Appendix A: Data Transparency

The data reported in this manuscript have been previously published and were collected as part of a larger data collection. Findings from the data collection have been reported in separate manuscripts. Other studies using MATCH data have looked at a large array of average treatment effects, including the specific hypothesized a priori interactions using the study's primary and secondary matching hypotheses and many of the other psychosocial processes that underlay alcohol misuse and treatment. The current study differs from and extends on previous research by using a personalized medicine approach to simultaneously test whether there is evidence for differential treatment effects when a large number of individual level variables are simultaneously used to test for differential effects of the treatment. We know of no previous study which has used a similar analytic approach to test for differences in the treatments used in the MATCH study.

Appendix B: Algorithm for Deriving PITE Predictions

```
#Import imputed CBT treatment group
cbt <- read.csv("M:/Project MATCH/mi_cbtimpFINAL.csv")

#select first imputed dataset
cbt <- cbt[which(cbt$.imp==1),]
colnames(cbt)

#Import imputed MET treatment group
met <- read.csv("M:/Project MATCH/mi_metimpFINAL.csv")

#select first imputed dataset
met <- met[which(met$.imp==1),]
colnames(met)

match <- rbind(cbt, met)                              #Combine datasets
dim(match)                                            #check # of rows and columns

match$txassgn<-match$txassgn-1            #Recode treatment variable as 0/1
table(match$txassgn)                                  ###met=1, cbt=0

#remove extraneous outcomes/non-baseline vars that had been used in
imputation model
match<-match[,c(-36,-37,-40:-43,-108)]

#remove extraneous vars created during imputation
match[,c(1:3)]<-NULL

###All changes saved in "M:/Project Match/Data/finalpitematch.csv"###
match<-read.csv("M:/Project Match/Data/finalpitematch.csv")

#######################Main Effect of Treatment##########################
library(lme4)

#Subset sample into outpatient and aftercare
match.out <- match[which(match$arm==1),]
match.aft <- match[which(match$arm==2),]

##For Table 2
#Outpatient
summary(m1<-glm(abstinent~Female + age + cogimp_CFS + angers_TAS + rbbtots +
socfuncs_PFI + selfef1C_AASE + selfef2T_AASE + idiscors + otcmi_Tot +
ydrkr0_Tot + ypror0_Tot +  asipsya_Tot + aai_int_Tot + ss2Famtot + ss1frietot
+ urica + typol + song_a, data=match.out.cbt, family="binomial"))
summary(m2<-glm(abstinent~Female + age + cogimp_CFS + angers_TAS + rbbtots +
socfuncs_PFI + selfef1C_AASE + selfef2T_AASE + idiscors + otcmi_Tot +
ydrkr0_Tot + ypror0_Tot +  asipsya_Tot + aai_int_Tot + ss2Famtot + ss1frietot
+ urica + typol + song_a, data=match.out.met, family="binomial"))
#Aftercare
summary(m3<-glm(abstinent~Female + age + cogimp_CFS + angers_TAS + rbbtots +
socfuncs_PFI + selfef1C_AASE + selfef2T_AASE + idiscors + otcmi_Tot +
ydrkr0_Tot + ypror0_Tot +  asipsya_Tot + aai_int_Tot + ss2Famtot + ss1frietot
+ urica + typol + song_a, data=match.aft.cbt, family="binomial"))
summary(m4<-glm(abstinent~Female + age + cogimp_CFS + angers_TAS + rbbtots +
socfuncs_PFI + selfef1C_AASE + selfef2T_AASE + idiscors + otcmi_Tot +
ydrkr0_Tot + ypror0_Tot +  asipsya_Tot + aai_int_Tot + ss2Famtot + ss1frietot
+ urica + typol + song_a, data=match.aft.met, family="binomial"))

###################PITE & PERMUTATION TEST, BY ARM########################
colnames(match.out)
out.cov<-match.out[,c("Female","age","cogimp_CFS","angers_TAS","rbbtots",
      #create matrix with baseline covariates among outpatient sample
      "socfuncs_PFI","selfef1C_AASE","selfef2T_AASE","idiscors",
      "otcmi_Tot","ydrkr0_Tot","ypror0_Tot","asipsya_Tot","aai_int_Tot",
      "ss2Famtot","ss1frietot","urica","song_a","typol")]

out.pite<-PiteBinary(match.out$txassgn,match.out$abstinent,out.cov)
      #PITE function
str(out.pite)
      #elements of results object
set.seed(9359503)
out.perm<-Pite.prmtYasin(out.pite,1000,responseType = 'binary')
      #Permutation test function using PITE results, 1000 permutations for
binary outcome
str(out.perm)
