Assignment 1

Benjamin Moran

03/09/2017

Question 1

i)

We are given the delta method:

$$Var[g(x)] = (\frac{d}{dx}g(x))^2 Var(x)$$

So, for $g(x) = \sqrt(\theta), x = \theta$ we get:

$$\frac{d}{d\theta}\sqrt(\theta) = \frac{1}{2\sqrt{\theta}} \implies (\frac{d}{d\theta}\sqrt(\theta))^2 = \frac{1}{4\theta}$$

We are given in the lecture notes that $Var(\theta) = \theta^2 \left(1/n_{11} + 1/n_{12} + 1/n_{21} + 1/n_{22}\right)$, therefore:

$$\begin{aligned} Var(\sqrt{\theta}) &= \frac{1}{4\theta} * \theta^2 \left(1/n_{11} + 1/n_{12} + 1/n_{21} + 1/n_{22}\right) \\ &= \frac{\theta}{4} * \left(1/n_{11} + 1/n_{12} + 1/n_{21} + 1/n_{22}\right) \end{aligned}$$

ii)

First, let's expand the expression $\left[(1 - g(\theta))(1 + g(\theta))\right]^2$, which will become useful later. We get:

$$\begin{aligned} \left[(1 - g(\theta))(1 + g(\theta))\right]^2 &= \left[\left(1 - \frac{\theta^b - 1}{\theta^b + 1}\right)\left(1 + \frac{\theta^b - 1}{\theta^b + 1}\right)\right]^2 \\ &= \frac{16\theta^{2b}}{(\theta^b + 1)^4} \end{aligned}$$

Now, we will use the delta method to find an expression for $Var(g(\theta))$:

$$\begin{aligned} Var(g(\theta)) &= \left(\frac{d}{d\theta}g(\theta)\right)^2 Var(\theta) \\ &= \left(\frac{d}{d\theta}\left(\frac{\theta^b - 1}{\theta^b + 1}\right)\right)^2 * \theta^2 \left(n_{11} + n_{12} + n_{21} + n_{22}\right)^{-1} \\ &= \frac{4b^2\theta^{2b-2}}{(\theta^b + 1)^4} * \theta^2 \left(n_{11} + n_{12} + n_{21} + n_{22}\right)^{-1} \\ &= \frac{4b^2\theta^{2b}}{(\theta^b + 1)^4} \left(n_{11} + n_{12} + n_{21} + n_{22}\right)^{-1} \end{aligned}$$

Returning to the question, if we substitute the expression we found for $\left[(1 - g(\theta))(1 + g(\theta))\right]^2$ into the required expression for $Var(g(\theta))$, we get:

$$
\begin{aligned}
Var(g(\theta)) &= \frac{b^2}{4}\left[(1-g(\theta))(1+g(\theta))\right]^2 (n_{11}+n_{12}+n_{21}+n_{22})^{-1} \\
&= \frac{b^2}{4}\frac{16\theta^{2b}}{(\theta^b+1)^4}(n_{11}+n_{12}+n_{21}+n_{22})^{-1} \\
&= \frac{4b^2\theta^{2b}}{(\theta^b+1)^4}(n_{11}+n_{12}+n_{21}+n_{22})^{-1}
\end{aligned}
$$

as required.

iii)

I've added the code to the `edwards.odds.exe` from the lectures, and then evaluated the code for Q, Y, H and J.

```
asbestos.dat <- matrix(c(522,53,203,339), nrow = 2)
dimnames(asbestos.dat) = list(c("0-19","20+"),c("No","Yes"))
oddsratio.exe <- function (N, acc) {
  round((N[1,1]*N[2,2])/(N[1,2]*N[2,1]), digits = acc)
}
oddsratio.exe(asbestos.dat, acc = 3)
```

```
## [1] 16.447
```

```
edwards.odds.exe <- function(N, b, acc){
  OR = oddsratio.exe(N, acc)
  EO = round((OR^b - 1)/(OR^b+1), digits = acc)
  dtheta2 = ((4*(b^2)*OR^(2*b))/(OR^b + 1)^(4))
  VarO = dtheta2*(N[1,1] + N[1,2] + N[2,1] + N[2,2])^(-1)
  return(c("OR Value" = EO, "OR Variance" = VarO))
}

# Yule's Q
edwards.odds.exe(asbestos.dat, b = 1, acc = 4)
```

```
##    OR Value OR Variance
## 8.85400e-01 1.04539e-05
```

```
# Yule's Y
edwards.odds.exe(asbestos.dat, b = 0.5, acc = 4)
```

```
##     OR Value  OR Variance
## 6.044000e-01 2.254105e-05
```

```
# Digby's H
edwards.odds.exe(asbestos.dat, b = 3/4, acc = 4)
```

```
##    OR Value OR Variance
## 7.81800e-01 1.90252e-05
```

```
# Edward's J
edwards.odds.exe(asbestos.dat, b = pi/4, acc = 4)
```

```
##     OR Value  OR Variance
## 8.004000e-01 1.783472e-05
```

Question 2

i)

The Mantel-Haenszel odds ratio

This estimate is given in the lecture notes as:

$$\widehat{\theta}_{MH} = \frac{\sum_g n_{11g} n_{22g}/n_g}{\sum_g n_{12g} n_{21g}/n_g} = \frac{\sum_g R_g}{\sum_g S_g} = \frac{R}{S}, \quad \forall g \in G$$

and gives the weighted (the n_g term) mean (the $\sum_g$) of G odds ratios. The weight can also be thought of as the inverse variance of each G stratum under independence. This ratio has no issue with zero cell counts (unlike some other ratios based on the OR - like Woolf's). As a result, it can give intelligible results when faced with a large proportion of the G strata having few/zero observations.

The Woolf odds ratio

This is based off of the log of the odds ratio and weights each G table by it's sampling error. By taking the inverse of this sampling error it gives more weight to those strata with the smallest variance. We have:

$$\widehat{\theta}_{Woolf} = exp\left\{\frac{\sum_g w_g * (log(OR_g))}{\sum_g w_g}\right\} w_g = \left(\frac{1}{n_{11g}} + \frac{1}{n_{12g}} + \frac{1}{n_{21g}} + \frac{1}{n_{22g}}\right)^{-1} OR_g = \frac{n_{11g} n_{22g}}{n_{12g} n_{21g}}$$

The Woolf ratio will have issues with zero cell counts (due to $log(0)$ being undefined) and we will therefore use either a Haldane of Yates correction (e.g. adding some suitably small amount - e.g. 0.5 - to each of the zeros), in order to account for this. However, this will introduce bias that will have a greater effect on the result when the cell counts are smaller.

ii)

The Cochran-Mantel-Haenszel Test is given in the lecture notes as:

$$\widehat{\theta}_{CMH} = \frac{\sum_g (n_{11g} - E(n_{11g}))}{\sum_g Var(n_{11g})}, \quad \forall g \in G E(n_{11g}) = \frac{n_{1*g} n_{*1g}}{n_{**g}} Var(n_{11g}) = \frac{n_{1*g} n_{2*g} n_{*1g} n_{*2g}}{(n_g)^2 (n_g - 1)}$$

