Devoir ACP et valeurs manquantes, Novembre 2018

Aubin de Belleroche, Camille Moatti

01 Novembre 2018

Contents

Introduction

Les données issues du monde réel sont souvent moins faciles à manipuler que celles formatées pour les salles de classe.
L'analyste doit fréquemment remanier les données afin d'obtenir un enregistrement par ligne et une variable par colonne (*tidy data*) ainsi que reformater les données (paramètres régionaux, formats des dates, codage des catégories...). Très souvent, les jeux de données contiennent également des valeurs manquantes. Cela peut-être dû à une multitude de facteurs :

- défaillance matérielle (capteur en panne par exemple)
- nouvelle variable apparue à une certaine date (valeurs manquantes pour les dates antérieures)
- sondage auxquel les sondés n'ont répondu que partiellement
- individu faisant parti d'une cohorte n'étant plus en mesure de répondre...

A chaque fois, l'analyste devra déterminer quels sont les raisons de la présence de ces valeurs manquantes avant de trouver une solution pour y remédier.
Il n'existe malheureusement pas de manière unique de traiter les valeurs manquantes. Plusieurs techniques pourront être essayés afin de trouver la meilleure solution dans un cas donné. Le but de l'imputation est donc de compléter un jeu de données sans changer la structure des données réelles. En effet, une mauvaise imputation ou l'abandon de certaines observations peut conduire à des analyses biasées.

Présentation du jeu et analyse exploratoire

Nous avons choisi d'analyser un jeu qui provient de la banque mondiale. L'institution financière internationale, dont le rôle est d'accorder des prêts afin d'encourager le développement, dispose d'un site permettant de requêter des données sur l'ensemble des pays. Nous avons choisi d'analyser les indicateurs les plus populaires (*Popular Indicators*). Le détail des variables ainsi que leurs sources sont disponibles dans le fichier `des.xlsx` à la racine du projet.

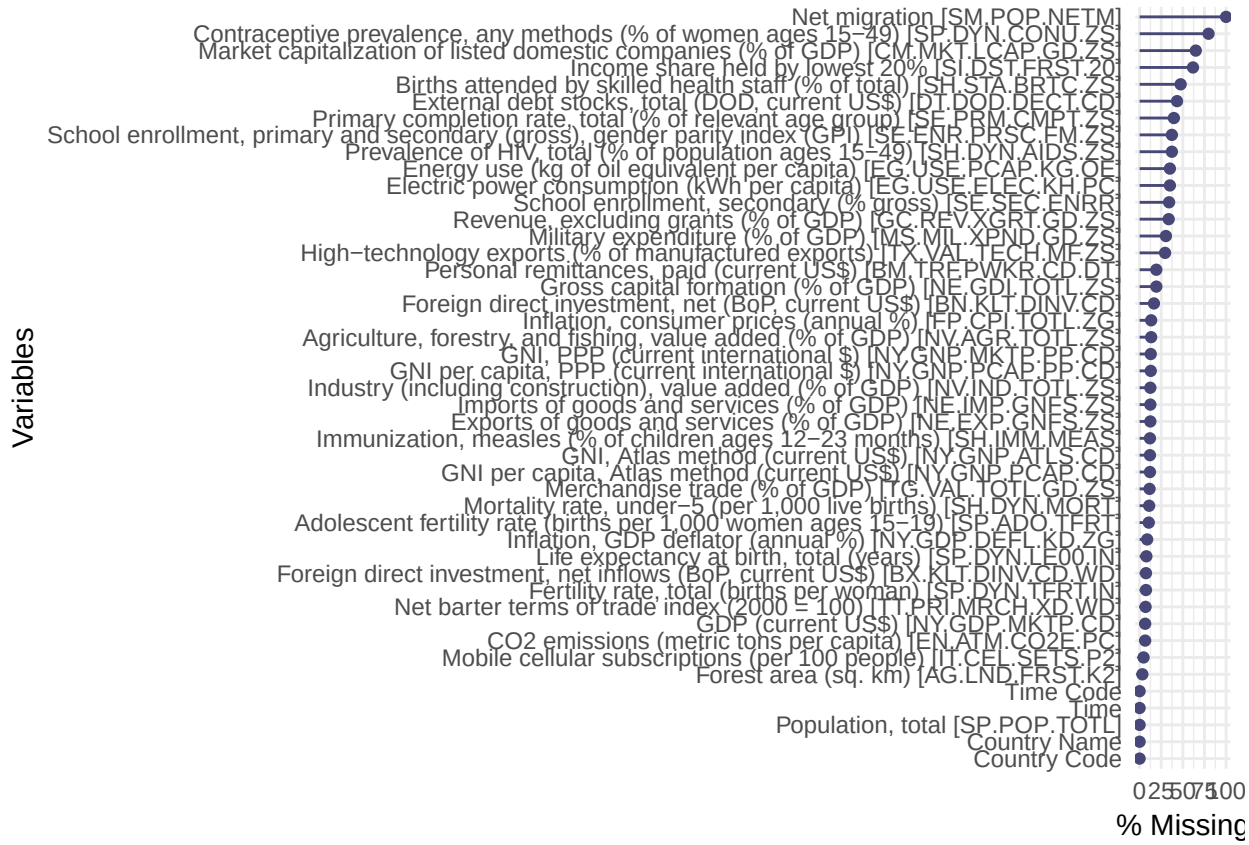
Le but final de notre étude sera d'implémenter un modèle pour expliquer et prévoir le nombre d'enfants par femme dans un pays donné.

Nous chargeons les *packages* nécessaires à l'analyse :

Code R masqué.

Après avoir chargé les données nous observons la répartition des valeurs manquantes en leur sein.

```
raw_data2 <- readxl::read_xlsx("../data/Popular Indicators.xlsx", na="..") %>% filter(Time==2010)
raw_data2 %>%
  summarise_all(funs(sum(is.na(.)))) / nrow(raw_data2) * 100 -> missing_values_pct
gg_miss_var(raw_data2, show_pct = TRUE)
```



Nous remarquons qu'un certain nombre de variables ont un pourcentage de valeurs manquantes très élevé. Etant donné le peu d'individus présents dans l'échantillon, nous avons considéré que les variables présentant un taux de valeurs manquantes supérieur à 20% ne présentaient pas assez de données pour imputer sans risque d'abimer le modèle. Les colonnes correspondantes ont donc été supprimées.

Nous abandonnons également les enregistrements sur lesquels notre *target variable* n'a pas de valeur. La variable `data clean` contient donc un `Tible` qui sera utilisé comme point de départ des imputations.

```
missing_values_pct <- missing_values_pct %>% gather() %>% arrange(-value)
inf_20_pct<-missing_values_pct %>% filter(value<=20)
data_limited_missing <- raw_data2 %>% select(inf_20_pct$key)
data_clean <- data_limited_missing %>%
  drop_na(`Fertility rate, total (births per woman) [SP.DYN.TFRT.IN]`)
```

Nous avons également choisi de supprimer les colonnes présentant des mesures de PIB par pays. En effet, nous disposons également du PIB par individu et de la population pour chaque pays, le PIB global n'apporte donc pas plus d'information.

```
data_clean %<>% select(-c(`GNI, PPP (current international $) [NY.GNP.MKTP.PP.CD]`,
                          `GNI, Atlas method (current US$) [NY.GNP.ATLS.CD]`,
                          `GDP (current US$) [NY.GDP.MKTP.CD]`))
```

Afin de faciliter la lecture dans la suite de notre analyse, nous renomons les différentes colonnes par des

lettres et conservons la correspondance dans un dictionnaire (Annexe 1).

```
saved_names <- names(data_clean)
LETTERS702 <- c(LETTERS, sapply(LETTERS, function(x) paste0(x, LETTERS)))
names(data_clean) <- LETTERS702[1:dim(data_clean)[2]]

letters_dict <- saved_names
names(letters_dict) <- LETTERS702[1:dim(data_clean)[2]]
```

La méthode de l'imputation multiple sera privilégiée tout au long de ce devoir afin de pouvoir estimer l'incertitude associée à notre imputation.

