Performing Restricted Cubic Splines in R

Lena Kristin Bache-Mathiesen

Contents

Introduction

This document is intended as a guide for modeling the relationship between a measure of training load and injury in sports research, using Restricted Cubic Splines (RCS). The examples will go through standard logistic regression, and later a mixed effects logistic regression model. However, the steps to model training load using splines in a Poisson model and other regression models are the same. The best-practice information on splines is based on Frank Harrell's "Regression Modeling Strategies" (available here: https://link.springer.com/book/10.1007/978-3-319-19425-7)

Preparation

First, load required packages.

```
library(ggplot2) # for creating figures
library(rms) # Harrell's rms package includes the functions we need for splines
library(lme4) # package for mixed model functions
library(sjPlot) # for plotting predicted values with splines
library(ggeffects) # for model predictions with splines
library(clubSandwich) # for cluster-robust confidence intervals
```

Next step is to load your data. Here we used the d_example_guide.rds data available from the GitHub repository. It is simulated from football data for the example to be reproducible. The object "d" in the r-code below can be replaced with your own data. At the minimum, your data should have:

- one column for load, measured in any metric (such as sRPE, GPS measures or ACWR)
- one column for injury, must either be a logical variable (TRUE/FALSE) or coded (0 for no injury, 1 for injury) to work in our logistic regression model.
- one column for athlete ID, coupling the load values and injuries to the right person

```
d = readRDS("d_example_guide.rds")
```

Standard Logistic Regression

Fitting the model

For a regular logistic regression model, we can write:

```
fit_logistic = glm(injury ~ load, data = d)
```

Running `fit_logistic` or `summary(fit_logistic)` will provide us the results and information from the fit.

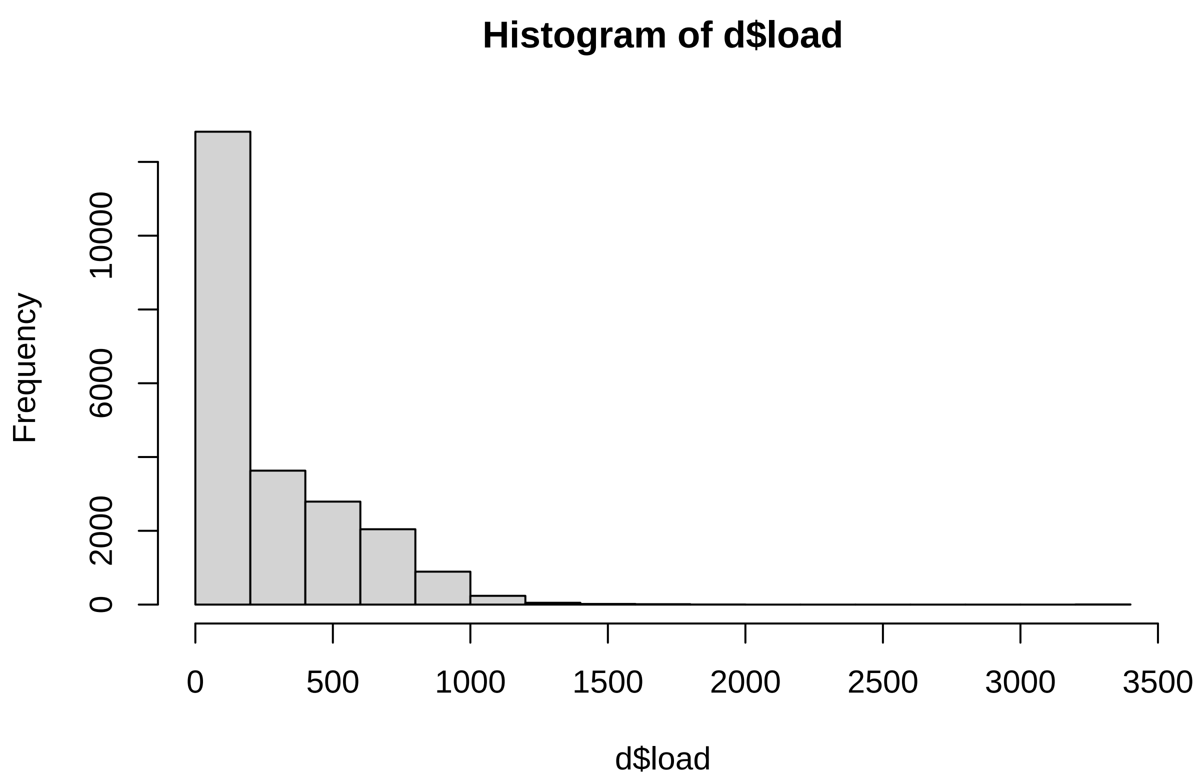
For restricted cubic splines, 3-5 knots are sufficient in the vast majority of cases. Choosing the number of knots can also be determined using Akaike's information Criterion. In this guide, we choose 3. For a restricted cubic splines with 3 knots, we simply write:

```
fit_splines = glm(injury ~ rcs(load, 3), data = d)
```

