Survival Model Diagnostics

Here, we will use two different diagnostic methods to show that the survival model in Stoffel et al. (2020) fits well to the data and is appropriate to investigate the questions in the manuscript.

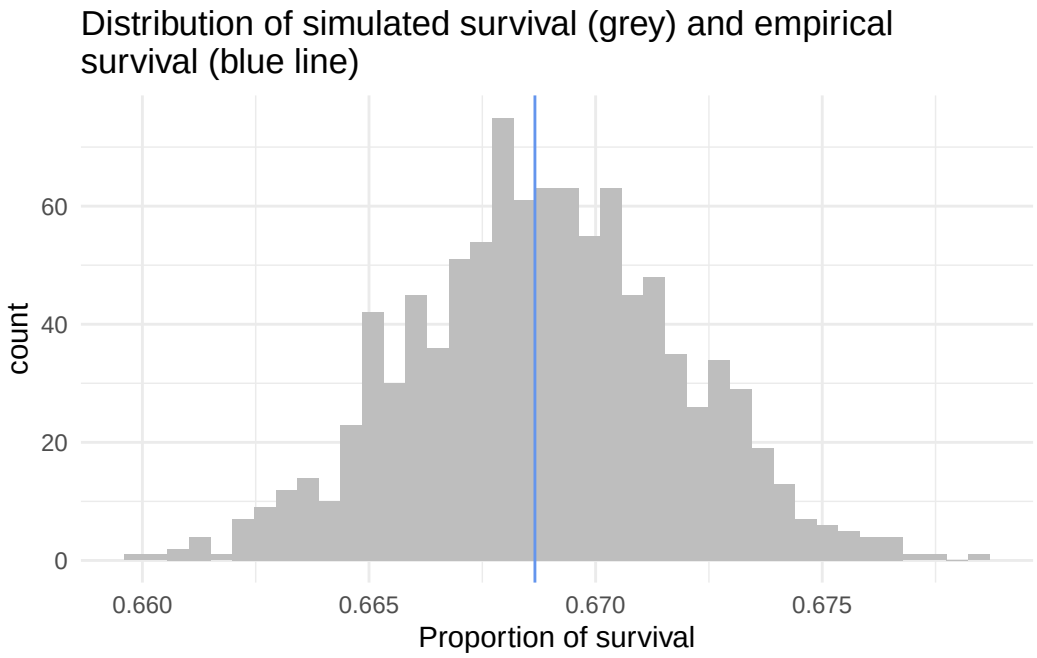
(1) Posterior predictive check

The idea of posterior predictive checks (Gelman,Meng & Stern, 1996) is to simulate new response variables (here: survival) from the fitted model. If the model fits well, these simulated response variables should be similar to the empirical response. As our response is binary, we take the ratio of survival to deaths in the dataset as a statistic. For ease of use, we fit the model in lme4 instead of INLA, without the additive genetic effect, which had a very low variance anyway and does not change the model estimates appreciably.

```
# fit the model
mod1 <- glmer(survival ~ froh_all10_cent * age_cent + froh_all10_cent * lamb +
              sex + twin + (1|sheep_year) + (1|birth_year) + (1|id),
              family = binomial(link = 'logit'), data = annual_survival,
              control = glmerControl(optimizer = "nloptwrap", calc.derivs = FALSE))
```

```
# calculate proportion of survival (i.e. 1s) in the empirical dataset
emp_ratio <- sum(annual_survival$survival) / nrow(annual_survival)
# simulate 1000 new response vectors based on the model
new_resps <- simulate(mod1, nsim = 1000, re.form = NULL)
# for each simulated response vector, calculate the proportion of survival
sim_ratio <- as_tibble(colSums(new_resps)/nrow(new_resps))

# plot simulated distribution and empirical ratio of survival / deaths
ggplot(sim_ratio, aes(value)) +
        geom_histogram(bins = 40, fill = "grey") +
        geom_vline(xintercept = emp_ratio, colour = "cornflowerblue") +
        xlab("Proportion of survival") +
        ggtitle("Distribution of simulated survival (grey) and empirical
survival (blue line)") +
        theme_minimal()
```




```
ggsave("figs/ppc_survival.jpg",)
```

```
## Saving 5.5 x 3.5 in image
```

Simulated survival data from the model is very similar to the empirical survival data, which is a good indicator that the model fits well.

