```r
data <- read.table("../../frequency.dat")
names(data) <- c("year", "freq")
plot(freq ~ year, data = data, ylab = "Freqency [MHz]",
    ylim = c(-1000, 6000))
```

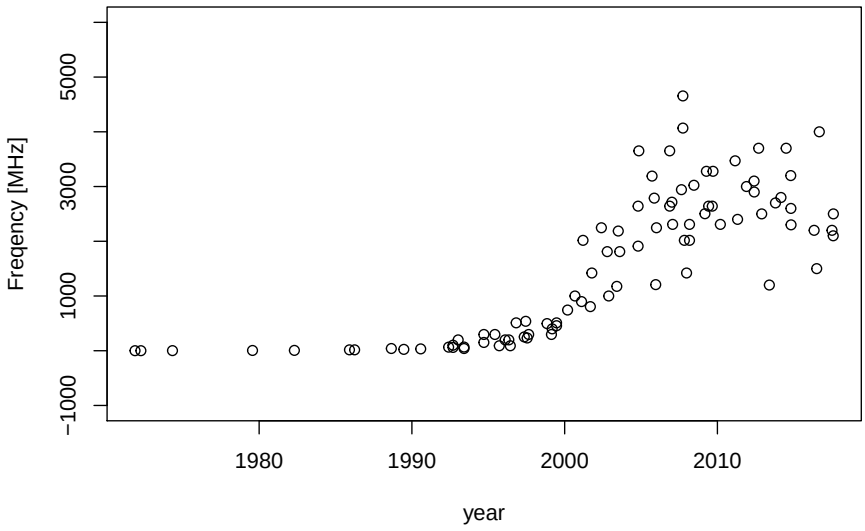

Figure 1: Microprocessor frequencies between 1971 and 2018.

1 Introduction

In this project the frequency dataset is analysed. The frequency dataset consists of 90 observations of microprocessor frequencies between 1971 and 2018 (Figure 1).

A first inspection of the data shows relatively low frequencies in the beginning with a sudden increase arround the year 1995. The fastest increase we see arround the year 2002, afterwards the frequency starts leveling off. This pattern is characteristic for logistic functions, which might be useful to model the data.

The variability increases with the level of frequency (heteroscedasticity) and all frequencies are naturally positive. Therefore, a log transformation might be useful to stabilize the variance.

The log transformed frequencies have a relatively stable variance over all years (Figure 2). There is a clear increase in log frequency until the year 2005. Afterwards the log frequency seems to be relatively stable. A piecewise linear

```r
plot(log(freq) ~ year, data = data, ylab = "log(Freqency)")
```

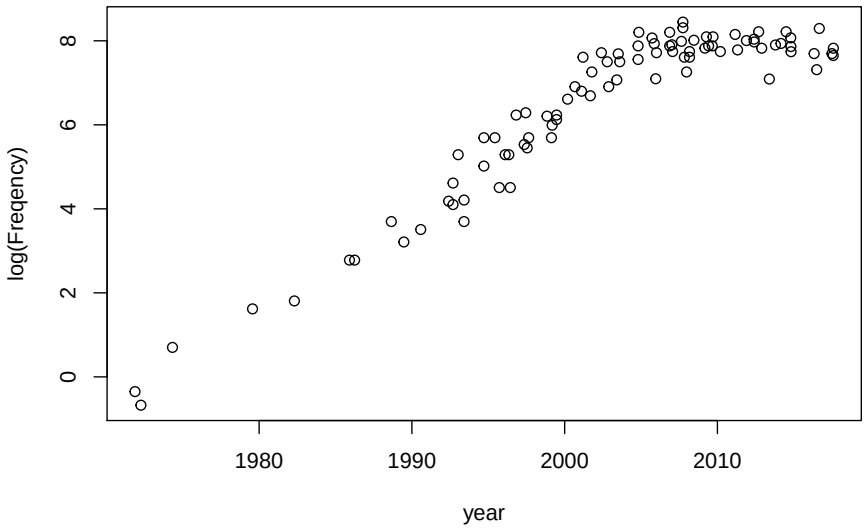

Figure 2: Microprocessor frequencies between 1971 and 2018 on log scale.

function might be a good choice to model these data.

