Fiche TD avec le logiciel R : microdendroCHS57

Manipulation de données temporelles appliquée au suivi horaire de la croissance de dix chênes pendant un an

Pr Jean R. Lobry

Table des matières

1 Des données temporelles utilisant le format POSIXct

1.1 La classe d'horodatage POSIXt

Dans cette fiche on cherche à illustrer toute la puissance qui est à notre disposition dans R quand les données temporelles sont au format POSIXt. C'est un format qui est adapté si votre résolution temporelle est infra-journalière, par exemple avec des heures, minutes et secondes. Si vous avez des données simplement calendaires, avec une résolution journalière, il vaut mieux travailler avec des objets de la classe Date[1]. Les données temporelles manipulées ici sont *déjà* dans le format POSIXt, mais on montrera en annexe page 39 comment les obtenir au moment de l'importation. La classe POSIXt a été introduite dans la version 1.2.0 de R en décembre 2000 [9, 5].

1.2 Les avatars POSIXct et POSIXlt

La classe POSIXt est en fait une classe abstraite qui peut s'incarner dans la classe POSIXct ou bien dans la classe POSIXlt. Voyons un exemple de chaque.

```
exemplePOSIXct <- as.POSIXct("2022-07-01 14:15:16")
class(exemplePOSIXct)
```

```
[1] "POSIXct" "POSIXt"
```

```
exemplePOSIXct
```

```
[1] "2022-07-01 14:15:16 CEST"
```

```
exemplePOSIXlt <- as.POSIXlt("2022-07-01 15:16:17")
class(exemplePOSIXlt)
```

```
[1] "POSIXlt" "POSIXt"
```

```
exemplePOSIXlt
```

```
[1] "2022-07-01 15:16:17 CEST"
```

On voit que les deux exemples dérivent tous les deux de la classe POSIXt, mais quand on demande de les afficher on ne voit pas bien de différence (le CEST fait référence au fuseau horaire utilisé). Ils diffèrent dans leur façon de représenter en interne les données temporelles.

```
dput(exemplePOSIXct)
```

```
structure(1656677716, class = c("POSIXct", "POSIXt"), tzone = "")
```

```
dput(exemplePOSIXlt)
```

```
structure(list(sec = 17, min = 16L, hour = 15L, mday = 1L, mon = 6L,
    year = 122L, wday = 5L, yday = 181L, isdst = 1L, zone = "CEST",
    gmtoff = NA_integer_), class = c("POSIXlt", "POSIXt"))
```

Les objets de la classe POSIXct sont représentés simplement par le nombre de secondes écoulées depuis le 1er janvier 1970 à minuit tandis que les objets de la classe POSIXlt sont représentés par une longue liste d'éléments tels que year pour le nombre d'années écoulées depuis 1900. La conséquence directe est que les objets de la classe POSIXlt sont beaucoup plus gourmands en espace mémoire que ceux de la classe POSIXct :

```
object.size(exemplePOSIXct)
```

[1] Voir https://pbil.univ-lyon1.fr/R/pdf/dendroCHS57.pdf

```
568 bytes
 object.size(exemplePOSIXlt)
2040 bytes
```

Il n'y a donc pas beaucoup d'intérêt à travailler avec des objets de la classe `POSIXlt`, et la documentation de R recommande d'utiliser la classe `POSIXct` dans les tableaux de données. La classe abstraite `POSIXt` permet de faire des opérations mélangeant des objets des deux types de classe, par exemple pour calculer une différence :

```
 exemplePOSIXlt - exemplePOSIXct
Time difference of 1.016944 hours
```

L'argument selon lequel la classe `POSIXlt` pourrait être préférable parce que son format de représentation interne est plus lisible par un être humain ne tient pas parce que la fonction `format()` permet d'obtenir à peu près tout ce que l'on veut (voir la table 1 page 6). On peut aussi utiliser les fonctions du paquet `lubridate` [4] pour un code plus lisible. Par exemple, pour retrouver les éléments de la classe `POSIXlt` à partir d'un objet de la classe `POSIXct` :

```
 library(lubridate)
 second(exemplePOSIXct) # sec
[1] 16
 minute(exemplePOSIXct) # min
[1] 15
 hour(exemplePOSIXct)   # hour
[1] 14
 mday(exemplePOSIXct)   # mday
[1] 1
 month(exemplePOSIXct) - 1    # mon
[1] 6
 year(exemplePOSIXct) - 1900  # year
[1] 122
 wday(exemplePOSIXct, week_start = 7) # wday
[1] 6
 yday(exemplePOSIXct) - 1  # yday
[1] 181
 ifelse(dst(exemplePOSIXct), 1, 0) # isdst
[1] 1
 format(exemplePOSIXct, "%Z") # zone
[1] "CEST"
 sHHMM <- format(exemplePOSIXct, "%z") # gmtoff
 gmtoff <- 3600*as.integer(substr(sHHMM, 2, 3)) + 60*as.integer(substr(sHHMM, 4, 5))
 if(substr(sHHMM, 1, 1) == "-") gmtoff <- -1L*gmtoff
 gmtoff
[1] 7200
```

Les objets de la classe `POSIXlt` sont particulièrement toxiques quand on utilise la fonction `range()` pour calculer les limites pour une représentation graphique (typiquement `plot.window(xlim = range(x))`). Si on regarde le résultat renvoyé par cette fonction, on ne voit pas bien de différence entre les objets des deux classes :

```
 testrange <- c("1970-01-01 12:00:00", "2022-01-01 12:00:00")
 range(as.POSIXct(testrange))
```

```
[1] "1970-01-01 12:00:00 CET" "2022-01-01 12:00:00 CET"
```

```
range(as.POSIXlt(testrange))
```

```
[1] "1970-01-01 12:00:00 CET" "2022-01-01 12:00:00 CET"
```

Mais si on regarde de plus près comment les résultats sont représentés en interne, on voit que ce n'est que pour les objets de la classe `POSIXct` que l'on a un vecteur de deux éléments directement exploitable :

```
dput(range(as.POSIXct(testrange)))
```

```
structure(c(39600, 1641034800), class = c("POSIXct", "POSIXt"
))
```

```
dput(range(as.POSIXlt(testrange)))
```

```
structure(list(sec = c(0, 0), min = c(0L, 0L), hour = c(12L,
12L), mday = c(1L, 1L), mon = c(0L, 0L), year = c(70L, 122L),
    wday = c(4L, 6L), yday = c(0L, 0L), isdst = c(0L, 0L), zone = c("CET",
    "CET"), gmtoff = c(3600L, 3600L)), class = c("POSIXlt", "POSIXt"
), tzone = c("", "CET", "CEST"))
```

On évitera donc dans le mesure du possible d'utiliser des objets de la classe `POSIXlt`. Attention, si vous utilisez directement la fonction `strptime()` (*string parser time*) pour convertir des chaînes de caractères, il faut savoir qu'elle renvoie des objets de la classe `POSIXlt`. En revanche, les fonctions `ISOdate()` et `ISOdatetime()`, pour convertir une représentation numérique des données temporelles, renvoient des objets de la classe `POSIXct`.

1.3 Les secondes intercalaires

Les secondes intercalaires sont ajoutées épisodiquement au temps universel coordonné (UTC) pour que le décalage avec le temps atomique international (TAI) soit inférieur à une seconde (à cause du ralentissement de la rotation de la Terre il y a une lente dérive de UTC par rapport à TAI). La norme POSIX spécifie qu'elles doivent être *ignorées* et ceci est vérifié au moment de la compilation de R ce qui garantit la reproductibilité des résultats quel que soit le système d'exploitation utilisé. La classe `POSIXct` n'est donc pas adaptée si vous avez besoin d'une précision astronomique, au sens propre du terme. Il est facile de vérifier que tous les jours depuis le 1er janvier 1970 ont duré exactement 86400 secondes selon la norme POSIX :

```
jours <- seq(as.POSIXct("1970-01-01 00:00:00"), Sys.time(), by = "day")
all(diff(as.numeric(jours)) == 86400L)
```

```
[1] TRUE
```

C'est évidement faux pour UTC puisque, depuis le 1er janvier 1970, 27 secondes ont été ajoutées pour compenser la dérive par rapport à TAI. Si besoin, les dates auxquelles les secondes intercalaires ont été ajoutées sont disponibles dans l'objet `.leap.seconds` :

```
.leap.seconds
```

```
 [1] "1972-07-01 GMT" "1973-01-01 GMT" "1974-01-01 GMT" "1975-01-01 GMT"
 [5] "1976-01-01 GMT" "1977-01-01 GMT" "1978-01-01 GMT" "1979-01-01 GMT"
 [9] "1980-01-01 GMT" "1981-07-01 GMT" "1982-07-01 GMT" "1983-07-01 GMT"
[13] "1985-07-01 GMT" "1988-01-01 GMT" "1990-01-01 GMT" "1991-01-01 GMT"
[17] "1992-07-01 GMT" "1993-07-01 GMT" "1994-07-01 GMT" "1996-01-01 GMT"
[21] "1997-07-01 GMT" "1999-01-01 GMT" "2006-01-01 GMT" "2009-01-01 GMT"
[25] "2012-07-01 GMT" "2015-07-01 GMT" "2017-01-01 GMT"
```

Ceci signifie par exemple, pour la seconde intercalaire la plus récente, que le `2016-12-31 23:59:59` a été suivi exceptionnellement par `2016-12-31 23:59:60` avant de passer à `2017-01-01 00:00:00`. Ce qu'il faut bien comprendre ici c'est qu'il n'y a pas de *bijection* entre les secondes informatiques POSIX et les secondes légales UTC. On a en fait une application *injective* des secondes POSIX vers les secondes UTC : les secondes UTC possèdent au plus un antécédent dans les secondes POSIX, ce qui revient à dire que deux secondes POSIX distinctes n'ont pas la même image dans les secondes UTC. Mais reste qu'il y a des secondes légales UTC sans antécédent dans les secondes POSIX, les fameuses secondes intercalaires.

1.4 Le paquet `lubridate`

Le paquet `lubridate` [4] définit de nombreuses fonctions qui simplifient la manipulation des données temporelles. La fonction `with_tz()` permet de changer très facilement de fuseau horaire.

```
exemplePOSIXct
```
```
[1] "2022-07-01 14:15:16 CEST"
```
```
with_tz(exemplePOSIXct, tz = "Pacific/Auckland")
```
```
[1] "2022-07-02 00:15:16 NZST"
```

Le premier juillet 2022 à 14 heures et des brouettes il sera minuit passé (donc le lendemain) à Auckland en Nouvelle-Zélande. La liste des noms des fuseaux horaires disponibles (plusieurs centaines) est donné par la fonction `OlsonNames()`.

```
sample(OlsonNames(), 15)
```
```
 [1] "Asia/Thimphu"         "US/Hawaii"            "America/Creston"
 [4] "Asia/Barnaul"         "Asia/Ho_Chi_Minh"     "America/Los_Angeles"
 [7] "Africa/Juba"          "Asia/Choibalsan"      "Africa/Libreville"
[10] "MST7MDT"              "America/El_Salvador"  "Asia/Qatar"
[13] "Europe/Uzhgorod"      "America/Knox_IN"      "Etc/GMT+11"
```

Dans le paquet `lubridate` sont définies plusieurs fonctions au nom intuitif : `second()`, `minute()`, `hour()`, `day()`, `wday()`, `yday()`, `week()`, `month()`, `year()` et `tz()`. Elles permettent de faire les opérations les plus courantes de la fonction `format()`, voir la table 1 page 6, mais de façon plus compacte et plus lisible. Ceci permet d'avoir un code plus facile à maintenir, par exemple :

```
as.numeric(format(exemplePOSIXct, "%S"))
```
```
[1] 16
```
```
second(exemplePOSIXct)
```
```
[1] 16
```

La table 2 page 2 illustre le comportement de ces fonctions. Elles sont également utilisables pour *modifier* la valeur d'un élément d'une variable de la classe `POSIXct`, par exemple pour mettre les secondes à zéro :

```
exemple2POSIXct <- exemplePOSIXct
second(exemple2POSIXct) <- 0
exemple2POSIXct
```
```
[1] "2022-07-01 14:15:00 CEST"
```

Toutes ces fonctions sont vectorisées, donc pas besoin de faire une boucle de façon explicite pour traiter tous les éléments d'un vecteur.

Format	Signification	Exemple 2022-07-01 14:15:16
%a	Nom abrégé du jour de la semaine dans le système d'exploitation local	Ven
%A	Nom complet du jour de la semaine dans le système d'exploitation local	Vendredi
%b	Nom abrégé du mois dans le système d'exploitation local	jul
%B	Nom complet du mois dans le système d'exploitation local	juillet
%c	Date et heure dans le format standard du système d'exploitation local	Ven 1 jul 14:15:16 2022
%C	« Siècle » : la partie entière de l'année divisée par 100	20
%d	Le jour du mois compris entre 01 et 31	01
%D	La date dans un format du type %m%d%y : pour le standard C99 c'est le format exact, mais il n'est pas honoré par tous les systèmes d'exploitation	07/01/22
%e	Le jour du mois compris entre 1 et 31, avec une espace en amont quand il n'y a qu'un seul chiffre	→ 1←
%F	Équivalent à %Y-%m-%d, c'est à dire le format ISO 8601 pour les dates	2022-07-01
%g	Les deux derniers chiffres de l'année basée sur les semaines (voir %V)	22
%h	Équivalent à %b	jul
%H	Les heures (00-23)	14
%I	Les heures (01-12)	02
%j	Rang du jour dans l'année compris entre 001 et 366	182
%m	Rang du mois dans l'année compris entre 01 et 12	07
%M	Les minutes (00-59)	15
%p	Indicateur AM/PM	PM
%r	L'heure sur 12 heures	02:15:16 PM
%R	Équivalent à %H:%M	14:15
%S	Secondes (00-61)	16
%T	Équivalent à %H:%M:%S	14:15:16
%u	Rang du jour dans la semaine compris entre 1 et 7, lundi vaut 1	5
%U	Rang de la semaine dans l'année compris entre 00 et 53 en utilisant dimanche comme le jour 1 de la semaine (et typiquement comme le premier dimanche de l'année comme le jour 1 de la semaine 1). Convention locale aux USA.	26
%V	Rang de la semaine dans l'année compris entre 01 et 53 comme défini dans la norme internationale ISO 8601. Si la semaine (commençant le lundi) qui contient le 1er janvier comporte au moins 4 autres jours dans la nouvelle année alors c'est la semaine 1. Sinon, c'est la dernière semaine de l'année précédente et c'est la semaine suivante qui est la semaine 1	26
%w	Rang du jour dans la semaine compris entre 0 et 6, dimanche vaut 0	5
%x	Date dans le format du système d'exploitation local	01.07.2022
%x	Heure dans le format du système d'exploitation local	14:15:16
%y	L'année sans le siècle compris entre 00 et 99	22
%Y	L'année avec le siècle.	2022
%z	Décalage horaire par rapport à l'UTC. En France +2 heures l'été, +1 heure l'hiver.	+0200
%Z	Nom abrégé du fuseau horaire. Par exemple CEST pour *Central European Summer Time* et CET pour *Central European Time*.	CEST

Table 1 : Liste des formats de conversions utilisables avec la fonction `format()` pour les objets de la classe `POSIXct`. Ils sont documentés dans l'aide de la fonction `strptime()`. La fonction `format()` renvoie une chaîne de caractères, pour exploiter des valeurs numériques il faudra les transtyper avec `as.numeric()` ou `as.integer()`.

