Homework 8 - Functions for Statistical Analysis in R

Morgan Southgate

March 8, 2017

Function: Linear Regression

Description: Fits a regression line to a scatterplot of x and y values

Input: continuous x & y variables

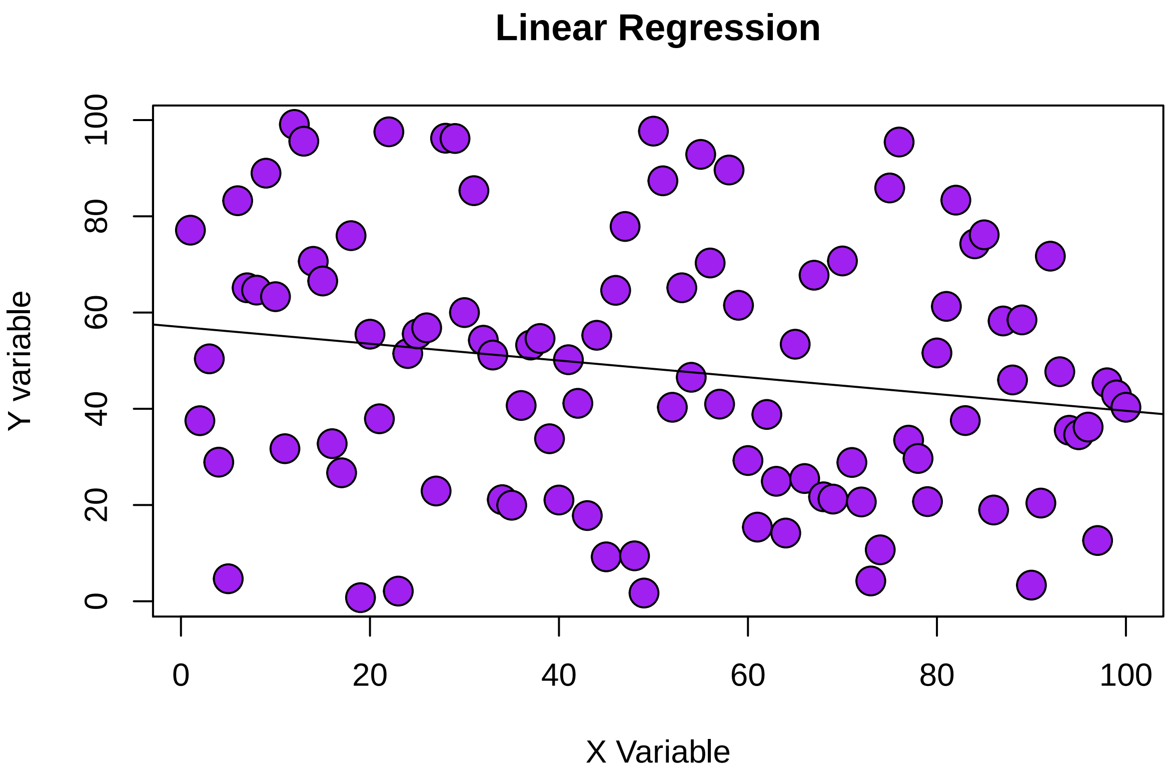
Output: abline & scatterplot

```
linReg <- function(xVar= 1:100,yVar=runif(100,min=0,max=100)){
  dataframe=data.frame(xVar,yVar)
  linRegMod <- lm(yVar~xVar,data=dataframe)
  linRegOut <- c(slope=summary(linRegMod)$coefficients[2,1],
                 pValue=summary(linRegMod)$coefficients[2,4])
    return(linRegOut)}

linReg()
```

```
##       slope      pValue
## -0.09198269  0.39187554
```

Function for plotting results of linear regression

```
linRegPlot <-  function(xVar=1:100,yVar=runif(100,min=0,max=100)){
  dataframe=data.frame(xVar,yVar)
  lrplot <- plot(y=dataframe$yVar,
                       x=dataframe$xVar,
                      cex=2,
                      pch=21,
                      bg="purple",
                      main="Linear Regression",
                      xlab="X Variable",
                      ylab="Y variable",
                      xlim=range(xVar),
                      ylim=range(yVar))
  linRegMod <- lm(yVar~xVar,data=dataframe)
  abline(linRegMod)
  return(lrplot)
}

linRegPlot()
```