      #elements of permutation test results object
##p=0.024

match.aft <- match[which(match$arm==2),]
      #subset aftercare sample
colnames(match.aft)
aft.cov<-match.aft[,c("Female","age","cogimp_CFS","angers_TAS","rbbtots",
      #create matrix with baseline covariates among aftercare sample
      "socfuncs_PFI","selfef1C_AASE","selfef2T_AASE","idiscors",
      "otcmi_Tot","ydrkr0_Tot","ypror0_Tot","asipsya_Tot","aai_int_Tot",
      "ss2Famtot","ss1frietot","urica","song_a","typol")]

aft.pite<-PiteBinary(match.aft$txassgn,mat.aft$abstinent,aft.cov)
      #PITE function
str(aft.pite)
      #elements of results object
set.seed(9359223)
aft.perm<-Pite.prmtYasin(aft.pite,1000,responseType = 'binary')
      #Permutation test function using PITE results, 1000 permutations for
binary outcome
str(aft.perm)
      #elements of permutation test results object
##p=0.277



##**********************PITE PREDICT FUNCTION**************************##
##PITE function for use with binary outcome##
PiteBinary <- function(trt, y, cov, inter = F, extract='ATE',
covToInc='all'){
  n <- nrow(cov)
  NumCov <- dim(cov)[2]
  if (length(table(trt)) != 2)
    stop("The treatment level is not two.")
```

```
#Code is designed to test treatment vs. control group, if there are too many
levels in the designated treatment variable, function will stop and produce
this error

  if( inter == TRUE & n/2 <= (factorial(ncol(cov)) / (factorial(ncol(cov)-
2)*factorial(2)) + ncol(cov) +1) )
    stop("degrees of freedom of residual is negative, sample size (n) needs
to be larger ")
  if( inter == FALSE & n/2 <= (ncol(cov) +1) )
    stop("degrees of freedom of residual is negative, sample size (n) needs
to be larger
          than twice of the number of total covariates plus 1 (NumCov +
NumNuiCov +1)")

#Default is set to not consider interactive effects, but if set to
inter=TRUE, they will be included in the calculation of the PITEs,
#if sample size is not large enough for adequate degrees of freedom, this
error will be returned

  if(extract !='model' & extract !='ATE'){
    stop("extraction method is not ATE or model")
  }

#Default is set to extract the average treatment effect of the intervention,
so that PITEs are calculated as individual treatment effects
#above and beyond the average effect of the intervention

  if(covToInc !='all' & length(covToInc)> ncol(cov)){
    stop("Number of covariates to include is greater than given number of
covariates")     }


#============================================================================#

  # Set up data into required format
  data <- data.frame(y, trt)
  dataxy <- data.frame(y, cov)
  # create the interaction between pair predictors x, add in to the dataset
  # dataint now have y, treatment condition, x, and the pairwise interaction
between x (if applicable)
  options(na.action = 'na.pass')
  dataint2 <- model.matrix(y ~ .^2, data = dataxy)    #creates matrix of each
case's coefficients for baseline covariates and pairwise interactions
  dataint <- data.frame(y, trt, x = dataint2[,-1])    #creates data frame of
y, treatment condition, and covariate + pairwise interactions (minus
intercept)


  # split covariates based on treatment assignment
  xt <- dataint[dataint$trt ==1, 3:ncol(dataint)]
  xc <- dataint[dataint$trt ==0, 3:ncol(dataint)]

  # split responses based on treatment assignment and order
  yt <- y[dataint$trt == 1]
  yc <- y[dataint$trt == 0]
  y.ord <- c(yt,yc)
  # separate the dataset (dataint) to treatment group and control group
  TRTmxi <- data.frame(xt)
  CNTLmxi <- data.frame(xc)

  # assigns treatment labels for ordered data
  trt1<- c(rep(1,length(yt)), rep(0, length(yc)))

    if(extract == 'model'){
            if(covToInc =='all'){ # set up which covariates to use for this
model
                dataint1 <- dataint
              }
            else{
                cov1 <- cov[,covToInc]
                dataint2.1 <- model.matrix(y ~ .^2, data =data.frame(y,cov1))
                dataint1 <- data.frame(y, trt, x = dataint2.1[,-1])
            }

        # extract coefficients from the control model
         mod <- glm(dataint1[,1] ~., family =binomial(link = 'logit'), data =
dataint1[,-1])
        lambda <- coefficients(mod)[2]
