# Analysis of multimodal condition

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

## Load libraries

```r
library(party)
library(lme4)
library(sjPlot)
```

## Load data

```r
d = read.csv("../../data/Final_Turn_data.csv", stringsAsFactors = F)
d = d[d$modalityCondition == "multi",]
```

## Prepare variables

```r
# Relabel modalities
d[d$turnModalityType=="multi",]$turnModalityType = "M"
d[d$turnModalityType=="unimodal acoustic",]$turnModalityType = "A"
d[d$turnModalityType=="unimodal visual",]$turnModalityType = "V"

# Only need one record per trial
d2 = d[!duplicated(d$trialString),]

# get turn modality type for T1
x = tapply(d[d$turnType=="T1",]$turnModalityType, d[d$turnType=="T1",]$trialString,head,n=1)
d2$turnModality.T1 = x[d2$trialString]

# remove NAs
d2 = d2[!is.na(d2$turnModality.T1),]
# relevel
d2$turnModality.T1 = relevel(factor(as.character(d2$turnModality.T1)),"V")

# get turn modality type for T2
x = tapply(d[d$turnType=="T2",]$turnModalityType, d[d$turnType=="T2",]$trialString,head,n=1)
d2$turnModality.T2 = x[d2$trialString]
d2$turnModality.T2[is.na(d2$turnModality.T2)] = "n"
d2$turnModality.T2 = relevel(factor(d2$turnModality.T2),'n')

# get turn modality type for T3
x = tapply(d[d$turnType=="T3",]$turnModalityType, d[d$turnType=="T3",]$trialString,head,n=1)
d2$turnModality.T3 = x[d2$trialString]
d2$turnModality.T3[is.na(d2$turnModality.T3)] = "n"
d2$turnModality.T3 = relevel(factor(d2$turnModality.T3),'n')
```

Make game variable.

```r
d2$trialTotal = d2$trial + (d2$game * (max(d2$trial)+1))
# Convert to proportion of games played, so that estimates reflect change per game.
d2$trialTotal = d2$trialTotal / 16
# Center the trialTotal variable so intercept reflects after the first game
d2$trialTotal = d2$trialTotal - 1

d2$incorrect = !d2$correct
```

Scale trial length variable.

```r
d2$trialLength.logcenter = log(d2$trialLength)
d2$trialLength.logcenter = d2$trialLength.logcenter - mean(d2$trialLength.logcenter)
```

## Simple mixed effects model

```r
m0 = lmer(trialLength.logcenter ~
            condition*trialTotal +
            I(trialTotal^2) +
            (1 | dyadNumber) +
            (1 | itemId),
          data = d2)

m1 = lmer(trialLength.logcenter ~
            condition*trialTotal +
            I(trialTotal^2) +
            turnModality.T1 +
            (1 | dyadNumber) +
            (1 | itemId),
          data = d2)

m2 = lmer(trialLength.logcenter ~
            condition*trialTotal +
            I(trialTotal^2) +
            turnModality.T1*condition +
            (1 | dyadNumber) +
            (1 | itemId),
          data = d2)
```

```
## fixed-effect model matrix is rank deficient so dropping 1 column / coefficient
```

```r
anova(m0,m1,m2)
```

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

```
## Data: d2
## Models:
## m0: trialLength.logcenter ~ condition * trialTotal + I(trialTotal^2) +
## m0:     (1 | dyadNumber) + (1 | itemId)
## m1: trialLength.logcenter ~ condition * trialTotal + I(trialTotal^2) +
## m1:     turnModality.T1 + (1 | dyadNumber) + (1 | itemId)
## m2: trialLength.logcenter ~ condition * trialTotal + I(trialTotal^2) +
## m2:     turnModality.T1 * condition + (1 | dyadNumber) + (1 | itemId)
##    Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## m0  8 944.90 980.35 -464.45   928.90
## m1 10 948.08 992.39 -464.04   928.08 0.8212      2     0.6633
## m2 11 950.02 998.77 -464.01   928.02 0.0549      1     0.8147
```

There was no significant main effect of T1 signal modality ( log likelihood difference = 0.41 , df = 2 , Chi Squared = 0.82 , p = 0.66 ).

There was no significant interaction between T1 signal modality and condition ( log likelihood difference = 0.027 , df = 1 , Chi Squared = 0.05 , p = 0.81 ).

