DualBreed

Sophie Kunz and Peter von Rohr

2 November 2018

1. Computing Economic Value For Dual Breed

Genetic Standard Deviations

Economic values can also be given in terms of a change of one genetic standard deviation. The used estimates for this parameter are

```
l_gen_sd <- list(CCc = 0.6336,
                 CCa = 0.6335,
                 CFc = 0.3474,
                 CFa = 0.3609,
                 CWc = 0.0557,
                 CWa = 0.1395)
```

Carcass conformation adults (CCa)

```
### # prices
vec_price_cca <- c(7.526960,7.938872,8.450784,8.800000,9.137304,9.392693,9.642693)
```

OB

```
n_mean_cca_ob <- 5.20
n_sd_cca_ob <- 1.02
vec_count_cca_ob <- c(4,43,218,1150,2106,1905,522)
vec_freq_cca_ob <- vec_count_cca_ob / sum(vec_count_cca_ob)
```

BV

```
n_mean_cca_bv <- 4.78
n_sd_cca_bv <- 1.27
vec_count_cca_bv <- c(273,1562,5982,17808,14303,11438,5875)
vec_freq_cca_bv <- vec_count_cca_bv / sum(vec_count_cca_bv)
```

SI

```
n_mean_cca_si <- 5.83
n_sd_cca_si <- 0.9
vec_count_cca_si <- c(4,38,377,3994,13917,24738,13535)
vec_freq_cca_si <- vec_count_cca_si / sum(vec_count_cca_si)
```

SF

```
n_mean_cca_sf <- 4.57
n_sd_cca_sf <- 1.18
vec_count_cca_sf <- c(165,1189,4814,11545,10445,6303,1755)
vec_freq_cca_sf <- vec_count_cca_sf / sum(vec_count_cca_sf)
```

MO

```
n_mean_cca_mo <- 5.39
n_sd_cca_mo <- 0.94
vec_count_cca_mo <- c(10,33,194,1341,3511,3730,979)
vec_freq_cca_mo <- vec_count_cca_mo / sum(vec_count_cca_mo)
```

Carcass conformation calves (CCc)

```
### # prices
vec_price_ccc <- c(11.2,12.7,13.6,14.2,14.7,15.2,15.7)
```

OB

```
n_mean_ccc_ob <- 4.98
n_sd_ccc_ob <- 1.01
vec_count_ccc_ob <- c(11,94,451,2266,3486,2376,472)
vec_freq_ccc_ob <- vec_count_ccc_ob / sum(vec_count_ccc_ob)
```

BV

```
n_mean_ccc_bv <- 4.08
n_sd_ccc_bv <- 1.08
vec_count_ccc_bv <- c(1759,10739,42522,94581,40759,15082,4855)
vec_freq_ccc_bv <- vec_count_ccc_bv / sum(vec_count_ccc_bv)
```

SI

```
n_mean_ccc_si <- 5.33
n_sd_ccc_si <- 1.02
vec_count_ccc_si <- c(16,64,247,1620,3359,3374,1087)
vec_freq_ccc_si <- vec_count_ccc_si / sum(vec_count_ccc_si)
```

SF

```
n_mean_ccc_sf <- 3.88
n_sd_ccc_sf <- 1.08
```

```
vec_count_ccc_sf <- c(645,4027,13631,21453,9150,3054,626)
vec_freq_ccc_sf <- vec_count_ccc_sf / sum(vec_count_ccc_sf)
```

MO

```
n_mean_ccc_mo <- 4.73
n_sd_ccc_mo <- 1.11
vec_count_ccc_mo <- c(14,57,232,774,920,523,118)
vec_freq_ccc_mo <- vec_count_ccc_mo / sum(vec_count_ccc_mo)
```

Carcass fatness adults (CFa)

```
### # prices
vec_price_cfa <- c(-0.9000000,
-0.3000000,
0.0000000,
-0.3926929,
-0.8480817)
```

