~~~
---
title: "Bayes"
author: "ME"
date: "10/11/2019"
output: html_document
---
Section 5 - Bayesian Statistics
```{r}
#Probability of a positive test given that you have the cystic fibrosis and the
probability of a negative test given that you don't have the disease
#P(+|D=1)=0.99, Prob(-|D=0)=0.99

#what is the probability that a person has cystic fibrosis given they have a positive
test
#P(D=1|+)? when P(D=1)=0.00025 (from random population)

#use Bayes -> P(A|B)=P(A and B)/P(B)=P(B|A)*P(A)/P(B)
#apply to our example
#We will need to use
#P(D=1|+)=P(+|D=1)*P(D=1)/P(+)=P(+|D=1)*P(D=1)/(P(+|D=1)*P(D=1)+P(+|D=0)*P(D=0))
#this comes out to a probability of 0.02 even though the test is 99% accurate
#this also assumes someone is chosen at random

#Monte Carlo to prove
prev<-0.00025
N<-100000
outcome<-sample(c("Disease", "Healthy"),N,replace=TRUE, prob=c(prev,1-prev))
mean(outcome=="Disease")
sum(outcome=="Disease")

#false positive
#P(D=0|+)=P(+|D=0)*P(D=0)/(P(+|D=0)*P(D=0)+P(+|D=1)*P(D=1))
#this comes out to 97.6%!!!

#prove it
accuracy<-0.99
test<-vector("character",N)
test[outcome=="Disease"]<-sample(c("+","-"),N,replace=TRUE,prob=c(accuracy,1-accuracy))
test[outcome=="Healthy"]<-sample(c("-","+"),N,replace=TRUE,prob=c(accuracy,1-accuracy))
table(outcome,test)

#outcome creates a population that we know is either diseased or healthy
#test applies the accuracy of the test to the population

#also can be done this way
N_D<-sum(outcome=="Disease")
N_H<-sum(outcome=="Healthy")
test2<-vector("character",N)
test2[outcome=="Disease"]<-sample(c("+","-"),N_D,replace=TRUE,prob=c(accuracy,1-
accuracy))
test2[outcome=="Healthy"]<-sample(c("-","+"),N_H,replace=TRUE,prob=c(accuracy,1-
accuracy))
table(outcome,test2)

```

BAYES IN PRACTICE

Bayesian stats allows prior knowledge to modify observed results, which alters our
~~~

conclusions about even probabilities.

THE HIERARCHICAL MODEL

level one descirbes general average -> called the prior distribution
p~N(mu=mean, tau=standard dev) describes player to player variability in natural ability to hit

second level describes individual -> called the sampling distribution
Y|p~N(p,sigma) describes a player's observed batting average given their ability p. This represents variability due to luck

The posterior distribution allows us to compute the probability distribution of p given that we have observed data Y.

By the continuous version of Bayes' rule, the expected value of the posterior distribution p given Y=y is a weighted average between the prior mean mu and the observed data Y:
E(p|y)=B * mu+(1-B)*Y=mu+(1-B)(Y-mu) where B=sigma^2/(sigma^2 + tau^2)

The standard error of the posterior distribution SE(p|Y)^2 is 1/(1/sigma^2 + 1/tau^2).
Taking the sqrt of both side solves for the SE

This Bayesian approach is also known as shrinking. When sigma is large, B is close to 1 and our prediction of p shrinks towards mu. When sigma is small B is close to 0 and our prediction of p is more weighted towards the observed data Y.
```{r}
#practical case in baseball
#a baseball player has 20 at bats and gets 9 hits -> batting average of .45
#will this average continue? from past experience we know batting average are ~0.275
with a standard devaition of ~0.027
#the values above are mu and tau respectively
#sigma=sqrt(p*(1-p)/N) N is the number of at bats

Y<-0.45 #observed batting average
N<-20
sigma<-sqrt(Y*(1-Y)/N)
sigma #0.111

mu<-0.275
tau<-0.027


B<-sigma^2/(sigma^2 + tau^2)
B #0.944
Expected<-B*mu+(1-B)*Y
Expected #0.285
SE_Expected<-sqrt(1/(1/sigma^2 + 1/tau^2))
SE_Expected #0.026

#95% credible interval -> Expected +- 2*SE_Expected
credible_int<-c(Expected - 2*SE_Expected,Expected + 2*SE_Expected)
credible_int #0.2322591 0.3372120

#so no, we don't expect this player to continue his >>above normal batting average

