

Case study - iviRA model

Xavier G.L.V. Pouwels

Introduction

This vignette shows how to use the functionalities of `pacheck` on the `iviRA` health economic model (Incerti et al. 2019).

Loading packages and data

```
#library(pacheck)
devtools::load_all()
```

i Loading `pacheck`

Warning: replacing previous import 'glmnet::na.replace' by 'gtools::na.replace' when loading 'pacheck'

Warning: replacing previous import 'boot::logit' by 'gtools::logit' when loading 'pacheck'

Warning: replacing previous import 'boot::inv.logit' by 'gtools::inv.logit' when loading 'pacheck'

Warning: replacing previous import 'magrittr::is_less_than' by 'testthat::is_less_than' when loading 'pacheck'

Warning: replacing previous import 'magrittr::not' by 'testthat::not' when loading 'pacheck'

Warning: replacing previous import 'magrittr::equals' by 'testthat::equals' when loading 'pacheck'

Warning: replacing previous import 'dplyr::matches' by 'testthat::matches' when loading 'pacheck'

Warning: replacing previous import 'assertthat::has_name' by 'tibble::has_name' when loading 'pacheck'

Warning: replacing previous import 'magrittr::extract' by 'tidyr::extract' when loading 'pacheck'

Warning: replacing previous import 'testthat::matches' by 'tidyr::matches' when loading 'pacheck'

```
set.seed(1234)
data("l_iviRA_pa_params")
data("l_iviRA_out_summ_1")
data("l_iviRA_out_summ_2")
```

Generating the probabilistic inputs and outputs

The source code to generate the model inputs and outputs of the `iviRA` cost-effectiveness model is available in the source code of the `pacheck` package on github (https://github.com/Xa4P/pacheck/blob/master/analysis/03-iviRA_data_generation.R).

Preparing dataset for use in pacheck

The following code chunk prepares the inputs and outputs list of the `iviRA` package to use the `pacheck` tests. In this example, only a subset of all inputs is used for convenience.

```
df_iviRA_pa <- data.frame(
  p_discount = l_iviRA_pa_params$tx.cost$discount,
  p_rebound = l_iviRA_pa_params$rebound,
  or_mort = exp(l_iviRA_pa_params$mort.logor),
  r_haq_prog = l_iviRA_pa_params$haq.lprog.tx,
  n_hosp_days = l_iviRA_pa_params$hosp.cost$hosp.days,
  c_hosp_pday = l_iviRA_pa_params$hosp.cost$cost.pday,
```

```

c_si    = l_iviRA_pa_params$si.cost,
c_prod_loss = l_iviRA_pa_params$prod.loss,
u_model_wailoo = l_iviRA_pa_params$utility.wailoo,
t_qaly_thx1 = l_iviRA_out_summ_1$means$qalys,
t_qaly_d_thx1 = l_iviRA_out_summ_1$means$dqalys,
t_cost_d_thx1 = l_iviRA_out_summ_1$means$dtot_cost,
t_qaly_thx2 = l_iviRA_out_summ_2$means$qalys,
t_qaly_d_thx2 = l_iviRA_out_summ_2$means$dqalys,
t_cost_d_thx2 = l_iviRA_out_summ_2$means$dtot_cost
)
df_iviRA_pa$inc_qaly <- df_iviRA_pa$t_qaly_d_thx1 - df_iviRA_pa$t_qaly_d_thx2
df_iviRA_pa$inc_cost <- df_iviRA_pa$t_cost_d_thx1 - df_iviRA_pa$t_cost_d_thx2

```

Testing plausibility of model inputs and outputs

One way to test the plausibility of model inputs and outputs is simply to inspect the summary statistics of each variable. This can be done with the `pacheck` package using the `generate_sum_stats` function.

```

# Summary stats
generate_sum_stats(df = df_iviRA_pa[, 1:10]) #only for the 10 first parameters for the sake of