Nom	Signification	Exemple 2022-07-01 14:15:16
`second()`	Les secondes (`%S`)	16
`minute()`	Les minutes (`%M`)	15
`hour()`	Les heures (`%H`)	14
`wday()`	Le rang du jour dans la semaine (1-7) avec dimanche valant 1.	6
`wday( , week_start = 1)`	Le rang du jour dans la semaine (1-7) avec lundi valant 1.	5
`wday( , label = TRUE)`	Le nom abrégé du jour de la semaine dans le système d'exploitation local.	Ven
`wday( , label = TRUE, abbr = FALSE)`	Le nom du jour de la semaine dans le système d'exploitation local.	Vendredi
`mday()` ou `day()`	Le rand du jour dans le mois (`%d`)	1
`yday()`	Le rang du jour dans l'année (`%j`)	182
`week()`	Le rang de la semaine dans l'année : le nombre de semaines complètes depuis le 1er janvier plus un.	26
`isoweek()`	Le rang de la semaine dans l'année selon la norme internationale ISO 8601 (`%V`)	26
`month()`	Le rang du mois dans l'année (`%m`).	7
`month( , label = TRUE)`	Le nom abrégé du mois dans le système d'exploitation local (`%b`).	jul
`month( , label = TRUE, abbr = FALSE)`	Le nom du mois dans le système d'exploitation local (`%B`).	juillet
`year()`	L'année (`%Y`)	2022
`tz()`	Le fuseau horaire. Une chaîne de caractères vide signifie le fuseau horaire du système d'exploitation local.	→ ←

Table 2 : Quelques fonctions utilitaires du paquet `lubridate`.

```
vecteur <- rep(exemple2POSIXct, 5)
vecteur
```

```
[1] "2022-07-01 14:15:00 CEST" "2022-07-01 14:15:00 CEST" "2022-07-01 14:15:00 CEST"
[4] "2022-07-01 14:15:00 CEST" "2022-07-01 14:15:00 CEST"
```

```
minute(vecteur) <- 59
vecteur
```

```
[1] "2022-07-01 14:59:00 CEST" "2022-07-01 14:59:00 CEST" "2022-07-01 14:59:00 CEST"
[4] "2022-07-01 14:59:00 CEST" "2022-07-01 14:59:00 CEST"
```

1.5 Les données illustratives

Les données, aimablement partagées par Nicolas Delpierre, ont été collectées avec des dendromètres électroniques (cf. figure 1 page 8) en 2021 sur le site CHS57A (donc majoritairement des chênes sessiles *Quercus petraea*) de l'unité territoriale du Saulnois (57590) dans le département de la Moselle en France. Les 10 chênes suivis avaient un diamètre du tronc compris entre 47 et 61 cm. Le modèle de dendromètre utilisé ici ne permet de faire que des mesures relatives : on mesure les variations de la circonférence du tronc mais on ne sait pas dans l'absolu quelle est sa valeur.

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/microdendroCHS57/microd.Rda"))
head(microd[ , c(1:4, 11)])
```

Figure 1 : Un dendromètre est une bande métallique (un feuillard) de faible coefficient de dilatation thermique posé à hauteur d'homme contre la circonférence du tronc d'un arbre (photo en en haut à gauche). Le feuillard est plaqué contre tronc à l'aide d'un dispositif de rappel (par exemple le ressort en haut à droite). La lecture de la valeur du diamètre du tronc peut se faire manuellement à l'aide d'un vernier (photo en bas à gauche) ou bien automatiquement avec un dispositif électronique d'enregistrement (photo en bas à droite). Crédit photo : Marie-Claude Venner.

```
  CHS57A_301 CHS57A_303 CHS57A_112 CHS57A_113              temps
1   14.10190   15.47888   12.23785   14.69743 2021-01-01 00:00:00
2   14.10190   15.47888   12.23785   14.69549 2021-01-01 01:00:00
3   14.10190   15.47694   12.23785   14.69743 2021-01-01 02:00:00
4   14.10384   15.47888   12.23979   14.69743 2021-01-01 03:00:00
5   14.10190   15.48081   12.23979   14.69743 2021-01-01 04:00:00
6   14.10287   15.47888   12.23979   14.69646 2021-01-01 05:00:00
```

```
tail(microd[ , c(1:4, 11)])
```

```
     CHS57A_301 CHS57A_303 CHS57A_112 CHS57A_113              temps
8195   27.52112   32.82955   29.03173   31.43127 2021-12-08 10:00:00
8196   27.52306   32.83052   29.03173   31.43127 2021-12-08 11:00:00
8197   27.52209   32.82858   29.03270   31.43127 2021-12-08 12:00:00
8198   27.52112   32.82955   29.03270   31.42933 2021-12-08 13:00:00
8199   27.52112   32.82858   29.03270   31.43127 2021-12-08 14:00:00
8200   27.52112   32.82858   29.03270   31.40706 2021-12-08 15:00:00
```

Les unités sont des millimètres, avec cinq chiffres après la virgule. Si on regarde la valeur absolue des écarts consécutifs et que l'on cherche les plus petites valeurs non nulles, on voit que l'on a une précision de la mesure qui est de l'ordre du μm, d'où le nom de micro-dendromètre donné à ce type d'instrument de mesure :

```
apply(microd[, 1:10], 2, \(x) min(abs(diff(x)[abs(diff(x)) > 0])))
```

```
CHS57A_301 CHS57A_303 CHS57A_112 CHS57A_113 CHS57A_302 CHS57A_115 CHS57A_107
   0.00096    0.00096    0.00096    0.00096    0.00096    0.00096    0.00096
CHS57A_106 CHS57A_105 CHS57A_101
   0.00096    0.00096    0.00096
```

Cette précision au micron peut sembler étonnante, mais cela n'a en fait rien d'extraordinaire puisque, dès la fin du XIX^{e} siècle, le dendromètre de Josef Friedrich (voir la figure 2 page 10) avait une précision de 10 μm permettant de mettre en évidence les fluctuations journalières du périmètre du tronc des arbres (voir la figure 3 page 11).

Les données temporelles sont de la classe `POSIXct`, d'une résolution d'une heure, dans le fuseau horaire UTC+02, et couvrent l'année 2021 jusqu'au 8 décembre 15h00. Convertissons un instant de l'hiver et un instant de l'été dans l'heure légale utilisée en France métropolitaine :

```
class(microd$temps)
```

```
[1] "POSIXct" "POSIXt"
```

```
range(microd$temps)
```

```
[1] "2021-01-01 00:00:00 +02" "2021-12-08 15:00:00 +02"
```

```
(hiver <- microd[8197, 11])
```

```
[1] "2021-12-08 12:00:00 +02"
```

```
with_tz(hiver, tz = "Europe/Paris")
```

```
[1] "2021-12-08 11:00:00 CET"
```

```
(été <- microd[4405, 11])
```

```
[1] "2021-07-03 12:00:00 +02"
```

```
with_tz(été, tz = "Europe/Paris")
```

```
[1] "2021-07-03 12:00:00 CEST"
```

On voit que UTC+02 correspond à l'heure d'été en France métropolitaine, comme c'est également la période de croissance des chênes, nous n'aurons pas trop de gymnastique intellectuelle à faire pour interpréter les données horaires.

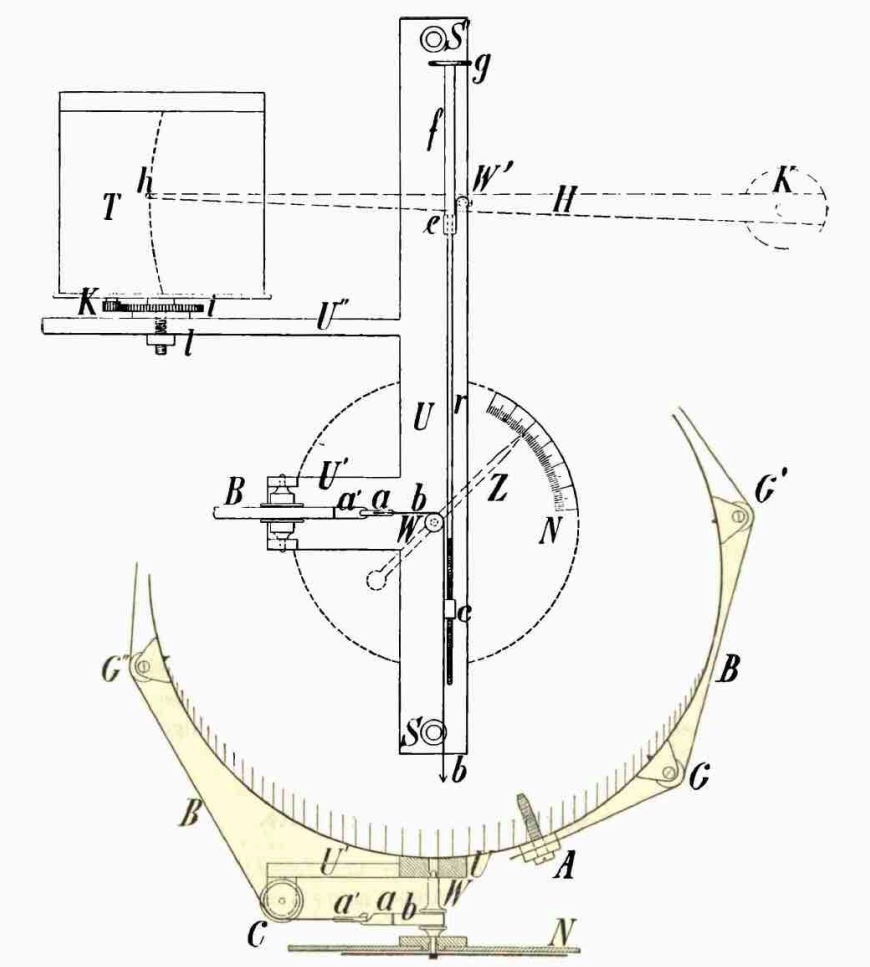


Figure 2 : Schéma du dendromètre de Josef Friedrich. Dans cette représentation il y a deux plans de coupe : un en bas, coloré en jaune, et un en haut sans coloration. Le plan de coupe du bas est le plan de coupe du bûcheron, au sens propre du terme, c'est à dire orthogonal à l'axe du tronc de l'arbre. L'extrémité d'un feuillard est vissée en **A** sur le tronc puis en fait le tour via des poulies **C**, **C'**, **C"** pour rejoindre le bloc fixé en **U** sur le tronc. Dans ce bloc les mouvements du feuillard qui se font dans le plan horizontal vont être transmis dans le plan vertical à autre feuillard via la jonction **a'-a** et la poulie **b**. Le plan de coupe du haut est dans le sens vertical, c'est ce que voit l'observateur. Il faut imaginer qu'au niveau de la flèche **b** une masse de 1 à 2 kg est attachée pour maintenir le dispositif sous tension. Une aiguille **Z** est attachée au niveau de la poulie **W** permettant de faire une lecture directe au centième de millimètre, soit 10 μm, sur le cadran **N**. Le feuillard est également solidaire en **c** d'une tige qui déplace un stylet fixé en **K** traçant en **h** le signal sur le tambour **T** entraîné par un dispositif d'horlogerie **K**. La précision temporelle est difficile à estimer pour un dispositif d'enregistrement en continu mais n'a rien à envier à celle des dendromètres modernes. Copie de la figure 1 page 2 de [3].

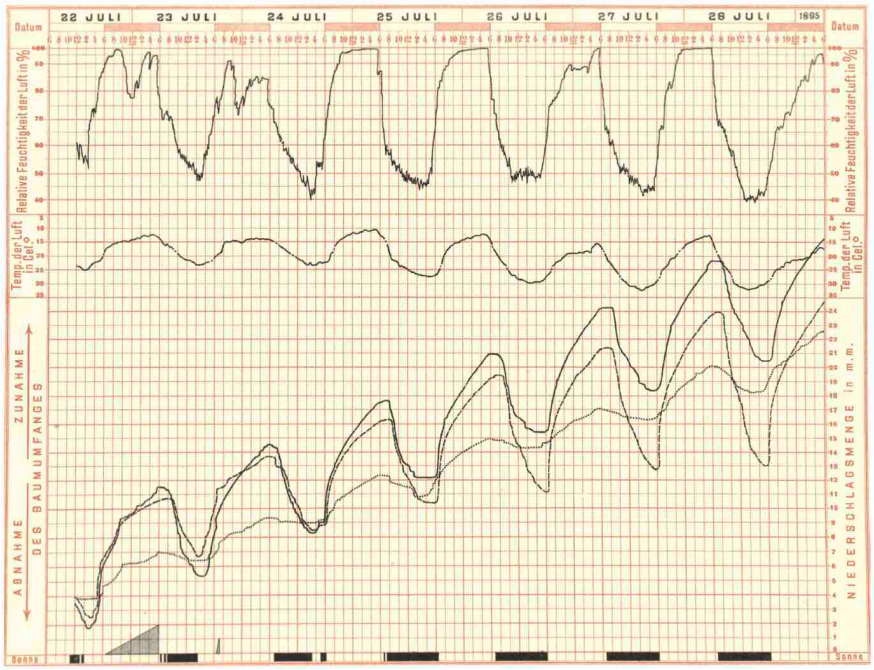


Figure 3 : Exemple de relevé dendrométrique produit pour trois arbres par Josef Friedrich pour la semaine du 22 juillet au 28 juillet 1895. La courbe du haut donne l'humidité relative de l'air, celle en dessous la température. Les triangles en bas donnent le niveau des précipitations et les rectangles du bas le niveau d'ensoleillement. La lente croissance des périmètres des troncs des trois arbres est parfaitement visible sur les courbes du bas. Le rythme circadien, avec une augmentation nocturne et une diminution diurne, est également évident ici. Josef Friedrich produit cet exemple pour illustrer un phénomène assez fin, à savoir que la variabilité de l'humidité de l'air pendant les nuits du 22 au 24 juillet a fait dévier les courbes de croissance de la forme régulière observée quand le temps est plus clément du 25 au 28 juillet. Copie de la planche IX de [3].

2 Représentation de séries temporelles

2.1 Représentation des données brutes

Comme nos données sont relatives, pour mieux pouvoir comparer les arbres nous allons translater verticalement nos séries de façon à ce qu'elles partent toutes de la valeur zéro dans l'objet `microd0`.

```
microd0 <- microd
for(j in seq_len(10)) microd0[ , j] <- microd0[ , j] - microd0[1, j]
```

Les noms des arbres sont donnés dans les noms des colonnes de `microd0` en accolant le nom du site et le numéro de l'arbre. Comme nous ne considérons qu'un seul site, pour gagner de la place dans les légendes des graphiques, nous n'allons conserver que le n° des arbres dans le vecteur `nomsa`.

```
colnames(microd0)
```

```
[1] "CHS57A_301" "CHS57A_303" "CHS57A_112" "CHS57A_113" "CHS57A_302" "CHS57A_115"
[7] "CHS57A_107" "CHS57A_106" "CHS57A_105" "CHS57A_101" "temps"
```

```
nomsa <- substr(colnames(microd0)[1:10], 8, 10)
nomsa
```

```
[1] "301" "303" "112" "113" "302" "115" "107" "106" "105" "101"
```

Nous allons faire une représentation graphique avec deux axes pour les temps. Un axe qui utilise le codage interne des temps, donc le nombre de secondes depuis le temps zéro d'UNIX, dont on se doute qu'il ne sera pas bien parlant pour un être humain. Pour que ce soit plus clair on va utiliser un deuxième axe dont les graduations correspondent au premier jour de chaque mois. Ce n'est pas si simple puisque le nombre de jours dans un mois est variable et qu'il peut, pour le mois de février, varier selon que l'année soit bissextile ou non. Heureusement pour nous, la fonction `seq.POSIXt()` va prendre en charge tous ces détails.

