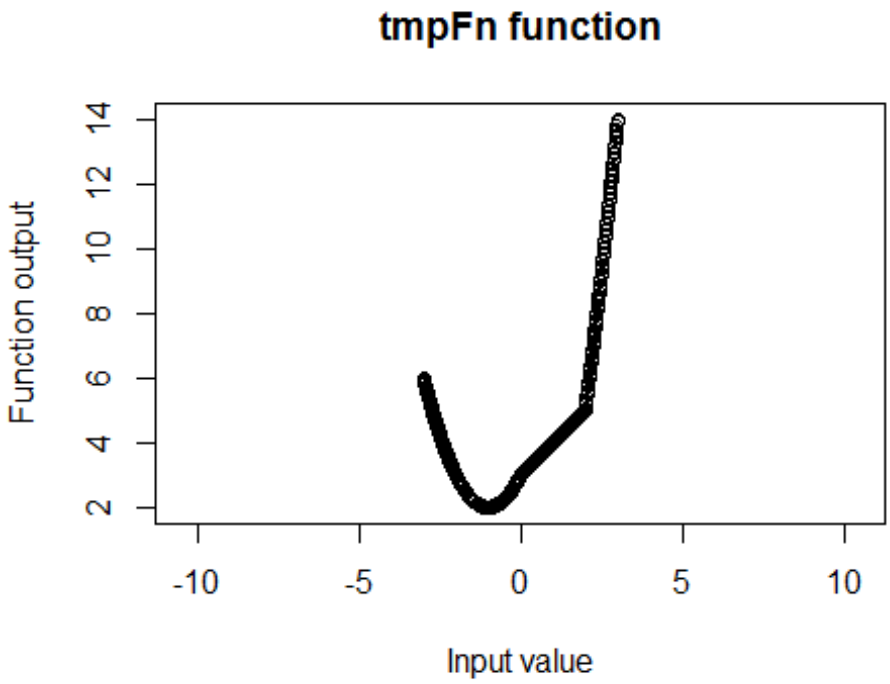
Hmwk9.R

Audrey McCombs

Fri Nov 10 22:02:27 2017

```
#Problem 1

matfunc <- function(n,k) {
  diag(x = k, nrow = n, ncol = n)
}
matfunc(6,5)

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    0    0    0    0    0
## [2,]    0    5    0    0    0    0
## [3,]    0    0    5    0    0    0
## [4,]    0    0    0    5    0    0
## [5,]    0    0    0    0    5    0
## [6,]    0    0    0    0    0    5

rm(list = ls())

#Problem 2

tmpFn <- function(xVec) {
  ifelse(test = xVec < 0,
      yes = ((xVec^2) + (2*xVec) + 3),
      no = ifelse(test = 0 <= xVec & xVec < 2,
          yes = (xVec + 3),
          no = ((xVec^2) + (4* xVec) - 7)))
  }

xVec <- seq(from = -3, to = 3, by = .01)
yVec <- tmpFn(xVec)
plot(xVec, yVec, xlab = "Input value", ylab = "Function output", main =
"tmpFn function", asp = 1)
```




```
rm(list = ls())

#Problem 3

gdc <- function(m,n) {
  firstm <- m
  firstn <- n
  r <- 1
  while (r != 0) {
    r <- m %% n
    m <- n
    n <- r
  }
print(c(firstm, firstn, m))
}

gdc(1420,95)

## [1] 1420   95    5

rm(list = ls())

#Problem 4

order.matrix <- function(mymat) {
ordvec <- sort(mymat)
indrow <- (rep(NA, length(ordvec)))
```

```
indcol <- (rep(NA, length(ordvec)))
for (i in 1:length(ordvec)) {
  rowcol <- which(mymat == ordvec[i], arr.ind = TRUE)
  indrow[i] <- rowcol[1,1]
  indcol[i] <- rowcol[1,2]
}
values <- data.frame(number = ordvec, rowindex = indrow, colindex = indcol)
return(values)
}

mymat <- matrix(rchisq(12, 1), nrow = 4)
order.matrix(mymat)
```

```
##       number rowindex colindex
## 1  0.1665623        3        2
## 2  0.2657899        3        1
## 3  0.4045661        1        1
## 4  0.4430517        4        1
## 5  0.5744085        1        2
## 6  0.7288544        2        3
## 7  0.7877767        4        2
## 8  0.9221183        2        1
## 9  0.9850161        3        3
## 10 1.5887811        2        2
## 11 3.1262032        1        3
## 12 4.5532919        4        3
```

```
mymat <- matrix(rchisq(20, 1), nrow = 5)
order.matrix(mymat)
```

```
##          number rowindex colindex
## 1  0.0001931904        1        4
## 2  0.0044179904        5        1
## 3  0.0139876989        1        2
## 4  0.0239771482        1        3
## 5  0.0349673022        2        4
## 6  0.0429590343        4        2
## 7  0.0484625034        4        4
## 8  0.0551572657        4        1
## 9  0.0600482125        2        1
## 10 0.0677640266        4        3
## 11 0.0792188237        5        3
## 12 0.1001339343        3        1
## 13 0.1037628103        1        1
## 14 0.1093666476        3        4
## 15 0.1869270890        2        2
## 16 0.2544719847        3        2
## 17 0.4785152108        2        3
## 18 0.7036197419        5        2
```

```
## 19 0.8737965178        3        3
## 20 0.9176613976        5        4
```

```
rm(list = ls())
```

```
#Problem 5
op <- par()

#Problem 5.a
polaroid <- function(x) {
  p <- length(x)
  r <- sqrt(sum(x^2))

  theta <- rep(0, p-1)
  den <- rep(0, p-2)

  theta[1] <- acos(x[1]/r)
  den[1] <- r

  for (i in 2:(p-1)) {
    den[i] <- den[i-1] * sin(theta[i-1])
    theta[i] <- acos(x[i]/den[i])
  }

  polar <- c(r, theta)
  return(polar)
}

x <- seq(from = 0, to = 10, by = 2)
polaroid(x)
```

