HW1

2023-04-17

Problem 1

```
library(qeML, quietly=T)
```

```
##
##
##
##
##
## *********************
##
##
##
## Latest version of regtools at GitHub.com/matloff
##
##
## Type ?regtools to see function list by category
```

```
## Latest version of partools at GitHub.com/matloff
```

```
##
##
##
##
##
## *********************
##
##
##
##   Navigating qeML:
##
##       Type vignette("Quick_Start") for a quick overview!
##
##       Type vignette("FtnList") for a categorized function list
##
##       Type vignette("ML_Overview") for an introduction to machine learning
```

```
data(pef)
head(pef)
```

```
##        age     educ occ sex wageinc wkswrkd
## 1 50.30082 zzzOther 102   2   75000      52
## 2 41.10139 zzzOther 101   1   12300      20
## 3 24.67374 zzzOther 102   2   15400      52
## 4 50.19951 zzzOther 100   1       0      52
## 5 51.18112 zzzOther 100   2     160       1
## 6 57.70413 zzzOther 100   1       0       0
```

Get accuracy of predicting different variables:

```
# Predict wage from all variables; MAE is > $25000
qeLin(pef,'wageinc')$testAcc
```

```
## holdout set has  1000 rows
```

```
## [1] 25013.19
```

```
# Predict gender from alll variables; Incorrect 24% of the time
qeLogit(pef,'sex')$testAcc
```

```
## holdout set has  1000 rows
```

```
## [1] 0.243
```

```
# Predict occupation (6 jobs); Incorrect 64% of the time
qeLogit(pef,'occ')$testAcc
```

```
## holdout set has  1000 rows
```

```
## [1] 0.638
```

Description:

Format: `takeALookAround(data,yName,sName,maxFeatureSetSize)`

`yName` and `sName` are the names of columns in the data frame `data`

The tool will investigate the impact of using various subsets of the feature set on

- **(a)** prediction accuracy for Y of this feature set
- **(b)** prediction accuracy for Y of this feature set plus S (how much are we giving up by NOT using S?)
- **(c)** prediction accuracy for S of this feature set (the better the accuracy, the less fairness value there is in omitting S from prediction model)

The return value will be a data frame with 4 columns.

1. A character string consisting of the names of the given feature set.
2. The prediction accuracy for (a)
3. The prediction accuracy for (b)
4. The prediction accuracy for (c)

There will be one row in the d.f. for each feature set. All possible feature sets of size up to **maxFeatureSetSize** will be investigated.

Case Study with 2 Features:

```
data <- pef
yName <- "wageinc"
sName <- "sex"
maxFeatureSetSize <- 1000

# Select columns excluding yName and sName
featureSet <- colnames(data[,!names(data) %in% c(yName, sName)])
results <- data.frame(matrix(nrow=0, ncol=4))
colnames(results) <- c("features","Y","YS","Accuracy")

i <- 1
j <- 2

feature1 <- featureSet[i] # age
feature2 <- featureSet[j] # educ

data_a <- data[,c(feature1, feature2, yName)]
data_b <- data[,c(feature1, feature2, yName, sName)]
data_c <- data[,c(feature1, feature2, sName)]

if (is.numeric(data[1,yName])) {
  acc_a <- qeLin(data_a,yName)$testAcc
  acc_b <- qeLin(data_b,yName)$testAcc
} else if (is.factor(data[1,yName])) {
  acc_a <- qeLogit(data_b,yName)$testAcc
  acc_b <- qeLogit(data_b,yName)$testAcc
} else {
  acc_a <- NA
  acc_b <- NA
}
```

```
## holdout set has  1000 rows
## holdout set has  1000 rows
```

```
if (is.numeric(data[1,sName])) {
  acc_c <- qeLin(data_c,sName)$testAcc
} else if (is.factor(data[1,sName])) {
  acc_c <- qeLogit(data_c,sName)$testAcc
} else {
  acc_c <- NA
}
```

```
## holdout set has  1000 rows
```

```
result <-c(paste(feature1,feature2,sep=","), acc_a, acc_b, acc_c)
results <- rbind.data.frame(results, result)
```

