Heat Increment of Feeding in Dolphins

Ioulia Koliopoulou, Stacy DeRuiter, Andreas Fahlman, et al.

Revisions: take out extra/misstep stuff; use predictor % over daily calories

Read in data

```
HIF_data <- read_xlsx('data/updated_data.xlsx',
                      .name_repair = 'unique_quiet') |>
  mutate(animal = factor(animal))
```

Model fitting

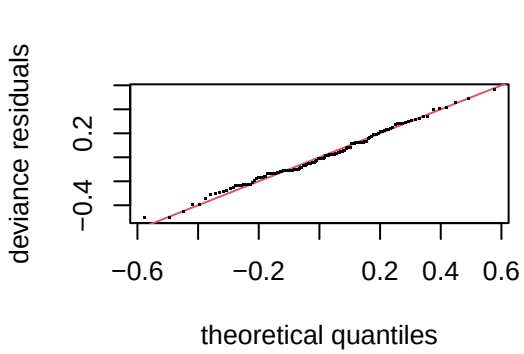
```
HIF_model <- gam(oxygen_cons ~ s(exact, k = 5) +
                   s(percentdailytotal, k = 4) +
                   s(age, k = 4) + sex + s(pool_temp, k = 4) +
                   s(animal, bs = 're'),
                 data = HIF_data,
                 method = "ML", select = TRUE)
summary(HIF_model)
```

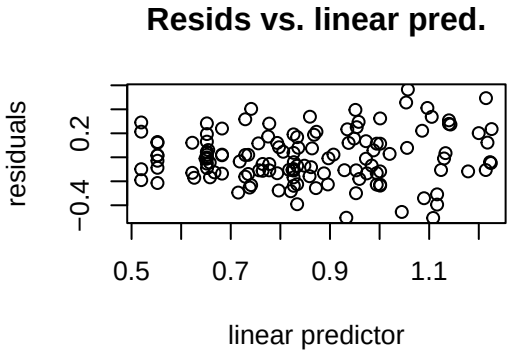
```
Family: gaussian
Link function: identity

Formula:
oxygen_cons ~ s(exact, k = 5) + s(percentdailytotal, k = 4) +
    s(age, k = 4) + sex + s(pool_temp, k = 4) + s(animal, bs = "re")

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

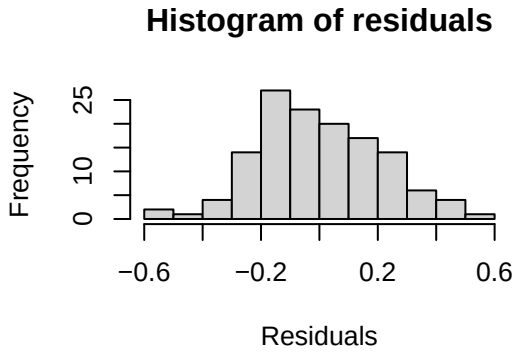
```
(Intercept)   0.6771     0.1016   6.663 7.92e-10 ***
sexM          0.2074     0.1180   1.757   0.0814 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                           edf Ref.df     F  p-value
s(exact)             2.049e+00      4 6.151 8.35e-05 ***
s(percentdailytotal) 5.786e-01      3 1.003    0.258
s(age)               1.196e-06      3 0.000    0.764
s(pool_temp)         1.560e-06      3 0.000    0.967
s(animal)            4.883e+00      6 8.775  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.454   Deviance explained = 48.9%
-ML = -4.4344  Scale est. = 0.046691  n = 133
```

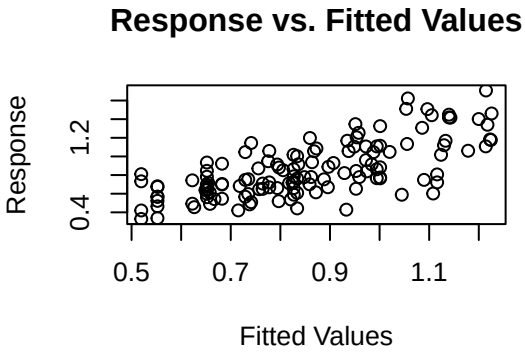
Model Checking

