6520 Project

Minjia Jia and Joia Zhang

Fall 2023

Set up

```
rm(list=ls())
set.seed(6520)
library(ggplot2)
library(expm) # for sqrtm
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'expm'
```

```
## The following object is masked from 'package:Matrix':
##
##     expm
```

```
library(devtools)
```

```
## Loading required package: usethis
```

```
library(roxygen2)

# load our package onlinegrad from Github
# install_github("joiazhang/6520project", subdir="onlinegrad") # only need to run once for installation
library(onlinegrad)
```

```
# data dimensions
n = 100 # sample size
p = 200 # number of predictors

# hyperparamters
lr = 1e-4 # learning rate
beta_0 = rep(0, p) # weight initialization
# beta_0 = runif(p)
```

Plotting

```r
# plot prediction or estimation error
# X: nxp matrix where rows are observations, columns are predictors
# Y: nx1 vector response variable
# betahats: n x p matrix where each ith row is the coefficients for the ith iteration and the columns a
# beta: px1 vector of true beta coefficient
# title: string for the title of the plot
# type: string that is "prediction" or "estimation" for prediction error or estimation error
# bs: vector of initial intercepts
plot_prediction_error = function(betahats, beta, X, Y, title, type, regression=T, bs) {
  n = nrow(X)
  p = ncol(X)
  if ((type!="prediction") && (type!="estimation")) {
    stop("type parameter must be 'prediction' or 'estimation'")
  }

  if (type=="prediction") {
    if (regression) {
      # prediction error
      err = colSums((X%*%t(betahats) - matrix(rep(Y, n), nrow=n, ncol=n, byrow=F))^2) # row of the insi
      ylab = "Prediction error"
    } else {
      # classification

      # predict y
      Z = as.matrix(rowSums(betahats*X))+bs
      Y_pred = 1/(1+1/exp(Z))
      Yhat = rep(NA, n)
      Yhat[Y_pred < 0.5] = 0
      Yhat[Y_pred >= 0.5] = 1

      #  misclassification rate
      err = cumsum(Yhat!=Y)/(1:n)
      ylab = "Misclassification rate"
    }
  } else {
    # estimation error
    beta = t(beta)
    beta = matrix(rep(beta, n),nrow=n, byrow=T) # row combine n number of t(beta)'s
    err = sqrt(rowSums((betahats - beta)^2))
    ylab = "Estimation error"
  }
  err = as.matrix(err)
  plot(err, xlab="Iteration", ylab=ylab, main=title)
  return(err)
}
```

```r
# plot last iteration of betahat for nonzero vs zero indexes, true beta overlaid
# betahats: n x p matrix where each ith row is the coefficients for the ith iteration and the columns a
# beta: px1 vector of true beta coefficient
# nonzero_indexes: kx1 vector of indexes where the k-sparse vector true beta has nonzero values
# nonzero: boolean, if true plot only nonzero indexes (k indexes of the true k-sparse beta), otherwise
plot_betas = function(betahats, beta, nonzero_indexes, nonzero=T) {
  n = nrow(betahats)
```

```
  p = ncol(betahats)
  dat = data.frame("p"=1:p, "bethat_n"=betahats[n, ], "beta"=beta)

  if (nonzero) {
    dat = dat[nonzero_indexes, ]
    title = "Last iteration of bethat (orange) and true beta (black) at nonzero indexes"
  } else {
    dat = dat[-nonzero_indexes, ]
    title = "Last iteration of bethat (orange) and true beta (black) at zero indexes"
  }

  ggplot(dat) + geom_point(aes(x=p, y=bethat_n), color="orange") +
    geom_point(aes(x=p, y=beta), color="black", shape=4) +
    xlab("p") +
    ylab("") +
    ggtitle(title)
}
```

Regression

Generate data for linear regression

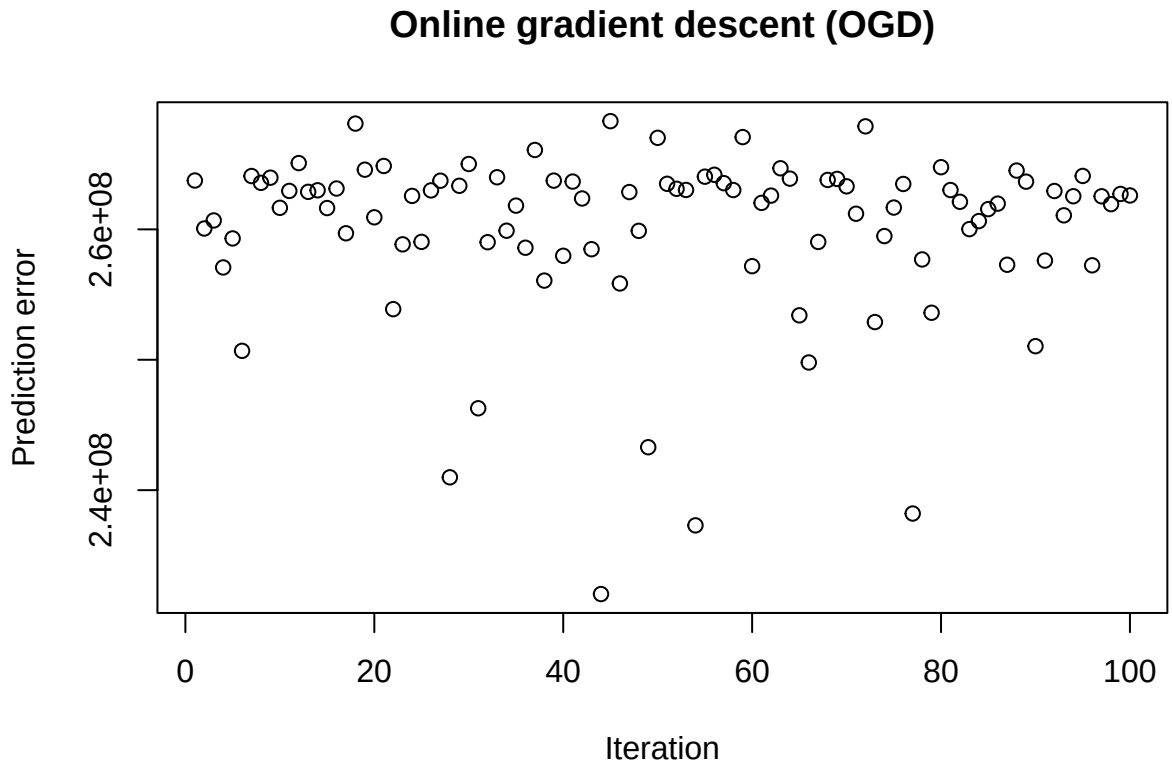
```
# simulate data: regression

# beta
k = round(0.05*p, 0) # number of nonzero coefficients
sd_beta = 0.01
nonzero_indexes = sample.int(n=p, size=k)
beta = rep(0, p)
beta[nonzero_indexes] = rnorm(n=k, mean=100, sd=sd_beta)
sum(which(beta !=0) != sort(nonzero_indexes)) # test that we made the right indexes nonzero
```

```
## [1] 0
```

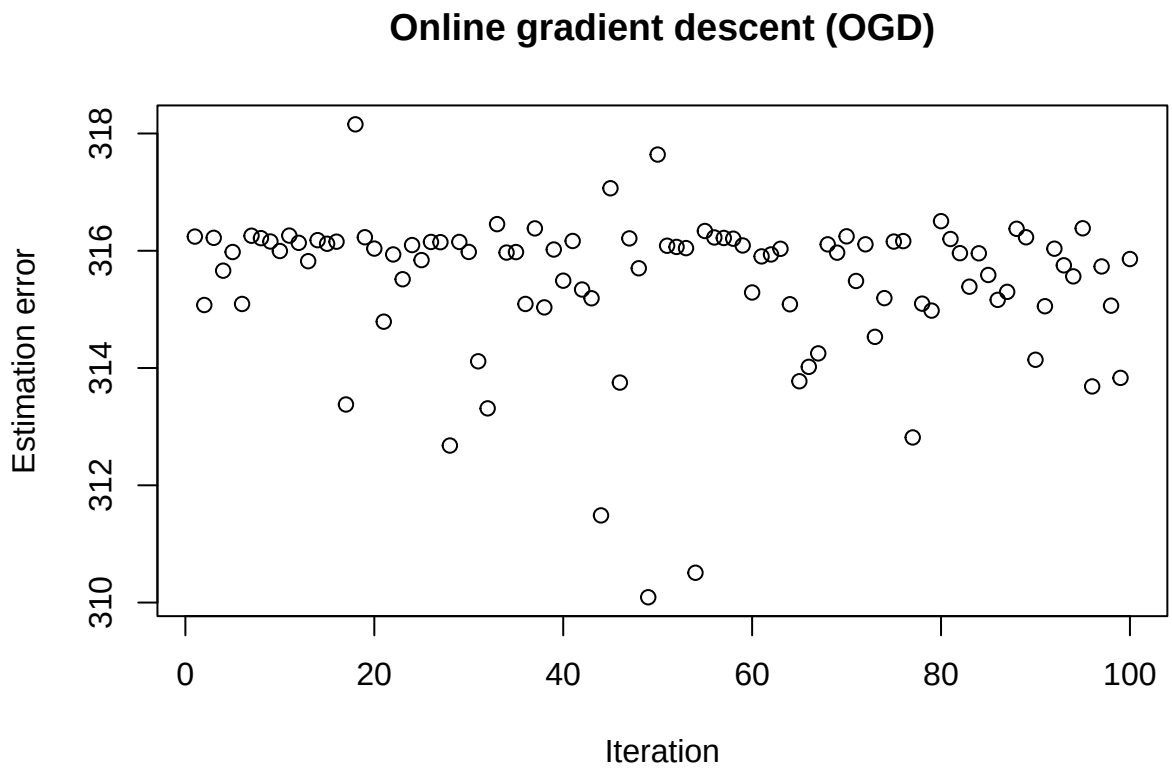
```
beta = as.matrix(beta)

# x
X = matrix(rnorm(n=n*p, mean=0, sd=5), nrow=n)

# epsilon
E = matrix(rnorm(n=n, mean=0, sd=1), nrow=n)

# y

Y = X%*%beta + E

# note that in the online setting, each t^th row of X and Y is for time t
```

OGD: regression

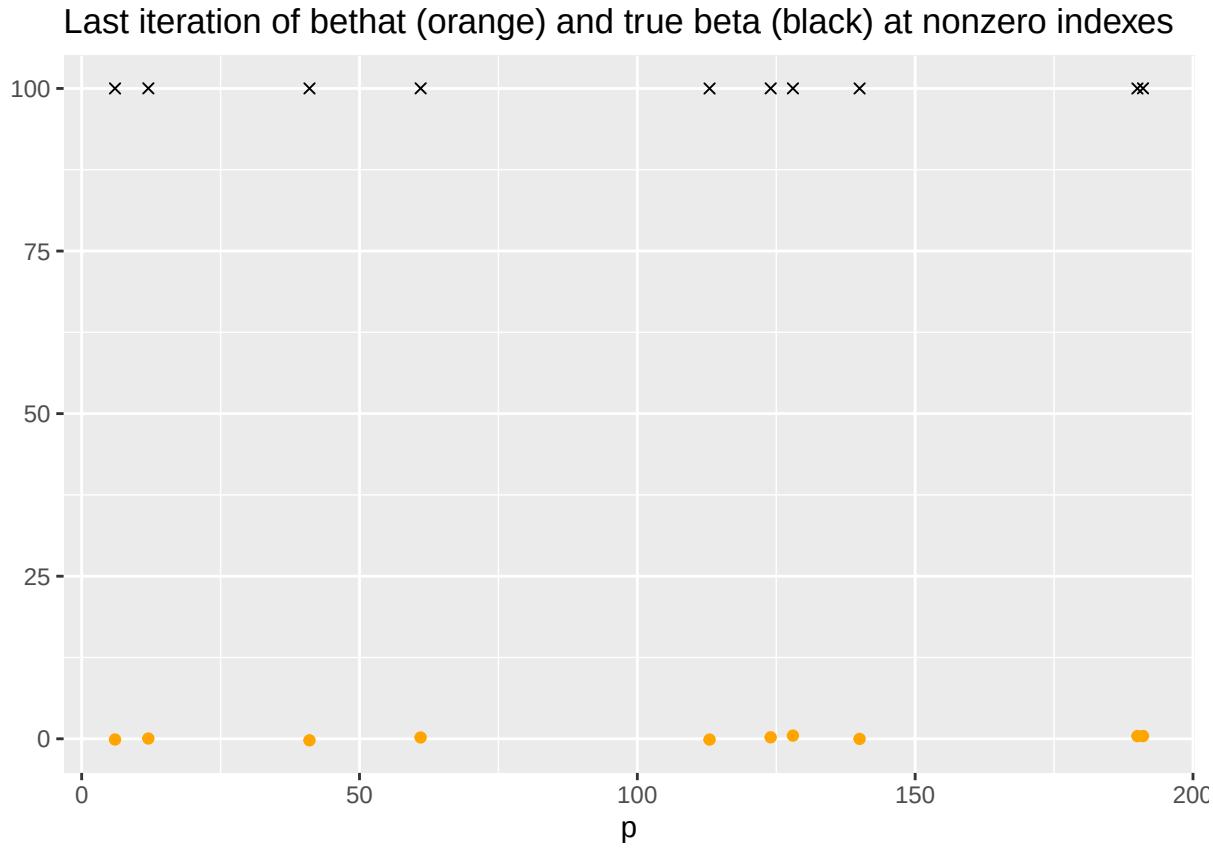
```
runtime_df = data.frame("n"=1:n)
pred_df = data.frame("n"=1:n) # prediction error
est_df = data.frame("n"=1:n) # estimation error

# OGD
temp = my_OGD(X=X, Y=Y, lr=lr, beta_0=rep(0, p))
betahats = temp[[1]]
runtimes = temp[[2]]
runtime_df$ogd = runtimes

pred_df$ogd = plot_prediction_error(betahats, beta, X, Y, title="Online gradient descent (OGD)", type="p
```