The above formulation takes advantabe of the binomial nature of the variables in a 2×2 contingency table to only consider the counts in a single cell of each strata, although it can be rewritten to look more like the other tests listed here. By incorporating the *conditional permutation variance* CMH statistic ensures that the test returns consisistent results even when used on data where many strata have small counts or contain zeroes. The Mantel-Haenzsel statistic does not incorporate this variance term and thus would return compromised results for the same data. The CMH shares a similar insight as the Woolf statistic regarding a consideration of the within strata variance, but does not go out of its way to give more weight to strata with lower variance.

iii)

First, let's import the data (availabel from Agresti's website)

```
# Data comes from Agresti's website.
table.6.9 <- data.frame(scan(file="Data/Agresti_6_9.txt",
                             what=list(Center="",
                                       Treatment="",
                                       Response="",
                                       Freq=0)))
levels(table.6.9$Treatment) <- c("Drug","Control")
levels(table.6.9$Response) <- c("Success","Failure")

array.6.9 <- xtabs(Freq~Treatment+Response+Center, data=table.6.9)
knitr::kable(table.6.9)
```

Center	Treatment	Response	Freq
a	Drug	Success	11
a	Drug	Failure	25
a	Control	Success	10
a	Control	Failure	27
b	Drug	Success	16
b	Drug	Failure	4
b	Control	Success	22
b	Control	Failure	10
c	Drug	Success	14
c	Drug	Failure	5
c	Control	Success	7
c	Control	Failure	12
d	Drug	Success	2
d	Drug	Failure	14
d	Control	Success	1
d	Control	Failure	16
e	Drug	Success	6
e	Drug	Failure	11
e	Control	Success	0
e	Control	Failure	12
f	Drug	Success	1
f	Drug	Failure	10
f	Control	Success	0
f	Control	Failure	10
g	Drug	Success	1
g	Drug	Failure	4
g	Control	Success	1
g	Control	Failure	8
h	Drug	Success	4
h	Drug	Failure	2
h	Control	Success	6
h	Control	Failure	1

Now, a function to calculate the Odds Ratio (using the Haldane correction for zero-valued cells).

```
odds.ratio <- function(ctable,correct = TRUE){
  if(correct){
    if(any(ctable==0)) # Haldane Correction
      ctable <- ctable + 0.5
  }
```

```
  or  = ctable[1,1]*ctable[2,2]/(ctable[2,1]*ctable[1,2])
}
apply(array.6.9 , 3, odds.ratio)
```

```
##          a          b          c          d          e          f
##  1.1880000  1.8181818  4.8000000  2.2857143 14.1304348  3.0000000
##          g          h
##  2.0000000  0.3333333
```

Next, a funciton to calculated the Hantel-Maenzsel pooled odds ratio, this time using the Yates continuity correction.

```
hantmaenz <- function(ctable, correct = TRUE) {

  sumctmd = apply(ctable, c(1, 3), sum) # sum of main diag for each centre
  sumctod = apply(ctable, c(2, 3), sum) # column sums for each centre
  ctsum   = apply(ctable, 3, sum) # sum of all elements for each centre
  or      = sum(ctable[1, 1, ] - sumctmd[1, ] * sumctod[1, ] /ctsum)
  yates   = ifelse((correct & (abs(or) >= 1/2)), 1/2, 0)
  stat    = ((abs(or)-yates)^2/sum(apply(rbind(sumctmd, sumctod),2,prod)/(ctsum^2*(ctsum - 1))))
  pval    = 1 - pchisq(stat,1)
  print(list("Statistic" = stat, "p value" = pval))
}

hantmaenz(array.6.9)
```

```
## $Statistic
## [1] 5.671647
##
## $`p value`
## [1] 0.01724126
```

Next, the Woolf with the Haldane correction (because it's easier and I only wanted to include Yates' once).

```
woolf <- function(ctable, correct = TRUE) {
  if(correct){
    if(any(ctable==0)) # Haldane Correction
      ctable <- ctable + 0.5
  }
  or   = apply(ctable, 3, function(ctable) (ctable[1,1]*ctable[2,2])/(ctable[1,2]*ctable[2,1]))
  w    =  apply(ctable, 3, function(ctable) 1 / sum(1 / ctable))
  stat = (sum(w * (log(or) - weighted.mean(log(or), w)) ^ 2))
  pval = 1 - pchisq(sum(w * (log(or) - weighted.mean(log(or), w)) ^ 2), 1)
  print(list("Statistic" = stat, "p value" = pval))
}

woolf(array.6.9)
```

```
## $Statistic
## [1] 5.817998
##
## $`p value`
## [1] 0.01586299
```

Lastly, the Cochran-Mantel-Haenzsel test, with the Haldane correction.

```r
cmh <- function(ctable, correct = TRUE) {
  if(correct){
    if(any(ctable==0)) # Haldane Correction
      ctable <- ctable + 0.5
  }
  k       = dim(ctable)[3]
  n1      = apply(ctable, 3, function(ctable) ctable[1,1])
  # the top left cell of each table
  en      = apply(ctable, 3, function(ctable) rowSums(ctable)[1]*colSums(ctable)[1])
  # first row sum * first col sum
  enwgt   = apply(ctable, 3, function(ctable) sum(ctable))
  # sum of all elements
  varn    = apply(ctable, 3, function(ctable) prod(rowSums(ctable))*prod(colSums(ctable)))
  #product of rowsums and colsums
  varnwgt = apply(ctable, 3, function(ctable) sum(ctable)^2 * (sum(ctable)-1))
  # variance denominator
  stat    = (sum((n1 - (en/enwgt))/(varn/varnwgt)))
  pval    = 1 - pchisq(sum((n1 - (en/enwgt))/(varn/varnwgt)), 1)
  print(list("Statistic" = stat, "p value" = pval))
}

cmh(array.6.9)
```

```
## $Statistic
## [1] 5.212985
##
## $`p value`
## [1] 0.02241881
```

Question 3

i)

First, create the dataset.

```r
drug.dat <- as.table(rbind(c(5,1,10,8,6),
                           c(5,3,3,8,12),
                           c(10,6,12,3,0),
                           c(7,12,8,1,1)))
dimnames(drug.dat) <- list(Drug = c("A","B","C","D"),
                       Effect = c("Poor","Fair","Good","VeryGood","Excellent"))
```

We can calculate the χ^2 statistic for the data using `chisq.test`.

```r
x2   = chisq.test(drug.dat)
stat = x2$statistic
dof  = x2$parameter
pval = x2$p.value
x2
```

```
##
##  Pearson's Chi-squared test
##
## data:  drug.dat
```

```
## X-squared = 47.072, df = 12, p-value = 4.53e-06
```

```
x2$observed
```

```
##     Effect
## Drug Poor Fair Good VeryGood Excellent
##    A    5    1   10        8         6
##    B    5    3    3        8        12
##    C   10    6   12        3         0
##    D    7   12    8        1         1
```

```
x2$expected
```

```
##     Effect
## Drug     Poor     Fair     Good VeryGood Excellent
##    A 6.694215 5.454545 8.181818 4.958678  4.710744
##    B 6.917355 5.636364 8.454545 5.123967  4.867769
##    C 6.917355 5.636364 8.454545 5.123967  4.867769
##    D 6.471074 5.272727 7.909091 4.793388  4.553719
```

