COVID STEROID 2: HTE Analysis

## Analysis

First the required R packages are loaded, as well as the data set. We also source a couple of functions to allow for running BART models with multiple MCMC chains (without needing multiple cores).

rm(list = ls())  
library(BART)  
library(caret)  
library(rpart)  
library(rpart.plot)  
source("clusterfunctions.R")  
  
# Load data from appropriate directory (edit as needed)  
#dat <- read.csv2("L:/LovbeskyttetMapper/Covid-Steroid/COVID-STEROID-2/Finale analyser/Data output/Dataset subset for BART HTE project.csv") #AG, real data  
dat <- read.csv2("~/Downloads/synth\_covid.csv") # Bryan, synthetic data # Bryan, synthetic data

Next we do a small amount of data cleaning/preparation.

# Clean up data variables types and remove the small amount of missing data  
dat$resp\_sup <- as.factor(dat$resp\_sup)  
dat$dead90 <- ifelse(dat$dead90 == TRUE, 1, 0)  
dat <- dat[complete.cases(dat), ]  
  
# Print out summaries of continuous covariates pre-standardization in case  
# needed for interpretation  
print(mean(dat$age, na.rm = T))

## [1] 63.82955

print(sd(dat$age, na.rm = T))

## [1] 12.46226

print(mean(dat$BL9\_Weight, na.rm = T))

## [1] 83.37087

print(sd(dat$BL9\_Weight, na.rm = T))

## [1] 20.71169

# Standardize continuous covariates  
dat$age <- (dat$age - mean(dat$age)) / sd(dat$age)  
dat$BL9\_Weight <- (dat$BL9\_Weight - mean(dat$BL9\_Weight)) / sd(dat$BL9\_Weight)  
  
# Make datasets under each counterfactual  
dat1 <- dat0 <- dat  
dat1$allocation <- TRUE  
dat0$allocation <- FALSE  
  
# Rename covariates so decision tree figures can be used without edits  
colnames(dat)[4:13] <- c("Age", "Limits\_Care", "Respiratory\_Support",  
 "Interleukin", "Early\_Corticosteroids", "Weight",  
 "Diabetes", "Ischemic\_Heart\_Disease",  
 "Pulmonary\_Disease", "Immunosuppression")

Then we run a BART analysis focused on the binary mortality outcome.

# Perform cross validation if not already done (may take >2 hours)  
cvcomplete <- TRUE  
cvresults\_mort <- c(3, 0.25, 400)  
cvresults\_dawols <- c(1, 0.95, 400)  
  
if (cvcomplete == FALSE) {  
   
 # Create 10 folds of the data set for cross-validation  
 set.seed(60622)  
 folds <- createFolds(dat$dead90, k = 10, list = TRUE, returnTrain = FALSE)  
   
 # Initialize output matrices for prediction error from each model  
 cvoutput <- expand.grid(1:3, c(0.25, 0.5, 0.95), c(50, 200, 400), NA)  
 colnames(cvoutput) <- c("Power", "Base", "Ntrees", "CVMSE")  
 mse <- array(NA, dim = c(27, 10))  
  
 for (hp in 1:27) {  
   
 for (i in 1:10) {  
   
 # BART model  
 bartmod <- lbart.cluster(x.train = dat[-folds[[i]], c(1, 4:13)],  
 y.train = dat$dead90[-folds[[i]]],  
 x.test = dat[folds[[i]], c(1, 4:13)],  
 power = cvoutput$Power[hp],  
 base = cvoutput$Base[hp],  
 ntree = cvoutput$Ntrees[hp], nchains = 4)  
 bartmod$yhat.test.collapse <- apply(bartmod$yhat.test, 2, rbind)  
 pred <- exp(colMeans(bartmod$yhat.test.collapse)) /  
 (1 + exp(colMeans(bartmod$yhat.test.collapse)))  
 mse[hp, i] <- mean((dat$dead90[folds[[i]]] - pred)^2)  
   
 }  
   
 }  
  
  
 # Calculate 10-fold CV error for each hyperparameter combination  
 cvoutput$CVMSE <- rowMeans(mse)  
   
 # Fit final model under hyperparameters with minimum CV error  
 set.seed(60622)  
 bartmod1 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE)])  
 bartmod0 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE)])  
   
}  
  
if (cvcomplete == TRUE) {  
   
 # Fit final model under hyperparameters with minimum CV error  
 set.seed(60622)  
 bartmod1 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_mort[1], base = cvresults\_mort[2],  
 ntree = cvresults\_mort[3])  
 bartmod0 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_mort[1], base = cvresults\_mort[2],  
 ntree = cvresults\_mort[3])  
   
}  
  
# Collapse predictions across chains for certain calculations  
bartmod1$yhat.train.collapse <- apply(bartmod1$yhat.train, 2, rbind)  
bartmod1$yhat.test.collapse <- apply(bartmod1$yhat.test, 2, rbind)  
bartmod0$yhat.train.collapse <- apply(bartmod0$yhat.train, 2, rbind)  
bartmod0$yhat.test.collapse <- apply(bartmod0$yhat.test, 2, rbind)

Print the chosen hyperparameters here for future reference:

if (cvcomplete == FALSE) {  
 print(cvoutput[which.min(cvoutput$CVMSE), ])  
}

Then conditional average treatment effects are estimated using the predictions under each counterfactual.

dat$cate <-  
 exp(colMeans(bartmod1$yhat.test.collapse)) /  
 (1 + exp(colMeans(bartmod1$yhat.test.collapse))) -  
 exp(colMeans(bartmod0$yhat.test.collapse)) /  
 (1 + exp(colMeans(bartmod0$yhat.test.collapse)))

This full process is then repeated for the continuous outcome (days alive without life support by day 90).

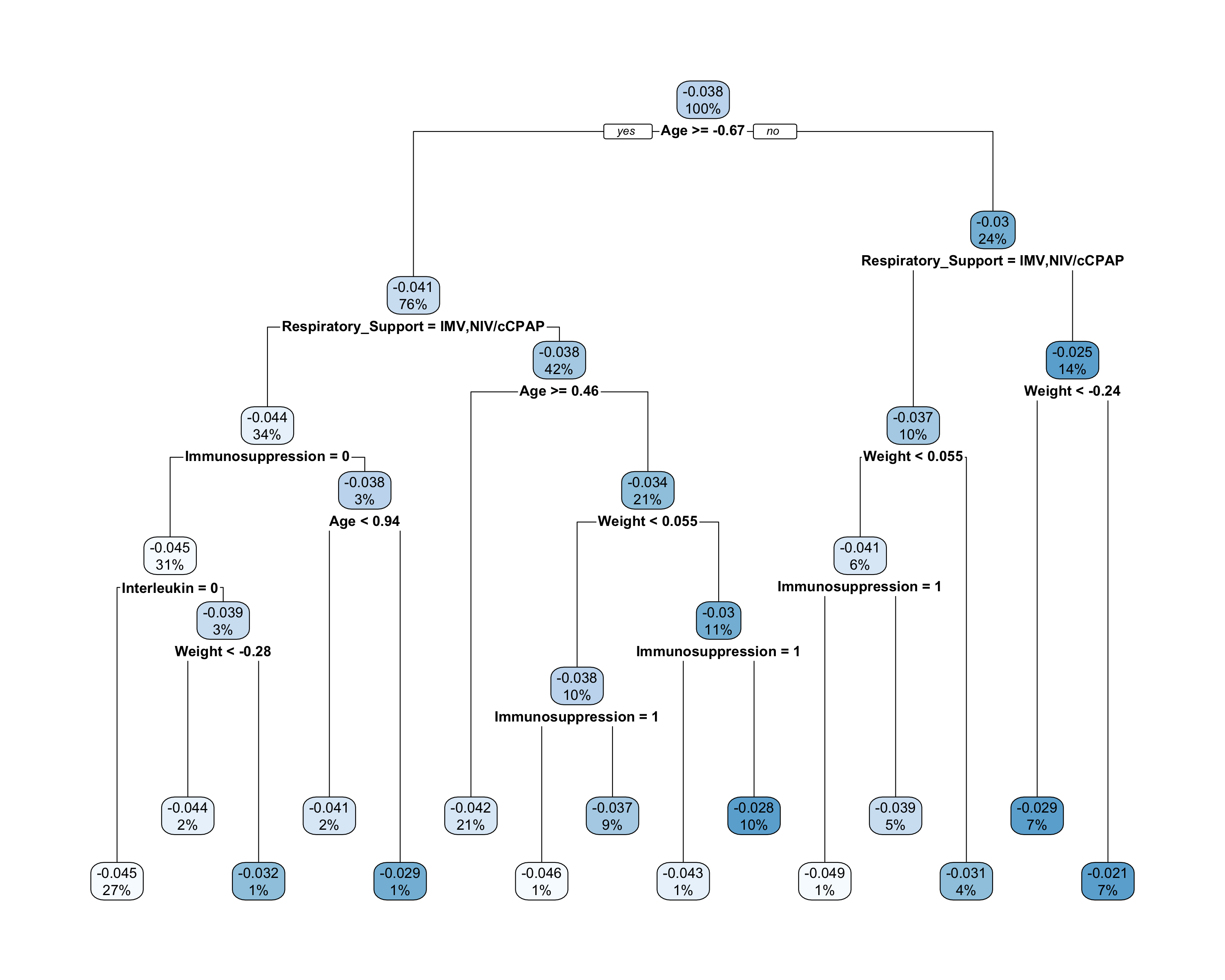
if (cvcomplete == FALSE) {  
  
 # Initialize output for prediction error from each model  
 cvoutput$CVMSE\_c <- NA  
 mse\_c <- array(NA, dim = c(27, 10))  
   
 # Perform cross validation (should take much less time than the binary outcome)  
 set.seed(60622)  
 for (hp in 1:27) {  
   
 for (i in 1:10) {  
   
 # BART model  
 bartmod\_c <- wbart.cluster(x.train = dat[-folds[[i]], c(1, 4:13)],  
 y.train = dat$dawols90[-folds[[i]]],  
 x.test = dat[folds[[i]], c(1, 4:13)],  
 power = cvoutput$Power[hp],  
 base = cvoutput$Base[hp],  
 ntree = cvoutput$Ntrees[hp], nchains = 4)  
 bartmod\_c$yhat.test.collapse <- apply(bartmod\_c$yhat.test, 2, rbind)  
 pred\_c <- colMeans(bartmod\_c$yhat.test.collapse)  
 mse\_c[hp, i] <- mean((dat$dawols90[folds[[i]]] - pred\_c)^2)  
   
