Stability selection pour la prédiction de la résistance d'une souche bactérienne à un antibiotique

M1 parcours SSD – UE Apprentissage Statistique I

Vadim Bertrand Lola Cottin Marie Gaffet

29/04/2022

1 Introduction

Certaines souches bactériennes présentent une résistance particulière aux antibiotiques développés pour lutter contre elles.
Cette résistance peut être modélisée par une régression logistique ayant pour variables explicatives la présence ou non de motifs génomiques dans le génome des souches. Seulement, le grand nombre de motifs (plusieurs dizaines de milliers) impose de sélectionner des variables explicatives avant d'entrainer le modèle de régression logistique. Pour cela, des méthodes de régression pénalisant le nombre de variables sélectionnées existent. L'approche du Lasso peut notamment être utilisée, mais quand les variables mises en jeu sont corrélées, elle présente l'inconvénient d'être instable et de sélectionner plus de variables que nécessaire.
Pour pallier ce problème, la technique de stability selection [1] peut être intéressante. Il s'agit de répéter plusieurs fois l'ajustement d'une régression logistique Lasso basée sur un sous-échantillonnage du jeu de données et d'en déduire la fréquence de sélection des variables par l'ensemble des régressions. Les variables dont la fréquence de sélection dépasse un seuil donné sont enfin utilisées pour l'ajustement d'une régression logistique non pénalisée.
Nous verrons ici comment implémenter la procédure de stability selection sur R, puis nous commenterons les résultats obtenus selon le seuil de sélection en les comparant à ceux d'une régression logistique Lasso "classique".

2 Implémentation de la stability selection

L'ensemble du code mis en oeuvre pour l'implémentation et l'application de la stability selection est organisé en quatre fichiers :

- ***stabsel_func.R*** contient les fonctions relatives à la stability selection (ajustement et évaluation) ;
- ***stabsel_data.R*** pour importer le jeu de données utilisé ;
- ***stabsel_run.R*** entraîne et évalue l'ensemble des modèles, puis sauvegarde un résumé des résultats ;
- ***stabsel_fig.R*** définie les fonctions utiles pour l'affichage des résultats.

Le code étant présent et commenté en annexes, nous ne reviendrons donc que brièvement sur les éléments-clés :

- nous nous sommes appuyés sur le package **R** *glmnet* [2] pour ajuster les régressions logistiques Lasso [3] ;

- par défaut, les sous-échantillons sélectionnés aléatoirement sont de taille $N/2$ avec N la taille de l'échantillon ;
- les modèles Lasso et sous-échantillons utilisés pour construire le chemin de stabilité sont au nombre de 100 ;
- le maximum des fréquences, ou probabilités, de sélection selon les valeurs de lambda du chemin de stabilité est utilisé pour sélectionner les variables stables selon le seuil de stabilité souhaité ;
- les seuils de stabilité considérés sont : $0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1$;
- nous avons également implémenté la procédure de validation croisée pour ajuster l'hyper-paramètre du seuil de stabilité :
  - la procédure utilise 10 folds par défaut ;
  - deux seuils de stabilité optimaux sont retournés : l'un minimisant l'erreur, l'autre minimisant le nombre de variables tel que l'erreur se situe à un écart type du minimum ;
  - l'erreur utilisée est $100 - t_{agrément}$ avec $t_{agrément}$ le taux d'agrément, ou taux de bonne classification global.

3 Prédiction de la résistance d'une souche bactérienne à un antibiotique

Le jeu de donnée d'apprentissage que nous avons utilisé permet d'associer la résistance à la *streptomycine* de 966 souches bactériennes de l'espèce *Mycobacterium tuberculosis* à la présence de 53677 motifs génomiques [4]. Nous avons ensuite utilisé un jeu de test correspondant de 200 souches bactériennes.
Les jeux de données ne présentant pas de données manquantes, la seule transformation que nous avons effectuée est l'encodage de la résistance à l'antibiotique de 1 pour résistant et -1 pour sensible à 1 pour résistant et 0 pour sensible.

La seconde étape de notre étude consistait à obtenir le chemin de régularisation et les hyper-paramètres λ d'un modèle de régression logistique Lasso, puis le chemin de stabilité pour ces mêmes λ du modèle de stability selection. Nous avons représenté ces chemins sur la ***Figure 1***, en colorant chacune des variables selon le maximum pris par leur probabilité de sélection en fonction de λ.

