Project III-Stat Method II

Craig Fick

4/24/2021

Required Packages:

```
#library(maps)
#library(rspatial)
#library(fields)
#library(lmtest)
#library(FitAR)
#library(forecast)
#library(fUnitRoots)
```

First Problem of Assignment:

Problem 2

In this problem we would like to analyze an ERA-Interim annual average temperature time series at a spatial location of your choice. For example you can choose the spatial location closest to where you live now. Below you can find a script to get the spatial location closest to New York City:

Load the Provided Data

```
load("NA_MaxT_ERA.RData")
NA_MaxT <- NA_MaxT - 273.15 # change from Kelvin to Celsius
nlon <- length(NA_lon); nlat <- length(NA_lat)
# numbers of lon/lat grid points
nmon <- 12; nyr <- 39 # numbers of month/year
avg_by_mon <- array(dim = c(nlon, nlat, nmon, nyr))
```

Code provided for Problem 2 with New York, New York coordinates implemented.

```
library(fields)
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

```
## Loading required package: grid
```

```
## Spam version 2.6-0 (2020-12-14) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

```r
ny.lon.lat <- c(-74.0060, 40.7128) # New York, New York


library(fields)
NewYork <- rdist.earth(matrix(ny.lon.lat, 1, 2),
expand.grid(NA_lon, NA_lat), miles = F)

# find closest location in data to home loacation
id <- which.min(NewYork)
(lon_id <- id %% nlon)
```

```
## [1] 69
```

```r
(lat_id <- id %/% nlon + 1)
```

```
## [1] 23
```

```r
# Check
(rdist.earth(matrix(ny.lon.lat, 1, 2),
matrix(c(NA_lon[lon_id], NA_lat[lat_id]), 1, 2), miles = F)
== min(NewYork))
```

```
##      [,1]
## [1,] TRUE
```

```r
# confirm the coordinates
NA_lon[lon_id]
```

```
## [1] -74.25
```

```r
NA_lat[lat_id]
```

```
## [1] 40.5
```

Closest location to New York, New York is at longitude -69 and latitude 23.

Problem 2 sub-parts.

(a) Compute and plot the annual average temperature values using the

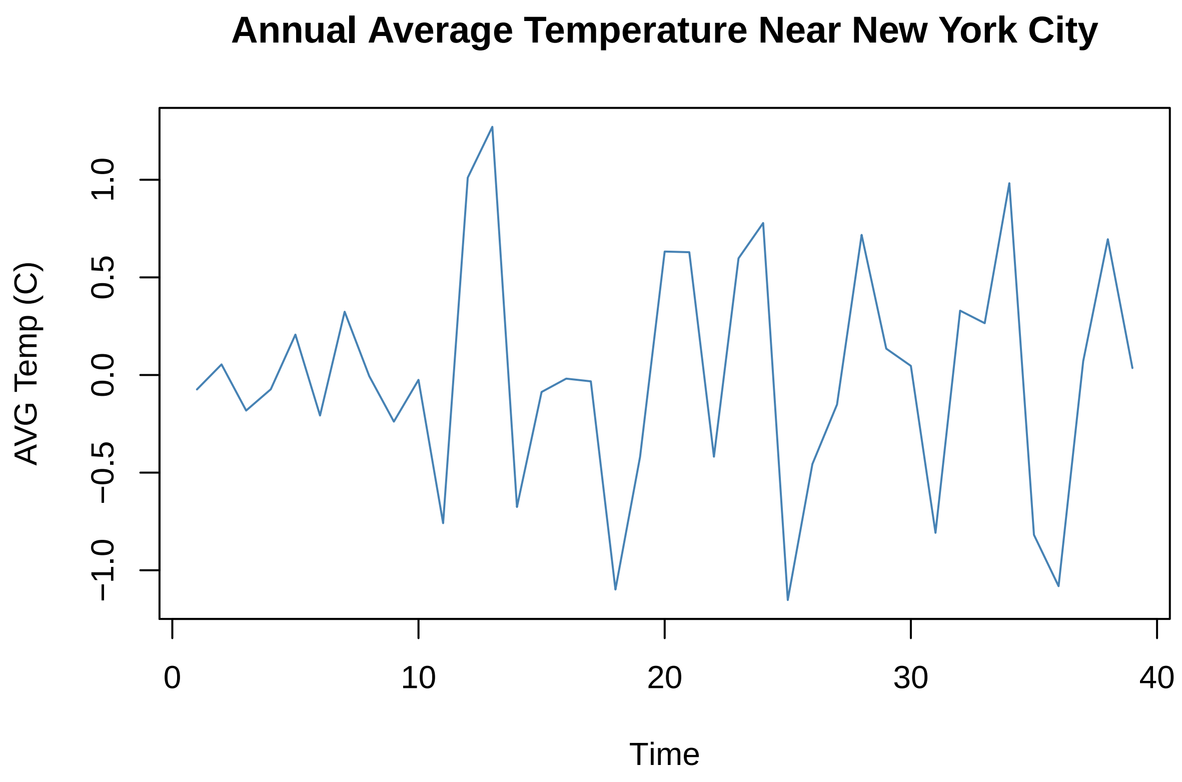
monthly average data we get from Problem 1. You can use the following script to compute the annual average temperature values:

Code provided for problem 4.a in lecture video:

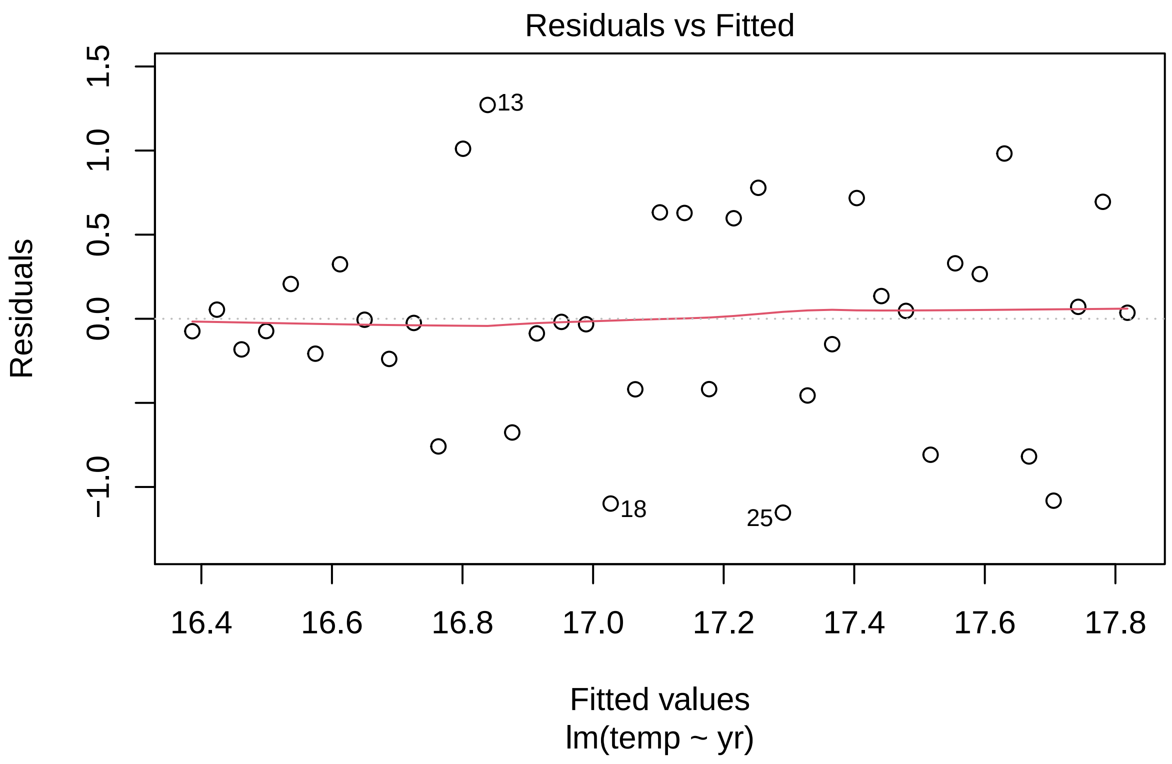
```r
# code used to create monthly averages
for (i in 1:nlon){
  for (j in 1:nlat){
    dat <- cbind(NA_MaxT[i,j,], as.factor(mon), as.factor(yr))
    avg_by_mon[i,j,,] <- tapply(dat[,1], list(dat[,2], dat[,3]),mean)
  }
}
save(avg_by_mon, file= "NA_MaxT_byMon_ERA.RData")
load("NA_MaxT_byMon_ERA.RData")
```