 }  
   
 }  
   
 # Calculate 10-fold CV error for each hyperparameter combination  
 cvoutput$CVMSE\_c <- rowMeans(mse\_c)  
   
 # Fit final models under hyperparameters with minimum CV error  
 set.seed(60622)  
 bartmod1\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE\_c)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE\_c)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE\_c)])  
 bartmod0\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE\_c)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE\_c)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE\_c)])  
   
}  
  
if (cvcomplete == TRUE) {  
   
 # Fit final models under hyperparameters with minimum CV error  
 set.seed(60622)  
 bartmod1\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_dawols[1], base = cvresults\_dawols[2],  
 ntree = cvresults\_dawols[3])  
 bartmod0\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_dawols[1], base = cvresults\_dawols[2],  
 ntree = cvresults\_dawols[3])  
   
}  
  
# Collapse predictions across chains for certain calculations  
bartmod1\_c$yhat.train.collapse <- apply(bartmod1\_c$yhat.train, 2, rbind)  
bartmod1\_c$yhat.test.collapse <- apply(bartmod1\_c$yhat.test, 2, rbind)  
bartmod0\_c$yhat.train.collapse <- apply(bartmod0\_c$yhat.train, 2, rbind)  
bartmod0\_c$yhat.test.collapse <- apply(bartmod0\_c$yhat.test, 2, rbind)  
  
# Estimate CATEs  
dat$cate\_c <- colMeans(bartmod1\_c$yhat.test.collapse) -  
 colMeans(bartmod0\_c$yhat.test.collapse)

Print the chosen hyperparameters here for future reference:

if (cvcomplete == FALSE) {  
 print(cvoutput[which.min(cvoutput$CVMSE\_c), ])  
}

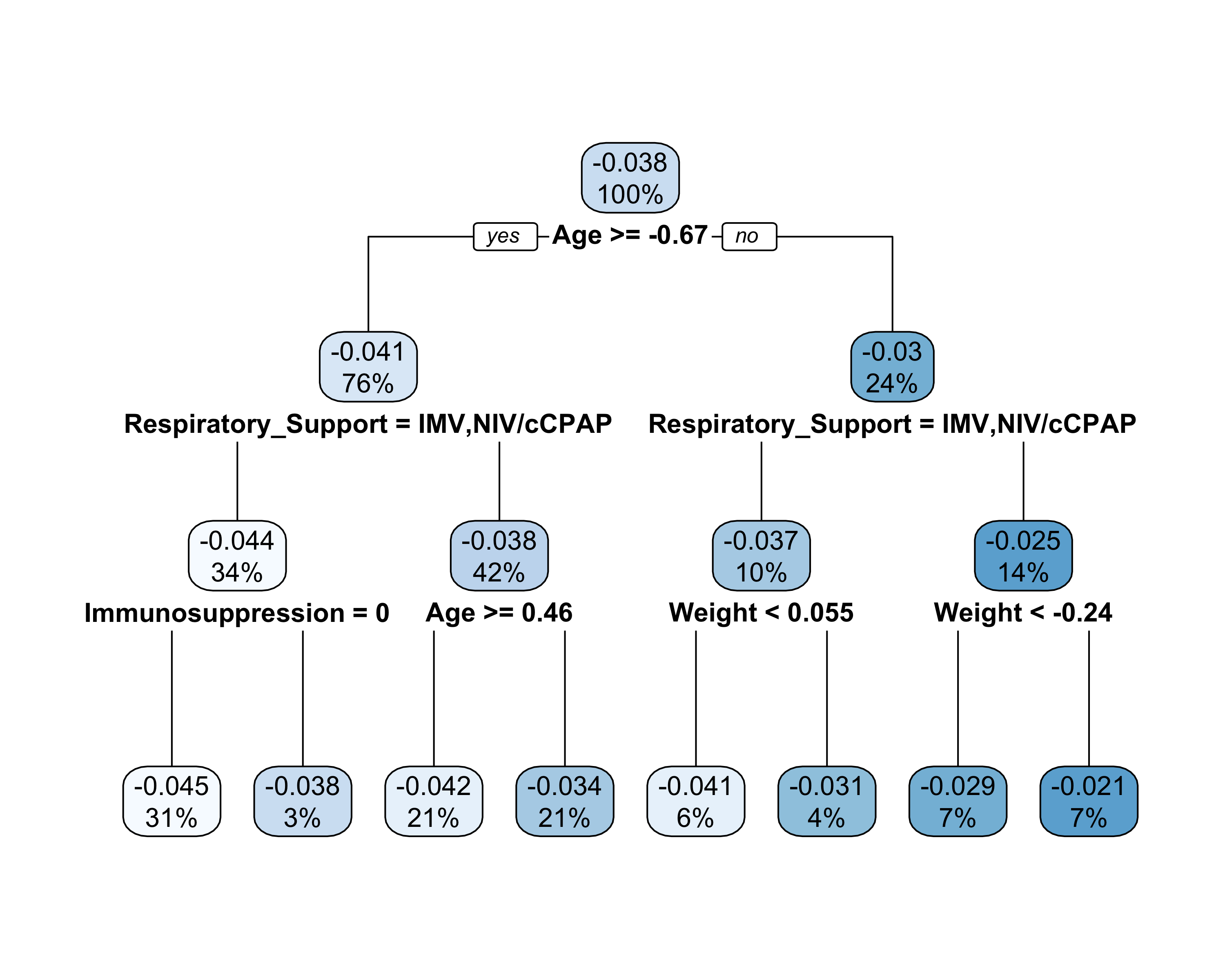
Finally, the “fit-the-fit” approach is used to find subgroups exhibiting heterogeneity of treatment effect, starting with the binary outcome. In particular, a CART model is fit with the CATE for 90-day mortality as the outcome and the covariates as possible predictors. The model is first fit under default CART hyperparameter settings.

# CART model for 90 day mortality with default CART hyperparameter and  
# all covariates considered  
cartmod <- rpart(cate ~ ., data = dat[, c(4:14)], method = "anova")  
rpart.plot(cartmod)



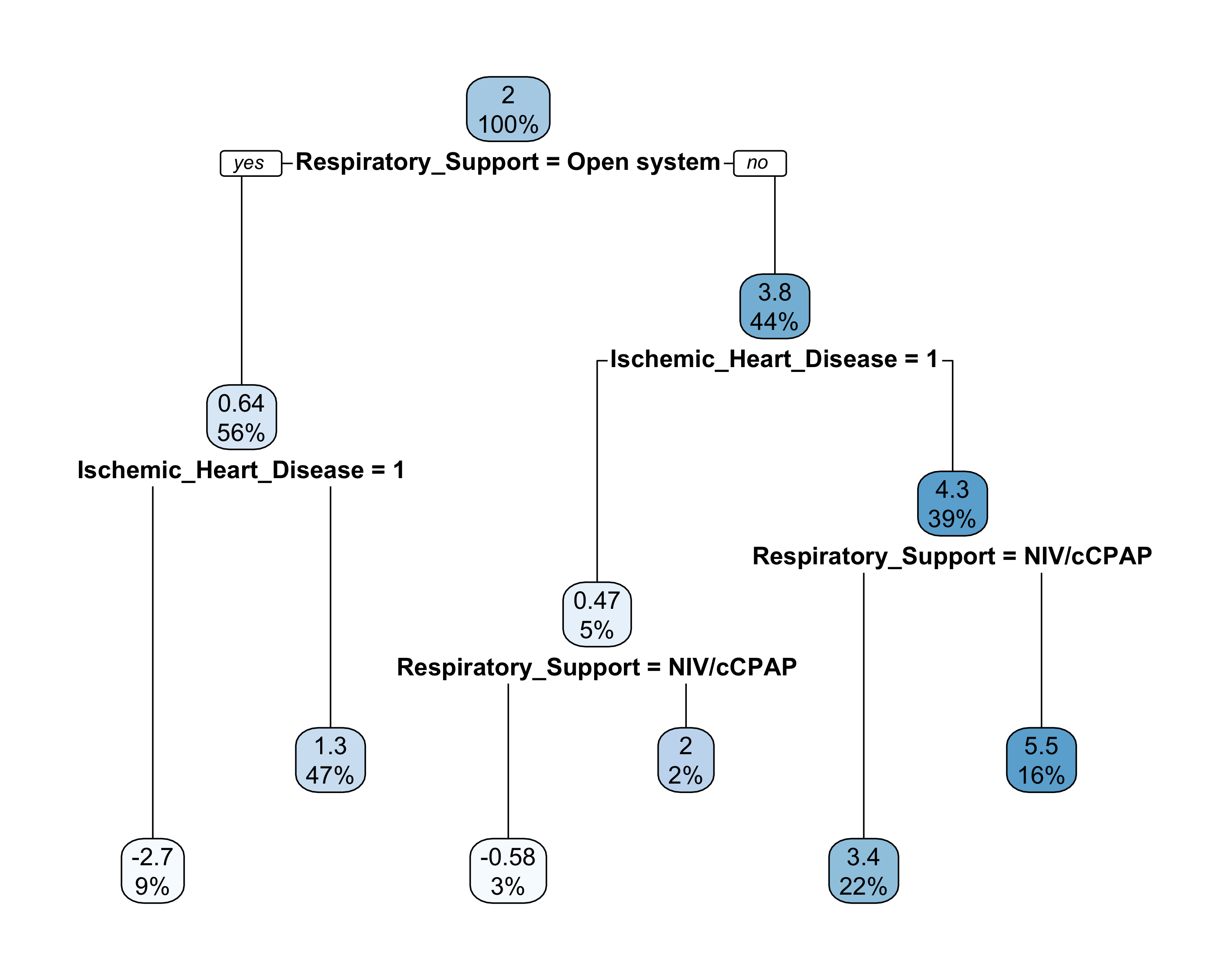
Now we prune the tree for interpretability using a maximum depth of 3 nodes.