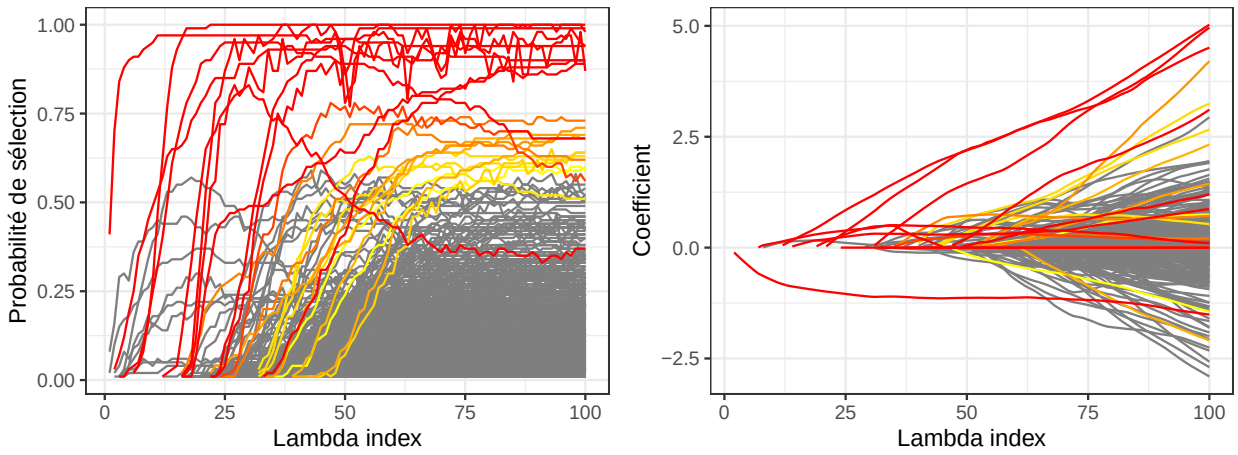

Figure 1: Chemins de stabilité (à gauche) et de régularisation (à droite)

Tout d'abord nous pouvons observer que pour les λ d'index 25 à 100, la probabilité de sélection des variables les plus "importantes" varie peu. Cette propriété mise en avant par Bühlmann et Meinshausen dans leur article [1] introduisant la stability selection conforte le choix du maximum des probabilités de sélection des variables comme statistique de sélection des variables selon le seuil de stabilité souhaité et permet de

contourner la question parfois difficile du choix de l'hyper-paramètre λ dans l'approche Lasso classique. Nous pouvons également remarquer que lorsque l'index de l'hyper-paramètre λ est supérieur à 35, de nombreuses variables instables sont introduites dans le modèle Lasso classique, ce qui augmente inutilement la dimension de son support ; à l'inverse quand λ est plus grand (pour des index inférieur à 35), la sélection des variables devient trop drastique et élimine des variables stables.

Nous avons ensuite souhaité observer si cette sélection de variables plus stable se traduit par l'ajustement de modèles de régression logistique plus performants. Pour cela, nous avons calculé le taux de bonne de classification de dix modèles : le Lasso classique et neuf modèles de stability selection à des seuils de stabilité différents. Etant donné que le nombre de variables mises en jeu dans les modèles est un facteur important, en particulier pour expliquer la résistance aux antibiotiques et pas uniquement la prédire, nous avons illustré sur la ***Figure 2*** la performance de chacun des modèles en fonction de la taille de leur support.