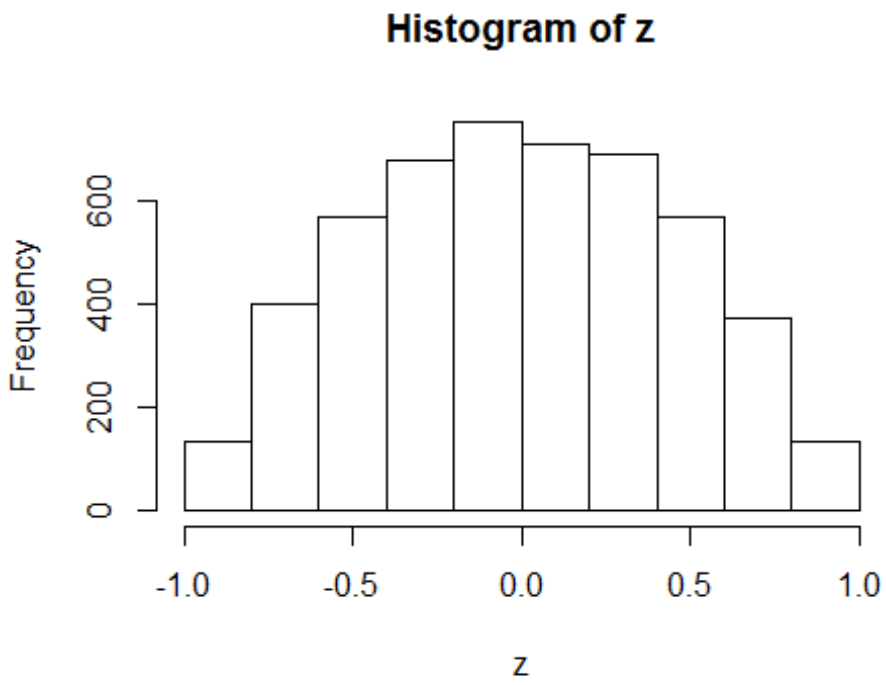
```
## [1] 14.8323970  1.5707963  1.4355444  1.2951535  1.1326473  0.8960554
```

```
#Problem 5.b
normalize <- function(vec) {
  den <- sqrt(sum(vec^2))
  output <- vec/den
}

#Problem 5.c
y <- matrix(rnorm(5000, mean = 0, sd = 1), nrow = 1000, ncol = 5)
zt <- apply(y, 1, normalize)
z <- t(zt)

ks.test(z,"punif",min=-1,max=1)
```

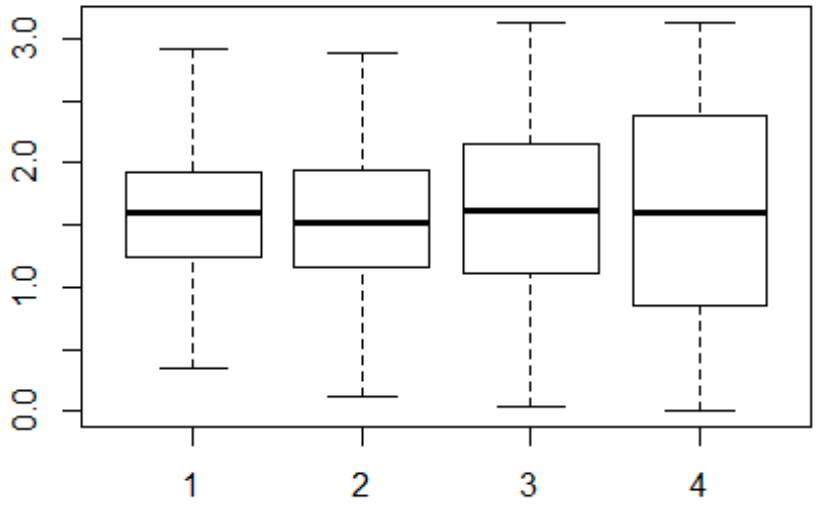
```
##
##  One-sample Kolmogorov-Smirnov test
```

```
##
## data:  z
## D = 0.099436, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
hist(z)
```



