```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
cancer <- read.table("cancer.txt", header = TRUE, sep = "\t")
dev <- apply(cancer[, -1], 2, sd)
dev
```

```
## All.cancers        Lung       Colon    Melanoma    F.breast    Pancreas
##   47.857634   14.724588    7.262243    5.245611    7.888396    1.482258
##    Leukemia     Ovarian      Cervix    Prostate       Liver
##    2.255171    1.358915    1.421959   16.858436    2.241009
```

```
#Unscaled Data
data <- cancer[,-1]
data <- data %>% mutate_all(as.numeric)

c <- cov(data)
eig <- eigen(c)
evalue <- eig$values
evector <- eig$vectors
evector[,1:2]
```

```
##              [,1]         [,2]
##  [1,] 0.948911183  0.048606436
##  [2,] 0.173707714  0.668729611
##  [3,] 0.098867058  0.127018189
##  [4,] 0.029715889 -0.112583098
##  [5,] 0.077227953 -0.127033747
##  [6,] 0.015969632  0.005649558
##  [7,] 0.019995381 -0.039163522
##  [8,] 0.004948948 -0.004370774
##  [9,] 0.014052752  0.035246938
## [10,] 0.227792155 -0.708933677
## [11,] 0.002286687  0.008824253
```

```
#Coefficient indications the relation between PC and random variable
#Determine the first 5 Principle Components: Coeffecients (loadings) - reflect correlation between RV a
```

```
#Scaled Data

normal <- scale(data)
c <- cor(normal)
eig <- eigen(c)
evalue <- eig$values