```
debmois <- seq(as.POSIXct("2021-01-01", tz = "Etc/GMT-2"), by = "month", length.out = 12)
debmois
```

```
[1] "2021-01-01 +02" "2021-02-01 +02" "2021-03-01 +02" "2021-04-01 +02"
[5] "2021-05-01 +02" "2021-06-01 +02" "2021-07-01 +02" "2021-08-01 +02"
[9] "2021-09-01 +02" "2021-10-01 +02" "2021-11-01 +02" "2021-12-01 +02"
```

```
dput(debmois)
```

```
structure(c(1609452000, 1612130400, 1614549600, 1617228000, 1619820000,
1622498400, 1625090400, 1627768800, 1630447200, 1633039200, 1635717600,
1638309600), class = c("POSIXct", "POSIXt"), tzone = "Etc/GMT-2")
```

Notre vecteur `debmois` contient donc le nombre de secondes écoulées depuis le temps zéro d'UNIX jusqu'au premier jour de chaque mois de l'année 2021. La fonction `month()` nous permet de récupérer très facilement les noms abrégés des mois. Notez que l'on peut utiliser un incrément du type `by = "2 months"`, et que l'on n'est pas limité aux mois, se reporter à l'aide de la fonction `seq.POSIXt()` pour plus de détails. Le nom en clair des mois, abrégé ou non, est donné par la fonction `month()` :

```
Sys.setlocale("LC_ALL", "C") # Si besoin en anglais
```

```
[1] "C/C/C/C/C/fr_FR.UTF-8"
```

```
month(debmois, label = TRUE)
```

```
 [1] Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < Oct < ... < Dec
```

```
month(debmois, label = TRUE, abbr = FALSE)
```

```
 [1] January   February  March     April     May       June      July      August
 [9] September October   November  December
12 Levels: January < February < March < April < May < June < July < ... < December
```

```
Sys.setlocale("LC_ALL", "")  # Retour aux conventions locales
```

```
[1] "fr_FR.UTF-8/fr_FR.UTF-8/fr_FR.UTF-8/C/fr_FR.UTF-8/fr_FR.UTF-8"
```

```
month(debmois, label = TRUE)
```

```
 [1] jan fév mar avr mai jui jul aoû sep oct nov déc
12 Levels: jan < fév < mar < avr < mai < jui < jul < aoû < sep < oct < ... < déc
```

```
month(debmois, label = TRUE, abbr = FALSE)
```

```
 [1] janvier   février   mars      avril     mai       juin      juillet   août
 [9] septembre octobre   novembre  décembre
12 Levels: janvier < février < mars < avril < mai < juin < juillet < ... < décembre
```

Pour distinguer les courbes de croissance de nos 10 arbres nous allons utiliser une palette de couleur de type qualitative et sombre pour plus de lisibilité. Notre capacité à distinguer les couleurs est malheureusement très limitée et cela sera insuffisant pour repérer facilement les arbres. Pour faciliter la lecture, nous allons ordonner les arbres dans la légende dans le même ordre que la dernière valeur observée.

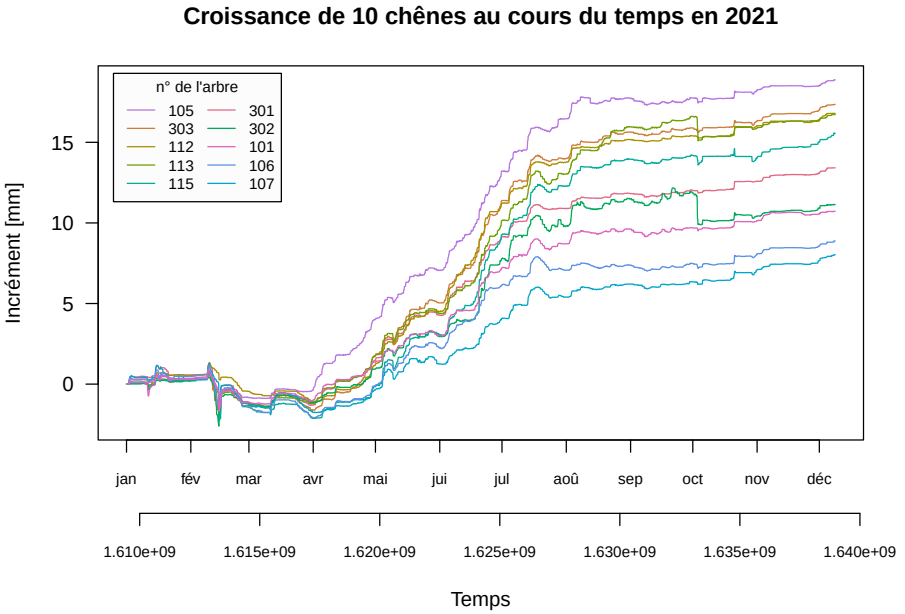
```
mycol <- hcl.colors(10, "Dark 3")
lastvals <- unlist(microd0[nrow(microd0), 1:10])
myo <- rev(order(lastvals))
nomsa[myo]
```

```
[1] "105" "303" "112" "113" "115" "301" "302" "101" "106" "107"
```

C'est donc pour l'arbre n° 105 que l'on a observé le plus grand accroissement de la circonférence, et pour l'arbre n° 107 la plus faible. Nous pouvons maintenant construire notre représentation graphique, cela ne pose pas de difficulté particulière, il faut simplement augmenter la taille de la marge du bas à huit lignes pour pouvoir caser nos deux axes.

```
par(mar  = c(8, 4, 4, 2) + 0.1)
plot.new()
plot.window(xlim = range(microd0$temps), ylim = range(microd0[ , 1:10]))
axis(1, line = 3, cex.axis = 0.75) # Axe en secondes
axis(1, at = debmois, labels = month(debmois, label = TRUE), cex.axis = 0.75)
axis(2, las = 1)
title(ylab = "Incrément [mm]")
title(xlab = "Temps", line = 6)
title(main = "Croissance de 10 chênes au cours du temps en 2021")
box()
for(j in seq_len(10)) lines(microd$temps, microd0[ , j], col = mycol[j])
legend("topleft", inset = 0.02, bg = grey(0.99), legend = nomsa[myo],
       lty = 1, col = mycol[myo], ncol = 2, cex = 0.75,
       title = "nř de l'arbre")
(pu <- par("usr"))
```

```
[1]  1.608271e+09  1.640149e+09 -3.465410e+00  1.974413e+01
```



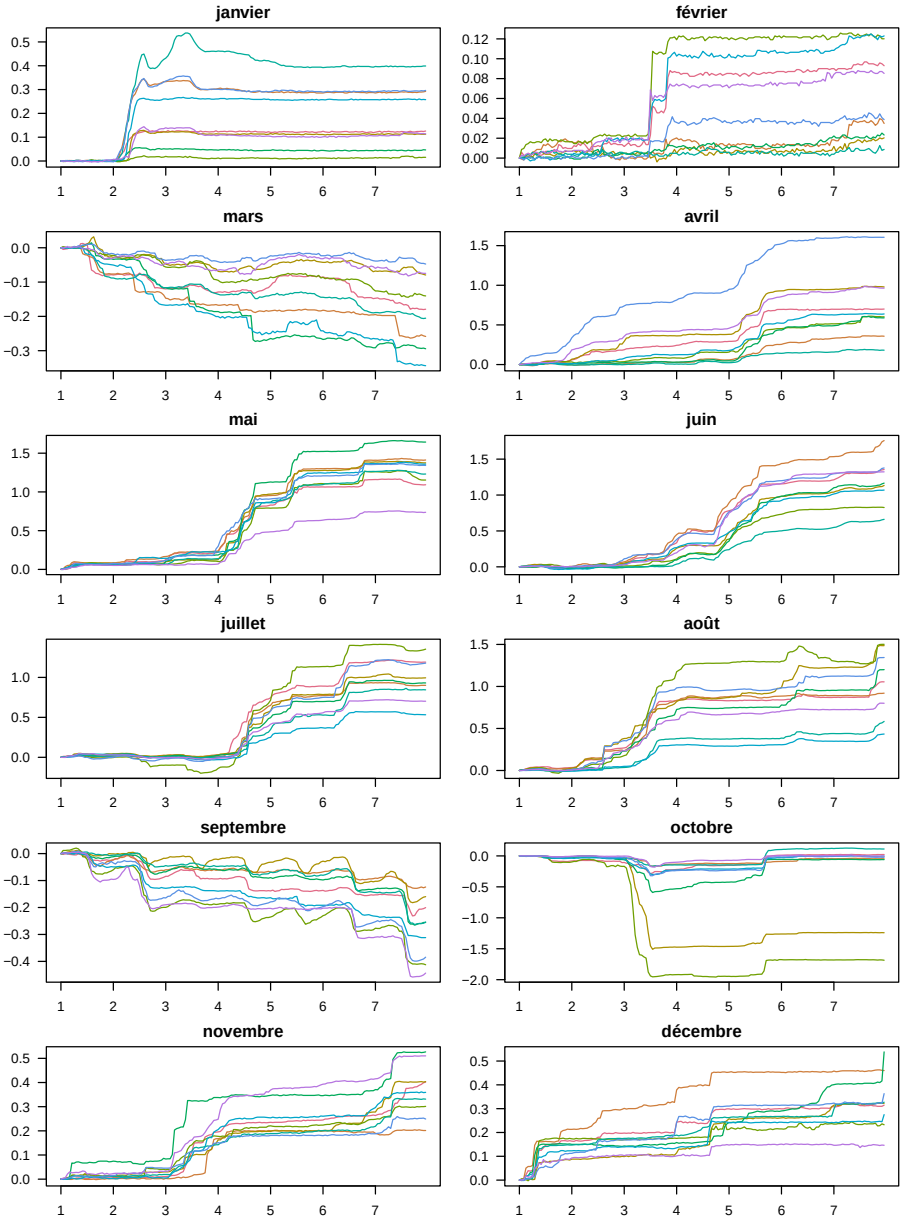
Notez que les coordonnées « utilisateur » sur ce graphique sont tout à fait raisonnables sur l'axe des ordonnées puisque l'on navigue entre -3.47 et 19.74 mais, en revanche, semblent étranges sur l'axe des abscisses puisque l'on va de $1.608.10^9$ à $1.640.10^9$. C'est simplement qu'il s'est écoulé plus d'un milliard de secondes depuis le 1er janvier 1970. Cela ne pose pas problème particulier ici puisque nous pouvons construire les étiquettes des axes pour y faire figurer les mois et non les secondes. Mais si nous voulions ajuster un modèle il serait préférable de changer pour des unités plus raisonnables, par exemple le nombre d'heures, ou le nombre de jours fractionnaires, écoulés depuis le début de l'année en cours (on verra un exemple dans la section 2.2 page 16).

Le gain annuel de circonférence est variable d'un arbre à l'autre puisqu'il va de près de 20 mm pour l'arbre n° 105 à environ 5 mm pour l'arbre n° 107. Ce sont des ordres de grandeur tout à fait corrects puisque, sur deux chênes âgés d'une quarantaine d'années, Robert Marsham [7] rapportait[2] en 1759 des accroissements annuels de la circonférence d'environ 30 mm. On note un bon synchronisme des courbes ce qui suggère que les arbres répondent de la même manière à un facteur environnemental commun. Les décrochages brutaux observés début octobre pour les arbres n° 113 et n° 302 pourraient sembler suspects. Zoomons un peu pour y voir plus clair en représentant la première semaine de chaque mois de façon à avoir une échelle comparable à celle des relevés de Josef Friedrich (figure 3 page 11).

```
plotmois7j <- function(the_mois){
  qui <- which(month(microd$temps) == month(the_mois) &
                 mday(microd$temps) <= 7)
  tmp <- microd[qui, ]
  for(j in seq_len(10)) tmp[ , j] <- tmp[ , j] - tmp[1, j]
  plot.new()
  plot.window(xlim = range(tmp$temps), ylim = range(tmp[ , 1:10]))
  xx <- seq(tmp[1, "temps"], tmp[nrow(tmp), "temps"], by = "day")
```

[2] Voir https://pbil.univ-lyon1.fr/R/pdf/MarshamR1759.pdf

```
  axis(1, at = xx, label = mday(xx))
  axis(2, las = 1)
  title(main = month(the_mois, label = TRUE, abbr = FALSE))
  box()
  for(j in 1:10){
    points(tmp$temps, tmp[ , j], type = "l", col = mycol[j - 1])
  }
}
par(mfrow = c(6, 2), mar = c(2, 3, 2, 1))
for(i in seq_len(12)) plotmois7j(debmois[i])
```



À cette échelle, les décrochages observés début octobre pour les arbres n° 113 et n° 302 n'apparaissent pas aussi brutaux et surtout on voit que tous les arbres sont concernés, même si c'est dans une moindre mesure. Nous pouvons

donc lever le doute quant à la suspicion d'artéfact évoquée *supra*. On voit également ici, en particulier pour le mois de septembre, la composante journalière du signal. Il faut bien admettre que les courbes ne sont pas aussi régulières que celles exhibées par Josef Friedrich. Je ne pense pas que cela soit la conséquence d'une différence dans la force de serrage des feuillards : elle est de 15 à 20 N pour les micro-dendromètres et de 10 à 20 N chez Josef Friedrich. La principale différence pourrait venir de ce que Josef Friedrich utilisait des poulies (voir la figure 2 page 10) qui devaient considérablement limiter les forces de friction entre le feuillard et l'écorce du tronc de l'arbre, et donc générer une réponse plus souple. On sait depuis longtemps que ce rythme circadien est lié à la photosynthèse à cause de sa disparition chez les arbres à feuilles caduques en hiver, ainsi Josef Friedrich rapportait[3] : « les conifères observés ont transpiré les jours secs et sans gel, même en hiver, alors que cela n'a pas été observé chez les arbres à feuilles caduques. La diminution de la circonférence des arbres les jours sans gel s'est produite chez les conifères exactement de la même manière qu'en été, à condition que le sol ne soit pas gelé ou que seule une fine couche de sa surface supérieure soit gelée. »

2.2 Représentation des données lissées

On utilise maintenant le modèle de croissance logistique de Verhulst [12, 13, 14, 15] pour lisser les séries temporelles de variation de la circonférence.

$$y = f(x) = \frac{y_\infty}{1 + e^{\frac{x_{0.5} - x}{\sigma}}} \tag{1}$$

La signification des paramètres est la suivante. y_∞ (`Asym`) représente la valeur de l'asymptote horizontale à droite, obtenue pour les très grandes valeurs de la variable de la fonction. $x_{0.5}$ (`xmid`) est un paramètre de position qui donne la valeur du point d'inflexion. La valeur de la courbe au point d'inflexion vaut la moitié de la valeur de l'asymptote horizontale à droite. s (`scale`) contrôle la rapidité de la transition, plus il est petit, plus elle sera brutale.

Pour les études de croissance des arbres, il est courant d'exprimer le temps par le rang des jours dans l'année. Une première possibilité avec nos données au format `POSIXct` est de retirer le nombre de secondes au début de l'année puis de diviser par le nombre de secondes, 86400, que comporte chaque jour, puis d'ajouter 1 pour que le premier janvier soit de rang 1 :

```
temps1 <- as.numeric(1 + (microd0$temps - microd0$temps[1])/86400)
head(temps1)
```

```
[1] 1.000000 1.041667 1.083333 1.125000 1.166667 1.208333
```

Une seconde possibilité, sans doute plus lisible, est d'utiliser les fonctions `yday()` et `hour()` du paquet `lubridate` :

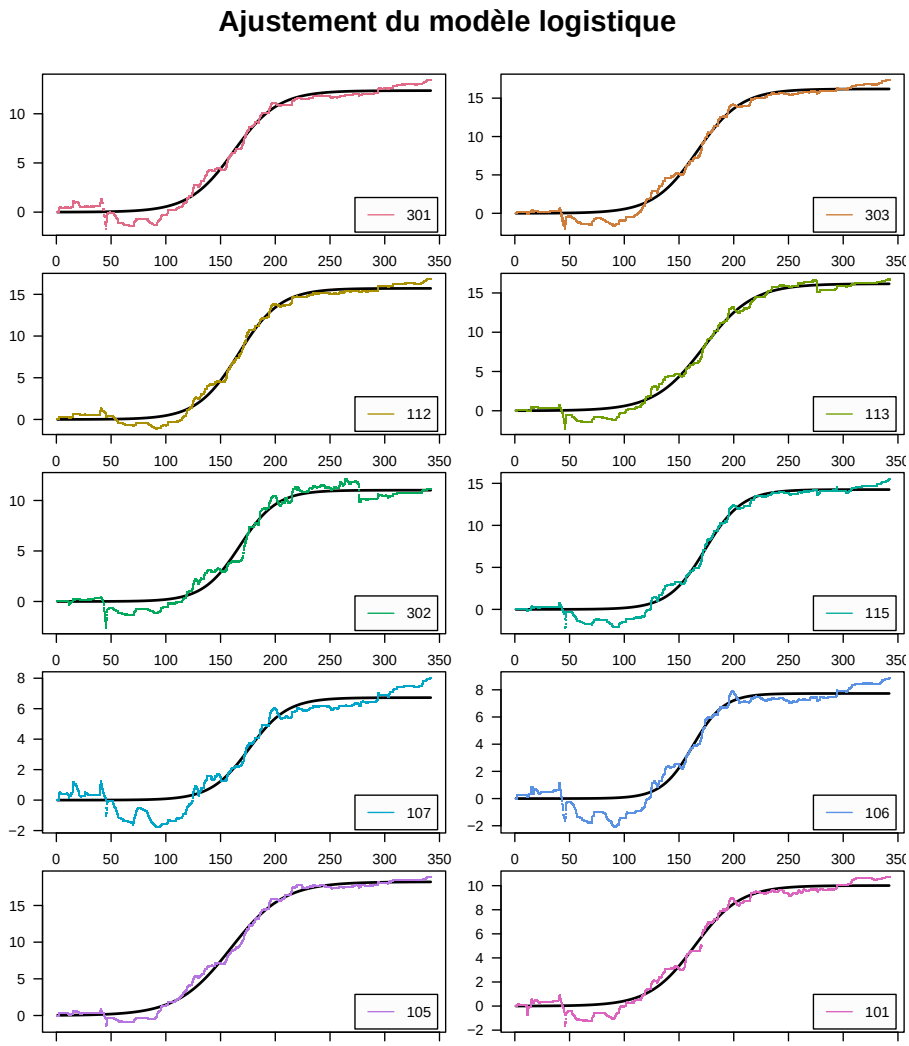
```
temps2 <- yday(microd0$temps) + hour(microd0$temps)/24
all.equal(temps1, temps2)
```

[3] Page 155 de [3] : Die beobachteten Nadelhölzer transpirirten an frostfreien trockenen Tagen auch zur Winterszeit, während dies bei den Laubhölzern nicht beobachtet wurde. Die Abnahme des Baumumfanges an frostfreien Tagen erfolgte bei den Nadelhölzern ganz gleich wie zur Sommerszeit, vorausgesetzt, dass der Boden entweder nicht oder nur eine dünne Schichte der OberHächc desselben gefroren war.

