R パッケージ midr の紹介

関数分解手法 MID の活用による予測モデルの可視化と解釈

上妻玲兒* 浅芝良一† 岩沢宏和‡

目次

1 はじめに

本稿は，2024 年度の研究発表大会で発表された岩沢・松森論文「ブラックボックスモデルの解釈を最大化する関数分解手法」(1) に関連して，その中で提案された手法である **MID**（Maximum Interpretation Decomposition）の計算を実装した R パッケージ **midr** の基本的な機能を紹介するものです.

同大会では，この発表のほかに，パネルディスカッション「機械学習技術のアクチュアリー実務への活用に

* rktkdm@gmail.com
† ryoichi.asashiba@gmail.com
‡ iwahiro@bb.mbn.or.jp

関する展望」においても，機械学習モデルを解釈する **IML**（Interpretable Machine Learning）手法の必要性が議論され，特に予測モデルの理論的な説明力やモデル構築の再現性などを向上させる目的で IML 手法を利用できる可能性が指摘されました．本稿の著者を含む MID の研究・開発チームは，MID および midr が手軽な IML 手法として多くの実務家に利用され，保険・年金分野における機械学習技術の普及に寄与するものとなることを願っています．

MID の実装に関する研究は，日本アクチュアリー会のデータサイエンス関連基礎調査ワーキンググループにおける勉強会での議論をきっかけに，岩沢・松森論文で紹介された理論研究の成果を広く利用可能なものにすることを目的として，当初は浅芝と岩沢の共同プロジェクトとして始まり，その後上妻をメンバーとして迎え，プロジェクトとして本格化しました．midr の実装の質に関する責任は浅芝が負っていますが，その理論的な土台は岩沢が松森氏とともに進めてきた研究の成果に基づいています．

1.1 MID とは

予測関数 f の**関数分解**とは，f を以下のように加法的に分解したもののことをいいます[*1]．

$$
\begin{aligned}
f(x_1, \cdots, x_d) = f_\emptyset & && \cdots\text{定数項} \\
+ f_1(x_1) + \cdots + f_d(x_d) & && \cdots\text{主効果（1 変数関数）} \\
+ f_{1,2}(x_1, x_2) + f_{1,3}(x_1, x_3) \cdots + f_{d-1,d}(x_{d-1}, x_d) & && \cdots\text{交互作用（2 変数関数）} \\
+ \epsilon(x_1, \cdots, x_d) & && \cdots\text{残差}
\end{aligned}
$$

そして，f を k 変数以下の多変数関数の集合と残差の和として分解する関数分解のうち，残差の二乗和が最小であり，各関数が中心化制約を満たすものを，k 次の MID といいます．

関数分解によって得られる個々の関数をさまざまな切り口で可視化することは，予測モデルの振る舞いを考察・理解するうえで非常に有用です．つまり，MID は，予測モデルを解釈するための IML 手法として活用することができます．

予測モデルの解釈手法としての MID には，以下のような特徴があります．

- 任意の予測モデルに適用可能．
- 関数分解手法としての原理がシンプルであり，特に，説明変数間の交互作用を理解しやすい形で示すことができる．
- データが存在しない領域における予測値の影響を一切受けないため，説明変数間に強い相関関係があるときに **PDP**（Partial Dependence Plot）(2) や **ALE**（Accumulated Local Effect）(3) よりも安定した解釈を得やすい (1)．
- **Friedman の H 統計量** (4) や **SHAP**（SHapley Additive exPlanations）(5) と比べて，現実的な計算コストで，より多くのサンプルを解釈することができる (6)．

[*1] ここでは 2 変数の交互作用までを考えた関数分解を表示していますが，理論上は 3 変数以上の交互作用を含めることもありえます．なお，midr パッケージでは，2 次の交互作用までの分解（2 次の MID）を行うことが可能です．

1.2 インストール

midr パッケージの最新版は，以下のコードでインストール可能です．

```
install.packages("devtools")
devtools::install_github("ryo-asashi/midr")
```

本稿の執筆時点における midr のバージョンを確認しておきます．バージョンが異なる場合，一部の関数の仕様が変更されている可能性があります．

```
# パッケージのバージョンを表示する
packageVersion("midr")
```

```
[1] '0.4.9.908'
```

2 準備

2.1 データの確認

本稿では，“Insurance data for homeowners and motor insurance customers monitored over five years”(7)[*2]というデータを使用します．このデータはスペインの住宅保険と自動車保険に関するもので，契約番号や観察年度などを含む 22 個の変数について，2010 年から 2014 年までの 5 年間における延べ 122,935 件の契約のデータが記録されています．

```
# 読み込むデータのファイルパスを指定
path_data <- "data_ex.csv"
# 変数のデータ型を指定する
dtypes <- c(gender = "factor", Car_2ndDriver_M = "factor", metro_code = "factor",
  Policy_PaymentMethodA = "factor", Policy_PaymentMethodH = "factor",
  appartment = "factor", Retention = "factor", Types = "factor")
# データを読み込む
df_all <- read.csv(path_data, sep = ",", colClasses = dtypes)
# データの概要を表示する
str(df_all)
```

```
'data.frame':   122935 obs. of  22 variables:
 $ X                          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ gender                     : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 2 2 2 2 ...
 $ Age_client                 : int  84 83 85 85 82 79 79 85 78 82 ...
```