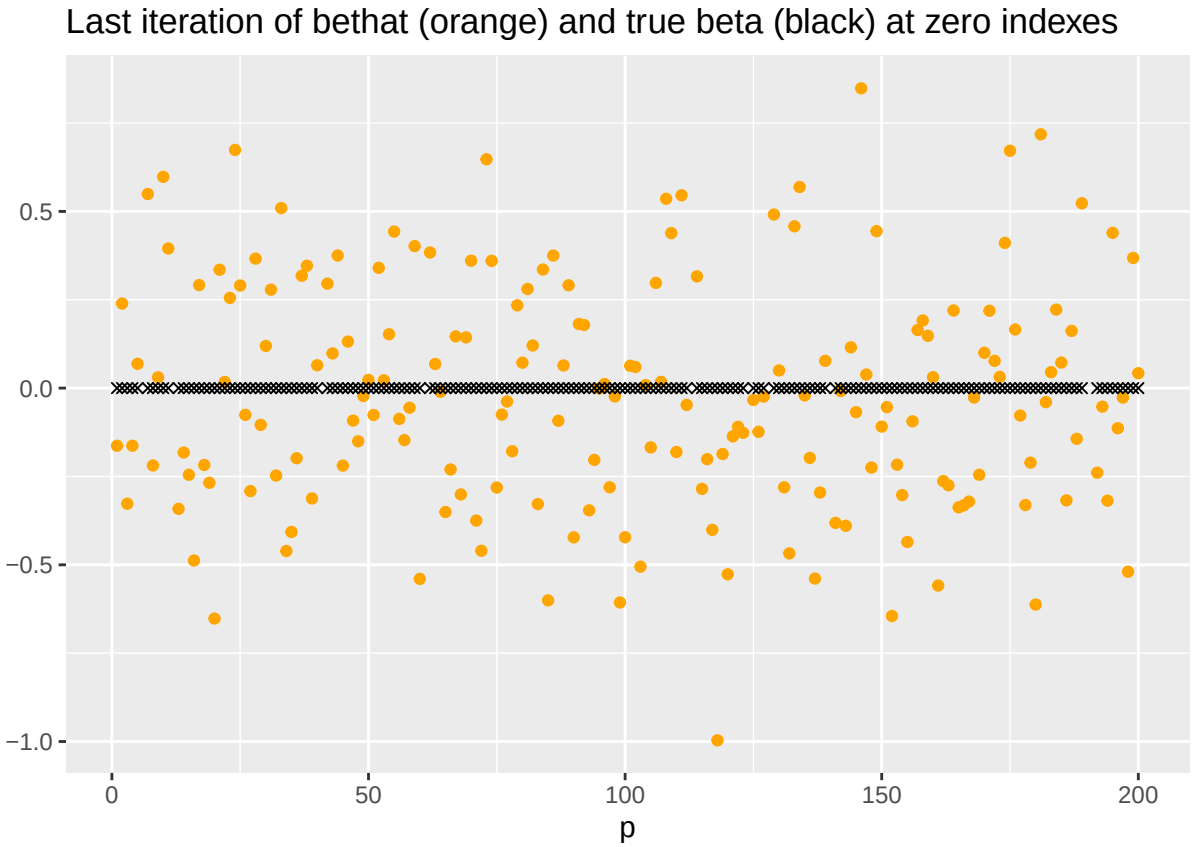

```
est_df$ogd = plot_prediction_error(betahats, beta, X, Y, title="Online gradient descent (OGD)", type="es
```




```
plot_betas(betahats, beta, nonzero_indexes, nonzero=T) # nonzero indexes
```

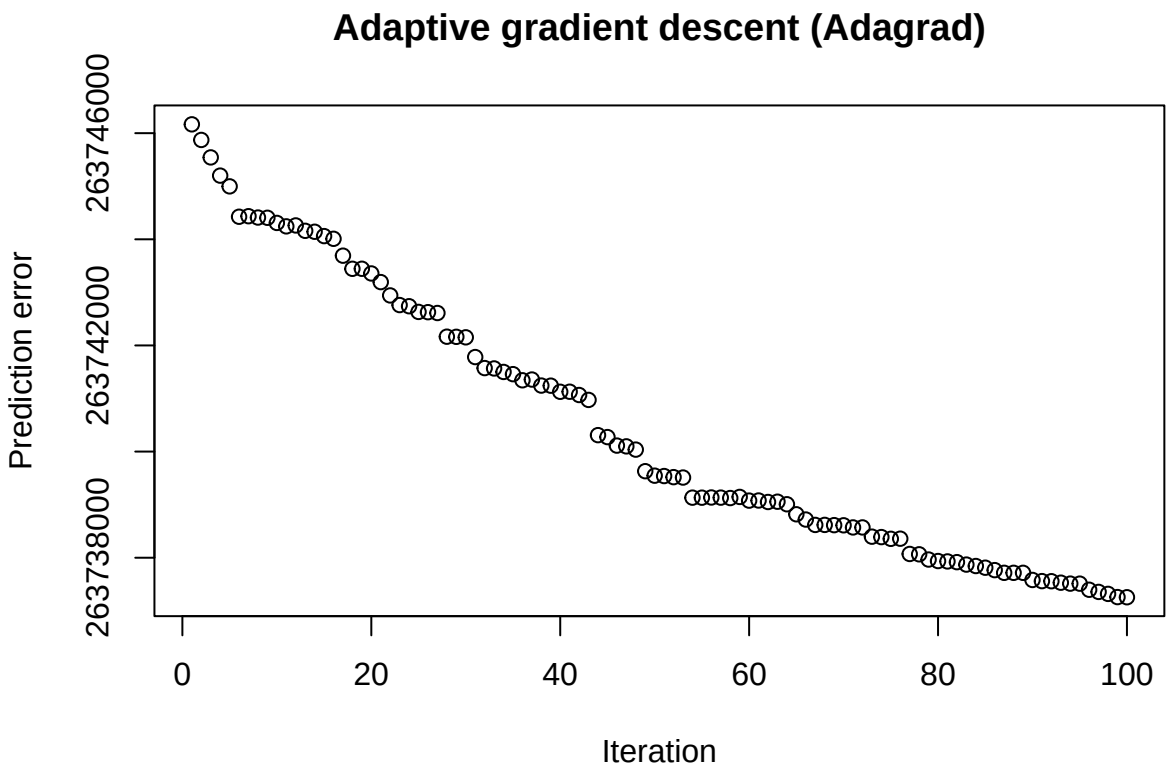



```
plot_betas(betahats, beta, nonzero_indexes, nonzero=F) # zero indexes
```

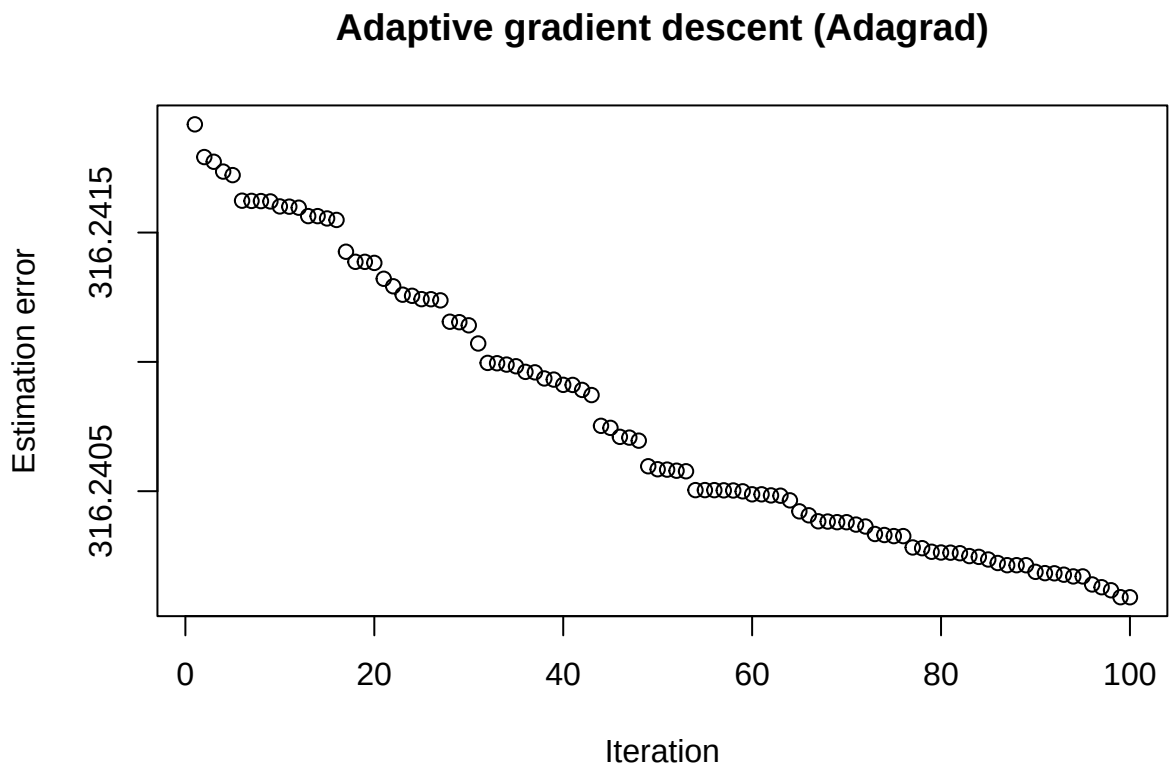


Adagrad: regression

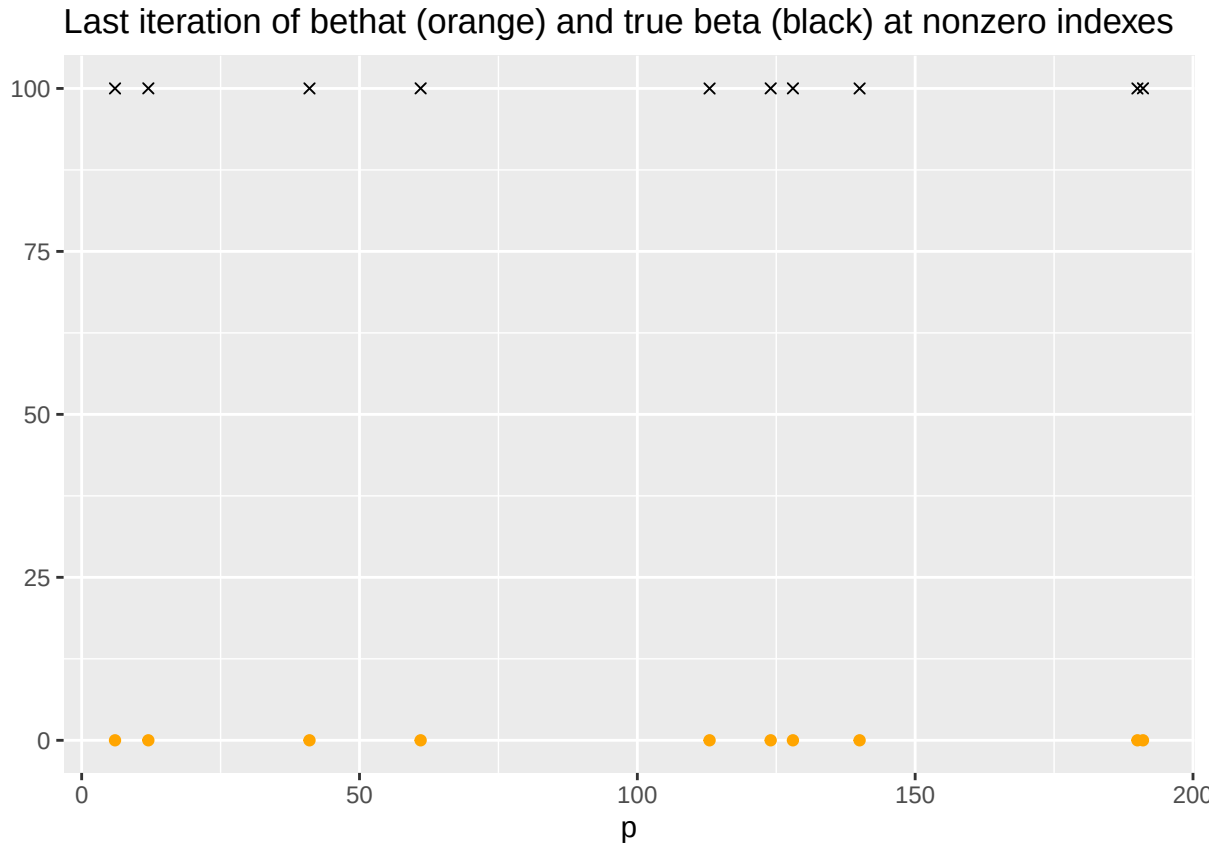
```
# Adagrad
temp = my_adagrad(X=X, Y=Y, lr=lr, beta_0=rep(0, p))
betahats = temp[[1]]
runtimes = temp[[2]]
runtime_df$adagrad = runtimes

pred_df$adagrad = plot_prediction_error(betahats, beta, X, Y, title="Adaptive gradient descent (Adagrad)
```



