

B: Worked examples of fitting extended moult models

The `moultmcmc` package: Bayesian inference for moult phenology models

In most free-living animal populations moult progression and duration in individuals can not be observed fully. Instead snapshot measurements of (re)captured individuals are typically used to infer these parameters on a population level. As an additional complication, recording of moult in the field may take various forms both in terms of the subset of the population that is sampled and whether moult is recorded as a categorical state, or a (semi-)continuous progression.

Underhill & Zucchini (1989; Ibis 130:358) proposed a general modelling framework to accommodate many of these features, implemented in the R package `moult` (Erni et al. 2013; J Stat Soft 52:8).

`moultmcmc` implements a Bayesian inference framework for this class of models with the aim of (eventually) allowing the inclusion of hierarchical model structures to accommodate 1) the integration of moult data sets using different modes of recording, 2) individual heterogeneity in moult timing, and 3) misclassified observations of non-moulting birds

`moultmcmc` implements fast inference for these models using Hamiltonian Monte Carlo samplers from Stan (<https://mc-stan.org/>). The currently implemented models are described in detail in Boersch-Supan et al. (2022) (<https://doi.org/10.48550/arXiv.2205.12120>) and the vignette ‘Moult data likelihoods’ (<https://pboesu.github.io/moultmcmc/articles/moult-likelihoods.html>).

Installation

The package `moultmcmc` is built around pre-compiled Stan models, the easiest and quickest way of installing it is to install the package from R-universe use the following code:

```
install.packages("moultmcmc", repos = "https://pboesu.r-universe.dev")
```

On Mac and Windows systems this will make use of pre-compiled binaries, which means the models can be run without having to install a C++ compiler. On Linux this will install the package from a source tarball. Because of the way the Stan models are currently structured, compilation from source can be a lengthy process (15-45 minutes), depending on system setup and compiler toolchain.

To install `moultmcmc` from the github source (not generally recommended for Windows users) use the following code. This requires a working C++ compiler and a working installation of `rstan`:

```
#not generally recommended for Windows/MacOS users  
install.packages("remotes")  
remotes::install_github("pboesu/moultmcmc")
```

Basic usage

Basic usage is described in the vignette ‘Getting started with `moultmcmc`’ (<https://pboesu.github.io/moultmcmc/articles/getting-started.html>).

Fitting standard and extended moult models to a subset of the Cape Weaver data

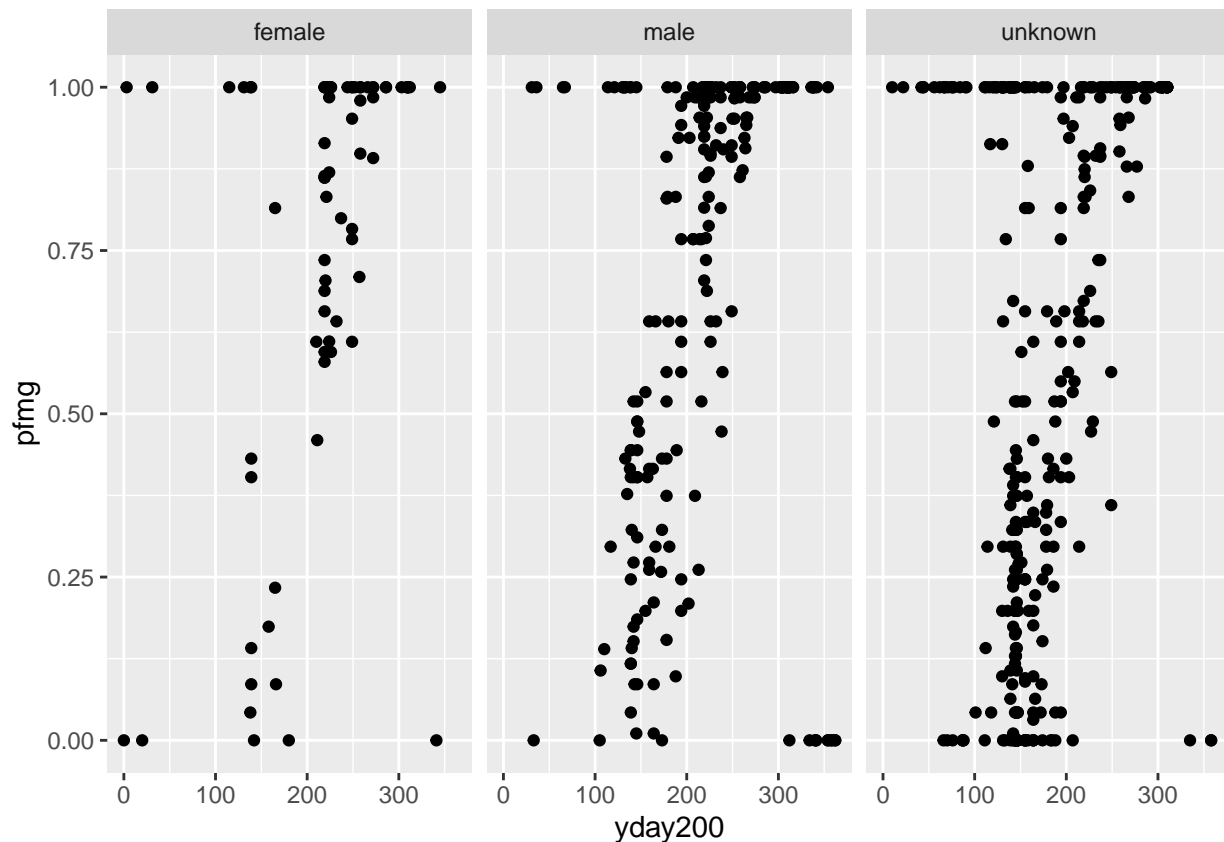
To fit the models described in the main text we load the necessary R packages and read in the example data set. This is a subset of the Cape Weaver data from HDO. The dataset has five columns: - `ringno_season`: a