[*2] https://data.mendeley.com/datasets/vfchtm5y7j/1 にて配布されています．

```
$ year                          : int  1 1 1 1 1 1 1 1 1 1 ...
$ age_of_car_M                  : int  13 0 0 0 20 8 10 10 8 6 ...
$ Car_power_M                   : num  90 177 163 90 115 75 200 70 115 105 ...
$ Car_2ndDriver_M               : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
$ num_policiesC                 : int  1 1 1 1 1 0 1 1 1 1 ...
$ metro_code                    : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 2 1 ...
$ Policy_PaymentMethodA         : Factor w/ 2 levels "0","1": 1 2 2 2 2 1 2 1 2 2 ...
$ Policy_PaymentMethodH         : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
$ Insuredcapital_content_re     : num  10.05 8.51 10.37 12.16 10.4 ...
$ Insuredcapital_continent_re: num  11.5 11.5 11.6 12.2 12.4 ...
$ appartment                    : Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 2 2 2 ...
$ Client_Seniority              : num  17 16.8 18.6 20.5 7.9 ...
$ Retention                     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 2 2 1 ...
$ NClaims1                      : int  0 0 0 0 0 0 0 0 0 0 ...
$ NClaims2                      : int  0 0 0 0 0 0 0 0 0 0 ...
$ Claims1                       : num  0 0 0 0 0 0 0 0 0 0 ...
$ Claims2                       : num  0 0 0 0 0 0 0 0 0 0 ...
$ Types                         : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1
 ↪  1 ...
$ PolID                         : int  1 2 3 4 5 6 7 8 9 10 ...
```

各列の内容は次のとおりです.

列名	型	説明
X	整数型	行番号（read.csv() 関数によって自動的に付与）
gender	因子型	1：男性，0：女性
Age_client	整数型	顧客の年齢
year	整数型	観察年度．1 から 5 が 2010 年から 2014 年に対応
age_of_car_M	整数型	顧客が何年前に購入した車か
Car_power_M	数値型	車の馬力
Car_2ndDriver_M	因子型	1：第 2 の臨時ドライバーに利用される旨の申告がある場合，0：それ以外
num_policiesC	整数型	保険会社における同じ顧客の保険契約の件数
metro_code	因子型	1：都会，0：地方
Policy_PaymentMethodA	因子型	自動車保険の保険料払込方法 1：年払，0：月払
Policy_PaymentMethodH	因子型	住宅保険[*3]の保険料払込方法 1：年払，0：月払
Insuredcapital_content_re	数値型	住宅保険の収容家財の価値
Insuredcapital_continent_re	数値型	住宅保険の建物の価値
appartment	因子型	1：建物がアパートの場合，0：それ以外

[*3] 建物のみならず収容家財をも補償対象としていますが，本稿では単に住宅保険と記載します.

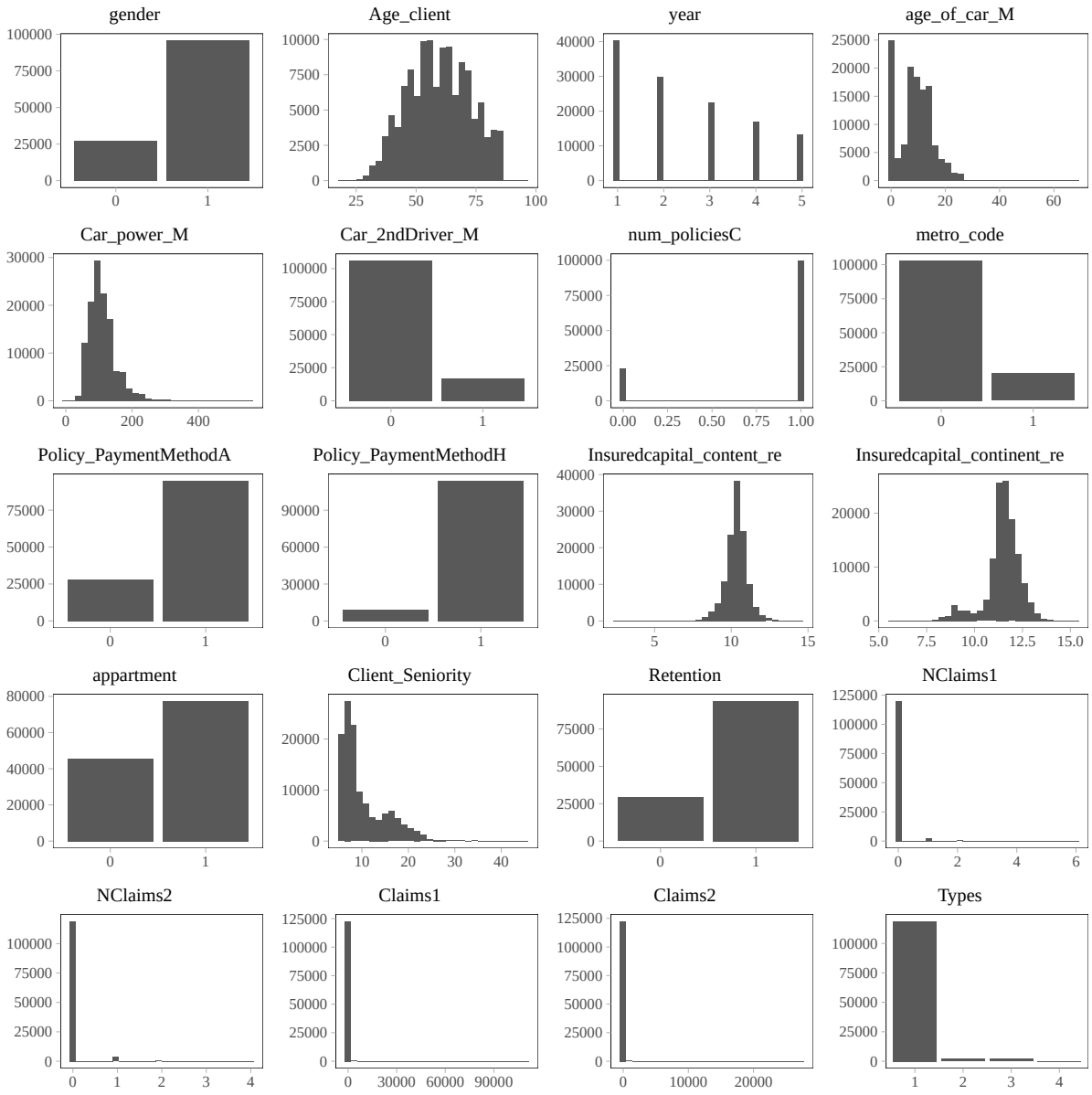
列名	型	説明
Client_Seniority	数値型	顧客が保険会社を選び続けている年数
Retention	因子型	1：契約が更新された，0：更新されなかった
NClaims1	整数型	対象の 1 年間における自動車保険のクレーム件数
NClaims2	整数型	対象の 1 年間における住宅保険のクレーム件数
Claims1	整数型	対象の 1 年間における自動車保険のクレーム総額
Claims2	整数型	対象の 1 年間における住宅保険のクレーム総額
Types	因子型	自動車，住宅それぞれのクレームの発生有無を表す．1：両方無，2：自動車のみ有，3：住宅のみ有，4：両方有
PolID	整数型	契約を特定する番号