We can get an inkling as to the significance of the result by observing how uniformly the rows/columns compare to each other in the expected table versus the lack of such similarity in the observed table. The results confirm this: we can see that the χ^2 statistic is 47.072; the degrees of freedom $(nrows - 1) * (ncols - 1)$ is 12; and the p.value reported by the test is 4.53e-06 which is < 0.01, suggesting that we have strong evidence to reject H_0 - that the observed cell counts would not differ from what we would expect under independence.

ii)

We can simulate p.values by using the following function. Note: setting plot = TRUE (the default) will plot a histogram of the simulated values - along with the theoretical χ^2 distribution and the calculated χ^2 statistic for the supplied data.

Then, the function for the simulations.

```
monte.carlo.p.exe <- function(ctable,
                              B = 10000,
                              plot = TRUE){

  # ctable: the contingency table that you wish to simulate p.values for.
  # B: the number of simulations to run.
  # plot: set to TRUE (default) if you wish to plot a histogram of the simulations

  options(digits = 7, warn = FALSE)
  dof  = (nrow(ctable)-1)*(ncol(ctable)-1) # degrees of freedom for supplied data.
  rr   = rowSums(ctable) # the row sums of the supplied data
  cc   = colSums(ctable) # the column sums of the supplied data
  x2   = as.numeric(chisq.test(ctable)$statistic) # the chi-squared statistic associated
                                                  # with the supplied data

  pvec = vector(mode = "numeric", length = B) # a blank vector for simulations

  # the following for loop will generate B contingency tables using the marginal sums of
  # the supplied data and calculate a chi-squared statistic for each simulated table.
  # It then stores this statistic in the vectore pvec.

  for(i in 1:B) {
```

```
    pvec[i] = chisq.test(r2dtable(1,rr,cc)[[1]])$statistic
  }

  # the simulated p.value can be calculated either this way or
  # by length(pvec[pvec > x2])/B

  MC.p.value = sum(pvec >= x2)/B
  print(c("Simulated p-value" = MC.p.value))
  if(plot){
    x <- rchisq(B, dof) # the theoretical chi-squared density
    xlims = c(0, x2*1.01) # expanding the limits to fit the calculated chi-square.
    hist(pvec,
         prob   = TRUE, # set prob = TRUE to compare with the theoretical density.
         breaks = 40,
         xlim   = xlims,
         xlab   = NULL,
         main   = paste("Monte-Carlo p-value for B =",B,
                        "\n p.value = ",MC.p.value,
                        ";","X^2 =",round(x2,3)),
         sub    = "Sims (histogram); theoretical (green), X^2 (red)")
    curve(dchisq(x, df=dof), col='green', add=TRUE) # add the theoretical density
    abline(v = x2, col = "red") # add the actual chi-square statistic.
  }
}
```

Now, we run the function for $B = 10, 100, 1000, 10000$.

```
monte.carlo.p.exe(drug.dat, B = 10)
```

```
## Simulated p-value
##                 0
```

```
monte.carlo.p.exe(drug.dat, B = 100)
```

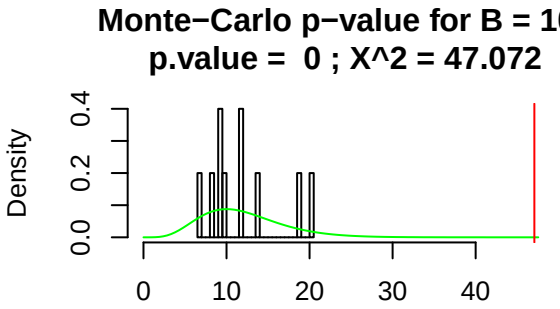
```
## Simulated p-value
##                 0
```

```
monte.carlo.p.exe(drug.dat, B = 1000)
```

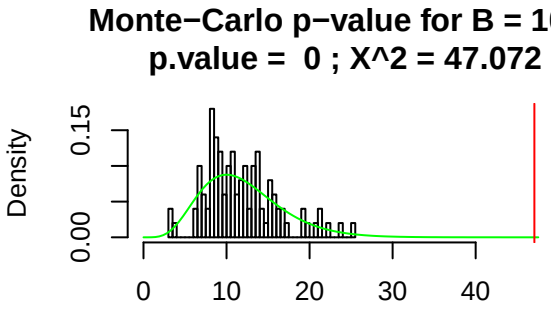
```
## Simulated p-value
##                 0
```

```
monte.carlo.p.exe(drug.dat, B = 10000)
```

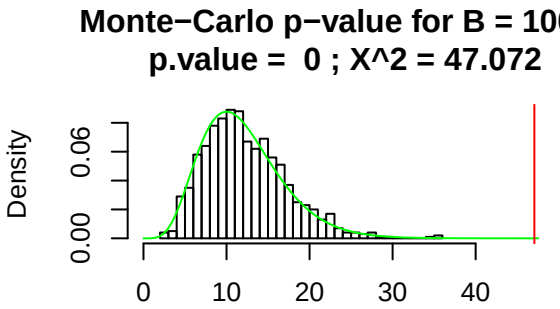
```
## Simulated p-value
##                 0
```



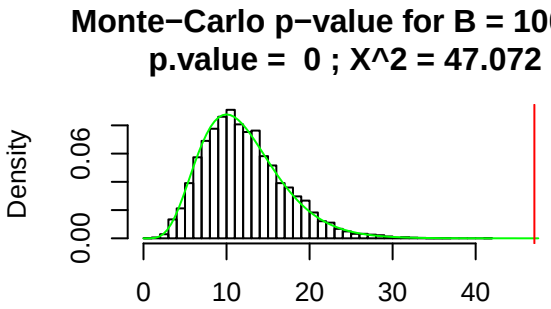
Sims (histogram); theoretical (green), X^2 (red)



Sims (histogram); theoretical (green), X^2 (red)



Sims (histogram); theoretical (green), X^2 (red)



Sims (histogram); theoretical (green), X^2 (red)

iii)