Implementation:

```
takeALookAround <- function(data, yName, sName, maxFeatureSetSize) {
  library(qeML)

  # Select columns excluding yName and sName
  featureSet <- colnames(data[,!names(data) %in% c(yName, sName)])
  results <- data.frame(matrix(nrow=0, ncol=4))

  # counter for feature set comparisons
  count <- 0

  # iterate through unique combinations of features
  for (i in 1:length(featureSet)) {
    for (j in i:length(featureSet)) {
      if (i != j) {
        feature1 <- featureSet[i] # i.e. age
        feature2 <- featureSet[j] # i.e. educ

        # a: predicting y from feature set
        data_a <- data[,c(feature1, feature2, yName)]
        # b: predicting y from feature set plus S
        data_b <- data[,c(feature1, feature2, yName, sName)]
        # c: predicting S from feature set
        data_c <- data[,c(feature1, feature2, sName)]

        # use qeLin on numeric data to predict y
        # use qeLogit on factor data to predict y
        if (is.numeric(data[1,yName])) {
          acc_a <- qeLin(data_a,yName)$testAcc # without S
          acc_b <- qeLin(data_b,yName)$testAcc # with S
        } else if (is.factor(data[1,yName])) {
          acc_a <- qeLogit(data_a,yName)$testAcc # without S
          acc_b <- qeLogit(data_b,yName)$testAcc # with S
        } else {
          acc_a <- NA # invalid data
          acc_b <- NA # invalid data
        }
        # use qeLin on numeric data to predict S
        # use qeLogit on factor data to predict S
        if (is.numeric(data[1,sName])) {
          acc_c <- qeLin(data_c,sName)$testAcc # predict S
        } else if (is.factor(data[1,sName])) {
          acc_c <- qeLogit(data_c,sName)$testAcc # predict S
        } else {
          acc_c <- NA
        }
        # names of features, accuracy a, b, and c
        result <-c(paste(feature1,feature2,sep=","), acc_a, acc_b, acc_c)
        results <- rbind.data.frame(results, result)
        count <- count+1
        # break if limit met
        if (count == maxFeatureSetSize) break
      }
    }
    # break if limit met
    if (count == maxFeatureSetSize) break
```

```
  }
  print(count)
  colnames(results) <- c("features","Y_acc","YS_acc","S_acc")
  # df with names and accuracies
  return(results)
}
```

Test Function on Example

```
wageinc_sex <- takeALookAround(data=pef,
                               yName="wageinc",
                               sName="sex",
                               maxFeatureSetSize=100)
```

```
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## [1] 6
```

```
print(wageinc_sex)
```

```
##       features            Y_acc           YS_acc S_acc
## 1     age,educ 29463.4035075258 30881.8935401557 0.262
## 2      age,occ 33317.2518673905 30940.0713715154  0.23
## 3  age,wkswrkd 24860.6038519676 26924.9547367294 0.255
## 4     educ,occ 29537.4498860143 32008.5139931088 0.255
## 5 educ,wkswrkd 26353.4885525869  25850.429543911 0.235
## 6  occ,wkswrkd 26543.2019554784 27689.0198615072 0.245
```

```
wageinc_educ <- takeALookAround(data=pef,
                               yName="wageinc",
                               sName="educ",
                               maxFeatureSetSize=100)
```

```
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## holdout set has  1000 rows
## [1] 6
```

```
print(wageinc_educ)
```

```
##      features            Y_acc           YS_acc S_acc
## 1     age,occ 30015.4152380695 30740.4514283973 0.252
## 2     age,sex 28542.9503874266 30309.8093888652 0.256
## 3 age,wkswrkd 26852.8543669288   25392.81913911  0.25
## 4     occ,sex 31146.1673153082 32336.5040799081 0.271
## 5 occ,wkswrkd 27386.9170284261 26105.3201175623 0.235
## 6 sex,wkswrkd 26325.1351417685 26045.4536856452 0.239
```

Problem 2

Description:

Simpson's Paradox

relation(X,Y) is + but relation(X,Y|Z) is -, or vice versa

Usually, e.g. in the **UCBAdmissions** data, the variables X, Y and Z are all categorical. Your code here will explore whether SP seems to hold for numeric X and Y (and maybe Z) for some given dataset

Variables

- **data** is an R data frame or equivalent
- **xName** is the name of the X column
- **yName** is the name of the Y column
- **zName** is the name of the Z column
- **numZvals** is the number of intervals to break Z into in the case where Z is a continuous variable

X and Y should either be continuous numeric variables or dichotomous R factors.

```
library(qeML)
library(Kendall)
data(pef)
head(pef)
```

```
##        age     educ occ sex wageinc wkswrkd
## 1 50.30082 zzzOther 102   2   75000      52
## 2 41.10139 zzzOther 101   1   12300      20
## 3 24.67374 zzzOther 102   2   15400      52
## 4 50.19951 zzzOther 100   1       0      52
## 5 51.18112 zzzOther 100   2     160       1
## 6 57.70413 zzzOther 100   1       0       0
```

We will define “relation” as one of two types of correlation:

- If both X and Y are continuous, use the classical Pearson correlation, available in R as **cor()**.
- If at least one of X and Y is dichotomous, use *Kendall’s tau* correlation, code available in the **Kendall** package. You should be able to use **install.packages()** to get this code.

```
cor(pef[,"age"],pef[,"wageinc"])
```

```
## [1] 0.1131102
```

