# Add stimulus condition
cond = lmer(trialLength.log ~ 1 + modalityCondition + condition +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
           data=d, REML = FALSE)
# Add trial total
game = lmer(trialLength.log ~ 1 + modalityCondition + condition + trialTotal +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
           data=d, REML = FALSE)
```

```
# Add interaction between condition and stimulus condition
modXcond =  lmer(trialLength.log ~ 1 + modalityCondition * condition + trialTotal +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
           data=d, REML = FALSE)
# Add interaction between condition and trial
conXgame = lmer(trialLength.log ~ 1 + (modalityCondition * condition) + trialTotal +
             (trialTotal:condition) +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
           data=d, REML = FALSE)
# Add interaction between modality and trial
modXgame = lmer(trialLength.log ~ 1 + (modalityCondition * condition) + trialTotal +
             (trialTotal:condition) + (modalityCondition:game) +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
           data=d, REML = FALSE)

# Add 3-way interaction
moXcoXga = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
             (1 + condition + incorrect |dyadNumber/playerId) +
             (1 + modalityCondition|itemId),
```

```
          data=d, REML = FALSE)

# Add number of turns
nTurns = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
                numberOfTurns +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
# interaction between turns and modality
nTurnXmo = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
                numberOfTurns*modalityCondition +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)

nTurnXco = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
                numberOfTurns*modalityCondition+ numberOfTurns:condition +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)

tuXmoXco = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
                numberOfTurns*modalityCondition*condition +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)

# Add whether the response was incorrect
incor = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            numberOfTurns*modalityCondition*condition +
            incorrect +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
```

```
# Add the interaction between modality and incorrectness
moXincor = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            numberOfTurns*modalityCondition*condition +
            incorrect + (modalityCondition:incorrect) +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
# Add the interaction between condition and incorrectness
coXincor = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            numberOfTurns*modalityCondition*condition +
            incorrect + (modalityCondition:incorrect) +
              (condition:incorrect) +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
# Add the three-way interaction between condition, modality and incorrectness
coXmoXin = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            numberOfTurns*modalityCondition*condition +
            (incorrect*condition*modalityCondition)+
```

```
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)

# Add multimodal signal

multim = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            numberOfTurns*modalityCondition*condition +
            (incorrect*condition*modalityCondition)+
            multimodal +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge: degenerate Hessian with 1 negative
## eigenvalues
```

```
multiXco = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            numberOfTurns*modalityCondition*condition +
            (incorrect*condition*modalityCondition)+
            multimodal*condition +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge: degenerate Hessian with 1 negative
## eigenvalues
```

```
# Add the quadratic effect of trial
gamQuad = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            I(trialTotal^2) +
            numberOfTurns*modalityCondition*condition +
            (incorrect*condition*modalityCondition)+
            multimodal*condition +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge: degenerate Hessian with 1 negative
## eigenvalues
```

```
# Add interaction between quadratic effect of trial and modality
modXgamQ = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            I(trialTotal^2) +
```

```
            numberOfTurns*modalityCondition*condition +
            (incorrect*condition*modalityCondition)+
            multimodal*condition +
            (modalityCondition:I(trialTotal^2)) +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge: degenerate Hessian with 1 negative
## eigenvalues
```

```
# Add block order
block = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            I(trialTotal^2) +
            numberOfTurns*modalityCondition*condition +
            (incorrect*condition*modalityCondition)+
            multimodal*condition +
            (modalityCondition:I(trialTotal^2)) +
            firstBlock +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = FALSE)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : unable to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge: degenerate Hessian with 1 negative
## eigenvalues
```

```
# Add interaction between block order and modality
blocXmod = lmer(trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
            I(trialTotal^2) +
            numberOfTurns*modalityCondition*condition +
            (incorrect*condition*modalityCondition)+
            multimodal*condition +
            (modalityCondition:I(trialTotal^2)) +
            firstBlock * modalityCondition +
            (1 + condition + incorrect |dyadNumber/playerId) +
            (1 + modalityCondition|itemId),
          data=d, REML = TRUE)   # Last model is REML to get estimates
```

Results

Compare the fit of the models:

```
modelComparison = anova(m0,modality,cond,game,modXcond,conXgame, modXgame,
      moXcoXga,nTurns,nTurnXmo,nTurnXco,tuXmoXco,
      incor,moXincor,coXincor,coXmoXin,
      multim,multiXco,
      gamQuad,modXgamQ,block, blocXmod)
```