The following will produce estimates of the Cressie-Read family of statistics using 10000 simulation and a Haldane correction.

```
divergeCR.exe <- function(ctable,
                          correct = TRUE,
                          correct.val = 0.5,
                          B = 10000,
                          plot = TRUE){

  # ctable: a contingency table
  # correct: if 0 values present, should they be replaced to avoid errors? Defaults to TRUE.
  # correct.val: if correct = TRUE, what value should be used to replace 0? Defaults to 0.5.

  options(digits = 4, warn = FALSE)
  rows = nrow(ctable)
  cols = ncol(ctable)
  ctsum = sum(ctable)

  if(correct){
    if(any(ctable) == 0){
      ctable[m,n] = correct.val
    }
  }
```

```
ctsum = sum(ctable)
ptable = prop.table(ctable)
rrc = rowSums(ctable)
ccc = colSums(ctable)
rr = rowSums(ptable)
cc = colSums(ptable)

# Pearson
pearsonmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    pearsonmatrix[m,n] = ((ptable[m,n] - rr[m]*cc[n])^2)/(rr[m]*cc[n])
  }
}

    # Monte-Carlo Sims
    pvecP = vector(mode = "numeric", length = B)

    for(i in 1:B){
      matsP = matrix(NA, nrow = rows, ncol = cols)
      pmatsP = r2dtable(1,rrc,ccc)[[1]]/ctsum
      for(m in 1:rows){
        for(n in 1:cols){
          matsP[m,n] = ((pmatsP[m,n] - rr[m]*cc[n])^2)/(rr[m]*cc[n])
        }
      }
      pvecP[i] = (ctsum)*sum(matsP)
    }

    # Calculate Score and p value

    pearsonstat = ctsum*sum(pearsonmatrix)
    p.pearsonstat = sum(pvecP >= pearsonstat)/B

# Log-Likelihood
llmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    llmatrix[m,n] = ptable[m,n]*log(ptable[m,n]/(rr[m]*cc[n]))
  }
}
    # Monte-Carlo Sims
    pvecL = vector(mode = "numeric", length = B)

    for(i in 1:B){
      matsL = matrix(NA, nrow = rows, ncol = cols)
      pmatsL = r2dtable(1,rrc,ccc)[[1]]/ctsum
      for(m in 1:rows){
        for(n in 1:cols){
          matsL[m,n] = pmatsL[m,n]*log(pmatsL[m,n]/(rr[m]*cc[n]))
        }
      }
      pvecL[i] = 2*ctsum*sum(matsL)
```

```
      if(is.na(sum(matsL))){
        pvecL[i] = pvecL[i-1]
      }
    }

    llstat = 2*ctsum*sum(llmatrix)
    p.llstat = sum(pvecL >= llstat)/B

# Freeman-Tukey
ftmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    ftmatrix[m,n] = (sqrt(ptable[m,n]) - sqrt(rr[m]*cc[n]))^(2)
  }
}

    # Monte-Carlo Sims
    pvecF = vector(mode = "numeric", length = B)

    for(i in 1:B){
      matsF = matrix(NA, nrow = rows, ncol = cols)
      pmatsF = r2dtable(1,rrc,ccc)[[1]]/ctsum
      for(m in 1:rows){
        for(n in 1:cols){
          matsF[m,n] = (sqrt(pmatsF[m,n]) - sqrt(rr[m]*cc[n]))^(2)
        }
      }
      pvecF[i] = 4*(ctsum)*sum(matsF)
    }

    ftstat = 4*ctsum*sum(ftmatrix)
    p.ftstat = sum(pvecF >= ftstat)/B

# Neyman's Modified
nmmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    nmmatrix[m,n] = ((ptable[m,n] - (rr[m]*cc[n]))^(2))/ptable[m,n]
  }
}

    # Monte-Carlo Sims
    pvecN = vector(mode = "numeric", length = B)

    for(i in 1:B){
      matsN = matrix(NA, nrow = rows, ncol = cols)
      pmatsN = r2dtable(1,rrc,ccc)[[1]]/ctsum
      for(m in 1:rows){
        for(n in 1:cols){
          matsN[m,n] = ((pmatsN[m,n] - (rr[m]*cc[n]))^(2))/pmatsN[m,n]
        }
      }
      pvecN[i] = (ctsum)*sum(matsN)
```

```
    }

    nmstat = ctsum*sum(nmmatrix)
    p.nmstat = sum(pvecN >= nmstat)/B


# Cressie-Read
crmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    crmatrix[m,n] = ptable[m,n]*((ptable[m,n]/(rr[m]*cc[n]))^(0.67) -1)
  }
}

    # Monte-Carlo Sims
    pvecCR = vector(mode = "numeric", length = B)

    for(i in 1:B){
      matsCR = matrix(NA, nrow = rows, ncol = cols)
      pmatsCR = r2dtable(1,rrc,ccc)[[1]]/ctsum
      for(m in 1:rows){
        for(n in 1:cols){
          matsCR[m,n] = pmatsCR[m,n]*((pmatsCR[m,n]/(rr[m]*cc[n]))^(0.67) -1)
        }
      }
      pvecCR[i] = (2 * ctsum/((0.67)*(0.67 + 1)))*sum(matsCR)
    }
    crstat = (2 * ctsum/((0.67)*(0.67 + 1)))*sum(crmatrix)
    p.crstat = sum(pvecCR >= crstat)/B

# Modified Log-Likelihood
mllmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    mllmatrix[m,n] = (rr[m]*cc[n])*log((rr[m]*cc[n])/ptable[m,n])
  }
}
    # Monte-Carlo Sims
    pvecML = vector(mode = "numeric", length = B)

    for(i in 1:B){
      matsML = matrix(NA, nrow = rows, ncol = cols)
      pmatsML = r2dtable(1,rrc,ccc)[[1]]/ctsum
      for(m in 1:rows){
        for(n in 1:cols){
          matsML[m,n] = (rr[m]*cc[n])*log((rr[m]*cc[n])/pmatsML[m,n])
        }
      }
      pvecML[i] = 2*ctsum*sum(matsML)
      if(is.na(sum(matsML))){
        pvecML[i] = pvecML[i-1]
      }
    }
```

