Computer Lab 3

Shipeng Liu (shili506)   Dongwei Ni (donni508)

Question 1: Stable distribution

Task 1

Plot $f(x)$ and $fp(x)$ together

```r
x=seq(0,10,0.05)

#one-sided strictly stable distribution of order 1/2 where c=2
f=function(x){
  c=2
  sapply(x,function(y){
    if(y>0){
      return(c*sqrt(2*pi)^(-1)*exp(-(c^2)/(2*y))*(y^(-3/2)))
    }else{
      return(0)
    }
  })
}


y_f=f(x)

#power-law distribution where a=1.5 and t_min=4/3
fp=function(x){
  a=1.5;t_min=4/3
  sapply(x,function(y){
    if(y>t_min){
      return(((a-1)/t_min)*(y/t_min)^(-a))
    }else{
      return(0)
    }
  })
}
y_fp=fp(x)

df=rbind(data.frame("x"=x,"y"=y_f,"class"="f"),data.frame("x"=x,"y"=y_fp,"class"="fp"))



p1=ggplot(data=df,aes(x=x,y=y))+geom_point(aes(color=class))+
  labs(tag="Fig. 1")+
  scale_color_discrete(
    name="distribution",
    labels=c("one-sided strictly stable distribution","power-law distribution")
```
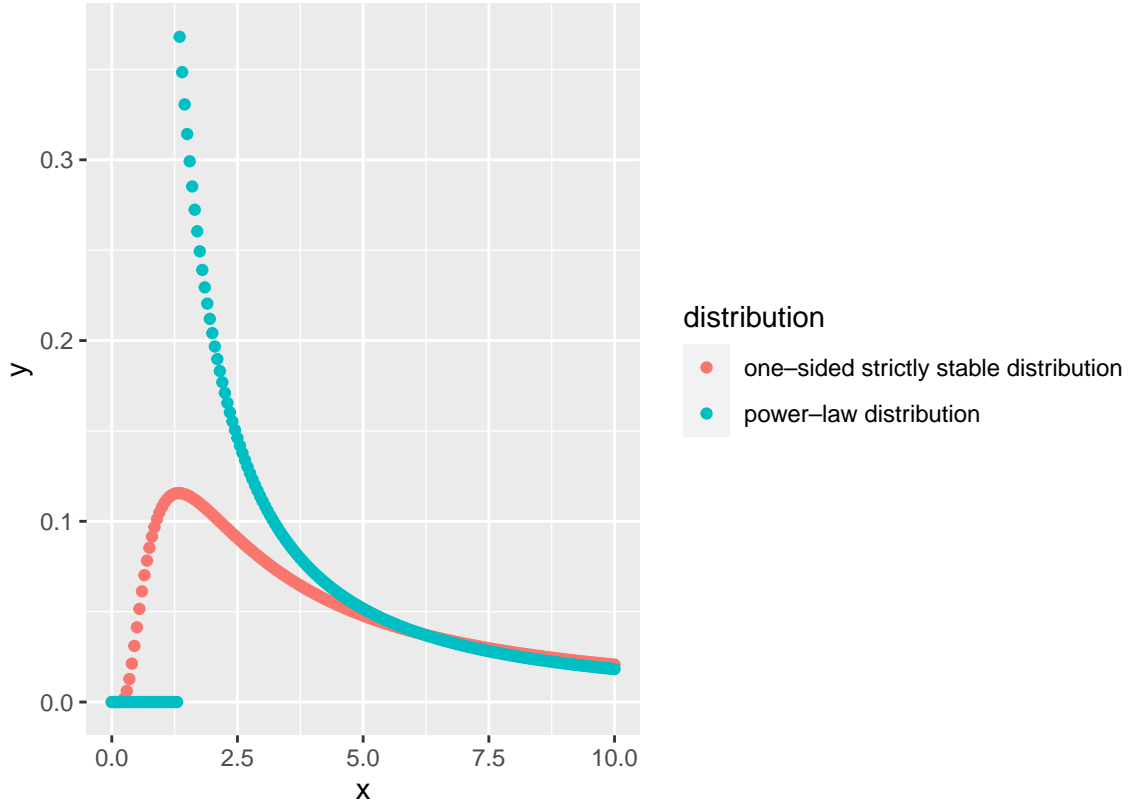
```
  )
p1
```

