Package `mactivate`
Examples II

Dave Zes

March 2, 2021

1 Example: Medium Data, Numerical Inputs, Numerical Response

```
library(mactivate)
set.seed(777)
d <- 56
N <- 100000
X <- matrix(rnorm(N*d, 0, 1), N, d) ####
colnames(X) <- paste0("x", I(1:d))
############# primary effects
b <- rep_len( c(-1/4, 1/4), d )
###########


ystar <-
  X %*% b +
  1/3 * (X[ , 11]+1) * (X[ , 12]-1) * (X[ , 33]+1) -
  1/2 * (X[ , 30]+0) * (X[ , 44]+1) * (X[ , 45]-0) * (X[ , 56]-1) +
  1/3 * (X[ , 6]+1) * (X[ , 47]-1) -
  1/2 * (X[ , 1]-1) * (X[ , 32]+0) * (X[ , 33]+1) * (X[ , 34]-0) *
  (X[ , 45]-0) * (X[ , 51]-1)
m_tot <- 12
#############



xs1 <-
```

```
   "y ~ . + x11:x12:x33 + x30:x44:x45:x56 + x6:x47 + x1:x32:x33:x34:x45:x51"
xs2 <-
   "y ~ . + x11*x12*x33 + x30*x44*x45*x56 + x6*x47 + x1*x32*x33*x34*x45*x51"
xnotQuiteTrue_formula <- eval(parse(text=xs1))
xtrue_formula <- eval(parse(text=xs2))
xnoint_formula <- eval(parse(text="y ~ ."))
yerrs <- rnorm(N, 0, 3)
y <- ystar + yerrs
## y <- (y - mean(y)) / sd(y)



########## standardize X
Xall <- t( ( t(X) - apply(X, 2, mean) ) / apply(X, 2, sd) )
yall <- y
Nall <- N
####### fold index
xxfoldNumber <- rep_len(1:2, N)
ufolds <- sort(unique(xxfoldNumber)) ; ufolds
############### predict
############### predict


dfx <- data.frame("y"=yall, Xall)
################## incorrectly fit LM: no interactions

xlm <- lm(xnoint_formula , data=dfx)
summary(xlm)
yhat <- predict(xlm, newdata=dfx)
sqrt( mean( (yall - yhat)^2 ) )
################## incorrectly fit LM: no lower-order interactions
xlm <- lm(xnotQuiteTrue_formula , data=dfx)
summary(xlm)
yhat <- predict(xlm, newdata=dfx)
sqrt( mean( (yall - yhat)^2 ) )
################# correctly fit LM
xlm <- lm(xtrue_formula, data=dfx)
summary(xlm)
yhat <- predict(xlm, newdata=dfx)
sqrt( mean( (yall - yhat)^2 ) )
############### fit using hybrid m-activation
```

```
###### takes about 1.5-3 hours

xcmact_hybrid <-
  f_control_mactivate(
  param_sensitivity = 10^12,
  bool_free_w = TRUE,
  w0_seed = 0.01,
  w_col_search = "alternate",
  max_internal_iter = 500, #####
  ss_stop = 10^(-14), ###
  escape_rate = 1.003,  #### 1.002,
  Wadj = 1/1,
  force_tries = 0,
  lambda = 0/10000, ### hybrid only
  tol = 10^(-14) ### hybrid only
  )
#### Fit


Uall <- Xall
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
X_train <- Xall[ xndx_train, , drop=FALSE ]
y_train <- yall[ xndx_train ]
xxnow <- Sys.time()
xxls_out <-
  f_fit_hybrid_01(
  X = X_train,
