Longitudinal Trees

Decision Trees and other statistical learning are mainly designed for the analysis of data collected at one time point. Although this dominates most books on statistical learning and programs written in languages like R, there are many resources available. We will cover a number of packages in R specifically created for longitudinal data.

Overview paper:

http://arxiv.org/pdf/1209.4690.pdf

These Include:

longRPart package. Not currently maintained on CRAN. Can be accessed from: http://cran.r-project.org/src/contrib/Archive/longRPart/

First Download the 1.0 tar.gz

and use: install.packages("~/Documents/Github/ATI_Labs/longRPart_1.0.tar.gz", repos = NULL, type = "source")

```r
library(longRPart)
library(REEMtree)
library(semtree)
```

For this demonstration, we are going to compare the different packages and functions in analyzing longitudinal WISC data. Data is WISC4VPE.DAT.

```r
wisc <- read.table("/Users/RJacobucci/Documents/Github/ATI_Labs/wisc4vpe.dat")
names(wisc)<- c("V1","V2","V4","V6","P1","P2","P4", "P6", "Moeducat")
# note: V1 refers to verbal scores at grade 1, P is performance
```

Visualization

To use many of the packages in R for longitudinal data, it is many times required to create a "long" data file, instead of the default wide.

How to do:

```r
# get rid of performance variables
wisc.verb <- wisc[,c(1:4,9)]

# create subset for plotting
ntot <- nrow(wisc.verb)    # total number of observations
wisc.verb.sel <- wisc.verb[sample(ntot, 30), ]



wisc.long <- reshape(wisc.verb, varying = c("V1", "V2", "V4", "V6"), v.names = "verbal",
                times = c(1, 2, 4, 6), direction = "long")
wisc.long.sel <- reshape(wisc.verb.sel, varying = c("V1", "V2", "V4", "V6"),
                         v.names = "verbal", times = c(1, 2, 4, 6), direction = "long")
head(wisc.long)
```

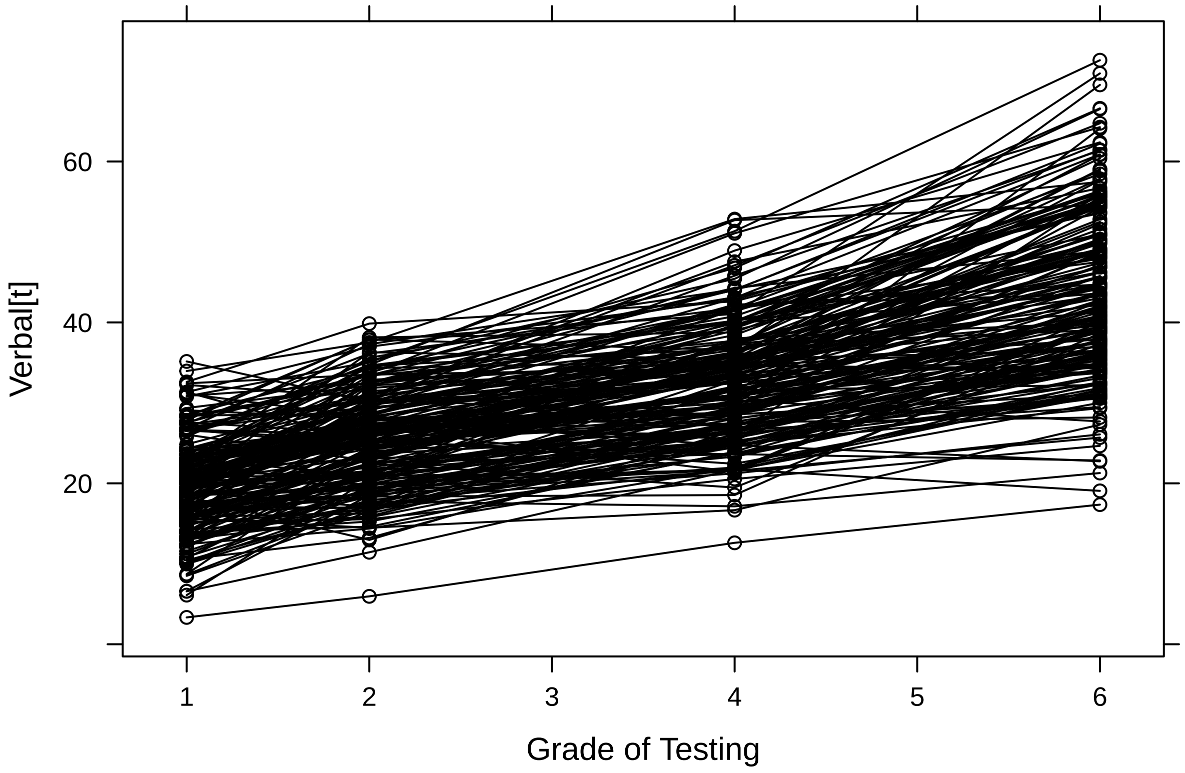
```
##     Moeducat time   verbal id
## 1.1        0    1 24.41964  1
## 2.1        0    1 12.44048  2
## 3.1        2    1 32.42560  3
## 4.1        2    1 22.69345  4
## 5.1        1    1 28.22917  5
## 6.1        2    1 16.05655  6
```

```
names(wisc.long)[2] <- "grade"
names(wisc.long.sel)[2] <- "grade"
```