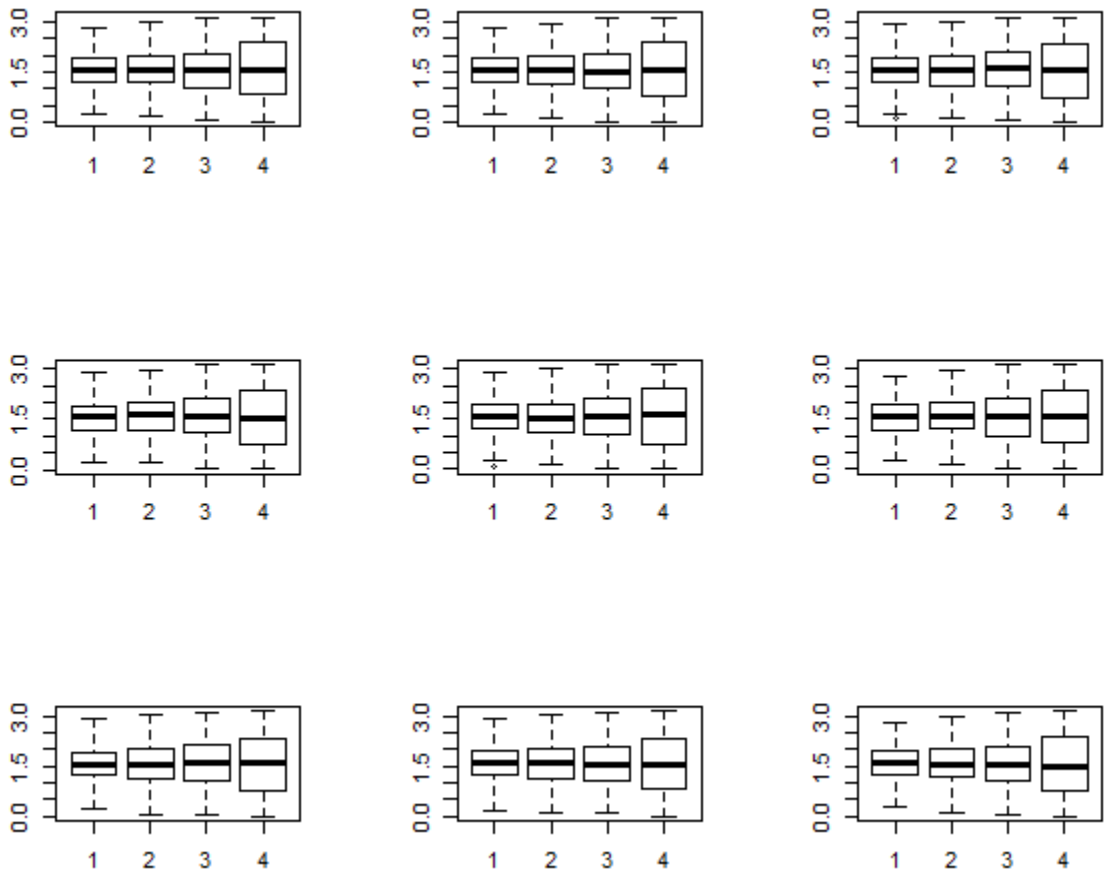

```
#Based on the Kolmogorov-Smirnov test, the z matrix is not uniformly
distributed.  We can see this clearly in the histogram of values in the
matrix.

#Problem 5.d
polarst <- apply(y, 1, polaroid)
polars <- t(polarst)

ks.test(polars[,1]^2, "pchisq", 5)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  polars[, 1]^2
## D = 0.026994, p-value = 0.4598
## alternative hypothesis: two-sided

boxplot(polars[,2:5])
```




```
ks.test(polars[,2], "punif", min = 0, max = 2*pi)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  polars[, 2]
## D = 0.57589, p-value < 2.2e-16
## alternative hypothesis: two-sided

ks.test(polars[,3], "punif", min = 0, max = pi)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  polars[, 3]
## D = 0.17876, p-value < 2.2e-16
## alternative hypothesis: two-sided

ks.test(polars[,4], "punif", min = 0, max = pi)

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  polars[, 4]
## D = 0.11135, p-value = 3.403e-11
## alternative hypothesis: two-sided

ks.test(polars[,5], "punif", min = 0, max = pi)
```