  y = y_train,
  m_tot = m_tot,
  U = X_train,
  m_start = 1,
  mact_control = xcmact_hybrid,
  verbosity = 1
  )
cat( difftime(Sys.time(), xxnow, units="mins"), "\n" )
######### check test error
```

```r
U_test <- Xall[ xndx_test, , drop=FALSE ]
X_test <- Xall[ xndx_test, , drop=FALSE ]
y_test <- yall[ xndx_test ]
yhatTT <- matrix(NA, length(xndx_test), m_tot+1)
for(iimm in 0:m_tot) {
      yhat_fold <- predict(object=xxls_out, X0=X_test, U0=U_test, mcols=iimm )
      yhatTT[ , iimm + 1 ] <- yhat_fold
  }
errs_by_m <- NULL
for(iimm in 1:ncol(yhatTT)) {
      yhatX <- yhatTT[ , iimm]
      errs_by_m[ iimm ] <- sqrt(mean( (y_test - yhatX)^2 ))
      cat(iimm, "::", errs_by_m[ iimm ])
  }
##### plot test RMSE vs m

plot(0:(length(errs_by_m)-1), errs_by_m, type="l", xlab="m", ylab="RMSE Cost")
#### 11 best




################# known 'true' for non zero-centered is
xtrue_formula_use <- xtrue_formula
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
xlm <- lm(xtrue_formula_use , data=dfx[ xndx_train, ])
yhat <- predict(xlm, newdata=dfx[ xndx_test, ])
sqrt( mean( (y_test - yhat)^2 ) )
##############




####### Let's dig in
####### We can use train W to construct test Xstar
####### Check which columns drive our response, y
```

```
X_test <- Xall[ xndx_test, ]
y_test <- yall[ xndx_test ]
Xstar_test <- f_mactivate(U=X_test, W=xxls_out[[ length(xxls_out) ]][[ "What" ]])
Xi <- cbind(X_test, Xstar_test)
xlm <- lm(y_test ~ Xi)
sumxlm <- summary(xlm)
```

```
print(sumxlm)
```

Call:
lm(formula = y_test ~ Xi)

Residuals:
     Min   1Q   Median   3Q   Max
-12.6721 -2.0218 -0.0189   2.0268 17.8341

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.706e+00 3.882e+00   0.697 0.485760
Xix1   -2.529e-01 5.311e-02   -4.762 1.92e-06 ***
Xix2   2.928e-01 4.076e-02   7.183 6.93e-13 ***
Xix3   -2.394e-01 6.772e-02   -3.535 0.000409 ***
Xix4   2.901e-01 5.291e-02   5.482 4.22e-08 ***
Xix5   -1.933e-01 8.021e-02   -2.409 0.015983 *
Xix6   -5.480e-02 3.740e-02   -1.465 0.142835
Xix7   -2.412e-01 5.105e-02   -4.725 2.31e-06 ***
Xix8   2.832e-01 4.973e-02   5.694 1.25e-08 ***
Xix9   -2.380e-01 4.192e-02   -5.677 1.38e-08 ***
Xix10   2.814e-01 4.999e-02   5.629 1.82e-08 ***
Xix11   -3.069e-01 6.118e-02   -5.017 5.27e-07 ***
Xix12   3.381e-01 4.511e-02   7.495 6.74e-14 ***
Xix13   -2.237e-01 5.192e-02   -4.308 1.65e-05 ***
Xix14   2.723e-01 3.932e-02   6.924 4.46e-12 ***
Xix15   -2.244e-01 4.134e-02   -5.427 5.75e-08 ***
