MTHM505 - Data Modelling in Space and Time

Joshua Harrison

2025-05-05

AI-supported/AI-integrated use is permitted in this assessment. I acknowledge the following uses of GenAI tools in this assessment:

- I have used GenAI tools to proofread and correct grammar or spelling errors
- I have used GenAI to help cross-check coding/plotting

I declare that I have referenced use of GenAI outputs within my assessment in line with the University referencing guidelines.

Table of contents

1. Sea Surface Temperature Modelling

```
library(ggplot2)
library(rmarkdown)
library(tidyverse)
library(lmerTest)
library(zoo)
library(MASS)
library(knitr)
library(patchwork)
library(lmtest)
library(AER)
library(geosphere)
library(ggplot2)
library(sp)
library(geoR)
library(ggmap)
library(sf)
library(rnaturalearth)
library(rnaturalearthdata)
library(MCMCpack)
library(dlm)
library(forecast)
library(lubridate)
library(reshape2)
library(ggspatial)
library(grid)
```

(a)

```
Kuroshio_data <- read.csv("C:/Users/joshh/OneDrive - University of Exeter/MSc Applied Data S
head(Kuroshio_data)
```

```
              date    lon   lat    id pt  sst sf   at af
1 01/01/1996 00:00 144.10 32.60  WATF  5 21.1  1 16.7  1
2 01/01/1996 00:00 141.40 36.50 MQWU7  5 11.7  1  7.2  1
3 01/01/1996 00:00 140.70 35.00 LATI4  5 18.2  1 11.0  1
4 01/01/1996 00:00 143.00 37.20 OWEB2  5 12.9  1  6.2  1
5 01/01/1996 00:00 149.44 30.19 21573  7 19.4  1 19.0  1
```

```
6 01/01/1996 01:00 142.10 38.70  MQWU  5  6.5  1  0.6  1
```

After printing out the head of the data, we can see the column names of 'date', 'lon', 'lat', 'id', 'pt', 'sst', 'sf', 'at' and 'af'. These columns describe the individual data points collected to measure Sea Surface Temperature in the Kuroshio current off Japan from the first two days of January 1996.

First, we will convert these data to a geodata object and remove NA values from the 'sst' column before visualising the data collected.

```
# Convert the dataset to a geodata object
Kuroshio_geodata <- as.geodata(Kuroshio_data, coords.col = c("lon", "lat"),
                               data.col = "sst")

# Removing rows with NA in the 'sst' column only
Kuroshio_data_clean <- Kuroshio_data %>%
  filter(!is.na(sst))  # This keeps rows where 'sst' is not NA
```

```
# Plot SST data on a map
ggplot(Kuroshio_data_clean, aes(x = lon, y = lat, color = sst)) +
  geom_point() +
  scale_color_viridis_c(option = "inferno", name = "SST (°C)") +
  labs(title = "Sea Surface Temperature (SST) Observations",
       x = "Longitude", y = "Latitude", color = "SST") +
  theme_minimal()
```

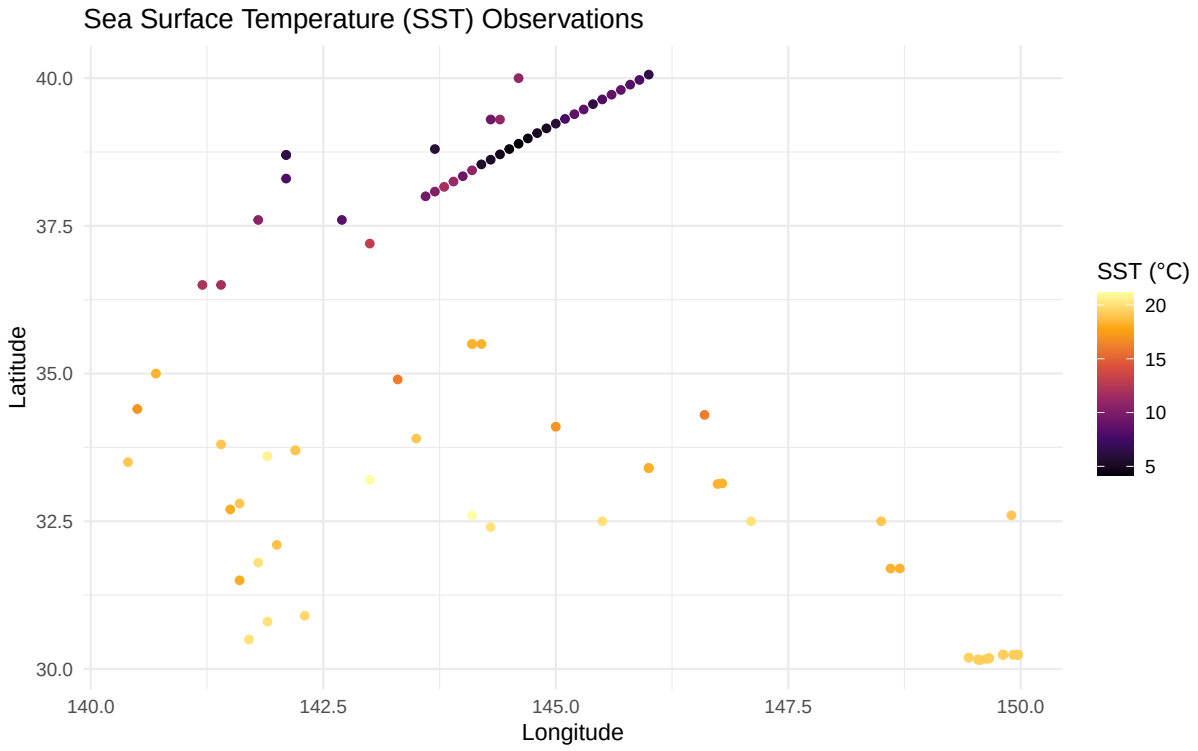


Figure 1: SST Observations and their Geographical Distribution

The scatterplot in Figure-1 above displays the Sea Surface Temperature (SST) against longitude and latitude in the Kuroshio current off Japan. The highest SST values are located closer to the 140-145 longitude range, while lower SST values are spread north above latitudes of 35.

High SST values around 20 degrees celsius indicate points that are likely influenced by the Kuroshio current. The current brings warm water from the tropics, which is evident in the elevated SST's. There is a cluster of high SST values around the 150 longitude and 30 latitude mark indicating a hotspot where SST is consistently high. These areas could refer to where the Kuroshio current is most active or areas where warm water is concentrated.

```
# Plotting the data in context with a map of Japan
Kuroshio_sf <- st_as_sf(Kuroshio_data_clean, coords = c("lon", "lat"),
                        crs = 4326)

# Getting country outlines for the basemap
japan_map <- ne_countries(scale = "medium", country = "Japan",
                          returnclass = "sf")
```

```r
# Plotting data on map of Japan
ggplot() +
  geom_sf(data = japan_map, fill = "grey90", color = "black") +
  geom_sf(data = Kuroshio_sf, aes(color = sst), size = 2.5, alpha = 0.8) +
  scale_color_viridis_c(name = "SST (°C)", option = "inferno") +
  coord_sf(xlim = c(125, 152), ylim = c(24, 46), expand = FALSE) +
  labs(
    title = "Sea Surface Temperature Observations in the Kuroshio Current",
    x = "Longitude",
    y = "Latitude"
  ) +
  theme_minimal() +
  theme(aspect.ratio = 1)
```