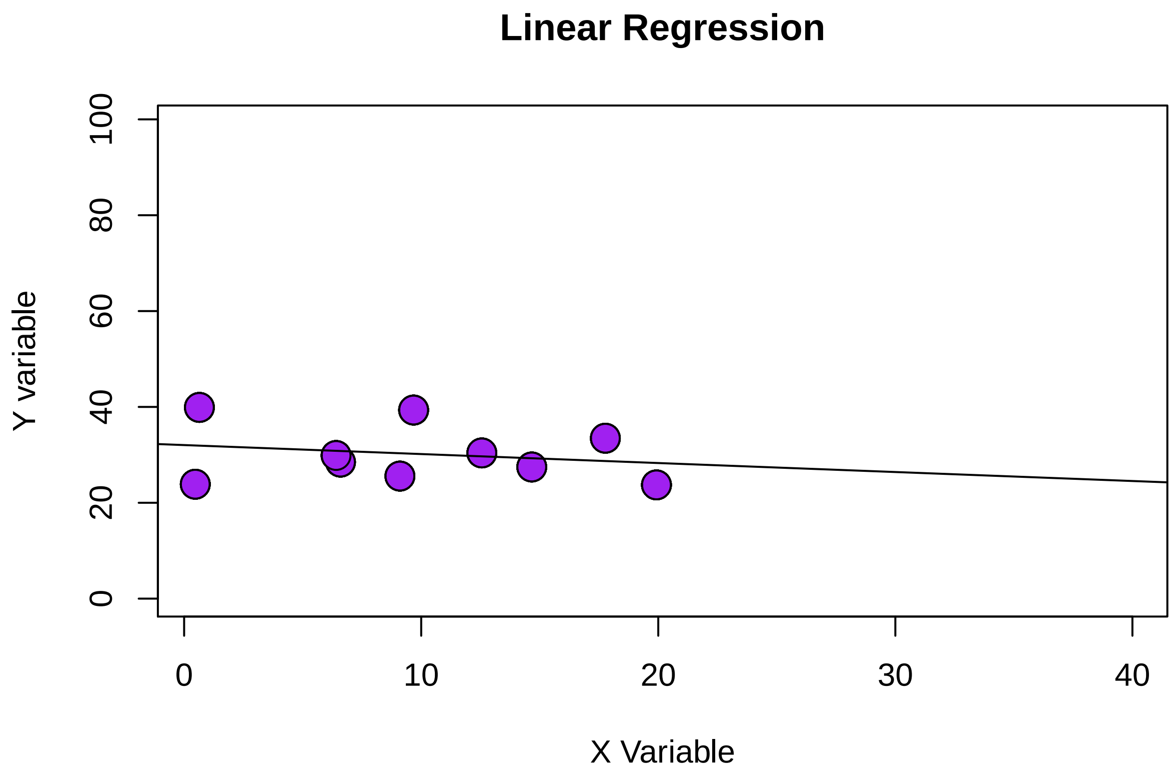

```
## NULL
```

Testing linReg() & linRegPlot() with tiny data set

```r
# make tiny data set
rdat <- data.frame(xVar=runif(10,min=0,max=20),yVar=runif(10,min=20,max=40))

# test linReg function with rdat
linReg(rdat)
```

```
##       slope      pValue
## -0.18751366  0.03330957
```

```r
# test linRegPlot function with rdat
linRegPlot(rdat)
```




```
## NULL
```

Function: Logistic Regression

Description:

