The Dirichlet Process and Clustering

The Dirichlet Process is a model for a *random discrete distribution* represented by pmf $\widetilde{f}$.

Suppose we want to sample

- $\widetilde{f} \sim DP(\alpha, G_X)$, then
- $Y_1, \ldots, Y_n \sim \widetilde{f}$ independently

and study the *marginal distribution* for $Y_1, \ldots, Y_n$

$$f_{Y_1,\ldots,Y_n}(y_1, \ldots, y_n) = \int \prod_{i=1}^{n} \widetilde{f}(y_i) \pi_0(d\widetilde{f})$$

where $\pi_0(d\widetilde{f}) \equiv DP(\alpha, G_X)$.

The marginal distribution is merely the *prior predictive distribution* under the nonparametric model.

The Dirichlet Process and Clustering

To sample from the marginal distribution *directly* using the *Polya Urn*:

- Sample $Y_1 \sim G_X$
- For $i = 2, \ldots, n$, sample $Y_i | Y_1, \ldots, Y_{i-1}$ from the conditional distribution, which is a mixture of the uniform *discrete* distribution on $\{y_1, \ldots, y_{i-1}\}$ and G_X, with weights

$$\frac{i-1}{\alpha + i - 1} \quad \text{and} \quad \frac{\alpha}{\alpha + i - 1}$$

respectively. That is, for $j = 1, \ldots, i-1$,

$$\Pr[Y_i = y_j | Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}] = \frac{1}{\alpha + i - 1}$$

and with the remaining probability $\alpha/(\alpha + i - 1)$, $Y_i \sim G_X$.

```
polya.urn<-function(nv,alv,muv,sigv){

    y<-rep(0,nv)
    y[1]<-rnorm(1,muv,sigv)
    u<-runif(nv)
    for(i in 2:nv){
        if(u[i] < alv/(alv+i-1)){
            y[i]<-rnorm(1,muv,sigv)
        }else{
            y[i]<-sample(y[1:(i-1)],size=1)
        }
    }
    return(y)

}
set.seed(3)
polya.urn(5,alv=2,0,1)

+ [1] -0.96193342 -0.53999302 -0.53999302 -0.53999302  0.08541773
```

```
n<-20
al<-2
nreps<-10000
Y.samp<-replicate(nreps,polya.urn(n,2,0,1))
library(dplyr,warn.conflicts = FALSE)
K.samp<-apply(Y.samp,2,n_distinct)
table(K.samp)          #Counts of numbers of distinct Y values

+ K.samp
+    1    2    3    4    5    6    7    8    9   10   11   12   14
+   41  283  980 1867 2320 2106 1376  684  241   84   15    2    1

prob.K.samp<-tabulate(K.samp,n)/nreps   #Proportions
```

Exact calculation:

$$\Pr[K = k] = \frac{\alpha^k B(n,k)}{\sum_{j=1}^{n} \alpha^j B(n,j)}$$

where $B(n,k)$ is an (unsigned) Stirling Number of the First Kind.

```
library(gmp)  #For the Stirling numbers
Stirling1.all(n)

+ Big Integer ('bigz') object of length 20:
+  [1] -121645100408832000 43156514681763840 0 -66860973034115328 0
+  [4] 61011607574049177 6 -3713847873452280 00 16142973653011896 0
+  [7] -5226090336251272 0 1295363698994389 6 -2503858755467550
+ [10] 381922055502195 -46280647751910 4465226757381
+ [13] -342252511900 20692933630 -973941900
+ [16] 34916946 -920550 16815
+ [19] -190 1
```

```
S.nk<-abs(as.numeric(Stirling1.all(n)))
lprob.vec<-c(1:n)*log(al)+log(S.nk)
prob.vec<-exp(lprob.vec-max(lprob.vec))
prob.vec<-prob.vec/sum(prob.vec)
round(prob.vec,6)

+  [1] 0.004762 0.033788 0.104693 0.191068 0.232611 0.202218 0.130931
+  [9] 0.025092 0.007655 0.001855 0.000358 0.000055 0.000007 0.000001
+ [17] 0.000000 0.000000 0.000000 0.000000

prob.compare<-cbind(prob.vec,prob.K.samp)
row.names(prob.compare)<-paste('K =',1:n)
colnames(prob.compare)<-c('True','Sample')
```

```
prob.compare[1:10,]

+                True Sample
+ K = 1  0.004761905 0.0041
+ K = 2  0.033787997 0.0283
+ K = 3  0.104693271 0.0980
+ K = 4  0.191068256 0.1867
+ K = 5  0.232610962 0.2320
+ K = 6  0.202217902 0.2106
+ K = 7  0.130931146 0.1376
+ K = 8  0.064906438 0.0684
+ K = 9  0.025092034 0.0241
+ K = 10 0.007654746 0.0084
```