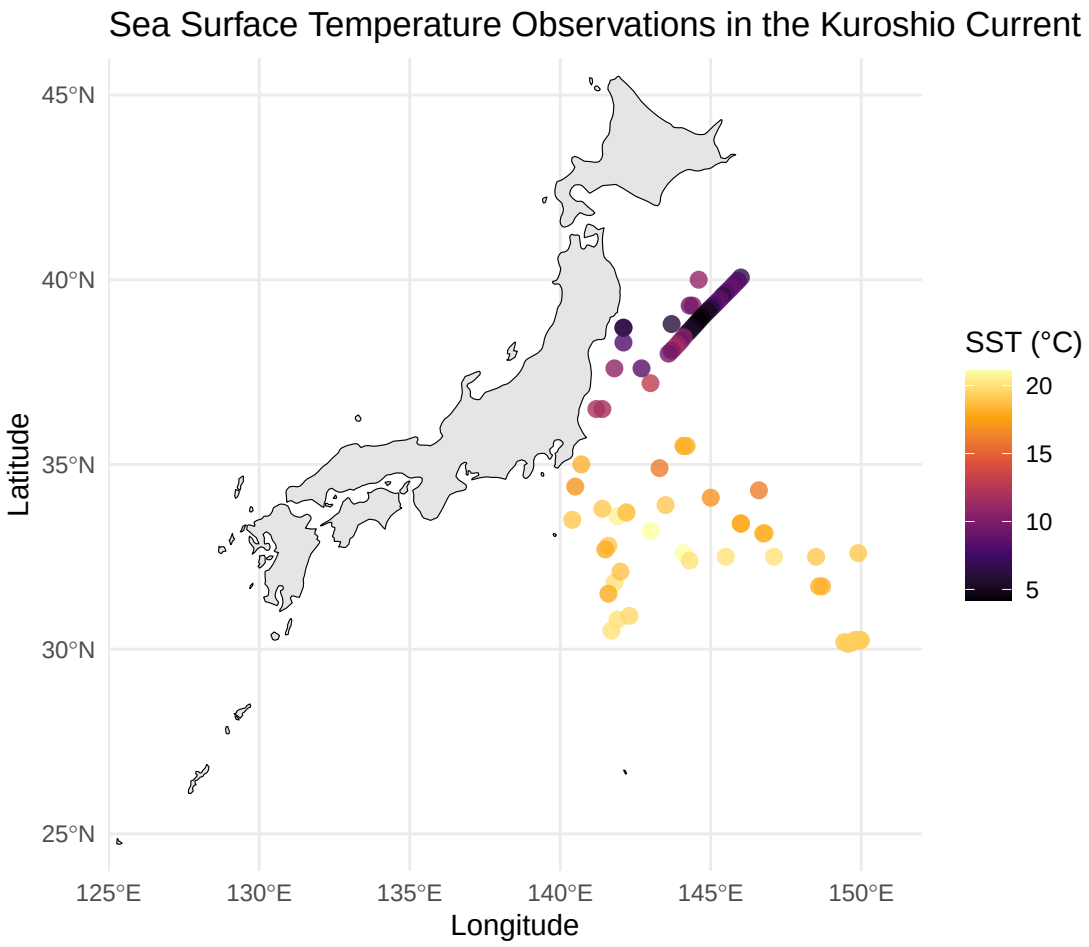


Figure 2: SST Observations on the Kurushio Coast

From figure-2 above, we can understand that areas closer to Japan have lower SST values (coloured dark purple to pink), with SST values ranging between 5-15 degrees celsius. As the data points move away from the coast and further south, the values increase in the range of 15-25 degrees celsius. The cluster of high SST values in the scatterplot from figure-1 can be seen in the south eastern part of the map.

Therefore, the oberserved regions in January 1996 further from Japan are warmer, while the coastal regions exhibit colder water temperatures.

(b)

We can select 5 random points from the dataset using `sample()` function to random select 5 rows.

```
# Aggregating the data by longitude and latitude, taking the mean SST for
# each group to deal with duplicates
trained_data_aggregated <- Kuroshio_data_clean %>%
  group_by(lon, lat) %>%
  summarise(
    date = first(date),
    id = first(id),
    pt = first(pt),
    sst = mean(sst, na.rm = TRUE),
    sf = first(sf),
    at = first(at),
    af = first(af),
    .groups = "drop"
  )

# Check the aggregated data
print(trained_data_aggregated)
```

```
# A tibble: 76 x 9
     lon   lat date             id       pt   sst    sf    at    af
   <dbl> <dbl> <chr>            <chr> <int> <dbl> <int> <dbl> <int>
 1  140.  33.5 01/01/1996 18:00 ZEOE      5  19       1  12       1
 2  140.  34.4 01/01/1996 03:00 VTFM      5  17       1  16       1
 3  141.  35   01/01/1996 00:00 LATI4     5  18.2     1  11       1
 4  141.  36.5 02/01/1996 06:00 3FFJ4     5  12       1   8.5     1
 5  141.  33.8 02/01/1996 15:00 JFYU      5  19       1  14       1
 6  141.  36.5 01/01/1996 00:00 MQWU7     5  11.7     1   7.2     1
 7  142.  32.7 02/01/1996 18:00 7MEW      5  18       1  13.5     1
 8  142.  31.5 02/01/1996 15:00 7MEW      5  18       1  14.5     1
```

```
 9  142.  32.8 02/01/1996 12:00 JFYU      5  19       1  14       1
10  142.  30.5 02/01/1996 12:00 7MEW      5  20       1  15       1
# i 66 more rows
```

```r
# Setting a random seed for reproducibility
set.seed(456)

# Randomly sample 5 rows from the dataset
random_points <- trained_data_aggregated[sample(nrow(trained_data_aggregated), 5), ]

# Viewing selected points
print(random_points)
```

```
# A tibble: 5 x 9
    lon   lat date             id           pt   sst    sf    at    af
  <dbl> <dbl> <chr>            <chr>     <int> <dbl> <int> <dbl> <int>
1  144.  38.5 02/01/1996 17:24 49  16760     5   5.6     1  NA      15
2  144.  39.3 02/01/1996 12:00 KGJB          5   9.4     1   2.8     1
3  144.  38.8 02/01/1996 06:00 JBKB          5   6       1   6       1
4  144.  38   02/01/1996 20:40 49  16760     5   9.5     1  NA      15
5  144.  32.6 01/01/1996 00:00 WATF          5  21.1     1  16.7     1
```

In order to report only the name, longitude, latitude and SST from the randomly selected points we can specify this as follows and then remove them from the dataset for the purpose of training models. We do this by filtering out the rows corresponding to the selected points.

```r
# Extracting relevant information
chosen_points <- random_points[, c("id", "lon", "lat", "sst")]

# Printing selected stations' details
print(chosen_points)
```

```
# A tibble: 5 x 4
  id          lon   lat   sst
  <chr>     <dbl> <dbl> <dbl>
1 49  16760  144.  38.5   5.6
2 KGJB       144.  39.3   9.4
3 JBKB       144.  38.8   6
4 49  16760  144.  38     9.5
5 WATF       144.  32.6  21.1
```

```
# Get the row numbers of the selected rows from chosen_points
selected_row_numbers <- row.names(chosen_points)

trained_data <- trained_data_aggregated %>%
  anti_join(chosen_points, by = c("lon", "lat"))

# Verify the number of rows left in the training dataset
print(nrow(trained_data))  # Should be 71 rows if 5 rows were removed
```

```
[1] 71
```

(c)

A variogram is the expected squared difference between data at (x) and (x + h), for distance (h):

$$2\gamma(h) = \mathbb{E}[(Y(x+h) - Y(x))^2] = \text{Var}[Y(x+h) - Y(x)].$$

```
# Creating geodata object
geodata_aggregated <- as.geodata(trained_data, coords.col = c("lon", "lat"),
                                 data.col = "sst")

# Calculating the sample variogram for the aggregated data
sample_vario_aggregated <- variog(geodata_aggregated, option = "bin")
```

```
variog: computing omnidirectional variogram
```

```
# Plotting the sample variogram
plot(sample_vario_aggregated, pch = 19, main = "Sample Variogram")
```

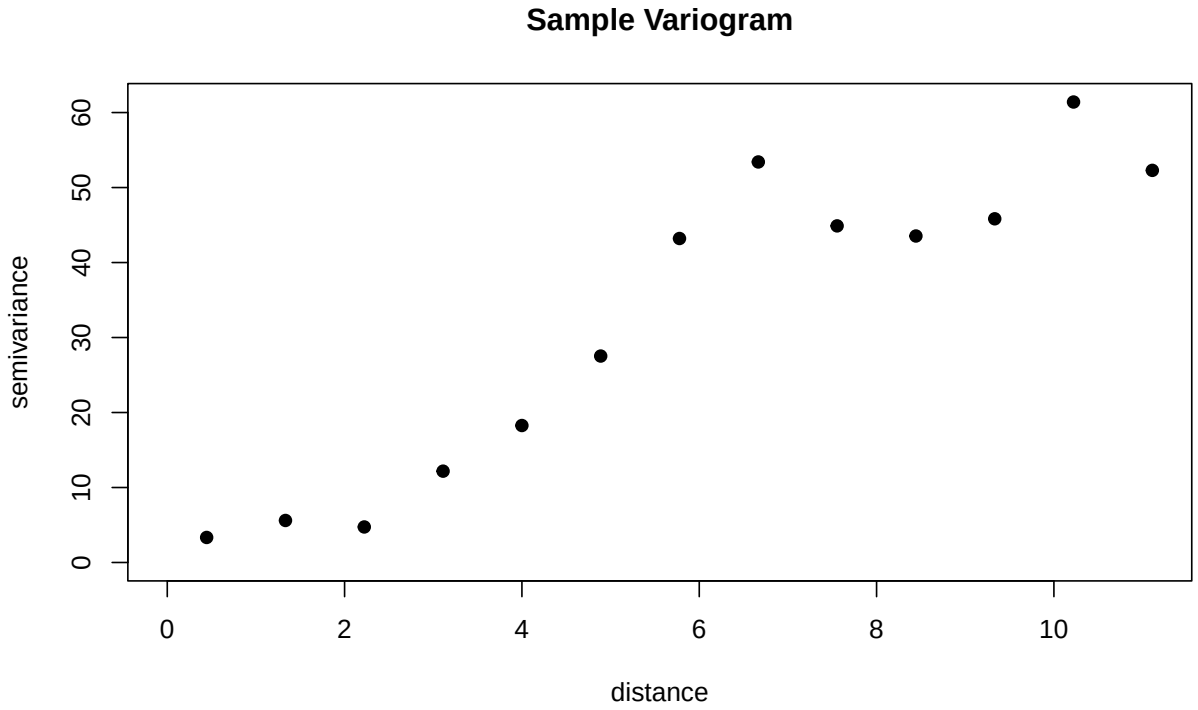


Figure 3: Sample Variogram

Looking at the variogram above, we can see that the semivariance increases with distance, but the pattern appears to be somewhat linear, with no major deviations or outliers. There is a steady increase which indicates a consistent spatial correlation between points over distance. The variogram does not show a clear plateau or flattening, suggesting that spatial correlation does not decay after a certain distance. Given the semivariance increases steadily with distance and does not appear to flatten, there is no clear cutoff point after which the further distance measurements become irrelevant. Therefore, since the variogram does not plateau, setting a maximum distance could be important to prevent the model from overfitting.

A nugget represents the variance at zero distance and typically accounts for measurement errors, small-scale variation, or unmodelled variability. Given that the semivariance in the variogram does not start at zero, there is a nugget effect present. This initial variance at zero distance is often attributed to spatial variability that is too fine-scale to be detected or modelled by the available data. In other words, it reflects small, localised errors or fine-scale processes that are not captured by the broader spatial trend modelled by the variogram.