Imputation des données manquantes et choix du modèle

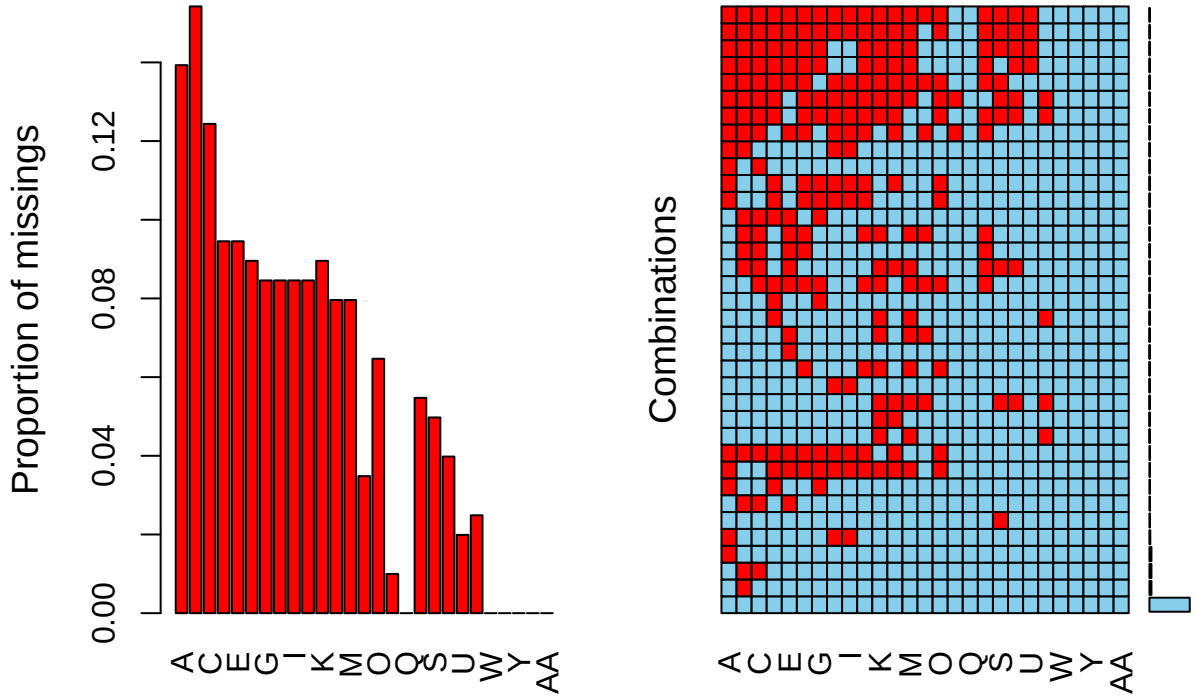
Détermination du mécanisme

Afin de choisir correctement la méthode d'imputation des données manquantes, il convient d'en déterminer le mécanisme. On considère 3 cas possibles (Little & Rubin) :

- MCAR (missing completely at random) si probabilité d'absence est la même pour toutes les observations, c'est à dire qu'elle ne dépend que de paramètres extérieurs;
- MAR (missing at random) lorsque la probabilité d'absence est liée à une ou plusieurs autres données observées;
- MNAR (Missing not at random) si les données observées ne suffisent pas à expliquer les données manquantes, c'est à dire que ces dernières dépendent également des données manquantes.

Afin d'analyser la répartition des valeurs manquantes, nous réalisons le graphique suivant à l'aide du package VIM

```
VIM::aggr(data_clean)
```



Nous en déduisons qu'il ne semble pas exister de schéma spécifique de répartion de nos données manquantes. En effet, aucune répartition de ces dernières en fonction des variables n'apparaît avec une fréquence sensiblement supérieure aux autres. En revanche la répartition des valeurs manquantes selon les variables est très inégale, le cas le cas MCAR semble donc exclu dans notre échantillon.

La distinction entre les cas MAR et MNAR n'est ici pas possible sans connaître la façon dont les données ont été collectées. Par défaut nous supposerons un mécanisme MAR et vérifierons la robustesse de ce choix à postériori.

Imputation et ACP

Dans un premier temps, nous implémentons l'algorithme de l'ACP itératif afin de réaliser une imputation multiple. L'explication détaillée de l'algorithme est disponible sur YouTube.

Le *chunk* suivant permet de régler quelques détails, notamment la conversion en `data.frame` base R et le centrage réduction des données.
Nous enlevons la variable cible afin d'éviter le **surapprentissage**.

```
data_clean_numeric <- data_clean %>%
  select(which(sapply(.,is.numeric))) %>%
  as.data.frame(.)

data_clean_numeric <- scale(data_clean_numeric) %>% as.data.frame(.)

target <- data_clean_numeric$Q
data_clean_numeric %<>% select(-Q)
```

Puis nous utilisons l'algorithme avec 10 *resampling* via *Parametric Bootstrap (Josse, J., Husson, F. (2010))*:

```
nbdim <- estim_ncpPCA(data_clean_numeric)
res.comp <- MIPCA(data_clean_numeric, ncp = nbdim$ncp, scale=TRUE, nboot = 10)
imp<-prelim(res.comp, data_clean_numeric)
```