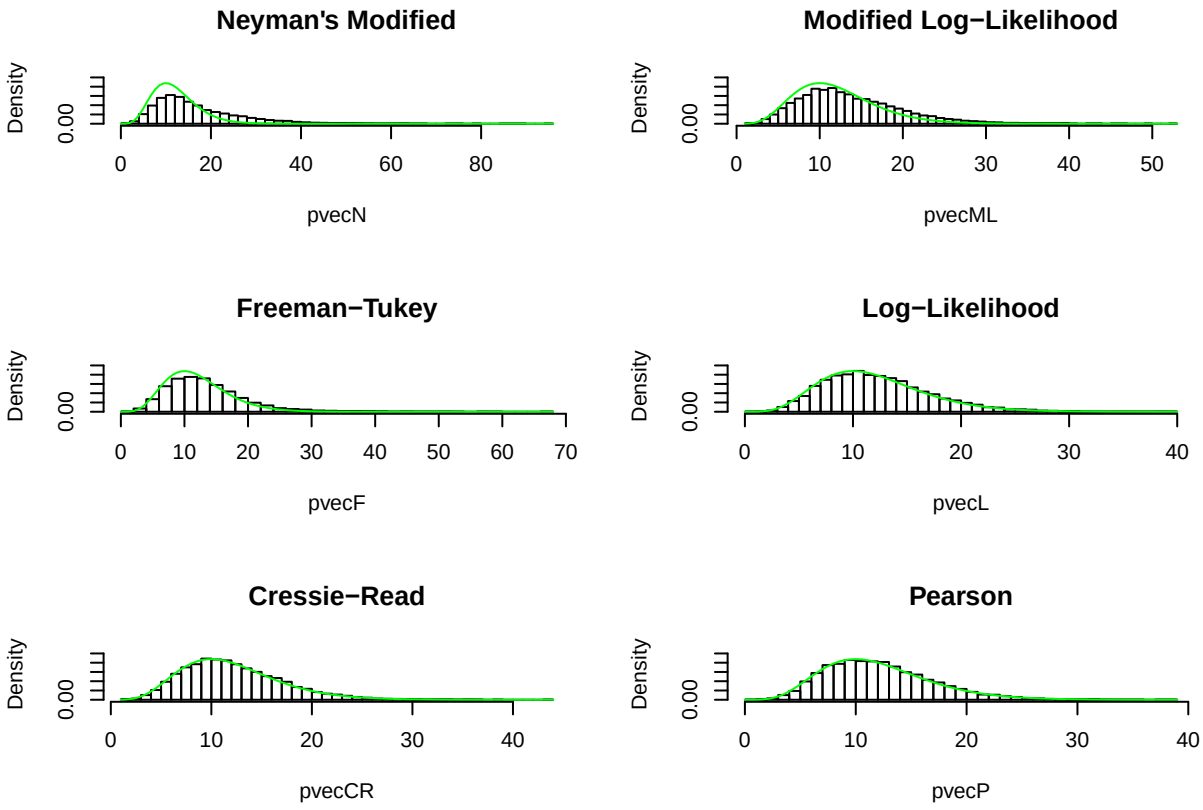
```
      mllstat = 2*ctsum*sum(mllmatrix)
      p.mllstat = sum(pvecML >= mllstat)/B

  # Output
  out = matrix(c(-2,-1,-1/2,0,0.67,1,
                 nmstat,  mllstat,  ftstat,  llstat,  crstat,  pearsonstat,
                 p.nmstat,p.mllstat,p.ftstat,p.llstat,p.crstat,p.pearsonstat),
               ncol = 3)

  dimnames(out) = list(c("Neyman's Modified","Modified Log-Likelihood",
                         "Freeman-Tukey","Log-Likelihood",
                         "Cressie-Read","Pearson"),
                       c("Lambda","Stat","p-value"))
  if(plot){
    par(mfrow = c(3,2))
    x <- rchisq(B, 12)
    ylims = c(0, max(dchisq(x, df=12))*1.25)
    hist(pvecN, prob = TRUE, breaks = 40, main = "Neyman's Modified", ylim = ylims)
    curve( dchisq(x, df=12), col='green', add=TRUE)
    hist(pvecML, prob = TRUE, breaks = 40, main = "Modified Log-Likelihood", ylim = ylims)
    curve(dchisq(x, df=12), col='green', add=TRUE)
    hist(pvecF, prob = TRUE, breaks = 40, main = "Freeman-Tukey", ylim = ylims)
    curve(dchisq(x, df=12), col='green', add=TRUE, ylim = ylims)
    hist(pvecL, prob = TRUE, breaks = 40, main = "Log-Likelihood", ylim = ylims)
    curve(dchisq(x, df=12), col='green', add=TRUE, ylim = ylims)
    hist(pvecCR, prob = TRUE, breaks = 40, main = "Cressie-Read", ylim = ylims)
    curve(dchisq(x, df=12), col='green', add=TRUE, ylim = ylims)
    hist(pvecP, prob = TRUE, breaks = 40, main = "Pearson", ylim = ylims)
    curve(dchisq(x, df=12), col='green', add=TRUE, ylim = ylims)
    par(mfrow = c(1,1))
  }
  out
}

divergeCR.exe(drug.dat)
```

```
## Warning in any(ctable): coercing argument of type 'double' to logical
```




```
##                          Lambda  Stat p-value
## Neyman's Modified         -2.00   Inf  0.0221
## Modified Log-Likelihood   -1.00   Inf  0.0206
## Freeman-Tukey             -0.50 64.45  0.0001
## Log-Likelihood             0.00   NaN      NA
## Cressie-Read               0.67 47.34  0.0000
## Pearson                    1.00 47.07  0.0000
```

From the above chart we can tell that every statistic in the family is approximately χ^2 distributed with a single degree of freedom. The accuracy of the approximation improves for values around the range specified by Cressie and Read, which we can tell by how much better the histograms trace the theoretical density for the Log-Likelihood, Cressie-Read and Pearson χ^2 statistics. The others are relatively poor approximations, although the overall shape of the data is obviously χ^2.

iv)