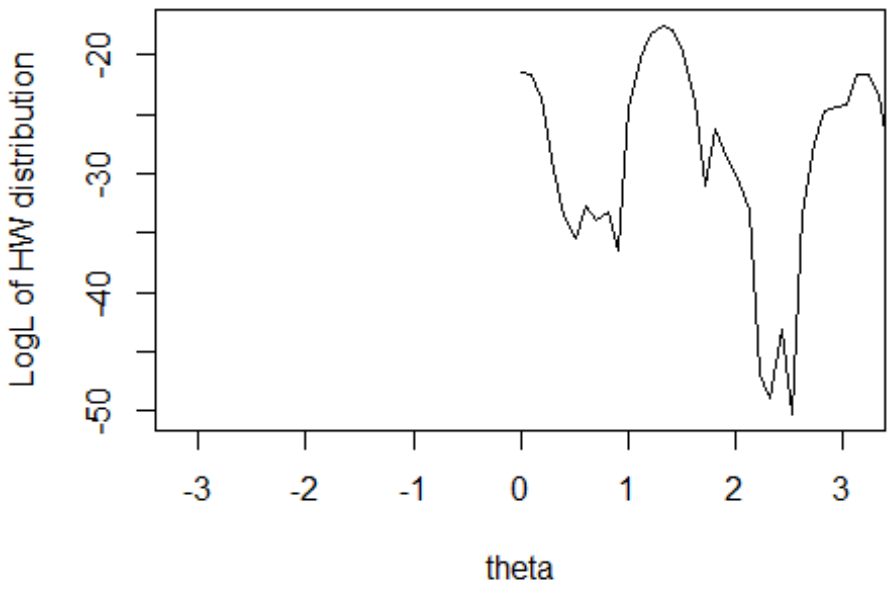
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  polars[, 5]
## D = 0.024207, p-value = 0.6012
## alternative hypothesis: two-sided
```

```r
#Multiple distributions
par(mfrow = c(3,3))
for (i in 1:9) {
  y <- matrix(rnorm(5000, mean = 0, sd = 1), nrow = 1000, ncol = 5)
  zt <- apply(y, 1, normalize)
  z <- t(zt)
  polarst <- apply(y, 1, polaroid)
  polars <- t(polarst)
  boxplot(polars[,2:5])
}
```




```r
par(op)
```

```r
#Problem 6

  #Problem 6.a
x <-c(3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96, 2.53, 3.88,
2.22, 3.47, 4.82, 2.46, 2.99, 2.54, 0.52, 2.50)
theta <- seq(0,10,,100)

hm <- function(theta = theta, x = 3.91) (1-cos(x - theta))/(2*pi)
loghm <- function(theta, x) (-log(2*pi)+sum(log(1-cos(x-theta)^2)))
plot(theta,sapply(theta,loghm,x),type="l",ylab="LogL of HW distribution",
xlim = c(-3.1415927, 3.1415927))
```




```r
plot(theta,sapply(theta,loghm,x),type="l",ylab="LogL of HW distribution",
xlim = c(0, 10))

  #Problem 6.b
optimize(function(theta) sapply(theta, loghm, x), interval = c(1,2), maximum
= T)

## $maximum
## [1] 1.326029
##
## $objective
## [1] -17.51277

  #Problem 6.c
ghm <- function(theta,x) sin(x-theta)/(1-cos(x-theta))
```

```r
derghm <- function(theta,x) 1/(1-cos(x-theta))

loglik <- function(fun, derf, x0, eps, nlim,...) {
  iter <- 0
  repeat {
    iter <- iter + 1
    if(iter > nlim) {
      cat(" Iteration Limit Exceeded: Current = ",iter, fill = T)
      x1 <- NA
      break
    }
    x1 <- x0 - fun(x0,...)/derf(x0,...)
    if(abs(x0 - x1) < eps||abs(fun(x1,...))<1.0e-12)
      break
    x0 <- x1
    cat("\n****** Iter. No: ", iter, " Current Iterate = ", x1, fill=T)
  }
  return(x1)
}

loglik(ghm, derghm, 0, 0.00001, 100, x)
```

```
##
## ****** Iter. No:  1  Current Iterate =  0.694991 0.9905465 -0.7588807
## 0.7946357 0.5298361 0.7823359 0.7333152 0.3787149 -0.7588807 -0.9252115
## -0.5741721 0.6731109 -0.7965655 0.3225359 0.9942155 -0.6300306 -0.1510127
## -0.5659562 -0.4968801 -0.5984721
##
## ****** Iter. No:  2  Current Iterate =  0.7683414 1.648321 -0.8614122
## 0.9180916 0.5584035 0.8981469 1.733213 0.3884072 -0.8614122 -1.178793
## -0.6115839 0.738361 -0.9212672 0.3284073 1.626262 -0.6815698 -0.1515927
## -0.6015851 -1.347351 -0.6415793

