DynTxRegime

An R Package for Dynamic Treatment Regimes

Shannon Holloway

DPHS, Duke University

February 6, 2025

Outline

Acknowledgements

Innovative Methods Program for Advancing Clinical Trials

- A joint venture of Duke, UNC-Chapel Hill, and NC State
- Supported by NCI Program Project P01 CA142538 (2010-2021)

- Statistical methods for precision cancer medicine

Goal

DynTxRegime was envisioned to be a toolkit for all things related to estimating DTRs

Guiding principles:

- User-friendly
  straightforward and consistent inputs and post-processing
- General
  minimal artificial limitation of coding choices
- Expandable design
  task oriented design (S4 dominated structure)

Goal

DynTxRegime was envisioned to be a toolkit for all things related to estimating DTRs

Guiding principles:

- User-friendly
  straightforward and consistent inputs and post-processing
- General
  minimal artificial limitation of coding choices
- Expandable design
  task oriented design (S4 dominated structure)
- Efficient?
  generalization first priority

Key Challenge

Existing DTR packages

- made limiting assumptions about the types of models
- difficult to extend to > 2 points

How to allow for general specification of multiple regression steps?

Key Challenge

Existing DTR packages

- made limiting assumptions about the types of models
- difficult to extend to > 2 points

How to allow for general specification of multiple regression steps?

Our solution – the **modelObj** package.

We start here

Key Challenge

Existing DTR packages

- made limiting assumptions about the types of models
- difficult to extend to > 2 points

How to allow for general specification of multiple regression steps?

Our solution – the **modelObj** package.

We start here

Do you have the package installed?

```
library(package = modelObj)
```

Outline

What is it?

- **modelObj** encapsulates the regression steps as modeling object
- Allows for simple "plug and play" structure for new methods
- Shifts the regression from method implementation $\rightarrow$ user input

What is it?

- **modelObj** encapsulates the regression steps as modeling object
- Allows for simple "plug and play" structure for new methods
- Shifts the regression from method implementation $\rightarrow$ user input
- A Class with state variables and behaviors
  - State variables define the
    - model,
    - method for obtaining parameter estimates, and
    - method for obtaining predictions
  - Behavior
    - 'obtain parameter estimates'
    - 'make predictions'

What is it?

- **modelObj** encapsulates the regression steps as modeling object
- Allows for simple "plug and play" structure for new methods
- Shifts the regression from method implementation $\rightarrow$ user input
- A Class with state variables and behaviors
  - State variables define the
    - model,
    - method for obtaining parameter estimates, and
    - method for obtaining predictions
  - Behavior
    - 'obtain parameter estimates'
    - 'make predictions'
- Users have complete control over all modeling steps.

Modeling object created through **buildModelObj()**

```
str(object = buildModelObj)
```

```
function (model, solver.method = NULL, solver.args = NULL,
    predict.method = NULL, predict.args = NULL)
```

Modeling object created through **buildModelObj()**

```
str(object = buildModelObj)
```

```
function (model, solver.method = NULL, solver.args = NULL,
    predict.method = NULL, predict.args = NULL)
```

- **model**: "formula"; the model
  - Standard R "formula" object $y \sim x$
  - Left-hand-side variable ignored; can (should) be omitted
    - response variable is input varianle of behavior `fit()`

Modeling object created through **buildModelObj()**

```
str(object = buildModelObj)
```

```
function (model, solver.method = NULL, solver.args = NULL,
    predict.method = NULL, predict.args = NULL)
```

- **solver.method**: "character"; name of the R function to be used to obtain parameter estimates
  - 'lm', 'glm', 'nls'
- **solver.args**: "list"; additional arguments for solver.method

Modeling object created through **buildModelObj()**

```
str(object = buildModelObj)
```

```
function (model, solver.method = NULL, solver.args = NULL,
    predict.method = NULL, predict.args = NULL)
```

- **solver.method**: "character"; name of the R function to be used to obtain parameter estimates
  - 'lm', 'glm', 'nls'
- **solver.args**: "list"; additional arguments for solver.method

```
str(object = glm)
```

```
function (formula, family = gaussian, data, weights,
    subset, na.action, start = NULL, etastart, mustart,
    offset, control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, singular.ok = TRUE, contrasts = NULL,
    ...)
```

```
solver.args = list(family = "binomial")
```

Modeling object created through **buildModelObj()**

```
str(object = buildModelObj)
```

```
function (model, solver.method = NULL, solver.args = NULL,
    predict.method = NULL, predict.args = NULL)
```

---

predict.method: "character"; name of the **R** function to be used to obtain predictions

- 'predict.lm', 'predict.glm', 'predict'

predict.args: "list"; additional arguments for **predict.method**

Modeling object created through **buildModelObj()**

```
str(object = buildModelObj)
```

```
function (model, solver.method = NULL, solver.args = NULL,
    predict.method = NULL, predict.args = NULL)
```

---

predict.method: "character"; name of the **R** function to be used to obtain predictions

- 'predict.lm', 'predict.glm', 'predict'

predict.args: "list"; additional arguments for **predict.method**

```
str(object = predict.glm)
```

```
function (object, newdata = NULL, type = c("link",
    "response", "terms"), se.fit = FALSE, dispersion = NULL,
    terms = NULL, na.action = na.pass, ...)
```

```
predict.args = list(type = "response")
```

Example 1

Assume we have covariates (x_1, x_2, x_3) and a continuous response variable y

Example 1

Assume we have covariates (x_1, x_2, x_3) and a continuous response variable y

- Postulate a model $y \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$

Example 1

Assume we have covariates (x_1, x_2, x_3) and a continuous response variable y

- Postulate a model $y \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$
- Specify the method to obtain parameter estimate
  - Ordinary least squares ('lm')
  - Maximum likelihood estimation ('glm')
  - Elastic net regression ('glmnet')
  - ...