Input: continuous x variable & categorical y variable

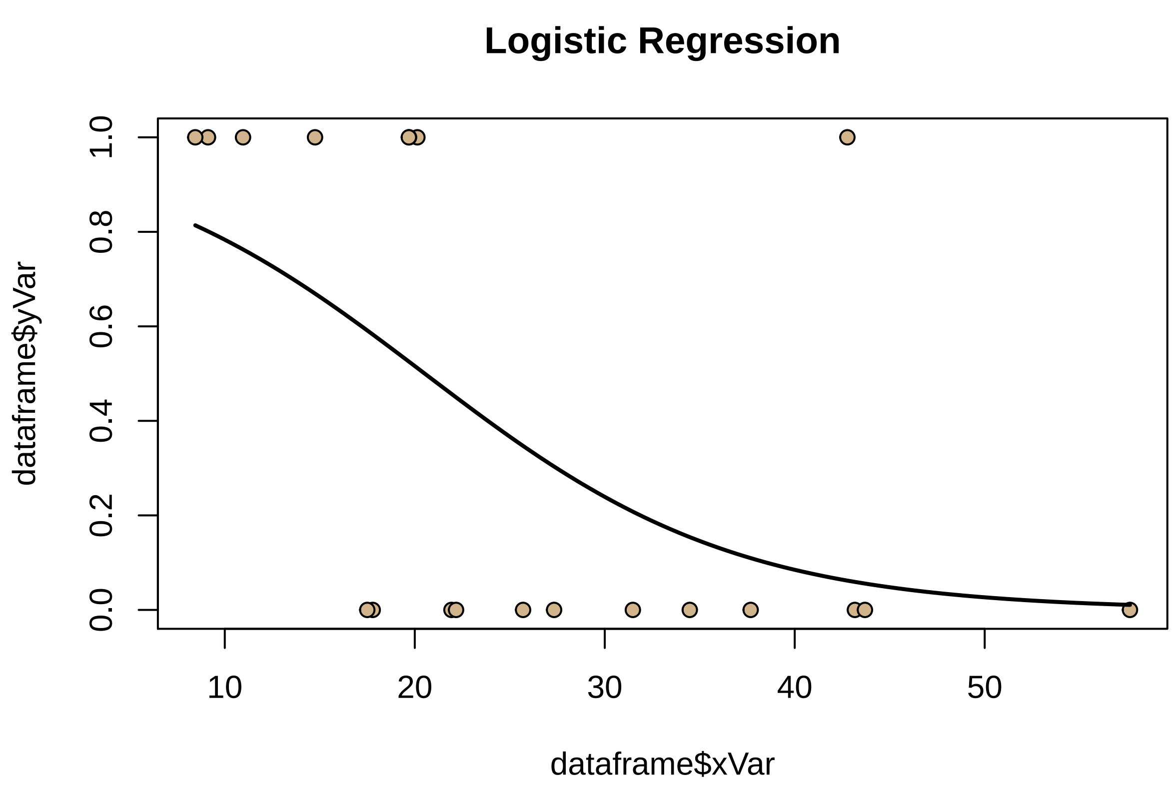
Output: estimate of xvar and p value of xvar

```r
logReg <- function(xVar=rgamma(n=20,shape=5,scale=5),
                   yVar=rbinom(n=20,size=1,p=0.5)){
  logRegMod <- glm(yVar~ xVar,
                   family=binomial(link="logit"))
  logRegOut <- c(xVarEst=summary(logRegMod)$coefficients[2,1],
                 pValue=summary(logRegMod)$coefficients[2,4])
   return(logRegOut)}

logReg()
```

```
##   xVarEst    pValue
## 0.0280732 0.6606187
```

