Modality effects in a signalling game: Accuracy

Intro

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

Load libraries

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

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$turnString, d$trialString,
                  function(X){length(unique(X))})
d$numberOfTurns = numTurns[d$trialString]

matcherResponds = tapply(d$turnType, d$trialString,
                         function(X){"T2" %in% X})
d$matcherResponds = matcherResponds[d$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 accuracy by conditions:

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
```

Make a variable for which stimuli the players experienced first.

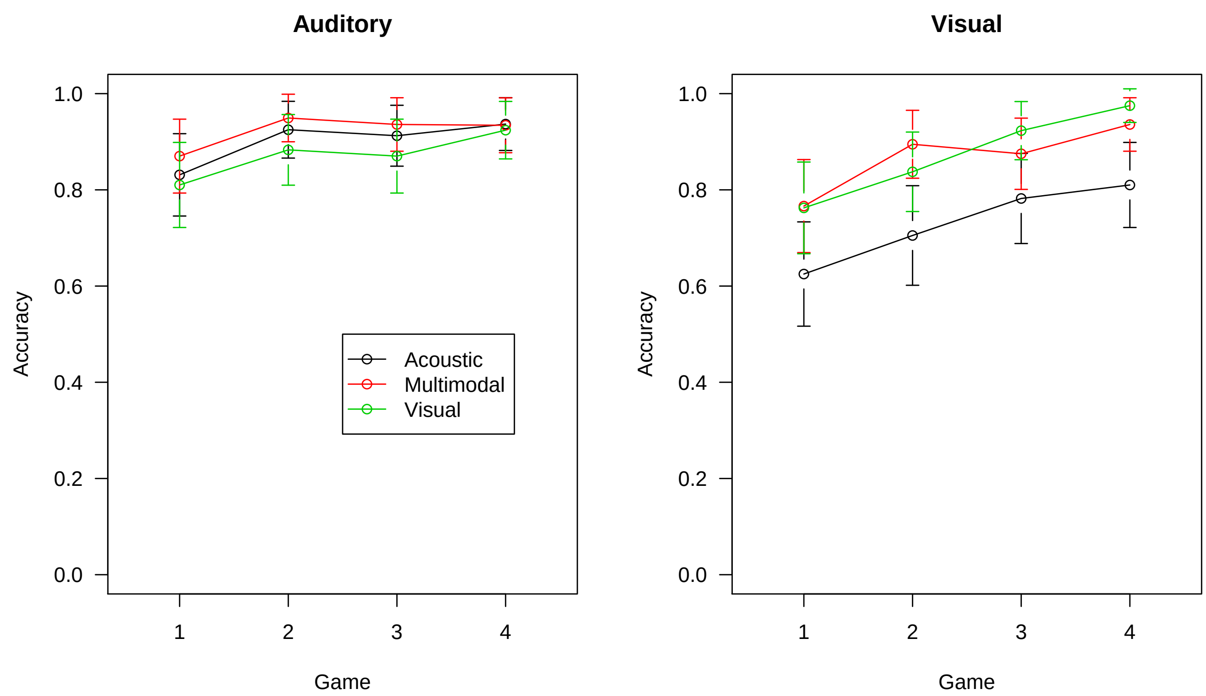


Figure 1: The efficiency of trials in different conditions

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

Variable to indicate whether T1 is multimodal.

