Notes on the Book: Introducing Monte Carlo Methods with R by Christian P. Robert and George Casella

John L. Weatherwax*

June 20, 2008

*wax@alum.mit.edu

Chapter 1 (Basic R Programming)

Introduction

To experiment with some of the `R` commands presented in this chapter I duplicated some of the code presented in the book in the `R` file `chap_1_code_examples.R`.

Exercise 1.3

We find

- seq: Generate regular sequences.
- sample: takes a sample of the specified size from the elements of x using either with or without replacement.
- order: returns a permutation which rearranges its first argument into ascending or descending order, breaking ties by further arguments.

Exercise 1.4

The function `order` returns the order in which we would need to take the elements of `x` to get them in a sorted order. The function `rank` returns the rank (spot in the sorted list) of each element of `x`. An example will help clarify

```
> x = runif(5)
> x
[1] 0.7887347 0.7654628 0.8868125 0.2918536 0.1194615
> order(x)
[1] 5 4 2 1 3
> x[order(x)] # is a sorted list
[1] 0.1194615 0.2918536 0.7654628 0.7887347 0.8868125
> rank(x) # gives the spot in the sorted list of each element of x
[1] 4 3 5 2 1
```

An example of the usage for the various options for `rep` will clarify its options

```
> x = c(1,2,3)
> rep(x,times=5)
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
```

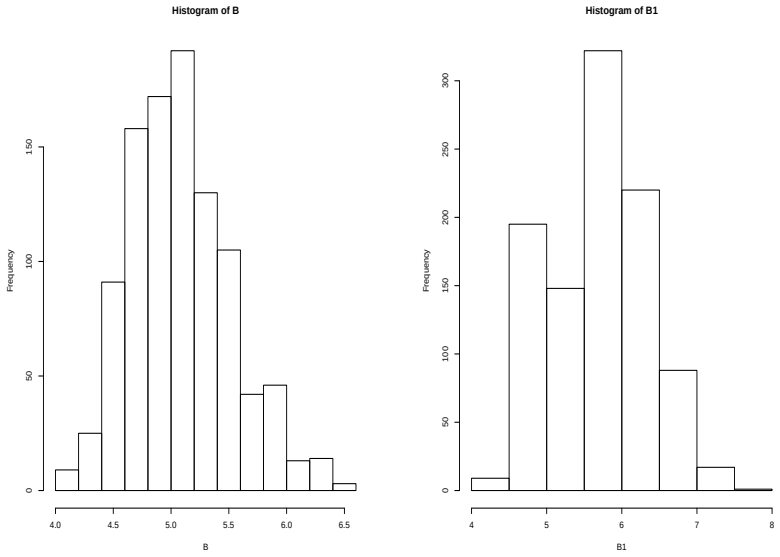

Figure 1: **Left:** The bootstrap samples of $\bar{y}$. **Right:** The bootstrap samples of $q_{0.95}(\bar{y})$.

```
> rep(x,length.out=5)
[1] 1 2 3 1 2
> rep(x,each=5)
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
```

Exercise 1.7 (bootstrap examples)

Part (a): See the `R` code in `ex_1_7.R`. When that code is run it generates the plot shown in Figure 1. We generate bootstrap samples by sampling with replacement 8 values from the y vector. We then compute the mean of that sample. We do this 1000 times generating 1000 values for $\bar{y}$, that are plotted in Figure 1 (left). From all of these values we can compute the 95% percentage point using the `R` function `quantile`. This gives the value 5.906.

Part (b): To get a confidence interval for $q_{0.95}(\bar{y})$ we need to do *two* bootstrap sample generation loops. For each pass through the outer loop we will compute a *single* sample of $q_{0.95}(\bar{y})$ in the same way as Part (a) above. We do this by first generating a bootstrap sample `ystar` from the original data `y` and then from this fixed sample we generate additional bootstrap samples on which we can compute the sample mean $\bar{y}$. Once we have a large number of bootstrapped samples of $\bar{y}$ we can compute the 95% quantile of these samples and save that number as a single sample for the outer bootstrap procedure. Once we have many samples of $q_{0.95}(\bar{y})$ we can plot all of them in a histogram. This plot is shown in Figure 1 (right). A 95% confidence interval for $q_{0.95}(\bar{y})$ can be obtained by computing the 0.025 and 0.975 percentage point of the outer bootstrap samples of $q_{0.95}(\bar{y})$. When we do that we find

```
[1] "the bootstrapped estimated confidence interval for q_{0.95} is"
```

```
[1] "(  4.613294,  6.915125)"
```

Exercise 1.8 (the bootstrap with linear models)

For this problem see the `R` code in `ex_1_8.R`. When we use the `R` code `lm` to get the t based standard error. This is extracted from the output of the `lm` command by using the `summary` command. For this problem we get

```
> fitsum$coefficients
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 2.669054  0.4023548  6.633582 0.0069789584
x           3.967274  0.1896719 20.916513 0.0002390236
```

Thus for the intercept we have $\hat{\beta}_0 = 2.6690$ and a t based standard error given by $\text{se}(\hat{\beta}_0) = 0.402$, while for the slope we have $\hat{\beta}_1 = 3.967274$ with a standard error of 0.1896719. From these numbers a $(1-\alpha)\times 100\%$ confidence interval for β_0 is given by [1]

$$\hat{\beta}_0 - t(\alpha/2, n-2)\text{se}(\hat{\beta}_0) \leq \beta_0 \leq \hat{\beta}_0 + t(\alpha/2, n-2)\text{se}(\hat{\beta}_0)\,,$$

here $t(\alpha/2, d)$ is the value that cuts off $\alpha/2 \times 100\%$ in the upper tail of the t-distribution with d degrees of freedom. The same type of expression hold for the confidence interval for β_1 as long as we use the correct expression/value for $\text{se}(\hat{\beta}_1)$. When we do that we find that

```
[1] "the t-based 0.90% confidence interval beta_0 is (  1.722167,  3.615941)"
[1] "the t-based 0.90% confidence interval beta_1 is (  3.520907,  4.413641)"
```

While the bootstrapped based confidence intervals are given by

```
> quantile( B[,1], c(0.05,0.95) ) # for beta_0
      5%      95%
2.138569 3.136529
> quantile( B[,2], c(0.05,0.95) ) # for beta_1
      5%      95%
3.727029 4.212291
```

These bootstrapped confidence intervals seem to be considerable tighter than their t distribution based alternative.

Exercise 1.10

The command `help(xor)` gives a help on these logical operators. Both & and && perform the logical "and" operation. The character | and || perform the logical or operation. These

commands differ mainly in what they return when operating on vector operands. The single forms returns a *vector* where the logical operation (and/or) is applied componentwise. The double form for example the expression, &&, returns a single logical value that depends on the individual elements in the vector. Because of this, the single character version is typically used for index extraction, while the double character version is typically used for logical expressions (i.e. "if" statements). An example will help clarify,

```
> # fix the random.seed, so I'll always get the same answer:
> set.seed(10131985)
>
> x = rnorm(10)
> x
 [1]  1.3034072 -0.4053758  1.4895198  0.2048860  0.7528309 -0.2749962
 [7]  1.3107793 -0.7214373  0.3383096  0.1270985
>
> indsLT = x < -0.25 # less than -0.25
> indsLT
 [1] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
>
> indsGT = x > +0.25 # greater than +0.25
> indsGT
 [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
>
> indsLT | indsGT # all points such that abs(value)>0.25
 [1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
> indsLT & indsGT # no points (all elements FALSE)
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>
> # return "or" of first element (indsLT[1]=TRUE) *or* (indsGT[1]=FALSE) = TRUE
> indsLT || indsGT
[1] TRUE
>
> # return "and" of first element (indsLT[1]=TRUE) *and* (indsGT[1]=FALSE) = FALSE
> indsLT && indsGT
[1] FALSE
```

Notice how the double logical expressions only return a single logical value (and not a vector). This example is worked in the `R` code `ex_1_10.R`.

Exercise 1.11

The `attach` command allows the user to have variable names searched for in the database specified when using the `attach` command. The `assign` command allows the user to specify a variable as a string and then assign a variable to this string.

Exercise 1.12

This exercise is worked in the R code ex_1_12.R.

Exercise 1.13

1. capture.output produces a string that represents what the R interpreter would produce when the given command is executed.
2. dput produces an ASCII representation of an R object to a file or a connection.
3. dump produces an ASCII representation from a vector of names.
4. save write several R object to a file.
5. sink writes R output to a connection.
6. write writes columns of data to a file.

Some examples will help clarify. To consider these examples we need some data. We will consider the following example

```
x = rnorm(20)
y = 3*x + 5 + rnorm(20,sd=0.3)
reslm = lm(y~x)
```

Then the command capture.output produces

```
> capture.output(summary(reslm))
 [1] ""
 [2] "Call:"
 [3] "lm(formula = y ~ x)"
 [4] ""
 [5] "Residuals:"
 [6] "     Min       1Q   Median       3Q      Max "
 [7] "-0.46481 -0.11923 -0.04518  0.14757  0.66035 "
 [8] ""
 [9] "Coefficients:"
[10] "            Estimate Std. Error t value Pr(>|t|)    "
[11] "(Intercept)  5.04771    0.06823   73.98   <2e-16 ***"
[12] "x            3.03041    0.09980   30.36   <2e-16 ***"
[13] "---"
[14] "Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1 "
[15] ""
```

```
[16] "Residual standard error: 0.291 on 18 degrees of freedom"
[17] "Multiple R-squared: 0.9809,\tAdjusted R-squared: 0.9798 "
[18] "F-statistic:   922 on 1 and 18 DF,  p-value: < 2.2e-16 "
[19] ""
```

The command `dput` would give (truncated for space)

```
> dput(reslm)
structure(list(coefficients = structure(c(5.04771134844566, 3.03041491288024
), .Names = c("(Intercept)", "x")), residuals = structure(c(-0.141513023558070,
0.313560591005267, -0.464808029425501, 0.420523983312902, 0.302041310583233,
-0.209328782477972, 0.156642822808795, -0.0690696235004164, 0.0427729747192703,
-0.0864893938662702, -0.0212822529638138, -0.080798389233946,
-0.110494650289592, 0.0571855672545044, -0.406005294280871, 0.660353066118607,
0.0300317438264137, 0.144544712400732, -0.426070608489334, -0.111796723943939
), .Names = c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"11", "12", "13", "14", "15", "16", "17", "18", "19", "20")),
    effects = structure(c(-25.3620821095076, 8.8356206498652,
```

The command `dump` creates a file called `dumpdata.R` which could be soured to get the output variables into to the workspace. For example entering the command `dump(‘‘reslm’’)` produces the above file that has contents given by

```
reslm <-
structure(list(coefficients = structure(c(5.04771134844566, 3.03041491288024
), .Names = c("(Intercept)", "x")), residuals = structure(c(-0.141513023558070,
0.313560591005267, -0.464808029425501, 0.420523983312902, 0.302041310583233,
-0.209328782477972, 0.156642822808795, -0.0690696235004164, 0.0427729747192703,
-0.0864893938662702, -0.0212822529638138, -0.080798389233946,
-0.110494650289592, 0.0571855672545044, -0.406005294280871, 0.660353066118607,
0.0300317438264137, 0.144544712400732, -0.426070608489334, -0.111796723943939
), .Names = c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
```

The `R` command `source('dumpdata.R')` will then bring the variables back into the workspace if needed.

The command `save` will save all `R` object into a file

```
save(file="r_workspace.dat")
```

produces a binary file that can be loaded into the workspace with the command `load`.

Exercise 1.14

The command `var` has an option `na.rm`, which if true will disregard rows (vector observations) that have any element which is NA. This is not the same issue that arises when we request the value of `var(1)`. In that case the denominator of the estimator is $n-1$ which with one sample results in a zero denominator giving the NA result. There is an option to the `lm` and `glm` commands called `na.action` that specifies what the commands should do when it finds an observation with a NA in it. The default action is `na.fail` which means the command will fail.

Exercise 1.15

In the `R` code `ex_1_15.R` we do some experiments with the commands `match` and `which`.

Exercise 1.16

The command `identical` is designed to test object for equality and for use in an logical (i.e. "if") statement. The command `all.equal` is used to test approximate equality.

Exercise 1.17

This exercise is worked in the `R` code `ex_1_17.R`. When we run that command we get

```
> N=100000
>
> y=c();
> system.time(for (t in 1:N){ y[t]=exp(t) })
   user  system elapsed
 54.391   0.000  54.635
>
> system.time(exp(1:N))
   user  system elapsed
  0.008   0.000   0.010
>
> system.time(sapply(1:N,exp))
   user  system elapsed
  0.409   0.000   0.410
```

Showing that the second form is the fastest form.

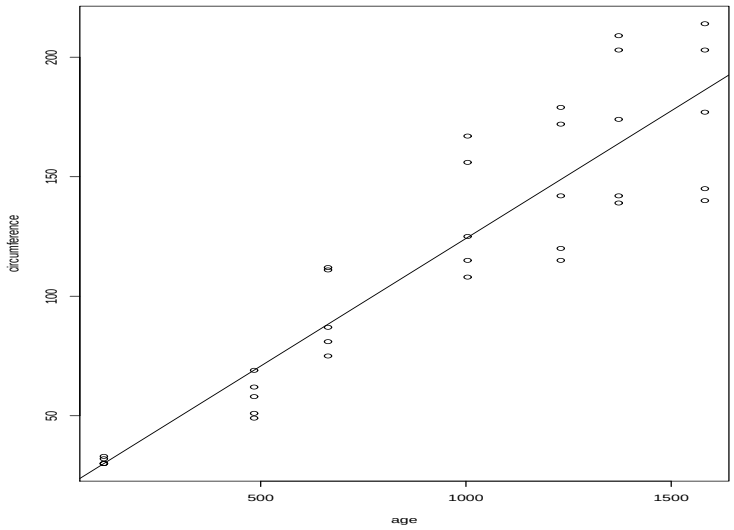

Figure 2: The requested linear regression for Exercise 1.21.