Xix16   2.955e-01 5.092e-02   5.803 6.54e-09 ***
Xix17   -2.419e-01 4.757e-02   -5.085 3.69e-07 ***
Xix18   2.535e-01 6.801e-02   3.728 0.000193 ***
Xix19   -2.004e-01 8.207e-02   -2.442 0.014614 *
Xix20   3.012e-01 4.416e-02   6.821 9.12e-12 ***
Xix21   -2.099e-01 5.010e-02   -4.190 2.79e-05 ***
Xix22   2.854e-01 6.367e-02   4.482 7.42e-06 ***
Xix23   -2.280e-01 5.350e-02   -4.261 2.04e-05 ***
Xix24   2.858e-01 4.097e-02   6.975 3.09e-12 ***
Xix25   -2.216e-01 4.775e-02   -4.642 3.46e-06 ***
Xix26   2.855e-01 4.316e-02   6.615 3.77e-11 ***
Xix27   -1.908e-01 7.456e-02   -2.560 0.010482 *

```
Xix28        3.146e-01  6.579e-02    4.781 1.75e-06 ***
Xix29       -2.218e-01  4.379e-02   -5.065 4.09e-07 ***
Xix30        2.936e-01  5.358e-02    5.480 4.26e-08 ***
Xix31       -2.128e-01  4.253e-02   -5.003 5.65e-07 ***
Xix32        2.775e-01  3.978e-02    6.977 3.06e-12 ***
Xix33       -3.369e-01  3.928e-02   -8.576  < 2e-16 ***
Xix34        2.802e-01  4.295e-02    6.524 6.91e-11 ***
Xix35       -2.099e-01  6.172e-02   -3.401 0.000672 ***
Xix36        2.902e-01  5.601e-02    5.180 2.22e-07 ***
Xix37       -2.130e-01  6.445e-02   -3.305 0.000951 ***
Xix38        3.002e-01  8.099e-02    3.707 0.000210 ***
Xix39       -2.601e-01  5.574e-02   -4.666 3.08e-06 ***
Xix40        2.896e-01  6.015e-02    4.815 1.48e-06 ***
Xix41       -2.460e-01  4.160e-02   -5.913 3.38e-09 ***
Xix42        3.086e-01  5.391e-02    5.725 1.04e-08 ***
Xix43       -2.435e-01  4.662e-02   -5.222 1.77e-07 ***
Xix44        2.848e-01  5.467e-02    5.209 1.91e-07 ***
Xix45       -2.204e-01  5.974e-02   -3.689 0.000225 ***
Xix46        2.473e-01  5.917e-02    4.179 2.93e-05 ***
Xix47        5.280e-01  6.007e-02    8.791  < 2e-16 ***
Xix48        2.576e-01  4.636e-02    5.557 2.76e-08 ***
Xix49       -2.533e-01  6.713e-02   -3.774 0.000161 ***
Xix50        2.834e-01  5.838e-02    4.855 1.21e-06 ***
Xix51       -2.187e-01  4.951e-02   -4.417 1.00e-05 ***
Xix52        2.511e-01  4.219e-02    5.952 2.67e-09 ***
Xix53       -2.323e-01  4.953e-02   -4.691 2.73e-06 ***
Xix54        2.729e-01  5.590e-02    4.881 1.06e-06 ***
Xix55       -2.274e-01  3.979e-02   -5.714 1.11e-08 ***
Xix56        2.814e-01  5.066e-02    5.554 2.81e-08 ***
Xi           1.746e-01  2.384e-03   73.242  < 2e-16 ***
Xi           2.684e-01  5.615e-03   47.809  < 2e-16 ***
Xi          -1.977e-04  8.386e-04   -0.236 0.813601
Xi          -1.414e-01  5.993e-03  -23.589  < 2e-16 ***
Xi           2.328e-04  7.586e-03    0.031 0.975514
