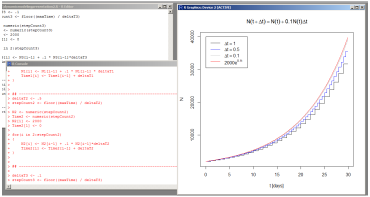
Modeling with R - Solutions to Assignments

Gunther Krauss

Crop Science Research Group

Institute of Crop Science and Resource Conservation (INRES)

February 3, 2021



Pflanzenbau Bonn

Exercise 1

First we calculate the number of new tillers `t` and the total number of tillers `tt` using the script `tillers.R` provided on E-Campus. As we don't know yet, how many stages will be reached, we perform the pre-calculation for 50 stages.

```r
# Maximum numbers of stages
m <- 50
# new tillers
t <- numeric(m)
t[1] <- 1
t[2] <- 1
for(n in 3:m)
{
  t[n] <- t[n-1] + t[n-2]
}
# total number of tillers
tt <- numeric(m)
for(n in 1:m)
{
  tt[n] <- sum(t[1:n])
}
```

E1: Phyllochrons

We put all the phyllochrons for different species into one vector `phyllochrons`.

```
phyllochrons <- c(60,70,80,90,100,110,120,130,140,150)
```

The tillering period is `100 days`

```
time <- 100
```

E1: Question A 1-2

Daily mean temperature `Tdmean` is `10 °C`:

```
TdMean <- 10
```

1. Temperature sum accumulated during 100 days

```
Tsum <- TdMean * time
```

```
# [1] 1000
```

2. Number of completed stages for each phylochron

```
stages <- floor(Tsum/phyllochrons)
```

```
#  [1] 16 14 12 11 10  9  8  7  7  6
```

Each value in `stages` corresponds to the value of the `phyllochrons` vector.

3. Total number of tillers for the calculated stages

```
tillers <- tt[stages]
```

```
#   [1] 2583  986  376  232  143   88   54   33   33   20
```

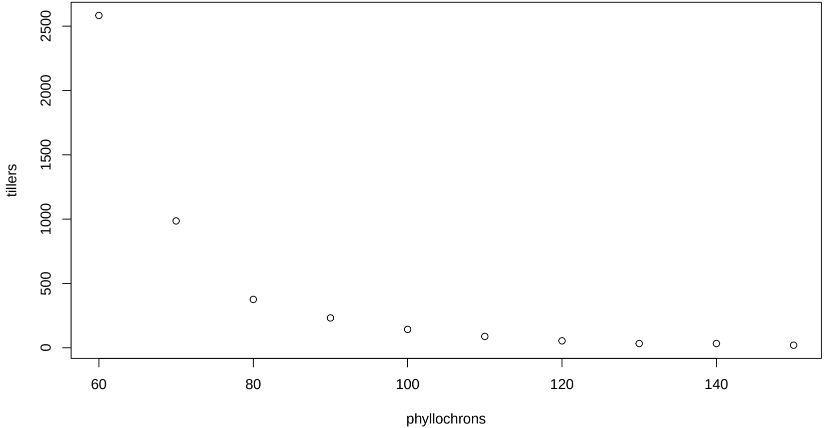
Each value in `tillers` corresponds to the value of the `stages` and therefore to the `phyllochrons`.

NB: If we plan to use the results for further calculations / visualisation, we would put all three vectors in a dataframe

```
tiller_df <- data.frame(phyllochrons, stages, tillers)
```

4. Plot

```
plot(phyllochrons, tillers)
```



E1: Question A 5

5. Relationship between phyllochrons and tillers

From the graph we see, that a higher phyllochron leads to less tillers.
Relationship looks not linear. [It seems to be rather exponential or hyperbolic.]