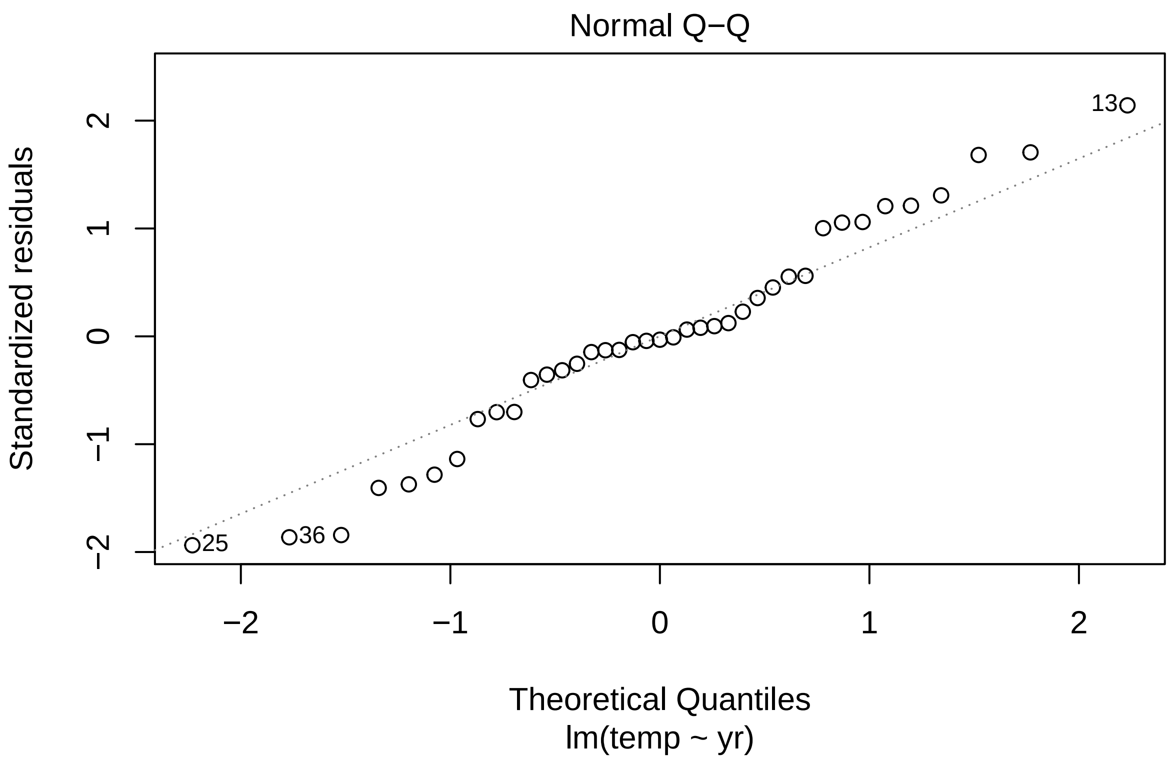
Code provided in project:

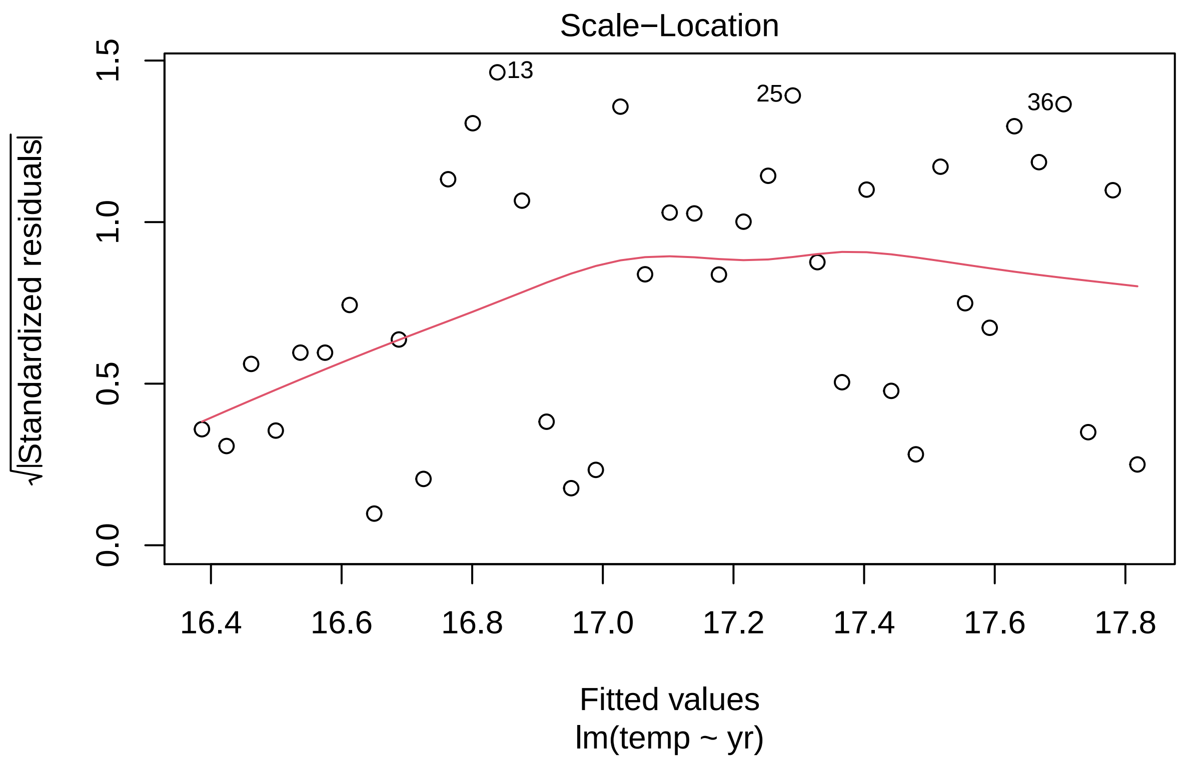
```
dat <- cbind(c(avg_by_mon[lon_id, lat_id,,]),
as.factor(rep(1979:2017, each = 12)))
ann_mean <- tapply(dat[,1], list(dat[, 2]), mean)


#fit the model for trend line
yr <- 1979:2017
df <- data.frame(temp = ann_mean, yr)
lm <- lm(temp~yr, df)

# Lecture plot
ts.plot(lm$residuals, main = "Annual Average Temperature Near New York City", col = "steelBlue", ylab =
points(ann_mean)
```