evector <- eig$vectors

evector
```

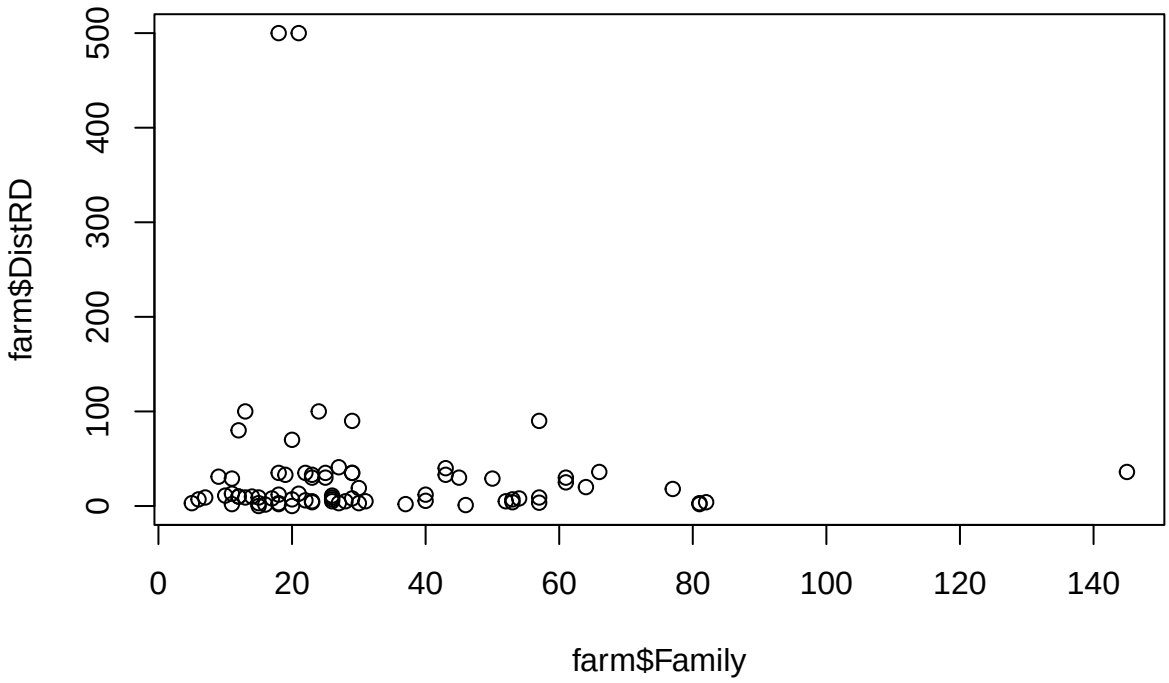
```
##              [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
##  [1,] -0.5010584 -0.03752827  0.13564906  0.10200653  0.00661866  0.10905402
##  [2,] -0.2867743  0.34463886  0.24975314  0.12961783  0.53167942  0.28217876
##  [3,] -0.3905015  0.24854728  0.20719239 -0.20988812 -0.03157362 -0.11291776
##  [4,] -0.1316843 -0.41857218 -0.05859752  0.60913595  0.15614063 -0.38162735
##  [5,] -0.2764510 -0.37218720 -0.27254464 -0.03686353  0.14210375  0.57513296
##  [6,] -0.3555025  0.02591798 -0.38107453  0.09964642  0.07993039 -0.45035470
##  [7,] -0.2406797 -0.28184171  0.30247121 -0.48821395 -0.28970563 -0.27410672
##  [8,] -0.1834695 -0.18665306 -0.48476086 -0.49457994  0.29556288 -0.06044586
##  [9,] -0.3183348  0.43674873 -0.03254542  0.07805234 -0.10765380 -0.18024636
## [10,] -0.3061206 -0.30469450  0.22752557  0.19475514 -0.44400656  0.24994785
## [11,] -0.0861796  0.32523824 -0.52350067  0.13936291 -0.53413119  0.20674709
##              [,7]         [,8]        [,9]       [,10]        [,11]
##  [1,]  0.04374209  0.006792296 -0.11185970 -0.18508041  0.811210603
##  [2,] -0.14537839  0.007252804 -0.29034347 -0.33824243 -0.370950840
##  [3,]  0.37193300  0.131962267  0.70550259 -0.07120336 -0.162640106
##  [4,] -0.30791519  