Function for plotting results of logistic regression analysis

```
logRegPlot <- function(xVar=rgamma(n=20,shape=5,scale=5),
                       yVar=rbinom(n=20,size=1,p=0.5),dataframe = data.frame(xVar,yVar)){
lrResults <- glm(yVar ~ xVar, family="binomial"(link="logit"))
LRplot <- plot(dataframe$xVar, y=dataframe$yVar,
                   pch=21,
                   bg="tan",
                   cex=1,
                   main="Logistic Regression")
LRplot1 <- curve(predict(lrResults,
              data.frame(xVar=x),
              type="response"),
      add=TRUE,
      lwd=2)
return(LRplot1)}

logRegPlot()
```




```
## $x
##   [1]  8.445644  8.937664  9.429684  9.921703 10.413723 10.905743 11.397763
##   [8] 11.889783 12.381803 12.873823 13.365843 13.857862 14.349882 14.841902
##  [15] 15.333922 15.825942 16.317962 16.809982 17.302002 17.794021 18.286041
##  [22] 18.778061 19.270081 19.762101 20.254121 20.746141 21.238161 21.730181
##  [29] 22.222200 22.714220 23.206240 23.698260 24.190280 24.682300 25.174320
##  [36] 25.666340 26.158359 26.650379 27.142399 27.634419 28.126439 28.618459
```

```
##  [43] 29.110479 29.602499 30.094518 30.586538 31.078558 31.570578 32.062598
##  [50] 32.554618 33.046638 33.538658 34.030677 34.522697 35.014717 35.506737
##  [57] 35.998757 36.490777 36.982797 37.474817 37.966837 38.458856 38.950876
##  [64] 39.442896 39.934916 40.426936 40.918956 41.410976 41.902996 42.395015
##  [71] 42.887035 43.379055 43.871075 44.363095 44.855115 45.347135 45.839155
##  [78] 46.331174 46.823194 47.315214 47.807234 48.299254 48.791274 49.283294
##  [85] 49.775314 50.267333 50.759353 51.251373 51.743393 52.235413 52.727433
##  [92] 53.219453 53.711473 54.203493 54.695512 55.187532 55.679552 56.171572
##  [99] 56.663592 57.155612 57.647632
##
## $y
##          1          2          3          4          5          6
## 0.81382057 0.80454494 0.79492365 0.78495533 0.77463985 0.76397832
##          7          8          9         10         11         12
## 0.75297325 0.74162859 0.72994978 0.71794384 0.70561939 0.69298672
##         13         14         15         16         17         18
## 0.68005774 0.66684605 0.65336688 0.63963705 0.62567497 0.61150050
##         19         20         21         22         23         24
## 0.59713489 0.58260067 0.56792150 0.55312208 0.53822791 0.52326518
##         25         26         27         28         29         30
## 0.50826058 0.49324108 0.47823377 0.46326565 0.44836343 0.43355337
##         31         32         33         34         35         36
## 0.41886106 0.40431126 0.38992775 0.37573319 0.36174895 0.34799506
##         37         38         39         40         41         42
## 0.33449003 0.32125088 0.30829300 0.29563013 0.28327438 0.27123617
##         43         44         45         46         47         48
## 0.25952431 0.24814596 0.23710673 0.22641072 0.21606055 0.20605751
##         49         50         51         52         53         54
## 0.19640156 0.18709146 0.17812487 0.16949840 0.16120777 0.15324783
##         55         56         57         58         59         60
## 0.14561269 0.13829583 0.13129013 0.12458801 0.11818148 0.11206221
##         61         62         63         64         65         66
## 0.10622162 0.10065093 0.09534123 0.09028351 0.08546875 0.08088793
##         67         68         69         70         71         72
## 0.07653207 0.07239231 0.06845988 0.06472615 0.06118268 0.05782121
##         73         74         75         76         77         78
## 0.05463367 0.05161223 0.04874927 0.04603741 0.04346952 0.04103869
##         79         80         81         82         83         84
## 0.03873830 0.03656193 0.03450345 0.03255694 0.03071675 0.02897746
##         85         86         87         88         89         90
## 0.02733388 0.02578104 0.02431422 0.02292889 0.02162074 0.02038566
##         91         92         93         94         95         96
## 0.01921975 0.01811929 0.01708075 0.01610075 0.01517611 0.01430380
##         97         98         99        100        101
## 0.01348094 0.01270481 0.01197282 0.01128252 0.01063160
```