The presence of this nugget indicates that a certain degree of randomness or unmodelled effects are inherent in the data, and these effects should be accounted for when fitting spatial models, as they could affect the accuracy of predictions, especially at smaller scales.

```r
# Setting the Maximum Distance and not including a nugget in the model

# Fitting the variogram model with the specificed maximum distance
sample_vario <- variog(geodata_aggregated, option = "bin", max.dist = 7)
```

```
variog: computing omnidirectional variogram
```

```r
# PLotting the sample variogram again with the maximum distance applied
plot(sample_vario, pch = 19, main = "Sample Variogram with Max Distance")
```

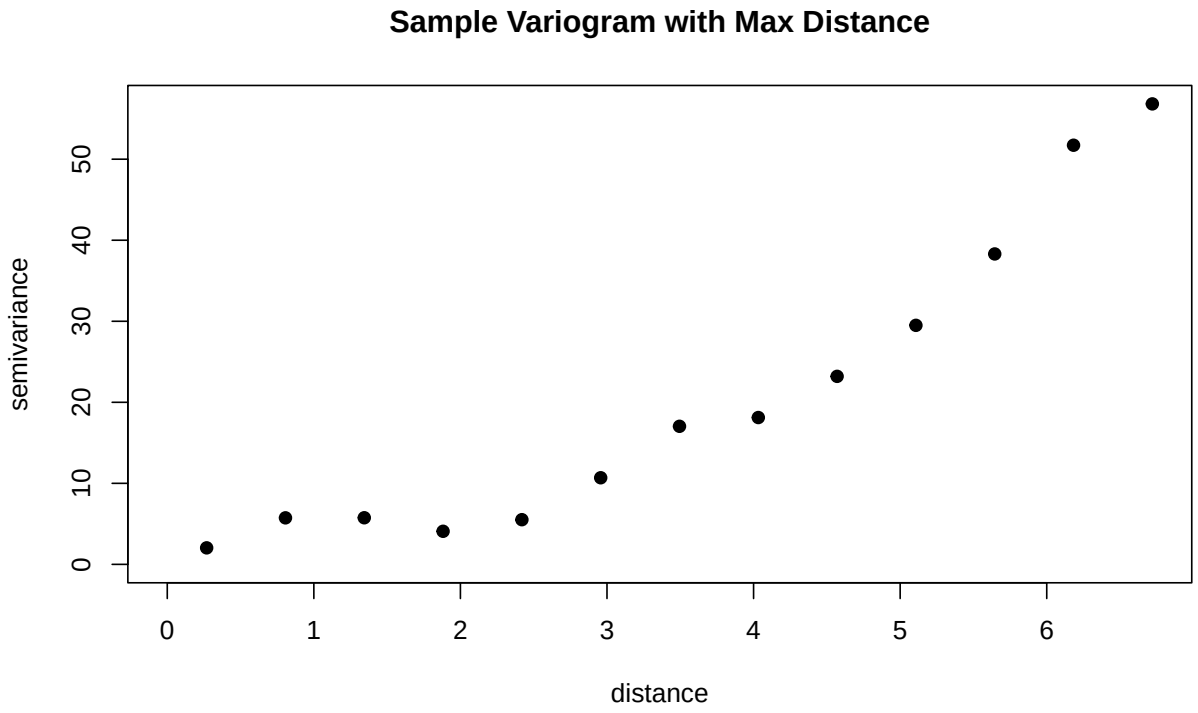


Figure 4: Sample Variogram with Max Distance

Based on the sample variogram, there is no significant trend in the data (e.g. no large-scale change in SST). Therefore, we will assume a constant mean function (i.e. no trend). We can use the function `variofit` to fit a parametric model, estimating the parameters using weighted least squares. We will initially use the Gaussian covariance function for the kriging model and then compare models using Powered Exponential and Matérn models to compare how well they fit the data. In all models we will include `fix.nugget = FALSE` which indicates that we are including a nugget in the model which will be estimated. We will keep a maximum distance of 7 for all models since the data begins to fall off after this point. We will use initial values of 60 and 7 for every model, reflecting the sill and range respectively based on the sample variogram.

Gaussian Model

We will now fit a Gaussian model. This model does not have a kappa parameter because the Gaussian covariance function is smooth by definition, with no explicit roughness control. This means that points remain correlated over long distances, gradually decaying in correlation. Based on the sample variogram, we again assume there is no significant trend in the data and therefore a constant mean function. We fit the model using the 'Gaussian' covariance function. We use `fix.nugget = FALSE` to allow the nugget to be estimated by the model. The initial values used are sill of 60 and range of 7.

```
# Specifying initial values for range and sill, and fitting the Gaussian model
# with a nugget
var_model_gaussian <- variofit(sample_vario, cov.model = "gaussian",
                               fix.nugget = FALSE, max.dist = 7,
                               ini.cov.pars = c(60, 7))
```

```
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
```

```
# Printing the fitted variogram model parameters for the Gaussian model print
(var_model_gaussian)
```

```
variofit: model parameters estimated by WLS (weighted least squares):
covariance model is: gaussian
parameter estimates:
     tausq    sigmasq        phi
    0.9084 15778.6679   113.8953
Practical Range with cor=0.05 for asymptotic range: 197.1321

variofit: minimised weighted sum of squares = 10582.21
```

The output summary from the Gaussian model displays the parameter estimates. The `optim` function was used for optimisation, which is typical for non-linear fitting in `geoR`. This minimises the weighted sum of squares to find the best-fitting model parameters.

The Gaussian Model estimated tausq at 0.9084. This represents the nugget effect or measurement error. This is positive, indicating there is some unexplained variability at zero distance. The model produced a sigmasq (σ^2) value of 15778.67, representing the sill value which is the total variance that the model can capture. This value suggests the total spatial variability in the data is quite high. This value represents the maximum variability that can be explained by the spatial correlation, after considering the nugget effect. The phi (ϕ) value is 113.89 which represents the range. Points separated by distances greater than 113.89 units will have very little spatial correlation. The practical range is 197.13, representing

the distance at which spatial correlation becomes negligible. A practical range of 197.13 units suggests that spatial correlation persists for longer distances compared to the range of 113.89 units. This model produces a minimised weighted sum of squares of 10582.21. This is the final loss function value after optimisation, representing the mode;'s goodness-of-fit. A smaller sum of squares indicates a better fit, but cross-validation should be performed to compare model accuracy.

Powered Exponential Model

Next, we will fit a powered exponential model with a fixed kappa of 1.5. The kappa parameter typically defines the smoothness of the spatial correlation. 1.5 is a reasonable assumption as it provides a balance between smooth and rough processes and is particularly useful when no strong prior knowledge is available for selecting specific smoothness. Based on the sample variogram, we again assume there is no significant trend in the data and therefore a constant mean function.

```
# Specifying initial values for range and sill,
# and fitting the powered exponential model with a nugget
# Fitting a variogram model with the powered exponential covariance function
var_model_powered_exponential <- variofit(sample_vario,
                                          cov.model = "powered.exponential",
                                          fix.nugget = FALSE, max.dist = 7,
                                          ini.cov.pars = c(60, 7), kappa = 1.5)
```

```
variofit: covariance model used is powered.exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
```

```
# Printing the fitted variogram model paramters for the exponential model
print(var_model_powered_exponential)
```

```
variofit: model parameters estimated by WLS (weighted least squares):
covariance model is: powered.exponential with fixed kappa = 1.5
parameter estimates:
     tausq    sigmasq        phi
    0.0000 70076.7906   828.0228
Practical Range with cor=0.05 for asymptotic range: 1720.723

variofit: minimised weighted sum of squares = 31076.18
```

This output summary shows us the estimates of the parameters for the powered exponential model. The value of 0 for tausq shows that there is no nugget effect in this model, meaning that the variance at zero

distance is negligible. The sigmasq (σ^2) value of 70076.79 represents the maximum semivariance, indicating that the model captures significant spatial variability in the data. The phi (ϕ) value of 828.02 indicates the range at which the spatial correlation becomes negligible. It shows that the spatial correlation diminishes after approximately 828 units. The practical range indicates the distance at which spatial correlation becomes practically zero. This range is 1720 units. The model has a minimised weighted sum of squares of 31076.18, representing how well the model fits the data. Smaller values generally indicate a better fit.

Matérn Model

We will now fit a Matérn model and compare its performance against the other two models. We will fit the model with a kappa value of 1.5 to fit smooth processes. We use `fix.nugget = FALSE` to allow the nugget to be estimated by the model. We keep the same parameters with max distance and initial values for range and sill. Based on the sample variogram, we again assume there is no significant trend in the data and therefore a constant mean function.

```
# Specifying initial values for range and sill, and fitting the Matérn model
# with a nugget
var_model_matern <- variofit(sample_vario, cov.model = "matern",
                             fix.nugget = FALSE, max.dist = 7,
                             ini.cov.pars = c(60, 7), kappa = 3/2)
```

```
variofit: covariance model used is matern
variofit: weights used: npairs
variofit: minimisation function used: optim
```

```
# Printing the vitted variogram model parameters for the Matérn model
print(var_model_matern)
```

```
variofit: model parameters estimated by WLS (weighted least squares):
covariance model is: matern with fixed kappa = 1.5
parameter estimates:
     tausq    sigmasq        phi
    0.8042 94012.6218   194.6294
Practical Range with cor=0.05 for asymptotic range: 923.2953

variofit: minimised weighted sum of squares = 10924.19
```

The output summary displays a tasq (nugget) value of 0.8042. This represents the nugget effect which is positive, indicating there is some unexplained variability at zero distance. Therefore, there is spatial variation at zero distance and measurement error to account for. The σ^2 (sill) value is 94012.62, which

is relatively high, suggesting that there is considerable spatial variability that the model can explain. The ϕ (range) is 194.62 representing the distance over which spatial correlation is significant. The practical range for this model is 923.29, representing the distance at which spatial correlation becomes practically zero. This value suggests that despite spatial correlation weakening beyond the range of 194 units, there are still long-range relationships that the model captures. The minimised weighted sum of squares is 10924.19 and this value is the result of minimising the weighted sum of squares during the fitting process. This value is only marginally larger than the Gaussian model. However both the Gaussian and Matérn models have considerably lower values than the powered exponential model, indicating better model fits.