0.098058275  0.31623819 -0.21670726 -0.107522730
##  [5,] -0.04175042  0.414550768  0.10494771  0.40179558 -0.111317074
##  [6,]  0.50502008  0.098916420 -0.43699136  0.16737802 -0.157430047
##  [7,] -0.33568094  0.364797251 -0.27142205 -0.15902722 -0.170336907
##  [8,] -0.19263901 -0.503971154  0.13826954 -0.22490690  0.009368924
##  [9,] -0.54255118 -0.168849994  0.04772674  0.57564539  0.012902563
## [10,]  0.15577866 -0.581436598 -0.09082973  0.03419142 -0.303946738
## [11,] -0.13637156  0.193887091  0.01248343 -0.44882159 -0.086554294
```

```
#Standardize all RV - Use correlation matrix (Cov of a standardized RV)

#Comparing the unscaled loadings to the standard deviations, we can observe that for most variables, th
```

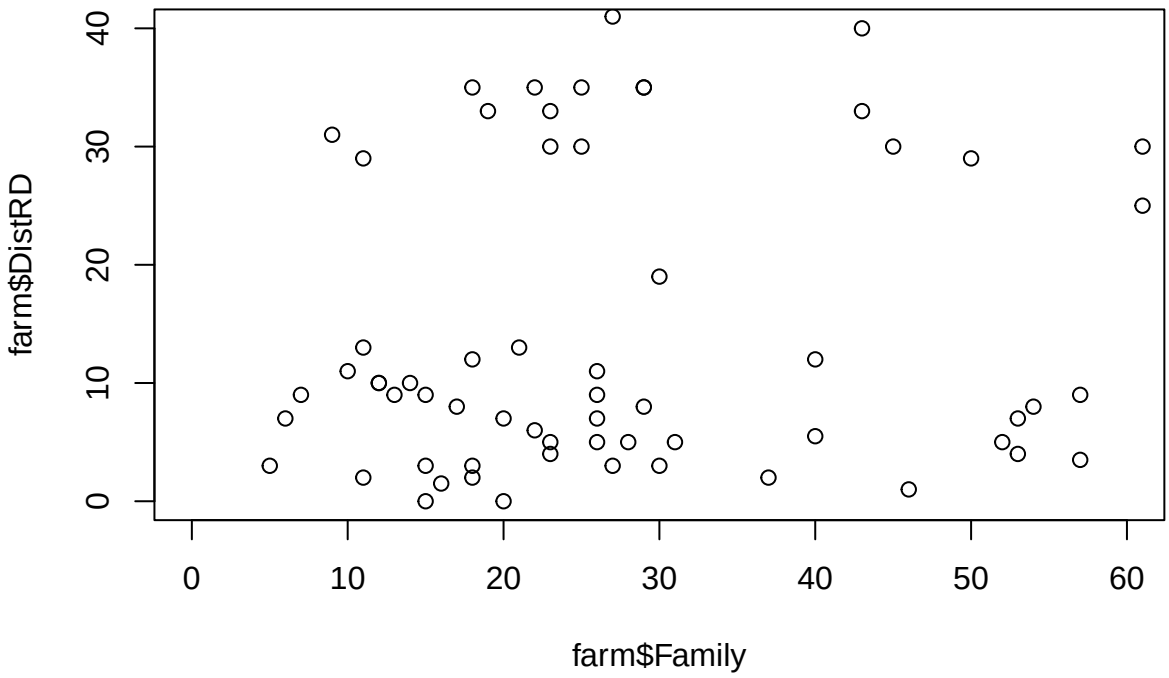
```
farm <- read.csv("farmers.csv")
head(farm)
```

```
##   Family DistRD Cotton Maize Sorg Millet Bull Cattle Goats
## 1     12     80    1.5   1.0    3   0.25    2      0     1
## 2     54      8    6.0   4.0    0   1.00    6     32     5
## 3     11     13    0.5   1.0    0   0.00    0      0     0
## 4     21     13    2.0   2.5    1   0.00    1      0     5
## 5     61     30    3.0   5.0    0   0.00    4     21     0
## 6     20     70    0.0   2.0    3   0.00    2      0     3
```

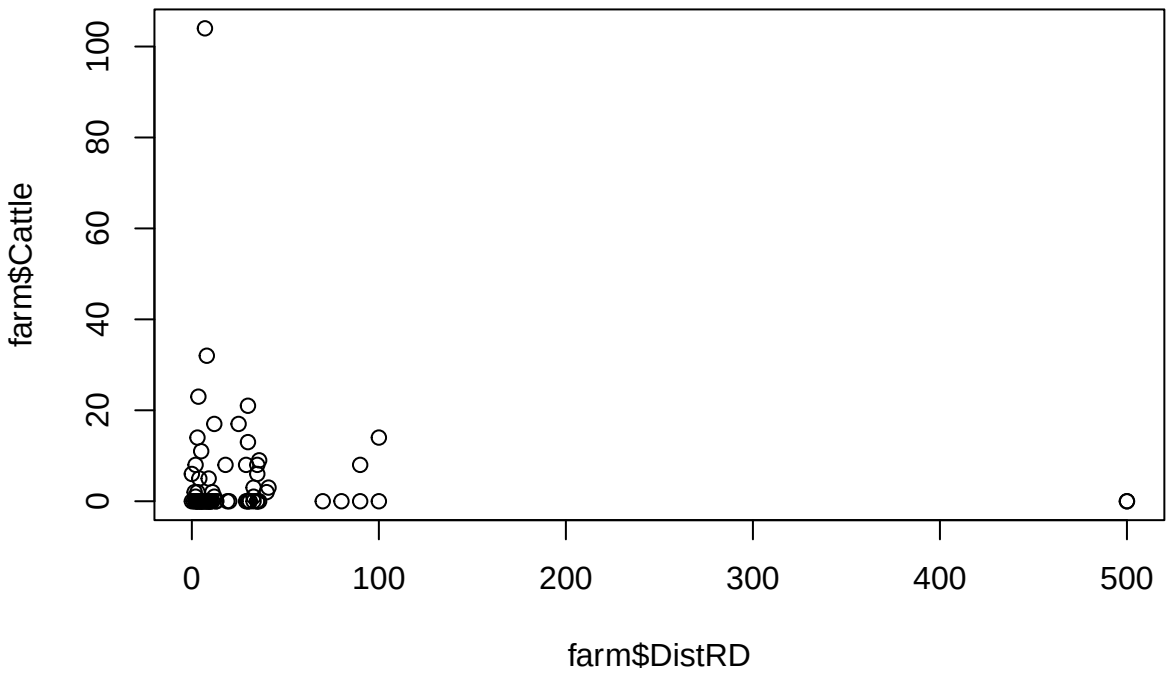
```
