Homework 2

Ginevra Carbone

DAAG Chapter 3

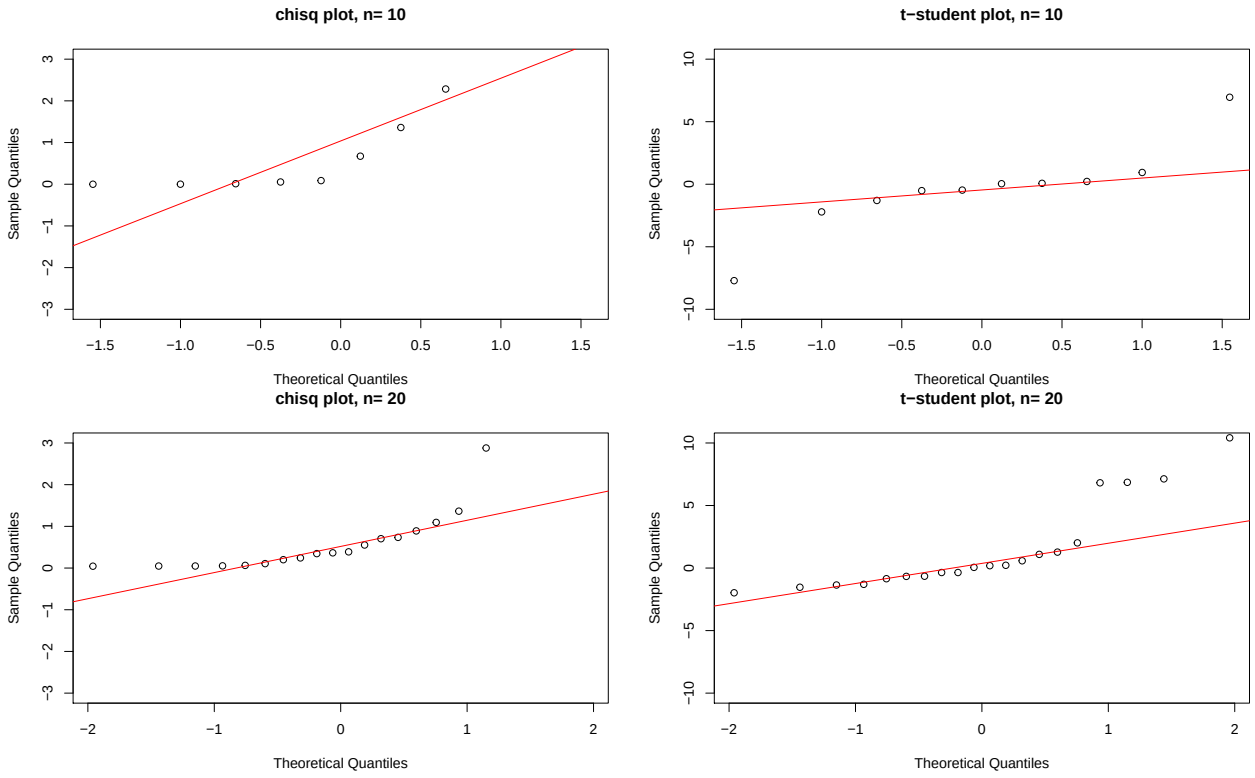
Exercise 10

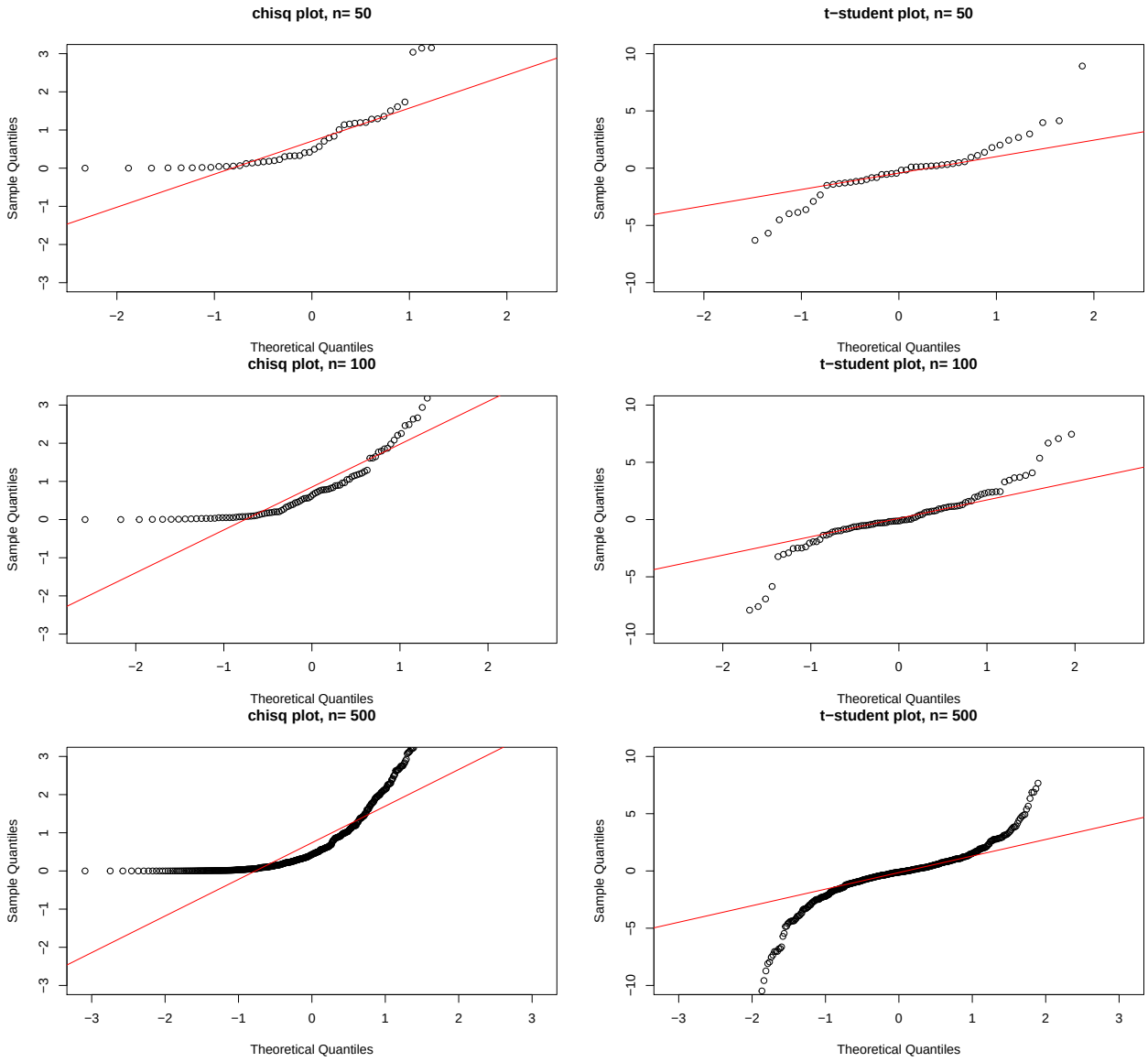
This exercise investigates simulation from other distributions. The statement `x <- rchisq(10, 1)` generates 10 random values from a chi-squared distribution with one degree of freedom. The statement `x <- rt(10, 1)` generates 10 random values from a t-distribution with one degree of freedom. Make normal probability plots for samples of various sizes from each of these distributions. How large a sample is necessary, in each instance, to obtain a consistent shape?

```
library(DAAG)
```

```
## Loading required package: lattice
```

```
set.seed(234)
par(mfrow=c(1,2))

for (n in c(10, 20, 50, 100, 500)) {
  chisq <- rchisq(n,1)
  qqnorm(chisq, ylim= c(-3,3),main = paste("chisq plot, n=", n))
  qqline(chisq, col=2)
  tstudent <- rt(n,1)
  qqnorm(tstudent, ylim= c(-10,10), main = paste("t-student plot, n=", n))
  qqline(tstudent, col=2)
}
```






```
par(mfrow=c(1,1))
```