We can sample the cluster membership *directly* in the Polya Urn

```
polya.urn.labels<-function(nv,alv,muv,sigv){
    z<-rep(0,nv)
    z[1]<-1
    u<-runif(nv)
    Kco<-1
    for(i in 2:nv){
        if(u[i] < alv/(alv+i-1)){
            Kco<-Kco+1
        }
        z[i]<-Kco
    }
    return(z)

}
set.seed(3)
polya.urn.labels(5,alv=2,0,1)

+ [1] 1 1 2 3 3
```

```
K.samp.labels<-replicate(nreps,polya.urn.labels(n,2,0,1))[,n]
prob.K.samp.labels<-tabulate(K.samp,n)/nreps  #Proportions
prob.compare<-cbind(prob.vec,prob.K.samp,prob.K.samp.labels)
row.names(prob.compare)<-paste('K =',1:n)
colnames(prob.compare)<-c('True','Sample','Labels')
prob.compare[1:10,]

+                True Sample Labels
+ K = 1  0.004761905 0.0041 0.0041
+ K = 2  0.033787997 0.0283 0.0283
+ K = 3  0.104693271 0.0980 0.0980
+ K = 4  0.191068256 0.1867 0.1867
+ K = 5  0.232610962 0.2320 0.2320
+ K = 6  0.202217902 0.2106 0.2106
+ K = 7  0.130931146 0.1376 0.1376
+ K = 8  0.064906438 0.0684 0.0684
+ K = 9  0.025092034 0.0241 0.0241
+ K = 10 0.007654746 0.0084 0.0084
```

In both approaches we can sample the *observables* Y *without* having to sample the components of $\widetilde{f}$.

In the *posterior distribution* we have $\alpha^\star = \alpha + n$, for any set B

$$G_X^\star(B) = \frac{\alpha}{\alpha + n} G_X(B) + \frac{1}{\alpha + n} \sum_{i=1}^{n} \delta_{y_i}(B).$$

and then

$$\pi_n(d\widetilde{f}) \equiv DP(\alpha^\star, G_X^\star).$$

Posterior predictive: To obtain the posterior predictive, we compute in the usual way

$$f_{Y_{n+1},\ldots,Y_{n+m}|Y_1,\ldots,Y_n}(y_{n+1},\ldots,y_{n+m}|y_1,\ldots,y_n) = \int \prod_{i=n+1}^{n+m} \widetilde{f}(y_i)\pi_n(d\widetilde{f})$$

To obtain the sample $Y_{n+1}, \ldots, Y_{n+m}$ from the posterior predictive we may use *direct* simulation

- sample $\widetilde{f} \sim \pi_n(d\widetilde{f})$
- sample $Y_{n+1}, \ldots, Y_{n+m}$ independently from $\widetilde{f}$

We may also use the *Polya Urn*

- Sample $Y_{n+1} \sim G_X^\star$, that is

$$Y_{n+1} \sim \begin{cases} G_X & \text{w.p. } \alpha/(\alpha+n) \\ \delta_{y_i}(y) & \text{w.p. } 1/(\alpha+n),\ i=1,\ldots,n. \end{cases}$$

- For $i = 2, \ldots, m$, sample

$$Y_{n+i} \sim \begin{cases} G_X^\star & \text{w.p. } \alpha^\star/(\alpha^\star+i-1) \\ \delta_{y_{n+j}}(y) & \text{w.p. } 1/(\alpha^\star+i-1),\ j=1,\ldots,i-1. \end{cases}$$

That is,

$$Y_{n+i} \sim \begin{cases} G_X & \text{w.p. } \alpha/(\alpha+n+i-1) \\ \delta_{y_j}(y) & \text{w.p. } 1/(\alpha+n+i-1),\ j=1,\ldots,n+i-1. \end{cases}$$

This sequential generation treats the *all* samples obtained *before* $n+i$,

$$y_1, \ldots, y_{n-1}, y_n, y_{n+1}, \ldots, y_{n+i-1}$$

on an equal basis.