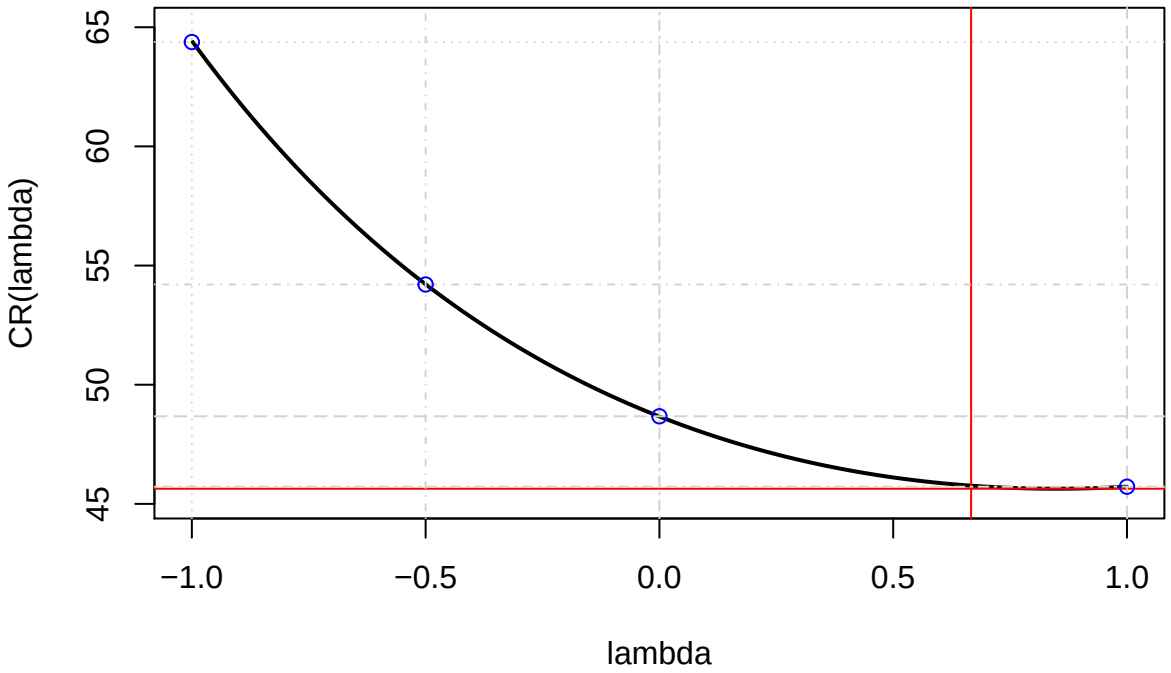
```
plotCR.exe <- function(ctable, resolution = 1000, correct = TRUE, correct.val = 0.5){
  options(digits = 7, warn = FALSE)
  x = seq(from = -1, to = 1, length.out = resolution)
  vcases = c(-1,-0.5,0,1)
  hcases = vector(mode = "numeric", length = length(vcases))
  for(i in 1:length(vcases)){
    sqvals = abs(x-vcases[i])^2
    val = min(abs(x + vcases[i])^2)
    hcases[i] = which(sqvals == val)
  }
  crstat = vector(mode = "numeric", length = resolution)
```

```
  crstat = cbind("lambda" = x,"CR(lambda)" = crstat)
  rows = nrow(ctable)
  cols = ncol(ctable)

  if(correct){
    for(m in 1:rows){
      for(n in 1:cols){
        if(ctable[m,n] == 0){
          ctable[m,n] = correct.val
        }
      }
    }
  }

  ctsum = sum(ctable)
  ptable = prop.table(ctable)
  rr = rowSums(ptable)
  cc = colSums(ptable)

  for(i in 1:resolution){
    lambda = crstat[i,1]
    crmatrix = matrix(NA, nrow = rows, ncol = cols)
    for(m in 1:rows){
      for(n in 1:cols){
        crmatrix[m,n] = ptable[m,n]*((ptable[m,n]/(rr[m]*cc[n]))^(lambda) - 1)
      }
    }
    crstat[i,2] = (2 * ctsum/(lambda*(lambda + 1)))*sum(crmatrix)
  }
  plot(crstat, type = "l", lwd = 2,
       ylim = c(min(crstat[,2]*0.99),
                max(crstat[2:nrow(crstat),2])*1.01))
  abline(v = 2/3,
         col = "red")
  abline(h = min(crstat[,2]),
         col = "red")
  abline(h = c(crstat[2,2],crstat[hcases[2:length(hcases)],2]),
         col = "lightgray", lty = c(3,4,5,6))
  abline(v = vcases,
         col = "lightgray", lty = c(3,4,6,5))
  points(vcases,
         c(crstat[2,2],crstat[hcases[2:length(hcases)],2]),
         col = "blue")
}
plotCR.exe(drug.dat)
```



Here we see the line (black) representing the value of the CR statistic for values between $[-1, 1]$ - with some trickery involving an approximation of the limit of the statistic as it approaches -1 - and the specific points (blue) corresponding to the particular statistics being study. The red vertical line represents the optimal value for λ specified by CR and the horizontal is the minimum of the statistic in the range. Although they don't quite match, it is obvious that this value for lambda produces a result close to both the $\chi^2 : \lambda = 1$ and the local minimum. Values in $(-1, 2/3)$ are progressively better estimators as they get closer to the optimal values, as we can see by the decreasing value of the statistic in that range.

In summary, the value of the statistic is generally decreasing over the range with a local minimum somewhere just greater than the value for λ corresponding with the optimal value specified by CR ($\lambda = 2/3$).

Question 4

```
galton.dat <- matrix(c(5,4,1,12,42,14,2,15,10),nrow = 3)
dimnames(galton.dat) <- list(c("Arches","Loops","Whorls"),
                             c("Arches","Loops","Whorls"))
galton.dat
```

```
##        Arches Loops Whorls
## Arches      5    12      2
## Loops       4    42     15
## Whorls      1    14     10
```

```
options(scipen=999)
monte.study.exe <- function(ctable,
                            B = 10000,
                            correct = TRUE,
                            correct.val = 0.5){

  # ctable: a contingency table
  # correct: if 0 values present, should they be replaced to avoid errors? Defaults to TRUE.
  # correct.val: if correct = TRUE, what value should be used to replace 0? Defaults to 0.5.
```

```
options(digits = 3, warn = FALSE)
rows  = nrow(ctable)
cols  = ncol(ctable)
dof   = min(nrow(ctable), ncol(ctable))-1
ctsum = sum(ctable)

if(correct){
  if(any(ctable == 0)){
        ctable = ctable + correct.val
      }
    }

ptable = prop.table(ctable)
pvec   = vector(mode = "numeric", length = B)
rrc    = rowSums(ctable)
ccc    = colSums(ctable)

for(i in 1:B){
  pvec[i] = chisq.test(r2dtable(1,rrc,ccc)[[1]])$statistic
}
rr = rowSums(ptable)
cc = colSums(ptable)


# Pearson's Phi
pphimatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    pphimatrix[m,n] = ((ptable[m,n] - rr[m]*cc[n])^2)/(rr[m]*cc[n])
  }
}

pphistat   = sum(pphimatrix)
p.pphistat = sum(pvec >= ctsum*pphistat)/B


# Cramer

cstat   = sqrt(pphistat/dof)
p.cstat = sum(pvec >= (cstat^2)*ctsum*dof)/B

# Chi-Square

xsstat   = chisq.test(ctable)$statistic
p.xsstat = sum(pvec >= xsstat)/B

# Tchouproff

tstat   = sqrt(pphistat/((nrow(ctable)-1)*(ncol(ctable)-1)))
p.tstat = sum(pvec >= (tstat^2)*ctsum*(rows - 1)*(cols - 1))/B

# Pearson's
```

```
pstat   = sqrt(pphistat/(1 + pphistat))
p.pstat = length(pvec[pvec > pstat])

# Sakoda's

sstat    = sqrt(pphistat/(1 + pphistat)*((dof+1)/dof))
sstatadj = ctsum*((1-(sstat^2 * dof)/(dof + 1))^(-1) - 1)
p.sstat  = length(pvec[pvec > sstatadj])/B

# Belson
bmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    bmatrix[m,n] = (ptable[m,n] - rr[m]*cc[n])^2
  }
}
pvecB = vector(mode = "numeric", length = B)

for(i in 1:B){
  matsB = matrix(NA, nrow = rows, ncol = cols)
  pmatsB = r2dtable(1,rrc,ccc)[[1]]/ctsum
  for(m in 1:rows){
    for(n in 1:cols){
      matsB[m,n] = (pmatsB[m,n] - rr[m]*cc[n])^2
    }
  }
  pvecB[i] = (ctsum^2)*sum(matsB)
}
bstat   = (ctsum^2)*sum(bmatrix)
p.bstat = sum(pvecB >= bstat)/B


# Jordan
jmatrix = matrix(NA, nrow = rows, ncol = cols)
for(m in 1:rows){
  for(n in 1:cols){
    jmatrix[m,n] = ptable[m,n]*(ptable[m,n] - rr[m]*cc[n])^2
  }
}

pvecJ = vector(mode = "numeric", length = B)

for(i in 1:B){
  matsJ = matrix(NA, nrow = rows, ncol = cols)
  pmatsJ = r2dtable(1,rrc,ccc)[[1]]/ctsum
  for(m in 1:rows){
    for(n in 1:cols){
      matsJ[m,n] = pmatsJ[m,n]*(pmatsJ[m,n] - rr[m]*cc[n])^2
    }
  }
  pvecJ[i] = (ctsum)*sum(matsJ)
}
jstat   = ctsum*sum(jmatrix)
```

```
  p.jstat = sum(pvecJ >= jstat)/B

  # Variation of Squares
  vosmatrix = matrix(NA, nrow = rows, ncol = cols)
  for(m in 1:rows){
    for(n in 1:cols){
      vosmatrix[m,n] = (ptable[m,n]-rr[m]*cc[n])*(ptable[m,n] + rr[m]*cc[n])
    }
  }
  pvecV = vector(mode = "numeric", length = B)

  for(i in 1:B){
    matsV = matrix(NA, nrow = rows, ncol = cols)
    pmatsV = r2dtable(1,rrc,ccc)[[1]]/ctsum
    for(m in 1:rows){
      for(n in 1:cols){
        matsV[m,n] = (pmatsV[m,n]-rr[m]*cc[n])*(pmatsV[m,n] + rr[m]*cc[n])
      }
    }
    pvecV[i] = (ctsum^2)*sum(matsV)
  }
  vosstat   = (ctsum^2)*sum(vosmatrix)
  p.vosstat = sum(pvecV >= vosstat)/B


  # Output
  out = matrix(c(xsstat,   bstat,   jstat,   vosstat,   pphistat,   sstat,   tstat,   cstat,
                 p.xsstat, p.bstat, p.jstat, p.vosstat, p.pphistat, p.sstat, p.tstat, p.cstat),
               ncol = 2)

  dimnames(out) = list(c("X-square","Belson's","Jordan's",
                         "Var. of Squares","Pearson's Phi", "Sakoda's",
                         "Tchouproff's","Cramer's"),
                       c("Stat","MC.p-value"))
  return(out)
}

monte.study.exe(galton.dat,correct = FALSE)
```

```
##                      Stat MC.p-value
## X-square         11.1699     0.0221
## Belson's         48.0305     0.1857
## Jordan's          0.0496     0.2957
## Var. of Squares 146.9029     0.1051
## Pearson's Phi     0.1064     0.0221
## Sakoda's          0.3798     0.0221
## Tchouproff's      0.1631     0.0221
## Cramer's          0.2306     0.0221
```