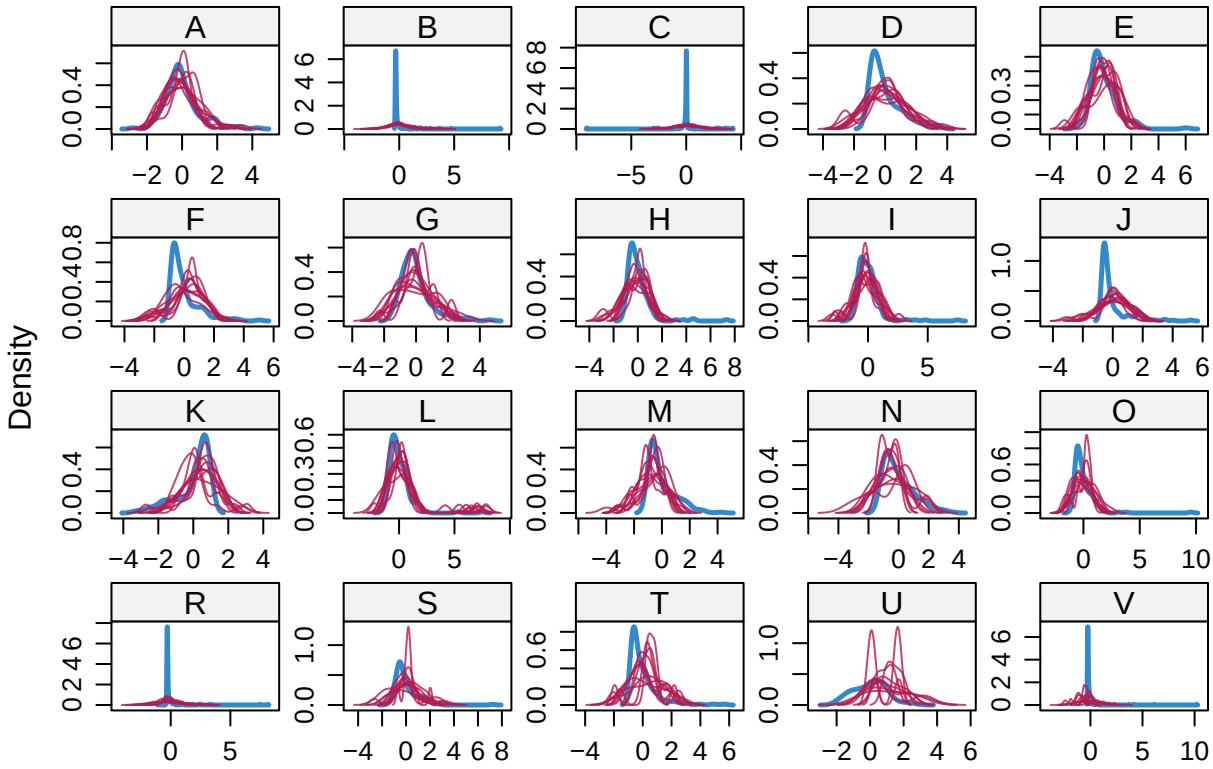
La fonction `prelim` permet d'obtenir un objet de type `mids` et d'utiliser les capacités graphiques du package `mice`. Nous commençons par les graphiques prévus par MissMDA :

```
plot(res.comp)
```

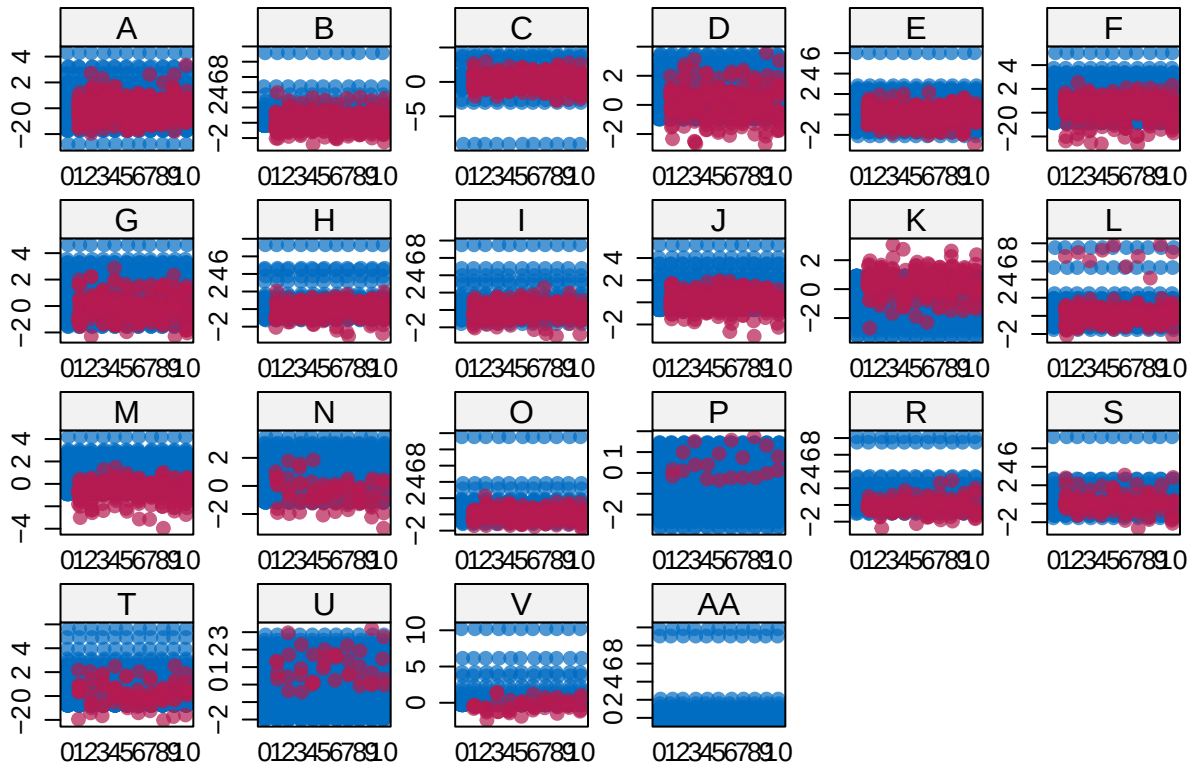
Les graphiques montrent que les axes sont stables entre les n imputations. De plus, les valeurs imputées ont une variabilité plutôt limitée. Nous pouvons noter que les variables A et C sont mal représentées.

Puis nous utilisons les fonctions `densityplot` et `stripplot` du *package* `mice` :

```
densityplot(imp)
```




```
stripplot(imp, pch = 20, cex = 1.2)
```



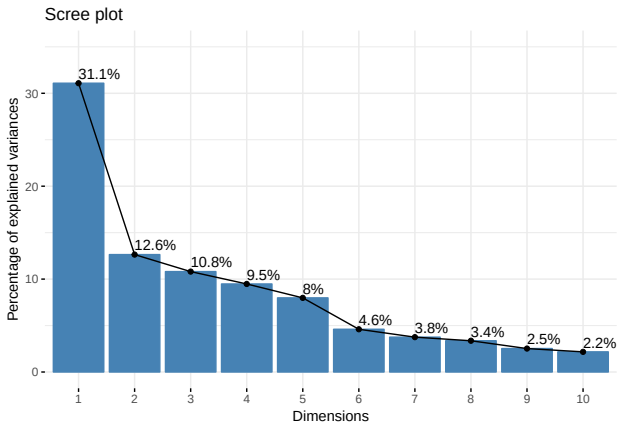
Le `stripplot` permet de bien visualiser la stabilité des valeurs imputées (points rouges) d'une imputation à l'autre. En abscisse, les n imputations sont représentées. En ordonnées, nous avons les valeurs imputées. Pour s'assurer de la robustesse de la procédure d'imputation, il faut que les points rouges soient similaires tout au long de l'axe des abscisses. Pour cette raison, les valeurs imputées sur les variables P et U ne sont pas satisfaisantes. Cependant, nous ne les enlevons pas à ce stade.