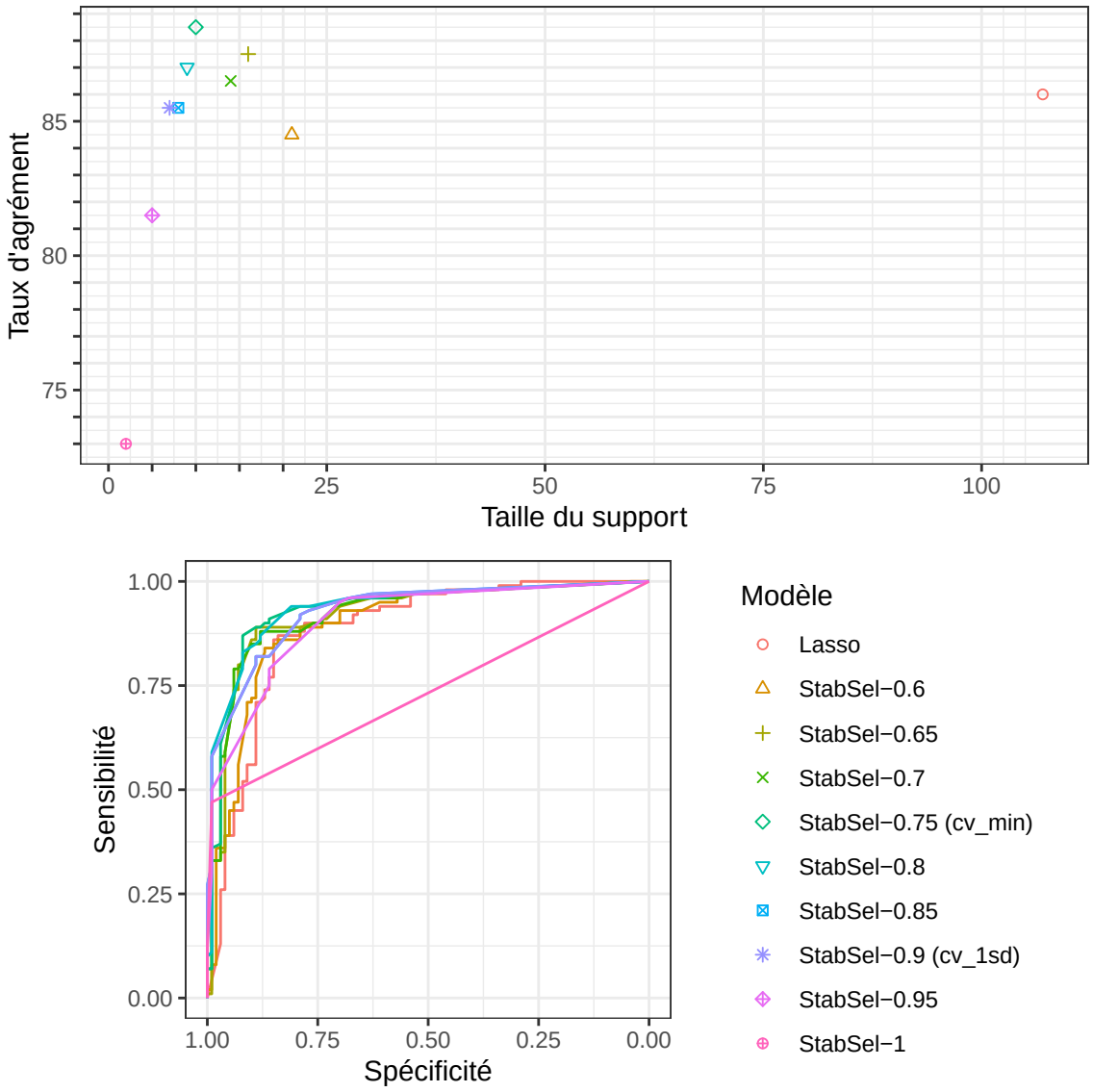

Figure 2: Taux d'agrément en fonction de la taille du support (en haut) et courbes ROC (en bas) des différents modèles

Il est visible que le taux d'agrément de sept des modèles de stability selection est comparable à celui du Lasso

(et meilleur pour quatre d'entre eux), tout en divisant par cinq, voire dix, le nombre de variables utilisées.
Comme énoncé par Bühlmann et Meinshausen [1], les résultats obtenus en terme de performance ou de taille du support sont proches pour des seuils de stabilité compris entre 0.6 et 0.9. Au delà de 0.9, les performances semblent se dégrader fortement en l'absence de variables visiblement importantes.
La similitude des résultats pour un seuil de stabilité cohérent et la facilité via ce genre de représentation à choisir ce seuil rendent l'utilisation de la validation croisée peu pertinente, d'autant que celle-ci est très coûteuse. Par ailleurs, les sous-échantillonnages impliqués par l'approche de stability selection et la construction des probabilités de sélection jouent déjà un rôle similaire à celui de la validation croisée.
S'attarder sur les courbes ROC des modèles peut également être intéressant pour observer leur différence avec un classifieur aléatoire. Le constat est semblable à celui fait via le taux d'agrément : les modèles de stability selection pour des seuils de stabilité dans l'intervalle $[0.6, 0.9]$ sont les plus proches du classifieur parfait.

3.1 Discussion sur ces bons résultats

Pour conclure cette étude, nous avons tenté de comprendre pourquoi la stability selection semble bien adaptée à la problèmatique de la résistance aux antibiotiques.
Nous nous sommes donc intéressés aux coefficients des différentes régression logistiques évoquées précédemment. Pour rappel, dans le cadre de la régression logistique avec des prédicteurs binaires, les coefficients des modèles indiquent la variation du logarithme de la probabilité d'avoir l'évènement prédit (ici, la résistance à l'antibiotique) quand le prédicteur passe de $FAUX$ à $VRAI$ (ici, de motif génomique absent à présent).

La ***Figure 3*** illustre la répartition des coefficients des différents modèles pour plusieurs niveaux de significativité.

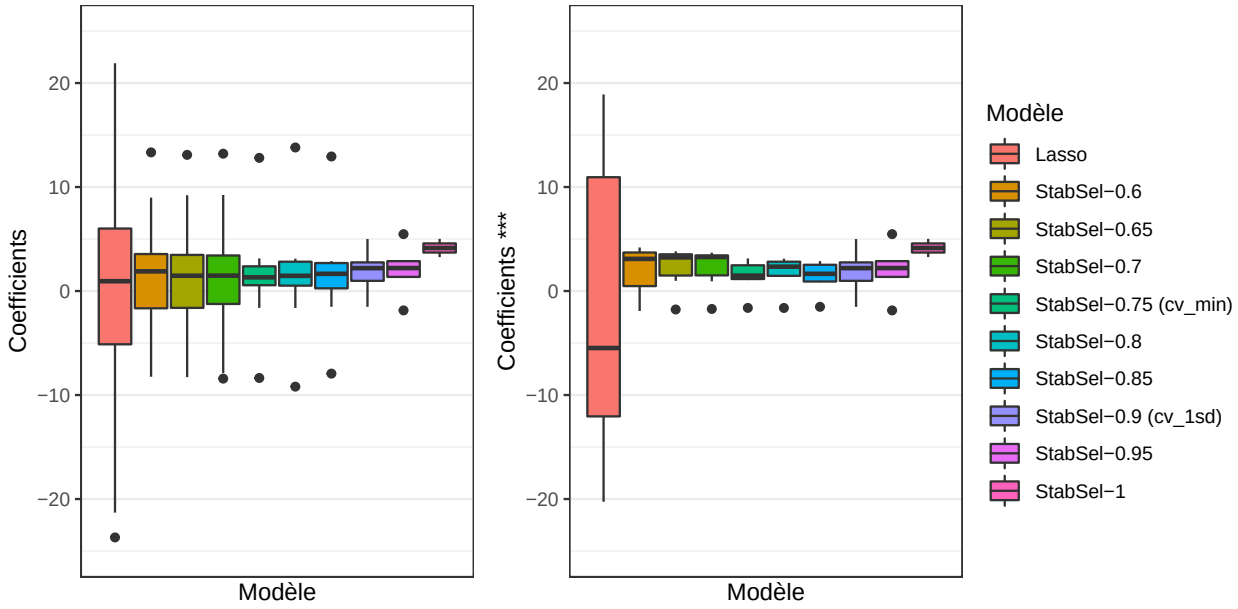

Figure 3: Boîtes à moustaches des coefficients. Sans restiction (en haut à gauche) ; aux niveaux 0.01 (en haut à droite), 0.05 (en bas à gauche) et 0.001 (en bas à droite)