I would say that in both cases the shape starts to be consistent with a sample of size 100.

Exercise 11

The following data represent the total number of aberrant crypt foci (abnormal growths in the colon) observed in seven rats that had been administered a single dose of the carcinogen azoxymethane and sacrificed after six weeks (thanks to Ranjana Bird, Faculty of Human Ecology, University of Manitoba for the use of these data): 87 53 72 90 78 85 83 Enter these data and compute their sample mean and variance.

```
data <- c(87, 53, 72, 90, 78, 85, 83)
mean(data)
```

```
## [1] 78.28571
```

```
var(data)
```

```
## [1] 159.9048
```

Is the Poisson model appropriate for these data? To investigate how the sample variance and sample mean differ under the Poisson assumption, repeat the following simulation experiment several times:

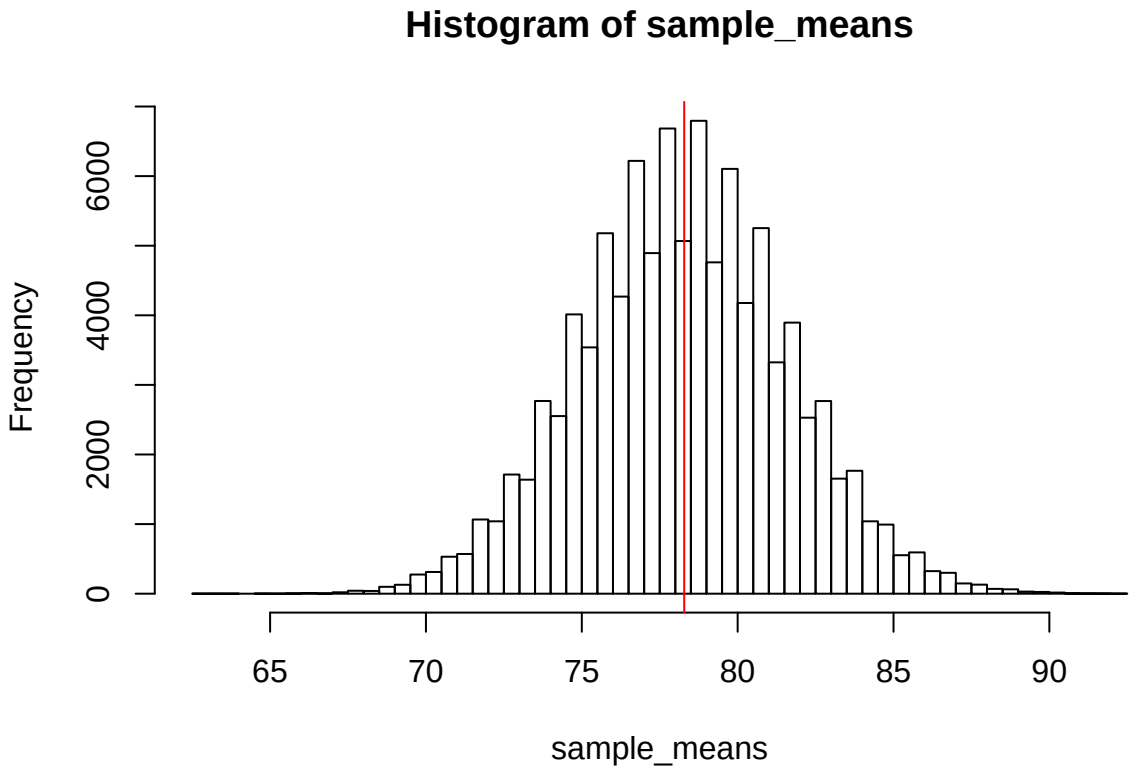
```
library(DAAG)

n <- 100000
sample_means <- replicate(n, mean(rpois(7, 78.3)))
sample_vars <- replicate(n, var(rpois(7, 78.3)))

mean(sample_means)
```

```
## [1] 78.29177
```

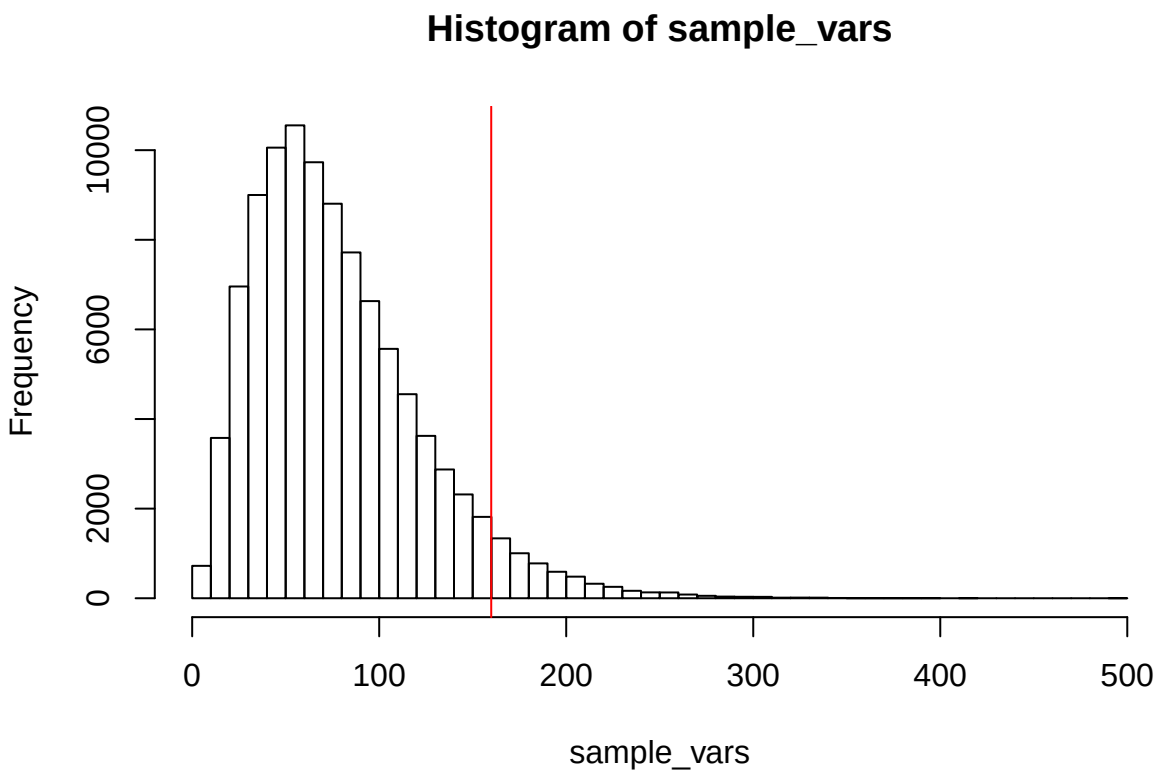
```
mean(sample_vars)
```

```
## [1] 78.05305
```

