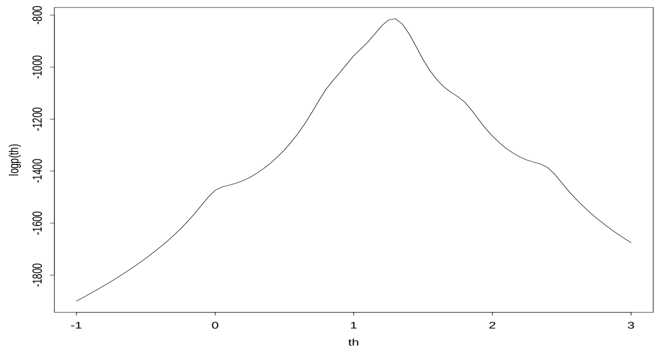
36-711 – FingerEx 13 Solutions

1

(a) $p(\theta|Y) \propto f(y|\theta)\pi(\theta) = \frac{1}{\pi s}\frac{1}{1+(\frac{y-\theta}{s})^2}\frac{1}{\sqrt{2\pi}}e^{-\frac{(\theta-1)^2}{2}}$
$\log p(\theta|Y) = -\sum_i \log(1+(\frac{y_i-\theta}{s})^2) - \frac{(\theta-1)^2}{2}$, maximized at $\tilde{\theta} = 1.284972$.



(b) Use `info.mtx.fun`, we can compute the SE of $\tilde{\theta} = 0.05365408$, hence an approximate 95% credible interval for θ is $(1.12401, 1.445935)$.

(c) Normalized integral of $p(\theta|Y)$ over the interval in part 1b is 0.9653856.

(d) $\mathrm{E}[\theta|Y] = 1.272403$, and $\sqrt{\mathrm{Var}[\theta|Y]} = 0.08608301$. The approximate 95% interval is $(1.014154, 1.530652)$, which is a little bit wider than that in part 1b.

(e) The numerical integration failed since repeating observations twice. This is because with repeated observations the likelihood is very peaked. We need to either adjust the step length in `integrate()`, or transform the likelihood to make it behave nicely in the interval.

2

(a) Define $p_i \equiv P[X = i]$, $i = 0, 1, 2, ...$, it's easy to see $p_i = F(i) - F(i-1)$. By construction, the returned value $\xi = k$ if found some k s.t. $F(k-1) < U \leq F(k)$.
Since $P[\xi = i] = P_u[F(i-1) < U <= F(i)] = F(i) - F(i-1) = p_i$ for all i, $\xi \sim F(x)$.

(b)(c) See attached output. Two disadvantages:

- This "search" method is slow since you have to search for the right k. Indeed, you have to wait a Poisson length of time, for each Poisson observation.
- When U is close to 1, the floating point representation will tend to allow U to match a value i that is too small. [i.e. the upper tail of the "simulated" Poisson distrib will be artificially truncated.]

3

See the simple proof on Monahan p.281. The number of samples until an "accept" is a geometric random variable with parameter $p = 1/a$, where a is some constant greater than 1. Hence on average, we need a pairs of (Y, U) to get one draw from the target density. Apparently we want a to be close to 1.

4

The density of $Cauchy(0,\sigma)$ is $g(x) = \frac{\sigma}{\pi}\frac{1}{\sigma^2+x^2}$, the density of $N(0,1)$ is $f(x) = \frac{1}{\sqrt{2\pi}}e^{x^2/2}$.
$\phi(x) = \frac{f(x)}{g(x)} = \sqrt{\frac{\pi}{2}}\frac{\sigma^2+x^2}{\sigma}e^{-\frac{x^2}{2}}$. Let $\phi'(x) = 0$, we have $x = 0$ or $x = \pm\sqrt{2-\sigma^2}$.
The possible extreme values are at $x = 0$ and $\sigma \geq \sqrt{2}$, $\phi(x) = \sqrt{\frac{\pi}{2}}\sigma$; or at $x = \pm\sqrt{2-\sigma^2}$ and $\sigma < \sqrt{2}$, $\phi(x) = \frac{\sqrt{2\pi}}{e\sigma}\cdot e^{\frac{\sigma^2}{2}}$. So $\frac{f(x)}{g(x)}$ has an upper bound C_σ.
Let $a = \text{argmin}_\sigma C_\sigma$, then we have $f(x) \leq a \cdot g(x)$, hence rejection method can be used.

Since $\lim_{x\to\infty}\phi(x) = 0$, $\frac{g(x)}{f(x)}$ cannot be bounded by a constant, the other way doesn't work.

5

Clearly, we can choose a discrete distrubtion $g(x)$, such that $p(x_i) \leq a \cdot g(x_i)$ for all x_i. Then generate Y from $g(x)$ and U from $Unif[0, a\cdot g(Y))$, if $U < p(Y)$ then accept Y.