Lets take a look at what the trajectories are:

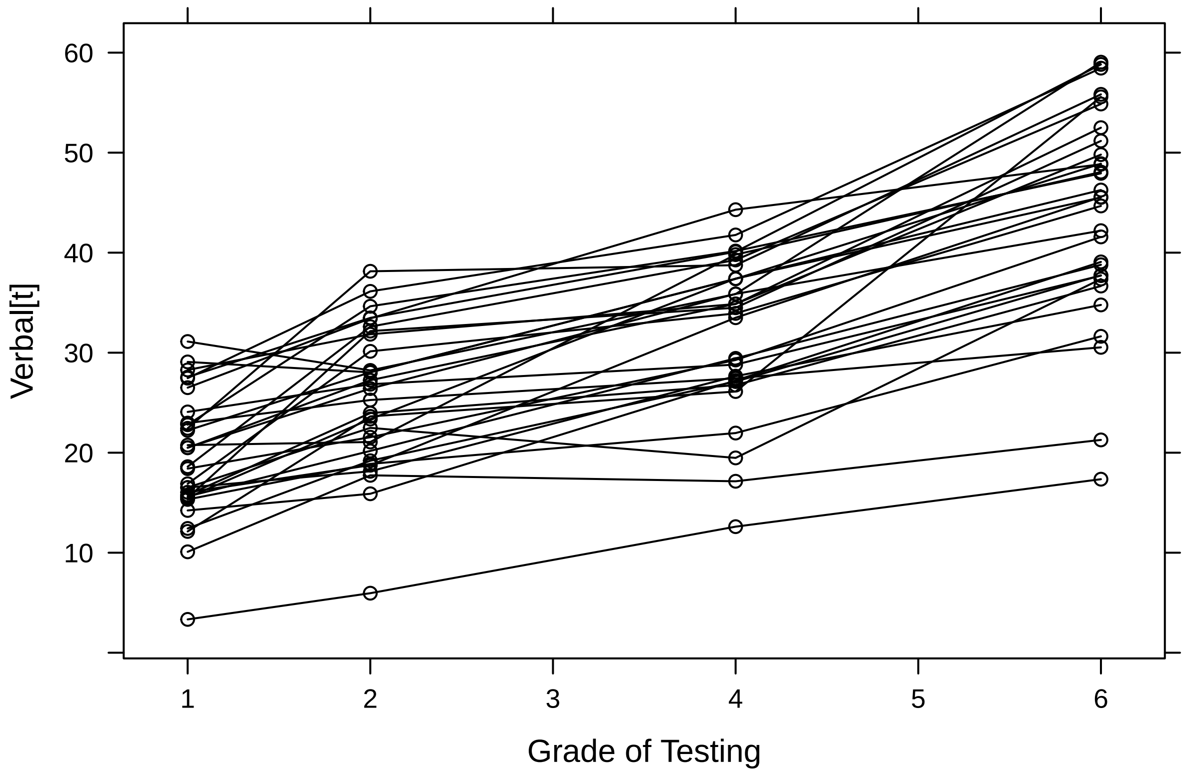
First using lattice package

```
library(lattice)
xyplot(verbal ~ grade, groups = id, data = wisc.long, type = "o", col = "black",
       xlab = "Grade of Testing", ylab = "Verbal[t]")
```