unique identifier of individuals encountered in a specific season - yday: day of year (since January 1) - yday: days since day of year 200 (this shifted variable ensures active moult falls within a 365 contiguous day period)
 - pfmg: proportion of feather mass grown based on relative feather masses from Underhill & Jouventin 1995.
 - sex: factor variable with levels "female", "male", "unknown"

```
library(moultmcmc) # for model fitting
library(ggplot2) # for plotting
```

```
## Warning: package 'ggplot2' was built under R version 4.2.1
```

```
#read data
juv_cw <- read.csv('juvenile_weavers_example_data.csv', stringsAsFactors = TRUE)
#plot data
ggplot(juv_cw, aes(x = yday200, y = pfmg)) + geom_point() + facet_wrap(~sex)
```



We can then fit the standard models using the `moultmcmc` function. Each MCMC chain takes about 30 seconds to run for these models.

```
T2 <- moultmcmc(date_column = 'yday200',
                 moult_column = 'pfmg',
                 start_formula = ~ sex,
                 duration_formula = ~ sex,
                 type = 2,
                 data = juv_cw,
                 chains = 2)
```

```
T3 <- moultmcmc(date_column = 'yday200',
```

```

moult_column = 'pfmg',
start_formula = ~ sex,
duration_formula = ~ sex,
type = 3,
data = juv_cw,
chains = 2)# 35 sec per chain

```

The extended moult models are also fitted using the `moultmcmc` function. The lumped model variants are activated by setting the argument `lump_non_moult = TRUE`

```

T2L <- moultmcmc(date_column = 'yday200',
moult_column = 'pfmg',
start_formula = ~ sex,
duration_formula = ~sex,
type = 2,
lump_non_moult = TRUE,
data = juv_cw,
chains = 2)#32 sec per chain

```

and the recaptures models are activated by providing individual identifiers to the `id_column` argument. The recaptures models are computationally much more challenging. Each chain for this example takes about 5 minutes to run. If multiple cores are available it is therefore recommended to run chains in parallel by setting the `cores` argument to greater than one.

```

T2LR <- moultmcmc(date_column = 'yday200',
moult_column = 'pfmg',
id_column = 'ringno_season',
start_formula = ~ sex,
duration_formula = ~sex,
type = 2,
lump_non_moult = TRUE,
data = juv_cw,
chains = 2, cores = 2, iter = 1000)#c. 5 mins per chain

```

```

T3R <- moultmcmc(date_column = 'yday200',
moult_column = 'pfmg',
id_column = 'ringno_season',
start_formula = ~ sex,
duration_formula = ~sex,
type = 3,
data = juv_cw,
chains = 2, cores = 2, iter = 1000)#c. 5 minutes per chain

```

Parameter tables for the fitted models can be extracted using the `summary` command:

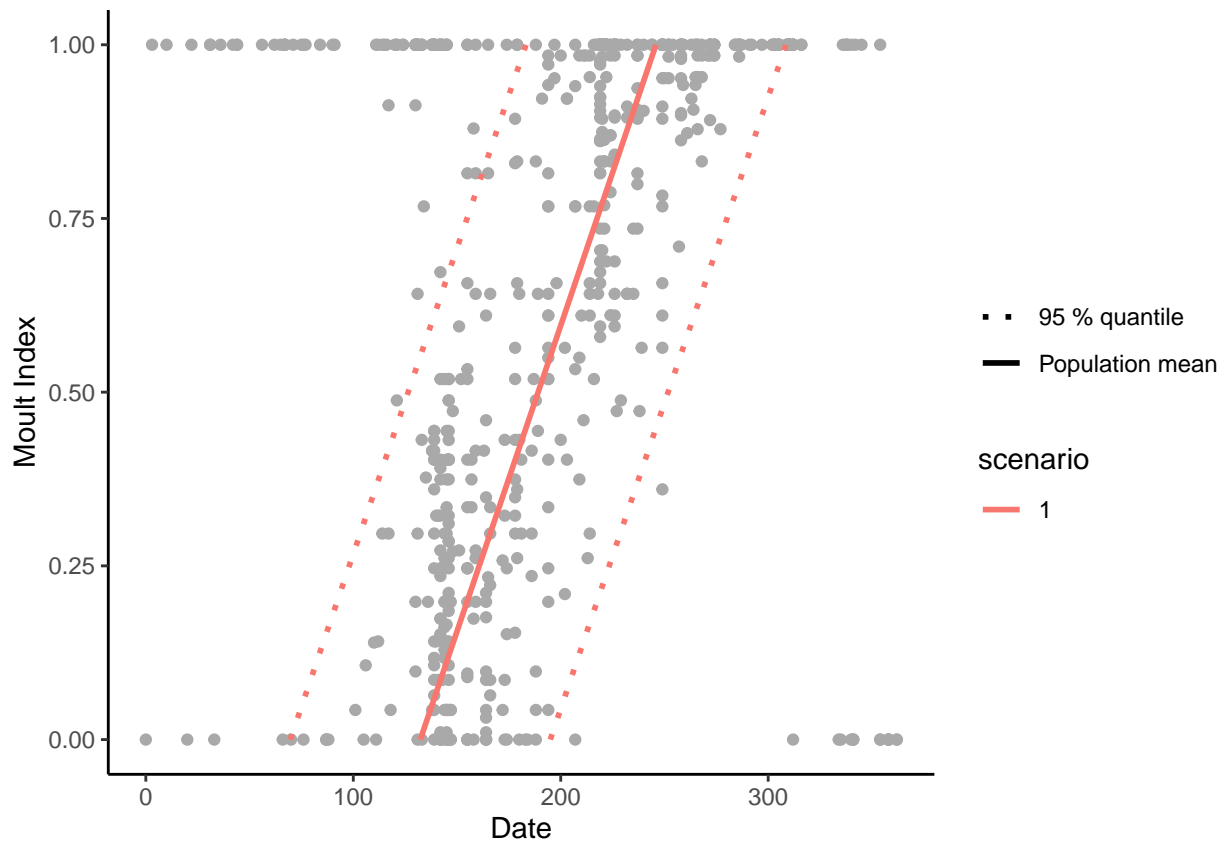
```
summary(T2L)
```

parameter	estimate	se_mean	sd	lci	uci	n_eff	Rhat	prob	method	type
mean_(Intercept)	127.03	0.45	9.46	109.29	145.90	437.58	1.00	0.95	MCMC	2L
mean_sexmale	-16.61	0.48	10.39	-37.15	3.49	461.46	1.00	0.95	MCMC	2L
mean_sexunknown	6.43	0.48	9.97	-13.19	25.46	432.32	1.00	0.95	MCMC	2L
duration_(Intercept)	112.18	0.51	11.11	90.82	133.97	471.25	1.00	0.95	MCMC	2L
duration_sexmale	13.79	0.54	12.21	-9.01	37.85	515.73	1.00	0.95	MCMC	2L
duration_sexunknown	-3.08	0.53	12.09	-26.69	20.23	510.96	1.00	0.95	MCMC	2L
log_sd_(Intercept)	3.50	0.00	0.05	3.41	3.59	1060.37	1.00	0.95	MCMC	2L

parameter	estimate	se_mean	sd	lci	uci	n_eff	Rhat	prob	method	type
sd_(Intercept)	33.01	0.05	1.50	30.20	36.11	1057.78	1.00	0.95	MCMC	2L
lp____	-199.56	0.07	1.85	-	-	676.24	1.01	0.95	MCMC	2L
				203.94	196.86					