Nous constatons que les coefficients obtenus par stability selection (quelque soit le seuil de stabilité) sont majoritairement positifs tandis que les coefficients du Lasso classique sont positifs et négatifs.
Nous pouvons donc en déduire que les modèles de stability selection retournent les motifs génomiques résistants à l'antibiotique (coefficients positifs) alors que le modèle Lasso classique retourne les motifs génomiques

résistants et sensibles (coefficients négatifs) à l'antibiotique. La nature de la stability selection semble suggérer que les motifs sensibles à l'antibiotique sont moins souvent retrouvés dans les souches bactériennes, à l'inverse des motifs résistants. Lorsqu'il s'agit d'utiliser le modèle ajusté sur des souches bactériennes nouvelles, non utilisées pour l'entraînement, il faut donc qu'un nombre de motifs sensibles bien plus importants que de motifs résistants soient connus du modèle pour que la potentielle sensibilité des nouvelles souches soit captée.

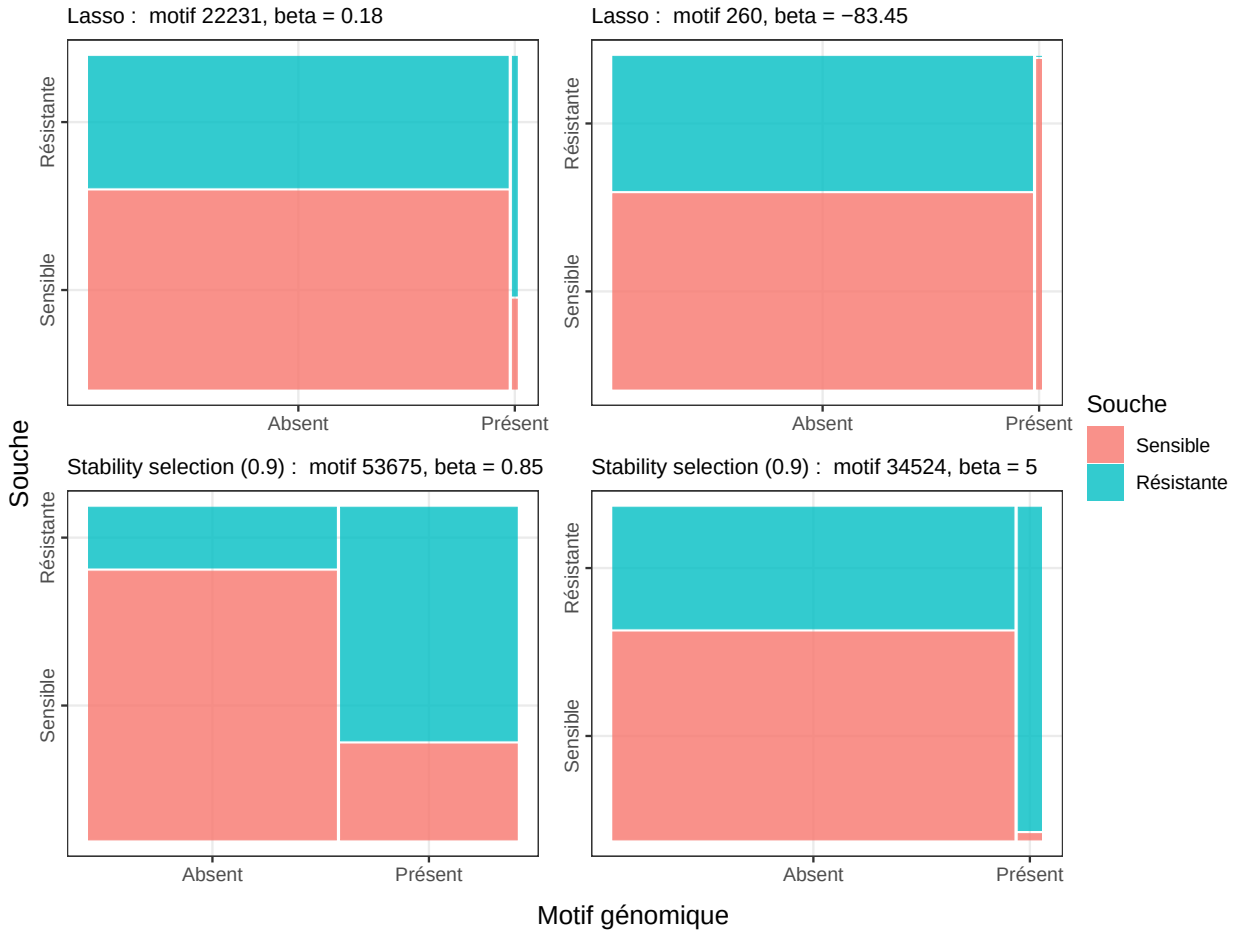

Figure 4: Diagrammes en mosaique de la résistance antibiotique en fonction de la présence de motifs génomiques. En haut pour le modèle Lasso et en bas pour le stability selection au seuil 0.9. A gauche le motif dont la valeur absolue du coefficient est la plus faible et à droite la plus élevée