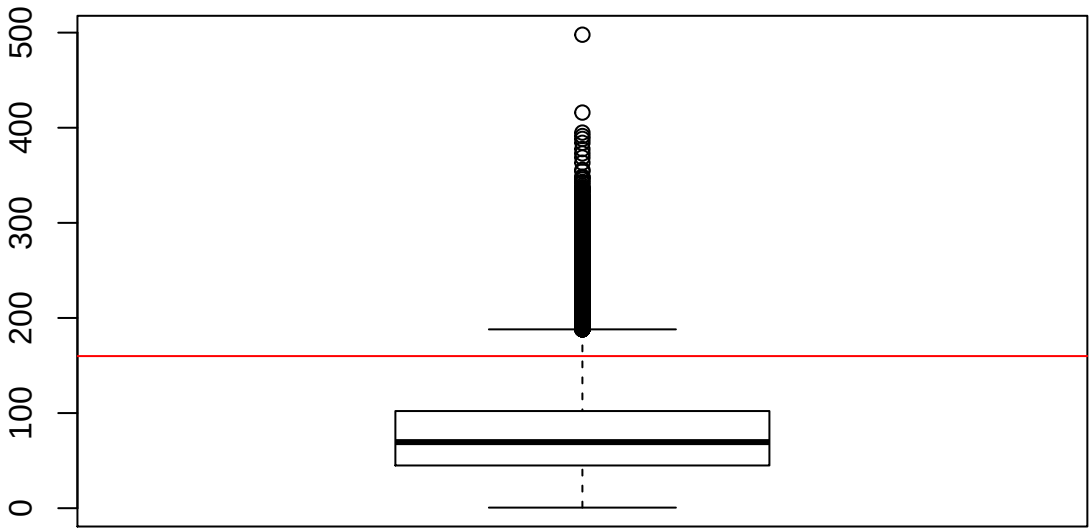
```
hist(sample_means, breaks = 50); abline(v=mean(data), col=2)
```




```
hist(sample_vars, breaks = 50); abline(v=var(data), col=2)
```




```
# boxplot(sample_means); abline(h=mean(data), col=2)
boxplot(sample_vars); abline(h=var(data), col=2)
```



The Poisson assumption has been done on the exact initial mean, so, as one would expect, the histogram of sample means is consistent with it. On the contrary, Poiosson distribution on this mean is not consistent with the original variance, and we can notice this fact from both the histogram and the boxplot of the random distribution.

Exercise 13

A Markov chain for the weather in a particular season of the year has the transition matrix, from one day to the next:

$$\begin{pmatrix} & Sun & Cloud & Rain \\ Sun & 0.6 & 0.2 & 0.2 \\ Cloud & 0.2 & 0.4 & 0.4 \\ Rain & 0.4 & 0.3 & 0.3 \end{pmatrix}$$

It can be shown, using linear algebra, that in the long run this Markov chain will visit the states according to the stationary distribution:

- Sun 0.641
- Cloud 0.208
- Rain 0.151

A result called the ergodic theorem allows us to estimate this distribution by simulating the Markov chain for a long enough time.

(a) Simulate 1000 values, and calculate the proportion of times the chain visits each of the states. Compare the proportions given by the simulation with the above theoretical proportions.

```
A = matrix(c(0.6,0.2,0.2,0.2,0.4,0.4,0.4,0.3,0.3), nrow=3, ncol=3, byrow=TRUE)

Markov <- function (N=1000, initial.value=1, A) {
  X <- numeric(N)
  X[1] <- initial.value + 1
  n <- nrow(A)
  for (i in 2:N){
  X[i] <- sample(1:n, size=1, prob=A[X[i-1], ])}
  X - 1
}

simulation <- Markov(1000000, 1, A)

Sun <- mean(simulation == 0)
Cloud <- mean(simulation == 1)
Rain <- mean(simulation == 2)

Sun
```

```
## [1] 0.428053
```

```
Cloud
```

```
## [1] 0.28618
```

```
Rain
```

```
## [1] 0.285767
```

The proportions are totally different from the ones showed in the textboox, which of course are wrong, since the second and third columns in the state matrix coincide.

(b) Here is code that calculates rolling averages of the proportions over a number of simulations and plots the result. It uses the function `rollmean()` from the zoo package.

```
library(zoo)
```

