Stat 341 Assignment 1

2022-09-26

Question 1: Basic R Calculations

```
3^4 # q1 a)
```

```
## [1] 81
```

```
log(100, base = 7) # 1.b)
```

```
## [1] 2.366589
```

```
# 1.c)
x <- seq(1, 100)
sum(sapply(x, function(x) {1/(x^2)}))
```

```
## [1] 1.634984
```

```
100 %% 7 # 1.d)
```

```
## [1] 2
```

```
# 1.e)
dx_steps <- 0.001
x_val <- seq(0, pi/2, by = dx_steps)
sum(sapply(x_val, function(x){ sin(x) * dx_steps }))
```

```
## [1] 0.9997036
```

```
# 1.f)
f <- function(x) {
  return (dexp(x, rate = 1/2))
}

dx_steps <- 0.001
x_val <- seq(0, 3, by = dx_steps)
sum(sapply(x_val, function(x){ dexp(x, rate = 1/2) * dx_steps }))
```

```
## [1] 0.7771756
```

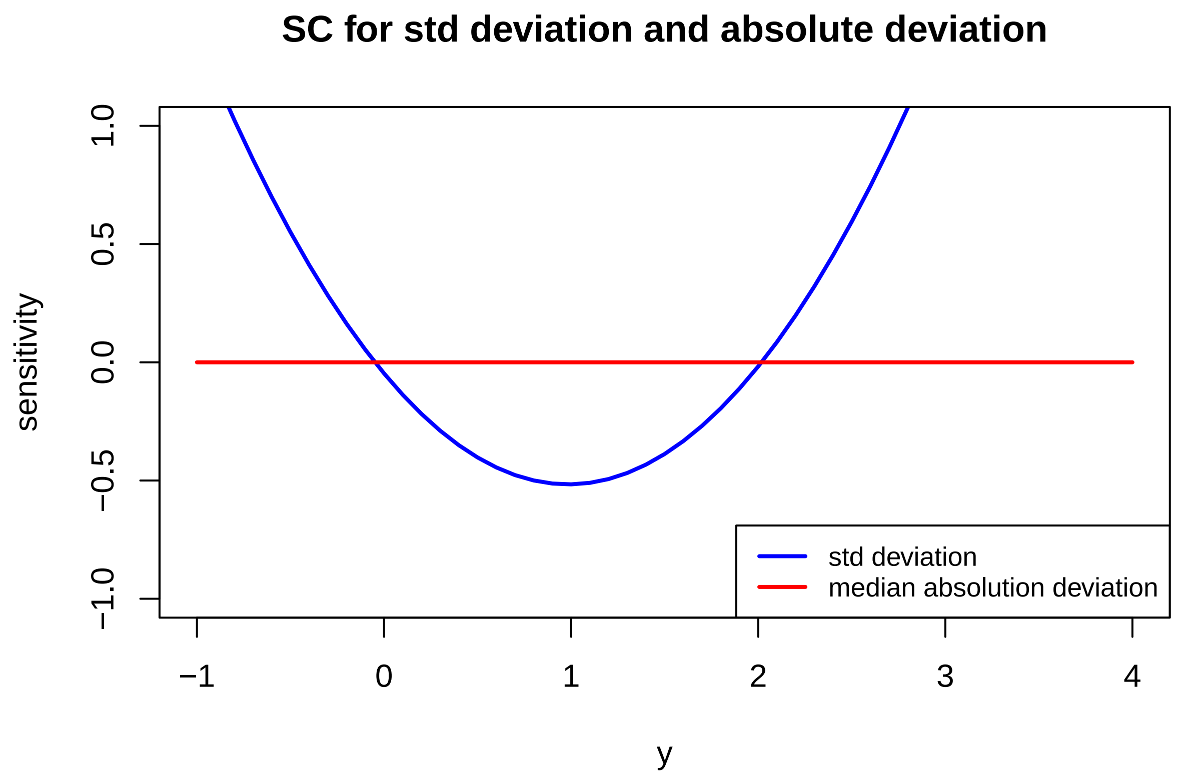
```
# 1.g)
f <- function(x) {
  return (x^2 + 3)
}

dx_steps <- 0.0001
x_val <- seq(-2, 2, by = dx_steps)
sum(sapply(x_val, function(x){ f(x) * dx_steps }))
```

```
## [1] 17.33403
```

