Fiche TD avec le logiciel R : tdr411

Sélection de modèles pour la création d'un thermomètre moléculaire protéique

J.R. Lobry

Illustration des méthodes de sélection de modèle en régréssion linéaire pour construire un modèle prédisant au mieux la température de croissance des procaryotes à partir de la composition en acide-aminés de leur protéome. Le critère AIC et le critère BIC.

Table des matières

1 Introduction

La fréquence des acides-aminés dans les protéines est connue pour être influencée par la température à laquelle vivent les procaryotes (T_{opt}, la température optimale de croissance, voir la fiche de TD tdr46 [1] pour les autres températures cardinales). On se pose la question suivante : comment à partir de la composition en acides-aminés prédire au mieux T_{opt} ? Nous avons donc 20 variables explicatives (les 20 fréquences en acides-aminés) et une variable à prédire, T_{opt}. Notez qu'il n'y a en fait que 19 variables prédictives indépendantes puisque leur somme est constante par construction.

2 Les données

2.1 Importation des données

Les données sont extraites de [12]. Les importer dans R puis les sauvegarder dans votre dossier de travail courant avec :

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/tdr411/afcinin.RData"))
save(afcinin, file = "afcinin.RData")
names(afcinin)
```

```
[1] "count"  "topt"   "domain" "gc"     "aa"
```

2.2 Retrait des outliers

Deux espèces, *Eubacterium acidaminophilum* et *Cenarchaeum symbiosum*, sont connues pour avoir un comportement particulier pour la composition de leur protéome [11, 12]. On les retire de l'analyse.

```
out1 <- which(rownames(afcinin$count) == toupper("Eubacterium acidaminophilum"))
out2 <- which(rownames(afcinin$count) == toupper("Cenarchaeum symbiosum"))
outs <- c(out1, out2)
afcinin$count <- afcinin$count[-outs, ]
afcinin$topt <- afcinin$topt[-outs, ]
afcinin$domain <- afcinin$domain[-outs]
afcinin$gc <- afcinin$gc[-outs]
```

Les psychrophiles sont trop peu documentés dans ce jeu de données, on décide de les retirer.

```
outpsy <- which(afcinin$topt$typephile == "psychrophile")
afcinin$count <- afcinin$count[-outpsy, ]
afcinin$topt <- afcinin$topt[-outpsy, ]
afcinin$topt$typephile <- factor(afcinin$topt$typephile)
afcinin$domain <- afcinin$domain[-outpsy]
afcinin$gc <- afcinin$gc[-outpsy]
```

1. Régression non-linéaire : http://pbil.univ-lyon1.fr/R/fichestd/tdr46.pdf.

2.3 Calcul des fréquences des acides aminés

```
head(afcinin$count[,1:5])
```

```
                                aaa  aac  aag  aat  aca
ACHROMOBACTER DENITRIFICANS     216  417  691  149  134
ACHROMOBACTER XYLOSOXIDANS      349  807 1225  283  169
ACIDIANUS AMBIVALENS            756  330  625  519  301
ACIDITHIOBACILLUS FERROOXIDANS  732  897 1252  662  270
ACINETOBACTER BAUMANNII        3442 1233 1429 2590 1271
ACINETOBACTER CALCOACETICUS    1449  441  435 1033  509
```

```
afcinin$aa
```

```
 [1] Lys Asn Lys Asn Thr Thr Thr Thr Arg Ser Arg Ser Ile Ile Met Ile Gln His Gln His
[21] Pro Pro Pro Pro Arg Arg Arg Arg Leu Leu Leu Leu Glu Asp Glu Asp Ala Ala Ala Ala
[41] Gly Gly Gly Gly Val Val Val Val Tyr Tyr Ser Ser Ser Ser Cys Trp Cys Leu Phe Leu
[61] Phe
20 Levels: Ala Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser ... Val
```

L'objet `afcinin$count` nous donne les fréquences absolues des 61 codons codants [2]. L'objet `afcinin$aa` nous donne les acides-aminés correspondants. On veut faire la somme des colonnes ayant la même modalité de la variable qualitative (on fait donc les sommes en ligne).

```
aa <- afcinin$aa
cnt <- afcinin$count
fraaa <- sapply(levels(aa), function(x) rowSums(cnt[,x==aa, drop=FALSE]))
head(fraaa[,1:5])
```

```
                                Ala  Arg  Asn  Asp Cys
ACHROMOBACTER DENITRIFICANS    2783 1601  566 1204 206
ACHROMOBACTER XYLOSOXIDANS     5031 2828 1090 2062 330
ACIDIANUS AMBIVALENS           1297  700  849  853 218
ACIDITHIOBACILLUS FERROOXIDANS 5876 3794 1559 2705 623
ACINETOBACTER BAUMANNII        7428 4257 3823 4476 745
ACINETOBACTER CALCOACETICUS    2973 1599 1474 1903 413
```

On veut des fréquences relatives exprimées en pourcentage :

```
frlaa <- 100*fraaa/rowSums(fraaa)
head(frlaa[,1:5])
```

```
                                    Ala      Arg      Asn      Asp       Cys
ACHROMOBACTER DENITRIFICANS    12.503931 7.193243 2.543020 5.409534 0.9255515
ACHROMOBACTER XYLOSOXIDANS     12.743484 7.163302 2.760962 5.223030 0.8358874
ACIDIANUS AMBIVALENS            6.752395 3.644315 4.420033 4.440858 1.1349438
ACIDITHIOBACILLUS FERROOXIDANS 10.967188 7.081265 2.909776 5.048714 1.1627907
ACINETOBACTER BAUMANNII         8.458306 4.847470 4.353272 5.096847 0.8483358
ACINETOBACTER CALCOACETICUS     8.703926 4.681324 4.315367 5.571332 1.2091226
```

3 Analyse univariée

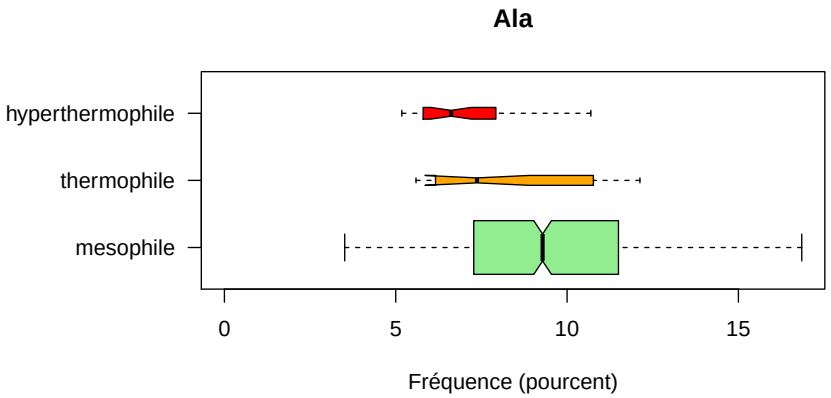
On définit une petite fonction utilitaire pour examiner rapidement les acides-aminés un par un :