This is hard to see, better to use subset

```
xyplot(verbal ~ grade, groups = id, data = wisc.long.sel, type = "o",
       col = "black", xlab = "Grade of Testing", ylab = "Verbal[t]")
```

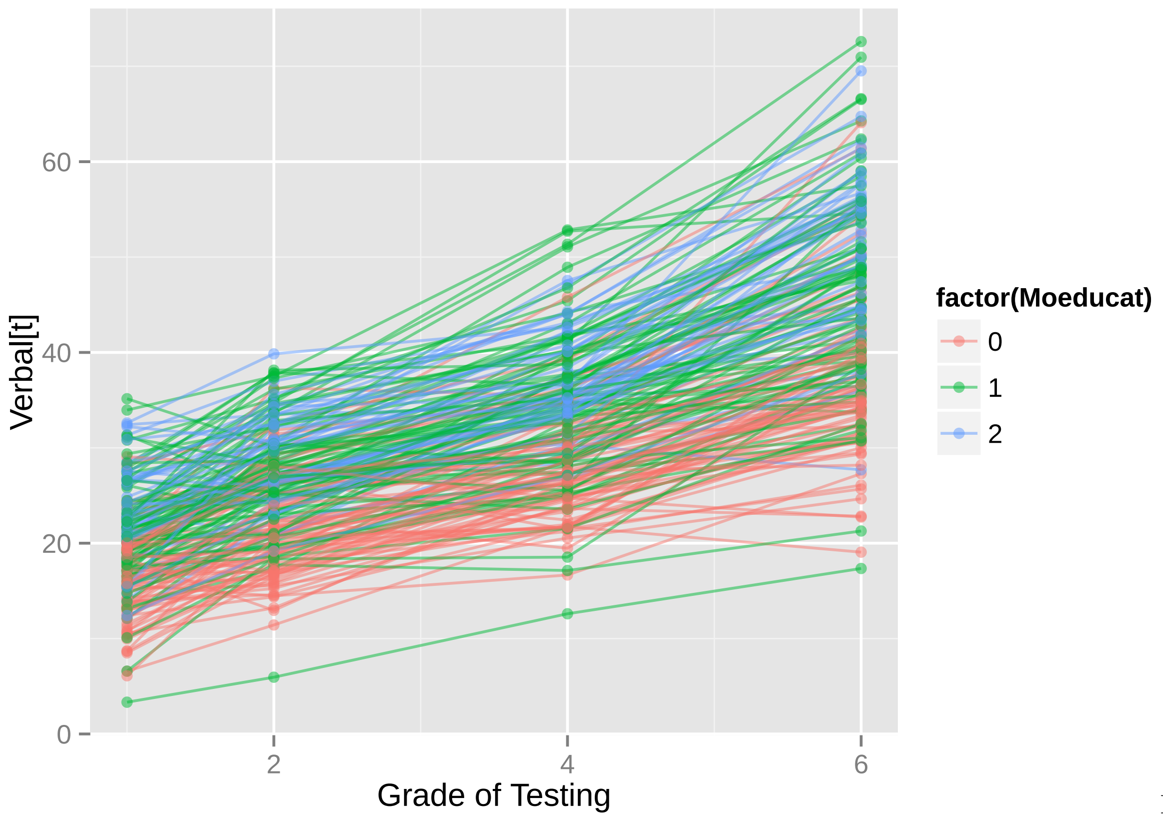


```
# on average, scores went up over time
```

Thats a little better.

But, can we simultaneously view the trajectories over time while seeing the influence that Mother's education may have?

```
library(ggplot2)
qplot(grade,verbal,group=id,data=wisc.long,alpha=I(1/2),colour=factor(Moeducat),
      geom = c("line","point"),xlab = "Grade of Testing", ylab = "Verbal[t]")
```



It seems pretty clear that mothers with higher levels of education have children that are consistently higher in verbal performance across time.