```
## refitting model(s) with ML (instead of REML)
```

```
modelComparison
```

```
## Data: d
## Models:
## m0: trialLength.log ~ 1 + (1 + condition + incorrect | dyadNumber/playerId) +
## m0:     (1 + modalityCondition | itemId)
## modality: trialLength.log ~ 1 + modalityCondition + (1 + condition + incorrect |
## modality:     dyadNumber/playerId) + (1 + modalityCondition | itemId)
## cond: trialLength.log ~ 1 + modalityCondition + condition + (1 + condition +
## cond:     incorrect | dyadNumber/playerId) + (1 + modalityCondition |
## cond:     itemId)
## game: trialLength.log ~ 1 + modalityCondition + condition + trialTotal +
## game:     (1 + condition + incorrect | dyadNumber/playerId) + (1 +
## game:     modalityCondition | itemId)
## modXcond: trialLength.log ~ 1 + modalityCondition * condition + trialTotal +
## modXcond:     (1 + condition + incorrect | dyadNumber/playerId) + (1 +
## modXcond:     modalityCondition | itemId)
## conXgame: trialLength.log ~ 1 + (modalityCondition * condition) + trialTotal +
## conXgame:     (trialTotal:condition) + (1 + condition + incorrect | dyadNumber/playerId) +
## conXgame:     (1 + modalityCondition | itemId)
## modXgame: trialLength.log ~ 1 + (modalityCondition * condition) + trialTotal +
## modXgame:     (trialTotal:condition) + (modalityCondition:game) + (1 +
## modXgame:     condition + incorrect | dyadNumber/playerId) + (1 + modalityCondition |
## modXgame:     itemId)
## moXcoXga: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## moXcoXga:     (1 + condition + incorrect | dyadNumber/playerId) + (1 +
## moXcoXga:     modalityCondition | itemId)
## nTurns: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## nTurns:     numberOfTurns + (1 + condition + incorrect | dyadNumber/playerId) +
## nTurns:     (1 + modalityCondition | itemId)
## nTurnXmo: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## nTurnXmo:     numberOfTurns * modalityCondition + (1 + condition + incorrect |
## nTurnXmo:     dyadNumber/playerId) + (1 + modalityCondition | itemId)
## nTurnXco: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## nTurnXco:     numberOfTurns * modalityCondition + numberOfTurns:condition +
## nTurnXco:     (1 + condition + incorrect | dyadNumber/playerId) + (1 +
## nTurnXco:     modalityCondition | itemId)
## tuXmoXco: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## tuXmoXco:     numberOfTurns * modalityCondition * condition + (1 + condition +
## tuXmoXco:     incorrect | dyadNumber/playerId) + (1 + modalityCondition |
## tuXmoXco:     itemId)
## incor: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## incor:     numberOfTurns * modalityCondition * condition + incorrect +
```