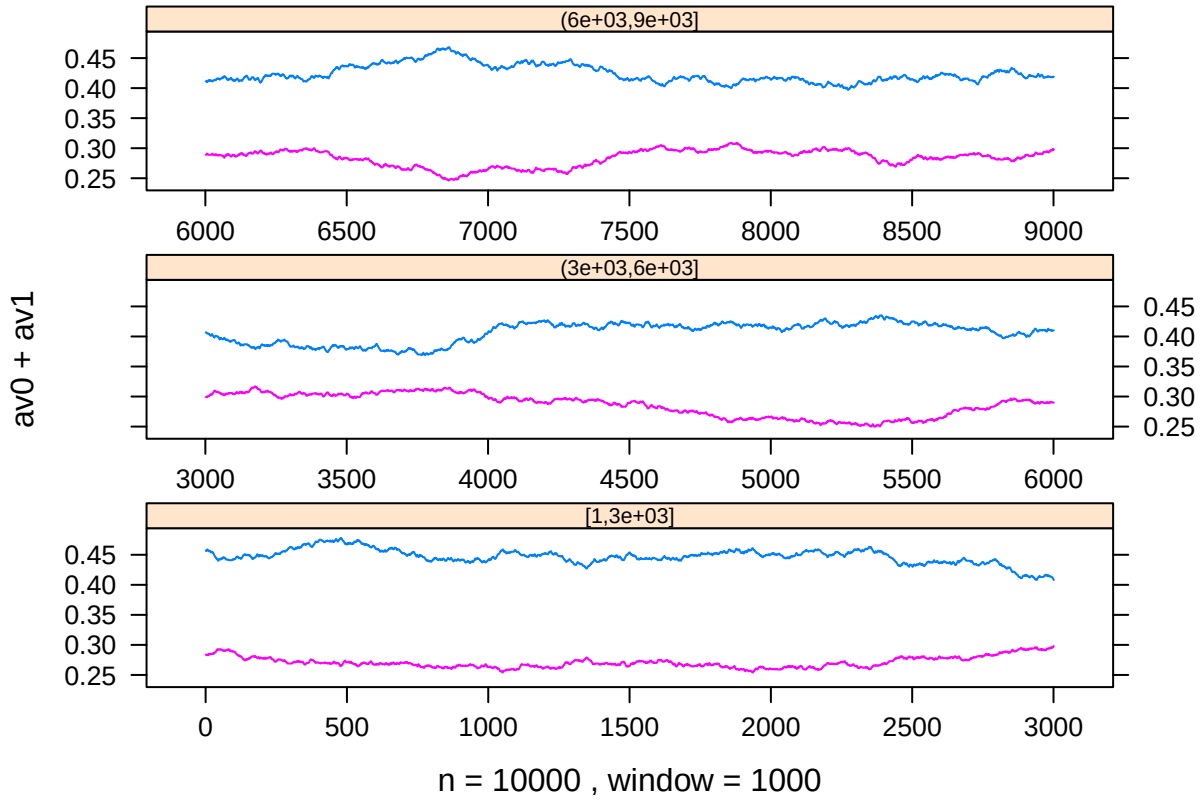
```
##
## Attaching package: 'zoo'
```

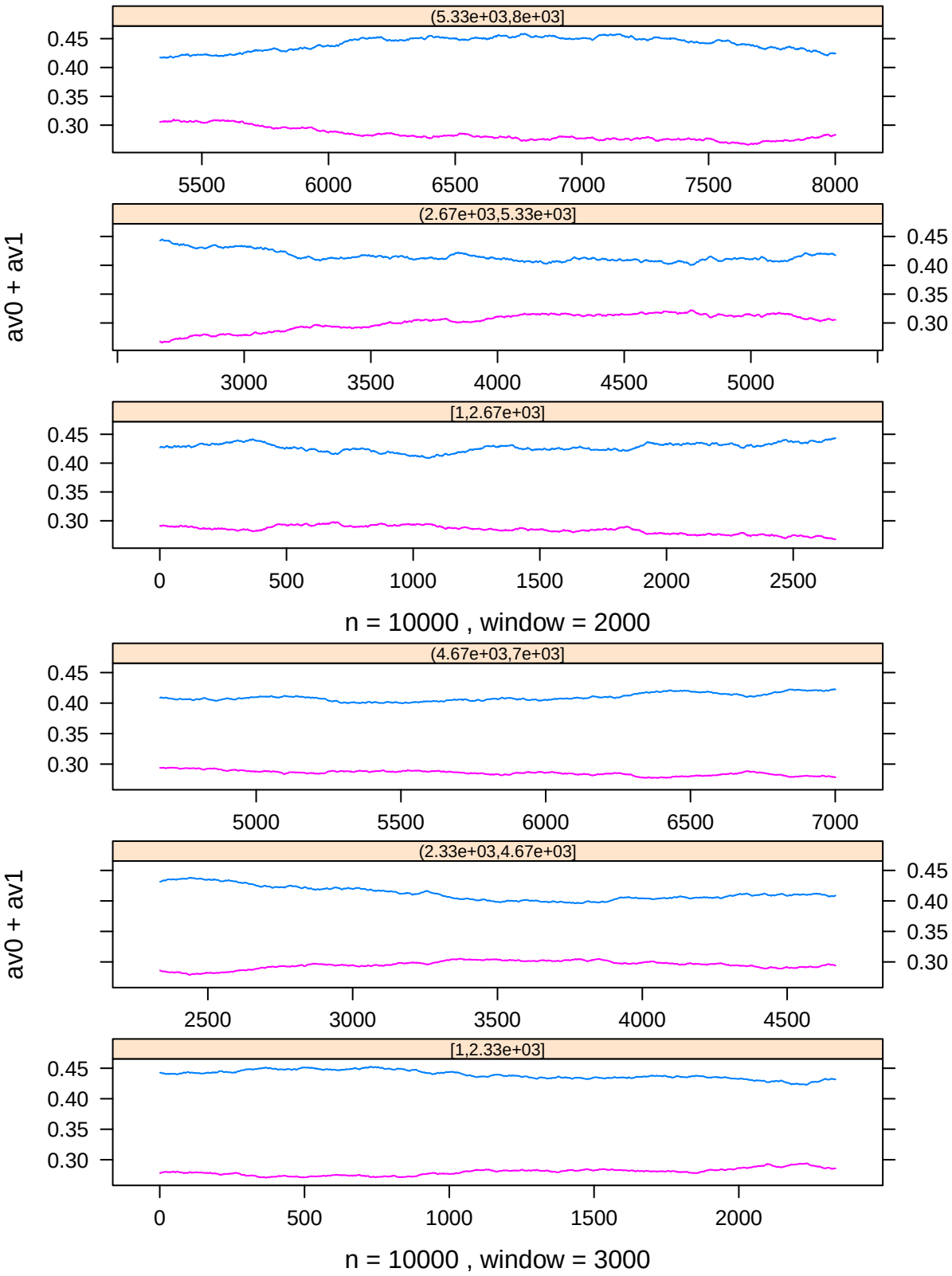
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