Question 2: Comparing Spread Attributes

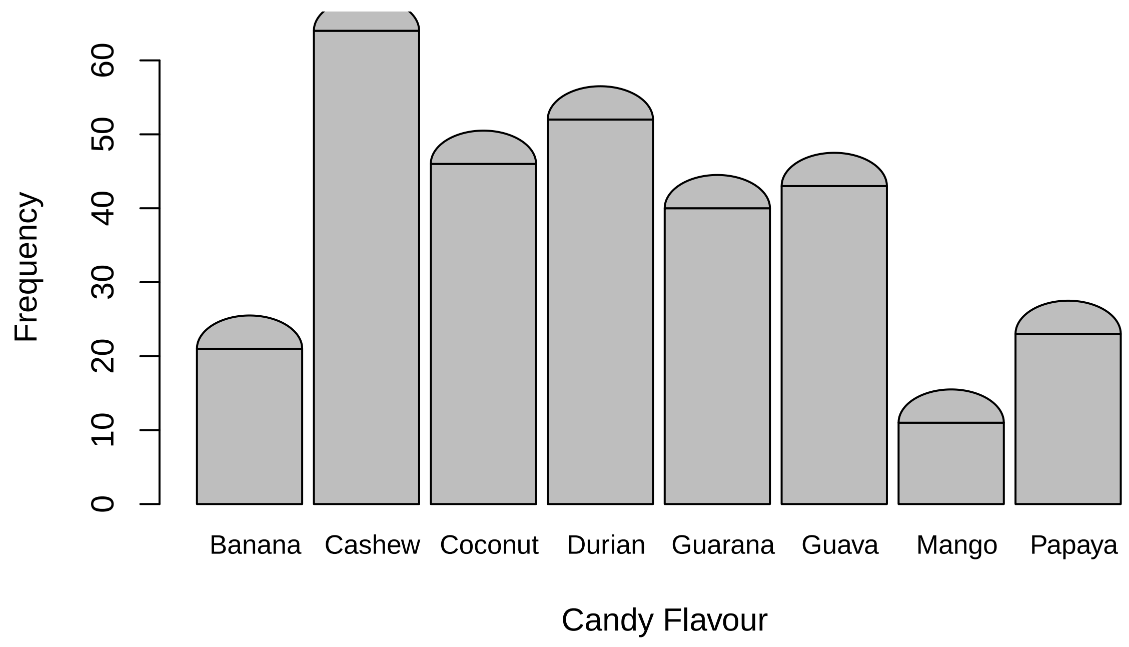
```r
# 2.d)
SD <- function(y) {
  return (sqrt(sum((y - mean(y))^2) / length(y)))
}

MAD <- function(y) {
  return (median(y - median(y)))
}
```

```r
# 2.e)
set.seed(341)
pop = rexp(1000)

y_val <- seq(-1, 4, by=0.1)

sc <- function(pop, y, attr){
  N <- length(pop) + 1
  return (sapply(y, function(y) { (N * (attr(c(pop, y)) - attr(pop))) }))
}

delta_sd <- sc(pop, y_val, SD)
delta_mad <- sc(pop, y_val, MAD)

plot(y_val, delta_sd, type="l", lwd = 2,
     main="SC for std deviation and absolute deviation", ylab="sensitivity", xlab="y",
     xlim=c(-1,4), ylim=c(-1, 1), col="blue")
lines(y_val, delta_mad, type="l", lwd = 2, main="Sensitivity curve for the median absolute deviation", 

legend(x = "bottomright",          # Position
       legend = c("std deviation", "median absolution deviation"),  # Legend texts
       col = c("blue", "red"), # Line colors
       cex = 0.8,
       lwd = 2)
```