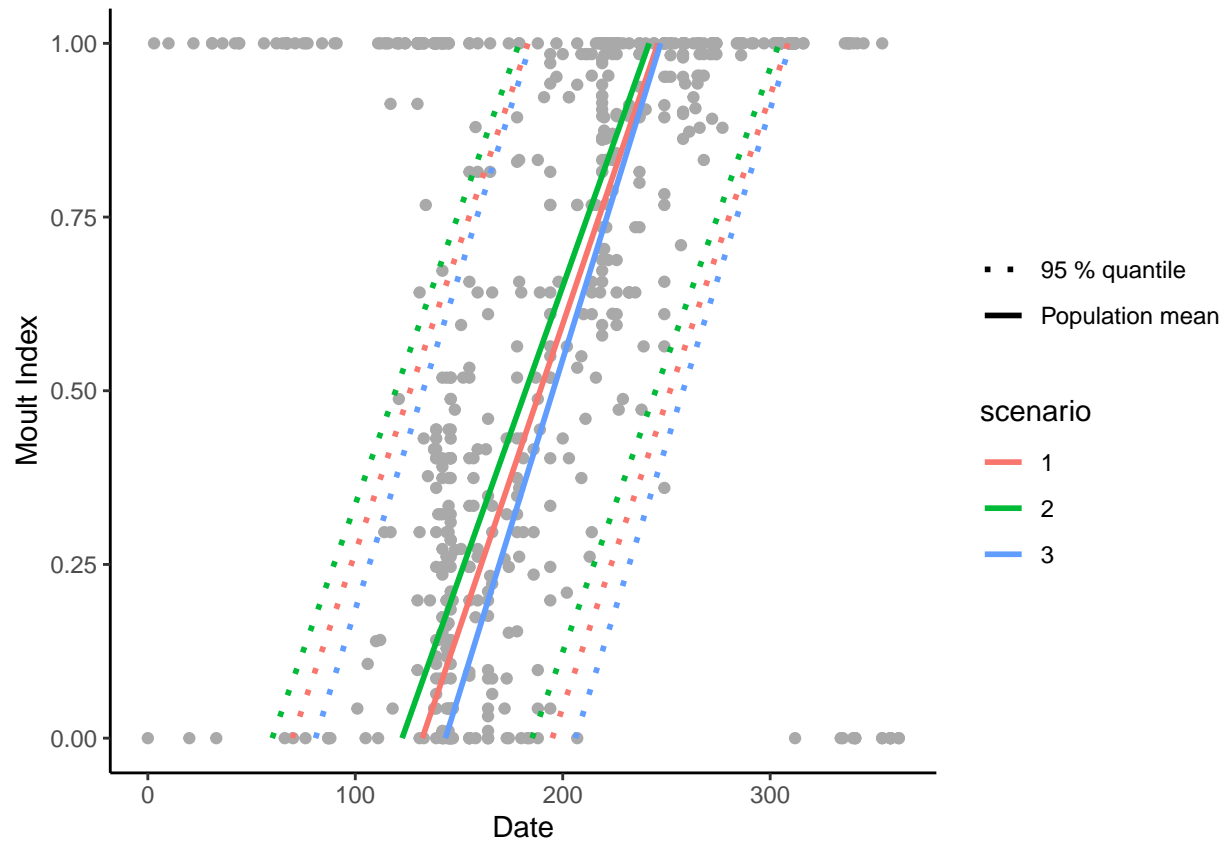
A basic visual summary can be created using the `moult_plot` function. Note that by default this only provides a moult trajectory for the intercepts of the linear predictor. In this particular case this would be the “female” group of the data.

```
moult_plot(T2LR)
```