##  [1]  0.7684073  1.7083712 -0.8615927  0.9184073  0.5584073  0.8984073
##  [7]  2.2855744  0.3884073 -0.8615927 -1.1815926 -0.6115927  0.7384073
## [13] -0.9215927  0.3284073  1.6783837 -0.6815927 -0.1515927 -0.6015927
## [19] -2.3037001 -0.6415927
```

```r
#This found the MLE value at -0.64: the local maximum near x = 0.

  #Problem 6.d
loglik(ghm, derghm, -2.0, 0.00001, 100, x)
```

```
##
## ****** Iter. No:  1  Current Iterate =  -1.635417 -2.536948 -1.092033
## -1.778663 -1.449314 -1.759205 -2.923388 -1.316034 -1.092033 -1.269942
## -1.016587 -1.60765 -1.118794 -1.27352 -2.511401 -1.031681 -1.038287
## -1.014822 -2.582331 -1.02247
##
## ****** Iter. No:  2  Current Iterate =  -0.9627782 -3.429855 -0.8636267
```

```
## -1.348636 -0.5432575 -1.293898 -3.786495 -0.3249513 -0.8636267 -1.181708
## -0.6225734 -0.8934112 -0.9228683 -0.2740048 -3.377936 -0.6887002
## -0.2633001 -0.6132531 -2.621583 -0.6507349
##
## ****** Iter. No:  3  Current Iterate =  0.02438699 -4.340533 -0.8615927
## -0.5813819 0.3487038 -0.4808975 -3.963835 0.3294259 -0.8615927 -1.181593
## -0.6115929 0.1047275 -0.9215927 0.2926269 -4.319364 -0.6815927 -0.1518248
## -0.6015929 -2.621593 -0.6415928
##
## ****** Iter. No:  4  Current Iterate =  0.7016384 -4.572642 -0.8615927
## 0.4160982 0.5568737 0.5008239 -3.964778 0.3883732 -0.8615927 -1.181593
## -0.6115927 0.6968417 -0.9215927 0.3283997 -4.600919 -0.6815927 -0.1515927
## -0.6015927 -2.621593 -0.6415927
##
## ****** Iter. No:  5  Current Iterate =  0.7683577 -4.574778 -0.8615927
## 0.8975489 0.5584073 0.8880153 -3.964778 0.3884073 -0.8615927 -1.181593
## -0.6115927 0.7383954 -0.9215927 0.3284073 -4.604778 -0.6815927 -0.1515927
## -0.6015927 -2.621593 -0.6415927

##  [1]  0.7684073 -4.5747780 -0.8615927  0.9184058  0.5584073  0.8984072
##  [7] -3.9647780  0.3884073 -0.8615927 -1.1815927 -0.6115927  0.7384073
## [13] -0.9215927  0.3284073 -4.6047780 -0.6815927 -0.1515927 -0.6015927
## [19] -2.6215927 -0.6415927

loglik(ghm, derghm, -2.7, 0.00001, 100, x)

##
## ****** Iter. No:  1  Current Iterate =  -3.021028 -3.654152 -1.735595
## -3.158951 -2.816549 -3.141092 -3.653541 -2.64684 -1.735595 -1.701372
## -1.830996 -2.992476 -1.721474 -2.587056 -3.644745 -1.798516 -2.140995
## -1.835988 -2.621673 -1.816545
##
## ****** Iter. No:  2  Current Iterate =  -3.624496 -4.450133 -0.9686912
## -3.964005 -3.047801 -3.923115 -3.959777 -2.540695 -0.9686912 -1.204683
## -0.892102 -3.548247 -1.004201 -2.362849 -4.463955 -0.8997605 -1.227339
## -0.892039 -2.621593 -0.8938737
##
## ****** Iter. No:  3  Current Iterate =  -4.573892 -4.574455 -0.8617973
## -4.949586 -3.49588 -4.917166 -3.964778 -2.329799 -0.8617973 -1.181595
## -0.6152569 -4.458982 -0.9216866 -1.927581 -4.604313 -0.6833192 -0.3473938
## -0.6056591 -2.621593 -0.6442603
##
## ****** Iter. No:  4  Current Iterate =  -5.381973 -4.574778 -0.8615927
## -5.352952 -4.287034 -5.367922 -3.964778 -1.918949 -0.8615927 -1.181593
## -0.6115927 -5.343657 -0.9215927 -1.153284 -4.604778 -0.6815927 -0.1528414
## -0.6015927 -2.621593 -0.6415927
##
## ****** Iter. No:  5  Current Iterate =  -5.514388 -4.574778 -0.8615927
## -5.364778 -5.278196 -5.384777 -3.964778 -1.178166 -0.8615927 -1.181593
## -0.6115927 -5.543425 -0.9215927 -0.1572512 -4.604778 -0.6815927 -0.1515927
```