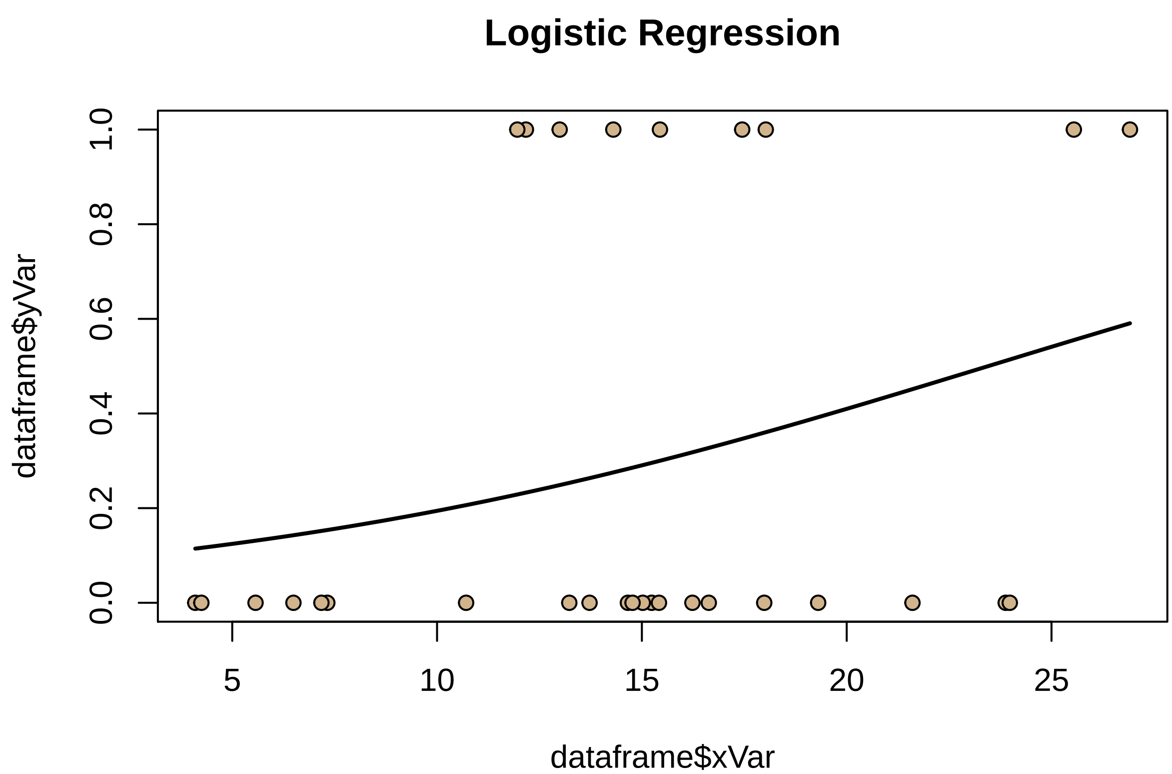
Test logReg and logRegPlot functions with tiny data set

```r
# tiny data set
xVar <- rgamma(n=30,shape=4,scale=4)
yVar <- rbinom(n=30,size=1,p=0.5)
df <- data.frame(xVar,yVar)
```

```r
# test logReg
logReg(xVar=xVar,yVar=yVar)
```

```
##    xVarEst     pValue
## 0.1057268 0.1523052
```

```r
# test logRegPlot
logRegPlot(xVar=xVar,yVar = yVar,dataframe=df)
```




```
## $x
##   [1]  4.096948  4.325136  4.553324  4.781512  5.009700  5.237888  5.466076
##   [8]  5.694264  5.922452  6.150640  6.378827  6.607015  6.835203  7.063391
##  [15]  7.291579  7.519767  7.747955  7.976143  8.204331  8.432519  8.660707
##  [22]  8.888895  9.117083  9.345271  9.573459  9.801647 10.029835 10.258023
##  [29] 10.486211 10.714399 10.942587 11.170775 11.398963 11.627151 11.855339
##  [36] 12.083526 12.311714 12.539902 12.768090 12.996278 13.224466 13.452654
##  [43] 13.680842 13.909030 14.137218 14.365406 14.593594 14.821782 15.049970
##  [50] 15.278158 15.506346 15.734534 15.962722 16.190910 16.419098 16.647286
##  [57] 16.875474 17.103662 17.331850 17.560038 17.788226 18.016413 18.244601
##  [64] 18.472789 18.700977 18.929165 19.157353 19.385541 19.613729 19.841917
##  [71] 20.070105 20.298293 20.526481 20.754669 20.982857 21.211045 21.439233
##  [78] 21.667421 21.895609 22.123797 22.351985 22.580173 22.808361 23.036549
##  [85] 23.264737 23.492925 23.721113 23.949300 24.177488 24.405676 24.633864
##  [92] 24.862052 25.090240 25.318428 25.546616 25.774804 26.002992 26.231180
##  [99] 26.459368 26.687556 26.915744
##
## $y
```

```
##         1         2         3         4         5         6         7
## 0.1144434 0.1169113 0.1194252 0.1219857 0.1245933 0.1272486 0.1299521
##         8         9        10        11        12        13        14
## 0.1327042 0.1355056 0.1383567 0.1412580 0.1442099 0.1472130 0.1502676
##        15        16        17        18        19        20        21
## 0.1533742 0.1565332 0.1597449 0.1630099 0.1663283 0.1697007 0.1731271
##        22        23        24        25        26        27        28
## 0.1766081 0.1801438 0.1837345 0.1873804 0.1910817 0.1948386 0.1986512
##        29        30        31        32        33        34        35
## 0.2025197 0.2064441 0.2104244 0.2144608 0.2185532 0.2227015 0.2269057
##        36        37        38        39        40        41        42
## 0.2311656 0.2354812 0.2398522 0.2442784 0.2487596 0.2532954 0.2578855
##        43        44        45        46        47        48        49
## 0.2625296 0.2672272 0.2719778 0.2767810 0.2816362 0.2865429 0.2915003
##        50        51        52        53        54        55        56
## 0.2965078 0.3015648 0.3066704 0.3118239 0.3170243 0.3222709 0.3275626
##        57        58        59        60        61        62        63
## 0.3328986 0.3382778 0.3436991 0.3491614 0.3546636 0.3602046 0.3657831
##        64        65        66        67        68        69        70
## 0.3713978 0.3770474 0.3827307 0.3884462 0.3941925 0.3999683 0.4057720
##        71        72        73        74        75        76        77
## 0.4116022 0.4174572 0.4233357 0.4292359 0.4351563 0.4410953 0.4470511
##        78        79        80        81        82        83        84
## 0.4530222 0.4590069 0.4650033 0.4710100 0.4770250 0.4830467 0.4890733
##        85        86        87        88        89        90        91
## 0.4951032 0.5011344 0.5071653 0.5131941 0.5192191 0.5252385 0.5312506
##        92        93        94        95        96        97        98
## 0.5372536 0.5432458 0.5492256 0.5551911 0.5611408 0.5670730 0.5729860
##        99       100       101
## 0.5788782 0.5847480 0.5905939
```