```
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
```

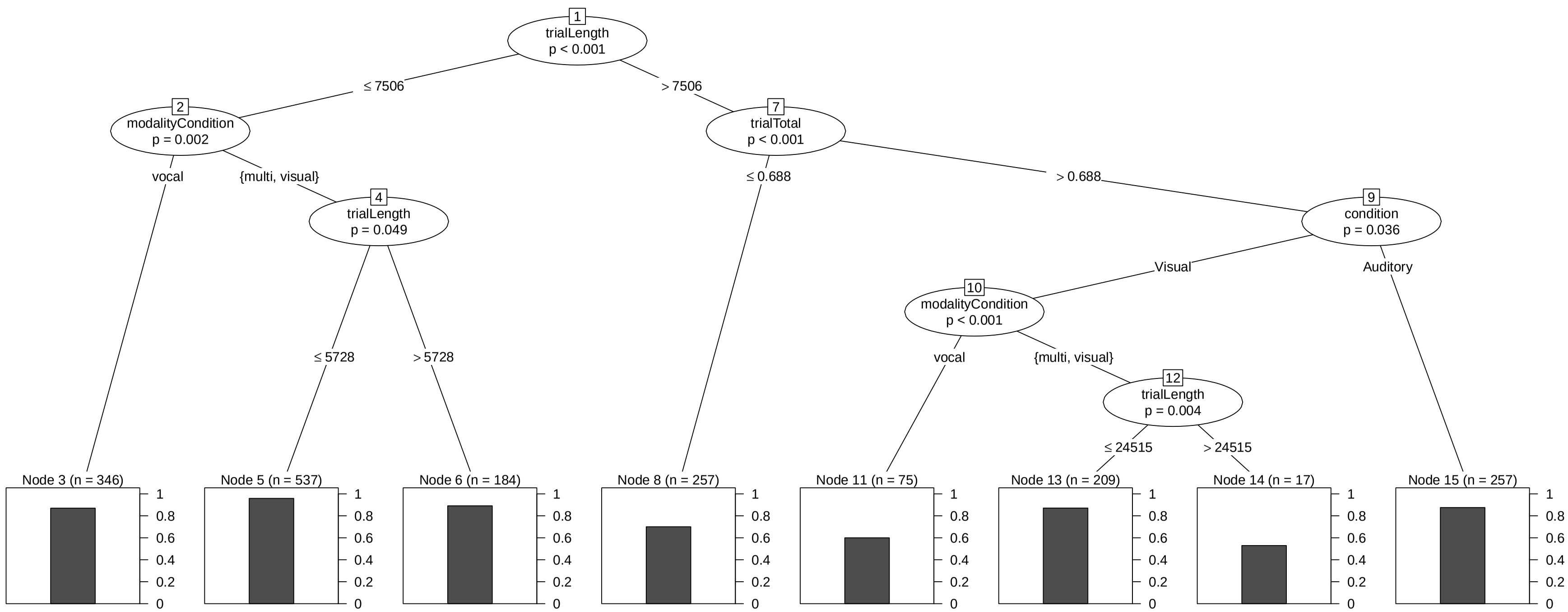
Transformed trial time.

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

Get an idea of the structure of the data from a binary tree:

```
cx = ctree(correct ~ modalityCondition + condition +
            trialTotal +
            trialLength +
            numberOfTurns +
            matcherResponds +
            multimodal+
            firstBlock,
          data=d)
```

```
plot(cx, terminal_panel=node_barplot(cx))
```



Mixed models

There are celing effects in the data, which reduces variance and makes model convergence difficult. Experimentation revealed that random effects other than random intercepts for dyad and item lead to non-convergence.

The final models do not converge within standard tolerances, but the convergence is acceptable.

```
# No fixed effects

gc = glmerControl(optimizer = "Nelder_Mead" ,optCtrl = list(maxfun=50000))

m0 = glmer(correct ~ 1 +
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)

mod = glmer(correct ~ 1 + modalityCondition +
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)

con = glmer(correct ~ 1 + modalityCondition + condition +
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)

modXcon = glmer(correct ~ 1 + modalityCondition * condition +
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)

game = glmer(correct ~ 1 + modalityCondition * condition +
            trialTotal +
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)

trialL = glmer(correct ~ 1 + modalityCondition * condition +
            trialTotal +
            trialLength.log+
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)

trialLXmo = glmer(correct ~ 1 + modalityCondition * condition +
            trialTotal +
            trialLength.log * modalityCondition+
            (1 |dyadNumber) +
            (1 |itemId) ,
```

```
          data=d, family=binomial,
          control = gc)

multi = glmer(correct ~ 1 + modalityCondition * condition +
            trialTotal +
            trialLength.log * modalityCondition+
            multimodal+
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge with max|grad| = 0.00228749 (tol =
## 0.001, component 1)
```

```
block = glmer(correct ~ 1 + modalityCondition * condition +
            trialTotal +
            trialLength.log * modalityCondition+
            multimodal+
            firstBlock +
            (1 |dyadNumber) +
            (1 |itemId) ,
          data=d, family=binomial,
          control = gc)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control
## $checkConv, : Model failed to converge with max|grad| = 0.00635234 (tol =
## 0.001, component 1)
```

Results

Compare the fit of the models:

```
modelComparison = anova(m0,mod,con,modXcon,
                        game, trialL,trialLXmo,
                        multi, block)
modelComparison
```