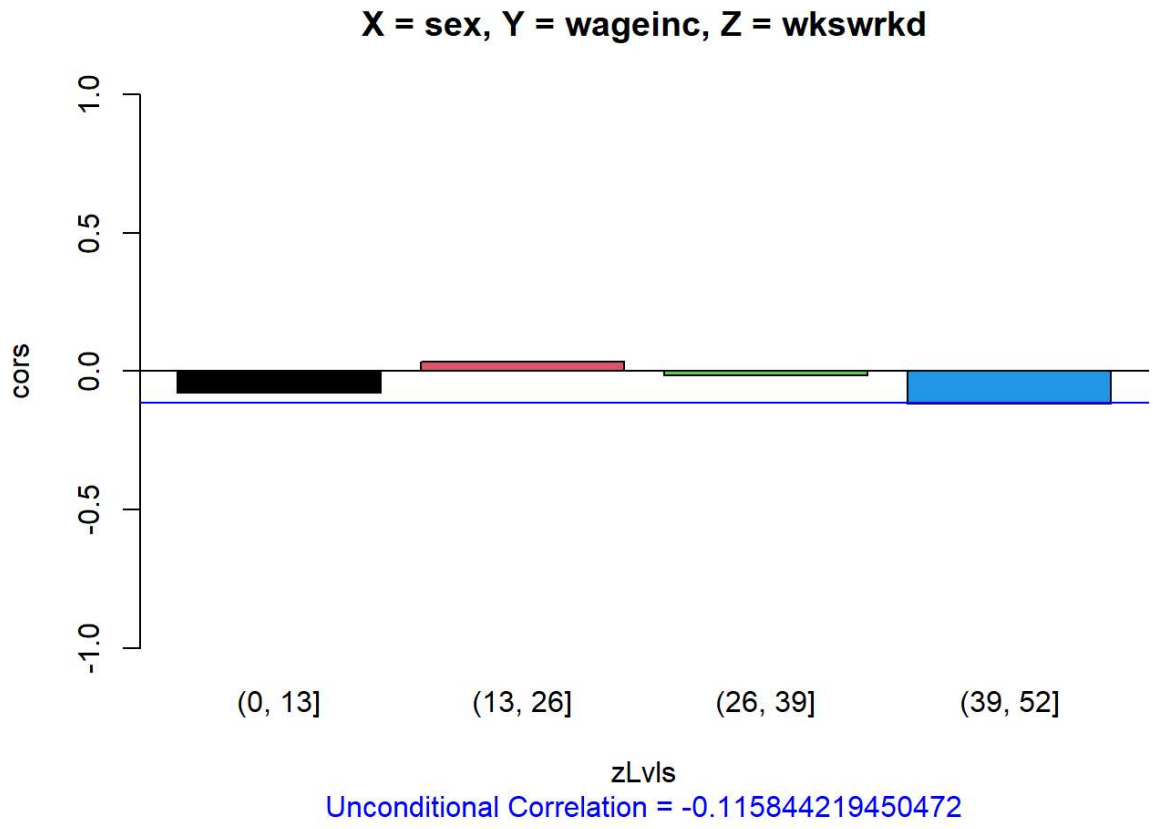
Implementation

```
numericSimpsonfunction <- function(data,xName,yName,zName,numZvals=NULL){
  #sort by Z value
  df_ordered <- data[order(data[,zName]),]
  size <- length(df_ordered[,xName])

  if (is.numeric(data[, zName])){
    interval_val <- ceiling(df_ordered[size, zName]/numZvals)
    i_val <- interval_val
  } else {
    #Separate Z into levels if it is categorical
    interval_val <- levels(data[, zName])
    numZvals = length(interval_val)
    #Initialize i_val to avoid processing error, however this assignment will not be
    #i_val contains the next item to search for so that the values in interval_val remain unchanged
    i_val <- interval_val[1]
  }
  #p_points is an array containing the index of the last occurrence of an element in the data
  p_points <- c(1)
  #Get a vector showing the indices of intervals
  for (i in 1:numZvals){
    val <- match(i_val, df_ordered[,zName])
    #If no match exists, find the closest element
    if (is.na(val)){
      val <- which.min(abs(df_ordered[,zName] - i_val))
    }
    if (is.numeric(data[,zName])){
      #Include the upper bound and exit the loops at the last iteration
      if (i == numZvals){
        p_points <- append(p_points, size)
        break
      }
      #otherwise continue through
      p_points <- append(p_points, val)
      i_val <- i_val + interval_val
    } else{
      #First element in p_points should be 1, not the the last occurrence of the first category
      #Therefore, set i_val to the next element to find and continue the loop
      if (i == 1){
        i_val <- interval_val[i+1]
        next
        }
      p_points <- append(p_points, val)
      #increment i_val
      i_val <- interval_val[i+1]
      #Include the upperbound
      if (i == numZvals){
        p_points <- append(p_points, size)
      }
    }
  }

  #CorrList will contain an array of all the calculated correlations
  CorrList <- c()
  for (i in 1:numZvals){
    #p_points[i] is lower bound of each partition, p_points[i+1] is the upper bound
```

```
    xVar <- as.numeric(df_ordered[p_points[i]:p_points[i+1], xName])
    yVar <- as.numeric(df_ordered[p_points[i]:p_points[i+1], yName])

    if (is.factor(data[1,xName]) || is.factor(data[1,yName])){
      Corr <- cor(xVar, yVar, method = 'kendall', use = 'complete.obs')
    }
    else if (is.numeric(data[1,xName]) && is.numeric(data[1, yName])){
      Corr <- cor(xVar, yVar, use = 'complete.obs')
    }
    CorrList <- append(CorrList, Corr)
  }

  #====================Graphing=========================
  header <- paste("X = ", xName, ", Y = ", yName, ", Z = ", zName, sep = "")
  lb_list <- c()
  for (i in 1:numZvals){
    #if its numeric none of the bounds information
    if (is.numeric(data[,zName])){
      if (i == 1){
        #first and second give the numeric values of the Z-Bounds
        first <- df_ordered[p_points[i]+1, zName]
        second <- df_ordered[p_points[i+1], zName]
      }else{
        first <- df_ordered[p_points[i], zName]
        second <- df_ordered[p_points[i+1], zName]
      }
      label <- paste("(", first, ", ", second, "]", sep = "")
    }else{
      if (i == 1){
        label <- df_ordered[p_points[i]+1, zName]
      }else{
        label <- df_ordered[p_points[i], zName]
      }
    }
    lb_list <- append(lb_list, label)
  }
  uncCor <- cor(as.numeric(data[,xName]), as.numeric(data[,yName]))
  sub <- paste("Unconditional Correlation = ", uncCor, sep = "")
  barplot(CorrList,
          col = palette(),
          names.arg = lb_list,
          main = header,
          ylab = "cors",
          xlab = "zLvls",
          ylim = c(-1,1),
          sub = sub,
          col.sub = "blue"
        )

  abline(h=0, col = "black")
  abline(h = uncCor, col = "blue")
}
```

