Applied Statistics (Chapter 2)

Yasuhiro Omori

A1A2 term, 2019-2020

University of Tokyo

Chapter 2

Section 2.3

```
library(LearnBayes)
# Generate a sequence:  0.05,0.15,0.25,...,0.95
# incremnet = 0.1
p = seq(0.05,0.95,by=0.1)
# Define the vector prior
prior=c(1,5.2,8,7.2,4.6,2.1,0.7,0.1,0,0)
# Normalize prior by dividing the sum of elements
prior=prior/sum(prior)
# Plot prior.type:histogram like vertical line
plot(p,prior,type="h",ylab="Prior Probability")
# Let us define a new vector data
# for 11 successes and 16 failures
data = c(11, 16)
```

```
# Compute posterior probabilities, post, at p.
# using the discrete prior, prior, and obs., data
post = pdisc(p, prior, data)
# Bind columns, p,prior,post
# Round them off to 2 decimal places, and output
round(cbind(p, prior, post),2)
# We will use lattice library
library(lattice)
# Construct a dataframe PRIOR using a character
# 'prior' and vectors p, prior
PRIOR=data.frame("prior",p,prior)
POST=data.frame("posterior",p,post)
```

Section 2.3 (continued)

```
# Change variable names for PRIOR and POST
names(PRIOR)=c("Type","P","Probability")
names(POST)=c("Type","P","Probability")
# Bind rows of PRIOR and POST and redefine data
data=rbind(PRIOR,POST)
windows()
# Plot posterior probabilities
plot(p,post,type="h",ylab="Posterior probability")
windows()
# Plot Probability vs P by Type using data
# layout:1 x 2 matrix
xyplot(Probability~P|Type,data=data,
layout=c(1,2), type="h",lwd=3,col="black")
```

```
library(LearnBayes)
# Construct list:quantile2 where
# quantile2$p=0.9 and quantile2$x=0.5
quantile2=list(p=.9,x=.5)
quantile1=list(p=.5,x=.3)
# Find p~beta(ab[1],ab[2]) prior which satisfies
# Pr( p < 0.5)=0.9 and Pr( p < 0.3)=0.5
ab=beta.select(quantile1,quantile2)
a=ab[1]; b=ab[2];s=11;f=16
# Draw the posterior density:  beta(a+s,b+f)
# lty:0(invisible),1(solid),2(dashed),3(dotted)
curve(dbeta(x,a+s,b+f),from=0,to=1,xlab="p",
ylab="Density",lty=1,lwd=4)
```

Section 2.4

```
# Overwrite the posterior density:  beta(1+s,1+f)
# with uniform prior
curve(dbeta(x,s+1,f+1),add=TRUE,lty=2,lwd=4)
# Overwrite the prior density:  beta(a,b)
curve(dbeta(x,a,b),add=TRUE,lty=3,lwd=4)
# Legend at (0.7, 0.4).
# Likelihood:Posterior with U(0,1) prior
legend(.7,4,c("Prior","Likelihood","Posterior"),
lty=c(3,2,1),lwd=c(3,3,3))
# Pr(p > 0.5|data)=1-Pr(p<=0.5|data)
1 - pbeta(0.5, a + s, b + f)
```

Section 2.4 (Continued)

```
# Find q1:  Pr(p<=q1|data)=0.05
# Find q2:  Pr(p<=q2|data)=0.95
# (q1,q2):  90% credible interval
qbeta(c(0.05, 0.95), a + s, b + f)
# Monte Carlo simulation
# Generate 1000 random samples from beta(a+s,b+f)
ps = rbeta(1000, a + s, b + f)
windows()
hist(ps,xlab="p")
# Estimate Pr(p>=0.5|data)
sum(ps >= 0.5)/1000
# Find sample quantiles to construct 90% interval
quantile(ps, c(0.05, 0.95))
```

```
# Discrete prior example
p=seq(0.05,0.95,by=.1)
prior=c(1,5.2,8,7.2,4.6,2.1,0.7,0.1,0,0)
prior=prior/sum(prior))
# m:number of new trials, ys:number of success
m=20; ys=0:20
# Compute the prior predictive density
pred=pdiscp(p, prior, m, ys)
cbind(0:20,pred)
# Beta prior example
ab=c(3.26,7.19)
m=20; ys=0:20
pred=pbetap(ab,m,ys)
```

```
# Monte Carlo simulation.  Generate 1000
# random samples (p1,...,p1000)~beta(3.26,7.19)
p=rbeta(1000,3.26,7.19)
# For each pi, generate yi~Bin(20,pi)
y=rbinom(1000,20,p)
table(y)
freq=table(y)
# Use names of freq  as integers
ys=as.integer(names(freq))
# Estimate the predictive density
predprob=freq/sum(freq)
```

```
# Draw the predictive density
plot(ys,predprob,type="h",xlab="y",
ylab="Predictive Probability")
# Define the matrix dist
dist=cbind(ys,predprob)
covprob=.9
# Find the (at least 90%) credible interval
discint(dist,covprob)
```