Fig. 1

The power–law distribution can not be used just by itself,because the two distribution don't have the same support. In $(0, t_{min})$,the density of the power–law distribution is 0,but for one–sided strictly stable distribution of order $1/2$ it isn't.The power–law distribution can't generate the proper samples in accept-reject method in this interval. We can apply a uniform distribution on the interval $(0, t_{min})$ as a majorizing function

For optimal Tmin: take the derivative of target density function

$$\frac{df(x)}{dx} = (\sqrt{2\pi}^{-1} e^{-\frac{c^2}{2x}} (\frac{c^2}{2x} x^{-3/2} - \frac{3}{2} x^{-5/2}))$$

take the root, we got

$$max f(x) = f(\frac{1}{3}c^2)$$

so the optimal $T_{min} = \frac{1}{3}c^2$

For optimal α: we need to make sure $\forall_x a f_p(x) \geq f(x)$ (a is a constant) hence we should take an appropriate value of a
s.t. the difference between $f(x)$ and $f_p(x)$ will not be large in the interval(T_{min}, ∞) according to the plot, we set $\alpha = 1.5$

Task 2

Implement an acceptance-rejection algorithm for sampling from the one-sided strictly stable distribution of order 1/2 with the proposal distribution built around the power-law distribution.

```r
sampling=function(samp_x,constant,c,t_min,alpha){
  a=sapply(samp_x,function(samp_x){
    x=NA
    while(is.na(x)){
      u=runif(1)
      if(samp_x>=0 && samp_x<=t_min+0.5){
        y=runif(1)
        if(u<=f(y)/constant*dunif(y)){x<-y}
      }else{
        y=rplcon(1,t_min,alpha)
        if(u<=f(y)/constant*dplcon(y,t_min,alpha)){x<-y}
      }
    }
    return(x)
  })
  return(a)
}

t_min=4/3
alpha=1.5
c=2
constant=1

samp_x=runif(2000,0,10)
samp=sampling(samp_x,constant,c,t_min,alpha)
p2=ggplot(data.frame("x"=samp),aes(x=x))+geom_density()+xlim(c(0,10))+
  labs(title="The sampler density",tag="Fig. 2")
df_f=df%>%filter(class=="f")
p3=ggplot(data=df_f,aes(x=x,y=y))+geom_line()+
  labs(title="The target density",tag="Fig. 3")
plot_comb=grid.arrange(p2,p3)
```
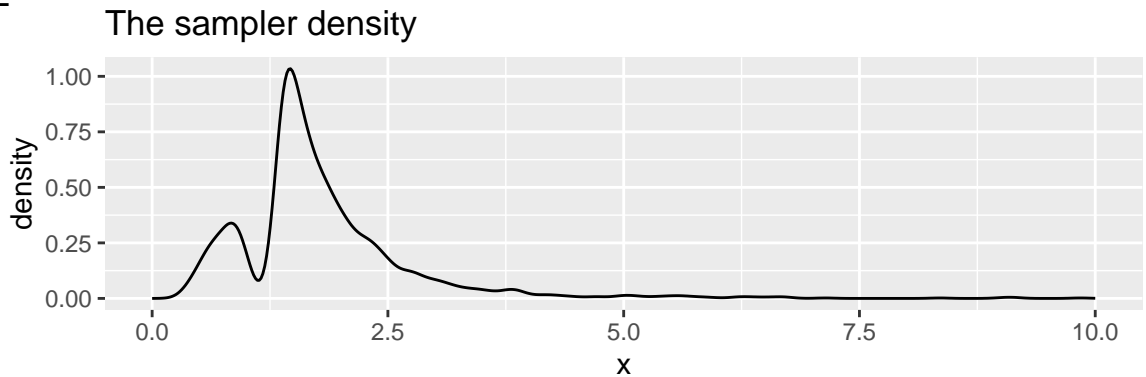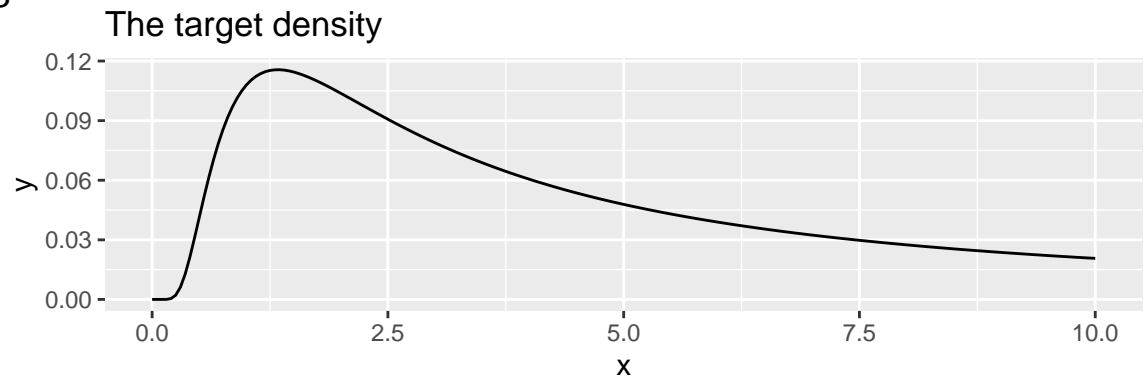
Fig. 2