本稿では，契約が更新されたかどうかを示す Retention を目的変数とする分類タスクを考えます．また，説明変数のうち行番号 X と契約番号 PolID は予測に利用しないため，データから除外しておきます．

```
# 行番号と契約番号を取り除いたデータを用意する
df_all <- df_all[, -which(names(df_all) == c("X", "PolID"))]
```

各変数の分布を確認しておきましょう．

```
# プロットのテーマを設定する
theme_set(theme_midr())
# 変数ごとの分布を表すプロットをリストに格納する
dist_plots <- list()
for (i in seq_len(ncol(df_all))) {
  name <- names(df_all)[i]
  dist_plots[[i]] <- ggplot(data = df_all) +
    labs(x = NULL, y = NULL, subtitle = name) +
    theme(plot.title.position = "plot", plot.subtitle = element_text(hjust = 0.5)) +
    if (is.numeric(df_all[[name]])) {
      geom_histogram(aes(x = .data[[name]]), bins = 30L)
    } else {
      geom_bar(aes(x = .data[[name]]))
    }
}
# リスト化されたプロットを整理して描画する
grid.arrange(grobs = dist_plots, ncol = 4L)
```



2.2 予測モデルの構築

ここでは，ranger パッケージを用いてランダムフォレストモデルを構築します[*4].

```
# ランダムフォレストモデルを構築する
library(ranger)
set.seed(1234)
model_rf <- ranger(
  Retention ~ ., # モデルを記述する式
```

[*4] ハイパーパラメータの設定は (1) に準じています.

```
  data = df_all, # 学習データ
  probability = TRUE, # 分類モデルで確率値を出力するかどうか
  mtry = 5, # 各ノードにおける分割基準の候補となる変数の数
  min.node.size = 250 # ノードの分割を進めるための最小の要素数
)
```

3 パッケージの使い方

3.1 関数分解の実行

MID による関数分解を実行するには `interpret()` 関数を使用します[*5]．`interpret()` の 1 番目の引数に渡す式（formula）により，分解に含める関数項を指定することが可能です．ここでは，定数項と主効果のみからなる 1 次の MID と 2 変数の交互作用を含む 2 次の MID を出力してみましょう．計算結果は “mid” というクラス属性を持つリスト（以下，単に mid オブジェクトといいます）として出力されます．後の分析で必要になるため，それぞれ `mid_1d`，`mid_2d` という変数に格納します．

```
# 1 次の関数分解を実行する
mid_1d <- interpret(
  Retention ~ ., # 関数項を指定する式
  data = df_all, # MID の計算に用いるデータ
  model = model_rf, # 解釈したい予測モデル
  k = 100 # 折れ線の節点の最大数
)
# 未解釈割合を表示する
mid_1d$ratio
```

```
[1] 0.223415
```

2 次の MID の場合，学習に 12 万件のデータをそのまま使うと計算量が大きくなりすぎるため，ここでは無作為に選んだ 1 万行のデータだけを使用します．

```
# 1 万行を抽出する
set.seed(42)
rows_mid <- sample(nrow(df_all), 10000)
# 2 次の関数分解を実行する
mid_2d <- interpret(
  Retention ~ (.)^2, data = df_all[rows_mid, ], model = model_rf,
```

[*5] ここでは (1) と同様に，ランダムフォレストモデルが出力する確率値（契約が更新される確率値）を加法的に分解します．確率値のオーダーが大きく変動する場合，0 から 1 の値をとる確率値そのものではなく，ロジット変換などにより実数値全体をとる数値に変換したものを加法的に分解することも考えられます．`interpret()` 関数の引数 `link` を活用することでそのような分解も可能ですが，本稿では割愛します．

```
  lambda = 0.001 # 正則化のための罰則項の係数
)
# 未解釈割合を表示する
mid_2d$ratio
```

```
[1] 0.1211476
```