Exercise 1.18

The stated commands return objects that can be modified and can be used as the left-hand-side of an expression.

Exercise 1.19

This exercise is worked in the `R` code `ex_1_19.R`. In that command we compute powers of a matrix who's rows sum to one. When I run the `R` code for this problem I get a quite stable result in that quite large powers of the base matrix seems to have rows that sum to one. This is in contrast to the answer that the book gives. If anyone sees anything wrong with what I've done for this problem please contact me.

Exercise 1.21

This exercise is worked in the `R` code `ex_1_21.R`. In that command we compute two linear regressions one based on simply the predicting `circumference` given `age` and another regression where we allow a factor predictor based on the actual tree. When we plot this first linear regression we get the result shown in Figure 2. We can then compare the two models using the `R` command `anova`. When we do that we find that we find

```
> anova(m1,m2)
Analysis of Variance Table
```

```
Model 1: circumference ~ age
Model 2: circumference ~ age + TF
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     33 18594.7
2     29  6753.9  4   11840.9 12.711 4.289e-06 ***
```

This table shows that by adding the tree as a factor we have restricted the degrees of freedom of the residuals by 4 but reduced the residual sum of squares by 11840.9 the probability that this might happen "by chance" is `Pr(>F)=4.289e-06`, thus this seems to be a modeling improvement i.e. it is rather unlikely that this change happened by chance.

Exercise 1.22 (further applications of the bootstrap)

Part (a): Warning: I thought this problem was a simple extension of the bootstrap for linear regression example presented in the book. The book says to sample the data set with replacement. The problem with this statement is that the data is a time series and as such we must sample the sequence *in the same order*. If we simply sample using the `R` command `sample` and then call the `acf` function we will end up computing autocorrelation functions on a different time series each time, thus there is no reason why the autocorrelation values (and the uncertainty in these values) should converge to the population expressions as the number of bootstrap samples goes to infinity. To use the bootstrap procedure on a time series must require a different procedure that specified in the text. If anyone knows more about this (or can tell me why the above logic is incorrect) please contact me.

Part (b): We implement this part of the problem in the `R` code `ex_1_22_b.R`. In that code we first fit a single spline curve to the full data set. Then to each fitted point in that curve we compute the residual with the true data. Each bootstrapped sample of output is obtained by sampling from these residuals and producing a new output vector by adding this sample of residuals to the original output. When the above `R` script is run we get the plot shown in Figure 3.

Exercise 1.23 (solving sudoku)

Part (a): We have

```
> s
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    0    0    0    0    0    6    0    4    0
[2,]    2    7    9    0    0    0    0    5    0
[3,]    0    5    0    8    0    0    0    0    2
[4,]    0    0    2    6    0    0    0    0    0
```

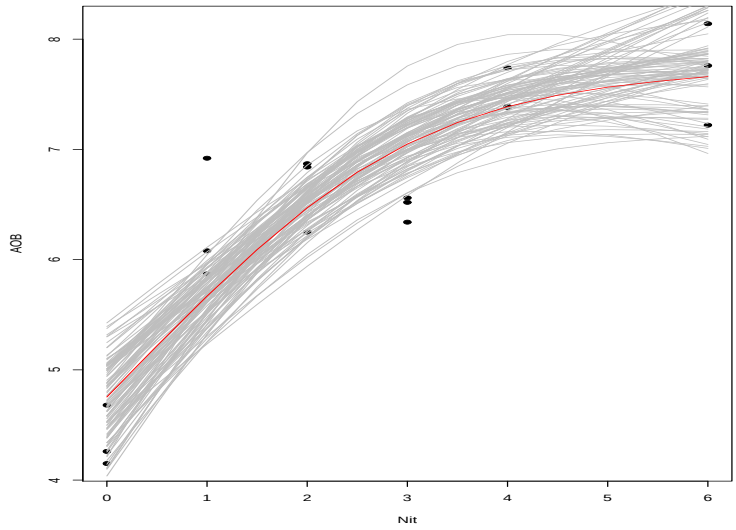

Figure 3: Many bootstrapped spline fits to the bacteria abundance *AOB* versus nitrogen levels.

```
[5,]    0    0    0    0    0    0    0    0    0
[6,]    0    0    1    0    9    0    6    7    3
[7,]    8    0    5    2    0    0    4    0    0
[8,]    3    0    0    0    0    0    0    8    5
[9,]    6    0    0    0    0    0    9    0    1
```

Part (b): See the `R` code `ex_1_23.R`.

Part (c): The value of `s[i]` is a way of expressing the elements of `s` by indexing down the columns of `s`. Thus the first index is the first element in the first column. The ninth index is the last element in the first column. The *tenth* index is the first element in the second column and so on.

Part (e): The first for loop with control given by `u in (1:9)[pool[a,b,]]` selects values for `u` that are possible to put into the spot `s[i,j]` as defined by the data structure `pool`. We can put `u` at the spot `a,b` if it is not already in the `a` row, which we can check with the elements of the vector `u==s[a,]`. If `u` is an element of this row one of the elements of this vector will be TRUE which when we sum all the elements in the vector would give a positive sum i.e. 1. This same logic holds for the `b` column with `u==s[,b]` and with the box around element `a,b` with `u==s[boxa,boxb]`. If any of the sums evaluate to 1 this value of `u` is not possible for the spot `a,b`.

Part (f): We can solve this grid with the following `R` code

```
while( sum(s==0)>0 ){
  # generate a random a,b element
```

```
  a = sample(1:9,1)
  b = sample(1:9,1)

  if( s[a,b]==0 ){  # this element has not alread been filled

    boxa = 3*trunc((a-1)/3)+1
    boxa = boxa:(boxa+2)
    boxb = 3*trunc((b-1)/3)+1
    boxb = boxb:(boxb+2)

    for ( u in (1:9)[pool[a,b,]])
      pool[a,b,u] = (sum(u==s[a,])+sum(u==s[,b])+sum(u==s[boxa,boxb]))==0

    if( sum(pool[a,b,])==1 ) s[a,b]=(1:9)[pool[a,b,]]
  }

}
```

This gives the following grid

```
> s
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    1    3    8    5    2    6    7    4    9
 [2,]    2    7    9    3    4    1    8    5    6
 [3,]    4    5    6    8    7    9    3    1    2
 [4,]    7    4    2    6    3    5    1    9    8
 [5,]    9    6    3    1    8    7    5    2    4
 [6,]    5    8    1    4    9    2    6    7    3
 [7,]    8    9    5    2    1    3    4    6    7
 [8,]    3    1    7    9    6    4    2    8    5
 [9,]    6    2    4    7    5    8    9    3    1
```

the same as in the book.

Chapter 2 (Random Variable Generation)

Exercise 2.2

To use the inverse transform method we draw a uniform random variable $u \sim \mathcal{U}_{[0,1]}$ and then find a sample x from the distribution of interest by solving the equation $F(x) = u$ for x, where $F(\cdot)$ is the distribution function of the density we desire to sample from.

Part (a): For the Logistic probability density $f(x)$ given by

$$f(x) = \frac{1}{\beta} \frac{e^{-(x-\mu)/\beta}}{(1+e^{-(x-\mu)/\beta})^2}\,, \tag{1}$$

is defined over $-\infty < x < +\infty$ so for the distribution function we find

$$F(x) = \int_{-\infty}^{x} \frac{1}{\beta} \frac{e^{-(x-\mu)/\beta}}{(1+e^{-(x-\mu)/\beta})^2} dx\,.$$

Let $v = 1 + e^{-(x-\mu)/\beta}$ so that $dv = -\frac{1}{\beta} e^{-(x-\mu)/\beta}$ and $dx = -\beta e^{(x-\mu)/\beta}$ and $F(x)$ then becomes

$$F(x) = -\int_{\infty}^{v} \frac{dv}{v^2} = -\left(-\frac{1}{v} \Big|_{\infty}^{v} = \frac{1}{v} = \frac{1}{1+e^{-(x-\mu)/\beta}}\,, \right. \tag{2}$$

as we were to show. For the Logistic probability density solving $F(x) = u$ for x gives

$$x = \mu - \beta \log\left(\frac{1}{u} - 1\right)\,.$$

In the first part of `ex_2_2.R` we plot a comparison of two ways to generate these variables. See Figure 4.

Part (b): For the Cauchy density $f(x)$ defined over $-\infty < x < +\infty$ we find

$$F(x) = \int_{-\infty}^{x} \frac{1}{\pi\sigma} \frac{dx}{1 + \left(\frac{x-\mu}{\sigma}\right)^2}\,.$$

Let $v = \frac{x-\mu}{\sigma}$ so that $dv = \frac{dx}{\sigma}$ and we get

$$F(x) = \frac{1}{\pi} \int_{-\infty}^{v} \frac{dv}{1+v^2} = \frac{1}{\pi} \tan^{-1}(v) - \tan^{-1}(-\infty))\,.$$

Since the limiting value of $\tan^{-1}(x)$ as $x \to -\infty$ is $-\frac{\pi}{2}$ the above becomes

$$F(x) = \frac{1}{\pi} \tan^{-1}\left(\frac{x-\mu}{\sigma}\right) + \frac{1}{2}\,,$$

as we were to show. For the Cauchy distribution when we solve $F(x) = u$ for x we have

$$x = \mu + \sigma \tan\left(\pi\left(u - \frac{1}{2}\right)\right)\,.$$

We generate random variables for a Cauchy distribution in the same way we did for the Logistic distribution.

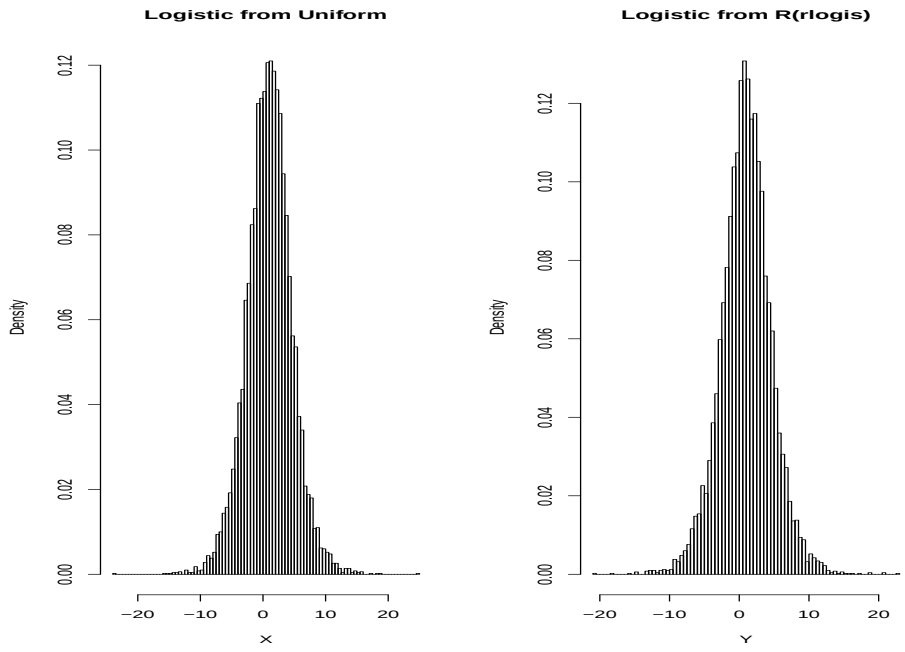

Figure 4: Generating Logistic random variables.

Exercise 2.3

Part (a): Since $E[U_i] = 0$ we have $E[Z] = 0$. Since

$$E[U_i] = \frac{1}{12}\left(\frac{1}{2} - \left(-\frac{1}{2}\right)\right) = \frac{1}{12}\,,$$

we have $E[Z] = 12\left(\frac{1}{12}\right) = 1$.

Part (b): In the `R` script `ex_2_3.R`, we generate random numbers using the central limit theorem (CLT), the Box Muller transform, and the `R` command `rnorm` as suggested an plot the resulting histograms in Figure 5. As commented in this problem the range of the generated variates in the CLT generator seems somewhat smaller than the other two generators produces.

Notes on Discrete Distributions

For example 2.4 (a binomial distribution) we can generate the values of p_k using the `R` command `pbinom(0:10,10,0.3)` which gives

```
> pbinom(0:10,10,0.3)
 [1] 0.02824752 0.14930835 0.38278279 0.64961072 0.84973167 0.95265101
 [7] 0.98940792 0.99840961 0.99985631 0.99999410 1.00000000
```

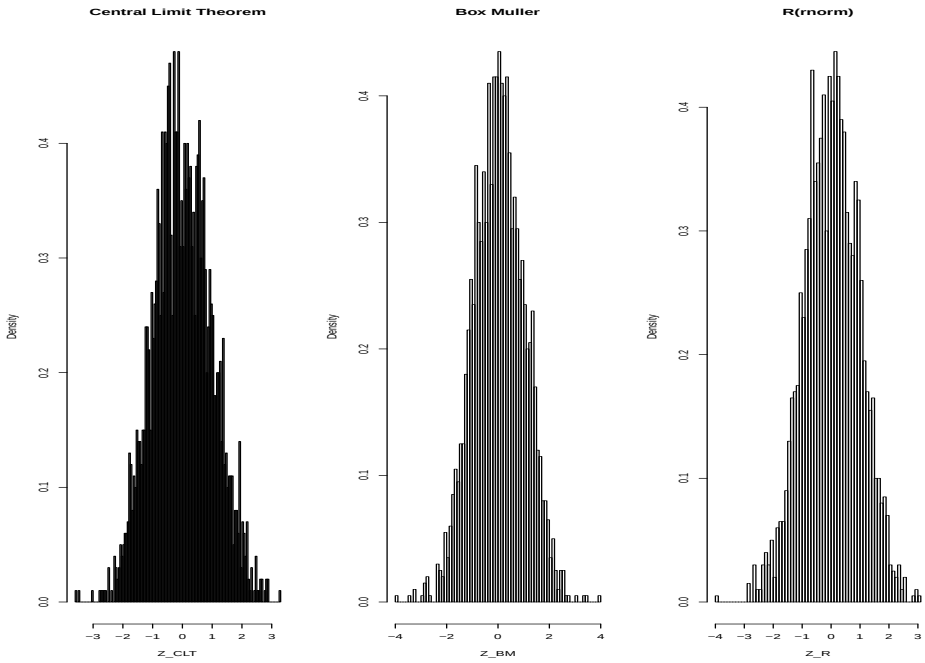

Figure 5: A histogram comparison of three normal random variable generators.