```
[1] TRUE
```

NOUS pouvons maintenant estimer pour chaque arbre les paramètres du modèle logistique. Nous conserverons ces valeurs dans l'objet `microtablog` pour une utilisation ultérieure. La figure suivante représente l'ajustement du modèle logistique aux données :

```
par(mar = c(0.5, 3, 1.5, 0) + 0.1, mfrow = c(5, 2), oma = c(0, 0, 3, 0))
microtablog <- as.data.frame(matrix(NA, ncol = 4, nrow = 10))
colnames(microtablog) <- c("Tree_name", "Asym", "xmid", "scal")
for(j in seq_len(10)){
  microtablog[j, "Tree_name"] <- nomsa[j]
  tmp <- list(y = microd0[, j], yday = temps1)
  resnls <- nls(y ~ SSlogis(yday, Asym, xmid, scal), data = tmp)
  xx <- 1:max(yday(microd0$temps))
  plot(xx, predict(resnls, list(yday = xx)), las = 1, lwd = 2,
       type = "l", ylim = range(tmp$y), ylab = "")
  points(tmp$yday, tmp$y, col = mycol[j], pch = ".")
  legend("bottomright", inset = 0.02, bg = grey(0.99), legend = nomsa[j],
         lty = 1, col = mycol[j])
  microtablog[j, 2:4] <- coef(resnls)
}
title(main = "Ajustement du modèle logistique", outer = TRUE, cex.main = 2)
```

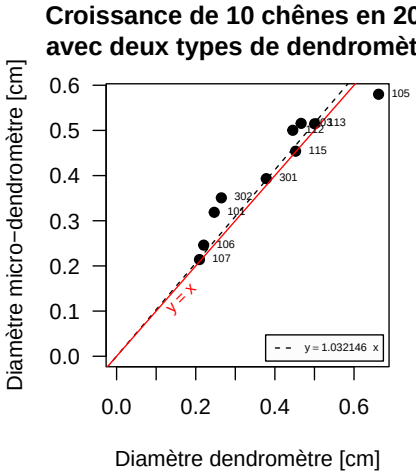


Le modèle logistique résume donc bien la tendance générale et a l'avantage de gommer automatiquement la petite phase de contraction que l'on observe parfois en début d'année (ce phénomène est particulièrement visible pour les arbres n^{os} 107 et 106). La valeur de l'asymptote nous donne ainsi directement l'accroissement annuel. Comme les chênes étaient équipés simultanément avec des dendromètres classiques[4] et des micro-dendromètres on peut comparer les accroissements annuels observés avec ces deux types de capteurs. Les données des dendromètres classiques sont en cm et pour les diamètres des troncs, il faut donc diviser les valeurs des micro-dendromètres par $10 \times \pi$ pour pouvoir les comparer directement. On considère ici que le dendromètre classique est la méthode de référence et on fait une régression par l'origine[5] pour résumer la tendance.

[4] Voir `https://pbil.univ-lyon1.fr/R/pdf/dendroCHS57.pdf`

[5] On pourrait ici préférer faire une régression orthogonale par l'origine pour ne pas privilégier un instrument de mesure par rapport à l'autre, mais on est déjà si proche de l'idéal $y = x$ que je doute que cela change grand chose.

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/dendroCHS57/tablog.Rda"))
tablog <- tablog[tablog$Year == 2021, ]
tablog$Tree_name <- substr(tablog$Tree_name, 7, 9)
comp <- merge(tablog, microtablog, by.x = "Tree_name", by.y = "Tree_name")
x <- comp$Asym.x ; y <- comp$Asym.y/(pi*10)
par(pty = "s")
plot(x, y, pch = 19, las = 1, xlim = c(0, max(x)), ylim = c(0, max(y)),
     main = "Croissance de 10 chênes en 2021\n avec deux types de dendromètres",
     xlab = "Diamètre dendromètre [cm]",
     ylab = "Diamètre micro-dendromètre [cm]")
abline(lm1 <- lm(y~x-1), lty = 2)
abline(c(0, 1), col = "red")
text(x, y, comp$Tree_name, pos = 4, cex = 0.5, xpd = NA)
text(0.1, 0.1, "y = x", srt = 45, pos = 4, col = "red", cex = 0.75)
myleg <- bquote(y == .(lm1$coefficients)*phantom(0)*x)
legend("bottomright", inset = 0.02, legend = myleg, lty = 2, bg = grey(0.99), cex = 0.5)
```

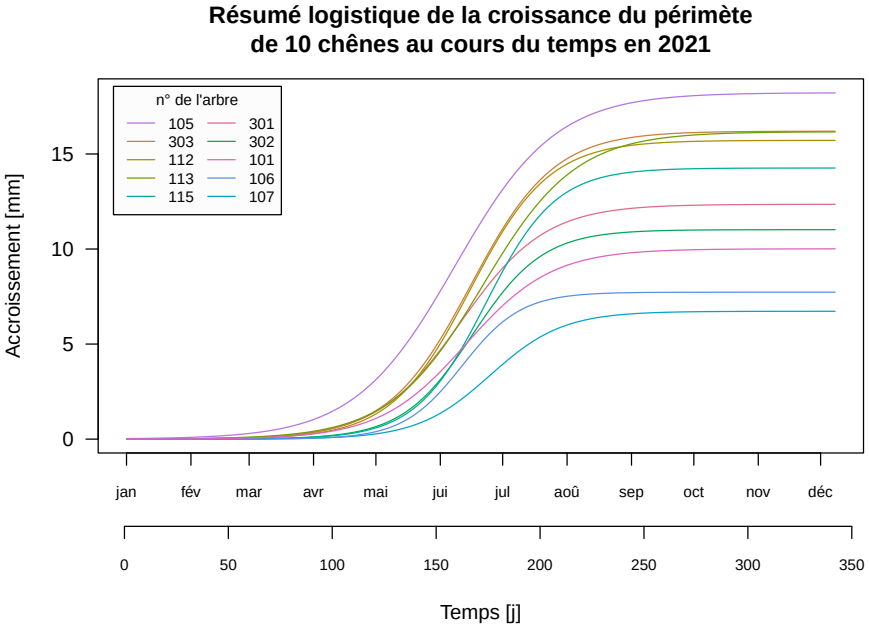


Les résultats sont donc tout à fait comparables avec une différence moyenne de 0.04 cm et une différence maximale de 0.1 cm. C'est rassurant dans le sens où si nous avions fait une grossière erreur de manipulation des données des micro-dendromètres nous ne nous attendrions pas à avoir une telle concordance.

2.3 Intérêt des données lissées

La table `microd0` comporte 90200 valeurs numériques alors que la table `microtablog` des paramètres du modèle logistique n'en comporte plus que 30 : nous avons considérablement résumé l'information.

```
par(mar  = c(8, 4, 4, 2) + 0.1)
plot.new()
plot.window(xlim = range(yday(microd0$temps)),
            ylim = c(0, max(microtablog$Asym)))
axis(1, line = 3, cex.axis = 0.75) # Axe en jours
axis(1, at = yday(debmois), labels = month(debmois, label = TRUE), cex.axis = 0.75)
axis(2, las = 1)
title(ylab = "Accroissement [mm]")
title(xlab = "Temps [j]", line = 6)
title(main = "Résumé logistique de la croissance du périmètre\nde 10 chênes au cours du temps en 2021")
box()
mlog <- function(x, p) p[1]/(1 + exp((p[2] - x)/p[3]))
xx <- seq(min(yday(microd0$temps)), max(yday(microd0$temps)), length.out = 256)
for(i in seq_len(10)) lines(xx, mlog(xx, unlist(microtablog[i, 2:4])), col = mycol[i])
legend("topleft", inset = 0.02, bg = grey(0.99), legend = nomsa[myo],
       lty = 1, col = mycol[myo], ncol = 2, cex = 0.75,
       title = "nř de l'arbre")
```



La représentation obtenue est beaucoup plus épurée que celle avec les données brutes (section 2.1 page 12). Ceci permet de voir que les arbres ayant la plus forte croissance sont également ceux qui sont les plus précoces. On gagne également en place puisque le fichier PDF correspondant à cette figure ne pèse que 18 Kio comparé aux 326 Kio pour le fichier de la figure avec les données brutes.

3 Fluctuations « saisonnières »

Utilisons le jeu de donnée R standard `co2` pour illustrer nos propos. Il s'agit d'un relevé mensuel de la concentration atmosphérique du gaz carbonique en ppm sur le volcan Mauna Loa dans l'archipel d'Hawaï entre 1959 et 1997, inclus. C'est un objet de la classe `ts` pour *time serie*. La fonction `decompose()` permet d'extraire la composante saisonnière :

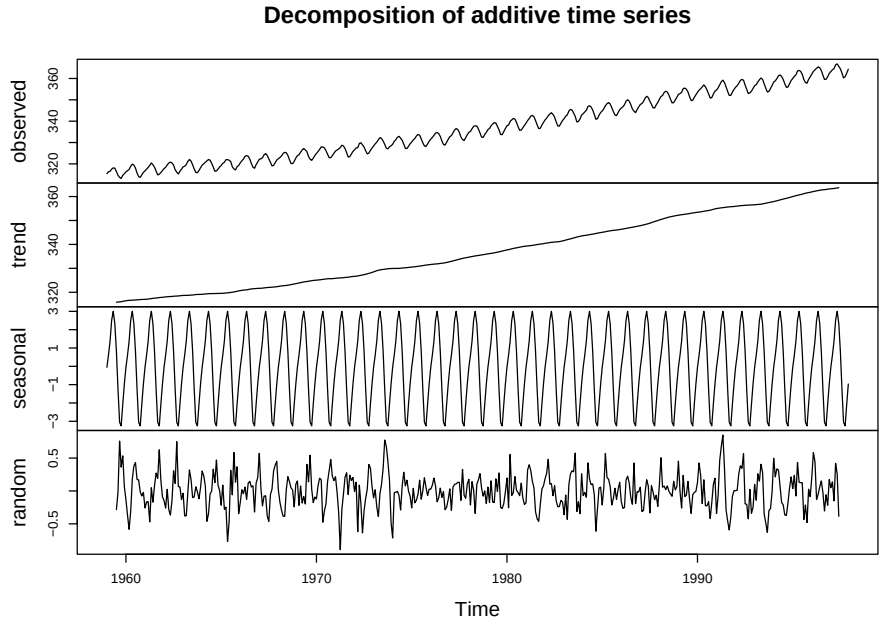
```
data(co2)
class(co2)
```

```
[1] "ts"
```

```
tsp(co2)
```

```
[1] 1959.000 1997.917   12.000
```

```
plot(decompose(co2))
```



Le signal observé, en haut, peut donc donc se décomposer comme la somme d'une tendance à la hausse, de fluctuations saisonnières et de résidus. Les objets de la classe `ts` étant extrêmement employés en économétrie, finance et sciences environnementales on trouvera beaucoup d'illustrations de fluctuations saisonnières. Mais rien ne nous empêche d'utiliser ces outils pour des phénomènes d'une autre nature, par exemple un rythme circadien. La seule petite difficulté vient de l'argument `frequency` de la fonction `ts()` : normalement pour un phénomène périodique la fréquence, exprimée en Hz, est l'inverse de la période exprimée en secondes, soit pour une période de 86400 secondes (un jour) une fréquence de 115.74 μHz. La signification de l'argument `frequency` est toute autre : c'est le nombre d'observations disponibles par période. Dans notre cas avec un suivi horaire nous avons donc 24 observations par jour. Comme la documentation de la fonction `decompose()` précise qu'il est préférable de travailler avec un nombre entier de périodes, nous allons censurer les données incomplètes du dernier jour de la série.

```
dts <- ts(microd0[1:((max(yday(microd0$temps)) - 1)*24) , 1:10],
          start = 0, frequency = 24)
class(dts)
```

```
[1] "mts"    "ts"     "matrix"
```

Nous avons créé un objet de la classe `mts` pour *multivariate time serie* qui hérite des classes `matrix` et `ts`. Nous pouvons donc lui appliquer les opérations habituelles sur les matrices :

```
colnames(dts) <- nomsa
head(dts[, 1:5])
```

```
         301      303     112      113      302
[1,] 0.00000  0.00000 0.00000  0.00000  0.00000
[2,] 0.00000  0.00000 0.00000 -0.00194 -0.00194
[3,] 0.00000 -0.00194 0.00000  0.00000 -0.00194
[4,] 0.00194  0.00000 0.00194  0.00000  0.00000
[5,] 0.00000  0.00193 0.00194  0.00000 -0.00097
[6,] 0.00097  0.00000 0.00194 -0.00097 -0.00097
```

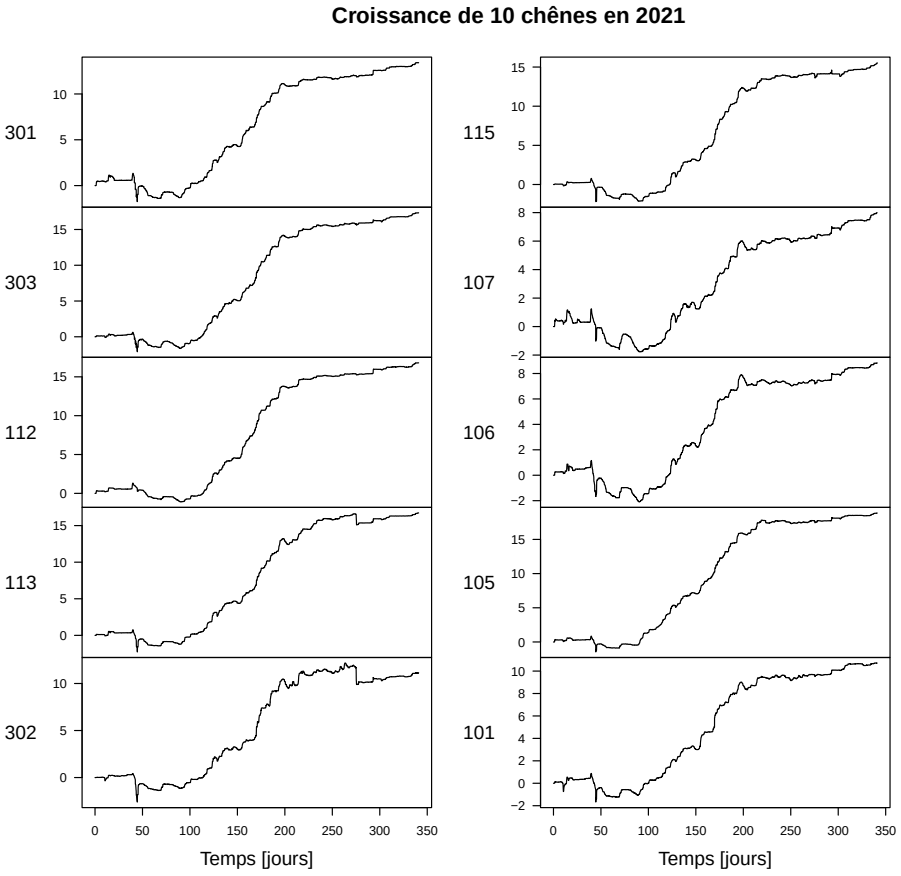
De la classe `ts` nous héritons d'un attribut `tsp` (*time serie parameters*) accessible avec la fonction éponyme `tsp()` qui nous donne l'instant du départ (pour nous le 1er janvier 2021 à minuit en UTC+2 qui est notre zéro), l'instant de fin (le 340.9583 correspond au 341e jour de l'année à 23h puisque nous partons de 0) et le nombre d'observations par période (24 heures par jour) :

```
tsp(dts)
```

```
[1]   0.0000 340.9583  24.0000
```