Predicting survival for new observations

As a next check, we will try to predict survival for new observations. To do this, we construct a training and a test dataset. The training dataset will contain 75% of the data in the full model and will be used to fit the model. After this, the fitted model will be used to predict survival in the remaining 25% of the dataset.

```
# get index for 25% of the observations
test <- sample(1:nrow(annual_survival), round(0.25 * nrow(annual_survival)))
# get index for the remaining 75%
train <- c(1:nrow(annual_survival))[-test]
# construct training dataset
df_train <- annual_survival[train, ]
# construct test dataset
df_test <- annual_survival[test, ]

# fit the model using the training dataset
mod <- glmer(survival ~ froh_all10_cent * age_cent + froh_all10_cent * lamb +
             sex + twin + (1|sheep_year) + (1|birth_year) + (1|id),
             family = binomial(link = 'logit'), data = df_train,
             control = glmerControl(optimizer = "nloptwrap", calc.derivs = FALSE))

# use the fitted model to predict survival in the test dataset
preds <- predict(mod, newdata = df_test, allow.new.levels = TRUE, re.form = NULL,
```

```
                type = "response")

# survival is predicted on the probability scale, but is 0/1 in the empirical data,
# to compare them, set to 1 if probability is higher than 0.5 and 0 otherwise
preds <- ifelse(preds > 0.5, 1, 0)
# confusion matrix for survival/death
cm <- as.matrix(table(actual = annual_survival[test, "survival"], predicted = preds))
cm
```

```
##       predicted
## actual    0    1
##      0  848  399
##      1  313 2159
```

```
# calculate the accuracy of the prediction, i.e. the proportion of cases classified
# correctly
accuracy <- sum(diag(cm)) /  sum(cm)
accuracy
```

```
## [1] 0.8085507
```

The model correctly predicts survival in 80.86 % of cases. We can now repeat this analysis 100 times by wrapping it in a function.

```
# prediction
predict_accuracy <- function(rep) {
  test <- sample(1:nrow(annual_survival), round(0.2 * nrow(annual_survival)))
  train <- c(1:nrow(annual_survival))[-test]
  df_train <- annual_survival[train, ]
  df_test <- annual_survival[test, ]
  mod <- glmer(survival ~ froh_all10_cent * age_cent + froh_all10_cent * lamb +
                 sex + twin + + (1|sheep_year) + (1|birth_year) + (1|id) + (froh_all10_cent|age),
               family = binomial(link = 'logit'), data = df_train,
               control = glmerControl(optimizer = "nloptwrap", calc.derivs = FALSE))

  preds <- predict(mod, newdata = df_test, allow.new.levels = TRUE, re.form = NULL,
                   type = "response")
  preds <- ifelse(preds > 0.5, 1, 0)
  cm <- as.matrix(table(actual = annual_survival[test, "survival"], predicted = preds))
  accuracy <- sum(diag(cm)) /  sum(cm)
  accuracy

}

library(future)
plan(multiprocess, workers = 3)
```

```
## Warning: [ONE-TIME WARNING] Forked processing ('multicore') is disabled
## in future (>= 1.13.0) when running R from RStudio, because it is
## considered unstable. Because of this, plan("multicore") will fall
## back to plan("sequential"), and plan("multiprocess") will fall back to
```

```
## plan("multisession") - not plan("multicore") as in the past. For more
## details, how to control forked processing or not, and how to silence this
## warning in future R sessions, see ?future::supportsMulticore
```

```
all_acc <- map_dbl(1:100, predict_accuracy)
```

```
p1 <- ggplot(as_tibble(all_acc), aes(value*100)) +
  geom_histogram(bins = 14, fill = "grey") +
  theme_minimal() +
  xlab("Survival prediction accuracy %")
p1
```