2 Model fitting and prediction

2.1 Model

Closesly related to piecewise linear functions are regression splines. Splines are picewise polynomials which are continuously differentiable up to a certain degree at the nodes. Splines can easliy be fitted with a generalized additive model (GAM). The model can be written as

$$g(E(Y)) = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_m(x_m) \tag{1}$$

where $g(\cdot)$ is a link function, linking the expected value of Y to the linear predictor and $f_i(\cdot)$ are smooth functions of the predictor x_i. The predictor is a linear combination of smooth functions $f_i(\cdot)$. The smooth function makes GAM more general than generalized linear models but they do not allow automatically for interactions (no curse of dimensionality). In our case we use splines as smooth functions.

The degrees of freedom of the spline have to be limited to avoid overfitting. This is done by penalizing the wiggliness of the spline in the loss function

$$\hat{f} = \underset{f \in F}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_a^b g''(x)^2 dx \tag{2}$$

where F is the class of possible spline functions, λ is the penalty parameter, a and b are the lower and upper boundary of the range of x, respectively. The implementation of GAM in R with the function 'gam()', automatically chooses the penalty parameter with cross validation.

Applying the model to the frequency data leads to

$$E(\log(Frequency)) = \beta_0 + f_t(t) \tag{3}$$

which means our $g(\cdot)$ is the identity function and we have only one smooth spline function for our predictor time t.

2.2 Fitting

We fit a GAM to the data using the function `gam()` of the package `mgcv` (Wood, 2019).

```
# packages
# ------------------------------------------------------------------
library(mgcv)
```

GAM is able to follow the trend without overfitting to the single observations (Figure 3).

```r
fit <- gam(log(freq) ~ s(year), data = data)
plot(log(freq) ~ year, data = data, ylab = "log(freqency)")
lines(data$year, predict(fit), col = "red", lwd = 2)
```

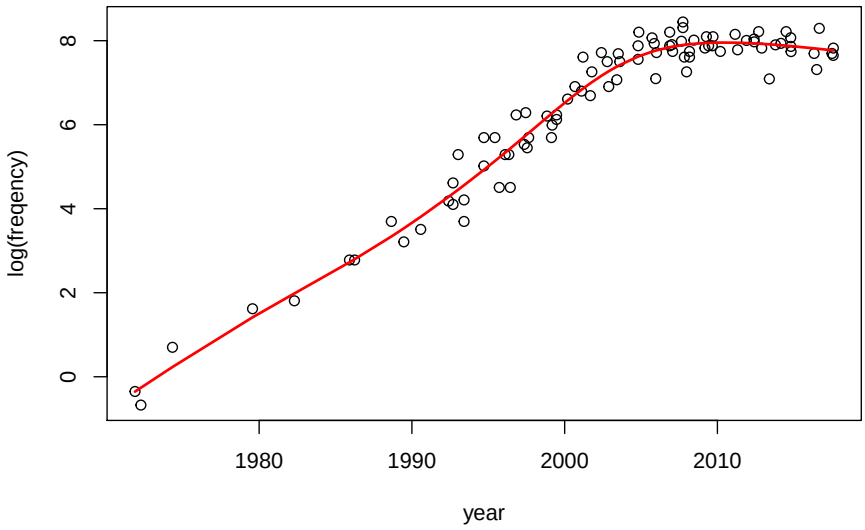

Figure 3: Fitted GAM model.

```
summary(fit)


Family: gaussian
Link function: identity

Formula:
log(freq) ~ s(year)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.45383    0.03913     165   <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(year) 5.266  6.378 413.3  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.967   Deviance explained = 96.9%
GCV = 0.14808  Scale est. = 0.13777   n = 90
```

The predictor year is highly significant with 5.266 estimated degrees of freedom.