De plus, nous remarquons que l'amplitude des coefficients du Lasso classique est très forte, y compris en se restreignant uniquement aux coefficients significatifs, ce qui est nettement moins le cas pour les modèles de stability selection. Cela peut également s'expliquer par la tendance du Lasso classique à capter des motifs résistants et sensibles revenant peu régulièrement dans le jeu d'entraînement.
Certains de ces motifs rares peuvent se montrer de bons classifieurs et la maximisation de la $log - vraisemblance$ qui est faite pour estimer les coefficients de la regression logistique va tirer les coefficients de ces classifieurs en négatifs pour les motifs sensibles, et en positifs pour les motifs résistants. Ce phénomène se retrouve moins dans les modèles de stability selection car en conservant les variables les plus fréquentes il est immaginable que, prises séparément, ces variables soient des classifieurs plus équivalent, et donc que leurs coefficients restent plus proches de zéro. La ***Figure 4*** présentant les tables de contingence de la présence de certains motifs génomiques et de la résistance à l'antibiotique vient illustrer ces suppositions.

La capacité des modèles de stability selection à capter la résistance plutôt que la sensibilité expliquerait donc les bons résultats prédictifs donnés par celle-ci malgré des supports nettement plus compacts. Dans l'optique d'expliquer la résistance aux antibiotiques plus que de la prédire il n'est pas nécessairement intéressant de limiter l'amplitude des coefficients, en revanche le nombre de variables à considérer reste un critère majeur.