```
## incor:     (1 + condition + incorrect | dyadNumber/playerId) + (1 +
## incor:     modalityCondition | itemId)
## moXincor: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## moXincor:     numberOfTurns * modalityCondition * condition + incorrect +
## moXincor:     (modalityCondition:incorrect) + (1 + condition + incorrect |
## moXincor:     dyadNumber/playerId) + (1 + modalityCondition | itemId)
## coXincor: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## coXincor:     numberOfTurns * modalityCondition * condition + incorrect +
## coXincor:     (modalityCondition:incorrect) + (condition:incorrect) + (1 +
## coXincor:     condition + incorrect | dyadNumber/playerId) + (1 + modalityCondition |
## coXincor:     itemId)
## coXmoXin: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## coXmoXin:     numberOfTurns * modalityCondition * condition + (incorrect *
## coXmoXin:     condition * modalityCondition) + (1 + condition + incorrect |
## coXmoXin:     dyadNumber/playerId) + (1 + modalityCondition | itemId)
## multim: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## multim:     numberOfTurns * modalityCondition * condition + (incorrect *
## multim:     condition * modalityCondition) + multimodal + (1 + condition +
## multim:     incorrect | dyadNumber/playerId) + (1 + modalityCondition |
## multim:     itemId)
## multiXco: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## multiXco:     numberOfTurns * modalityCondition * condition + (incorrect *
## multiXco:     condition * modalityCondition) + multimodal * condition +
## multiXco:     (1 + condition + incorrect | dyadNumber/playerId) + (1 +
## multiXco:     modalityCondition | itemId)
## gamQuad: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## gamQuad:     I(trialTotal^2) + numberOfTurns * modalityCondition * condition +
## gamQuad:     (incorrect * condition * modalityCondition) + multimodal *
## gamQuad:     condition + (1 + condition + incorrect | dyadNumber/playerId) +
## gamQuad:     (1 + modalityCondition | itemId)
## modXgamQ: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## modXgamQ:     I(trialTotal^2) + numberOfTurns * modalityCondition * condition +
## modXgamQ:     (incorrect * condition * modalityCondition) + multimodal *
## modXgamQ:     condition + (modalityCondition:I(trialTotal^2)) + (1 + condition +
## modXgamQ:     incorrect | dyadNumber/playerId) + (1 + modalityCondition |
## modXgamQ:     itemId)
## block: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## block:     I(trialTotal^2) + numberOfTurns * modalityCondition * condition +
## block:     (incorrect * condition * modalityCondition) + multimodal *
## block:     condition + (modalityCondition:I(trialTotal^2)) + firstBlock +
## block:     (1 + condition + incorrect | dyadNumber/playerId) + (1 +
## block:     modalityCondition | itemId)
## blocXmod: trialLength.log ~ 1 + modalityCondition * condition * trialTotal +
## blocXmod:     I(trialTotal^2) + numberOfTurns * modalityCondition * condition +
## blocXmod:     (incorrect * condition * modalityCondition) + multimodal *
## blocXmod:     condition + (modalityCondition:I(trialTotal^2)) + firstBlock *
## blocXmod:     modalityCondition + (1 + condition + incorrect | dyadNumber/playerId) +
## blocXmod:     (1 + modalityCondition | itemId)
##           Df    AIC    BIC   logLik deviance    Chisq Chi Df Pr(>Chisq)
## m0        20 2694.4 2805.2 -1327.21   2654.4
## modality  22 2696.0 2817.9 -1326.01   2652.0   2.4050      2   0.300444
## cond      23 2697.3 2824.7 -1325.65   2651.3   0.7147      1   0.397881
## game      24 2309.0 2441.9 -1130.48   2261.0 390.3345      1  < 2.2e-16 ***
## modXcond  26 2301.2 2445.2 -1124.59   2249.2  11.7749      2   0.002774 **
```

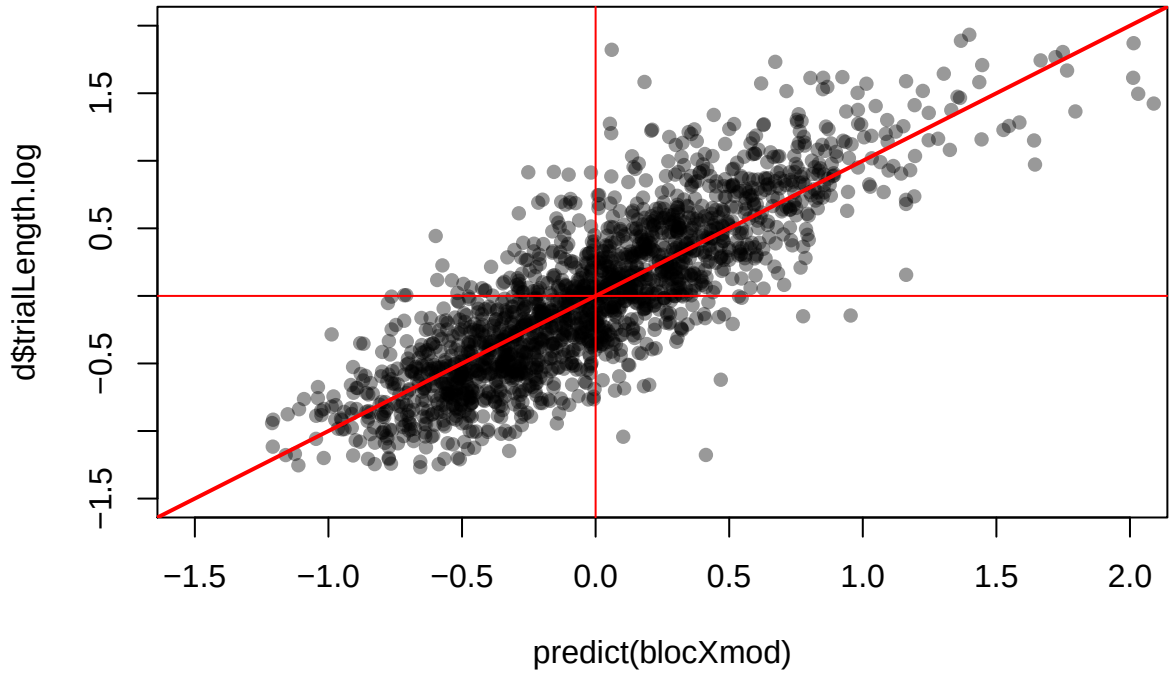
```
## conXgame 27 2302.9 2452.4 -1124.44   2248.9   0.3160      1   0.574037
## modXgame 30 2303.3 2469.5 -1121.66   2243.3   5.5434      3   0.136068
## moXcoXga 31 2302.1 2473.9 -1120.06   2240.1   3.2019      1   0.073555 .
## nTurns   32 1733.7 1910.9  -834.83   1669.7 570.4654      1  < 2.2e-16 ***
## nTurnXmo 34 1678.5 1866.9  -805.27   1610.5  59.1307      2  1.445e-13 ***
## nTurnXco 35 1672.5 1866.3  -801.22   1602.5   8.0855      1   0.004462 **
## tuXmoXco 37 1670.7 1875.7  -798.36   1596.7   5.7226      2   0.057194 .
## incor    38 1655.6 1866.1  -789.79   1579.6  17.1487      1  3.456e-05 ***
## moXincor 40 1656.4 1878.0  -788.22   1576.4   3.1389      2   0.208159
## coXincor 41 1657.8 1884.9  -787.89   1575.8   0.6519      1   0.419420
## coXmoXin 43 1660.6 1898.8  -787.30   1574.6   1.1741      2   0.555976
## multim   44 1643.0 1886.7  -777.49   1555.0  19.6374      1  9.362e-06 ***
## multiXco 45 1644.4 1893.7  -777.21   1554.4   0.5494      1   0.458545
## gamQuad  46 1592.9 1847.7  -750.43   1500.9  53.5663      1  2.500e-13 ***
## modXgamQ 48 1587.7 1853.6  -745.85   1491.7   9.1637      2   0.010236 *
## block    49 1589.5 1860.9  -745.72   1491.5   0.2431      1   0.621980
## blocXmod 51 1591.2 1873.7  -744.59   1489.2   2.2635      2   0.322475
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pick final model for estimates:

```
finalModel = blocXmod
```