Xi          -4.343e-02  4.186e-04 -103.753  < 2e-16 ***
Xi          -2.076e-02  1.866e-02   -1.113 0.265808
Xi           3.501e-03  8.045e-03    0.435 0.663476
Xi           3.621e+04  3.032e+05    0.119 0.904920
Xi          -8.776e+04  9.212e+05   -0.095 0.924103
```

```
Xi              5.259e+04  8.601e+05    0.061 0.951251
Xi             -1.046e+03  2.450e+05   -0.004 0.996593
Xi             -4.014e-03  1.411e-02   -0.284 0.776060
Xi              1.569e-02  1.524e-02    1.030 0.302984
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 3.019 on 49929 degrees of freedom
Multiple R-squared:  0.4611,        Adjusted R-squared:  0.4603
F-statistic: 610.2 on 70 and 49929 DF,  p-value: < 2.2e-16
```

```
 ########### Looks like cols 1,2,4,6 are where all the action is
 ########### (in other word, mactivate didn't track signal on passes 3 and 5)

 bWhat <- xxls_out[[ length(xxls_out) ]][[ "What" ]][ ,  c(1,2,4,6) ]
 bWhat
 wwmag <- apply(bWhat, 1, function(x) { return(sum(abs(x)))} ) ; wwmag
 plot(wwmag, type="h", lwd=4,
   main="W Coefficient Total Magnitute vs Input Term",
   xlab="Column of U (in this case, same as X)",
   ylab="Sum of magnitudes in fitted W",
   cex.lab=1.3
   )
```

W Coefficient Total Magnitude vs Input Term
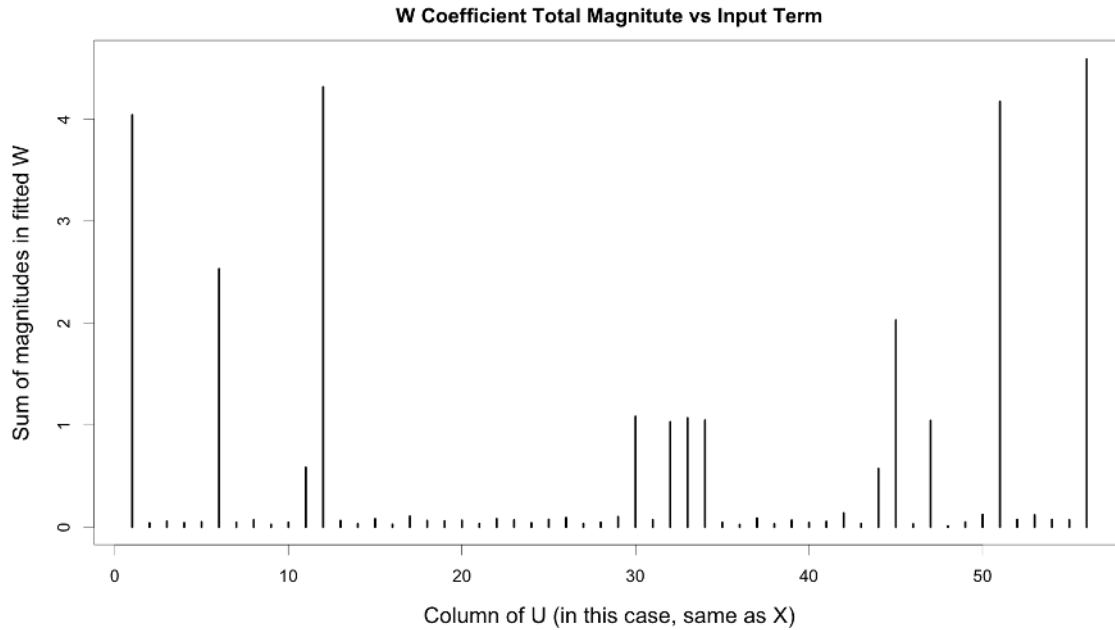

```
###########################


########### these are the terms in X that
########### appear to have a polynomial contribution
########### to our response, y

xxhigh <- which(wwmag > 0.5) ; xxhigh


#####################
```

2 Example: Orthopedic Sales Data, Numerical Inputs, Numerical Response, 10-Fold CV

```
data(df_hospitals_ortho)
xvars <- attr(df_hospitals_ortho, "modelvars")
xmx <- as.matrix(df_hospitals_ortho[ , xvars])
old_par <- par(mfrow=c(3,4))
for(jj in 1:ncol(xmx)) {
     hist(xmx[ ,jj], main=colnames(xmx)[jj])
```

```
  }
######## pause and look


############### transform

xlog_list <-
  c(
  "tot_sales",
  "tot_knee",
  "tot_hip",
  "beds",
  "rehab_beds",
  "outpatient_visits",
  "adm_costs",
  "revenue_inpatient"
  )
ymx <- xmx
for(jj in 1:ncol(xmx)) {
      ymx[ ,jj] <- log( xmx[ ,jj] + 1 )
  }
for(jj in 1:ncol(ymx)) {
      hist(ymx[ ,jj], main=colnames(ymx)[jj])
  }
######## pause and look




#### standardize

ymx_stnd <- t( ( t( ymx ) - apply(ymx, 2, mean)) / apply(ymx, 2, sd) )
for(jj in 1:ncol(ymx_stnd)) {
      hist(ymx_stnd[ ,jj], main=colnames(ymx_stnd)[jj])
  }
######## pause and look




