Assume we have iid covariates $X \in \mathbb{R}^{n\times p}$ and $Y \in \mathbb{R}^p$ response. Also, we have $\Omega \in \mathbb{R}^{k\times p}$ random projection matrix which follows $\mathbb{E}[\Omega^T\Omega] = kI_p$ and has Rademacher random variable and iid elements. Let

$$\beta_n^{\Omega} = \Omega^T \operatorname*{argmin}_{\beta\in\mathbb{R}^k} \mathbb{P}_n\{Y_(\Omega X)^T\beta\}^2$$

Now assume we have a distribution for the dimension parameter k, i.e. $k_1,\ldots,k_B \sim Q$, thus giving us B values of Ω (call each individual realization $\Omega(k(b))$ for an integer $k \in [1,B]$ and a potential estimator]

$$\widehat{\beta}_n^{\text{ave}} = \frac{1}{B}\sum_{b=1}^{B} \beta_n^{\Omega(k_b)}$$

I implemented this streaming approach with k following a distribution by starting with a specific case and then using that to create a generalized function. The following gives some context for the R code.

We assume k follows a uniform distribution centered at an integer value near log(p) in order to guarantee that $p >> k$. Using this k, B omega matrices are generated, each unique. For my algorithm, I made each entry of a given omega matrix a standard normal draw. The rest of the information needed to complete the problem must be inputted by the user. In order to generate a general algorithm for all cases of p,X,Y,B, I first started with a specific case (i.e. I chose p,X,Y,B). The first thing I did was use an optimization function from class.The first part of the below code is the gradient/tuning function. The second part is a general case, which is a function that returns the desired $\widehat{\beta}_n^{ave}$. The final part is my specific case, where I used $p = 100$, X follows a uniform distribution between 5 and 15, $Y = 10$, and $B = 1000$. For my specific case, the general optimization solution turns out to be equivalent to OLS.

This general case function utilizes the **readline()** function in R, which asks the user to input values for p,Y, and B. For this reason, do not simply ctrl+A to run the code. If you would like to scroll to the general case, it's the first instance of the line with multiple # in a row (to signify a break), the second being the general case. The **genav** (general) function requires an input of X and returns $\widehat{\beta}_n^{ave}$ (assuming the user inputted p,y, and B).

```r
# gradient function, takes beta_hat, and sigma matrix as input
compute_min <- function (beta_hat,sig, maxit=500, tol=1e-8){
  fn = function (b){
    return (as.numeric(
      t(b-beta_hat) %*% sig %*% (b-beta_hat)))
  }
  gr = function (b){
    return(as.numeric(2*sig%*%(b-beta_hat)))
  }
  beta_cur <- beta_hat
  direction_cur <- gr (beta_cur)

  # Line search objective
  ls_obj = function (alpha){
    return (fn(beta_cur - alpha*direction_cur))
  }

  alpha_opt <- optimize (ls_obj, interval=c(0,1))$minimum
  beta_next <- beta_cur - alpha_opt*gr(beta_cur)
  for (k in 1:maxit){
    direction_next <- gr (beta_next)
    dx = beta_next - beta_cur
    if (max(abs(dx)) < tol){
      break
```

```
    }
    dg <- direction_next - direction_cur
    alpha_opt <- sum(dg*dx)/sum(dg^2)
    beta_cur <- beta_next
    direction_cur <- direction_next
    beta_next <- beta_next - alpha_opt*direction_next
  }
  return (beta_next)



}

###################
# this is the general case
# this assumes x is defined correctly
# we assume a specific structure of omega
# however each omega will be different
p <- as.numeric(readline("enter value for p  "))
y <- as.numeric(readline("enter scalar for y  "))
B <- as.numeric(readline("enter large integer for B  "))
genav <- function(x){
  bomg <- 0
  for (i in 1:B){
    # we draw K in a uniform distribution centered at log(N)
    k <- runif(1,1,2*ceiling(log(p)))
    omg <- matrix(0,nrow = k,ncol = p)
    for (c in 1:p){
      for (d in 1:k){
        omg[d,c] <- rnorm(1,0,1)
      }
    }
    q <- omg%*%x
    bt <- t(solve(t(q)%*%q)%*%t(q)*y)
    sig <- q%*%t(q)
    bomg1 <- t(omg)%*%compute_min(bt,sig)
    bomg <- (bomg*(i-1))/i + bomg1/i
    rm(omg)
  }
  return(bomg)
}
far <- matrix(0,nrow=2,ncol = 2)
dim(far)
dim

###########

#example with a specific x, omega, y, B, p
# B from the problem
B <- 1000
# bomg is running average, see code from #2
bomg <- 0
# p is fixed far greater than what k could be
p <- 100
```

```
# y is a scalar
y <- 10
for (i in 1:B){
  # we draw K in a uniform distribution centered at log(N), per Dr. Laber
  k <- runif(1,1,10)
  x <- as.matrix(runif(p,5,15))
  omg <- matrix(0,nrow = k,ncol = p)
  for (c in 1:p){
    for (d in 1:k){
      omg[d,c] <- rnorm(1,0,1)
    }
  }
  q <- omg%*%x
  bt <- t(solve(t(q)%*%q)%*%t(q)*y)
  sig <- q%*%t(q)
  bomg1 <- t(omg)%*%compute_min(bt,sig)
  bomg <- (bomg*(i-1))/i + bomg1/i
  rm(omg)
}
# the next line returns the target
bomg
```