    }

    else if(extract=='ATE'){
        # simple model
        mod <- glm(y.ord ~trt1, family =binomial(link = 'logit'))
        lambda <- coefficients(mod)[2]
    }

  # if the interaction of predictors are not taken into consideration
  if (inter == FALSE) {

    # fit model for trt group
    mod.t <- glm(yt ~ .,family =binomial(link = 'logit'), data =
TRTmxi[,1:(ncol(cov))])
    # fit model for control group
    mod.c <- glm(yc ~ .,family =binomial(link = 'logit'), data =
CNTLmxi[,1:(ncol(cov))])
    et <- coefficients(mod.t)
    ec <- coefficients(mod.c)
    cov <- as.matrix(cov)

    # predictions of response on logit scale
    pred_t <- et[1] + (cov%*%et[2:(length(et))])
    pred_c <- ec[1] + (cov%*%ec[2:(length(ec))])

    # predictions of responses on probability scale
    ppPred_t <- sapply(pred_t, logistic)
    ppPred_c <- sapply(pred_c, logistic)

    # FOR PERMUTATION TEST first on logit scale followed by probability
scale;
    predt.perm <- pred_t - lambda                     #subtract out average
treatment effect (ATE) from individual effects
    ppPredt.perm <- sapply(predt.perm, logistic)      #convert to probability
scale
```

```
  }

  # if the pairwise interaction of predictors are taken into consideration
  if (inter == TRUE) {

    mod.t <- glm(yt ~ .,family =binomial(link = 'logit'), data = TRTmxi) #
fit model for trt group
    mod.c <- glm(yc ~ .,family =binomial(link = 'logit'), data = CNTLmxi) #
fit model for control group
    et <- coefficients(mod.t)
    ec <- coefficients(mod.c)
    cov <- as.matrix(cov)
    # predictions of response on trt on logit scale
    pred_t <- et[1] + (dataint[,3:ncol(dataint)]%*%et[2:(length(et))])
    pred_c <- ec[1] + (dataint[,3:ncol(dataint)]%*%ec[2:(length(ec))])

    ppPred_t <- sapply(pred_t, logistic)   #predictions on probability scale
    ppPred_c <- sapply(pred_c, logistic)   #predictions on probability scale

    # FOR PERMUTATION TEST first on logit scale followed by probability
scale;
    predt.perm <- pred_t - lambda
    ppPredt.perm <- sapply(predt.perm, logistic)

  }

  # calculate the estimate: individual PITE estimate and the standard
deviation
  # of PITE estimates among individuals
  pite.ind <- ppPred_t - ppPred_c         #predicted response under treatment
condition - predicted response under control
  pite.sd <- sd(pite.ind)

  pite.ind.perm <-  ppPredt.perm - ppPred_c
  pite.sd.perm <- sd(pite.ind.perm)

  return(list("trt.coef" = et,
              "cntl.coef" = ec,
              "pite.trt.cond" = ppPred_t,
              "pite.cntl.cond" = ppPred_c,
              "pite.trt.PT" =ppPredt.perm,
              "pite.ind" = pite.ind,
              "pite.sd" = pite.sd,
              "pite.ind.perm" =pite.ind.perm,
              "pite.sd.perm" = pite.sd.perm,
              "TOTCov" = ncol(cov),
              "gen.data" = data,
              "gen.data.int" = dataint,
              "NumCov" = NumCov,
              "gen.xMain" = cov,
              "gen.xMainInt" = dataint[,3:ncol(dataint)],
              "interaction" = inter,
              "TOTCov" = ncol(cov)
  ))
}
```

```
#========================PERMUTATION FUNCTION============================#

Pite.prmtYasin <- function(pite.rslt, nperm, inter = F,
responseType='normal', extract = 'ATE', covToInc='all'){
  if(covToInc !='all' & length(covToInc)> ncol(pite.rslt$gen.xMain)){
    stop("Number of covariates to include is greater than given number of
covariates")     }
  if(responseType!='binary' & responseType !='normal'){
    stop("Response type is not normal or binary")
  }
  if(extract !='model' & extract !='ATE'){
    stop("extraction method is not ATE or model")
  }


  # use the output from 'pite.rslt' to know where the PITE SD lies for
original treatment labels
  # use output which has subtracted treatment effect
  pite.ind <- pite.rslt$pite.ind.perm
  pite.sd <- pite.rslt$pite.sd.perm
  trt <- pite.rslt$gen.data$trt # original treatment labels
  y <- pite.rslt$gen.data$y  # observed responses

  # setting up covariates if interactions F, only covariates, if T, include
interactions
  if (inter == F) {
    x <- pite.rslt$gen.xMain
  }

  if (inter == T) {
    x <- pite.rslt$gen.xMainInt
  }