plot(farm$Family, farm$DistRD)
```



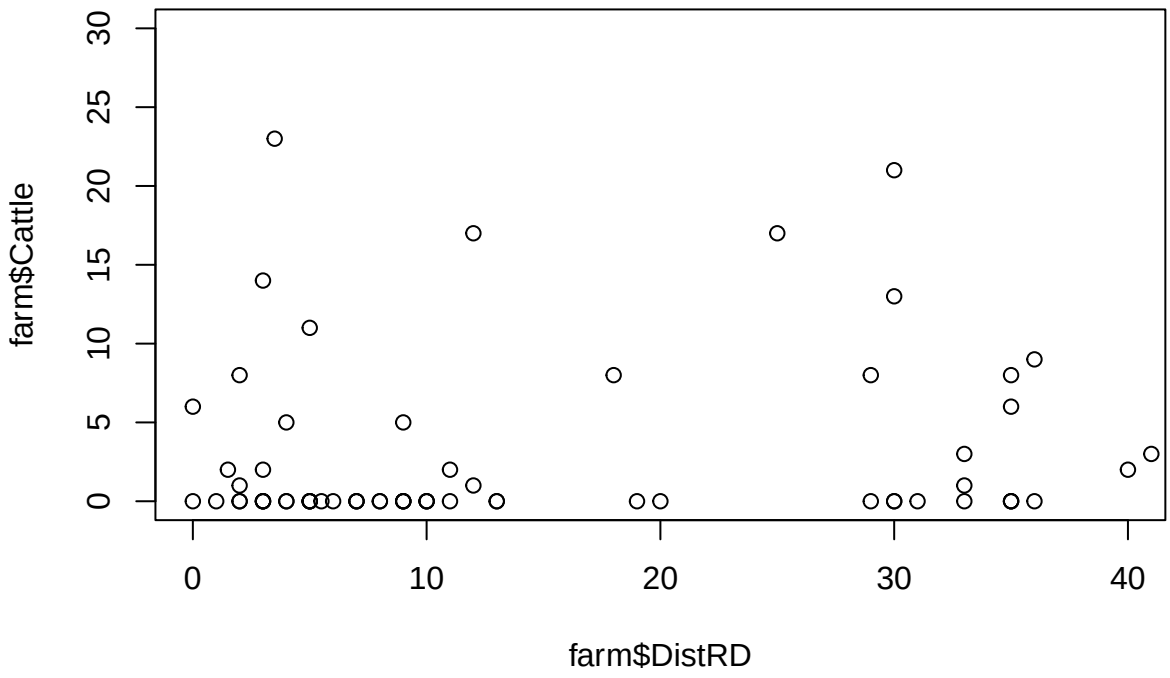

```
plot(farm$Family, farm$DistRD, ylim = c(0,40), xlim = c(0, 60))
```




```
plot(farm$DistRD, farm$Cattle)
```




```
plot(farm$DistRD, farm$Cattle, ylim = c(0,30), xlim = c(0, 40))
```




```
normal <- scale(farm)
c <- cor(normal)
eig <- eigen(c)
evalue <- eig$values
evector <- eig$vectors

evector[,1:5]
```

```
##              [,1]        [,2]         [,3]        [,4]        [,5]
##  [1,]  0.44428879 -0.09567009  0.008833192  0.12789030 -0.09000166
##  [2,] -0.03325733 -0.05415301  0.825143323 -0.51441340 -0.19338477
##  [3,]  0.41179605 -0.33581984  0.083498728 -0.01627394  0.09170542
##  [4,]  0.33247471 -0.56919455 -0.154619270 -0.15056819 -0.13415448
##  [5,]  0.31097871  0.46059658  0.059319486  0.21998950 -0.35482566
##  [6,]  0.26984647  0.07262355  0.392125115  0.59471442 -0.17451653
##  [7,]  0.44047172 -0.04395364 -0.126521044 -0.19728624  0.12572018
##  [8,]  0.24794805  0.43776487 -0.303032632 -0.49033335 -0.39511528
##  [9,]  0.31043123  0.37438468  0.151868551 -0.12107034  0.77447776
```

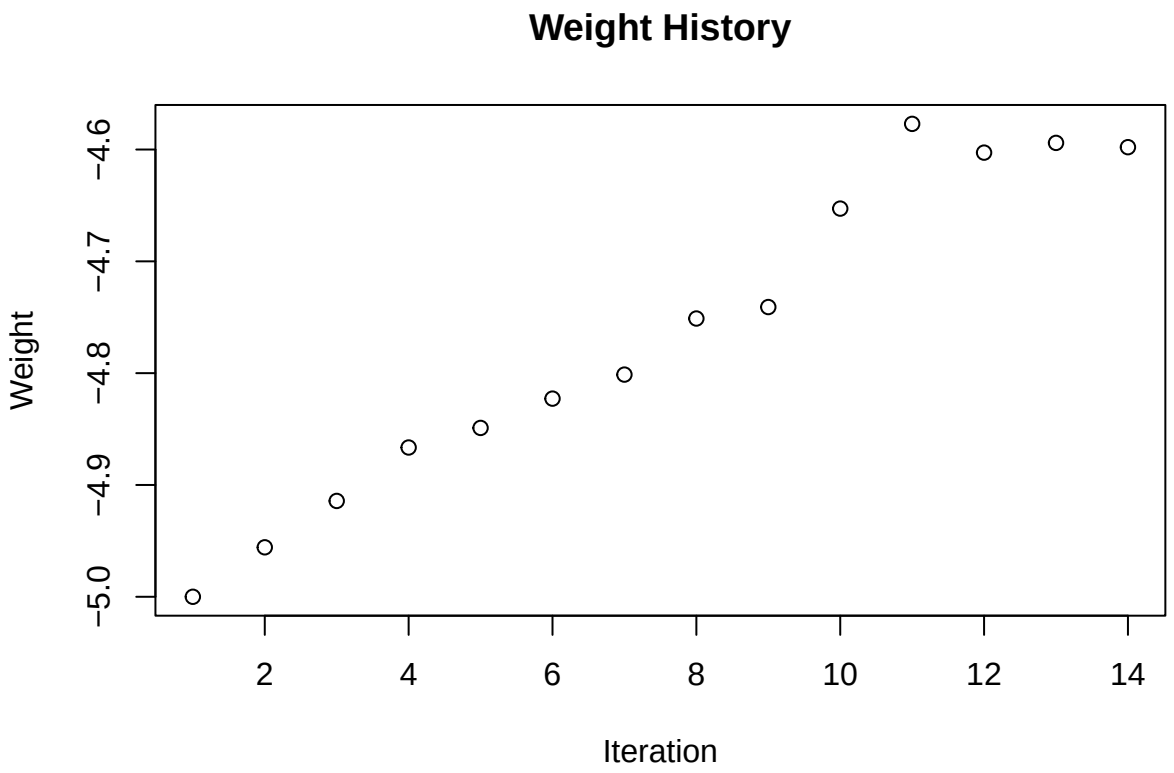
```
#The first principal component is a combination of all variables, as a majority are positive loadings.
```

```
g <- function(w) {
  return(sin(3 * w) + (1/3 * w)^2)
}
```