E1: Question B 1

1. Reading weather table

```r
weather <- read.table('data/weather.txt', header=TRUE)
```

```r
head(weather)
```

```
#   DAY       Date Radiation MinTemp MaxTemp MeanTemp
# 1   1 01.01.1996    1.9968     0.3     2.6     1.45
# 2   2 02.01.1996    2.1911    -0.9     2.1     0.60
# 3   3 03.01.1996    0.7564    -0.5     0.9     0.20
# 4   4 04.01.1996    4.3056    -0.9     1.1     0.10
# 5   5 05.01.1996    3.8569     0.0     3.6     1.80
# 6   6 06.01.1996    1.3625    -5.5    -1.6    -3.55
```

E1: Question B 2

2. Temperature sum from day 61 to 160

```r
Tsum <- sum(weather$MeanTemp[61:160])
```

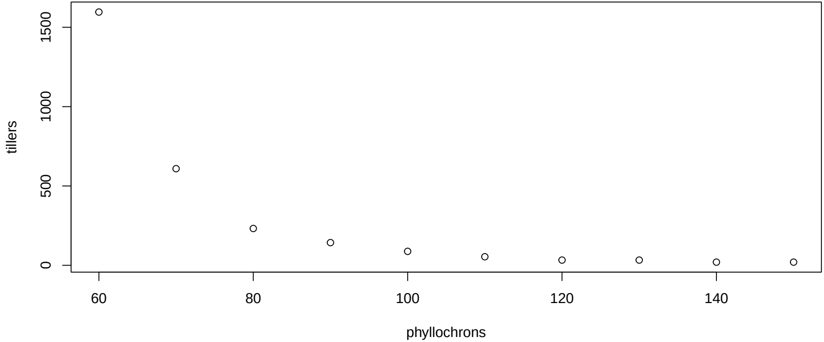
```r
Tsum
```

```
# [1] 934.5
```

E1: Question B 3-5

The rest of the calculations is now identical to part A:

```
stages <- floor(Tsum/phyllochrons)
tillers <- tt[stages]
plot(phyllochrons, tillers)
```



Exercise 2 - Task A

Reading weather

```
weather <- read.table('data/weather.txt', header=TRUE)
```

To make life easier (less typing, better readability), we store the temperature and radiation columns in extra variables

```
MT <- weather$MeanTemp
R <- weather$Radiation
```

E2: Task A - preparation

End time is day 365

```
EndTime <- 365
```

We want to calculate the weight for each day, so we prepare vector of zeros, that we will fill later with our calculated values:

```
deltaW <- numeric(EndTime)
W <- numeric(EndTime)
```

E2: Task A - initial value

We have to store the initial weight on day 1 in our weights variable:

```
W[1] <- 1.53
```

E2: Task A - looping

We want to calculate values from day 2 to `EndTime` 365, so our loop has to run through `2:EndTime`.

```
for(n in 2:EndTime)
{
 ...
}
```

The variable `n` denotes the day!

E2: Task A - rate calculation

To calculate the state of day `n`, we first need to calculate the rate of the previous day `n-1`.

```
deltaW[n-1] = ...
```

(In the assignment, the general formula was given as $\Delta W_n = ...$ As we need to calculate for `n-1`, we substitute each `n` with `n-1` in the formula.)

To get mean Temperature or Radiation for a specific day, we just subset the coresponding vector by the day, e.g. for day `n-1` Temperature is `MT[n-1]` and radiation is `R[n-1]`.

E2: Task A - condition

`deltaW` is calculated with first formula on the condition that the day is smaller or equal or 21, and with second formula on the condition that the day is greater than 21 So the if statements looks like:

```
if(n-1 <= 21)
{
  ... # formula 1
} else {
  ... # formula 2
}
```

Remember that `n` denotes the day!