In the same way for example 2.4 (a Poisson distribution) we generate the values of p_k using the `R` command `ppois(0:20,7)`, which gives

```
> ppois(0:20,7)
 [1] 0.000911882 0.007295056 0.029636164 0.081765416 0.172991608 0.300708276
 [7] 0.449711056 0.598713836 0.729091268 0.830495937 0.901479206 0.946650377
[13] 0.973000227 0.987188607 0.994282798 0.997593420 0.999041817 0.999638216
[19] 0.999870149 0.999955598 0.999985505
```

where the upper limit (here 20) can be specified by the user to make the final probability as close to one as desired.

Exercise 2.5 (the probability of acceptance)

The probability of acceptance is defined as

$$P\left(U \leq \frac{f(Y)}{Mg(Y)}\right),$$

since it is in these cases where we will take $X = Y$ i.e. accept the sample Y from $g(y)$ as a sample from $f(X)$. This is the denominator in the proof of the accept-reject method and

we find

$$P\left(U \leq \frac{f(Y)}{Mg(Y)}\right) = \int_{-\infty}^{\infty} \int_{0}^{\frac{f(y)}{Mg(y)}} 1\, dug(y)dy$$
$$= \int_{-\infty}^{\infty} \frac{f(y)}{Mg(y)} g(y)dy = \frac{1}{M}\int_{-\infty}^{\infty} f(y)dy = \frac{1}{M}\,.$$

For the second question we assume that we don't know the *exact* normalization factors for f or g to make them proper densities. Thus we have access to approximate densities $\tilde{f}$ and $\tilde{g}$ and we then compute or derive a bound $\tilde{M}$ for these functions such that $\tilde{f} \leq \tilde{M}\tilde{g}$ for use with the accept-reject algorithm. When we run this algorithm using these functions we can compute an *empirical* acceptance rate $P\left(U \leq \frac{\tilde{f}(Y)}{\tilde{M}\tilde{g}(Y)}\right)$. This empirical acceptance rate is equal to

$$P\left(U \leq \frac{\tilde{f}(Y)}{\tilde{M}\tilde{g}(Y)}\right) = \int_{-\infty}^{\infty} \int_{0}^{\frac{\tilde{f}(y)}{\tilde{M}\tilde{g}(y)}} 1\, dug(y)dy = \int_{-\infty}^{\infty} \frac{\tilde{f}(y)}{\tilde{M}\tilde{g}(y)} g(y)dy\,.$$

If our functions $\tilde{f}$ and $\tilde{g}$ are only incorrect in the sense that they have errors in the normalizing constants then there exist other constants $\hat{f}$, $\hat{g}$, and $\hat{M}$ that when multiplied by the "tilde" variables give us the correct densities and reciprocal acceptance rate M. That is

$$f = \hat{f}\tilde{f}\,, \quad g = \hat{g}\tilde{g}\,, \quad \text{and} \quad M = \hat{M}\tilde{M}\,. \tag{3}$$

Thus the empirical acceptance probability we compute when running the accept-reject algorithm using $\tilde{f}$, $\tilde{g}$, and $\tilde{M}$ becomes

$$\tilde{P}_A = \int_{-\infty}^{\infty} \frac{\tilde{f}(y)}{\tilde{M}\tilde{g}(y)} g(y)dy = \left(\frac{\hat{g}\hat{M}}{\hat{f}}\right) \int_{-\infty}^{\infty} \left(\frac{f(y)}{Mg(y)}\right) g(y)dy = k\frac{1}{M}\,. \tag{4}$$

where the constant $k \equiv \frac{\hat{g}\hat{M}}{\hat{f}}$. This statement says that when we don't know the true normalization factors for the densities of interest the empirical acceptance-rate will be *proportional* to the true acceptance-rate when using the true densities.

Notes on the number of trials needed in the Accept-Reject Algorithm

The book comments that typically in a Monte-Carlo setting we want to generate `Nsim` draws from the density $f(x)$ of interest where the value of `Nsim` is given. When we use the accept-reject algorithm (since it is a random algorithm) we then don't know exactly how many samples from $f(x)$ we will actually obtain. However, since the probability of acceptance (or the probability of getting a draw from $f(x)$ is $\frac{1}{M}$) if we call the accept-reject algorithm MNsim times then *on average* we will expect to get

$$\frac{1}{M}(M\text{Nsim}) = \text{Nsim}\,,$$

draws from $f(x)$. We can use this fact to code our iterations in a somewhat more intelligent way.

Exercise 2.7

When $f \sim \mathcal{B}(\alpha, \beta)$ and $g \sim \mathcal{B}(a, b)$ then we have

$$\frac{f}{g} = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \cdot \frac{B(a,b)}{x^{a-1}(1-x)^{b-1}} = x^{\alpha-a}(1-x)^{\beta-b}\left(\frac{B(a,b)}{B(\alpha,\beta)}\right), \tag{5}$$

with

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}. \tag{6}$$

We want to show that there is a M such that $\frac{f}{g} \leq M$ for $0 < x < 1$. If the powers of x and $1-x$ are non-negative then over the domain $0 < x < 1$ the ratio $\frac{f}{g}$ well be bounded, thus we require

$$\alpha - a \geq 0 \quad \text{and} \quad \beta - b \geq 0\,.$$

or

$$a \leq \alpha \quad \text{and} \quad b \leq \beta\,.$$

If a and b were to *equal* α and β then drawing samples from g would be the same as drawing samples from f. Thus we want a and b to be as close as α and β as possible to make the acceptance probability as large as possible. If a and b must be integers than we should take them to be $a = \lfloor \alpha \rfloor$ and $b = \lfloor \beta \rfloor$, to make the acceptance probability as large as possible. The maximum of $\frac{f}{g}$ as a function of x is given by solving $\frac{d}{dx}\left(\frac{f}{g}\right) = 0$ for x or solving

$$(\alpha-a)x^{\alpha-a-1}(1-x)^{\beta-b} - (\beta-b)x^{\alpha-a}(1-x)^{\beta-b-1} = 0\,.$$

or when we divide by $x^{\alpha-a}(1-x)^{\beta-b}$ we get

$$\frac{\alpha-a}{x} - \frac{\beta-b}{1-x} = 0\,.$$

Solving this for x we get

$$x = -\frac{\alpha-a}{\beta-b-\alpha+a} \quad \text{so} \quad 1-x = \frac{\beta-b}{\beta-b-\alpha+a}\,.$$

When we put this value into the ratio given by Equation 5, we get

$$\left.\frac{f}{g}\right|_{\max} = \frac{B(a,b)}{B(\alpha,\beta)}\left(\frac{(a-\alpha)^{\alpha-a}(\beta-b)^{\beta-b}}{(\beta-b-\alpha+a)^{\alpha-a+\beta-b}}\right)$$

Exercise 2.8 (accept-reject with a double exponential distribution)

Part (a): For the densities specified we find the ratio given by

$$\frac{f(x)}{g(x|\alpha)} = \frac{\frac{1}{\sqrt{2\pi}}e^{-x^2/2}}{\left(\frac{\alpha}{2}\right)e^{-\alpha|x|}} = \sqrt{\frac{2}{\pi}}\alpha^{-1}\exp\left(-\frac{x^2}{2}+\alpha|x|\right).$$

Lets consider the case when $x > 0$ then we have

$$x^2 - 2\alpha|x| = x^2 - 2\alpha x = (x^2 - 2\alpha x + \alpha^2) - \alpha^2 = (x-\alpha)^2 - \alpha^2 \,.$$

Thus we see that the argument of the exponent above becomes

$$-\frac{1}{2}(x^2 - 2\alpha|x|) = -\frac{1}{2}(x-\alpha)^2 + \frac{1}{2}\alpha^2 \leq \frac{1}{2}\alpha^2 \,.$$

When $x < 0$ the same type of transformation holds

$$-\frac{1}{2}(x^2 - 2\alpha|x|) = -\frac{1}{2}(x^2 + 2\alpha x + \alpha^2) + \frac{1}{2}\alpha^2 = -\frac{1}{2}(x+\alpha)^2 + \frac{1}{2}\alpha^2 \leq \frac{1}{2}\alpha^2 \,.$$

Thus we conclude that $e^{-\frac{x^2}{2}+\alpha|x|} \leq e^{\frac{\alpha^2}{2}}$ and we have shown

$$\frac{f(x)}{g(x|\alpha)} \leq \sqrt{\frac{2}{\pi}}\alpha^{-1}e^{\frac{\alpha^2}{2}} \,.$$

When we try to *minimize* this with respect to the parameter α we take the derivative with respect to α, set the result equal to zero, and solve for α. Setting the derivative equal to zero gives me

$$\frac{d}{d\alpha}\left(\alpha^{-1}e^{\frac{\alpha^2}{2}}\right) = -\alpha^{-2}e^{\frac{\alpha^2}{2}} + \alpha^{-1}\alpha e^{\frac{\alpha^2}{2}} = 0 \,,$$

or

$$-\alpha^{-2} + 1 = 0 \,, \quad \text{so} \quad \alpha = \pm 1 \,.$$

If we consider the second derivative of the α expression

$$\frac{d^2}{d\alpha^2}\left(\alpha^{-1}e^{\frac{\alpha^2}{2}}\right) = \frac{d}{d\alpha}\left((1-\alpha^{-2})e^{\frac{\alpha^2}{2}}\right) = 2\alpha^{-3}e^{\frac{\alpha^2}{2}} + (1-\alpha^{-2})\alpha e^{\frac{\alpha^2}{2}} = (\alpha - \alpha^{-1} + 2\alpha^{-3})e^{\frac{\alpha^2}{2}} \,.$$

Thus we find that

$$\left.\frac{d^2}{d\alpha^2}\left(\alpha^{-1}e^{\frac{\alpha^2}{2}}\right)\right|_{\alpha=-1} = e^{1/2}(-1+1-2) < 0 \quad \text{is a maximum}$$
$$\left.\frac{d^2}{d\alpha^2}\left(\alpha^{-1}e^{\frac{\alpha^2}{2}}\right)\right|_{\alpha=+1} = e^{1/2}(+1-1+2) < 0 \quad \text{is a minimum} \,.$$

Part (b): Thus using this bound with $\alpha = +1$ we get

$$\frac{f(x)}{g(x|\alpha)} = \sqrt{\frac{2}{\pi}}e^{1/2} \quad \text{so} \quad M = \sqrt{\frac{2}{\pi}}e^{1/2} \,,$$

and our acceptance rate is given by

$$\frac{1}{M} = \sqrt{\frac{\pi}{2}}e^{-1/2} = 0.760173 \,.$$

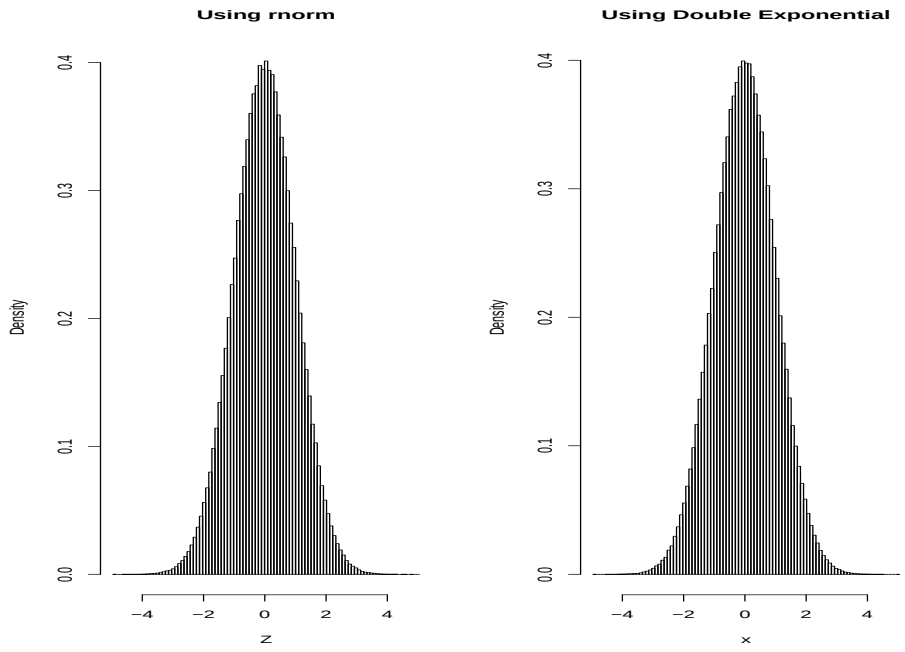

Figure 6: Random draws from a $\mathcal{N}(0,1)$ distribution using the `R` code `rnorm` (left) and the accept-reject algorithm with a double exponential as the proposal distribution (right).