cartmod <- rpart(cate ~ ., data = dat[, c(4:14)], method = "anova",  
 maxdepth = 3)  
rpart.plot(cartmod)



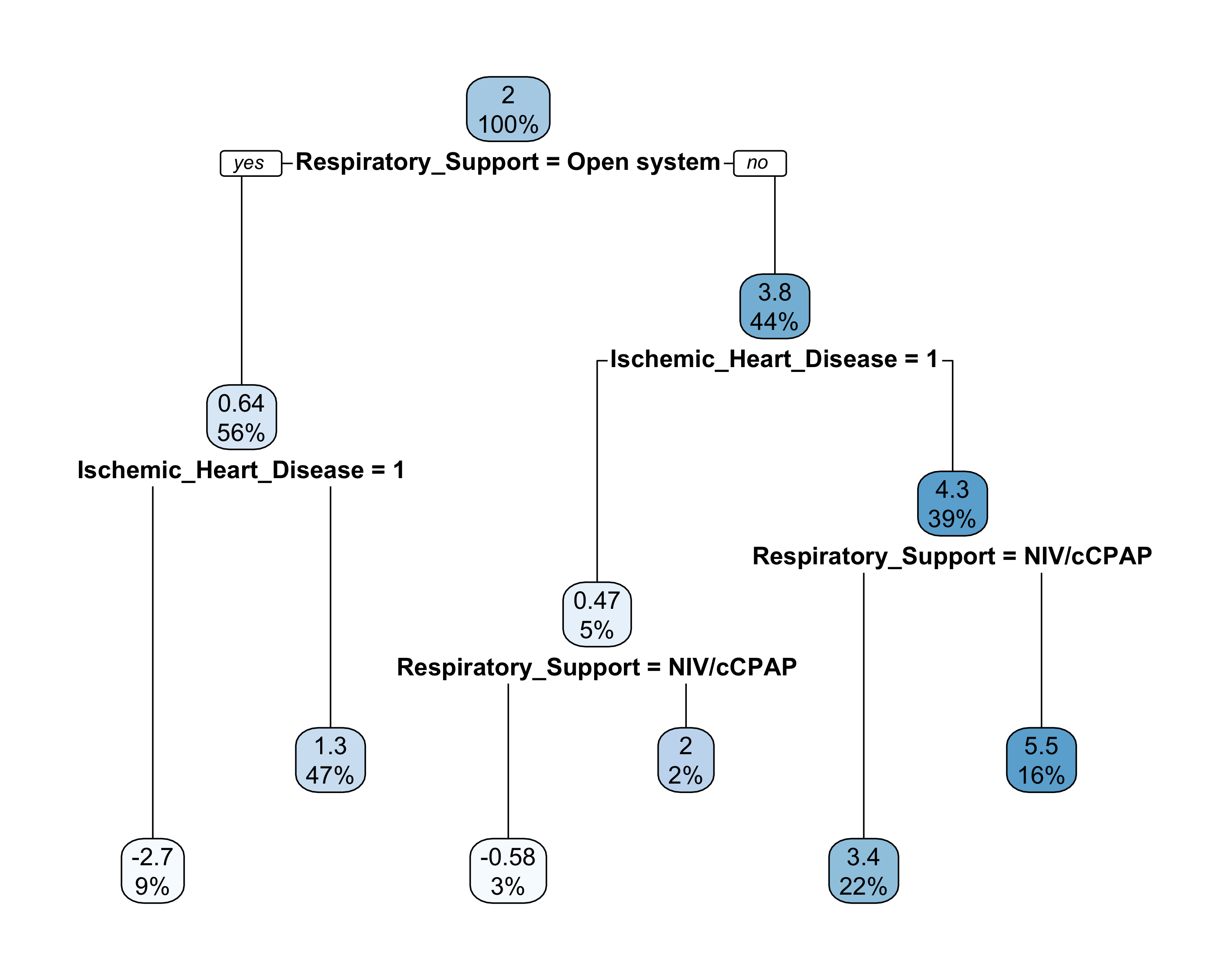
Next the same fit-the-fit approach is used to summarize the results for the continuous outcome, starting with a CART model using all covariates and default hyperparameter.

# CART model for 90 day mortality with default CART hyperparameter and  
# all covariates considered  
cartmod <- rpart(cate\_c ~ ., data = dat[, c(4:13, 15)], method = "anova")  
rpart.plot(cartmod)



Now we prune the tree for interpretability using a maximum depth of 3 nodes.

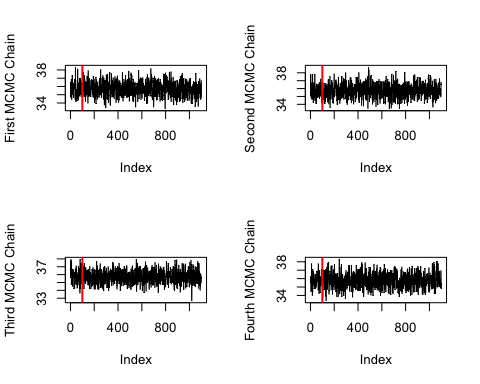
cartmod <- rpart(cate\_c ~ ., data = dat[, c(4:13, 15)], method = "anova",  
 maxdepth = 3)  
rpart.plot(cartmod)



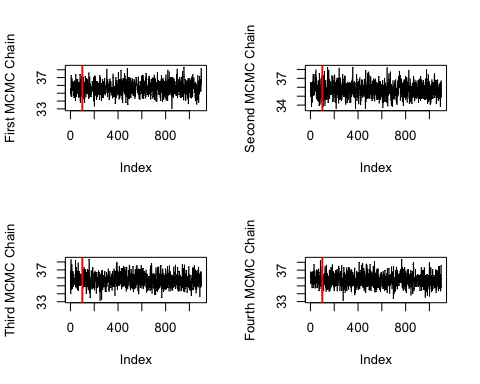
## Diagnostics

Next we run standard diagnostics for each of the models. We begin with diagnostics for the continuous outcome models, which are simpler.

# MCMC chains for parameter in models of continuous outcome  
par(mfrow = c(2, 2))  
plot(bartmod0\_c$sigma[, 1], type = "l", ylab = "First MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")  
plot(bartmod0\_c$sigma[, 2], type = "l", ylab = "Second MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")  
plot(bartmod0\_c$sigma[, 3], type = "l", ylab = "Third MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")  
plot(bartmod0\_c$sigma[, 4], type = "l", ylab = "Fourth MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")



plot(bartmod1\_c$sigma[, 1], type = "l", ylab = "First MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")  
plot(bartmod1\_c$sigma[, 2], type = "l", ylab = "Second MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")  
plot(bartmod1\_c$sigma[, 3], type = "l", ylab = "Third MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")  
plot(bartmod1\_c$sigma[, 4], type = "l", ylab = "Fourth MCMC Chain")  
abline(v = 100, lwd = 2, col = "red")



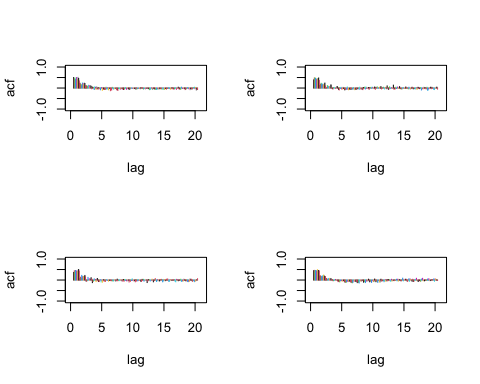
# Posterior diagnostics of sigma:  
posterior::summarise\_draws(posterior::draws\_array(sigma0 = bartmod0\_c$sigma[101:1100, ],  
 sigma1 = bartmod1\_c$sigma[101:1100, ],  
 .nchains = 4))

## # A tibble: 2 × 10  
## variable mean median sd mad q5 q95 rhat ess\_bulk ess\_tail  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 sigma0 35.7 35.7 0.824 0.830 34.4 37.1 1.00 3471. 3848.  
## 2 sigma1 35.7 35.7 0.808 0.818 34.4 37.1 1.00 3594. 3648.

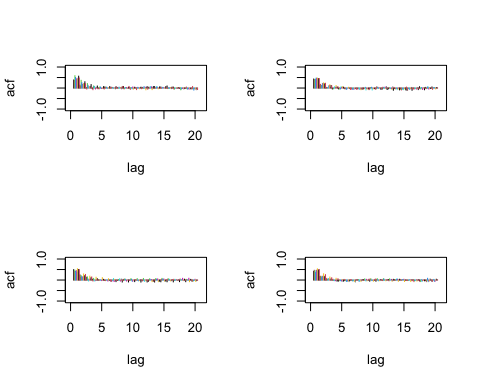
One should check that each chain has converged after burn-in (designated by the vertical red lines). In general, the chains should be converging to approximately the same value.

Next consider diagnostics for the models with the binary mortality outcome as described in Sparapani (2021). First consider the autocorrelation of the estimated response surface from BART from 10 randomly selected subjects. This may start somewhat correlated for nearby observations, but should reduce to 0 correlation for observations further apart.