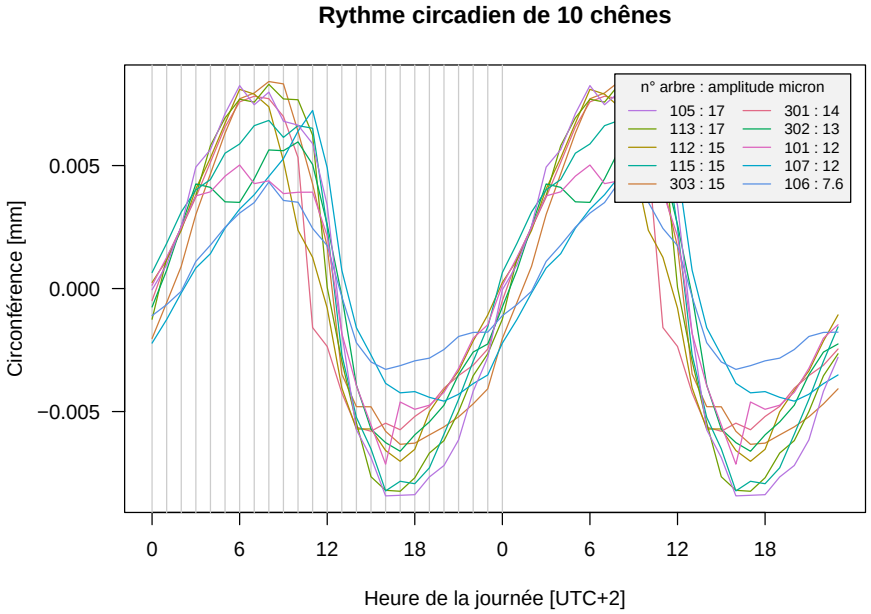
Notons qu'il existe une fonction `plot()` spécialisée pour les objets de la classe `mts` qui permet de produire rapidement un graphique pour avoir une vue d'ensemble des données :

```
plot(dts, las = 1, main = "Croissance de 10 chênes en 2021",
     xlab = "Temps [jours]")
```



Une utilisation un peu brutale de la fonction `plot(decompose())` s'avère assez décevante parce que, évidement, avec 341 périodes on ne peut pas visualiser correctement la composante périodique du signal. Nous allons zoomer de façon à ne représenter que deux périodes, soit de 0 à 47 heures, et utiliser l'opérateur modulo `%%` pour calculer l'heure de la journée :

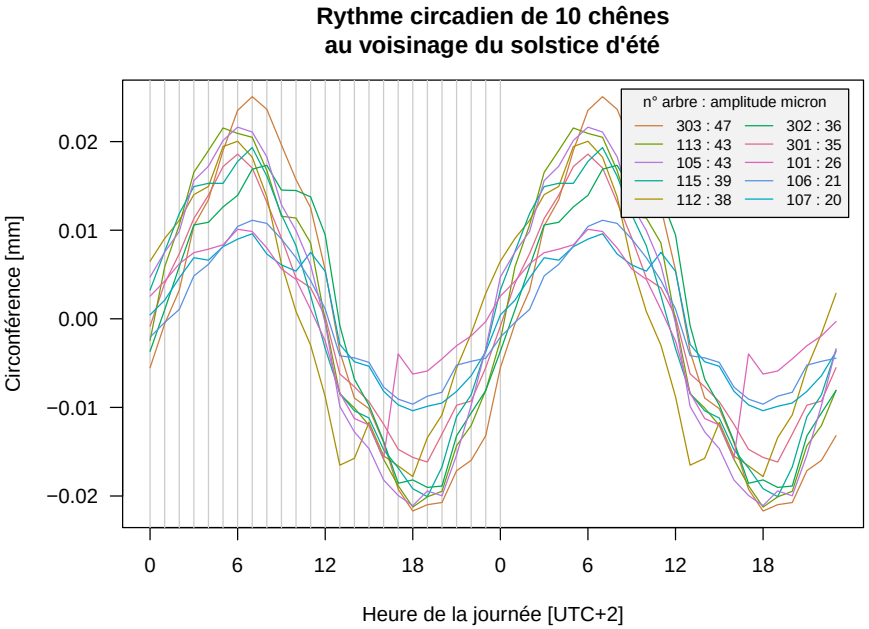
```
resdec <- list()
for(i in seq_len(10)) resdec[[i]] <- decompose(dts[ , i])
plotcirca <- function(resdec, xx = 0:47,
          xlab = "Heure de la journée [UTC+2]",
          ylab = "Circonférence [mm]",
          main = "Rythme circadien de 10 chênes",
          unit = "micron", fac = 1000){
  ylim <- range(unlist(lapply(resdec, \(x) x$seasonal[xx + 1])))
  ampl <- unlist(lapply(resdec, \(x) max(x$seasonal -min(x$seasonal))))
  myo <- rev(order(ampl))
  par(mar = c(5, 5, 4, 2) + 0.1)
  plot.new() ; plot.window(xlim = range(xx), ylim = ylim)
  title(xlab = xlab, main = main) ; box()
  title(ylab = ylab, line = 4)
  hseq <- seq(0, max(xx), by = 6)
  axis(1, at =  hseq, labels = hseq %% 24)
  axis(2, las = 1)
  abline(v = 0:24, lty = 1, col = grey(0.8))
  for(i in seq_len(10)) lines(xx, resdec[[i]]$seasonal[xx + 1], col = mycol[i])
  legend <- paste(nomsa[myo], ":", signif(fac*ampl[myo], 2))
  legend("topright", inset = 0.02, bg = grey(0.95), legend = legend,
         lty = 1, col = mycol[myo], ncol = 2, cex = 0.75,
         title = paste("nř arbre : amplitude", unit))
}
plotcirca(resdec)
```



La croissance nocturne de la circonférence des troncs est bien visible, elle commence en fait vers 16h (heure d'été) pour culminer le lendemain matin vers 8h. Elle est asymétrique dans le sens où la contraction est plus brutale que l'extension. L'intensité du phénomène n'est pas non plus dramatique puisque l'on a une amplitude allant jusqu'à 17 μm pour la variation de la circonférence du tronc, à comparer à une croissance annuelle de l'ordre de 1 cm.

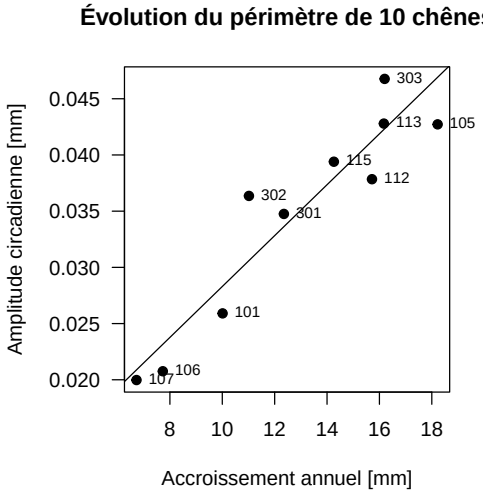
Le signal est quand même assez bruité. Essayons d'améliorer les choses en ne considérant pas les périodes sans croissance et en travaillant au voisinage du solstice d'été (entre le 21 mai et le 21 juillet, inclus) lorsque la durée du jour solaire est relativement constante.

```
sol <- subset(microd0, temps >= as.POSIXct("2021-05-21") &
                       temps <  as.POSIXct("2021-07-22"))
solts <- ts(sol[ , 1:10], start = 0, frequency = 24)
soldec <- list()
for(i in seq_len(10)) soldec[[i]] <- decompose(solts[ , i])
plotcirca(soldec, main = "Rythme circadien de 10 chênes\nau voisinage du solstice d'été")
```



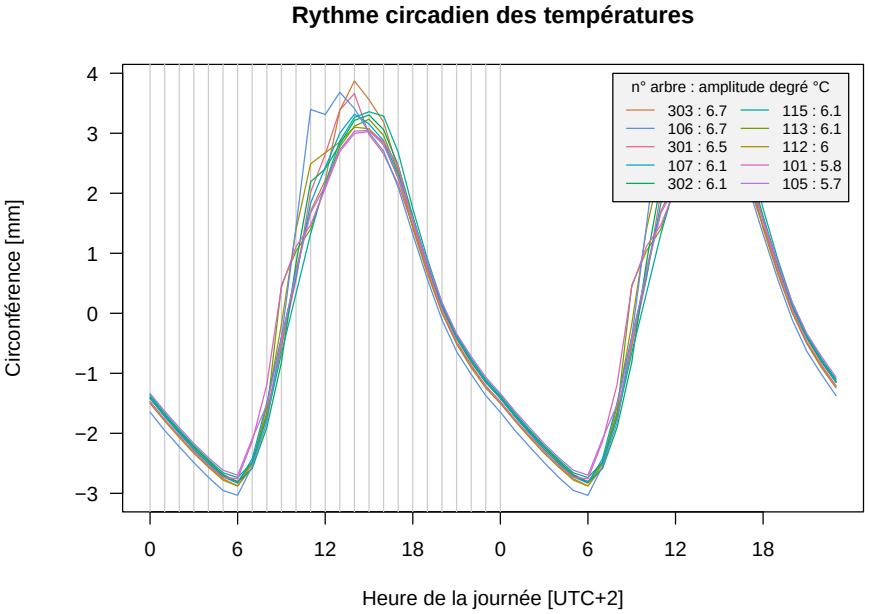
On obtient un signal plus régulier et plus symétrique puisque maintenant les phases d'expansion et de contraction alternent entre 6 heures du soir et 6 heures du matin. L'amplitude est également plus importante : jusqu'à 47 μm. Si on compare l'amplitude des variations circadiennes avec l'accroissement annuel on constate sans surprise qu'il y a une forte corrélation entre les deux : les arbres avec des gros troncs on une plus forte croissance annuelle du périmètre et une plus grande amplitude de la variation circadienne d'icelui.

```
ampl <- unlist(lapply(soldec, \(x) max(x$seasonal -min(x$seasonal))))
par(pty = "s", mar = c(5, 5, 4, 2) + 0.1)
x <- microtablog$Asym ; y <- ampl
plot(x, y, las = 1, ylab = "", pch = 19, xlab = "Accroissement annuel [mm]",
     main = "Évolution du périmètre de 10 chênes")
abline(lm(y~x))
title(ylab = "Amplitude circadienne [mm]", line = 4)
text(x, y, nomsa, pos = 4, xpd = NA, cex = 0.75)
```



Comme exercice on laissera au lecteur le soin d'essayer de faire la même chose avec les relevés horaires des températures :

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/microdendroCHS57/meteo.Rda"))
```



4 Confrontation de séries temporelles

4.1 Séries de même résolution temporelle

Quand on dispose de séries de même résolution temporelle, la principale opération que l'on peut-être amené à faire est de trouver une plage temporelle

commune à deux séries. Par exemple, on pourrait vouloir restreindre nos séries de température et de croissance à la seule saison de végétation des arbres.

D'après [6] pour le site CHS57A le débourrement des chênes commence en moyenne le 9 avril et la durée moyenne de la saison de végétation est de 186 jours. La fonction `ISOdatetime()` nous permet de calculer très facilement au format `POSIXct` la variable `ddd`, pour l'instant du « début du débourrement », dans le fuseau horaire UTC+02 qui nous intéresse :

```
ddd <- ISOdatetime(year = 2021, month = 4, day = 9,
                   hour = 0, min = 0, sec = 0,
                   tz = "Etc/GMT-2")
class(ddd)
```

```
[1] "POSIXct" "POSIXt"
```

```
ddd
```

```
[1] "2021-04-09 +02"
```

Mais comment calculer la variable `fsv`, pour l'instant de la « fin de saison de végétation », au format `POSIXct` ? C'est très simple grâce à la fonction `yday()` du paquet `lubridate` qui permet de modifier le rang du jour dans l'année d'une variable de la classe `POSIXct` :

```
fsv <- ddd
yday(fsv) <- yday(fsv) + 186
fsv
```

```
[1] "2021-10-12 +02"
```

```
fsv - ddd
```

```
Time difference of 186 days
```

Il nous est maintenant très facile avec la fonction `subset()` de construire des tables de données restreintes à la saison de végétation :

```
microdV <- subset(microd, temps >= ddd & temps < fsv)
# Pour partir de 0
for(j in seq_len(10)) microdV[ , j] <- microdV[ , j] - microdV[1 , j]
meteoV <- subset(meteo, temps >= ddd & temps < fsv)
```

Les plages temporelles peuvent être facilement manipulées avec les objets de la classe `Interval` du paquet `lubridate`. Ils peuvent être créés de deux façons : avec la fonction `interval()` ou bien avec l'opérateur infixe `%-%` :

```
plageV <- interval(ddd, fsv)
plageV
```

```
[1] 2021-04-09 +02--2021-10-12 +02
```

```
plageUNIX <- origin %--% now()
plageUNIX
```

```
[1] 1970-01-01 UTC--2022-07-27 09:41:36 UTC
```

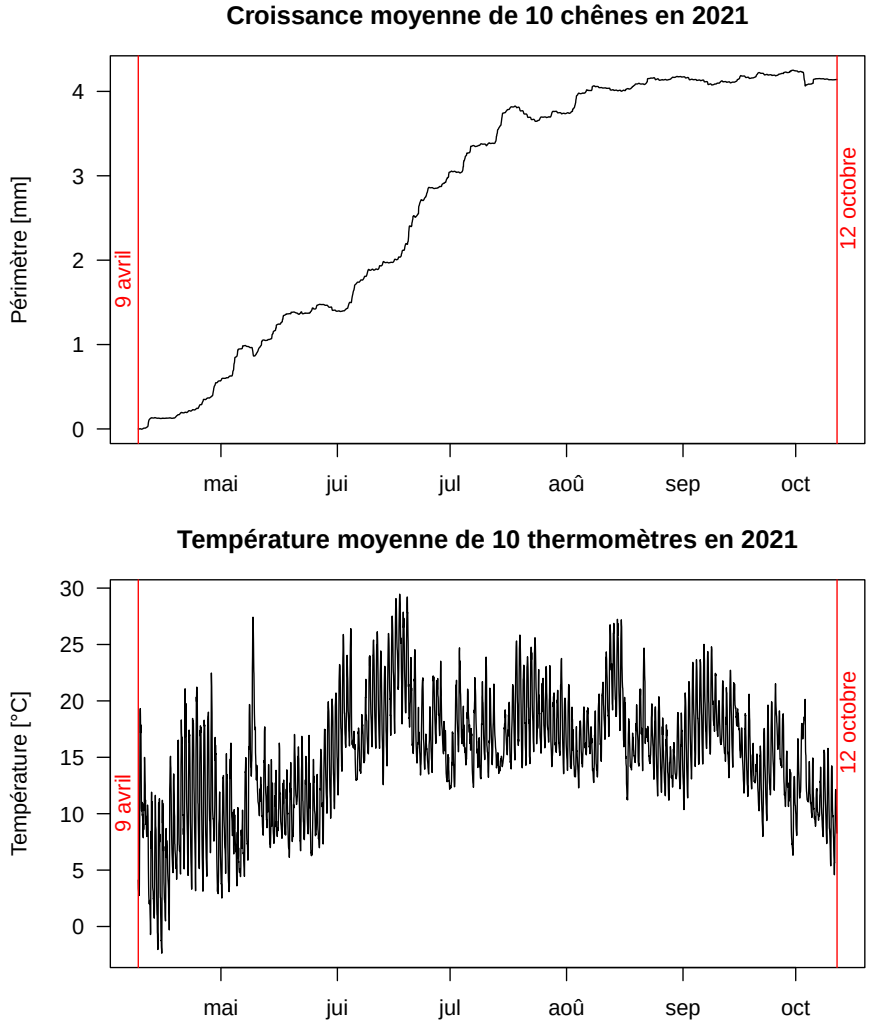
Les opérateurs ensemblistes classiques[6] d'intersection et d'union définis dans le paquet `base` sont également définis dans le paquet `lubridate` pour pouvoir travailler avec des objets de la classe `Interval`. On dispose également de l'opérateur infixe `%within%` au lieu du classique `%in%` :

[6] L'opérateur de différence asymétrique `setdiff()` est également implémenté mais non défini si le résultat n'est pas un intervalle continu, par exemple avec `setdiff(plageUNIX, plageV)`. On y perd la belle généralité vraie pour tout ensemble `x` et `y` : `setequal(union(x, y), c(setdiff(x, y), intersect(x, y), setdiff(y, x)))`. Pour la conserver il faudrait que les objets de la classe `Interval` soient des ensembles de plages temporelles disjointes, ce qui serait beaucoup plus compliqué à gérer.

