Plotting Fixed Effects Solutions From EBV-Routine For MAR in RH

2020-01-30

Pre-Requisites

Required packages are loaded

```
require(readr)
require(dplyr)
require(ggplot2)
```

Disclaimer

Experiments and trials to plot fixed effect solutions from routine EBV for MAR in RH are shown in this document.

Data

The data is located in the work directory of the routine EBV for MAR

```
s_breed <- "rh"
s_trait <- "mar"
s_data_dir_pre <- "/Volumes/data_zws/health"
s_data_dir_post <- "zws"
s_data_dir <- file.path(s_data_dir_pre, s_trait, "work", s_breed, s_data_dir_post)
# check whether data directory exists
if (!dir.exists(s_data_dir)) stop(" *** * ERROR: Cannot find data directory: ", s_data_dir)
```

Effect Levels

Try to extract effect levels from the re-coding log file. Determine the path to the logfile.

```
s_log_fn <- "codeEff.log"
s_log_file_path <- file.path(s_data_dir, s_log_fn)
```

Extractor Helper Function

For a given row in the logfile, the values for calvingYear (cy), calvingMonth (cm) and the corresponding level are extracted and returned as a one-row tibble

```
# extractor function to get from single record, the values for cy, cm and the level
tbl_extract_code_info <- function(pvec_log_info){
  # get index in pvec_log_info where cy info is
  vec_cy_idx <- grep(pattern = "calvingYear", pvec_log_info)
  # get cy info and convert to number
  n_cy <- as.integer(gsub(pattern = "cycm(calvingYear=",
                          replacement = "",
                          pvec_log_info[vec_cy_idx],
                          fixed = TRUE))
  # get index for cm info
  vec_cm_idx <- grep(pattern = "calvingMonth", pvec_log_info)
  # get level code
  vec_cm_info <- unlist(strsplit(pvec_log_info[vec_cm_idx], split = ") ", fixed = TRUE))
  n_code <- as.integer(vec_cm_info[2])
  n_cm <- as.integer(gsub(pattern = "calvingMonth=",
                          replacement = "",
                          vec_cm_info[1],
                          fixed = TRUE))
  return(tibble::tibble(calvingYear = n_cy, calvingMonth = n_cm, level = n_code))
}
```

Main Function producing the code-level mapping

The function below reads the logfile and extracts the part with the mapping. The different code levels are extracted from each line using the above helper function.

```
tbl_get_level_code <- function(ps_log_file_path) {
  # check whether log file exists
  if (!file.exists(ps_log_file_path)) stop(" *** Error: Cannot find log file: ", ps_log_file_path)
  # read logfile into vector of string
  con_code_eff_log <- file(ps_log_file_path)
  vec_code_eff_log <- readLines(con = con_code_eff_log)
  close(con = con_code_eff_log)
  # get idx where mapping of level-to-codes is
  vec_code_eff_idx <- grep(pattern = "cycm", vec_code_eff_log, fixed = TRUE)
  vec_code_eff_level <- vec_code_eff_log[vec_code_eff_idx]
  # split every line into entries for calvingYear and calvingMonth
  l_code_info <- strsplit(vec_code_eff_level, split = ", ", fixed = TRUE)

  # return the result
  return(purrr::map_dfr(l_code_info, tbl_extract_code_info))

}
```

Code Extraction

The function `tbl_get_level_code` is used to extract the codes of the different fixed effect levels

```
tbl_level_code <- tbl_get_level_code(ps_log_file_path = s_log_file_path)
```

Limit Number of Years

The analysis is done for the last few years and therefore, the codes are only kept for these years

```
n_yr_back <- 5
n_min_year <- max(tbl_level_code$calvingYear) - n_yr_back
tbl_level_code <- tbl_level_code %>%
  filter(calvingYear > n_min_year)
```

Same Code for Multiple Levels

Some codes are used for multiple levels. This is verified in the following statement.

```
tbl_mult_level_code <- tbl_level_code %>%
  group_by(level) %>%
  summarise(cnt = n()) %>%
  filter(cnt > 1)
tbl_mult_level_code
```

```
## # A tibble: 1 x 2
##   level   cnt
##   <int> <int>
## 1     1     2
```

At the moment we are not eliminating any level-code mappings.

```
# eliminate duplicate code explicitly, this a manual step and can only be done once
# if (nrow(tbl_mult_level_code) > 0){
#   tbl_level_code %>% filter(level %in% tbl_mult_level_code$level)
#   tbl_level_code <- tbl_level_code[-1,]
#
# }
```