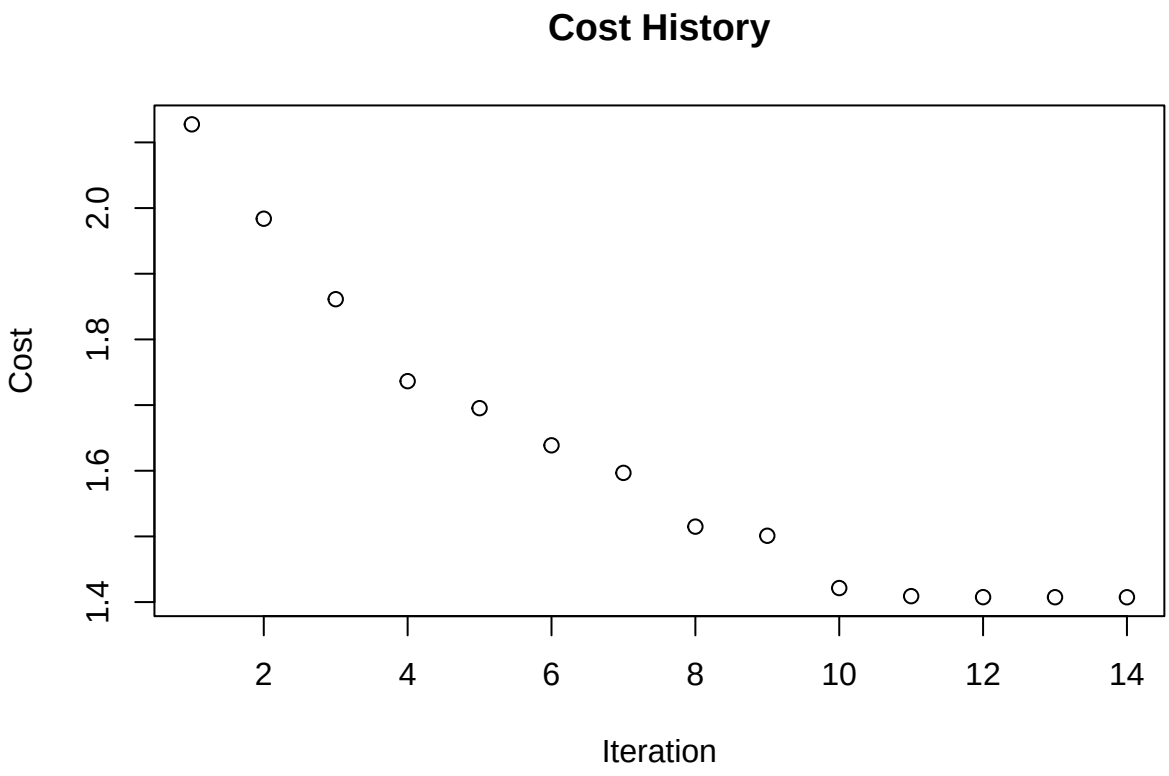
```r
random_search1 <- function(g, alpha = 1, max_its = 100, n = 2, w0 = 0) {
  weight_history <- vector()
  cost_history <- vector()

  weight <- w0
  cost <- g(weight)

  weight_history <- c(weight_history, weight)
  cost_history <- c(cost_history, cost)

  if (alpha == "diminishing") {
    alpha <- 1
  } else {
    alpha <- alpha
  }


  for (i in 1:max_its) {
    for (j in 1:n) {
      direction <- runif(1, -1, 1)
      a <- weight + (alpha * direction)
      a <- g(a)
      if (cost > a){

      weight <- weight + (alpha * direction)
      cost <- g(weight)

      weight_history <- c(weight_history, weight)
      cost_history <- c(cost_history, cost)
      }
    }

   if (alpha == "diminishing") {
      alpha <- alpha / sqrt(i)
    }
  }


  list(weight_history = weight_history, cost_history = cost_history)
}

# Example usage
results <- random_search1(g, alpha = 0.1, max_its = 100, n = 2, w0 = -5)
results
```

```
## $weight_history
##  [1] -5.000000 -4.955771 -4.914289 -4.866498 -4.848990 -4.822775 -4.801304
##  [8] -4.751146 -4.740885 -4.652842 -4.577068 -4.602777 -4.594075 -4.597897
##
## $cost_history
##  [1] 2.127490 1.983776 1.861254 1.736406 1.695324 1.638684 1.596757 1.514906
##  [9] 1.500984 1.421351 1.409005 1.407532 1.407392 1.407372
```

```r
