BIOS 740 - Computing HW 2

Andrew Walther

October 13, 2021

1 Introduction

Precision Medicine is a sub-field of machine learning and statistics that has received significantly increased attention and research efforts in recent years. The field hinges around the goal of using observed characteristics about patients and populations to develop data-driven individualized treatment rules that are generalizable and reproducible. These individualized treatment rules involve compiling relevant information about patients and utilizing that data, along with a selection of available treatments to select the appropriate treatment that maximizes an outcome of interest with a treatment rule. Furthermore, effective treatment rules (or regimes) can be applied to larger groups of patients to provide a beneficial outcome across the whole sample. In short, individualized treatment rules or regimes (ITRs) seek to use data to determine who to apply a treatment to and how to do so in order to achieve an optimized clinical outcome. In addition to single-stage treatment plans where only a single decision is made, multi-stage treatment plans can also be assessed via precision medicine methods. Techniques such as Q-learning[1][4], Backward outcome weighted learning[5], and list-based dynamic treatment regimes [4] can be utilized to obtain value estimates for specific treatment decisions as well as optimal treatment decision plans based on a set of patient information.

In this report, we consider a set of simulated data intended to represent data that may be obtained from patients in a clinic to then be used for developing individualized dynamic treatment regimes. The factors included in this data are noted below in Table 1. The simulation includes four covariates $(X_{11}, X_{12}, X_{21}, X_{22})$ that represent treatments at stages 1 and 2. These covariates are independent and follow standard normal distributions. Additionally, there are two treatments A_1 and A_2 that follow binomial distributions with probability $expit(x) = (1 + e^{-x})^{-1}$. There are two history components denoted by $H_1 = \{X_{11}, X_{12}\}$ and $H_2 = \{X_{21}, X_{22}\}$. Finally, the outcome of interest is Y, represented by (with $\epsilon \sim N(0, 1)$)

$$Y = \exp(X_{11}) + \exp(X_{21}) + X_{11}X_{21} + A_1(1 + X_{11}) + A_2(1 + X_{11} + X_{21} + A_1X_{21}) + \epsilon.$$

In this set of simulated data, there are $n = 1000$ observations and for the outcome Y, larger values are considered to be better, similar to a longer survival time for a patient being considered optimal.

Table 1: Simulation Data Description

Factor	Description
X_{11}	Stage 1 treatment 1
X_{12}	Stage 1 treatment 2
X_{21}	Stage 2 treatment 1
X_{22}	Stage 2 treatment 2
A_1	Treatment 1
A_2	Treatment 2
H_1	Stage 1 history (X_{11}, X_{12})
H_2	Stage 2 history $(X_{11}, X_{12}, A_1, X_{21}, X_{22})$

2 Diagnostics

Our simulated data contains 1000 total observations and every observation is a complete case (no missing entries for any factor). Additionally, every observation was assigned a binary outcome for each of the treatments A_1 and A_2 of 1 or 0 which roughly translates to "give treatment" or "do not give treatment".

For purposes or data pre-processing, we considered a histogram plot of the response variable Y. From the raw data, this distribution has apparent right skew. Due to this right-skew, we considered the possibility of performing a log-transformation on the response variable in order to obtain a distribution conforming better to normality. However, the specified simulated data resulted in a non-trivial number of negative-valued Y observations such that a log-transformation would not be possible on these observations ($n = 52$). For that purpose, log-transformed data would not be feasible considering we had the intention of retaining all of the data for our analysis in the 2-stage individualized treatment regime.

Additionally, to assess normality for both the response variable as is and after a log-transformation (omitting 52 observations), we carried out a Kolmogorov-Smirnov test for normality. For this test, H_0 : the distribution of $Y/\log(y)$ follows a normal distribution and H_A : the distribution of $Y/\log(y)$ does not follow a normal distribution. For both tests, we obtained p-values much smaller than the $\alpha = 0.05$ significance threshold so it is apparent that the response variable does not follow a normal distribution. This is unfortunate as some of the methods to be used have regression components where normality is an assumption. However, we will proceed with the raw simulated as is without any transformation applied for the sake of simplicity and continuity among other studies.

Finally, to investigate possible outsize factor importance by some individual factors in the simulated data, we trained an rPart model and found that the covariates X_{21} and X_{11} are the most important for predicting the outcome Y. Regardless, all covariates specified in the data will be included in the subsequent treatment regime analysis.