To use a continuous $g(x)$ we might reformulate the problem as follows: First, consider the case that X has integer values. Let Z be the continuous random variable $Z = X + U$, where U is an independent Unif$[-\frac{1}{2}, +\frac{1}{2}]$ random variable. Then the density $f_Z(x)$ for Z looks like a barplot for the probabilities $p(i) = P[X = i]$. Find a convenient, continuous density $g(x)$ such that $f_Z(x) \leq ag(x)$ for all x. Simulate Z using the rejection method on samples from $g(x)$; let X be Z rounded to the nearest integer. Then (you can show) that X is discrete with density $f(x)$.

If X is not integer valued, a similar trick works, but now the range of U also depends on the value of X (this doesn't affect the basic method, which involves finding $g(x)$ to dominate the "barplot" $f_Z(x)$ of $f(x)$, and then "rounding" rejection-method draws of Z to the nearest value of X).

6

Let $\theta = x/y$, where $x = \frac{1}{n}\tilde{I}_n$, and $y = \frac{1}{n}\tilde{O}_n$. By delta method, $Var(\hat{I}_n) = \nabla g(\theta)Var(\hat{\theta})\nabla g(\theta)^T \approx$
$(1/y, -x/y^2)\begin{pmatrix} Var(\tilde{I}_n) & Cov(\tilde{I}_n,\tilde{O}_n) \\ Cov(\tilde{I}_n,\tilde{O}_n) & Var(\tilde{O}_n)\end{pmatrix}\begin{pmatrix} 1/y \\ -x/y^2\end{pmatrix}|_{y=1,x=E_f[h(X)]}$
$= Var(\tilde{I}_n) - 2E_f[h(X)]Cov(\tilde{I}_n,\tilde{O}_n) + E_f^2[h(X)]Var(\tilde{O}_n)$.
By making $Cov(\tilde{I}_n,\tilde{O}_n)$ large and $Var(\tilde{O}_n)$ large we may possibly make
$E_f^2[h(X)]Var(\tilde{O}_n) - 2E_f[h(X)]Cov(\tilde{I}_n,\tilde{O}_n) \leq 0$, therefore $Var(\hat{I}_n) \leq Var(\tilde{I}_n)$.

Splus source code

```
##Q1(a)

s_0.1
PI_3.1415926535
y_Y_c(1.2, 2.4, 1.3, 1.3, 0.0, 1.0, 1.8, 0.8, 4.6, 1.4)

logp_function(th){
    ans_0
    for(i in 1:length(y)){
        ans_ans - log(1+((y[i]-th)/s)^2)
    }
    ans_ans - (th-1)^2/2
}

nlogp_function(th){
    -logp(th)
}

th_seq(-1,3,.05)
plot(th, logp(th), type="n")
lines(th, logp(th), type="l")

##Q1(b)

th0_nlmin(nlogp, 1)$x
> th0
[1] 1.284972

se_sqrt(1/info.mtx.fun(logp, th0))
> se
[1,] 0.05365408

##Q1(c)

p_function(th){   ## posterior p(\theta|Y)
    exp(logp(th))
}

plot(th, p(th), type="n")
lines(th, p(th), type="l")

## normalized integral
n_integrate(p, -1, 3)$integral
integrate(p, 1.12401, 1.445935)$integral/n

## Q1(d)

n.p_function(th) { p(th)/n }        ## normalized p(\theta|Y)
npx_function(th) { n.p(th)*th }     ## p(x)*x
npxx_function(th) { n.p(th)*th^2 }  ## p(x)*x^2
npv _function(th) {n.p(th)*(th-E)^2 } # p(x)*(x-E)^2

E_integrate(npx, -1, 3)$integral
> E
[1] 1.272403

V_integrate(npxx, -1, 3)$integral - E^2
> V
[1] 0.007410285

## Q1(e)

r_c(1,2,3,5,10,20,30)

for(j in 1:length(r)){
```

```
    y_rep(Y, r[j])
    E_integrate(npx, -1, 3)$integral
    V1_integrate(npxx, -1, 3)$integral - E^2
    V2_integrate(npv,-1,3)$integral
    print(c(E, sqrt(V1),sqrt(V2)))
}


[1] 1.27240330 0.08608301 0.07034798
[1] 1.596371e-12 1.417894e-06 1.417894e-06
[1] 2.358513e-24 1.722881e-12 1.722881e-12
[1] 5.330201e-48 2.589960e-24 2.589960e-24
[1] 4.113901e-107  7.195287e-54  7.195287e-54
[1] 2.450719e-225 5.553515e-113 5.553515e-113
[1] 0 0 0

########################################

## Q2(b) function to generate poisson draws

random.poisson_function(n, lambda){  ## generate n poisson draws
with parameter lambda
    u_runif(n, 0, 1)
    for(i in 1:n){
        k_0
        while(1) {
            if(ppois(k-1, lambda)<u[i] && u[i]<=ppois(k, lambda)) {
                u[i]_k
                break
            }
            k_k+1
        }
    }
    return(u)
}

> random.poisson(10, 1)
 [1] 3 3 0 2 1 1 3 2 1 1
```