OB

```
n_mean_cfa_ob <- 2.88
n_sd_cfa_ob <- 0.58
vec_count_cfa_ob <- c(161,889,4429,448,21)
vec_freq_cfa_ob <- vec_count_cfa_ob / sum(vec_count_cfa_ob)
```

BV

```
n_mean_cfa_bv <- 2.85
n_sd_cfa_bv <- 0.6
vec_count_cfa_bv <- c(1704,9941,41215,4167,214)
vec_freq_cfa_bv <- vec_count_cfa_bv / sum(vec_count_cfa_bv)
```

SI

```
n_mean_cfa_si <- 2.82
n_sd_cfa_si <- 0.55
vec_count_cfa_si <- c(1259,10942,41326,3004,72)
vec_freq_cfa_si <- vec_count_cfa_si / sum(vec_count_cfa_si)
```

SF

```
n_mean_cfa_sf <- 2.87
n_sd_cfa_sf <- 0.55
```

```
vec_count_cfa_sf <- c(787,5694,27214,2442,79)
vec_freq_cfa_sf <- vec_count_cfa_sf / sum(vec_count_cfa_sf)
```

MO

```
n_mean_cfa_mo <- 2.68
n_sd_cfa_mo <- 0.6
vec_count_cfa_mo <- c(373,2721,6420,280,4)
vec_freq_cfa_mo <- vec_count_cfa_mo / sum(vec_count_cfa_mo)
```

Carcass fatness calves (CFc)

```
### # prices
vec_price_cfc <- c(-1.5,
-0.6,
0.0,
-0.4,
-1.0)
```

OB

```
n_mean_cfc_ob <- 2.62
n_sd_cfc_ob <- 0.69
vec_count_cfc_ob <- c(632,2671,5379,471,3)
vec_freq_cfc_ob <- vec_count_cfc_ob / sum(vec_count_cfc_ob)
```

BV

```
n_mean_cfc_bv <- 2.68
n_sd_cfc_bv <- 0.67
vec_count_cfc_bv <- c(12790,52626,133521,11316,44)
vec_freq_cfc_bv <- vec_count_cfc_bv / sum(vec_count_cfc_bv)
```

SI

```
n_mean_cfc_si <- 2.66
n_sd_cfc_si <- 0.7
vec_count_cfc_si <- c(691,2522,5974,578,2)
vec_freq_cfc_si <- vec_count_cfc_si / sum(vec_count_cfc_si)
```

SF

```
n_mean_cfc_sf <- 2.76
n_sd_cfc_sf <- 0.65
```

```
vec_count_cfc_sf <- c(2475,11464,35025,3602,19)
vec_freq_cfc_sf <- vec_count_cfc_sf / sum(vec_count_cfc_sf)
```

MO

```
n_mean_cfc_mo <- 2.64
n_sd_cfc_mo <- 0.67
vec_count_cfc_mo <- c(191,676,1675,96,0)
vec_freq_cfc_mo <- vec_count_cfc_mo / sum(vec_count_cfc_mo)
```

Carcass weight adults (CWa)

```
 n_scale_fact_cwa <- 100
vec_price_cwa <- c(0.0,
-0.1,
-0.2,
-0.3,
-0.5,
-0.7,
-0.9,
-1.2,
-1.4,
-1.6,
-1.8)
vec_thre_cwa <- c(2.9,
3.0,
3.1,
3.2,
3.3,
3.4,
3.5,
3.6,
3.7,
3.8) * n_scale_fact_cwa
```

OB

```
n_mean_cwa_ob <- 2.61 * n_scale_fact_cwa
n_sd_cwa_ob <- 0.41 * n_scale_fact_cwa
```

BV

```
n_mean_cwa_bv <- 2.77 * n_scale_fact_cwa
n_sd_cwa_bv <- 0.36 * n_scale_fact_cwa
```

SI

```
n_mean_cwa_si <- 2.79 * n_scale_fact_cwa
n_sd_cwa_si <- 0.42 * n_scale_fact_cwa
```