Example 1

Assume we have covariates (x_1, x_2, x_3) and a continuous response variable y

- Postulate a model $y \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$
- Specify the method to obtain parameter estimate
  - Ordinary least squares ('lm')
  - Maximum likelihood estimation ('glm')
  - Elastic net regression ('glmnet')
  - ...
- Specify the method to obtain predictions:
  - predict.lm(), predict.glm()
  - Most tools are designed to dispatch **predict()**

Example 1

Assume we have covariates (x_1, x_2, x_3) and a continuous response variable y

- Postulate a model $y \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$
- Specify the method to obtain parameter estimate
  - Ordinary least squares ('lm')
  - Maximum likelihood estimation ('glm')
  - Elastic net regression ('glmnet')
  - ...
- Specify the method to obtain predictions:
  - predict.lm(), predict.glm()
  - Most tools are designed to dispatch **predict()**

Construct modeling object assuming OLS using `lm`...

Example 1

Assume we have covariates (x_1, x_2, x_3) and a continuous response variable y

- Postulate a model $y \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$
- Specify the method to obtain parameter estimate
  - Ordinary least squares ('lm')
  - Maximum likelihood estimation ('glm')
  - Elastic net regression ('glmnet')
  - ...
- Specify the method to obtain predictions:
  - predict.lm(), predict.glm()
  - Most tools are designed to dispatch **predict()**

Construct modeling object assuming OLS using `lm`...

```
buildModelObj(model = ~ x1 + x2 + x3,
              solver.method = "lm",
              predict.method = "predict.lm")
```

Example 1

Assume we have covariates (x_1, x_2, x_3) and a continuous response variable y

- Postulate a model $y \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$
- Specify the method to obtain parameter estimate
  - Ordinary least squares ('lm')
  - Maximum likelihood estimation ('glm')
  - Elastic net regression ('glmnet')
  - ...
- Specify the method to obtain predictions:
  - predict.lm(), predict.glm()
  - Most tools are designed to dispatch **predict()**

Construct modeling object assuming OLS using `lm`...

```
buildModelObj(model = ~ x1 + x2 + x3,
              solver.method = "lm",
              predict.method = "predict.lm")
```

```
buildModelObj(model = ~ x1 + x2 + x3,
              solver.method = "lm")
```

Example 2

If outcome is binary

- Postulate a logistic regression model

$$\text{logit}(y) \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$$

- Specify the method to obtain parameter estimate
  - Maximum likelihood estimation ('glm')
  - Elastic net regression ('glmnet')
  - ...

If we choose `glm`, how would this model object differ from the previous?

```
previous <- buildModelObj(model = ~ x1 + x2 + x3,
                          solver.method = "lm")
```

Example 2

If outcome is binary

- Postulate a logistic regression model

$$\text{logit}(y) \sim \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + \beta_3\ x_3$$

- Specify the method to obtain parameter estimate
  - Maximum likelihood estimation ('glm')
  - Elastic net regression ('glmnet')
  - ...

If we choose `glm`, how would this model object differ from the previous?

```
previous <- buildModelObj(model = ~ x1 + x2 + x3,
                          solver.method = "lm")
```

```
buildModelObj(model = ~ x1 + x2 + x3,
              solver.method = "glm",
              solver.args = list(family = "binomial"))
```

There's a lot more . . .

We've glossed over a few details. For the applications in this class, they are not important.

For the record:

- **solver.method** must have a corresponding **predict.method**
  It does not have to have **coef()**, **residuals()**, **plot()**, etc.
- Function specified in **solver.method** must take
  - A "formula" object and a "data.frame" object; OR
  - A design matrix (X) and a response vector (Y)
- Function specified by **predict.method** must take the regression value object and a "data.frame" object
- Can define your own regression and predictions methods.

Requirements for modelObj in DynTxRegime

Propensity Regression

"modelObj" must specify that predictions are returned on the scale of the probability, i.e., in the interval (0,1)

For **glm()**/predict.glm(), default is scale of linear predictors

```
str(object = predict.glm)
```

```
function (object, newdata = NULL, type = c("link",
    "response", "terms"), se.fit = FALSE, dispersion = NULL,
    terms = NULL, na.action = na.pass, ...)
```

```
buildModelObj(model = ~ x1 + x2 + x3,
              solver.method = "glm",
              solver.args = list(family = "binomial"),
              predict.args = list(type = "response"))
```

Outcome Regression

"modelObj" must specify that predictions are returned on the scale of the response

For **lm()**/predict.lm(), default is scale of the response

```
str(object = predict.lm)
```

```
function (object, newdata, se.fit = FALSE, scale = NULL,
    df = Inf, interval = c("none", "confidence", "prediction"),
    level = 0.95, type = c("response", "terms"), terms = NULL,
    na.action = na.pass, pred.var = res.var/weights,
    weights = 1, rankdeficient = c("warnif", "simple",
        "non-estim", "NA", "NAwarn"), tol = 1e-06,
    verbose = FALSE, ...)
```

Outcome Regression

Specified using two "modelObj" objects

$$y \sim \mu + A\ C$$

- Specify μ as a "modelObj" object and/or C as a "modelObj" object
- Can specify different model structures for each component
  - a linear model for μ and a non-linear model for C
- μ and C are combined into a single model if possible, iterative procedure used otherwise

Responsible Model Selection

WARNING!!