```

	Parameter	Mean	SD	Percentile_2.5th	Percentile_97.5th
1	p_discount.abtiv	0.249	0.028	0.202	0.297
2	p_discount.abtsc	0.250	0.030	0.202	0.298
3	p_discount.ada	0.248	0.029	0.202	0.297
4	p_discount.adabiosbwtd	0.249	0.029	0.203	0.297
5	p_discount.ana	0.251	0.029	0.203	0.297
6	p_discount.bct	0.250	0.029	0.202	0.298
7	p_discount.czp	0.250	0.029	0.203	0.298
8	p_discount.etn	0.250	0.030	0.201	0.298
9	p_discount.etnbiosszzs	0.250	0.029	0.202	0.297
10	p_discount.etnbiosykro	0.250	0.029	0.203	0.297
	Minimum	Maximum	Median	Skewness	Kurtosis
1	0.2	0.3	0.249	0.044	1.843
2	0.2	0.3	0.251	-0.029	1.745
3	0.2	0.3	0.248	0.092	1.843
4	0.2	0.3	0.248	0.026	1.857
5	0.2	0.3	0.251	-0.042	1.786
6	0.2	0.3	0.250	0.035	1.821

7	0.2	0.3	0.250	0.026	1.732
8	0.2	0.3	0.250	-0.028	1.786
9	0.2	0.3	0.250	0.006	1.790
10	0.2	0.3	0.251	-0.035	1.798

```
# Specific checks
```

```
## Proportions & probabilities between 0 and 1
```

```
v_vars_discount <- grep("p_discount", names(df_ivIRA_pa), value = TRUE) # returns variable names
check_binary(c(v_vars_discount, "p_rebound"), df = df_ivIRA_pa)
```

	Input	Negative_values	Values_above_1
1	p_discount.abtiv	None	None
2	p_discount.abtsc	None	None
3	p_discount.ada	None	None
4	p_discount.adabiosbwtd	None	None
5	p_discount.ana	None	None
6	p_discount.bct	None	None
7	p_discount.czp	None	None
8	p_discount.etn	None	None
9	p_discount.etnbiosszs	None	None
10	p_discount.etnbiosykro	None	None
11	p_discount.gol	None	None
12	p_discount.hcl	None	None
13	p_discount.ifx	None	None
14	p_discount.ifxbiosqbtx	None	None
15	p_discount.cdmards	None	None
16	p_discount.rtx	None	None
17	p_discount.sar	None	None
18	p_discount.ssz	None	None
19	p_discount.tcz	None	None
20	p_discount.tof	None	None
21	p_discount.upa	None	None
22	p_rebound	None	None

```
## Costs, counts, OR positive
```

```
v_vars_n <- grep("n_", names(df_ivIRA_pa), value = TRUE)
```

```
v_vars_c <- grep("c_", names(df_ivIRA_pa), value = TRUE)
```

```
v_vars_c <- v_vars_c[-grep("inc_", v_vars_c)]
```

```
check_positive(c(v_vars_n, v_vars_c, "or_mort"), df = df_ivIRA_pa)
```

Input Negative_values

1	n_hosp_days.haq_less0.5	None
2	n_hosp_days.haq0.5to1	None
3	n_hosp_days.haq1to1.5	None
4	n_hosp_days.haq1.5to2	None
5	n_hosp_days.haq2to2.5	None
6	n_hosp_days.haq2.5plus	None
7	c_hosp_pday.haq_less0.5	None
8	c_hosp_pday.haq0.5to1	None
9	c_hosp_pday.haq1to1.5	None
10	c_hosp_pday.haq1.5to2	None
11	c_hosp_pday.haq2to2.5	None
12	c_hosp_pday.haq2.5plus	None
13	c_si	None
14	c_prod_loss	None
15	or_mort	None

```
do_check(df = df_iviRA_pa,
  v_vars = c(v_vars_n, v_vars_c, "or_mort"),
  check = ~ .x >= 0,
  label_check = "positive") # alternative
```

\$check

n_hosp_days.haq_less0.5	n_hosp_days.haq0.5to1	n_hosp_days.haq1to1.5
TRUE	TRUE	TRUE
n_hosp_days.haq1.5to2	n_hosp_days.haq2to2.5	n_hosp_days.haq2.5plus
TRUE	TRUE	TRUE
c_hosp_pday.haq_less0.5	c_hosp_pday.haq0.5to1	c_hosp_pday.haq1to1.5
TRUE	TRUE	TRUE
c_hosp_pday.haq1.5to2	c_hosp_pday.haq2to2.5	c_hosp_pday.haq2.5plus
TRUE	TRUE	TRUE
c_si	c_prod_loss	or_mort
TRUE	TRUE	TRUE

\$messages

A tibble: 1 x 2

ok message

<lg1> <glue>

1 TRUE all variables are positive

Metamodelling

In this section, we will perform metamodelling using two techniques: linear regression modelling (lm) and random forest (rf) using all input parameters of the probabilistic analysis data frame. Before performing metamodelling, we will standardise the inputs using their respective mean and standard deviation.