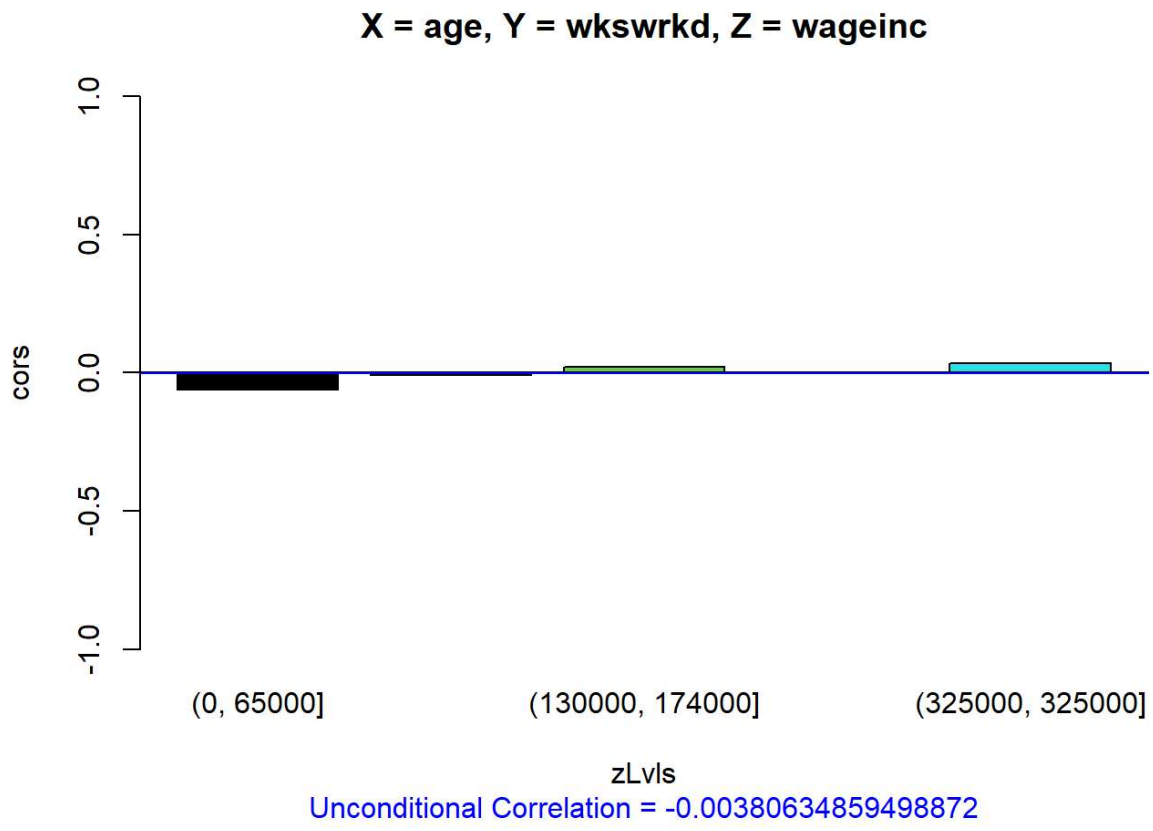
Test Samples

```
numericSimpsonfunction(pef, 'sex', 'wageinc','wkswrkd',4)
```