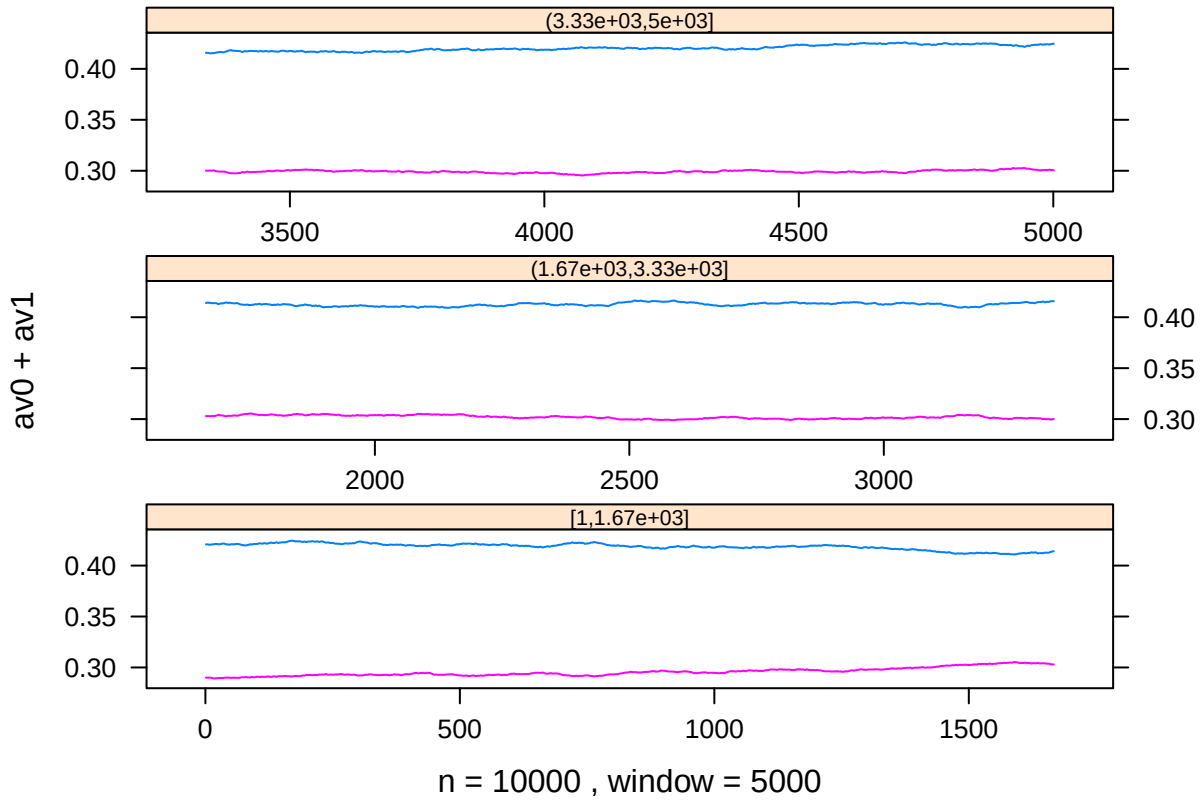
```
library(lattice)
plotmarkov <- function(n=10000, start=0, window=100, transition=Pb, npanels=5){
  xc2 <- Markov(n, start, transition)
  mav0 <- rollmean(as.integer(xc2==0), window)
  mav1 <- rollmean(as.integer(xc2==1), window)
  npanel <- cut(1:length(mav0), breaks=seq(from=1, to=length(mav0), length=npanels+1), include.lowest=TR
  df <- data.frame(av0=mav0, av1=mav1, x=1:length(mav0),
  gp=npanel)
  print(xyplot(av0+av1 ~ x | gp, data=df ,layout=c(1,npanels),
  type="l", par.strip.text=list(cex=0.65),
  scales=list(x=list(relation="free")), xlab = paste("n =",n,", window =",window)))
}
```

Try varying the number of simulations and the width of the window. How wide a window is needed to get a good sense of the stationary distribution? This series settles down rather quickly to its stationary distribution (it "burns in" quite quickly). A reasonable width of window is, however, needed to give an accurate indication of the stationary distribution.

```
for(window in c(1000, 2000, 3000, 5000)){
  plotmarkov(transition = A, n=10000, window= window, npanels=3)
}
```




(5.33e+03,8e+03]
(2.67e+03,5.33e+03]
[1,2.67e+03]
av0 + av1
n = 10000 , window = 2000
(4.67e+03,7e+03]
(2.33e+03,4.67e+03]
[1,2.33e+03]
av0 + av1
n = 10000 , window = 3000



Taking a look at these plots, a window of size 3000 should be enough.