# First for bartmod0, one panel for each chain  
par(mfrow = c(2, 2))  
  
auto.corr <- acf(bartmod0$yhat.train[ , sample(1:dim(dat)[1], 10), 1],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}  
  
auto.corr <- acf(bartmod0$yhat.train[ , sample(1:dim(dat)[1], 10), 2],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}  
  
auto.corr <- acf(bartmod0$yhat.train[ , sample(1:dim(dat)[1], 10), 3],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}  
  
auto.corr <- acf(bartmod0$yhat.train[ , sample(1:dim(dat)[1], 10), 4],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}

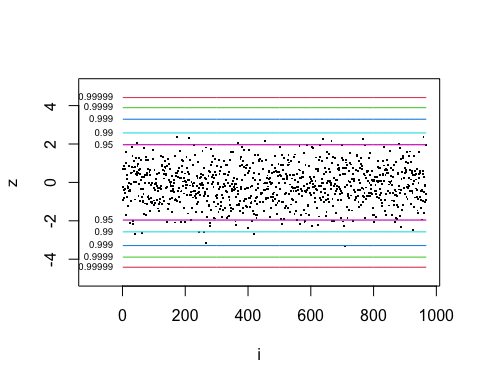


# Then for bartmod1  
auto.corr <- acf(bartmod1$yhat.train[ , sample(1:dim(dat)[1], 10), 1],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}  
  
auto.corr <- acf(bartmod1$yhat.train[ , sample(1:dim(dat)[1], 10), 2],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}  
  
auto.corr <- acf(bartmod1$yhat.train[ , sample(1:dim(dat)[1], 10), 3],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}  
  
auto.corr <- acf(bartmod1$yhat.train[ , sample(1:dim(dat)[1], 10), 4],  
 plot = FALSE)  
max.lag <- max(auto.corr$lag[ , 1, 1])  
  
j <- seq(-0.5, 0.4, length.out = 10)  
for (h in 1:10) {  
 if (h == 1) {  
 plot(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', xlim = c(0, max.lag + 1), ylim = c(-1, 1),  
 ylab = 'acf', xlab = 'lag')  
 } else {  
 lines(1:max.lag + j[h], auto.corr$acf[1 + (1:max.lag), h, h],  
 type = 'h', col = h)  
 }  
}

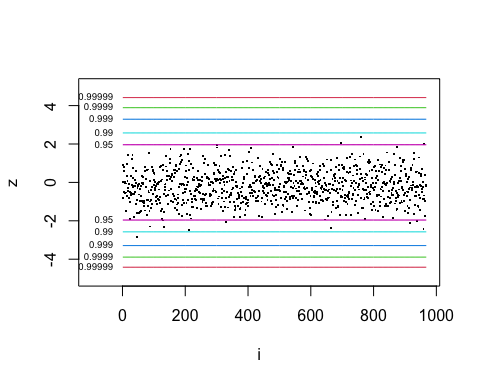


Next, we plot the Geweke Z statistics for each individual, which should be approximately distributed as a standard Normal.

# First for bartmod0  
geweke <- gewekediag(bartmod0$yhat.train.collapse)  
n <- dim(dat)[1]  
j <- -10^(log10(n) - 1)  
plot(geweke$z, pch = '.', cex = 2, ylab = 'z', xlab = 'i',  
 xlim=c(j, n), ylim=c(-5, 5))  
lines(1:n, rep(-1.96, n), type='l', col=6)  
lines(1:n, rep(+1.96, n), type='l', col=6)  
lines(1:n, rep(-2.576, n), type='l', col=5)  
lines(1:n, rep(+2.576, n), type='l', col=5)  
lines(1:n, rep(-3.291, n), type='l', col=4)  
lines(1:n, rep(+3.291, n), type='l', col=4)  
lines(1:n, rep(-3.891, n), type='l', col=3)  
lines(1:n, rep(+3.891, n), type='l', col=3)  
lines(1:n, rep(-4.417, n), type='l', col=2)  
lines(1:n, rep(+4.417, n), type='l', col=2)  
text(c(1, 1), c(-1.96, 1.96), pos=2, cex=0.6, labels='0.95')  
text(c(1, 1), c(-2.576, 2.576), pos=2, cex=0.6, labels='0.99')  
text(c(1, 1), c(-3.291, 3.291), pos=2, cex=0.6, labels='0.999')  
text(c(1, 1), c(-3.891, 3.891), pos=2, cex=0.6, labels='0.9999')  
text(c(1, 1), c(-4.417, 4.417), pos=2, cex=0.6, labels='0.99999')



# Then for bartmod1  
geweke <- gewekediag(bartmod1$yhat.train.collapse)  
plot(geweke$z, pch = '.', cex = 2, ylab = 'z', xlab = 'i',  
 xlim=c(j, n), ylim=c(-5, 5))  
lines(1:n, rep(-1.96, n), type='l', col=6)  
lines(1:n, rep(+1.96, n), type='l', col=6)  
lines(1:n, rep(-2.576, n), type='l', col=5)  
lines(1:n, rep(+2.576, n), type='l', col=5)  
lines(1:n, rep(-3.291, n), type='l', col=4)  
lines(1:n, rep(+3.291, n), type='l', col=4)  
lines(1:n, rep(-3.891, n), type='l', col=3)  
lines(1:n, rep(+3.891, n), type='l', col=3)  
lines(1:n, rep(-4.417, n), type='l', col=2)  
lines(1:n, rep(+4.417, n), type='l', col=2)  
text(c(1, 1), c(-1.96, 1.96), pos=2, cex=0.6, labels='0.95')  
text(c(1, 1), c(-2.576, 2.576), pos=2, cex=0.6, labels='0.99')  
text(c(1, 1), c(-3.291, 3.291), pos=2, cex=0.6, labels='0.999')  
text(c(1, 1), c(-3.891, 3.891), pos=2, cex=0.6, labels='0.9999')  
text(c(1, 1), c(-4.417, 4.417), pos=2, cex=0.6, labels='0.99999')



If several points lie beyond the dark blue line or further, consider using more thinning when fitting the models.

## Sensitivity analysis

We conduct two simple sensitivity analyses, one assuming that all missing mortality data correspond to an alive status and one assuming that all missing mortality data correspond to a deceased status. In the former scenario, missing days without life support will be set to a random sample with replacement from the observed days without life support among those known to be alive at day 90; in the latter scenario it will be set to a random sample with replacement from the observed days without life support among those not alive at day 90. Default hyperparameters are used in these analyses.

Note that these analyses are unlikely to output the exact same trees as the originl analysis, but they should hopefully output trees which largely tell the same HTE story as the original analysis.

### Sensitivity analysis 1: Impute missing outcomes as alive

First the memory is cleared and the data set is reloaded (the cross validation output is saved for future use).

rm(list = ls()[!(ls() %in% list("cvoutput", "cvcomplete",  
 "cvresults\_mort", "cvresults\_dawols"))])  
library(BART)  
library(caret)  
library(rpart)  
library(rpart.plot)  
source("clusterfunctions.R")  
  
# Load data from appropriate directory (edit as needed)  
#dat <- read.csv2("L:/LovbeskyttetMapper/Covid-Steroid/COVID-STEROID-2/Finale analyser/Data output/Dataset subset for BART HTE project.csv") #AG, real data  
dat <- read.csv2("~/Downloads/synth\_covid.csv") # Bryan, synthetic data # Bryan, synthetic data

Next we do a small amount of data cleaning/preparation.

# Clean up data variables types and impute the small amount of missing data  
dat$resp\_sup <- as.factor(dat$resp\_sup)  
dat$dead90 <- ifelse(dat$dead90 == TRUE, 1, 0)  
dat$dawols90[is.na(dat$dawols90)] <-  
 sample(dat$dawols90[!is.na(dat$dawols90) & dat$dead90 == 0],  
 sum(is.na(dat$dawols90)), replace = TRUE)  
dat$dead90[is.na(dat$dead90)] <- 0  
  
# Standardize continuous covariates  
dat$age <- (dat$age - mean(dat$age)) / sd(dat$age)  
dat$BL9\_Weight <- (dat$BL9\_Weight - mean(dat$BL9\_Weight)) / sd(dat$BL9\_Weight)  
  
# Make datasets under each counterfactual  
dat1 <- dat0 <- dat  
dat1$allocation <- TRUE  
dat0$allocation <- FALSE

Then we run a BART analysis focused on the binary mortality outcome. Note that we use the hyperparameters selected during the cross-validation procedures in the main analysis.

if (cvcomplete == FALSE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE)])  
 bartmod0 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE)])  
   
}  
  
if (cvcomplete == TRUE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_mort[1], base = cvresults\_mort[2],  
 ntree = cvresults\_mort[3])  
 bartmod0 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_mort[1], base = cvresults\_mort[2],  
 ntree = cvresults\_mort[3])  
   
}  
  
  
# Collapse predictions across chains for certain calculations  
bartmod1$yhat.train.collapse <- apply(bartmod1$yhat.train, 2, rbind)  
bartmod1$yhat.test.collapse <- apply(bartmod1$yhat.test, 2, rbind)  
bartmod0$yhat.train.collapse <- apply(bartmod0$yhat.train, 2, rbind)  
bartmod0$yhat.test.collapse <- apply(bartmod0$yhat.test, 2, rbind)

Then conditional average treatment effects are estimated using the predictions under each counterfactual.