```
gam.check(HIF_model)
```





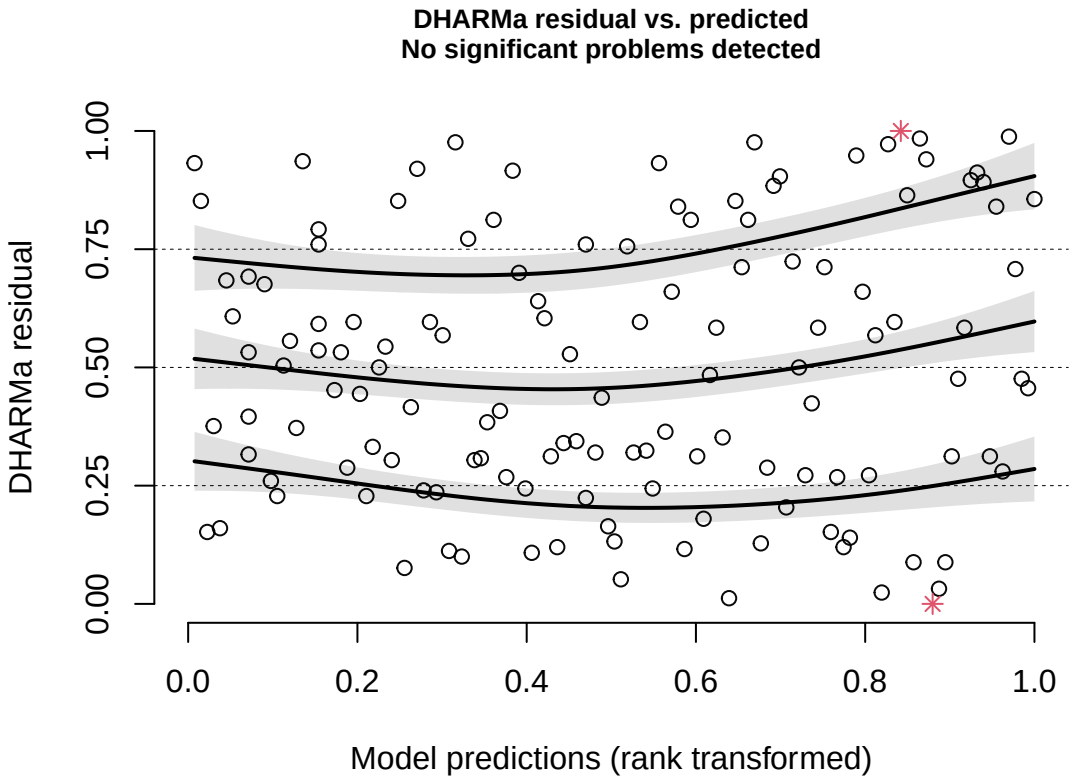




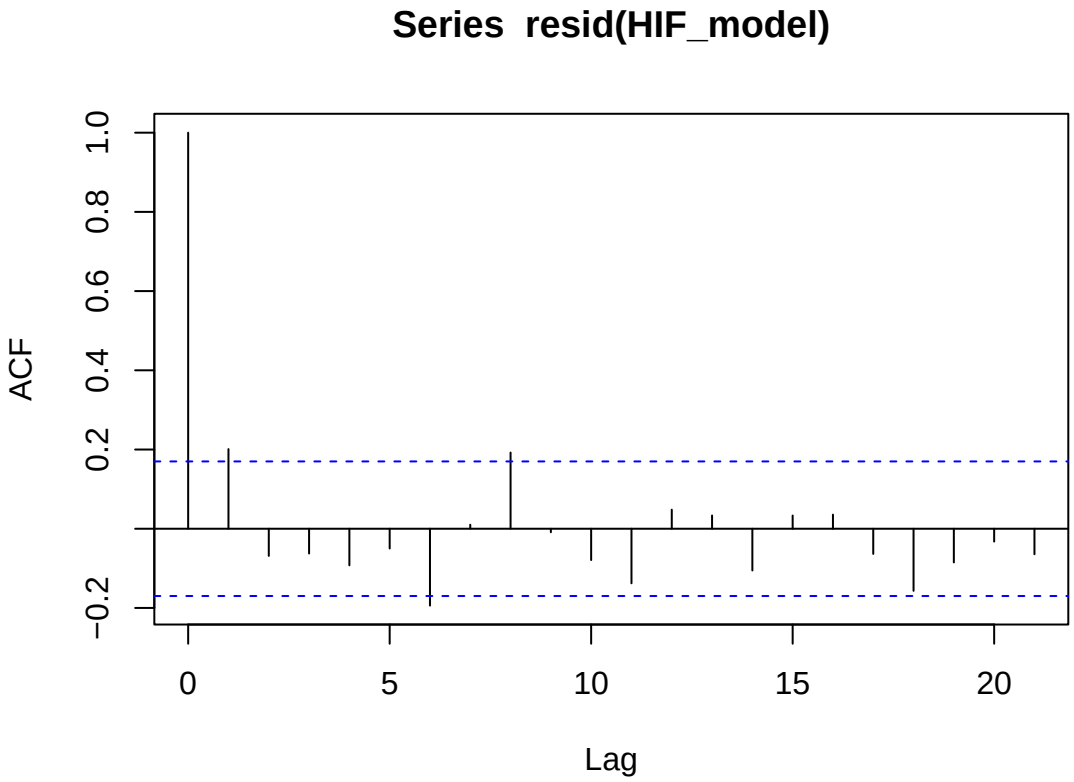

```
Method: ML   Optimizer: outer newton
full convergence after 14 iterations.
Gradient range [-5.759846e-07,8.287999e-07]
(score -4.434367 & scale 0.04669088).
Hessian positive definite, eigenvalue range [2.218469e-08,66.68651].
Model rank =  23 / 23

Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.