3 Methods

In this study, the individualized treatment rule (ITR) methods of interest in the two-stage decision setting are Q-learning, Backward outcome weighted learning, and List-based dynamic treatment regimes. In all cases, the ultimate goal is to select an optimal treatment regime, $\hat{d}^{opt}$, for each patient that is based on $\{\hat{d}_1^{opt}, \hat{d}_2^{opt}\}$, the optimal treatments at stages 1 and 2. From this optimal treatment protocol in the two-stage decision treatment plan, we want to estimate the value function of the treatment plan, represented by

$$\hat{V}\left(\hat{d}^{opt}\right) = \frac{\sum_{i=1}^{n} Y_i I\{A_{1i} = \hat{d}_1^{opt}\left(H_{1i}\right), A_{2i} = \hat{d}_2^{opt}\left(H_{2i}\right)\} / \left[P\left(A_{1i}|H_{1i}\right) P\left(A_{2i}|H_{2i}\right)\right]}{\sum_{i=1}^{n} I\{A_{1i} = \hat{d}_1^{opt}\left(H_{1i}\right), A_{2i} = \hat{d}_2^{opt}\left(H_{2i}\right)\} / \left[P\left(A_{1i}|H_{1i}\right) P\left(A_{2i}|H_{2i}\right)\right]}$$

In the value function, Y_i represents the outcome value from the data, A_{1i} is a selected treatment in stage 1, $\hat{d}_1^{opt}\left(H_{1i}\right)$ is the optimal treatment for the given history in stage 1, A_{2i} is a selected treatment in stage 2, and $\hat{d}_2^{opt}\left(H_{2i}\right)$ is the optimal treatment for the given history in stage 2. These parameters were all specified in the introduction. For each of the following methods, we utilize k-fold cross-validation with a typical number of 10 folds. Therefore, the total number of observations was split up into 10 subsets of 100 observations each. Then in each iteration of model evaluation, 9 of the 10 folds are utilized for training and the 10th is used for testing. There are 10 total iterations such that each fold is held out for testing/making predictions on each model one time. This methodology allows for cross validation and a reduction in generalization error.

3.1 Q-Learning

Q-learning has been a method primarily used just in randomized trials, but recently it has been extended to be useful for developing optimal dynamic treatment regimes for observational data as well[1]. In Q-learning, we begin estimation at the final stage of a given process and seek to determine an optimal treatment. Estimation is then carried out on preceding stages via a "pseudo-outcome" that is built on the foundational assumption that every treatment in the following stages (that has already been estimate) is the optimal treatment. The process of implementing Q-learning can be found in Section 2.2 of Moodie et al. but as a primer it begins by proposing a set of "Q-functions",

$$Q_2\left(H_2, A_2\right) = E\left[Y|H_2, A_2\right] \text{ and } Q_1\left(H_1, A_1\right) = E\left[\max Q_2\left(H_2, a_2\right)|H_1, A_1\right]$$

that are based on the known data. The optimal treatment for a given stage is subsequently found by maximizing the Q-function. Further optimization at each stage then yields a series of treatments over multiple stages that is considered to be the optimal dynamic treatment regime in the multi-decision setting. Estimation of the final stage is given by

$$\left(\hat{\beta}_2, \hat{\psi}_2\right) = \text{argmin}_{\beta_2, \psi_2} \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - Q_2\left(H_{2i}, A_{2i}; \beta_2, \psi_2\right)\right)^2.$$

Then the optimal rule for the final stage is given by

$$\hat{d}_2(h_2) = \text{argmax}_{a_2} Q_2\left(h_2, a_2; \hat{\beta}_2, \hat{\psi}_2\right).$$

This then provides a pseudo-outcome for the prior stage of

$$\hat{Y}_{1i} = \max_{a_2} Q_2\left(H_{2i}, a_2; \hat{\beta}_2, \hat{\psi}_2\right).$$

Subsequent estimation of prior stages proceeds in a similar fashion. In our implementation of Q-learning, we utilized the qLearn() function from the 'DynTxRegime'[2] R package. The main effects and constrasts models were both specified to be linear regression models due to the continuous nature of the Y output in the simulated data. Additionally, the responses were simply designated as Y and treatments A_1, A_2 were appropriately assigned to stages 1 and 2. Overall, Q-learning is relatively simple to implement in R, but it is necessary to properly specify the models that determine the Q-functions in order to achieve valid estimation.

