MTHM506 - Statistical Data Modelling

Topic 3 - A GAM example

Preliminaries

In this session, we consider an example of fitting a GAM in `R`. In the last session we discussed the theory of GAMs and how it relates to GLMs, where covariates enter the model in flexible way.

Here we show how it works in practice, illustrating the similarities between GLMs and GAMs. These notes refer to Topics 3.1-3.8 from the lecture notes. All practical aspects will be done using the `globalMeanTemp` dataframe in `datasets.RData`.

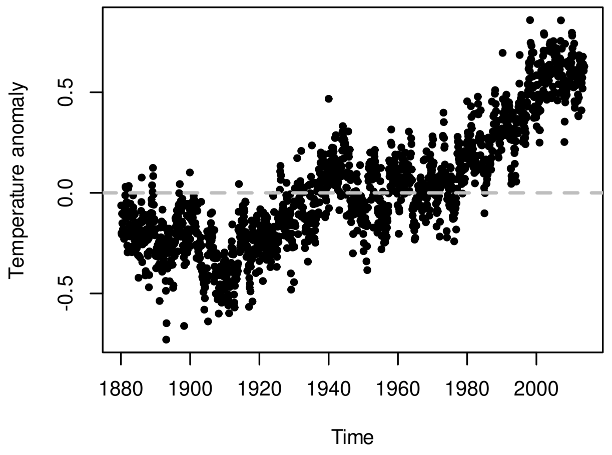
```
# Load the data
load('datasets.RData')
library(mgcv)
```

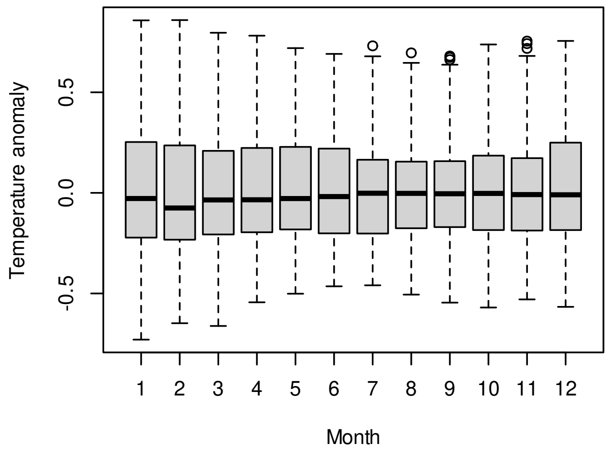
Example: `globalMeanTemp` dataframe

The data relates to monthly temperature anomalies (average global mean temperature minus the mean across the time period) between 1880 and 2013. Interest lies in modelling the temperature anomalies over time and seeing if global mean temperature is increasing.

```
# Fist 6 rows of globalMeanTemp
head(globalMeanTemp)
  year month    temp timeStep
1 1880     1 -0.0627        1
2 1880     2 -0.1998        2
3 1880     3 -0.2067        3
4 1880     4 -0.1034        4
5 1880     5 -0.1508        5
6 1880     6 -0.2012        6

# Create time step for nice plot
globalMeanTemp$Time <- globalMeanTemp$year + globalMeanTemp$month/12

# Split plot
par(mfrow = c(1,2))
# plot time series
plot(globalMeanTemp$Time, globalMeanTemp$temp, pch=20,
```

```r
     xlab = 'Time', ylab = 'Temperature anomaly')
abline(h=0,col="grey",lwd=2,lty=2)

# Looking at a boxplot to see if there is a seasonal sycle
boxplot(temp ~ month, data = globalMeanTemp,
        xlab = 'Month', ylab = 'Temperature anomaly')
```





This is a dataset that is often used as evidence of global warming. The plot indicates that the global mean temperature is consistently increasing. The question is: can we estimate the trend along with any associated uncertainty? For instance, are temperatures signifcantly higher in 2013 than in 1880?

These are temperature anomalies so we can use the Normal distribution as the basis for a model. However it is clear that the mean is a highly non-linear function of time. This is a classic example where we would use a GAM. Let's formulate a model:

$$\begin{aligned} Y_t &\sim N(\mu_t, \sigma^2), \quad Y_t \text{ indep.} \\ \mu_t &= \beta_0 + f(t) \end{aligned}$$

where Y_t is the global mean temperature anomalies, t is the monthly time step, and $f(t)$ is a smooth function. Note that it's better to write the intercept β_0 on its own, so that the function $f(t)$ is centered on zero. (This is happens automatically.)

