Upward Mobility analysis

Alex Haase

# External Requirements: none - all computations used standard R libraries

rm(list=ls()) # clear global environment  
getwd() # double check wd

## [1] "C:/Users/Alex Haase/Documents/DS-CS GitHub Projects/Upward-Mobility"

mobility = read.csv("http://dept.stat.lsa.umich.edu/~bbh/s485/data/mobility5.csv")  
class(mobility) #turns the data 'mobility' into a dataframe

## [1] "data.frame"

# Function to Generate Confidence Intervals using the Wilson Method

wilsonCI = function(n,p) {  
 k = qnorm(0.975)  
 q = 1 - p  
 lowerbound = (n\*p + (k^2)/2)/(n+k^2) - (k\*n^(1/2)\*((p\*q+(k^2)/(4\*n))^(1/2)))/(n+k^2)  
 upperbound = (n\*p + (k^2)/2)/(n+k^2) + (k\*n^(1/2)\*((p\*q+(k^2)/(4\*n))^(1/2)))/(n+k^2)  
 CI = c(lowerbound, upperbound)  
 print(CI) #prints the CI  
}

# Creating Dataframes and Generating Confidence Intervals

zone = mobility$zone  
n.lowstart = mobility$n.lowstart  
p.upmover = mobility$p.upmover  
  
for (i in 1:nrow(mobility)) {  
 wilCI = wilsonCI(n.lowstart[i], p.upmover[i])   
 print(as.character(zone)[i])  
 print(wilCI)  
 #sprintf(as.character(zone)[i], wilCI)  
}

## [1] 0.04537659 0.32127482  
## [1] "Anchorage"  
## [1] 0.04537659 0.32127482  
## [1] 0.01787478 0.20718637  
## [1] "Phoenix"  
## [1] 0.01787478 0.20718637  
## [1] 0.01495098 0.17704654  
## [1] "St. George"  
## [1] 0.01495098 0.17704654  
## [1] 0.07926677 0.29366205  
## [1] "Modesto"  
## [1] 0.07926677 0.29366205  
## [1] 0.05309655 0.29681327  
## [1] "San Jose"  
## [1] 0.05309655 0.29681327  
## [1] 0.0000000 0.1486549  
## [1] "San Francisco"  
## [1] 0.0000000 0.1486549  
## [1] 0.06449441 0.30128056  
## [1] "San Diego"  
## [1] 0.06449441 0.30128056  
## [1] 0.05401527 0.15556884  
## [1] "Los Angeles"  
## [1] 0.05401527 0.15556884  
## [1] 0.06403451 0.34653622  
## [1] "Portland"  
## [1] 0.06403451 0.34653622  
## [1] 0.08179529 0.50256376  
## [1] "Seattle"  
## [1] 0.08179529 0.50256376  
## [1] 0.03852035 0.28057801  
## [1] "Detroit"  
## [1] 0.03852035 0.28057801  
## [1] 0.005717222 0.161941052  
## [1] "Dayton"  
## [1] 0.005717222 0.161941052  
## [1] 0.02651877 0.28914139  
## [1] "Lima"  
## [1] 0.02651877 0.28914139  
## [1] 0.005369411 0.153187254  
## [1] "Canton"  
## [1] 0.005369411 0.153187254  
## [1] 0.00000000 0.09641863  
## [1] "Cleveland"  
## [1] 0.00000000 0.09641863  
## [1] 0.0000000 0.1071792  
## [1] "Minneapolis"  
## [1] 0.0000000 0.1071792  
## [1] 0.03957953 0.23051775  
## [1] "Galesburg"  
## [1] 0.03957953 0.23051775  
## [1] 6.938894e-18 9.180987e-02  
## [1] "Chicago"  
## [1] 6.938894e-18 9.180987e-02  
## [1] 6.938894e-18 1.206433e-01  
## [1] "Wichita"  
## [1] 6.938894e-18 1.206433e-01  
## [1] 0.02055465 0.23369593  
## [1] "Kansas City"  
## [1] 0.02055465 0.23369593  
## [1] 0.01981206 0.22645363  
## [1] "Erie"  
## [1] 0.01981206 0.22645363  
## [1] 0.05602818 0.26705743  
## [1] "Olean"  
## [1] 0.05602818 0.26705743  
## [1] 0.05134284 0.28852406  
## [1] "Sunbury"  
## [1] 0.05134284 0.28852406  
## [1] 0.05951087 0.17791206  
## [1] "New York"  
## [1] 0.05951087 0.17791206  
## [1] 0.02135509 0.24141554  
## [1] "Toms River"  
## [1] 0.02135509 0.24141554  
## [1] 0.02529543 0.27814947  
## [1] "Newark"  
## [1] 0.02529543 0.27814947  
## [1] 0.04260807 0.20978235  
## [1] "Philadelphia"  
## [1] 0.04260807 0.20978235  
## [1] 0.007393456 0.202418065  
## [1] "Boston"  
## [1] 0.007393456 0.202418065  
## [1] -6.938894e-18 7.865160e-02  
## [1] "Springfield"  
## [1] -6.938894e-18 7.865160e-02  
## [1] 0.02529543 0.27814947  
## [1] "Bridgeport"  
## [1] 0.02529543 0.27814947  
## [1] 0.03957953 0.23051775  
## [1] "Charlotte"  
## [1] 0.03957953 0.23051775  
## [1] 0.01079714 0.08877099  
## [1] "Florence"  
## [1] 0.01079714 0.08877099  
## [1] 0.05193022 0.22194701  
## [1] "Lafayette"  
## [1] 0.05193022 0.22194701  
## [1] 0.03821375 0.16794395  
## [1] "Shreveport"  
## [1] 0.03821375 0.16794395  
## [1] 0.0180603 0.1436955  
## [1] "Tampa"  
## [1] 0.0180603 0.1436955  
## [1] 0.01021637 0.12535177  
## [1] "Macon"  
## [1] 0.01021637 0.12535177  
## [1] 0.03046594 0.22960487  
## [1] "Baltimore"  
## [1] 0.03046594 0.22960487  
## [1] 0.04586458 0.19844607  
## [1] "San Antonio"  
## [1] 0.04586458 0.19844607  
## [1] 0.04438348 0.19261701  
## [1] "Kerrville"  
## [1] 0.04438348 0.19261701  
## [1] 0.01787478 0.20718637  
## [1] "Dallas"  
## [1] 0.01787478 0.20718637