```
intersect(plageV, plageUNIX)
[1] 2021-04-09 +02--2021-10-12 +02
union(plageV, plageUNIX)
[1] 1970-01-01 02:00:00 +02--2022-07-27 11:41:36 +02
plageV %within% plageUNIX
[1] TRUE
plageUNIX %within% plageV
[1] FALSE
```

Pour vérifier que nous avons bien sélectionné les données de la plage temporelle qui nous intéresse, on désire produire un graphique avec des traits verticaux au début et à la fin de la saison de végétation. C'est assez facile, avec par exemple `abline(v = ddd)`, mais on désire également documenter ces lignes pour qu'elles soient interprétables par un être humain. Ce sera le travail de la fonction `vérif()` ci-après. Comme nos données sont très synchrones entre les 10 capteurs, tant pour la croissance que pour la température, on ne travaillera dorénavant qu'avec les valeurs moyennes.

```
vérif <- function(){
  abline(v = ddd, col = "red")
  abline(v = fsv, col = "red")
  pu <- par("usr")
  ymid <- (pu[3] + pu[4])/2
  human <- function(x) paste(day(x), month(x, label = TRUE, abb = FALSE))
  text(ddd, ymid, human(ddd), srt = 90, col = "red", pos = 2)
  text(fsv, ymid, human(fsv), srt = 90, col = "red", pos = 4)
}
par(mfrow = c(2, 1), mar = c(2, 4, 3, 2) + 0.1)
Pmoy <- rowMeans(microdV[ , 1:10])/pi
plot(microdV$temps, Pmoy, type = "l", xlab = "",
     main = "Croissance moyenne de 10 chênes en 2021", las = 1,
     ylab = "Périmètre [mm]")
vérif()
Tmoy <- rowMeans(meteoV[ , 1:10])
plot(meteoV$temps, Tmoy, type = "l", xlab = "",
     main = "Température moyenne de 10 thermomètres en 2021", las = 1,
     ylab = "Température [ř C]")
vérif()
```



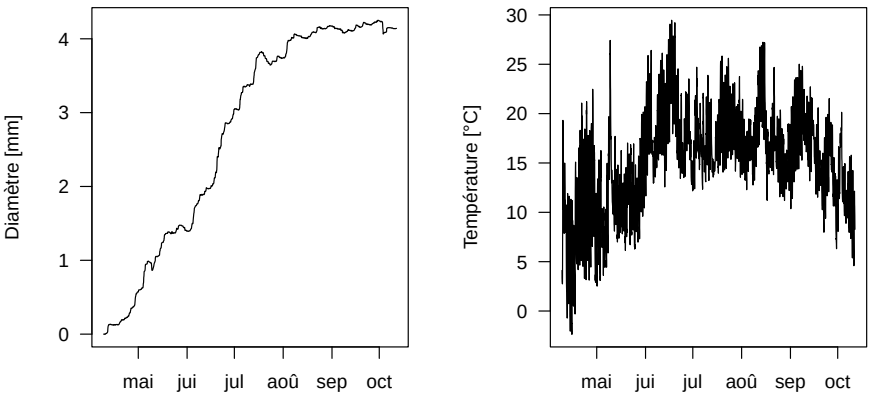
Cette représentation graphique nous permet de vérifier qu'il n'y a pas eu d'erreur *grossière* et que nous travaillons bien avec les données de la saison de la végétation entre le 9 avril et le 12 octobre 2021. Mais ne peut-il pas y avoir une erreur plus subtile ? En particulier, pour optimiser l'utilisation de la fonction `decompose()`, nous souhaitons avoir un nombre *entier* de périodes. Vérifions ce qu'il en est.

```
nrow(microdV)/24
```
```
[1] 186
```

Nous avons donc bien exactement 186 jours pour la saison de végétation. Notez que ce résultat n'est obtenu que grâce à la condition d'inégalité stricte `temps < fsv` utilisée lors de l'utilisation de la fonction `subset()`. Avec une condition `temps <= fsv` nous aurions inclus la valeur à minuit le jour suivant et eu un nombre non entier de périodes.

Dans la représentation graphique *supra* nous avons confronté les deux séries temporelles en les plaçant l'une au-dessus de l'autre et non côte à côte. Il est évident que l'on facilitera grandement la lecture en procédant ainsi :

Ne pas représenter les séries temporelles côte à côte

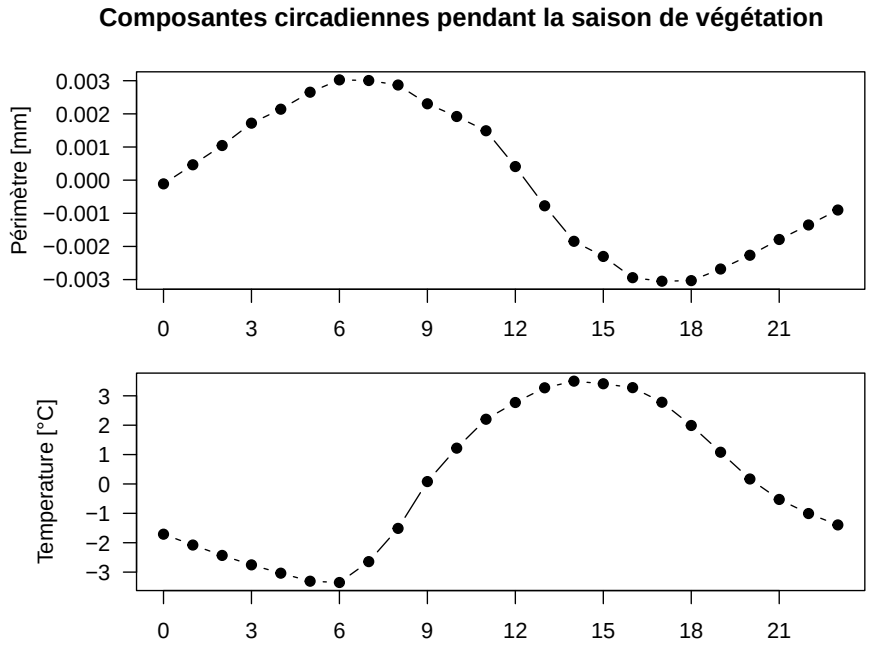


Nos deux séries ayant la même plage temporelle on pourrait vouloir les convertir en un seul objet de la classe `mts`, c'est à dire une série temporelle multivariée. Le problème de procéder ainsi est que lorsque nous allons invoquer la fonction `decompose()` elle va tenter d'extraire un signal circadien *commun* aux deux séries, alors qu'il n'y a pas de raison de penser qu'elles soient synchrones. Il nous faut donc traiter nos deux séries séparément :

```
tsVP <- ts(Pmoy, start = 0, frequency = 24)
tsVT <- ts(Tmoy, start = 0, frequency = 24)
dtsVP <- decompose(tsVP)
dtsVT <- decompose(tsVT)
```

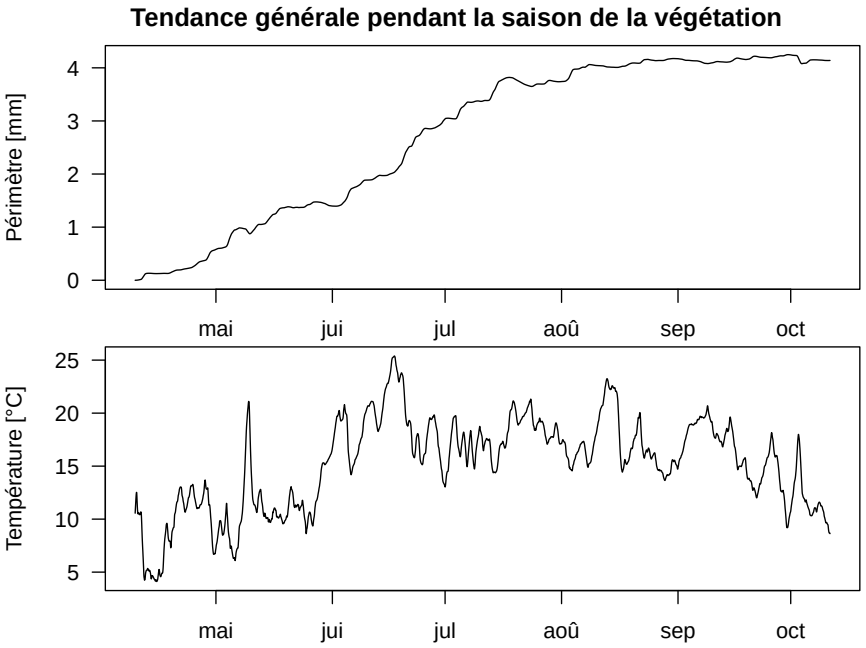
Commençons par nous pencher sur la composante circadienne du signal. Nous avions déjà vu dans la section 3 page 20 qu'il y avait une fluctuation périodique et synchrone des périmètres des arbres et des températures au cours de la journée, on s'attend donc à les retrouver avec les données moyennées.

```
par(mfrow = c(2, 1), mar = c(2, 5, 1, 2) + 0.1, pch = 19, las = 1,
    oma = c(0, 0, 2, 0))
xx <- seq(0, 23, by = 3)
plot(0:23, dtsVP$seasonal[1:24], ylab = "", type = "b", xaxt = "n")
axis(1, at = xx)
title(ylab = "Périmètre [mm]", line = 4)
plot(0:23, dtsVT$seasonal[1:24], ylab = "Temperature [ř̆C]", type = "b", xaxt = "n")
axis(1, at = xx)
title(main = "Composantes circadiennes pendant la saison de végétation", outer = TRUE)
```



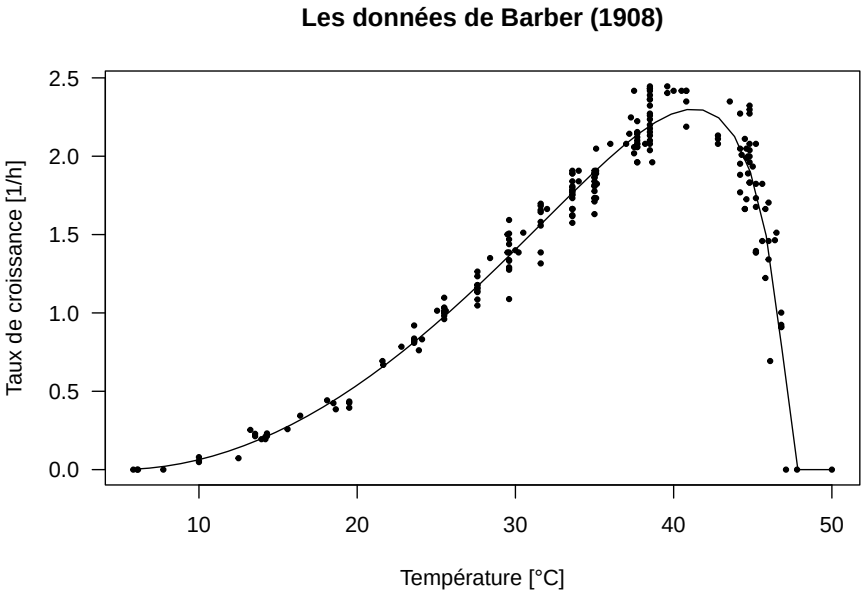
Les deux composantes circadiennes sont quasiment en opposition de phase, mais pas exactement puisque le maximum de température est observé à 14h (c'est à dire midi heure solaire puisque nous sommes en UTC+2) alors que la contraction maximale est à 17h. La température est en fait un *proxy* de la photopériode, dès qu'il fait jour les chênes commencent à transpirer, ce qui pompe la sève brute et contracte le tronc. Pour donner un ordre de grandeur, un chêne de la même classe d'âge que ceux étudiés ici peut transpirer jusqu'à 400 litres d'eau par jour [2]. Le maximum de contraction est observé alors qu'il fait encore jour, sans doute parce qu'il finit par s'établir un équilibre avec la sève brute pompée par les racines et la sève élaborée descendante. Voyons maintenant la tendance générale au cours de la saison.

```
par(mfrow = c(2, 1), mar = c(2, 4, 0, 2) + 0.1, oma = c(0, 0, 2, 0), las = 1)
plot(microdV$temps, dtsVP$trend, type = "l", ylab = "Périmètre [mm]")
plot(meteoV$temps, dtsVT$trend, type = "l", ylab = "Température [ř C]")
title(main = "Tendance générale pendant la saison de la végétation", outer = TRUE)
```



En éliminant la composante circadienne, nous avons considérablement amélioré la lisibilité de nos graphiques, en particulier pour la courbe des températures. Mais ce n'est pas encore complètement satisfaisant à cause de la nature très différente de nos variables : nous avons en haut une variable *extensive* en mm et en bas une variable *intensive* en degrés Celcius. L'influence de la température sur la vivacité des organismes poïkilothermes est étudiée depuis longtemps, voici par exemple l'effet d'icelle sur le taux de croissance de la bactérie *Escherichia coli* telle qu'observée par Barber [1] en 1908 et résumée par le modèle CTMI [10, 8].

```
barber <- read.table(file="http://pbil.univ-lyon1.fr/R/donnees/barber.txt", header = TRUE)
CTMI <- function(T, param){
  Tmin <- param[1] ; Topt <- param[2] ; Tmax <- param[3] ;Muopt <- param[4]
  if( T <= Tmin || T >= Tmax ) return(0)
  Num <- (T-Tmax)*(T-Tmin)^2
  Den <- (Topt-Tmin)*((Topt-Tmin)*(T-Topt)-(Topt-Tmax)*(Topt+Tmin-2*T))
  return(Muopt*Num/Den)
}
sceCTMI <- function(param, data){
  xobs <- data[ ,1] ; yobs <- data[ ,2] ; ytheo <- sapply(xobs, CTMI, param)
  return( sum((yobs-ytheo)^2) )
}
nlm2 <- nlm( f = sceCTMI, p = c( 10, 40, 50, 2.5 ), data = barber )
xaxis <- seq(from = min(barber$Temp), to = max(barber$Temp), by = 1)
ytheo <- sapply(xaxis, \(x) CTMI(x, nlm2$estimate))
plot(barber, main = "Les données de Barber (1908)",
     las = 1, pch = 19, cex = 0.5, xlab = "Température [řC]",
     ylab = "Taux de croissance [1/h]")
lines(xaxis, ytheo)
```



Nous souhaitons donc confronter les températures avec la vitesse de croissance des périmètres des troncs. Si on note, pour un arbre donné, t_i la date de la i^{e} observation ($i \in \{1, 2, \ldots, n\}$) du périmètre du tronc, P_i, alors la vitesse instantanée, V_i, est définie par :

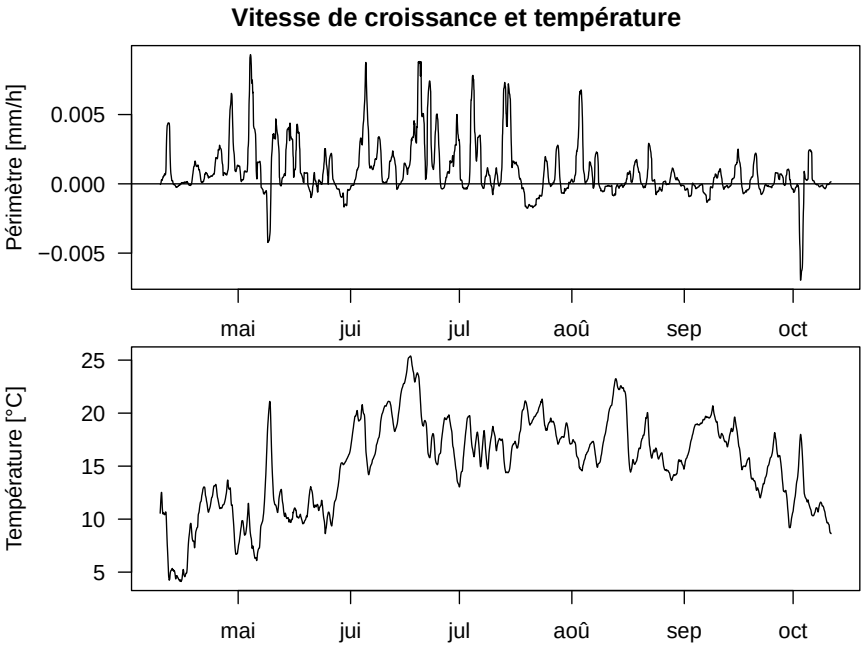
$$\forall i \in \{2, 3, \ldots, n\} : V_i = \frac{P_i - P_{i-1}}{t_i - t_{i-1}} = \frac{\Delta P_i}{\Delta t_i}$$

La vitesse instantanée n'est pas calculable pour l'instant initial, t_1, ce qui explique les constructions du type `temps[-1]` ci-après pour censurer le premier instant. Dans le cas présent, si nous décidons d'exprimer les vitesses instantanées en mm.h^{-1}, l'expression se simplifie en :

$$\forall i \in \{2, 3, \ldots, n\} : V_i = \Delta P_i$$

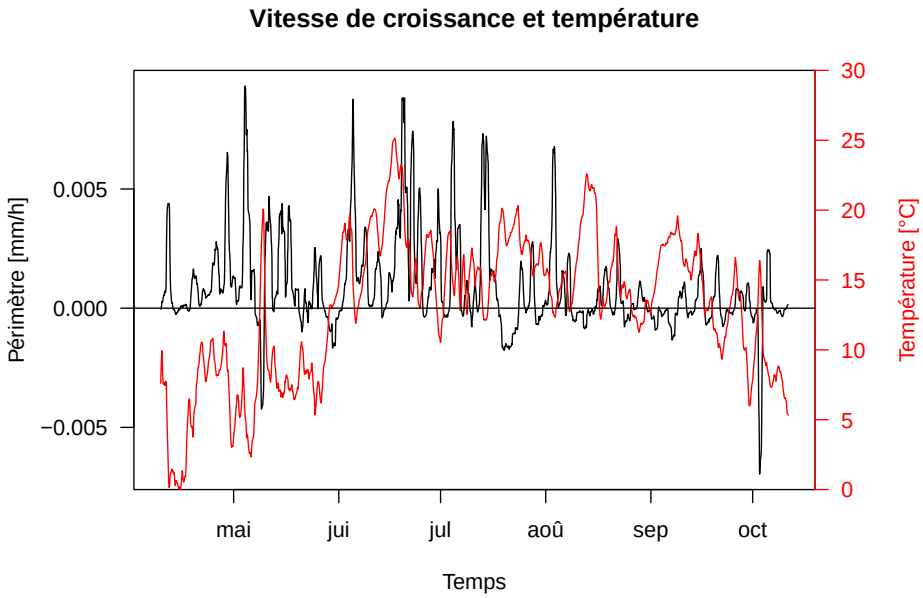
puisque nous avons un pas temporel constant d'une heure ($\forall i \in \{2, 3, \ldots, n\} : \Delta t_i = 1$). La fonction `diff()` nous permet de calculer directement les ΔP_i :