3.2 Backward Outcome Weighted Learning (BOWL)

In backward outcome weighted learning, the optimal decision rule at a given stage t maximizes

$$E\left[\frac{\left(\sum_{j=t}^{T} R_j\right) \Pi_{j=t+1}^{T} I\left(A_j = d_j^*\left(H_j\right)\right)}{\Pi_{j=t}^{T} \pi_j\left(A_j, H_j\right)} I(A_t = d_t(H_t)) | H_t = h_t\right].$$

BOWL is a backward recursive procedure that estimates an optimal decision rule at future stages initially. Then the optimal decision rule at a current stage is estimated by only considering subjects who are known to have followed the optimal decision rule in subsequent stages. In doing so, BOWL only focuses on observations that follow the recommended plan and ignores those that deviate from recommended treatment regimes. The algorithm for implementation of BOWL estimation can be found in Section 2.3 of Zhao et al.[5]. Fisher consistency also confirms that the optimizer for BOWL over a population of data is indeed the optimal dynamic treatment regime as well[5].

Our implementation of BOWL utilized the bowl() function from the R package 'DynTxRegime'[2]. This relatively simple implementation required a constant propensity model with generalized linear model solver method (binomial) to predict a binary outcome for treatment inclusion/exclusion. In addition, treatments A_1, A_2 were appropriately assigned to specific bowl() functions related to estimation for stages 1 and 2, respectively. In both cases, the reward was denoted as the given outcome Y from the simulation data and a typical number 10 cross-validation folds was specified for the analysis. Finally, the specified regime simply included all of the given 'X' covariates from the data.

3.3 List-based dynamic treatment regime (LIST-DTR)

The list-based dynamic treatment regime is a method of estimation of an optimal treatment regime where a sequence of decision rules is related to a specific treatment recommendation for one or more stages of treatment[4]. These decision rules are all if-then statements where the "if" statement is a condition that can be satisfied or not satisfied and the "then" statement is an action to take provided the condition is satisfied. These sequential statements allow one rule to be considered and a possible action taken followed by the next rule if the prior condition is not satisfied, and in turn the prior action not taken. In essence, these decision lists follow a single branch of a decision tree where each individual rule is a branch off of the main branch. Therefore an action can be taken or we continue on to the next possible decision point. The specific framework to develop these decision lists can be found in Section 2 of Zhang et al 2018[4] as the necessary notation is far from concise. However, the overall goal is to develop a set of decision rules that are the optimal regime out of all the possible regimes or $\pi^{opt} \in \Pi$.

In our analysis, we implemented the list-based dynamic treatment regime decision rule estimator via the listdtr() function from the R package 'listdtr'[3]. This relatively simple function takes in inputs of the covariates of interest from the data as x, the treatments for each stage A_1, A_2, and the outcomes Y. We specified the number of folds to be used as 10 in order to keep with the convention that we noted earlier on. Additionally, the randomization seed was specified to be identical as the seed used for generating the simulation data. The output of this function is presented in both a list form and as a flow chart that directs a reader through each decision to find the appropriate action for a given set of covariates.

4 Results

Following the implementation and evaluation of each of proposed ITR estimator, we obtained a set of value estimates and their respective standard errors in Table 2. In addition, the estimated decision rule for the list-based dynamic treatment regime method is provided in Table 3. The outcome for this simulated data is simply denoted by Y, a unitless quanitity. Based on the nature of the methods tested, we only have an explicit value estimate and standard error for the Q-learning and BOWL methods. Of these, the BOWL method obtained an estimated value of 21.22, which is significantly larger than the estimated value of 5.43 for Q-learning. However, BOWL had a standard error about an order of magnitude larger than Q-learning where the standard errors are 0.514 compared to 0.045. Additionally, for Q-learning, an average of 522 observations were recommended to receive treatment in stage 1 and an average of 400 observations were recommended to receive treatment in stage 2. Furthermore, for Backward outcome weighted learning, an average of 638 observations were recommended to receive treatment in stage 2.

Table 2: Estimated Outcome Values

Method	Estimated Value $\hat{V}$	Standard Error of $\hat{V}$
Q-Learning	5.43	0.045
BOWL	21.22	0.514
LIST-DTR	NA	NA

When we consider the list-based dynamic treatment regime method, from the simulated data we obtain the series of treatment rules for stages 1 and 2 given in Table 3. The rules are to be read in sequential order until a condition is satisfied such that a treatment decision can be applied. Furthermore, the list-dtr method found that 174 observations had the recommended plan and optimal plan to be no treatment, 376 observations had the recommended plan of giving the treatment but the optimal plan of not giving treatment, and 450 observations where the recommended and optimal plans were both to give treatment. This comes out to about 63 percent of the time where the model recommendations and the optimal plan agreed.