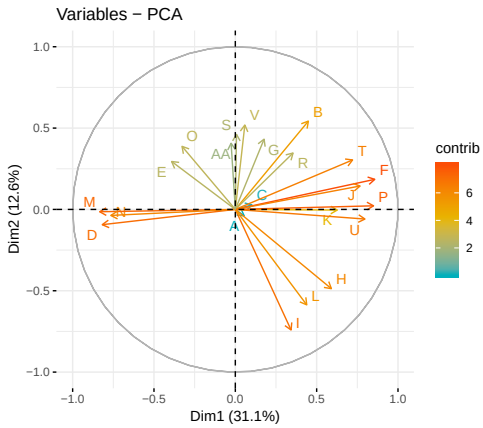
Afin de continuer vers la modélisation, nous choisissons dans un premier temps de réduire le nombre de dimensions :

```
data_imputed <- res.comp[["res.imputePCA"]]

res<-PCA(data_imputed, graph = F)
```

Même si la première dimension domine largement les autres, il semble opportun de garder quatre dimensions supplémentaires (72% de variance conservée), en effet la première ne modélise que 31,1% de l'inertie de notre jeu de donnée. On constate également un coude après la cinquième dimension, ce qui nous conforte dans notre choix.

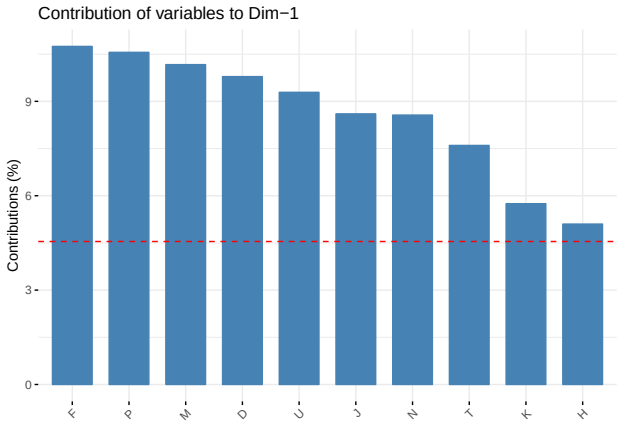
```
fviz_screeplot(res, addlabels = TRUE, ylim = c(0, 35))

fviz_pca_var(res, col.var="contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE # Avoid text overlapping
             )
```

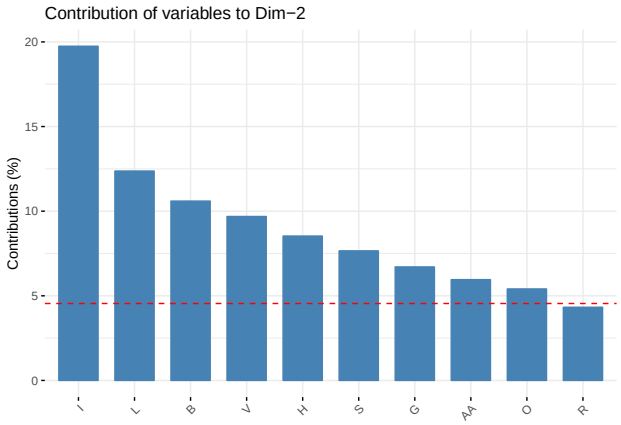




Nous notons que, hormis A et C, les variables contribuent toutes correctement aux premier plan, certaines bien plus que d'autres. Il est apparent que M, N et D contribuent à l'axe 1 très négativement et qu'à l'inverse T, J, P, F, U et K ont un impact très positif. Ces variables n'ont en revanche que peu d'impact sur l'axe 2 à l'inverse de I, L et H qui contribuent négativent à ce dernier et positivement au premier. Enfin un le groupe restant a une plus faible contribution positive à l'axe 2 tout en étant réparti sur l'axe 1.

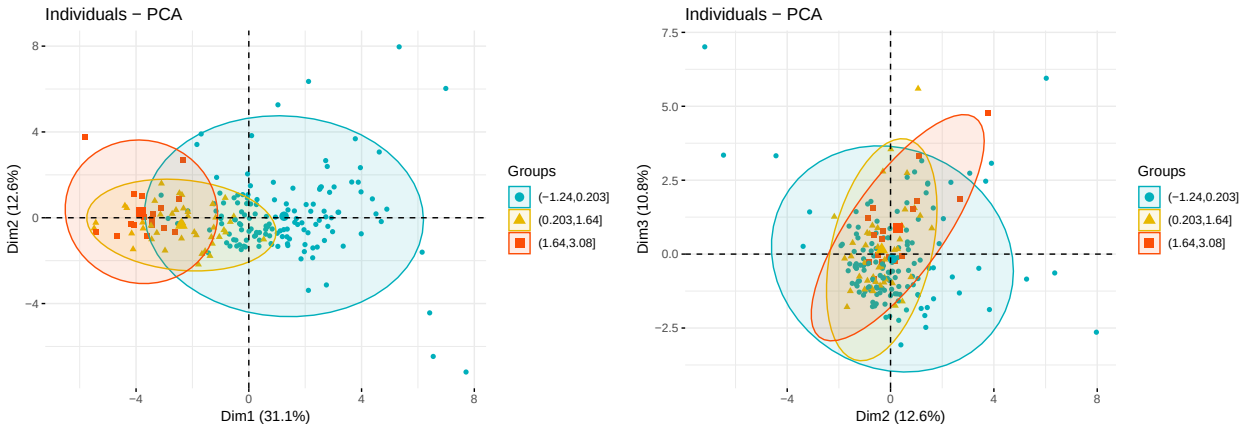
```
fviz_contrib(res, choice = "var", axes = 1, top = 10)

fviz_contrib(res, choice = "var", axes = 2, top = 10)
```





Nous pouvons d'ores et déjà visualiser la fertilitité sur les deux premiers axes de notre ACP. Pour ce faire, nous créons trois catégorie à partir de la variable continue cible. Ces trois catégories peuvent être interprétées comme des niveaux du nombre d'enfants par femme (élevé, moyen et faible). Pour ce faire, nous utilisons la fonction `cut` :

```
fviz_pca_ind(res,
             label = "none",
             habillage = cut(target,3),
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE)
fviz_pca_ind(res,
             axes=c(2,3),
             label = "none",
             habillage = cut(target,3),
             palette = c("#00AFBB", "#E7B800", "#FC4E07"),
             addEllipses = TRUE)
```



Avec seulement deux dimensions, nous observons qu'un point ayant une valeur positive sur la première dimension appartient presque toujours à la première catégorie (faible fertilité).
Les deux autres catégories, bien que plus proches, permettent d'observer que les pays où les femmes ont beaucoup d'enfants tendent à avoir une valeur très négative sur la première dimension.
Le visualisation de la seconde dimension par rapport à la troisième est plus difficile à interpréter de prime abord.