DynTxRegime and **modelObj** do **NOT** check for appropriateness of models

Responsible model selection is the **responsibility of the *user***

Outline

https://github.com/sth1402/ST790/ Download st790Data.txt

Load Data

```
df <- read.csv(file = "path/st790Data.txt", header = TRUE)
```

Summary Statistics

```
summary(object = df)
```

```
       x1                  x2               x3
 Min.   :-3.40000   Min.   :0.000   Min.   : 0.640
 1st Qu.:-0.67000   1st Qu.:0.000   1st Qu.: 7.997
 Median :-0.04000   Median :0.000   Median :10.120
 Mean   :-0.02656   Mean   :0.306   Mean   :10.072
 3rd Qu.: 0.62000   3rd Qu.:1.000   3rd Qu.:12.105
 Max.   : 3.20000   Max.   :1.000   Max.   :19.500
       A                  y
 Min.   :0.000   Min.   :-5.0600
 1st Qu.:0.000   1st Qu.:-1.3100
 Median :0.000   Median :-0.3000
 Mean   :0.374   Mean   :-0.3138
 3rd Qu.:1.000   3rd Qu.: 0.7025
 Max.   :1.000   Max.   : 3.9900
```

Outline

Ready?

Do you have the package installed?

```
library(package = DynTxRegime)
```

Outcome Regression (Q-Learning)

Outcome Regression (Q-Learning)

Recall:

Estimator for d^{opt}

$$\widehat{d}_Q^{opt}(h) = \arg\max_{a \in \mathcal{A}} Q(h, a; \widehat{\beta}),$$

where

$$E(Y|H = h, A = a) = Q(h, a)$$

and $Q(h, a; \beta)$ is the posited model for $Q(h, a)$.

Estimator for the value $\mathcal{V}(d^{opt}) = E\{Y^*(d^{opt})\}$

$$\widehat{\mathcal{V}}_Q(d^{opt}) = \frac{1}{n} \sum_{i=1}^{n} \max_{a \in \mathcal{A}} Q(H_i, a; \widehat{\beta})$$

Both estimators are obtained using `qLearn()`

```
str(object = qLearn)
```

```
function (..., moMain, moCont, data, response, txName,
    fSet = NULL, iter = 0L, verbose = TRUE)
```

Argument	Class	Description
...		Ignored. Included to require named input.
moMain	modelObj	for main effects terms (mu)
moCont	modelObj	for terms interacting with treatment (C)
data	data.frame	covariates and treatment history
response	vector	outcome of interest
txName	character	treatment variable name
fSet		not used in single stage analyses
iter	integer	if >0, iterative methods used
verbose	logical	if FALSE, screen prints suppressed

```
str(object = qLearn)
```

```
function (..., moMain, moCont, data, response, txName,
    fSet = NULL, iter = 0L, verbose = TRUE)
```

Postulate a model for $Q(h, a)$. Recall, $\sim \mu + AC$

$$Q(h, a; \beta) = \overbrace{\beta_0 + \beta_1\ x_1 + \beta_2\ x_2}^{\mu} + A\{\ \underbrace{\beta_3 + \beta_4\ x_2 + \beta_5\ x_3}_{C}\ \}$$

Use OLS with a single model

Try to create the "modelObj".

```
str(object = qLearn)
```

```
function (..., moMain, moCont, data, response, txName,
    fSet = NULL, iter = 0L, verbose = TRUE)
```

Postulate a model for $Q(h, a)$. Recall, $\sim \mu + AC$

$$Q(h, a; \beta) = \overbrace{\beta_0 + \beta_1\ x_1 + \beta_2\ x_2}^{\mu} + A\{\ \underbrace{\beta_3 + \beta_4\ x_2 + \beta_5\ x_3}_{C}\ \}$$

Use OLS with a single model

Try to create the "modelObj".

```
moMain <- buildModelObj(model = ~ x1 + x2,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~ x2 + x3,
                        solver.method = 'lm')
```

Should we specify the iterative algorithm (`iter` > 0)?

```
str(object = qLearn)
```

```
function (..., moMain, moCont, data, response, txName,
    fSet = NULL, iter = 0L, verbose = TRUE)
```

- **data** is a complete dataset, (df)
- **response** is the "vector" outcome of interest (df$y)
- **txName** is the treatment variable name, ("A")
- **fSet**, **iter**, and **verbose** will stay their default values

```
moMain <- buildModelObj(model = ~ x1 + x2,
                        solver.method = 'lm')

moCont <- buildModelObj(model = ~ x2 + x3,
                        solver.method = 'lm')

qObj <- qLearn(moMain = moMain, moCont = moCont,
               data = df, response = df$y, txName = 'A')
```

```
qObj <- qLearn(moMain = moMain, moCont = moCont,
               data = df, response = df$y, txName = 'A')
```

```
First step of the Q-Learning Algorithm.

Outcome regression.
Combined outcome regression model: ~ x1+x2 + A + A:(x2+x3) .
Regression analysis for Combined:

Call:
lm(formula = YinternalY ~ x1 + x2 + A + x2:A + A:x3, data = data)

Coefficients:
(Intercept)           x1           x2            A
   -0.29653      0.82166      0.38095      1.28550
       x2:A         A:x3
   -0.02217     -0.13897


Recommended Treatments:
  0   1
617 383

Estimated value: -0.08377312
```

Methods available:

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`optTx(object)`	retrieve recommended optimal treatments for training data
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`summary(object)`	retrieve regression summaries

Methods available: Model Diagnostics

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`optTx(object)`	retrieve recommended optimal treatments for training data
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`summary(object)`	retrieve regression summaries

```
coef(object = qObj)
```

```
$outcome
$outcome$Combined
(Intercept)          x1          x2           A        x2:A
-0.29653461  0.82165693  0.38094720  1.28549787 -0.02217292
       A:x3
-0.13897217
```