SF

```
n_mean_cwa_sf <- 2.88 * n_scale_fact_cwa
n_sd_cwa_sf <- 0.32 * n_scale_fact_cwa
```

MO

```
n_mean_cwa_mo <- 2.99 * n_scale_fact_cwa
n_sd_cwa_mo <- 0.25 * n_scale_fact_cwa
```

Carcass weight calves (CWc)

```
n_scale_fact_cwc <- 100
vec_price_cwc <- seq(0.0,-1.1,-0.1);vec_price_cwc
```

```
##  [1]  0.0 -0.1 -0.2 -0.3 -0.4 -0.5 -0.6 -0.7 -0.8 -0.9 -1.0 -1.1
```

```
vec_thre_cwc <- seq(1.4, 1.5, 0.01) * n_scale_fact_cwc
vec_thre_cwc
```

```
##  [1] 140 141 142 143 144 145 146 147 148 149 150
```

OB

```
n_mean_cwc_ob <- 1.25 * n_scale_fact_cwc
n_sd_cwc_ob <- 0.13 * n_scale_fact_cwc
```

BV

```
n_mean_cwc_bv <- 1.26 * n_scale_fact_cwc
n_sd_cwc_bv <- 0.14 * n_scale_fact_cwc
```

SI

```
n_mean_cwc_si <- 1.27 * n_scale_fact_cwc
n_sd_cwc_si <- 0.13 * n_scale_fact_cwc
```

SF

```
n_mean_cwc_sf <- 1.24 * n_scale_fact_cwc
n_sd_cwc_sf <- 0.13 * n_scale_fact_cwc
```

MO

```
n_mean_cwc_mo <- 1.28 * n_scale_fact_cwc
n_sd_cwc_mo <- 0.15 * n_scale_fact_cwc
```

Overview of the phenotypic mean

Traits	OB	BV	SI	SF	MO
cca	5.20	4.78	5.83	4.57	5.39
ccc	4.98	4.08	5.33	3.88	4.73
cfa	2.88	2.85	2.82	2.87	2.68
cfc	2.62	2.68	2.66	2.76	2.64
cwa	261.00	277.00	279.00	288.00	299.00
cwc	125.00	126.00	127.00	124.00	128.00

Presenting output for economic values

The computed economic values are shown in the following tables:

1.1) Table are presenting economic value in trait unit.

Traits	OB	BV	SI	SF	MO
cca	0.2786366	0.3015193	0.2474393	0.3120456	0.2676898
ccc	0.5004597	0.5680155	0.4893199	0.5992985	0.5278464
cfa	0.0553657	0.0729381	0.1006845	0.0615499	0.1893340
cfc	0.4011634	0.3732053	0.3796711	0.3137966	0.4235119
cwa	-0.0038515	-0.0057708	-0.0065763	-0.0076378	-0.0100679
cwc	-0.0113522	-0.0133317	-0.0139556	-0.0101531	-0.0160444

1.2) Table are presenting economic value in genetic standard deviation.

Traits	OB	BV	SI	SF	MO
cca	0.1765163	0.1910125	0.1567528	0.1976809	0.1695815
ccc	0.3170913	0.3598946	0.3100331	0.3797155	0.3344435
cfa	0.0199815	0.0263234	0.0363370	0.0222133	0.0683306
cfc	0.1393642	0.1296515	0.1318978	0.1090129	0.1471280
cwa	-0.0537280	-0.0805030	-0.0917393	-0.1065469	-0.1404470
cwc	-0.0632318	-0.0742578	-0.0777326	-0.0565530	-0.0893674