```
colmes <- "palegreen2"
colther <- "orange"
colhyp <- "red"
colphil <- c(colmes,colther,colhyp)
oneaa <- function(aa){
        opar <- par(no.readonly = TRUE)
        on.exit(par(opar))
        par(mar = par("mar") + c(0,5,0,0))
        y <- frlaa[, which(colnames(frlaa)==aa)]
        x <- afcinin$topt$typephile
  boxplot(y~x, ylim = c(0,max(frlaa)),
  horizontal=T,varwidth=T, main = aa,
  las = 1, col = colphil, notch = T,
  xlab = "Fréquence (pourcent)")
}
```

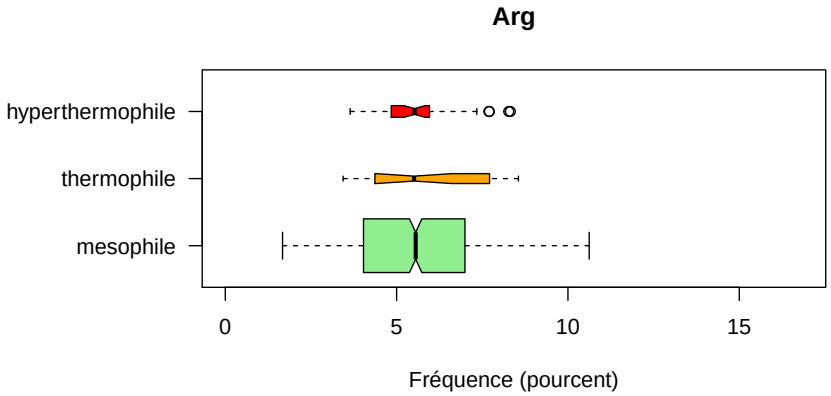
2. Les espèces de ce jeu de données utilisent toutes le code génétique standard.

Il y a moins ($\Delta \approx -3$ %) d'alanine chez les thermophiles :

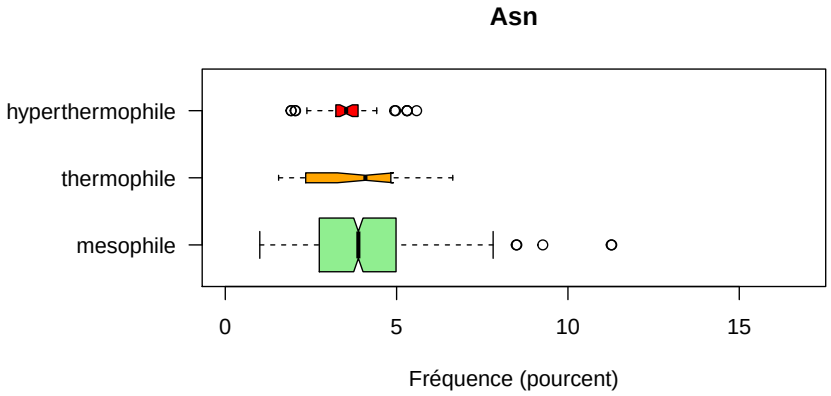
```
oneaa("Ala")
```



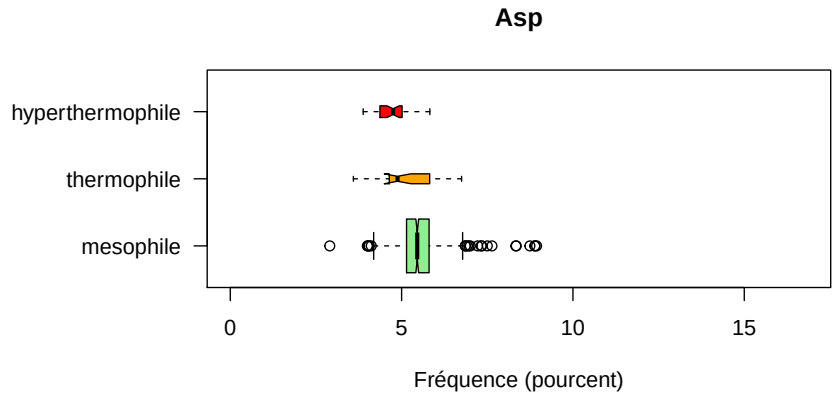
L'arginine est peu affectée :



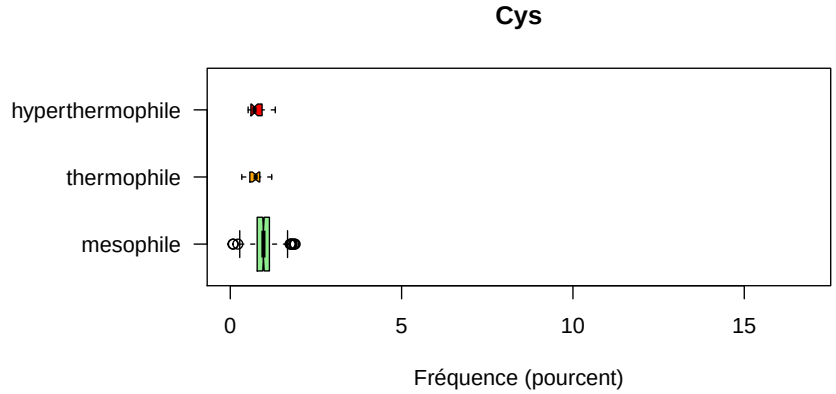
L'asparagine est peu affectée :



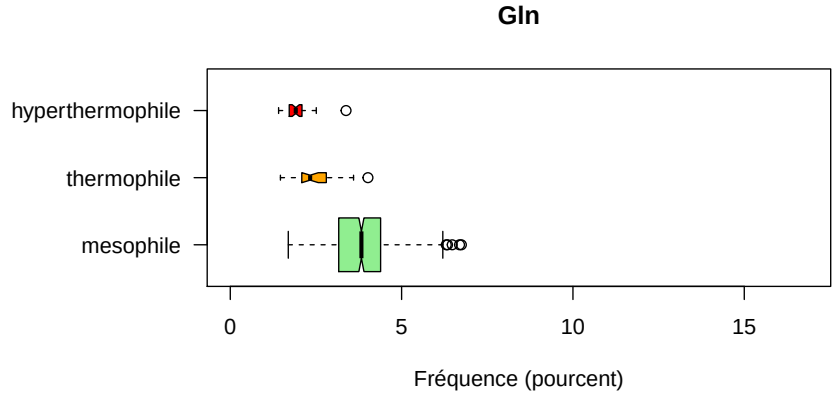
Il y a moins ($\Delta \approx -1$ %) d'aspartate chez les thermophiles :



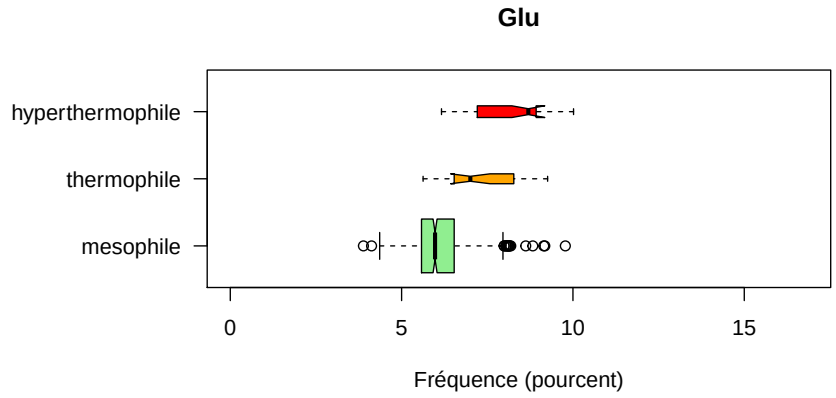
La cystéine est peu affectée :



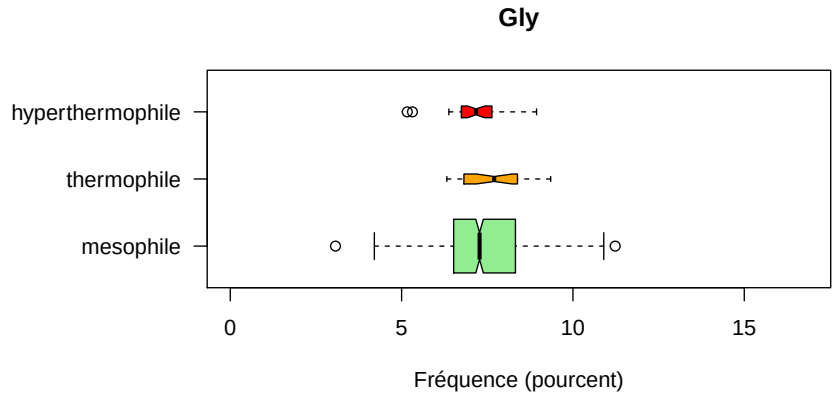
Il y a moins ($\Delta \approx -3$ %) de glutamine chez les thermophiles :



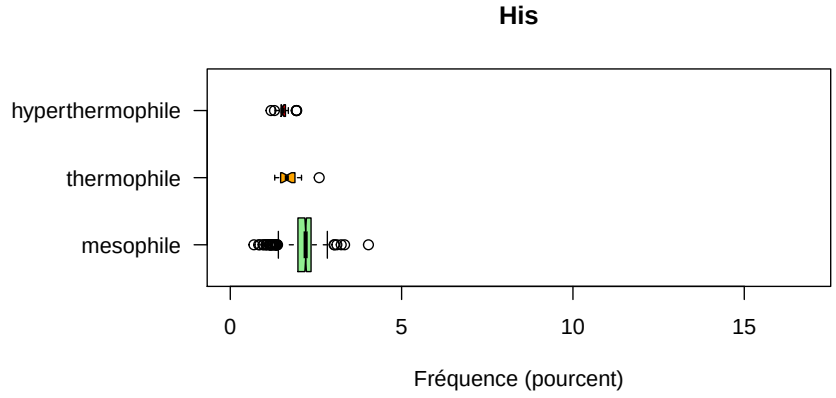
Il y a plus ($\Delta \approx +3$ %) de glutamate chez les thermophiles :



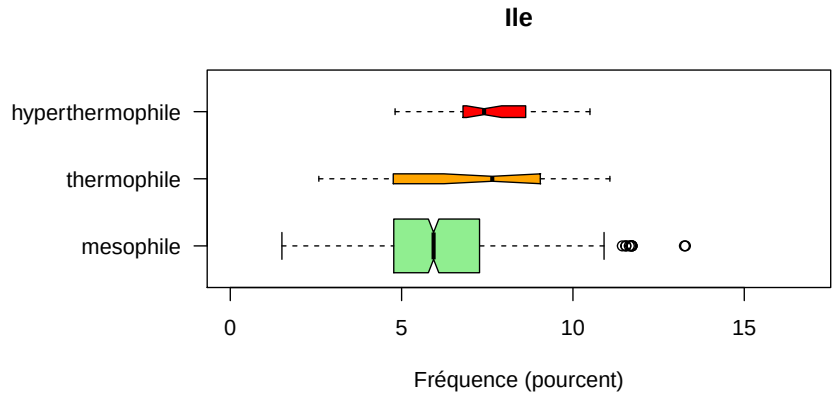
La glycine est peu affectée :



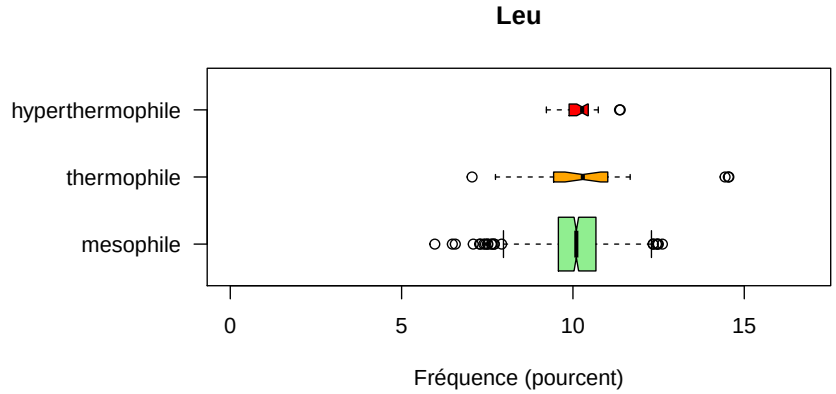
Il y a moins ($\Delta \approx -1$ %) d'histidine chez les thermophiles :



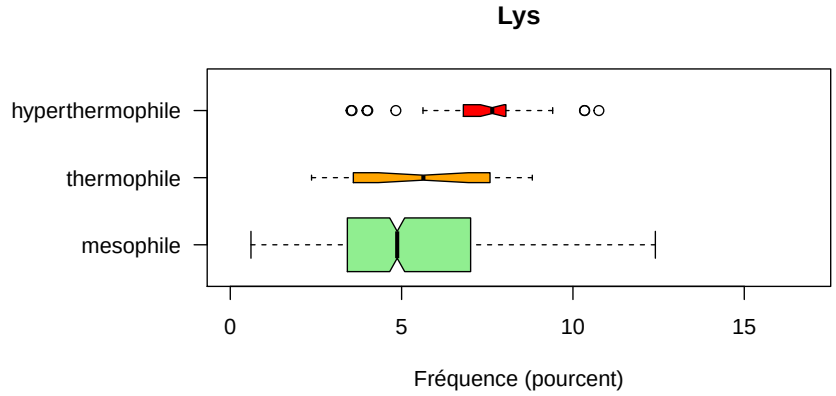
Il y a plus ($\Delta \approx +2$ %) d'isoleucine chez les thermophiles :



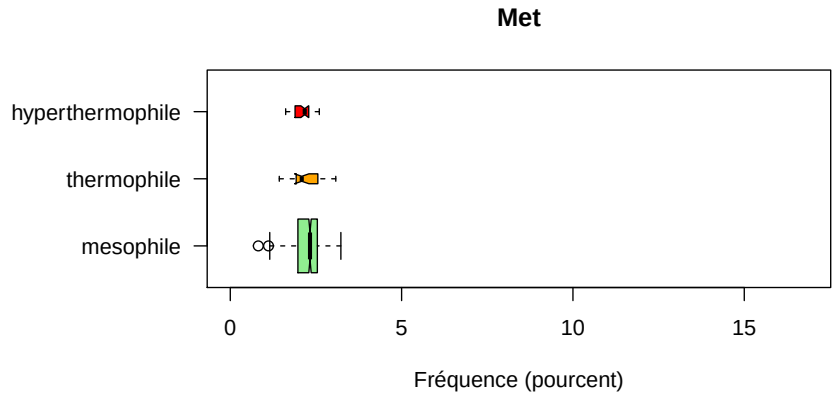
La leucine est peu affectée :



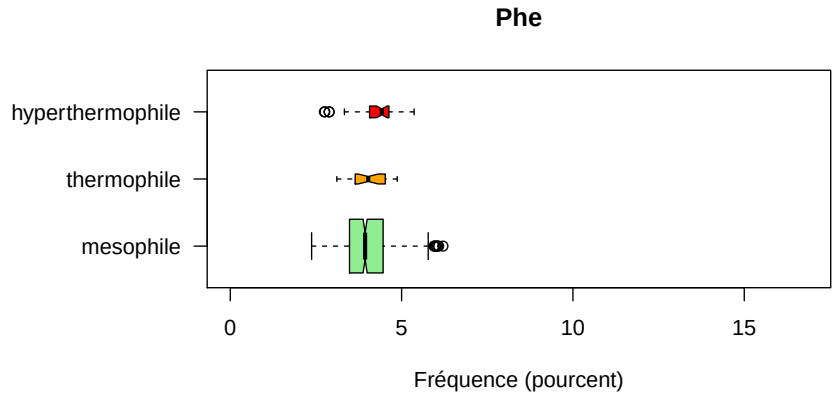
Il y a plus ($\Delta \approx +3$ %) de lysine chez les thermophiles :



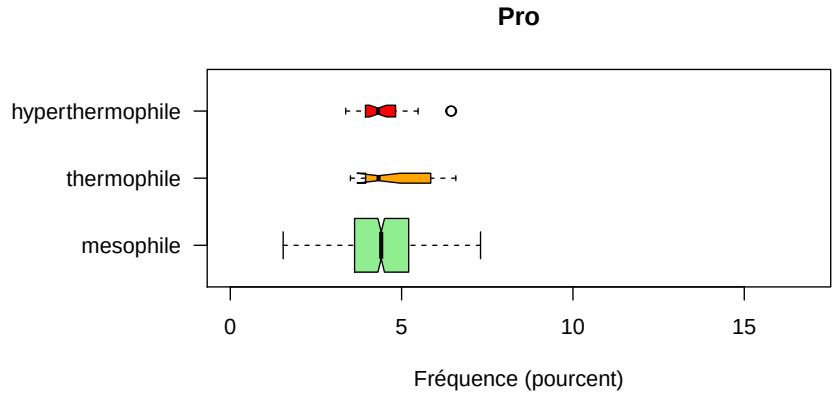
La méthionine est peu affectée :



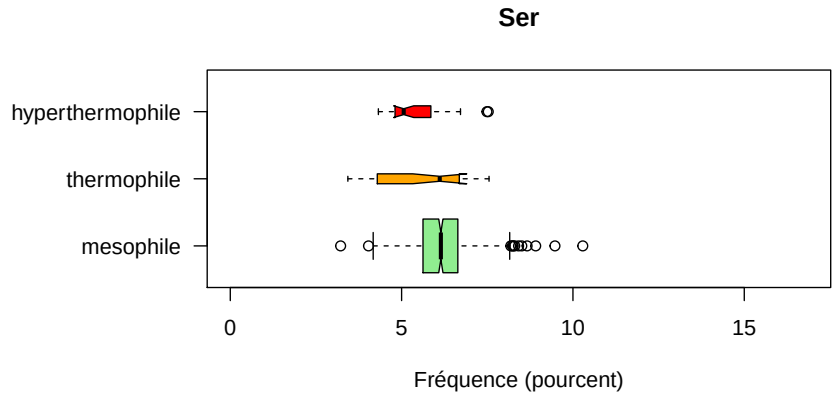
La phénylalanine est peu affectée :



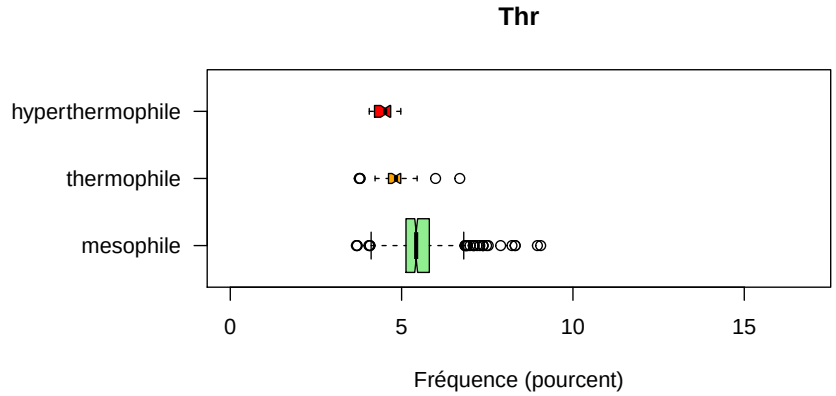
La proline est peu affectée :



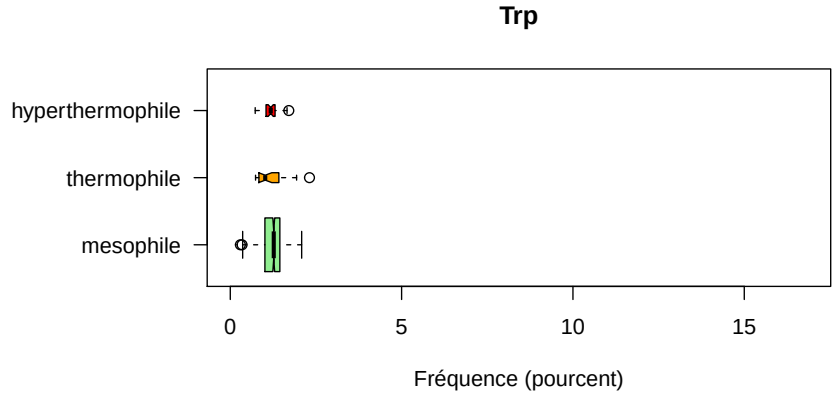
Il y a moins ($\Delta \approx -1$ %) de sérine chez les thermophiles :



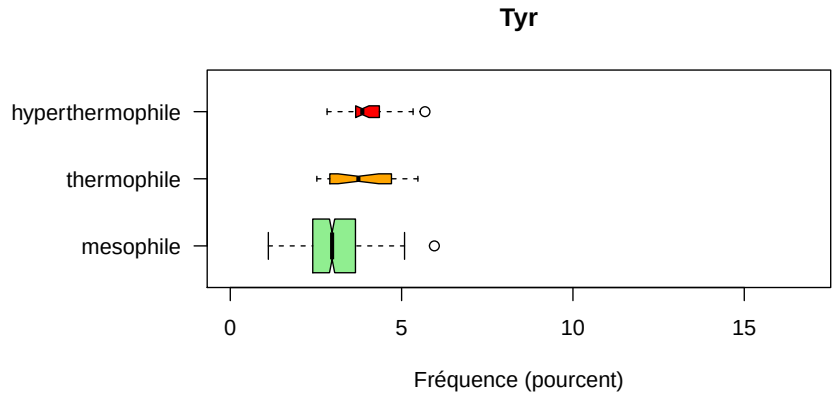
Il y a moins ($\Delta \approx -1$ %) de thréonine chez les thermophiles :



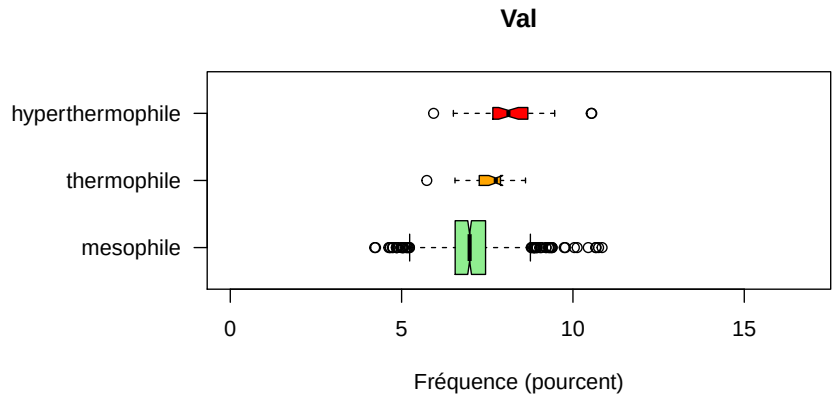
Le tryptophane est peu affecté :



Il y a plus ($\Delta \approx +1$ %) de tyrosine chez les thermophiles :



Il y a plus ($\Delta \approx +2$ %) de valine chez les thermophiles :



4 Modèle linéaire

5 Choix des variables explicatives

Il n'y a que 19 variables indépendantes, il faut en retirer une. On choisit de ne pas prendre le tryptophane parce qu'il est peu fréquent et peu influencé par la température.