La fonction `Investigate(res)` du package `FactoInvestigate` permet de générer un rapport automatisé sur notre ACP. Cette analyse peut servir de point de départ afin d'affiner l'interprétation. L'analyse de la première dimension est reproduite ci-dessous :

```
The dimension 1 opposes individuals such as 95, 197, 156 and 194 (to the right of the graph, characteri

The group 1 (characterized by a positive coordinate on the axis) is sharing :

high values for the variables P, U, K, J, F, T and H (variables are sorted from the strongest).
low values for the variables M, N, D, O, E, S and AA (variables are sorted from the weakest).
The group in which the individuals 156 and 194 stand (characterized by a positive coordinate on the axi

high values for variables like B, T, F, J, G, S, P, C, R and V (variables are sorted from the strongest
low values for the variables D, N, M, I and E (variables are sorted from the weakest).
The group in which the individuals 95 and 197 stand (characterized by a positive coordinate on the axis

high values for the variables L, I, H, U, F, R, A and P (variables are sorted from the strongest).
The group 4 (characterized by a negative coordinate on the axis) is sharing :

high values for the variables M, D, N, E and O (variables are sorted from the strongest).
low values for variables like P, U, F, J, T, K, H, B, L and R (variables are sorted from the weakest).
```

L'analyse automatique retient que le groupe qui a des valeurs élevées pour P (*Life expectancy at birth*), U (*Mobile cellular subscriptions (per 100 people)*) et K Immunization, measles (*% of children ages 12-23 months*) a des valeurs basses pour M (*Mortality rate, under-5*) et D (*Agriculture, forestry, and fishing, value added (% of GDP)*). Nous verrons à la fin du devoir que ce sont les variables qui apparaissent pertinentes lors d'une modélisation par régression linéaire sans réduction de la dimension. Pour le groupe 4, cela s'inverse.

Notre première modélisation, effectuée après imputation des valeurs manquantes et réduction de la dimension à cinq axes, permet d'obtenir un R ajusté de 78%.

```
ind_coord <- res[["ind"]][["coord"]] %>% as_tibble()
ind_coord_target <- ind_coord %>% mutate(target = target)
fit <- lm(target~.-1, ind_coord_target)
summary(fit)
```

```
##
## Call:
## lm(formula = target ~ . - 1, data = ind_coord_target)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30183 -0.31774 -0.03584  0.28358  1.35056
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## Dim.1 -0.300997   0.012598 -23.892  < 2e-16 ***
## Dim.2  0.006351   0.019756   0.321 0.748201
## Dim.3  0.163717   0.021364   7.663 8.18e-13 ***
## Dim.4 -0.082369   0.022815  -3.610 0.000388 ***
## Dim.5  0.219716   0.024863   8.837 5.62e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4671 on 196 degrees of freedom
## Multiple R-squared:  0.7862, Adjusted R-squared:  0.7808
## F-statistic: 144.2 on 5 and 196 DF,  p-value: < 2.2e-16
```

Imputation et Algorithme Mice

L'algorithme utilisé dans le package Mice (**M**ultivariate **I**mputation by **C**hained **E**quations) a été dévelopé par Stef van Buuren. Tout comme l'algorithme implémenté dans MissMDA, dévelopé par François Husson, Mice utilise une approche itérative afin d'imputer les valeurs manquantes. L'approche multiple permet de mesurer l'incertitude quant aux valeurs imputées.
Afin d'effectuer notre analyse, nous allons utiliser la fonction `mice()` en variant les méthodes d'imputation. `mice` est une méthode de Monte Carlo par chaines de Markov qui utilise la structure des corrélations pour trouver des valeurs plausibles sur les jeux incomplets.
Nous allons essayer trois méthodes :

- *Predictive Mean Matching (Rubin 1986, Little 1988)* : Cette méthode est la méthode par défaut de la fonction `mice()`. Cependant, aucune théorie mathématique n'a prouvé la pertinence de cette méthode. Toutefois, cette méthode permet d'imputer avec des valeurs empruntées aux autres observations. Par exemple, sur une colonne d'entiers, *pmm* imputera des entiers. Les valeurs imputées seront contenues entre la borne minimum et maximum de la variable. Ces qualités ont rendu cette méthode assez populaire. Cet article explique l'algorithme en détail ainsi que ses limites.
- *Linear Regression Ignoring Model Error* : Cette seconde méthode utilise un modèle linéaire afin d'estimer les valeurs manquantes. Cependant, l'algorithme ajoute un bruit gaussien à la prédiction afin d'eviter l'*overfitting*. Mice propose d'autres algorithmes d'imputations s'appuyant sur le modèle linéaire gaussien. Cette page les explique en détail. Nous utiliserons donc `norm.nob`.
- *Random Forest* : Cet algorithme ensembliste est une amélioration de l'arbre de décision simple. Il est utilisé dans le cadre de l'apprentissage statistique afin de réduire la tendance à l'*overfitting* des arbres. Tout comme les arbres, il peut être utilisé pour prédire des variables continues. Ces méthodes sont robustes aux *outliers*, gèrent la multicollinéarité et les distributions assymétriques. Cela est appréciable dans le cadre de l'imputation. Cette page peut être consultée pour plus de détails.

Tout comme précédemment, nous remanions légèrement notre tableau afin d'assurer sa compatibilité avec `mice`. Grâce à `predictorMatrix`, nous pouvons empêcher la variable `Q` d'être utilisée lors de l'imputation :

```
data_clean_numeric_mice <- data_clean %>%
  select(which(sapply(.,is.numeric)))
data_clean_numeric_mice <- scale(data_clean_numeric_mice) %>%
  as.data.frame(.)
```

```
imp <- mice(data_clean_numeric_mice, print = FALSE)
pred <- imp$predictorMatrix
pred[, "Q"] <- 0
```

Nous imputons avec les trois méthodes décrites précédemment. Dix imputations sont réalisées pour chaque méthode :

```
imputed_pmm = mice(data_clean_numeric_mice, pred = pred, method="pmm", m=10)

imputed_normnob = mice(data_clean_numeric_mice, pred = pred, method="norm.nob", m=10)

imputed_rf = mice(data_clean_numeric_mice, pred = pred,method="rf", m=10)
```

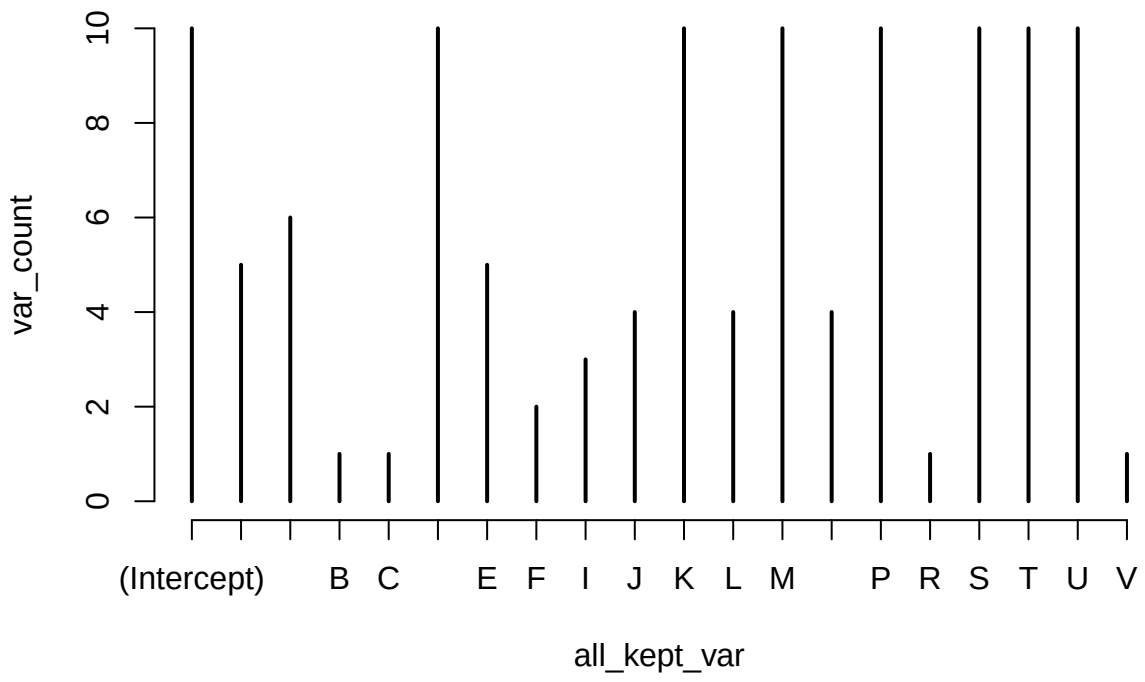
Nous cherchons ensuite à déterminer les variables à conserver dans nos 3 cas à l'aide d'une méthode pas à pas. Pour réaliser cela, nous suivons le mode opératoire décris par Brand (1999) : nous appliquons une méthode pas à pas sur chaque table imputée séparement, puis nous comptons le nombre d'apparition de chaque variable dans les 10 modèles ainsi créés pour ne conserver que celles qui apparaissent dans plus de la moitié dans un "supermodèle" final.