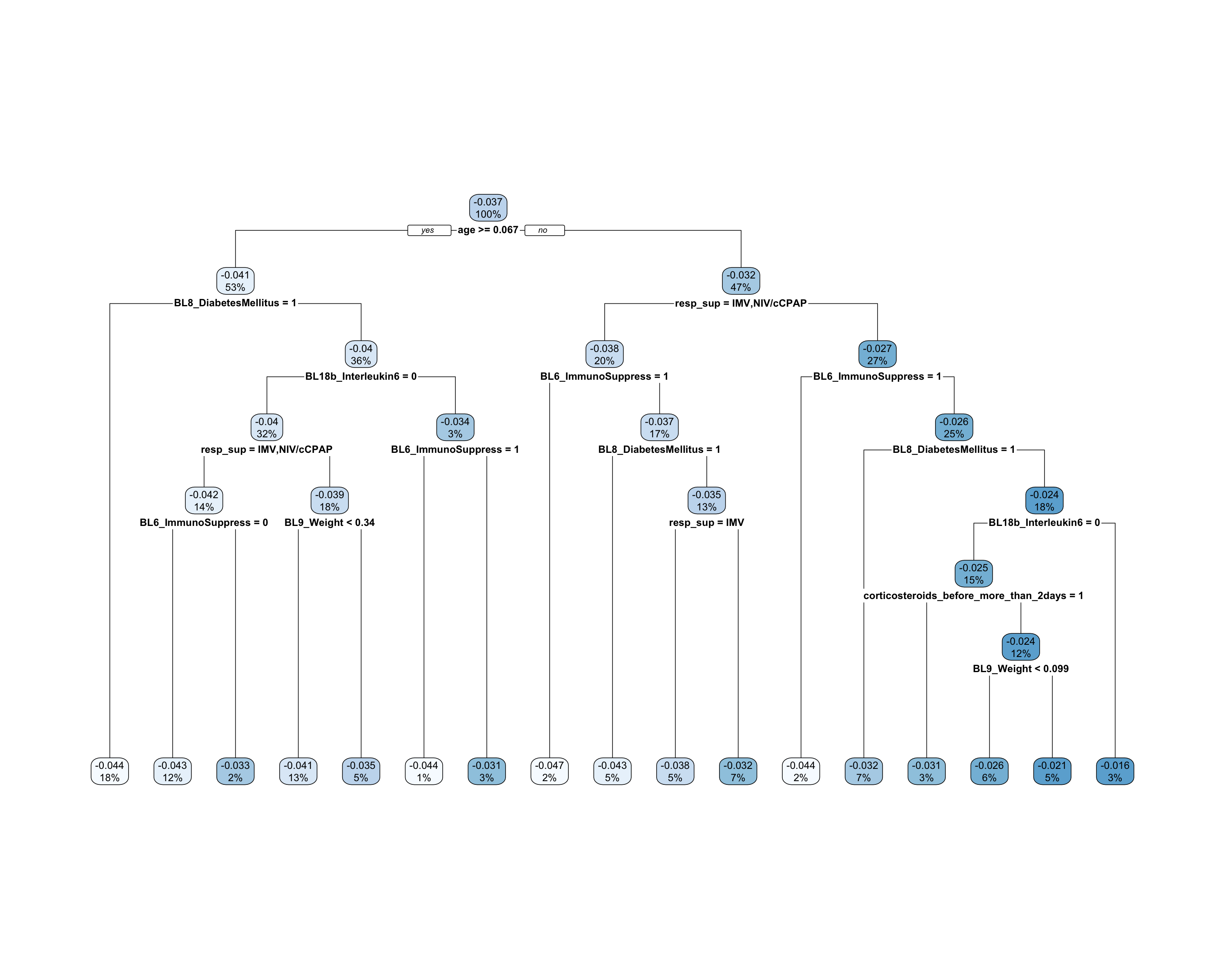
dat$cate <-  
 exp(colMeans(bartmod1$yhat.test.collapse)) /  
 (1 + exp(colMeans(bartmod1$yhat.test.collapse))) -  
 exp(colMeans(bartmod0$yhat.test.collapse)) /  
 (1 + exp(colMeans(bartmod0$yhat.test.collapse)))

This full process is then repeated for the continuous outcome (days alive without life support by day 90).

if (cvcomplete == FALSE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE\_c)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE\_c)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE\_c)])  
 bartmod0\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE\_c)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE\_c)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE\_c)])  
  
}  
  
if (cvcomplete == TRUE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_dawols[1], base = cvresults\_dawols[2],  
 ntree = cvresults\_dawols[3])  
 bartmod0\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_dawols[1], base = cvresults\_dawols[2],  
 ntree = cvresults\_dawols[3])  
  
}  
  
# Collapse predictions across chains for certain calculations  
bartmod1\_c$yhat.train.collapse <- apply(bartmod1\_c$yhat.train, 2, rbind)  
bartmod1\_c$yhat.test.collapse <- apply(bartmod1\_c$yhat.test, 2, rbind)  
bartmod0\_c$yhat.train.collapse <- apply(bartmod0\_c$yhat.train, 2, rbind)  
bartmod0\_c$yhat.test.collapse <- apply(bartmod0\_c$yhat.test, 2, rbind)  
  
# Estimate CATEs  
dat$cate\_c <- colMeans(bartmod1\_c$yhat.test.collapse) -  
 colMeans(bartmod0\_c$yhat.test.collapse)

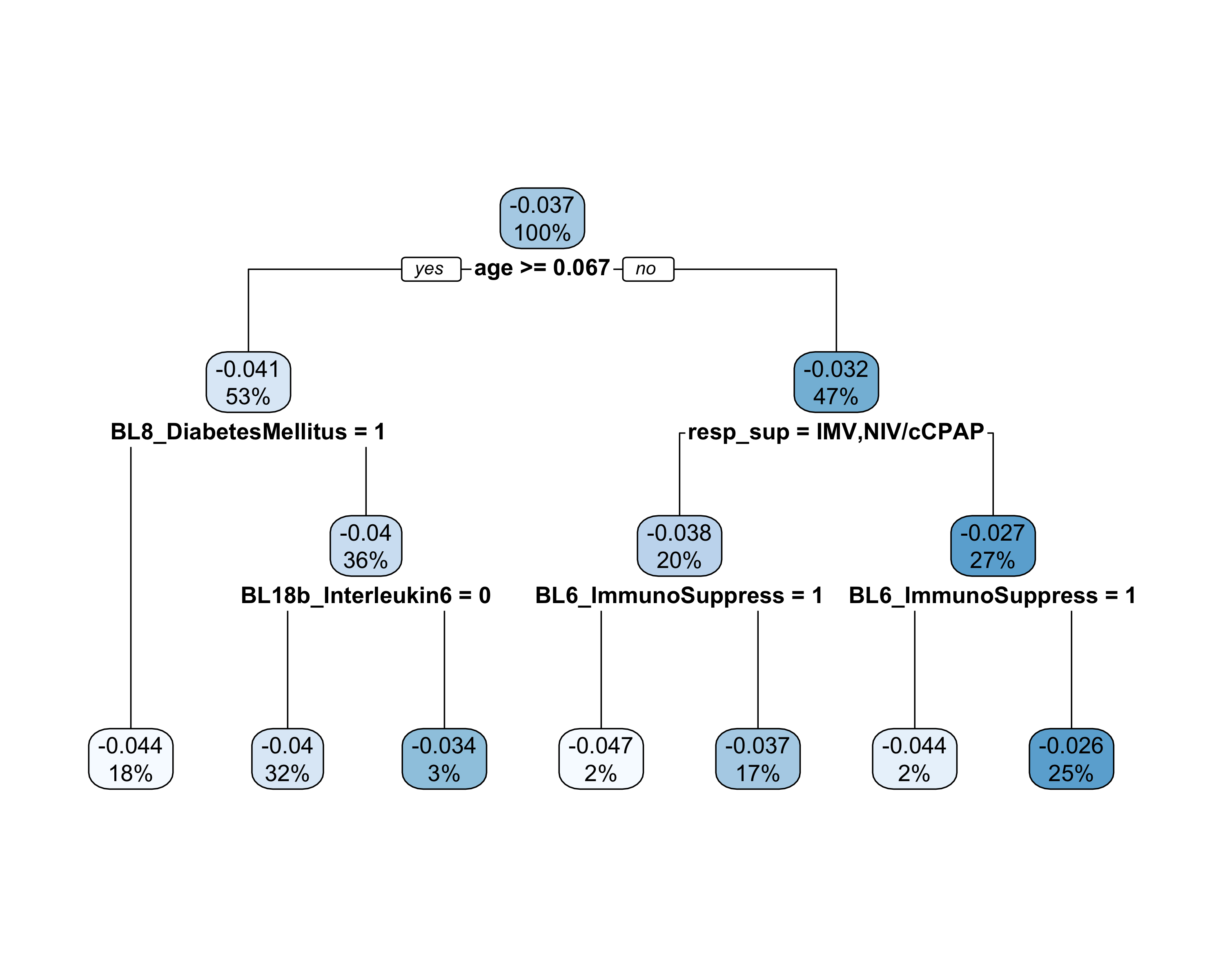
Finally, the “fit-the-fit” approach is used to find subgroups exhibiting heterogeneity of treatment effect, starting with the binary outcome. In particular, a CART model is fit with the CATE for 90-day mortality as the outcome and the covariates as possible predictors. The model is first fit under default CART hyperparameter settings.

# CART model for 90 day mortality with default CART hyperparameter and  
# all covariates considered  
cartmod <- rpart(cate ~ ., data = dat[, c(4:14)], method = "anova")  
rpart.plot(cartmod)



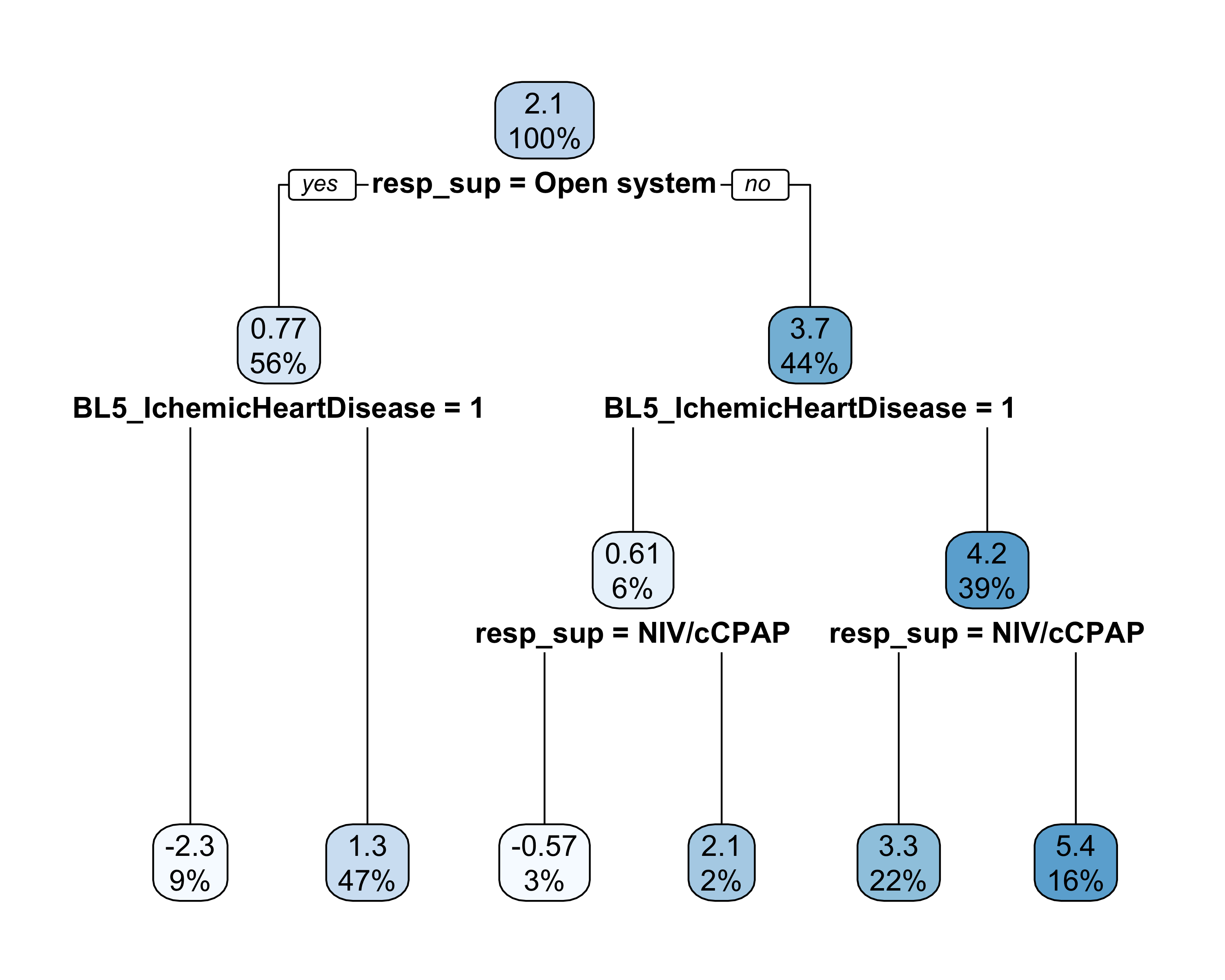
Now we prune the tree for interpretability using a maximum depth of 3 nodes.

