Question 1

(a)

Firstly, I implemented polynomial regression with order 3. But when we look at the r-square value, it only explains the 7% of the variance, and there are not many significant variables as well, that's not a good fit.

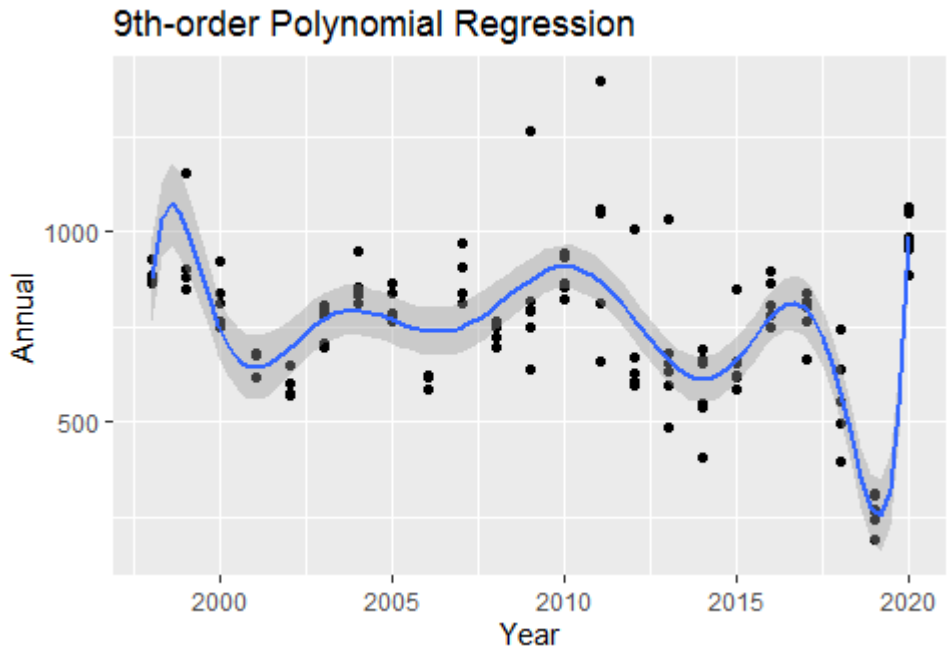
```
> model=lm(Annual~poly(Year,3),data=df)
> summary(model)

Call:
lm(formula = Annual ~ poly(Year, 3), data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-464.26 -122.03   -7.85   89.98  641.17

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    752.52      17.84  42.172  < 2e-16 ***
poly(Year, 3)1 -577.88     193.01  -2.994  0.00338 **
poly(Year, 3)2 -123.16     193.01  -0.638  0.52470
poly(Year, 3)3  -64.25     193.01  -0.333  0.73985
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 193 on 113 degrees of freedom
Multiple R-squared:  0.07742,   Adjusted R-squared:  0.05292
F-statistic: 3.161 on 3 and 113 DF,  p-value: 0.02742
```

Let's increase the order of polynomial so we can find a better-fitted model. I choose the 9th order polynomial to fit the model which gives us almost 60% explanation about the variance and I have much more significant variables.

```
Call:
lm(formula = Annual ~ poly(Year, 9), data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-237.34  -76.41   -4.46   53.00  524.66

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    752.52      11.57  65.031  < 2e-16 ***
poly(Year, 9)1 -577.88     125.17  -4.617 1.09e-05 ***
poly(Year, 9)2 -123.16     125.17  -0.984   0.3274
poly(Year, 9)3  -64.25     125.17  -0.513   0.6088
poly(Year, 9)4  737.49     125.17   5.892 4.48e-08 ***
poly(Year, 9)5  324.45     125.17   2.592   0.0109 *
poly(Year, 9)6  254.10     125.17   2.030   0.0448 *
poly(Year, 9)7  680.73     125.17   5.439 3.42e-07 ***
poly(Year, 9)8  626.86     125.17   5.008 2.18e-06 ***
poly(Year, 9)9  981.45     125.17   7.841 3.55e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 125.2 on 107 degrees of freedom
Multiple R-squared:  0.6326,    Adjusted R-squared:  0.6017
F-statistic: 20.47 on 9 and 107 DF,  p-value: < 2.2e-16
```

I got four strongly significant variables.