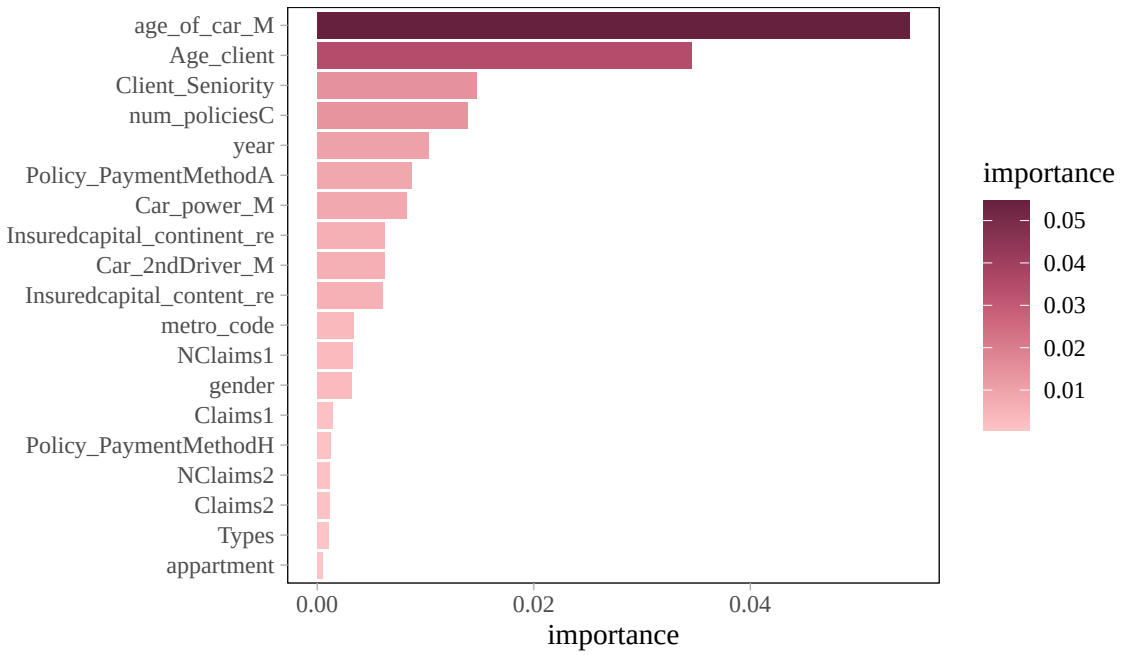
ここで，**未解釈割合**（Uninterpreted Rate）とは，解釈される予測モデルの予測値のばらつきのうち，関数分解の残差以外の項では説明されない割合のことをいい，予測値の偏差平方和（TSS）に対する残差平方和（RSS）の割合として計算されます．今回の場合，2 変数の交互作用までを含む 2 次の MID によって，ランダムフォレストモデルの予測値の分散の約 88% が説明できていることになります．

3.2 説明変数の重要度の可視化

MID では，主効果や交互作用などの各関数項は，その期待値がゼロとなるように中心化されています．そのため，たとえば「効果の絶対値の期待値」が大きい関数項ほど予測における重要性が高いと考えられます．

`mid.importance()` 関数に mid オブジェクトを渡すことで，各関数項の重要度を計算することができます．結果を可視化するには，出力されたデータフレームに対して `ggmid()` 関数を使用します．なお，引数 `theme` にカラーテーマの名前を指定することで，出力されるプロットの見た目を手軽に変更することができます．

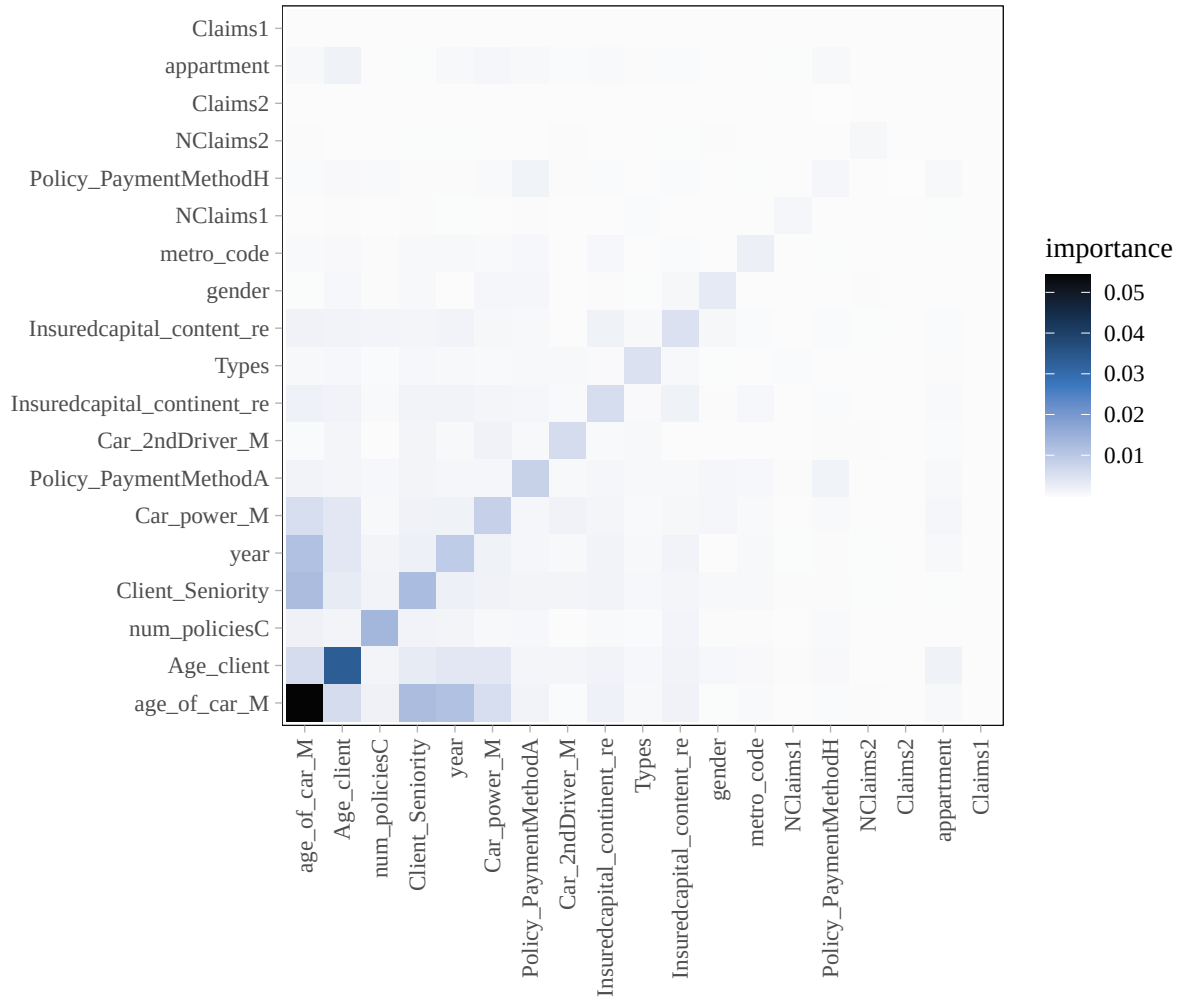
```
# 関数項の重要度を可視化する
imp_1d <- mid.importance(mid_1d)
ggmid(imp_1d, theme = "Burg_r")
```