```
data <- as.data.frame(frlaa[,-18])
topt <- afcinin$topt$topt
```

5.1 Modèle linéaire complet sans interactions

```
lm(topt~.,data)->lm1
summary(lm1)
```

```
Call:
lm(formula = topt ~ ., data = data)
Residuals:
    Min      1Q  Median      3Q     Max
```

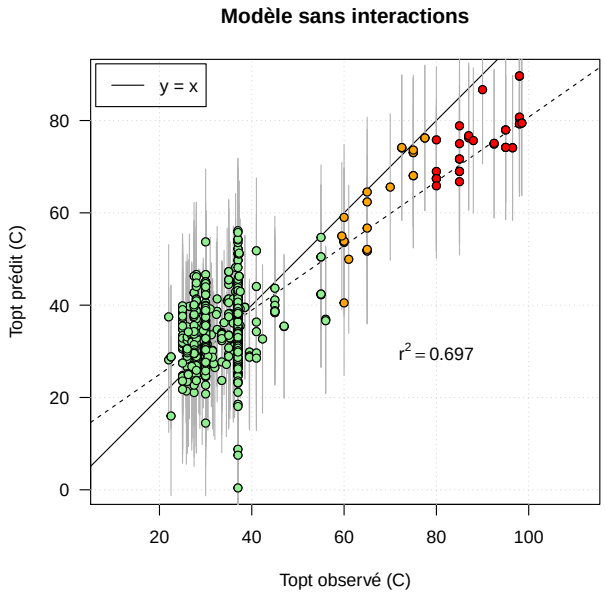
```
-23.713  -4.643  -0.222   4.848  36.594

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  458.411    161.812   2.833 0.004742 **
Ala           -5.413      1.586  -3.414 0.000678 ***
Arg           -2.845      1.781  -1.598 0.110556
Asn           -8.094      1.785  -4.535 6.75e-06 ***
Asp           -7.209      1.730  -4.166 3.48e-05 ***
Cys           -6.369      2.141  -2.975 0.003034 **
Gln           -6.182      1.791  -3.451 0.000592 ***
Glu           -3.256      1.620  -2.010 0.044758 *
Gly           -5.321      1.851  -2.874 0.004178 **
His          -11.670      1.919  -6.080 1.96e-09 ***
Ile           -2.990      1.618  -1.848 0.064987 .
Leu           -2.672      1.709  -1.563 0.118454
Lys           -2.528      1.607  -1.573 0.116144
Met           -6.210      2.035  -3.051 0.002365 **
Phe          -10.417      1.905  -5.468 6.31e-08 ***
Pro           -3.522      2.068  -1.703 0.089016 .
Ser           -5.420      1.630  -3.325 0.000929 ***
Thr           -6.671      1.813  -3.679 0.000252 ***
Tyr            6.572      1.981   3.318 0.000952 ***
Val            1.179      1.683   0.700 0.484020
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.884 on 710 degrees of freedom
Multiple R-squared:  0.697,        Adjusted R-squared:  0.6889
F-statistic: 85.98 on 19 and 710 DF,  p-value: < 2.2e-16
```

Le modèle a tendance à sous-estimer pour les thermophiles :

```
perfplot <- function(lm0, main){
        x <- topt
        y <- predict(lm0)
  plot(x, y,
    xlab = "Topt observé (C)",
    ylab = "Topt prédit (C)",
    las = 1,
    main = main,
    asp =1)
  grid()
  abline(c(0,1))
  abline(lm(y~x), lty = 2)
  r2 <- round(cor(x,y)^2, 3)
  text(80, 30, bquote(r^2 == .(r2)))
  legend("topleft", inset = 0.01, legend = "y = x", lwd = 1)
  suppressWarnings(cint <- predict(lm0,interval="predict"))
  segments(topt,cint[,"lwr"],topt,cint[,"upr"],
    col = grey(0.7))
  hyp <- afcinin$topt$typephile == "hyperthermophile"
  ther <- afcinin$topt$typephile == "thermophile"
  mes <- afcinin$topt$typephile == "mesophile"
  points(x[hyp],y[hyp],bg=colhyp, pch = 21)
  points(x[ther],y[ther],bg=colther, pch = 21)
  points(x[mes],y[mes],bg=colmes, pch = 21)
}
perfplot(lm1, main = "Modèle sans interactions")
```




```
lm1$coefficients
```

```
(Intercept)         Ala         Arg         Asn         Asp         Cys         Gln
 458.410573   -5.412725   -2.844889   -8.094452   -7.209022   -6.369386   -6.181949
        Glu         Gly         His         Ile         Leu         Lys         Met
  -3.256064   -5.320609  -11.669755   -2.990021   -2.672202   -2.527969   -6.210023
        Phe         Pro         Ser         Thr         Tyr         Val
 -10.417021   -3.522326   -5.420426   -6.670925    6.572472    1.178605
```

Si on force par l'origine pour avoir une simple forme linéaire sur les fréquences d'acides aminés :

```
lm(topt~.-1,data)->lm11
summary(lm11)
```

```
Call:
lm(formula = topt ~ . - 1, data = data)
Residuals:
    Min      1Q  Median      3Q     Max
-22.873  -4.821  -0.266   4.809  36.736

Coefficients:
    Estimate Std. Error t value Pr(>|t|)
Ala  -1.0501     0.3798  -2.765 0.005847 **
Arg   1.9589     0.5461   3.587 0.000357 ***
Asn  -3.4012     0.6673  -5.097 4.43e-07 ***
Asp  -2.7234     0.7015  -3.882 0.000113 ***
Cys  -1.5885     1.3246  -1.199 0.230819
Gln  -1.3285     0.5260  -2.526 0.011763 *
Glu   1.1059     0.5047   2.191 0.028766 *
Gly  -0.4104     0.6542  -0.627 0.530656
His  -7.3518     1.1724  -6.271 6.24e-10 ***
Ile   1.4098     0.4552   3.097 0.002031 **
Leu   1.9977     0.4550   4.390 1.30e-05 ***
Lys   1.8371     0.4587   4.005 6.85e-05 ***
Met  -1.2027     1.0143  -1.186 0.236105
Phe  -5.6081     0.8691  -6.453 2.03e-10 ***
Pro   1.8344     0.8425   2.177 0.029785 *
Ser  -0.9972     0.4709  -2.118 0.034558 *
Thr  -1.8639     0.6428  -2.900 0.003848 **
Tyr  11.6760     0.8273  14.114  < 2e-16 ***
Val   5.6479     0.5897   9.578  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.922 on 711 degrees of freedom
Multiple R-squared:   0.96,        Adjusted R-squared:  0.959
F-statistic: 898.9 on 19 and 711 DF,  p-value: < 2.2e-16
```

