# Problem Set 1

September 17, 2010

## 1 Problem 1

1. Write a function in R that given a vector of temperature for the entire growing season calculates the frost free period and also reports the day in the spring after the last frost and the first day in the fall before the first fall frost.

2. Modify the previous function to consider temperatures other than 0°. For example, -2°.

3. Test your function with the data set "cmi2005.txt".

### 1.1 Solution

The important thing about these problems is that there is no single solution. This is what I propose for problem 1.

**Pseudocode**

1. Split the temperature vector in two

2. Take the first half of the year

3. Loop (starting at the end) and check each value to see if it is smaller than zero

4. When a value smaller than zero is found stop and record the date

5. Do a similar process for the first frost in the fall.

**R code**

```
## Read in the data

cmi <- read.table("cmi2005.txt", header = TRUE)

temp <- cmi$DailyTemp.C
doy <- cmi$doy

## Let us calculate the length of the temp vector

l.temp <- length(temp)

## Then we divide by two to split the vector in half

l.temp2 <- l.temp/2

start.loop <- as.integer(l.temp2)

for(i in start.loop:1){

  if(temp[i] < 0)
    break

}

doy.first.frost <- doy[i]

for(i in start.loop:l.temp){

  if(temp[i] < 0)
    break

}

doy.last.frost <- doy[i]

length.growing.season <- doy.last.frost - doy.first.frost
```

```
## Let's put it into a function

lgs <- function(temp){

  l.temp <- length(temp)

  if(l.temp != 8760)
    stop("This function works with 8760 length vectors")

  ## Then we divide by two to split the vector in half

  l.temp2 <- l.temp/2

  start.loop <- as.integer(l.temp2)

  for(i in start.loop:1){

    if(temp[i] < 0)
      break

  }

  doy.first.frost <- doy[i]

  for(i in start.loop:l.temp){

    if(temp[i] < 0)
      break

  }

  doy.last.frost <- doy[i]

  length.growing.season <- doy.last.frost - doy.first.frost

  ans <- list(first.frost = doy.first.frost,
              last.frost = doy.last.frost,
```

```
              length.growing.season = length.growing.season)
  ans

}

## Test the function

lengs <- lgs(temp)
lengs
$first.frost
[1] 124

$last.frost
[1] 301

$length.growing.season
[1] 177


## Modify the function to accept different criteria for low temperature

lgs <- function(temp, cTemp = 0){

  l.temp <- length(temp)

  if(l.temp != 8760)
    stop("This function works with 8760 length vectors")

  ## Then we divide by two to split the vector in half

  l.temp2 <- l.temp/2

  start.loop <- as.integer(l.temp2)

  for(i in start.loop:1){

    if(temp[i] < cTemp)
      break
```

```
  }

  doy.first.frost <- doy[i]

  for(i in start.loop:l.temp){

    if(temp[i] < cTemp)
      break

  }

  doy.last.frost <- doy[i]

  length.growing.season <- doy.last.frost - doy.first.frost

  ans <- list(first.frost = doy.first.frost,
              last.frost = doy.last.frost,
              length.growing.season = length.growing.season)
  ans

}

## Test it

lengs2 <- lgs(temp, cTemp = -2)
$first.frost
[1] 76

$last.frost
[1] 315

$length.growing.season
[1] 239
```

# 2 Problem 2

1. Write an R function that can compute the mean of meteorological variables by day of the year
2. Adapt the previous function to return the sum instead of the mean
3. Test your function with the data set "cmi2005.txt"

Do not use built-in functions for this, but rather use explicit loops.

## 2.1 Solution

**note:** useful built-in functions that do this are `aggregate`, `ave`, `summarize:package Hmisc` and you might want to use the by statement. There are other useful functions in packages created by Hadley Wickham (`reshape`,`plyr`).

**Pseudo-code**

1. Create an empty vector to store results
2. Create a loop that cycles through the days of the year
3. Select a subset of the data for each day of the year
4. Apply the function of interest
5. Store the results in the empty vector

**R code**

```
## Create function that cylcles through days and returns the mean
## temperature

## This function only works if we have a dataset that has 1 through 365 days

avg <- function(x, var.col = 5){

  ## x is the numeric variable to average

  ## group is the grouping numeric variable which will be
  ## used to construct the averages
```

```
  ## Some error checking

  if(!is.data.frame(x)) stop("x should be a dataframe")

  res <- numeric(365)

  for(i in 1:365){

    tmp <- x[x$doy == i, var.col]

    res[i] <- mean(tmp)

  }

  ans <- data.frame(doy = 1:365, variable = res)

  ans

}


## If we want the sum

sumg <- function(x, var.col=5){

  ## x is the numeric variable to average

  ## group is the grouping numeric variable which will be
  ## used to construct the averages

  ## Some error checking

  if(!is.data.frame(x)) stop("x should be a dataframe")

  res <- numeric(365)

  for(i in 1:365){
```

```
    tmp <- x[x$doy == i, var.col]
    res[i] <- sum(tmp)
  }
  ans <- data.frame(doy = 1:365, variable = res)
  ans
}
## Testing with cmi
aveTemp <- avg(cmi)
head(aveTemp)
## Now the sumg function
sumTemp <- sumg(cmi)
head(sumTemp)
```

# 3 Problem 3

1. Write your won version of a function which does cumulative sums.

## 3.1 Solution

```
## Creating a function that does cumulative sums
CumSum <- function(x){
  ## First create an empty vector
```

```
  lenx <- length(x)

  res <- numeric(lenx)

  ## The first element will be the same

  res[1] <- x[1]

  for(i in 2:lenx){

    res[i] <- res[i - 1] + x[i]

  }

  res

}

## Testing the function
## Cummulative precipitation for the cmi

precip <- cmi$precip

cum.precip <- CumSum(precip)

cum.precip2 <- cumsum(precip)

library(lattice)

xyplot(cum.precip + cum.precip2 ~ 1:8760, type="l")
## The graphs show that the function I create and the built-in
## cummulative function are the same
```

# 4 Comments

As I mentioned before this will be graded based on participation alone. Try to solve the problems on your own first. After you give this problems some thought feel free to discuss approaches with classmates. We will discuss the answers in class. This assignment is due 9-17-2010.