Using the R package polyapost

As described in class the R package polyapost lets one simulate complete copies of a population give the values in the sample. Here will well give a couple of examples. More information can be found in the package vignette.

We begin by taking 20 polya draws from an urn which contains 5 balls with the values one through five. Each ball is given a weight of one.

```
> library(polyapost)
> set.seed(4747)
> x<-1:5
> out<-polyap(x,20)
> out

 [1] 1 2 3 4 5 1 2 2 2 2 5 2 5 2 5 2 1 4 4 4 5 3 5 5 4

> tabulate(out)/25

[1] 0.12 0.32 0.08 0.20 0.28
```

Note the final line gives us the proportion of balls of each type in the urn when we stop.

In class it was noted that when we take many polya draws from the urn the proportion of balls of each type in the urn, when we stop, follows a Dirichlet distribution. The R package gtools lets one simulate from this distribution.

```
> library(gtools)
> as.vector(rdirichlet(1,rep(1,5)))

[1] 0.136684210 0.123537677 0.002573571 0.245979135 0.491225406

> rdirichlet(2,rep(1,5))

          [,1]     [,2]      [,3]      [,4]      [,5]
[1,] 0.1046188 0.219989 0.1987861 0.1179756 0.3586305
[2,] 0.0660507 0.150971 0.2892068 0.2719058 0.2218657
```

Here we drew one such vector of the possible proportions followed by simulating the results for two such simulated populations.

When we want to give different weights to the balls in the urn we can use the function wtpolya. In the next bit of code we do this where we give ball i the weight of i. The last line does the same thing using the Dirichlet distribution

```
> wts<-5*(1:5)/sum(1:5)
> round(wts,digits=3)

[1] 0.333 0.667 1.000 1.333 1.667

> out<-wtpolyap(x,wts,20)
> out
```

```
 [1] 1 2 3 4 5 3 3 5 5 4 4 3 3 5 3 3 3 5 4 2 5 5 3 3 5
```

```
> tabulate(out)/25
```

```
[1] 0.04 0.08 0.40 0.16 0.32
```

```
> as.vector(rdirichlet(1,wts))
```

```
[1] 0.05509815 0.02145851 0.45475248 0.03465052 0.43404035
```

Suppose we have a random sample of size five from a large population and the value of the auxiliary variable, say x, associated with the units in the population, for our observed sample values, are one through five. Suppose that in addition we know that the population mean of x is 3.5 and that the proportion of the population that takes on the values 1 and 2 is 0.2. Let $p = (p_1, \ldots, p_5)$ be the unknown true proportion of each type of unit in the population. In this case we would like our simulated population to satisfy these known constraints. Here is one way to get approximate independent simulated copies of the population which satisfy the two constraints. We use the hitrun function. This function allows one to include equality and inequality constraints from prior information when making a large number of draws from the urn.

```
>  da2<-rbind(x,c(1,1,0,0,0))
>  db2<-c(3.5,0.2)
>  da1<-NULL
>  db1<-NULL
>  alpha<-rep(1,5)
>  dum<-hitrun(alpha,a1=da1,b1=db1,a2=da2,b2=db2,nbatch=2,blen=5000)
>  out<-dum$batch
>  out
```

```
           [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.09897366 0.1010263 0.2129798 0.3750668 0.2119534
[2,] 0.09963292 0.1003671 0.2121730 0.3760210 0.2118059
```

```
>  sum(da2[1,]*out[1,])
```

```
[1] 3.5
```

```
>  sum(da2[2,]*out[2,])
```

```
[1] 0.2
```

```
>  alpha<-wts
>  dum<-hitrun(alpha,a1=da1,b1=db1,a2=da2,b2=db2,nbatch=2,blen=5000)
>  out<-dum$batch
>  out
```

```
           [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.07750271 0.1224973 0.2460846 0.3303281 0.2235873
[2,] 0.08289912 0.1171009 0.2429530 0.3311949 0.2258521
```

```
>  sum(da2[1,]*out[1,])
```

```
[1] 3.5
```

```
>  sum(da2[2,]*out[1,])
```

```
[1] 0.2
```