```
lm11$coef
```

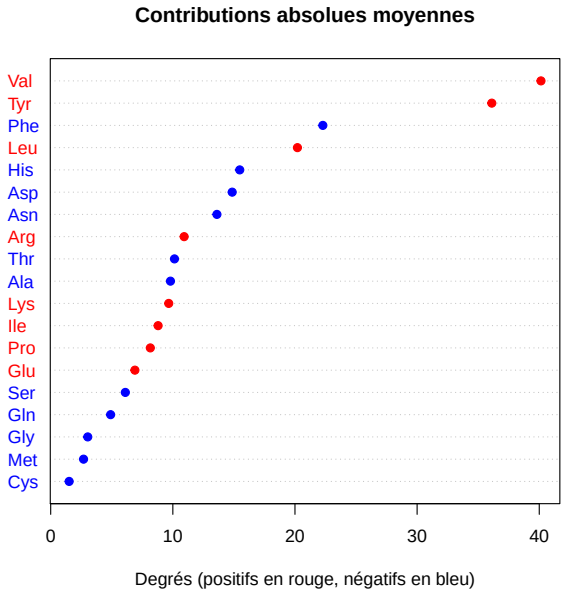
```
       Ala        Arg        Asn        Asp        Cys        Gln        Glu
-1.0500590  1.9589462 -3.4012169 -2.7234339 -1.5885225 -1.3285448  1.1058856
       Gly        His        Ile        Leu        Lys        Met        Phe
-0.4103932 -7.3518194  1.4098175  1.9976712  1.8371330 -1.2026915 -5.6081021
       Pro        Ser        Thr        Tyr        Val
 1.8343999 -0.9971857 -1.8639056 11.6759569  5.6479109
```

C'est beaucoup plus direct à lire ainsi : le signe des coefficients indique directement si c'est un acide-aminé dont la fréquence diminue (*e.g.* Ala) ou augmente (*e.g.* Arg) avec la température. La lecture est assez directe, par exemple chaque % de lysine en plus fait gagner 11.8 degrés. Pour avoir une idée des contributions de chaque acide-aminés il suffit de multiplier les coefficients par les fréquences moyennes :

```
(contribs <- lm11$coef*colMeans(data))
```

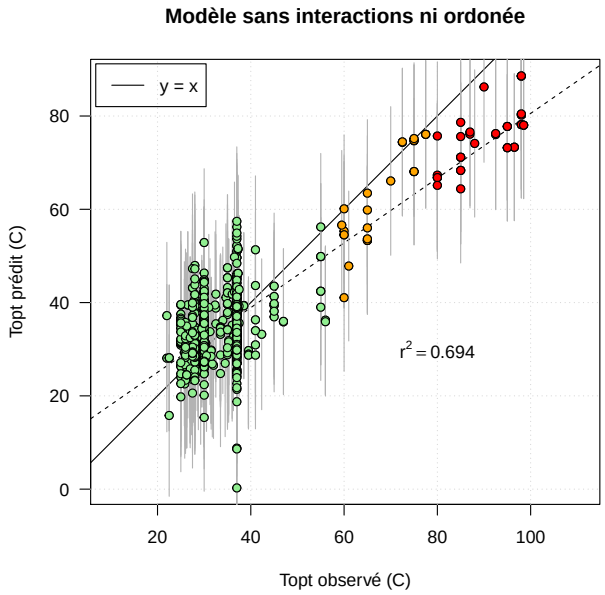
```
       Ala        Arg        Asn        Asp        Cys        Gln        Glu
-9.806351  10.925714 -13.609385 -14.859825  -1.511186  -4.906857   6.895349
       Gly        His        Ile        Leu        Lys        Met        Phe
-3.033894 -15.477755   8.794482  20.201625   9.663411  -2.691441 -22.281114
       Pro        Ser        Thr        Tyr        Val
 8.157108  -6.111179 -10.135332  36.113964  40.141039
```

```
contrabs <- abs(contribs)
dotchart(contrabs[order(contrabs)],
col = ifelse(sign(contribs[order(contrabs)])==1,"red","blue"),pch=19,
main = "Contributions absolues moyennes",
xlab = "Degrés (positifs en rouge, négatifs en bleu)")
```



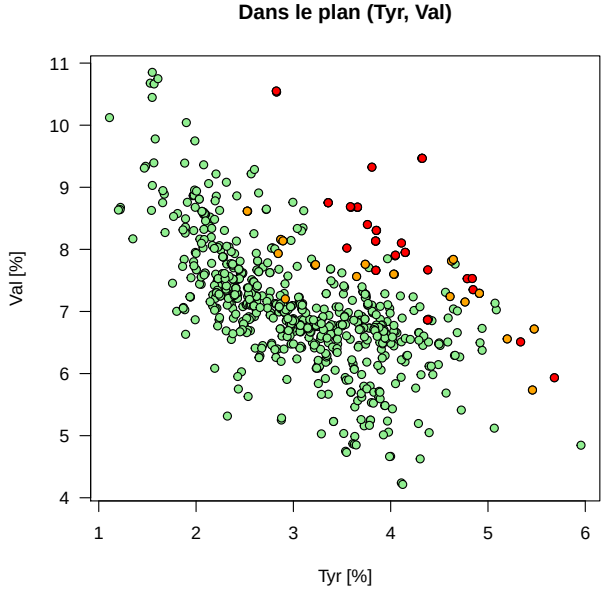
D'un point de vue purement prédictif cela ne change pas grand chose avec le modèle avec ordonnée libre :

```
perfplot(lm11, main = "Modèle sans interactions ni ordonée")
```



On jette un oeil dans le plan tyrosine valine pour voir :

```
x <- data$Tyr
y <- data$Val
plot(x, y, pch = 21, las = 1,
xlab = "Tyr [%]", ylab = "Val [%]",
main = "Dans le plan (Tyr, Val)",
bg = colmes)
hyp <- afcinin$topt$typephile == "hyperthermophile"
ther <- afcinin$topt$typephile == "thermophile"
points(x[hyp],y[hyp],bg=colhyp, pch = 21)
points(x[ther],y[ther],bg=colther, pch = 21)
```



On décide de conserver un modèle sans ordonnée à l'origine pour faciliter l'interprétation des coefficients. Voyons maintenant si on peut le simplifier un peu sans pertes de performance. On peut procéder par élimination descendante ou sélection ascendante des prédicteurs.

5.1.1 Elimination descendante des prédicteurs

L'algorithme est le suivant :

1. On part du modèle complet avec tous les prédicteurs.
2. On enlève le predicteur avec la plus grande p-value supérieure à un seuil donné α_{crit}. On prendra ici par exemple $\alpha_{crit} < 10^{-4}$.
3. Réajuster le modèle et retourner à l'étape 2.
4. On arrête lorsque toutes les p-values sont inférieures à α_{crit}.

L'avantage dans R c'est que l'on peut programmer cet algorithme pour ne pas avoir à faire les opérations à la main :

```
selback <- function(lm0, data, alphac = 0.0001, verbose = FALSE){
        lmsb <- lm0
        coef <- summary(lmsb)$coef
        while(max(coef[,4]) > alphac){
          imax <- which.max(coef[,4])
          quimax <- rownames(coef)[imax]
          if(verbose) print (paste("Deleting", quimax))
          old <- formula(lmsb)
          new <- update.formula(old, as.formula(paste(". ~ . -", quimax )))
          lmsb <- lm(new, data)
          coef <- summary(lmsb)$coef
        }
        return(lmsb)
}
lm11sb <- selback(lm11, data, verbose = TRUE)
```

```
[1] "Deleting Gly"
[1] "Deleting Cys"
[1] "Deleting Met"
[1] "Deleting Glu"
[1] "Deleting Ile"
[1] "Deleting Ser"
[1] "Deleting Pro"
[1] "Deleting Thr"
```

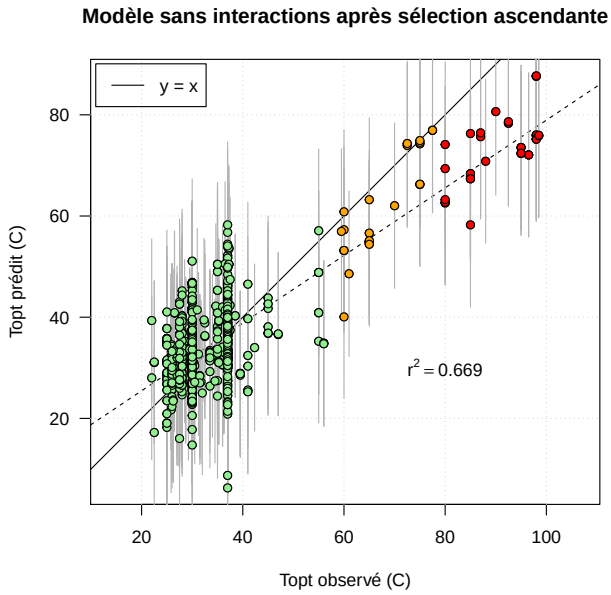
On arrive à un modèle plus économique avec 11 prédicteurs au lieu de 19 avec le modèle initial :

```
lm11sb$coef
```

```
      Ala       Arg       Asn       Asp       Gln       His       Leu       Lys
-1.405383  2.522879 -4.422590 -3.291889 -2.253405 -8.408416  2.837760  2.767280
      Phe       Tyr       Val
-5.833455 12.492138  5.415776
```

Sans grosse perte de performances :

```
perfplot(lm11sb,"Modèle sans interactions après élimination descendante")
```



5.1.2 Sélection ascendante des prédicteurs

L'algorithme est le suivant :

1. On part d'un modèle sans aucun prédicteur.
2. Pour chaque prédicteur du modèle, regarder sa p-value s'il est ajouté au modèle. Sélectioner celui avec la plus petite p-value inférieure à α_{crit}.
3. Continuer jusqu'à ce que l'on ne puisse plus ajouter de nouveau prédicteur.

Ce qui nous donne :

```
selfor <- function(lm0, data, alphac = 0.0001, verbose = FALSE){
        lmsf <- lm(topt ~ -1, data) # empty model
        coef <- summary(lm0)$coef
        ncoef <- nrow(coef)
        isadded <- logical(ncoef) # FALSE
        names(isadded) <- rownames(coef)
        continuer <- TRUE
        while(continuer){
                npos <- sum(!isadded)
                if(npos == 0) return(lmsf)
                newpval <- numeric(ncoef) # default 0
                names(newpval) <- rownames(coef)
                for(i in seq_len(ncoef)){
                  if(isadded[i]) next
                  qui <- names(newpval)[i]
                  tmp <- update.formula(formula(lmsf), as.formula(paste(". ~ . +", qui)))
                  tmpcoef <- summary(lm(tmp,data))$coef
                  ou <- which(rownames(tmpcoef) == qui)
                  newpval[i] <-tmpcoef[ou,4]
                }
                tocheck <- newpval[!isadded]
                if(min(tocheck) < alphac){
                  continuer <- TRUE
                  imin <- which.min(tocheck)
                  qui <- names(tocheck[imin])
                  if(verbose) print(paste("Adding", qui))
                  new <- update.formula(formula(lmsf),
                         as.formula(paste(". ~ . +", qui )))
```

```
                lmsf <- lm(new, data)
                isadded[qui] <- TRUE
              } else {
                continuer <- FALSE
              }

        }
        return(lmsf)
}
lm11sf <- selfor(lm11, data, verbose = TRUE)
```

```
[1] "Adding Glu"
[1] "Adding Gln"
[1] "Adding Tyr"
[1] "Adding Asn"
[1] "Adding Leu"
[1] "Adding His"
[1] "Adding Phe"
[1] "Adding Ile"
[1] "Adding Val"
[1] "Adding Asp"
```

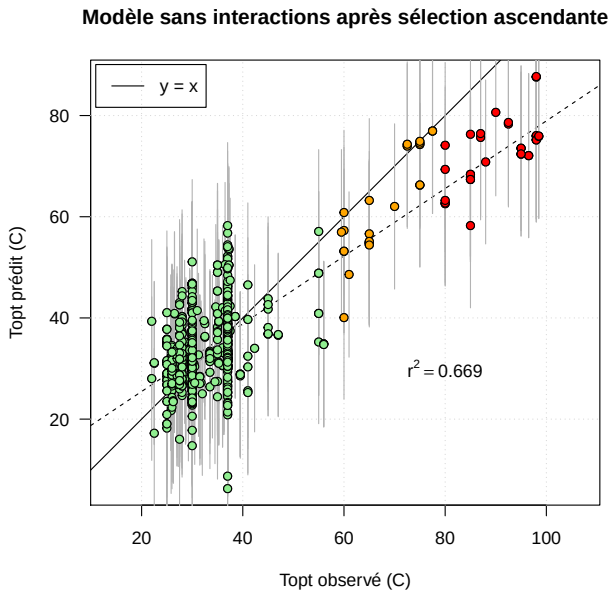
On arrive à un modèle plus économique avec 10 prédicteurs au lieu de 19 avec le modèle initial :

