Generalized Additive Model

Daniel Shang

```
# Load the packages needed
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.0.3
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(mgcv)
```

```
## Warning: package 'mgcv' was built under R version 4.0.3

## Loading required package: nlme

## Warning: package 'nlme' was built under R version 4.0.3

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-33. For overview type 'help("mgcv-package")'.
```

```
library(ggplot2)
```

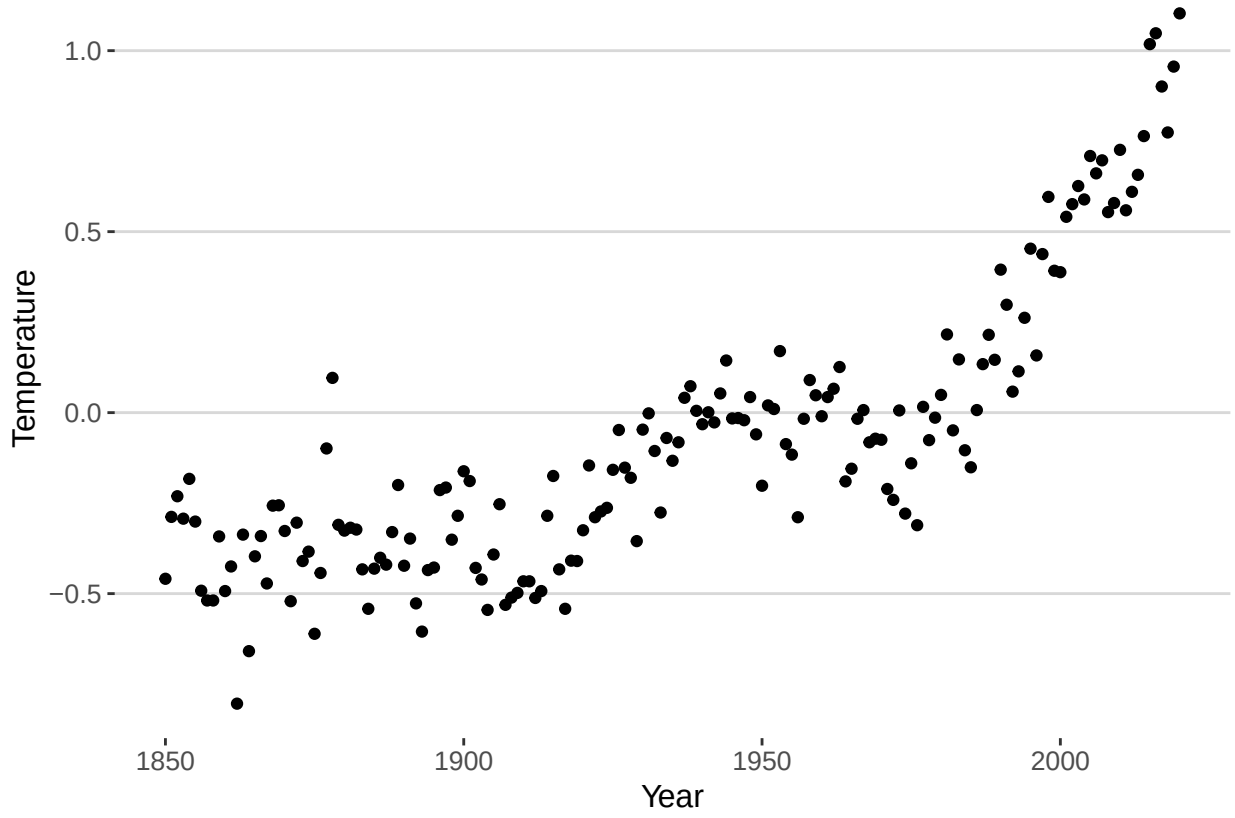
```
## Warning: package 'ggplot2' was built under R version 4.0.3
```

```
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 4.0.3
```

```
# The data used in this illustration is from a URL defined below. The data is about
## temperature measured each year
url = 'https://bit.ly/hadcrutv4'
data = read_delim(url, delim = ' ', col_types = 'nnnnnnnnnnnn', col_names = FALSE) %>%
  select(num_range('X', c(1, 5))) %>% setNames(nm = c('Year', 'Temperature'))
```