                          k'      edf k-index p-value
s(exact)             4.00e+00 2.05e+00    1.09   0.845
s(percentdailytotal) 3.00e+00 5.79e-01    0.80   0.015 *
s(age)               3.00e+00 1.20e-06    0.83   0.025 *
s(pool_temp)         3.00e+00 1.56e-06    0.82   0.010 **
s(animal)            8.00e+00 4.88e+00      NA      NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plotResiduals(simulateResiduals(HIF_model))
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2

Registered S3 methods overwritten by 'mgcViz':
  method       from
  +.gg         GGally
  simulate.gam gratia
```




```
acf(resid(HIF_model))
```



There are no apparent issues with model conditions (independence and normality of residuals, constant variance of residuals).

Hypothesis Testing

```
anova(HIF_model)
```

```
Family: gaussian
Link function: identity

Formula:
oxygen_cons ~ s(exact, k = 5) + s(percentdailytotal, k = 4) +
    s(age, k = 4) + sex + s(pool_temp, k = 4) + s(animal, bs = "re")

Parametric Terms:
    df     F p-value
sex  1 3.087  0.0814
```

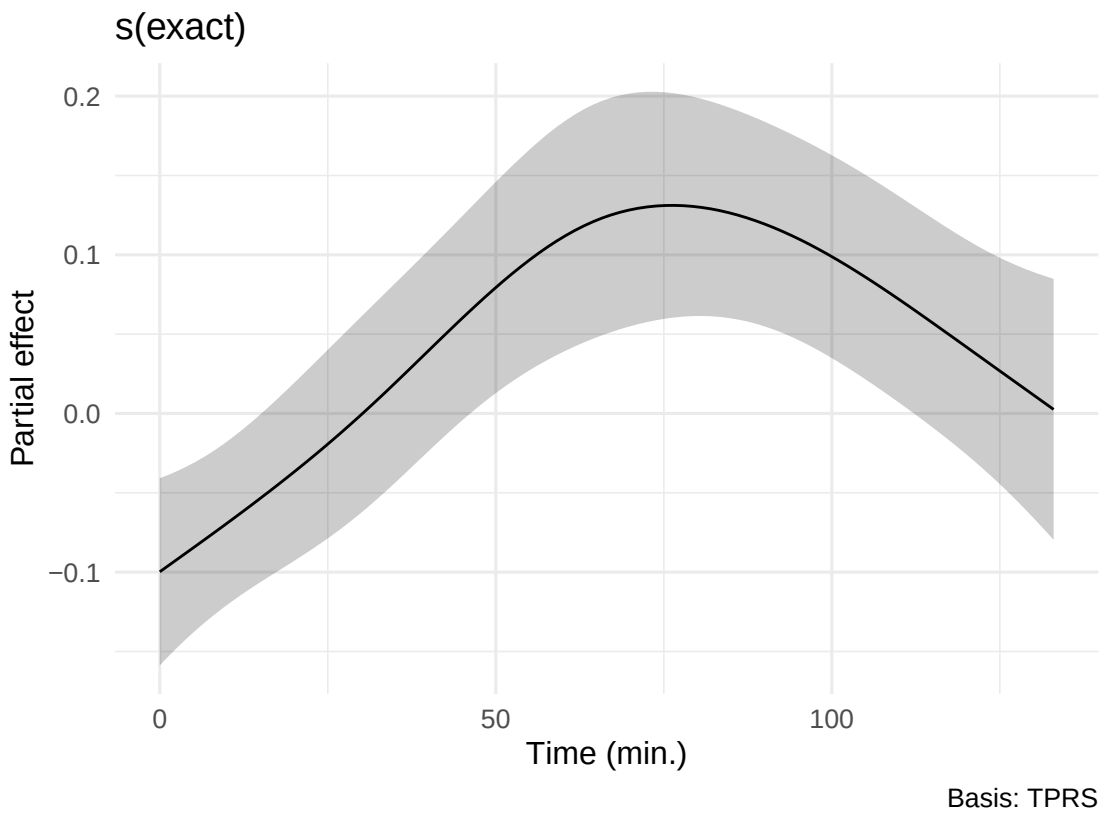
```
Approximate significance of smooth terms:
                           edf    Ref.df     F  p-value
s(exact)             2.049e+00 4.000e+00 6.151 8.35e-05
s(percentdailytotal) 5.786e-01 3.000e+00 1.003    0.258
s(age)               1.196e-06 3.000e+00 0.000    0.764
s(pool_temp)         1.560e-06 3.000e+00 0.000    0.967
s(animal)            4.883e+00 6.000e+00 8.775  < 2e-16
```

Model predictions

Partial Smooths

This is a partial plot for the smooth for `Exact`:

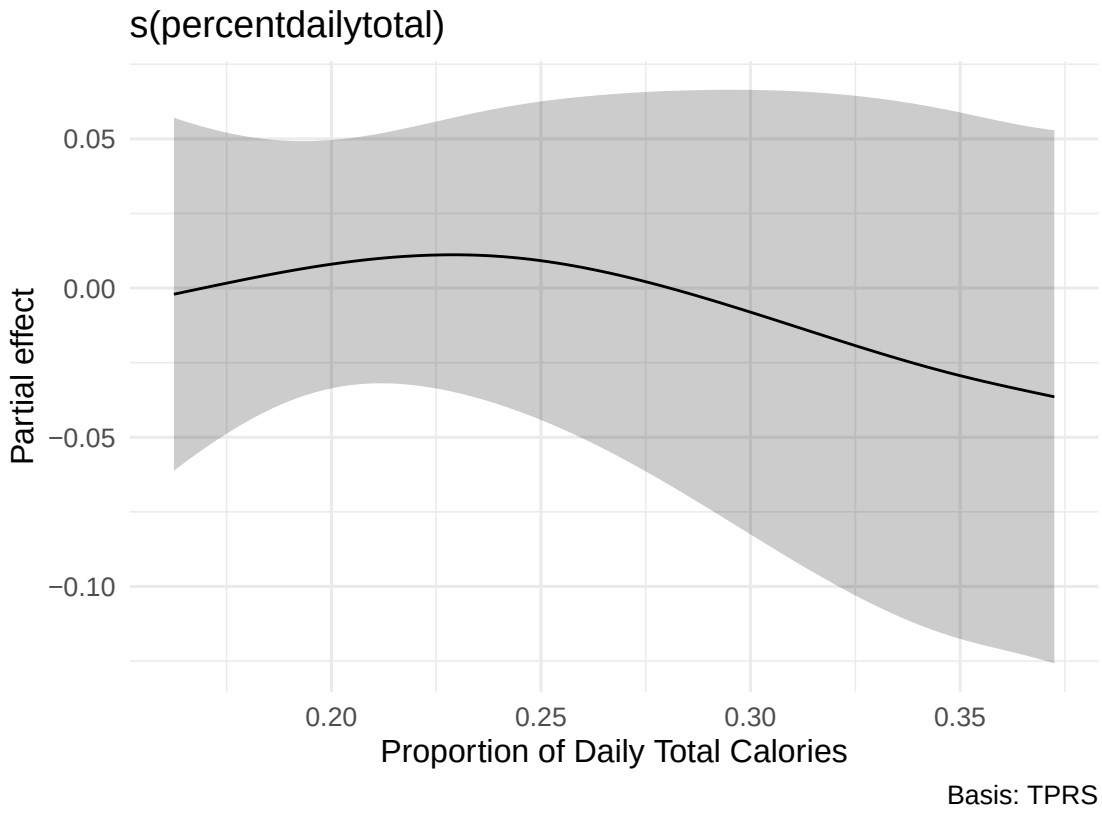
```
exact_partial_smooth <- smooth_estimates(HIF_model, select = 's(exact)')