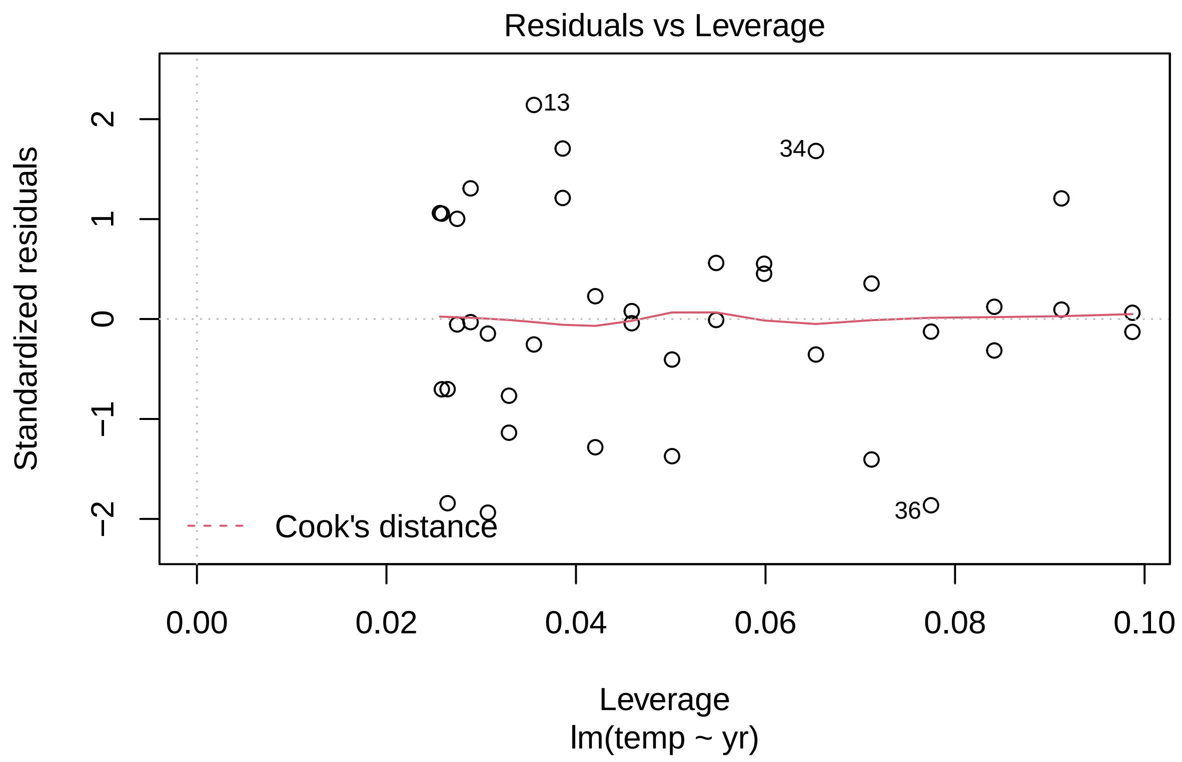


```
# residual plot and trends
plot(lm)
```


Residuals vs Fitted
Residuals
Fitted values
lm(temp ~ yr)
13
18
25
1.5
1.0
0.5
0.0
−1.0
16.4
16.6
16.8
17.0
17.2
17.4
17.6
17.8


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(temp ~ yr)
13
36
25
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
16.4
16.6
16.8
17.0
17.2
17.4
17.6
17.8
13
25
36
Fitted values
lm(temp ~ yr)



(b) Describe the main features of the annual time series data in (a).

Solution:

There is a slight positive trend with a lot of sudden shifts all being between 16 and 19 degrees. The sudden shifts are probably due to seasonal patterns and although they seem to have random variation, they are still indicative of multiplicative changes over time seeing as how the variance at the start is very small and increases over time.

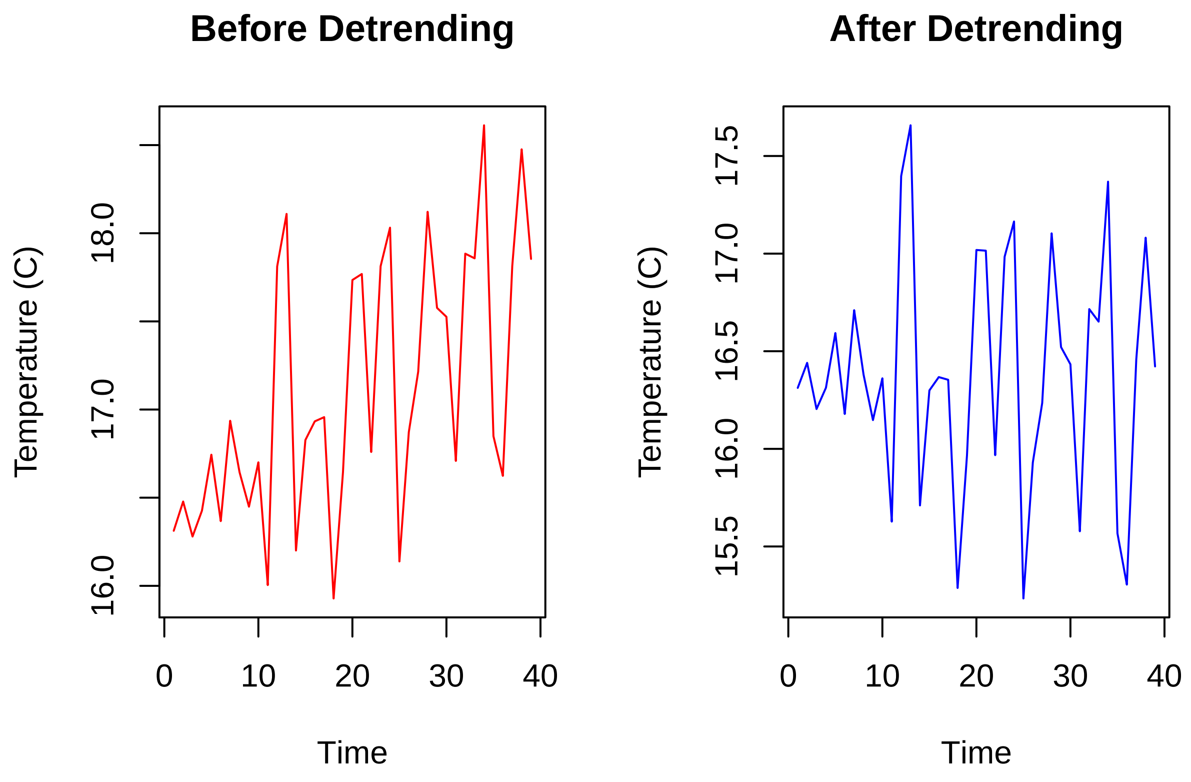
(c) Is it reasonable to assume that there is a linear trend? If so, estimate and remove the trend to get the detrended time series.

```
# See what the trend model is
summary(lm)
```

Code:

```
##
## Call:
## lm(formula = temp ~ yr, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.15250 -0.32859 -0.01856  0.32679  1.27111
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -58.204527  17.179607  -3.388  0.00168 **
## yr            0.037691   0.008598   4.384 9.29e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6043 on 37 degrees of freedom
## Multiple R-squared:  0.3418, Adjusted R-squared:  0.324
## F-statistic: 19.22 on 1 and 37 DF,  p-value: 9.29e-05
```

```
# Create new ts for ease of use; difficulty with prior fit for forecast
temp <- ts(
  c(16.31222, 16.47818, 16.27945, 16.42619 ,16.74389 ,16.36712, 16.93632, 16.64439, 16.44893, 16.70055,
frequency = 1)

# make sure our data was plotted correctly and should be the same as above
par(mfrow=c(1,2))
plot(temp, main = "Before Detrending", ylab = "Temperature (C)", col = "red")

# create new detrended time series using linear model
linear_fit <- lm(temp ~ time(temp))

# detrend plot
plot(temp - linear_fit$fitted.values + linear_fit$fitted.values[1],
    ylab = "Temperature (C)", main = "After Detrending", col = "blue")
```



```
# intercept and slope
coef(lm)
```

```
##   (Intercept)           yr
## -58.20452662   0.03769111
```

```
# New detrended intercept and slope
coef(linear_fit)
```