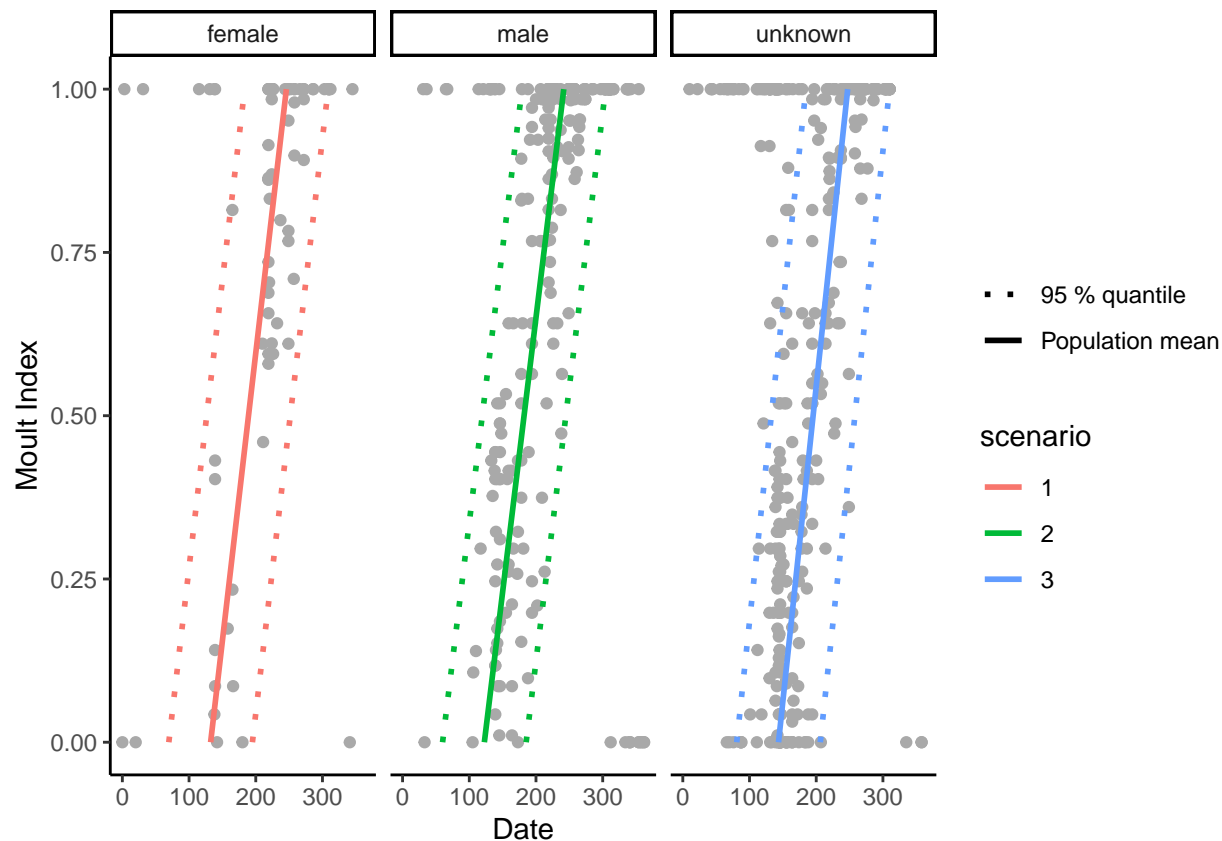
more elaborate plots can be created by providing the desired covariate data to the `newdata` argument of `moult_plot` (in a similar way as most `predict` methods in R work):

```
prediction_data <- data.frame(sex = factor(c('female', 'male', 'unknown')))
moult_plot(T2LR, newdata = prediction_data)
```



And using the facetting capabilities of `ggplot` (<https://ggplot2-book.org/facet.html>), the group-specific estimates can be shown in separate panels as well:

```
moult_plot(T2LR, newdata = prediction_data) + facet_wrap(~sex)
```



Session Information

```
sessionInfo()

## R version 4.2.0 (2022-04-22 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.3.6      moultmcmc_0.0.0.9014
##
## loaded via a namespace (and not attached):
## [1] rstan_2.26.11      tidyselect_1.2.0    xfun_0.30
## [4] lattice_0.20-45    V8_4.1.0            colorspace_2.0-3
```

## [7] vctrs_0.5.1	generics_0.1.3	htmltools_0.5.2
## [10] stats4_4.2.0	loo_2.5.1	yaml_2.3.5
## [13] utf8_1.2.2	rlang_1.0.6	pkgbuild_1.3.1
## [16] pillar_1.8.1	glue_1.6.2	withr_2.5.0
## [19] matrixStats_0.62.0	lifecycle_1.0.3	stringr_1.4.1
## [22] munsell_0.5.0	gtable_0.3.1	codetools_0.2-18
## [25] evaluate_0.15	labeling_0.4.2	inline_0.3.19
## [28] knitr_1.39	callr_3.7.2	fastmap_1.1.0
## [31] ps_1.7.1	parallel_4.2.0	curl_4.3.2
## [34] fansi_1.0.3	highr_0.9	Rcpp_1.0.9
## [37] scales_1.2.1	RcppParallel_5.1.5	StanHeaders_2.26.11
## [40] jsonlite_1.8.0	farver_2.1.1	gridExtra_2.3
## [43] digest_0.6.29	stringi_1.7.8	processx_3.7.0
## [46] dplyr_1.0.99.9000	grid_4.2.0	cli_3.4.1
## [49] tools_4.2.0	magrittr_2.0.3	tibble_3.1.8
## [52] crayon_1.5.1	pkgconfig_2.0.3	prettyunits_1.1.1
## [55] rmarkdown_2.14	rstudioapi_0.13	R6_2.5.1
## [58] nlme_3.1-157	compiler_4.2.0	