cartmod <- rpart(cate ~ ., data = dat[, c(4:14)], method = "anova",  
 maxdepth = 3)  
rpart.plot(cartmod)



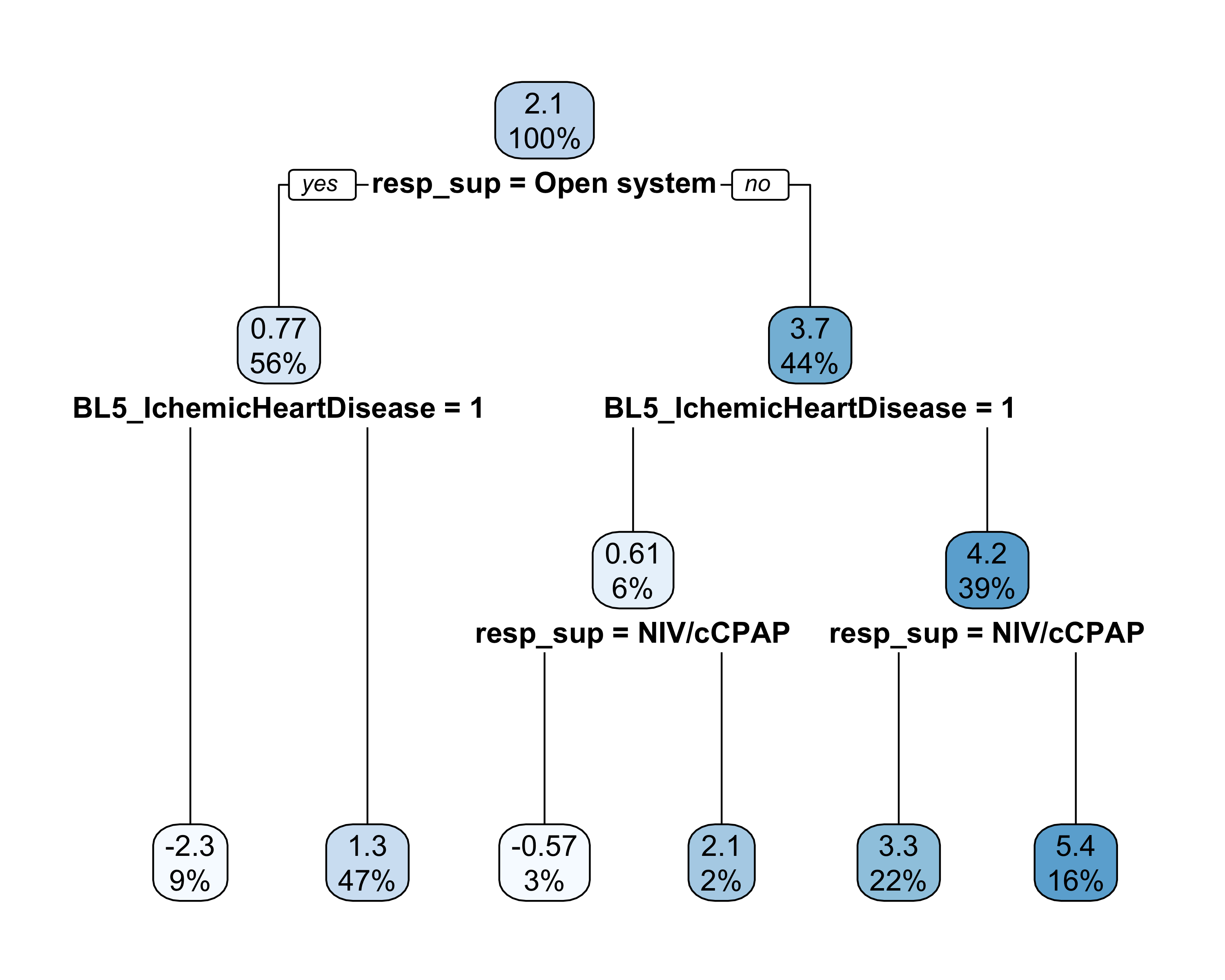
Next the same fit-the-fit approach is used to summarize the results for the continuous outcome, starting with a CART model using all covariates and default hyperparameter.

# CART model for 90 day mortality with default CART hyperparameter and  
# all covariates considered  
cartmod <- rpart(cate\_c ~ ., data = dat[, c(4:13, 15)], method = "anova")  
rpart.plot(cartmod)



Now we prune the tree for interpretability using a maximum depth of 3 nodes.

cartmod <- rpart(cate\_c ~ ., data = dat[, c(4:13, 15)], method = "anova",  
 maxdepth = 3)  
rpart.plot(cartmod)



### Sensitivity analysis 2: Impute missing outcomes as deceased

First the memory is cleared and the data set is reloaded.

rm(list = ls()[!(ls() %in% list("cvoutput", "cvcomplete",  
 "cvresults\_mort", "cvresults\_dawols"))])  
library(BART)  
library(caret)  
library(rpart)  
library(rpart.plot)  
source("clusterfunctions.R")  
  
# Load data from appropriate directory (edit as needed)  
#dat <- read.csv2("L:/LovbeskyttetMapper/Covid-Steroid/COVID-STEROID-2/Finale analyser/Data output/Dataset subset for BART HTE project.csv") #AG, real data  
dat <- read.csv2("~/Downloads/synth\_covid.csv") # Bryan, synthetic data # Bryan, synthetic data

Next we do a small amount of data cleaning/preparation.

# Clean up data variables types and impute the small amount of missing data  
dat$resp\_sup <- as.factor(dat$resp\_sup)  
dat$dead90 <- ifelse(dat$dead90 == TRUE, 1, 0)  
dat$dawols90[is.na(dat$dawols90)] <-  
 sample(dat$dawols90[!is.na(dat$dawols90) & dat$dead90 == 1],  
 sum(is.na(dat$dawols90)), replace = TRUE)  
dat$dead90[is.na(dat$dead90)] <- 1  
  
# Standardize continuous covariates  
dat$age <- (dat$age - mean(dat$age)) / sd(dat$age)  
dat$BL9\_Weight <- (dat$BL9\_Weight - mean(dat$BL9\_Weight)) / sd(dat$BL9\_Weight)  
  
# Make datasets under each counterfactual  
dat1 <- dat0 <- dat  
dat1$allocation <- TRUE  
dat0$allocation <- FALSE

Then we run a BART analysis focused on the binary mortality outcome.

if (cvcomplete == FALSE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE)])  
 bartmod0 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE)])  
  
}  
  
if (cvcomplete == TRUE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_mort[1], base = cvresults\_mort[2],  
 ntree = cvresults\_mort[3])  
 bartmod0 <-  
 lbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dead90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_mort[1], base = cvresults\_mort[2],  
 ntree = cvresults\_mort[3])  
  
}  
  
# Collapse predictions across chains for certain calculations  
bartmod1$yhat.train.collapse <- apply(bartmod1$yhat.train, 2, rbind)  
bartmod1$yhat.test.collapse <- apply(bartmod1$yhat.test, 2, rbind)  
bartmod0$yhat.train.collapse <- apply(bartmod0$yhat.train, 2, rbind)  
bartmod0$yhat.test.collapse <- apply(bartmod0$yhat.test, 2, rbind)

Then conditional average treatment effects are estimated using the predictions under each counterfactual.