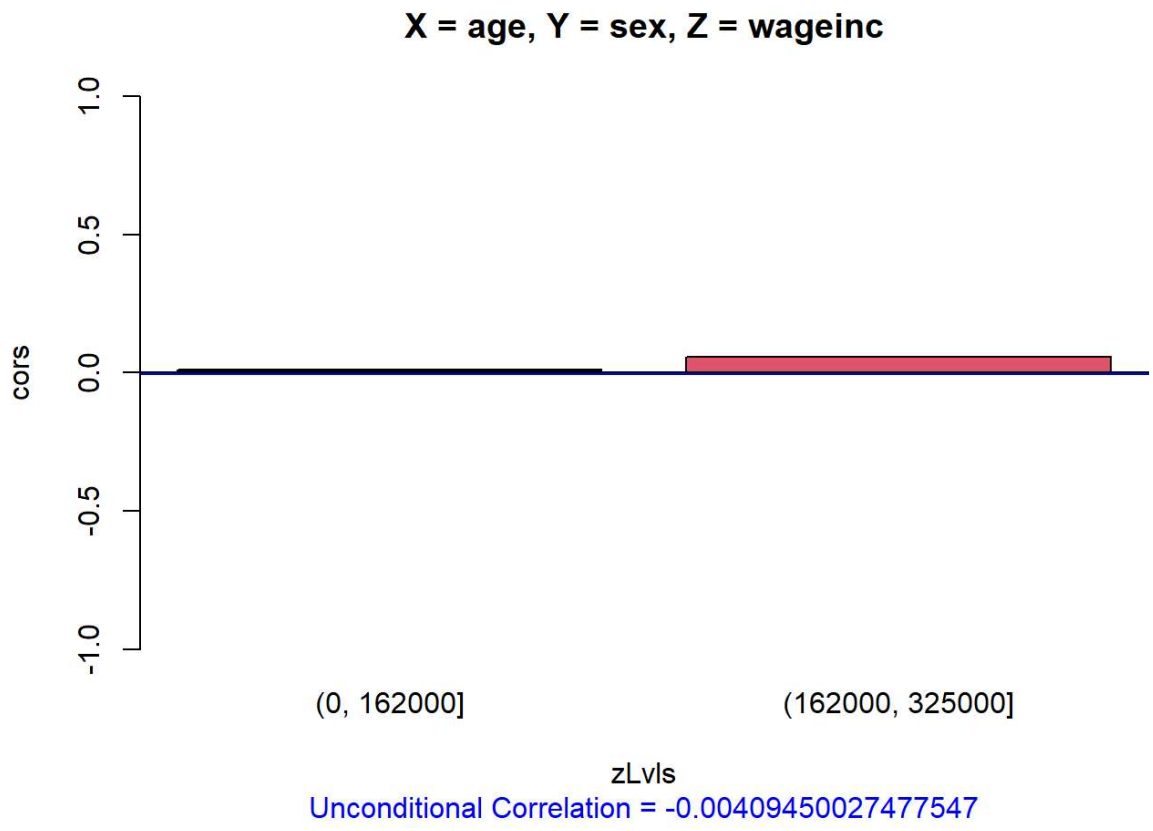

```
data <- pef
xName <- "age" # continuous numeric
yName <- "wkswrkd" # continuous numeric
zName <- "wageinc" # numeric
numZvals <- 5
numericSimpsonfunction(data,xName,yName,zName,numZvals)
```

```
## Warning in cor(xVar, yVar, use = "complete.obs"): the standard deviation is
## zero
```