In the limiting case of $\alpha \longrightarrow 0$, we have that $\alpha^\star = n$ and

$$G_X^\star(B) = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i}(B).$$

The limiting posterior distribution places random probabilities

$$\pi \sim Dirichlet(n-1; 1, \ldots, 1)$$

on the observed data $\{y_1, \ldots, y_n\}$.

Sample the limiting posterior using stick-breaking:

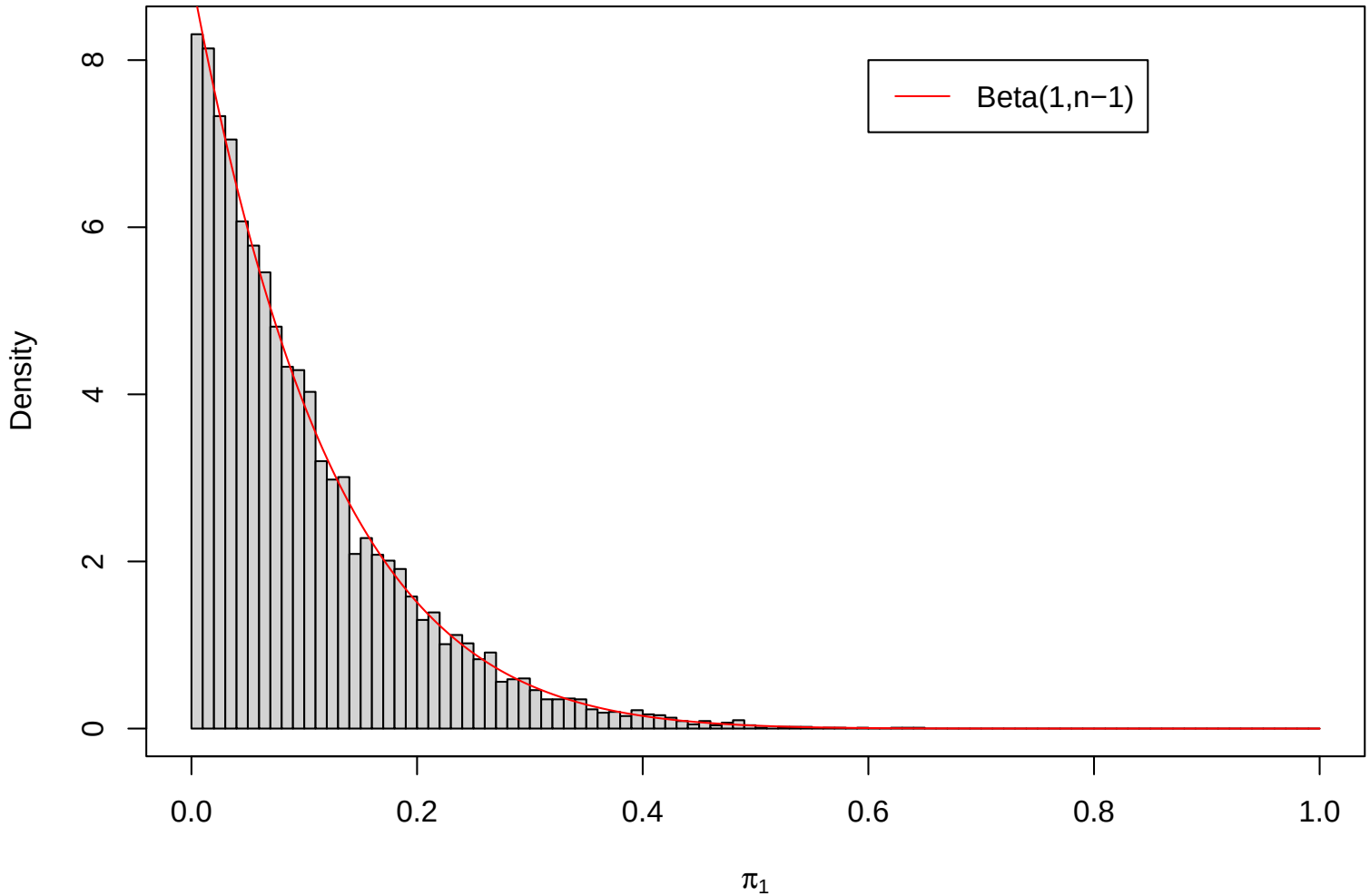
```
limit.DP<-function(nv,Yv,Mv=10000){
        V<-rbeta(Mv,1,nv)
        pivec<-c(V[1],cumprod(1-V[-Mv])*V[-1])
        Xvec<-sample(Yv,size=Mv,rep=T)
        ptot<-rep(0,nv)
        for(i in 1:nv){
                ptot[i]<-sum(pivec[Xvec==Yv[i]])
        }
        return(ptot)
}

nreps<-10000
n<-10
Y0<-runif(n)
pimat<-replicate(nreps,limit.DP(n,Y0))
```