```
par(mfrow = c(2, 1), mar = c(2, 5, 0, 2) + 0.1, oma = c(0, 0, 2, 0), las = 1)
plot(microdV$temps[-1], diff(dtsVP$trend), type = "l", ylab = "")
abline(h = 0)
title(ylab = "Périmètre [mm/h]", line = 4)
plot(meteoV$temps[-1], dtsVT$trend[-1], type = "l", ylab = "")
title(ylab = "Température [řC]", line = 4)
title(main = "Vitesse de croissance et température", outer = TRUE)
```



DANS le cas particulier où on cherche à ne confronter que deux séries temporelles ayant des unités différentes, on peut essayer de produire un graphique avec deux axes en ordonnée :

```
plot2 <- function(x, y1, y2, ylab1, ylab2, line1 = 3, line2 = 3, las = 1, ...){
  par(mar = c(5, line1 + 1, 4, line2 + 1))
  plot(x, y1, type = "l", ylab = "", las = las, ...)
  title(ylab = ylab1, line = line1)
  pu <- par("usr")
  yconv <- function(x)
    (pu[4]-pu[3])*(x - min(x, na.rm = T))/max(x, na.rm = TRUE) + pu[3]
  lines(x, yconv(y2), col = "red")
  axis(4, at = yconv(pretty(y2)), labels = pretty(y2), col = "red",
       col.tick = "red", las = las, col.axis = "red")
  mtext(text = ylab2, line = line2, side = 4, col = "red")
}
plot2(microdV$temps[-1], diff(dtsVP$trend), dtsVT$trend[-1],
      ylab1 = "Périmètre [mm/h]", line1 = 4,
      ylab2 = "Température [ř C]",
      main = "Vitesse de croissance et température",
      xlab = "Temps")
abline(h = 0)
```



Comme on peut le constater il n'y a pas de lien évident entre la température et la vitesse de croissance. Il y a sans doute bien d'autres aspects des conditions environnementales (ensoleillement, précipitations, humidité de l'air et du sol) qu'il faudrait prendre en considération pour être plus prédictif.

4.2 Séries de résolution temporelle différente (*e.g.* `Date` *vs.* `POSIXct`)

Quand on dispose de deux séries avec une résolution temporelle différente on a besoin de construire une échelle commune pour pouvoir les comparer. Il y a deux approches possibles :

1° injecter la série à basse résolution dans la série à haute résolution et donc travailler avec une échelle commune à haute résolution ;

2° compresser la série à haute résolution en une série à basse résolution et donc travailler avec une échelle commune à basse résolution.

La série à basse résolution temporelle que nous allons utiliser ici est issue du suivi hebdomadaire de la croissance avec des dendromètres classiques (voir la figure 1 page 8) des mêmes dix chênes la même année[7]. Les données sont les diamètres des troncs en cm, on multipliera donc par $10 \times \pi$ pour avoir les circonférences en mm. On translatera les valeurs de façon à partir de zéro et on homogénéisera le nom des arbres.

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/dendroCHS57/diacm.Rda"))
Pmm <- diacm
# On ne garde que l'année 2021
Pmm <- Pmm[year(Pmm$Date) == 2021, ]
```

[7] Pour plus de détails on consultera la fiche « Manipulation de données calendaires appliquée au suivi hebdomadaire de la croissance de dix chênes pendant sept ans » à `https://pbil.univ-lyon1.fr/R/pdf/dendroCHS57.pdf`

```
# On passe en mm et à la circonférence
Pmm[ , 1:10] <- pi*10*Pmm[ , 1:10]
# On part de zéro
for(j in 1:10) Pmm[ , j] <- Pmm[ , j] - Pmm[1, j]
# Homogénéisation du nom des arbres
colnames(Pmm)[1:10] <- paste0("CHS57A_", substr(colnames(Pmm)[1:10], 7, 9))
# Vérification de cohérence
stopifnot(all(names(Pmm)[1:10] %in% names(microd0)[1:10]))
names(Pmm)
```

```
[1] "CHS57A_101" "CHS57A_105" "CHS57A_106" "CHS57A_107" "CHS57A_115" "CHS57A_302"
[7] "CHS57A_303" "CHS57A_112" "CHS57A_113" "CHS57A_301" "Date"
```

```
names(microd0)
```

```
[1] "CHS57A_301" "CHS57A_303" "CHS57A_112" "CHS57A_113" "CHS57A_302" "CHS57A_115"
[7] "CHS57A_107" "CHS57A_106" "CHS57A_105" "CHS57A_101" "temps"
```

Les noms des arbres ne sont pas dans le même ordre dans nos deux jeux de données. Ce n'est pas très grave, ce qui est important c'est d'avoir vérifié que nous avions bien construit une clef d'identification commune pour les désigner. Pour ne pas nous enquiquiner dans la suite nous allons ré-ordonner les dix premières colonnes de la table `Pmm` de façon à suivre celles de la table `microd0`, nous pourrons ainsi utiliser directement le « petit nom » des arbres déjà défini *supra* dans la variable `nomsa`.

```
Pmm <- Pmm[ , c(match(names(microd0)[1:10], names(Pmm)[1:10]), 11)]
names(microd0)
```

```
[1] "CHS57A_301" "CHS57A_303" "CHS57A_112" "CHS57A_113" "CHS57A_302" "CHS57A_115"
[7] "CHS57A_107" "CHS57A_106" "CHS57A_105" "CHS57A_101" "temps"
```

```
names(Pmm)
```

```
[1] "CHS57A_301" "CHS57A_303" "CHS57A_112" "CHS57A_113" "CHS57A_302" "CHS57A_115"
[7] "CHS57A_107" "CHS57A_106" "CHS57A_105" "CHS57A_101" "Date"
```

```
all(substr(names(Pmm)[1:10], 8, 10) == nomsa)
```

```
[1] TRUE
```

4.2.1 Échelle commune à haute résolution

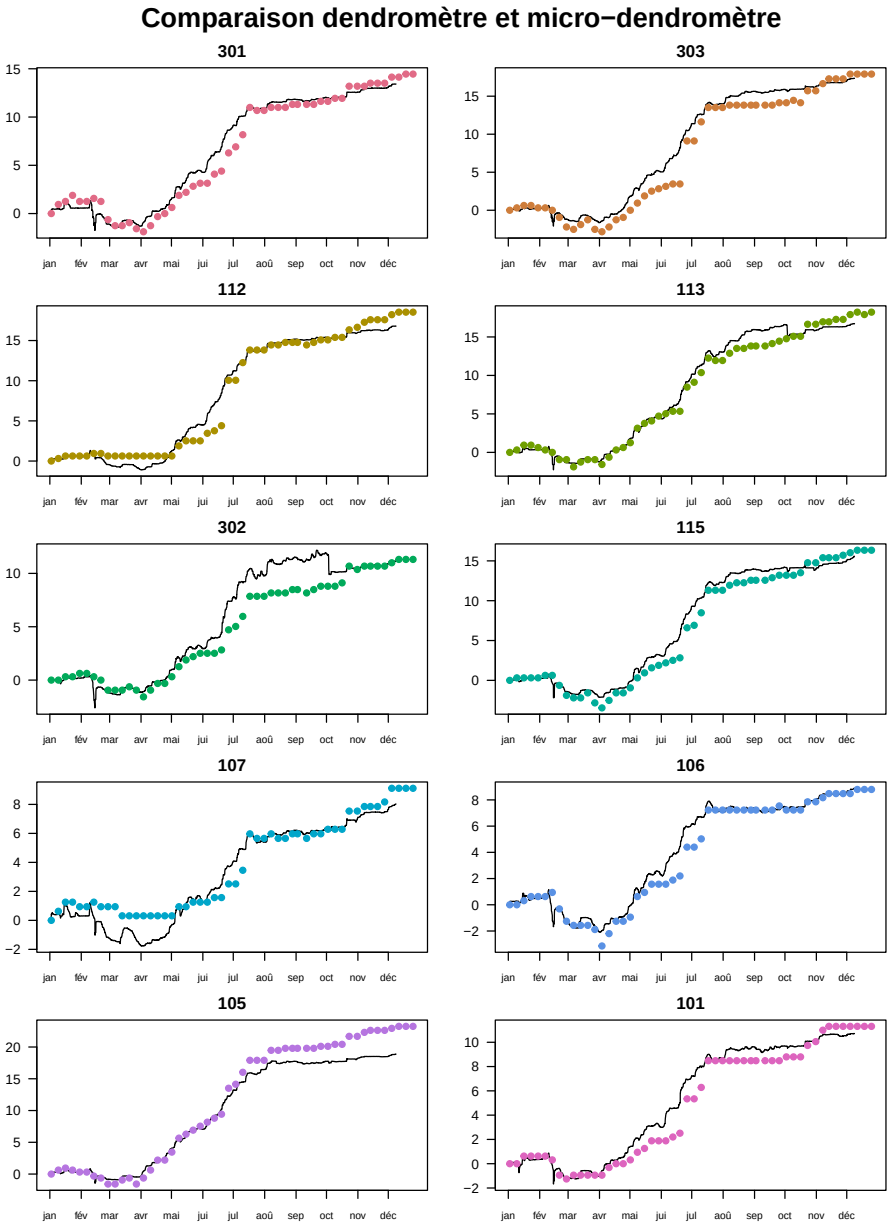
La résolution temporelle de notre série temporelle à basse résolution est d'une semaine mais on a une *précision* au jour près. Mieux, on sait que les relevés ont été faits en général le matin. Pour garder le maximum d'information, on va supposer que c'est à 10 heures UTC+2. La conversion du format `Date` au format `POSIXct` va se faire très simplement en précisant pour chaque jour l'heure et le fuseau horaire :

```
Pmm$temps <- as.POSIXct(paste(Pmm$Date, "10:00:00"), tz = "Etc/GMT-2")
range(Pmm$temps)
```

```
[1] "2021-01-02 10:00:00 +02" "2021-12-25 10:00:00 +02"
```

Maintenant que nous avons une échelle commune pour nos deux séries, il est facile de les représenter simultanément. Dans le graphique ci-après les points correspondent à la série temporelle à basse résolution (dendromètre classique) et les courbes à la série temporelle à haute résolution (micro-dendromètre).

```
compdir <- function(i){
  plot.new()
  plot.window(xlim = range(Pmm$temps), ylim = range(c(microd0[ , i], Pmm[, i])))
  axis(1, at = debmois, labels = month(debmois, label = TRUE), cex.axis = 0.75)
  axis(2, las = 1) ; box()
  title(main = nomsa[i])
  lines(microd0$temps, microd0[ , i], main = nomsa[i], type = "l")
  points(Pmm$temps, Pmm[, i], pch = 19, col = mycol[i])
}
par(mfrow = c(5, 2), mar = c(2, 2, 2, 2) + 0.1, oma = c(0, 0, 2, 0))
for(i in seq_len(10)) compdir(i)
title(main = "Comparaison dendromètre et micro-dendromètre", outer = TRUE,
      cex.main = 2)
```



Pour les dix chênes on constate qu'il y a globalement une bonne corrélation entre les mesures faites avec les deux types d'instruments. La phase de contraction du début de l'année n'est pas toujours observée avec le dendromètre classique (cas des arbres n^{os} 112 et 107). On peut émettre l'hypothèse d'une force de rappel insuffisante des ressorts de ces instruments. Pendant la phase de croissance les dendromètres classiques sont en général en retard par rapport au microdendromètres, comme s'ils avaient à lutter contre des forces de friction plus importantes. Les décrochages importants observés début octobre pour les arbres n^{os} 302 et 113 ne sont pas visibles avec les dendromètres classiques. Il n'y a que pour l'arbre n^{o} 105 que l'on observe une importante différence en fin d'année : avec le dendromètre classique on a une croissance annuelle de 23.2 mm

contre 18.9 mm avec le micro-dendromètre, soit un écart de 4.3 mm.