Fit

The metamodels are first developed on the full data sets, variable selection for the Note that we did not perform variable selection first, which is an important step if the model is used for prediction. We have used the `stepAIC` function from the `MASS` package to perform backward variable selection using the Akaike Information Criteria (AIC) using the linear regression metamodel. For the sake of simplicity, we will use the same selected variable for the linear regression metamodel and the random forest metamodel.

In the `fit_rf_metamodel`, we set the `var_importance` argument to `TRUE` to be able to extract variable importance plots and for the random forest metamodel.

```
# Select parameters for metamodelling
v_x_vars <- names(df_iviRA_pa)
v_x_vars <- v_x_vars[-grep("t_qaly", v_x_vars)]
v_x_vars <- v_x_vars[-grep("t_cost", v_x_vars)]
v_x_vars <- v_x_vars[-grep("inc_cost", v_x_vars)]
v_x_vars <- v_x_vars[-grep("inc_qaly", v_x_vars)]

## standardisation
df_iviRA_mm <- df_iviRA_pa
df_iviRA_mm[, v_x_vars] <- as.data.frame(apply(df_iviRA_mm[, v_x_vars], 2, function(col) (col - mean(col)) / sd(col))))

# Fit metamodels using all input parameters contained in the parameter data frame
metamodel_lm <- fit_lm_metamodel(df = df_iviRA_mm,
                                y_var = "t_qaly_d_thx1",
                                x_vars = v_x_vars)

# Features (variable selection)
mm_lm_select <- MASS::stepAIC(do.call("lm", list(metamodel_lm$model_info$form,
                                                data = metamodel_lm$model_info$data)),
                              direction = "backward",
                              trace = FALSE)
v_select_vars <- names(mm_lm_select$coefficients)[-1]
```

```
# Fit based on selected values
metamodel_lm <- fit_lm_metamodel(df = df_iviRA_mm,
                                y_var = "t_qaly_d_thx1",
                                x_vars = v_select_vars)
summary(metamodel_lm$fit)
```