We can fit this model using `gam()`, which is uses the same formulation as `glm()`. The smooth functions of any covariate `x` are specified via `s(x)`:

```r
# Fit the model
Amodel <- gam(temp ~ s(timeStep, k = 4, bs = "cs"),
              data = globalMeanTemp,
              family = gaussian(link = "identity"))

# Summarise the model
summary(Amodel)
```

```
Family: gaussian
Link function: identity

Formula:
temp ~ s(timeStep, k = 4, bs = "cs")

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.025102   0.003623   6.929 6.13e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
              edf Ref.df    F p-value
s(timeStep) 2.923      3 1749  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.766   Deviance explained = 76.6%
GCV = 0.021131  Scale est. = 0.02108   n = 1606
```

The number of knots `k` is the number of coefficients β_k that make up the smooth function **minus 1**. This minus 1 is the center-to-zero constraint. So with `k=4`, our smooth spline function of time step will have 3 coefficients. The type of basis function is chosen via `bs`, here chosen to be a cubic spline with `bs=cs`.

The above code has fitted the model, estimated the parameters and found an optimal λ to penalise the smooth function/likelihood:

```
Amodel$sp
s(timeStep)
   11.25057
```

The output is very similar to a GLM but we now get two tables:

1) for any parametric coefficients we might have in our model (in this case we only have an intercept)

2) for any smooth function.

Let's proceed to understand the output.

Parameter inference

The parametric coefficients table is exactly the same as in a GLM. Remember that for Normal GLMs we perform hypothesis testing (e.g. $H_0 : \beta_i = a$) on the basis that if H_0 is true

$$T = \frac{\hat{\beta}_i - a}{SE(\hat{\beta}_i)} \sim t_{n-p-1}$$

With penalised likelihood, conditional on the penalty parameters, we replace the $p+1$ parameters in our model with the EDF

$$T = \frac{\hat{\beta}_i - a}{SE(\hat{\beta}_i)} \sim t_{n-EDF}$$

The smooth function table provides inference on the smooth functions **as a whole**. It does not give us information on the β_k inside the function, as they are often not really interpretable. It does report on the approximate significance of the function, which is effectively the result of a likelihood ratio test comparing the model with another that does not include $f(\cdot)$.

Here, the p-value of $f(\cdot)$ is much less than 0.05 so the **function as a whole** is significantly different from zero.

The `gam()` function also contains everything else of course. The estimate of σ^2:

```
Amodel$sig2
[1] 0.02107975
```

the three coefficient estimates of $f(\cdot)$:

```
Amodel$coefficients
  (Intercept) s(timeStep).1 s(timeStep).2 s(timeStep).3
   0.02510205   -0.04367741    0.22284245    0.66901910
```

the rank of the spline:

```
Amodel$rank
[1] 4
```

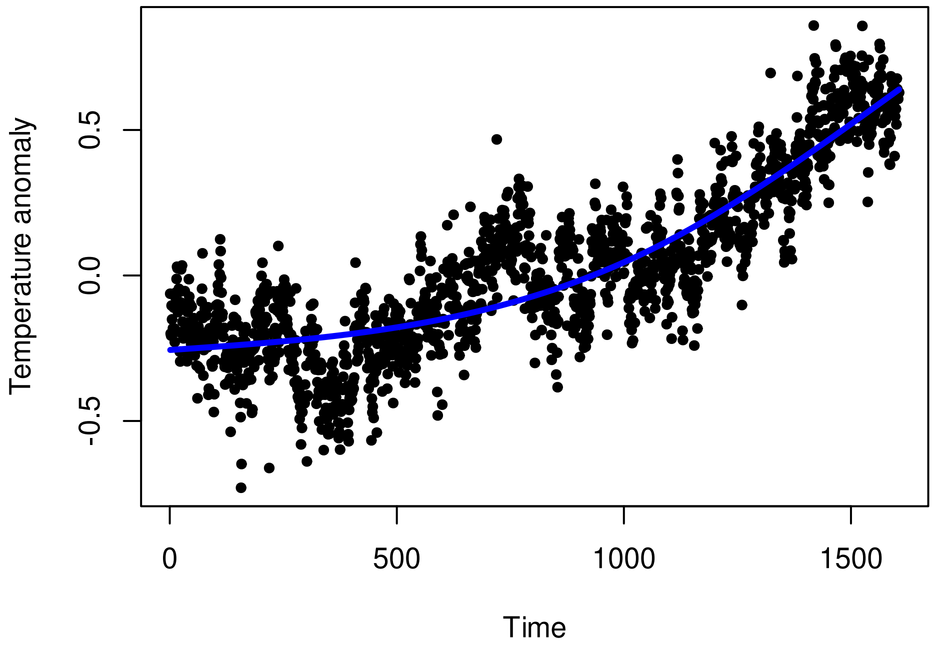
and so on.