We can also change the kappa value to 2 to explore whether the matérn model improves.

```
# Changing kappa value from 1.5 to 2.5
var_model_matern_2 <- variofit(sample_vario, cov.model = "matern",
                               fix.nugget = FALSE, max.dist = 7,
                               ini.cov.pars = c(60, 7), kappa = 5/2)
```

```
variofit: covariance model used is matern
variofit: weights used: npairs
variofit: minimisation function used: optim
```

```
# Printing the vitted variogram model parameters for the Matérn model
print(var_model_matern)
```

```
variofit: model parameters estimated by WLS (weighted least squares):
covariance model is: matern with fixed kappa = 1.5
parameter estimates:
     tausq    sigmasq        phi
    0.8042 94012.6218   194.6294
Practical Range with cor=0.05 for asymptotic range: 923.2953

variofit: minimised weighted sum of squares = 10924.19
```

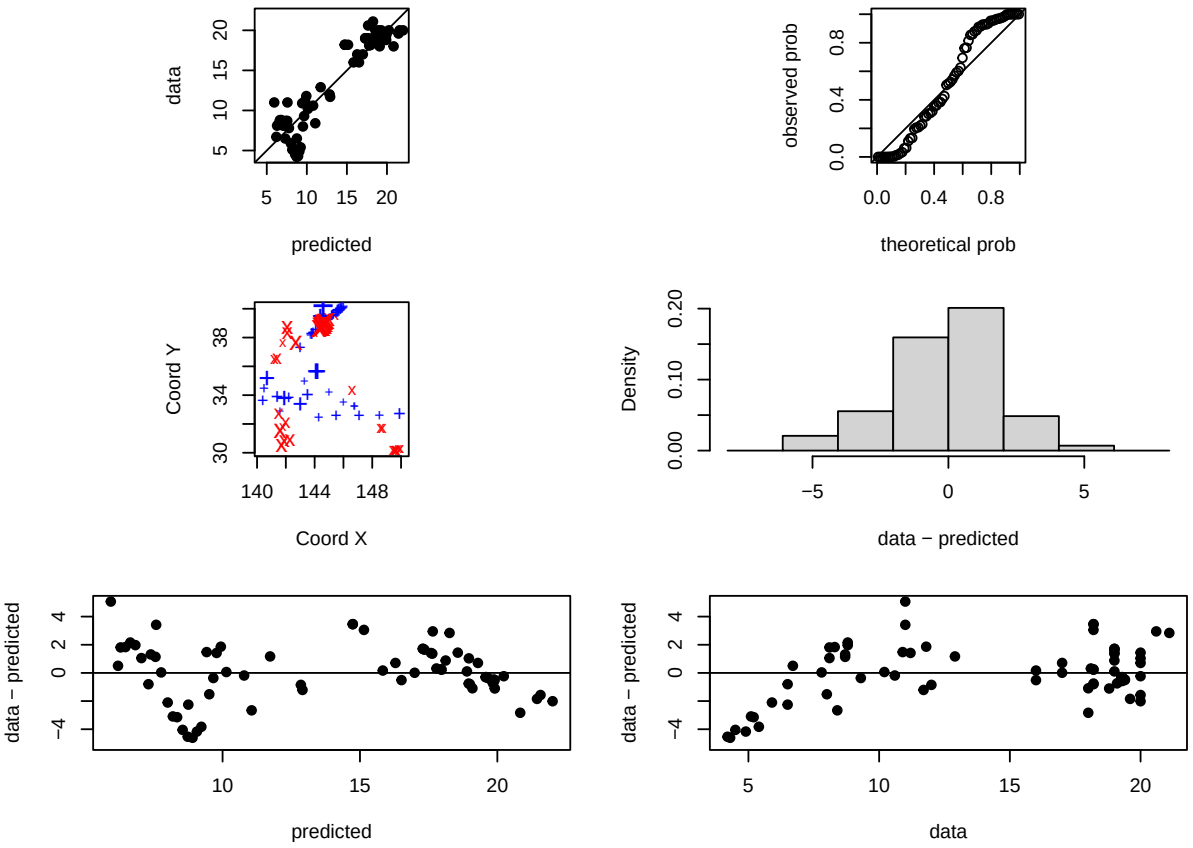
Changing the kappa value to 5/2 does not change any of the parameter estimates as shown.

Model Validation

Cross-Validation for the Gaussian model

```
# Leave-one-out cross-validation for the Gaussian model
xv_gaussian <- xvalid(geodata_aggregated, model = var_model_gaussian)
```

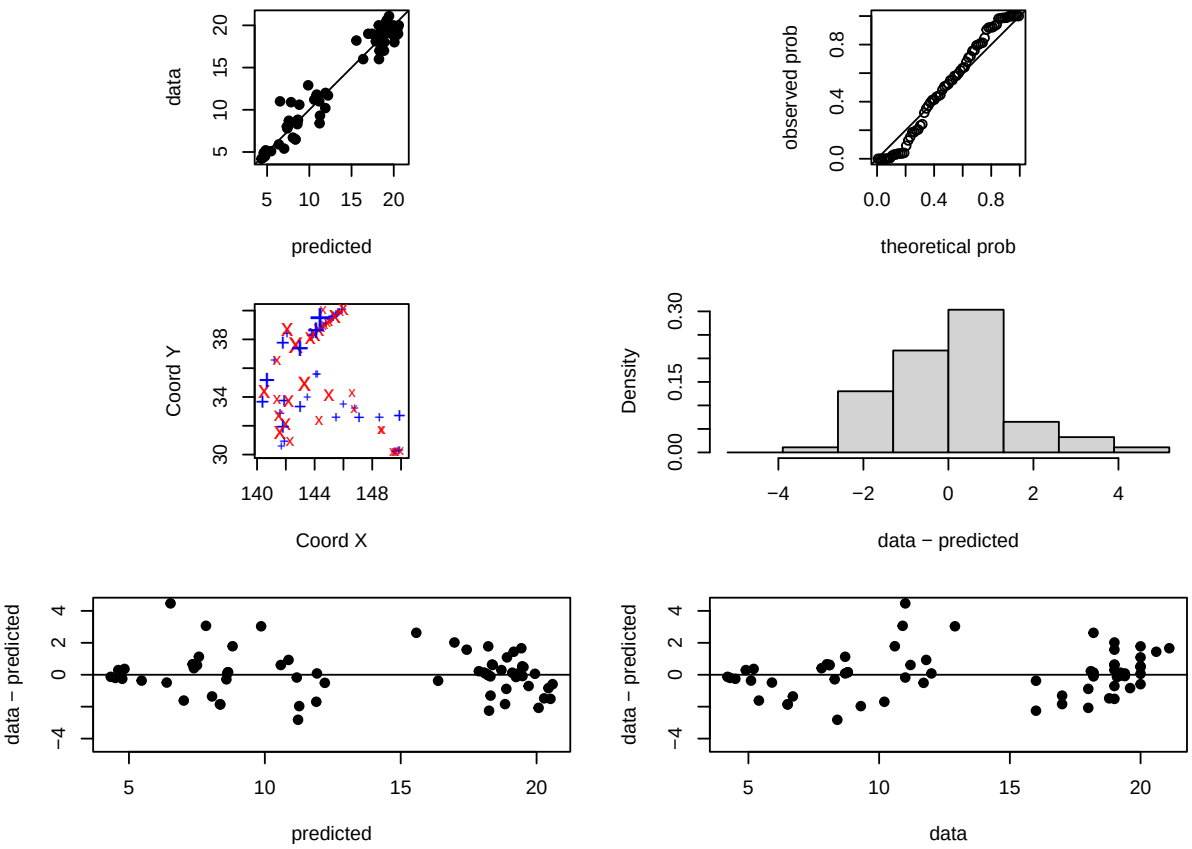
```
par(mfrow = c(3, 2), mar = c(4, 4, 2, 2))
plot(xv_gaussian, error = TRUE, std.error = FALSE, pch = 19)
```



Looking at the predicted vs observed plot in the top-left, we can see that the points align reasonably well along the diagonal, suggesting the model is making fairly accurate predictions as the residuals are reasonably normal. There are some small deviations from the diagonal. The QQ plot in the top-right suggests that the residuals are approximately normally distributed, which is a good sign for model fit. Both residuals vs spatial location and residuals vs data plots on the bottom row show mostly random scatter, indicating no systematic errors in the model's residuals.