So now that we have an idea what will we find if we look at the relationship between mother's education and trajectory, lets test it with statistical models

SEM Trees LGCM

Our first example is going to be using a latent growth curve model (lgcm) as our outcome, and attempting to find subgroups based on mother's education and the performance scores

Previous demonstrations using SEM Trees have used OpenMx. In this case, we will use lavaan.

```
library(lavaan)

linearGCM <- '
    inter =~ 1*V1 + 1*V2 + 1*V4 + 1*V6
    slope =~ 1*V1 + 2*V2 + 4*V4 + 6*V6
    inter ~~ vari*inter; inter ~ meani*1;
    slope ~~ vars*slope; slope ~ means*1;
    inter ~~ cov*slope;
    V1 ~~ residual*V1; V1 ~ 0*1;
    V2 ~~ residual*V2; V2 ~ 0*1;
    V4 ~~ residual*V4; V4 ~ 0*1;
    V6 ~~ residual*V6; V6 ~ 0*1;
'
run <- lavaan(linearGCM,wisc) # could also have used growth()
```

```
#summary(run)
coef(run)
```

```
##     vari    meani     vars    means      cov residual
##   15.196   15.151    1.529    4.673    1.565   12.828
```

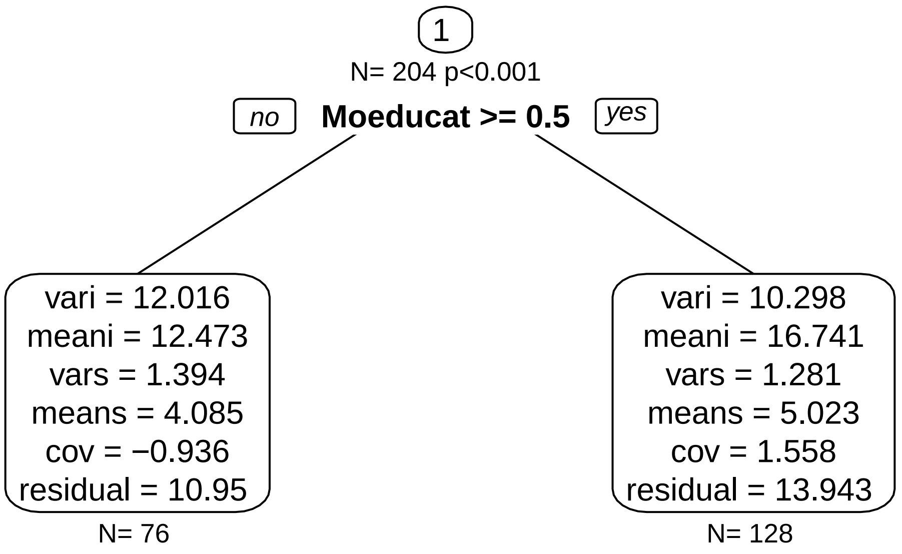
Now use "run" with semtree()

Just used defaults

```
# wisc.sub <- wisc[,c(1:4,9)]
mymodel <- lavaan(linearGCM,wisc,do.fit=FALSE)
mytree <- semtree(mymodel,wisc[,c(1:4,9)]) # only moeducat as covariate
```