Here, the placement is determined by the default settings for the rcs function in the rms package. According to the `rms`-package documentation, knots are placed based on quartiles.

We can run the following to create a histogram:

```
hist(d$load)
```



The plot above shows the example data is fairly skewed. Partitioning by quartiles may not be the best fit. Running a histogram to check the data should be done before running the first model.

In the code below, instead of the number of knots, we feed the argument with a vector. The vector lists the locations of where along our load variable we wish our knots to be placed.

```
fit_splines_loc = glm(injury ~ rcs(load, c(500, 1500, 2500)), data = d)
```

We can check which was better using the model with the lowest AIC:

```
AIC(fit_splines)
```

```
## [1] 12355.2
```

```
AIC(fit_splines_loc)
```

```
## [1] 12282.19
```

Since the last AIC was lower, it indicates that placing the knots had a better fit.

We can run `summary()` to recieve the coefficients (Estimate), standard error and p-values of our model:

```
summary(fit_splines_loc)
```

```
##
## Call:
## glm(formula = injury ~ rcs(load, c(500, 1500, 2500)), data = d)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.98867   0.01133   0.01133   0.16780   0.72098
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      9.887e-01  2.787e-03  354.80   <2e-16 ***
## rcs(load, c(500, 1500, 2500))load  -5.795e-04  7.622e-06  -76.03   <2e-16 ***
## rcs(load, c(500, 1500, 2500))load'  6.545e-04  5.713e-05   11.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1010415)
##
##     Null deviance: 2872.2  on 22499  degrees of freedom
## Residual deviance: 2273.1  on 22497  degrees of freedom
## AIC: 12282
##
## Number of Fisher Scoring iterations: 2
```

To obtain Odds Ratios, we can run:

```
exp(coef(fit_splines_loc))
```

```
##                        (Intercept)  rcs(load, c(500, 1500, 2500))load
##                          2.6876561                          0.9994207
## rcs(load, c(500, 1500, 2500))load'
##                          1.0006547
```