DAAG Chapter 4

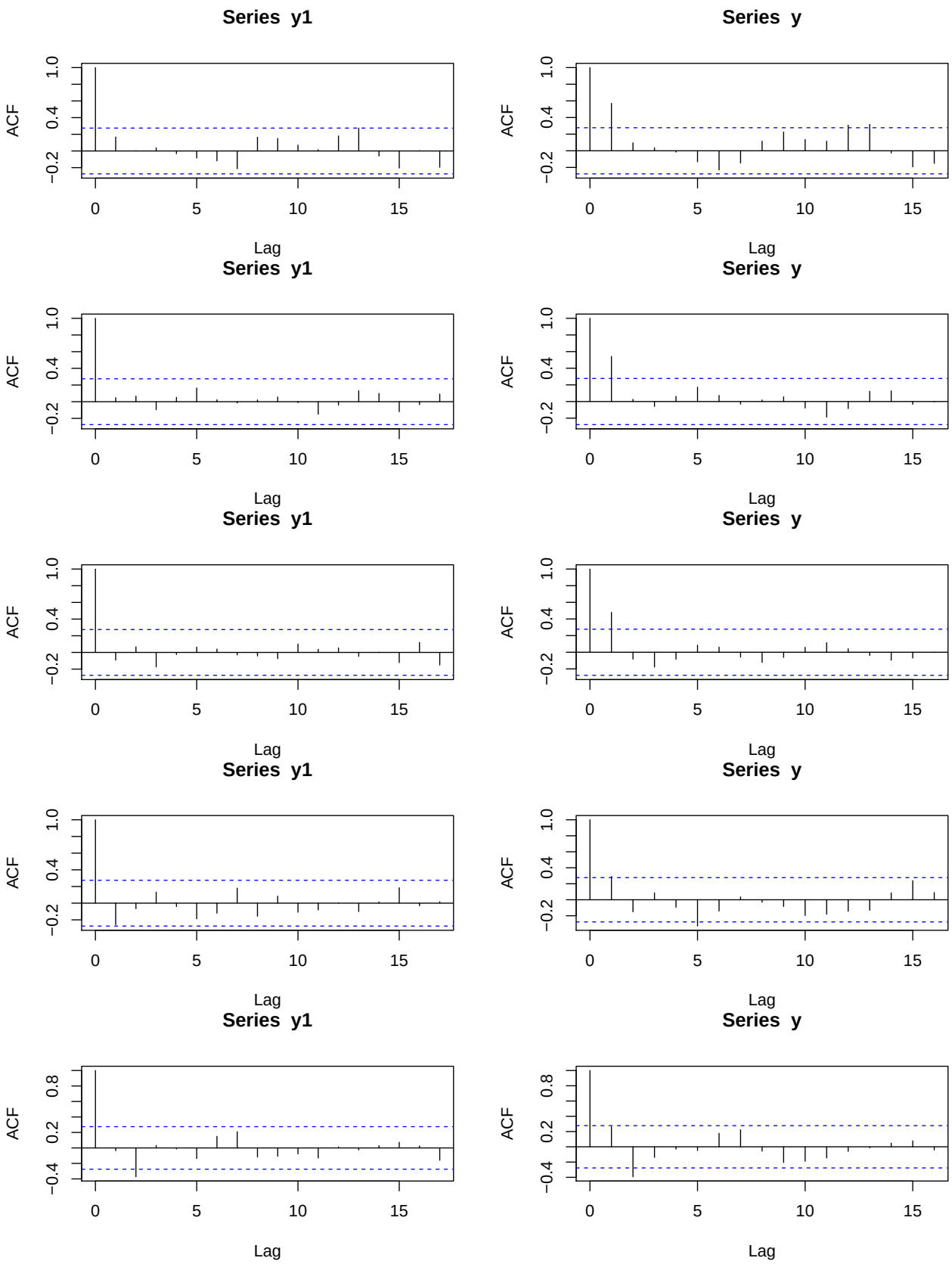
Exercise 6

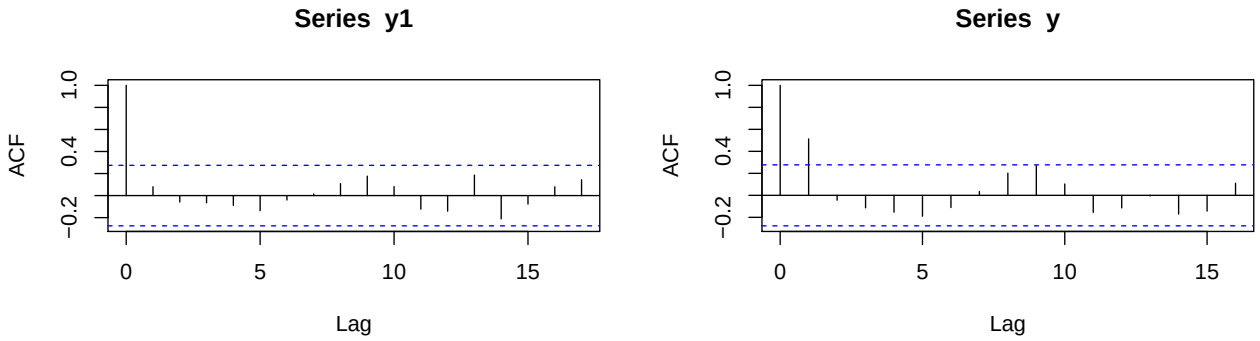
Here we generate random normal numbers with a sequential dependence structure.

```
y1 <- rnorm(51)
y <- y1[-1] + y1[-51]
acf(y1)
# acf is 'autocorrelation function'
# (see Chapter 9)
acf(y)
```

Repeat this several times. There should be no consistent pattern in the `acf` plot for different random samples `y1`. There will be a fairly consistent pattern in the `acf` plot for `y`, a result of the correlation that is introduced by adding to each value the next value in the sequence.

```
par(mfrow=c(1,2))
for(i in seq(1:6)){
  y1 <- rnorm(51)
  acf(y1)

  y <- y1[-1] + y1[-51]
  acf(y)
}
```


Series y1
Series y
ACF
Lag
1.0
0.4
−0.2
0.8
0.2
−0.4
0
5
10
15




```
par(mfrow=c(1,1))
```

Exercise 7

Create a function that does the calculations in the first two lines of the previous exercise. Put the calculation in a loop that repeats 25 times. Calculate the mean and variance for each vector `y` that is returned. Store the 25 means in the vector `av`, and store the 25 variances in the vector `v`. Calculate the variance of `av`.

```
sums <- function(x) x[-1]+x[-51]
av <- array(0)
v <- array(0)
for (i in seq(1:25)) {
  y1 = rnorm(51)
  y = sums(y1)
  av[i] = mean(y)
  v[i] = var(y)
}
var(av)
```

```
## [1] 0.06972232
```

Exercise 9

In a study that examined the use of acupuncture to treat migraine headaches, consenting patients on a waiting list for treatment for migraine were randomly assigned in a 2:1:1 ratio to acupuncture treatment, a "sham" acupuncture treatment in which needles were inserted at non-acupuncture points, and waiting-list patients whose only treatment was self-administered (Linde et al., 2005). (The "sham" acupuncture treatment was described to trial participants as an acupuncture treatment that did not follow the principles of Chinese medicine.) Analyze the following two tables. What, in each case, are the conclusions that should be drawn from the analyses? Comment on implications for patient treatment and for further research:

(a) Outcome is classified according to numbers of patients who experienced a greater than 50% reduction in headaches over a four-week period, relative to a pre-randomization baseline:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
##     intersect, setdiff, setequal, union
```

```
acupuncture <- c(74,71)
sham_acup <- c(43,38)
wait_list <- c(11,65)
df <- data.frame(acupuncture, sham_acup, wait_list, row.names = c(">=50%", "<50%"))
# sum <- summarise_all(df, funs(sum))
# mean <- summarise_all(df, funs(mean))
# df <- bind_rows(df, mean, sum)
row.names(df) <- c(">=50%", "<50%")
df
```

```
##       acupuncture sham_acup wait_list
## >=50%          74        43        11
## <50%           71        38        65
```

```
# % over treatment
# sum_treated <- sum$acupuncture + sum$sham_acup
percentages <- data.frame(perc_chinese=acupuncture/sum(acupuncture), perc_sham=sham_acup/sum(sham_acup)
row.names(percentages) <- c(">=50%", "<50%")
percentages
```

```
##       perc_chinese perc_sham perc_wait
## >=50%    0.5103448 0.5308642 0.1447368
## <50%     0.4896552 0.4691358 0.8552632
```

The table above show the percentages of people falling in each category over the number of people who received a certain treatment. Over 51% of people experienced a reduction in headaches greater than 50% after being treated with one of the aucpunture treatments, while 85% of people in the waiting list experienced a reduction in headaches smaller than 50%. This suggests that both treatments have an effect on the reduction in headaches experienced, but a smaller reduction happens anyways over time without any treatment.

(b) Patients who received the acupuncture and sham acupuncture treatments were asked to guess their treatment. Results were:

```
acupuncture <- c(82, 17, 30)
sham_acup <- c(30,26,16)
df <- data.frame(acupuncture, sham_acup)
sum <- summarise_all(df, funs(sum))
row.names(df) <- c("Chinese", "Other", "Don't know")
df
```

```
##            acupuncture sham_acup
## Chinese             82        30
## Other               17        26
## Don't know          30        16
```

```
# % over treatment
percentages <- data.frame(perc_chinese=acupuncture/sum(acupuncture), perc_sham=sham_acup/sum(sham_acup)
row.names(percentages) <- c("Chinese", "Other", "Don't know")
percentages
```