draw(exact_partial_smooth) + xlab('Time (min.)')
```



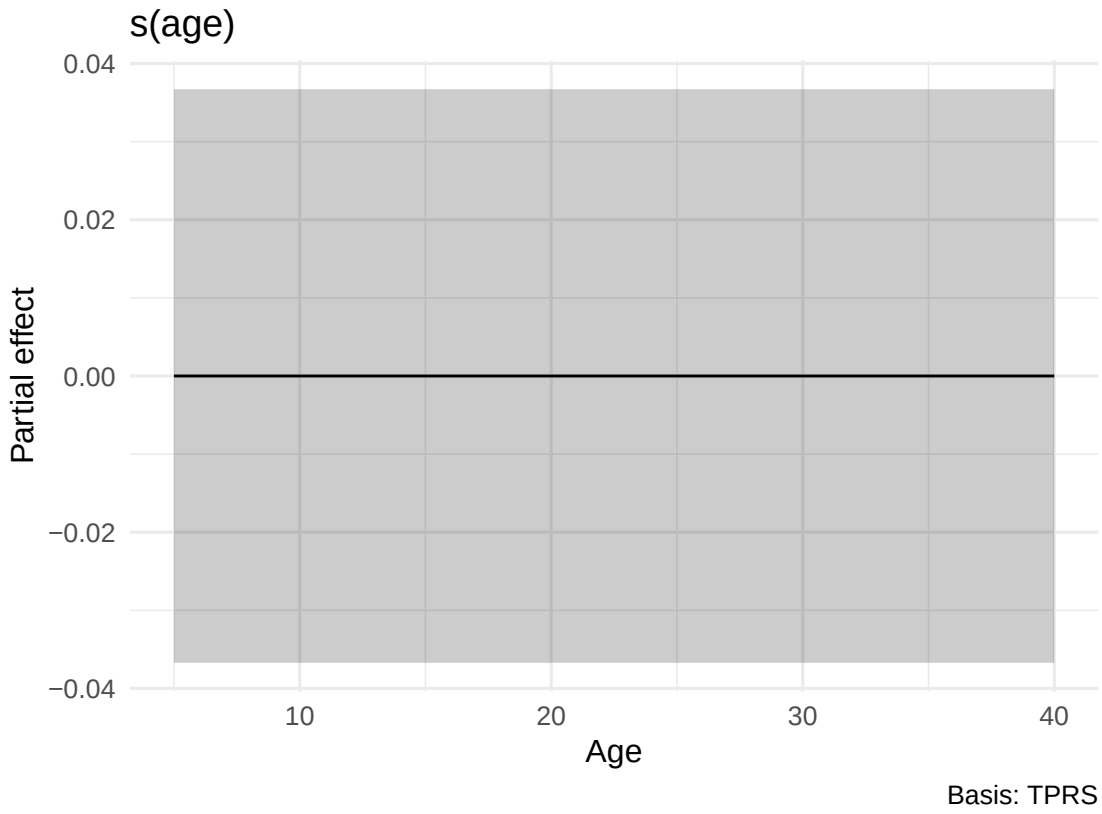
In a partial plot, the units of the y-axis are the same as the units of the response, but the values are not actually the expected oxygen consumption – it's the amount added on to (or taken away from) the oxygen consumption total due to the value of `exact`. It shows the shape of the relationship but the a realistic oxygen consumption total – only a relative value.

We can do the same for other predictors - but note that the ANOVA did not indicate evidence of a detectable association between any of them, and the oxygen consumption.

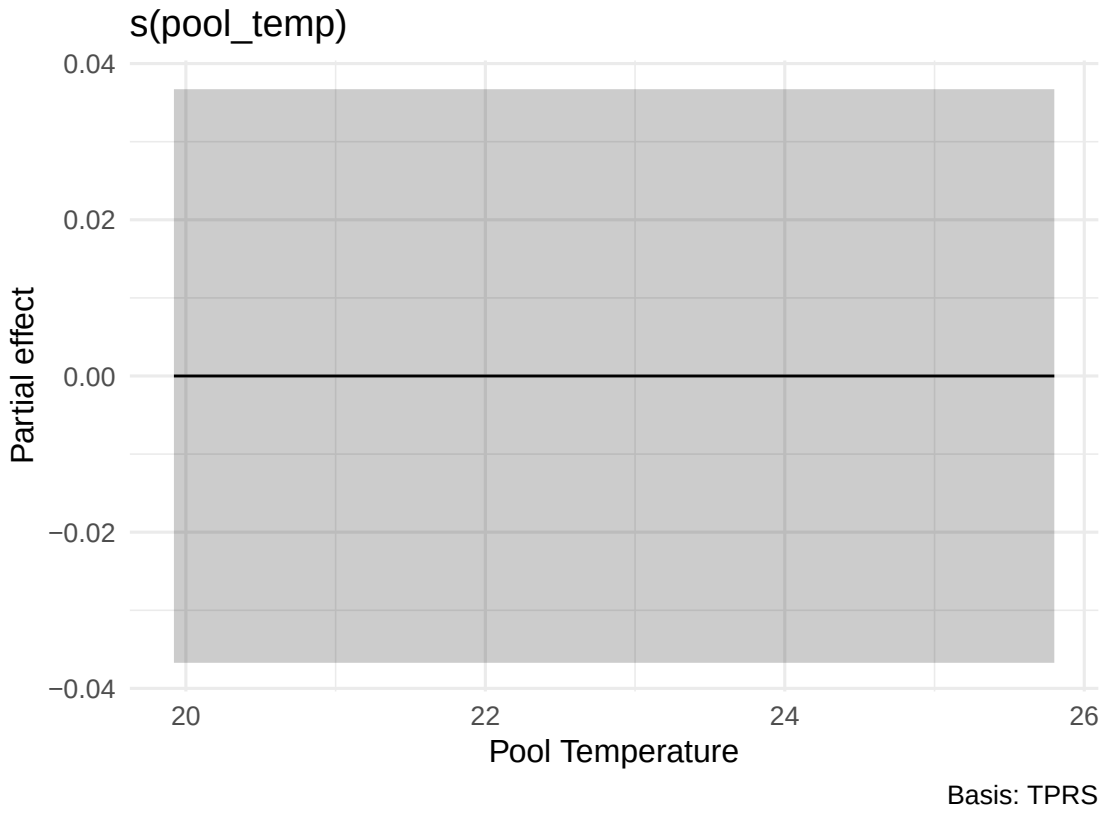
```
exact_partial_smooth <- smooth_estimates(HIF_model, select = 's(percentdailytotal)')
draw(exact_partial_smooth) + xlab('Proportion of Daily Total Calories')
```