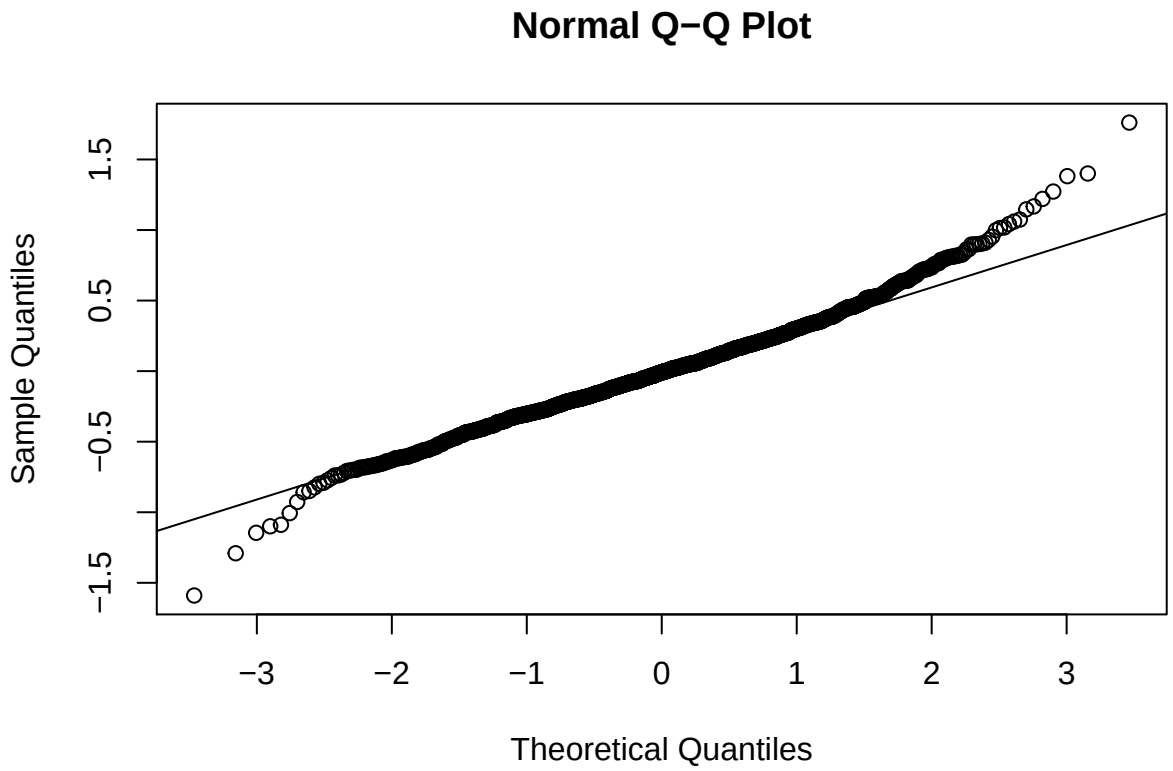
Check model predictions. The model predictions are in the right range and direction, fitting linear quite well:

```
plot(predict(blocXmod),d$trialLength.log, pch=16, col=rgb(0,0,0,0.4),
     ylim=c(-1.5,2),xlim=c(-1.5,2))
abline(a=0,b=1, col=2, lwd=2)
abline(h=0, col=2)
abline(v=0, col=2)
```



The residuals are ok, though it tends to do worse at higher values. This is expected from using the log scale.

```
qqnorm(resid(blocXmod))
qqline(resid(blocXmod))
```