このプロットによれば，ランダムフォレストモデルの予測において，説明変数の中でも車両購入からの年数 age_of_car_M や顧客の年齢 Age_client などが特に重要であることがわかります．

また，2 次の MID に含まれる各関数項の重要度を計算するときは，出力されるプロットの種類をヒートマップにすることも可能です．

```
# 交互作用を含む関数項の重要度を可視化する
imp_2d <- mid.importance(mid_2d)
ggmid(imp_2d, type = "heatmap", theme = "Oslo") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

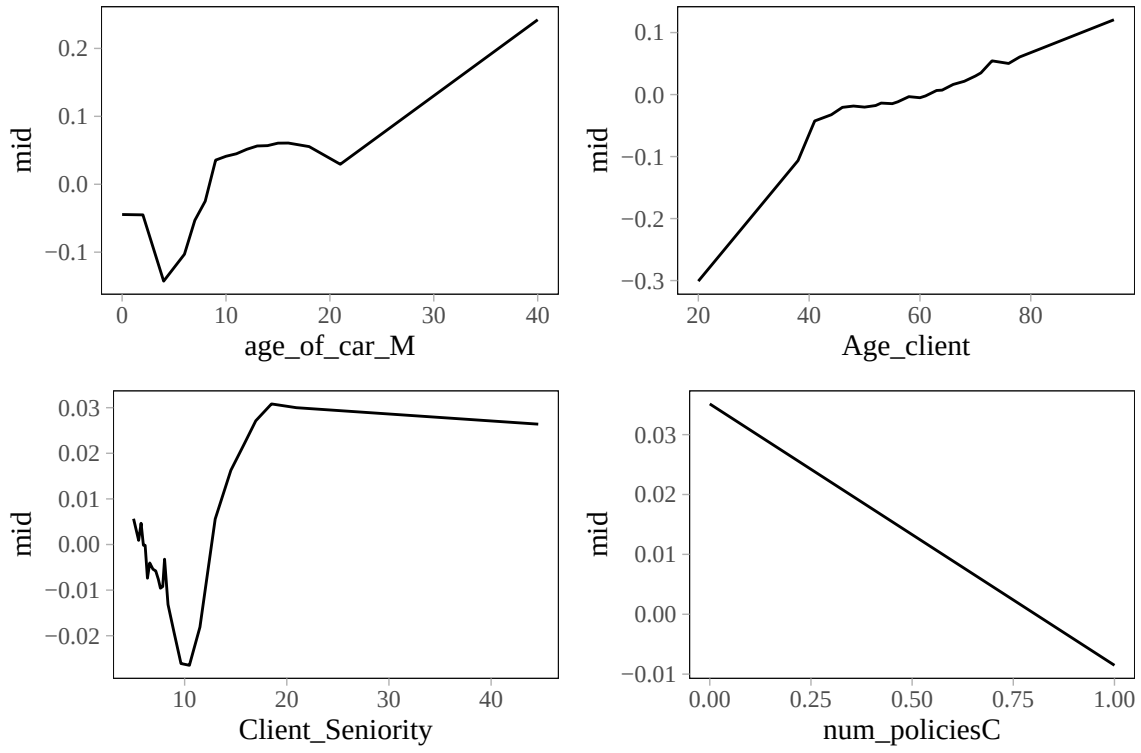


このヒートマップからは，ランダムフォレストモデルの予測において，車両購入からの年数 age_of_car_M と，顧客の継続年数 Client_Seniority や年度 year などとの間に交互作用があることがわかります．

3.3 主効果・交互作用の可視化

mid オブジェクトに対して直接 ggmid() 関数を使用し，引数 term に可視化したい関数項の名称を指定することで，主効果や交互作用の形状を観察することができます．たとえば，主効果の関数項（1 変数関数）をプロットするには次のようにします．