Question 3: Write a rounded-barplot-making function

```
# 3.a)
rounded.barplot <- function(x, xlab){
  table_x <- table(x)
  categories <- names(table_x)
  categories_frequencies <- as.numeric(table_x)

  plot.new()
  plot(NULL, type="n", xlim=c(0, 10*length(categories_frequencies)), ylim=c(0, max(categories_frequenci

  axis(2, at=seq(from=0, to=max(categories_frequencies), by=10))
  mtext(xlab, side=1, line=2)
  mtext("Frequency", side=2, line=3)

  x_semi <- seq(-4.5, 4.5, by=0.01)
  y_semi <- sqrt(20.25-x_semi^2)

  for (i in c(1: length(categories_frequencies))){
    rect(10*(i-1), 0, 10*i-1, categories_frequencies[i], col = "gray", border = "black")
    mtext(categories[i], 1, at=10*i-5, cex=0.85)
    polygon(x_semi + 4.5 + 10*(i-1), y_semi + categories_frequencies[i], col = "gray")
  }
}
# 3.b)
set.seed(12345)
flavours = c("Mango","Papaya","Banana","Coconut","Guava","Guarana","Durian","Cashew")
candies = sample(flavours, size=300, prob=(1:8)/sum(1:8), replace=TRUE)

rounded.barplot(candies, xlab="Candy Flavour")
```



Question 4: R Analysis Question

```
# 4.a)
setwd("C:/Users/2baja/OneDrive/Desktop/STAT 341/A1")
apartment_eval <- read.csv("Apartment_Building_Evaluation.csv")

score_90 <- apartment_eval[,"SCORE"] >= 90
sum(score_90)
```

```
## [1] 410
```

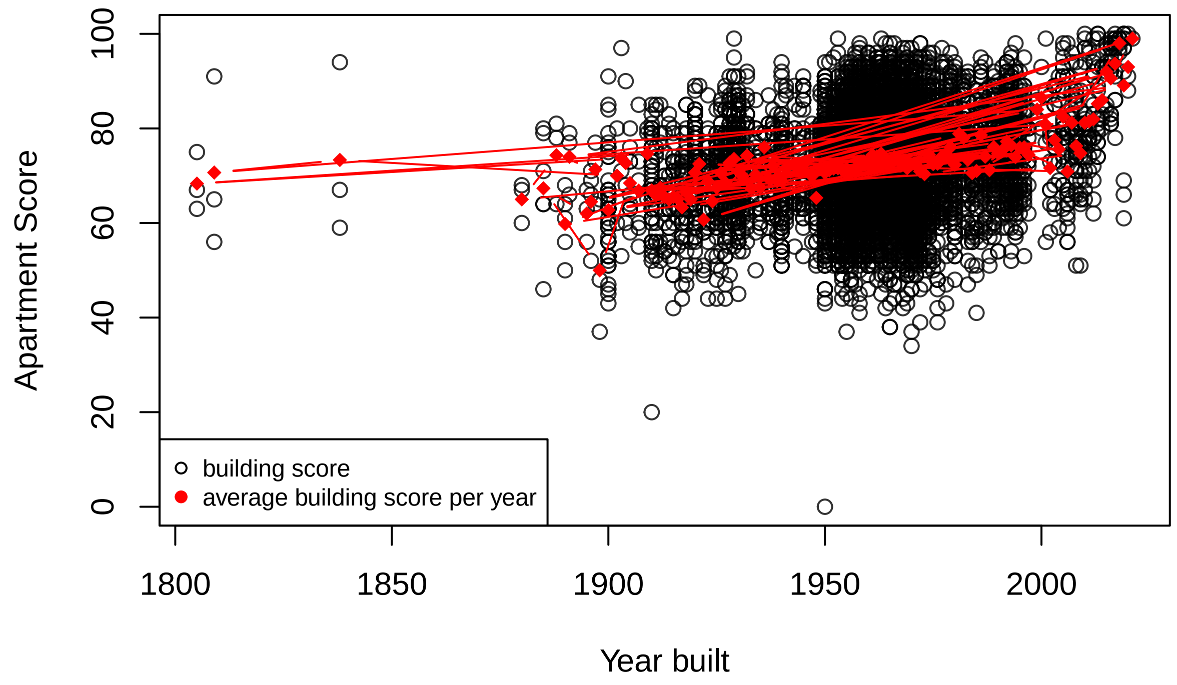
```
# 4.b)
davenport <- which(apartment_eval[,"WARDNAME"] == "Davenport")
davenport_apartments <- apartment_eval[davenport,]
davenport_apartments_sorted_addresses <- davenport_apartments[order(-davenport_apartments$SCORE),"SITE_
davenport_apartments_sorted_addresses[c(1:5)]
```

```
## [1] "1544 DUNDAS ST W"  "1544 DUNDAS ST W"  "1289 DUNDAS ST W"
## [4] "19-21 RUSHOLME RD" "410 DOVERCOURT RD"
```