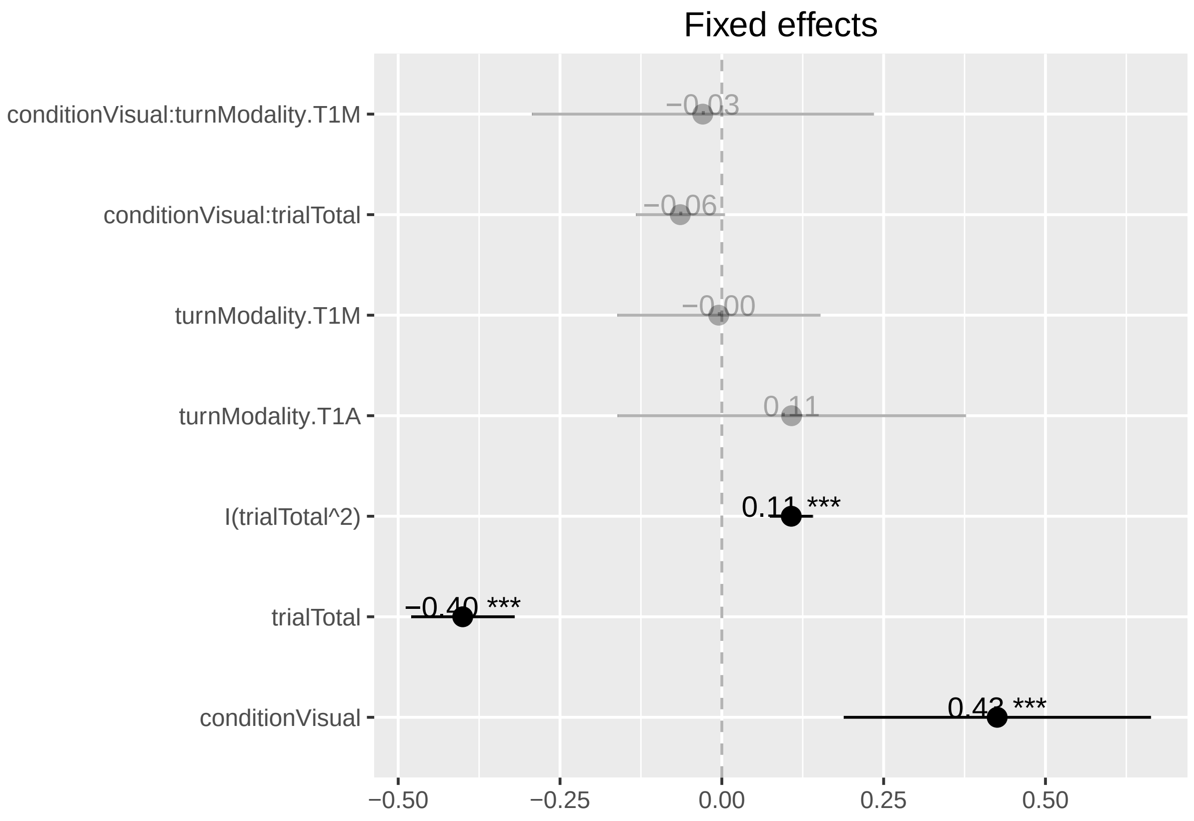
```
sjp.lmer(m2, 'fe',
         geom.colors = c(1,1),
         fade.ns = T)
```

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

```
## Computing p-values via Kenward-Roger approximation. Use `p.kr = FALSE` if computation takes too long
```