- The returned object is a list
  - The coefficient are for the 'outcome' model
  - 'Combined' = moMain + A:moCont
  - Iterative procedure returns element 'moMain' and 'moCont' individually

```
fitObj <- fitObject(object = qObj)
fitObj
```

```
$outcome
$outcome$Combined

Call:
lm(formula = YinternalY ~ x1 + x2 + A + x2:A + A:x3, data = data)

Coefficients:
(Intercept)           x1           x2            A
   -0.29653      0.82166      0.38095      1.28550
       x2:A          A:x3
   -0.02217     -0.13897
```

```
fitObj <- fitObject(object = qObj)
fitObj
```

```
$outcome
$outcome$Combined

Call:
lm(formula = YinternalY ~ x1 + x2 + A + x2:A + A:x3, data = data)

Coefficients:
(Intercept)           x1           x2            A
   -0.29653      0.82166      0.38095      1.28550
       x2:A          A:x3
   -0.02217     -0.13897
```

```
is(object = fitObj$outcome$Combined)
```

```
[1] "lm"       "oldClass"
```

```
utils::methods(class = is(object = fitObj$outcome$Combined)[1L])[1:10]
```

```
 [1] "add1.lm"                      "alias.lm"
 [3] "anova.lm"                     "case.names.lm"
 [5] "coerce,oldClass,S3-method"    "confint.lm"
 [7] "cooks.distance.lm"            "deviance.lm"
 [9] "dfbeta.lm"                    "dfbetas.lm"
```

```
summary(object = qObj)
```

```
$outcome
$outcome$Combined

Call:
lm(formula = YinternalY ~ x1 + x2 + A + x2:A + A:x3, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-3.8193 -0.8168  0.0365  0.8045  3.2639

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.29653    0.06155  -4.818 1.68e-06 ***
x1           0.82166    0.04714  17.429  < 2e-16 ***
x2           0.38095    0.09932   3.836 0.000133 ***
A            1.28550    0.32231   3.988 7.14e-05 ***
x2:A        -0.02217    0.16761  -0.132 0.894783
A:x3        -0.13897    0.02643  -5.257 1.79e-07 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.163 on 994 degrees of freedom
Multiple R-squared:  0.3138,    Adjusted R-squared:  0.3103
F-statistic: 90.89 on 5 and 994 DF,  p-value: < 2.2e-16



$optTx
  0   1
617 383

$value
[1] -0.08377312
```

Methods available: Training Diagnostics

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`optTx(object)`	retrieve recommended optimal treatments for training data
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`summary(object)`	retrieve regression summaries

```
ot <- optTx(x = q0bj)
```

A list is returned containing

- **optimalTx** The recommended treatment $\widehat{d}_Q^{opt}$
- **decisionFunc*}** The $Q(h, a; \widehat{\beta})$ for each $a \in \mathcal{A}$

```
table(ot$optimalTx)
```

```
  0   1
617 383
```

```
head(ot$decisionFunc)
```

```
                0           1
[1,] -1.29073949 -2.14263362
[2,] -0.06647067  0.39770167
[3,]  0.59085487  0.78125203
[4,] -2.22742839 -2.09956871
[5,]  0.05677787  0.08179814
[6,]  0.12251042  0.17532513
```

The estimated value $\widehat{\mathcal{V}}_Q(d^{opt})$

```
estimator(x = q0bj)
```

```
[1] -0.08377312
```

Methods available: Predictions

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`optTx(object)`	retrieve recommended optimal treatments for training data
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`summary(object)`	retrieve regression summaries

Suppose an individual has covariates $x_1 = 2$, $x_2 = 0$, and $x_3 = 8$.

To estimate their optimal treatment:

```
newPatient <- data.frame(x1 = 2, x2 = 0, x3 = 8)
optTx(x = qObj, newdata = newPatient)
```

```
$optimalTx
[1] 1

$decisionFunc
             0      1
[1,] 1.346779 1.5205
```

Value Search

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{C_{d_\eta,i}\,Y_i}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})} - \frac{C_{d_\eta,i} - \pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}\,Q_{d_\eta}(H_i;\eta,\widehat{\beta})\right],$$

where

$$Q_{d_\eta}(H;\eta,\beta) = Q(H,1;\beta)\; d(H;\eta) + Q(H,0;\beta)\; \{1 - d(H;\eta)\}$$

and

$$\pi_{d_\eta}(H;\eta,\gamma) = \pi(H;\gamma)\; d(H;\eta) + \{1 - \pi(H;\gamma)\}\;\; \{1 - d(H;\eta)\}$$

- $Q(h,a;\beta)$ is a model for $Q(h,a) = E(Y|H=h,A=a)$
- $\pi(H;\gamma)$ is a model for the propensity score
- $d(H;\eta)$ is the restricted class of regimes

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{C_{d_\eta,i}\,Y_i}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})} - \frac{C_{d_\eta,i} - \pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}\,Q_{d_\eta}(H_i;\eta,\widehat{\beta})\right],$$

where

$$Q_{d_\eta}(H;\eta,\beta) = Q(H,1;\beta)\; d(H;\eta) + Q(H,0;\beta)\; \{1 - d(H;\eta)\}$$

and

$$\pi_{d_\eta}(H;\eta,\gamma) = \pi(H;\gamma)\; d(H;\eta) + \{1 - \pi(H;\gamma)\}\;\; \{1 - d(H;\eta)\}$$

- $Q(h,a;\beta)$ is a model for $Q(h,a) = E(Y|H=h,A=a)$
- $\pi(H;\gamma)$ is a model for the propensity score
- $d(H;\eta)$ is the restricted class of regimes

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{C_{d_\eta,i}\,Y_i}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})} - \frac{C_{d_\eta,i} - \pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}\,Q_{d_\eta}(H_i;\eta,\widehat{\beta})\right],$$

where

$$Q_{d_\eta}(H;\eta,\beta) = Q(H,1;\beta)\; d(H;\eta) + Q(H,0;\beta)\; \{1 - d(H;\eta)\}$$

and

$$\pi_{d_\eta}(H;\eta,\gamma) = \pi(H;\gamma)\; d(H;\eta) + \{1 - \pi(H;\gamma)\}\; \{1 - d(H;\eta)\}$$