Prediction

Assuming the smoothing parameter λ is fixed and known, we predict in the same way as GLMs, i.e. $\hat{y}_i = \hat{\mu}_i$. As before, we can use the `predict()` function to predict the mean at various values of the covariate (monthly time step):

```
# grid of monthly time steps
xx <- seq(min(globalMeanTemp$timeStep),max(globalMeanTemp$timeStep),length=200)

# Predict model mean
yfitAM <- predict(Amodel,newdata=data.frame(timeStep=xx))
```

```r
# Plot model results
par(mar = c(4, 4, 1, 1),cex=1.2)
plot(globalMeanTemp$timeStep,globalMeanTemp$temp,pch=20,
     xlab = 'Time', ylab = 'Temperature anomaly')
lines(xx, yfitAM, col = "blue",lwd=3)
```



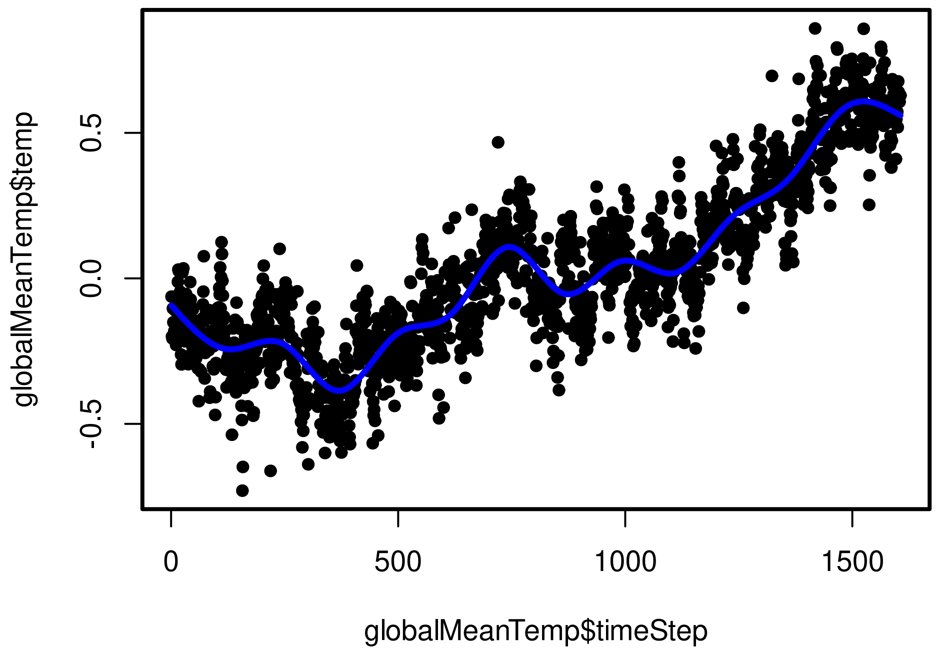
A few things to notice from the plot:

1) Adding unknown non-parametric smooth functions creates a powerful way to model complex relationships in data with little subjective input.
2) It looks like we don't have enough flexibility in our mean. In other words, the rank of the model may be too small. The mean picks up the overall trend yes but it is not flexible enough to pick up **decadal variability**.

Let's increase the rank from $K = 4$ to $K = 14$ to increase flexibility.

```r
# Fit the model
Amodel2 <- gam(temp ~ s(timeStep, k = 14, bs = "cs"),
              data = globalMeanTemp,
              family = gaussian(link = "identity"))

# Produce fitted line from this model using predict
xx <- seq(min(globalMeanTemp$timeStep),max(globalMeanTemp$timeStep),length=200)

# Predicting model mean
yfitAM <- predict(Amodel2, newdata=data.frame(timeStep=xx))

# Plot
```

```r
par(mar = c(4, 4, 1, 1),cex=1.2,lwd=2)
# Plot data
plot(globalMeanTemp$timeStep, globalMeanTemp$temp, pch = 20)
lines(xx, yfitAM, col = "blue",lwd=3)
```



Looks much better now, but how do we know that we have an adequate value for K? Fortunately, `mgcv` provides function `k.check()` as an **informal way** to assess whether we should think about increasing K:

```r
k.check(Amodel)
             k'      edf   k-index p-value
s(timeStep)  3 2.923338 0.2588401       0
```

- The first column shows the number of coefficients $K-1$
- The second columns shows EDF, which in effect tells us “how many parameters out of `k'` did the model “use” after penalisation?” So despite the fact that our model has been penalised to not overfit the data, we still use nearly all of the 3 degrees of freedom.
  - This is the **main** indicator for increasing the rank. As a rule of thumb, if k' and the EDF of a function are less than 1 (i.e. less than one parameter between them) then we may need more degrees of freedom.
- The third and fourth columns are the k-index and a corresponding p-values. These are informal metrics whereby if k-index is less than 1 **and** the p-value is small then we may need to increase K.