Plot the fixed effects

Relabel the effects:

```
feLabels = matrix(c(
"(Intercept)"              ,"Intercept"      , NA,
"modalityConditionvisual" ,"Visual modality", "modality",
"modalityConditionvocal"  , "Acoustic modality", "modality",
"conditionVisual" , "Visual stimuli","cond",
"trialTotal"              , "Game","game",
"modalityConditionvisual:conditionVisual"  , "Visual modality:Visual stimuli", "modXcond",
"modalityConditionvocal:conditionVisual" , "Acoustic modality:Visual stimuli","modXcond",
"modalityConditionvisual:trialTotal"    , "Visual modality:Game","modXgame",
"modalityConditionvocal:trialTotal"     , "Acoustic modality:Game", "modXgame",
"conditionVisual:trialTotal"             , "Visual stimuli:Game","conXgame",
"modalityConditionvisual:conditionVisual:trialTotal", "Visual modality:Visual stimuli:Game", "moXcoXga"
"modalityConditionvocal:conditionVisual:trialTotal", "Acoustic modality:Visual stimuli:Game", "moXcoXga
"incorrectTRUE","Inorrect","incor",
"modalityConditionvisual:incorrectTRUE","Visual modality:Incorrect","moXincor",
"modalityConditionvocal:incorrectTRUE","Acoustic modality:Incorrect","moXincor",
"modalityConditionvisual:I(trialTotal^2)", "Visual modality:Game^2","modXgamQ",
"modalityConditionvocal:I(trialTotal^2)", "Acoustic modality:Game^2","modXgamQ",
"I(trialTotal^2)","Game^2","gamQuad",
"firstBlockVisual","Visual stims first","block",
"modalityConditionvisual:firstBlockVisual","Visual modality:Visual stim first","blocXmod",
```

```
"modalityConditionvocal:firstBlockVisual","Acoustic modality:Visual stim first","blocXmod",
"conditionVisual:incorrectTRUE","Visual stimuli:incorrect","coXincor",
"modalityConditionvisual:conditionVisual:incorrectTRUE","Visual modality:Visual stimuli:incorrect","coXm
"modalityConditionvocal:conditionVisual:incorrectTRUE","Acoustic modality:Visual stimuli:incorrect","coX

"modalityConditionvisual:conditionVisual:numberOfTurns","VisualModality:Visual stim:NumTurns","tuXmoXco
"modalityConditionvocal:conditionVisual:numberOfTurns","Vocal Modality:Visual stim:NumTurns","tuXmoXco"
"conditionVisual:numberOfTurns","Visual stim:NumTurns","nTurnXco",
"modalityConditionvisual:numberOfTurns","VisualModality:NumTurns","nTurnXmo",
"modalityConditionvocal:numberOfTurns","Vocal Modality:NumTurns","nTurnXmo",
"numberOfTurns","Number of turns","nTurns",
"multimodalTRUE","Multimodal T1","multim",
"conditionVisual:multimodalTRUE","VisualStim:MultimodalT1","multiXco"
), ncol=3, byrow = T)

feLabels2 = as.vector(feLabels[match(names(fixef(finalModel)),feLabels[,1]),2])
feModel = as.vector(feLabels[match(names(fixef(finalModel)),feLabels[,1]),3])

sig = modelComparison$`Pr(>Chisq)`
names(sig) = rownames(modelComparison)

sig.data = data.frame(estimate = fixef(finalModel),
                      y=1:length(fixef(finalModel)),
                      sig=sig[feModel])

cols= c("black",'red')
sig.data$pointCol = cols[1]
sig.data$pointCol[!is.na(sig.data$sig)] =
  cols[1 + (sig.data$sig[!is.na(sig.data$sig)] < 0.05)]
# Mark marginal effects
#sig.data$pointCol[!is.na(sig.data$sig) &
#                    sig.data$sig < 0.1 &
#                    sig.data$sig >=0.05] = "orange"

sig.data$fade = F
```

Plot the strength of the fixed effects:

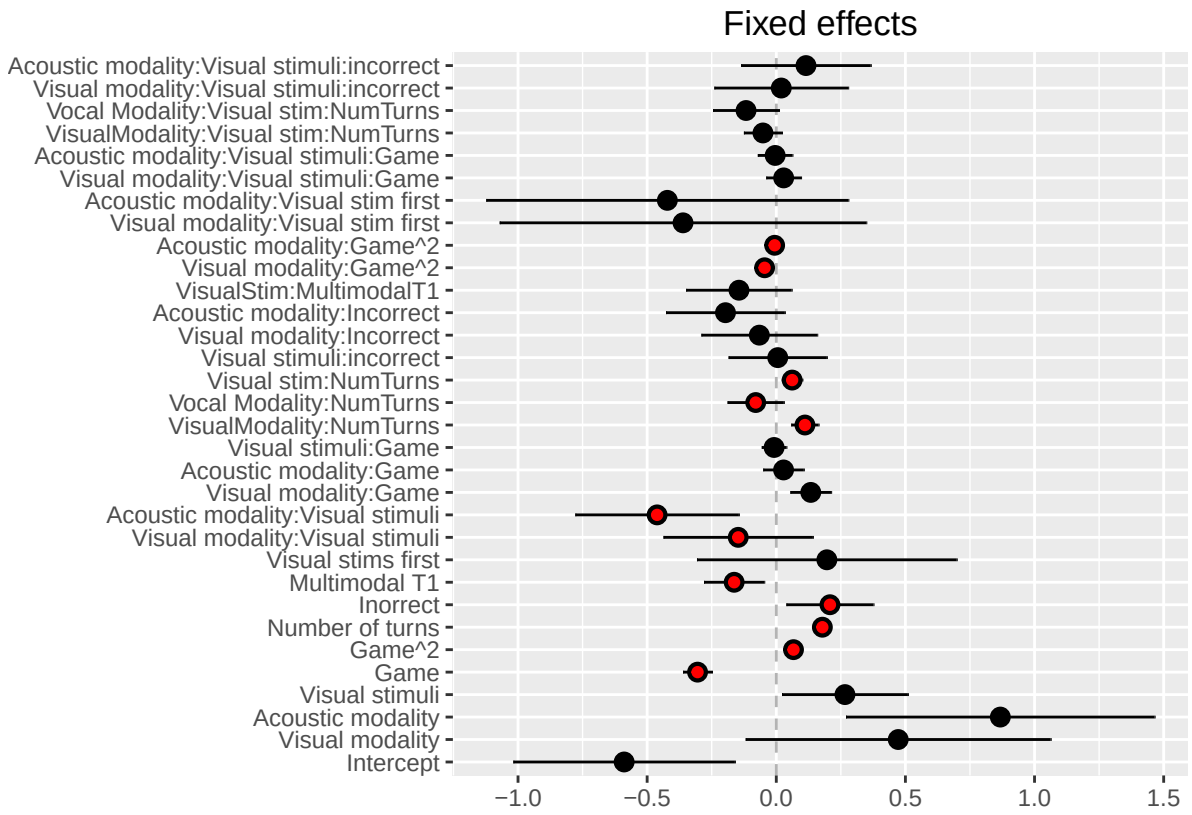
```
x = sjp.lmer(finalModel, 'fe',
         show.intercept = T,
         sort.est=NULL,
         axis.labels = feLabels2[2:length(feLabels2)],
         xlab="Trial time (ms)",
         geom.colors = c(1,1),
         show.p=F,
         show.values = F,
         p.kr = FALSE,
         string.interc="Intercept",
         prnt.plot = F)
```

```
## Warning: replacing previous import 'lme4::sigma' by 'stats::sigma' when
## loading 'pbkrtest'
```

```
## Computing p-values via Wald-statistics approximation (treating t as Wald z).
```

```
## Warning: Deprecated, use tibble::rownames_to_column() instead.
```

```
x$plot.list[[1]] +  geom_point(data=sig.data,aes(y=estimate,x=y,fade=fade), color=sig.data$pointCol)
```