The sampler density

Fig. 3

The target density


```
plot_comb
```

```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells     name            grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (2-2,1-1) arrange gtable[layout]
```

Task 3

Question 2: Laplace distribution

Task 1

Write a code generating double exponential distribution DE(0, 1) from Unif(0, 1) by using the inverse CDF method.

The PDF of the Laplace distribution is given:

$$f(x) = \frac{\alpha}{2} exp(-\alpha \mid x - \mu \mid)$$

And we calculate the CDF:

$$F(x) = \int_{-\infty}^{x} f(u)du = \begin{cases} \frac{1}{2}e^{(\frac{x-\mu}{\alpha})} & \text{if } x < \mu \\ 1 - \frac{1}{2}e^{(\frac{x-\mu}{\alpha})} & \text{if } x \geq \mu \end{cases}$$

The inverse of CDF:

$$F^{-1}(x) = \begin{cases} -\frac{1}{\alpha} \ln 2(1 - F(x)) + \mu & \text{if } x \geq \mu \\ \frac{ln(2F(x))}{\alpha} + \mu & \text{if } x < \mu \end{cases}$$

4

when $x = \mu, F(x) = \frac{1}{2}$ hence The inverse of CDF:

$$F^{-1}(x) = \left\{ \begin{array}{ll} -\frac{1}{\alpha}\ln 2(1 - F(x)) + \mu & \text{if } F(x) \geq \frac{1}{2} \\ \frac{\ln(2F(x))}{\alpha} + \mu & \text{if } F(x) < \frac{1}{2} \end{array} \right.$$

```r
r_laplace=function(gen_num,mu,alpha){
  unif=runif(gen_num,0,1)
  temp=sapply(unif,function(unif){
    if(unif>=0.5){
      return((-1/alpha)*log(2*(1-unif))+mu)
    }else{
      return((log(2*unif)/alpha)+mu)
    }
  })
  return(temp)
}
```

Generate 10000 random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

```r
p4=ggplot(data=data.frame("y"=r_laplace(10000,0,1)),aes(x=y))+
  geom_histogram(bins=50)+
  labs(title="Laplace distribution generate by Inverse CDP method")
p4
```