Reading The Solutions

The solution file from MiX99 for the trait MAR for RH is read

```
s_solfix_fn <- "Solfix"
s_solfix_path <- file.path(s_data_dir, s_solfix_fn)
if (!file.exists(s_solfix_path)) stop(" *** * ERROR: Cannot find solfix file: ", s_solfix_path)
tbl_sol_fix <- read_delim(file      = s_solfix_path,
                          delim     = " ")
```

```
## Parsed with column specification:
## cols(
##   ` Fact.` = col_character(),
##   Trt = col_character(),
```

```
##   `     Level` = col_character(),
##   `   N-Obs` = col_character(),
##   ` Solution` = col_character(),
##   `              Factor` = col_character(),
##   Trait = col_character()
## )
```

```
## Warning: 1266 parsing failures.
## row col  expected    actual                                                file
##   1  -- 7 columns 8 columns '/Volumes/data_zws/health/mar/work/rh/zws/Solfix'
##   2  -- 7 columns 8 columns '/Volumes/data_zws/health/mar/work/rh/zws/Solfix'
##   3  -- 7 columns 8 columns '/Volumes/data_zws/health/mar/work/rh/zws/Solfix'
##   4  -- 7 columns 8 columns '/Volumes/data_zws/health/mar/work/rh/zws/Solfix'
##   5  -- 7 columns 8 columns '/Volumes/data_zws/health/mar/work/rh/zws/Solfix'
## ... ... ......... ......... ...............................................
## See problems(...) for more details.
```

Remove leading and trailing spaces from column names

```
colnames(tbl_sol_fix) <- purrr::map(colnames(tbl_sol_fix), function(x) gsub(pattern = "\\s+", replaceme
#head(tbl_sol_fix)
```

Remove spaces from entries in the tibble.

```
tbl_sol_fix <- purrr::modify(tbl_sol_fix,  function(x) gsub(pattern = "\\s+", replacement = "", x))
#head(tbl_sol_fix)
```

Data-types for the columns `Fact.`, `Trt`, `Level`, `N-Obs` and `Solution` are changed

```
tbl_sol_fix$Fact. <- as.integer(tbl_sol_fix$Fact.)
tbl_sol_fix$Trt <- as.integer(tbl_sol_fix$Trt)
tbl_sol_fix$Level <- as.integer(tbl_sol_fix$Level)
tbl_sol_fix$`N-Obs` <- as.integer(tbl_sol_fix$`N-Obs`)
tbl_sol_fix$Solution <- as.double(tbl_sol_fix$Solution)
#head(tbl_sol_fix)
```

The solutions read from the solution file are filtered such that only solutions for the trait `mar` and the effect `cycm` are kept.

```
tbl_sol_fix_mar_cycm <- tbl_sol_fix %>%
  filter(Trait == 'mar' & Factor == 'cycm')
#head(tbl_sol_fix_mar_cycm)
```

Merge The Levels To the Solutions Based on Code-Level-Mapping

The levels are required for plot as a meaningful x-Axis display

```
tbl_sol_fix_mar_cycm <- tbl_sol_fix_mar_cycm %>%
  inner_join(tbl_level_code, by = c('Level' = 'level'))
#head(tbl_sol_fix_mar_cycm)
```

Restrict Range of Calving Years

The analysis is only done for a given range of calving years. The data is filtered to keep only records within the desired range.

```
n_min_year <- max(tbl_sol_fix_mar_cycm$calvingYear) - n_yr_back
tbl_sol_fix_mar_cycm <- tbl_sol_fix_mar_cycm %>%
  filter(calvingYear > n_min_year)
#head(tbl_sol_fix_mar_cycm)
```

Calving Year Times Month as Date

Calving year and calving month which are in separate columns are combined into a single column and are converted into a `Date` as data-type. This makes the use of a `data_scale` on the x-axis.

```
tbl_sol_fix_mar_cycm <- tbl_sol_fix_mar_cycm %>%
  mutate(`Calving Year x Month` = as.Date(paste(ifelse(nchar(calvingMonth) == 1,
                                                       paste(calvingYear, paste0("0",calvingMonth), sep = '-'),
                                                       paste(calvingYear, calvingMonth, sep = '-')),
                                                "01", sep = "-")))
#head(tbl_sol_fix_mar_cycm,15)
```

Levels in Chronological Order

The order of the levels should be sorted chronologically.

```
# sort-levels
tbl_sol_fix_mar_cycm <- tbl_sol_fix_mar_cycm %>%
  arrange(desc(Level))
#head(tbl_sol_fix_mar_cycm)
```

The levels of the cycm effects should be unique.

```
tbl_sol_fix_mar_cycm %>%
  group_by(`Calving Year x Month`) %>%
  summarise(cnt = n()) %>%
  filter(cnt > 1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: `Calving Year x Month` <date>, cnt <int>
```

The first six records of the prepared data-set are shown below.

```
head(tbl_sol_fix_mar_cycm)
```