Part (c): To generate random variates from the double-exponential density $g(x|\alpha)$ we can use the probability inverse transform. For $g(x|\alpha)$ we have a distribution function given by

$$\begin{aligned} G(x) &= \int_{-\infty}^{x} g(\xi|\alpha)d\xi = \frac{\alpha}{2}\int_{-\infty}^{x} e^{-\alpha|\xi|}d\xi \\ &= \frac{\alpha}{2}\begin{cases} \int_{\infty}^{x} e^{\alpha\xi}d\xi & x<0 \\ \int_{\infty}^{0} e^{\alpha\xi}d\xi + \int_{0}^{x} e^{-\alpha\xi}d\xi & x>0 \end{cases} = \frac{\alpha}{2}\begin{cases} \left(\frac{1}{\alpha}e^{\alpha\xi}\right|_{-\infty}^{x} & x<0 \\ \left(\frac{1}{\alpha}e^{\alpha\xi}\right|_{-\infty}^{0} + \left(\frac{1}{\alpha}e^{-\alpha\xi}\right|_{-\infty}^{x} & x>0 \end{cases} \\ &= \begin{cases} \frac{1}{2}e^{\alpha x} & x<0 \\ 1-\frac{1}{2}e^{-\alpha x} & x>0 \end{cases} . \end{aligned}$$

Then to use the inverse transform we draw a random sample $U \sim \mathcal{U}_{[0,1]}$ and solve for x in $x = G(x)$. Given u then if $u < \frac{1}{2}$ we have

$$u = \frac{1}{2}e^{\alpha x} \quad \text{so} \quad x = \frac{1}{\alpha}\log(2u)\,,$$

while if $\frac{1}{2} < u < 1$ we have

$$u = 1 - \frac{1}{2}e^{-\alpha x} \quad \text{so} \quad x = -\frac{1}{\alpha}\log(2(1-u))\,.$$

Part (c): For this part of the problem we use the accept-reject algorithm to generate data from a normal distribution by using as a candidate density the double exponential. This problem is worked in the `R` file `ex_2_8.R`. When that code is run it generates the results given in Figure 6.

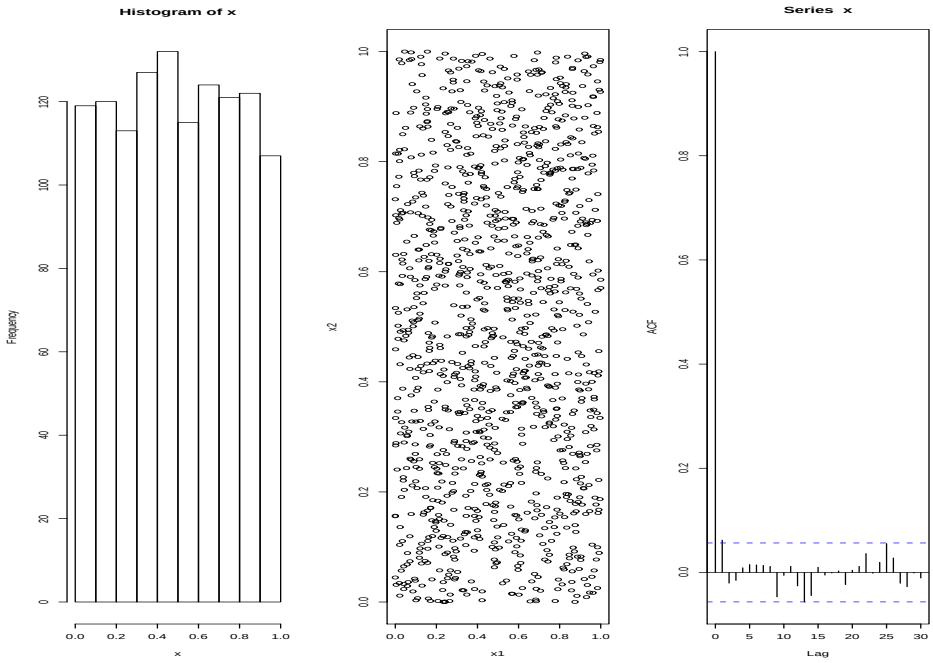

Figure 7: Data from the `randu` generator.

Exercise 2.10 (the `RANDU` data set)

Part (a): This exercise is worked in the `R` file `ex_2_10.R`. When we run that script we obtain the plot given in Figure 7.

Exercise 2.11

This exercise is worked in the `R` file `ex_2_11.R`.

Part (a): We implement code to generate random samples from a binomial distribution using the inverse transform method. We then compare the timing of this routine with the default `R` routine `rbinom`. We find

```
> system.time(wrbinom(10000,n,p)) # My generator
   user  system elapsed
  0.160   0.000   0.161
>
> system.time(rbinom(10000,n,p)) # The R generator is rbinom
   user  system elapsed
  0.000   0.000   0.002
```

Part (b): In the above `R` file we implement the suggested algorithm. A plot of the histogram

of the data generated from the suggested loop verifies that the points are indeed from $\mathcal{U}_{[0,\alpha]}$. When we compare the various generators for a couple of value of α we find that for $\alpha = 0.1$ the computational time is given by

```
> system.time( wUniformAlpha(Nsim,alpha) )
   user  system elapsed
  9.016   0.008   9.336
>
> system.time( runif(Nsim,max=alpha) )
   user  system elapsed
  0.012   0.000   0.013
>
> system.time( alpha*runif(Nsim) )
   user  system elapsed
  0.012   0.000   0.013
```

While for $\alpha = 0.9$ we find

```
> system.time( wUniformAlpha(Nsim,alpha) )
   user  system elapsed
  1.408   0.000   1.416
>
> system.time( runif(Nsim,max=alpha) )
   user  system elapsed
  0.012   0.000   0.012
>
> system.time( alpha*runif(Nsim) )
   user  system elapsed
  0.016   0.000   0.014
```

The hand coded generator takes more time when α is small. There is almost no difference in the time of generation for the second two methods.

Exercise 2.12 (the inverse transformation method)

Part (a): This problem is worked in the `R` program `ex_2_12.R`. When we run that program we get the plot shown in Figure 8. In working this problem one has to make sure to use the *same* random draws from an $\mathcal{E}(1)$ distribution in computing the needed sums in the numerator and the denominator to computing a correct sample from the beta distribution.

Part (b): For an exponential distribution its density and distribution functions are given by

$$f(x) = \lambda e^{-\lambda x} \quad \text{and} \quad F(x) = 1 - e^{-\lambda x} \quad \text{for} \quad 0 \leq x < \infty \,.$$

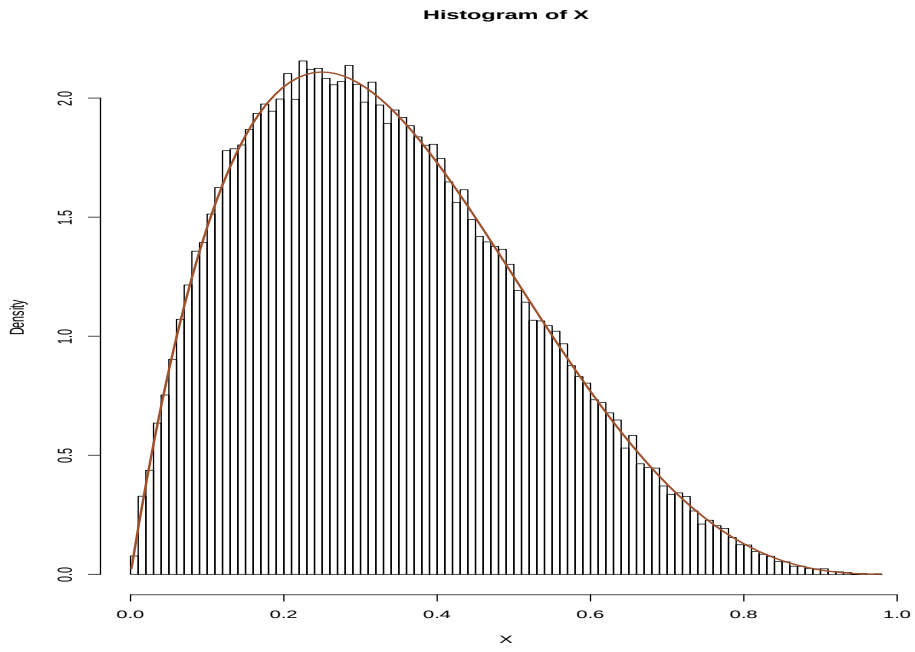

Figure 8: Samples (and the exact p.d.f) from a beta $\mathcal{B}(2,4)$ distribution.

when we solve $F(x) = u$ for x we have

$$x = -\frac{\log(1-u)}{\lambda}\,.$$

When $u \sim \mathcal{U}_{[0,1]}$ so is $1-u$ and we can generate exponentially generated random variables x using

$$x = -\frac{\log(u)}{\lambda}\,.$$

Part (c): For the Logistic distribution its density is given by Equation 1 and its distribution function is given by Equation 2. Thus when we solve $F(x) = u$ for x we have

$$x = \mu + \beta \log\left(\frac{u}{1-u}\right)\,.$$

If we want $\mu = 0$ and $\beta = 1$ then the above becomes

$$x = \log\left(\frac{u}{1-u}\right)\,.$$

Exercise 2.13 (moments of the univariate Pareto distribution)

To begin we provide some background on the Pareto distribution. The general p.d.f for a univariate Pareto distribution is given by

$$f(x|x_0,\alpha) = \frac{\alpha x_0^\alpha}{x^{\alpha+1}}\,, \tag{7}$$

when $x > x_0$ and is 0 otherwise. In this problem $x_0 = 1$. Note that $E(X^k)$ will only exist the following integral

$$\int_{x_0}^{\infty} \frac{x^k}{x^{\alpha+1}} dx = \int_{x_0}^{\infty} x^{k-\alpha-1} dx \,,$$

converges. The convergence of this later integral requires $k - \alpha - 1 < -1$ or

$$k < \alpha \,.$$

We can compute various statistics for this distribution. We find

$$\begin{aligned} E(X) &= \int_{x_0}^{\infty} x \frac{\alpha x_0^\alpha}{x^{\alpha+1}} dx = \int_{x_0}^{\infty} \frac{\alpha x_0^\alpha}{x^\alpha} dx = \alpha x_0^\alpha \left(\frac{x^{-\alpha+1}}{-\alpha+1} \Big|_{x_0}^{\infty} = \alpha x_0^\alpha \left(0 + \frac{x_0^{-\alpha+1}}{\alpha - 1} \right) \right. \\ &= \frac{\alpha x_0}{\alpha - 1} \,. \end{aligned} \tag{8}$$

Next assuming $\alpha > 2$ we compute $E(X^2)$. We find that

$$\begin{aligned} E(X^2) &= \int_{x_0}^{\infty} \frac{\alpha x_0^\alpha}{x^{\alpha-1}} dx = \alpha x_0^\alpha \left(\frac{x^{-\alpha+1+1}}{-\alpha+2} \Big|_{x_0}^{\infty} = \alpha x_0^\alpha \left(0 + \frac{x_0^{-\alpha+2}}{\alpha - 2} \right) \right. \\ &= \frac{\alpha x_0^2}{\alpha - 2} \,. \end{aligned} \tag{9}$$

Thus from these two expressions we can compute the variance of X and find

$$\begin{aligned} \mathrm{Var}(X) &= E(X^2) - E(X)^2 = \frac{\alpha x_0^2}{\alpha - 2} - \frac{\alpha^2 x_0^2}{(\alpha - 1)^2} \\ &= \frac{\alpha x_0^2}{(\alpha - 1)^2 (\alpha - 2)} \,. \end{aligned} \tag{10}$$

The density function $F(x)$ for a Pareto distribution is given by

$$\begin{aligned} F(x) &= \int_{x_0}^{x} f(\xi) d\xi = \int_{x_0}^{x} \frac{\alpha x_0^\alpha}{\xi^{\alpha+1}} d\xi = \alpha x_0^\alpha \left(\frac{\xi^{-\alpha}}{-\alpha} \Big|_{x_0}^{x} \right. \\ &= -x_0^\alpha \left(\frac{1}{x^\alpha} - \frac{1}{x_0^\alpha} \right) = 1 - \left(\frac{x_0}{x} \right)^\alpha \,. \end{aligned} \tag{11}$$

Then to generate random variables from a Pareto distribution we can use the inverse transform method. We generate a a random variable from $\mathcal{U}_{[0,1]}$ and then solve $F(x) = u$ for x. We have

$$1 - u = \frac{x_0^\alpha}{x^\alpha} \,,$$

or

$$x = \frac{x_0}{(1-u)^{1/\alpha}} = x_0 (1-u)^{-1/\alpha} \,.$$

Since $u \sim \mathcal{U}_{[0,1]}$, the variable $1 - u$ is also a draw from a $\mathcal{U}_{[0,1]}$ distribution, and we can generate a Pareto random variable using

$$x = x_0 u^{-1/\alpha} \,. \tag{12}$$

In the `R` code `ex_2_13.R` we generate Pareto random variables this way. The resulting histogram is presented in Figure 9

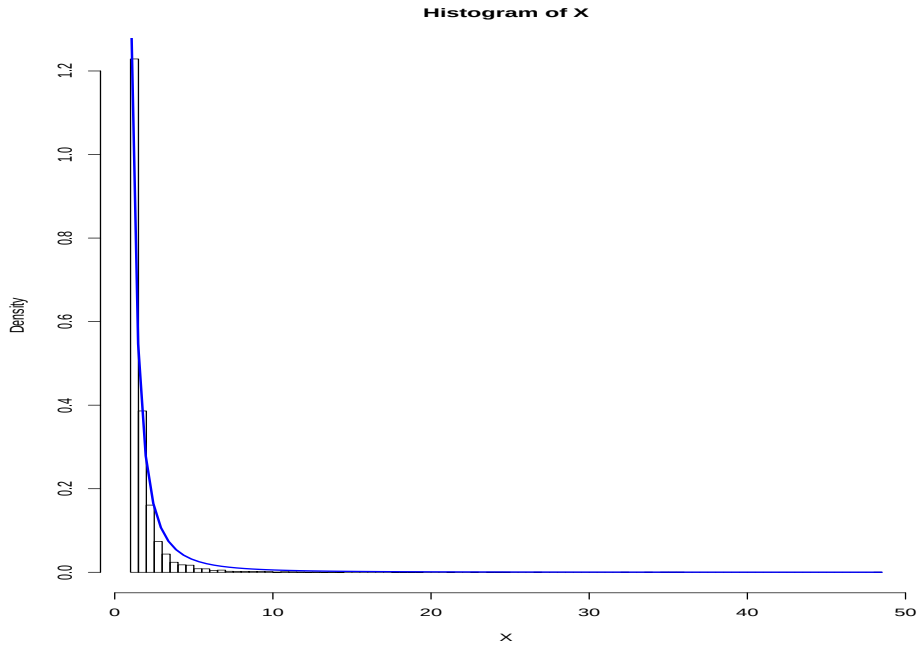

Figure 9: Samples (and the exact p.d.f) of a Pareto $\mathcal{P}(\alpha)$ distribution when $\alpha = 2.4$ and $x_0 = 1$. Note the extremely long tails that this distribution has.