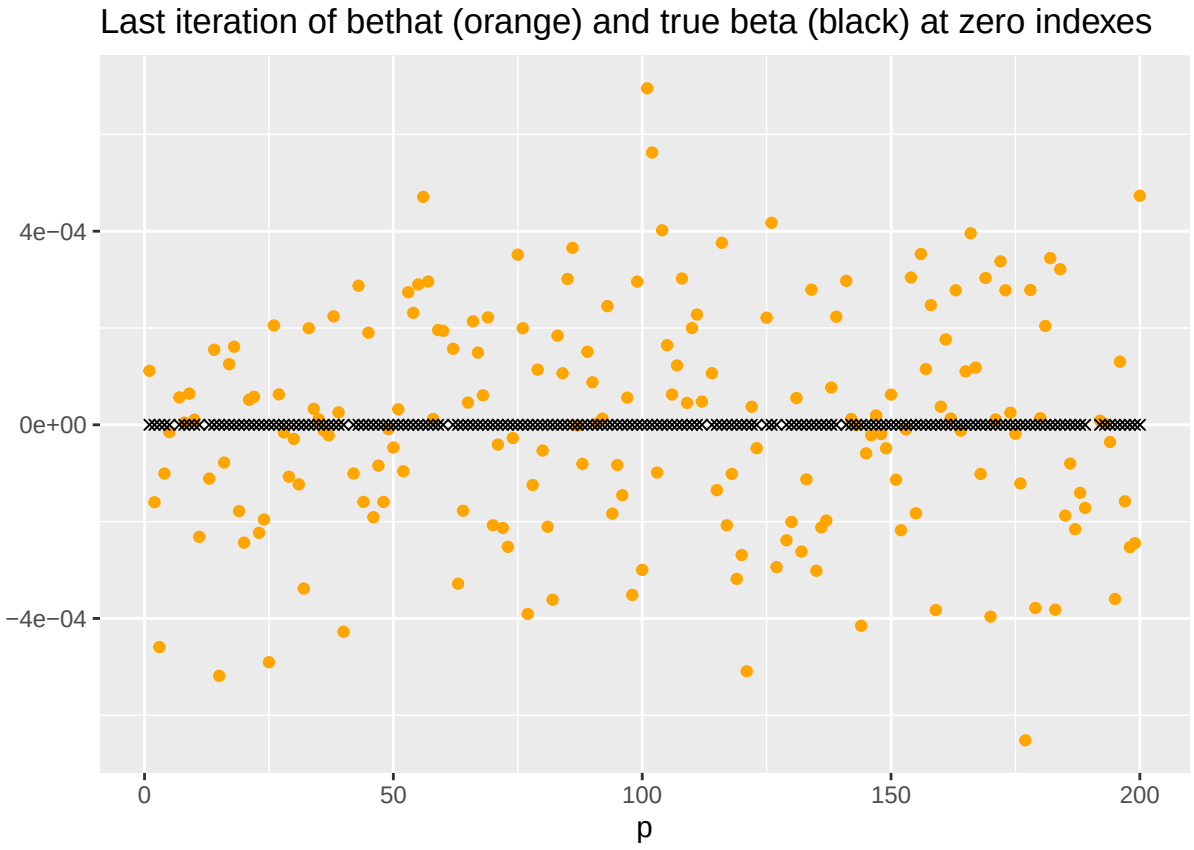

```
est_df$adagrad = plot_prediction_error(betahats, beta, X, Y, title="Adaptive gradient descent (Adagrad)")
```




```
plot_betas(betahats, beta, nonzero_indexes, nonzero=T)
```

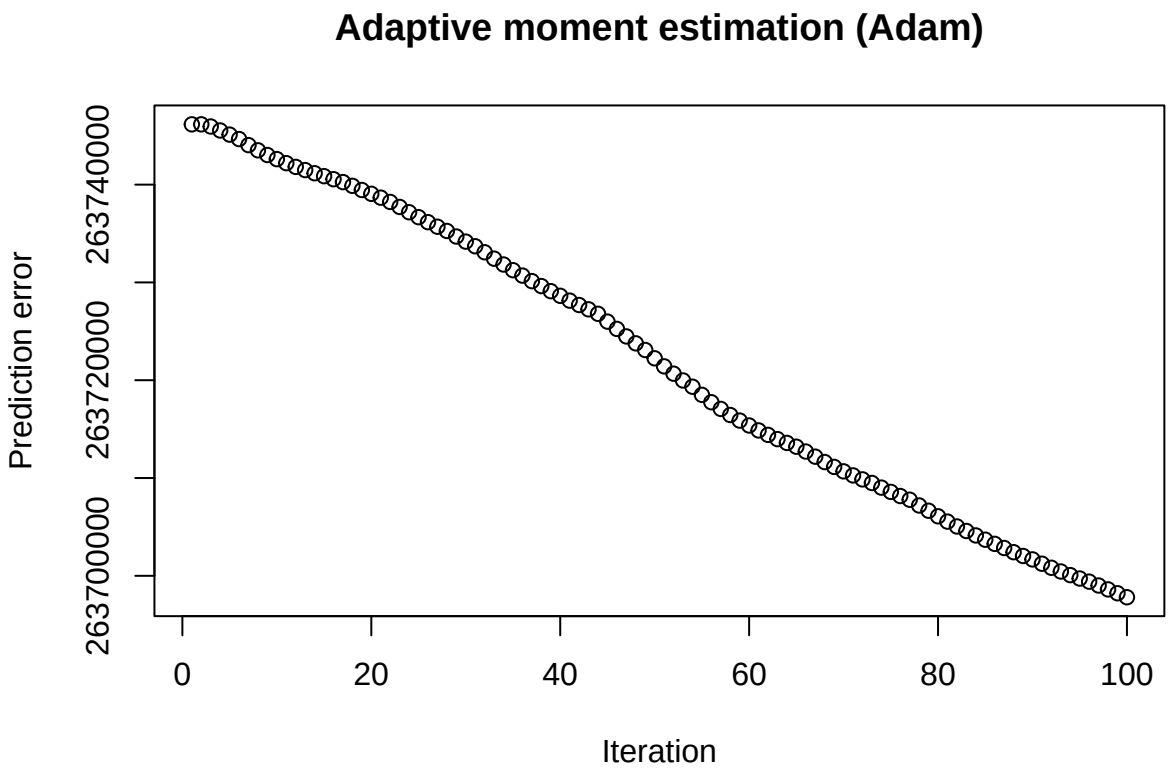



```
plot_betas(betahats, beta, nonzero_indexes, nonzero=F)
```

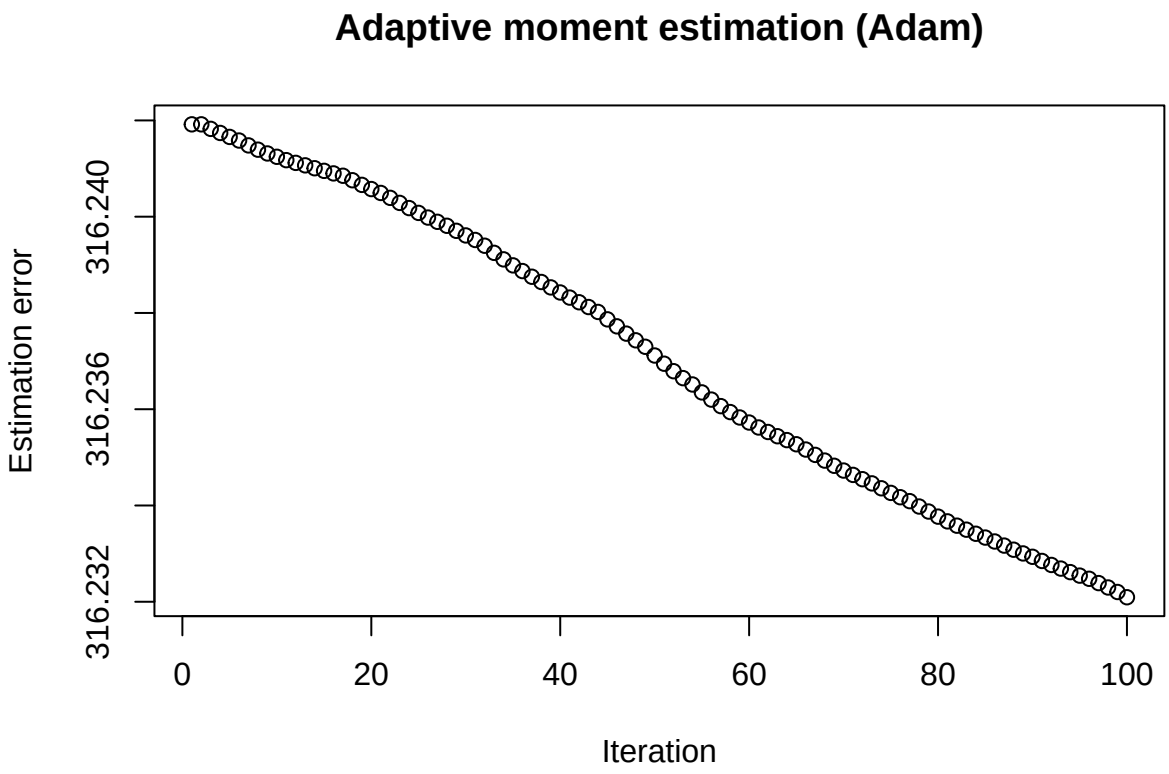


Adam: regression

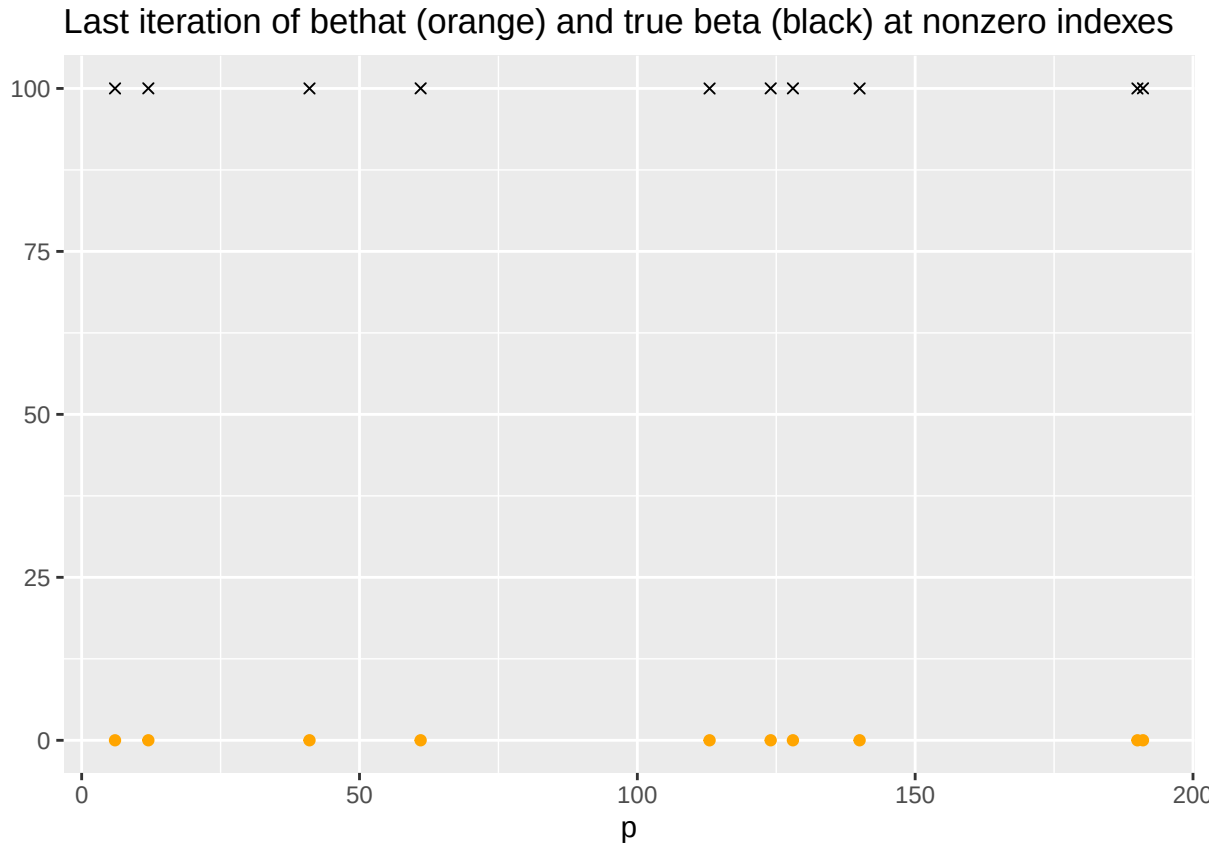
```
# Adam
temp = my_adam(X=X, Y=Y, lr=lr, beta_0=rep(0, p), rho_1=0.9, rho_2=0.999, epsilon=1e-8)
betahats = temp[[1]]
runtimes = temp[[2]]
runtime_df$adam = runtimes

pred_df$adam = plot_prediction_error(betahats, beta, X, Y, title="Adaptive moment estimation (Adam)", ty
```



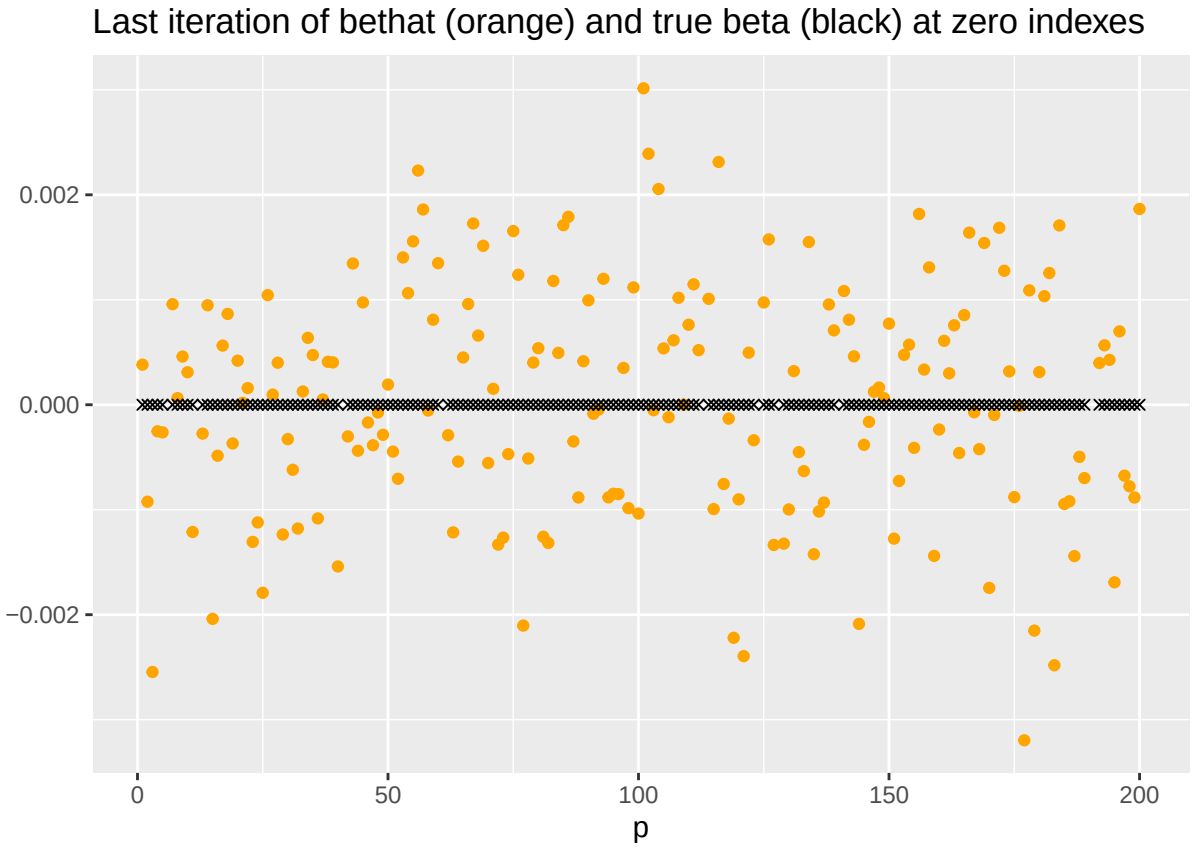

```
est_df$adam = plot_prediction_error(betahats, beta, X, Y, title="Adaptive moment estimation (Adam)", ty
```




```
plot_betas(betahats, beta, nonzero_indexes, nonzero=T)
```