```
## Data: d
## Models:
## m0: correct ~ 1 + (1 | dyadNumber) + (1 | itemId)
## mod: correct ~ 1 + modalityCondition + (1 | dyadNumber) + (1 | itemId)
## con: correct ~ 1 + modalityCondition + condition + (1 | dyadNumber) +
## con:     (1 | itemId)
## modXcon: correct ~ 1 + modalityCondition * condition + (1 | dyadNumber) +
## modXcon:     (1 | itemId)
## game: correct ~ 1 + modalityCondition * condition + trialTotal + (1 |
## game:     dyadNumber) + (1 | itemId)
## trialL: correct ~ 1 + modalityCondition * condition + trialTotal + trialLength.log +
## trialL:     (1 | dyadNumber) + (1 | itemId)
## trialLXmo: correct ~ 1 + modalityCondition * condition + trialTotal + trialLength.log *
## trialLXmo:     modalityCondition + (1 | dyadNumber) + (1 | itemId)
## multi: correct ~ 1 + modalityCondition * condition + trialTotal + trialLength.log *
## multi:     modalityCondition + multimodal + (1 | dyadNumber) + (1 |
## multi:     itemId)
## block: correct ~ 1 + modalityCondition * condition + trialTotal + trialLength.log *
## block:     modalityCondition + multimodal + firstBlock + (1 | dyadNumber) +
## block:     (1 | itemId)
##           Df    AIC    BIC  logLik deviance   Chisq Chi Df Pr(>Chisq)
## m0         3 1405.1 1421.8 -699.56   1399.1
## mod        5 1404.5 1432.2 -697.25   1394.5  4.6377      2   0.098388 .
## con        6 1404.3 1437.5 -696.14   1392.3  2.2210      1   0.136149
## modXcon    8 1392.4 1436.7 -688.19   1376.4 15.8923      2   0.000354 ***
## game       9 1342.8 1392.6 -662.39   1324.8 51.5987      1  6.809e-13 ***
## trialL    10 1302.5 1357.9 -641.23   1282.5 42.3102      1  7.789e-11 ***
## trialLXmo 12 1302.5 1369.0 -639.26   1278.5  3.9397      2   0.139480
## multi     13 1304.5 1376.5 -639.24   1278.5  0.0442      1   0.833472
## block     14 1306.4 1384.0 -639.21   1278.4  0.0710      1   0.789896
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pick final model for estimates:

```
finalModel = block
```

Plot the fixed effects

Relabel the effects:

```
feLabels = matrix(c(
"(Intercept)"              ,"Intercept"      , NA,
"modalityConditionvisual" ,"Visual modality", "mod",
"modalityConditionvocal"  , "Acoustic modality", "mod",
"conditionVisual" , "Visual stimuli","con",
```