Function: ANOVA

input: categorical x variable and continuous y variable

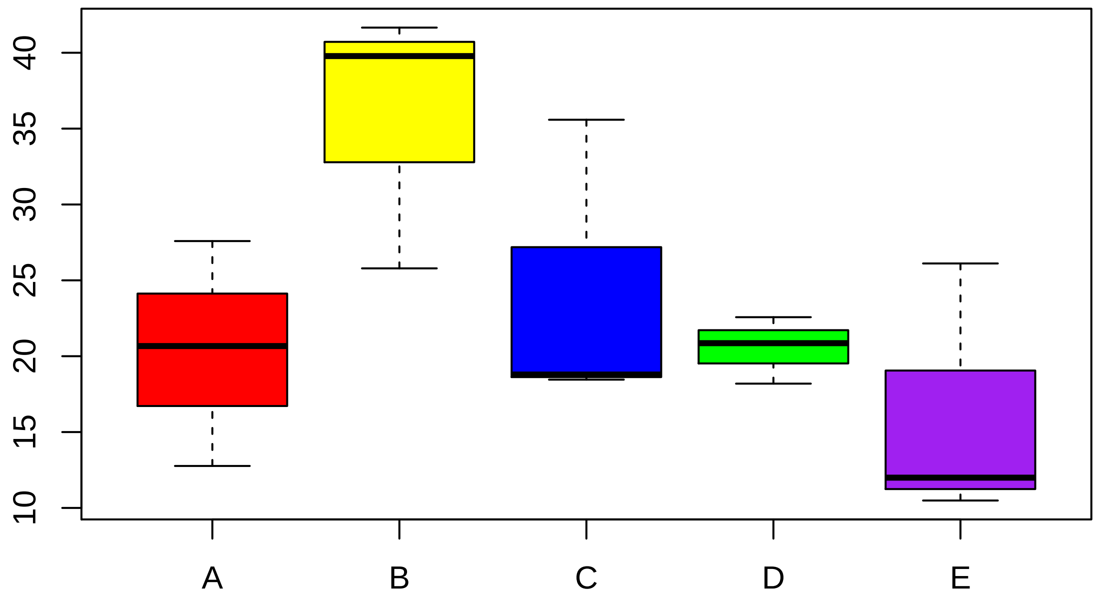
output: p value

```
ANOV <- function(xVar=as.factor(rep(c("A","B","C","D","E"),each=3)),
                    yVar=c(rgamma(10,shape=5,scale=5),rgamma(5,shape=5,scale=10))){
                      df=data.frame(xVar,yVar)
          aovMod <- aov(yVar~xVar,data=df)
          aovOut <- summary(aovMod)[[1]][["Pr(>F)"]][1]
         return(aovOut)}

ANOV()
```

```
## [1] 0.02202106
```

Function for plotting boxplot to represent ANOVA results

```
ANOVplot <- function(xVar=as.factor(rep(c("A","B","C","D","E"),each=3)),yVar=c(rgamma(15,shape=5,scale=5
  df <- data.frame(xVar,yVar)
  aovMod <- aov(yVar~xVar,data=df)
  aovPlot <-  boxplot(yVar~xVar,
                      data=df,
                      col=c("red","yellow","blue","green","purple"),
                    xlab=names(xVar),ylab=names(yVar))
  return(aovPlot)}

ANOVplot()
```




```
## $stats
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 12.76699 25.78971 18.45454 18.19135 10.48889
## [2,] 16.71721 32.78512 18.62314 19.52466 11.24212
## [3,] 20.66742 39.78052 18.79174 20.85796 11.99535
## [4,] 24.12758 40.71855 27.18788 21.71514 19.05439
## [5,] 27.58774 41.65658 35.58403 22.57232 26.11342
##
## $n
## [1] 3 3 3 3 3
##
## $conf
##          [,1]     [,2]     [,3]     [,4]      [,5]
```