Exercise 2.14 (the inverse transformation with discrete densities)

In this problem we are asked to generate 1000 draws from each of the distributions. When we compare the histograms that result with the true density function this small number of samples can result in a histogram that looks rather different than the expected true density functions. To verify that we have coded everything correctly one can run the above code with a much larger number of samples to draw from and observe that the histograms in that case. One then observes that the histograms do indeed converge to the theoretical curves.

This exercise is worked in the `R` code `ex_2_14.R`.

Part (a): For 1000 samples, the resulting histogram for Poisson random variables is presented in Figure 10.

Part (b): For 1000 samples, the resulting histogram for the negative binomial distribution is presented in Figure 11.

Part (c): For 1000 samples, the resulting histogram for the logarithmic distribution distribution is presented in Figure 13.

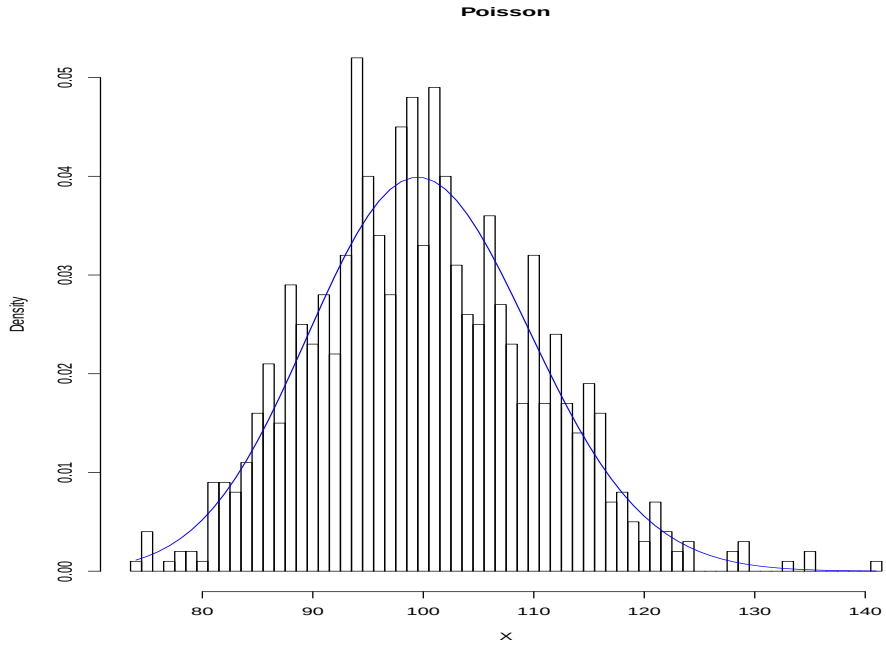

Figure 10: 1000 samples from a Poisson distribution with $\lambda = 100$.

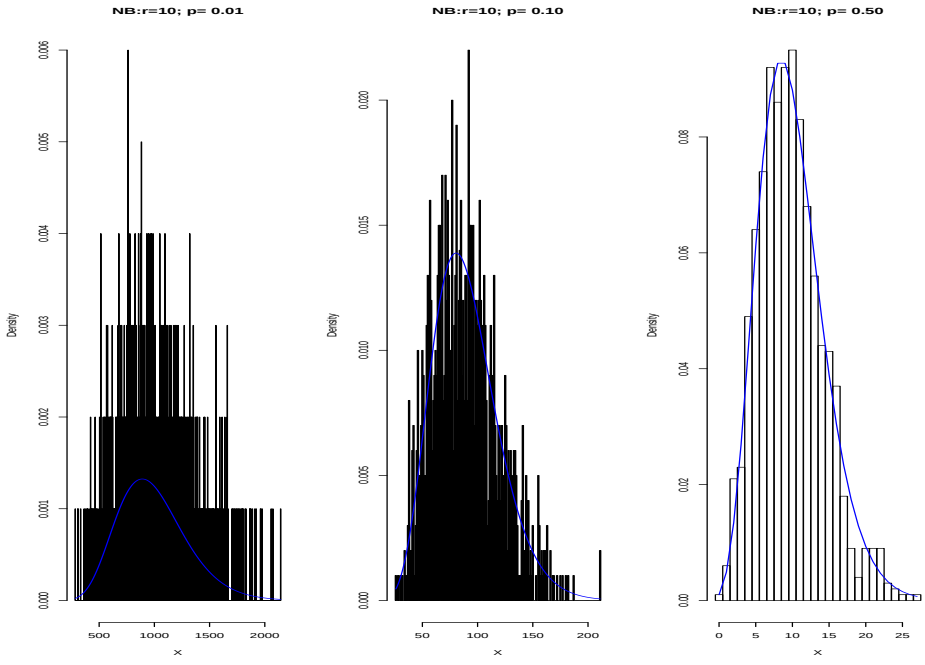

Figure 11: 1000 samples from a negative binomial distribution with $r = 10$ and (from left to right) for $p = 0.01$, 0.1, and 0.5.

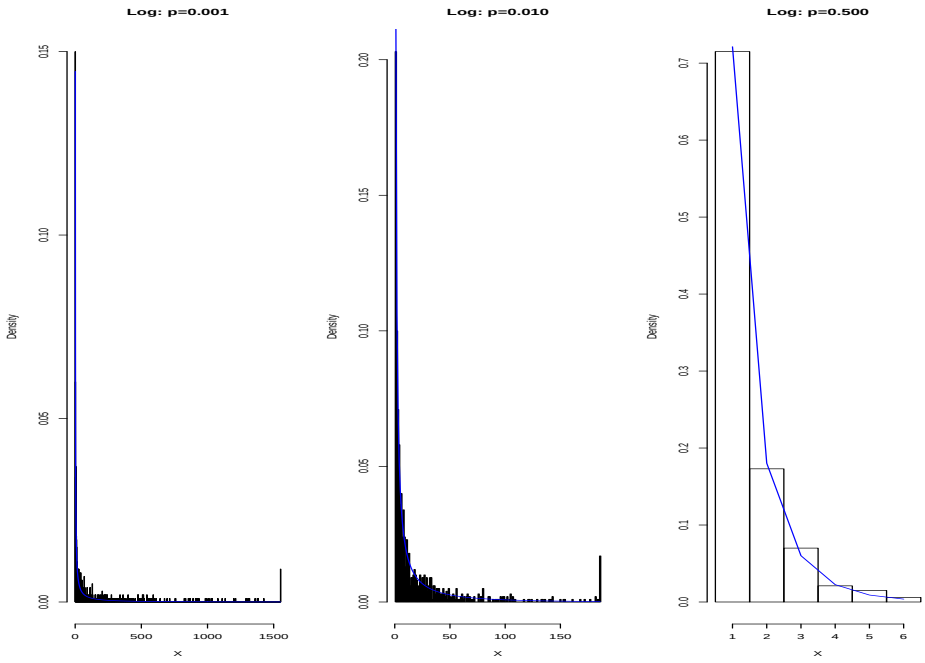

Figure 12: 1000 samples from a logarithmic distribution (from left to right) for $p = 0.001$, 0.01, and 0.5.

Exercise 2.15 (the Poisson from exponential RV's)

Note that the `R` definition of the exponential distribution matches that from the book. In other words the density function for X when $X \sim \mathcal{E}xp(\lambda)$ is given by

$$f(x) = \lambda e^{-\lambda x} ,$$

and one sample from this distribution is generated with the `R` call `rexp(1,rate=lambda)`. We expect that the generator described will not be very good when λ is small, since in that case many random draws X_i will be small and it will require a large number of draws from $\mathcal{E}xp(\lambda)$ to produce a sum larger than one.

This exercise is worked in the `R` code `ex_2_15.R`.

Exercise 2.16 (more Beta RV's)

In the `R` code `ex_2_16.R`, the suggested algorithm is implemented. It seems in the small α and β cases the algorithm converges relatively quickly. In the large α and β cases this does not seem to be true at all.

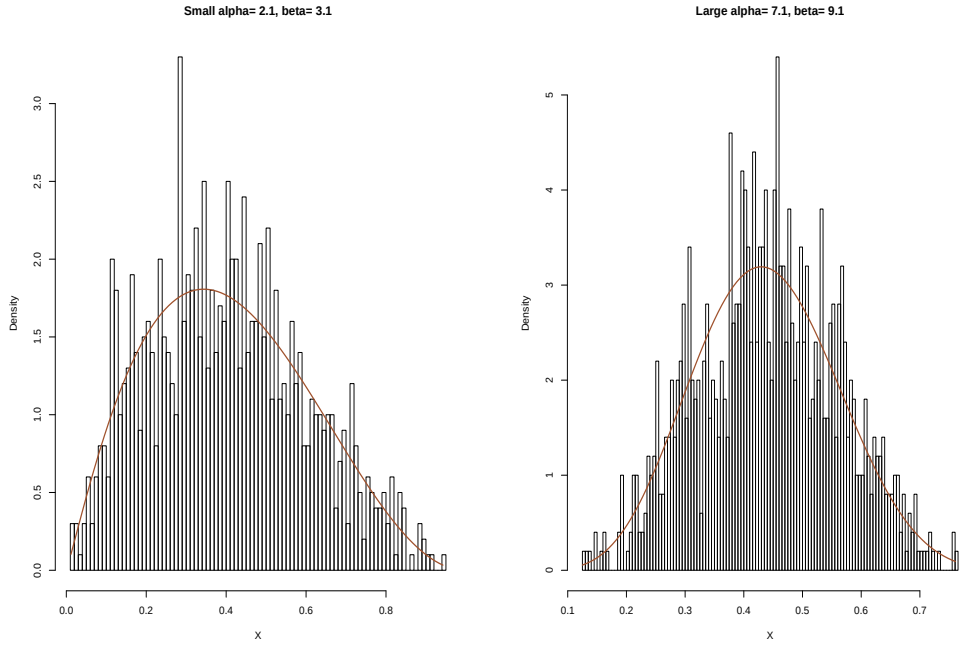

Figure 13: **Left:** Generating 1000 RV's from a beta distribution with $\alpha = 2.1$ and $\beta = 3.1$. **Right:** Generating 1000 RV's from a beta distribution with $\alpha = 7.1$ and $\beta = 9.1$. This second run took *much* more computational time to run than the first.

Exercise 2.17 (the gamma distribution)

Part (a): We want to draw x from a $\mathcal{G}a(\alpha, \beta)$ distribution. If we draw y from a $\mathcal{G}a(\alpha, 1)$ distribution and then compute $x = \frac{y}{\beta}$ then we claim that x will be drawn from a $\mathcal{G}a(\alpha, \beta)$. To see this note that the p.d.f of y is given by by

$$g(y) = \frac{1}{\Gamma(\alpha)} y^{\alpha-1} e^{-y} .$$

Using this the p.d.f of x, denoted by $f(x)$, is given by

$$\begin{aligned} f(x) &= g(y(x)) \left| \frac{dy}{dx} \right| = \beta g(y(x)) \\ &= \frac{\beta}{\Gamma(\alpha)} (\beta x)^{\alpha-1} e^{-\beta x} = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} , \end{aligned} \tag{13}$$

which is the p.d.f of a $\mathcal{G}a(\alpha, \beta)$ as we were to show. Thus to generate random variables from a $\mathcal{G}a(\alpha, \beta)$ it is sufficient to scale random variables from a $\mathcal{G}a(\alpha, 1)$ distribution.