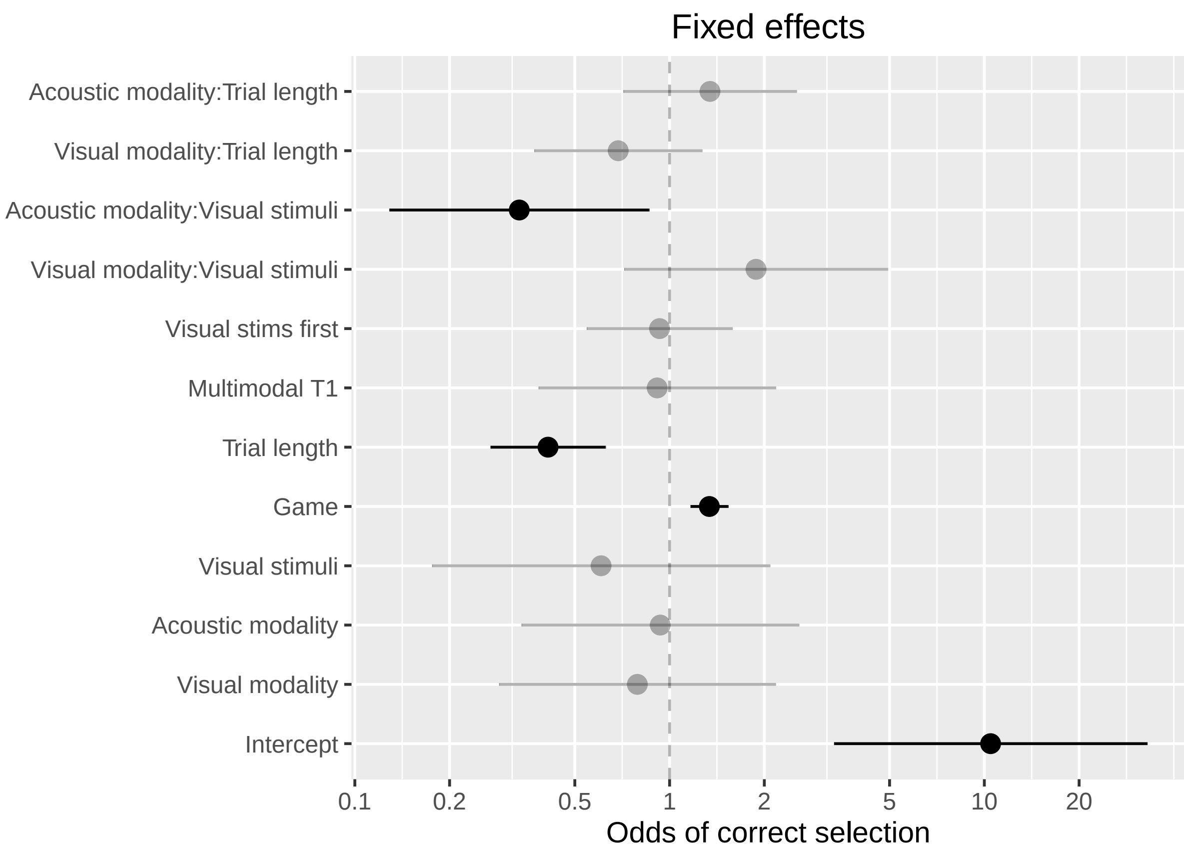
```
"trialTotal"              , "Game","game",
"modalityConditionvisual:conditionVisual"  , "Visual modality:Visual stimuli", "modXcon",
"modalityConditionvocal:conditionVisual" , "Acoustic modality:Visual stimuli","modXcon",
"firstBlockVisual","Visual stims first","block",
"trialLength.log","Trial length","trialL",
"modalityConditionvisual:trialLength.log","Visual modality:Trial length",'trialLXmo',
"modalityConditionvocal:trialLength.log","Acoustic modality:Trial length",'trialLXmo',
"multimodalTRUE","Multimodal T1","multi",
"trialLength.log",'Trial Length','trialL'