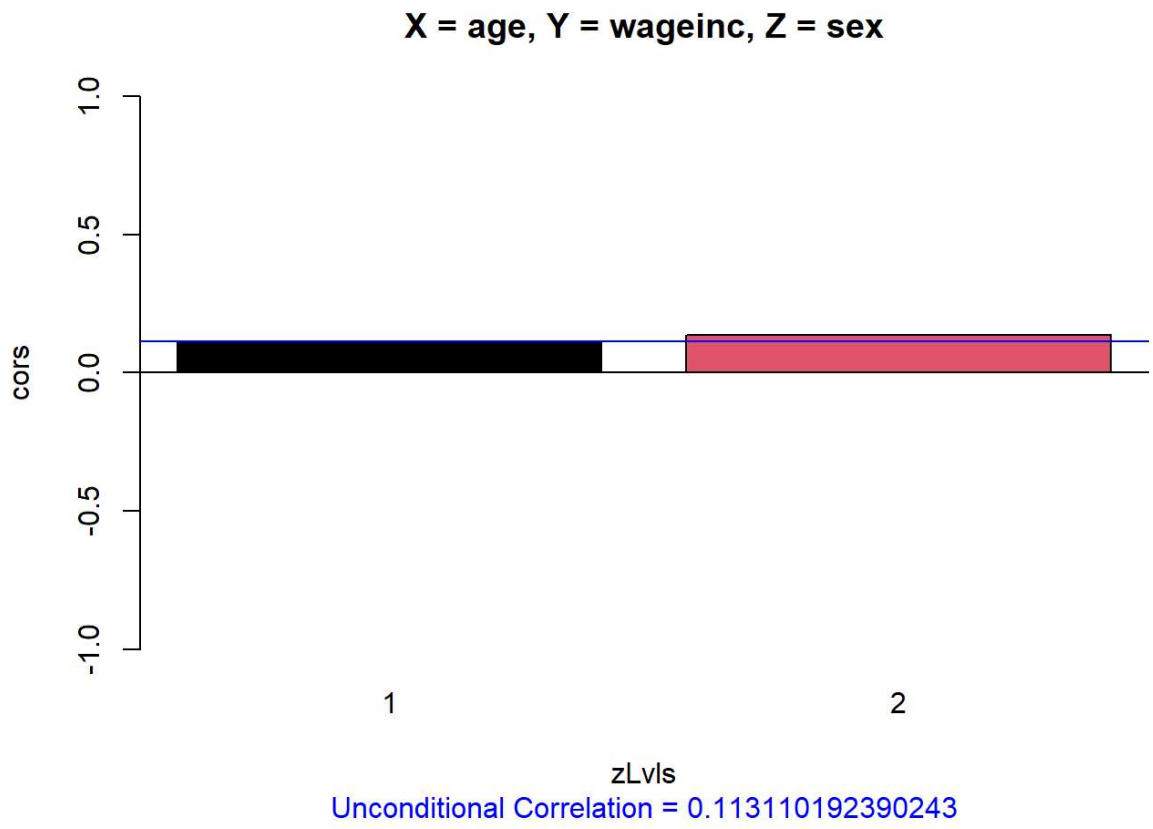
Unconditional Correlation = -0.00380634859498872

```
data <- pef
xName <- "age" # continuous numeric
yName <- "sex" # dichotomous factor
zName <- "wageinc" # numeric
numericSimpsonfunction(data,xName,yName,zName,2)
```



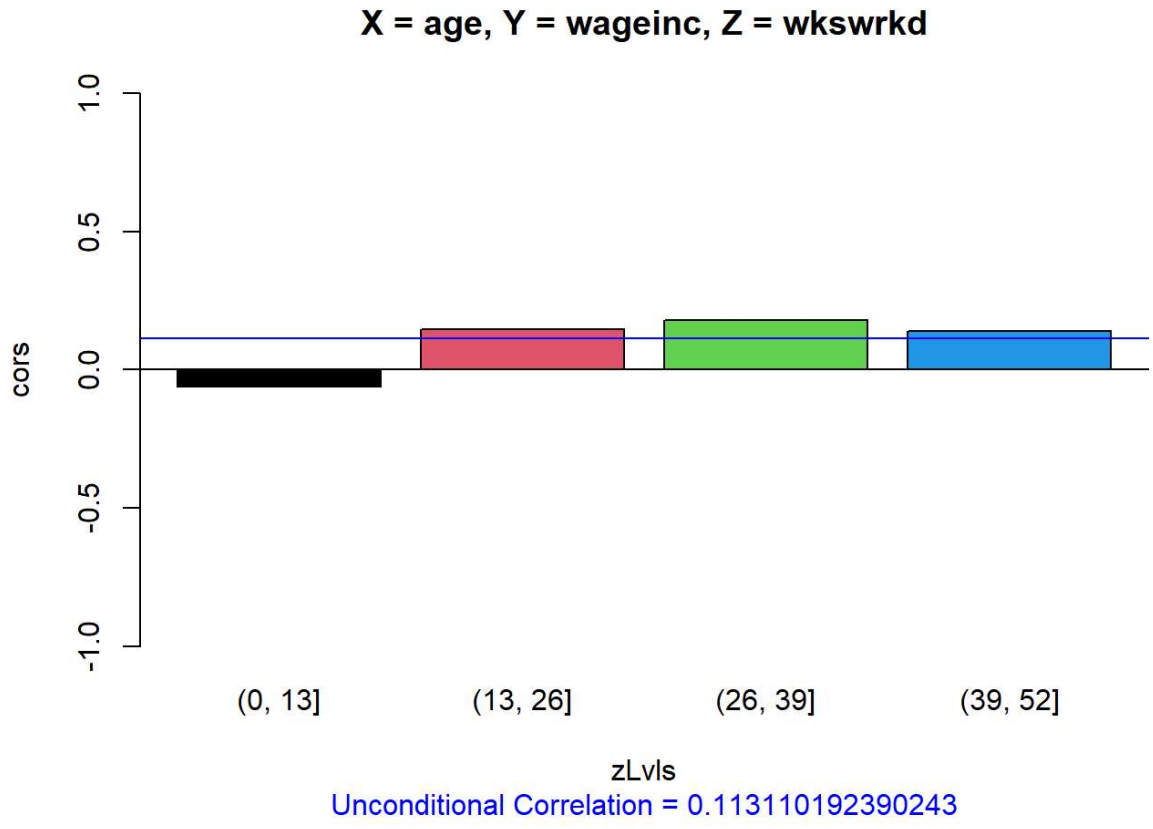
Unconditional Correlation = -0.00409450027477547

```
data <- pef
xName <- "age" # continuous numeric
yName <- "wageinc" # continuous numeric
zName <- "sex" # categorical
numericSimpsonfunction(data,xName,yName,zName,numZvals=NULL)
```