```
##            perc_chinese perc_sham
## Chinese       0.6356589 0.4166667
## Other         0.1317829 0.3611111
## Don't know    0.2325581 0.2222222
```

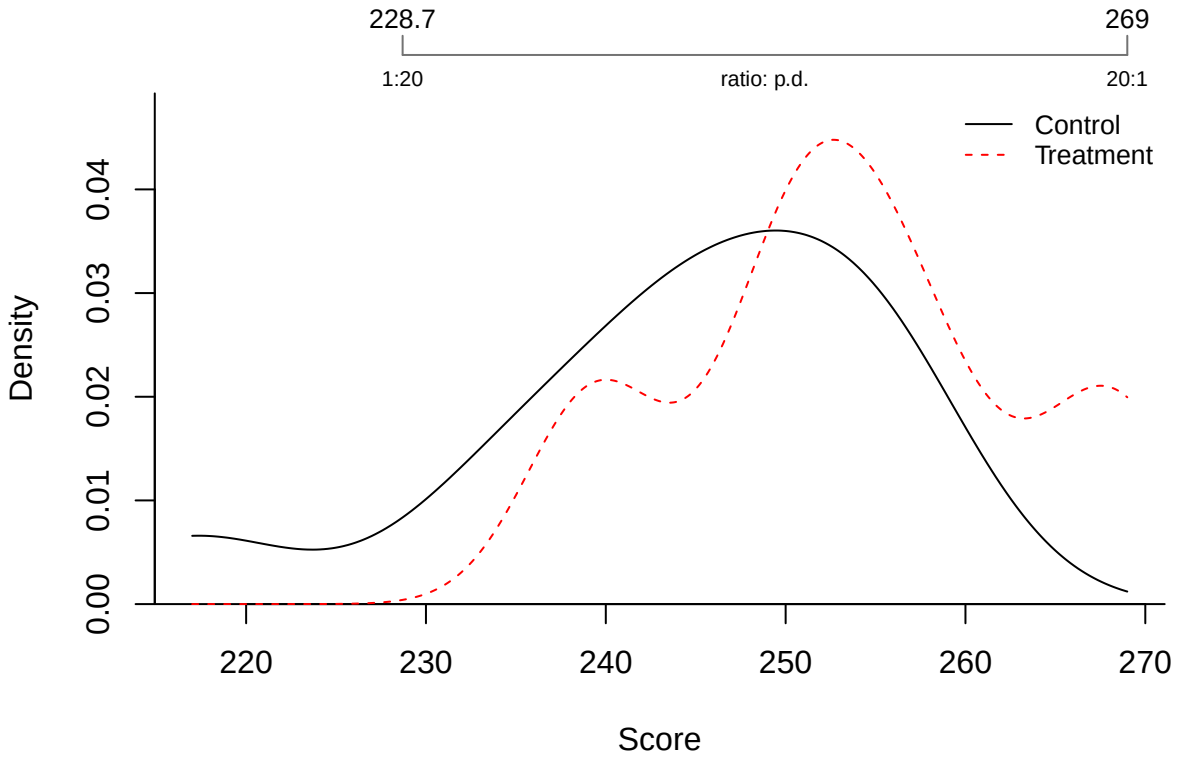
63% of people treated with acupuncture guessed their treatment, while only 41% of people treated with sham

aucpunture guessed it. 22% of people in both cases could not guess the treatment. This suggest that the majority of people were able to guess the correct treatment.

Exercise 13

The function `overlapDensity()` in the DAAG package can be used to visualize the unpaired version of the t-test. Type in

```
library(DAAG)
## Compare densities for ambient & heated: list two65 (DAAG)
with(two65, overlapDensity(ambient, heated))
```




```
## [1] 228.7025 269.0000
```

```
# Do overlapDensity(ambient, heated) with ambient and heated
# taken, if not found elsewhere, from the columns of two65
```

in order to observe estimates of the stretch distributions of the ambient (control) and heated (treatment) elastic bands.

This plot estimates densities for the stretch of ambient and heated elastic bands, and compares them. Of course this kind of comparison only works if the two samples have similar scale, which is indeed the case.

CS Chapter 3

Exercise 3.3

Rewrite the following, replacing the loop with efficient code:

```
n <- 100000; z <- rnorm(n)
zneg <- 0;j <- 1
```

```
# given function
system.time(
  for (i in 1:n) {
    if (z[i]<0) {
      zneg[j] <- z[i]
      j <- j + 1
    }
  }
)
```

```
##    user  system elapsed
##   0.022   0.000   0.023
```

Confirm that your rewrite is faster but gives the same result.

```
# my function
system.time(
  new_zneg <- z[which(z<0)]
)
```

```
##    user  system elapsed
##   0.001   0.000   0.001
```

```
# checking the result
all.equal(zneg, new_zneg)
```

```
## [1] TRUE
```

Exercise 3.5

Consider solving the matrix equation $Ax = y$ for x , where y is a known n vector and A is a known $n»n$ matrix. The formal solution to the problem is $x = A^{-1}y$, but it is possible to solve the equation directly, without actually forming A^{-1} . This question explores this direct solution. Read the help file for solve before trying it.

a. First create an A , x and y satisfying $Ax = y$.

```
set.seed(0); n <- 1000
A <- matrix(runif(n*n),n,n); x.true <- runif(n)
y <- A%*%x.true
```

The idea is to experiment with solving $Ax = y$ for x , but with a known truth to compare the answer to.

b. Using solve, form the matrix A^{-1} explicitly and then form $x_1 = A^{-1}y$. Note how long this takes. Also assess the mean absolute difference between `x1` and `x.true` (the approximate mean absolute 'error' in the solution).

```
system.time({
  A_inv <- solve(A)
  x1 <- A_inv %*% y
})
```

```
##    user  system elapsed
##   0.630   0.264   0.260
```

```
mean_diff = mad(x1-x.true); mean_diff
```

```
## [1] 4.042222e-11
```

c. Now use solve to directly solve for x without forming A^{-1} . Note how long this takes and assess the mean absolute error of the result.

```
system.time(
  x1 <- solve(A, y)
)
```

```
##    user  system elapsed
##   0.185   0.064   0.070
```

```
mean_diff = mad(x1-x.true); mean_diff
```

```
## [1] 1.305456e-12
```

d. What do you conclude?

By avoiding matrix multiplication we get a smaller total elapsed time. In both cases my laptop is executing the calculations using its multicore architecture. In the second code, for example, ***user+system*** *time is greater than* ***elapsed*** *time, meaning that there are multiple cores running.*

Exercise 3.6

The empirical cumulative distribution function for a set of measurements $x_i : i = 1, \ldots, n$ is

$$\hat{F}(x) = \frac{\#\{x_i < x\}}{n}$$

where $\#\{x_i < x\}$ denotes 'number of x_i values less than x'. When answering the following, try to ensure that your code is commented, clearly structured, and tested. To test your code, generate random samples using `rnorm` , `runif` , etc.

a. Write an R function that takes an unordered vector of observations `x` and returns the values of the empirical c.d.f. for each value, in the order corresponding to the original `x` vector. See `?sort.int` .

```
my_ecdf <- function(x){
  sorted <- sort.int(x, index.return = TRUE)
  out <- c()
  n <- length(x)
  t <- seq(0,1,1/n)