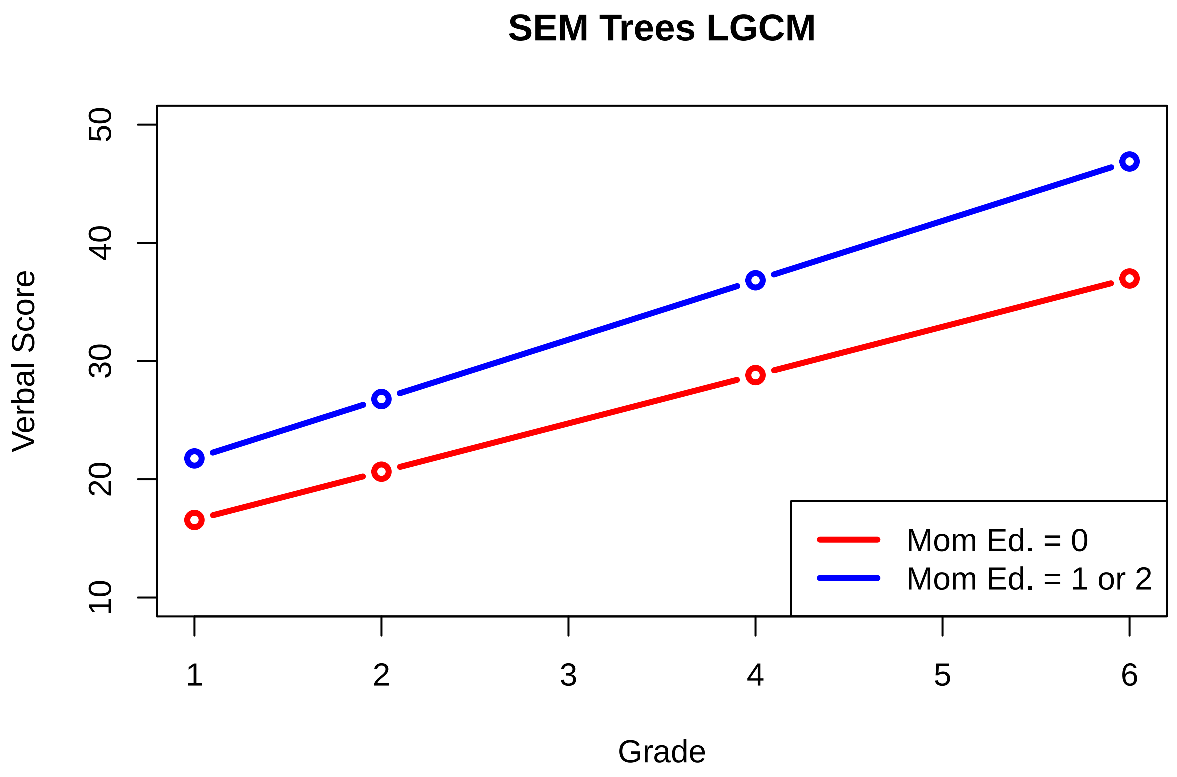
```
## Default SEMtree settings established since no Controls provided.
## [x] Tree construction finished regularly!
```

```
plot(mytree)
```



Plot Trajectories

```
# create expected trajectories from parameters
expected.growth <- matrix(
    rep(t(parameters(mytree))[, "meani"], each=4) +
    rep(t(parameters(mytree))[, "means"], each=4)*c(1,2,4,6), nrow=2, byrow=T)
# plot expected trajectories for each leaf
plot(c(1,6), c(10,50), xlab="Grade", ylab="Verbal Score", type="n",main="SEM Trees LGCM")
lines(c(1,2,4,6), expected.growth[1,], col="red", type="b", lw=3)
lines(c(1,2,4,6), expected.growth[2,], col="blue", type="b", lw=3)
legend("bottomright", c("Mom Ed. = 0", "Mom Ed. = 1 or 2"),col=c("red","blue"), lw=3)
```



We should get same results as in the left node of the tree by just subsetting the dataset based on Moeducat = 0

```
wisc.sub <- wisc[wisc$Moeducat == 0,]
run.sub <- lavaan(linearGCM,wisc.sub)
coef(run.sub)
```

```
##     vari    meani     vars    means      cov residual
##   12.016   12.473    1.394    4.085   -0.936   10.950
```

Yup, everything checks out. This should make it clear that SEM Trees is really just subsetting the dataset into subgroups based on values of the covariates entered.

Note: this is really just a multiple group model, with the groups now Mother's Education = 0, and Mother's Education = 1 or 2:

```
wisc.group <- wisc
wisc.group[wisc.group$Moeducat == 1,] <- 2

run.group <- lavaan(linearGCM,wisc.group,group="Moeducat")
coef(run.group)
```

Note, you can constrain the SEM Tree model to only split on specific parameters. For instance, maybe we want to only find group differences based on the mean slope. Very similar thing to forcing invariance with semtree()

This currently fails, must have to use OpenMx to specify original model

```r
constr <- names(coef(run.sub))[-4]
mytree2 <- semtree(mymodel,wisc[,c(1:4,9)],global.constraints=constr)
plot(mytree2)
```

longRPart

Instead of running the models in a SEM framework, longRPart uses mixed-effects models. This works just as well, as many LGCM can be re-specified as mixed-effects models.