Call:

```
lm(formula = form, data = df)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.30454	-0.24527	-0.01384	0.23620	1.18534

Coefficients:

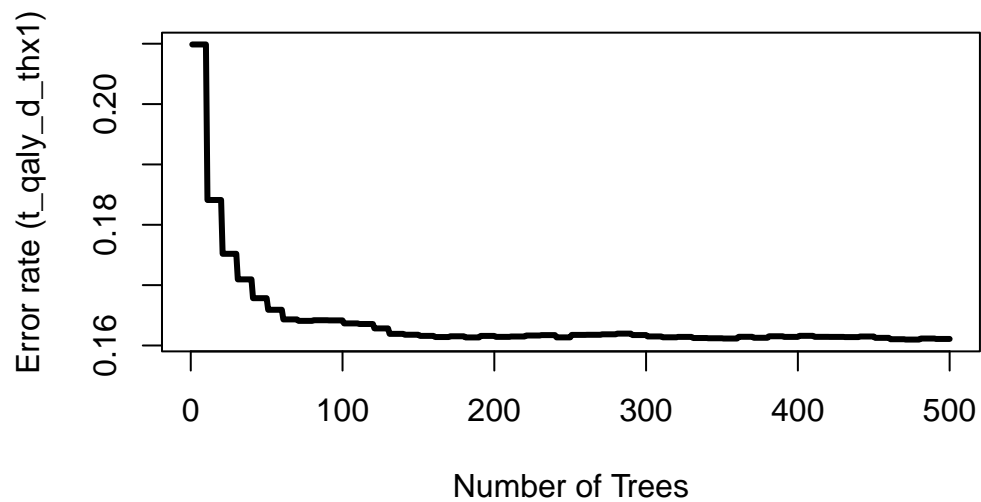
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	10.00297	0.01157	864.785	< 2e-16 ***
p_discount.ana	0.01883	0.01167	1.614	0.106888
p_discount.ssz	0.02810	0.01172	2.398	0.016652 *
p_rebound	-0.18679	0.01166	-16.021	< 2e-16 ***
or_mort	-0.36768	0.01169	-31.458	< 2e-16 ***
r_haq_prog.adamtx	-0.06240	0.01175	-5.311	1.35e-07 ***
r_haq_prog.czp	0.01704	0.01178	1.446	0.148513
r_haq_prog.etnmtx	-0.02047	0.01167	-1.755	0.079574 .
r_haq_prog.ifxmtx	-0.02702	0.01167	-2.316	0.020750 *
r_haq_prog.nbt	-0.04794	0.01173	-4.087	4.73e-05 ***
r_haq_prog.sarmtx	-0.01932	0.01174	-1.646	0.100042
r_haq_prog.tczmtx	-0.01658	0.01174	-1.413	0.157916
n_hosp_days.haq_less0.5	-0.02353	0.01174	-2.005	0.045259 *
c_hosp_pday.haq1.5to2	0.02565	0.01169	2.194	0.028489 *
c_hosp_pday.haq2.5plus	-0.01885	0.01169	-1.612	0.107233
c_si	0.02475	0.01167	2.122	0.034105 *
u_model_wailoo.int	0.06030	0.01169	5.159	3.00e-07 ***
u_model_wailoo.age	0.05307	0.01164	4.558	5.81e-06 ***
u_model_wailoo.dis_dur	0.03498	0.01167	2.998	0.002789 **
u_model_wailoo.haq0	0.04363	0.01164	3.748	0.000189 ***
u_model_wailoo.prev_dmards	0.03342	0.01166	2.866	0.004246 **
u_model_wailoo.haq	0.02446	0.01168	2.094	0.036489 *

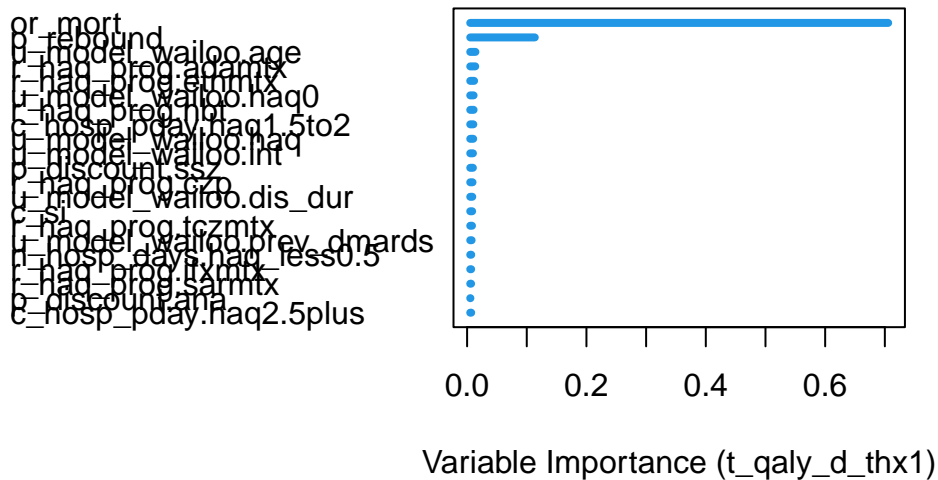
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3658 on 978 degrees of freedom

Multiple R-squared: 0.5822, Adjusted R-squared: 0.5732
F-statistic: 64.89 on 21 and 978 DF, p-value: < 2.2e-16

```
metamodel_rf <- fit_rf_metamodel(df = df_iviRA_mm,  
                                y_var = "t_qaly_d_thx1",  
                                x_vars = v_select_vars,  
                                var_importance = TRUE) # automatically plots variable import
```





Based on the variable importance plots (but also coefficient values of the linear regression metamodel), one can conclude that the `or_mort` and `p_rebound` model parameters have the biggest impact on the results.

Validation

Validation of the model will be done using cross-validation with ten folds. Note that for the validation purposes that the metamodel is re-fitted in each fold.