Ces opérations seront réalisées à l'aide des fonctions suivantes

```
step_on_mice <- function (mice_object, direction = "both") {
  fit <- with(data = mice_object, exp = lm(Q ~ A+B+C+D+E+F+G+H+I+J+
                                           K+L+M+O+P+R+S+T+U+V+AA))
  len <- length(fit$analyses)
  all_kept_var <- vector()
  for (i in 1:len) {
    mod_step <- step(fit$analyses[[i]], direction = direction,trace=0)
    kept_var <- names(mod_step[["coefficients"]])
    all_kept_var <- c(all_kept_var, kept_var)
  }
  table(all_kept_var)
}

make_linear_model <- function(mice_object, step_on_mice_res, threshold = 5, intercept=TRUE){
  final_var <- step_on_mice_res[step_on_mice_res > threshold]
  fin_var_name <- names(final_var)
  fin_var_name <- fin_var_name[fin_var_name!= "(Intercept)"]
  if (intercept){
  formula_ <- as.formula(paste("Q", paste(fin_var_name, collapse=" + "), sep=" ~ "))
  } else {
    formula_ <- as.formula(paste("Q", paste(paste(fin_var_name, collapse=" + "), "-1"), sep=" ~ "))
  }
  print(formula_)
  with(data = mice_object, exp=lm(formula(format(formula_))))
}
```

- Dans le cas de la méthode PMM, l'ensemble des variables présentes dans plus de 50% des modèles le sont dans tous, ce sont donc celles que nous retenons pour notre modèle final.

```
var_count <- step_on_mice(imputed_pmm)
plot(var_count)
```




```
fit_selected_var <- make_linear_model(imputed_pmm, var_count)
```

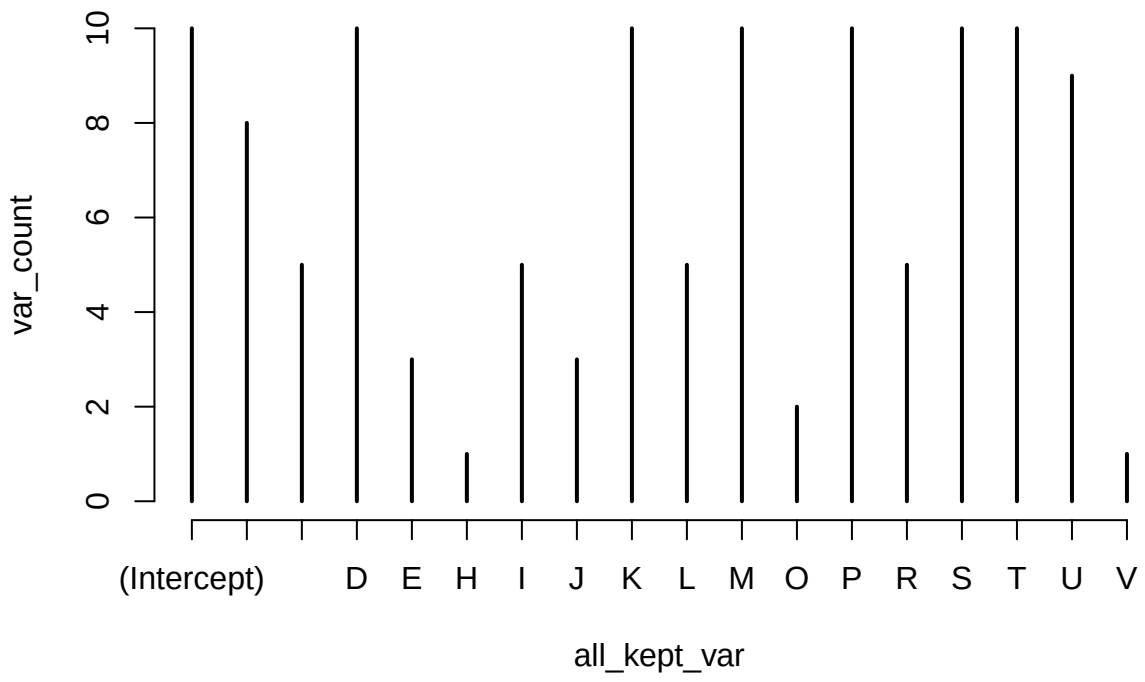
```
## Q ~ AA + D + K + M + P + S + T + U
## <environment: 0x55821cb29488>
```

```
summary(pool(fit_selected_var))
```

```
##                estimate  std.error  statistic       df      p.value
## (Intercept)  0.01840431 0.03353632  0.5487875 185.2385 5.837973e-01
## AA          -0.05848372 0.03337862 -1.7521311 189.4785 8.136894e-02
## D            0.10999496 0.05840070  1.8834528 183.5566 6.117097e-02
## K           -0.13551162 0.04649087 -2.9148008 160.8346 3.987997e-03
## M            0.28570490 0.09646402  2.9617769 175.3080 3.450273e-03
## P           -0.29026347 0.08214276 -3.5336465 179.5781 5.149455e-04
## S            0.16252862 0.04049079  4.0139652 128.5019 8.593893e-05
## T           -0.09423519 0.04558006 -2.0674654 162.8025 4.004930e-02
## U           -0.11169698 0.05072745 -2.2019039 180.8015 2.887922e-02
```

- Pour la méthode utilisant un modèle linéaire, nous retrouvons les mêmes variables que précédement dans les 10 modèles mais 3 autres apparaissent dans plus de la moitié des cas, sans pour autant être systématiquement présentes. Par ailleurs ces mêmes variables apparaissent peu significatives avec des p-value élevées.

```
var_count <- step_on_mice(imputed_normnob)
plot(var_count)
```




```
fit_selected_var <- make_linear_model(imputed_normnob, var_count)
```

```
## Q ~ A + D + K + M + P + S + T + U
## <environment: 0x55821b4d0d38>
```

```
summary(pool(fit_selected_var))
```

```
##                estimate  std.error  statistic       df      p.value
## (Intercept)  0.02749805 0.03429827  0.8017327 178.8289 4.237327e-01
## A           -0.06327181 0.03512513 -1.8013258 167.8430 7.327441e-02
## D            0.11197789 0.06049494  1.8510291 167.2158 6.575411e-02
## K           -0.11021119 0.04859031 -2.2681724 149.7679 2.447026e-02
## M            0.28251536 0.09399297  3.0057074 185.6354 3.016121e-03
## P           -0.31105653 0.08264623 -3.7637112 181.5827 2.244373e-04
## S            0.17352989 0.03807542  4.5575301 183.1804 9.366093e-06
## T           -0.09316151 0.04486557 -2.0764589 183.8227 3.922890e-02
## U           -0.08539838 0.05300528 -1.6111295 145.4963 1.088505e-01
```