nlme

```r
library(nlme)
#no growth baseline
mix0 <- lme(fixed=verbal~1,data=wisc.long,random=~1|id,method="ML")
#summary(mix0)
summary(mix0)$logLik
```

```
## [1] -3170.966
```

```r
noGrowth <- '
    inter =~ 1*V1 + 1*V2 + 1*V4 + 1*V6
    inter ~~ vari*inter; inter ~ meani*1;
    V1 ~~ residual*V1; V1 ~ 0*1;
    V2 ~~ residual*V2; V2 ~ 0*1;
    V4 ~~ residual*V4; V4 ~ 0*1;
    V6 ~~ residual*V6; V6 ~ 0*1;
'
lgc0 <- lavaan(noGrowth,wisc) # could also have used growth()
#summary(lgc0,fit=T)
fitMeasures(lgc0)["logl"]
```

```
##      logl
## -3170.966
```

```r
#summary(run)
#coef(lgc0)




#Linear growth
mix1 <- lme(fixed = verbal ~ grade, random = ~ grade | id, data = wisc.long, method="ML" )
#summary(mix1)
anova.lme(mix1, mix0) #test of linearity
```

```
##      Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## mix1     1  6 5050.817 5079.043 -2519.408
## mix0     2  3 6347.933 6362.046 -3170.966 1 vs 2 1303.116  <.0001
```

```
summary(mix1)$logLik
```

```
## [1] -2519.408
```

```
#summary(mix1)

# LGC from before
fitMeasures(run)["logl"]
```

```
##      logl
## -2519.408
```

```
coef(run)
```

```
##     vari    meani     vars    means      cov residual
##   15.196   15.151    1.529    4.673    1.565   12.828
```

Now that we see how we can specify growth curves as mixed-effects models, lets test out w/ longRPart and see if we get the same answer to SEM Trees

longRPart

```
library(longRPart)
#initial split on the basis based on Moeducat
lcart.mod1 <- longRPart(verbal ~ grade, ~ Moeducat, ~1 | id,wisc.long)
```

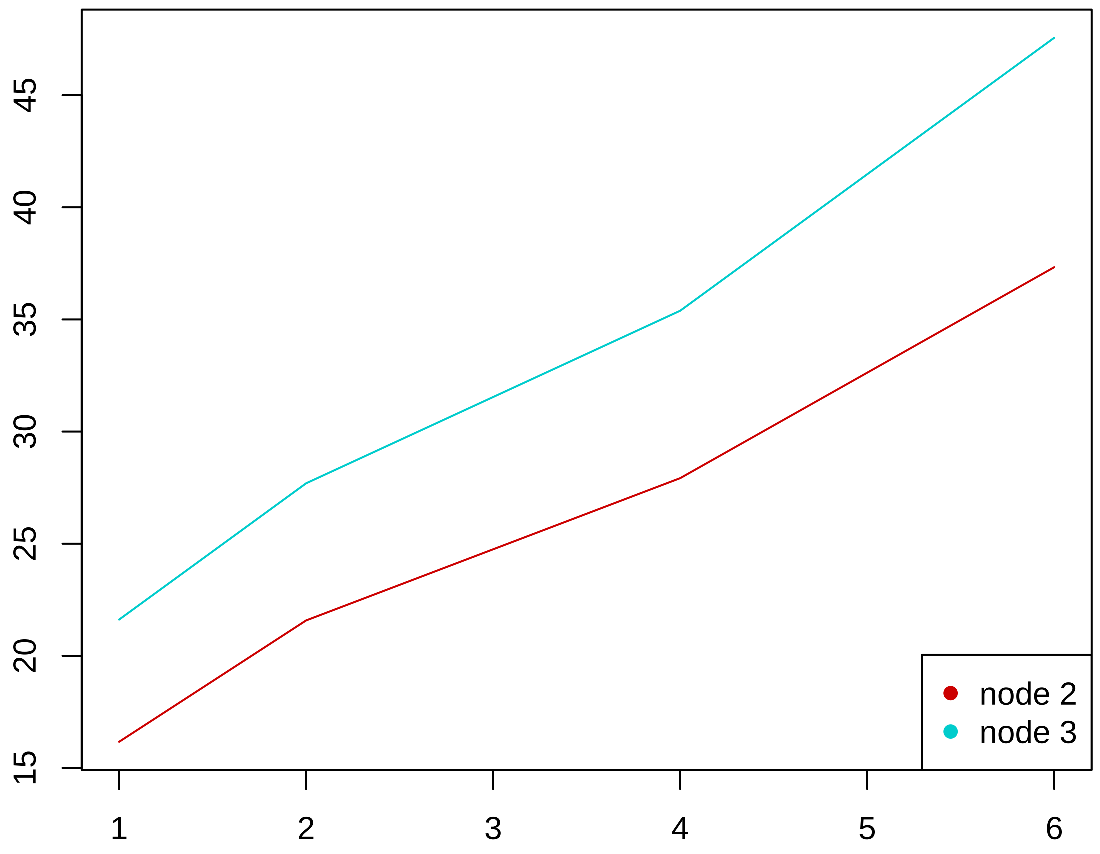
```
## [1] "splitting: 3 values"
## [1] "splitting: 2 values"
```

```
summary(lcart.mod1)
```