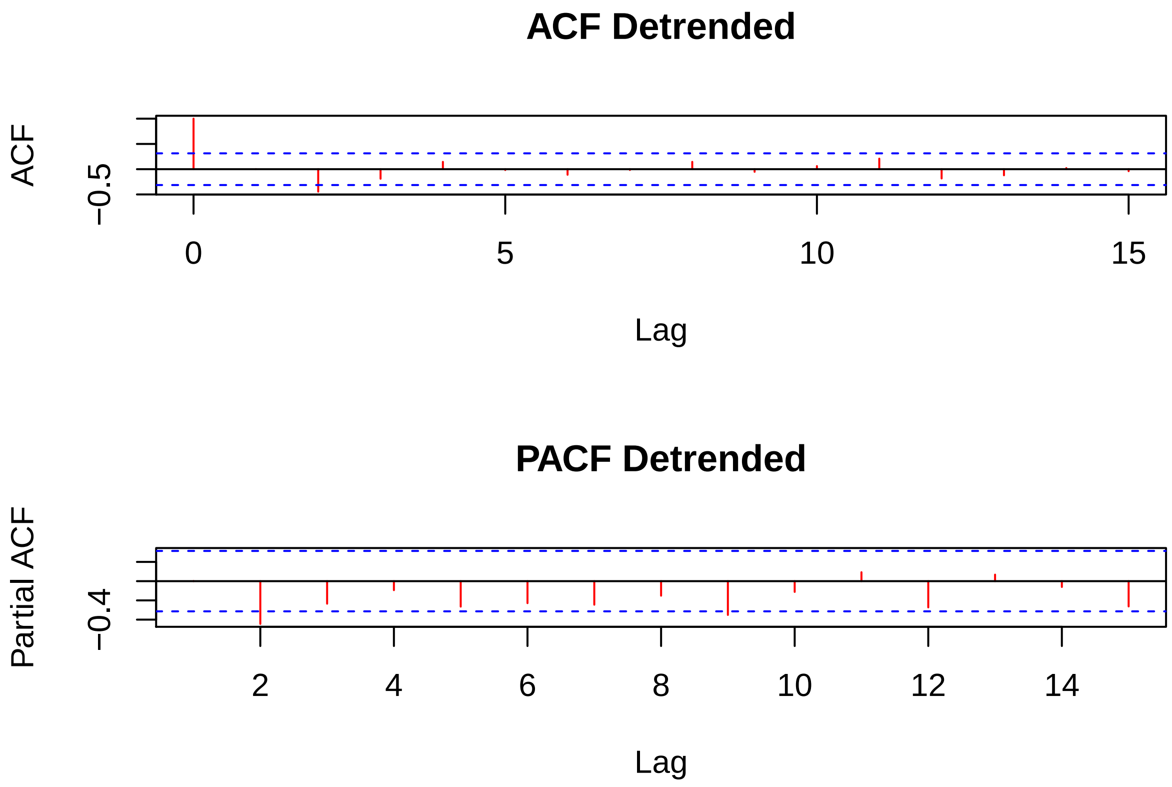
```
## (Intercept)  time(temp)
## 16.34849714  0.03769104
```

Solution:

It is reasonable to assume there is a linear trend in the data we have used. After using the summary function, we can see that there is a slope of 0.037691 with a y-intercept of -58.204. After detrending the time series data and making the plot a stationary series we get a new slope of 0.2638373 and a y-intercept of 16.1223509. This makes the graph more parallel to the x axis where there is less of a trend.

(d) Make acf and pacf plots for the detrended time series.

```
par(mfrow= c(2,1))

# acf plot of detrended ts
acf(linear_fit$residuals, main = "ACF Detrended", col = "red")

# pacf plot of detrended ts
pacf(linear_fit$residuals, main = "PACF Detrended", col = "red")
```



Code:

Solution:

After plotting the ACF and PACF, we can use the plots to identify the orders of AR and MA in the terms of the ARMA model.

Best model selection:

AR = the ACF plot will gradually decrease and simultaneously the PACF should have a sharp drop after p significant lags.

MA = opposite from the ACF and PACF plots in AR, meaning that: the ACF should show a sharp drop after a certain q number of lags while PACF should show a geometric or gradual decreasing trend.

ARMA = both ACF and PACF plots demonstrate a gradual decreasing pattern.

Given the descriptions and the plots above, it looks like either MA or ARMA could be the appropriate model to pursue. So in part e, I will test both.

(e) Fit some ARMA models and perform a model selection to determine the "best"

model.

```
library(lmtest)
```

Code:

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

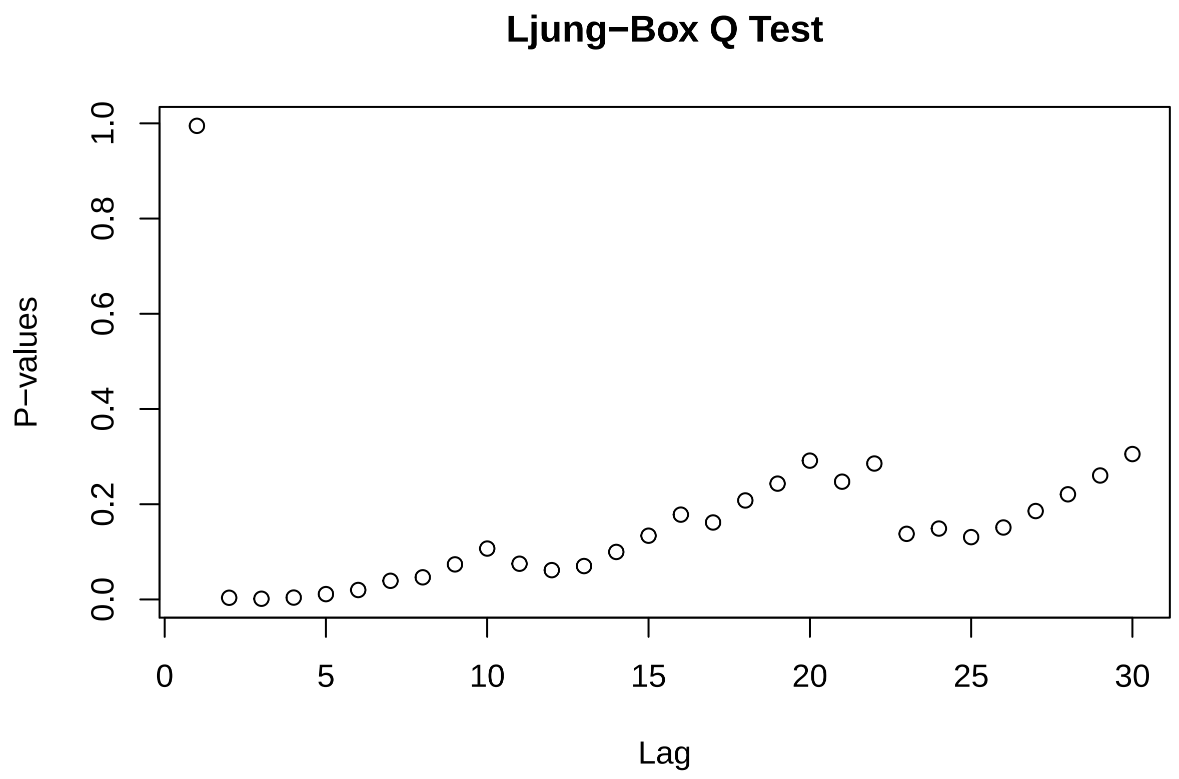
```
library(FitAR)
```

```
## Loading required package: lattice

## Loading required package: leaps

## Loading required package: ltsa

## Loading required package: bestglm
```