Sampled probabilities on y_1 are distributed as $Beta(1, n-1)$.

```
hist(pimat[1,],freq=FALSE,br=seq(0,1,by=0.01),
        main='',xlab=expression(pi[1]));box()
xv<-seq(0,1,by=0.001);yv<-dbeta(xv,1,n-1)
lines(xv,yv,col='red')
legend(0.6,8,c('Beta(1,n-1)'),lty=1,col='red')
```

The Dirichlet Process and Clustering



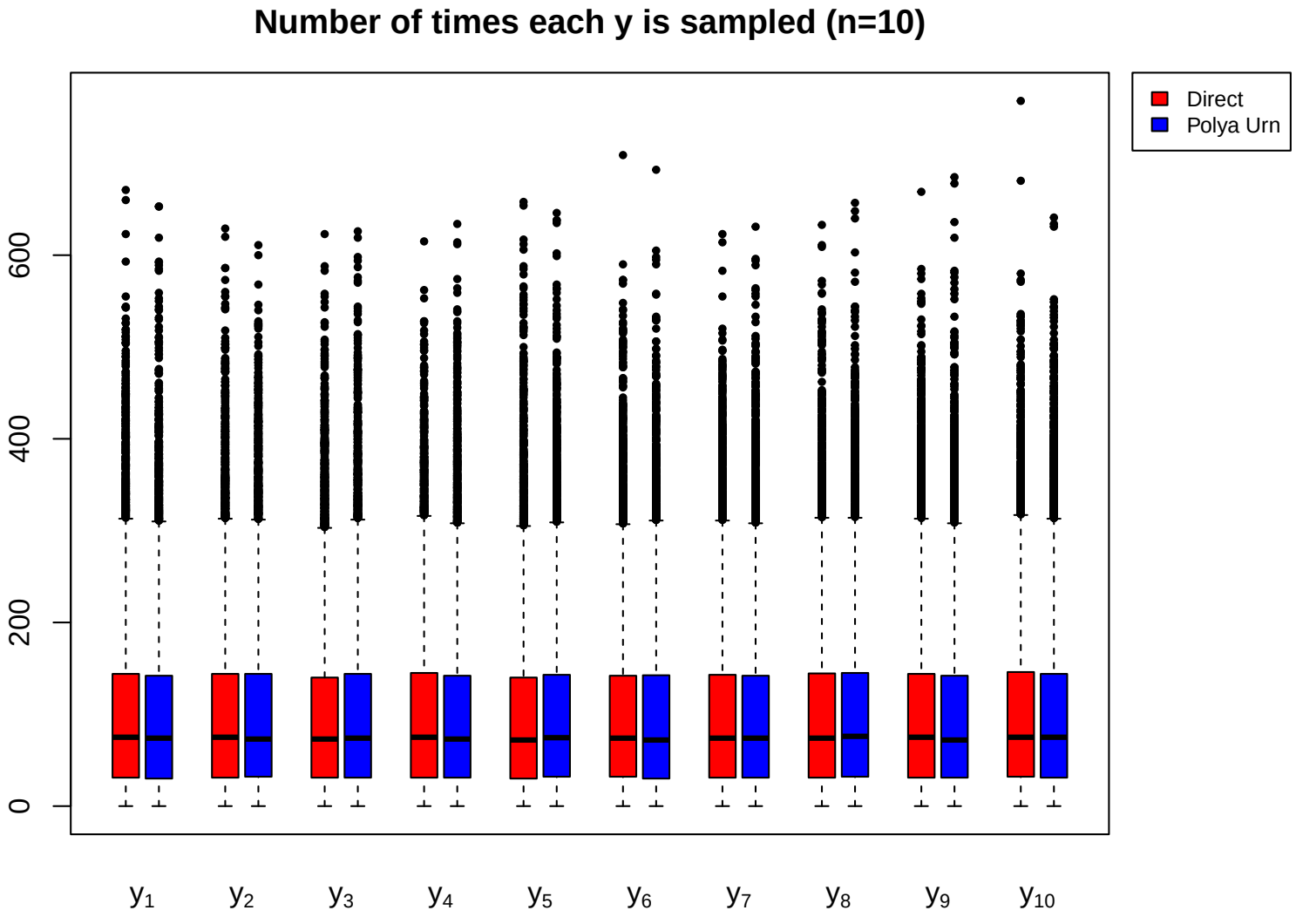
Posterior predictive: Direct sampling

```
library(extraDistr)
n<-10
m<-1000
Zsamp<-matrix(0,nrow=nreps,ncol=n)
for(i in 1:nreps){
    pi.vec<-rdirichlet(1,rep(1,n))
    Z<-sample(1:n,size=m,prob=pi.vec,rep=T)
    Zsamp[i,]<-tabulate(Z,nbins=n)
}
```