Attempt plot with axes in milliseconds.

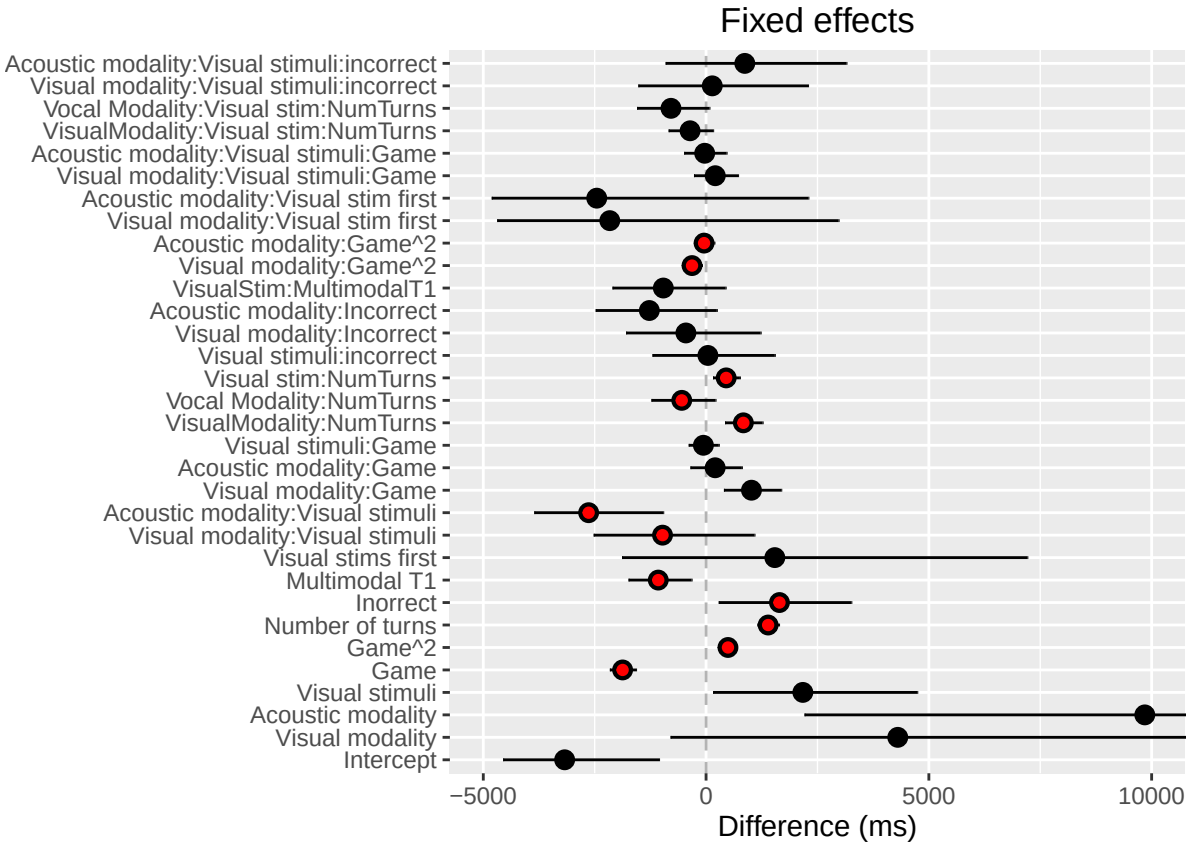
```
convertEst = function(X){
  exp(meanLogTrialLength+X) - exp(meanLogTrialLength)
}

x$plot.list[[1]]$data$estimate =convertEst(x$plot.list[[1]]$data$estimate)
x$plot.list[[1]]$data$conf.low = convertEst(x$plot.list[[1]]$data$conf.low)
x$plot.list[[1]]$data$conf.high =  convertEst(x$plot.list[[1]]$data$conf.high)

sig.data2 = sig.data
sig.data2$estimate = x$plot.list[[1]]$data$estimate

x$plot.list[[1]] +