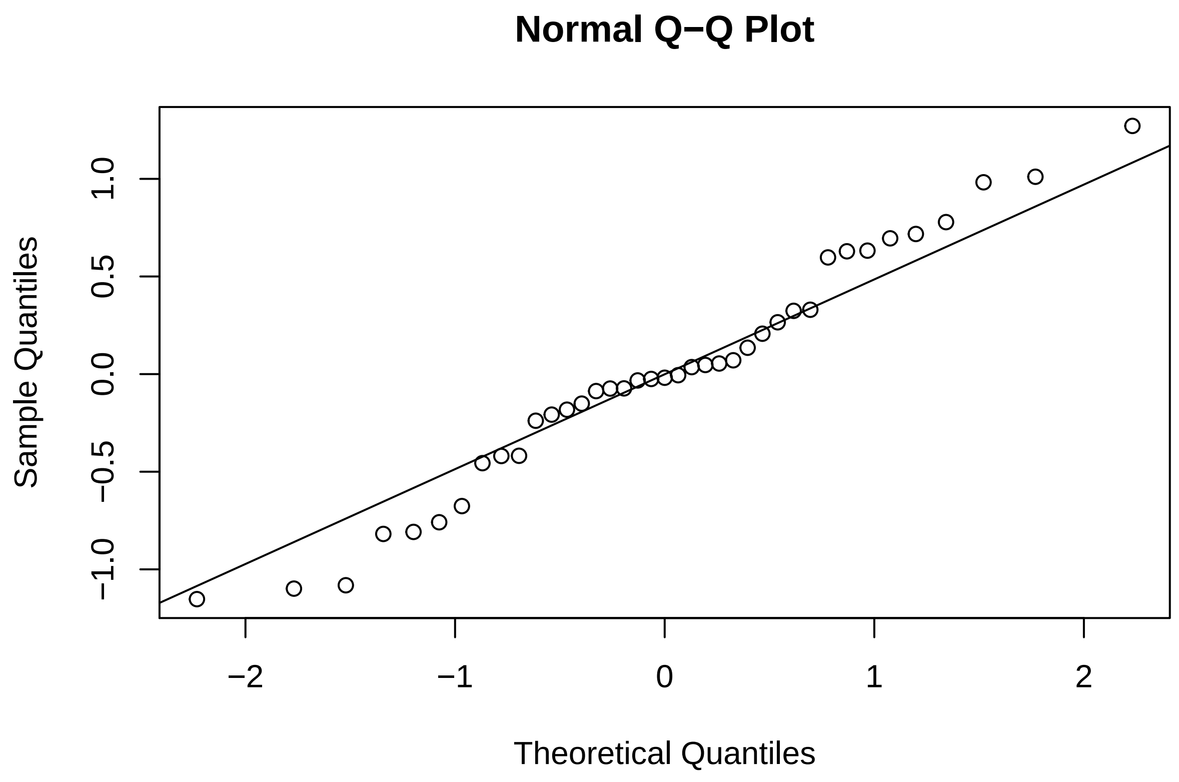
```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

##
## Attaching package: 'forecast'

## The following object is masked from 'package:FitAR':
##
##     BoxCox
```

```
# Ljung-Box Q test to see if residuals have autocorrelation or not
boxresult <-LjungBoxTest(linear_fit$residuals,k=2,StartLag=1)
plot(boxresult[,3],main= "Ljung-Box Q Test", ylab= "P-values", xlab= "Lag")
```



```
qqnorm(linear_fit$residuals)
qqline(linear_fit$residuals)
```



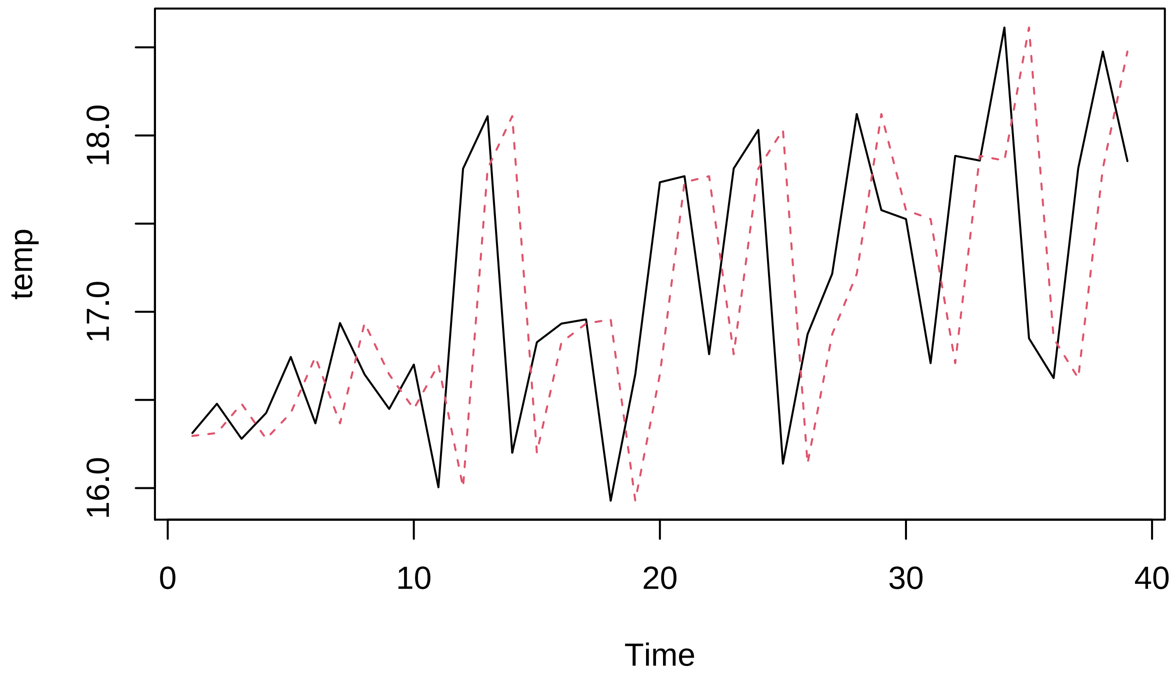
```r
# combination of unit root tests
bestm <- auto.arima(temp, trace=TRUE)
```

```
##
##  ARIMA(2,1,2) with drift         : Inf
##  ARIMA(0,1,0) with drift         : 99.1835
##  ARIMA(1,1,0) with drift         : 98.52023
##  ARIMA(0,1,1) with drift         : Inf
##  ARIMA(0,1,0)                    : 97.03975
##  ARIMA(1,1,1) with drift         : Inf
##
##  Best model: ARIMA(0,1,0)
```

```r
summary(bestm)
```

```
## Series: temp
## ARIMA(0,1,0)
##
## sigma^2 estimated as 0.712:  log likelihood=-47.46
## AIC=96.93   AICc=97.04   BIC=98.57
##
## Training set error measures:
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.03995903 0.8328845 0.6373872 0.1156108 3.733443 0.9749988
##                    ACF1
## Training set -0.277891
```

```r
#plotting the series along with the fitted values
ts.plot(temp)
BestFit <- temp - residuals(bestm)
points(BestFit, type = "l", col = 2, lty = 2)
```



Solution:

Using the function auto.arima, we use the maximum likelihood estimation (MLE) to estimate the ARIMA model. This package is used to determine the best order of the model to be fitted to the data. It will test three variables p, d, and q for the order of the autoregressive parts of the model. The function has determined that (0,1,0) is our best order. This is determined by finding the smallest Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC) values possible. The AIC value is 96.93 and the BIC value is 98.57.