4 Conclusion

Blabla

5 Annexes

stabsel_func.R

```
library(car)
library(glmnet)
library(pROC)
library(purrr)


## ---- eval_func ----------------------------------------------------------
# retourne un modèle de regression logistique entrainé
get_glm <- function(X_train, y_train, vars_idx) {
  data <- cbind(as.data.frame(as.matrix(X_train[, vars_idx])),
                data.frame(y = y_train))
  colnames(data) <- c(vars_idx, "y")
  return(glm("y ~ .", data = data, family = "binomial"))
}

# retourne les probabilités des classes
get_predictions <- function(model, X_test, vars_idx) {
  data <- as.data.frame(as.matrix(X_test[, vars_idx]))
  colnames(data) <- vars_idx
  return(predict(model, newdata = data, type = "response"))
}

# retourne le taux d'agrément
get_score <- function(y_test, y_pred) {
  return(sum(y_test == y_pred) / length(y_test) * 100)
}

# estime les classes
# retourne le taux d'agrément
get_performance <- function(model, X_test, y_test, vars_idx) {
  y_pred <- as.integer(get_predictions(model, X_test, vars_idx) > .5)
  return(get_score(y_test, y_pred))
}

# retourne les coefficients (sans l'intercept)
get_coef <- function(model) {
  coefs <- model$coefficients
  coefs <- coefs[setdiff(names(coefs), "(Intercept)")]
```

```
  return(coefs)
}

# entraine un modèle de régression logistique non pénalisé (avec un jeu
#   restreint aux variables stables)
# estime les probabilités des classes
# retourne la courbe ROC
get_roc <- function(X_train, y_train, X_test, y_test, vars_idx) {
  model <- get_glm(X_train, y_train, vars_idx)
  y_pred <- get_predictions(model, X_test, vars_idx)
  return(roc(y_test, y_pred))
}
get_rocs <- function(X_train, y_train, X_test, y_test, mods_summary) {
  mods_roc <- lapply(mods_summary$vars.idx,
                     function(vars) get_roc(X_train, y_train, X_test, y_test,
                                            vars))
  names(mods_roc) <- rownames(mods_summary)
  return(mods_roc)
}

# à partir d'une matrice de coefficients (chemins) retourne un data.frame
get_path_as_df <- function(path, colnames) {
  vars <- which(path != 0, arr.ind = TRUE)
  path_df <- data.frame(cbind(path[vars], vars))
  colnames(path_df) <- colnames
  path_df$lambda <- as.integer(path_df$lambda)
  return(path_df)
}


## ---- stab_sel_func ------------------------------------------------------
# retourne une matrice de 0 / 1 indiquant si le coefficient d'une variable
# est non nul pour un lambda donné
get_selected_vars <- function(X_train, y_train, lambda, sample_size_coef) {
  full_size <- nrow(X_train)
  fold_idx <- sample(full_size, full_size * sample_size_coef)
  X_fold <- X_train[fold_idx, ]
  y_fold <- y_train[fold_idx]
  return(glmnet(X_fold, y_fold, family = "binomial", lambda = lambda)$beta != 0)
}

# retourne un chemin de stabilité
get_stability_path <- function(X_train, y_train, lambda, n_models = 100,
                               sample_size_coef = .5) {
  # liste de n_models matrices de 0 / 1 indiquant la sélection ou non d'une
  # variable pour un lambda donné
  vars_select <- sapply(1:n_models,
                        function(i) get_selected_vars(X_train, y_train, lambda,
                                                      sample_size_coef))
  # passage à une matrice de fréquences des variables
  # (pour des lambda différents)
  return(reduce(vars_select, `+`) / n_models)
}
```

```r
# retourne un data.frame donnant le score et la taille du support selon le
# seuil de stabilité
run_stability_selection_model <- function(X_train, y_train, X_test, y_test,
                                          stability_path) {
  # on garde la fréquence max sur les différents lambdas de chaque variable
  vars_max_freq <- apply(stability_path, 1, max)
  stability_indices <- seq(.6, 1, by = .05)
  names(stability_indices) <- stability_indices
  # index des variables stables pour différents seuils
  vars_idx <- lapply(stability_indices,
                     function(s_idx) which(vars_max_freq >= s_idx))
  models <- lapply(vars_idx,
                   function(v_idx) get_glm(X_train, y_train, v_idx))
  # scores des régressions logistiques sans pénalisation associés aux
  # différents seuils
  scores <- sapply(1:length(vars_idx),
                   function(i) get_performance(models[[i]], X_test, y_test,
                                               vars_idx[[i]]))
  # coefficients
  coefs <- lapply(models, function(model) get_coef(model))
  # NA coefs
  na_coefs <- lapply(coefs, is.na)
  # MAJ coefs, vars
  vars_idx <- lapply(1:length(vars_idx),
                     function(i) vars_idx[[i]][!na_coefs[[i]]])
  coefs <- lapply(1:length(coefs), function(i) coefs[[i]][!na_coefs[[i]]])
  coefs_p_val <- lapply(vars_idx, function(v_idx) {
    p_val <- Anova(get_glm(X_train, y_train, v_idx))$`Pr(>Chisq)`
    names(p_val) <- names(v_idx)
    return(p_val)
  })
  nzero <- sapply(vars_idx, length)
  return(data.frame(indice = stability_indices, score = scores, nzero = nzero,
                    vars.idx = I(vars_idx), coefs = I(coefs),
                    coefs.p.val = I(coefs_p_val)))
}


## ---- cv_func ----------------------------------------------------------------
# retourne un data.frame donnant le score et la taille du support selon le
# seuil de stabilité pour n_folds modèles
cv_run_stability_selection_model <- function(X_train, y_train, lambda,
                                             n_folds = 10) {
  # assignation de chaque observation à un fold
  folds_id <- sample(rep(seq(n_folds), length = nrow(X_train)))
  cv_stability_indices_summary <- lapply(1:n_folds, function(fid) {
    whichs <- folds_id == fid
    # restriction du jeu d'entrainement aux observations n'appartenant pas au
    # fold fid
    fold_X_train <- X_train[! whichs, ]
    fold_y_train <- y_train[! whichs]
    # et du jeu de test à celles appartenant à fid
    fold_X_test <- X_train[whichs, ]
```

```
    fold_y_test <- y_train[whichs]
    # appel à la fonction générique
    stability_path <- get_stability_path(fold_X_train, fold_y_train, lambda)
    return(run_stability_selection_model(fold_X_train, fold_y_train,
                                         fold_X_test, fold_y_test,
                                         stability_path))
  })
  return(do.call(rbind, cv_stability_indices_summary))
}

# à partir d'un data.frame contenant les résultats de la cross-validation
# retourne les indices optimaux
cv_get_optimal_stability_indices <- function(stability_indices_summary) {
  stability_indices <- unique(stability_indices_summary$indice)
  mean_sd <- lapply(stability_indices, function(sid) {
    scores <-
      stability_indices_summary$score[stability_indices_summary$indice == sid]
    return(data.frame(mean = mean(100 - scores), sd = sd(100 - scores)))
  })
  mean_sd <- do.call(rbind, mean_sd)
  mean_sd$indice <- stability_indices
  # indice de stabilité pour la meilleure classification
  indice_min <- max(mean_sd$indice[which(mean_sd$mean == min(mean_sd$mean))])
  # indice de stabilité pour le meilleur
  # compromis classification / taille du support
  indice_1sd <- max(mean_sd$indice[which(mean_sd$mean <=
                                           mean_sd$mean[[indice_min]] +
                                             mean_sd$sd[[indice_min]])])
  return(list(indice.min = indice_min, indice.1sd = indice_1sd))
}
```

stabsel_data.R

```
load("data/TB.Rdata")
# transformation de la réponse en 0 / 1
y.train <- (y.train + 1) / 2
y.test <- (y.test + 1) / 2
```