Laplace distribution generate by Inverse CDP method
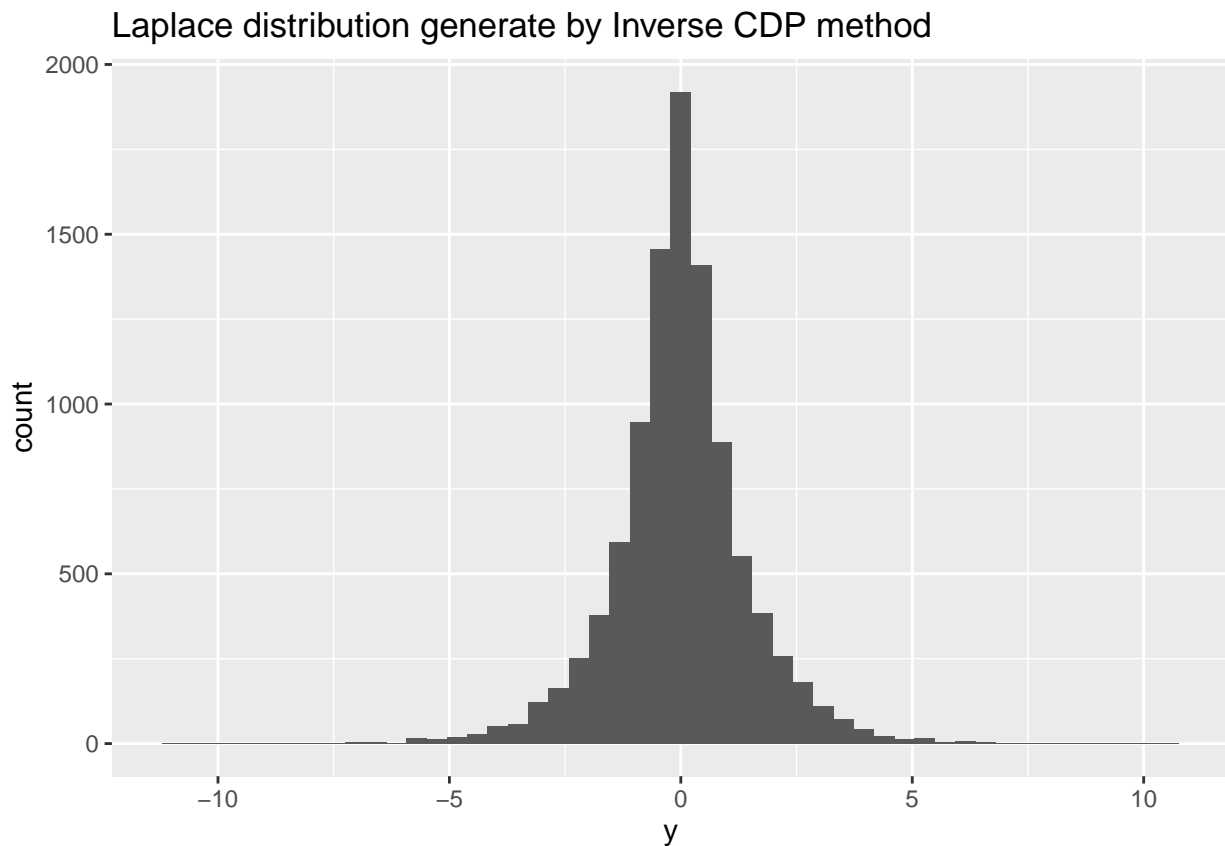
Compare to the plot of Laplace distribution probability density function(it can be thought of as two exponential distributions spliced together along the abscissa),the result seems resonable.

Task 2

Use the Acceptance/rejection method with DE(0, 1) as a majorizing density to generate N (0,1) variables.

```r
ar_norm<-function(c){
  x<-NA
  num.reject<-0
  while (is.na(x)){
    y<-r_laplace(1,0,1)
    u<-runif(1)
    if (u<=dnorm(y,0,1)/(c*(exp(-1*abs(y))/2))){x<-y}
    else{num.reject<-num.reject+1}
  }
  c(x,num.reject)
}
```

We sample Y from the majorizing distribution, sample U from the uniform distribution, and then filter out samples that satisfy $U \leq \frac{f_X(Y)}{cf_Y(Y)}$, and resample if not satisfied , until a sample that satisfies the condition is obtained.

majorizing constant c

We need to choose majorizing constant c such that:

$$\forall_x cf_Y(x) \geq f_X(x)$$

where $f_Y(x)$ is PDF of Laplace distribution,$f_X(x)$ is PDF of Normal distribution. c mush be large enough,but too large c may cause large rejection rates,we need to choose carefully.