```
# 主効果を可視化する
grid.arrange(ggmid(mid_2d, term = "age_of_car_M"),
             ggmid(mid_2d, term = "Age_client"),
             ggmid(mid_2d, term = "Client_Seniority"),
             ggmid(mid_2d, term = "num_policiesC"))
```

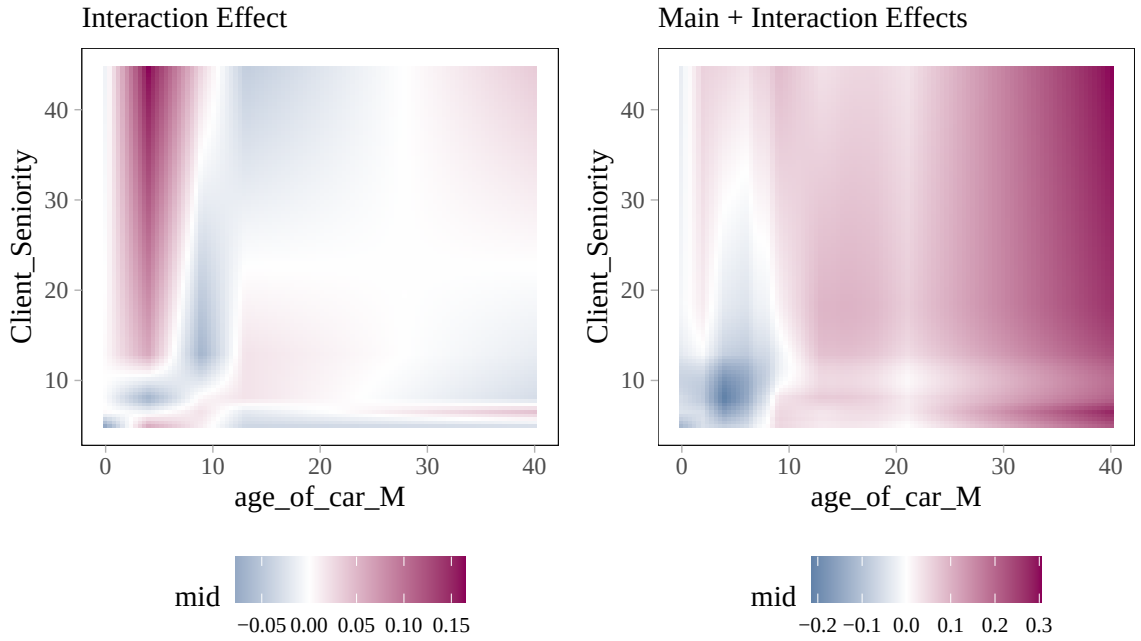


これらの主効果プロットからは，次のようなことがわかります．

- 車両購入からの年数 age_of_car_M が少ない方が更新率が低く，とりわけ 4 年目頃の更新率が非常に低い．
- 顧客の年齢 Age_client が高齢であるほど更新率が高い．
- 同じ保険会社での継続年数 Client_Seniority について，20 年ほど利用している顧客の更新率が高い．
- この保険会社における同じ顧客の保険契約の件数 num_policiesC について，もう一つ別の契約を保有している顧客の更新率が低い．

交互作用の関数項（2 変数関数）をプロットする場合は，ggmid() 関数の引数 term を "x1:x2" の形で指定します．ここで，引数 main.effects = TRUE を指定すれば，対応する主効果の関数項を加算することも可能です．