2.3 Validation

The Tukey-Anscombe plot and the QQ plot of the model fit shows that the model assumptions are largely fulfilled (Figure 4). The variance stays constant over the entire range of fitted value and the residuals follow a normal distribution.

2.4 Extrapolation

GAM can also be used to predict future frequencies (Figure 5).

3 Simulation study

First, I created a true model which can lead to similar data as we observed. I used a logistic model with normally distributed heteroscadistic errors.

Next we fit two models to the new data. One model is a GAM which I already used before, the other is a logistic model fitted with nonlinear least squares.

To compare how well the two models fit at year 2008 and 2017, we repeat the simulation 1000 times and plot the density of the difference to the true model.

```r
par(mfrow = c(1, 2))
with(fit, plot(fitted.values, residuals))
qq.gam(fit)
```

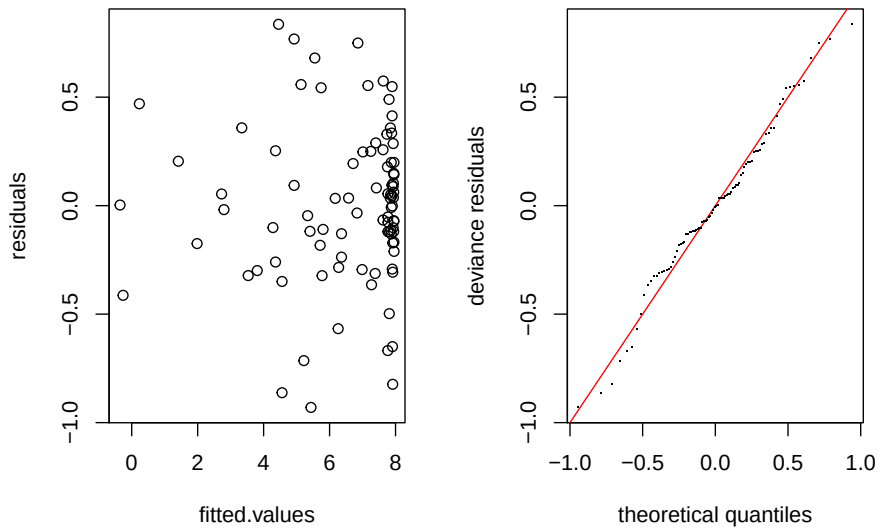

Figure 4: Residual analysis of the fitted GAM model with Tukey-anscombe plot (left) and QQ plot (right).

```r
plot(log(freq) ~ year, data = data, ylab = "Freqency [MHz]",
    xlim = c(1970, 2050), ylim = c(0, 10))
lines(data$year, predict(fit), col = "red", lwd = 2)
pred_period <- 2018:2050
mypred <- predict.gam(fit, newdata = list(year = pred_period),
    se.fit = T)
polygon(x = c(pred_period, rev(pred_period)),
    y = c(mypred$fit - 1.96 * mypred$se.fit, rev(mypred$fit +
        1.96 * mypred$se.fit)), border = NA, col = "lightgray")
lines(pred_period, mypred$fit, col = "gray", lwd = 2)
```