stabsel_run.R

```
## ---- stabsel_func ------------------------------------------------------------
source("stabsel_func.R")


## ---- stabsel_data ------------------------------------------------------------
source("stabsel_data.R")


## ---- lasso -------------------------------------------------------------------
lasso_mod <- cv.glmnet(X.train, y.train, type.measure = "class",
                       family = "binomial")
lasso_lambda <- lasso_mod$lambda
lasso_pred <- predict(lasso_mod, newx = X.test, type = "class",
```

```
                      s = lasso_mod$lambda.1se)[, 1]
lasso_vars_idx <- which(lasso_mod$glmnet.fit$beta[,
                          which(lasso_mod$lambda == lasso_mod$lambda.1se)] != 0)
lasso_coefs <- get_coef(X.train, y.train, lasso_vars_idx)
# non NA
lasso_na_coefs <- is.na(lasso_coefs)
# MAJ coefs, vars
lasso_vars_idx <- lasso_vars_idx[!lasso_na_coefs]
lasso_coefs <- lasso_coefs[!lasso_na_coefs]
lasso_coefs_p_val <- Anova(get_glm(X.train, y.train,
                                   lasso_vars_idx))$`Pr(>Chisq)`
names(lasso_coefs_p_val) <- names(lasso_vars_idx)
lasso_summary <- data.frame(indice = NA,
                            score = get_score(y.test, lasso_pred),
                            nzero = length(lasso_vars_idx),
                            vars.idx = I(list(lasso_vars_idx)),
                            coefs = I(list(lasso_coefs)),
                            coefs.p.val = I(list(lasso_coefs_p_val)))
rownames(lasso_summary) <- "Lasso"


## ---- stab_sel ----------------------------------------------------------
stability_path <- get_stability_path(X.train, y.train, lasso_lambda)
stab_sel_summary <- run_stability_selection_model(X.train, y.train, X.test,
                                                  y.test, stability_path)
rownames(stab_sel_summary) <- paste0("StabSel-", round(stab_sel_summary$indice,
                                                       digits = 2))


## ---- cv_stab_sel -------------------------------------------------------
cv_stab_sel_summary <- cv_run_stability_selection_model(X.train, y.train,
                                                        lasso_lambda)
best_indices <- cv_get_optimal_stability_indices(cv_stab_sel_summary)
indice_min_idx <- which(stab_sel_summary$indice == best_indices$indice.min)
indice_1sd_idx <- which(stab_sel_summary$indice == best_indices$indice.1sd)
rownames(stab_sel_summary)[[indice_min_idx]] <-
  paste0(rownames(stab_sel_summary)[[indice_min_idx]], " (cv_min)")
rownames(stab_sel_summary)[[indice_1sd_idx]] <-
  paste0(rownames(stab_sel_summary)[[indice_1sd_idx]], " (cv_1sd)")


## ---- paths -------------------------------------------------------------
# Construction des data.frames des chemins
# De régularisation
reg_path <- get_path_as_df(lasso_mod$glmnet.fit$beta,
                           c("beta", "var", "lambda"))

## De sélection
stab_path <- get_path_as_df(stability_path, c("freq", "var", "lambda"))

# Jointure avec les fréquences/probabilités de sélection maximales pour la
# coloration
max_freq <- by(stab_path, list(stab_path$var),
```

```
              function(df) data.frame(var = unique(df$var),
                                      max_freq = max(df$freq)))
max_freq <- as.data.frame(do.call(rbind, max_freq))
max_freq$var <- as.factor(max_freq$var)
max_freq$max_freq <- as.numeric(max_freq$max_freq)
reg_path <- merge(reg_path, max_freq, by = "var", all.x = TRUE)
reg_path[is.na(reg_path$max_freq), "max_freq"] <- 0
stab_path <- merge(stab_path, max_freq, by = "var")

# cast en factor avec un ordre de levels precis pour que les chemins des
# variables les plus stables apparaissent biens
reg_path$var <- factor(reg_path$var,
                       levels = unique(reg_path[order(reg_path$max_freq),
                                                "var"]))
stab_path$var <- factor(stab_path$var,
                        levels = unique(stab_path[order(stab_path$max_freq),
                                                  "var"]))


## ---- save ------------------------------------------------------------------
save(lasso_summary, stab_sel_summary, file = "data/summaries.Rdata")
save(reg_path, stab_path, file = "data/paths.Rdata")
```