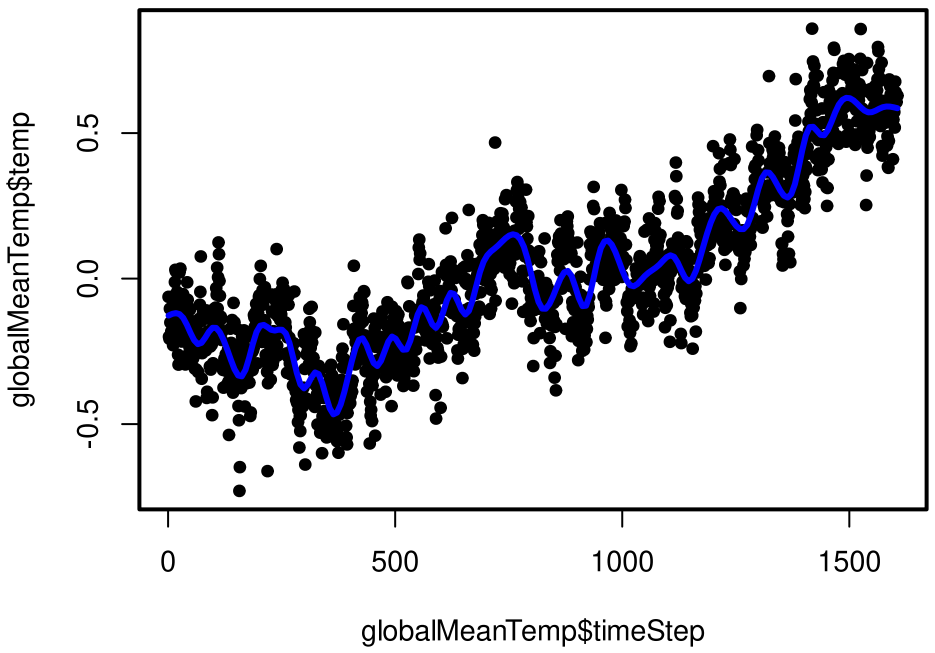
Here all evidence points to needing to increase K, including the visual check. Let’s look at the second model:

```r
k.check(Amodel2)
             k'      edf   k-index p-value
```

```
s(timeStep) 13 12.92242 0.3851522       0
```

This also indicates that $K = 14$ is too small. So let's increase even more:

```
# Fit the model
Amodel3 <- gam(temp ~ s(timeStep, k = 50, bs = "cs"),
               data = globalMeanTemp,
               family = gaussian(link = "identity"))
k.check(Amodel3)
            k'      edf   k-index p-value
s(timeStep) 49 46.7473 0.4753799       0
# Predicting model mean
yfitAM <- predict(Amodel3, newdata=data.frame(timeStep=xx))

# Plot
par(mar = c(4, 4, 1, 1),cex=1.2,lwd=2)
# Plot data
plot(globalMeanTemp$timeStep, globalMeanTemp$temp, pch = 20)
lines(xx, yfitAM, col = "blue",lwd=3)
```



This is now better (46.7 parameters out of 49) and although `k-index` and the p-value are small, we probably don't want a much "wigglier" line so best stop there (especially if model fits well).

Uncertainty

What about uncertainty of the estimated trend? Well, same game as with GLMs. We can construct a 95% confiodence interval of the linear predictor:

$$\hat{\eta}_i \pm 1.96 \times SE(\hat{\eta}_i)$$

and then transform at the scale of the mean μ_i

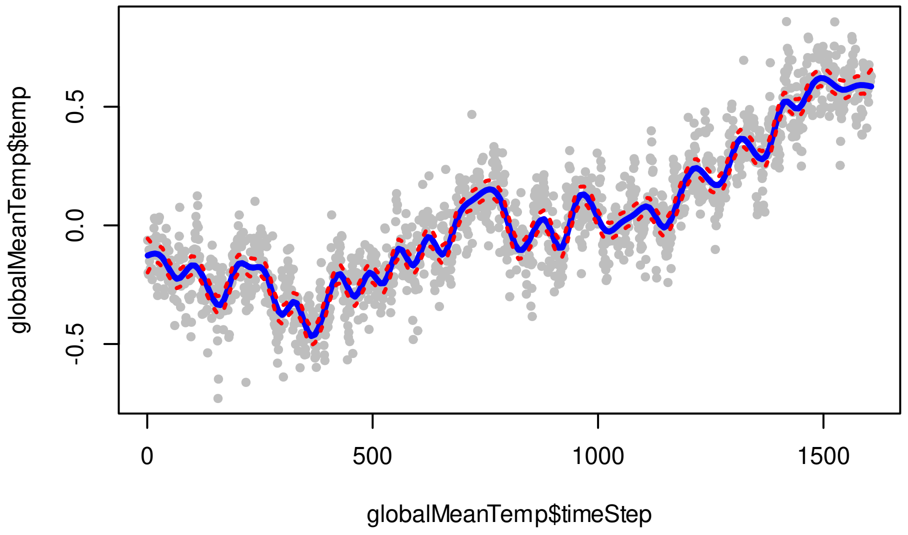
$$g^{-1}(\hat{\eta}_i \pm 1.96 \times \times SE(\hat{\eta}_i))$$

Here we are using the identity link so

$$\hat{\mu}_i \pm 1.96 \times SE(\hat{\mu}_i)$$

is obtained via:

```r
preds <- predict(Amodel3, newdata = data.frame(timeStep = xx),
                 se.fit = T)
# Plot data
plot(globalMeanTemp$timeStep, globalMeanTemp$temp, pch = 20,col="grey")
# Predictions and CIs
lines(xx,preds$fit,col="blue",lwd=3)
lines(xx,preds$fit+1.96*preds$se.fit,col="red",lwd=2,lty=3)
lines(xx,preds$fit-1.96*preds$se.fit,col="red",lwd=2,lty=3)
```