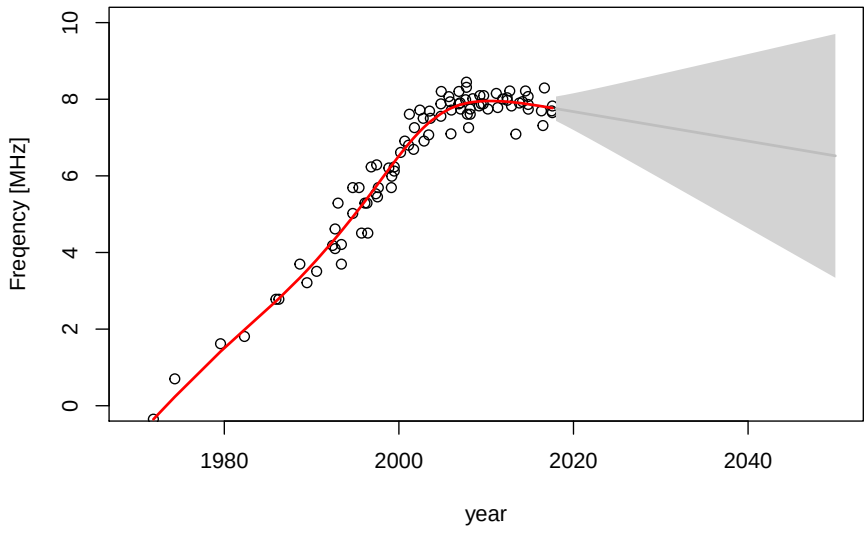

Figure 5: Linear extrapolation of the trend fitted by the GAM model. The gray area indicate the 95% CI for the trend prediction.

```r
# Single example
# ------------------------------------------------------------
plot(freq ~ year, data = data, ylab = "Freqency [MHz]",
    ylim = c(-1000, 6000))
# Assume a true function
fpl <- function(x, A = 0, B = 3000, xmid = 2003,
    scal = 3) {
    A + (B - A)/(1 + exp((xmid - x)/scal))
}
curve(fpl, from = 1970, to = 2020, add = T)
# sample new points
set.seed(1)
true_value <- fpl(data$year)
value <- abs(true_value + rnorm(length(true_value),
    mean = 0, sd = true_value/3))
sim_data <- data.frame(year = data$year, freq = value)
points(sim_data$year, sim_data$freq, col = "green")
legend("topleft", legend = c("Original observations",
    "Observations with new model"), pch = 1, col = c("black",
    "green"))
```

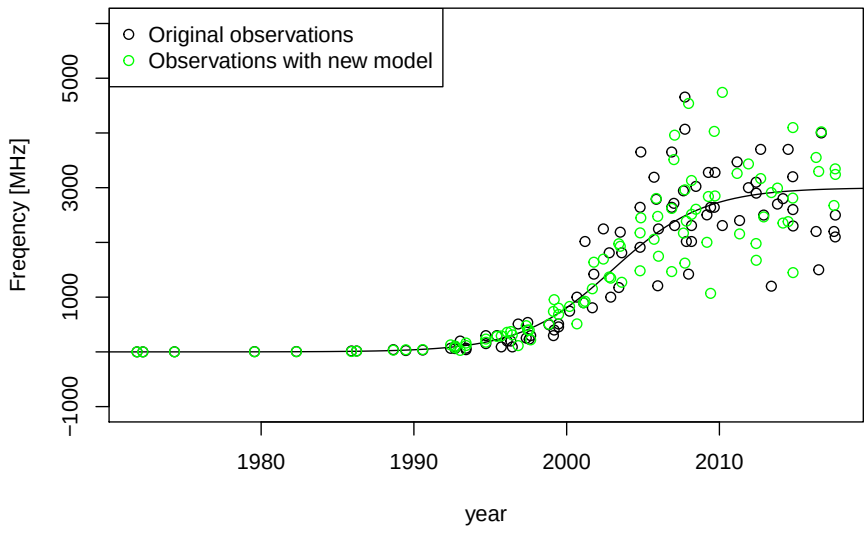

Figure 6: Random observations using the new model. The black line shows the true mean of the new model.

```r
# fit with GAM
fit <- gam(log(freq) ~ s(year), data = sim_data)
# fit with SSfpl()
fit_ssfpl <- nls(freq ~ SSfpl(year, A, B, xmid,
    scal), data = sim_data)
# plot all
plot(freq ~ year, data = sim_data, ylab = "Freqency [MHz]",
    ylim = c(-1000, 6000))
lines(sim_data$year, exp(predict(fit)), col = "red",
    lwd = 2)
lines(sim_data$year, predict(fit_ssfpl), col = "green",
    lwd = 2)
curve(fpl, lwd = 2, add = T)
legend("topleft", legend = c("True model", "GAM fit",
    "FPL fit"), lwd = 2, col = c("black", "red",
    "green"))
abline(v = 2008, lty = 2)
abline(v = 2017, lty = 2)
```