```
## -0.6015927 -2.621593 -0.6415927
##
## ****** Iter. No:  6  Current Iterate =  -5.514778 -4.574778 -0.8615927
## -5.364778 -5.710081 -5.384778 -3.964778 -0.1781749 -0.8615927 -1.181593
## -0.6115927 -5.544778 -0.9215927 0.3095397 -4.604778 -0.6815927 -0.1515927
## -0.6015927 -2.621593 -0.6415927

##  [1] -5.5147780 -4.5747780 -0.8615927 -5.3647780 -5.7247774 -5.3847780
##  [7] -3.9647780  0.3585766 -0.8615927 -1.1815927 -0.6115927 -5.5447780
## [13] -0.9215927  0.3284062 -4.6047780 -0.6815927 -0.1515927 -0.6015927
## [19] -2.6215927 -0.6415927

# At a starting value of -2.0, the first iteration found an MLE of -1.02,
which is a local maximum near -2.0.  Later iterations found the same MLE as
with starting point of 0 (i.e., -0.64).  At a starting value of 2.7, however,
the first iteration found the MLE at -1.8165, and it took more iterations to
find the MLE of -0.64.  This function bounces around a lot, so it's not
surprising that the function finds local maxima and takes a while to settle.

#Problem 7 - Go Galton!

  #Problem 7.a
men <- rnorm(n = 100, mean = 125, sd =  25)
women <- rnorm(n = 100, mean = 125, sd = 15)
t0 <- data.frame(M = men, W = women)
head(t0)

##           M        W
## 1  94.18540 142.2488
## 2  98.84936 112.6996
## 3 103.38450 121.5094
## 4 185.52787 119.4877
## 5 145.68981 119.1958
## 6  93.34521 139.0001

  #Problem 7.b
permute <- function(t0, iter) {
  t <- as.list(rep(NA, iter))
  output <- as.list(rep(NA, iter))
  ttemp <- t0
  for (i in 1:iter) {
    t[[i]] <- data.frame(M = sample(x = ttemp$M, 100), W = ttemp$W)
    output[[i]] <- apply(t[[i]], 1, mean)
    ttemp <- data.frame(M = output[[i]], W = output[[i]])
  }
  return(output)
}

  #Problem 7.c
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 3.2.5
```

```
heights <- permute(t0, 9)
heights <- as.data.frame(heights)
names(heights) <- paste("G", 1:9, sep = "")
head(heights)
```