Cross-Validation for the Powered Exponential model

```
# Leave-one-out cross-validation for the Powered Exponential model
xv_powered_expo <- xvalid(geodata_aggregated, model = var_model_powered_exponential)
par(mfrow = c(3, 2), mar = c(4, 4, 2, 2))
plot(xv_powered_expo, error = TRUE, std.error = FALSE, pch = 19)
```

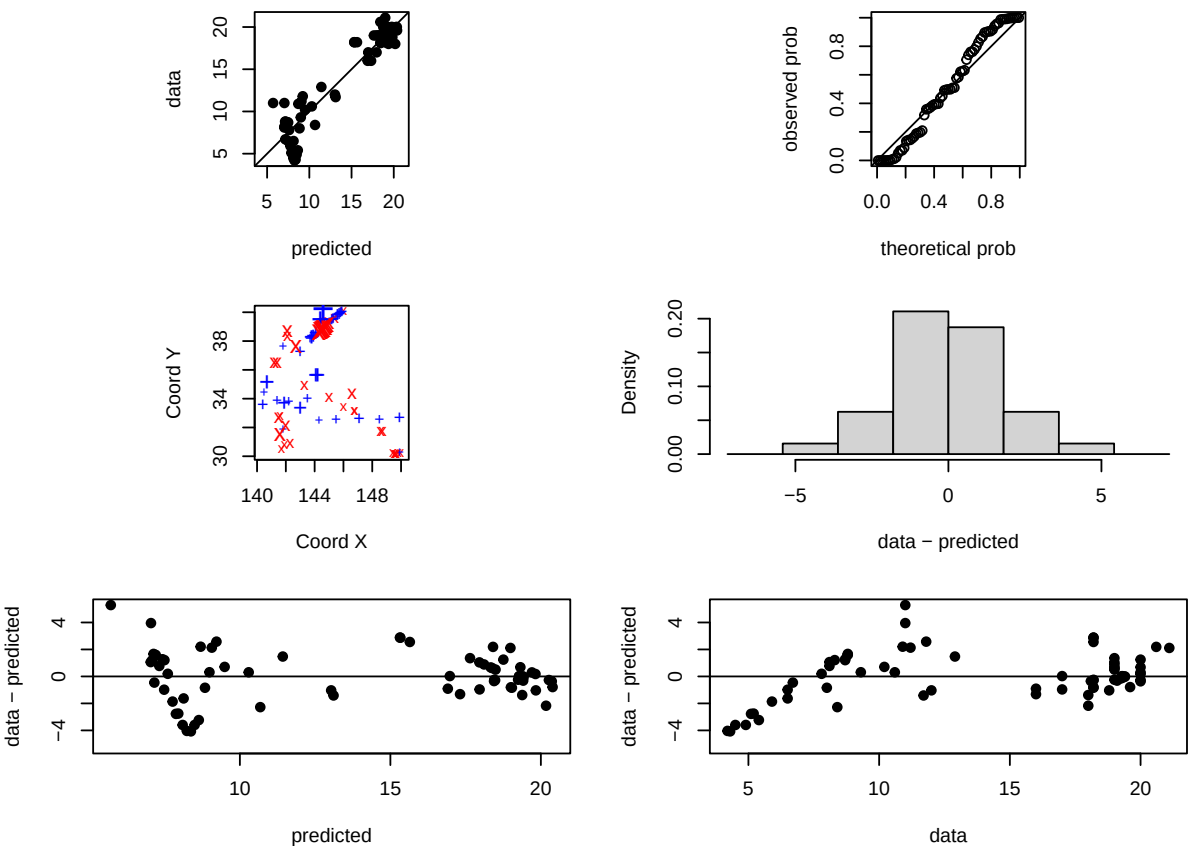


For the powered exponential model, the predicted vs observed plot in the top left performs slightly worse than the Gaussian model. While the points still align relatively well, there are larger deviations, particularly for higher predicted values. The residuals are normally distributed in the QQ plot but show some deviation from normality. The residuals vs data and residuals vs spatial location plots are very similar to the Gaussian model, showing mostly random scatter. However, the powered exponential model shows some clusters with some spatial patterns, indicating that the model may not fully account for spatial dependencies.

Cross-Validation for the Matérn Model

```
# Leave-one-out cross-validation for the Powered Exponential model
xv_Matérn <- xvalid(geodata_aggregated, model = var_model_matern)
```

```
par(mfrow = c(3, 2), mar = c(4, 4, 2, 2))
plot(xv_Matérn, error = TRUE, std.error = FALSE, pch = 19)
```



The predicted vs observed visual for the Matérn model fits well with points aligning with the diagonal well, suggesting accurate predictions. The QQ plot in the top-right shows little deviation away from the normal line, suggesting the residuals are approximately normally distributed. This is important because Gaussian process models, including the Matérn model, assume normally distributed errors. A close fit indicates the model assumptions are likely valid. The Matérn model has randomly scattered residuals for spatial location, suggesting the model fits well and captures the spatial dependencies properly. The residuals vs data graph shows that residuals are randomly distributed, with no noticeable structure, suggesting good model performance.

When picking a preferred model, the Matérn model is likely best. Despite not having the lowest minimised weighted sum of squares values, with a value of 16327.63, marginally higher than the Gaussian model, it provides more flexibility as a model than the exponential or Gaussian models. This is because the additional kappa parameter allows the models to better capture spatial dependencies, especially at varying scales. This model effectively captures the spatial structure of the data evidenced by predicted values that closely match the observed data and residuals that do not show any significant patterns.

Comparison Plot for Variograms

```
# Plotting the sample variogram with black points
plot(sample_vario, main = "Comparison of Models Against Sample Variogram",
     pch = 19, col = "black", cex = 0.6)

# Overlaying the fitted models
lines(var_model_matern, col = "cyan", lwd = 2, lty = 1)  # Matérn model
lines(var_model_gaussian, col = "red", lwd = 2, lty = 2)  # Gaussian model
lines(var_model_powered_exponential, col = "blueviolet", lwd = 2)  # Powered Exponential Mode


legend("bottomright",
       legend = c("Sample Variogram", "Matérn", "Gaussian", "Powered Exponential"),
       col = c("black", "cyan", "red", "blueviolet"),
       lwd = 2,
       cex = 0.8,
       lty = c(NA, 1, 2, 1),
       pch = c(19, NA, NA, NA))
```

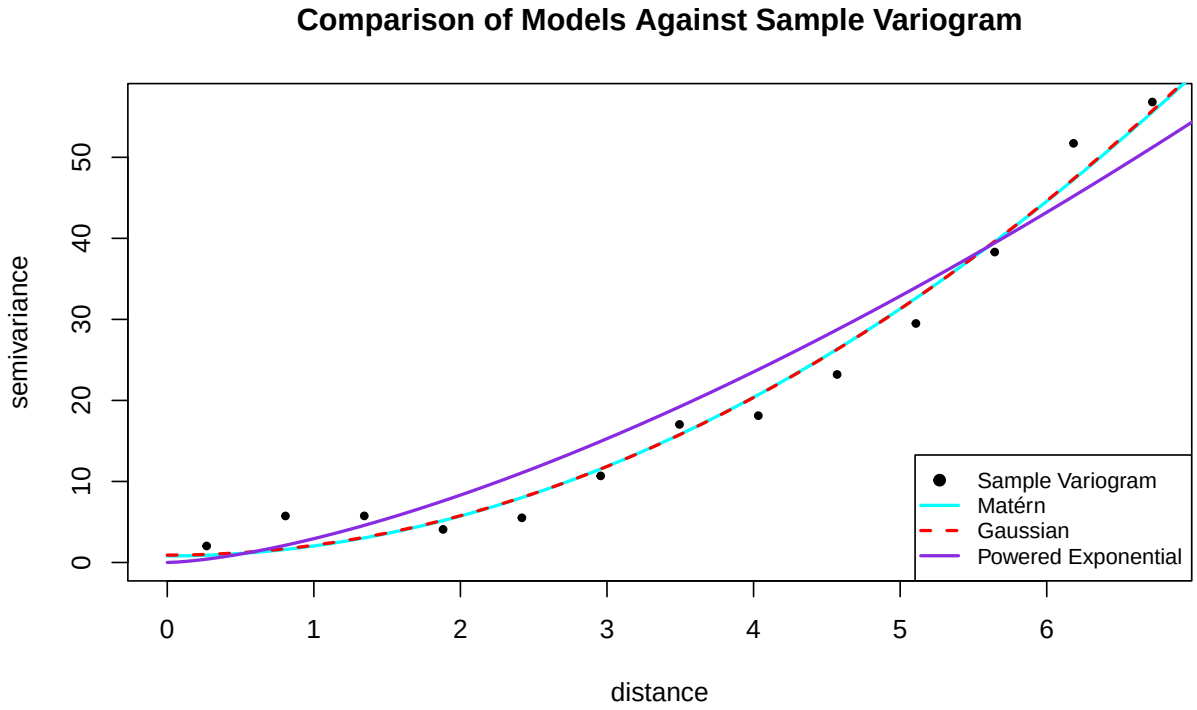


Figure 5: Compairson of Models against Sample Variogram

Comparing all of the models together, the Matérn model and Gaussian models provide the best overall fit, especially in distances between 2 and 4. Both these models out perform the Powered Exponential model as they follow the Sample Variogram more closely and therefore are a better fit for the data. The Gaussian model is extremely similar to the Matérn model because the Matérn model is behaving similarly to the Gaussian model with a kappa of 1.5. The key difference between the two models is that the Matérn model has more flexibility. Through adjusting the kappa, we can control the smoothness of the model, allowing for adjustments to how the spatial correlation decays over distance.