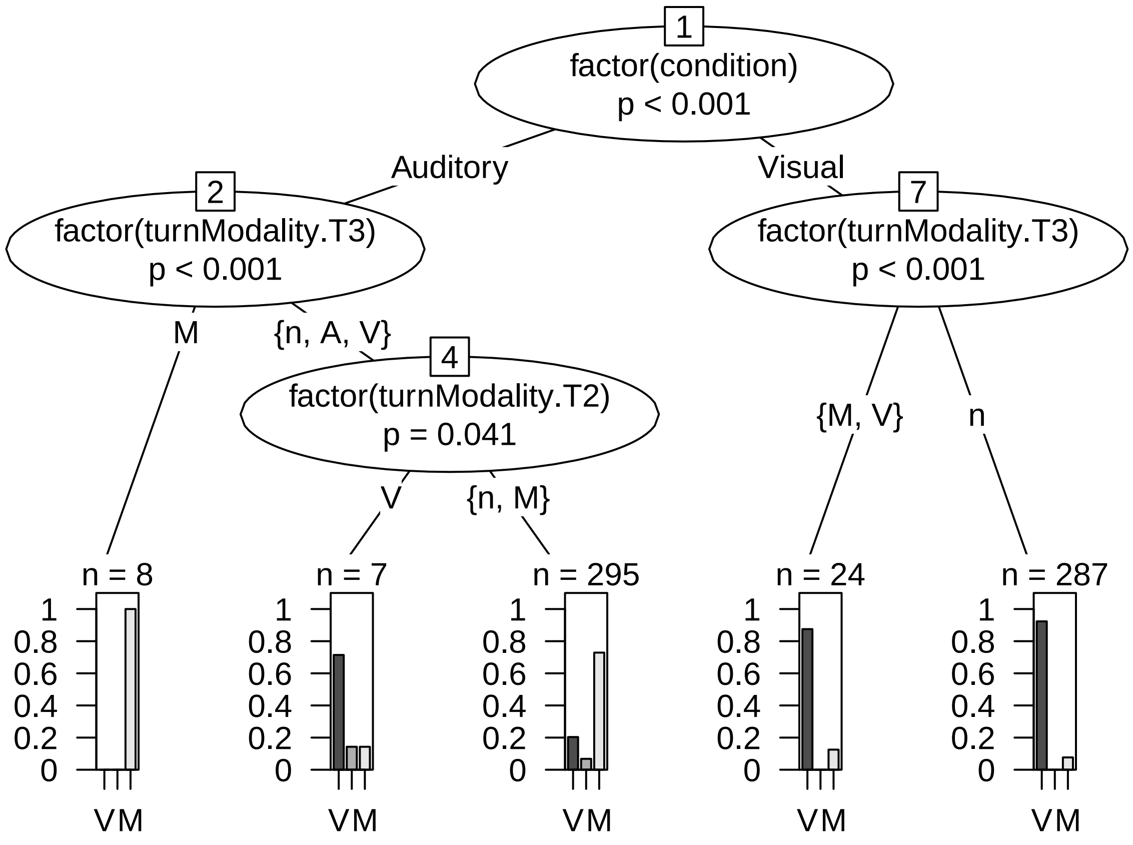
```
## Warning in deviance.merMod(object, ...): deviance() is deprecated for REML
## fits; use REMLcrit for the REML criterion or deviance(.,REML=FALSE) for
## deviance calculated at the REML fit
```