Posterior predictive: Polya Urn

```
set.seed(2413)
Zsamp0<-matrix(0,nrow=nreps,ncol=n)
for(i in 1:nreps){
    Z<-1:n
    for(j in 1:m){Z<-c(Z,sample(Z,size=1))}
    Zsamp0[i,]<-tabulate(Z,nbins=n)-1
}
```

```
all.Z<-cbind(Zsamp[,1],Zsamp0[,1])
for(j in 2:n){all.Z<-cbind(all.Z,Zsamp[,j],Zsamp0[,j])}
par(mar=c(5,4,3,6),xpd=TRUE,xaxt='n')
avec<-1:(3*n);avec<-avec[-3*(1:n)]
boxplot(all.Z,pch=19,cex=0.5,col=c('red','blue'),at=avec)
legend("topright", c("Direct", "Polya Urn"),cex=0.75,
            fill = c("red", "blue"),inset=c(-0.175,0))
nvec<-c(expression(y[1]),expression(y[2]),
            expression(y[3]),expression(y[4]),
            expression(y[5]),expression(y[6]),
            expression(y[7]),expression(y[8]),
            expression(y[9]),expression(y[10]))
text(1.5+c(0:(n-1))*3,rep(-100,10),nvec)
title('Number of times each y is sampled (n=10)')
```