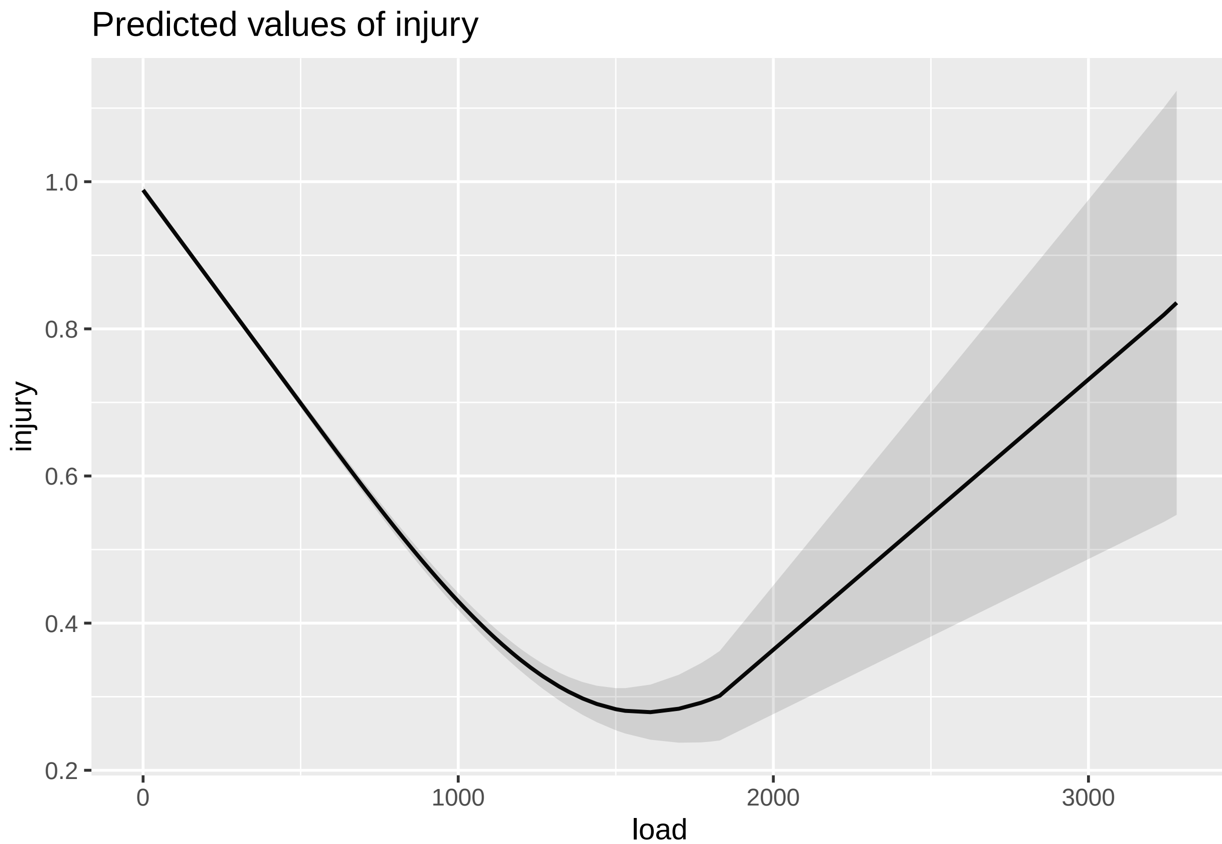
Even should we transform our estimates to an Odds Ratio, the coefficients from a splines-results make little sense. The p-values can be used and understood as usual.

Visualization

The best way to interpret splines is by visualizing predictions.

The sjPlot package has a handy function To create a simple form of ggplot2-plot:

```
plot_model(fit_splines_loc, type = "pred", terms = "load [all]")
```



Mapping the load values to the predicted probabilities (figure above) showed that the splines modelled a U-shaped relationship for the example data.

Mixed Effects Regression Model

Fitting the model

A mixed effects logistic regression model is a bit more complicated. Most notably, because running a general linear mixed model (GLMM) isn't part of base R. Here, we use the lme4 package. Not everything in base R or other packages is compatible with the lme4 package.

For a mixed model with binomial distribution and random intercept per athlete one can run:

```
fit_mixed = glmer(injury ~ load + (1 | p_id), family = "binomial", data = d)
```

For a mixed model with binomial distribution, random intercept and random slope per athlete:

```
fit_mixed_slope = glmer(injury ~ load + (load | p_id), family = "binomial", data = d)
```

Finally, our splines model with knots placed where we think they should be, based on our histogram in the previous section. Note that GLMMs can take some time to run. In this example we use an intercept-only model, as the random slope model failed to converge.

```
fit_mixed_splines = glmer(injury ~ rcs(load, c(500, 1500, 2500)) + (1 | p_id),
                          family = "binomial", data = d)
```

Visualization

A visualization can be created in the same manner as above with `plot_model()` However, since this is a mixed model, we can calculate cluster-robust confidence intervals which take into account the uncertainty stemming from random effects variance. We can do this by calculating the predictions with `ggpredict()` and specifying the function for obtaining the variance-covariance matrix from the model. We use `vcovCR()` from the `clubSandwich` package and specify the clusters in a list in the `vcov.args` argument.

```
preds = ggpredict(
  fit_mixed_splines,
  "load [all]",
  vcov.fun = "vcovCR",
  vcov.type = "CR0",
  vcov.args = list(p_id = d$p_id),
  type = "re.zi"
  )
plot(preds)
```