(d)

Gaussian Process Model using Maximum Likelihood

```
# Fitting a Gaussian Process model using Maximum Likelihood
Gaussian_ML <- likfit(geodata_aggregated,
                      cov.model = "exponential",
                      ini.cov.pars = c(60, 7),
                      fix.nugget = FALSE,
                      kappa = 1.99,
                      lik.method = "ML")
```

```
kappa not used for the exponential correlation function
-------------------------------------------------------------
likfit: likelihood maximisation using the function optim.
likfit: Use control() to pass additional
         arguments for the maximisation function.
        For further details see documentation for optim.
likfit: It is highly advisable to run this function several
        times with different initial values for the parameters.
likfit: WARNING: This step can be time demanding!
-------------------------------------------------------------
likfit: end of numerical maximisation.
```

```
# Printing the fitted model paramters
print(Gaussian_ML)
```

```
likfit: estimated model parameters:
    beta    tausq  sigmasq      phi
"15.149" " 0.000" "25.170" " 5.822"
Practical Range with cor=0.05 for asymptotic range: 17.44183

likfit: maximised log-likelihood = -131.5
```

The Gaussian Process Model using Maximum Likelihood provides beta parameters referring to the mean function. The Beta value from the output refers to the intercept and trend parameters. The beta value shown of 15.14 represents the baseline value of the SST data when other trend terms are zero. The tausq value is 0 which indicates there is no nugget in the model, suggesting that there data does not have small-scale variability or measurement error. The sigmasq (σ^2) value is 25.17 which refers to the sill of the model which represents the variance of the process. It is the overall variability in the SST values. The phi (ϕ) value is 5.822 which is the range parameter. This indicates the distance at which the spatial correlation decays. A value of 5.822 suggests that SST values are highly correlated within relatively short distances and the correlation drops off rapidly beyond that range. The practical range is shown here as 17.44 indicating the distance beyond which spatial correlation becomes negligible. The Maximum log-likelihood is -131.5 representing how well the model fits the data. The higher the log-likelihood, the better the fit.

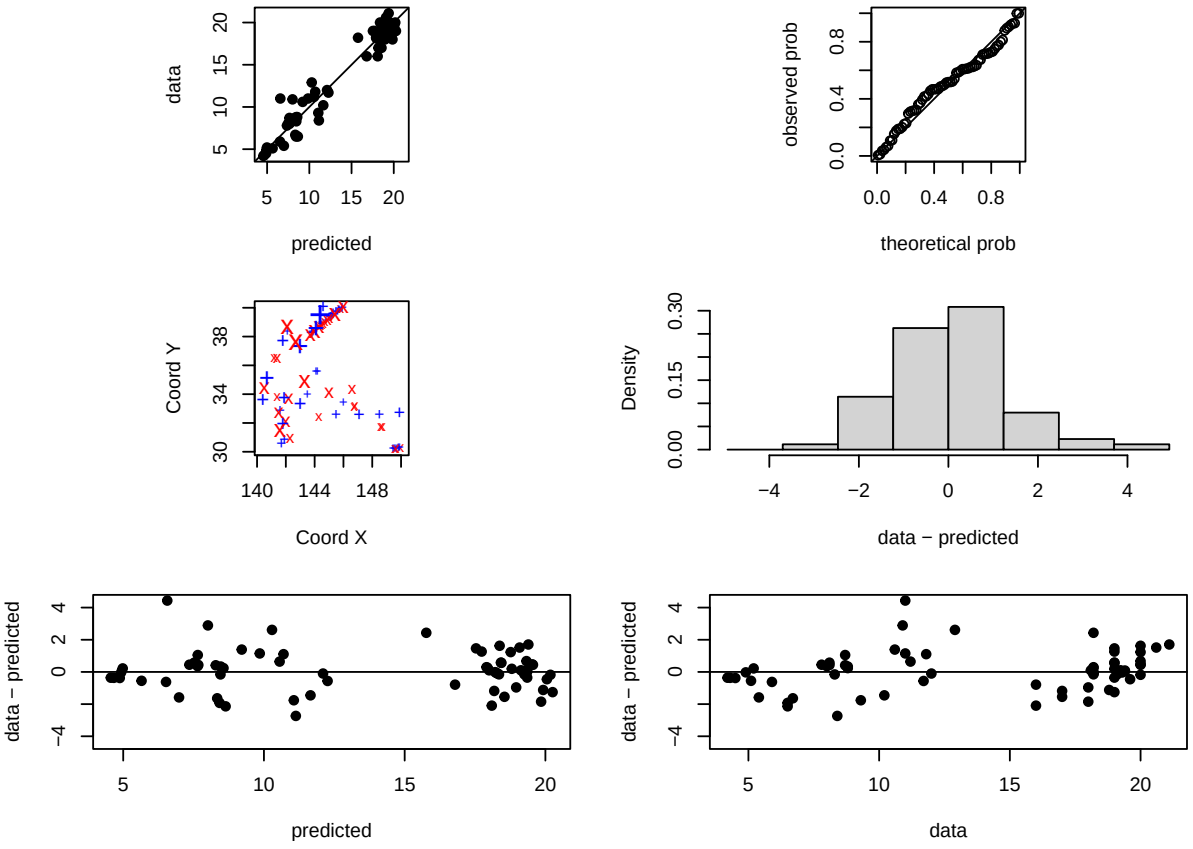
We see that our estimates for the variance σ^2 and correlation length ϕ (25.17, 5.82) are not too far away from the true values we used to generate the field (60, 5). Through maximising the likelihood, we have generated point estimates for the parameters.

Cross Validation for the Gaussian Process Model using Maximum Likelihood

```
# Leave-one-out cross-validation for the Gaussian Process Model using Maximum Likelihood
xv_Gaussian_ML <- xvalid(geodata_aggregated, model = Gaussian_ML)
```

```
xvalid: number of data locations       = 71
xvalid: number of validation locations = 71
xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
xvalid: end of cross-validation
```

```
par(mfrow = c(3, 2), mar = c(4, 4, 2, 2))
plot(xv_Gaussian_ML, error = TRUE, std.error = FALSE, pch = 19)
```



Based on the Cross Validation plots above of the Gaussian Process Model using Maximum Likelihood, the model has good predictive performance. The predicted vs actual data plot shows a strong relationship and indication of a good model fit. The QQ plot shows a comparison between the observed and

expected probabilities. The closer the points are to the diagonal line, the more normally distributed the residuals are. The spatial plot in the bottom left shows mostly randomly scattered residuals. The histogram of residuals in the bottom right shows that residuals are randomly distributed, with no noticeable structure, suggesting good model performance. This verifies that the errors are randomly distributed and follow a normal distribution. This model is likely a better fit than the other models because it shows strong predictive performance with minimal errors.

(e)

Bayesian Approach

We will fit the model using a Bayesian approach with discrete priors. This involves setting the priors for the correlation length (range phi) and nugget (tasuq), which are key parameters of the model. We will choose starting points for the Bayesian priors from our maximum likelihood estimates. For the nugget parameter (tasuq) we will choose a uniform prior assuming that the nugget is equally likely to take any value between 0 and 1. We use a discrete uniform distribution for this prior with 50 bins. For the range parameter prior for phi (ϕ), we will choose a uniform prior between 0 and 10. 5.822 was the phi value from the Gaussian Process Model using Maximum Likelihood. This prior is set as a discrete uniform distribution between 0 and 10 with 100 bins.

The model uses a Matérn covariance function with a kappa of 3/2. This provides a flexible representation of spatial correlation. This choice was made based on the assumption that the data exhibits smooth spatial correlation and this value provides a good balance between smooth and rough correlation.

```
set.seed(456)

# Setting grid for predictions
ex.grid <- as.matrix(expand.grid(seq(0,1,l=21), seq(0,1,l=21)))

# Fitting Bayesian Model
ex.bayes <- krige.bayes(
              geodata = geodata_aggregated,
              model = model.control(cov.model = "matern", kappa = 3/2),
              prior = prior.control(phi.discrete =seq(0, 10, l=101),
                                    phi.prior = "uniform",
                                    tausq.rel.discrete =seq(0, 1, l=51),
                                    tausq.rel.prior = "uniform"
                                   ))
```

```
# Printing Summary of the posterior samples
summary(ex.bayes$posterior$sample)
```

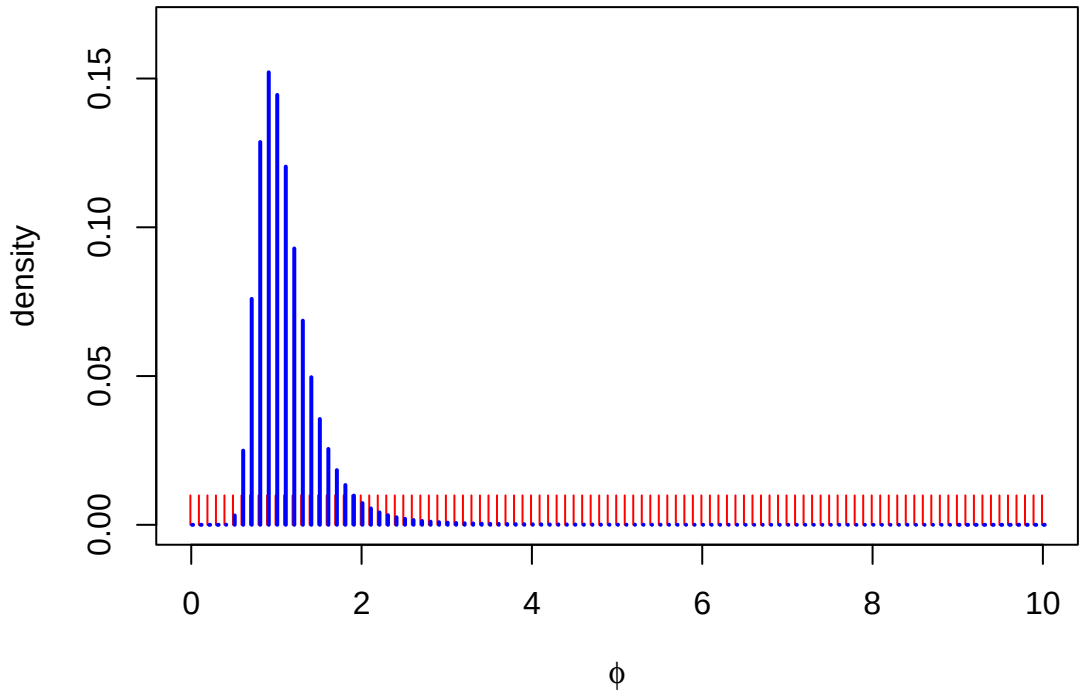
```
      beta            sigmasq             phi           tausq.rel
 Min.   : 1.032   Min.   :  7.298   Min.   :0.500   Min.   :0.0200
 1st Qu.:14.472   1st Qu.: 14.446   1st Qu.:0.900   1st Qu.:0.0200
 Median :15.554   Median : 18.746   Median :1.000   Median :0.0400
 Mean   :15.513   Mean   : 21.501   Mean   :1.153   Mean   :0.0436
 3rd Qu.:16.565   3rd Qu.: 25.364   3rd Qu.:1.300   3rd Qu.:0.0600
 Max.   :26.923   Max.   :133.810   Max.   :8.000   Max.   :0.3000
```

