Problem 1

Chapter 8, Exercise 4 (p. 332).

Problem 2

Chapter 8, Exercise 8 (p. 333).

Problem 3

Chapter 8, Exercise 10 (p. 334).

Problem 4

Chapter 9, Exercise 4 (p. 369).

Problem 5

Chapter 9, Exercise 7 (p. 371). You may skip part (d).

Problem 6

The package `kernlab` in R implements Support Vector Machines for a wide variety of kernels, which are applicable to non-standard data types. In this example, from Jean-Philippe Vert, we will compare various string kernels.

We start by installing the packages needed. Download the source file for the `stringkernels` package from http://cran.r-project.org/src/contrib/Archive/stringkernels/stringkernels_0.8.9.tar.gz. Then run the following commands:

```
install.packages('kernlab')
library(kernlab)
install.packages("~/Downloads/stringkernels_0.8.9.tar.gz", repos = NULL, type='source')
library(stringkernels)
```

You will have to replace '~/Downloads/' above by the path to the directory where you saved the source file stringkernels_0.8.9.tar.gz. **If you have trouble installing the package, see the instructions at the end of the problem.**

Now, we can load our data:

```
data(reuters)
y <- rlabels
x <- reuters
```

The variable `x` is a list of articles from the Reuters agency. The variable `y` is a response, which indicates the topic of the article, 'acquisitions' or 'crude oil'. Your goal is to predict the topic using the information in the articles.

We will work with two kernels. The *spectrum kernel* is equivalent to using a very long vector of features, which count for every possible string of a given length the number of times that the string appears in the article. The computation of this kernel is very efficient. We first define the kernel by calling

```
sk <- stringdot(type="spectrum", length=2, normalized=TRUE)
```

where we specify the length of the words. The *gappy kernel* is able to detect the similarity between strings which have almost matching subsequences. This kernel is defined by calling

```
sgk <- gapweightkernel(length=2,lambda=0.1,normalized=TRUE,use_characters=TRUE)
```

The names `sgk` and `sk` are functions which compute the kernel. We can compute the kernel between two words by calling, for example:

```
sk('abracadabra', 'radar')
```

To run an SVM with the kernel `sk`, we call

```
svp <- ksvm(x,y,kernel=sk,scale=c(),cross=5)
```

The argument `cross=5` allows us to do 5-fold cross validation. The test error estimate is obtained by

```
cross(svp)
```

Your job is to fit the model with the two kernels, using a range of lengths for each one. Then, plot the cross validation errors as a function of the length and select the optimal kernel. Report the best error rate you obtain.

Problems installing stringkernels?

You can download pre-computed kernel matrices built with the gappy kernel with a range of lengths from this directory. Then, call the function `ksvm` as follows:

```
ker = read.csv('len2lam0.1.csv')
ker = as.kernelMatrix(as.matrix(ker))
svp = ksvm(x=ker[,-1],y=rlabels,cross=5)
```