  for(i in sorted$ix){
    out[i] <- length(which(x<=x[i]))/n
  }
  return(out)
}

my_ecdf(rnorm(50, 0, 1))
```

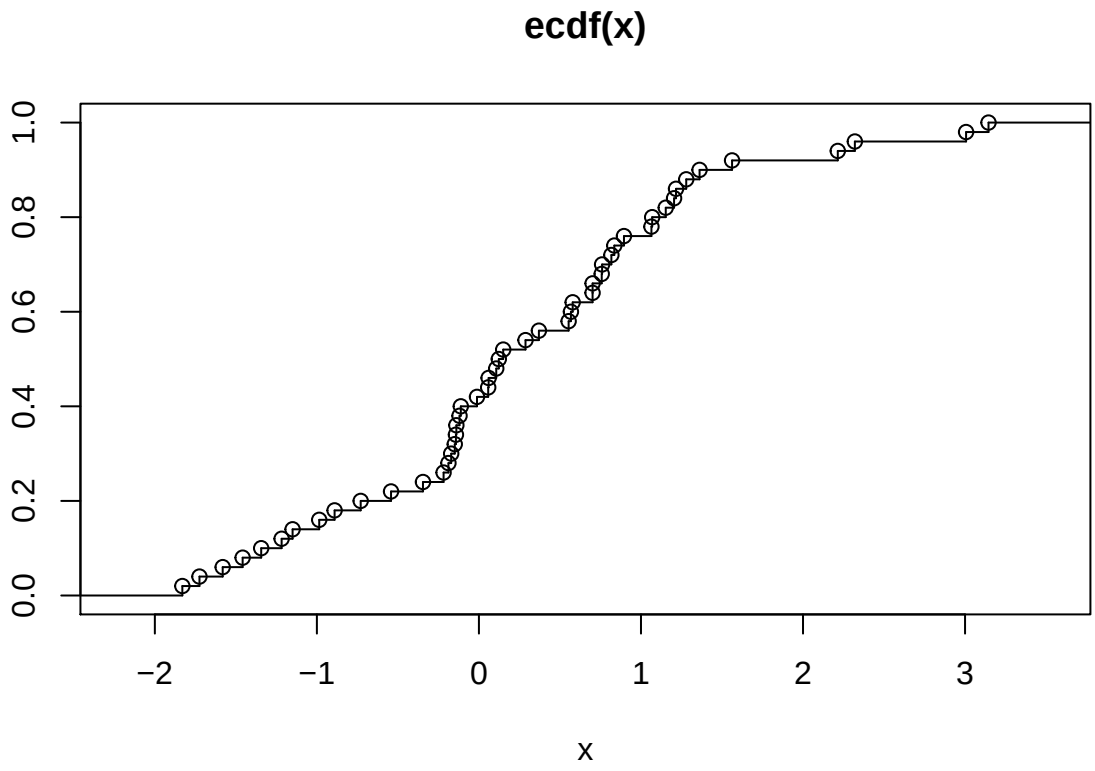
```
##  [1] 0.60 0.34 0.88 0.10 0.96 0.12 0.02 0.80 0.54 0.92 0.68 0.50 0.32 0.72
## [15] 0.08 1.00 0.28 0.36 0.94 0.56 0.74 0.58 0.40 0.38 0.44 0.66 0.82 0.48
## [29] 0.90 0.62 0.78 0.76 0.86 0.46 0.26 0.84 0.14 0.22 0.70 0.06 0.18 0.04
## [43] 0.30 0.20 0.16 0.24 0.52 0.98 0.64 0.42
```

```
my_ecdf(runif(50))
```

```
##  [1] 0.30 0.88 0.26 0.72 0.34 0.40 0.48 0.86 0.12 0.60 0.28 0.56 0.92 0.62
## [15] 0.02 0.18 0.66 0.46 0.44 0.82 0.74 0.70 0.94 0.64 0.84 0.36 0.76 1.00
## [29] 0.80 0.98 0.50 0.38 0.68 0.90 0.10 0.96 0.42 0.14 0.58 0.78 0.06 0.16
## [43] 0.54 0.20 0.52 0.08 0.24 0.32 0.22 0.04
```

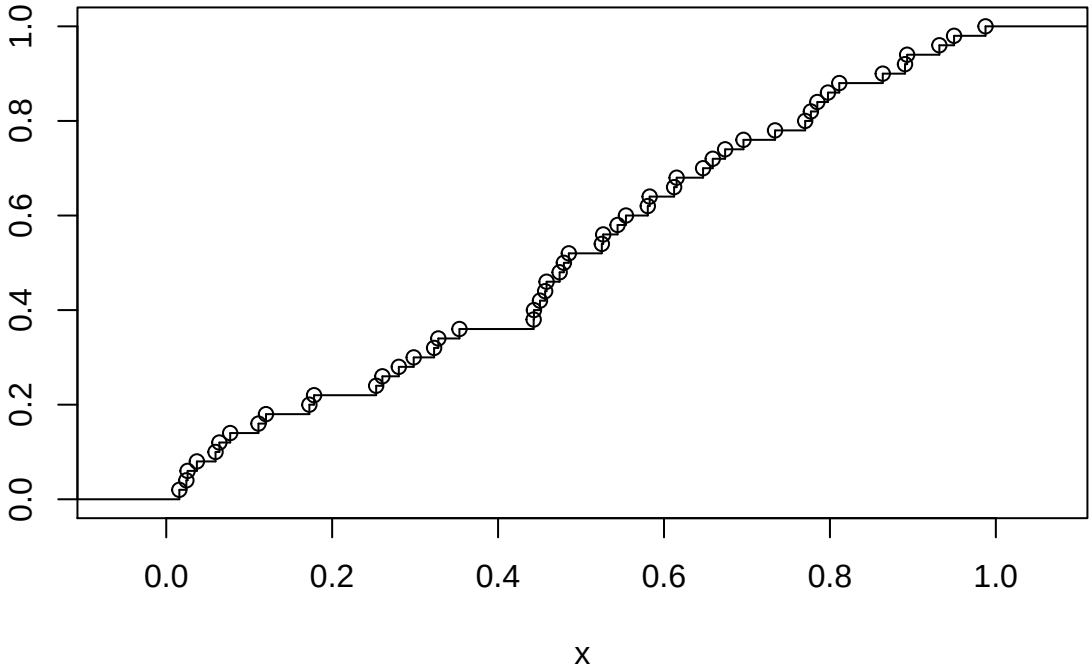
b. Modify your function to take an extra argument plot.cdf , that when TRUE will cause the empirical c.d.f. to be plotted as a step function over a suitable x range.

```
library(graphics)

my_ecdf <- function(x, cond){
  sorted <- sort.int(x, index.return = TRUE)
  out <- c()
  n <- length(x)
  t <- seq(0,1,1/n)
  for(i in sorted$ix){
    out[i] <- length(which(x<=x[i]))/n
  }

  if (cond == TRUE){
    plot.stepfun(x, ylab = " ")
  }
  return(out)
}

my_ecdf(rnorm(50, 0, 1), TRUE)
```




```
##  [1] 0.28 0.36 0.08 0.50 0.20 0.30 0.76 0.72 0.22 0.86 0.82 0.46 0.60 0.40
## [15] 1.00 0.64 0.02 0.96 0.06 0.48 0.68 0.88 0.16 0.44 0.66 0.18 0.04 0.62
## [29] 0.34 0.14 0.74 0.56 0.80 0.98 0.38 0.84 0.54 0.26 0.12 0.42 0.78 0.10
## [43] 0.92 0.24 0.90 0.32 0.52 0.58 0.70 0.94
```

```
my_ecdf(runif(50), TRUE)
```




```
##  [1] 0.08 0.38 0.12 0.20 0.46 0.28 0.06 0.58 0.16 0.82 0.64 0.84 0.18 0.50
## [15] 0.26 0.04 0.22 0.14 0.30 0.98 0.42 0.60 0.48 0.72 0.54 1.00 0.40 0.34
## [29] 0.52 0.02 0.76 0.24 0.94 0.70 0.92 0.44 0.10 0.56 0.86 0.66 0.62 0.88
## [43] 0.80 0.68 0.74 0.90 0.78 0.36 0.32 0.96
```