############# let's fit an MLR model (in place, no train-test)
ydf_stnd <- as.data.frame(ymx_stnd)
```

```
xlm <- lm( tot_sales ~ . , data=ydf_stnd )
summary(xlm)
yhat <- xlm$fitted
yy <- ydf_stnd[ , "tot_sales" ]
rmse <- sqrt( mean( (yy - yhat)^2 ) ) ; rmse
1 - rmse^2 / 1  #### r-squared
######## now let's break out m-activation using 10-fold CV


library(mactivate)
yall <- ymx_stnd[ , "tot_sales" ]
Xall <- ymx_stnd[ , -1 ]
Uall <- Xall
xfolds <- rep_len( 1:10, length(yall) ) ## 10-fold CV
m_tot <- 5
xmcont <-
  f_control_mactivate(
  param_sensitivity = 10^11,
  w0_seed = 0.1,
  max_internal_iter = 500,
  w_col_search = "one",
  bool_headStart = FALSE,
  ss_stop = 10^(-11),
  escape_rate = 1.001,
  step_size = 1/100,
  Wadj = 1/1,
  force_tries = 0,
  lambda = 0,
  tol = 10^(-8)
  )
ufolds <- sort(unique(xfolds))
yout <- numeric(length(yall))
#### takes about 5-10 minutes

for(iif in 1:length(ufolds)) {

    xmask_fold <- xfolds %in% ufolds[ iif ]

    xxls_out <-
```

```
        f_fit_hybrid_01(
        X=Xall[ !xmask_fold, ],
        y=yall[ !xmask_fold ],
        m_tot=m_tot,
        U = Xall[ !xmask_fold, ],
        m_start = 1,
        mact_control = xmcont,
        verbosity = 0
        )


        xxls_out


        yhatG <- predict(
            object=xxls_out,
            X0=Xall[ xmask_fold, ],
            U0=Uall[ xmask_fold, ],
            mcols=m_tot
            )


        yout[ xmask_fold ] <- yhatG


        cat("Done this fold:", iif, "\n\n")


    }
mact_rmse <- sqrt( mean( (yall  -  yout)^2 ) ) ; mact_rmse
cat("TT R2:", 1 - mact_rmse^2 / 1, "\n") #### m-activation R^2
par(old_par)
```

3   Example: Medium Data, Numerical Inputs, Dichotomous Response

```
library(mactivate)
set.seed(777)
d <- 25
N <- 100000
X <- matrix(rnorm(N*d, 0, 1), N, d) ####
colnames(X) <- paste0("x", I(1:d))
```

```
############# primary effects
b <- rep_len( c(-1/4, 1/4), d )
ystar <-
  X %*% b +
  1/3 * (X[ , 1]+1) * (X[ , 2]-1) * (X[ , 3]+1) -
  1/2 * (X[ , 3]+0) * (X[ , 4]+1) * (X[ , 5]-0) * (X[ , 6]-1) +
  1/3 * (X[ , 6]+1) * (X[ , 7]-1) -
  1/2 * (X[ , 1]-1) * (X[ , 2]+0) * (X[ , 3]+1) * (X[ , 4]-0) *
  (X[ , 5]-0) * (X[ , 7]-1)
m_tot <- 10
#############


xs1 <- "y ~ . + x1:x2:x3 + x3:x4:x5:x6 + x6:x7 + x1:x2:x3:x4:x5:x7"
xs2 <- "y ~ . + x1*x2*x3 + x3*x4*x5*x6 + x6*x7 + x1*x2*x3*x4*x5*x7"
xnotQuiteTrue_formula <- eval(parse(text=xs1))
xtrue_formula <- eval(parse(text=xs2))
xnoint_formula <- eval(parse(text="y ~ ."))