```
term <- "age_of_car_M:Client_Seniority"
# 交互作用を可視化する
plot1 <- ggmid(mid_2d, term) +
  theme(legend.position = "bottom", legend.text = element_text(size = 8)) +
  labs(subtitle = "Interaction Effect")
# 主効果を含む交互作用プロットを作成する
plot2 <- ggmid(mid_2d, term, main.effects = TRUE) +
  theme(legend.position = "bottom", legend.text = element_text(size = 8)) +
  labs(subtitle = "Main + Interaction Effects")
grid.arrange(plot1, plot2, nrow = 1)
```



この交互作用プロットからは，車両購入からの年数 age_of_car_M が短い場合でも，同じ保険会社での継続年数 Client_Seniority が 10 年を超えている契約者の更新率は，2 つの説明変数の主効果の単純な和に比べて相当高くなることがわかります．

3.4 サンプル別の予測値の変化

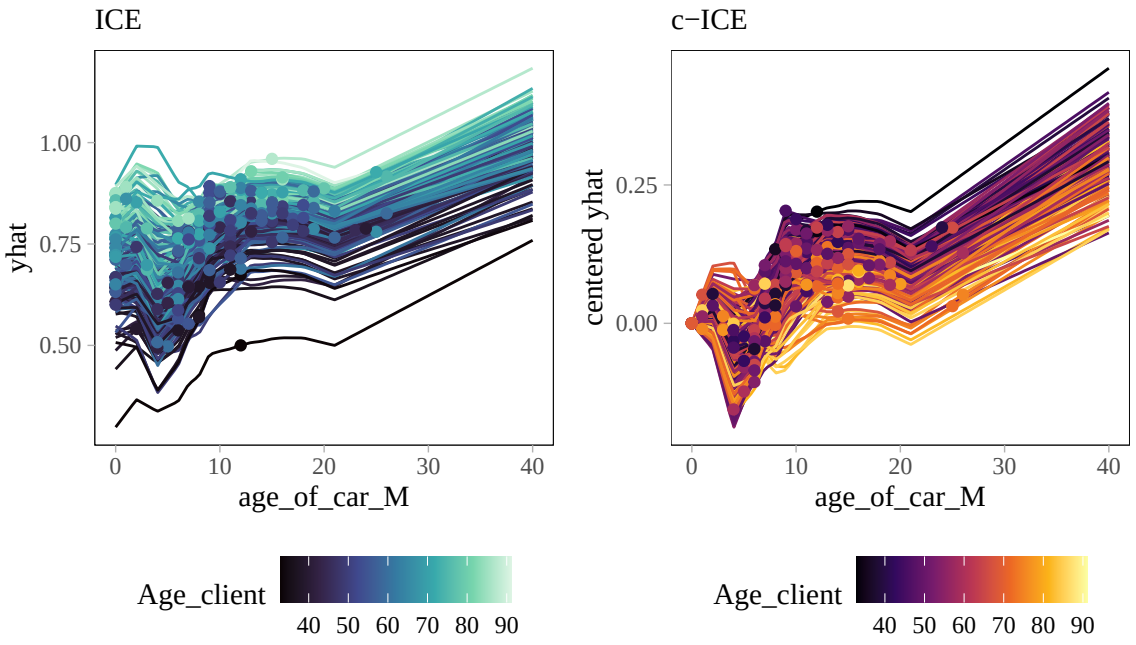
主効果プロットは，各説明変数が予測値に及ぼす影響を，サンプルによらない 1 変数関数として可視化するものです．しかし，予測モデルに交互作用がある場合，ある説明変数が予測値に及ぼす影響は，ほかの説明変数の値に応じてサンプルごとに変化するはずです．そうした影響の違いを可視化するために，サンプル別に予

測値の変化をプロットする手法を **ICE**（Individual Conditional Expectation）(8) といいます．

midr パッケージでは，`mid.conditional()` 関数を用いることで ICE を計算することが可能です．このとき，引数のうち，ICE 曲線を出力したい説明変数 `variable` と，計算に利用するサンプルを含むデータセット `data` を指定することが必要です．

出力されるオブジェクトに `ggmid()` 関数を適用することで，ICE 曲線を作成できます．また，引数 `type = "centered"` を指定すると，各折れ線グラフの左端における値がゼロに基準化され，交互作用が折れ線グラフの広がりとして可視化されます．このようなプロットを **c-ICE**（Centered ICE）といいます．

```
# ICE をプロットするサンプルを抽出する
set.seed(42)
df_ice <- df_all[sample(nrow(df_all), 200L), ]
# ICE を計算する
ice <- mid.conditional(mid_2d, variable = "age_of_car_M", data = df_ice)
# ICE プロットを作成する
plot1 <- ggmid(ice, var.color = Age_client, theme = "mako") +
  theme(legend.position = "bottom") +
  labs(subtitle = "ICE")
# Centered ICE プロットを作成する
plot2 <- ggmid(ice, type = "centered", var.color = Age_client, theme = "inferno") +
  theme(legend.position = "bottom") +
  labs(subtitle = "c-ICE")
grid.arrange(plot1, plot2, nrow = 1)
```

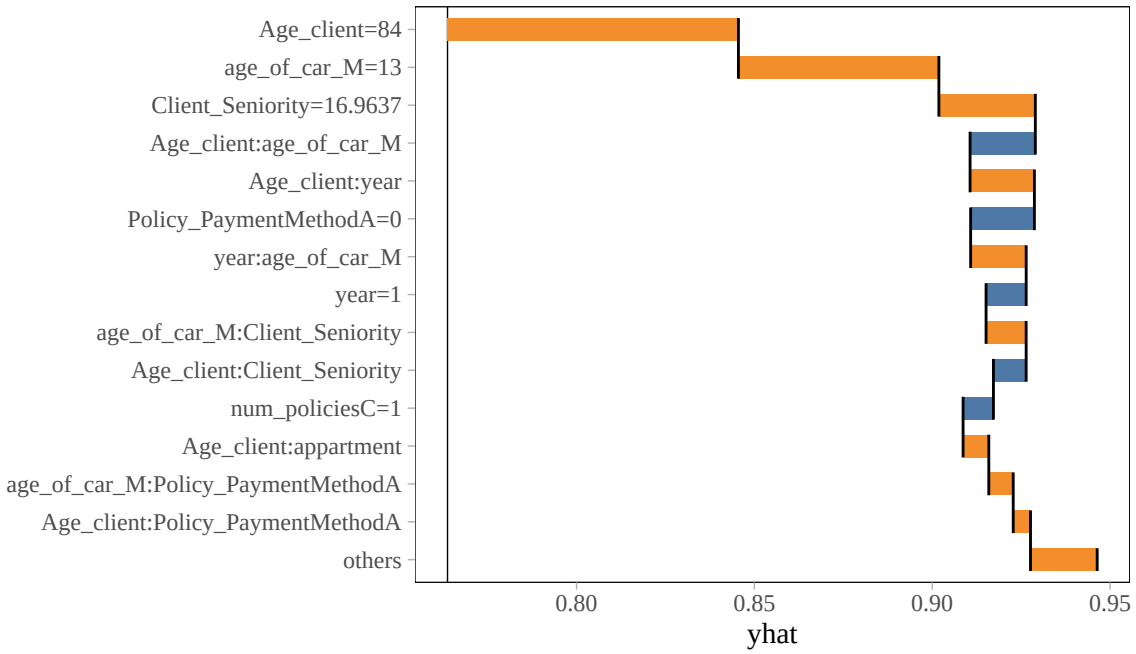


プロット内に存在する点は，実サンプルにおける説明変数 age_of_car_M の値と更新確率の予測値を示しており，各点に対応する折れ線グラフが，age_of_car_M の値が変化したときの予測値の変化を表しています．

3.5 個別の予測値の分解

サンプルごとの個別の予測値は，関数分解に基づいて平均（定数項）と各関数項の寄与に分解することができます．mid.breakdown() 関数に 1 行のデータを渡すことで，そのサンプルにおける各項の寄与を可視化することが可能です．