```
exact_partial_smooth <- smooth_estimates(HIF_model, select = 's(age)')
draw(exact_partial_smooth) + xlab('Age')
```




```
exact_partial_smooth <- smooth_estimates(HIF_model, select = 's(pool_temp)')
draw(exact_partial_smooth) + xlab('Pool Temperature')
```



Prediction plots

Another way of visualizing the results of a GAM (or other regression model) is a prediction plot, where you select specific values at which to fix all the predictors other than `exact` time (age, proportion total calories, etc.) and then show predicted oxygen consumption given those values, and varying values of exact. Such a plot shows the same shape, but perhaps more easy-to-understand values on the y axis.

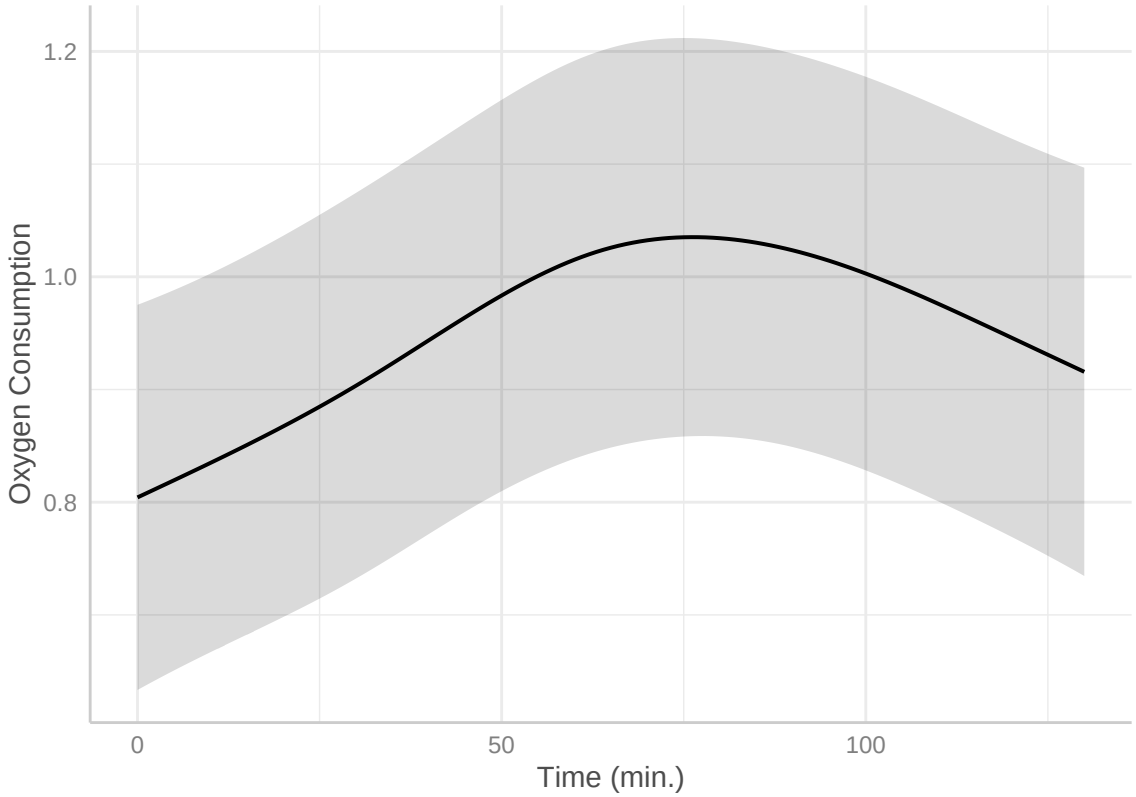
The table below shows some such predictions, but the important part to us is the notes at the bottom, which specify what fixed values were used for other predictors:

- percentdailytotal = 0.23
- age = 21.4
- sex: M (most common in data)
- pool_temp: 22.8
- the input `type = 'fixed'` means that the output will be population-level predictions for an average individual (so although one individual's name is listed below, it's not actually used – predictions are for a hypothetical average dolphin, not for `Tt1`.)

```r
ggpredict(HIF_model,
          terms = 'exact',
          type = 'fixed')
```

```
# Predicted values of oxygen_cons

exact | Predicted |     95% CI
------------------------------
    0 |      0.80 | 0.63, 0.98
   31 |      0.91 | 0.74, 1.08
   38 |      0.94 | 0.76, 1.11
   61 |      1.02 | 0.84, 1.19
   67 |      1.03 | 0.85, 1.21
  108 |      0.98 | 0.81, 1.16
  119 |      0.95 | 0.77, 1.13
  133 |      0.91 | 0.72, 1.09

Adjusted for:
* percentdailytotal =  0.23
*               age = 21.40
*               sex =     M
*         pool_temp = 22.80
*            animal =   Tt1


Not all rows are shown in the output. Use `print(..., n = Inf)` to show
  all rows.
```

```r
ggpredict(HIF_model,
          terms = 'exact [0:130]',
          type = 'fixed') |>
  plot() +
  ggtitle("") + xlab('Time (min.)') + ylab('Oxygen Consumption')
```



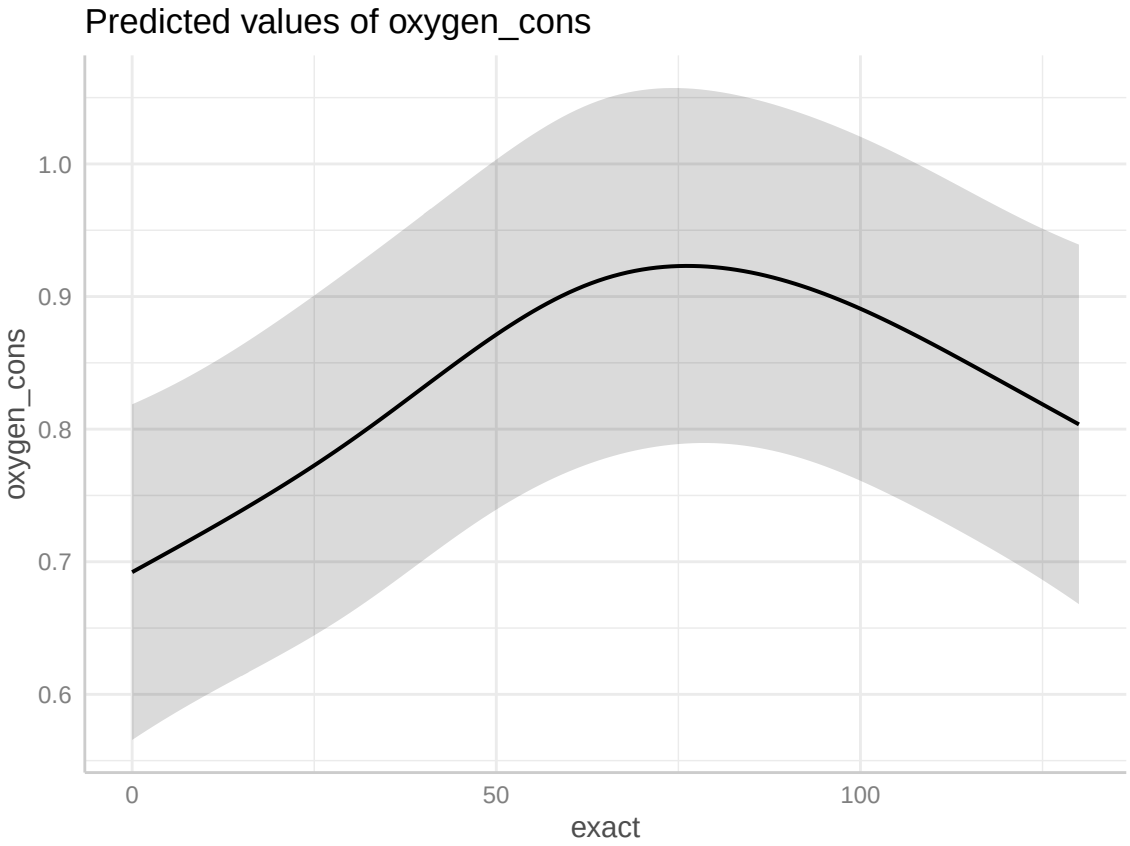
In a prediction plot like this, the CIs will be wider because they include uncertainty in the intercept and the estimates of the effects of other predictors as well as just the one being shown.

Predictions for marginal means

Another way to show predictions treats "other" predictors in the model in yet another way. In this approach quantitative predictors are set to the mean value observed in the data, but this is a weighted average across the observed values of all categorical predictors. The idea is to try to show the expected response value for "an average observation" (in some sense) in the actual data…

This would look like:

```
mmeans <- predict_response(HIF_model, terms = 'exact [0:130]',
                margin = 'marginalmeans')

mmeans |>
  plot()
```



(looks very similar to above but determined in a slightly different way; and this one *might* make more sense to overlay on top of the data…)

Data Plot with Model Fit

Setup

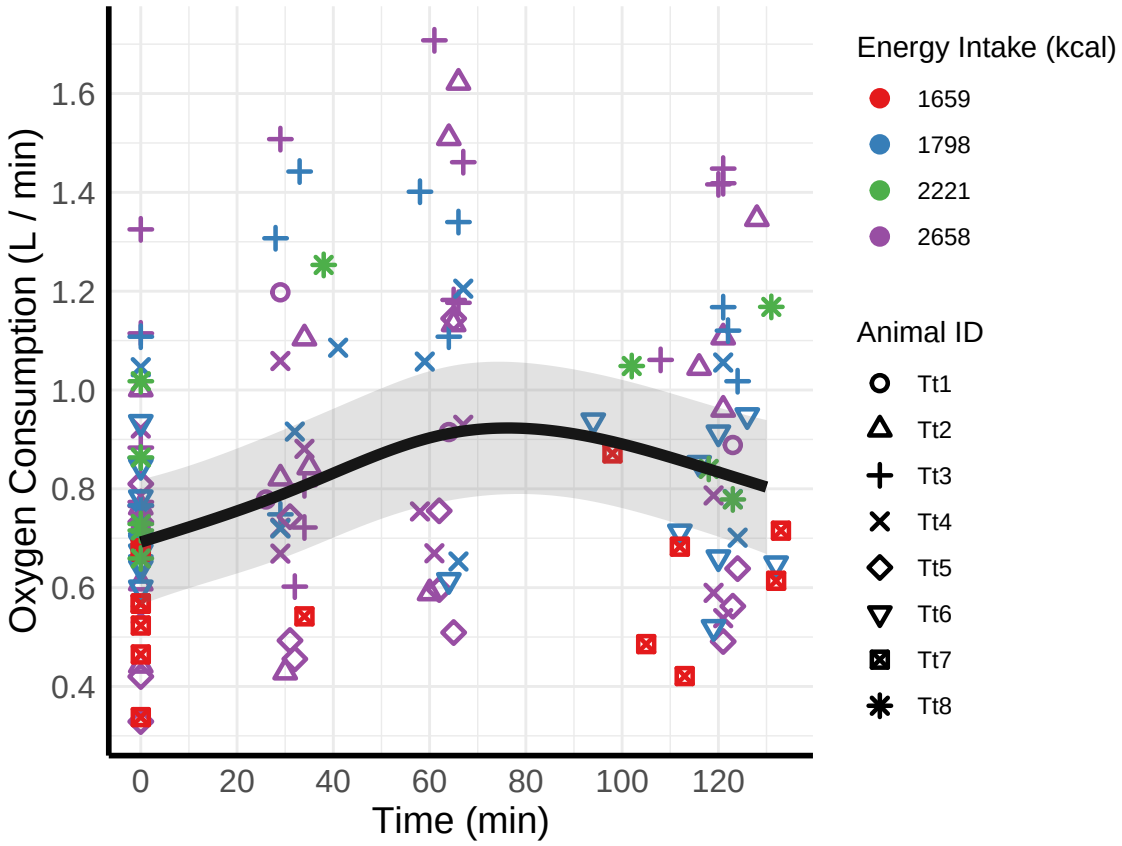
Define colors and shapes to use for plotting

```
kcal_colors <- c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3")
shape_vector <- c("Tt1" = 1, "Tt2" = 2, "Tt3" = 3, "Tt4" = 4, "Tt5" = 5, "Tt6" = 6, "Tt7" =
```

Data plot with maringal mean predictions

If it was really desirable to "draw a line" showing the model expectations on top of the graph above, maybe the marginal means predictions would be the thing to show.