2. Computing Relative Economic Factors

Relative economic factors are defined as the ratio of each economic value on the basis of one genetic standard deviation to the sum of all economic values in a given breed. The principle of how the relative economic factors are computed is shown in the chunk below.

```
tbl_ev_result_ev_per_gen_sd
```

```
## # A tibble: 6 x 6
##   Traits      OB      BV      SI      SF      MO
##   <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 cca     0.177   0.191   0.157   0.198   0.170
## 2 ccc     0.317   0.360   0.310   0.380   0.334
## 3 cfa     0.0200  0.0263  0.0363  0.0222  0.0683
## 4 cfc     0.139   0.130   0.132   0.109   0.147
## 5 cwa    -0.0537 -0.0805 -0.0917 -0.107  -0.140
## 6 cwc    -0.0632 -0.0743 -0.0777 -0.0566 -0.0894
```

```
# convert the tibble with economic values on the basis of one genotypic standard deviation to a matrix
mat_ev_per_gen_sd <- as.matrix(tbl_ev_result_ev_per_gen_sd[,2:ncol(tbl_ev_result_ev_per_gen_sd)])
mat_ev_per_gen_sd
```

```
##               OB          BV          SI          SF          MO
## [1,]  0.17651628  0.19101250  0.15675277  0.19768091  0.16958147
## [2,]  0.31709128  0.35989461  0.31003309  0.37971555  0.33444348
## [3,]  0.01998150  0.02632336  0.03633702  0.02221335  0.06833064
## [4,]  0.13936415  0.12965152  0.13189775  0.10901295  0.14712802
## [5,] -0.05372804 -0.08050301 -0.09173934 -0.10654687 -0.14044696
## [6,] -0.06323182 -0.07425784 -0.07773262 -0.05655304 -0.08936742
```

```
# compute sum of absolute economic values within each breed
vec_abs_sum_ev <- apply(abs(mat_ev_per_gen_sd), 2, sum)
vec_abs_sum_ev
```

```
##        OB        BV        SI        SF        MO
## 0.7699131 0.8616428 0.8044926 0.8717227 0.9492980
```

```
# inverse of sum
vec_inv_abs_sum_ev <- 1/vec_abs_sum_ev
vec_inv_abs_sum_ev
```

```
##       OB       BV       SI       SF       MO
## 1.298848 1.160574 1.243020 1.147154 1.053410
```

```
# extend inverse factors into a matrix
mat_inv_abs_sum_ev <- matrix(vec_inv_abs_sum_ev, nrow = nrow(mat_ev_per_gen_sd), ncol = ncol(mat_ev_per
# element-wise multiplication of matrix of economic values and matrix of inverse sums to get ratios
(mat_factors_ev <- mat_ev_per_gen_sd * mat_inv_abs_sum_ev)
```

```
##               OB          BV          SI          SF          MO
## [1,]  0.22926781  0.22168408  0.19484675  0.22677042  0.17863882
## [2,]  0.41185336  0.41768421  0.38537718  0.43559214  0.35230611
## [3,]  0.02595293  0.03055019  0.04516763  0.02548213  0.07198018
## [4,]  0.18101284  0.15047014  0.16395148  0.12505462  0.15498613
## [5,] -0.06978455 -0.09342968 -0.11403379 -0.12222565 -0.14794824
## [6,] -0.08212852 -0.08618169 -0.09662317 -0.06487504 -0.09414053
```

```
mat_factors_ev
```

```
##               OB          BV          SI          SF          MO
## [1,]  0.22926781  0.22168408  0.19484675  0.22677042  0.17863882
## [2,]  0.41185336  0.41768421  0.38537718  0.43559214  0.35230611
## [3,]  0.02595293  0.03055019  0.04516763  0.02548213  0.07198018
## [4,]  0.18101284  0.15047014  0.16395148  0.12505462  0.15498613
## [5,] -0.06978455 -0.09342968 -0.11403379 -0.12222565 -0.14794824
## [6,] -0.08212852 -0.08618169 -0.09662317 -0.06487504 -0.09414053
```

```
# check
apply(abs(mat_factors_ev), 2, sum)
```

```
## OB BV SI SF MO
##  1  1  1  1  1
```

```
all.equal(sum(apply(abs(mat_factors_ev), 2, sum)),ncol(mat_factors_ev))
```

```
## [1] TRUE
```

```
tbl_rel_fact <- tibble::as_tibble(mat_factors_ev)
tbl_rel_fact <- bind_cols(tbl_ev_result_ev_per_gen_sd[,1],tbl_rel_fact)
```

The whole computation is now done in a function called `get_relative_economic_factors()`. This function takes as input the tibble of all economic values.