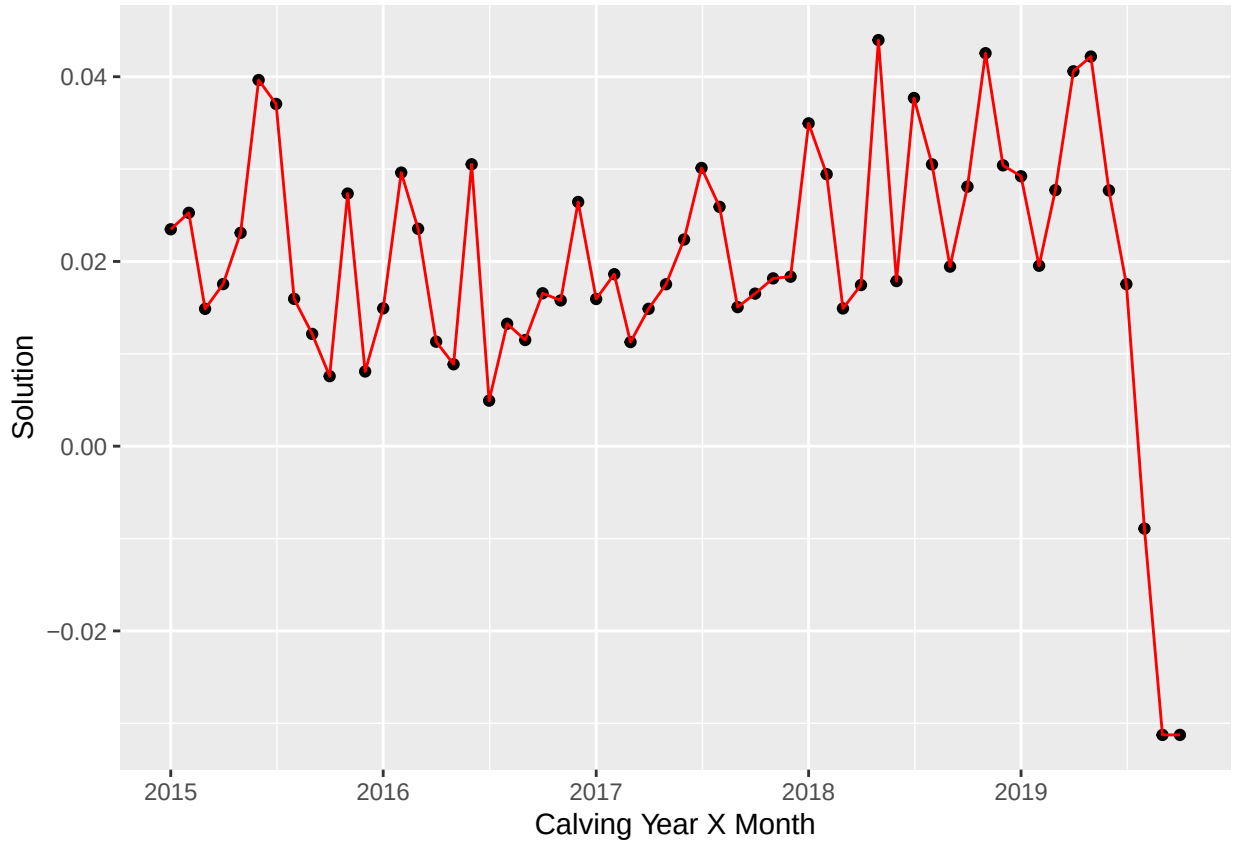
```
## # A tibble: 6 x 10
##   Fact.   Trt Level `N-Obs` Solution Factor Trait calvingYear calvingMonth
##   <int> <int> <int>   <int>    <dbl> <chr>  <chr>       <int>        <int>
## 1     1     1    57    1241   0.0235 cycm   mar          2015            1
## 2     1     1    56    1140   0.0253 cycm   mar          2015            2
```

```
## 3     1     1    55    1004   0.0149 cycm   mar          2015            3
## 4     1     1    54     819   0.0175 cycm   mar          2015            4
## 5     1     1    53     777   0.0231 cycm   mar          2015            5
## 6     1     1    52     797   0.0396 cycm   mar          2015            6
## # ... with 1 more variable: `Calving Year x Month` <date>
```

Plots

The plots show the solutions for the different levels of fixed effect. Because the effect-levels are time-dependent, the x-axis is shown as a time scale.

```
# plots using date on x-axis
p_cycm <- ggplot(data = tbl_sol_fix_mar_cycm, aes(x = `Calving Year x Month`, y=Solution)) +
  geom_point() +
  geom_line(color = "red") +
  scale_x_date(name = "Calving Year X Month",
               date_breaks = "1 year",
               date_labels = "%Y"
               )
p_cycm
```



title: “20200129_plot_fixed_effect_solutions.R” author: “pvr” date: “2020-01-30” —