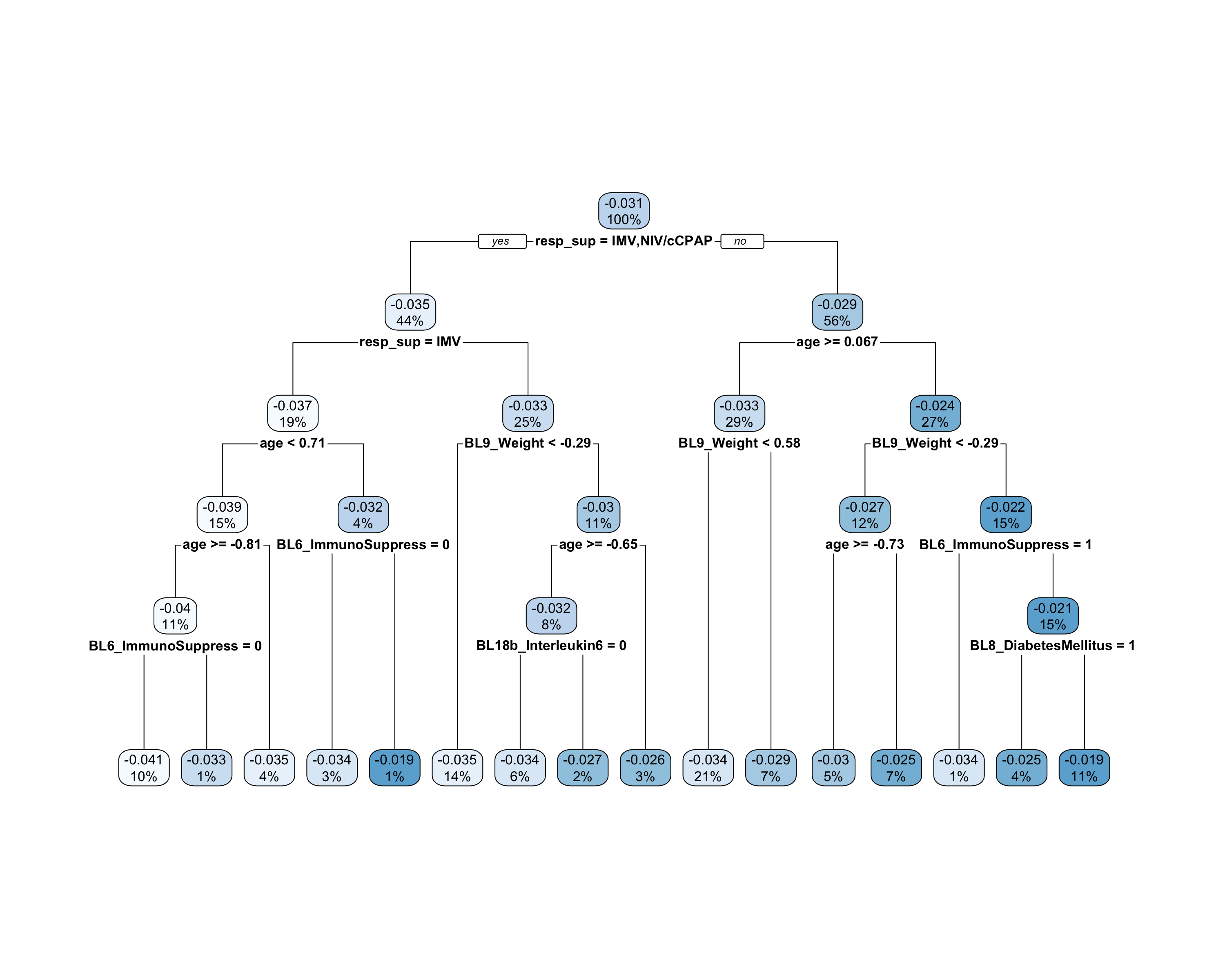
dat$cate <-  
 exp(colMeans(bartmod1$yhat.test.collapse)) /  
 (1 + exp(colMeans(bartmod1$yhat.test.collapse))) -  
 exp(colMeans(bartmod0$yhat.test.collapse)) /  
 (1 + exp(colMeans(bartmod0$yhat.test.collapse)))

This full process is then repeated for the continuous outcome (days alive without life support by day 90).

if (cvcomplete == FALSE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE\_c)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE\_c)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE\_c)])  
 bartmod0\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvoutput$Power[which.min(cvoutput$CVMSE\_c)],  
 base = cvoutput$Base[which.min(cvoutput$CVMSE\_c)],  
 ntree = cvoutput$Ntrees[which.min(cvoutput$CVMSE\_c)])  
  
}  
  
if (cvcomplete == TRUE) {  
   
 # Fit BART models, get predictions under each trt  
 set.seed(60622)  
 bartmod1\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat1[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_dawols[1], base = cvresults\_dawols[2],  
 ntree = cvresults\_dawols[3])  
 bartmod0\_c <-  
 wbart.cluster(x.train = dat[, c(1, 4:13)], y.train = dat$dawols90,  
 x.test = dat0[, c(1, 4:13)], nchains = 4,  
 power = cvresults\_dawols[1], base = cvresults\_dawols[2],  
 ntree = cvresults\_dawols[3])  
  
}  
  
# Collapse predictions across chains for certain calculations  
bartmod1\_c$yhat.train.collapse <- apply(bartmod1\_c$yhat.train, 2, rbind)  
bartmod1\_c$yhat.test.collapse <- apply(bartmod1\_c$yhat.test, 2, rbind)  
bartmod0\_c$yhat.train.collapse <- apply(bartmod0\_c$yhat.train, 2, rbind)  
bartmod0\_c$yhat.test.collapse <- apply(bartmod0\_c$yhat.test, 2, rbind)  
  
# Estimate CATEs  
dat$cate\_c <- colMeans(bartmod1\_c$yhat.test.collapse) -  
 colMeans(bartmod0\_c$yhat.test.collapse)

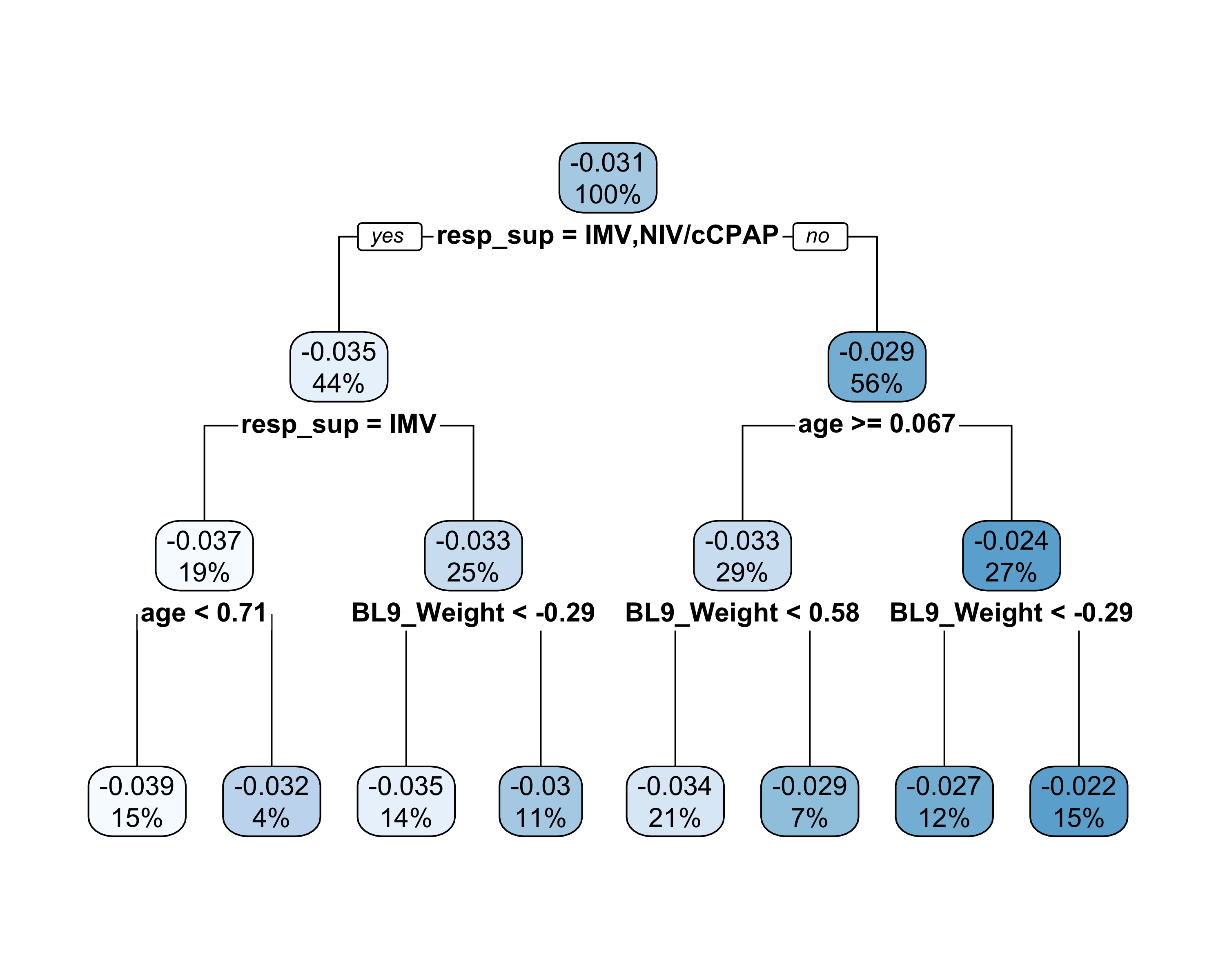
Finally, the “fit-the-fit” approach is used to find subgroups exhibiting heterogeneity of treatment effect, starting with the binary outcome. In particular, a CART model is fit with the CATE for 90-day mortality as the outcome and the covariates as possible predictors. The model is first fit under default CART hyperparameter settings.

# CART model for 90 day mortality with default CART hyperparameter and  
# all covariates considered  
cartmod <- rpart(cate ~ ., data = dat[, c(4:14)], method = "anova")  
rpart.plot(cartmod)