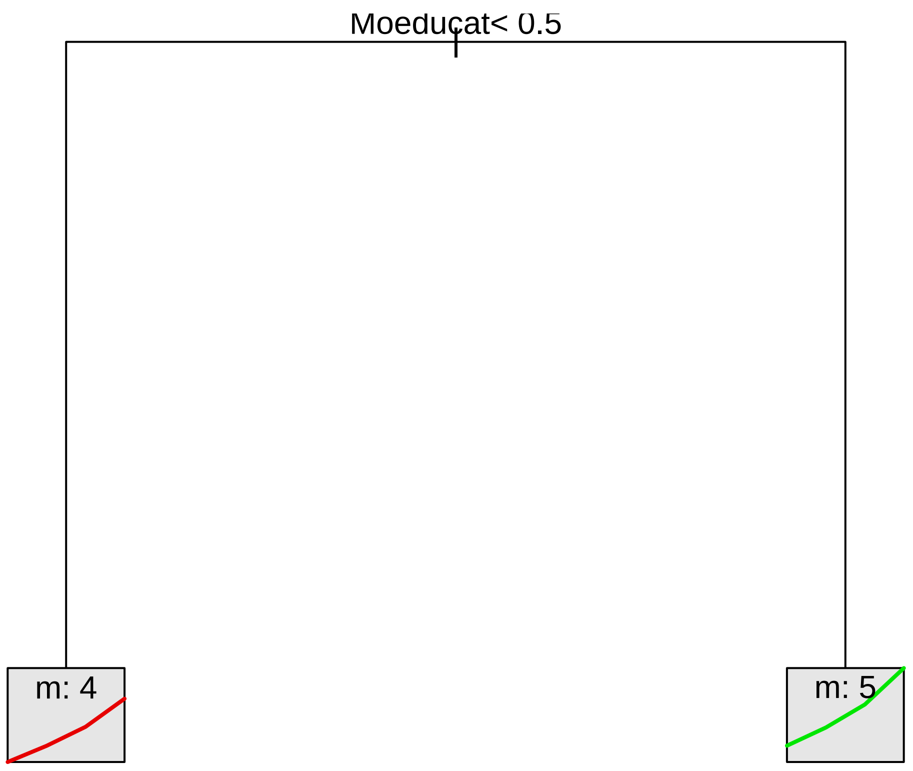
```
## Call:
## rpart(formula = paste(groupingName, c(rPartFormula)), data = data,
##     method = list(eval = evaluation, split = split, init = initialize),
##     parms = data, control = control)
##   n= 816
##
##           CP nsplit rel error
## 1 0.01981397      0  1.000000
## 2 0.01000000      1  0.980186
##
## Variable importance
## Moeducat
##      100
##
## Node number 1: 816 observations,    complexity param=0.01981397
## deviance (-2logLik) 5220.46 slope 4.7
##   left son=2 (304 obs) right son=3 (512 obs)
##   Primary splits:
##       Moeducat < 0.5 to the left,  improve=2661.947, (0 missing)
##
## Node number 2: 304 observations
## deviance (-2logLik) 1865.76 slope 4.1
##
## Node number 3: 512 observations
## deviance (-2logLik) 3251.26 slope 5
```

```
lrpPlot(lcart.mod1)
```

```
## [[1]]
## [1] "1"
##
## [[1]]
## [1] "1"
```



```
lrpTreePlot(lcart.mod1,use.n=F)
```



Get almost identical results as from SEM Trees, but it looks as though longRPart allows more flexibility in the slopes between time points (grade).