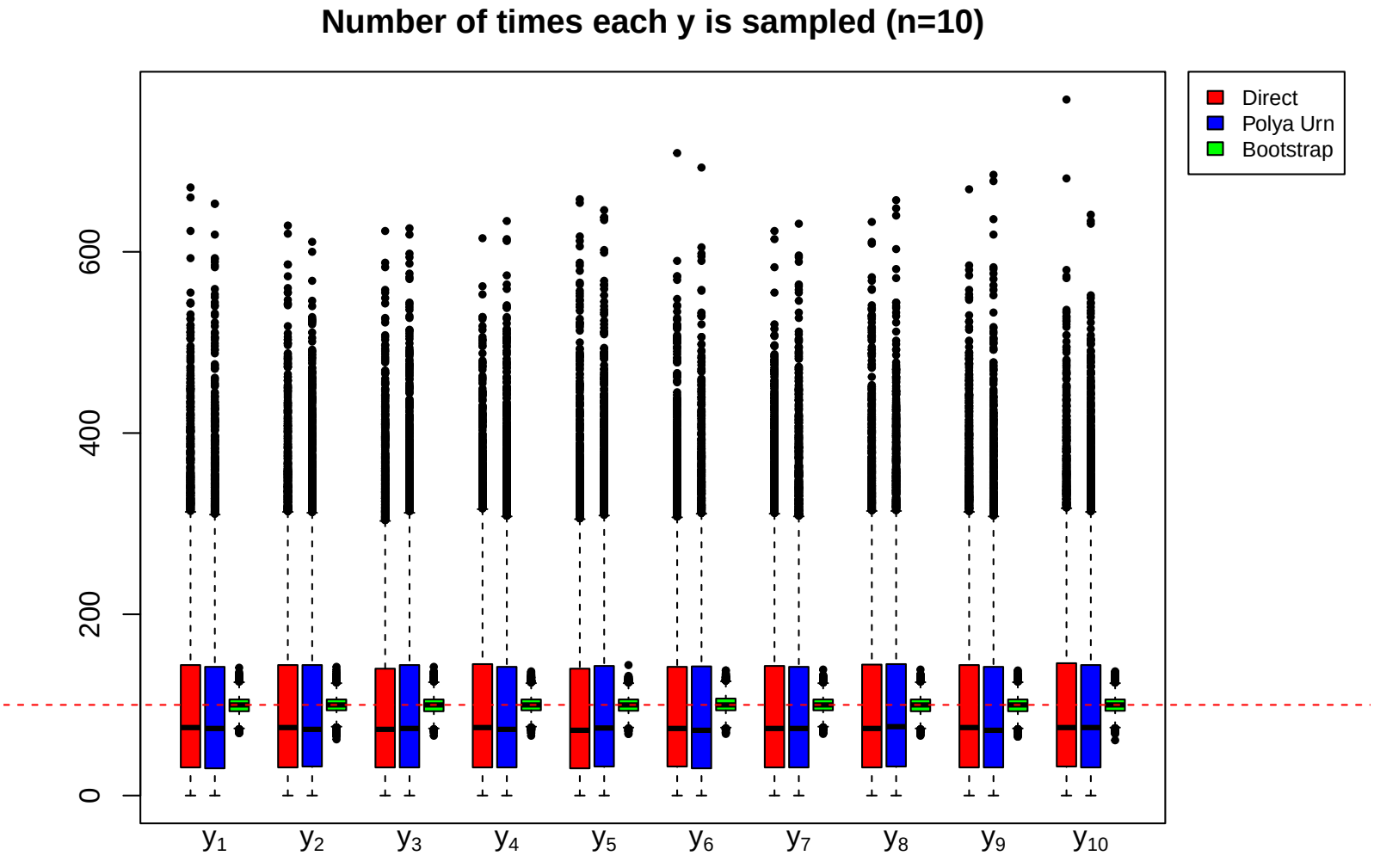

Number of times each y is sampled (n=10)
Direct
Polya Urn
0
200
400
600
y1
y2
y3
y4
y5
y6
y7
y8
y9
y10