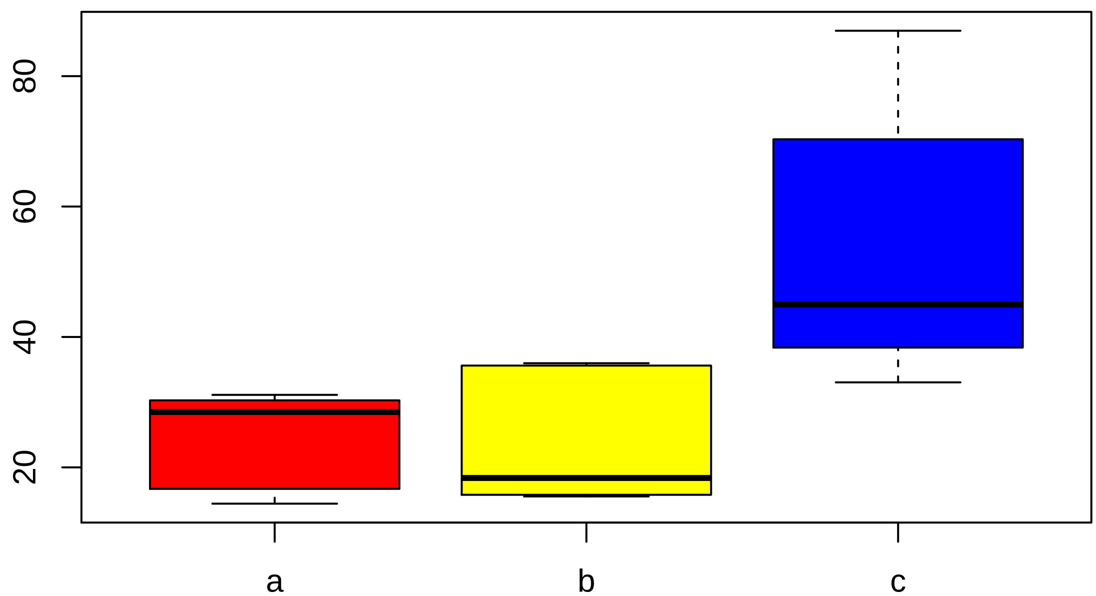
```
## [1,] 13.90758 32.54354 10.97886 18.85977  4.868899
## [2,] 27.42727 47.01751 26.60461 22.85615 19.121809
##
## $out
## numeric(0)
##
## $group
## numeric(0)
##
## $names
## [1] "A" "B" "C" "D" "E"
```

test ANOV and ANOVplot using tiny data set

```
# make tiny data set
xVar1 <- as.factor(rep(c("a","b","c"),each=5))
yVar1 <- c(rgamma(10,shape=5,scale=5),rgamma(5,shape=5,scale=10))

# test analVar() using tiny data set
ANOV(xVar=xVar1,yVar=yVar1)
```

```
## [1] 0.01165875
```

```
# test ANOVplot() using tiny data set
ANOVplot(xVar=xVar1,yVar=yVar1)
```



```
## $stats
##          [,1]     [,2]     [,3]
## [1,] 14.43559 15.54148 33.03642
## [2,] 16.70259 15.79013 38.36970
## [3,] 28.48510 18.37590 44.96497
## [4,] 30.27788 35.61230 70.32108
## [5,] 31.11918 35.97518 86.95887
##
## $n
## [1] 5 5 5
##
## $conf
##          [,1]      [,2]     [,3]
## [1,] 18.89283  4.369605 22.38821
## [2,] 38.07737 32.382193 67.54173
##
## $out
## numeric(0)
##
## $group
## numeric(0)
##
## $names
## [1] "a" "b" "c"
```

Function: Contingency Table

Input: discrete independent variable and discrete dependent variable

Output:

```
contTable <- function(x=c(22,40,60),y=c(40,80,45),datamatrix=rbind(x,y)){
  rownames(datamatrix)=c("Cold","Warm")
  colnames(datamatrix)=c("Species1","Species2","Species3")
  contTableMod <- chisq.test(datamatrix)
  contTableOut <- print(chisq.test(datamatrix)[3])
          return(contTableOut)}

contTable()
```