REEMtree

Tree partitioning for longitudinal data where random effects exist. This doesn't really accomplish what we did previously with longRPart or SEM Trees. Interested, see the examples in following links.

http://pages.stern.nyu.edu/~jsimonof/REEMtree/

http://www.r-bloggers.com/a-brief-tour-of-the-trees-and-forests/

mvpart

Like longRPart, also archived:

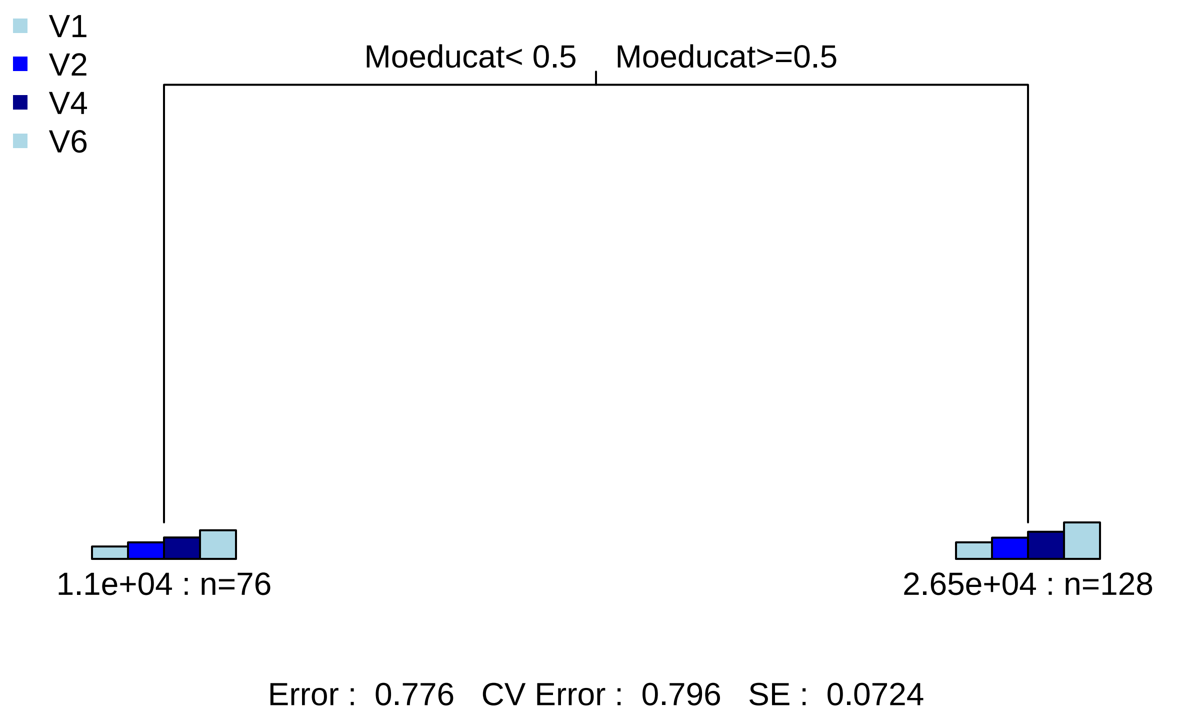
http://cran.r-project.org/src/contrib/Archive/mvpart/

If we treat the longitudinal data just as a multivariate outcome, we can accomplish a very similar process.

```r
library(mvpart)
```

```
##
## Attaching package: 'mvpart'
##
## The following objects are masked from 'package:rpart':
##
##     meanvar, na.rpart, path.rpart, plotcp, post, printcp, prune,
##     prune.rpart, rpart, rpart.control, rsq.rpart, snip.rpart,
##     xpred.rpart
```

```
mvpart(data.matrix(wisc[,1:4]) ~ Moeducat,wisc)
```


V1
V2
V4
V6
Moeducat< 0.5
Moeducat>=0.5
1.1e+04 : n=76
2.65e+04 : n=128
Error : 0.776 CV Error : 0.796 SE : 0.0724