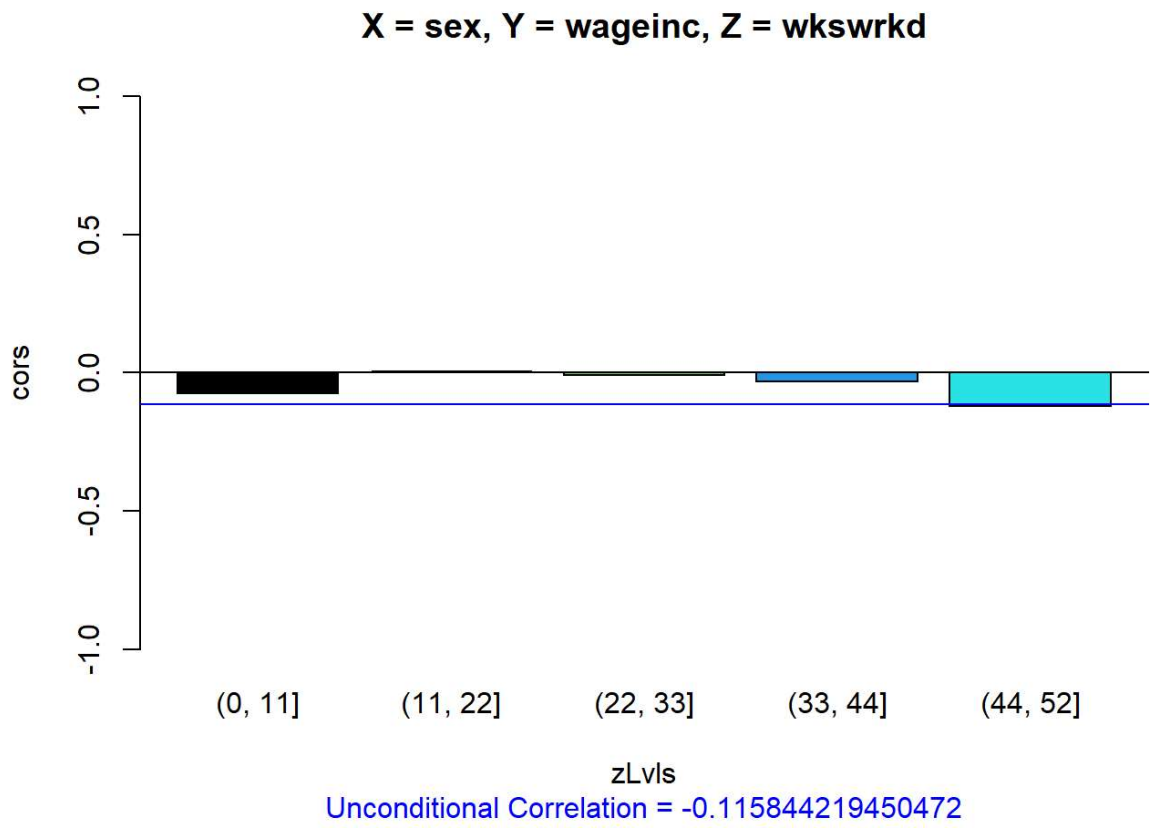
Unconditional Correlation = 0.113110192390243

```
numericSimpsonfunction(pef,'age','wageinc','wkswrkd',4)
```