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

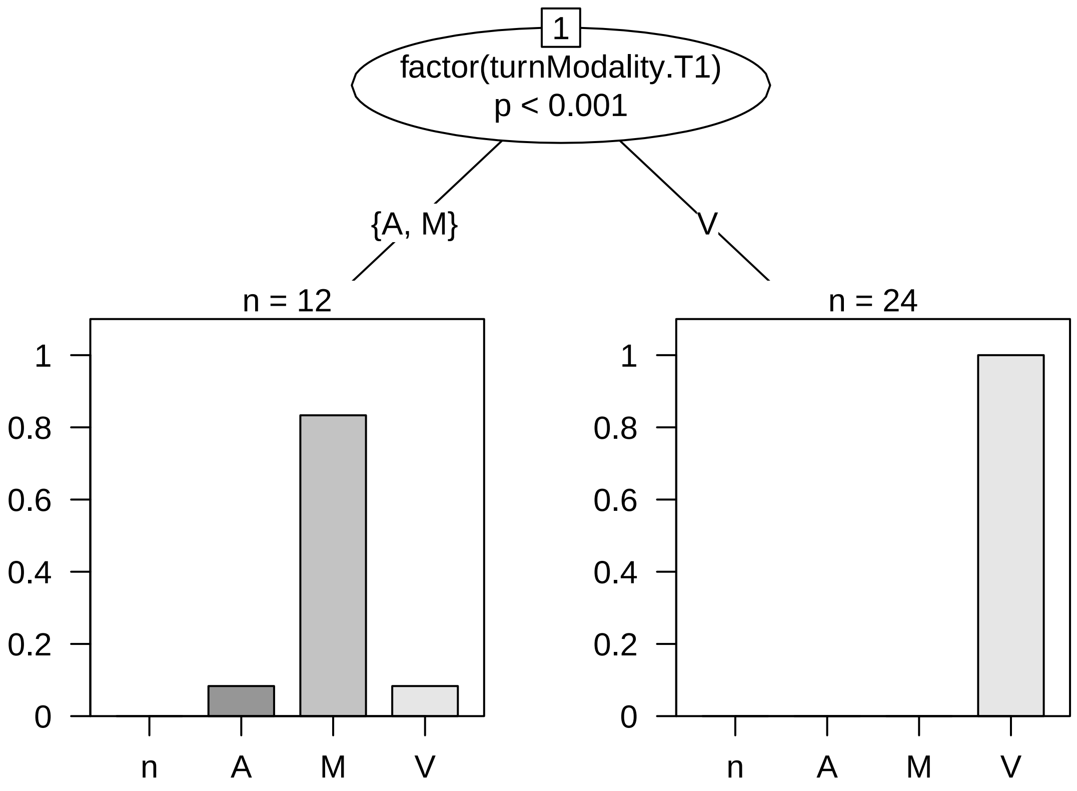


## Binary trees

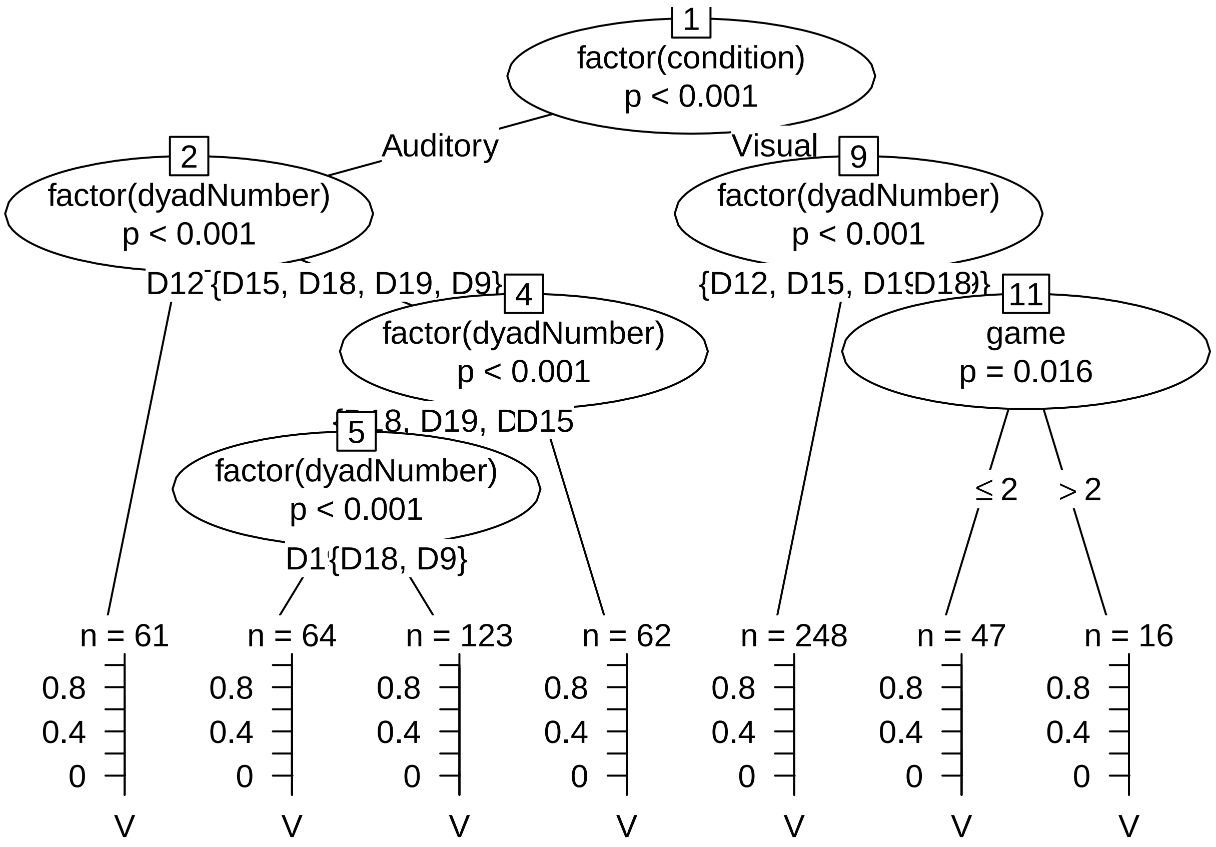
```
cx = ctree(turnModality.T1~
              factor(turnModality.T2)+
              factor(turnModality.T3) +
              factor(condition) +
              game
            , data=d2)
plot(cx, terminal_panel = node_barplot(cx, id=F))
```



```r
cxT3 = ctree(turnModality.T3~
             factor(turnModality.T2)+
             factor(turnModality.T1) +
             factor(condition) +
             game
           , data=d2[d2$turnModality.T3!="n",])
plot(cxT3, terminal_panel = node_barplot(cxT3, id=F))
```



```
cx2 = ctree(turnModality.T1~
             factor(turnModality.T2)+
             factor(turnModality.T3) +
             factor(condition) +
             game +
             factor(dyadNumber)
           , data=d2)
plot(cx2, terminal_panel = node_barplot(cx2, id=F))
```



Graphs are also written to results/graphs/cTree/

```
## pdf
##   2
```

```
## pdf
##   2
```

```
## pdf
##   2
```