Table 3: Estimated Dynamic Treatment Regime Rules

Stage	If (Condition)	Then (Action)
1	$X_{11} > -0.609$	Give treatment
1	$X_{21} > 0.267$	Give treatment
1	$X_{11} > -1.932$	Do not give treatment
1	$X_{21} > -0.461$	Give treatment
1	Else	Do not give treatment
2	$Y_1 \leq 10.823$	Do not give treatment
2	$X_{11} \leq 0.510$	Give treatment
2	$Y_1 \leq 11.815$	Do not give treatment
2	$Y_1 \leq 23.006$	Give treatment
2	$X_{11} \leq 1.673$	Do not give treatment
2	$X_{11} \leq 3.402$	Give treatment
2	Else	Give treatment

5 Discussion of Advantages & Disadvantages

1. Q-learning

   Q-learning is a regression-based estimation method that utilizes a technique known as reinforcement learning to estimate optimal treatment regimes. Some strong aspects of Q-learning include the fact that it is appealing due to its ease of implementation[1]. I can concur with this statement by Moodie et al. as I found the R Q-learning method quite simple to implement and execute. In addition, Q-learning also allows for flexible specifications such as splines to be made for modeling when relationships in the data are not thoroughly understood. The paper by Moodie et al. also confirmed that Q-learning can be used for observational data in addition to randomized trials. On the other hand, Q-learning can have serious negative outcomes in terms of bias and poor coverage if the Q-functions are incorrectly specified[1]. In addition, true Q-functions are never explicitly known so we are forced to make educated guesses and hope that it doesn't seriously hinder the performance of the estimator. Finally, this method ended up returning and significantly smaller value estimate relative to the BOWL method so it may not have been quite as effective at determining the optimal treatment regime for the simulation data in this case.

2. Backward outcome weighted learning

   Backward outcome weighted learning (BOWL) was developed explicitly as an alternative to a regression-based dynamic treatment regime estimation algorithm (such as Q-learning) and is therefore a type of policy learning-based method. In fact, BOWL lays out the task of estimation of a DTR as a sequence of weighted classification problems[5]. One of the strong advantages of BOWL, as noted by both Zhao et al.[5] and our own results is that BOWL is capable of providing a better estimated value for the dynamic treatment regime relative to Q-learning. Additionally, relative to Q-learning, BOWL eliminates the need to properly specify "guesses" for Q-functions and rather considers a series of classification problems to identify the DTR. Not needing to be concerned about possible model mis-specification is a positive aspect of BOWL. On the other hand, while BOWL achieved a greater value estimate relative to Q-learning, it also had more variability among its estimates as shown by a larger standard error between cross-validation folds. Additionally, the run-time of BOWL is significantly longer than Q-learning as multiple folds are considered for each individual evaluation of the model. Finally, another drawback of BOWL is that the number of observations used for learning decision rules decreases geometrically as the considered stage number approaches the first possible stage.

3. List-based dynamic treatment regime

   List-based dynamic treatment regimes are quite unique from the prior two methods that we investigated and are a policy learning-based method. Regardless, they have their own set of advantages and drawbacks to be discussed as well. For one, list-based decision rules are incredibly simple to interpret once the list result is obtained. They are also able to be consumed as either a paragraph/list form or as a flow chart. Their simplicity and logical nature allows for thoughtful discussion and consideration among stakeholders that are not directly involved with analysis[4]. In addition, after proper specification of the model, it is quite easy to obtain predictions for which treatment series should be administered to a particular patient or group of patients. These lists are also simple but sufficient as to determine valuable treatment regimes. In general, this method is also advantageous as a means of avoiding the possibility of incorrectly specifying a model that is necessary for accurate estimation with other methods. On the other hand, the algorithm that "computes" decision lists for LIST-DTRs is quite complicated to understand so it is not easy to get a grasp of why certain decision rules are determined, but simply that the decision rules should be followed. Furthermore, as a contrast from many other dynamic treatment regime estimation methods, LIST-DTRs do not provide an explicit value estimate. In fact, they just provide decision rules based on the covariates and the option to predict optimal treatments based on that set of rules.

References

[1] Erica EM Moodie, Bibhas Chakraborty, and Michael S Kramer. Q-learning for estimating optimal dynamic treatment rules from observational data. *Canadian Journal of Statistics*, 40(4):629–645, 2012.