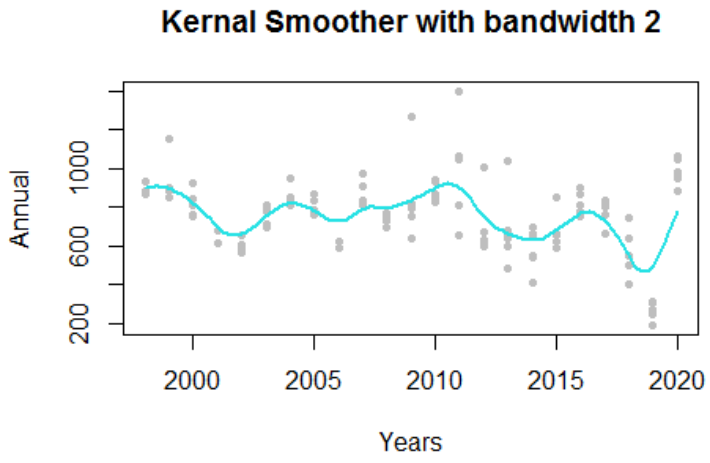
For Visualization purposes, I am using the grammar of graphics (ggplot2) library to visualize the fit of the model on the data with a 95% confidence interval.

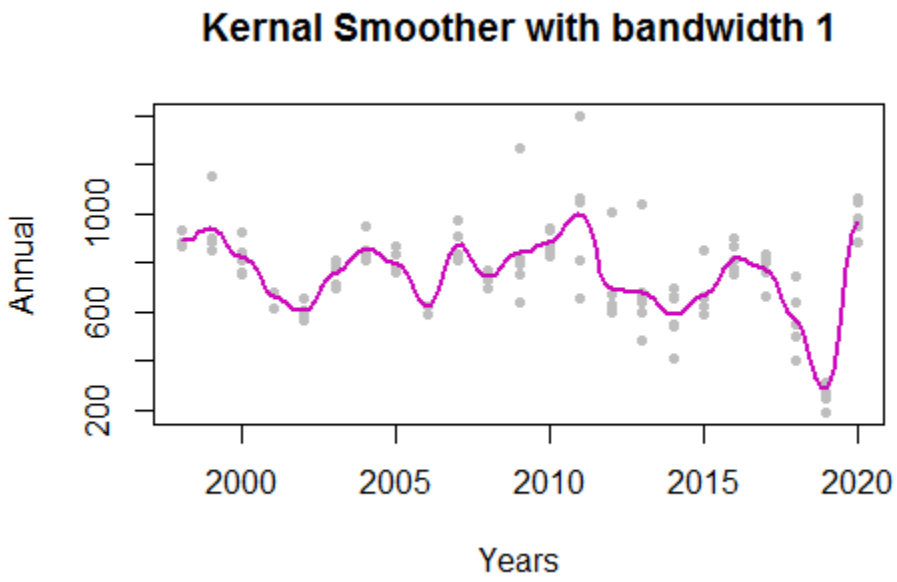


(b)

Kernal smoother:

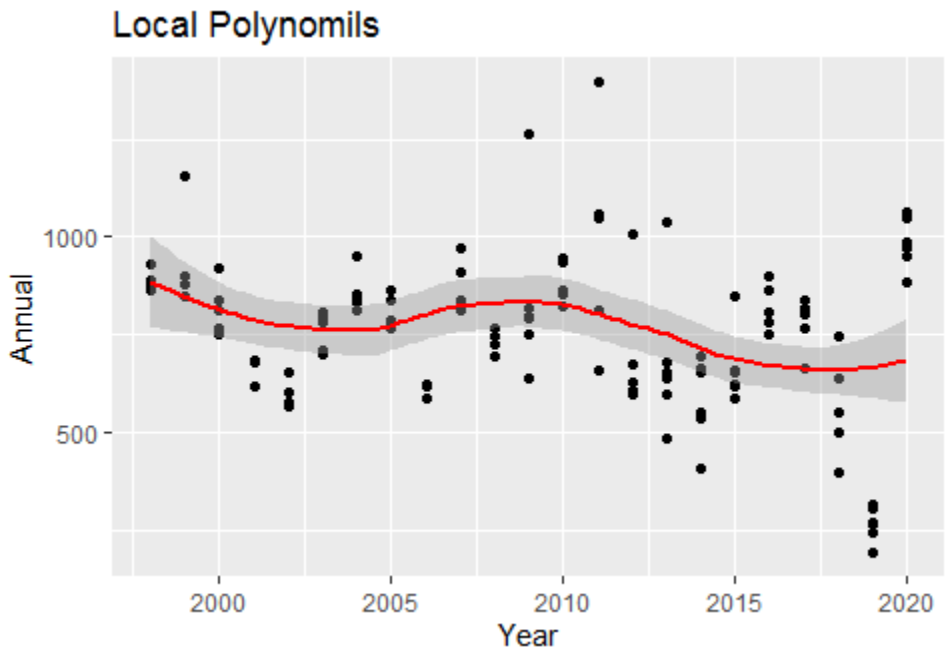
The kernels are scaled so that their quartiles (viewed as probability densities) are at +/- 0.25*bandwidth. Where the bandwidth is specified manually. Lesser the bandwidth, the better fit we will get. I analyzed fitting with bandwidth 2 and bandwidth 1. The results are below.





Local polynomials

Local polynomials fitting each subset of data are usually of the first or second degree, i.e., either locally linear (in the straight line sense) or locally quadratic. LOESS becomes a weighted moving average when a zero-degree polynomial is used. In theory, higher-degree polynomials would work, but they would produce models that were not in the spirit of LOESS. LOESS is built on the principles that a low-order polynomial may accurately estimate any function in a narrow region and that simple models can be easily fitted to data. High-degree polynomials are numerically unstable and tend to overfit the data in each subset, making accurate computations challenging.



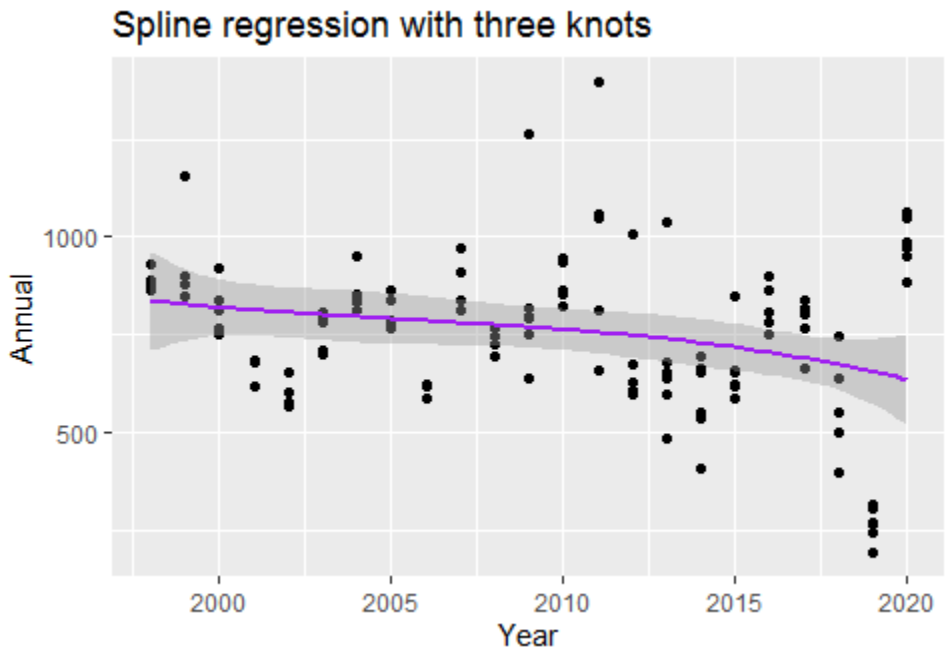
Splines

Polynomial regression only captures a definite amount of curvature in a nonlinear association. Another approach to modeling nonlinear relationships is to use splines.