Part (b): We want to generate random variables from a $\mathcal{G}a(n, 1)$ using random variables from an exponential $\mathcal{E}xp(\lambda)$ distribution as the candidate density using the accept-reject algorithm. Now the p.d.f for the $\mathcal{E}xp(\lambda)$ distribution is given by $g(x) = \lambda e^{-\lambda x}$ and to find the best value for the parameter λ we want to find the smallest value for M such that

$$\frac{f(x)}{g(x)} \leq M ,$$

where $f(x)$ is the p.d.f for the $\mathcal{G}a(n,1)$ distribution. Recall that the functional form of a $\mathcal{G}a(n,1)$ distribution is $\frac{1}{\Gamma(n)}x^{n-1}e^{-x}$. Thus this ratio in terms of x is given by

$$\frac{1}{\Gamma(n)}\frac{x^{n-1}e^{-x}}{\lambda e^{-\lambda x}} = \frac{1}{\lambda\Gamma(n)}x^{n-1}e^{-(1-\lambda)x}\,.$$

If $-(1-\lambda) > 0$ or $\lambda > 1$ then as $x \to +\infty$ this fraction diverges. Thus we must have $\lambda \leq 1$ to use this method. Lets assume that $\lambda < 1$ and find the maximum value of $x^{n-1}e^{-(1-\lambda)x}$ as a function of x. Taking the x derivative and setting the result equal to zero gives

$$(n-1)x^{n-2}e^{-(1-\lambda)x} - (1-\lambda)x^{n-1}e^{-(1-\lambda)x} = 0\,.$$

When we solve for x we get

$$x = \frac{n-1}{1-\lambda}\,.$$

With this value of x we get M given by

$$M(n;\lambda) = \frac{1}{\lambda\Gamma(n)}\left(\frac{n-1}{1-\lambda}\right)^{n-1}e^{-(n-1)}\,.$$

We now ask what value of $\lambda \leq 1$ makes this value as small as possible. To find that value, we take the first derivative with respect to λ, set the result equal to zero, and solve for λ. The first derivative gives

$$-\frac{1}{\lambda^2}\frac{1}{(1-\lambda)^{n-1}} - (n-1)\frac{(-1)}{\lambda(1-\lambda)^n} = 0\,.$$

We then multiply by $(1-\lambda)^n\lambda^2$ to get $-(1-\lambda)+(n-1)\lambda = 0$ or

$$\lambda = \frac{1}{n} \quad \text{so} \quad 1-\lambda = \frac{n-1}{n}\,. \tag{14}$$

This is the optimal value for λ. When we use this value in M we get

$$M = \frac{n}{\Gamma(n)}\frac{(n-1)^{n-1}}{(n-1)^{n-1}}n^{n-1}e^{-(n-1)} = \frac{n^n e^{-(n-1)}}{\Gamma(n)}\,. \tag{15}$$

Part (c): For this part of the problem we want to us random draws from the candidate density of $\mathcal{G}a(a,b)$ to generate random variables from a $\mathcal{G}a(\alpha,1)$. In this case the ratio of $\frac{f}{g}$ is given by

$$\frac{f(x)}{g(x)} = \frac{1}{\Gamma(\alpha)}x^{\alpha-1}e^{-x}\frac{\Gamma(a)}{b^a}\frac{1}{x^{a-1}e^{-bx}} = \frac{\Gamma(a)}{\Gamma(\alpha)}\frac{1}{b^a}x^{\alpha-a}e^{-(1-b)x}\,.$$

Then for this ratio to not diverge as $x \to +\infty$ we must have $1-b > 0$ or $b < 1$. For this to not diverge as $x \to 0$ we must have $\alpha - a \geq 0$ or $a \leq \alpha$. If we maximize this as a function of x we must solve for x in

$$(\alpha-a)x^{\alpha-a-1}e^{-(1-b)x} - (1-b)x^{\alpha-a}e^{-(1-b)x} = 0\,.$$

Solving for x we get

$$x = \frac{\alpha-a}{1-b}\,, \tag{16}$$

and M becomes

$$M = \frac{\Gamma(a)}{\Gamma(\alpha)} \frac{1}{b^a} \left(\frac{\alpha - a}{1 - b} \right)^{\alpha - a} e^{-(\alpha - a)} ,$$

which is the desired expression.

Part (d): We would next want to *minimize* the above expression for M with respect to b. This will make the acceptance rate $1/M$ as large as possible and make the accept-reject algorithm as efficient as possible. To find this minimum we take the derivative of the above expression with respect to b, set the result equal to zero, and solve for b. We find this derivative given by

$$-ab^{-a-1}(1-b)^{-(\alpha-a)} + (\alpha - a)b^{-a}(1-b)^{-(\alpha-a)-1} = 0 .$$

If we multiply by $b^{a+1}(1-b)^{\alpha-a+1}$ we get

$$-a(1-b) + (\alpha - a)b = 0 ,$$

or

$$b = \frac{a}{\alpha} \quad \text{so} \quad 1 - b = \frac{\alpha - a}{\alpha} .$$

Recall that the *mean* of a $\mathcal{G}a(a, b)$ distribution if given by $\frac{a}{b}$. Then the mean of the $\mathcal{G}a(\alpha, 1)$ is α, and the mean of the distribution $\mathcal{G}a(a, b)$ with b as specified as above is given by

$$\frac{a}{b} = \alpha ,$$

showing these two distributions have the same mean. With that value for b we find that the expression for M above becomes proportional to

$$\left(\frac{\alpha - a}{e} \right)^{\alpha - a} \left(\frac{a}{\alpha} \right)^{-a} \left(\frac{\alpha - a}{\alpha} \right)^{-(\alpha - a)} = \frac{\alpha^{\alpha - a}}{e^{\alpha - a}} \left(\frac{\alpha}{a} \right)^{a} = \frac{\alpha^{\alpha}}{e^{\alpha - a} a^{a}} .$$

Since $e^{-a}a^a$ increases as a increases to make M as small as possible we want to make a as large as possible. Since $a \leq \alpha$ this motivates selecting $a = \lfloor \alpha \rfloor$ if we must restrict a to be an integer.

Exercise 2.18 (an unnormalized density)

Part (a): We first plot the given function f in Figure 14 (left). We next consider the ratio $f(x)/g(x)$ where g is the standard normal density. We thus find

$$\frac{f(x)}{g(x)} = \sqrt{2\pi}(\sin(6x)^2 + 3\cos(x)^2 \sin(4x)^2 + 1) .$$

Since the trigonometric functions above are bounded we expect this ratio to have an upper bound. To find this upper bound (as a function of x we can use the `R` command `optimize`. When we do that we find that the above ratio is bounded above by $\tilde{M} = 10.94031$.

Part (b): This exercise is worked in the `R` code `ex_2_18.R`.

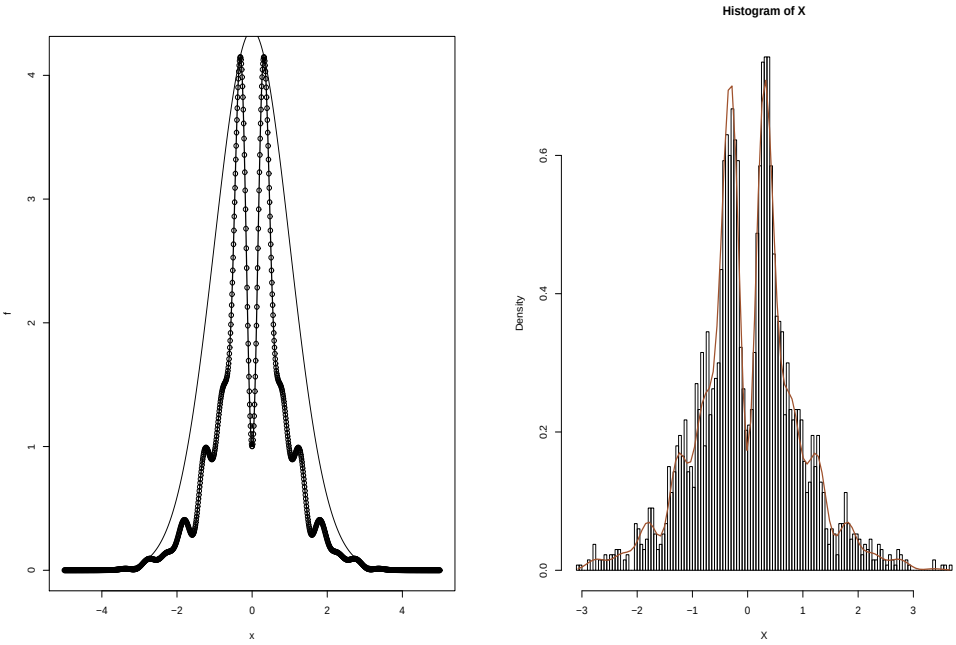

Figure 14: **Left:** The unnormalized density $f(x)$ for Exercise 2.18 and the upper bound on $f(x)$ or $Mg(x)$. **Right:** A histogram of the samples drawn from the *normalized* density $f(x)$ and a plot of this functions probability density function. The normalization factor for $f(x)$ is obtained from the empirical probability of acceptance (see the text for more details).

Part (c): In the above `R` script we also compute the empirical probability of acceptance $\tilde{P}_A$ when using the functions given. From Equation 4 we have that $\tilde{P}_A = \frac{\hat{g}\hat{M}}{\hat{f}}\frac{1}{M}$, which when we solve for $\hat{f}$ the normalizing factor for the $\tilde{f}(x)$ density and use Equations 3 we have that

$$\hat{f} = \frac{\hat{g}}{\tilde{P}_A}\frac{1}{\tilde{M}}\,.$$

In this problem, we have that $\hat{g} = 1$, $\tilde{M} = 10.94031$, and we measure $\tilde{P}_A$ to be 0.5338. Using this and the above expression for $\hat{f}$ we can plot the p.d.f for f on top of the empirical histogram of samples drawn using the accept-reject algorithm. See Figure 14 (right).

Exercise 2.19

This exercise has been already performed. See Exercise 2.8 on Page 17

Exercise 2.20 (some more examples with the accept-reject algorithm)

Part (a): Our two densities in this case are

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \quad \text{and} \quad g(x) = \frac{1}{\pi}\left(\frac{\gamma}{\gamma^2 + x^2}\right) .$$

We will pick the parameter γ to make the accept-reject algorithm as efficient as possible. We have the ratio of the desired density to the candidate density given by

$$\frac{f(x)}{g(x)} = \frac{\pi e^{-\frac{x^2}{2}}(\gamma^2 + x^2)}{\sqrt{2\pi}\gamma} = \sqrt{\frac{\pi}{2}}\left(\frac{1}{\gamma}\right)(\gamma^2 + x^2)e^{-\frac{x^2}{2}} .$$

We next find the x value the makes the above expression a maximum. The first derivative gives

$$2xe^{-\frac{x^2}{2}} + (\gamma^2 + x^2)e^{-\frac{x^2}{2}}(-x) = 0 .$$

or assuming $x \neq 0$ this means

$$x^2 = 2 - \gamma^2 \quad \text{so} \quad x = \pm\sqrt{2 - \gamma^2} .$$

We thus have

$$\frac{f(x)}{g(x)} \leq \sqrt{\frac{\pi}{2}}\left(\frac{2}{\gamma}\right) e^{\frac{1}{2}(\gamma^2 - 2)} .$$

Lets now pick γ to make this expression as *small* as possible. The first derivative gives

$$-\frac{1}{\gamma^2} e^{\frac{1}{2}(\gamma^2-2)} + \frac{1}{\gamma} e^{\frac{1}{2}(\gamma^2-2)}\gamma = 0 .$$

or

$$-\frac{1}{\gamma^2} + 1 = 0 \quad \text{or} \quad \gamma = \pm 1 .$$

Using this value for γ we find that

$$\frac{f(x)}{g(x)} \leq \sqrt{2\pi} e^{-1/2} .$$

Thus we will take as our upper bound for the accept-reject algorithm as $M = \sqrt{2\pi}e^{-1/2}$. This problem is worked in the `R` script `ex_2_20.R`.

Part (b): To begin with we need to find a value of M such that

$$\mathcal{G}a(x; 4.3, 6.2) \leq M\mathcal{G}a(x; 4, 7) ,$$

Since we can evaluate the gamma distribution $\mathcal{G}(x; a, b)$ (with the parametrization above) using the `R` command `pgamma(x,a,1/b)` we can find the needed value of M using the `R` function `optimize`. We find $M = 1.0894$. A plot of the functions $f(x)$ and $Mg(x)$ are shown in Figure 15 (left). Using the accept-reject algorithm the histogram of the drawn samples is shown in Figure 15 (right). This problem is worked in the `R` script `ex_2_20.R`.

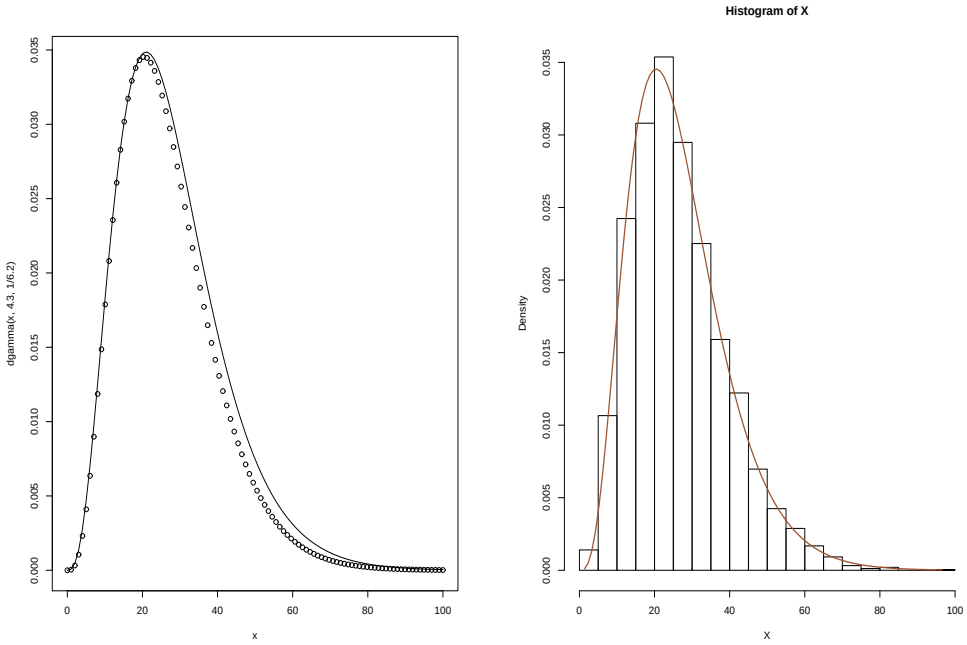

Figure 15: **Left:** The density $f(x)$ for Exercise 2.20 and the product $Mg(x)$ where M is selected to be the smallest value such that $\frac{f(x)}{g(x)} < M$. **Right:** A histogram of the samples from a $\mathcal{G}a(4.3, 6.2)$ drawn using the accept-reject algorithm with the candidate density $\mathcal{G}a(4, 7)$.