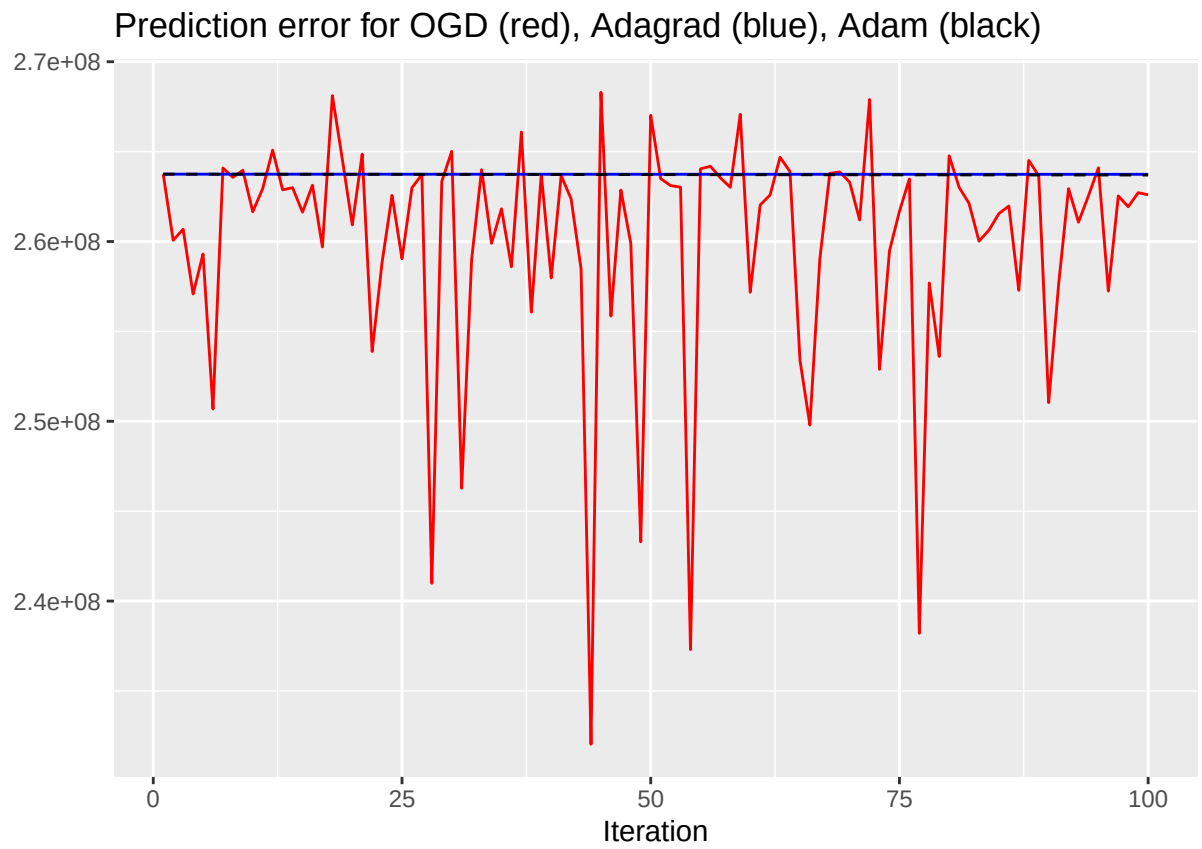



```
plot_betas(betahats, beta, nonzero_indexes, nonzero=F)
```

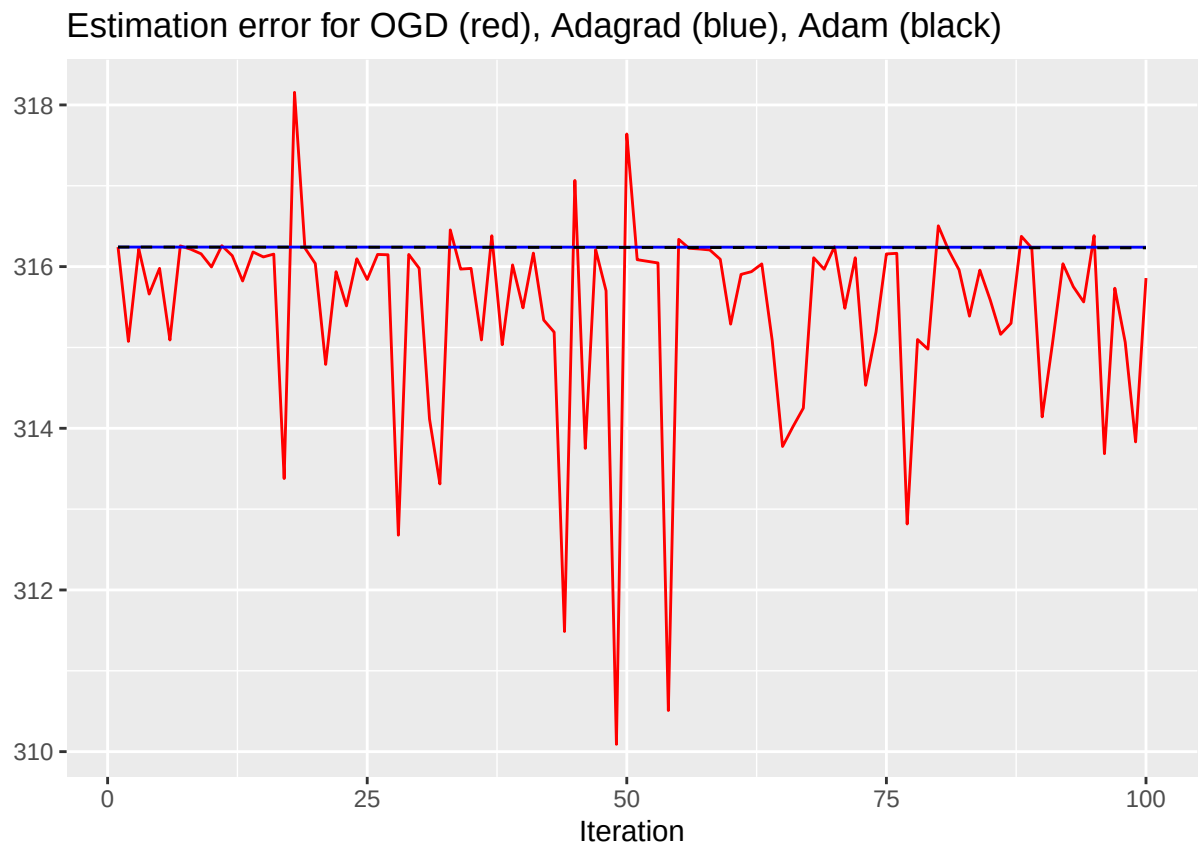


Performance comparison: regression

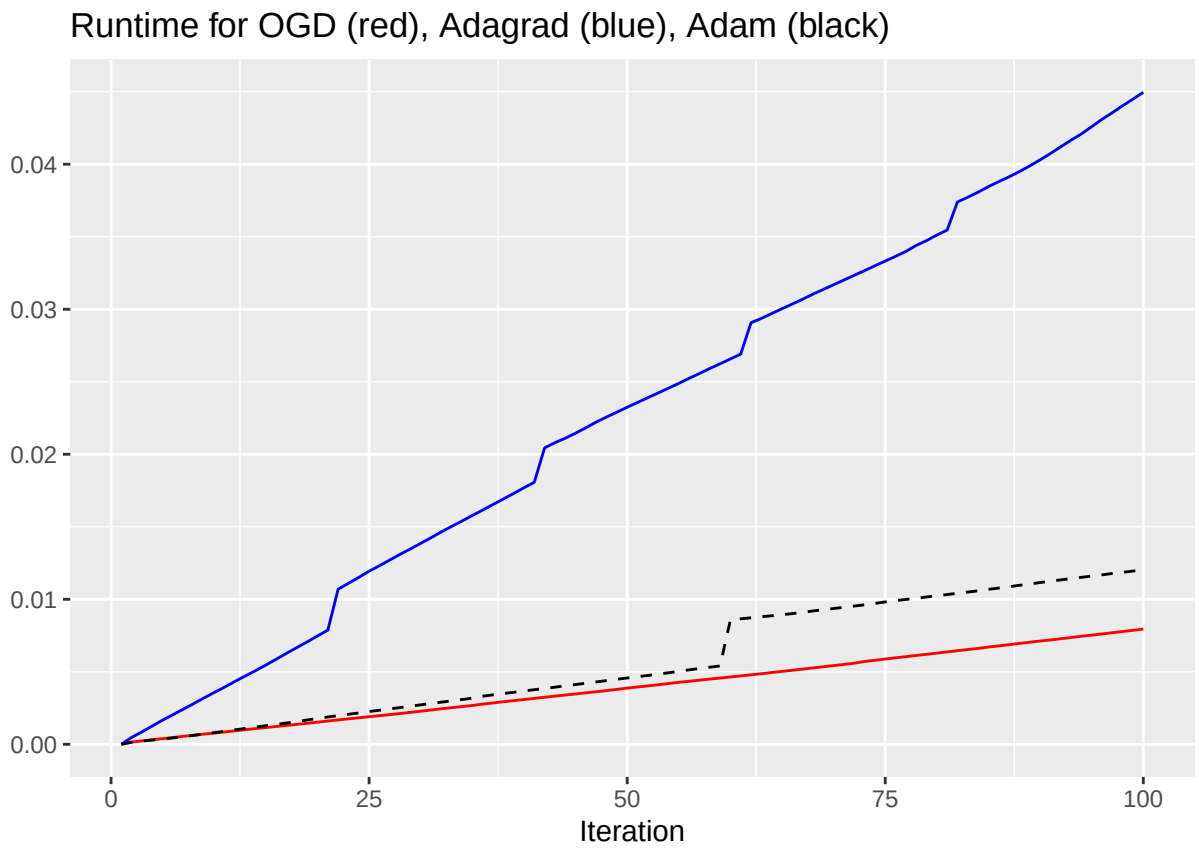
```
# plot prediction error for ogd, adagrad, adam
ggplot(pred_df, aes(x=n)) +
  geom_line(aes(y=ogd), color="red") +
  geom_line(aes(y=adagrad), color="blue") +
  geom_line(aes(y=adam), linetype="dashed") +
  xlab("Iteration") +
  ylab("") +
  ggtitle("Prediction error for OGD (red), Adagrad (blue), Adam (black)")
```



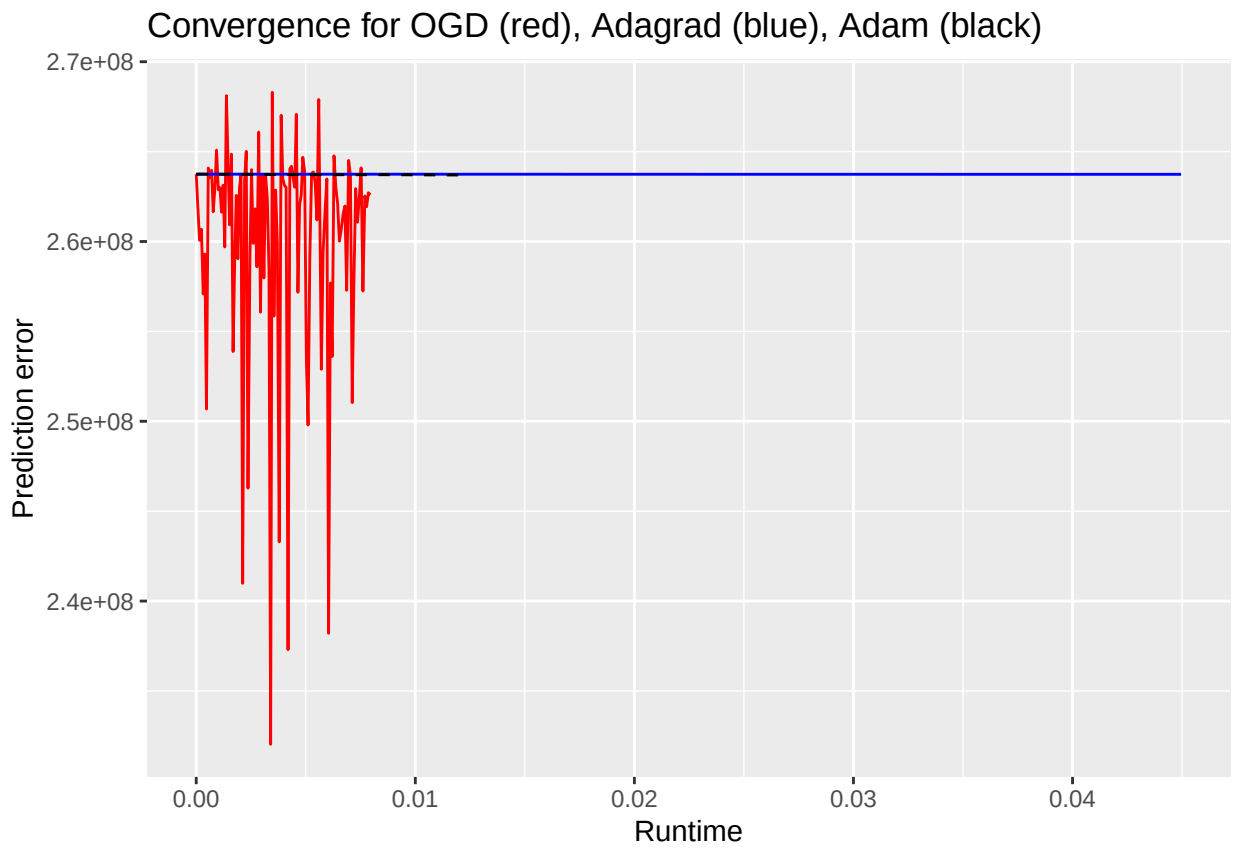

```
# plot estimation error for ogd, adagrad, adam
ggplot(est_df, aes(x=n)) +
  geom_line(aes(y=ogd), color="red") +
  geom_line(aes(y=adagrad), color="blue") +
  geom_line(aes(y=adam), linetype="dashed") +
  xlab("Iteration") +
  ylab("") +
  ggtitle("Estimation error for OGD (red), Adagrad (blue), Adam (black)")
```




```
# plot runtime for ogd, adagrad, adam
ggplot(runtime_df, aes(x=n)) +
  geom_line(aes(y=ogd), color="red") +
  geom_line(aes(y=adagrad), color="blue") +
  geom_line(aes(y=adam), linetype="dashed") +
  xlab("Iteration") +
  ylab("") +
  ggtitle("Runtime for OGD (red), Adagrad (blue), Adam (black)")
```




```
conv_df = data.frame("n"=1:n) # plot convergence
conv_df$err_ogd = pred_df$ogd
conv_df$err_adagrad = pred_df$adagrad
conv_df$err_adam = pred_df$adam
conv_df$time_ogd = runtime_df$ogd
conv_df$time_adagrad = runtime_df$adagrad
conv_df$time_adam = runtime_df$adam

# plot convergence for ogd, adagrad, adam
ggplot(conv_df) +
  geom_line(aes(x=time_ogd, y=err_ogd), color="red") +
  geom_line(aes(x=time_adagrad, y=err_adagrad), color="blue") +
  geom_line(aes(x=time_adam, y=err_adam), linetype="dashed") +
  xlab("Runtime") +
  ylab("Prediction error") +
  ggtitle("Convergence for OGD (red), Adagrad (blue), Adam (black)")
```