2.1) Computing Relative Economic Factors For All Categories

Traits	OB	BV	SI	SF	MO
cca	0.2292678	0.2216841	0.1948468	0.2267704	0.1786388
ccc	0.4118534	0.4176842	0.3853772	0.4355921	0.3523061
cfa	0.0259529	0.0305502	0.0451676	0.0254821	0.0719802
cfc	0.1810128	0.1504701	0.1639515	0.1250546	0.1549861
cwa	-0.0697845	-0.0934297	-0.1140338	-0.1222257	-0.1479482
cwc	-0.0821285	-0.0861817	-0.0966232	-0.0648750	-0.0941405

2.2) Computing Relative Economic Factors For Adults

Computing the factors for different animal categories separately can be done with two separate function calls.We start with the category “addults”

Traits	OB	BV	SI	SF	MO
cca	0.7054279	0.6413283	0.5503397	0.6055637	0.4482025
cfa	0.0798539	0.0883812	0.1275748	0.0680470	0.1805973
cwa	-0.2147182	-0.2702905	-0.3220855	-0.3263892	-0.3712002

2.3) Computing Relative Economic Factors For Calves

The same is done for the category “calves”

Traits	OB	BV	SI	SF	MO

Traits	OB	BV	SI	SF	MO
ccc	0.6101579	0.6383329	0.5966036	0.6963660	0.5857780
cfc	0.2681693	0.2299585	0.2538138	0.1999205	0.2576949
cwc	-0.1216728	-0.1317086	-0.1495826	-0.1037135	-0.1565271

3. Importance of calves versus adults for each population

Categories	OB	BV	SI	SF	MO
adults	0.393803	0.2139547	0.8528401	0.4078332	0.7878739
calves	0.606197	0.7860453	0.1471599	0.5921668	0.2121261

4. Determine Relative Factors Based on Given Restrictions

animal1: normal growth animal2: growing fast Animal 1 and 2 have the same slaughterweight -> breeding value for animal2 is higher than animal1. In the index, the economic value (ev) resulting of the payment system are negative for slaughterweight.

Consequence: Index animal 1 would be higher than index animal 2.

A fast solution would be according to Urs Schnyder: ev_cwa_n = ev_cca_n

ev_cwc_n = ev_ccc_n

ev_cfa_n = alphaa * ev_cca_n

ev_cfc_n = alphac * ev_ccc_n

Using the scale factors to compute the weights

ev_total_n = 1 = ev_cca_n + ev_cfa_n + ev_cwa_n + ev_ccc_n + ev_cfc_n + ev_cwc_n

replace all the terms in the formula to solve the equation.

ev_ccc_n = beta * ev_cca_n

ev_cwa_n = 1 / (2 + 2beta + alphaa + alphac*beta)

Traits	OB	BV	SI	SF	MO
cca	0.1539528	0.1518637	0.1422689	0.1537154	0.1386026
ccc	0.2765586	0.2861327	0.2813862	0.2952643	0.2733479
cfa	0.0174273	0.0209283	0.0329795	0.0172730	0.0558481
cfc	0.1215497	0.1030789	0.1197105	0.0847677	0.1202509
cwa	0.1539528	0.1518637	0.1422689	0.1537154	0.1386026
cwc	0.2765586	0.2861327	0.2813862	0.2952643	0.2733479

5. Writing Output To A File

The economic values that have been computed so far are collected into a dataframe and are written to a csv-formatted file.