[2] K. A. Linn B. Zhang M. Davidian S. T. Holloway, E. B. Laber and Maintainer Shannon T. Holloway A. A. Tsiatis. Package 'dyntxregime'. *R software*, 2020.

[3] Yichi Zhang. Package 'listdtr'. *R software*, 2021.

[4] Yichi Zhang, Eric B Laber, Marie Davidian, and Anastasios A Tsiatis. Interpretable dynamic treatment regimes. *Journal of the American Statistical Association*, 113(524):1541–1549, 2018.

[5] Ying-Qi Zhao, Donglin Zeng, Eric B Laber, and Michael R Kosorok. New statistical learning methods for estimating optimal dynamic treatment regimes. *Journal of the American Statistical Association*, 110(510):583–598, 2015.

```
###BIOS 740 Computing HW 2 Code - Andrew Walther

##Packages
#data manipulation/visualizations
library(tidyverse)
#summary statistics
library(mosaic)
#cross-validation
library(caret)
#ML method implementation
library(DynTxRegime)
library(listdtr)
##simulation dataset
#set a seed for reproducibility
set.seed(2019)
#number of samples
n <- 1000
#expit function
expit <- function(x) {1 / (1 + exp(-x))}
#covariates for stage 1
X11 <- rnorm(n)
X12 <- rnorm(n)
#covariates for stage 2
X21 <- rnorm(n)
X22 <- rnorm(n)
#treatment 1 & 2
A1 <- rbinom(n, 1, expit(X11))
A2 <- rbinom(n, 1, expit(X21))
#gamma components of outcome
gamma1 <- A1 * (1 + X11)
gamma2 <- A2 * (1 + X11 + X21 + A1 * X21)
#outcome
Y <- exp(X11) + exp(X21) + X11 * X21 + gamma1 + gamma2 + rnorm(n)
#build dataset and view first few observations
dat = cbind(X11, X12, X21, X22, A1, A2, Y) %>% as.data.frame()
#######################################################################
##Data Processing
#add column for log transform of Y
dat_log <- dat %>% mutate(log_Y = log(Y))
#Create 10 folds of the dataset and add to TEST/TRAIN lists
folds <- createFolds(dat$Y, 10)
TEST_DATA <- list()
TRAIN_DATA <- list()
for(i in 1:10){
        TEST_DATA[[i]] <- dat[folds[[i]],]
        TRAIN_DATA[[i]] <- dat[-folds[[i]],]}
#######################################################################
##EDA & Feature Selection
#summary stats for Y
favstats(dat$Y)
#summary stats for log(Y)
favstats(dat_log$log_Y)
#hist of Y (right skew is present in the data, some negative values)
```

```
dat %>% ggplot() + geom_histogram(aes(x=dat$Y),binwidth = .1) + labs(title
= "Histogram of Y outcome",x = "Y outcome")
#hist of log(Y) (now there's left skew in the data, must omit NAs from
negative Y's)
dat_log %>% ggplot() + geom_histogram(aes(x=log(Y)),binwidth = 0.1) +
labs(title = "Histogram of log Y",x = "Y outcome (log)")
##K-S test for normality (H0: data follows Normal distribution)
#raw data: not normally distributed
ks.test(dat$Y, "pnorm", mean=mean(dat$Y), sd=sd(dat$Y))
#log-transform: still not normally distributed (remove NA/negative values)
dat_log_reduced <- dat_log[complete.cases(dat_log),]
ks.test(dat_log_reduced$log_Y, "pnorm", mean=mean(dat_log_reduced$log_Y),
sd=sd(dat_log_reduced$log_Y))
##Feature Selection
#Train RPart Model to compute variable importance
rPartMod <- train(Y~ ., data=dat, method="rpart")
rpartImp <- varImp(rPartMod)
print(rpartImp)
#use all factors since n>>p (X11, X21, A1 seem to be influential)
########################################################################
##Q-learning
#Initialize lists to hold value/opt trt predictions
Q.estimates <- list()
Q.opt.Tx <- list()
#loop over 10 data folds with Q-learning method
for(i in 1:10){
        #Outcome Model (linear model method for continuous Y, all
covariates)
        moMain <- buildModelObj(model = ~X11+X12+X21+X22,
                                solver.method = 'lm')
        moCont <- buildModelObj(model = ~X11+X12+X21+X22,
                                solver.method = 'lm')
        #2nd Stage Analysis (A2 is 2nd stage treatment)