```r
oxygen_time_plot <- ggplot(HIF_data,
                           aes(x = exact, y = oxygen_cons)) +
  geom_point(aes(color = factor(kcal), shape = animal),
             size = 2,
             stroke = 1.2) +
  geom_line(data = mmeans,
            aes(x = x, y = predicted),
            color = 'black',
            linewidth = 2) +
  geom_ribbon(data = mmeans,
              aes(ymin = conf.low, ymax = conf.high, x = x),
              inherit.aes = FALSE,
              fill = 'grey44', alpha = 0.2) +
  labs(x = "Time (min)",
       y = "Oxygen Consumption (L / min)",
       color = "Energy Intake (kcal)",
       shape = "Animal ID") +
  scale_color_manual(values = kcal_colors) +
  scale_shape_manual(values = shape_vector) +
  theme_minimal() +
  theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        axis.line = element_line(linewidth = 1)) +
  scale_x_continuous(breaks = seq(0, 120, by = 20)) +
  scale_y_continuous(breaks = seq(0, 2, by = 0.2))

print(oxygen_time_plot)
```



Area Under Curve

The calculations below show the AUC under the black line in the graph above.

For additional reference if needed, the initial value of oxygen consumption at time 0 is: 0.6921595

Cumulative AUC?

It might be of interest to compute the AUC every so often (not just the total over all 100+ minutes)?

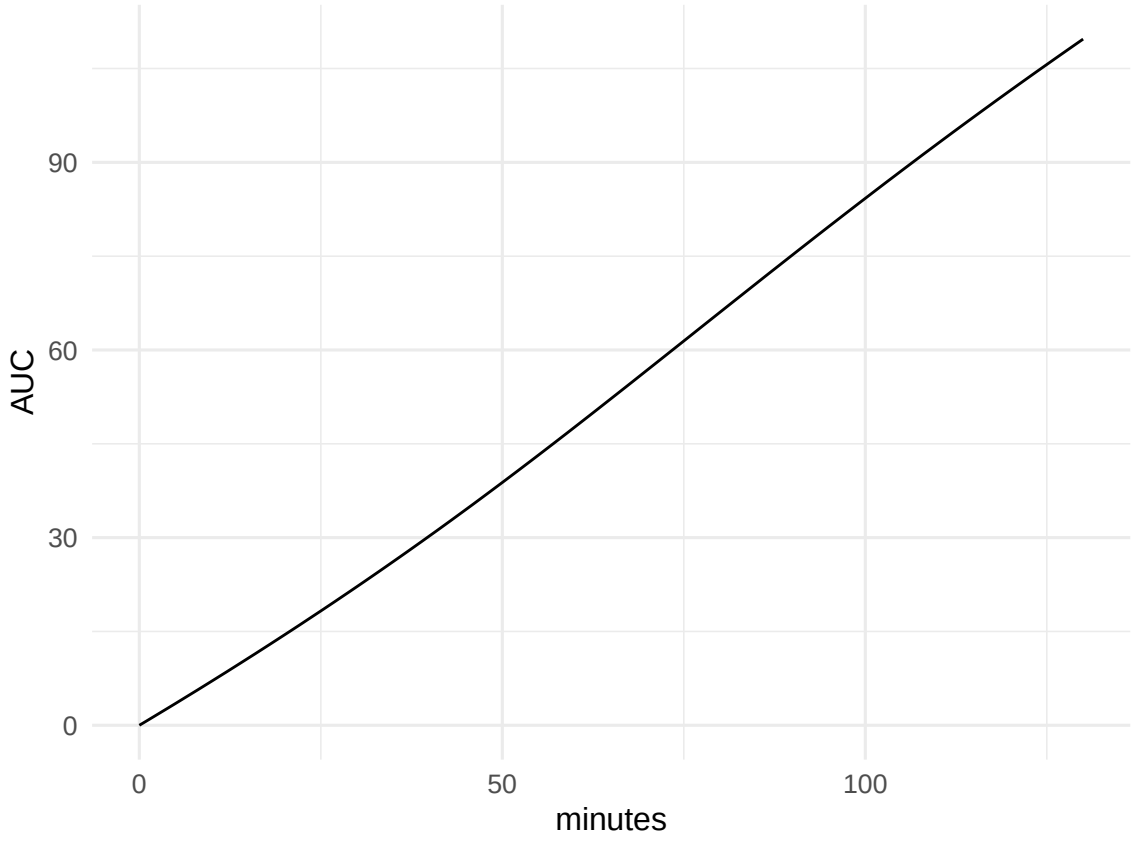
If so – and using the marginal mean expected 0_2 consumption values, we could do something like:

```
predicted_AUC <- data.frame(minutes = mmeans$x,
                            AUC = cumtrapz(x = mmeans$x,
                                           y = mmeans$predicted))
max(predicted_AUC$AUC)
```

```
[1] 109.7025
```

The total at 130 minutes is: 109.7025268. Below the cumulative sums are shown:

```
ggplot(predicted_AUC,
       aes(x = minutes, y = AUC)) +
  geom_path()
```



To compute this quantity with uncertainty, we would probably employ a parametric bootstrap (not done yet).