stabsel_fig.R

```
## ---- stabsel_func ----------------------------------------------------------
source("stabsel_func.R")


library(ggmosaic)
library(ggplot2)


## ---- fig_paths -------------------------------------------------------------
get_fig_stab <- function(stab_path) {
  # meme couleur de .8 à 1
  stab_path$color <- sapply(stab_path$max_freq, function(f) min(f, .8))
  ggplot(data = stab_path) +
    geom_line(aes(x = lambda, y = freq, group = var, color = color)) +
    scale_color_gradient(low = "yellow", high = "red", limits = c(.6, .8),
                         guide = "none") +
    labs(x = "Lambda index", y = "Probabilité de sélection") +
    theme_bw()
}

get_fig_reg <- function(reg_path) {
  # meme couleur de .8 à 1
  reg_path$color <- sapply(reg_path$max_freq, function(f) min(f, .8))
  ggplot(data = reg_path) +
    geom_line(aes(x = lambda, y = beta, group = var, color = color)) +
    scale_color_gradient(low = "yellow", high = "red", limits = c(0.6, .8),
                         guide = "none") +
    labs(x = "Lambda index", y = "Coefficient") +
```

```
    theme_bw()
}


## ---- fig_roc ---------------------------------------------------------------
get_fig_roc <- function(mods_roc) {
  ggroc(mods_roc) +
    labs(x = "Spécificité", y = "Sensibilité") +
    scale_color_discrete(name = "Modèle") +
    theme_bw()
}


## ---- fig_score -------------------------------------------------------------
get_fig_nz_score <- function(mods_summary) {
  ggplot(data = mods_summary) +
    geom_point(aes(x = nzero, y = score, colour = rownames(mods_summary),
                   shape = rownames(mods_summary))) +
    scale_shape_manual(name = "Modèle", values = 1:nrow(mods_summary)) +
    scale_color_discrete(name = "Modèle") +
    labs(x = "Taille du support", y = "Taux d'agrément") +
    theme_bw()
}


## ---- fig_coef --------------------------------------------------------------
get_coefs_df <- function(mods_summary, signif_level) {
  df <-  lapply(rownames(mods_summary), function(mod_name) {
    r <- mods_summary[mod_name, ]
    coef <- unlist(r$coefs)
    vars_idx <- unlist(r$vars.idx)
    if (!is.null(signif_level)) {
      idx <- unlist(r$coefs.p.val) <= signif_level
      coef <- coef[idx]
      vars_idx <- vars_idx[idx]
    }
    f <- data.frame(model = mod_name, coef = coef, vars.idx = vars_idx)
    rownames(f) <- NULL
    return(f)
  })
  df <- do.call(rbind, df)
  return(df)
}

get_fig_coefs <- function(mods_summary, signif_level = NULL) {
  df <- get_coefs_df(mods_summary, signif_level)
  ggplot(data = df) +
    geom_boxplot(aes(y = coef, fill = model)) +
    scale_fill_discrete(name = "Modèle") +
    scale_x_continuous(breaks = NULL) +
    labs(y = "Coefficients", x = "Modèle") +
    theme_bw()
}
```

```r
signif_codes <- c("0.001" = "***", "0.01" = "**", "0.05" = "*")
get_signif_code <- function(signif_level) {
  return(signif_codes[[as.character(signif_level)]])
}

get_fig_signif_coefs <- function(mods_summary, signif_level) {
  get_fig_coefs(mods_summary, signif_level) +
    labs(y = paste0("Coefficients ", get_signif_code(signif_level)))
}


get_fig_prensence <- function(X_train, mods_summary, model_name,
                              signif_level = NULL) {
  coefs_df <- get_coefs_df(mods_summary[model_name, ], signif_level)
  vars_idx <- coefs_df$vars.idx
  coefs <- coefs_df$coef
  names(coefs) <- vars_idx
  coefs <- sort(abs(coefs))

  df_presence <- as.data.frame(t(apply(X_train[, vars_idx], 2, table)))
  df_presence$var <- factor(vars_idx, ordered = T, levels = names(coefs))
  df_presence$coef <- coefs
  presents <- df_presence[, c("1", "var", "coef")]
  colnames(presents)[[1]] <- "n"
  presents$presence <- 1
  absents <- df_presence[, c("0", "var", "coef")]
  colnames(absents)[[1]] <- "n"
  absents$presence <- 0
  df_presence <- rbind(presents, absents)
  df_presence$presence <- as.factor(df_presence$presence)
  levels(df_presence$presence) <- c("Non", "Oui")

  colors <- c("TRUE" = "green", "FALSE" = "red")
  coefs_col <- sapply(coefs_df$coef,
                      function(i) return(colors[[as.character(i < 0)]]))
  ggplot(data = df_presence) +
    geom_bar(aes(x = var, y = n, fill = presence), stat = "identity") +
    labs(x = "Valeur absolue du coefficient du motif génomique",
         y = "n souches") +
    scale_fill_discrete(name = "Présence") +
    scale_x_discrete(labels = round(coefs, digits = 2)) +
    theme_bw() +
    theme(axis.text.x = element_text(angle = 90, hjust = .5, vjust = .5,
                                     color = coefs_col))
}


get_mosaic <- function(X_train, y_train, var_idx, coef, mod_name) {
  df <- data.frame(x = as.factor(X_train[, var_idx]), y = as.factor(y_train))
  levels(df$x) <- c("Absent", "Présent")
  levels(df$y) <- c("Sensible", "Résistante")
  ggplot(data = df) +
    geom_mosaic(aes(x = product(y, x), fill = y)) +
```

```
    labs(title = paste0(mod_name, " : ",
                        " motif ", var_idx,
                        ", beta = ", round(coef, 2)),
         x = "Motif génomique", y = "Souche") +
    scale_fill_discrete(name = "Souche") +
    theme_bw() +
    theme(axis.text.y = element_text(angle = 90, hjust = .5, vjust = .5),
          plot.title = element_text(size = 10))
}
```

Références

[1] Peter Bühlmann Nicolai Meinshausen. "Stability selection". In: *Journal of the Royal Statistical Society, Series B* 72.4 (2010), pp. 417–473. DOI: https://doi.org/10.48550/arXiv.0809.2932.

[2] Robert Tibshirani Jerome Friedman Trevor Hastie. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22. URL: https://www.jstatsoft.org/v33/i01/.

[3] Robert Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society, Series B* 58.1 (1996), pp. 267–288. DOI: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[4] James J Davis Sébastien Boisvert Thomas Brettin Ronald W Kenyon Chunhong Mao Robert Olson Ross Overbeek John Santerre Maulik Shukla Alice R Wattam Rebecca Will Fangfang Xia Rick Stevens. "Antimicrobial resistance prediction in PATRIC and RAST". In: *Scientific Reports* 6.27930 (2016). DOI: https://doi.org/10.1038/srep27930.