- $Q(h,a;\beta)$ is a model for $Q(h,a) = E(Y|H=h,A=a)$
- $\pi(H;\gamma)$ is a model for the propensity score
- $d(H;\eta)$ is the restricted class of regimes

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{C_{d_\eta,i}\,Y_i}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})} - \frac{C_{d_\eta,i} - \pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}\,Q_{d_\eta}(H_i;\eta,\widehat{\beta})\right],$$

where

$$Q_{d_\eta}(H;\eta,\beta) = Q(H,1;\beta)\; d(H;\eta) + Q(H,0;\beta)\; \{1 - d(H;\eta)\}$$

and

$$\pi_{d_\eta}(H;\eta,\gamma) = \pi(H;\gamma)\; d(H;\eta) + \{1 - \pi(H;\gamma)\}\; \{1 - d(H;\eta)\}$$

- $Q(h,a;\beta)$ is a model for $Q(h,a) = E(Y|H=h,A=a)$
- $\pi(H;\gamma)$ is a model for the propensity score
- $d(H;\eta)$ is the restricted class of regimes

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{C_{d_{\eta,i}}\, Y_i}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})} - \frac{C_{d_{\eta,i}} - \pi_{d_\eta}(H_i;\eta,\widehat{\gamma})}{\pi_{d_\eta}(H_i;\eta,\widehat{\gamma})} Q_{d_\eta}(H_i;\eta,\widehat{\beta})\right],$$

where

$$Q_{d_\eta}(H;\eta,\beta) = Q(H,1;\beta)\; d(H;\eta) + Q(H,0;\beta)\; \{1 - d(H;\eta)\}$$

and

$$\pi_{d_\eta}(H;\eta,\gamma) = \pi(H;\gamma)\; d(H;\eta) + \{1 - \pi(H;\gamma)\}\; \{1 - d(H;\eta)\}$$

- $Q(h,a;\beta)$ is a model for $Q(h,a) = E(Y|H=h,A=a)$
- $\pi(H;\gamma)$ is a model for the propensity score
- $d(H;\eta)$ is the restricted class of regimes

Estimators obtained using `optionalSeq()`

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

Argument	Class	Description
...		Additional inputs for rgenoud()
moPropen	modelObj	for propensity regression
moMain	modelObj	for main effects terms (mu)
moCont	modelObj	for terms interacting with treatment (C)
data	data.frame	covariates and treatment history
response	vector	outcome of interest
txName	character	treatment variable name
regimes	function	definition of restricted class of regimes
fSet		not used in single stage analyses
refit		deprecated
iter	integer	if >0, iterative methods used
verbose	logical	if FALSE, screen prints suppressed

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

Postulate a model for propensity score:

$$\text{logit}(A) \sim \gamma_1$$

Use maximum likelihood.

What is the modelObj?

- What is the regression method?
  - Should we keep its default input values?

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

Postulate a model for propensity score:

$$\text{logit}(A) \sim \gamma_1$$

Use maximum likelihood.

What is the modelObj?

- What is the regression method?
  - Should we keep its default input values?
- What is the prediction method?
  - Should we keep its default input values?

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

Postulate a model for propensity score:

$$\text{logit}(A) \sim \gamma_1$$

Use maximum likelihood.

What is the modelObj?

- What is the regression method?
  - Should we keep its default input values?
- What is the prediction method?
  - Should we keep its default input values?

```
moPropen <- buildModelObj(model = ~1,
                          solver.method = 'glm',
                          solver.args = list(family = 'binomial'),
                          predict.args = list(type = "response"))
```

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

We'll use the same model for $Q(h, a)$ that we postulated for the Outcome Regression method!

$$Q(h, a; \beta) = \beta_0 + \beta_1 \; x_1 + \beta_2 \; x_2 + A \; \beta_3 + \beta_4 \; A \; x_2 + \beta_5 A \; x_3$$

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

We'll use the same model for $Q(h, a)$ that we postulated for the Outcome Regression method!

$$Q(h, a; \beta) = \beta_0 + \beta_1\ x_1 + \beta_2\ x_2 + A\ \beta_3 + \beta_4\ A\ x_2 + \beta_5 A\ x_3$$

- **moMain**, **moCont**, **data**, **response**, and **txName** are the same as for `qLearn()`

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

Define the elements of the restricted class of regimes as

$$d(h, \eta) = \mathrm{I}(x_1 < \eta_1 \;\; \& \;\; x_3 < \eta_2)$$

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

Define the elements of the restricted class of regimes as

$$d(h, \eta) = \mathrm{I}(x_1 < \eta_1 \;\; \& \;\; x_3 < \eta_2)$$

```
regimes <- function(eta1, eta2, data) {
             d1 <- {data$x1 < eta1} & {data$x3 < eta2}
             return(as.integer(x = d1))
           }
```

- data must be the last input
- No limit on the length of η.
- Must return a vector of the same type as that of the treatment provided in the data.

```
str(object = optimalSeq)
```

```
function (..., moPropen, moMain, moCont, data, response,
    txName, regimes, fSet = NULL, refit = FALSE, iter = 0L,
    verbose = TRUE)
```