```

Assessment
```{r}
```

```
# Load the libraries and poll data
library(dplyr)
library(dslabs)
data(polls_us_election_2016)

# Create an object `polls` that contains the spread of predictions for each candidate in
Florida during the last polling days
polls <- polls_us_election_2016 %>%
  filter(state == "Florida" & enddate >= "2016-11-04" ) %>%
  mutate(spread = rawpoll_clinton/100 - rawpoll_trump/100)

# Examine the `polls` object using the `head` function
head(polls)

# Create an object called `results` that has two columns containing the average spread
(`avg`) and the standard error (`se`). Print the results to the console.
results<-polls %>% summarize(avg=mean(polls$spread),se=sd(polls$spread)/sqrt(n()))
results

#we don't use the sd function on its own because the standard error of the mean is the
sd divided by the sqrt of n, n being the number of polls.

# Define `mu` and `tau` because the elections in FL are generally close
mu <- 0
tau <- 0.01

# Define a variable called `sigma` that contains the standard error in the object
`results
sigma<-results$se

# Define a variable called `Y` that contains the average in the object `results`
Y<-results$avg

# Define a variable `B` using `sigma` and `tau`. Print this value to the console.
B<-sigma^2/(sigma^2+tau^2)

# Calculate the expected value of the posterior distribution
Expected<-B*mu+(1-B)*Y

# Compute the standard error of the posterior distribution. Print this value to the
console.
SE_Expected<-sqrt(1/(1/sigma^2 + 1/tau^2))

# Construct the 95% credible interval. Save the lower and then the upper confidence
interval to a variable called `ci`.
ci<-c(Expected - 2*SE_Expected,Expected + 2*SE_Expected)

# Using the `pnorm` function, calculate the probability that the actual spread was less
than 0 (in Trump's favor). Print this value to the console.
pnorm(0,Expected,SE_Expected)

# Define the variables from previous exercises
mu <- 0
sigma <- results$se
Y <- results$avg

# Define a variable `taus` as different values of tau
taus <- seq(0.005, 0.05, len = 100)

# Create a function called `p_calc` that generates `B` and calculates the probability of
```

```
the spread being less than 0
p_calc<-function(tau){
  B<-sigma^2/(sigma^2+tau^2)
  exp_value<-B*mu+(1-B)*Y
  se<-sqrt(1/(1/sigma^2+1/tau^2))
  pnorm(0,exp_value,se)
}

# Create a vector called `ps` by applying the function `p_calc` across values in `taus`
ps<-sapply(taus,p_calc)

# Plot `taus` on the x-axis and `ps` on the y-axis
plot(taus,ps)
```

Assessment 2

Bayesian search theory is the use of Bayes' Theorem to find lost objects. It has been used to find planes crashed at sea, sunken submarines, missing people and more. Prior assumptions about the probability of finding the object in various locations are continually revised as the search proceeds.

A search team is tasked with finding a crashed plane. Their initial data suggest the plane is found somewhere within one of four areas with the following probabilities:

Pr(in A)=0.2
Pr(in B)=0.6
Pr(in C)=0.15
Pr(in D)=0.05

There is a 90% chance that if the plane is in the area, they will find it.

```{r}
options(digits=3)

pa<-.2
pb<-0.6
pc<-.15
pd<-.05

p_nf_in<-.1

#What is the probability the plane is not in B?
p_notb<-1-pb
#What is the probability the plane is in B but is not found?
pb_nf<-pb*p_nf_in
#What is the probability the plane is not found in B on day 1?
p_nf_b<-p_notb+pb_nf

#find P(in B|nf in B)
p_b_nfb<-p_nf_in*pb/p_nf_b
p_b_nfb

#find P(in C|nf in B)
p_c_nfb<-pc/p_nf_b
p_c_nfb

#find P(f in A| nf in B)
p_fa_nfb<-pa*(1-p_nf_in)/p_nf_b
#multiply this by P(not found in B) to get the probability that the plane is not found
```

```
on the first day but is found on the second
p_fa_nfb*p_nf_b

#what is the probability the plane is found in 2 days
p_fa_nfb*p_nf_b+(1-p_nf_b)
```