  # set up a matrix with all permutations of treatment assignment.
  # each column corresponds to one set of permutated labels.
  trtperms <- replicate(n=nperm, sample(pite.rslt$gen.data$trt, replace = F))

 # dependent on response type, we apply the correct function: Normal or
Binary
  if( responseType == 'normal'){
    prmt <- sapply(apply(trtperms,2,PiteNormalPRMT,trt=trt,cov=x,y=y,
                         extract=extract, inter=inter, covToInc=covToInc),
'[[', 'pite.ind')
  }
#prmt=pites for each permutation
  else if(responseType == 'binary'){
    prmt <- sapply(apply(trtperms,2,PiteBinaryPRMT,trt=trt,cov=x,y=y,
                         extract=extract, inter=inter, covToInc=covToInc),
'[[', 'pite.ind')
  }

#calculating SD for each permutation
  prmt.sd <- apply(prmt, 2, sd)

#p value of permutation test calculated based on proportion of permuted SDs
that are greater than observed SD
#divided by total number of permutations
```

```
  alpha <-  sum(pite.sd <=prmt.sd)/nperm

# histogram of permuted SDs, line for SD of PITE in observed data (figure 2)
  hist(prmt.sd, xlim = c(min(pite.sd, range(prmt.sd)[1]) -.05, max(pite.sd,
range(prmt.sd)[2])+.05),
       main = (deparse(substitute(pite.rslt))), sub = paste0('p-value = ',
alpha))
  abline(v=pite.sd, col=2)

  return(list(
    "pite.ind" = pite.ind,
    "permute" = prmt,
    "pite.sd" = pite.sd,
    "prmt.sd" = prmt.sd,
    "alpha" = alpha
  ))
}


#=========== EXTRA FUNCTIONS REQUIRED FOR PERMUATION TEST ===============#
#This function randomly assigns treatment condition for as many times as
specified by nperm
#and then runs the original function to produce PITEs with these random
assignments
#Those results are collated and returned by Pite.prmtYasin

PiteBinaryPRMT <- function(trt1, trt, y, cov, inter = F, extract = 'ATE',
covToInc = 'all'){
  if(extract !='model' & extract !='ATE'){
    stop("extract method is not ATE or model")
  }
        n <- nrow(cov)
  NumCov <- dim(cov)[2]
  if (length(table(trt)) != 2)
    stop("The treatment level is not two.")