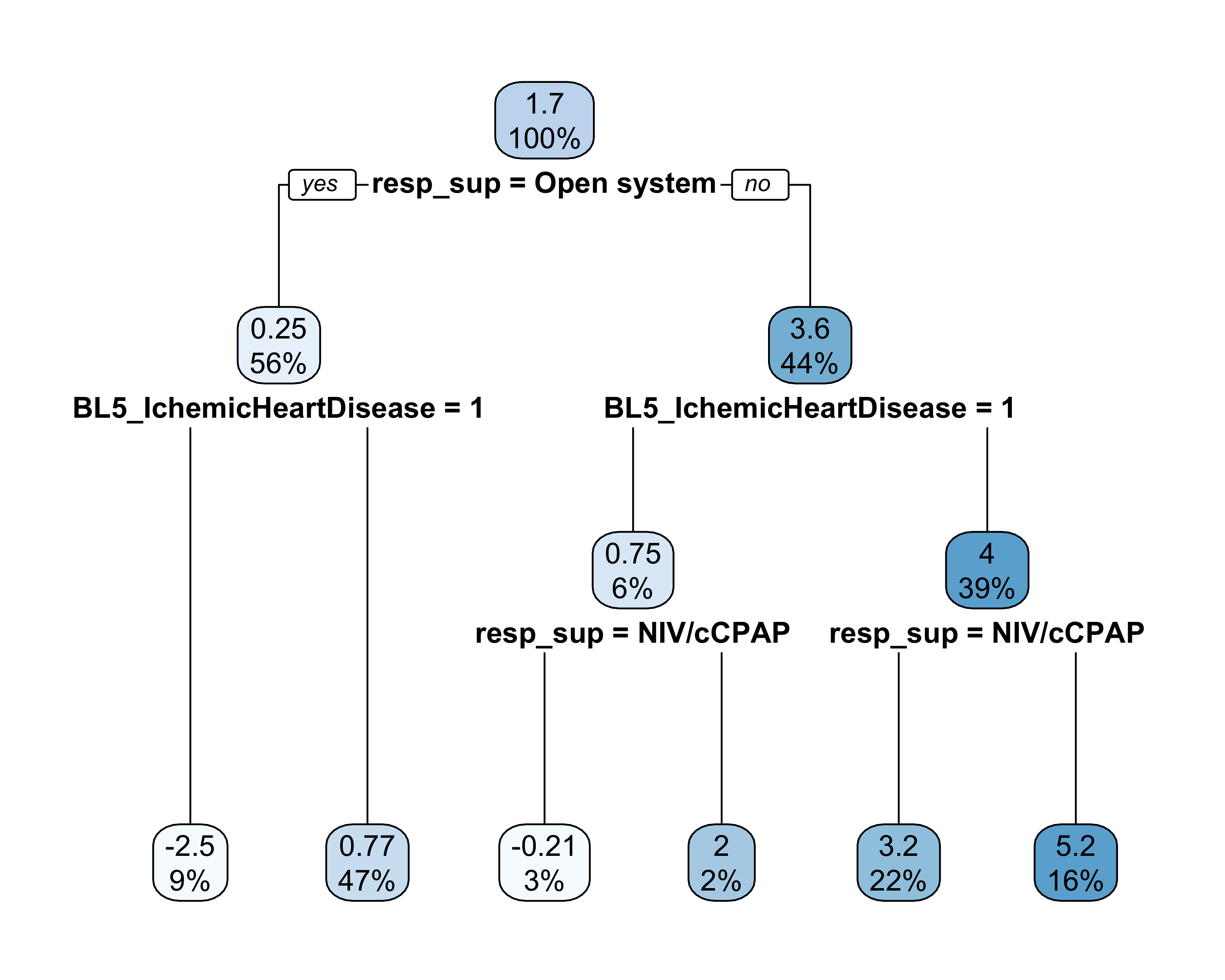
Now we prune the tree for interpretability using a maximum depth of 3 nodes.

cartmod <- rpart(cate ~ ., data = dat[, c(4:14)], method = "anova",  
 maxdepth = 3)  
rpart.plot(cartmod)



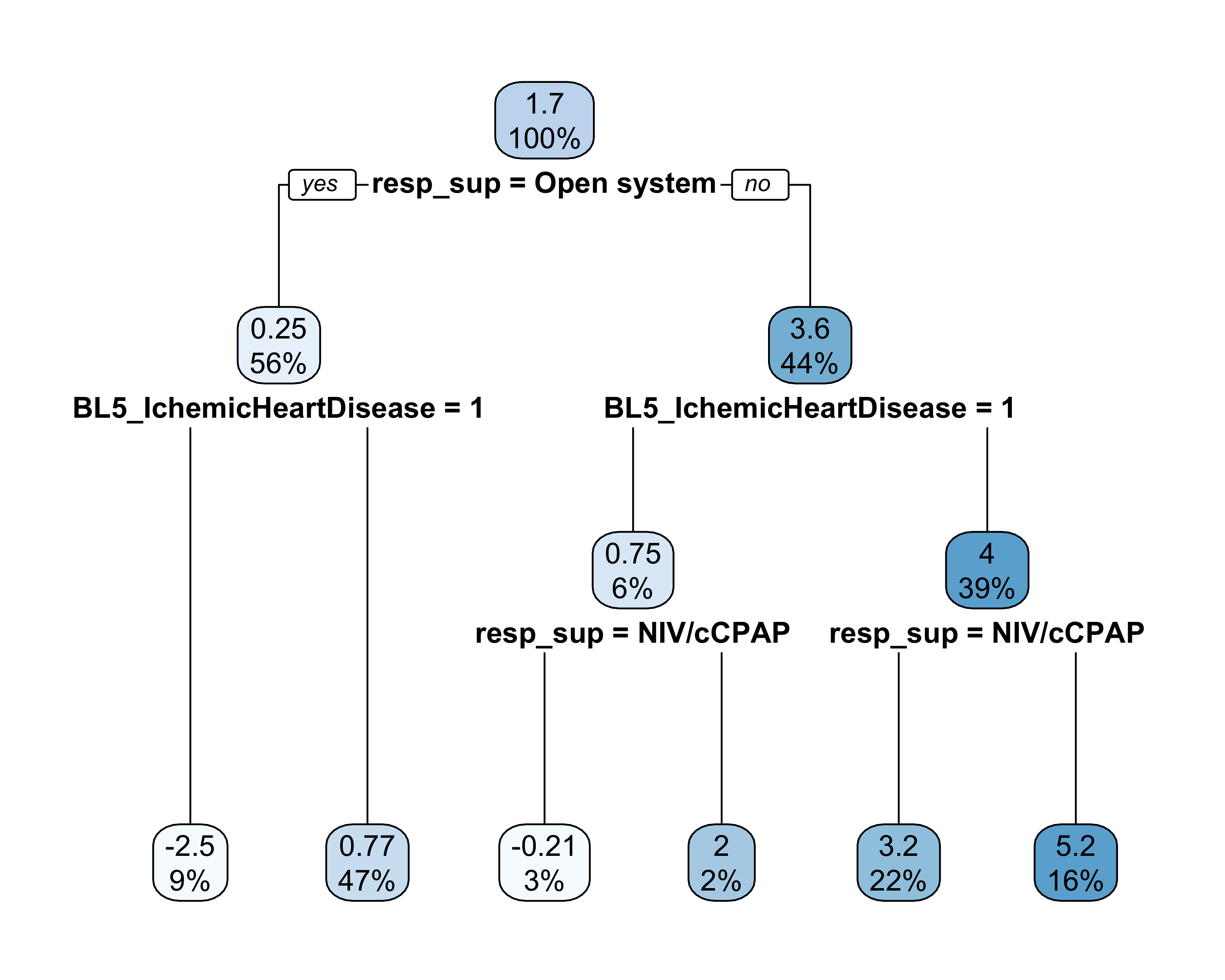
Next the same fit-the-fit approach is used to summarize the results for the continuous outcome, starting with a CART model using all covariates and default hyperparameter.

# CART model for 90 day mortality with default CART hyperparameter and  
# all covariates considered  
cartmod <- rpart(cate\_c ~ ., data = dat[, c(4:13, 15)], method = "anova")  
rpart.plot(cartmod)



Now we prune the tree for interpretability using a maximum depth of 3 nodes.

cartmod <- rpart(cate\_c ~ ., data = dat[, c(4:13, 15)], method = "anova",  
 maxdepth = 3)  
rpart.plot(cartmod)



AG: Log date and session info (reproducibility/save package versions etc.):

date()

## [1] "Mon Sep 12 21:48:19 2022"

sessionInfo()

## R version 4.1.2 (2021-11-01)  
## Platform: aarch64-apple-darwin20 (64-bit)  
## Running under: macOS Monterey 12.3.1  
##   
## Matrix products: default  
## BLAS: /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRblas.0.dylib  
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRlapack.dylib  
##   
## locale:  
## [1] en\_US.UTF-8/en\_US.UTF-8/en\_US.UTF-8/C/en\_US.UTF-8/en\_US.UTF-8  
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] rpart.plot\_3.1.0 rpart\_4.1-15 caret\_6.0-92 lattice\_0.20-45   
## [5] ggplot2\_3.3.5 BART\_2.9 survival\_3.2-13 nnet\_7.3-16   
## [9] nlme\_3.1-153   
##   
## loaded via a namespace (and not attached):  
## [1] splines\_4.1.2 foreach\_1.5.2 prodlim\_2019.11.13   
## [4] assertthat\_0.2.1 posterior\_1.2.0 distributional\_0.3.0  
## [7] highr\_0.9 stats4\_4.1.2 tensorA\_0.36.2   
## [10] yaml\_2.2.2 globals\_0.14.0 ipred\_0.9-12   
## [13] pillar\_1.7.0 backports\_1.4.1 glue\_1.6.1   
## [16] pROC\_1.18.0 digest\_0.6.29 checkmate\_2.0.0   
## [19] hardhat\_0.2.0 colorspace\_2.0-2 recipes\_0.2.0   
## [22] htmltools\_0.5.2 Matrix\_1.3-4 plyr\_1.8.6   
## [25] timeDate\_3043.102 pkgconfig\_2.0.3 listenv\_0.8.0   
## [28] purrr\_0.3.4 scales\_1.1.1 gower\_1.0.0   
## [31] lava\_1.6.10 tibble\_3.1.6 generics\_0.1.2   
## [34] farver\_2.1.0 ellipsis\_0.3.2 withr\_2.4.3   
## [37] cli\_3.2.0 magrittr\_2.0.2 crayon\_1.5.0   
## [40] evaluate\_0.14 future\_1.23.0 fansi\_1.0.2   
## [43] parallelly\_1.30.0 MASS\_7.3-54 class\_7.3-19   
## [46] tools\_4.1.2 data.table\_1.14.2 matrixStats\_0.61.0   
## [49] lifecycle\_1.0.1 stringr\_1.4.0 munsell\_0.5.0   
## [52] compiler\_4.1.2 rlang\_1.0.1 grid\_4.1.2   
## [55] iterators\_1.0.14 rstudioapi\_0.13 rmarkdown\_2.11   
## [58] gtable\_0.3.0 ModelMetrics\_1.2.2.2 codetools\_0.2-18   
## [61] abind\_1.4-5 DBI\_1.1.2 reshape2\_1.4.4   
## [64] R6\_2.5.1 lubridate\_1.8.0 knitr\_1.37   
## [67] dplyr\_1.0.8 fastmap\_1.1.0 future.apply\_1.8.1   
## [70] utf8\_1.2.2 stringi\_1.7.6 parallel\_4.1.2   
## [73] Rcpp\_1.0.8 vctrs\_0.3.8 tidyselect\_1.1.1   
## [76] xfun\_0.29