```
lm11sb$coef
```

```
      Ala       Arg       Asn       Asp       Gln       His       Leu       Lys
-1.405383  2.522879 -4.422590 -3.291889 -2.253405 -8.408416  2.837760  2.767280
      Phe       Tyr       Val
-5.833455 12.492138  5.415776
```

Toujours sans grosse perte de performances :

```
perfplot(lm11sf,"Modèle sans interactions après sélection ascendante")
```



Notez que l'on ne converge pas vers le même modèle qu'avec la procédure descendante, c'est tout à fait normal.

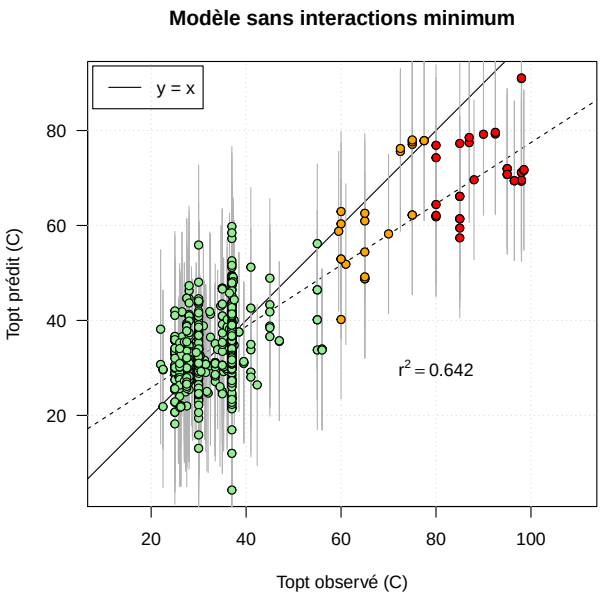
5.1.3 Modèle linéaire minimaliste

On décide de ne conserver que les prédicteurs communs aux modèles obtenus par élimination descendante et sélection ascendante en se basant sur l'heuristique que les prédicteurs communs sont dignes d'être considérés.

```
pred <- names(lm11$coef)
pred[pred %in% names(lm11sb$coef)]
 [1] "Ala" "Arg" "Asn" "Asp" "Gln" "His" "Leu" "Lys" "Phe" "Tyr" "Val"
pred[pred %in% names(lm11sf$coef)]
 [1] "Asn" "Asp" "Gln" "Glu" "His" "Ile" "Leu" "Phe" "Tyr" "Val"
(predmin <- pred[pred %in% names(lm11sb$coef) & pred %in% names(lm11sf$coef)])
[1] "Asn" "Asp" "Gln" "His" "Leu" "Phe" "Tyr" "Val"
length(predmin)
[1] 8
```

Ce qui correspond à un modèle avec 8 prédicteurs. Voyons son comportement :

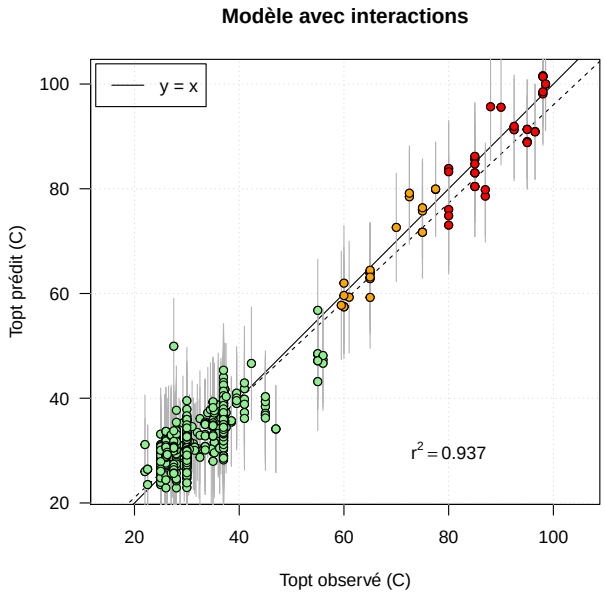
```
lm11min <-lm(as.formula(paste("topt ~ -1 +", paste(predmin, collapse = " + "))),data)
 perfplot(lm11min,"Modèle sans interactions minimum")
```



Ce n'est pas mal du tout. Les performances sont dégradées, mais pas tant que ça.

5.2 Modèle linéaire complet avec toutes les interactions

```
lm(topt~.*.-1,data)->lm2
perfplot(lm2, main = "Modèle avec interactions")
```



Très joli mais il y a beaucoup trop de predicteurs ici. Quand on part de n variables et que l'on ajoute toutes les interactions on se retrouve avec $\frac{n(n+1)}{2}$ prédicteurs, soit 190 ici, alors que nous n'avons que 730 données ici. C'est un peu fort de café.

```
nrow(data)/length(lm2$coef)
```
```
[1] 3.842105
```

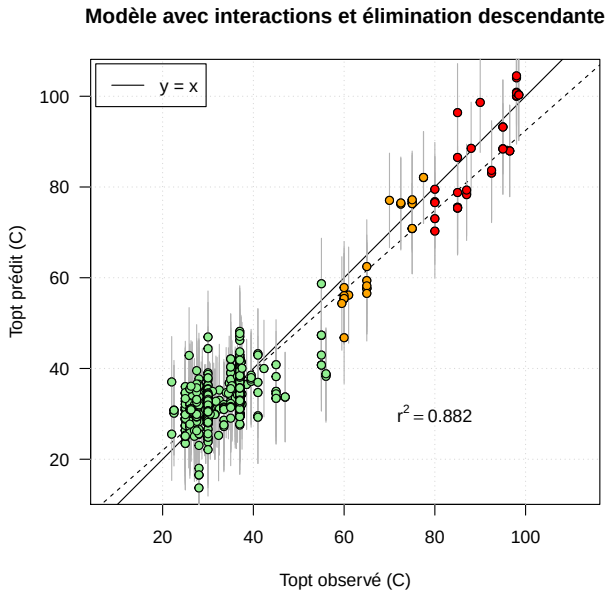
On dispose en moyenne de moins de 4 observation par coefficient, c'est vraiment très léger. Il faut essayer de rédruire drastiquement le nombre de prédicteurs.

5.2.1 Elimination descendante des prédicteurs

```
lm2sb <- selback(lm2, data)
save(lm2sb, file = "lm2sb.RData")
```

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/tdr411/lm2sb.RData"))
```

```
length(lm2sb$coef)
```
```
[1] 39
```
```
perfplot(lm2sb,"Modèle avec interactions et élimination descendante")
```



Nous n'avons plus que 39 prédicteurs ici.

5.2.2 Sélection ascendante des prédicteurs

Il faut modifier un peu le script précédent pour que cela marche, les interactions pouvant être notées aussi bien `Gly:Ala` que `Ala:Gly`.

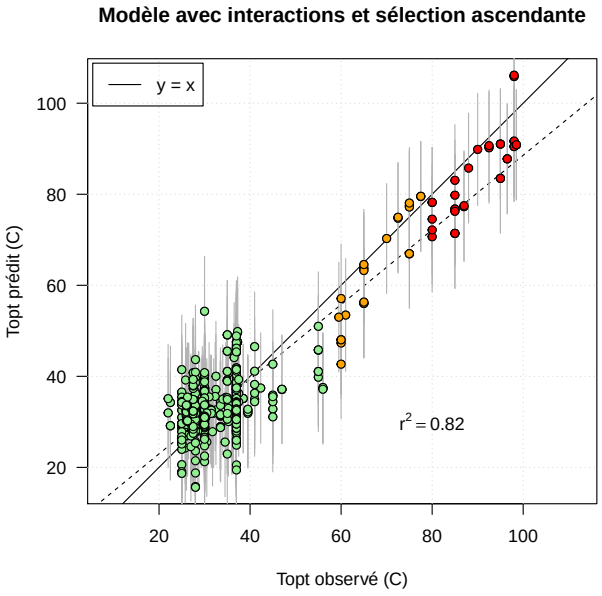
```
equal.pred <-function(pred1, pred2){
  nddp1 <- length(grep(":", pred1))
  nddp2 <- length(grep(":", pred2))
  if(nddp1 == 0 & nddp2 == 0) return(pred1 == pred2)
  if(nddp1 == 1 & nddp2 == 0) return(FALSE)
  if(nddp1 == 0 & nddp2 == 1) return(FALSE)
  if(nddp1 == 1 & nddp2 == 1){
            pred1s <- unlist(strsplit(pred1, split = ":"))
            pred2s <- unlist(strsplit(pred2, split = ":"))
    return(all(pred1s %in% pred2s))
          }
}
selfor <- function(lm0, data, alphac = 0.0001, verbose = FALSE){
        lmsf <- lm(topt ~ -1, data) # empty model
        coef <- summary(lm0)$coef
        ncoef <- nrow(coef)
        isadded <- logical(ncoef) # FALSE
        names(isadded) <- rownames(coef)
        continuer <- TRUE
        while(continuer){
                npos <- sum(!isadded)
                if(npos == 0) return(lmsf)
                newpval <- numeric(ncoef) # default 0
                names(newpval) <- rownames(coef)
                for(i in seq_len(ncoef)){
                  if(isadded[i]) next
                  qui <- names(newpval)[i]
                  tmp <- update.formula(formula(lmsf), as.formula(paste(". ~ . +", qui)))
                  tmpcoef <- summary(lm(tmp,data))$coef
                  ou <- which(sapply(rownames(tmpcoef), equal.pred, qui))
                  newpval[i] <-tmpcoef[ou,4]
                }
                tocheck <- newpval[!isadded]
                if(min(tocheck) < alphac){
                  continuer <- TRUE
```

```
                imin <- which.min(tocheck)
                qui <- names(tocheck[imin])
                if(verbose) print(paste("Adding", qui))
                new <- update.formula(formula(lmsf),
                      as.formula(paste(". ~ . +", qui )))
                lmsf <- lm(new, data)
                isadded[qui] <- TRUE
              } else {
                continuer <- FALSE
              }
        }
        return(lmsf)
}


lm2sf <- selfor(lm2, data, verbose = TRUE)
save(lm2sf, file = "lm2sf.RData")


load(url("https://pbil.univ-lyon1.fr/R/donnees/tdr411/lm2sf.RData"))


length(lm2sf$coef)
[1] 14
perfplot(lm2sf,"Modèle avec interactions et sélection ascendante")
```



Nous n'avons conservé que 14 prédicteurs ici, et le modèle n'est pas mauvais du tout.