We also conducted a Ljung-Box Q test to see if residuals have autocorrelation or not. What we determined is that the values are normal and the p-values indicate being significant since they are below 0.05

(f) Perform a one-year-ahead forecast. Calculate the prediction and provide a

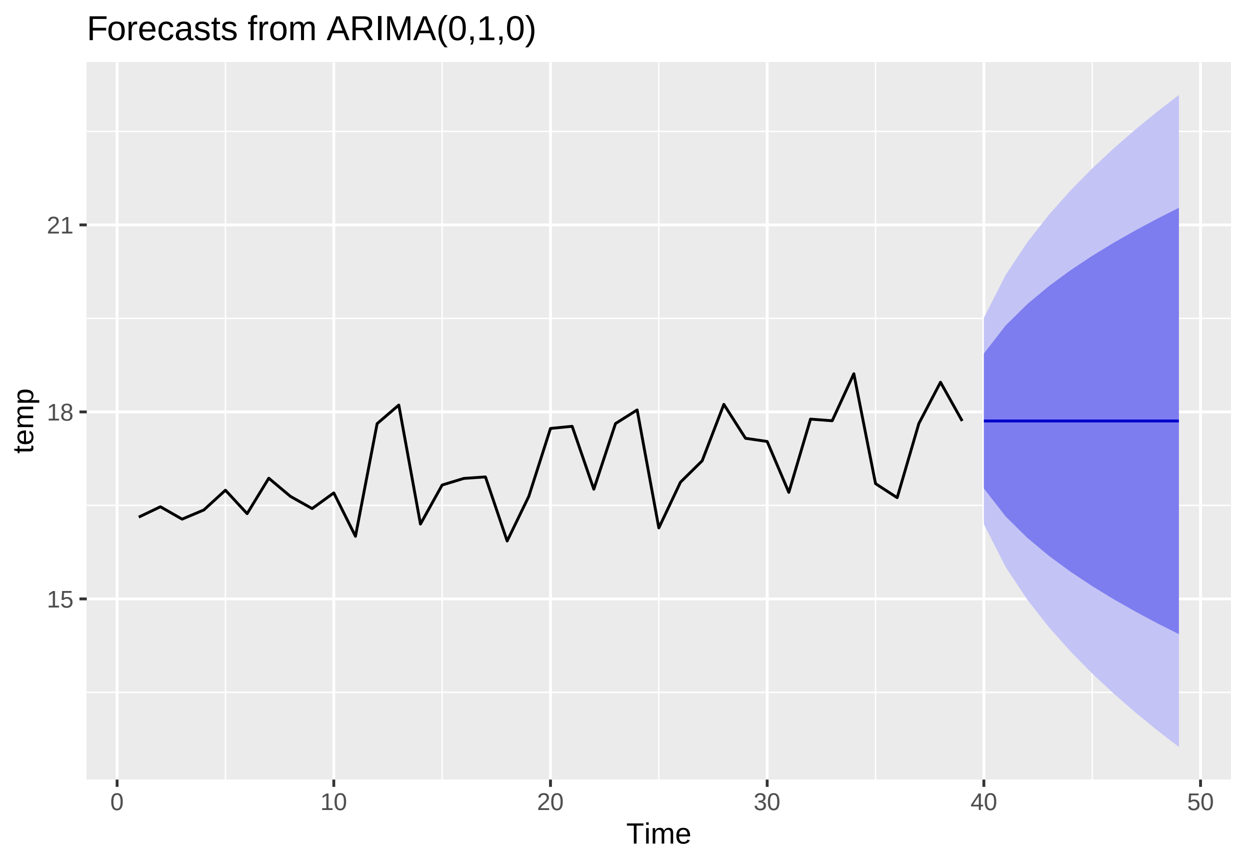
95% prediction interval.

```r
#Load Libraries
library(fUnitRoots)
```

Code:

```
## Loading required package: timeDate
## Loading required package: timeSeries

##
## Attaching package: 'timeSeries'

## The following object is masked from 'package:zoo':
##
##     time<-

## Loading required package: fBasics
```

```
library(lmtest)
library(FitAR)

# reapplying the moving average
model1 <- arima(bestm$residuals, order = c(0,1,0))

# plot for prediction
autoplot(forecast(bestm))
```

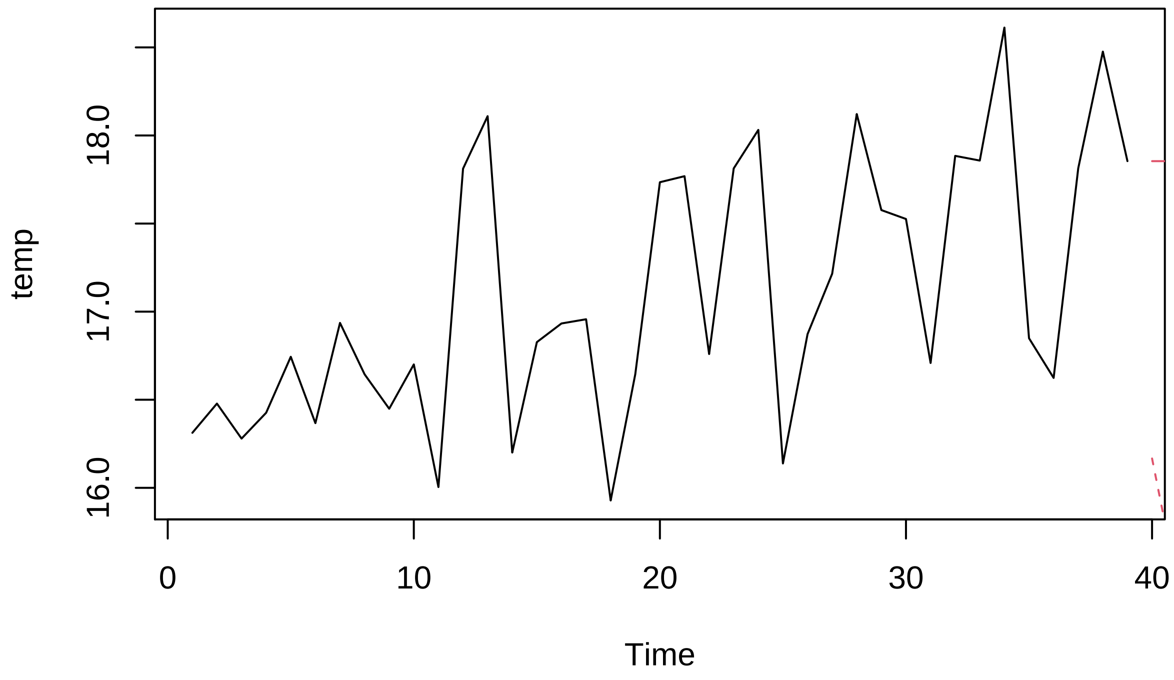


```
# Create prediction for the best model, 95% by default
predict(lm,data.frame(yr=2018),interval = "prediction", level=0.95)
```

```
##        fit      lwr      upr
## 1 17.85614 16.56803 19.14425
```

```
# a one year ahead forecast
forecast1 <- forecast(bestm)
```

```r
# plot with predict function applied
ts.plot(temp)
AR_forecast <- predict(bestm, n.ahead = 10)$pred
AR_forecast_se <- predict(bestm, n.ahead = 10)$se
points(AR_forecast, type = "l", col = 2)
points(AR_forecast - 2*AR_forecast_se, type = "l", col = 2, lty = 2)
points(AR_forecast + 2*AR_forecast_se, type = "l", col = 2, lty = 2)
```



Solution:

For the prediction, I got a value of 17.85614, which is accurate with the slight positive trend we concluded above. We then get the lower and upper bounds of 19.80841 and 20.58012.