```
##         G1       G2       G3       G4       G5       G6       G7       G8
## 1 135.8820 136.1448 137.6520 129.1269 125.2955 125.6155 124.9099 124.7827
## 2 116.8171 115.3775 112.4821 118.9764 118.8376 123.0381 124.2781 124.9299
## 3 118.8127 127.7004 129.5418 126.7541 127.3588 124.2797 124.1995 124.2551
## 4 112.0907 122.3789 119.9758 120.4831 128.1408 128.5821 127.7421 127.9181
## 5 123.5298 125.8862 121.2932 123.3369 122.2617 121.9554 123.9541 124.9214
## 6 121.1923 121.1923 120.8602 126.5974 125.4531 126.4668 126.6820 126.0116
##         G9
## 1 126.1103
## 2 125.1944
## 3 125.1173
## 4 125.3302
## 5 126.4198
## 6 125.7347
```

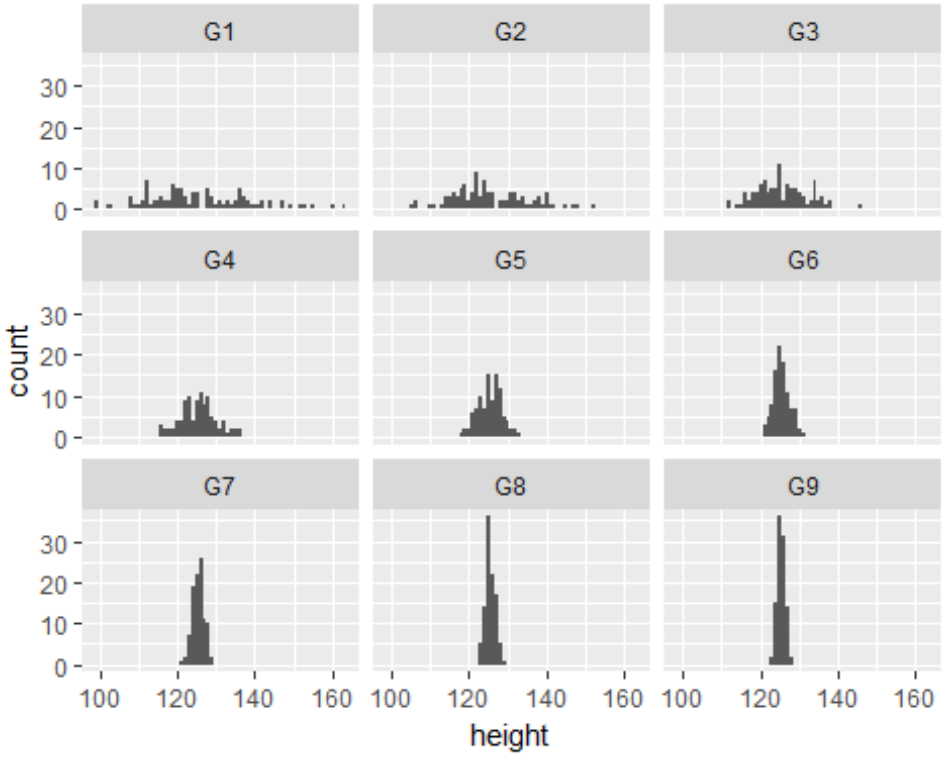
```
heights <- melt(heights)
```

```
## No id variables; using all as measure variables
```

```
names(heights) <- c("gen", "height")
head(heights)
```

```
##   gen   height
## 1  G1 135.8820
## 2  G1 116.8171
## 3  G1 118.8127
## 4  G1 112.0907
## 5  G1 123.5298
## 6  G1 121.1923
```

```
ggplot(heights, aes(x=height)) + geom_histogram(binwidth = 1) + facet_wrap(~
gen)
```




```
#Problem 8

  #Problem 8.a

prettyvec <- function(filename) {
  iris <- readLines(con = filename)
  iris <- iris[-c(1:2, length(iris))]  #remove first 2 lines and blank last
line
  indices <- grep(pattern = "\\s", iris)

  iris <- iris[-c(indices-1)]   #remove blank lines
  indices <- grep(pattern = "\\s", iris)

  final <- as.list(rep(NA, length(indices)))
  for (i in 1:length(indices)) {
    grpsize <- strsplit(x = iris[indices[i]], split = "=")
    final[[i]] <- rep(i-1, grpsize[[1]][2])
  }

  finalvec <- unlist(final)
  iris <- iris[-c(indices)]

  finaldf <- as.data.frame(cbind(finalvec, iris))
  names(finaldf) <- c("group", "observation")
```

```
  return(finaldf)
}

  #Problem 8.b
prettyvec("Iris1.out")
```

```
##     group observation
## 1       0         100
## 2       0         102
## 3       0         103
## 4       0         104
## 5       0         105
## 6       0         107
## 7       0         108
## 8       0         109
## 9       0         110
## 10      0         111
## 11      0         112
## 12      0         114
## 13      0         115
## 14      0         116
## 15      0         117
## 16      0         118
## 17      0         120
## 18      0         122
## 19      0         124
## 20      0         125
## 21      0         128
## 22      0         129
## 23      0         130
## 24      0         131
## 25      0         132
## 26      0         134
## 27      0         135
## 28      0         136
## 29      0         137
## 30      0         139
## 31      0         140
## 32      0         141
## 33      0         143
## 34      0         144
## 35      0         145
## 36      0         147
## 37      0         148
## 38      1           0
## 39      1           1
## 40      1           2
## 41      1           3
## 42      1           4
## 43      1           5
```

```
## 44      1           6
## 45      1           7
## 46      1           8
## 47      1           9
## 48      1          10
## 49      1          11
## 50      1          12
## 51      1          13
## 52      1          14
## 53      1          15
## 54      1          16
## 55      1          17
## 56      1          18
## 57      1          19
## 58      1          20
## 59      1          21
## 60      1          22
## 61      1          23
## 62      1          24
## 63      1          25
## 64      1          26
## 65      1          27
## 66      1          28
## 67      1          29
## 68      1          30
## 69      1          31
## 70      1          32
## 71      1          33
## 72      1          34
## 73      1          35
## 74      1          36
## 75      1          37
## 76      1          38
## 77      1          39
## 78      1          40
## 79      1          41
## 80      1          42
## 81      1          43
## 82      1          44
## 83      1          45
## 84      1          46
## 85      1          47
## 86      1          48
## 87      1          49
## 88      2          50
## 89      2          51
## 90      2          52
## 91      2          53
## 92      2          54
## 93      2          55
```