Splines provide a way to smoothly interpolate between fixed points, called knots. Polynomial regression is computed between knots. In other words, splines are series of polynomial segments strung together, joining at knots.

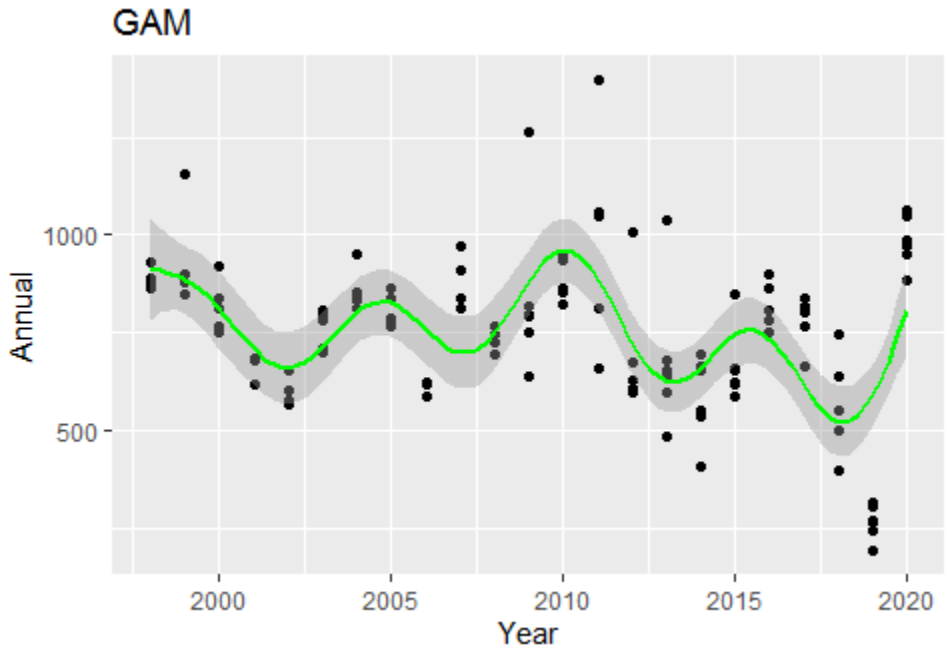
The R package “splines” includes the function ‘bs’ for creating a b-spline term in a regression model.

I need to specify two parameters: the degree of the polynomial and the location of the knots.



Generalized additive model

Once you have detected a non-linear relationship in your data, the polynomial terms may not be flexible enough to capture the relationship, and spline terms require specifying the knots. Generalized additive models, or GAM, are a technique to automatically fit a spline regression without putting the knots manually.