Second Problem of the Assignment:

Problem 3

In this problem we will use the fields R package to conduct spatial interpolation using Gaussian process model. We will use airqual dataset in the rspatial package to interpolate average ozone levels (from 1980 to 2009) for California.

Use the following script to load the data:

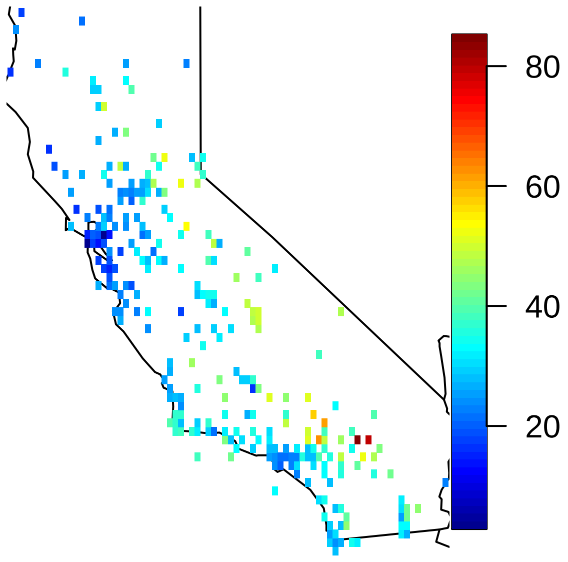
```r
library(rspatial)
```

```
## Loading required package: raster

## Loading required package: sp
```

```
x <- sp_data("airqual")
loc <- cbind(x$LONGITUDE, x$LATITUDE)
rg <- apply(loc, 2, range)
y <- x$OZDLYAV * 1000
```

Visualize the spatial data using the script below and describe your findings:

```
library(maps)
library(fields)
map("state", xlim = rg[, 1], ylim = rg[, 2])
quilt.plot(loc, y, nx = 80, ny = 64, add = T)
```



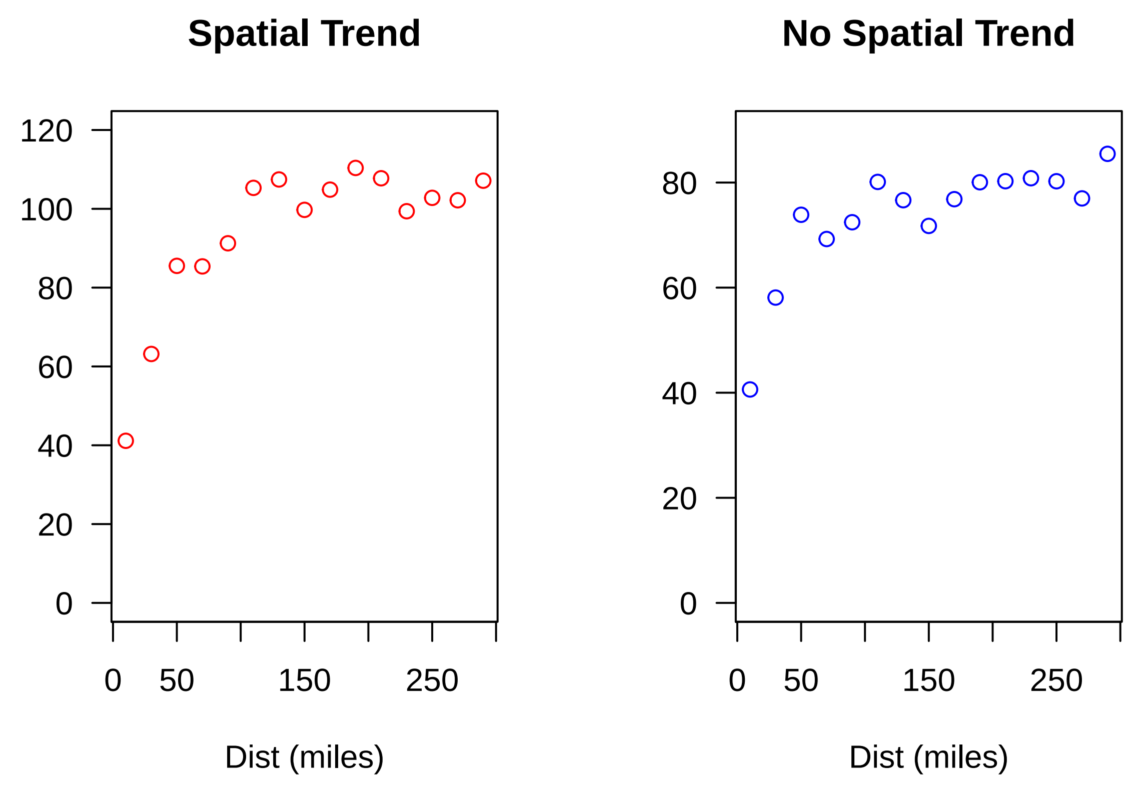
Findings:

From this plot we can see that for the most part California is around a measurement of 30 or lower for ozone levels, especially along the coast. There are more extreme ozone levels that are more central state with the highest being in the south central area of the state. My guess is the area with an ozone level of 80+ is Los Angeles, which is more populated.

Make two variograms, one without spatial trend and another one with linear

spatial trend. Explain the differences between these two variograms. You may use the following script to make variograms:

```r
library(maps)
library(fields)
par(mfrow = c(1,2))

vgram <- vgram(loc = loc, y = y, N = 30, lon.lat = T)
plot(vgram$stats["mean", 1:15] ~ vgram$centers[1:15],
main = "Spatial Trend", las = 1, ylim = c(0, 120), col = "red", ylab = "", xlab = "Dist (miles)")

# remove linear spatial trend
lm <- lm(y ~ loc)

vgram <- vgram(loc = loc, y = lm$residuals, N = 30, lon.lat = TRUE)
plot(vgram$stats["mean", 1:15] ~ vgram$centers[1:15],
main = "No Spatial Trend", las = 1, col = "blue", ylim = c(0, 90), ylab = "",
xlab = "Dist (miles)")
```



Explain the difference between the variograms:

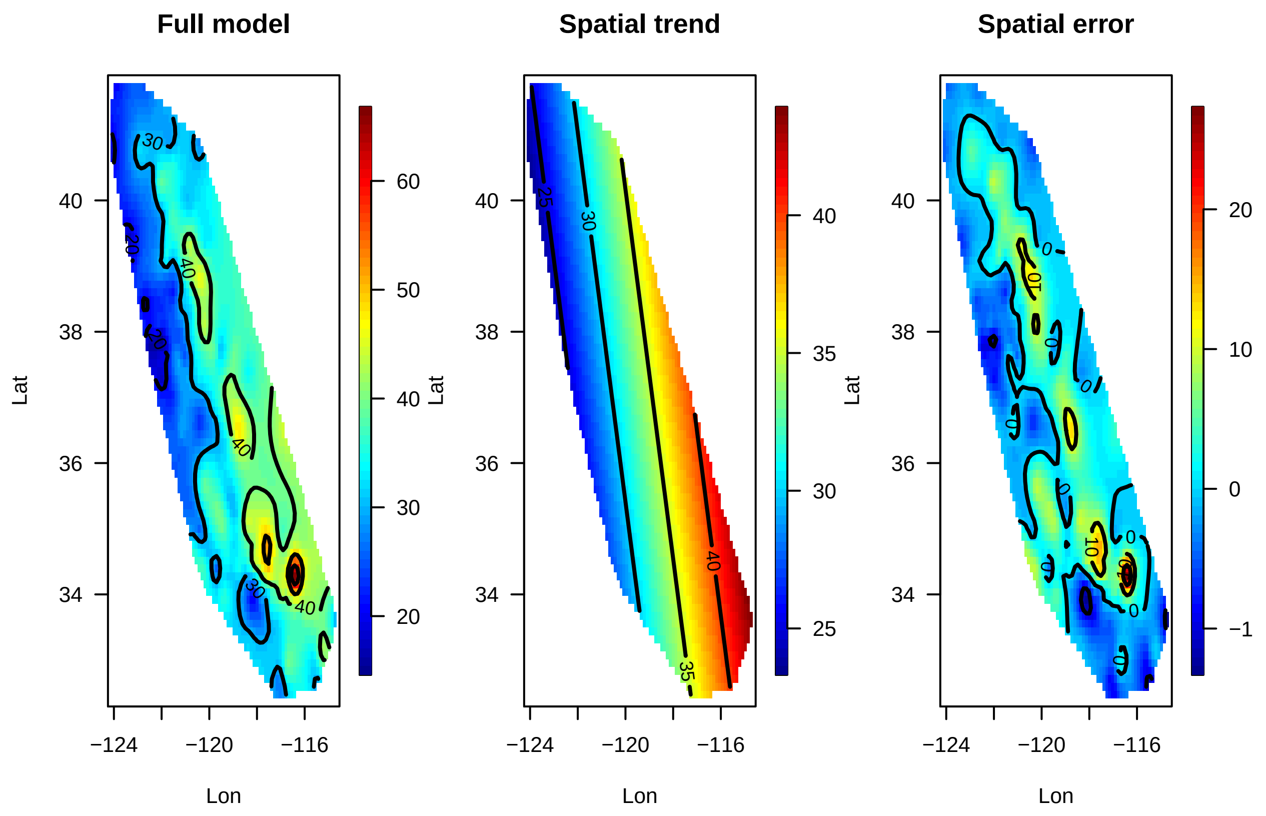
some of the differences we see in the trends is both shape and in the values for semivariance. The shape for both seem to be between a spherical and circular/exponential fit. A spherical shape has more of an extreme curve while circular/expontential is a subtle curve. I would say that spatial has a spherical shape and the non-spatial trend has a circular fit.Although there is not a huge noticable difference in shape. The y axis values are quite large for spatial trend and much shorter for non-spatial.

Use the following script to fit a Gaussian process to the data and to decompose the prediction into spatial trend part and spatially correlated error part:

```
fit <- spatialProcess(loc, y)
out.full <- predictSurface(fit)
out.poly <- predictSurface(fit, just.fixed = T)
out.spatial <- out.full
out.spatial$z <- out.full$z - out.poly$z
set.panel(1, 3)
```

```
## plot window will lay out plots in a 1 by 3 matrix
```

```
surface(out.full, las = 1, xlab = "Lon", ylab = "Lat")
title("Full model")
surface(out.poly, las = 1, xlab = "Lon", ylab = "Lat")
title("Spatial trend")
surface(out.spatial, las = 1, xlab = "Lon", ylab = "Lat")
title("Spatial error")
```

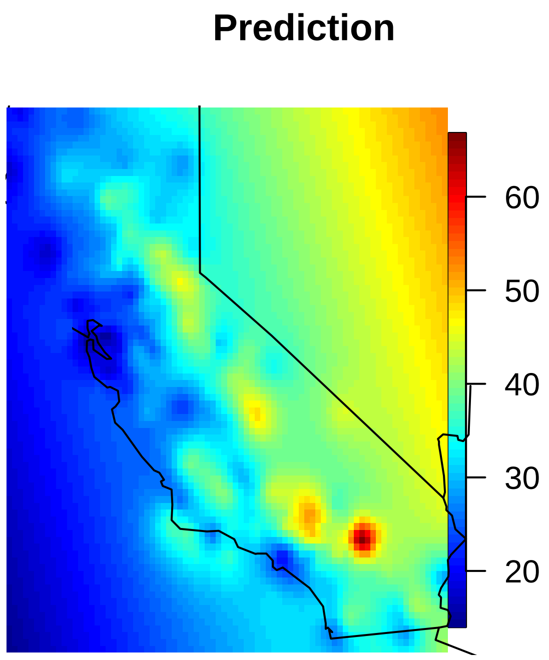


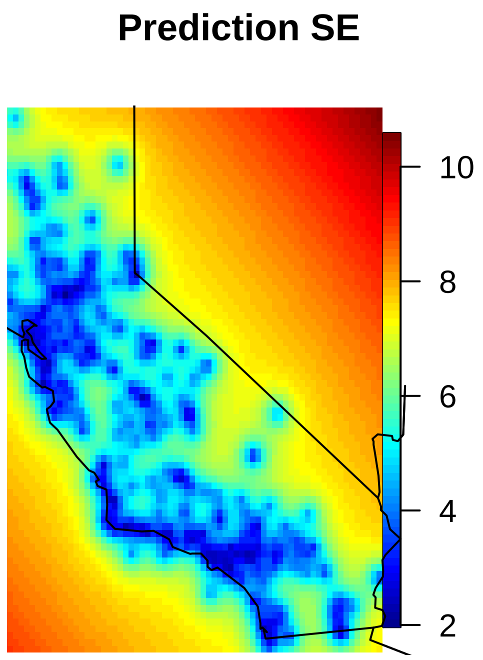
Make prediction map and the associated prediction uncertainty map using the

script below. Can you explain the spatial variation in the uncertainty map?

```
library(fields)
xg <- fields.x.to.grid(loc)
Pred <- predict.mKrig(fit, xnew = expand.grid(xg))
SE <- predictSE(fit, xnew = expand.grid(xg))
```

```r
par(mfrow = c(1, 2), las = 1)

map("state", xlim = rg[, 1], ylim = rg[, 2])
image.plot(xg[[1]], xg[[2]], matrix(Pred, 80), add = T)
map("state", xlim = rg[, 1], ylim = rg[, 2], add = T)
title("Prediction")

map("state", xlim = rg[, 1], ylim = rg[, 2])
image.plot(xg[[1]], xg[[2]], matrix(SE, 80), add = T)
map("state", xlim = rg[, 1], ylim = rg[, 2], add = T)
title("Prediction SE")
```





Can you explain the spatial variation in the uncertainty map?

In our uncertainty map the higher levels ("red") are high certainty and lower levels (blue) is the lower lever of certainty. Since we don't really have data outside the state we should not pay much attention to the outside of the border. The data in our prediction map is very similar to our original spatial plot above, which is expected as population increases in the state and slowly the values will rise anywhere the population does. This is indicative of a linear pattern. Uncertainty is related to the amount of spatial variability and dependent of the sampling. The function we used predictSE(), computes predicted values on abundance and standard errors based on the estimates from our fitted model. This means there are fluctuations in the data that make the normally high levels of ozone have less uncertainty. This is indicative of somewhat higher standard errors than our prediction map, which the higher the number for standard error the more spread out your data is.