  scale_y_continuous(name="Difference (ms)") +
  scale_x_discrete(labels=feLabels2) +
  geom_point(data=sig.data2,aes(y=estimate,x=y,fade=fade), color=sig.data$pointCol) +
  coord_flip(ylim=c(-5000,10000))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

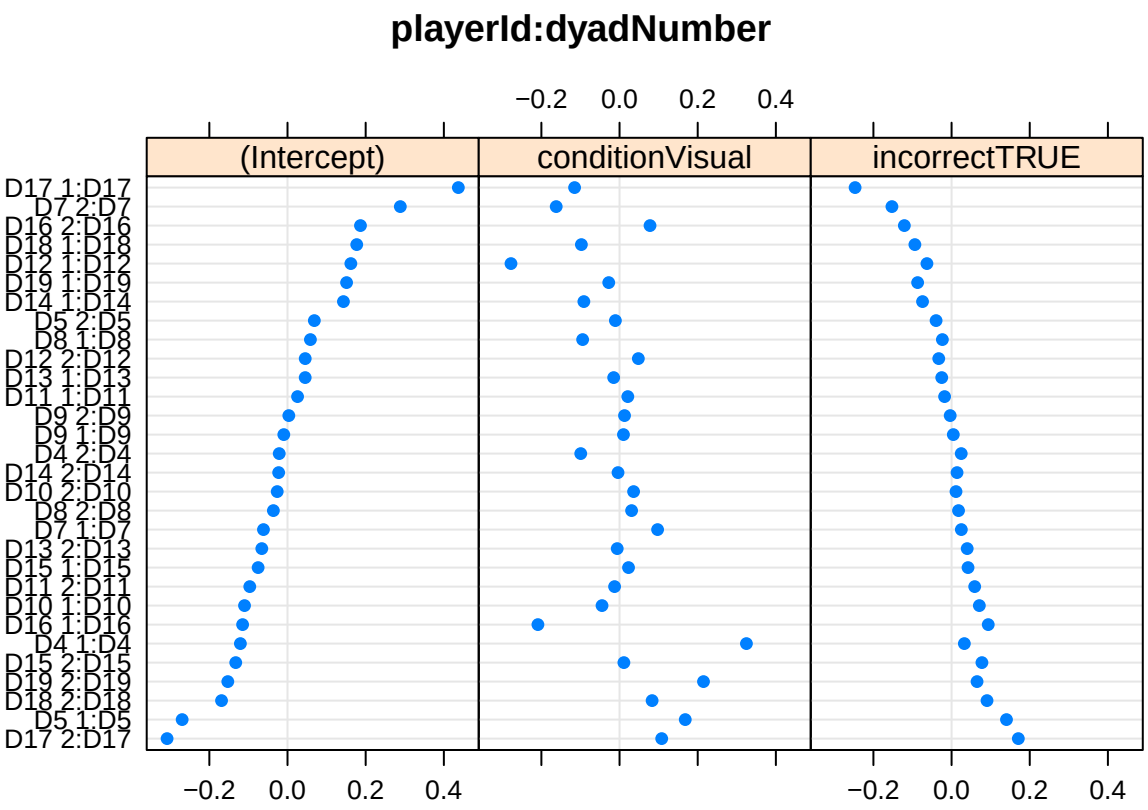


Random effects

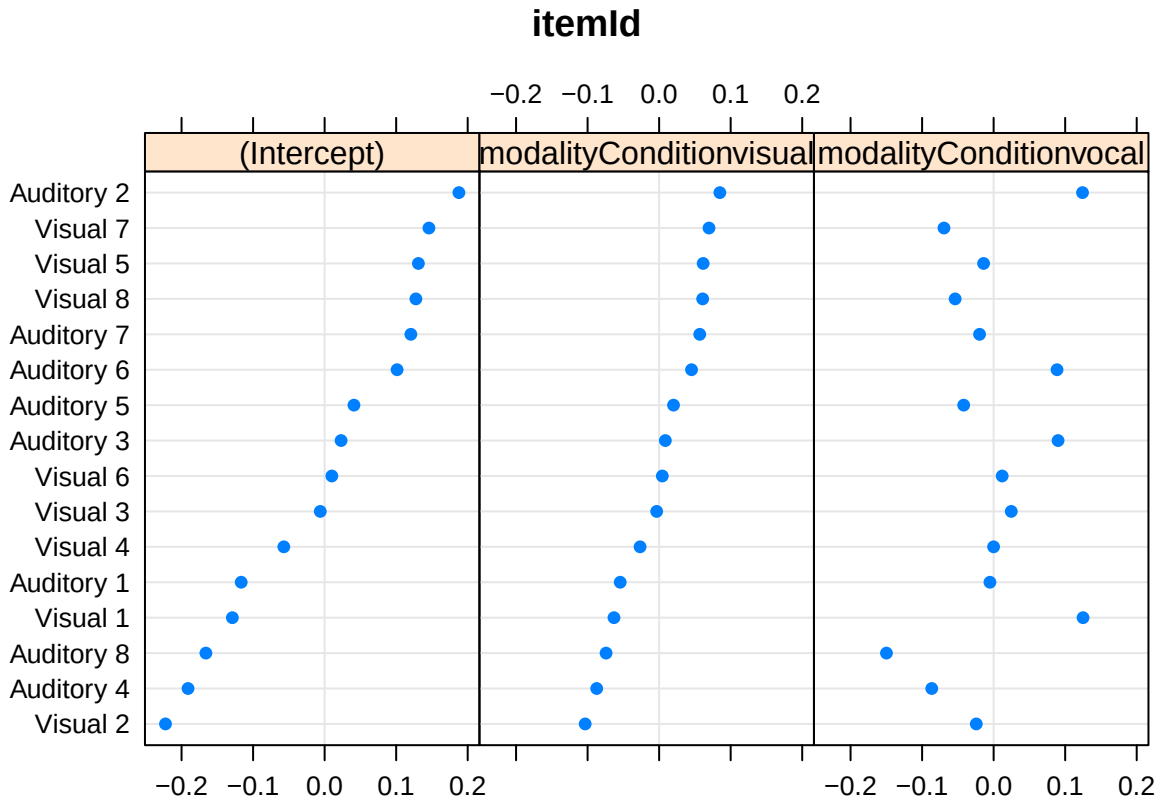
There is a reasonable amount of variaition in the random effects, suggesting that dyads and players differ. This justifies the use of mixed effects modelling.