Nous réalisons alors des tests afin de vérifier si ces variables sont nécessaires au modèle. Nous testons tout d'abord l'intérêt de la variable A :

```
fit_without <- with(imputed_normnob, lm(Q ~ AA + D + K + L + M + P + S + T + U))
fit_with <- with(imputed_normnob, lm(Q ~ A + AA + D + K + L + M + P + S + T + U))
anova(fit_with, fit_without)
```

```
##    test statistic df1      df2 df.com   p.value        riv
##  1 ~~ 2  2.206525   1 157.6045    190 0.1394254 0.06090468
```

La p-value du test est égale à 0,21, nous considérons donc que A n'est pas nécessaire au modèle. Nous testons

maintenant la pertinence de L :

```
fit_without <- with(imputed_normnob, lm(Q ~ AA + D + K + M + P + S + T + U))
fit_with <- with(imputed_normnob, lm(Q ~ AA + D + K + L + M + P + S + T + U))
anova(fit_with, fit_without)
```

```
##    test   statistic df1      df2 df.com   p.value       riv
##  1 ~~ 2 0.004230924   1 64.64155    191 0.9483385 0.2368071
```

La p-value est toujours importante (0,88), nous ne conservons pas la variable L. Enfin nous vérifions la nécessité de conserver AA :

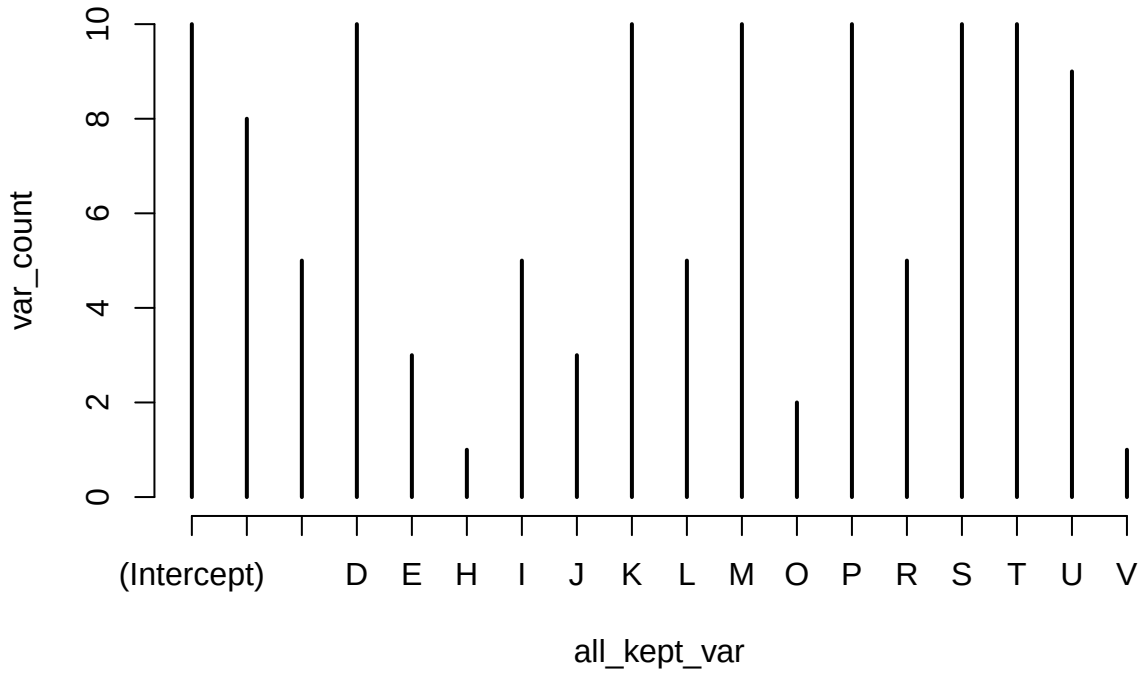
```
fit_without <- with(imputed_normnob, lm(Q ~ D + K + M + P + S + T + U))
fit_with <- with(imputed_normnob, lm(Q ~ AA + D + K + M + P + S + T + U))
anova(fit_with, fit_without)
```

```
##    test statistic df1     df2 df.com    p.value          riv
##  1 ~~ 2  3.081693   1 190.018    192 0.08078945 0.0008530449
```

Cette fois ci la p-value est bien meilleure, à 0,08, nous faisons le choix de conserver la variable.

- Enfin, pour la méthode de Random Forest, nous observons à nouveau des varibles apparaissant plus de 5 fois mais pas à chaque occurence, certaines présentant des p-values élevées :

```
var_count <- step_on_mice(imputed_normnob)
plot(var_count)
```




```
fit_selected_var <- make_linear_model(imputed_normnob, var_count)
```

```
## Q ~ A + D + K + M + P + S + T + U
```

```
## <environment: 0x558219b6a128>
```

```
summary(pool(fit_selected_var))
```

```
##                estimate  std.error  statistic       df      p.value
## (Intercept)  0.02749805 0.03429827  0.8017327 178.8289 4.237327e-01
## A           -0.06327181 0.03512513 -1.8013258 167.8430 7.327441e-02
## D            0.11197789 0.06049494  1.8510291 167.2158 6.575411e-02
## K           -0.11021119 0.04859031 -2.2681724 149.7679 2.447026e-02
## M            0.28251536 0.09399297  3.0057074 185.6354 3.016121e-03
## P           -0.31105653 0.08264623 -3.7637112 181.5827 2.244373e-04
## S            0.17352989 0.03807542  4.5575301 183.1804 9.366093e-06
## T           -0.09316151 0.04486557 -2.0764589 183.8227 3.922890e-02
## U           -0.08539838 0.05300528 -1.6111295 145.4963 1.088505e-01
```

En testant à nouveau le modèle comme précédemment, nous décidons de retirer L et T dont les p-value sont supérieures à 0,1 mais de conserver AA.

```
fit.without <- with(imputed_rf, lm(Q ~ D + K + M + P + S + T + U))
fit.with <- with(imputed_rf, lm(Q ~ AA + D + K + M + P + S + T + U))
anova(fit.with, fit.without)
```

```
##    test statistic df1      df2 df.com    p.value         riv
##  1 ~~ 2   3.30621   1 189.9944    192 0.07059329 0.001568457
```

```
fit.without <- with(imputed_rf, lm(Q ~ AA + K + M + P + S + T + U))
fit.with <- with(imputed_rf, lm(Q ~ AA + D + K + M + P + S + T + U))
anova(fit.with, fit.without)
```

```
##    test statistic df1      df2 df.com   p.value       riv
##  1 ~~ 2  2.386583   1 59.17426    192 0.1277125 0.2596616
```

```
fit.without <- with(imputed_rf, lm(Q ~ AA + D + K + M + P + S + U))
fit.with <- with(imputed_rf, lm(Q ~ AA + D + K + M + P + S + T + U))
anova(fit.with, fit.without)
```

```
##    test statistic df1      df2 df.com   p.value       riv
##  1 ~~ 2  2.540244   1 99.40039    192 0.1141526 0.1464281
```

Au final, avec cette deuxième façon de faire, nous obtenons 3 modèles relativement similaires, incluant systématiquement les variables D, K, M, P, S et U, avec T et AA selon les cas.