Uses genetic algorithm implemented by **genoud()**

Additional inputs for **genoud()** passed through ellipsis. Must include

- **starting.values** initial estimates for regime parameters
- **Domains** matrix defining search space
- **pop.size** population size

```
starting.values <- c(0, 0)
Domains <- matrix(data = c(-10, -10, 10, 10), ncol = 2L)
pop.size <- 500 #TOO SMALL
```

```
moPropen <- buildModelObj(model = ~ 1,
                          solver.method = 'glm',
                          solver.args = list(family = 'binomial'),
                          predict.args = list(type = 'response'))

moMain <- buildModelObj(model = ~ x1 + x2, solver.method = 'lm')
moCont <- buildModelObj(model = ~ x2 + x3, solver.method = 'lm')

regimes <- function(eta1, eta2, data) {
             d1 <- {data$x1 < eta1} & {data$x3 < eta2}
             return( as.integer(x = d1) )
            }

vsObj <- optimalSeq(moPropen = moPropen,
                    moMain = moMain, moCont = moCont,
                    data = df, response = df$y, txName = 'A',
                    regimes = regimes,
                    Domains = matrix(data = c(-10, -10, 10, 10), ncol = 2L),
                    starting.values = c(0, 0), pop.size = 500)
```

```
vsObj <- optimalSeq(moPropen = moPropen,
                    moMain = moMain, moCont = moCont, iter = 0L,
                    data = df, response = df$y, txName = 'A',
                    regimes = regimes, Domains = matrix(data = c(-10,-10,10,10), ncol = 2L),
                    starting.values = c(0,0), pop.size = 500,
                    verbose = TRUE)
```

```
Value Search - Missing Data Perspective.

Propensity for treatment regression.
Regression analysis for moPropen:

Call:  glm(formula = YinternalY ~ 1, family = "binomial", data = data)

Coefficients:
(Intercept)
    -0.5151

Degrees of Freedom: 999 Total (i.e. Null);  999 Residual
Null Deviance:      1322
Residual Deviance: 1322     AIC: 1324

Outcome regression.
Combined outcome regression model: ~ x1+x2 + A + A:(x2+x3) .
Regression analysis for Combined:

Call:
lm(formula = YinternalY ~ x1 + x2 + A + x2:A + A:x3, data = data)

Coefficients:
(Intercept)           x1           x2            A
   -0.29653      0.82166      0.38095      1.28550
       x2:A         A:x3
   -0.02217     -0.13897



Wed Feb  5 17:37:40 2025
Domains:
 -1.000000e+01   <=  X1   <=    1.000000e+01
 -1.000000e+01   <=  X2   <=    1.000000e+01
```

Methods available:

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`genetic(object)`	retrieve value returned by genoud()
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`optTx(object)`	retrieve recommended optimal treatments for training data
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`propen(object)`	retrieve value returned by propensity regression analysis
`regimeCoef(object)`	retrieve estimated regime parameters
`summary(object)`	retrieve regression summaries

Methods available: New Model Diagnostics

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`genetic(object)`	retrieve value returned by genoud()
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`optTx(object)`	retrieve recommended optimal treatments for training data
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`propen(object)`	retrieve value returned by propensity regression analysis
`regimeCoef(object)`	retrieve estimated regime parameters
`summary(object)`	retrieve regression summaries

```
coef(object = vsObj)
```

```
$propensity
(Intercept)
 -0.5150946

$outcome
$outcome$Combined
(Intercept)          x1          x2           A        x2:A
-0.29653461  0.82165693  0.38094720  1.28549787 -0.02217292
       A:x3
-0.13897217
```

```
fitObj <- fitObject(object = vsObj)
print(x = fitObj)
```

```
$propensity

Call:  glm(formula = YinternalY ~ 1, family = "binomial", data = data)

Coefficients:
(Intercept)
    -0.5151

Degrees of Freedom: 999 Total (i.e. Null);  999 Residual
Null Deviance:      1322
Residual Deviance: 1322     AIC: 1324

$outcome
$outcome$Combined
```

```
genetic(object = vsObj)
```

```
$value
[1] -0.1847581

$par
[1] 1.182027 8.479420

$gradients
[1] NA NA

$generations
[1] 13

$peakgeneration
[1] 2

$popsize
[1] 500

$operators
[1] 65 62 62 62 62 62 62 62  0
```

Methods available: New Training Diagnostics

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`genetic(object)`	retrieve value returned by genoud()
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`optTx(object)`	retrieve recommended optimal treatments for training data
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`propen(object)`	retrieve value returned by propensity regression analysis
`regimeCoef(object)`	retrieve estimated regime parameters
`summary(object)`	retrieve regression summaries

The estimated optimal treatment for all training data:

```
ot <- optTx(x = vsObj)
table(ot$optimalTx)
```

```
  0   1
733 267
```

```
ot$decisionFunc
```

```
[1] NA
```

The estimated value $\widehat{\mathcal{V}}_{AIPW}(d_\eta)$

```
estimator(x = vsObj)
```

```
[1] -0.1847581
```

The estimated parameters of $d(h;\eta)$

```
regimeCoef(object = vsObj)
```

```
    eta1     eta2
1.182027 8.479420
```

Methods available: Predictions

Method	Description
`Call(name)`	retrieve the unevaluated call
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`genetic(object)`	retrieve value returned by genoud()
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`optTx(object)`	retrieve recommended optimal treatments for training data
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`propen(object)`	retrieve value returned by propensity regression analysis
`regimeCoef(object)`	retrieve estimated regime parameters
`summary(object)`	retrieve regression summaries

For the same individual we considered previously ($x_1 = 2$, $x_2 = 0$, and $x_3 = 8$), the estimated optimal treatment is:

```
optTx(x = vsObj, newdata = newPatient)
```

```
$optimalTx
[1] 0

$decisionFunc
[1] NA
```