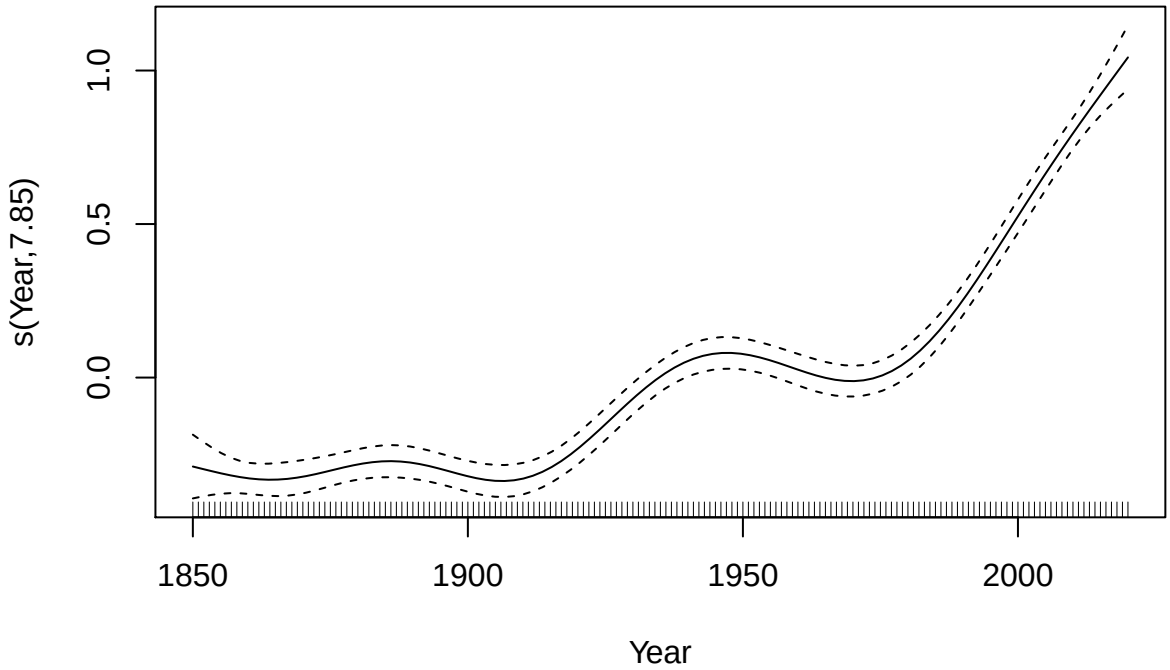
```
# Plot the data to get a basic understanding. As the plot shows, the relationship
## between year and temperature is not linear. Thus, we cannot use a linear model.
## Generally, polynomial functions can fit the nonlinear pattern. Therefore, we use
## Generalized Additive Model (GAM) here. The logic behind GAM is that the model
## takes several pieces of smaller functions together and smooth them to form a
## polynomial function, which fit the model.
ggplot(data, aes(x = Year, y = Temperature)) +
  geom_point() +
  theme_hc()
```




```
# Fit a GAM model and summarize the model.
gam_1 = gam(Temperature ~ s(Year), data = data, method = 'REML')
summary(gam_1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Temperature ~ s(Year)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.07590    0.00974  -7.792 7.45e-13 ***
## ---
## 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) 7.851  8.658 159.5  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.89   Deviance explained = 89.5%
## -REML = -90.886  Scale est. = 0.016221  n = 171
```

```
# Make a simple plot of the model to check its pattern
plot(gam_1)
```



```r
# To see how well the model fits the data, we make prediction using the current
## model and current data. We also plot a confidence interval with three standard
## error away as its upper bound and lower bound. We also plot the original data.
## As the plot shows, the model fit the original data well.
new_year = as_tibble(with(data, data.frame(Year = seq(min(Year), max(Year), length = nrow(data)))))
pred_1 = as_tibble(data.frame(predict(gam_1, newdata = new_year, se.fit = TRUE,
                                      unconditional = TRUE)))
pred_1 = cbind(new_year, pred_1) %>%
    mutate(upr = fit + 3 * se.fit, lwr = fit - 3 * se.fit)

ggplot(data, aes(x = Year, y = Temperature)) +
  geom_point() +
  geom_ribbon(data = pred_1, mapping = aes(ymin = lwr, ymax = upr, x = Year),
              alpha = 0.4, inherit.aes = FALSE, fill = "yellow3") +
  geom_line(data = pred_1, mapping = aes(y = fit, x = Year), inherit.aes = FALSE,
            size = 1, colour = "blue3") +
  labs(x = 'Year', y = 'Temperature', title = 'GAM Model Fit') +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_hc()
```