), ncol=3, byrow = T)

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

Plot the strength of the fixed effects:

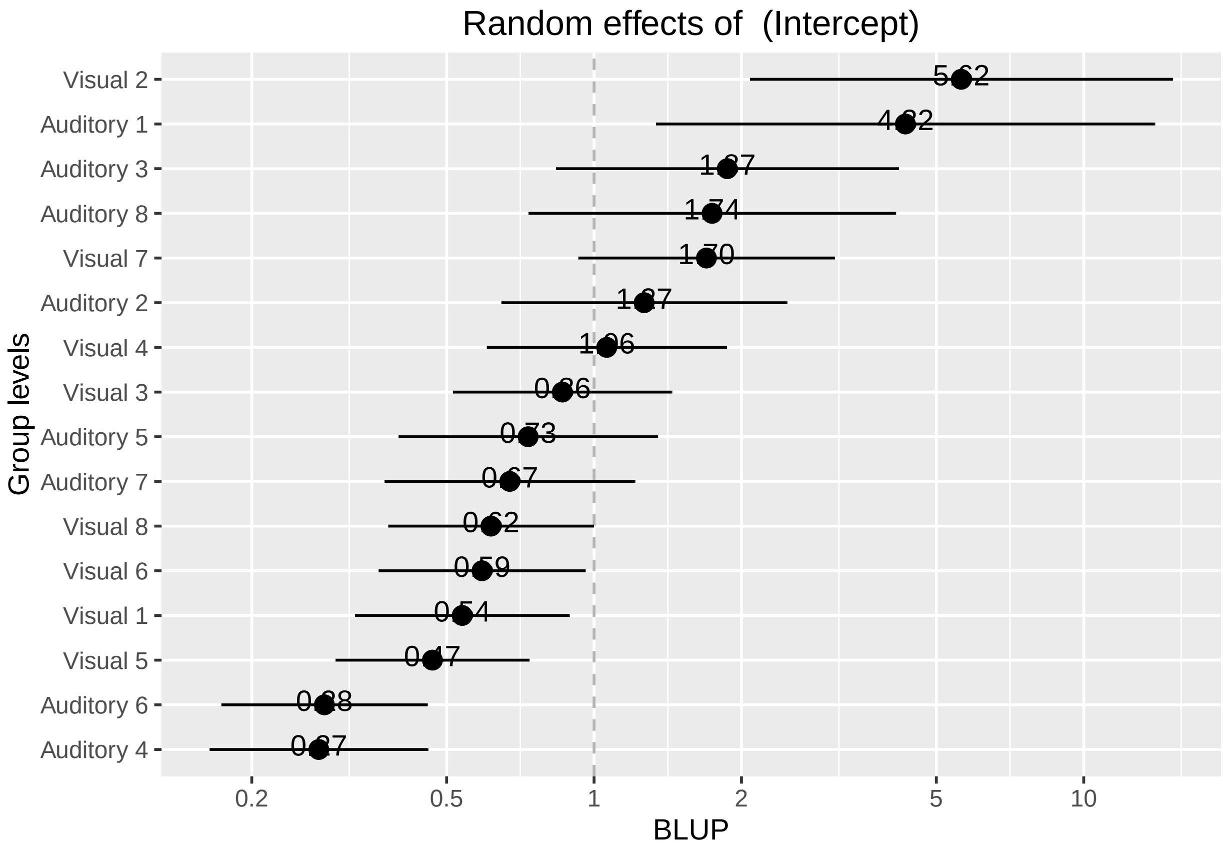
```
sjp.glmer(finalModel, 'fe',
         show.intercept = T,
         sort.est=NULL,
         axis.labels = feLabels2[2:length(feLabels2)],
         axis.title="Odds of correct selection",
         geom.colors = c(1,1),
         show.values = F,
         show.p = F,
         fade.ns = T,
         string.interc="Intercept")
```

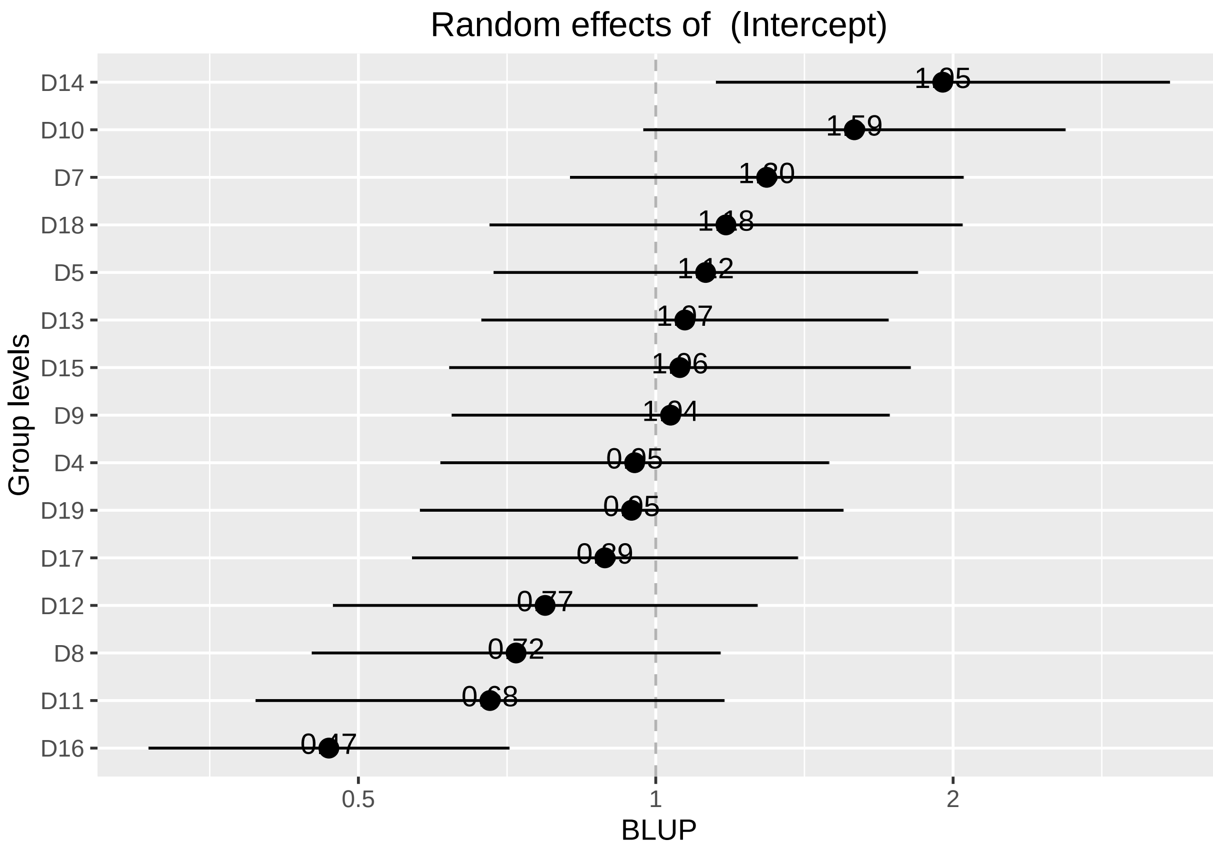
```
## Warning: Deprecated, use tibble::rownames_to_column() instead.
```



Random effects