Nous en tirons que la fécondité d'un pays dépend positivement de la part de l'agriculture et de la pêche dans le PIB (poids du secteur primaire dans la création de richesse), du taux de mortalité infantile et des termes de l'échange (rapport entre les prix des exportations et les prix des importations). Elle est en revanche négativement affectée par le taux de vaccination infantile à la rubéole (probablement négativement corrélé à la mortalité infantile), l'espérance de vie et le nombre de souscription à des portables.

Ces résultats semblent assez intuitifs, les pays les plus développés et les plus riches sont réputés pour avoir des taux de fécondité plus faibles. En revanche les variables explicatives selectionnées peuvent avoir de quoi surprendre par rapport à d'autres indicateurs à priori plus directs.

Conclusion

Nous avons donc essayé plusieurs méthodes pour gérer les valeurs manquantes. La première méthode s'appuie sur l'ACP pour imputer les variables. L'algorithme Mice, la seconde option, englobe plusieurs sous-méthodes. Nous en avons essayé trois avant de mettre en place une méthode de sélection de variables inspirée par cette réponse de Stef van Buuren.

Pour continuer l'analyse nous pourions par exemple essayer d'évaluer les différentes méthodes d'imputation que propose mice entre elles. Nous pourrions également évaluer missMDA en parralèle. Cependant :

- mice propose de choisir une méthode d'imputation différente pour chaque variable. Il est peu probable qu'une même méthode d'imputation soit efficace pour toutes les colonnes. Nous avons mis en annexe les imputations pour la variable *CO2 Emissions* pour montrer que cette variable doit être imputée par une méthode qui a conscience du *range min max*. La méthode d'imputation par régression est susceptible de créer des émissions de CO2 négatives (les variables ont été centrées et réduites ce qui explique que le minimum soit inférieur à 0 ici).
  Ce travail peut donc s'avérer complexe.
- mice ne propose pas de méthode `predict`. Cela peut être handicapant si l'on souhaite évaluer nos modèles de manière simple avec la RMSE. La fonction est prévue pour une prochaine *release* (cf. ce lien).

Annexes

Annexe 1

```
letters_dict
```

```
##                                                                                A
##                           "Gross capital formation (% of GDP) [NE.GDI.TOTL.ZS]"
##                                                                                B
##                  "Personal remittances, paid (current US$) [BM.TRF.PWKR.CD.DT]"
##                                                                                C
##            "Foreign direct investment, net (BoP, current US$) [BN.KLT.DINV.CD]"
##                                                                                D
##   "Agriculture, forestry, and fishing, value added (% of GDP) [NV.AGR.TOTL.ZS]"
##                                                                                E
##                         "Inflation, consumer prices (annual %) [FP.CPI.TOTL.ZG]"
##                                                                                F
##             "GNI per capita, PPP (current international $) [NY.GNP.PCAP.PP.CD]"
##                                                                                G
##    "Industry (including construction), value added (% of GDP) [NV.IND.TOTL.ZS]"
##                                                                                H
##                       "Exports of goods and services (% of GDP) [NE.EXP.GNFS.ZS]"
##                                                                                I
##                       "Imports of goods and services (% of GDP) [NE.IMP.GNFS.ZS]"
##                                                                                J
##                    "GNI per capita, Atlas method (current US$) [NY.GNP.PCAP.CD]"
##                                                                                K
##         "Immunization, measles (% of children ages 12-23 months) [SH.IMM.MEAS]"
##                                                                                L
##                                "Merchandise trade (% of GDP) [TG.VAL.TOTL.GD.ZS]"
##                                                                                M
##                  "Mortality rate, under-5 (per 1,000 live births) [SH.DYN.MORT]"
##                                                                                N
##   "Adolescent fertility rate (births per 1,000 women ages 15-19) [SP.ADO.TFRT]"
##                                                                                O
##                           "Inflation, GDP deflator (annual %) [NY.GDP.DEFL.KD.ZG]"
##                                                                                P
##                        "Life expectancy at birth, total (years) [SP.DYN.LE00.IN]"
##                                                                                Q
##                       "Fertility rate, total (births per woman) [SP.DYN.TFRT.IN]"
```

```
##                                                                          R
## "Foreign direct investment, net inflows (BoP, current US$) [BX.KLT.DINV.CD.WD]"
##                                                                          S
##              "Net barter terms of trade index (2000 = 100) [TT.PRI.MRCH.XD.WD]"
##                                                                          T
##                       "CO2 emissions (metric tons per capita) [EN.ATM.CO2E.PC]"
##                                                                          U
##               "Mobile cellular subscriptions (per 100 people) [IT.CEL.SETS.P2]"
##                                                                          V
##                                          "Forest area (sq. km) [AG.LND.FRST.K2]"
##                                                                          W
##                                                                     "Time"
##                                                                          X
##                                                                "Time Code"
##                                                                          Y
##                                                              "Country Name"
##                                                                          Z
##                                                              "Country Code"
##                                                                         AA
##                                                "Population, total [SP.POP.TOTL]"
```

Annexe 2

Fonction visualisation des données imputées avec mice :

```
plot_imputed <- function(mice_object, original_dataset, column, plot_type="model", se=F, method="auto"){
  x_imputations <- mice_object[["imp"]][[column]]
  index <- row.names(x_imputations[1])
  ordo <- original_dataset[["Q"]][strtoi(index)]
  imputed_points <- x_imputations %>%
    as_tibble() %>%
    gather() %>%
    mutate(ordo=rep(ordo, length(x_imputations)))
  if (plot_type=="simple") {
    ggplot() +
      geom_point(data = original_dataset, aes_string(y = "Q", x = column)) +
      geom_point(data = imputed_points, aes(y = ordo, x = value), colour="#CC0000", alpha = 0.5) +
      xlab(letters_dict[column]) + ylab("Fertility rate, births per woman")
  } else if (plot_type=="model"){
    ggplot() +
      geom_point(data = original_dataset, aes_string(y = "Q", x = column)) +
      geom_smooth(data = original_dataset, aes_string(y = "Q", x = column), se=se, method=method) +
      geom_point(data = imputed_points, aes(y = ordo, x = value), colour="#CC0000", alpha = 0.5) +
      geom_smooth(data = imputed_points, aes(y = ordo, x = value), colour="#CC0000", alpha = 0.5, se=se
      xlab(letters_dict[column]) + ylab("Fertility rate, births per woman")
  }

}
```

```
plot_imputed(imputed_pmm, data_clean_numeric_mice, "T", method = "auto")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 8 rows containing non-finite values (stat_smooth).
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 8 rows containing missing values (geom_point).
```

```
plot_imputed(imputed_normnob, data_clean_numeric_mice, "T", method = "auto")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## Warning: Removed 8 rows containing non-finite values (stat_smooth).
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## Warning: Removed 8 rows containing missing values (geom_point).
```