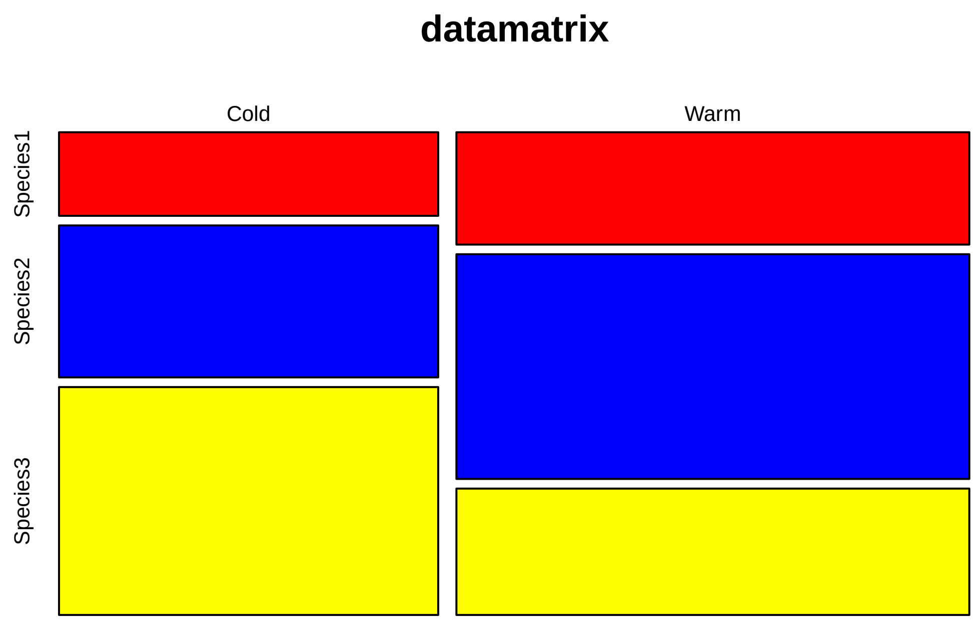
```
## $p.value
## [1] 0.0006799671
```

```
## $p.value
## [1] 0.0006799671
```

Function for plotting mosaic plot of data complementary to contingency table results

```
contPlot <- function(x=c(22,40,60),y=c(40,80,45),datamatrix=rbind(x,y)){
  rownames(datamatrix)=c("Cold","Warm")
```

```
  colnames(datamatrix)=c("Species1","Species2","Species3")
  mplot <- mosaicplot(x=datamatrix,col=c("red","blue","yellow"),shade=F)
  return(mplot)}

contPlot()
```




```
## NULL
```

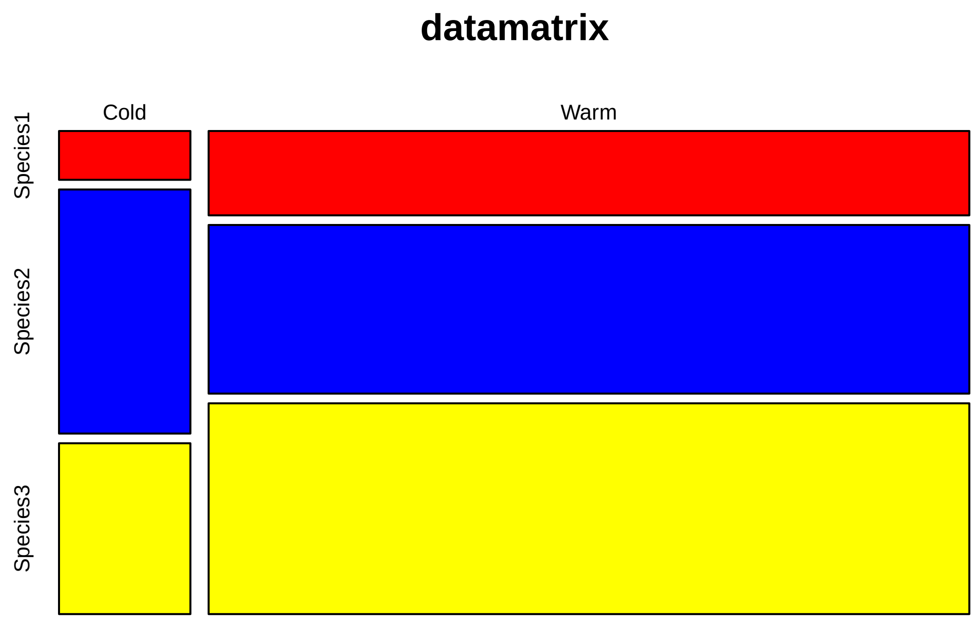
Test contingency table functions using tiny data set

```
x <- c(2,10,7)
y <- c(20,40,50)
dm <- rbind(x,y)

# test contTable()
contTable(x,y,dm)
```

```
## $p.value
## [1] 0.3800403
```

```
## $p.value
## [1] 0.3800403
```

```
# test contPlot()
contPlot(x,y,dm)
```




```
## NULL
```