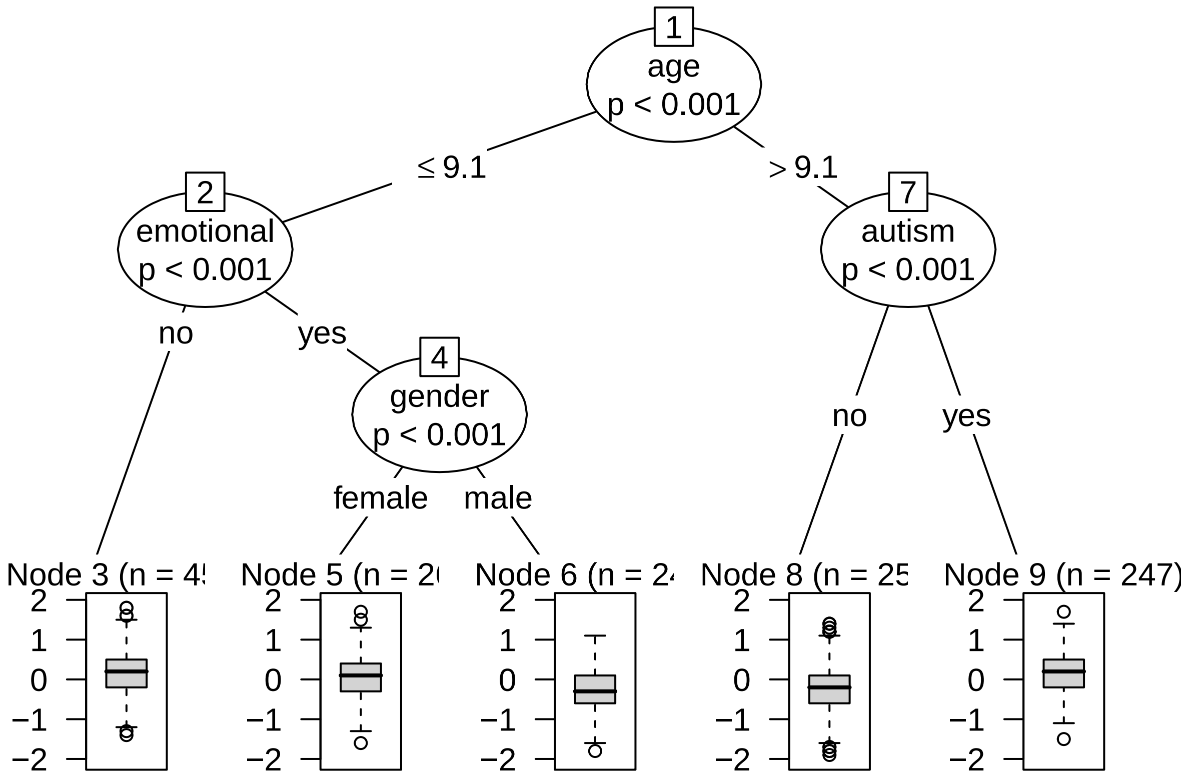
Fit glmertree and calculate k-fold cross-validated prediction error

Load package and data:

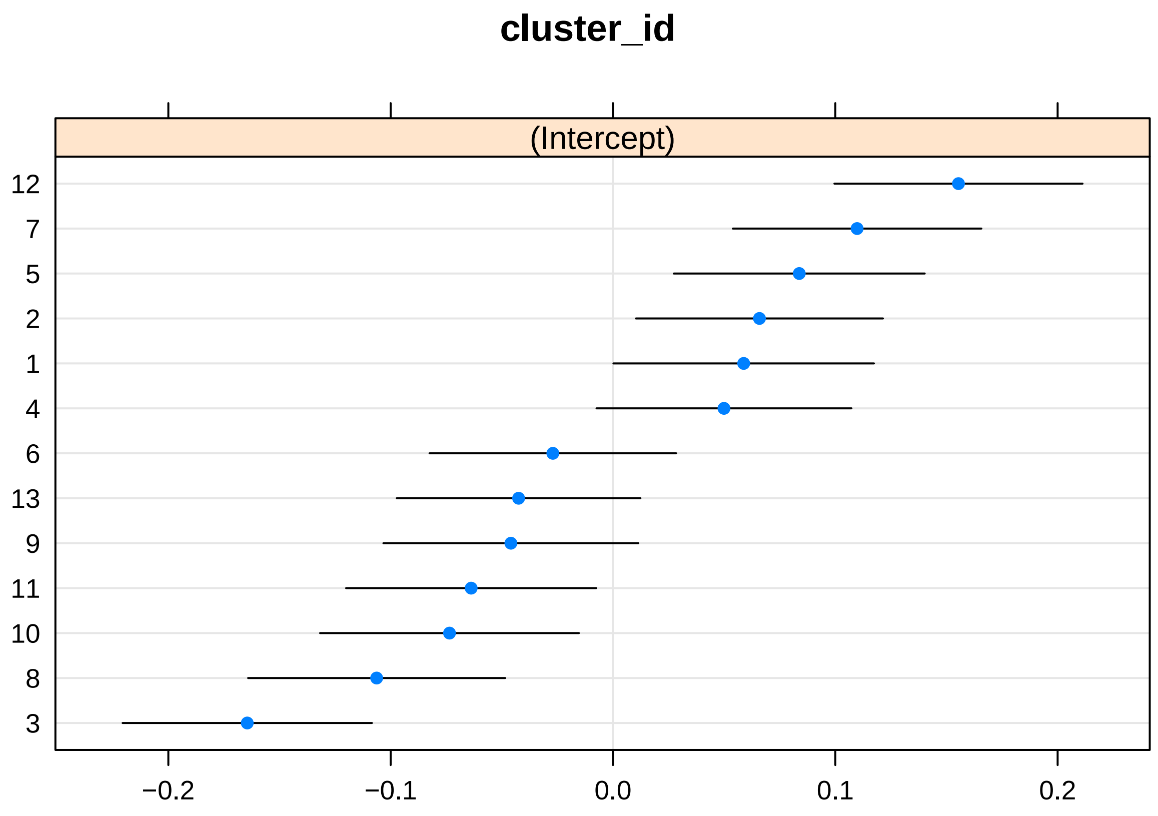
```
library("glmertree")
data <- read.table("UK_MH mimic data.txt")
```