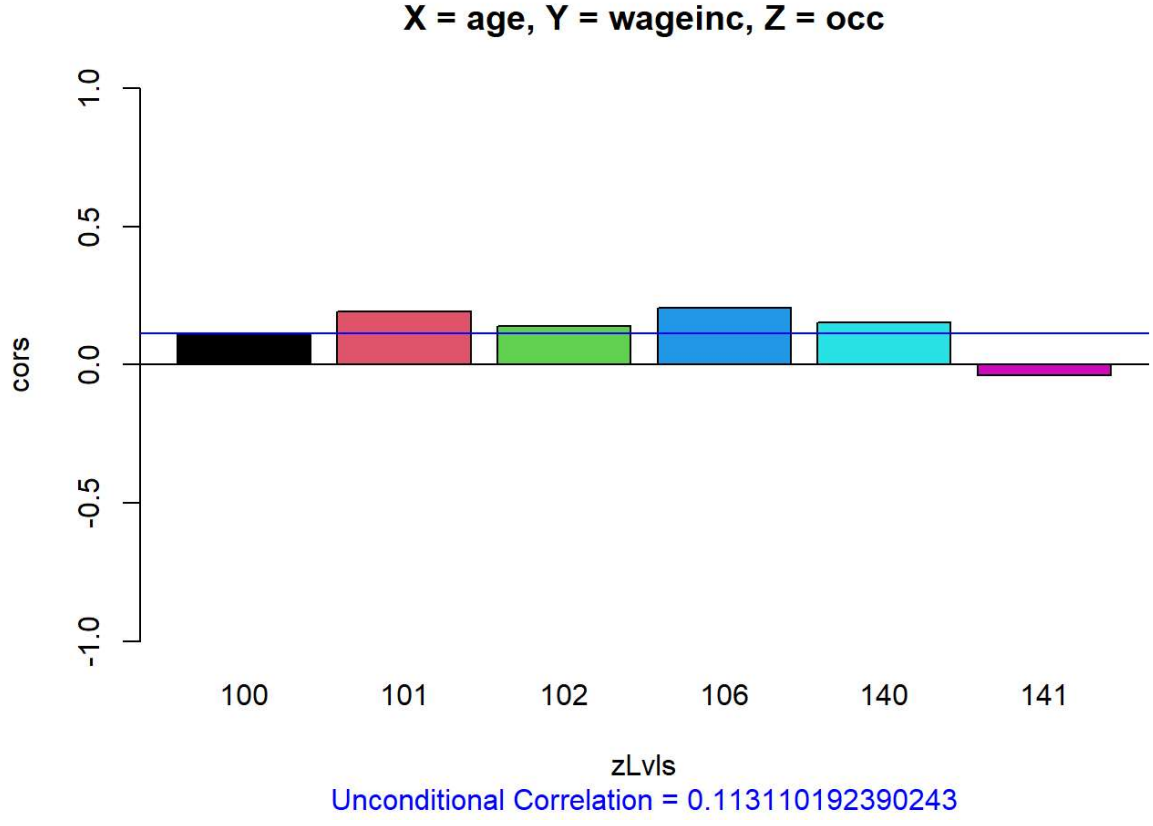
Unconditional Correlation = 0.113110192390243

```
numericSimpsonfunction(pef,'sex','wageinc','wkswrkd',5)
```



Unconditional Correlation = -0.115844219450472

```
numericSimpsonfunction(pef,'age','wageinc','occ',4)
```



Unconditional Correlation = 0.113110192390243

Extra Credit B

It would be nice to condition on two factors, e.g. gender and occupation. There Z would be an R factor with 2 X 6 = 12 levels, representing a categorical variable of 12 categories.

Write a function with call form

```
superFactor(f1,f2)
```

which returns a new factor that is a combination of factors **f1** and **f2** as described above.

```
superFactor <- function(f1,f2) {
  f3 <- as.factor(paste(f1,f2,sep="_"))
  return (f3)
}
```

```
unique(superFactor(pef$sex, pef$occ))
```

```
##  [1] 2_102 1_101 1_100 2_100 2_101 1_141 1_102 2_140 1_140 2_106 2_141 1_106
## 12 Levels: 1_100 1_101 1_102 1_106 1_140 1_141 2_100 2_101 2_102 ... 2_141
```

```
unique(superFactor(pef$sex, pef$educ))
```

```
## [1] 2_zzzOther 1_zzzOther 2_14       1_14       1_16       2_16
## Levels: 1_14 1_16 1_zzzOther 2_14 2_16 2_zzzOther
```

Extra Credit A

Find a real dataset, in which at least one of X and Y is a continuous variable, in which SP hold. Write your jode as a function call **spExample()** (no arguments). It will fetch the dataset, perform any needed preprocessing, and then call **numericSimpson()**.

```
#install.packages(c("devtools"))
#devtools::install_github("ldurazo/kaggler")
library(readr)
library(kaggler)
kgl_auth(creds_file = 'kaggle.json')
```

```
## <request>
## Options:
## * httpauth: 1
## * userpwd: sydneymw:7cc8736b11250e3c6ea35ae13e7933f0
```

```
response <- kgl_datasets_download_all(owner_dataset = "wduckett/californiaddsexpenditures")

download.file(response[["url"]], "temp.zip", mode="wb")
unzip_result <- unzip("temp.zip", overwrite = TRUE)
dds <- read_csv("californiaDDSDataV2.csv")
```

```
## Rows: 1000 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Age Cohort, Gender, Ethnicity
## dbl (3): Id, Age, Expenditures
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dds <- data.frame(dds)
```

Ethnicity-Expenditure Correlation Reduces when Age-Stratified

```
dds$Ethnicity <- as.factor(dds$Ethnicity)
dds$GenderBin <- as.numeric(dds$Gender=="Female")
dds$Age.Cohort <- as.factor(dds$Age.Cohort)

#numericSimpson(dds, "Expenditures", "GenderBin","Ethnicity")

data <- dds
xName <- "Expenditures" # continuous numeric
yName <- "Ethnicity" # continuous numeric
zName <- "Age.Cohort" # categorical
numericSimpsonfunction(data,xName,yName,zName,numZvals=NULL)
```