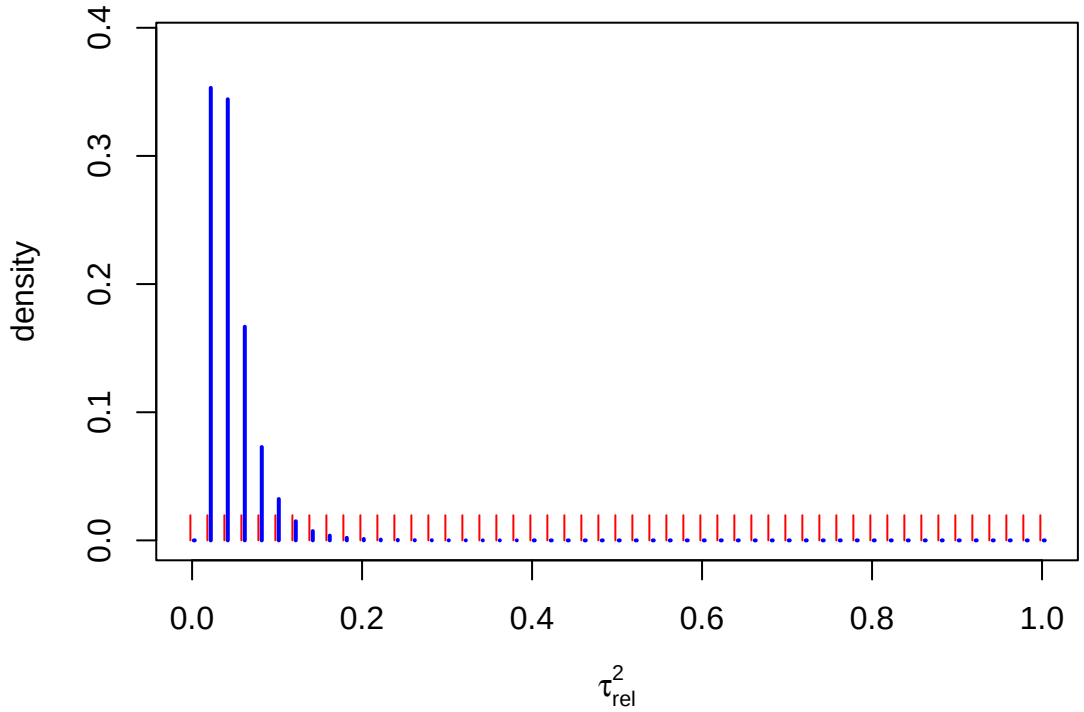
The summary output above from the Bayesian model contains 5151 posterior distributions for the parameters (phi and tausq), each of which have been sampled using Markov Chain Monte Carlo (MCMC).

The beta values representing the intercept in the spatial model is consistent with the maximum likelihood estimate, centering around 15.5 which is similar to the previous estimate of 15.1. The sigmasq (σ^2) value represents the overall variance of the process. This value is roughly around 21.501 as shown by the mean of the posterior above. The mean value from the Bayesian model is only slightly larger than the value from Gaussian Process Model using maximum likelihood, which estimated the parameter with a value of 25.1.

The phi (range) values in the posterior appear to be clustered around 1.15, which was lower than the initial estimate from the maximum likelihood estimate. This suggests that the maximum likelihood and Bayesian approaches have aligned differently for the range parameter. The Bayesian model provides credible intervals that are wider compared to the point estimates from MLE, reflecting the uncertainty in the model. This larger range for phi may indicate a more flexible model for the spatial correlation. The phi values range from 0.5 to 8.0. The tausq (nugget) value has posterior values around 0.04, indicating that the nugget is small in the data and there is very little unexplained variability beyond spatial correlation. The maximum likelihood estimate from showed the nugget was close to zero, and this is consistent with the Bayesian posterior where tausq is estimated to be around 0.04.

```r
# Plotting posterior for phi (range) and tausq (nugget)
par(mar = c(4, 4, 2, 2))
plot(ex.bayes, type = "h", phi.rel = FALSE, col = c("red", "blue"))
```





Looking at the prior and posterior distribution plots above for phi (range) and tausq (nugget), we can understand the uncertainty surrounding the parameters. The prior distribution for phi (range) is uniform

from 0 to 10 and this is shown by the red bars, displaying that all these values for phi before model fitting are considered equally likely. The posterior distribution shown for phi by the blue bars peak at lower values of phi, ranging from 0.0 to 2.0 with a cluster around 1.2. The data has reduced uncertainty and helped the model focus on a specific range for phi. The data has led the model to focus on values of phi closer to 0, compared to the maximum likelihood estimate of phi which was calculated at 5.822. This indicates a shorter range of spatial correlation.

The prior distribution for tasuq (nugget) is unform from 0 to 1 and this is shown by the red bars, indicating all these values for tausq before model fitting were considered equally likely. The posterior distribution shows a peak around 0.04, suggesting that after observing the data, the model became confident that the nugget (small-scale variability) is quite small. The model has effectively reduced the uncertainty about the nugget parameter compared to the prior after model fitting.

(f)

Predicting Precipitation at the 5 removed locations

We will predict the predict SST at the 5 removed locations using 3 different models that we have previously fitted. First we will predict using the Matérn variogram-based kriging model. Then we will predict SST using the Gaussian Process model using Maximum Likelihood. Finally we will predict SST using the Bayesian Approach model before comparing all of our predictions together.

```
# Defining the grid for prediction using the locations of the removed points
prediction_points_matrix <- as.matrix(chosen_points[, c("lon", "lat")])


# Making predictions using the variogram-based model (Matérn)
predictions_matern <- krige.conv(geodata = geodata_aggregated,
                            locations = prediction_points_matrix,
                            krige = krige.control(obj.model = var_model_matern))

Matern_predictions <- predictions_matern$predict

# Prediction with the Gaussian Process model using Maximum Likelihood
predictions_GaussianML <- krige.conv(geodata = geodata_aggregated,
                                locations = prediction_points_matrix,
                                krige = krige.control(obj.model = Gaussian_ML))

GaussianML_predictions <- predictions_GaussianML$predict

# Predictions with Bayesian Approach model
predictions_Bayesian <- krige.bayes(
                      geodata = geodata_aggregated,
```

```
                        locations = prediction_points_matrix,
                        model = model.control(cov.model = "matern", kappa = 3/2),
                        prior = prior.control(phi.discrete =seq(0, 10, l=101),
                        phi.prior = "uniform",
                        tausq.rel.discrete =seq(0, 1, l=51),
                        tausq.rel.prior = "uniform"))

Bayesian_predictions <- predictions_Bayesian$predict$mean
```

```
# Formating the predictions from all three models into a table
comparison_table <- data.frame(
  Location = paste(chosen_points$lon, chosen_points$lat, sep = ", "),
  True_SST = chosen_points$sst, # True original SST values
  Matern_Prediction = Matern_predictions,  # Matérn model predictions
  Gaussian_Prediction = GaussianML_predictions, # Gaussian model predictions
  Bayesian_Prediction = Bayesian_predictions   # Bayesian model predictions
)


kable(comparison_table,
caption = "Comparison of Predictions from Matérn, Gaussian, and Bayesian Models",
      col.names = c("Location (Longitude, Latitude)", "True SST", "Matérn Model",
                    "Gaussian Model", "Bayesian Model"))
```

Table 1: Comparison of Predictions from Matérn, Gaussian, and Bayesian Models

Location (Longitude, Latitude)	True SST	Matérn Model	Gaussian Model	Bayesian Model
144.2, 38.54	5.6	8.620291	8.058818	7.742083
144.3, 39.3	9.4	7.218415	10.688214	9.778947
143.7, 38.8	6.0	7.960053	9.585232	10.280123
143.6, 38	9.5	9.703833	10.236769	10.955739
144.1, 32.6	21.1	19.627264	20.036785	20.122546

Based on the comparison of True SST values and predictions from the Matérn, Gaussian, and Bayesian models for five locations, we observe that each model provides reasonable predictions with slight deviations from the true values. At location (144.2, 38.54), where the true SST is 5.6, all models overestimate the SST, with the Matérn model predicting 8.62, the Gaussian predicting 8.05, and the Bayesian model predicting 7.74. For location (144.3, 39.3), where the true SST is 9.4, the Matérn model predicts 7.22, the Gaussian model predicts 8.69, and the Bayesian model predicts 9.78. In this case the Matérn model underestimates the SST, with the Gaussian model being the most accurate. For location (143.6,

38), where the true SST is 9.5, the Matérn model predicts 9.70, the Gaussian model predicts 10.24, and the Bayesian model predicts 10.96. The Matérn model performs best here, with the other models overestimating SST again. At the 5th location (144.1, 32.6), where the true SST is 21.1, all models underestimate the SST, with the Matérn model predicting 19.63, the Gaussian model predicting 20.04, and the Bayesian model predicting 20.12. Here, the Bayesian model provides the closest prediction.