  if( inter == TRUE & n/2 <= (factorial(ncol(cov)) / (factorial(ncol(cov)-
2)*factorial(2)) + ncol(cov) +1) )
    stop("degrees of freedom of residual is negative, sample size (n) needs
to be larger ")
  if( inter == FALSE & n/2 <= (ncol(cov) +1) )
    stop("degrees of freedom of residual is negative, sample size (n) needs
to be larger than twice
         of the number of total covariates plus 1 (NumCov + NumNuiCov +1)")
  if(covToInc !='all' & length(covToInc)> ncol(cov)){
    stop("Number of covariates to include is greater than given number of
covariates")     }
#========================================================================#

  data <- data.frame(y, trt)
    dataxy <- data.frame(y, cov)

  # create the interaction between pair predictors x, add in to the dataset
  # dataint now have y, treatment condition, x, and the pairwise interaction
between x

  options(na.action = 'na.pass')
  dataint2 <- model.matrix(y ~ .^2, data = dataxy)
  dataint <- data.frame(y, trt, x = dataint2[,-1])

  # SPLIT DATASETS ACCORDING TO PERMUTED TREATMENT LABELS
##This is where the permutation test code differs from the PITE function
above
##Here, the dataset is being split up according to randomly /permuted/
treatment condition (trt1)
##Not according to observed treatment assignment (above as "trt")

  xt1 <- dataint[trt1 ==1, 3:ncol(dataint)] # covariates of treatment group
  xc1 <- dataint[trt1 ==0, 3:ncol(dataint)] # covariates of control group

  yt1 <- y[trt1 == 1]       #responses on treatment
  yc1 <- y[trt1 == 0]       #responses on control
  y.ord1 <- c(yt1,yc1)

  TRTmxi1 <- data.frame(xt1)
  CNTLmxi1 <- data.frame(xc1)

  if (extract == 'model'){
    if(covToInc =='all'){
      dataint1 <- dataint
    }
    else{
      cov1 <- cov[,covToInc]
      dataint2.1 <- model.matrix(y ~ .^2, data =data.frame(y,cov1))
      dataint1 <- data.frame(y, trt, x = dataint2.1[,-1])
    }

  # extract coefficients from the control model
  mod <- glm(dataint[,1] ~., family =binomial(link = 'logit'), data =
dataint[,-1])
  lambda <- coefficients(mod)[2]
  }
  else if(extract == 'ATE'){
  mod <- glm(dataint[,1] ~dataint[,2], family =binomial(link = 'logit'))
  lambda <- coefficients(mod)[2]
  }

  # finding those patients who are trt=1 in the group of people who are
trt1=0 and trt1=1
  offset.t <- trt1[trt==1]
  offset.c <- trt1[trt==0]

  # if the interaction of predictors are not taken into considertion
  if (inter == FALSE) {

    # set up model on responses from permuted treatment labels including
offset
    mod.t <- glm(yt1 ~ .,family =binomial(link = 'logit'),
                 data = TRTmxi1[,1:(ncol(cov))], offset = lambda*offset.t)
    mod.c <- glm(yc1 ~ .,family =binomial(link = 'logit'),
                 data = CNTLmxi1[,1:(ncol(cov))], offset = lambda *offset.c)
    et <- coefficients(mod.t)
    ec <- coefficients(mod.c)
    cov <- as.matrix(cov)
    # predictions of responses on logit scale
    xbeta_t <- et[1] + (cov%*%et[2:(length(et))])
    xbeta_c <- ec[1] + (cov%*%ec[2:(length(ec))])

    # predictions of response on probability scale
    ppPred_t <- sapply(xbeta_t, logistic)
    ppPred_c <- sapply(xbeta_c, logistic)

  }

  # if the pairwise interaction of predictors are taken into consideration
  if (inter == TRUE) {

    # set up model on responses from permuted treatment labels including
offset
    mod.t <- glm(yt1 ~ .,family =binomial(link = 'logit'), data = TRTmxi1,
                 offset = lambda*offset.t)
    mod.c <- glm(yc1 ~ .,family =binomial(link = 'logit'), data = CNTLmxi1,
                 offset = lambda*offset.c)
    et <- coefficients(mod.t)
    ec <- coefficients(mod.c)
    dataint <- as.matrix(dataint)
    # predictions of responses on logit scale
    xbeta_t <- et[1] + (dataint[,3:ncol(dataint)]%*%et[2:(length(et))])
    xbeta_c <- ec[1] + (dataint[,3:ncol(dataint)]%*%ec[2:(length(ec))])

    #predictions on probability scale
    ppPred_t <- sapply(xbeta_t, logistic)
    ppPred_c <- sapply(xbeta_c, logistic)

  }

  # calculate PITE estimate and the standard deviation of pite estimates
  pite.ind <- ppPred_t - ppPred_c
  pite.sd <- sd(pite.ind)


  return(list("pite.trt.cond" = ppPred_t,
              "pite.cntl.cond" = ppPred_c,
              "pite.ind" = pite.ind,
              "pite.sd" = pite.sd,
              "TOTCov" = ncol(cov),
              "gen.data" = data,
              "gen.data.int" = dataint,
              "NumCov" = NumCov,
              "gen.xMain" = cov,
              "gen.xMainInt" = dataint[,3:ncol(dataint)],
              "interaction" = inter,
              "TOTCov" = ncol(cov)
  ))
}

#Function for converting estimates on the logit scale to the probability
scale
logistic <- function(x){
  return(exp(x)/(1+exp(x)))
}
```