In each case we checked to see that our constraints on the vector $p = (p_1, p_2, \dots, p_5)$, where p_i is the proportion of the units in the population that take on the value i, are satisfied for the first simulated copy of the population. What hitrun is doing here is generating a sequence of 10,000 dependent p vectors and keeping the 5,000th and 10,000th which we can treat as approximate independent draws from the distribution of interest.

One includes inequality constraints by using $a1$ and $b1$ in the hitrun function. In the next bit of code we we change the constraint that $p_1 + p_2 = 0.2$ to $p_1 + p_2 \leq 0.2$.

```
>  da2<-rbind(x)
>  db2<-c(3.5)
>  da1<-rbind(c(1,1,0,0,0))
>  db1<-c(0.2)
>  alpha<-rep(1,5)
>  dum<-hitrun(alpha,a1=da1,b1=db1,a2=da2,b2=db2,nbatch=2,blen=5000)
>  out<-dum$batch
>  out
```

```
           [,1]       [,2]      [,3]      [,4]      [,5]
[1,] 0.06999345 0.06868940 0.3281943 0.3575694 0.1755535
[2,] 0.07241511 0.06471793 0.3431569 0.3298720 0.1898381
```

```
>  sum(da2[1,]*out[1,])
```

```
[1] 3.5
```

```
>  sum(da1[1,]*out[1,])
```

```
[1] 0.1386828
```

Suppose we have our sample and the prior information in the constraints but no other information about the population. Then we could put the uniform distribution over the set of all p vectors which satisfy the constraints. Call this set Λ. The hitrun fuction lets us find the expected value of p under the uniform distribution on Λ. Let μ denote this vector of expected values. If N is the population size then $N\mu$ is a set of possible weights for the units in the sample. This is done by setting nbatch=1 and then hitrun returns the average of all 10,000 simulated p vectors.

```
>  N<-500
>  da2<-rbind(x)
>  db2<-c(3.5)
>  da1<-rbind(c(1,1,0,0,0))
>  db1<-c(0.2)
>  alpha<-rep(1,5)
>  dum<-hitrun(alpha,a1=da1,b1=db1,a2=da2,b2=db2,nbatch=1,blen=10000)
>  out<-colMeans(dum$batch) # blen=1 so just getting the overall mean
>  out

[1] 0.07096613 0.06628465 0.33578924 0.34570305 0.18125694

>  sum(da2[1,]*out)

[1] 3.5

>  sum(da1[1,]*out)

[1] 0.1372508

>  weights<-N*out
>  weights

[1]  35.48307  33.14233 167.89462 172.85152  90.62847

> sum(weights)

[1] 500
```

Instead of using the uniform distribution over Λ we could take any Dirichlet distribution and constrain it to live on Λ. To find the $E(p)$ under this distribution we just replace alpha in the above code by the parameter defining the constrained Dirichlet distribution.

```
>  N<-500
>  da2<-rbind(x)
>  db2<-c(3.5)
>  da1<-rbind(c(1,1,0,0,0))
>  db1<-c(0.2)
>  alpha<-wts
>  dum<-hitrun(alpha,a1=da1,b1=db1,a2=da2,b2=db2,nbatch=1,blen=10000)
>  out<-colMeans(dum$batch) # blen=1 so just getting the overall mean
>  out

[1] 0.04494239 0.06623318 0.41376490 0.29400107 0.18105845

>  sum(da2[1,]*out)

[1] 3.5
```

```
>  sum(da1[1,]*out)

[1] 0.1111756

>  weights<-N*out
>  weights

[1]  22.47120  33.11659 206.88245 147.00053  90.52923

> sum(weights)

[1] 500
```