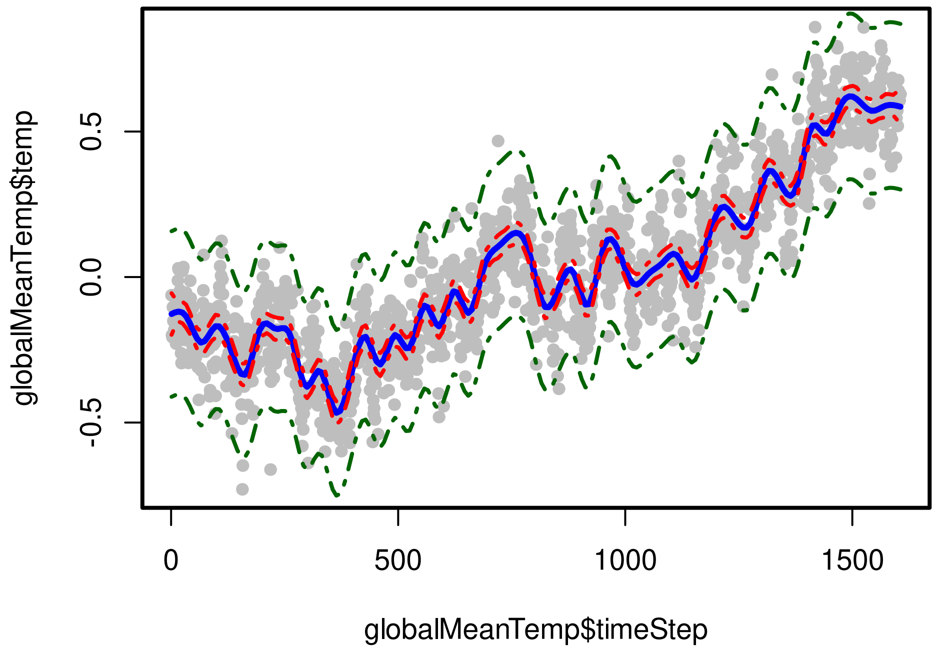


We can see that the CIs are pretty narrow for our model. This is because there is a lot of data and therefore certainty about the mean. Remember though, CIs tell us nothing about predicting the individual data y_i just the mean anomaly.

For predicting y_i we need prediction intervals (PIs). We have seen from Topics 1 and 2 that we can use the quantiles of the probability distribution of the response (Normal in this case) to compute PIs. To generate these for our model, we use `predict()` in conjunction with `qnorm()`.

```r
# Predictions and stand. erros for the mean
preds <- predict(Amodel3, newdata = data.frame(timeStep = xx), se.fit = T)

# Plot data
par(mar = c(4, 4, 1, 1),cex=1.2,lwd=2)
plot(globalMeanTemp$timeStep, globalMeanTemp$temp, pch = 20,col="grey")
```

```r
# Predictions and CIs
lines(xx,preds$fit,col="blue",lwd=3)
lines(xx,preds$fit+1.96*preds$se.fit,col="red",lwd=2,lty=4)
lines(xx,preds$fit-1.96*preds$se.fit,col="red",lwd=2,lty=4)
lines(xx, qnorm(0.025, mean = preds$fit, sd = sqrt(Amodel$sig2)),
      col="darkgreen",lwd=2,lty=4)
lines(xx, qnorm(0.975, mean = preds$fit, sd = sqrt(Amodel$sig2)),
      col="darkgreen",lwd=2,lty=4)
```



The PIs are much wider than CIs and capture most of the data. These tell us the range of monthly anomalies we would expect to see, and in this example we can pinpoint months with exceptionally high or low temperature anomalies.

Model checking

This is a Normal GAM with an estimated dispersion parameter so we don't need to check whether it fits with respect to the saturated model. But for models where the dispersion parameter is known (Binomial, Poisson, Exponential) we will need to do so. So let's see how we do this with GAMs.

Recall the test is (Slide 17 Topic 2 Notes):

$$H_0 : M \text{ is as good as } M_S \quad vs. \quad H_1 : M \text{ is NOT as good as } M_S$$

We saw that if H_0 is true then

$$\frac{D_M}{\phi} \sim_{approx} \chi^2_{n-p-1}$$

where D_M is the the deviance and D_M/ϕ is the scaled deviance (see Slide 18 Topic 2 Notes).