Fit tree on full data:

```
formula <- outcome ~ 1 | cluster_id | age + gender + emotional +
                         autism + impact + conduct
full_tree <- lmertree(formula = formula, data = data)
plot(full_tree)
```



```
## $cluster_id
```



The tree based on the full data would be the main result of the analysis. However, often we want to get an estimate of how accurate our predictions would be, if we were to apply this model (tree) to new observations. A good way to do this is with k-fold cross validation, where setting $k = 10$ should provide reasonable (not too optimistic, not too pessimistic) estimates.

Calculate CV error using observation-level sampling

Divide dataset k random subsamples:

```
## Set up fold indicators:
k <- 10
set.seed(42)
fold_ids <- sample(rep(1:k, length = nrow(data)))
## Set up vector to save CV predictions:
cv_preds <- numeric(length = nrow(data))
## Perform the CV:
for(i in 1:k) {
  test_ids <- which(fold_ids == i)
  train_ids <- which(fold_ids != i)
  tree <- lmertree(formula = formula, data = data[train_ids, ])
  cv_preds[test_ids] <- predict(tree, newdata = data[test_ids, ],
                                re.form = NULL)
}
```

Then we can calculate mean squared error or correlation between predicted and observed values:

```r
mean((cv_preds - data$outcome)^2) # computes MSE
```

```
## [1] 0.2633459
```

```r
var(data$outcome) # as reference to compare MSE with
```

```
## [1] 0.3001537
```

```r
cor(cv_preds, data$outcome)
```

```
## [1] 0.3498842
```

Calculate CV error using cluster-level sampling

Above, I sampled observations, not clusters (sites, in your dataset). If you have $> k$ sites, you could also sample sites for the cross validation. Then code would be as follows:

```r
## Set up fold indicators:
k <- 10
data$cluster_id <- factor(data$cluster_id)
set.seed(43)
fold_id <- sample(rep(1:k, length = nlevels(data$cluster_id)))
fold_ids <- list()
for (i in 1:k) {
  fold_ids[[i]] <- levels(data$cluster)[which(fold_id == i)]
}
## Set up vector to save predictions:
cv_preds <- numeric(length = nrow(data))
## Perform the CV:
for (i in 1:k) {
  test_ids <- which(data$cluster_id %in% fold_ids[[i]])
  train_ids <- which(!data$cluster_id %in% fold_ids[[i]])
  tree <- lmertree(formula = formula, data = data[train_ids, ])
  cv_preds[test_ids] <- predict(tree, newdata = data[test_ids, ],
                                re.form = NA)
}
```

Note that earlier, I used `re.form = NULL` in the `predict()` function. That means that random effects are accounted for in the predictions. If you have many clusters (sites) and at least some of them are quite small, this could yield an error. Then you can have one or morefolds where all observations in a given cluster appearing only in the test, and not in the training data. Then the random effects were not estimated for that cluster (site) and predictions cannot be calculated, yielding an error.

In the cluster-level sampling CV I used `re.form = NA`, so random effects are not included in the predictions.

If the random effects are substantial (i.e., the intraclass correlation is substantial), I expect the observation-level sampling to indicate better predictive accuracy than cluster-level sampling. That is what we see here, too:

```r
mean((cv_preds - data$outcome)^2) # computes MSE
```

```
## [1] 0.2741758
```

```r
var(data$outcome) # as reference to compare MSE with
```

```
## [1] 0.3001537
```

```r
cor(cv_preds, data$outcome)
```

```
## [1] 0.2946909
```

I do not think there is a right or wrong approach. Some researchers / statisticians / reviewers may prefer observation-level sampling, others cluster-level sampling approach.