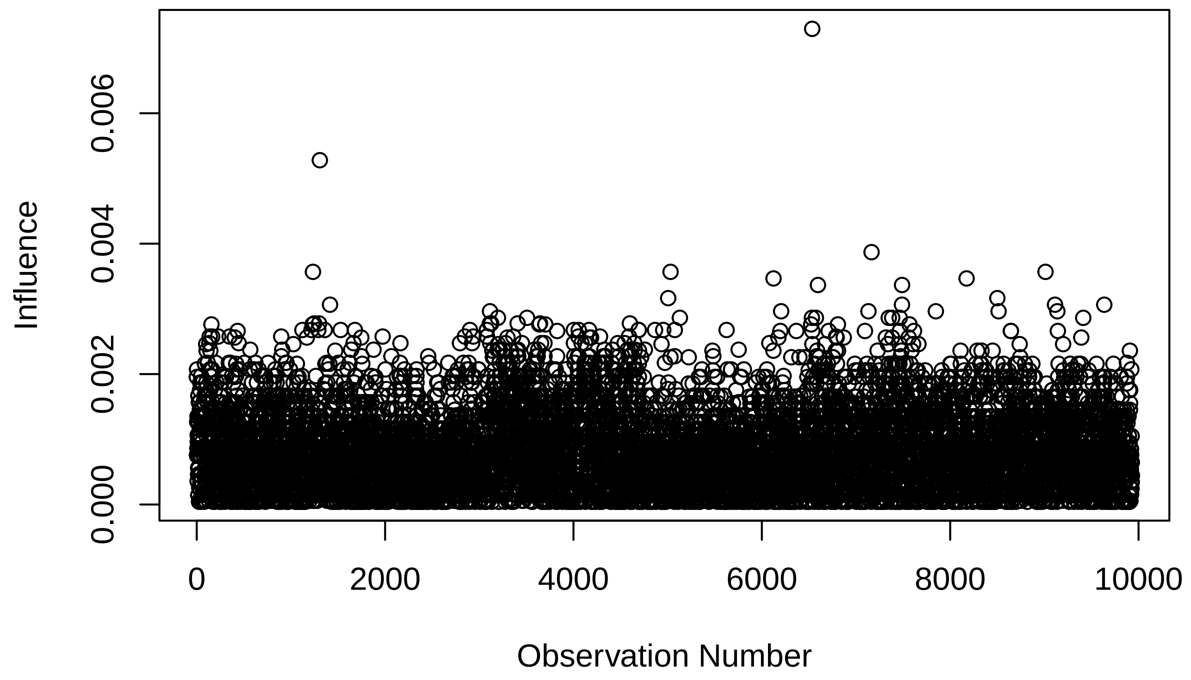
```
# 4.c)
unique_wardnames <- unique(apartment_eval[,"WARDNAME"])
sapply(unique_wardnames, function(name) { mean(apartment_eval[which(apartment_eval$WARDNAME == name), "S
```

```
##    Scarborough Southwest       Eglinton-Lawrence    Scarborough-Agincourt
##                 72.03354                72.17902                 78.33333
##        Beaches-East York               Davenport        Spadina-Fort York
##                 72.44581                68.86260                 75.14400
##         Toronto-Danforth          Toronto Centre        Toronto-St. Paul's
##                 73.21563                71.90877                 73.62217
##      University-Rosedale      York South-Weston Humber River-Black Creek
##                 71.81912                70.28017                 68.79331
##               Willowdale    Scarborough-Guildwood       Scarborough Centre
##                 76.86667                72.28054                 74.51587
##         Etobicoke Centre          Don Valley East              York Centre
##                 72.14054                76.30913                 71.53305
##          Don Valley West        Parkdale-High Park      Etobicoke-Lakeshore
##                 76.69196                69.34385                 71.47331
##          Etobicoke North        Scarborough North        Don Valley North
##                 69.30645                81.50000                 79.19310
##   Scarborough-Rouge Park
##                 75.05479
```

```
# 4.d)
plot(apartment_eval$YEAR_BUILT, apartment_eval$SCORE, pch = 1, col=adjustcolor("black", alpha = 0.8), xl

unique_years <- unique(apartment_eval[,"YEAR_BUILT"])
average_score_by_year <- sapply(unique_years, function(year_built) { mean(apartment_eval[which(apartmen

lines(unique_years, average_score_by_year, pch = 18, col="red", type="b")

legend(x = "bottomleft",          # Position
       legend = c("building score", "average building score per year"),  # Legend texts
       col = c("black", "red"),           # Line colors
       cex = 0.75,
       pch = c(1, 19))
```