```
sjp.glmer(finalModel,'re', sort.est ="sort.all",
          facet.grid = F,
          geom.colors= c(1,1))
```

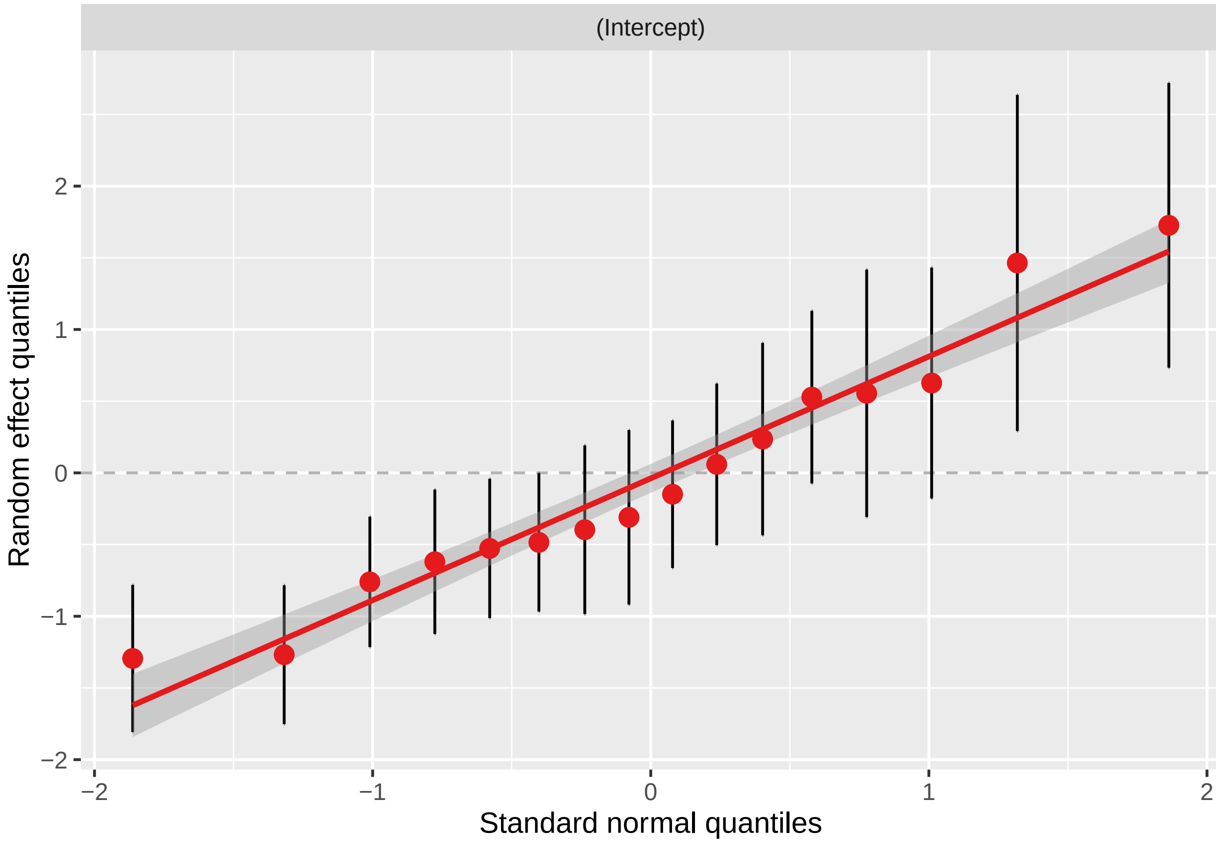




qq-plots of random effects

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

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


(Intercept)
Random effect quantiles
2
1
0
−1
−2
−2
−1
0
1
2
Standard normal quantiles