Classification

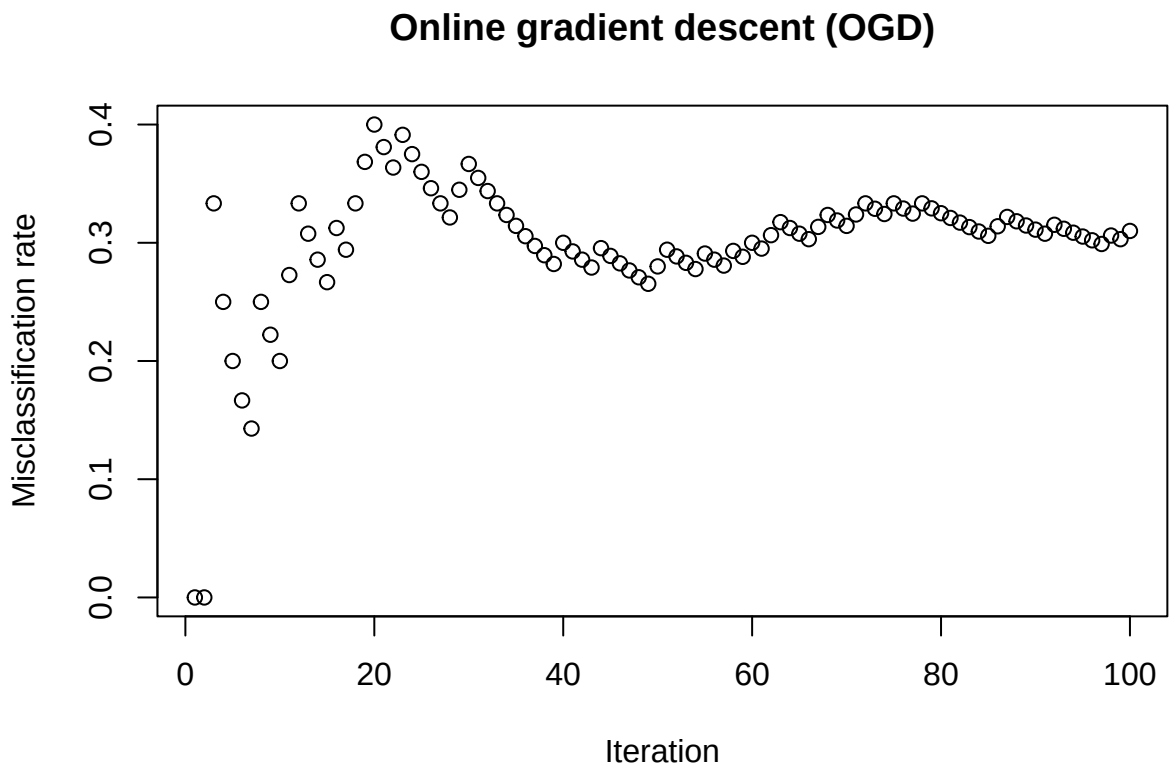
Generate data for logistic regression

```
# simulate data: classification
# X, beta same as above
probs = 1/(1+exp(-X%*%beta))
Y = rbinom(n=n, size=1, prob = probs) # Bernoulli
```

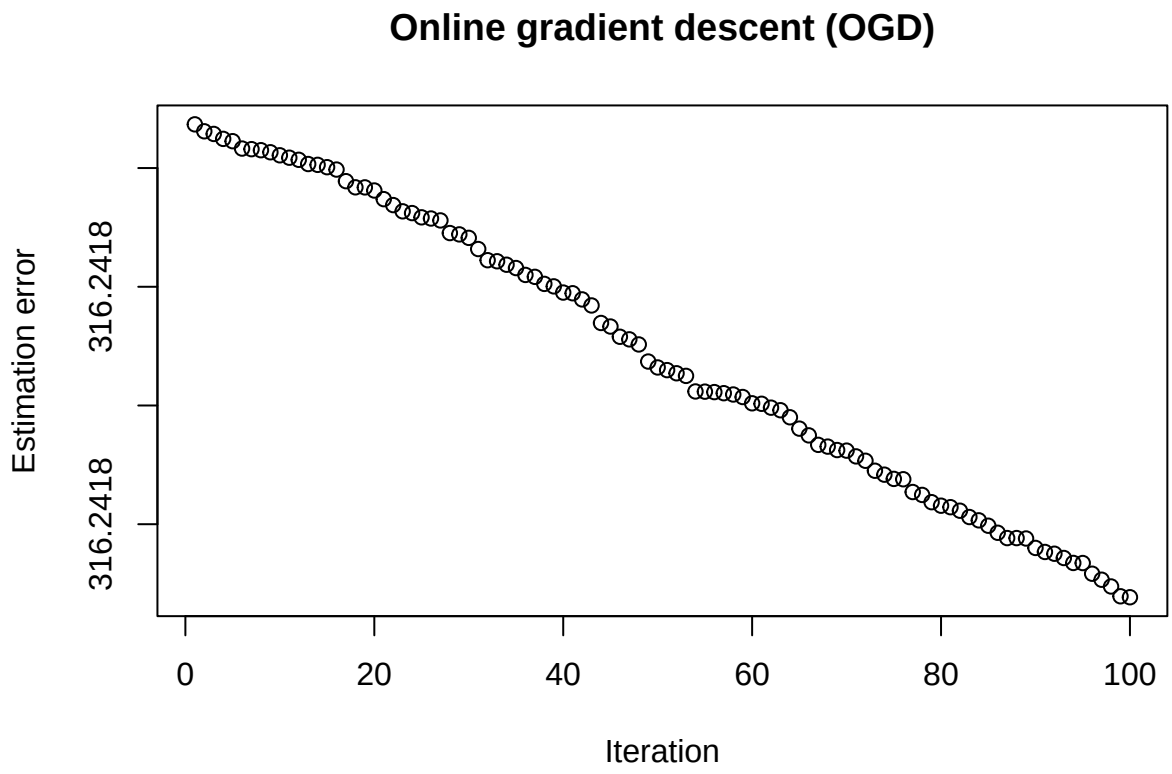
OGD: classification

```
runtime_df = data.frame("n"=1:n)
pred_df = data.frame("n"=1:n) # prediction error
est_df = data.frame("n"=1:n) # estimation error

# OGD
temp = my_OGD(X=X, Y=Y, lr=lr, beta_0=rep(0, p), regression=F)
betahats = temp[[1]]
runtimes = temp[[2]]
bs = temp[[3]]
runtime_df$ogd = runtimes
```

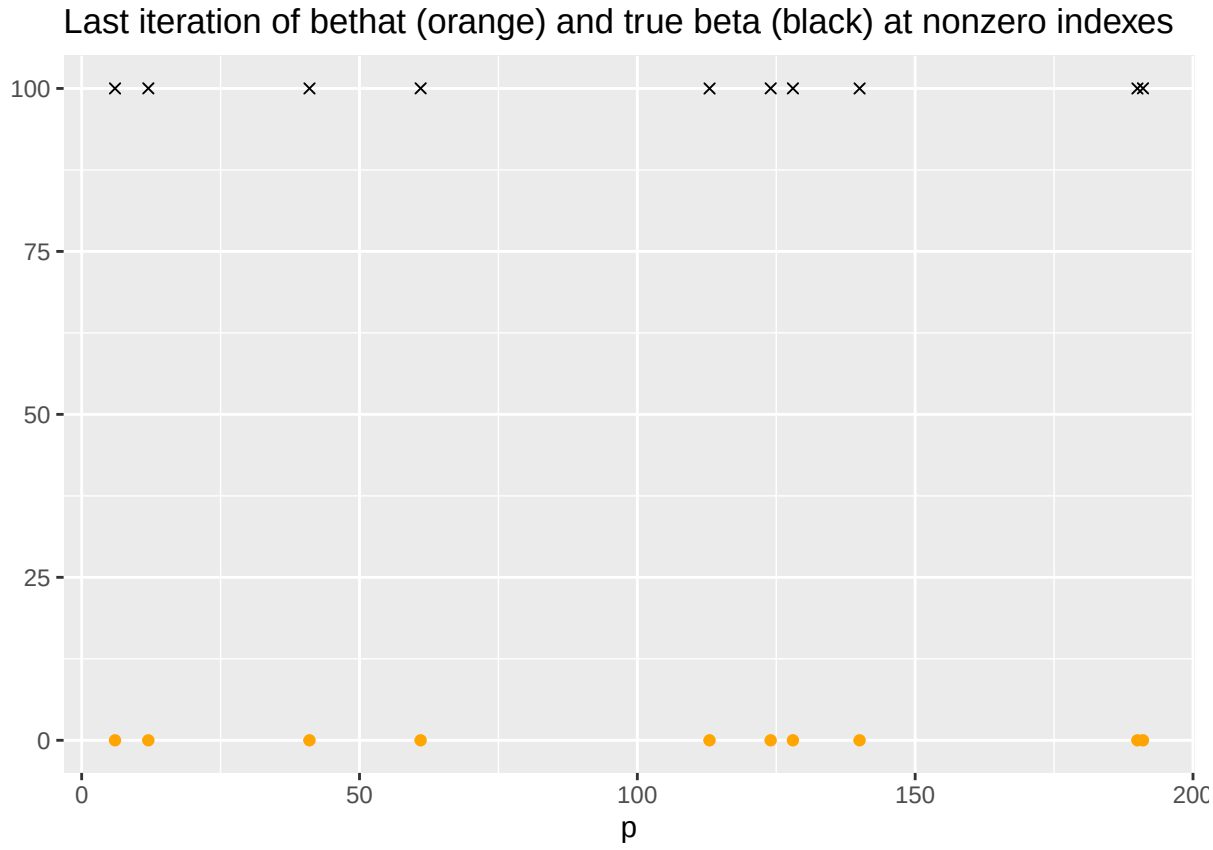
```
pred_df$ogd = plot_prediction_error(betahats, beta, X, Y, title="Online gradient descent (OGD)", type="p
```



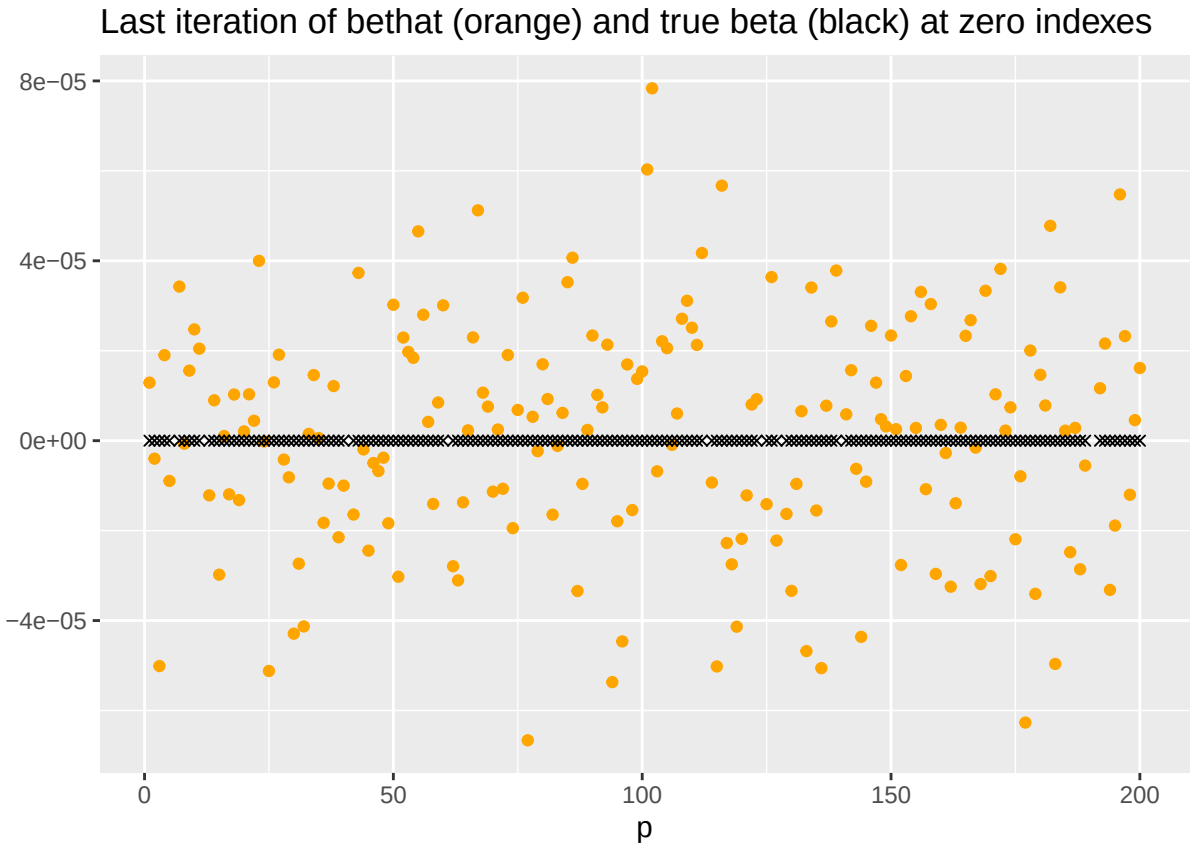

```
est_df$ogd = plot_prediction_error(betahats, beta, X, Y, title="Online gradient descent (OGD)", type="es
```




```
plot_betas(betahats, beta, nonzero_indexes, nonzero=T) # nonzero indexes
```