With penalised likelihood, the number of parameters is not $p + 1$ but EDF, so

$$\frac{D_M}{\phi} \sim_{approx} \chi^2_{n-EDF}$$

if the model fits. We can get $n - EDF$ via:

```
# The model degrees of freedom (analogous to n-p-1 in GLMs) is
Amodel3$df.residual
[1] 1558.253
```

Let's do the test:

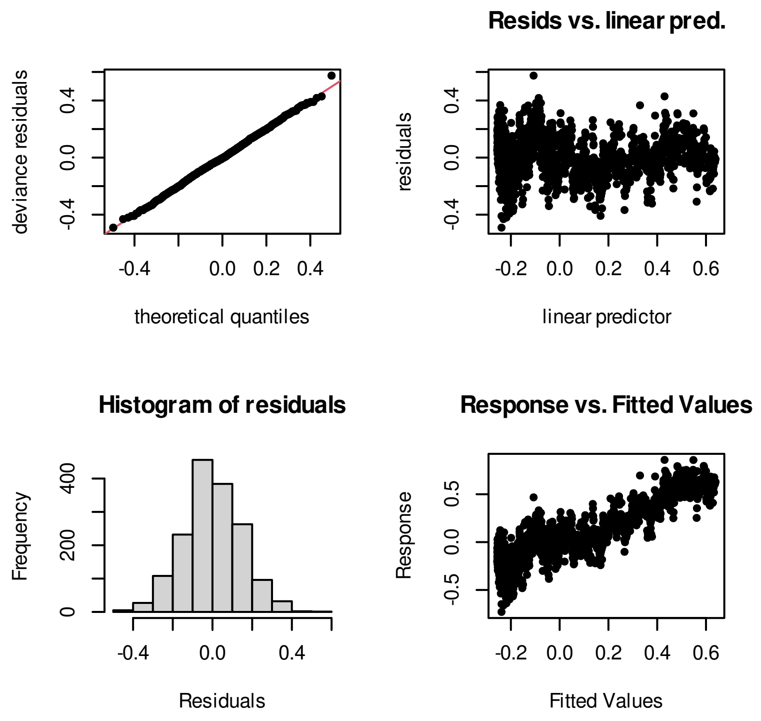
```
phi <- Amodel3$sig2
LRT <- Amodel3$deviance / phi
1 - pchisq(LRT, Amodel3$df.residual)
[1] 0.4952358
```

As expected, the p-value is much larger than 0.05 as this is a Normal GAM.

Residual checking

Residual plots are produced automatically via function `gam.check()`, which also prints out the `k.check()` table. Let's see do this for our original model with rank 4.

```
# 2x2 plot for the residuals
par(mfrow=c(2,2))

# Run gam.check on our original model
gam.check(Amodel,pch=20)
```



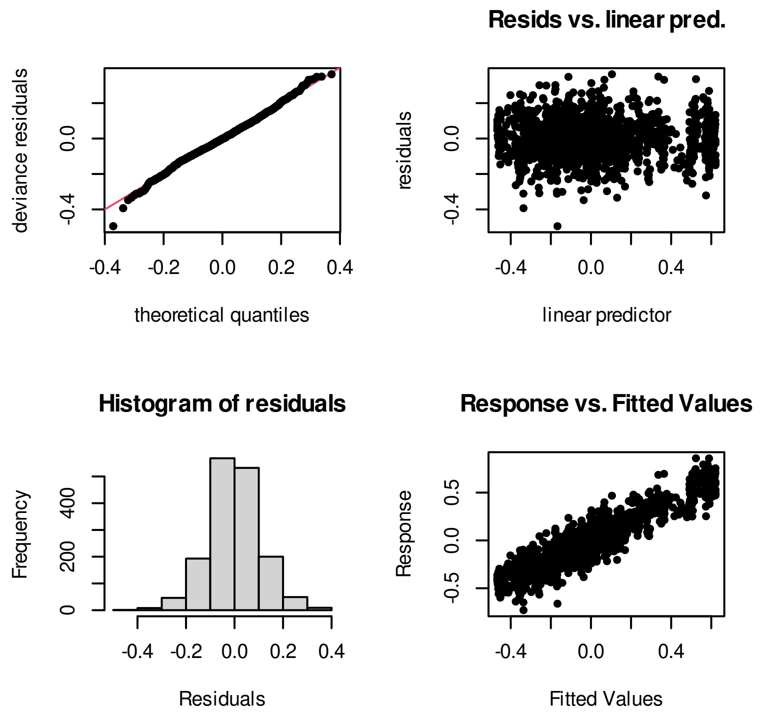
```
Method: GCV   Optimizer: magic
Smoothing parameter selection converged after 9 iterations.
The RMS GCV score gradient at convergence was 2.309279e-07 .
The Hessian was positive definite.
Model rank =  4 / 4

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(timeStep) 3.00 2.92    0.26  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The plots are:

- The first (top left) plot is the QQ plot (as we've seen before in GLMs) and if the model is a good fit, the residuals should lie on a diagonal line. In the case of our model, it seems to fit very well as nearly every point lies on the diagonal.
- The second (top right) plot is the residuals vs the linear predictor values. In this we should not see patterns about the $y = 0$ line. Here, we can see a definite pattern since we know that the function is not flexible enough.
- The third (bottom left) plot is a simply a histogram of the residuals, which we would use to see if the residuals follow a $N(0, 1)$ distribution. However, since we have the QQ plots, this is a bit useless.
- The fourth (bottom right) plot is the response data (y_i) versus the fitted/predicted values $(\hat{y}_i)$. If your model is a good fit then the points should lie or scatter evenly on a diagonal line. But again since we have the residuals vs the fitted values, this doesn't tell us much more.

Let's look at the residual plots from `Amodel3`:

```r
# 2x2 plot for the residuals
par(mfrow=c(2,2))
gam.check(Amodel3,pch=20)
```



```
Method: GCV   Optimizer: magic
Smoothing parameter selection converged after 4 iterations.
The RMS GCV score gradient at convergence was 6.675114e-07 .
The Hessian was positive definite.
Model rank =  50 / 50

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(timeStep) 49.0 46.7    0.48  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

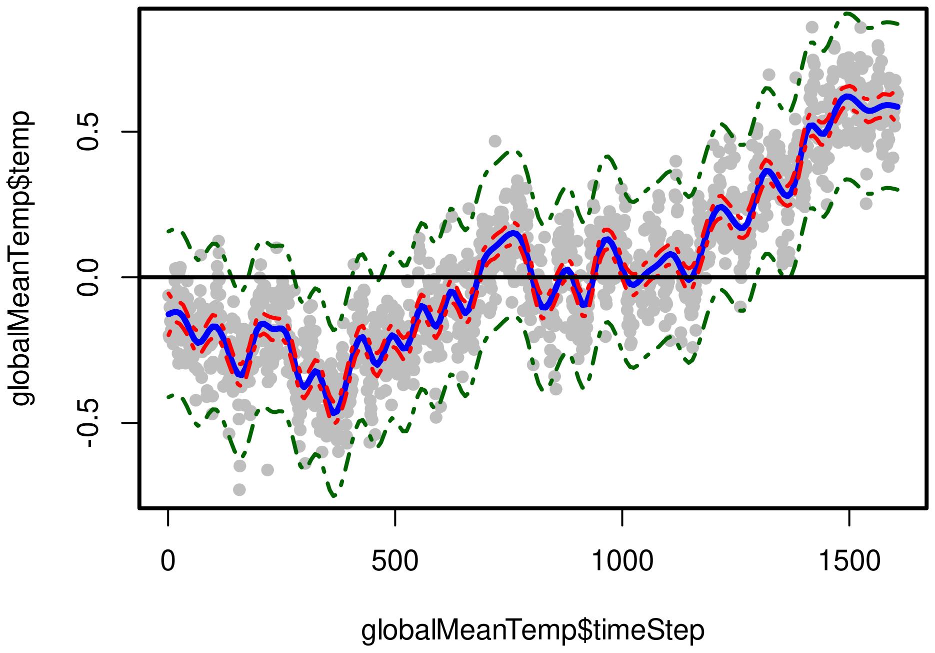
We can see that from the residuals vs the linear predictor values look much better with an even scatter about the $y = 0$ line and the patterns that we saw before have largely gone.

Interpretation

Let's look at the predicted trend again:

```
# Predictions and stand. erros for the mean
preds <- predict(Amodel3, newdata = data.frame(timeStep = xx), se.fit = T)

# Plot data
par(mar = c(4, 4, 1, 1),cex=1.2,lwd=2)
```

```r
plot(globalMeanTemp$timeStep, globalMeanTemp$temp, pch = 20,col="grey")

# Predictions and CIs
lines(xx,preds$fit,col="blue",lwd=3)
lines(xx,preds$fit+1.96*preds$se.fit,col="red",lwd=2,lty=4)
lines(xx,preds$fit-1.96*preds$se.fit,col="red",lwd=2,lty=4)
lines(xx, qnorm(0.025, mean = preds$fit, sd = sqrt(Amodel$sig2)),
      col="darkgreen",lwd=2,lty=4)
lines(xx, qnorm(0.975, mean = preds$fit, sd = sqrt(Amodel$sig2)),
      col="darkgreen",lwd=2,lty=4)
abline(h=0)
```



The model suggests that the global warming seems to flatten out, rather than increasing. This is what climate skeptics did with this data to show that global warming had stopped. This data only goes to 2013, but if you look up to 2020, we see that the trend flattens for a bit before increasing again, which is similar to the trend that we see from previous decades.

Model comparison

So, conditional on fixing the smoothing parameter λ is known, the inference is very similar to GLMs. The only think we haven't seen is model comparison.

Recall the exploratory plots at the beginning of the session, where we looked at a boxplot to see how the distribution of temperatures varies with month-of-the-year (MoY). The plot showed little evidence of a seasonal cycle, either in the mean or the variance of temperatures.

Nevertheless let's add a term in our GAM to model the seasonal cycle in order to:

- see how this can be done

- see how to do model selection.