Question 5

For large N, the lecture notes tell us that the Freeman-Tukey transformation gives us the following representation of the Freeman-Tukey statistic:

$$T^2 = 4n \sum_i \sum_j (\sqrt{p_{ij}} - \sqrt{p_{i*}p_{j*}})^2$$

which is a χ^2 RV with $(I-1)(J-1)$ degrees of freedom. In turn, the Freeman-Tukey statistic is a member of the Cressie-Read family of Power Divergence statistics:

$$CR(\lambda) = \frac{2n}{\lambda(\lambda+1)} \sum_i \sum_j p_{ij} \left[\left(\frac{p_{ij}}{p_{i*}p_{*j}} \right)^{\lambda} - 1 \right]$$

when $\lambda = -1/2$. The χ^2 statistic is also a member of Cressie-Read family - $CR(1)$. By definition, all members of this family of statistics have a χ^2 as their limiting distribution, which implies that - due it being a member of this family - the Freeman-Tukey statistic derived from the Freeman-Tukey variance transformation can approximate the χ^2 distribution as $n \to \infty$. However, it must be noted that Cressie and Read stated that an optimal value for λ was somewhere in the range $[0, 2/3]$, excluding the Freeman-Tukey statistic from the list of those considered to be good approximations of the χ^2 for all n.

An actual proof that the limiting distribution of the Freeman-Tukey statistic is the χ^2 is found in *Discrete Mulivariate Analysis: Theory and Practice* by Bishop, Feinberg and Holland. I haven't included it here because I wasn't able to come to grips with it myself, meaning that any working towards a proper answer to this question would simply be transcribing from their text *sans* understanding on my part. Given that T^2 is written in terms of the difference between expected and observed counts with observed variance equal to 1, it is obviously a χ^2 RV.

However the process is not my own - and neither is any further proof stating the variance of $\sqrt{n_{ij}}$, so I'll leave my answer here.

Question 6

i) We note that a property that a fundamental property of the χ^2 distribution is as follows. If X_i are independently distributed, standard normal variables, then a random variable X such that

$$X := X_1 + X_2 + X_3 + \cdots + X_{n-1} + X_n$$

will follow a χ^2 distribution with mean v and standard deviation $\sqrt{2v}$. By standardising (subtracting the mean and dividing by the standard deviation) we get:

$$z = \frac{X - v}{\sqrt{2v}} \implies X = v + z * \sqrt{2v}$$

due to the fact that - by the CLT - the distribution of z tends to the standardised gaussian distribution as $v \to \infty$. Thus $\chi^2 \approx X = v + z * \sqrt{2v}$.

ii) Using a similar argument to part i), if X_i are independently distributed, standard normal variables, then a random variable X such that

$$X := X_1 + X_2 + X_3 + \cdots + X_{n-1} + X_n \implies \frac{X}{v} := \frac{X_1}{v} + \frac{X_2}{v} + \frac{X_3}{v} + \cdots + \frac{X_{n-1}}{v} + \frac{X_n}{v} \implies \left(\frac{X}{v} \right)^{1/3} := \left(\frac{X_1}{v} + \frac{X_2}{v} + \frac{X_3}{v} +$$

is also χ^2 distributed. From this we derive the following:

$$z = \frac{(X/v)^{1/3} - (1 - 2/9v)}{\sqrt{2/9v}} \implies X = v * \left(1 - 2/9v + z * \sqrt{2/9v} \right)^3$$

First, I'll add in the data from the the question sheet.

```r
# values of Z_a for a = 0.1,0.05,0.01,0.005,0.001
z_scores <- c(1.28155,1.64485,2.32635,2.57583,3.09023)
v_values <- c(2,5,10,20,50)

X2_approx_i <- matrix(c(4.60517,09.2364,15.9872,28.4120,63.1671,
                        5.9915,11.0705,18.3070,31.4104,67.5048,
                        9.2103,15.0863,23.2093,37.5662,76.1539,
                        10.5966,16.7496,25.1882,39.9968,79.4900,
                        13.8155,20.5150,29.5883,45.3147,86.6608),
                      nrow = 5)
dimnames(X2_approx_i) <- list(c(2,5,10,20,50),c(0.1,0.05,0.01,0.005,0.001))
knitr::kable(X2_approx_i)
```

	0.1	0.05	0.01	0.005	0.001
2	4.61	5.99	9.21	10.6	13.8
5	9.24	11.07	15.09	16.8	20.5
10	15.99	18.31	23.21	25.2	29.6
20	28.41	31.41	37.57	40.0	45.3
50	63.17	67.50	76.15	79.5	86.7