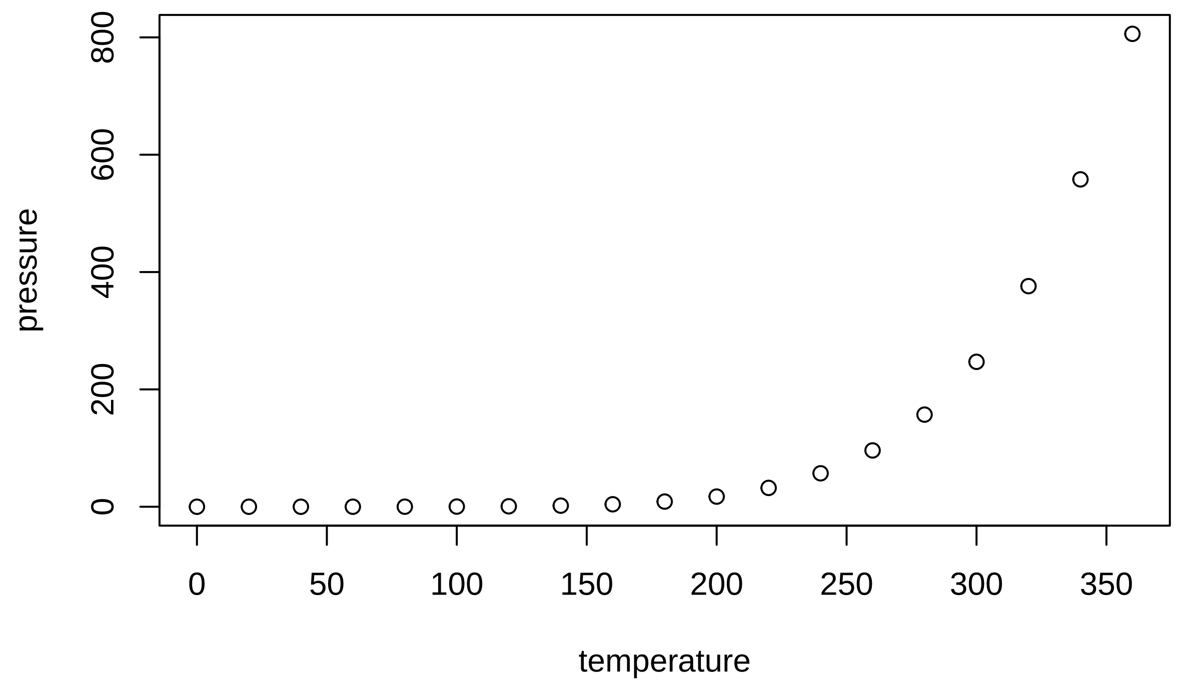


```
# 4.e)
influence_values <- function(pop, attribute){
  N <- length(pop)
  attribute_total_pop <- attribute(pop)

  return (sapply(1:N, function(x) { abs(attribute_total_pop - attribute(pop[-x])) }))
}

mean_influence <- influence_values(apartment_eval$SCORE, mean)

plot(1:length(apartment_eval$SCORE), mean_influence, xlab = "Observation Number", ylab = "Influence")
```



Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.