Manual conversion and table1.2 output are shown below.

```
vec_breed <- c("OB", "BV", "SI", "SF", "MO")
vec_trait <- c("cca", "ccc", "cfa", "cfc", "cwa", "cwc")
n_nr_trait <- length(vec_trait)

tbl_ev_input <- NULL
 for (b in vec_breed){
# b <- vec_breed[2]
  ### # put together
  if (is.null(tbl_ev_input)){
    tbl_ev_input <- tibble::data_frame(Trait = vec_trait,
                                       Breed = rep(b, length(n_nr_trait)),
                                       Ev    = tbl_ev_result_ev_per_gen_sd[[b]])
  } else {
    tbl_ev_current <- tibble::data_frame(Trait = vec_trait,
                                       Breed = rep(b, length(n_nr_trait)),
                                       Ev    = tbl_ev_result_ev_per_gen_sd[[b]])
    tbl_ev_input <- rbind(tbl_ev_input, tbl_ev_current)
  }

}
readr::write_csv(tbl_ev_input, path = "ev_meat_input.csv")
```

Use the function `write_ev_to_file()` A good initial test is to write the same tibble (table1.2) as with the manual conversion.

```
MeatValueIndex::write_ev_to_file(ptbl_economic_value = tbl_ev_result_ev_per_gen_sd,
ps_out_path = "economic_value_raw.csv",
pb_first_col_trait_name = TRUE)
```

Szenario A) we are using the relative economic factors (table 2.1, File name: economic_value_relative.csv)

Traits	OB	BV	SI	SF	MO
cca	0.2292678	0.2216841	0.1948468	0.2267704	0.1786388
ccc	0.4118534	0.4176842	0.3853772	0.4355921	0.3523061
cfa	0.0259529	0.0305502	0.0451676	0.0254821	0.0719802
cfc	0.1810128	0.1504701	0.1639515	0.1250546	0.1549861
cwa	-0.0697845	-0.0934297	-0.1140338	-0.1222257	-0.1479482
cwc	-0.0821285	-0.0861817	-0.0966232	-0.0648750	-0.0941405

Szenario B) The relative factors are weighted with animal categories to get weighted relative factors (File name: weighted_economic_value_relative.csv)

Traits	OB	BV	SI	SF	MO
cca	0.0902863	0.0474303	0.1661731	0.0924845	0.1407449
ccc	0.2496643	0.3283187	0.0567121	0.2579432	0.0747333
cfa	0.0102203	0.0065364	0.0385208	0.0103925	0.0567113
cfc	0.1097294	0.1182764	0.0241271	0.0740532	0.0328766
cwa	-0.0274814	-0.0199897	-0.0972526	-0.0498477	-0.1165646

Traits	OB	BV	SI	SF	MO
cwc	-0.0497861	-0.0677427	-0.0142191	-0.0384168	-0.0199697

Szenario C) This set consist of the creative political values not weighted for animal classes (table 4, File name: political_unweighted.csv)

Traits	OB	BV	SI	SF	MO
cca	0.1539528	0.1518637	0.1422689	0.1537154	0.1386026
ccc	0.2765586	0.2861327	0.2813862	0.2952643	0.2733479
cfa	0.0174273	0.0209283	0.0329795	0.0172730	0.0558481
cfc	0.1215497	0.1030789	0.1197105	0.0847677	0.1202509
cwa	0.1539528	0.1518637	0.1422689	0.1537154	0.1386026
cwc	0.2765586	0.2861327	0.2813862	0.2952643	0.2733479

Szenario D) The last set is the above factors weighted with the proportions of the animal classes (File name: political_weighted.csv)

Traits	OB	BV	SI	SF	MO
cca	0.0606271	0.0324919	0.1213326	0.0626902	0.1092014
ccc	0.1676490	0.2249133	0.0414087	0.1748457	0.0579842
cfa	0.0068629	0.0044777	0.0281262	0.0070445	0.0440013
cfc	0.0736831	0.0810247	0.0176166	0.0501966	0.0255084
cwa	0.0606271	0.0324919	0.1213326	0.0626902	0.1092014
cwc	0.1676490	0.2249133	0.0414087	0.1748457	0.0579842