5.2.3 Modèle minimaliste avec interactions

Voyons les prédicteurs communs :

```
pred2 <- names(lm2$coef)
pred2sf <- names(lm2sf$coef)
pred2sb <- names(lm2sb$coef)
ressf <- logical(length(pred2))
ressb <- logical(length(pred2))
for(i in seq_len(length(pred2))){
  ressf[i] <- ressb[i] <- FALSE
```

```
  for(j in seq_len(length(pred2sf)))
    if(equal.pred(pred2sf[j],pred2[i]))
      ressf[i] <- TRUE
  for(j in seq_len(length(pred2sb)))
    if(equal.pred(pred2sb[j],pred2[i]))
      ressb[i] <- TRUE
}
pred2[ressf]
```

```
 [1] "Asp"     "Glu"     "Ala:Asn" "Ala:Tyr" "Arg:Lys" "Asn:Pro" "Asp:Gln" "Gln:Glu"
 [9] "Glu:Val" "Gly:Phe" "His:Ile" "Ile:Leu" "Met:Thr" "Pro:Tyr"
```

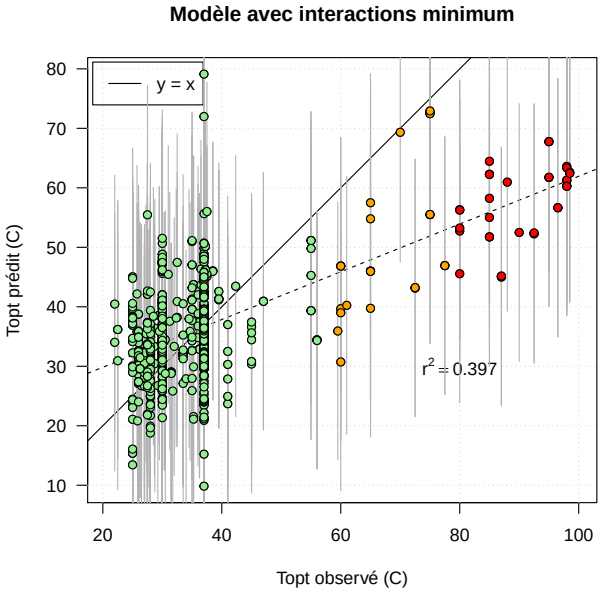
```
pred2[ressb]
```

```
 [1] "Ala"     "Asn"     "Asp"     "Glu"     "Lys"     "Pro"     "Ser"     "Thr"
 [9] "Tyr"     "Val"     "Ala:Arg" "Ala:Gln" "Ala:Lys" "Ala:Tyr" "Arg:Glu" "Arg:Pro"
[17] "Arg:Ser" "Asn:Glu" "Asn:Tyr" "Asn:Val" "Asp:His" "Asp:Leu" "Asp:Met" "Asp:Pro"
[25] "Cys:Glu" "Cys:Leu" "Gln:Gly" "Gln:Pro" "Glu:Lys" "His:Ile" "His:Lys" "His:Phe"
[33] "His:Tyr" "His:Val" "Ile:Pro" "Ile:Ser" "Lys:Pro" "Phe:Pro" "Ser:Tyr"
```

```
(pred2min <- pred2[ressb & ressf])
```

```
[1] "Asp"     "Glu"     "Ala:Tyr" "His:Ile"
```

```
lm2min <-lm(as.formula(paste("topt ~ -1 +", paste(pred2min, collapse = " + "))),data)
perfplot(lm2min,"Modèle avec interactions minimum")
```



On a forcé un peu loin dans le minimalisme ici. Si on refait la manip avec un $\alpha_{crit} = 0.05$ pour être moins sévère, on a :

```
lm2sb.5 <- selback(lm2,data,alpha=0.05)
lm2sf.5 <- selfor(lm2,data,alpha=0.05)
save(lm2sb.5, file = "lm2sb5.RData")
save(lm2sf.5, file = "lm2sf5.RData")
```

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/tdr411/lm2sb5.RData"))
load(url("https://pbil.univ-lyon1.fr/R/donnees/tdr411/lm2sf5.RData"))
```

```
pred2 <- names(lm2$coef)
pred2sf.5 <- names(lm2sf.5$coef)
pred2sb.5 <- names(lm2sb.5$coef)
ressf.5 <- logical(length(pred2))
ressb.5 <- logical(length(pred2))
```

```
for(i in seq_len(length(pred2))){
  ressf.5[i] <- ressb.5[i] <- FALSE
  for(j in seq_len(length(pred2sf.5)))
    if(equal.pred(pred2sf.5[j],pred2[i]))
      ressf.5[i] <- TRUE
  for(j in seq_len(length(pred2sb.5)))
    if(equal.pred(pred2sb.5[j],pred2[i]))
      ressb.5[i] <- TRUE
}
```

```
pred2[ressf]
```

```
[1] "Asp"     "Glu"     "Ala:Asn" "Ala:Tyr" "Arg:Lys" "Asn:Pro" "Asp:Gln" "Gln:Glu"
[9] "Glu:Val" "Gly:Phe" "His:Ile" "Ile:Leu" "Met:Thr" "Pro:Tyr"
```

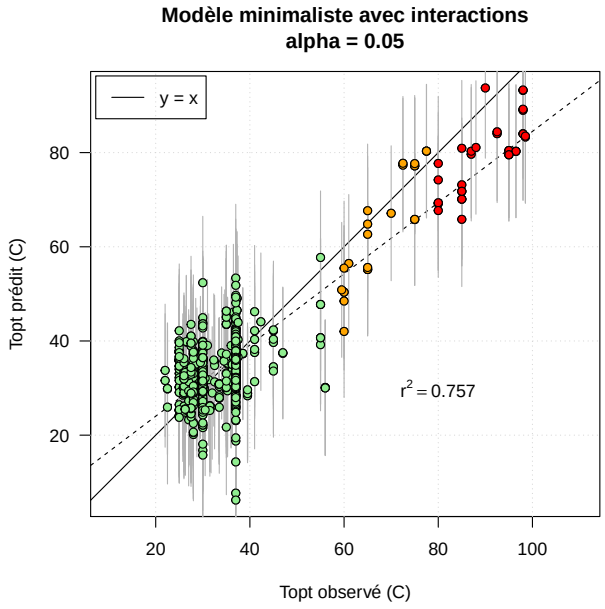
```
pred2[ressb]
```

```
 [1] "Ala"     "Asn"     "Asp"     "Glu"     "Lys"     "Pro"     "Ser"     "Thr"
 [9] "Tyr"     "Val"     "Ala:Arg" "Ala:Gln" "Ala:Lys" "Ala:Tyr" "Arg:Glu" "Arg:Pro"
[17] "Arg:Ser" "Asn:Glu" "Asn:Tyr" "Asn:Val" "Asp:His" "Asp:Leu" "Asp:Met" "Asp:Pro"
[25] "Cys:Glu" "Cys:Leu" "Gln:Gly" "Gln:Pro" "Glu:Lys" "His:Ile" "His:Lys" "His:Phe"
[33] "His:Tyr" "His:Val" "Ile:Pro" "Ile:Ser" "Lys:Pro" "Phe:Pro" "Ser:Tyr"
```

```
(pred2[ressb.5 & ressf.5] -> pred2min.5)
```

```
 [1] "Asp"     "Gln"     "Glu"     "Ala:Ser" "Ala:Tyr" "Asn:Gln" "Asn:Glu" "Asn:His"
 [9] "Asn:Met" "Asn:Val" "Asp:Cys" "Asp:Pro" "Gln:His" "Gln:Val" "Glu:Gly" "Glu:Phe"
[17] "Gly:His" "Gly:Val" "His:Ile" "Ile:Leu" "Ile:Ser" "Pro:Tyr"
```

```
lm2min.5 <-lm(as.formula(paste("topt ~ -1 +", paste(pred2min.5, collapse = " + "))),data)
perfplot(lm2min.5,"Modèle minimaliste avec interactions\nalpha = 0.05")
```



On a donc ici un modèle avec 22 prédicteurs qui ne se comporte pas trop mal.

6 Sélection de modèle sur critère

On part du modèle complet avec interactions sans ordonnée. On veut le simplifier sur des critères un peu plus objectifs que les bricolages vus ci-dessus.

Il existe une riche littérature sur les critères possibles [3] on ne consière que les deux plus courants ici. Les outils sont disponibles dans la distribution de base de R.

6.1 Critère AIC

Avec le critère AIC [1, 2] on trouve [4] :

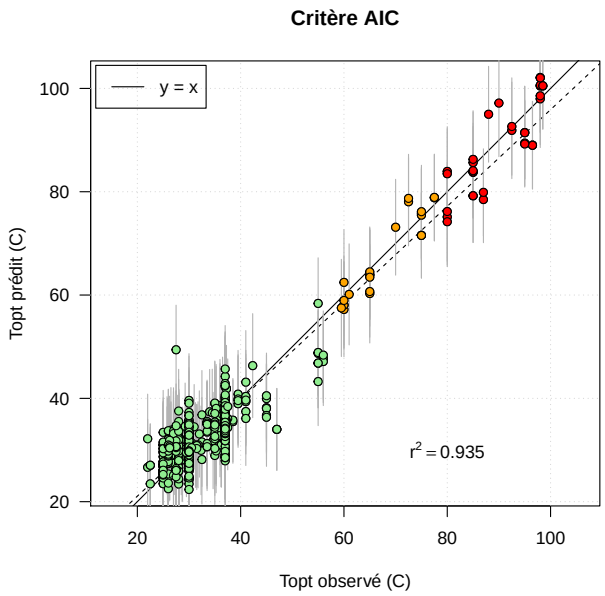
```
lm2AIC <- step(lm2, trace = 0)
save(lm2AIC, file = "lm2AIC.RData")
```

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/tdr411/lm2AIC.RData"))
```

```
length(lm2AIC$coef)
```

```
[1] 134
```

```
perfplot(lm2AIC,"Critère AIC")
```



Bon c'est pas mal, mais 134 prédicteurs c'est beaucoup trop, même si on a gagné un peu par rapport aux 190 de départ. Le problème du critère AIC c'est qu'il ne pénalise que de 2 points de vraisemblance pour tout ajout de paramètre. Ce n'est pas très stringent quand on a beaucoup de données.

6.2 Critère BIC

Avec le critère BIC [15] on trouve [5] :

3. De quoi se perdre : AIC [1, 2], TIC [17], BIC [15], AIC_c [10], NIC [13], QAIC et $QAIC_c$ [5], RIC [3], DIC [16], FIC et FRIC [6].
4. Attention : les calculs sont un peu longs, de l'ordre de 8 minutes sur un MacBook Pro en 2008.
5. Environ 10 minutes de calcul

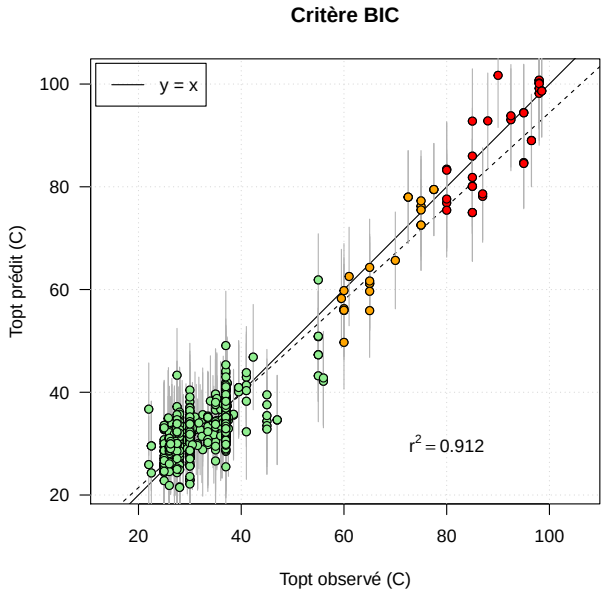
```
lm2BIC <- step(lm2, trace = 0, k = log(nrow(data)))
save(lm2BIC, file = "lm2BIC.RData")
```

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/tdr411/lm2BIC.RData"))
```

```
length(lm2BIC$coef)
```

```
[1] 66
```

```
perfplot(lm2BIC,"Critère BIC")
```



On garde le modèle BIC comme thermomètre moléculaire, avec plus de 10 points par coefficient cela reste dans les limites de l'acceptable :