$$\frac{c}{2}e^{-|x|} \geq \frac{1}{\sqrt{2\pi}}e^{|x|-\frac{x^2}{2}}$$

solve it and we'll get:

$$c \geq \sqrt{\frac{2}{\pi}}e^{(|x|-\frac{x^2}{2})}$$

when$x = 1$,we get the maximum c:

$$c = \sqrt{\frac{2e}{\pi}}$$

Generate 2000 random numbers N (0, 1) using your code and plot the histogram.

```r
df_norm=data.frame(t(data.frame(sapply(rep(sqrt((2*exp(1))/pi),2000),ar_norm))))
colnames(df_norm)=c("sample","reject")
p4=ggplot(data=df_norm,aes(x=sample))+
  geom_histogram(bins=50)+
  labs(title="Normal distribution generate by Acceptance/rejection method")
p4
```

Normal distribution generate by Acceptance/rejection method
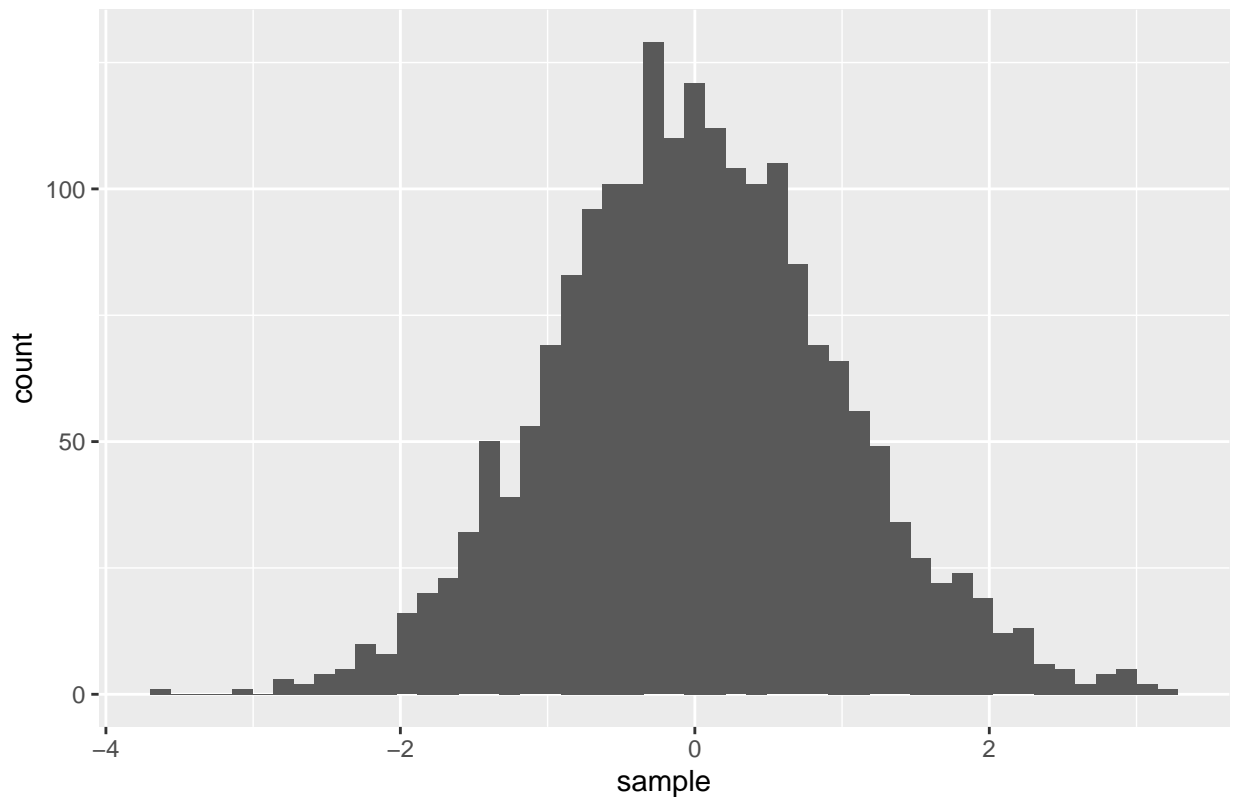
Rejection rate

```
mean_r=sum(df_norm[,2])/(2000+sum(df_norm[,2]))
ER=1-(1/sqrt((2*exp(1))/pi))

cat("The average rejection rate R: ",mean_r,"\nThe expected rejection rate ER: ",
    ER,"\nThe difference:",abs(ER-mean_r))
```

```
## The average rejection rate R:  0.2337165
## The expected rejection rate ER:  0.2398265
## The difference: 0.006110074
```
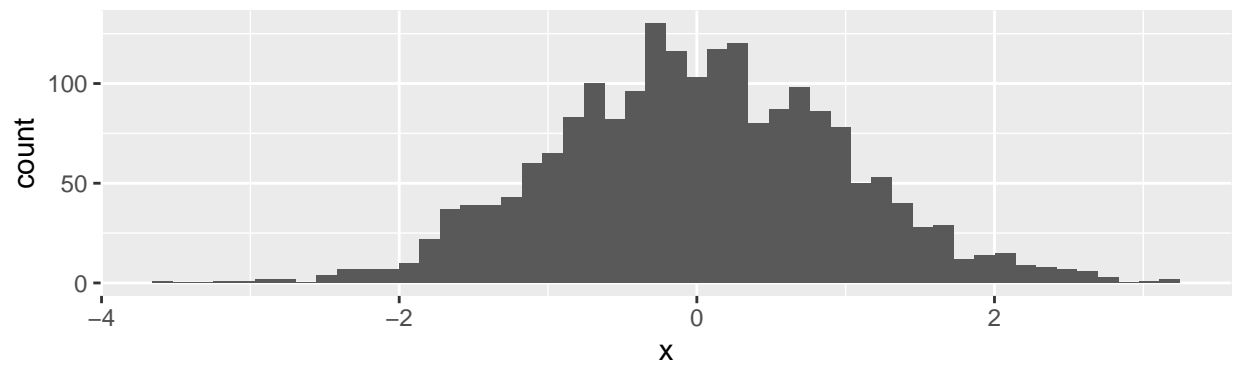
The R and ER are very close to each other.