Classification

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left\{d(H_i,\eta)\widehat{C}(H_i,A_i,Y_i) + \widehat{\psi}_0(H_i,A_i,Y_i)\right\}$$

where

$$\widehat{\psi}_1(H,A,Y) = \frac{AY}{\pi(H;\widehat{\gamma})} - \frac{A-\pi(H;\widehat{\gamma})}{\pi(H;\widehat{\gamma})}Q(H,1;\widehat{\beta}),$$

$$\widehat{\psi}_0(H,A,Y) = \frac{(1-A)Y}{1-\pi(H;\widehat{\gamma})} + \frac{A-\pi(H;\widehat{\gamma})}{1-\pi(H;\widehat{\gamma})}Q(H,0;\widehat{\beta}),$$

$$\widehat{C}(H_i,A_i,Y_i) = \widehat{\psi}_1(H_i,A_i,Y_i) - \widehat{\psi}_0(H_i,A_i,Y_i)$$

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left\{d(H_i,\eta)\widehat{C}(H_i,A_i,Y_i) + \widehat{\psi}_0(H_i,A_i,Y_i)\right\}$$

where

$$\widehat{\psi}_1(H,A,Y) = \frac{AY}{\pi(H;\widehat{\gamma})} - \frac{A-\pi(H;\widehat{\gamma})}{\pi(H;\widehat{\gamma})}Q(H,1;\widehat{\beta}),$$

$$\widehat{\psi}_0(H,A,Y) = \frac{(1-A)Y}{1-\pi(H;\widehat{\gamma})} + \frac{A-\pi(H;\widehat{\gamma})}{1-\pi(H;\widehat{\gamma})}Q(H,0;\widehat{\beta}),$$

$$\widehat{C}(H_i,A_i,Y_i) = \widehat{\psi}_1(H_i,A_i,Y_i) - \widehat{\psi}_0(H_i,A_i,Y_i)$$

Recall:

$$\widehat{\mathcal{V}}_{AIPW}(d_\eta) = \frac{1}{n}\sum_{i=1}^{n}\left\{d(H_i,\eta)\widehat{C}(H_i,A_i,Y_i) + \widehat{\psi}_0(H_i,A_i,Y_i)\right\}$$

where

$$\widehat{\psi}_1(H,A,Y) = \frac{AY}{\pi(H;\widehat{\gamma})} - \frac{A-\pi(H;\widehat{\gamma})}{\pi(H;\widehat{\gamma})}Q(H,1;\widehat{\beta}),$$

$$\widehat{\psi}_0(H,A,Y) = \frac{(1-A)Y}{1-\pi(H;\widehat{\gamma})} + \frac{A-\pi(H;\widehat{\gamma})}{1-\pi(H;\widehat{\gamma})}Q(H,0;\widehat{\beta}),$$

$$\widehat{C}(H_i,A_i,Y_i) = \widehat{\psi}_1(H_i,A_i,Y_i) - \widehat{\psi}_0(H_i,A_i,Y_i)$$

Estimator implemented as `optionalClass()`.

```
str(object = optimalClass)
```

```
function (..., moPropen, moMain, moCont, moClass, data,
    response, txName, iter = 0L, fSet = NULL, verbose = TRUE)
```

Argument	Class	Description
...		Ignored. Included to require named input.
moPropen	modelObj	for propensity regression
moMain	modelObj	for main effects terms (mu)
moCont	modelObj	for terms interacting with treatment (C)
moClass	modelObj	for classification regression
data	data.frame	covariates and treatment history
response	vector	outcome of interest
txName	character	treatment variable name
iter	integer	if >0, iterative methods used
fSet		not used in single stage analyses
verbose	logical	if FALSE, screen prints suppressed

```
str(object = optimalClass)
```

```
function (..., moPropen, moMain, moCont, moClass, data,
    response, txName, iter = 0L, fSet = NULL, verbose = TRUE)
```

We'll use the same "modelObj" previously defined for $Q(H, A; \beta)$ and $\pi(H; \gamma)$.

```
str(object = optimalClass)
```

```
function (..., moPropen, moMain, moCont, moClass, data,
    response, txName, iter = 0L, fSet = NULL, verbose = TRUE)
```

We'll use the same "modelObj" previously defined for $Q(H, A; \beta)$ and $\pi(H; \gamma)$.

Need to define the classification method.

Method must use input **weights** and return predictions as the class.

```
library(rpart)
moClass <- buildModelObj(model = ~x1 + x2 + x3,
                         solver.method = 'rpart',
                         predict.args = list(type = "class"))
```

```
str(object = optimalClass)
```

```
function (..., moPropen, moMain, moCont, moClass, data,
    response, txName, iter = 0L, fSet = NULL, verbose = TRUE)
```

We'll use the same "modelObj" previously defined for $Q(H, A; \beta)$ and $\pi(H; \gamma)$.

Need to define the classification method.

Method must use input **weights** and return predictions as the class.

```
library(rpart)
moClass <- buildModelObj(model = ~x1 + x2 + x3,
                         solver.method = 'rpart',
                         predict.args = list(type = "class"))
```

All other inputs are the same!

```
moPropen <- buildModelObj(model = ~ 1,
                          solver.method = 'glm',
                          solver.args = list(family = 'binomial'),
                          predict.args = list(type = 'response'))

moMain <- buildModelObj(model = ~ x1 + x2, solver.method = 'lm')
moCont <- buildModelObj(model = ~ x2 + x3, solver.method = 'lm')

moClass <- buildModelObj(model = ~x1 + x2 + x3,
                         solver.method = 'rpart',
                         predict.args = list(type = "class"))

clObj <- optimalClass(moPropen = moPropen,
                      moMain = moMain, moCont = moCont,
                      moClass = moClass,
                      data = df, response = df$y, txName = 'A')
```

```
clObj <- optimalClass(moPropen = moPropen,
                      moMain = moMain, moCont = moCont,
                      moClass = moClass,
                      data = df, response = df$y, txName = 'A')
```

```
AIPW value estimator

First step of the Classification Algorithm.

Classification Perspective.


Propensity for treatment regression.
Regression analysis for moPropen:

Call:  glm(formula = YinternalY ~ 1, family = "binomial", data = data)

Coefficients:
(Intercept)
    -0.5151

Degrees of Freedom: 999 Total (i.e. Null);  999 Residual
Null Deviance:      1322
Residual Deviance: 1322     AIC: 1324

Outcome regression.
Combined outcome regression model: ~ x1+x2 + A + A:(x2+x3) .
Regression analysis for Combined:

Call:
lm(formula = YinternalY ~ x1 + x2 + A + x2:A + A:x3, data = data)

Coefficients:
(Intercept)           x1           x2            A
   -0.29653      0.82166      0.38095      1.28550
       x2:A         A:x3
   -0.02217     -0.13897


Classification Analysis
Regression analysis for moClass:
n= 1000
```