Exercise 2.21 (the noncentral chi-squared distribution)

Part (a): To show that at noncentral chi-squared distribution $\chi^2_p(\lambda)$ is a mixture representation of of χ^2_{p+2y} and $\mathcal{P}(\lambda/2)$ we will simply calculate the proposed mixture distribution and show that it equals the probability density function for a noncentral chi-squared distribution. To begin we recall the density function for a noncentral chi-squared distribution $f(x|\lambda)$ which is

$$f(x|\lambda) = \frac{1}{2}\left(\frac{x}{\lambda}\right)^{\frac{p-2}{4}} I_{\frac{p-2}{2}}\left(\sqrt{\lambda x}\right) e^{-\frac{\lambda+x}{2}}, \tag{17}$$

where $I_\nu(x)$ is the **modified Bessel function of the first kind** and has a series representation given by

$$I_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{j=0}^{\infty} \frac{1}{j!\Gamma(\nu + j + 1)} \left(\frac{z^2}{4}\right)^j. \tag{18}$$

We now construct the mixture density suggested. We want to evaluate $\sum_{i=0}^{\infty} p_i f_i(x)$ where $p_i = \mathcal{P}(\lambda/2)$ and $f_i(x) = \chi^2_{p+2i}$. The density function for a χ^2_ν is a special case of the Gamma distribution $\mathcal{G}a(\alpha, \beta)$ given by Equation 13 with $\alpha = \frac{\nu}{2}$ and $\beta = \frac{1}{2}$. Using these facts we find

that the mixture density is given by

$$\begin{aligned}\tilde{f}(x|\lambda) &= \sum_{i=0}^{\infty} \mathcal{P}\left(i;\frac{\lambda}{2}\right)\chi^2_{p+2i}(x) = \sum_{i=0}^{\infty} \mathcal{P}\left(i;\frac{\lambda}{2}\right)\mathcal{G}a\left(x;\frac{p+2i}{2},\frac{1}{2}\right) \\ &= \sum_{i=0}^{\infty}\left(\frac{e^{-\lambda/2}\lambda^i}{i!2^i}\right)\left(\frac{1}{2}\right)^{\frac{p+2i}{2}}\frac{1}{\Gamma\left(\frac{p+2i}{2}\right)}x^{\frac{p+2i}{2}-1}e^{-\frac{x}{2}} \\ &= \frac{x^{\frac{p}{2}-1}}{2^{p/2}}\left(\sum_{i=0}^{\infty}\frac{\lambda^i}{i!2^{2i}}\frac{1}{\Gamma\left(\frac{p}{2}+i\right)}x^i\right)e^{-\frac{\lambda+x}{2}}.\end{aligned}$$

We now consider what the expression $I_{\frac{p-2}{2}}(\sqrt{\lambda x})$ evaluates to. Using Equation 18 We find

$$I_{\frac{p-2}{2}}(\sqrt{\lambda x}) = \left(\frac{(\lambda x)^{\frac{p-2}{2}}}{2}\right)^{\frac{p-2}{2}}\sum_{i=0}^{\infty}\frac{\left((\lambda x)/4\right)^i}{i!\Gamma\left(\frac{p}{2}+i\right)} = \frac{(\lambda x)^{\frac{p-2}{2}}}{2^{\frac{p}{2}-1}}\sum_{i=0}^{\infty}\frac{\lambda^i x^i}{i!2^{2i}\Gamma\left(\frac{p}{2}+i\right)}.$$

Thus the sum in the expression for $\tilde{f}(x|\lambda)$ in terms of $I_{\frac{p-2}{2}}(\sqrt{\lambda x})$ is given by

$$\sum_{i=0}^{\infty}\frac{\lambda^i x^i}{i!2^{2i}\Gamma\left(\frac{p}{2}+i\right)} = \frac{2^{\frac{p}{2}-1}}{(\lambda x)^{\frac{p-2}{4}}}I_{\frac{p-2}{2}}(\sqrt{\lambda x}),$$

and we can write $\tilde{f}(x|\lambda)$ as

$$\tilde{f}(x|\lambda) = \frac{x^{\frac{p}{2}-1}}{2^{\frac{p}{2}}}I_{\frac{p-2}{2}}(\sqrt{\lambda x})\frac{2^{\frac{p}{2}-1}}{(\lambda x)^{\frac{p}{4}-\frac{1}{2}}} = \frac{1}{2}\left(\frac{x}{\lambda}\right)^{\frac{p-2}{4}}I_{\frac{p-2}{2}}(\sqrt{\lambda x}),$$

which is the probability density function for the noncentral chi-squared distribution showing the desired equivalence.

Exercise 2.22 (the truncated normal distribution)

Part (b): We will generate a sample from $\mathcal{N}^+(\mu,\sigma^2,a)$ using the given method if the given sample z from the normal $\mathcal{N}(\mu,\sigma^2)$ fall in the region $z \geq a$. This will happen with a probability

$$\frac{1}{\sqrt{2\pi}\sigma}\int_a^{\infty} e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx.$$

If we let $v = -\left(\frac{x-\mu}{\sigma}\right)$ then $dv = -\frac{dx}{\sigma}$ and the above probability can be written as

$$\int_{-\infty}^{\frac{\mu-a}{\sigma}}\frac{1}{\sqrt{2\pi}}e^{-\frac{v^2}{2}}dv \equiv \Phi\left(\frac{\mu-a}{\sigma}\right),$$

the result we were trying to show. Since this is the probability of one acceptance then by recalling the probabilities of the negative binomial random variable the number of trials (on average) to get one acceptance is the reciprocal of this number or

$$\frac{1}{\Phi\left(\frac{\mu-a}{\sigma}\right)}.$$

If a is large then $\Phi\left(\frac{\mu-a}{\sigma}\right)$ is small and the number of simulations to get one valid sample will be large.

Part (c): We find

$$\frac{f(x)}{g(x)} = \left(\frac{\exp\left(-\frac{x^2}{2}\right)}{\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x-\bar{\mu})^2}{2}\right)}\right)\mathbb{I}_{x\geq a} = \sqrt{2\pi}\exp\left(-\frac{x^2}{2}+\frac{1}{2}(x-\bar{\mu})^2\right)\mathbb{I}_{x\geq a}$$
$$= \sqrt{2\pi}\exp\left(-x\bar{\mu}+\frac{1}{2}\bar{\mu}^2\right)\mathbb{I}_{x\geq a}\,.$$

We next want to maximize this as a function of x over the domain $x \geq a$. As $x \to +\infty$ when $\bar{\mu} > 0$ the above expression decreases to zero. Thus the largest value for this ratio happens when $x = a$, where it equals

$$\sqrt{2\pi}\exp\left(-a\bar{\mu}+\frac{1}{2}\bar{\mu}^2\right)\,.$$

Now with this expression we want to pick the value of $\bar{\mu}$ that *minimizes* this expression. This will happen at the minimum of the argument of the exponential or $-a\bar{\mu}+\frac{1}{2}\bar{\mu}^2$. Taking the derivative of this expression, setting the result equal to zero and solving for $\bar{\mu}$ we get $\bar{\mu} = a$. Since the second derivative of this linear expression with respect to $\bar{\mu}$ is $+1 > 0$ the value $\bar{\mu} = a$ is a minimum. Thus this minimum is given by

$$\sqrt{2\pi}\exp\left\{-\frac{a^2}{2}\right\}\,.$$

Random draws from a truncated normal distribution are generated using the accept-reject algorithm with a Gaussian for the candidate density are performed in the `R` script `ex_2_21.R`. When that script is run it produces the plot shown in Figure 16 (left).

Part (d): We find

$$\frac{f(x)}{g(x)} = \left(\frac{\exp\left(-\frac{x^2}{2}\right)}{\alpha\exp\left(-\alpha(x-a)\right)}\right)\mathbb{I}_{x\geq a} = \alpha\exp\left(-\frac{x^2}{2}+\alpha(x-a)\right)\mathbb{I}_{x\geq a}\,.$$

We need to pick the value of x that maximizes the right-hand-side of the above. Taking the derivative of the argument of the exponent gives $-x+\alpha = 0$ or $x = \alpha$. The second derivative of this argument is $-1 < 0$ showing that $x = \alpha$ is indeed a maximum. Since $x \geq a$ by the requirements of the truncated normal distribution in order to use the solution $x = \alpha$ we must have $\alpha \geq a$. When we put in $x = \alpha$ we get

$$\frac{f(x)}{g(x)} \leq \frac{1}{\alpha}\exp\left\{\frac{\alpha^2}{2}-\alpha a\right\}\,. \tag{19}$$

Next we want to pick α such that $\alpha \geq a$ and the above expression is a *minimum*. As α increases the expression $\exp\left\{\frac{\alpha^2}{2}-\alpha a\right\}$ increases while the fraction $\frac{1}{\alpha}$ decreases, thus there maybe an α point, such that the above expression is a minimum. Taking the first derivative of the right-hand-side with respect to α and setting the result equal to zero gives

$$-\frac{1}{\alpha^2}\exp\left\{\frac{\alpha^2}{2}-\alpha a\right\}+\frac{1}{\alpha}\exp\left\{\frac{\alpha^2}{2}-\alpha a\right\}(\alpha-a) = 0\,.$$

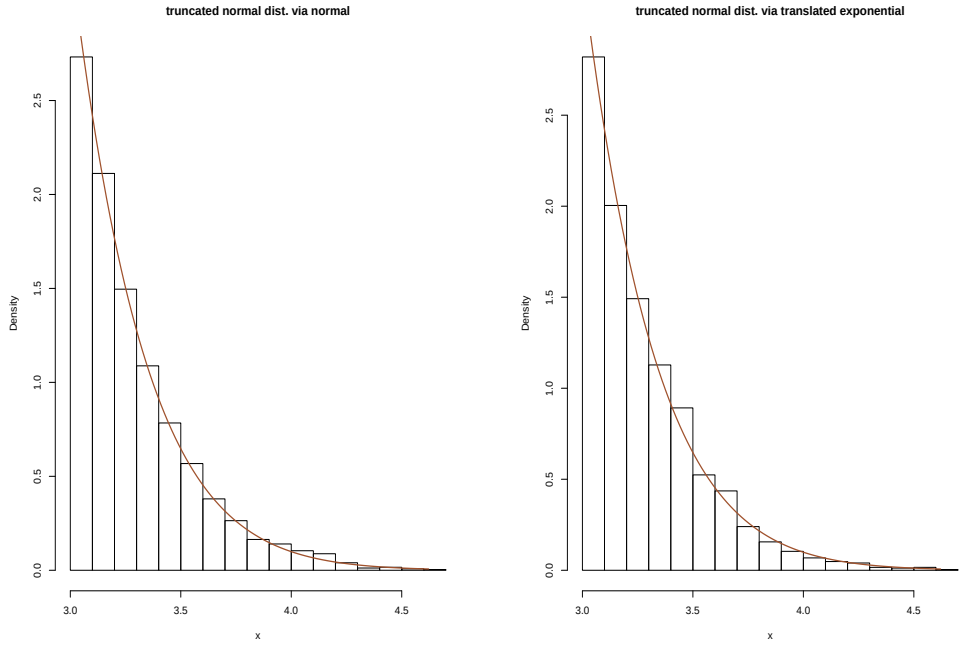

Figure 16: **Left:** The normalized density $f(x)$ (in brown) for Exercise 2.21 and a histogram of samples drawn using the accept reject algorithm and a Gaussian candidate density **Right:** Using the translated exponential density as the candidate density. These plots look the same, showing that both methods are drawing correct samples from $f(x)$.

When we solve for α we get

$$\alpha = \frac{a \pm \sqrt{a^2+4}}{2} .$$

To make sure that $\alpha \geq a$ we must take the positive sign in the above expression. With this value for α specified we can evaluate the right-hand-side of the fraction $\frac{f(x)}{g(x)}$ using Equation 19 do determine the upper bound M needed in the accept-reject algorithm. Random draws from a truncated normal distribution are generated using the accept-reject algorithm with a translated exponential distribution for the candidate density are computed in the `R` script `ex_2_21.R`. When that script is run it produces the plot shown in Figure 16 (right).

Exercise 2.23 (Monte-Carlo samples from the posterior distribution)

Part (a): The bound M in the accept-reject algorithm when the target density is $\pi(\theta|x)$ and the candidate density is $\pi(\theta)$ must satisfy $\frac{\pi(\theta|x)}{\pi(\theta)} \leq M$. From the Bayes' rule statement and ignoring constants (that don't affect the results of the accept-reject algorithm) we have

$$\frac{\pi(\theta|x)}{\pi(\theta)} = \prod_i f(x_i|\theta) .$$

Thus the smallest value (and optimal) M can be is when we maximize the right-hand-side with respect to θ which is the maximum likelihood estimator (MLE). Thus

$$\frac{\pi(\theta|x)}{\pi(\theta)} \leq \max_\theta \prod_i f(x_i|\theta)\,.$$

Part (b): If $X_i \sim N(\theta, 1)$ and $\theta \sim C(0,1) = \frac{1}{\pi(1+\theta^2)}$, then $\pi(\theta|x)$ is proportional to

$$\pi(\theta|x) \propto \frac{1}{\pi(1+\theta^2)} \frac{1}{(2\pi)^{n/2}} \prod_{i=1}^n e^{-\frac{(x_i-\theta)^2}{2}}\,.$$

We need to find the MLE for the factor $\prod_{i=1}^n e^{-\frac{(x_i-\theta)^2}{2}}$. By taking natural logarithms of this expression, taking the derivative of the result with respect to θ, setting the result equal to zero, and solving for θ we find that the maximum likelihood estimate is given by $\bar{\theta} = \frac{1}{n}\sum_{i=1}^n x_i$ or the sample average. Then the value of M to use in the accept-reject algorithm is given by

$$M = \prod_{i=1}^n f(x_i|\bar{\theta}) = \frac{1}{(2\pi)^{n/2}} \prod_{i=1}^n e^{-\frac{(x_i-\bar{\theta})^2}{2}}\,.$$

Part (b): In the `R` script `ex_2_23.R` we perform the required computations for this problem. We do this for $n = 10$, $n = 100$, and $n = 1000$. Since for each value of n we are getting some number of samples from the posteriori distribution, $\pi(\theta|x)$, we get a distribution of possible values of θ. We can evaluate how our estimation procedure is performing by computing the mean and variance of this empirical posteriori distribution. By running the above code several times for different values of n we find

```
n=  10; mean(theta samples)=   3.126708; var(theta samples)=   0.099886
n= 100; mean(theta samples)=   2.959917; var(theta samples)=   0.010182
n= 300; mean(theta samples)=   2.875398; var(theta samples)=   0.003364
n= 500; mean(theta samples)=   2.999528; var(theta samples)=   0.002046
```

showing improvement and convergence to 3 as n increases.