4.2.2 Échelle commune à basse résolution

Si les séries sont un multiple exact l'une de l'autre il assez simple de compresser la série à haute résolution dans l'échelle de celle à basse résolution. Dans notre cas de figure c'est un peu plus compliqué parce que les relevés ne sont pas exactement équidistants :

```
diff(Pmm$temps)
```

```
Time differences in days
 [1] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 9 7 7 7 7
[41] 7 7 8 7 6 7 7 7 7 7 7
```

Nous commençons par calculer les bornes des intervalles centrés sur les instants de la série à basse résolution. Ces bornes ne seront bien définies que depuis celle entre la première et la deuxième mesure jusqu'à celle entre l'avant-dernière et la dernière mesure. Pour les calculer on ajoute à chaque instant la moitié de l'écart à l'instant suivant :

```
bornes <- Pmm$temps[-nrow(Pmm)] + diff(Pmm$temps)/2
bornes[1:2]
```

```
[1] "2021-01-05 22:00:00 +02" "2021-01-12 22:00:00 +02"
```

```
Pmm$temps[1:3]
```

```
[1] "2021-01-02 10:00:00 +02" "2021-01-09 10:00:00 +02" "2021-01-16 10:00:00 +02"
```

```
diff(bornes)
```

```
Time differences in days
 [1] 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0
[21] 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0 6.0 7.0 8.0 7.0 7.0 7.0 7.0
[41] 7.0 7.5 7.5 6.5 6.5 7.0 7.0 7.0 7.0 7.0
```

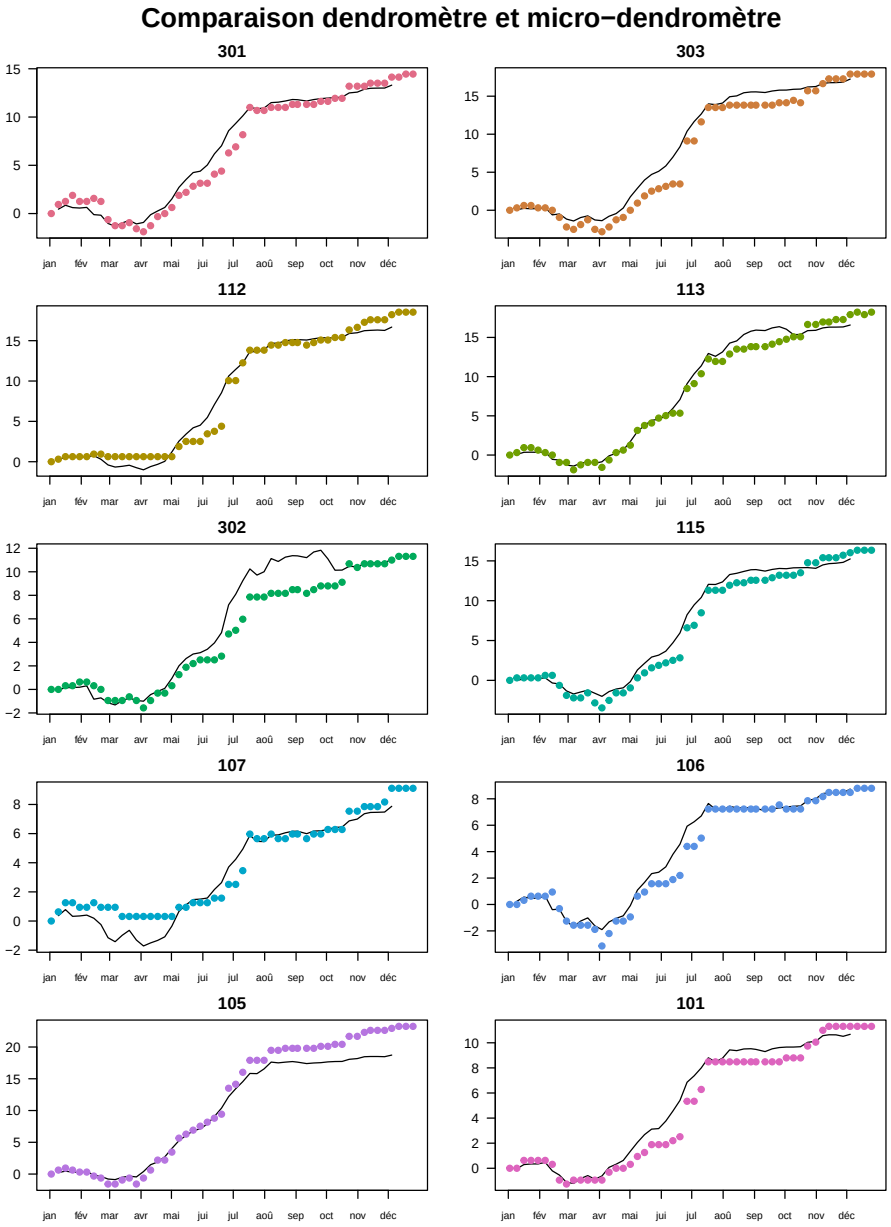
Le premier intervalle est centré sur le 9 janvier à 10h et court du 5 janvier 22h au 12 janvier 22h. À quelques exceptions près, il y a un écart de 7 jours entre la borne inférieure et supérieure des intervalles. La série à haute résolution est incomplète puisqu'elle s'arrête le 8 décembre à 15h. On ne conserve donc que les bornes des intervalles qui sont renseignées dans la série à haute résolution :

```
bornes <- bornes[bornes %in% microd0$temps]
```

Nous pouvons maintenant calculer nos séries compressées dans `microdC` en calculant pour chaque arbre la valeur moyenne observée pour chaque intervalle.

```
microdC <- as.data.frame(matrix(NA, nrow = length(bornes) - 1, ncol = 10))
colnames(microdC) <- nomsa
for(i in seq_len(nrow(microdC))){
  qui <- which(microd0$temps >= bornes[i] & microd0$temps < bornes[i + 1])
  microdC[i, 1:10] <- colMeans(microd0[qui, 1:10])
}
microdC$temps <- Pmm$temps[2:(nrow(microdC) + 1)]
```

En adaptant *mutatis mutandis* le code précédent on peut facilement comparer les séries temporelles. La représentation graphique est essentiellement la même sauf que la série à haute résolution est maintenant lissée :


Comparaison dendromètre et micro−dendromètre
301
303
112
113
302
115
107
106
105
101
jan fév mar avr mai jui jul aoû sep oct nov déc

5 Annexes

5.1 Remerciements

Un grand merci à Nicolas Delpierre, maître de conférences universitaire au laboratoire « Écologie Systématique Évolution » de l'université Paris-Saclay, et à Gilles Sinicco, technicien forestier territorial de l'ONF de l'unité territoriale du Saulnois (57590) dans le département de la Moselle, d'avoir accepté de partager leur données.

5.2 Importation au format `POSIXct`

On a illustré ici toute la puissance du format `POSIXct`, tout ceci est bel et bon, mais que fait-on si on ne dispose pas de ce format ? J'explique ici comment les données ont été importées dans R. Le point de départ[8] est un fichier de type tableur[9] ressemblant à ceci :

	A	B	C	D	S	T	U
1	DATE TIME	1. 05 Increment [mm]	2. 05 Temperature [oC]	3. 06 Increment [mm]	18. 13 Temperature [oC]	19. 95 Increment [mm]	20. 95 Temperature [oC]
2	14/08/2020 15:00	12,49446	25,98047	14,03896	25,125	13,68842	25,42578
3	14/08/2020 16:00	12,49252	25,09375	14,03121	25,22266	13,68358	25,83594
4	14/08/2020 17:00	12,49059	24,67188	14,02831	24,78125	13,68358	25,16797
5	14/08/2020 18:00	12,48962	23,22656	14,0254	23,34375	13,68551	23,55859
6	14/08/2020 19:00	12,48671	20,51563	14,02153	20,77734	13,68164	20,84375
7	14/08/2020 20:00	12,48865	19,29688	14,01765	19,10156	13,68358	18,96094
11542	08/12/2021 11:00	27,52306	4,167969	32,83052	4,207031	25,6561	4,160156
11543	08/12/2021 12:00	27,52209	4,621094	32,82858	4,628906	25,65513	4,609375
11544	08/12/2021 13:00	27,52112	5,351563	32,82955	5,332031	25,6532	5,355469
11545	08/12/2021 14:00	27,52112	5,464844	32,82858	5,273438	25,65417	5,328125
11546	08/12/2021 15:00	27,52112	5,035156	32,82858	4,839844	25,65417	4,835938

On dispose également d'un fichier[10] indiquant quels dendromètres sont affectés à quels arbres :

	A	B	C	D	E	F
1	Site_arbre	année	date	diametre_tronc	num_dendro_electronique	Ref dendro
2	CHS57A_101	2019	09/12/2019	48,12	20065295	95
3	CHS57A_105	2019	09/12/2019	59,45	20065313	13
4	CHS57A_106	2019	09/12/2019	55,28	20065312	12
5	CHS57A_107	2019	09/12/2019	46,94	20065311	11
6	CHS57A_112	2019	09/12/2019	52,47	20065307	07
7	CHS57A_113	2019	09/12/2019	58,74	20065308	08
8	CHS57A_115	2019	09/12/2019	61,16	20065310	10
9	CHS57A_301	2019	09/12/2019	57,5	20065305	05
10	CHS57A_302	2019	09/12/2019	51	20065309	09
11	CHS57A_303	2019	09/12/2019	58,27	20065306	06

J'ai commencé par l'enregistrer sous la forme d'un fichier texte[11] en utilisant des tabulations comme séparateur de colonnes et en ne conservant que les informations qui m'intéressaient, et en mettant le nom des arbres comme titre des colonnes :

[8] Les fichiers de départ sont à `https://pbil.univ-lyon1.fr/R/donnees/microdendroCHS57/`

[9] `CHS57_microdendro_Jun2020_Dec2021.xlsx`

[10] `corresp_dendros.xlsx`

[11] `microdendroCHS57.csv`

	A	B	C	D	K
1	DATE TIME	CHS57A_301	CHS57A_303	CHS57A_112	CHS57A_101
2	14/08/2020 15:00	12,49446	14,03896	11,89409	13,68842
3	14/08/2020 16:00	12,49252	14,03121	11,87182	13,68358
4	14/08/2020 17:00	12,49059	14,02831	11,86407	13,68358
5	14/08/2020 18:00	12,48962	14,0254	11,86117	13,68551
6	14/08/2020 19:00	12,48671	14,02153	11,86213	13,68164
7	14/08/2020 20:00	12,48865	14,01765	11,8631	13,68358
11542	08/12/2021 11:00	27,52306	32,83052	29,03173	25,6561
11543	08/12/2021 12:00	27,52209	32,82858	29,0327	25,65513
11544	08/12/2021 13:00	27,52112	32,82955	29,0327	25,6532
11545	08/12/2021 14:00	27,52112	32,82858	29,0327	25,65417
11546	08/12/2021 15:00	27,52112	32,82858	29,0327	25,65417

```
chmin <- "https://pbil.univ-lyon1.fr/R/donnees/microdendroCHS57/microdendroCHS57.csv"
microd <- read.table(chmin, sep = "\t", header = TRUE, dec = ",")
```

Les données temporelles sont dans une chaîne de caractères du type `14/08/2020 15:00:00`, on pourrait donc penser les convertir au format `POSIXct` avec simplement :

```
microd$tbad <- as.POSIXct(microd$DATE.TIME, format = "%d/%m/%Y %H:%M:%S")
```

Mais si on s'amuse à regarder les différences successives on voit qu'elles sont toutes d'une heure, ce qui est logique pour un suivi horaire, sauf dans deux cas où on a une différence de *deux* heures et dans deux cas où la différence n'est même pas définie ! Si on regarde de plus près on constate que les problèmes apparaissent au moment des passages entre l'heure d'été (`CEST`) et l'heure d'hiver (`CET`).

```
deltat <- diff(microd$tbad)
table(deltat, useNA = "always")
```

```
deltat
    1     2  <NA>
11540     2     2
```

```
ii <- which(deltat == 2 | is.na(deltat))
ii <- sort(unique((c(ii, ii + 1))))
for(i in ii) print(microd$DATE.TIME[i])
```

```
[1] "25/10/2020 02:00:00"
[1] "25/10/2020 03:00:00"
[1] "28/03/2021 01:00:00"
[1] "28/03/2021 02:00:00"
[1] "28/03/2021 03:00:00"
[1] "31/10/2021 02:00:00"
[1] "31/10/2021 03:00:00"
```

```
for(i in ii) print(microd$tbad[i])
```

```
[1] "2020-10-25 02:00:00 CEST"
[1] "2020-10-25 03:00:00 CET"
[1] "2021-03-28 01:00:00 CET"
[1] NA
[1] "2021-03-28 03:00:00 CEST"
[1] "2021-10-31 02:00:00 CEST"
[1] "2021-10-31 03:00:00 CET"
```

Le problème vient de ce que si on ne précise rien, l'importation se fera dans le fuseau horaire du système d'exploitation local, ici celui utilisé en France métropolitaine. Par exemple, dans la nuit du 27 au 28 mars 2021, le changement s'est fait dans le « mauvais » sens puisque nous avons perdu une heure de sommeil. À 2 heures du matin nous sommes passé sans transition à 3 heures du matin, 2 heures du matin n'a jamais eu d'existence légale, d'où le `NA` généré à l'importation (ce qui donnera deux `NA` lors du calcul des différences). À l'automne, dans les nuits du 24 au 25 octobre 2020 et du 30 au 31 octobre 2021, la

transition s'est faite dans le « bon » sens puisque nous avons gagné une heure de sommeil en retardant nos réveils d'une heure à 3 heures du matin. Ici par de problème à l'importation puisque toutes les heures ont eu une existence légale, mais il y a un écart de deux heures entre 2 heures du matin et 3 heures du matin.

Pour ne pas subir ces inconvénients nous allons importer les données en UTC+2, qui est le réglage par défaut en usine des dendromètres (ce qui correspond à l'heure d'été en France métropolitaine) en utilisant l'argument `tz` avec l'argument `Etc/GMT-2` pour signifier que nous sommes en *retard* de 2 heures par rapport à UTC.

```
microd$temps <- as.POSIXct(microd$DATE.TIME, tz = "Etc/GMT-2",
                           format = "%d/%m/%Y %H:%M:%S")
deltat <- diff(microd$temps)
table(deltat, useNA = "always")
```

```
deltat
    1  <NA>
11544     0
```

Il ne ne reste plus qu'à sauvegarder notre tableau de données `microd` au format XDR [11], qui est un format binaire compatible multi-plateformes. On enlève au préalable les colonnes qui ne nous servent plus. On ne conserve que les données de l'année 2021.

```
microd <- microd[ , c(2:11, 13)]
microd <- microd[year(microd$temps) == 2021, ]
rownames(microd) <- NULL
comment(microd) <- paste("Généré le :", Sys.time())
save(microd, file = "microdendroCHS57/microd.Rda")
```

Les micro-dendromètres sont également équipés d'un thermomètre donnant la température en degrés Celcius. On laissera au lecteur à titre d'exercice faire les manipulations analogues pour importer les données dans l'objet `meteo`.

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/microdendroCHS57/meteo.Rda"))
head(meteo[ ,c(1:5,11)])
```

```
  CHS57A_301 CHS57A_303 CHS57A_112 CHS57A_113 CHS57A_302              temps
1  0.9726563  0.8789063  0.8515625  0.9570313  0.7695313 2021-01-01 00:00:00
2  1.0000000  0.8789063  0.8671875  0.9609375  0.7929688 2021-01-01 01:00:00
3  0.8437500  0.8007813  0.7031250  0.8085938  0.6875000 2021-01-01 02:00:00
4  0.6328125  0.6250000  0.5351563  0.6406250  0.5039063 2021-01-01 03:00:00
5  0.9296875  0.8710938  0.8164063  0.9062500  0.7929688 2021-01-01 04:00:00
6  0.8320313  0.7148438  0.6953125  0.7500000  0.6289063 2021-01-01 05:00:00
```

Références

[1] M.A. Barber. The rate of multiplication of *Bacillus coli* at different temperatures. *Journal of Infectious diseases*, 5 :379–400, 1908.

[2] J. Čermák, J. Ulehla, J. Kučera, and M. Penka. Sap flow rate and transpiration dynamics in the full-grown oak (*Quercus robus* L.) in floodplain forest exposed to seasonal floods as related to potential evapotranspiration and tree dimensions. *Biologia Plantarum*, 24(6) :446–460, 1982.

[3] J. Friedrich. Über den Einfluss der Witterung auf den Baumzuwachs. *Mitteilungen Forstlichen Bundes-Versuchsanstalt Wien*, 22 :1–160, 1897.

[4] G. Grolemund and H. Wickham. Dates and times made easy with lubridate. *Journal of Statistical Software*, 40(3) :1–25, 2011.

[5] G. Grothendieck and T. Petzoldt. Date and time classes in R. *R News*, 4(1) :29–31, 2004.

[6] F. Lebourgeois, J. Differt, A. Granier, N. Bréda, and E. Ulrich. Premières observations phénologiques des peuplements du réseau national de suivi à long terme des écosystèmes forestiers (RENECOFOR). *Revue forestière française*, 54(5) :407–418, 2002.

[7] R. Marsham. Observations on the growth of trees. *Philosophical Transactions of the Royal Society of London*, (51) :7–12, 1759.

[8] D.A. Ratkowsky and G.V.P. Reddy. Empirical model with excellent statistical properties for describing temperature-dependent developmental rates of insects and mites. *Annals of the Entomological Society of America*, 110 :302–309, 2017.

[9] B.D. Ripley and K. Hornik. Date-time classes. *R News*, 1(2) :8–11, 2001.

[10] L. Rosso, J.R. Lobry, and J.-P. Flandrois. An unexpected correlation between cardinal temperatures of microbial growth highlighted by a new model. *Journal of Theoretical Biology*, 162(4) :447–463, 1993.

[11] Sun Microsystems. XDR : external data representation standard. RFC 1014. Technical report, Network Working Group, 1987.

[12] P.-F. Verhulst. Notice sur la loi que la population suit dans son accroissement. *Correspondance mathématique et physique*, 10 :113–121, 1838.

[13] P.-F. Verhulst. Recherches mathématiques sur la loi d'accroissement de la population. *Nouveaux mémoires de l'académie royale des sciences, des lettres et des beaux-arts de Belgique*, 18 :1–38, 1845. Lu à la séance du 30 novembre 1844.

[14] P.-F. Verhulst. Note sur la loi d'accroissement de la population. *Bulletins de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles*, 13, 1846.

[15] P.-F. Verhulst. Deuxième mémoire sur la loi d'accroissement de la population. *Mémoires de l'académie royale des sciences, des lettres et des beaux-arts de Belgique*, 20 :1–32, 1847. Lu à la séance de l'Académie royale du 15 mai 1846.