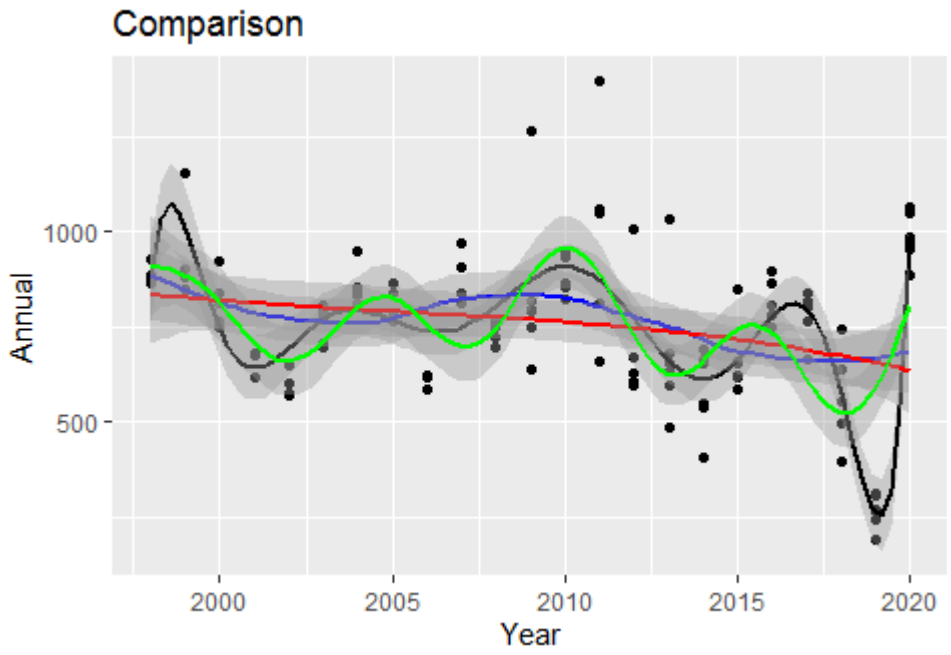


(c)

Let's plot all the curves on the same graph to compare the results from all methods.

Color codes:

Black	9th order polynomial regression
Green	GAM
Blue	Local polynomial regression
Red	Spline with three knots



As we can see, *all the models clearly show the decrease in annual rainfall over time.*

9th order polynomial is a best-fitted model from all of them where the spline with three knots is the most poorly fitted mode. But there is a possibility that the best-fitted model does not predict well on the unseen data, because it might be the case of overfitting. Where GAM is the 2nd model with a good fit. Spline with three knots barely explains any variance in data.

Appendix

Code

```
#Loading required libraries

library(tidyverse)

library(ggplot2)
library(splines)


#Importing our dataset

df <- read.delim("ArmidaleRainfall.txt")


# Part(a)        Polynomial regression
# Modeling


model=lm(Annual~poly(Year,3),data=df)
summary(model)

model=lm(Annual~poly(Year,9),data=df)
summary(model)

#Visualizing Polynomial Regression

ggplot(data=df, aes(Year,Annual)) +
   geom_point() +
   geom_smooth(method="lm", formula=y~poly(x,9))+ labs(title = "9th-order Pol
ynomial Regression")

#Part b

# Kernal smoother
plot(df$Year,df$Annual,pch=20,col='grey', xlab="Years",ylab = "Annual",main =
"Kernal Smoother with bandwidth 1")+lines(ksmooth(df$Year,df$Annual, "normal"
, bandwidth = 1), col = 7862,lwd = 2)



plot(df$Year,df$Annual,pch=20,col='grey', xlab="Years",ylab = "Annual",main =
"Kernal Smoother with bandwidth 2")+lines(ksmooth(df$Year,df$Annual, "normal"
, bandwidth = 2), col = 765,lwd = 2)

# Splines

# Build the model
```

```r
knots <- quantile(df$Year, p = c(0.25,0.5,0.75))
model <- lm (Annual ~ bs(Year, knots = knots), data = df)
summary(model)

# Visualization

ggplot(df, aes(Year, Annual) ) +
    geom_point() +
    stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 3),col = "purp
le")+labs(title = "Spline regression with three knots")

# Local Polynomial

local=loess(Annual ~ Year, df)
summary(local)

plot(y=df$Annual, x=df$Year, type="p", main="Loess Smoothing and Prediction",
xlab="Date", ylab="Unemployment (Median)")+lines(df$Annual, df$Year)

ggplot(data=df, aes(Year,Annual)) +
    geom_point() +
    geom_smooth(method="loess", formula= y ~ x ,col = "red")+labs(title = "Lo
cal Polynomils")

# GAM
ggplot(df, aes(x = Year, y = Annual)) + geom_point()+ stat_smooth(method = "g
am",
              formula = y ~s(x), size = 1, se = T, colour = "green")+labs(tit
le = "GAM")

# Comparison on a same plot


m <- ggplot(df, aes(x = Year, y = Annual)) + geom_point()
print(m)

m + stat_smooth(method = "lm", formula = y~poly(x,9), size = 1, se = T,
                colour = "black") + stat_smooth(method = "loess", formula = y
~ x,
                size = 1, se = T, colour = "blue") + stat_smooth(method = "lm
",
                formula =  y ~ splines::bs(x, df = 3), size = 1, se = T, colo
ur = "red") + stat_smooth(method = "gam",
                formula = y ~s(x), size = 1, se = T, colour = "green") +labs(
title = "Comparison")
```