Recall that t is the monthly time step and let $MoY(t)$ be the month-of-year for any t. So $MoY(1) = 1$ (January), $MoY(12) = 12$, $MoY(13) = 1$ and so on. Let's add a cyclic smooth function of $MoY(1)$:

$$\mu_t = \beta_0 + f_1(t) + f_2(MoY(t))$$

We construct both functions using cubic splines, except that for f_2 we must add an extra constraint so that the effect is cyclical (we don't want a discontinuity between December and Janunary).

Fortunately this is very easy to do in `mgcv` using `bs="cc"` (cyclic cubic spline). We continue with rank 50 for f_1 and let's choose rank 5 for our seasonal effect:

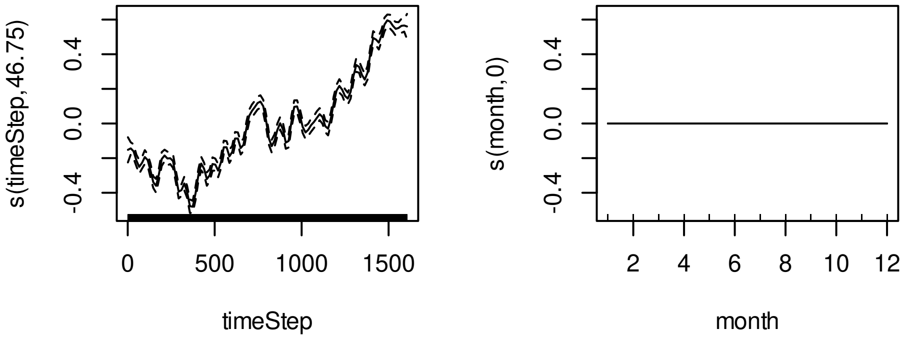
```
# Fit the model
Amodel4 <- gam(temp~s(timeStep,k=50,bs="cs") + s(month,bs="cc",k=5),
               data = globalMeanTemp,
               family = gaussian(link='identity'),
               knots=list(month=c(0,12)))
```

There are two smooth functions now and each has its own penalty (that was estimated):

```
Amodel4$sp
```

We can use the `plot()` function to visualise f_1 and f_2:

```
# Visualise the smooth functions at the linear predictor level using plot():
par(mfrow = c(1,2))
plot(Amodel4)
```



Function f_1 is basicallt the same as before, while the seasonal cycle function f_2 is not really doing anything (as we would expect from the exploratory plots).

In GLMs, we would have conducted a LRT to compare `Amodel3` with `Amodel4` to assess whether we need f_2 or not. With GAMs however, this is no longer the case, as the models are no longer nested because of the penalisation. The f_1 from `Amodel3` is not the same as the `f_1` from `Amodel4` so the former is not a special case of the former:

```
Amodel3$coefficients[1:6]
  (Intercept) s(timeStep).1 s(timeStep).2 s(timeStep).3 s(timeStep).4
   0.02510205   -0.13782745   -0.23098010   -0.17219229   -0.25608750
s(timeStep).5
```

```
  -0.33647665
Amodel4$coefficients[1:6]
  (Intercept) s(timeStep).1 s(timeStep).2 s(timeStep).3 s(timeStep).4
   0.02510205   -0.13782477   -0.23098253   -0.17219000   -0.25608621
s(timeStep).5
  -0.33647989
```

Fortunately, the `summary()` returns an appropriate F-test however to that effect:

```
# Summarise the model
summary(Amodel4)

Family: gaussian
Link function: identity

Formula:
temp ~ s(timeStep, k = 50, bs = "cs") + s(month, bs = "cc", k = 5)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.025102   0.002709   9.266   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                  edf Ref.df     F p-value
s(timeStep) 4.675e+01     49 218.1  <2e-16 ***
s(month)    3.429e-07      3   0.0   0.598
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.869   Deviance explained = 87.3%
GCV = 0.012149  Scale est. = 0.011787  n = 1606
```

and as expected, the p-value for `s(month)` is bigger than 0.05, which means we can get rid of this term!

And of course for multiple models we can simply compare the AIC. In Topic 2, we saw that the AIC is twice the negative log-likelihood plus the number of parameters (see Slide 21 Topic 2)

$$\text{AIC} = 2(-\ell(\boldsymbol{\theta}, \phi; \boldsymbol{y}) + p)$$

but again as we are using a penalised likelihood we replace the number of parameters p with the effective degrees of freedom

$$\text{AIC} = 2(-\ell(\boldsymbol{\theta}, \phi; \boldsymbol{y}) + EDF)$$

The model with the lower AIC is better. We can extract the AIC's using the `AIC` function:

```r
# Extract the model AIC
AIC(Amodel3)
[1] -2525.148
AIC(Amodel4)
[1] -2525.149

# Doing it by hand
as.numeric(2*(-logLik(Amodel3) + sum(Amodel3$edf)))
[1] -2527.148
as.numeric(2*(-logLik(Amodel4) + sum(Amodel4$edf)))
[1] -2527.149
```

The AICs are near identical, and combining it with the above we can conclude that the models are basically the same so there is no point adding a seasonal component to our model.