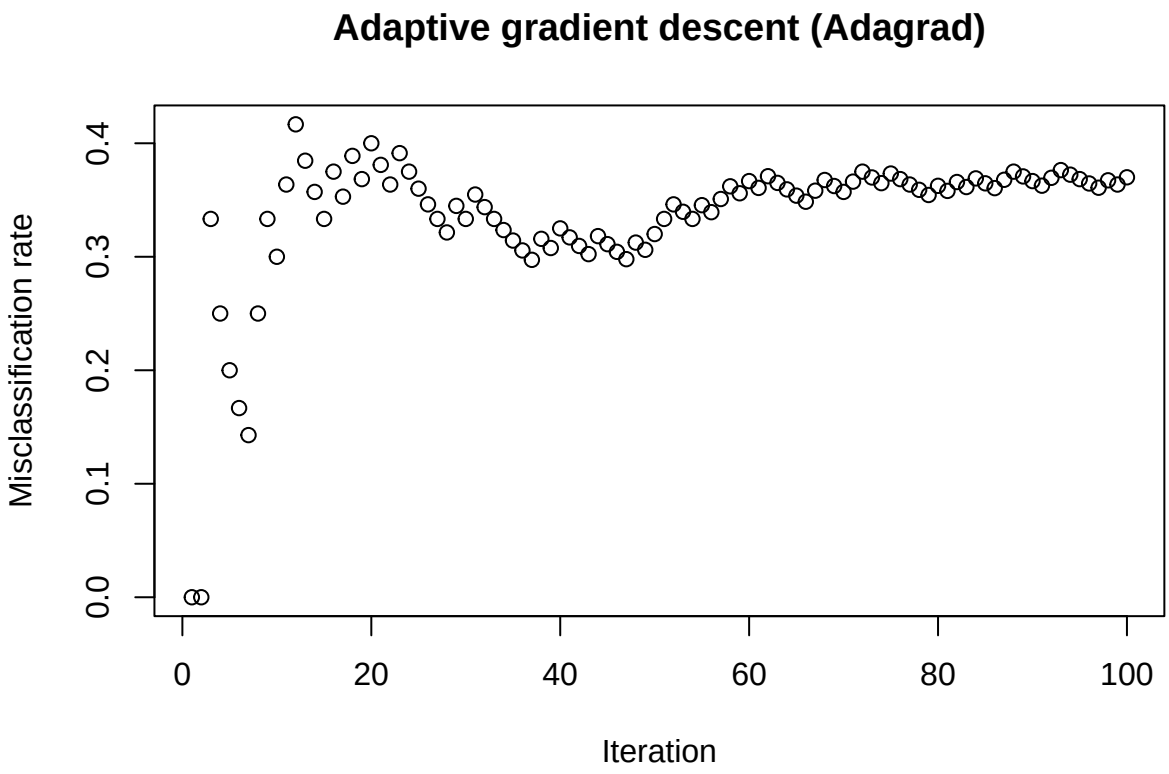



```
plot_betas(betahats, beta, nonzero_indexes, nonzero=F) # zero indexes
```

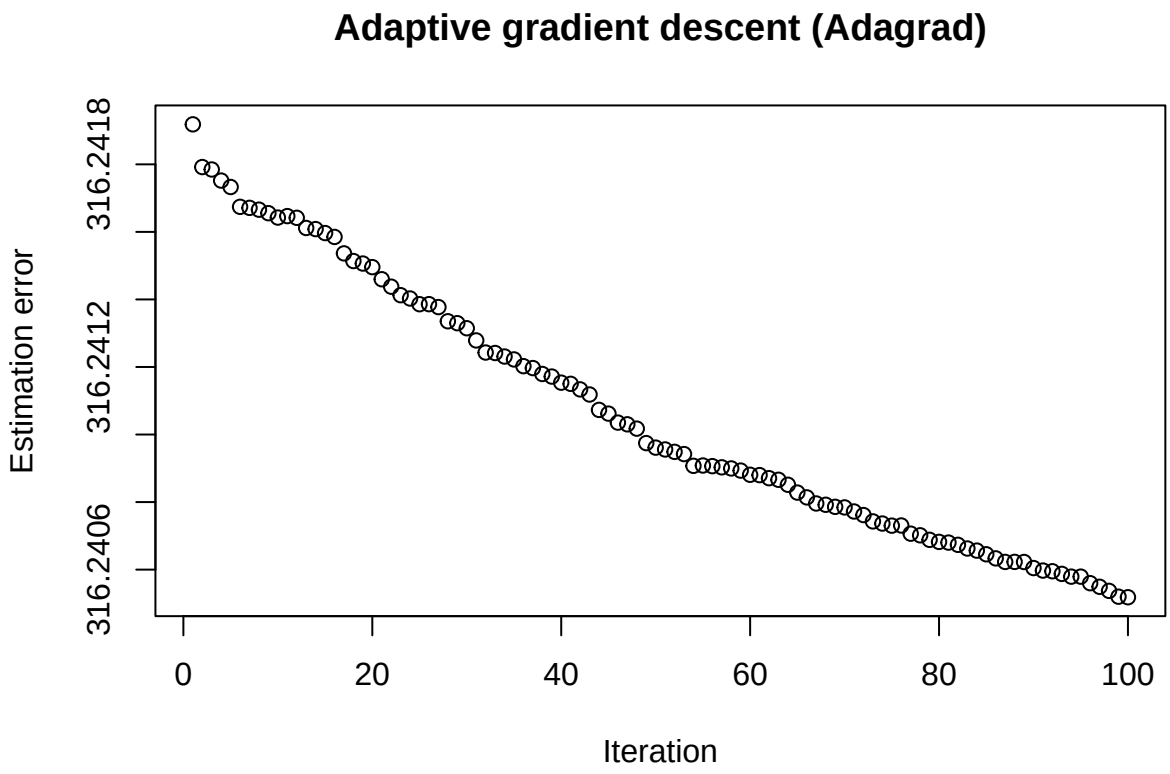


Adagrad: classification

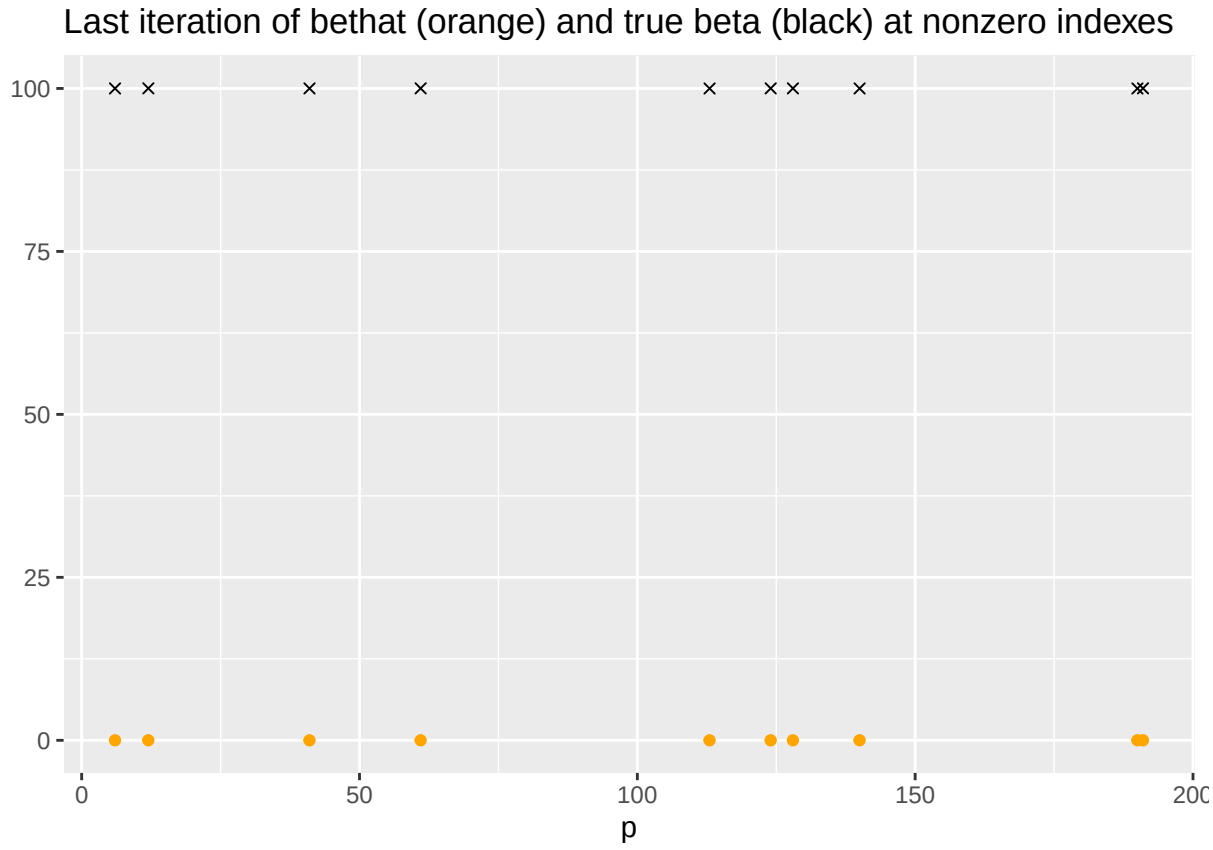
```
# Adagrad
temp = my_adagrad(X=X, Y=Y, lr=lr, beta_0=rep(0, p), regression=F)
betahats = temp[[1]]
runtimes = temp[[2]]
runtime_df$adagrad = runtimes

pred_df$adagrad = plot_prediction_error(betahats, beta, X, Y, title="Adaptive gradient descent (Adagrad
```



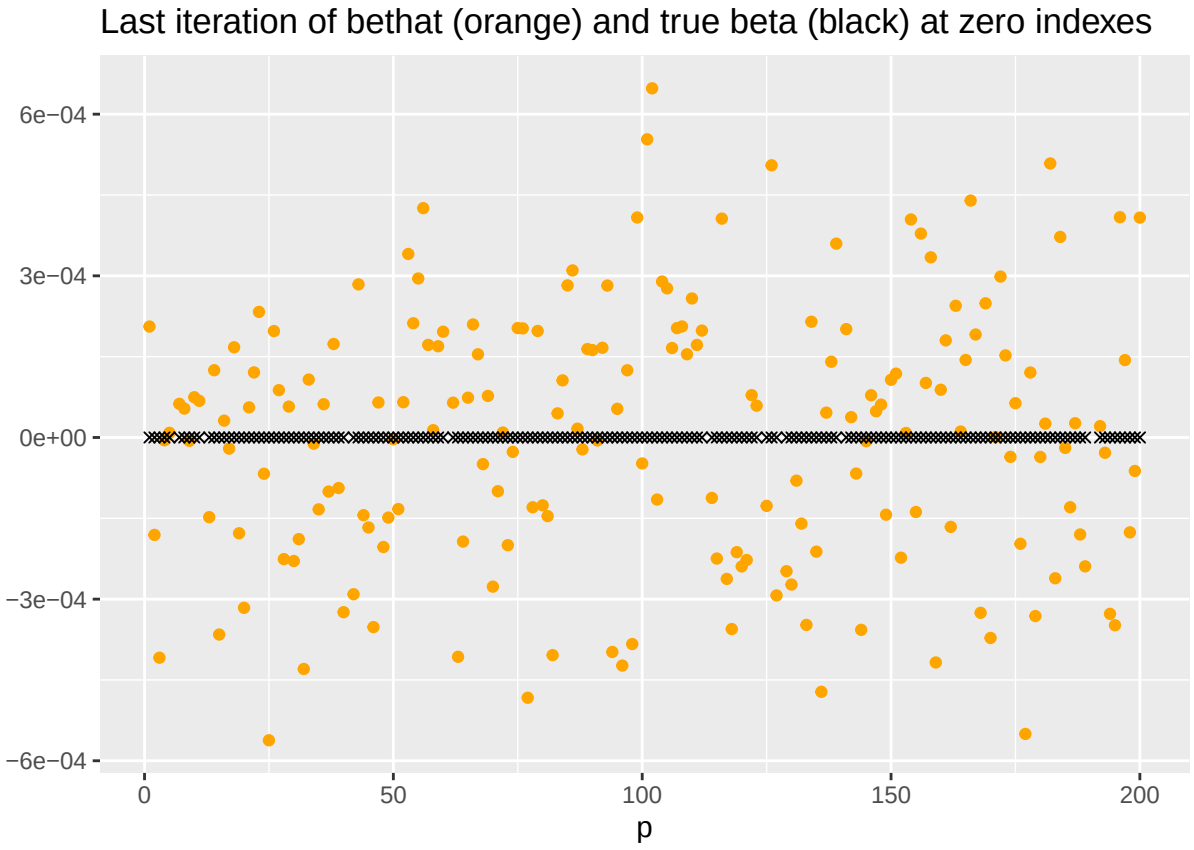

```
est_df$adagrad = plot_prediction_error(betahats, beta, X, Y, title="Adaptive gradient descent (Adagrad)")
```




```
plot_betas(betahats, beta, nonzero_indexes, nonzero=T)
```