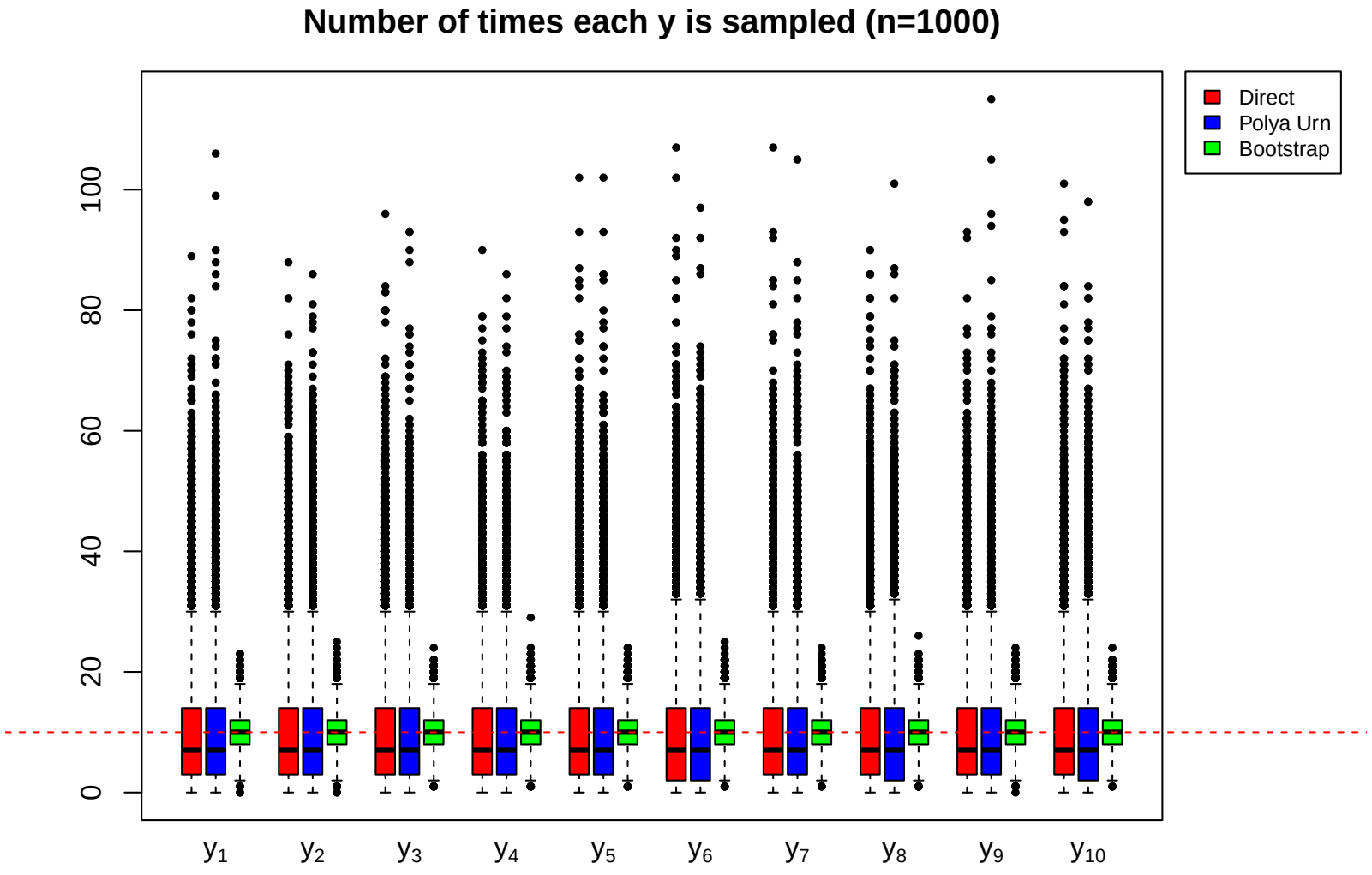
Contrast this with ordinary *bootstrap* resampling, which resamples y values with *deterministic* probabilities

$$\left\{\frac{1}{n}, \ldots \frac{1}{n}\right\}$$

```
set.seed(2413)
ZsampB<-matrix(0,nrow=nreps,ncol=n)
for(i in 1:nreps){
    Z<-sample(1:n,size=m,rep=T)
    ZsampB[i,]<-tabulate(Z,nbins=n)
}
```

```
all.Z<-cbind(Zsamp[,1],Zsamp0[,1],ZsampB[,1])
for(j in 2:n){all.Z<-cbind(all.Z,Zsamp[,j],Zsamp0[,j],ZsampB[,j])}
par(mar=c(5,4,3,6),xpd=TRUE,xaxt='n')
avec<-1:(4*n);avec<-avec[-4*(1:n)]
boxplot(all.Z,pch=19,cex=0.5,col=c('red','blue','green'),at=avec)
legend("topright", c("Direct", "Polya Urn",'Bootstrap'),cex=0.75,
            fill = c("red", "blue","green"),inset=c(-0.175,0))
nvec<-c(expression(y[1]),expression(y[2]),
            expression(y[3]),expression(y[4]),
            expression(y[5]),expression(y[6]),
            expression(y[7]),expression(y[8]),
            expression(y[9]),expression(y[10]))
text(2+c(0:(n-1))*4,rep(-50,10),nvec)
title('Number of times each y is sampled (n=10)')
abline(m/n,0,col='red',lty=2)
```


Number of times each y is sampled (n=10)
Direct
Polya Urn
Bootstrap
0
200
400
600
y1
y2
y3
y4
y5
y6
y7
y8
y9
y10


Number of times each y is sampled (n=1000)
Direct
Polya Urn
Bootstrap
0
20
40
60
80
100
y1
y2
y3
y4
y5
y6
y7
y8
y9
y10

The Bayesian nonparametric formulations always yield *more variable* samples, although the difference gets less marked as n increases.