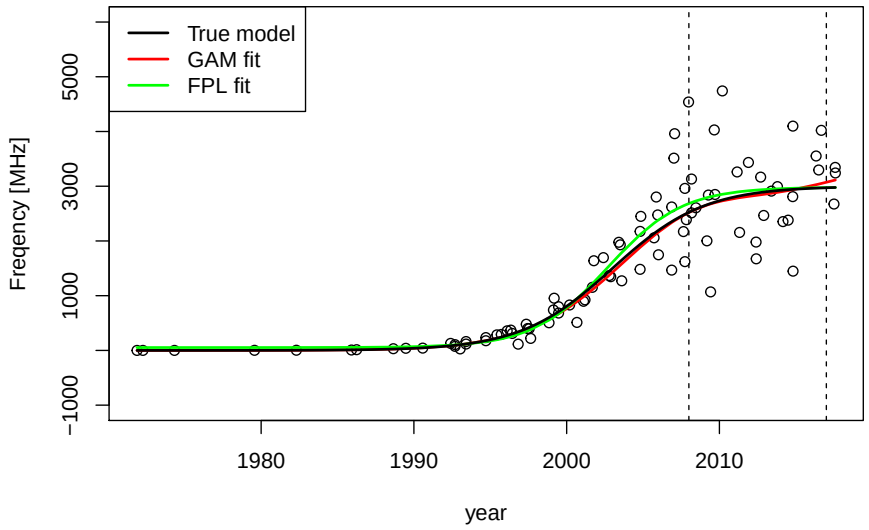

Figure 7: GAM and four paramteric logistic model (FPL model) fitted to a random sample of the new model.

```r
# simulation
# ------------------------------------------------------------
sim_diff <- function() {
    true_value <- fpl(data$year)
    value <- abs(true_value + rnorm(length(true_value),
        mean = 0, sd = true_value/3))
    sim_data <- data.frame(year = data$year, freq = value)
    fit <- gam(log(freq) ~ s(year), data = sim_data)
    fit_ssfpl <- nls(freq ~ SSfpl(year, A, B,
        xmid, scal), data = sim_data)
    gam_pred <- unname(exp(predict(fit, newdata = list(year = c(2008,
        2017)))))
    ssfpl_pred <- predict(fit_ssfpl, newdata = list(year = c(2008,
        2017)))
    true_values <- fpl(c(2008, 2017))
    c(gam_pred - true_values, ssfpl_pred - true_values)
}
out <- replicate(1000, sim_diff())
out <- t(out)
# plot simulation results
# ------------------------------------------------
plot(density(out[, 1]), main = "Difference to true function at 2008",
    xlim = c(-1000, 1000), ylim = c(0, 0.005),
    col = "red", lwd = 2)
lines(density(out[, 3]), col = "green", lwd = 2)
abline(v = 0, lty = 2)
legend("topleft", legend = c("True model", "GAM fit",
    "FPL fit"), lwd = 2, col = c("black", "red",
    "green"))
```