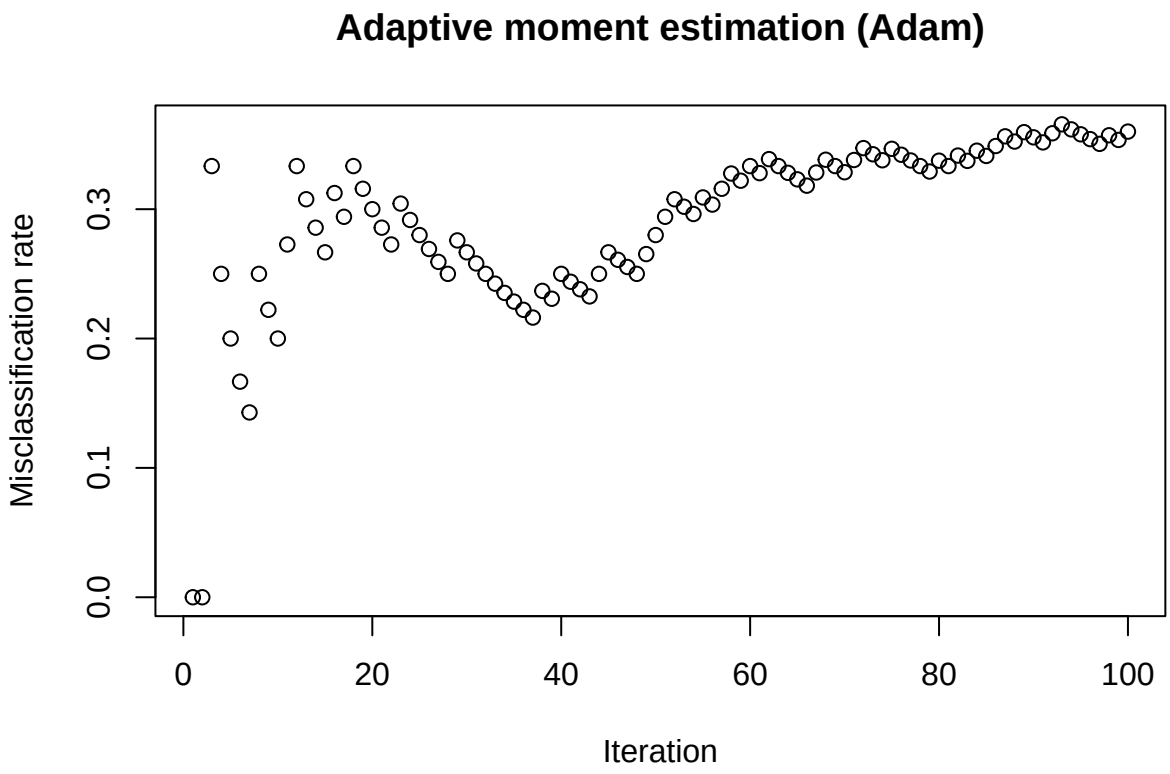



```
plot_betas(betahats, beta, nonzero_indexes, nonzero=F)
```

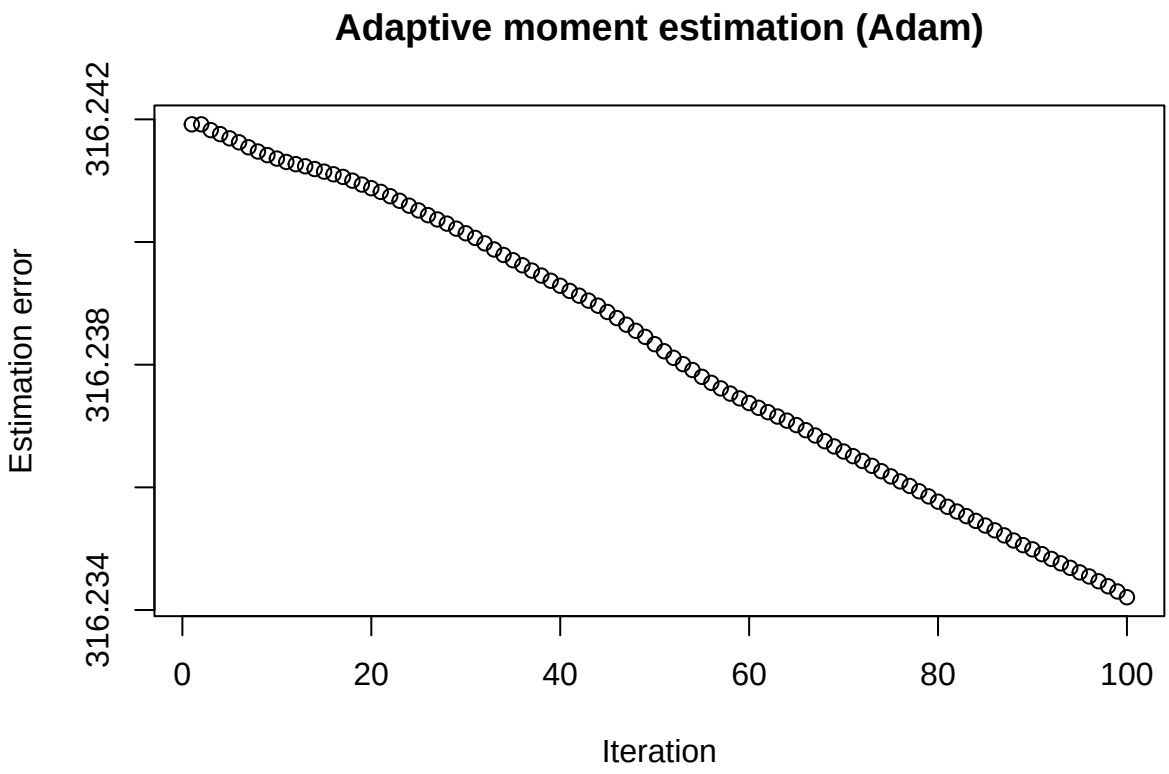


Adam: classification

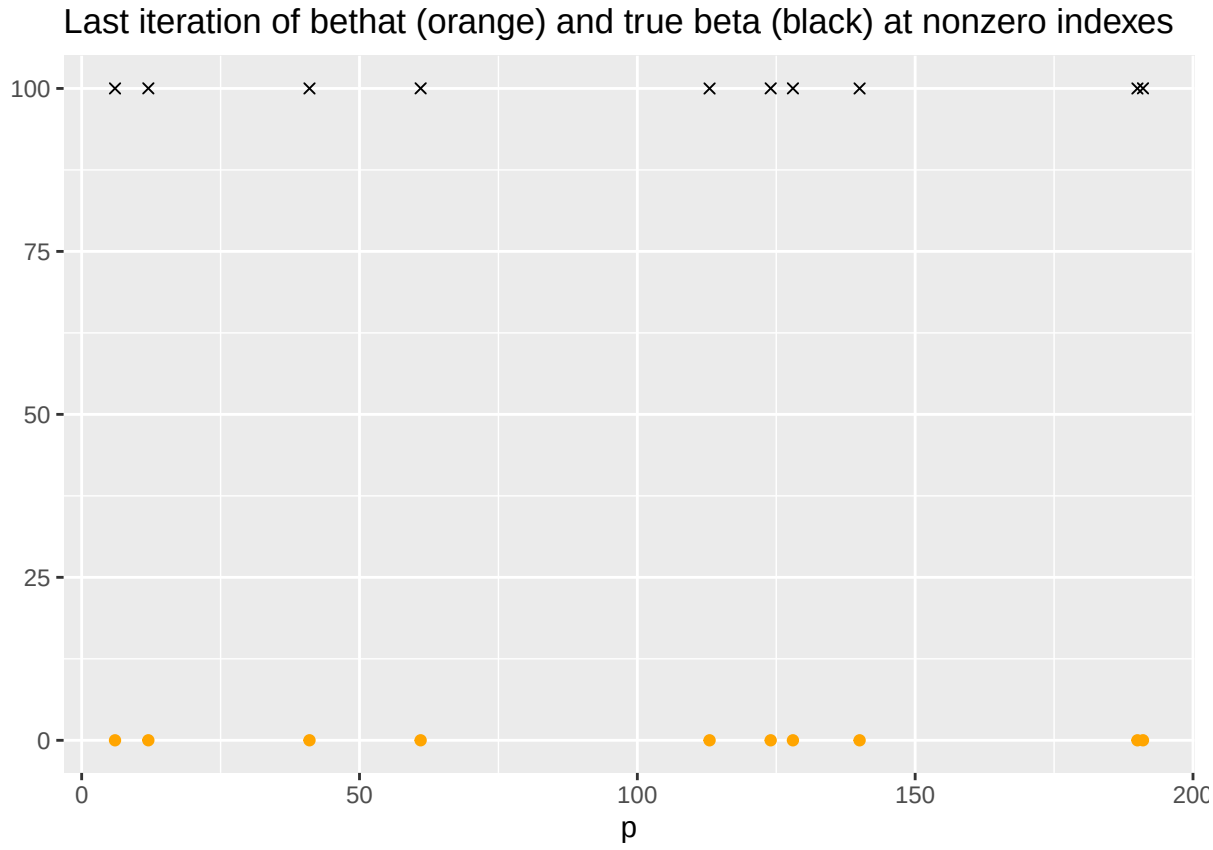
```
# Adam
temp = my_adam(X=X, Y=Y, lr=lr, beta_0=rep(0, p), rho_1=0.9, rho_2=0.999, epsilon=1e-8, regression=F)
betahats = temp[[1]]
runtimes = temp[[2]]
runtime_df$adam = runtimes

pred_df$adam = plot_prediction_error(betahats, beta, X, Y, title="Adaptive moment estimation (Adam)", ty
```



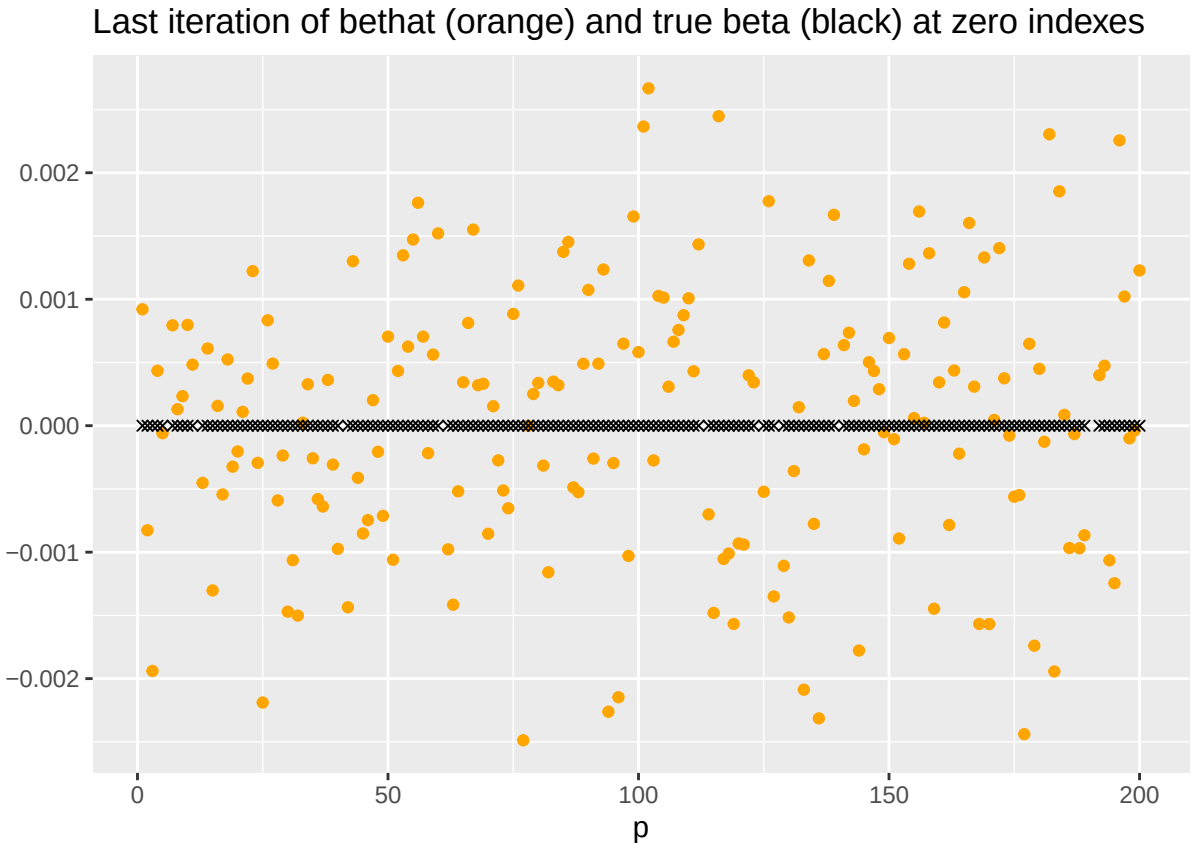

```
est_df$adam = plot_prediction_error(betahats, beta, X, Y, title="Adaptive moment estimation (Adam)", ty
```




```
plot_betas(betahats, beta, nonzero_indexes, nonzero=T)
```