ysigmoid <- 1 / (1 + exp(-ystar))
range(ysigmoid)
y <- rbinom(size=1, n=N, prob=ysigmoid)
########## standardize X
Xall <- t( ( t(X) - apply(X, 2, mean) ) / apply(X, 2, sd) )
yall <- y
Nall <- N
####### fold index
xxfoldNumber <- rep_len(1:2, N)
ufolds <- sort(unique(xxfoldNumber)) ; ufolds
############### predict
############### predict


dfx <- data.frame("y"=yall, Xall)
##################


xglm <- glm(xnoint_formula , data=dfx, family=binomial(link="logit"))
summary(xglm)
yhat <- predict(xglm, newdata=dfx, type="response")
mean( f_logit_cost(y=yall, yhat=yhat) )
####### known true when zero centered
```

```
xglm <- glm(xnotQuiteTrue_formula , data=dfx, family=binomial(link="logit"))
summary(xglm)
yhat <- predict(xglm, newdata=dfx, type="response")
mean( f_logit_cost(y=yall, yhat=yhat) )
####### known true when not zero centered
xglm <- glm(xtrue_formula , data=dfx, family=binomial(link="logit"))
summary(xglm)
yhat <- predict(xglm, newdata=dfx, type="response")
mean( f_logit_cost(y=yall, yhat=yhat) )
############################### alternate
############################### alternate -- about 1.5 hours

xcmact_gradient <-
  f_control_mactivate(
  param_sensitivity = 10^9,
  bool_free_w = TRUE,
  w0_seed = 0.05,
  w_col_search = "alternate",
  bool_headStart = TRUE,
  ss_stop = 10^(-9), ###
  escape_rate = 1.003,
  Wadj = 1/1,
  force_tries = 0
  )
Uall <- Xall
xthis_fold <- ufolds[ 1 ]
xndx_test <- which( xxfoldNumber %in% xthis_fold )
xndx_train <- setdiff( 1:Nall, xndx_test )
X_train <- Xall[ xndx_train, , drop=FALSE ]
y_train <- yall[ xndx_train ]
xxnow <- Sys.time()
xxls_out <-
  f_fit_gradient_logistic_01(
  X = X_train,
  y = y_train,
  m_tot = m_tot,
  U = X_train,
  m_start = 1,
  mact_control = xcmact_gradient,
```

```
    verbosity = 1
  )
cat( difftime(Sys.time(), xxnow, units="mins"), "\n" )
U_test <- Xall[ xndx_test, , drop=FALSE ]
X_test <- Xall[ xndx_test, , drop=FALSE ]
y_test <- yall[ xndx_test ]
yhatTT <- matrix(NA, length(xndx_test), m_tot+1)
for(iimm in 0:m_tot) {
      yhat_fold <- predict(object=xxls_out, X0=X_test, U0=U_test, mcols=iimm )
      yhatTT[ , iimm + 1 ] <- yhat_fold[[ "p0hat" ]]
  }
errs_by_m <- NULL
for(iimm in 1:ncol(yhatTT)) {
      yhatX <- yhatTT[ , iimm]
      errs_by_m[ iimm ] <- mean( f_logit_cost(y=y_test, yhat=yhatX) )
      cat(iimm, "::", errs_by_m[ iimm ])
  }
##### plot test Logit vs m

plot(0:(length(errs_by_m)-1), errs_by_m, type="l", xlab="m", ylab="Logit Cost")
################# use known 'true' in glm()

xglm <- glm(xtrue_formula , data=dfx[ xndx_train, ], family=binomial(link="logit"))
yhat <- predict(xglm, newdata=dfx[ xndx_test, ], type="response")
mean( f_logit_cost(y=y_test, yhat=yhat) )
```