```
# Metamodels validation
validation_mm_lm <- validate_metamodel(
  model = metamodel_lm,
  method = 'cross_validation',
  folds = 10
)
validation_mm_rf <- validate_metamodel(
  model = metamodel_rf,
  method = 'cross_validation',
  folds = 10
)

knitr::kable(validation_mm_lm,
  caption = "Validation metrics of the linear regression metamodel.")
```

Table 1: Validation metrics of the linear regression metamodel.

Statistic	Value (method: cross-validation)
R-squared	0.563
Mean absolute error	0.296
Mean relative error	0.030
Mean squared error	0.137

Table 2: Validation metrics of the random forest metamodel.

Statistic	Value (method: cross-validation)
R-squared	0.505
Mean absolute error	0.318
Mean relative error	0.032
Mean squared error	0.162

```
knitr::kable(validation_mm_rf,
              caption = "Validation metrics of the random forest metamodel.")
```

Based on this validation step, we can see that the linear regression metamodel results in a higher proportion of the variance being explained by the parameters included in the meta-models (higher R^2), and lower errors than the random forest metamodel.

Prediction

The following code snippet shows how to make prediction using the fitted metamodels.

```
# Create dataframe containing the 10th and 90th percentile of the probabilistic values of the
df_inputs_pred <- data.frame(matrix(ncol = length(v_select_vars),
                                   nrow = 2,
                                   dimnames = list(c(),
                                                    v_select_vars))
                             )
df_inputs_pred[1, ] <- unname(apply(df_iviRA_mm[, v_select_vars],
                                   2,
                                   function(rows) {quantile(rows, probs = 0.1)}})
df_inputs_pred[2, ] <- unname(apply(df_iviRA_mm[, v_select_vars],
                                   2,
                                   function(rows) {quantile(rows, probs = 0.9)}})

# Predict
```

```

df_lm_preds <- predict_metamodel(model = metamodel_lm,
                                inputs = df_inputs_pred,
                                output_type = "long_df")
df_rf_preds <- predict_metamodel(model = metamodel_rf,
                                inputs = df_inputs_pred,
                                output_type = "long_df")

#Display
knitr::kable(df_lm_preds,
              caption = "Predictions using the linear regression metamodel.")

```

Table 3: Predictions using the linear regression metamodel.

Name	Value
p_discount.ana	-1.3878898
p_discount.ssz	-1.3240638
p_rebound	-1.3662859
or_mort	-1.2655790
r_haq_prog.adamtx	-1.2988305
r_haq_prog.czp	-1.2896426
r_haq_prog.etnmtx	-1.2847053
r_haq_prog.ifxmtx	-1.2440296
r_haq_prog.nbt	-1.2832023
r_haq_prog.sarmtx	-1.2696626
r_haq_prog.tczmtx	-1.2919564
n_hosp_days.haq_less0.5	-0.5051284
c_hosp_pday.haq1.5to2	-1.2270428
c_hosp_pday.haq2.5plus	-1.2387378
c_si	-1.4115996
u_model_wailoo.int	-1.3129443
u_model_wailoo.age	-1.2157838
u_model_wailoo.dis_dur	-1.2820319
u_model_wailoo.haq0	-1.3017471
u_model_wailoo.prev_dmards	-1.2944664
u_model_wailoo.haq	-1.1943372
predictions	10.5376134

```

knitr::kable(df_rf_preds,
              caption = "Predictions using the random forest metamodel.")

```

Table 4: Predictions using the random forest metamodel.

Name	Value
p_discount.ana	-1.3878898
p_discount.ssz	-1.3240638
p_rebound	-1.3662859
or_mort	-1.2655790
r_haq_prog.adamtx	-1.2988305
r_haq_prog.czp	-1.2896426
r_haq_prog.etnmtx	-1.2847053
r_haq_prog.ifxmtx	-1.2440296
r_haq_prog.nbt	-1.2832023
r_haq_prog.sarmtx	-1.2696626
r_haq_prog.tczmtx	-1.2919564
n_hosp_days.haq_less0.5	-0.5051284
c_hosp_pday.haq1.5to2	-1.2270428
c_hosp_pday.haq2.5plus	-1.2387378
c_si	-1.4115996
u_model_wailoo.int	-1.3129443
u_model_wailoo.age	-1.2157838
u_model_wailoo.dis_dur	-1.2820319
u_model_wailoo.haq0	-1.3017471
u_model_wailoo.prev_dmards	-1.2944664
u_model_wailoo.haq	-1.1943372
predictions	10.5141224

References

- Incerti, Devin, Jeffrey R. Curtis, Jason Shafrin, Darius N. Lakdawalla, and Jeroen P. Jansen. 2019. “A Flexible Open-Source Decision Model for Value Assessment of Biologic Treatment for Rheumatoid Arthritis.” *PharmacoEconomics* 37 (6): 829–43. <https://doi.org/10.1007/s40273-018-00765-2>.