```
nrow(data)/length(lm2BIC$coef)
```

```
[1] 11.06061
```

```
lmthermometre <- lm2BIC
lmthermometre$coef
```

```
       Ala        Arg        Asn        Asp        Cys        Gln        Glu 
33.2620220  9.3994645 69.3889114 -40.4814340 -99.6463176 67.8508947 -98.8551492 
       Gly        His        Ile        Leu        Lys        Met        Phe 
-35.4129567 -45.3504166  8.3666676 -24.2942559 -46.0259463 -65.5686702 -56.5693457 
       Pro        Ser        Thr        Tyr        Val     Ala:Cys     Ala:Glu 
105.9237536 83.1197206 -25.2421853 105.8507576 29.5671224  2.0898681  0.9674142 
    Ala:Phe     Ala:Pro     Ala:Ser     Ala:Tyr     Ala:Val     Arg:Glu     Arg:Pro 
  1.4669891 -2.1718585 -2.4546401 -3.0311876 -1.9090660  3.7147447 -2.5292819 
    Arg:Ser     Asn:Gln     Asn:Glu     Asn:Gly     Asn:Tyr     Asn:Val     Asp:Glu 
 -3.3216456 -3.9817729 -2.2576524 -1.6560784 -4.5185561 -2.2952890  2.5700124 
    Asp:Gly     Asp:His     Asp:Leu     Asp:Lys     Asp:Pro     Cys:Leu     Cys:Met 
  4.1632119  7.4020418 -1.6893344  1.3853689 -3.2171532  6.9854063 12.9904833 
    Cys:Pro     Gln:Pro     Gln:Ser     Gln:Val     Glu:His     Glu:Ile     Glu:Leu 
 -4.9780987 -5.3553737 -1.8861284 -2.7736593 -4.7175415  1.5113714  2.1201749 
    Glu:Lys     Glu:Phe     Glu:Thr     Gly:Leu     Gly:Lys     His:Ile     His:Lys 
  2.9826846  3.7073224  2.1439774  0.8935774  1.2471362 -2.9365716  3.6239663 
    His:Phe     His:Tyr     Ile:Met     Ile:Ser     Leu:Thr     Lys:Phe     Lys:Pro 
 13.8818176 -8.6030183  3.5057173 -2.9928139  0.9862787  2.3249820 -1.5539076 
    Met:Val     Phe:Pro     Ser:Tyr 
  4.3187384 -5.4313360 -6.1424547 
```

7 Comparaison avec d'autres thermomètres moléculaires

7.1 Di giullio

Un collègue italien, Massimo Di Giulio, a proposé [7] d'utiliser la statistique suivante (TI pour Thermophily Index) pour une protéine comportant N acides aminés :

$$\mathrm{TI} = \sum_{i=1}^{N} \frac{R_i}{N}$$

Les coefficients sont donnés dans la table 1 de l'article :

Table 1
Thermophily ranks (See text for their definition)

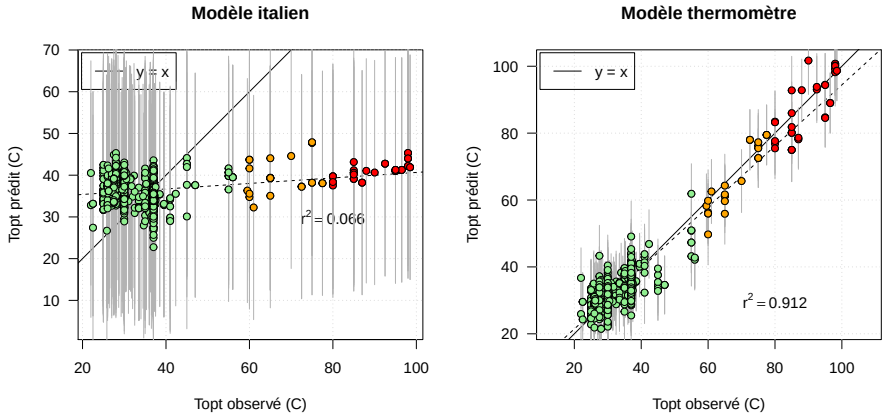
Arg = 19.50	Cys = 13.75	Phe = 10.25	Asp = 6.00
Trp = 18.25	Leu = 13.75	Lys = 10.00	Gln = 5.25
Pro = 17.25	Val = 13.00	His = 9.25	Thr = 5.00
Ile = 15.50	Glu = 11.25	Met = 7.00	Asn = 2.25
Tyr = 14.75	Ala = 11.00	Gly = 6.00	Ser = 1.00

C'est donc une forme linéaire sur les fréquences des 20 acides aminés. Calculons et comparons :

```
data20 <- as.data.frame(frlaa/100)
tir <- numeric(20)
names(tir) <- names(data20)
tir[c("Arg","Trp","Pro","Ile","Tyr")] <- c(19.5,18.25,17.25,15.5,14.75)
tir[c("Cys","Leu","Val","Glu","Ala")] <- c(13.75,13.75,13,11.25,11)
tir[c("Phe","Lys","His","Met","Gly")] <- c(10.25,10,9.25,7,6)
tir[c("Asp","Gln","Thr","Asn","Ser")] <- c(6,5.25,5,2.25,1)
tir
```

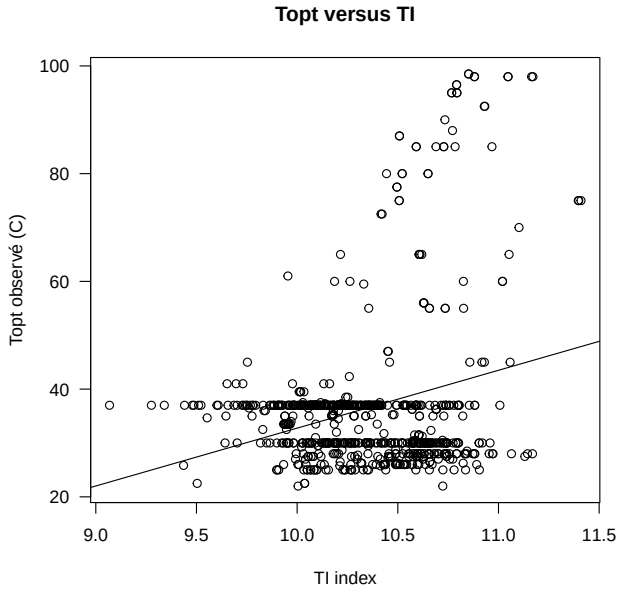
```
  Ala   Arg   Asn   Asp   Cys   Gln   Glu   Gly   His   Ile   Leu   Lys   Met   Phe
11.00 19.50  2.25  6.00 13.75  5.25 11.25  6.00  9.25 15.50 13.75 10.00  7.00 10.25
  Pro   Ser   Thr   Trp   Tyr   Val
17.25  1.00  5.00 18.25 14.75 13.00
```

```
TI <- as.matrix(data20) %*% tir
lmTI <- lm(topt~TI)
par(mfrow=c(1,2))
perfplot(lmTI, "Modèle italien")
perfplot(lmthermometre, "Modèle thermomètre")
```



La mauvaise performance de cette statistique comme variable prédictive s'explique par la très forte dispersion de ses valeurs dans le groupe des mésophiles de ce jeu de données, ce qui tire la droite de régression vers le bas et ôte tout espoir de prédire des températures élevées :

```
plot(TI, topt, las = 1, xlab = "TI index",
ylab = "Topt observé (C)",
main = "Topt versus TI")
abline(lm(topt~TI))
```



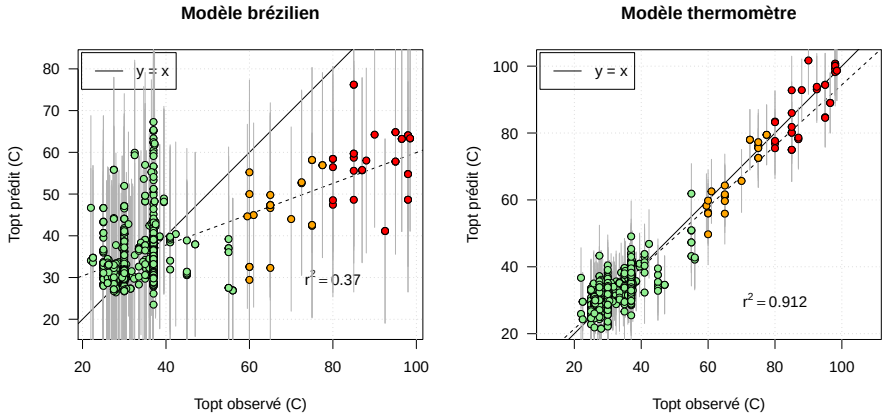
7.2 (E+K)/(Q+H)

Deux collègues brésiliens, Sávio T. Farias et Maria Christina M. Bonato, ont proposé [8] d'utiliser le rapport (E + K)/(Q + H) comme variable prédictive. Retrouvons les notations trois lettres pour les acides-aminés :

```
library(seqinr)
aaa(s2c("EKQH"))
```
```
[1] "Glu" "Lys" "Gln" "His"
```

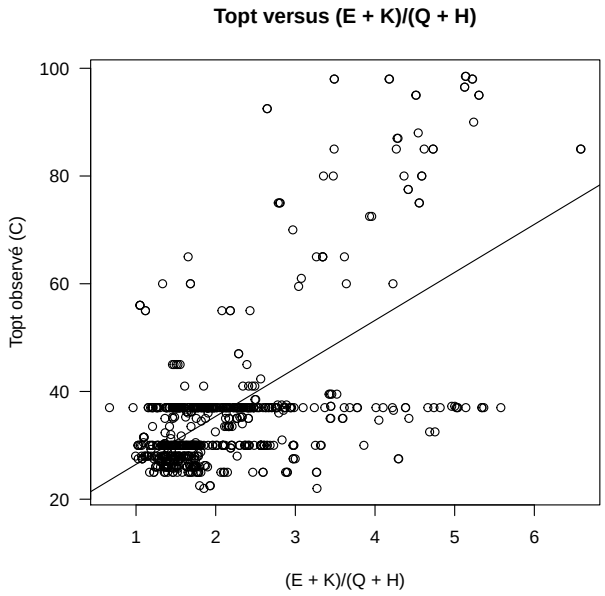
Calculons ce rapport et comparons :

```
ekqh <- with(data, (Glu+Lys)/(Gln+His))
lmBrazil <- lm(topt~ekqh)
par(mfrow=c(1,2))
perfplot(lmBrazil, "Modèle brézilien")
perfplot(lmthermometre, "Modèle thermomètre")
```



Ici encore, la mauvaise performance de la statistique s'explique par la très forte dispersion de ses valeurs chez les mésophiles :

```
plot(ekqh, topt, las = 1, xlab = "(E + K)/(Q + H)",
ylab = "Topt observé (C)",
main = "Topt versus (E + K)/(Q + H)")
abline(lm(topt~ekqh))
```



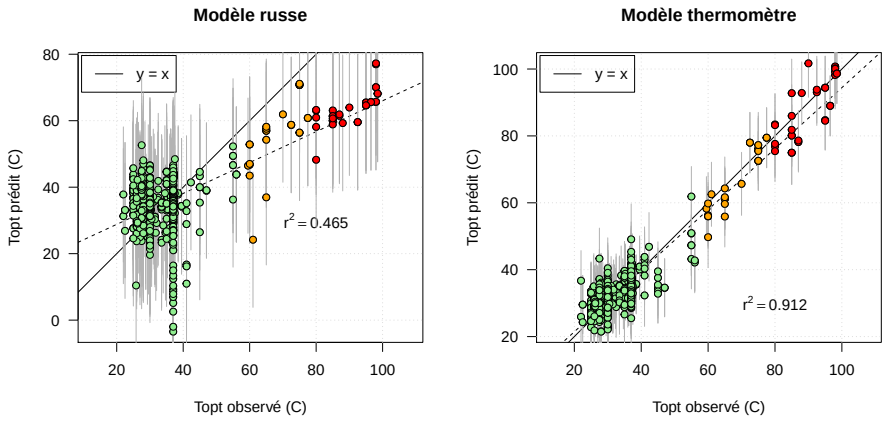
7.3 IVYWREL

Trois collègues travaillant aux états unis d'Amérique, Konstantin B. Zeldovich, Igor N. Berezovsky et Eugene I. Shakhnovich, ont proposé [18] d'utiliser la fréquence totale en 7 acides-aminés comme variable prédictive pour T_{opt}. C'est encore ici une forme linéaire sur les fréquences en acides-aminés qui se calcule simplement sous R :

```
Fr <- numeric(20)
names(Fr) <- names(data20)
Fr[aaa(s2c("IVYWREL"))] <- 1
Fr
```

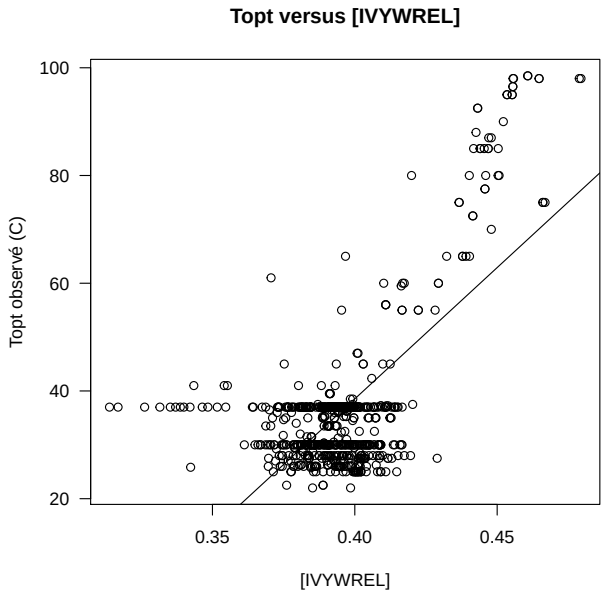
```
Ala Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser Thr Trp Tyr Val
  0   1   0   0   0   0   1   0   0   1   1   0   0   0   0   0   0   1   1   1
```

```
F <- as.matrix(data20) %*% Fr
lmF <- lm(topt~F)
par(mfrow=c(1,2))
perfplot(lmF, "Modèle russe")
perfplot(lmthermometre, "Modèle thermomètre")
```



Ici encore c'est la variabilité des mésophiles qui plombe la régression :