```
## 94      2          56
## 95      2          57
## 96      2          58
## 97      2          59
## 98      2          60
## 99      2          61
## 100     2          62
## 101     2          63
## 102     2          64
## 103     2          65
## 104     2          66
## 105     2          67
## 106     2          68
## 107     2          69
## 108     2          70
## 109     2          71
## 110     2          72
## 111     2          73
## 112     2          74
## 113     2          75
## 114     2          76
## 115     2          77
## 116     2          78
## 117     2          79
## 118     2          80
## 119     2          81
## 120     2          82
## 121     2          83
## 122     2          84
## 123     2          85
## 124     2          86
## 125     2          87
## 126     2          88
## 127     2          89
## 128     2          90
## 129     2          91
## 130     2          92
## 131     2          93
## 132     2          94
## 133     2          95
## 134     2          96
## 135     2          97
## 136     2          98
## 137     2          99
## 138     2         101
## 139     2         106
## 140     2         113
## 141     2         119
## 142     2         121
## 143     2         123
```

```
## 144     2         126
## 145     2         127
## 146     2         133
## 147     2         138
## 148     2         142
## 149     2         146
## 150     2         149
```

```
prettyvec("Iris2.out")
```

```
##     group observation
## 1       0          50
## 2       0          51
## 3       0          52
## 4       0          53
## 5       0          54
## 6       0          55
## 7       0          56
## 8       0          57
## 9       0          58
## 10      0          59
## 11      0          60
## 12      0          61
## 13      0          62
## 14      0          63
## 15      0          64
## 16      0          65
## 17      0          66
## 18      0          67
## 19      0          68
## 20      0          69
## 21      0          70
## 22      0          71
## 23      0          72
## 24      0          73
## 25      0          74
## 26      0          75
## 27      0          76
## 28      0          77
## 29      0          78
## 30      0          79
## 31      0          80
## 32      0          81
## 33      0          82
## 34      0          83
## 35      0          84
## 36      0          85
## 37      0          86
## 38      0          87
## 39      0          88
```

```
## 40      0          89
## 41      0          90
## 42      0          91
## 43      0          92
## 44      0          93
## 45      0          94
## 46      0          95
## 47      0          96
## 48      0          97
## 49      0          98
## 50      0          99
## 51      0         100
## 52      0         101
## 53      0         102
## 54      0         103
## 55      0         104
## 56      0         105
## 57      0         106
## 58      0         107
## 59      0         108
## 60      0         109
## 61      0         110
## 62      0         111
## 63      0         112
## 64      0         113
## 65      0         114
## 66      0         115
## 67      0         116
## 68      0         117
## 69      0         118
## 70      0         119
## 71      0         120
## 72      0         121
## 73      0         122
## 74      0         123
## 75      0         124
## 76      0         125
## 77      0         126
## 78      0         127
## 79      0         128
## 80      0         129
## 81      0         130
## 82      0         131
## 83      0         132
## 84      0         133
## 85      0         134
## 86      0         135
## 87      0         136
## 88      0         137
## 89      0         138
```

```
## 90      0         139
## 91      0         140
## 92      0         141
## 93      0         142
## 94      0         143
## 95      0         144
## 96      0         145
## 97      0         146
## 98      0         147
## 99      0         148
## 100     0         149
## 101     1           0
## 102     1           1
## 103     1           2
## 104     1           3
## 105     1           4
## 106     1           5
## 107     1           6
## 108     1           7
## 109     1           8
## 110     1           9
## 111     1          10
## 112     1          11
## 113     1          12
## 114     1          13
## 115     1          14
## 116     1          15
## 117     1          16
## 118     1          17
## 119     1          18
## 120     1          19
## 121     1          20
## 122     1          21
## 123     1          22
## 124     1          23
## 125     1          24
## 126     1          25
## 127     1          26
## 128     1          27
## 129     1          28
## 130     1          29
## 131     1          30
## 132     1          31
## 133     1          32
## 134     1          33
## 135     1          34
## 136     1          35
## 137     1          36
## 138     1          37
## 139     1          38
```

```
## 140     1          39
## 141     1          40
## 142     1          41
## 143     1          42
## 144     1          43
## 145     1          44
## 146     1          45
## 147     1          46
## 148     1          47
## 149     1          48
## 150     1          49
```