Methods available:

Method	Description
`Call(name)`	retrieve the unevaluated call
`classif(object)`	retrieve value returned by classification method
`coef(object)`	retrieve parameter estimates
`DTRstep(object)`	print description of method
`estimator(x)`	retrieve estimated value
`fitObject(object)`	retrieve value object returned for regression steps
`optTx(object, newdata)`	estimate optimal treatments based on a prior analysis
`optTx(object)`	retrieve recommended optimal treatments for training data
`outcome(object)`	retrieve value object returned for outcome regression
`plot(x, suppress)`	plot regression results
`propen(object)`	retrieve value returned by propensity regression analysis
`summary(object)`	retrieve regression summaries

```
classObj <- classif(object = clObj)
is(object = classObj)
```

```
[1] "rpart"
```

```
utils::methods(class = is(object = classObj)[1L])
```

```
 [1] labels       meanvar      model.frame plot
 [5] post         predict      print       prune
 [9] residuals    summary      text
see '?methods' for accessing help and source code
```

Training Diagnostics

```
ot <- optTx(x = clObj)
names(ot)
```

```
[1] "optimalTx"    "decisionFunc"
```

```
table(ot$optimalTx)
```

```
  0   1
590 410
```

```
ot$decisionFunc
```

```
[1] NA
```

```
estimator(x = clObj)
```

```
[1] 0.08087631
```

Predictions

```
newPatient <- data.frame(x1 = 2, x2 = 0, x3 = 8)
optTx(x = clObj, newdata = newPatient)
```

```
$optimalTx
[1] 0

$decisionFunc
[1] NA
```

Outcome Weighted Learning

Recall:

The decision function

$$f(X;\eta) = \sum_{i=1}^{n} \widehat{\alpha}_i A_i k(X, X_i) + \widehat{\beta}_0$$

is estimated by minimizing

$$\frac{1}{n}\sum_{i=1}^{n} \frac{Y_i}{A_i\pi(H;\gamma) + (1-A_i)/2} \ell_{0-1}\left\{A_i f(H_i;\eta)\right\} + \lambda_n ||f||^2$$

Recall:

The decision function

$$f(X;\eta) = \sum_{i=1}^{n} \widehat{\alpha}_i A_i k(X, X_i) + \widehat{\beta}_0$$

is estimated by minimizing

$$\frac{1}{n}\sum_{i=1}^{n} \frac{Y_i}{A_i\pi(H;\gamma) + (1-A_i)/2} \ell_{0-1}\left\{A_i f(H_i;\eta)\right\} + \lambda_n ||f||^2$$

```
str(object = owl)
```

```
function (..., moPropen, data, reward, txName, regime,
    response, lambdas = 2, cvFolds = 0L, kernel = "linear",
    kparam = NULL, surrogate = "hinge", verbose = 2L)
```

Argument	Class	Description
...		Ignored. Included to require named input.
moPropen	modelObj	for propensity regression
data	data.frame	covariates and treatment history
reward	vector	outcome of interest
txName	character	treatment variable name
regime	formula	covariates of the decision function
response	vector	outcome of interest
lambdas	numeric	one or more tuning parameters
cvFolds	integer	number of cross-validation steps to select lambda and/or kparam
kernel	character	one of linear, poly, radial
kparam	numeric	kernel parameter
surrogate	character	surrogate 0-1 loss function logit, exp, hinge, sqhinge, huber
verbose	integer or logical	level of screen printing

```
owl0bj <- owl(moPropen = moPropen, data = df, reward = df$y,
              txName = "A", regime = ~x1 + x2 + x3,
              lambdas = 0.01, kernel = "linear", surrogate = "sqhinge")
```

```
owlObj <- owl(moPropen = moPropen, data = df, reward = df$y,
              txName = "A", regime = ~ x1,
              lambdas = 0.01, kernel = "linear", surrogate = "sqhinge")
```

```
Outcome Weighted Learning

Propensity for treatment regression.
Regression analysis for moPropen:

Call:  glm(formula = YinternalY ~ 1, family = "binomial", data = data)

Coefficients:
(Intercept)
    -0.5151

Degrees of Freedom: 999 Total (i.e. Null);  999 Residual
Null Deviance:      1322
Residual Deviance: 1322     AIC: 1324

Outcome regression.
No outcome regression performed.

Final optimization step.
initial  value 2.281378
final  value 1.934095
converged

Results returned by optimization method
$par
[1]  0.332536 -0.200740

$value
[1] 1.934095

$counts
function gradient
      14        5

$convergence
[1] 0

$message
```

```
methods(class = "OWL")
```

```
 [1] Call       coef       cvInfo     DTRstep    estimator
 [6] fitObject  optimObj   optTx      outcome    plot
[11] print      propen     regimeCoef show       summary
see '?methods' for accessing help and source code
```

Outline

These slides are available from
https://github.com/sth1402/ST790/

If you have **ANY** questions or encounter any problems, please do not hesitate to contact me

Shannon Holloway shannon.t.holloway@gmail.com