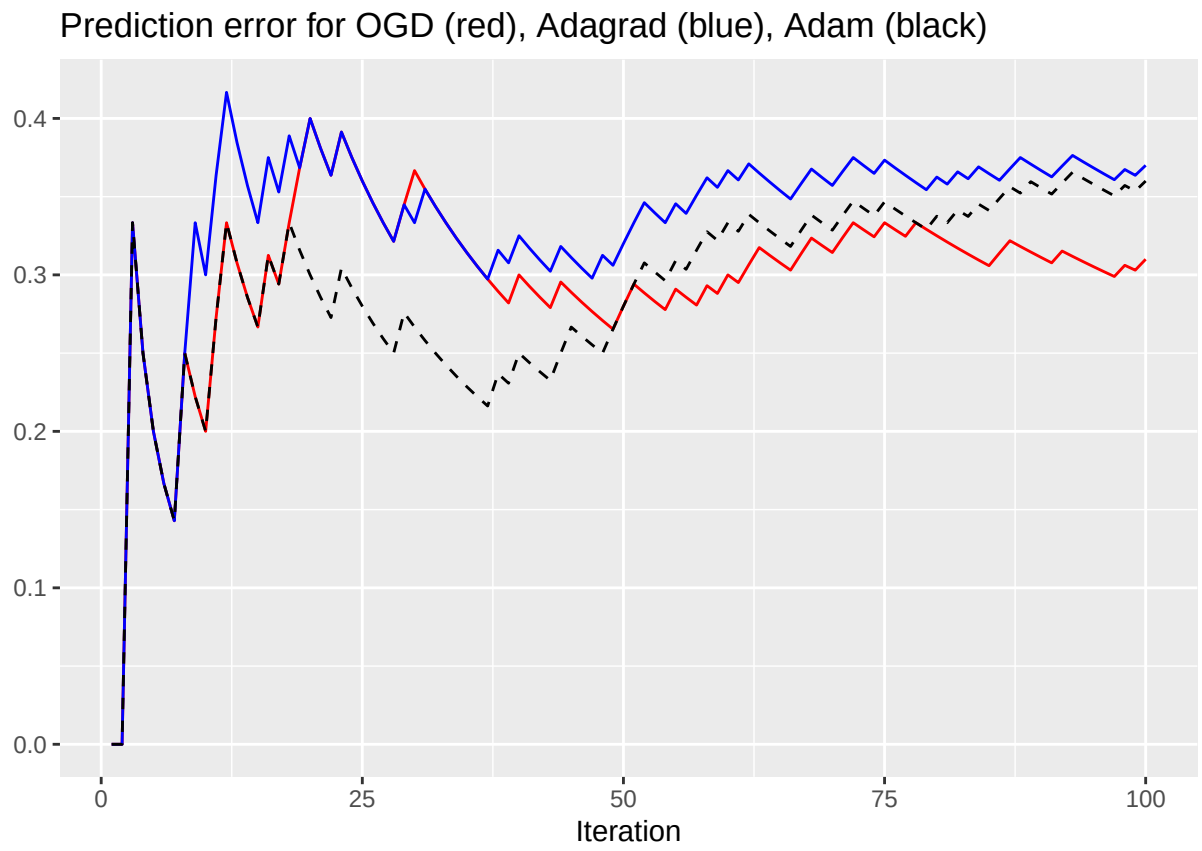



```
plot_betas(betahats, beta, nonzero_indexes, nonzero=F)
```

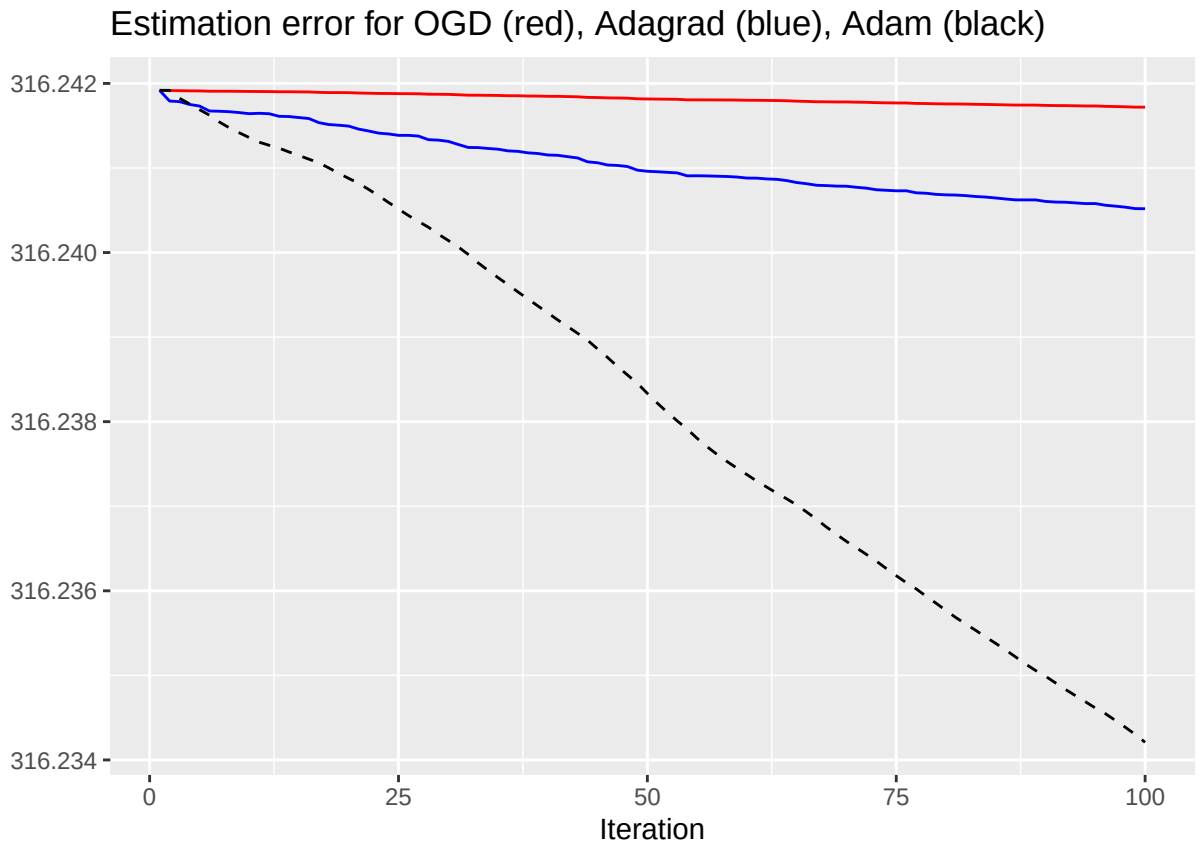


Performance comparison: classification

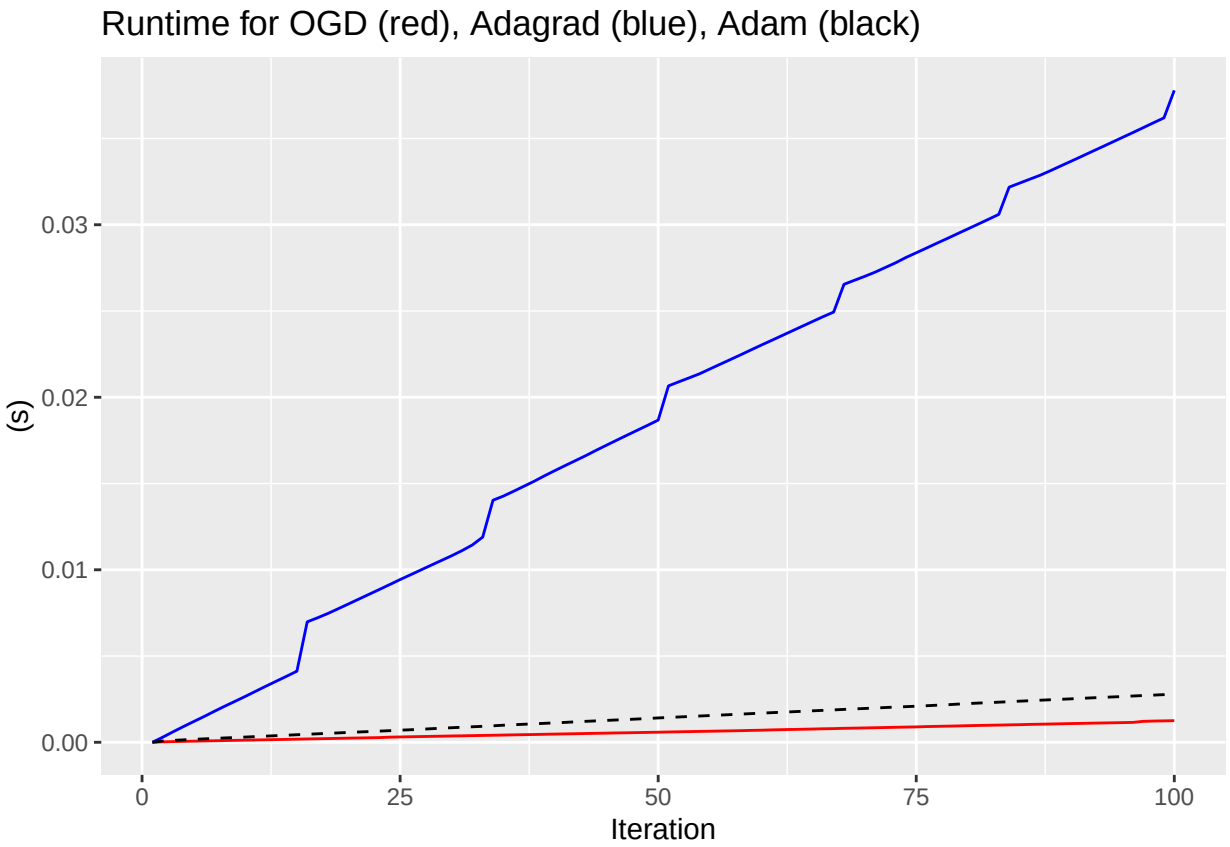
```
# plot prediction error for ogd, adagrad, adam
ggplot(pred_df, aes(x=n)) +
  geom_line(aes(y=ogd), color="red") +
  geom_line(aes(y=adagrad), color="blue") +
  geom_line(aes(y=adam), linetype="dashed") +
  xlab("Iteration") +
  ylab("") +
  ggtitle("Prediction error for OGD (red), Adagrad (blue), Adam (black)")
```



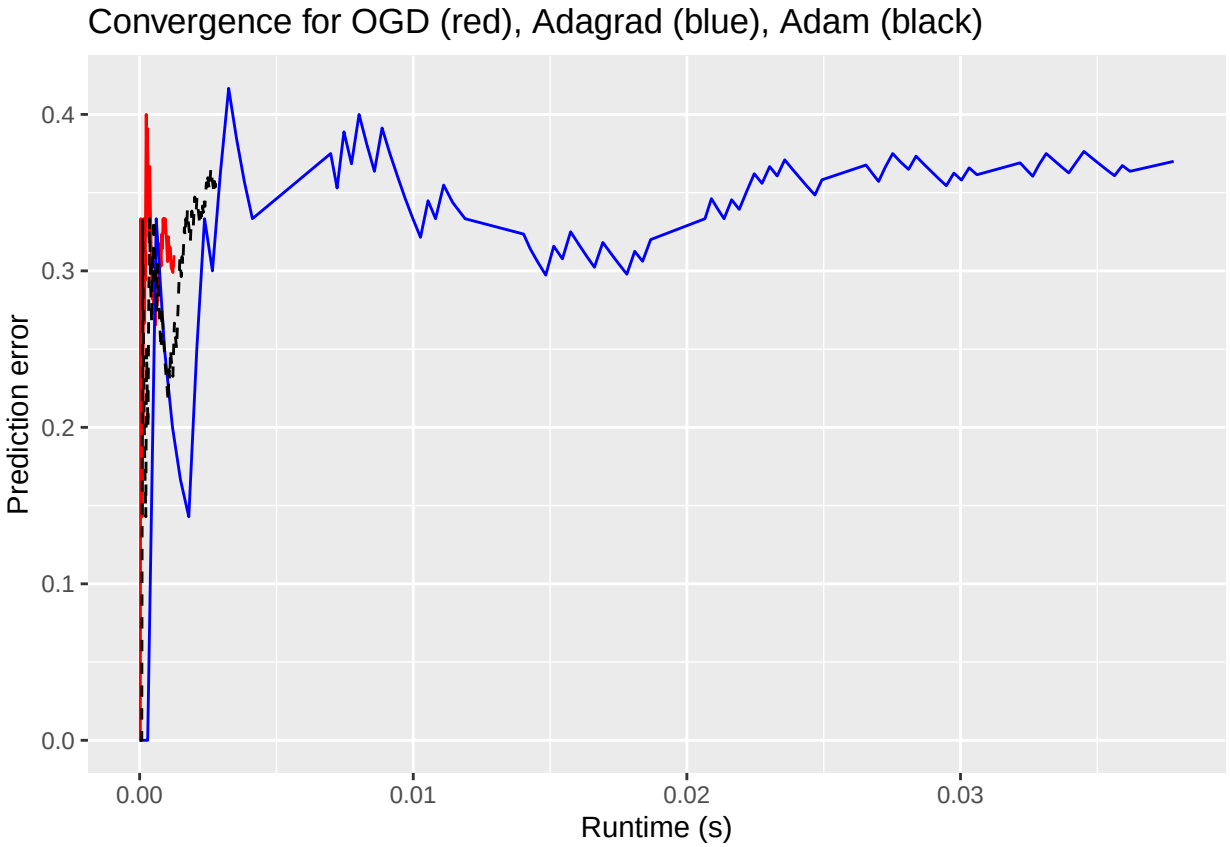

```
# plot estimation error for ogd, adagrad, adam
ggplot(est_df, aes(x=n)) +
  geom_line(aes(y=ogd), color="red") +
  geom_line(aes(y=adagrad), color="blue") +
  geom_line(aes(y=adam), linetype="dashed") +
  xlab("Iteration") +
  ylab("") +
  ggtitle("Estimation error for OGD (red), Adagrad (blue), Adam (black)")
```




```
# plot runtime for ogd, adagrad, adam
ggplot(runtime_df, aes(x=n)) +
  geom_line(aes(y=ogd), color="red") +
  geom_line(aes(y=adagrad), color="blue") +
  geom_line(aes(y=adam), linetype="dashed") +
  xlab("Iteration") +
  ylab("(s)") +
  ggtitle("Runtime for OGD (red), Adagrad (blue), Adam (black)")
```




```
conv_df = data.frame("n"=1:n) # plot convergence
conv_df$err_ogd = pred_df$ogd
conv_df$err_adagrad = pred_df$adagrad
conv_df$err_adam = pred_df$adam
conv_df$time_ogd = runtime_df$ogd
conv_df$time_adagrad = runtime_df$adagrad
conv_df$time_adam = runtime_df$adam

# plot convergence for ogd, adagrad, adam
ggplot(conv_df) +
  geom_line(aes(x=time_ogd, y=err_ogd), color="red") +
  geom_line(aes(x=time_adagrad, y=err_adagrad), color="blue") +
  geom_line(aes(x=time_adam, y=err_adam), linetype="dashed") +
  xlab("Runtime (s)") +
  ylab("Prediction error") +
  ggtitle("Convergence for OGD (red), Adagrad (blue), Adam (black)")
```


Convergence for OGD (red), Adagrad (blue), Adam (black)
Prediction error
0.0
0.1
0.2
0.3
0.4
0.00
0.01
0.02
0.03
Runtime (s)