```
# 関数分解に沿って予測値を分解する
bd1 <- mid.breakdown(mid_2d, data = df_all[1, ])
# 予測値への各項の寄与を可視化する
ggmid(bd1, theme = "Tableau 10") +
  theme(legend.position = "none")
```



3.6 解釈精度の検証

以下では，解釈の対象となる予測モデルを「対象モデル」と呼び，また，対象モデルの予測関数の MID による関数分解から残差を取り除くことで得られる関数（「対象モデルの予測値」を，MID による関数分解で近似するモデル）を，その対象モデルの「解釈モデル」と呼ぶことにします．

MID による解釈とは，解釈モデルをさまざまな方法で指標化，可視化したものにほかなりません．それらが対象モデルに関する良い解釈であると言うためには，解釈モデルが対象モデルを良く表すモデルになっていることを確認する必要があります．未解釈割合は一つの重要な基準ですが，mid オブジェクトに格納されてい

る未解釈割合は，あくまで関数分解の学習に用いたデータにおける残差をもとにした指標であり，新しいデータについても対象モデルと解釈モデルの予測が近しいものになる保証はありません．つまり，解釈モデルは学習データに対する対象モデルの予測値を「過学習」している可能性があります．

ここでは，2 次の MID による解釈の精度を，学習に用いなかった 11 万件のデータ（評価データ）における RMSE および未解釈割合で評価してみましょう．予測モデルの予測値を数値ベクトルとして得たい場合[*6]は，midr パッケージの `get.yhat()` 関数を用いることができます．

```
# 対象モデルの予測値を計算する
preds_rf <- get.yhat(model_rf, df_all)
print(head(preds_rf))
```

```
[1] 0.9117521 0.7289752 0.8889507 0.7855020 0.8878572 0.7345995
```

```
# 解釈モデルの予測値を計算する
preds_mid <- get.yhat(mid_2d, df_all)
print(head(preds_mid))
```

```
[1] 0.9463842 0.8175311 0.8617871 0.7930823 0.9102221 0.8078979
```

評価指標の計算のために，ここでは Metrics パッケージの `sse()` 関数を利用して予測値の偏差平方和と残差平方和を計算し，その結果を用いて RMSE と未解釈割合を確認します．

```
library(Metrics)
# 学習データでの評価指標を確認する
RSS <- sse(preds_rf[rows_mid], preds_mid[rows_mid])
TSS <- sse(preds_rf[rows_mid], mean(preds_rf[rows_mid]))
n1 <- length(rows_mid)
cat("RMSE:", sqrt(RSS / n1))
```

```
RMSE: 0.03677978
```

```
cat("Uninterpreted Rate:", RSS / TSS)
```

```
Uninterpreted Rate: 0.1211476
```

```
# 関数分解に利用しなかった評価データでの評価指標を確認する
RSS <- sse(preds_rf[-rows_mid], preds_mid[-rows_mid])
TSS <- sse(preds_rf[-rows_mid], mean(preds_rf[-rows_mid]))
n2 <- nrow(df_all) - n1
```

[*6] R 標準の `predict()` 関数と役割は似ていますが，`predict()` 関数の実行結果は常に数値ベクトルとなるとは限りません．場合によっては，リスト等になっていることもあります．`get.yhat()` 関数を用いることで，予測結果を常に数値ベクトルとして出力することができます．

```
cat("RMSE:", sqrt(RSS / n2))
```

```
RMSE: 0.04097837
```

```
cat("Uninterpreted Rate:", RSS/TSS)
```

```
Uninterpreted Rate: 0.1490109
```

RMSE，未解釈割合ともに，評価データでの数値は学習データでの数値よりも悪化しているため，解釈モデルは学習データの予測値をわずかに過学習している可能性があります．しかし，評価データにおいても予測値の分散の約 85% を説明できている点を踏まえれば，2 次の MID による解釈モデルは，ランダムフォレストモデルによる予測の特徴を十分に捉えているものと考えられます．

4 参考文献

(1) 岩沢宏和・松森至宏．ブラックボックスモデルの解釈を最大化する関数分解手法．日本保険・年金リスク学会研究発表大会．2024.

(2) Friedman, J. H. Greedy function approximation: A gradient boosting machine. Annals of Statistics. 2001, vol. 29, p. 1189–1232.

(3) Apley, D. W. & Zhu, J. Visualizing the effects of predictor variables in black box supervised learning models. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2020, vol. 82(4), p. 1059–1086.

(4) Friedman, J. H. & Popescu, B. E. Predictive learning via rule ensembles. Annals of Applied Statistics. 2008, vol. 2(3), p. 916–954.

(5) Lundberg, Scott M. & Lee, S. A unified approach to interpreting model predictions. Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, p. 4768–4777.

(6) データサイエンス関連基礎調査 WG. 関数分解手法 MID の応用による機械学習モデルの解釈手法の提案．日本アクチュアリー会年次大会．2024.

(7) Guillen, M., Bolancé, C., Frees, E. & Valdez, E. A. Insurance data for homeowners and motor insurance customers monitored over five years. 2021. https://data.mendeley.com/datasets/vfchtm5y7j/1.

(8) Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. Journal of Computational and Graphical Statistics. 2015, vol. 24(1), p. 44–65.