Generate 2000 numbers from N (0, 1) using standard rnorm() procedure

```
p5=ggplot(data=data.frame("x"=rnorm(2000,0,1)),aes(x=x))+
  geom_histogram(bins=50)+
  labs(title="Normal distribution generate by rnorm")
plot_comb1=grid.arrange(p4,p5)
```

Normal distribution generate by Acceptance/rejection method
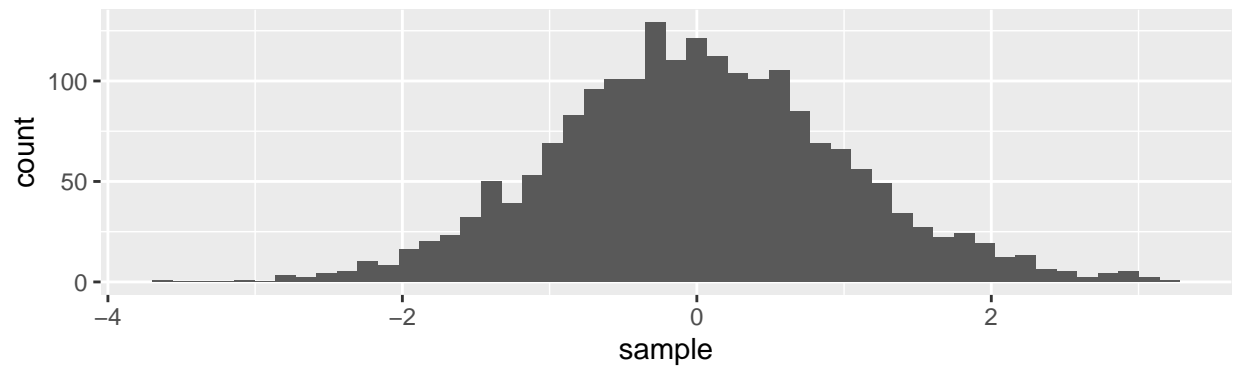
Normal distribution generate by rnorm

The obtained two histograms has similar distribution. The number of samples concentrated around 0 is the largest, and the number of samples decreases as the absolute value of the sample increases.