#random_search1(g, alpha = 0.1, max_its = 100, n = 2, w0 = 2.5)

plot(results$weight_history , xlab = "Iteration", ylab = "Weight", main = "Weight History")
```




```r
# Plot cost_history
plot(results$cost_history, xlab = "Iteration", ylab = "Cost", main = "Cost History")
```




```r
random_search <- function(g, a = "diminishing", max_its = 100, n = 1000, w0 = c(0, 0)) {
  weight_history <- list(w0)
  cost_history <- c(g(w0[1], w0[2]))

  if (a == "diminishing") {
    ap1 <- 1.0
  } else {
    ap1 <- a
  }

  for (i in 1:max_its) {
    directions <- matrix(rnorm(n * 2), ncol = 2)

    for (j in 1:n) {
      w <- weight_history[[length(weight_history)]]
      w_new <- w + ap1 * directions[j, ]
      cost <- g(w_new[1], w_new[2])

      if (cost < cost_history[length(cost_history)]) {
        weight_history <- c(weight_history, list(w_new))
        cost_history <- c(cost_history, cost)
      }
    }

    if (a == "diminishing") {
      ap1 <- ap1 / 2.0
```

```
    }
  }

  return(list(weight_history = do.call(rbind, weight_history), cost_history = cost_history))
}

g <- function(w1, w2) {
  return(100 * (w2 - w1^2)^2 + (1 - w1)^2)
}

result <- random_search(g, a = 1, max_its = 100, w0 = c(-3, -3))

# Plot cost history
plot(result$cost_history, xlab = "Iteration", ylab = "Cost", main = "Cost History")
```