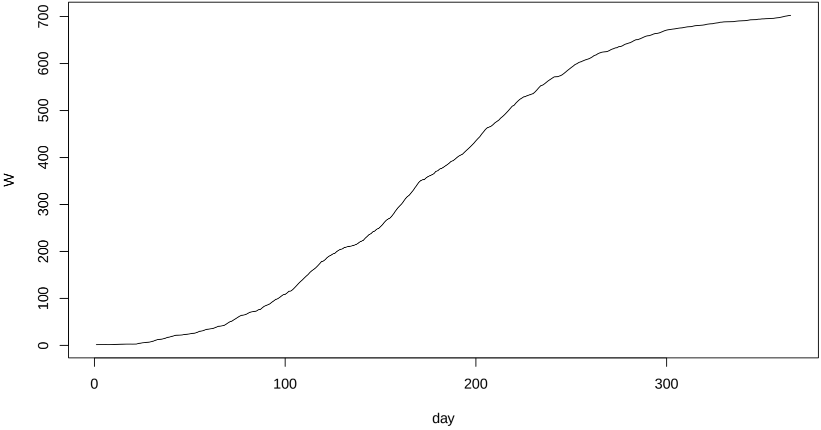
E2: Task A - putting everything together

```
for(n in 2:EndTime)
{
  if(n-1 <= 21)
  {
    deltaW[n-1] <- 0.0193 * MT[n-1] * W[n-1]
  } else {
    deltaW[n-1] <- 0.201 * R[n-1]
  }
  W[n] <- W[n-1] + deltaW[n-1]
}
W[EndTime]
```

```
# [1] 702.4693
```

E2: Task A - plotting

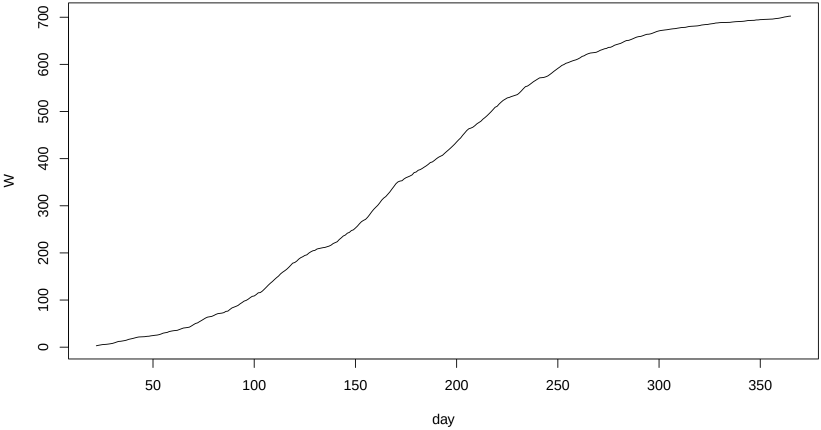
```
plot(W,type="l",xlab="day")
```



E2: Task A - plotting only phase 2

Second phase is from day 22 to EndTime

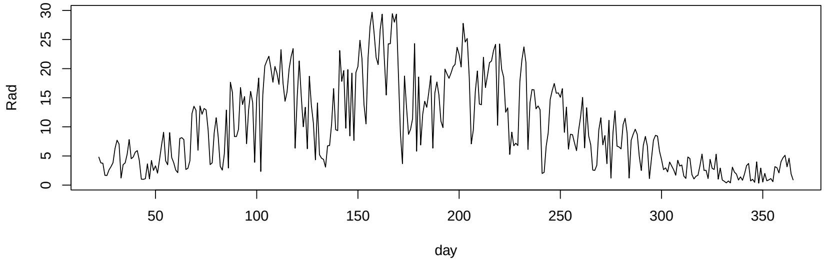
```
plot(22:EndTime, W[22:EndTime],type="l",xlab="day",ylab="W")
```



E2: Task B

The curve is not linear. From the formula we see, that the increment $\Delta W_n = 0.201 R_n$ depends on radiation. Plotting the radiation

```
plot(22:EndTime,R[22:EndTime], type="l", xlab="day",ylab="Rad")
```



shows that radiation is not constant, therefore the rate is not constant and the state not linear.