# Likelihood Ratio Test - multiple parameters

Note: p\_upmover and n\_lowstart are vectors for proportions for all the cities and their sample sizes

n\_lowstart = as.vector(mobility$n.lowstart)  
p\_upmover = as.vector(mobility$p.upmover)  
  
#n\_lowstart #n.lowstart data (vector)  
#p\_upmover #p.upmover data (vector)  
  
#under H0, the MLE is:  
p\_hat\_H0 = sum(n\_lowstart\*p\_upmover)/sum(n\_lowstart)  
p\_hat\_H0 # = 0.07850242

## [1] 0.07850242

#under Ha, the MLE is:   
p\_hat\_Ha = p\_upmover  
  
#region generation  
west = subset(mobility, region == 'west')  
midwest = subset(mobility, region == 'midwest')  
northeast = subset(mobility, region == 'northeast')  
south = subset(mobility, region == 'south')  
  
p\_w = sum(west$n.lowstart\*west$p.upmover)/sum(west$n.lowstart) #west  
p\_mw = sum(midwest$n.lowstart\*midwest$p.upmover)/sum(midwest$n.lowstart) #midwest  
p\_ne = sum(northeast$n.lowstart\*northeast$p.upmover)/sum(northeast$n.lowstart) #northeast  
p\_s = sum(south$n.lowstart\*south$p.upmover)/sum(south$n.lowstart) #south  
  
p\_hat\_vec = c(p\_w, p\_mw, p\_ne, p\_s)  
p\_0\_vec = c(p\_hat\_H0,p\_hat\_H0)  
  
n\_l\_region = c(sum(west$n.lowstart,sum(midwest$n.lowstart)))  
n\_l\_region

## [1] 701

n\_l\_region = c(sum(west$n.lowstart),sum(midwest$n.lowstart))  
  
p\_u\_r = c(p\_w, p\_mw, p\_ne, p\_s)  
  
#log likelihood  
log\_lik = function(p) {  
 sum(dbinom(round(n\_l\_region\*p\_u\_r), size = n\_l\_region, prob = p, log = T))  
}  
  
#creating test statistic  
testStatistic = 2\*(log\_lik(p\_hat\_vec) - log\_lik(p\_0\_vec)) #test statistic  
testStatistic

## [1] 11.86681

#compare test stat to chi-square on 3 degree's of freedom  
chisq = pchisq(testStatistic, 3, lower = F)  
chisq

## [1] 0.007853613