Chapter 3 (Monte Carlo Integration)

Exercise 3.1 (Monte Carlo convergence)

Part (a): We work this exercise in the `R` file `ex_3_1.R`. The three integrands in the numerator/denominator are plotted in Figure 17 (top/bottom). We see that the integral of the integrand of the denominator should be equal for each value of x, while the value of the numerator will change depending on x. Note that we can write the given expression for $\delta(x)$ in the form

$$\delta(x) = \frac{\int_{-\infty}^{\infty} \theta \left(\frac{1}{\pi(1+\theta^2)}\right)\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\theta)^2}{2}}\right) d\theta}{\int_{-\infty}^{\infty} \left(\frac{1}{\pi(1+\theta^2)}\right)\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\theta)^2}{2}}\right) d\theta} ,$$

which shows explicitly that we have the expressions for a Cauchy *and* a normal density in each integrand. Thus we can approximate these two integrals by either drawing samples from a Cauchy distribution (and inserting them into a normal density) or drawing samples from a normal density (and inserting them into a Cauchy distribution). When we do the first of these two procedures for 10^4 random Cauchy draws we get

```
[1] "MC integration estimation using cauchy draws ..."
[1] "x=   0; N=    10000; MC estimate=   0.001614"
[1] "x=   2; N=    10000; MC estimate=   1.263821"
[1] "x=   4; N=    10000; MC estimate=   3.411461"
```

The exact numbers one gets depends on the value of the numerical random seed.

Part (b): To monitor the convergence of a Monte Carlo integration we can generate the estimates of the integrals (with their standard errors) as a function of the number of samples used to compute the estimate. Rather than do this for both he numerator and the denominator of the function $\delta(x)$ for the remaining parts of this problem I'll just consider convergence of the integral in the numerator for various values of x. I'll look at estimating the integral using each of the Monte Carlo methods discussed above. In Figure 18 we plot the integrals estimate using Cauchy draws in black (with error boundaries in red) and the integral estimate using normal draws in gray (with pink error bars). In general, it looks like the normal random draws give better estimates much sooner.

Notes on Example 3.4

For the estimator

$$\hat{\Phi}(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{x_i \leq t} ,$$

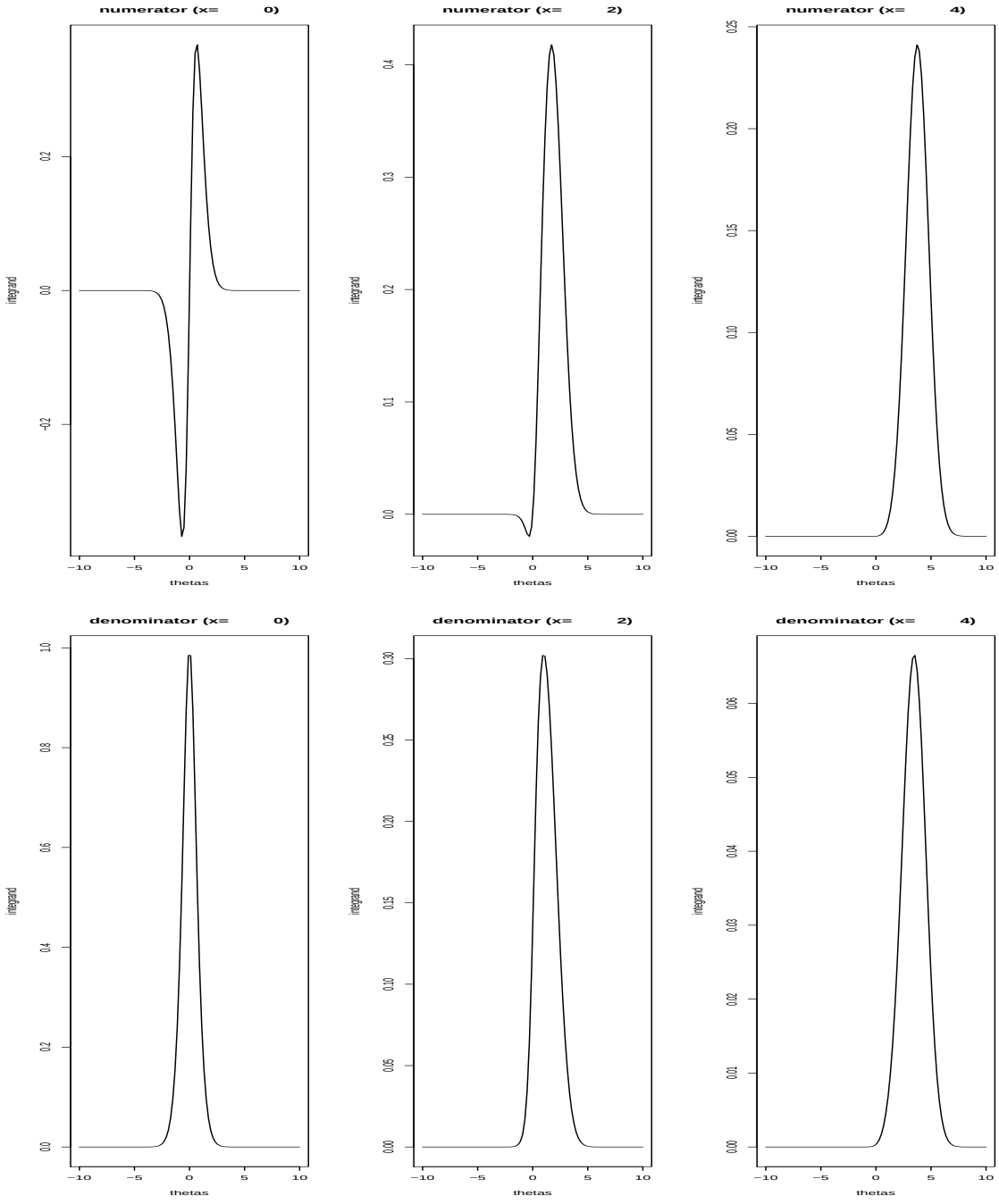

Figure 17: The integrands for Exercise 3.1.

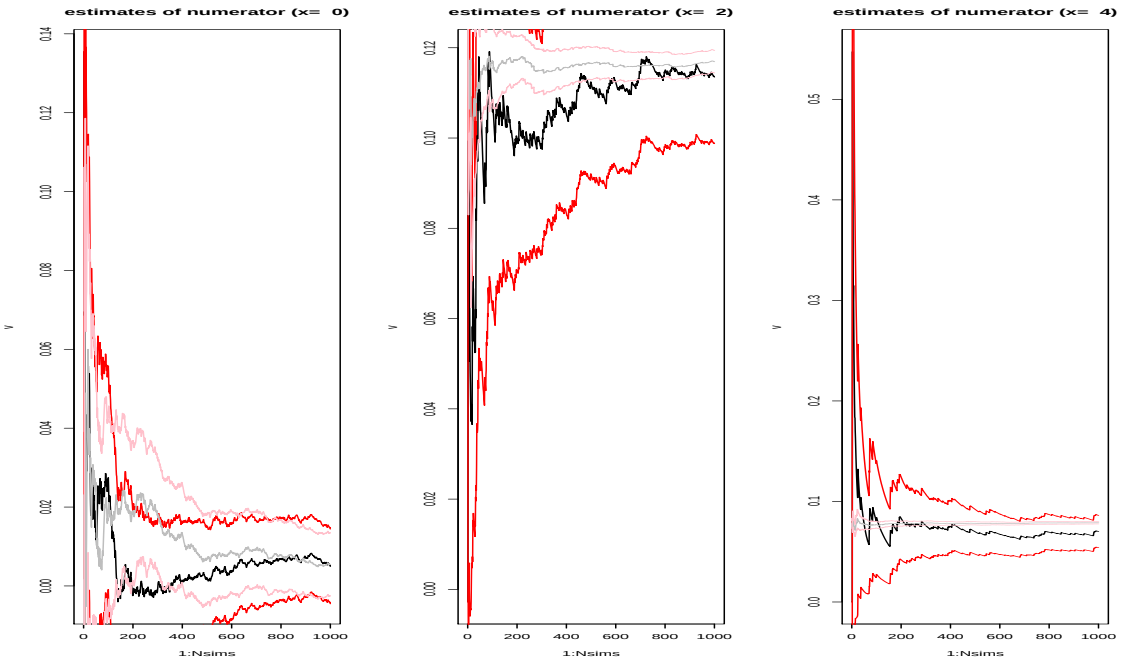

Figure 18: Plots of the MC convergence of the integral in the numerator for Exercise 3.1 using two different methods.

we can compute the exact variance using

$$\text{Var}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}_{x_i\leq t}\right) = \frac{1}{n^2}\sum_{i=1}^{n}\text{Var}(\mathbb{I}_{x_i\leq t}) = \frac{n}{n^2}\text{Var}(\mathbb{I}_{x_i\leq t}) = \frac{1}{n}\Phi(t)(1-\Phi(t))\,,$$

since $\mathbb{I}_{X_i\leq t}$ is a Bernoulli RV (each X_i is random). Since $\Phi(0) = 1/2$ we have that when $t = 0$ the variance of this estimator has variance approximately given by $\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)\frac{1}{n} = \frac{1}{4n}$.

Exercise 3.2 (the variance of the ratio $\frac{\mathbb{I}_{X_i\leq t}}{\Phi(t)}$)

We compute

$$\text{Var}\left(\frac{\mathbb{I}_{X_i\leq t}}{\Phi(t)}\right) = \frac{1}{\Phi(t)^2}\text{Var}\left(\mathbb{I}_{X_i\leq t}\right) = \frac{1}{\Phi(t)^2}\Phi(t)(1-\Phi(t)) = \frac{1-\Phi(t)}{\Phi(t)}\,,$$

since $\mathbb{I}_{X_i\leq t}$ is a Bernoulli RV and using the formula for what the variance of a Bernoulli is given by. Since $\Phi(t)\to 0$ as $t\to -\infty$ this variance goes to $+\infty$.

If we are attempting to estimate the error in the estimate of $\Phi(t)$ using the approximation

$\hat{\Phi}(t)$ we can compute the relative error from

$$\begin{aligned}\text{Var}\left(\frac{\hat{\Phi}-\Phi}{\Phi}\right) &= \text{Var}\left(\frac{\hat{\Phi}}{\Phi}-1\right) = \text{Var}\left(\frac{\hat{\Phi}}{\Phi}\right) \\ &= \text{Var}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\mathbb{I}_{X_i\leq t}}{\Phi(t)}\right) = \frac{n}{n^2}\text{Var}\left(\frac{\mathbb{I}_{X_i\leq t}}{\Phi(t)}\right) \\ &= \frac{1}{n}\left(\frac{1-\Phi(t)}{\Phi(t)}\right) .\end{aligned}$$

To get a variance less than some accuracy (say 10^{-8}) we would require n such that

$$\frac{1}{n}\left(\frac{1-\Phi(t)}{\Phi(t)}\right) < 10^{-8} .$$

We can solve for n in the above expression to determine the number of samples to use.

Exercise 3.3 (tail probabilities)

To evaluate the tail probability we need to evaluate the integral

$$\Pr(X > 20) = \int_{20}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy .$$

To do that we let $v = \frac{1}{y}$. Then $y = \frac{1}{v}$ so $dy = -\frac{1}{v^2}dv$. The limits of the integral transform as $y = +\infty$ we have $v = 0$ and when $y = +20$ we have $v = \frac{1}{20}$. Thus we get the above is equal to

$$\Pr(X > 20) = \int_{0}^{\frac{1}{20}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2v^2}} \left(\frac{dv}{v^2}\right) = \frac{1}{\sqrt{2\pi}} \int_{0}^{\frac{1}{20}} \frac{e^{-\frac{1}{2v^2}}}{v^2} dv .$$

This last integral can be approximated using random draws from a $\mathcal{U}(0, \frac{1}{20})$ distribution and evaluating them in the integrand of the integral. This is done in the `R` program `ex_3_3.R`. When that script is run we get the plot given in Figure 19.

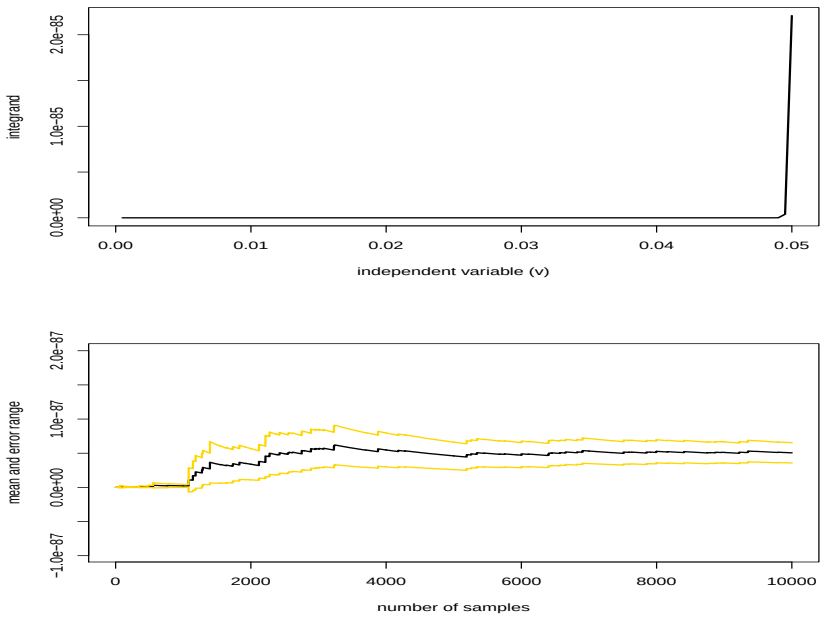

Figure 19: Plots of the integrand (top) and the MC convergence (as a function of samples) of the integral given in Exercise 3.3 (bottom).

References

[1] S. Weisberg. *Applied Linear Regression.* Wiley, New York, 1985.