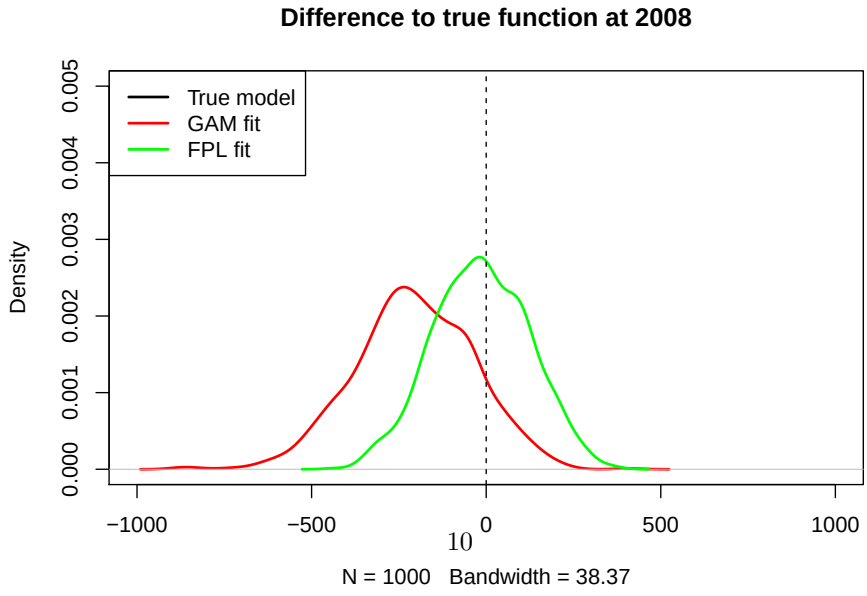

Figure 8: Distribution of the difference between the true model and the estimates of the GAM and FPL model at 2008.

```r
plot(density(out[, 2]), main = "Difference to true function at 2017",
    xlim = c(-1000, 1000), ylim = c(0, 0.005),
    col = "red", lwd = 2)
lines(density(out[, 4]), col = "green", lwd = 2)
abline(v = 0, lty = 2)
legend("topleft", legend = c("True model", "GAM fit",
    "FPL fit"), lwd = 2, col = c("black", "red",
    "green"))
```

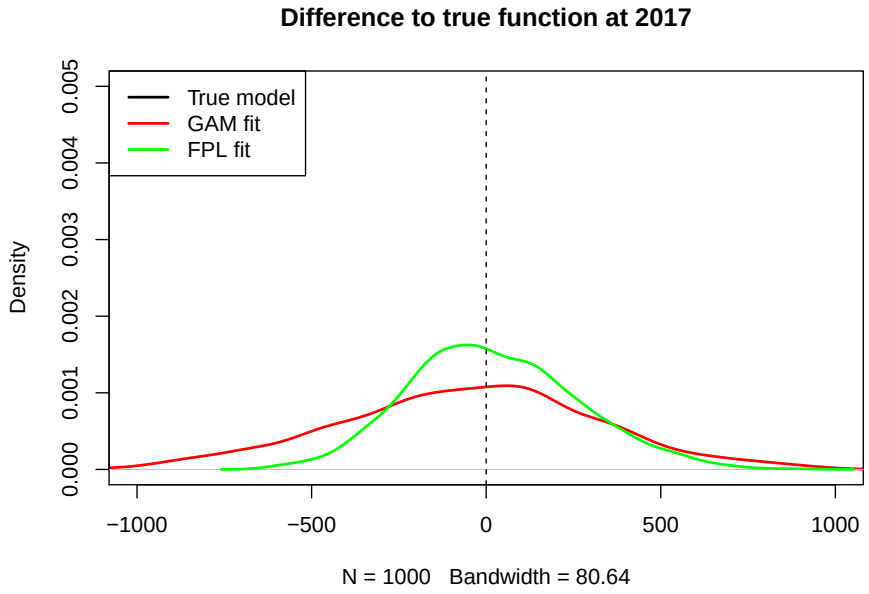

Figure 9: Distribution of the difference between the true model and the estimates of the GAM and FPL model at 2017.

We see that GAM is negatively biased at 2008. The logistic model has no bias and lower variance at both time points. This is not very surprising because the true underlying model was also a logistic model.

4 Conclusion

We can conclude that GAM is a powerful tool to fit a curve to data which have some piecewise functional form. In our simulation, the logistic model was slightly better in fitting the true curve. However, the logistic model is much less flexible compared to GAM. Therefore, we recomend GAM as fast and easy to use model which can fit a wide variety of functions.

References

Simon Wood. *mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation*, 2019. URL `https://CRAN.R-project.org/package=mgcv`. R package version 1.8-29.