        fitSS <- qLearn(moMain = moMain, moCont = moCont,
                        data = TRAIN_DATA[[i]],
                        response = TRAIN_DATA[[i]]$Y,  txName = 'A2')
        #outcome model
        moMain <- buildModelObj(model = ~X11+X12+X21+X22,
                                solver.method = 'lm')
        moCont <- buildModelObj(model = ~X11+X12+X21+X22,
                                solver.method = 'lm')
        fitFS <- qLearn(moMain = moMain, moCont = moCont,
                        data = TRAIN_DATA[[i]],
                        response = fitSS,  txName = 'A1')
        #extract value & optimal treatment predictions for each fold
        Q.estimates[[i]] <- as.numeric(estimator(fitFS))
        Q.opt.Tx[[i]] <- optTx(fitFS, newdata = TEST_DATA[[i]])}
#compute mean & sd for each fold prediction mean
mean.Q <- mean(as.numeric(Q.estimates))
se.Q <- sd(as.numeric(Q.estimates))
#count patients assigned to receive treatment vs. not
trt1.total <- 0
trt0.total <- 0
for(i in 1:10){
        trt1 <- count(Q.opt.Tx[[i]]$optimalTx == 1)
        trt0 <- count(Q.opt.Tx[[i]]$optimalTx == 0)
        trt1.total <- trt1.total + trt1
        trt0.total <- trt0.total + trt0}
#proportion of patients assigned to get treatment
print(trt1.total/(trt1.total+trt0.total))
#######################################################################
##BOWL
#Initialize lists to hold value/opt trt predictions
BOWL.estimates <- list()
BOWL.opt.Tx <- list()
#loop over 10 data folds with BOWL method
for(i in 1:10){
        #2nd Stage Regression - Constant propensity model
        moPropen <- buildModelObj(model = ~ 1,solver.method = 'glm',
                                  solver.args = list('family'='binomial'),
                                  predict.method = 'predict.glm',
                                  predict.args = list(type='response'))
        fitSS <- bowl(moPropen = moPropen,
                      data = TRAIN_DATA[[i]], reward = TRAIN_DATA[[i]]$Y,
txName = 'A2',
                      regime = ~ X11+X12+X21+X22)
        #1st Stage Regression - Constant Propensity model
        fitFS <- bowl(moPropen = moPropen,
                      data = TRAIN_DATA[[i]], reward = TRAIN_DATA[[i]]$Y,
txName = 'A1',
                      regime = ~ X11+X12+X21+X22,
                      BOWLObj = fitSS, lambdas = c(0.5, 1.0), cvFolds = 10)
        #extract value & optimal treatment predictions for each fold
        BOWL.estimates[[i]] <- as.numeric(estimator(fitFS))
        BOWL.opt.Tx[[i]] <- optTx(fitFS, newdata = TEST_DATA[[i]])}
#compute mean & sd for each fold prediction mean
mean.BOWL <- mean(as.numeric(BOWL.estimates))
se.BOWL <- sd(as.numeric(BOWL.estimates))
#count patients assigned to receive treatment vs. not
trt1.total <- 0
trt0.total <- 0
for(i in 1:10){
        trt1 <- count(BOWL.opt.Tx[[i]]$optimalTx == 1)
        trt0 <- count(BOWL.opt.Tx[[i]]$optimalTx == 0)
        trt1.total <- trt1.total + trt1
        trt0.total <- trt0.total + trt0}
#proportion of patients to receive treatment
print(trt1.total/(trt1.total+trt0.total))
#######################################################################
##List-DTR
#feature covariates to base decision rules on
x <- cbind(dat$X11,dat$X12,dat$X21,dat$X22)
stage.x <- rep(1,4)
#Treatments (stage 1 & 2)
a1 <- dat$A1
a2 <- dat$A2
#Outcomes (stage 1 & 2)
y1 <- dat$Y
```

```
y2 <- dat$Y
#dynamic treatment regime function (10 folds)
dtr <- listdtr(cbind(y1,y2), cbind(a1,a2), x, stage.x, kfolds = 10, seed = 
2019)
#print decision rule If/Then & plot flow chart of decision rules
print(dtr)
plot(dtr)
#predictions for dtr (recommended and optimal)
yrec <- predict(dtr, x, 1)
yopt <- ifelse(x[,1] > 0, 1, 0)
table(yrec, yopt)
#######################################################################
```