```
plot(F, topt, las = 1, xlab = "[IVYWREL]",
ylab = "Topt observé (C)",
main = "Topt versus [IVYWREL]")
abline(lm(topt~F))
```



8 Utilisation prédictive du modèle

D.J. Brooks et collaborateurs ont estimé [4] la composition en acides aminés du dernier ancêtre commun à tous les êtres vivants sur terre (LUCA [6]). Quelle prédiction peut-on faire pour le T_{opt} de LUCA ? Quelle précision peut-on attendre de cette estimation ?

Les données sont extraites de la table 1 de l'article reproduite ci-dessous.

Table 1. Amino acid frequencies within a set of 65 proteins observed in 8 modern species and inferred in the LUA

	Aae	Tma	Ssp	Bsu	Eco	Sce	Mth	Afu	Average Modern Set	Inferred LUA	Average Modern Proteome
Ala	0.0651	0.0660	0.0932	0.0826	0.0990	0.0688	0.0779	0.0814	0.0792	0.0819	0.0726
Arg	0.0571	0.0626	0.0613	0.0519	0.0620	0.0524	0.0744	0.0675	0.0612	0.0685	0.0527
Asn	0.0310	0.0332	0.0324	0.0348	0.0348	0.0429	0.0244	0.0272	0.0326	0.0272	0.0398
Asp	0.0481	0.0491	0.0527	0.0546	0.0564	0.0542	0.0601	0.0518	0.0534	0.0456	0.0516
Cys	0.0077	0.0060	0.0094	0.0043	0.0087	0.0121	0.0104	0.0095	0.0085	0.0040	0.0102
Gln	0.0253	0.0242	0.0503	0.0345	0.0361	0.0368	0.0237	0.0207	0.0315	0.0166	0.0319
Glu	0.0964	0.0930	0.0685	0.0873	0.0699	0.0653	0.0924	0.0944	0.0834	0.1182	0.0764
Gly	0.0705	0.0713	0.0781	0.0751	0.0780	0.0680	0.0745	0.0720	0.0734	0.0733	0.0694
His	0.0191	0.0192	0.0183	0.0202	0.0205	0.0219	0.0227	0.0200	0.0202	0.0237	0.0189
Ile	0.0711	0.0712	0.0635	0.0674	0.0592	0.0680	0.0770	0.0719	0.0687	0.0806	0.0696
Leu	0.0951	0.0907	0.1045	0.0883	0.0899	0.0919	0.0848	0.0857	0.0914	0.0832	0.1010
Lys	0.0957	0.0865	0.0540	0.0745	0.0616	0.0772	0.0581	0.0776	0.0732	0.0874	0.0642
Met	0.0191	0.0227	0.0193	0.0241	0.0282	0.0230	0.0285	0.0234	0.0236	0.0202	0.0246
Phe	0.0393	0.0397	0.0319	0.0340	0.0340	0.0419	0.0362	0.0386	0.0369	0.0308	0.0443
Pro	0.0435	0.0428	0.0463	0.0394	0.0406	0.0411	0.0445	0.0414	0.0425	0.0406	0.0423
Ser	0.0361	0.0417	0.0461	0.0466	0.0438	0.0586	0.0475	0.0413	0.0452	0.0213	0.0614
Thr	0.0449	0.0459	0.0544	0.0550	0.0534	0.0543	0.0450	0.0411	0.0492	0.0390	0.0501
Trp	0.0111	0.0099	0.0106	0.0078	0.0096	0.0111	0.0101	0.0109	0.0101	0.0067	0.0113
Tyr	0.0364	0.0325	0.0255	0.0296	0.0257	0.0329	0.0276	0.0311	0.0301	0.0231	0.0340
Val	0.0873	0.0918	0.0797	0.0880	0.0887	0.0777	0.0801	0.0925	0.0857	0.1080	0.0736

Species abbreviations are as follows: Aae, *Aquifex aeolicus*; Tma, *Thermotoga maritima*; Ssp, *Synechocystis* PCC6803; Bsu, *Bacillus subtilis*; Eco, *Escherichia coli K12*; Sce, *Saccharomyces cerevisiae*; Mth, *Methanobacterium thermoautotrophicum*; Afu, *Archaeoglobus fulgidus*. The column headed Inferred LUA gives the inferred frequencies within the LUA of the set of 65 proteins; that headed Average Modern Set gives the average frequencies in this set within the 8 extant species included in the study; that headed Average Modern Proteome gives the average frequencies of the whole proteomes for the 8 species.

Entrer les données sous R et prédire :

6. Last Universal Common Ancestor, le "Common" est un peu redondant pour un ancêtre, on abbrège aussi en LUA

```
brooks <- read.table("http://pbil.univ-lyon1.fr/R/donnees/brooks2004.txt", head =T, sep="\t", row.names = 1)
all(rownames(brooks) == names(data20))
```

```
[1] TRUE
```

```
lua <- as.data.frame(t(brooks[-18,"LUA", drop=FALSE]))*100
(lua.topt <- predict(lmthermometre,lua, interval = "prediction"))
```

```
        fit      lwr      upr
LUA 99.35771 79.80726 118.9082
```

```
diff(lua.topt[,2:3])
```

```
    upr
39.10092
```

Avec cette composition pour son protéome, LUCA serait un hyperthermophile avec $T_{opt} = 100 \pm 20$ °C. Si l'on considère la variabilité observée pour les mésophiles les mieux documentés (autour de 37 °C), il est clair que l'on ne peut pas espérer faire mieux du point de vue de la précision de l'estimation. Pour ce qui est de son exactitude, elle est conditionnelle à l'inférence faite de la composition du protéome de LUCA. Ces inférences sont connues pour être particulièrement délicates [9].

On veut faire une représentation graphique pour visualiser cette prédiction. L'influence de la température T, sur le taux de croissance en phase exponentielle des micro-organismes, μ, est donné en première approximation, par :

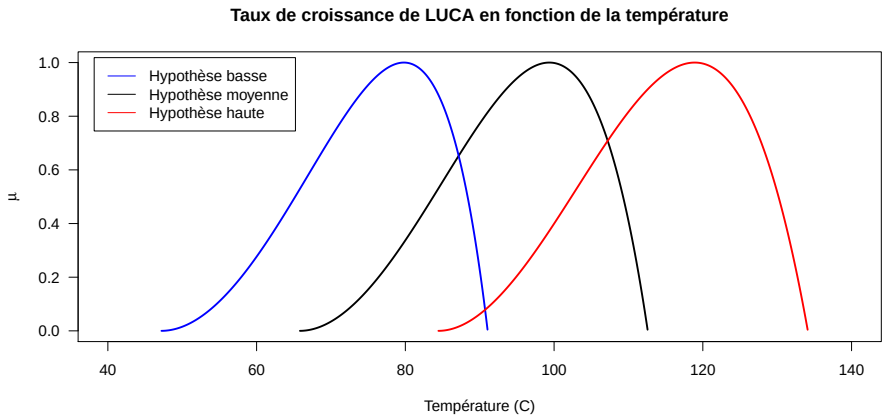
$$\mu(T) = \begin{cases} 0 & \text{si } T \notin [T_{min}, T_{max}] \\ \frac{\mu_{opt}(T-T_{max})(T-T_{min})^2}{(T_{opt}-T_{min})[(T_{opt}-T_{min})(T-T_{opt})-(T_{opt}-T_{max})(T_{opt}+T_{min}-2T)]} & \text{si } T \in [T_{min}, T_{max}] \end{cases}$$

où T_{min} représente la température en deça de laquelle il n'y a plus de croissance, T_{max} la température au delà de laquelle il n'y a plus de croissance, T_{opt} la température pour laquelle le taux de croissance atteint son maximum μ_{opt}. C'est le modèle dit des températures cardinales [14]. On commence par définir une fonction correspondant au modèle dans R :

```
CTMI <- function(T, Tmin, Topt, Tmax, Muopt)
{
  if( T <= Tmin || T >= Tmax ) return(NA)
  Num <- (T-Tmax)*(T-Tmin)^2
  Den <- (Topt-Tmin)*((Topt-Tmin)*(T-Topt)-(Topt-Tmax)*(Topt+Tmin-2*T))
  return(Muopt*Num/Den)
}
```

C'est une fonction avec quatre paramètres alors que nous n'en avons estimé qu'un seul. Que faire ? Pour μ_{opt} on décide de travailler en valeur relatives, autrement dit on fixe $\mu_{opt} = 1$. Pour T_{min} et T_{max} on exploite la forte correlation qui existe entre les trois températures cardinales [14].

```
futil <- function(x, topt){
        sapply(x, function(x)
  CTMI(x, Tmin = 0.953*topt - 28.913, Topt = topt,
  Tmax = 1.101*topt + 3.203, Muopt = 1))
}
x <- seq(from = 40, to = 140, length = 1000)
plot(x, futil(x, lua.topt[1]), type = "l",
  las = 1, ylab = expression(mu),
  xlab = "Température (C)",
  main = "Taux de croissance de LUCA en fonction de la température", lwd = 2)
lines(x, futil(x, lua.topt[2]), col = "blue", lwd = 2)
lines(x, futil(x, lua.topt[3]), col = "red", lwd = 2)
legend("topleft", inset = 0.02,
legend = c("Hypothèse basse", "Hypothèse moyenne", "Hypothèse haute"), lty = 1, col = c("blue","black","red"))
```



D'après ces prédictions, à prendre *cum grano salis*, LUCA devait être capable de croître entre 50 et 130 °C environ. On retiendra l'extrême imprécision de ces prédictions par rapport à la gamme de température possible pour l'eau en phase liquide.

Références

[1] H. Akaike. Information theory and an extention of the maximum likelihood principle. In B.N. Petrov and P. Cźaki, editors, *2nd International Symposium on Information Theory*, pages 267–281, Budapest, 1973. Akademiai Kiadó.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19 :716–723, 1974.

[3] A. Basu, I.R. Harris, N.L. Hjort, and M.C. Jones. Robust and efficient estimation by minimizing a density power divergence. *Biometrika*, 85 :549–559, 1998.

[4] D.J. Brooks, J.R. Fresco, and M Singh. A novel method for estimating ancestral amino acid composition and its application to proteins of the Last Universal Ancestor. *Bioinformatics*, 20 :2251–2257, 2004.

[5] K.P. Burnham and D.R. Anderson. *Model Selection and Inference : A Practical Information-Theoretic Approach*. Springer-Verlag, New York, USA, 1998.

[6] G. Claeskens and N.L. Hjort. The focused information criterion. *Journal of the American Statistical Association*, 98 :900–916, 2003.

[7] M. Di Giulio. The late stage of genetic code structuring took place at high temperature. *Gene*, 261 :189–195, 2000.

[8] S.T. Farias and M.C.M Bonato. Preferred amino acids and thermostability. *Genetics and Molecular Research*, 2 :383–393, 2003.

[9] M. Gouy and M. Chaussidon. Ancient bacteria liked it hot. *Nature*, 451 :635–636, 2008.

[10] C.M. Hurvich and C.-L. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76 :297–307, 1989.

[11] J.R. Lobry and D. Chessel. Internal correspondence analysis of codon and amino-acid usage in thermophilic bacteria. *Journal of Applied Genetics*, 44 :235–261, 2003.

[12] J.R. Lobry and A. Necşulea. Synonymous codon usage and its potential link with optimal growth temperature in prokaryotes. *Gene*, 385 :128–136, 2006.

[13] N. Murata, S. Yoshizawa, and S. Amari. Network information criterion - determining the number of hidden units for artificial natural network models. *IEEE Transactions on Neural Networks*, 5 :865–872, 1994.

[14] 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.

[15] G. Schwarz. Estimating the dimension of a model. *Annals of Statistcs*, 6 :461–464, 1978.

[16] D.J. Spiegelhalter, N.G. Best, B.P. Carlin, and A. van der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Ser. B*, 64 :583–639, 2002.

[17] K. Takeuchi. Distribution of informational statistics and a criterion of model fitting (in japanese). *Suri-Kagaku (Mathematical Sciences)*, 153 :12–18, 1976.

[18] K.B. Zeldovich, I.N. Berezovsky, and E.I. Shakhnovich. Protein and DNA sequence determinants of thermophilic adaptation. *PLoS Comput. Biol.*, 3 :e5, 2007.