```
dotplot(ranef(finalModel))
```

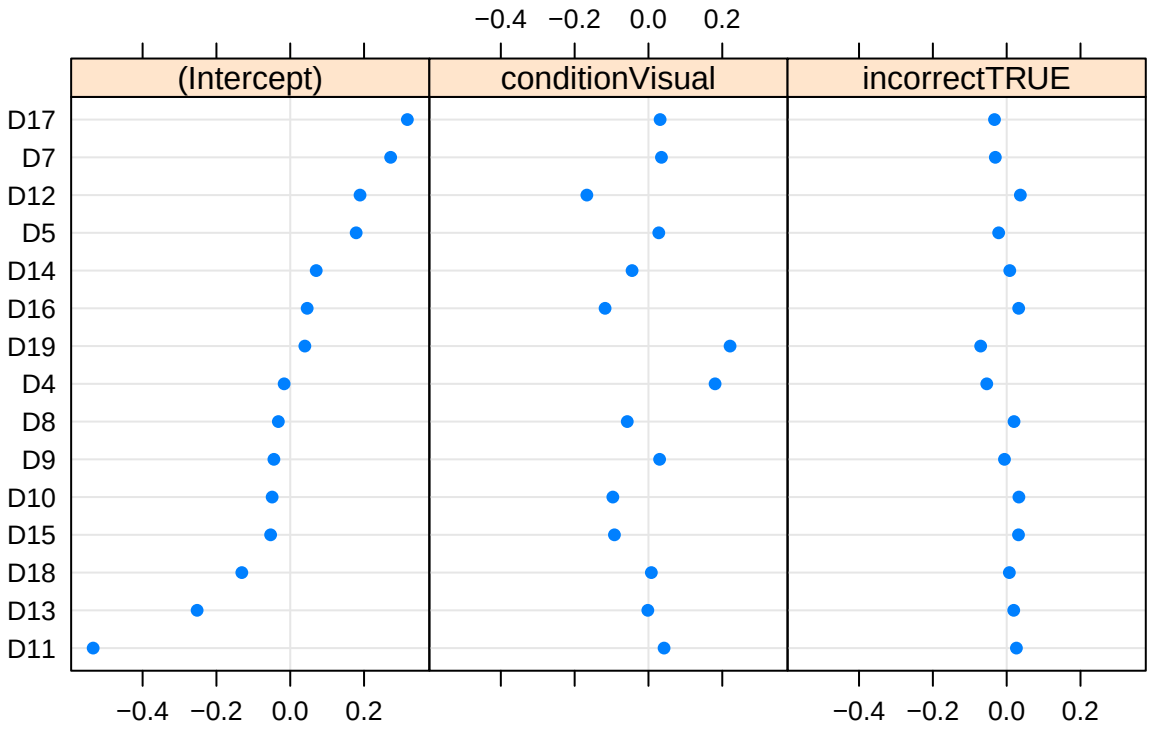
```
## $`playerId:dyadNumber`
```




```
##
## $itemId
```


itemId
(Intercept)
modalityConditionvisual
modalityConditionvocal
Auditory 2
Visual 7
Visual 5
Visual 8
Auditory 7
Auditory 6
Auditory 5
Auditory 3
Visual 6
Visual 3
Visual 4
Auditory 1
Visual 1
Auditory 8
Auditory 4
Visual 2
−0.2 −0.1 0.0 0.1 0.2

```
##
## $dyadNumber
```


dyadNumber
(Intercept)
conditionVisual
incorrectTRUE
D17
D7
D12
D5
D14
D16
D19
D4
D8
D9
D10
D15
D18
D13
D11
−0.4 −0.2 0.0 0.2

qq-plots of random effects

```
sjp.lmer(finalModel, type = "re.qq")
```

```
## Testing for normal distribution. Dots should be plotted along the line.
```