Next, we'll simulate values using the approximation in part i).

```r
X2_approx_ii <- matrix(nrow = 5,ncol = 5)
dimnames(X2_approx_ii) <- list(c(2,5,10,20,50),c(0.1,0.05,0.01,0.005,0.001))
for(i in 1:5){
  for(j in 1:5){
    X2_approx_ii[i,j] = v_values[i] + z_scores[j]*sqrt(2*v_values[i])
  }
}
knitr::kable(X2_approx_ii)
```

	0.1	0.05	0.01	0.005	0.001
2	4.56	5.29	6.65	7.15	8.18
5	9.05	10.20	12.36	13.14	14.77
10	15.73	17.36	20.40	21.52	23.82
20	28.11	30.40	34.71	36.29	39.54
50	62.82	66.45	73.26	75.76	80.90

Lastly, we'll simulate values using the Wilson and Hilferty approximation.

```r
X2_approx_iii <- matrix(nrow = 5,ncol = 5)
dimnames(X2_approx_iii) <- list(c(2,5,10,20,50),c(0.1,0.05,0.01,0.005,0.001))
for(i in 1:5){
  for(j in 1:5){
    X2_approx_iii[i,j] = v_values[i]*(1-(2/(9*v_values[i])) + z_scores[j]*sqrt(2/(9*v_values[i])))^3
  }
}
knitr::kable(X2_approx_iii)
```

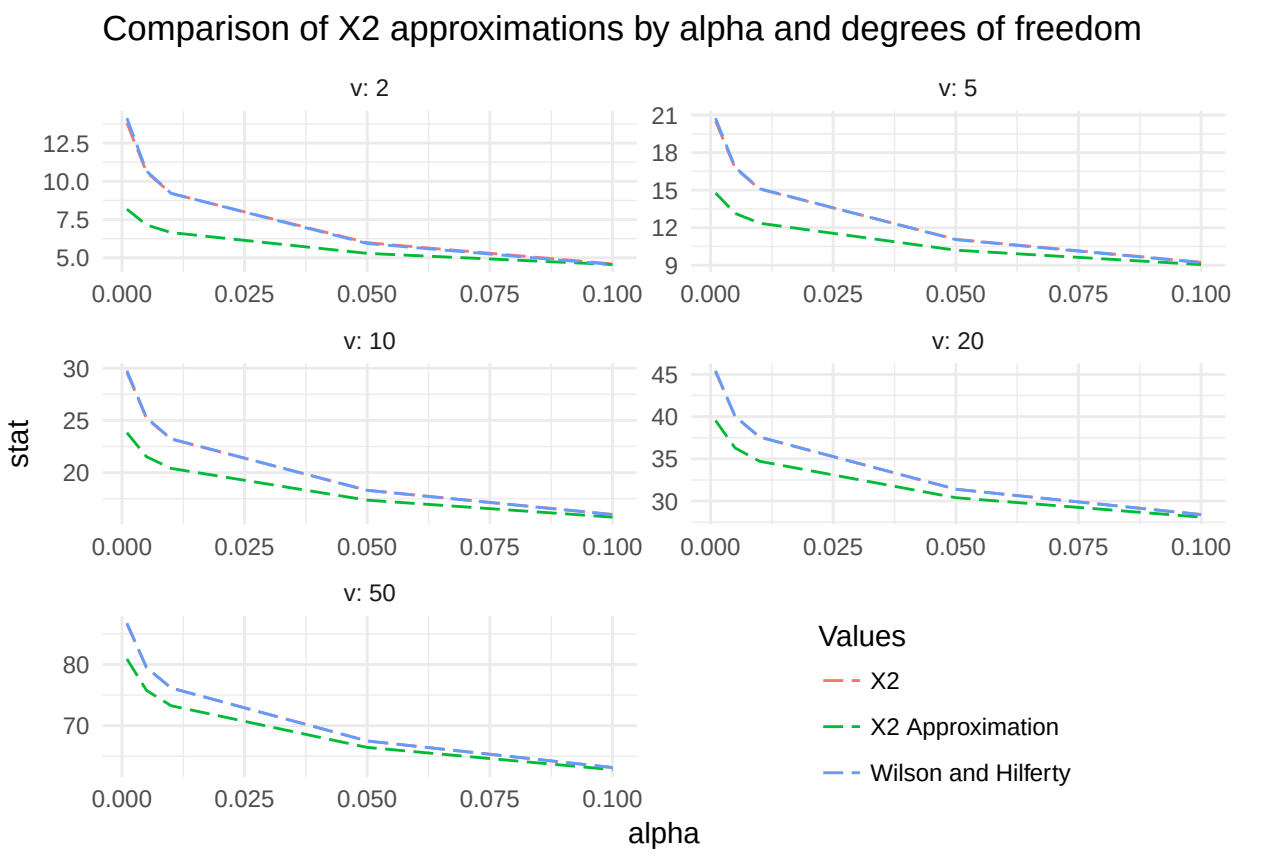
	0.1	0.05	0.01	0.005	0.001
2	4.56	5.94	9.22	10.7	14.1

	0.1	0.05	0.01	0.005	0.001
5	9.21	11.04	15.12	16.8	20.8
10	15.97	18.29	23.24	25.3	29.8
20	28.40	31.40	37.59	40.0	45.4
50	63.16	67.50	76.17	79.5	86.7

Now, we'll combine the data sets and plot each against the others to compare.

```
options(digits=7)
library(reshape2)
library(ggplot2)
X2_i <- cbind(rep("X2",25),melt(X2_approx_i))
colnames(X2_i) <- c("Values","v","alpha","stat")
X2_ii <- cbind(rep("X2 Approximation",25),melt(X2_approx_ii))
colnames(X2_ii) <- c("Values","v","alpha","stat")
X2_iii <- cbind(rep("Wilson and Hilferty",25),melt(X2_approx_iii))
colnames(X2_iii) <- c("Values","v","alpha","stat")
X2_all <- rbind(X2_i,X2_ii,X2_iii)

ggplot(data = X2_all, aes(x = alpha, y = stat)) +
  geom_line(aes(group = Values, col = Values), linetype = 5) +
  facet_wrap(~v, nrow = 3, scales = "free", labeller = "label_both") +
  theme_minimal() +
  labs(title = "Comparison of X2 approximations by alpha and degrees of freedom") +
  theme(legend.position = c(0.75, 0.1))
```



The fact that we can't see the χ^2 (red) line in any of the facets is telling: the Wilson and Hilferty approximation is close enough for all of the values for α and v tested as to completely obscure it. The approximation derived in part i) however is not a good one for values of α close to 0. Increasing the degrees of freedom may make it a better one, but it still falls well short compared to the Wilson and Hilferty approximation.

Lastly, we should note that as the value for α increases (i.e. the conditions for significance are reduced), the approximations converge towards a common value.

Note: I've altered the linetype in the above chart to try and show that more red is visible for lower values of v: i.e. that the Wilson and Hilferty approximation is not perfect, but like the other approximation it improves as v increases.