Overall, based on the comparison of predictions, the Matérn model performs the best. While all models offer reasonable predictions, the Matérn model consistently provides results closest to the true SST values across all locations.

```
# Calculating the absolute error for each model (Matérn, Gaussian, Bayesian)
comparison_table$Matern_Error <- abs(comparison_table$True_SST -
                                       comparison_table$Matern_Prediction)
comparison_table$Gaussian_Error <- abs(comparison_table$True_SST -
                                         comparison_table$Gaussian_Prediction)
comparison_table$Bayesian_Error <- abs(comparison_table$True_SST -
                                         comparison_table$Bayesian_Prediction)

# Calculate the Mean Absolute Error (MAE) for each model
mae_matern <- mean(comparison_table$Matern_Error)
mae_gaussian <- mean(comparison_table$Gaussian_Error)
mae_bayesian <- mean(comparison_table$Bayesian_Error)


# Creating the error table with True SST and predictions for each model
comparison_table_errors <- data.frame(
  Location = comparison_table$Location,
  Matern_Error = comparison_table$Matern_Error,  # Matérn Error
  Gaussian_Error = comparison_table$Gaussian_Error,  # Gaussian Error
  Bayesian_Error = comparison_table$Bayesian_Error  # Bayesian Error
)

# Addding MAE at the bottom of the table
comparison_table_errors <- rbind(comparison_table_errors,
                                 data.frame(
                                   Location = "Average Error",
                                   Matern_Error = mae_matern,
                                   Gaussian_Error = mae_gaussian,
                                   Bayesian_Error = mae_bayesian
                                 ))


kable(comparison_table_errors,
```

```
    caption = "Comparison of Absolute Errors for Each Model",
    col.names = c("Location (Longitude, Latitude)", "Matérn Error",
                  "Gaussian Error", "Bayesian Error"))
```

Table 2: Comparison of Absolute Errors for Each Model

Location (Longitude, Latitude)	Matérn Error	Gaussian Error	Bayesian Error
144.2, 38.54	3.0202910	2.4588183	2.1420835
144.3, 39.3	2.1815848	1.2882144	0.3789470
143.7, 38.8	1.9600532	3.5852324	4.2801229
143.6, 38	0.2038332	0.7367688	1.4557385
144.1, 32.6	1.4727360	1.0632151	0.9774536
Average Error	1.7676996	1.8264498	1.8468691

The Comparison of Absolute Errors for Each Model table above gives an overview of the predictive accuracy of the models for predicting SST in Kuroshio. The Matérn model consistently has the lowest errors across all locations, as indicated by its average error of 1.767, which is the smallest among the three models. For instance, at location (144.2, 38.54), the Matérn model has an error of 3.02, which is larger than the Gaussian (2.46) and Bayesian (2.14) errors, but when averaged across all locations, the Matérn model proves to be the most reliable. The Gaussian model, with an average error of 1.826, performs reasonably well, especially at location (143.6, 38), where its error is 0.74, but it slightly under performs compared to the Matérn model in most cases. For location (143.6, 38), the Matérn model had an error of only 0.2. The Bayesian model, with the highest average error of 1.84, shows larger deviations, especially at location (143.7, 38.8), where its error is 3.58. This table corroborates our previous analysis, confirming that the Matérn model is marginally the most accurate overall based on the absolute errors.

2. The Atlantic Overturning Circulation

(a)

```
# Loading the data
MOC_data <- load("MOC.Rdata")

# Checking the structure of the object
str(MOC_data)
```

```
 chr [1:2] ".Random.seed" "MOCmean"
```

```
# Viewing the first few entries
head(MOCmean)
```

```
 2017-10  2017-11  2017-12   2017-8   2017-9   2018-1
19.00617 20.26551 20.36827 18.53505 15.49253 24.06516
```

After loading the data, which considers measurements of the Atlantic meridional overturning circulation (AMOC) at 26 degrees north, we check the data's structure and first few entries. We can see that the data `MOCmean` is a named numeric vector, where each name represents a year-month period (e.g. "2017-10", "2017-10"), but the names are not structured chronologically. Before plotting to observe any trends or patterns, we must wrangle the data.

```
# Extracting the dates and values
dates <- names(MOCmean)
values <- as.numeric(MOCmean)

# Convert string dates to yearmon class
date_series <- as.yearmon(dates, format = "%Y-%m")

# Combine and sort
MOC_df <- data.frame(date = date_series, value = values)
MOC_df <- MOC_df[order(MOC_df$date), ]

# Checking the structure of the wrangled dataframe
str(MOC_df)
```

```
'data.frame':   67 obs. of  2 variables:
 $ date : 'yearmon' num  Aug 2017 Sep 2017 Oct 2017 Nov 2017 ...
 $ value: num  18.5 15.5 19 20.3 20.4 ...
```

```r
head(MOC_df)
```

```
      date    value
4 Aug 2017 18.53505
5 Sep 2017 15.49253
1 Oct 2017 19.00617
2 Nov 2017 20.26551
3 Dec 2017 20.36827
6 Jan 2018 24.06516
```

```r
# Plotting the data to observe trends and patterns
options(repr.plot.width = 10, repr.plot.height = 5)


tick_dates <- seq(from = min(MOC_df$date), to = max(MOC_df$date), by = 1)


plot(MOC_df$date, MOC_df$value, type = "l",
     col = "steelblue", lwd = 2,
     xlab = "Year", ylab = "AMOC Strength",
     main = "Atlantic Meridional Overturning Circulation at 26°N",
     xaxt = "n")

axis(1, at = tick_dates,
     labels = format(tick_dates, "%Y"),
     las = 1, cex.axis = 0.9, padj = 0.5)

# 12-month rolling mean
lines(MOC_df$date, rollmean(MOC_df$value, k = 12, fill = NA), col = "red", lwd = 3, lty = 2)

legend("topright", legend = c("Monthly AMOC Strength", "12-Month Rolling Average"),
       col = c("steelblue", "red"), lwd = 2.5, lty = c(1, 2), bty = "n", cex = 0.9)
```

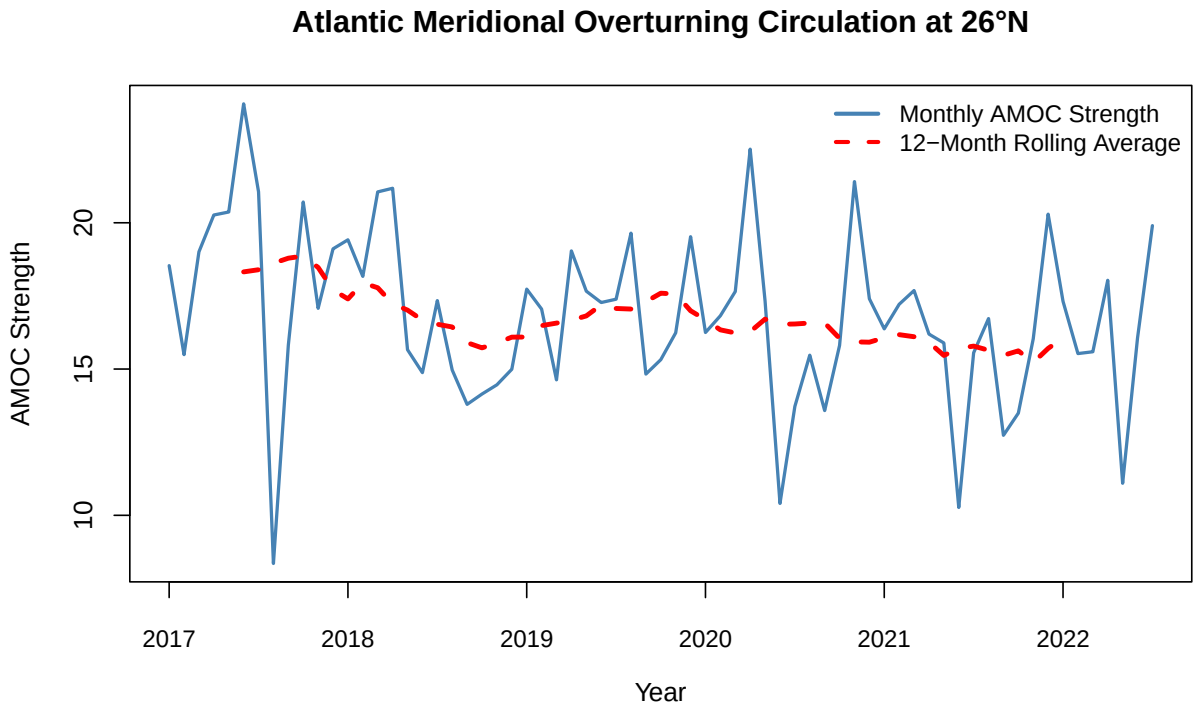


Figure 6: Monthly Strength of the AMOC at 26 degrees North from late 2017 to early 2023

The plot in figure-6 shows the monthly strength of the Atlantic Meridional Overturning Circulation (AMOC) at 26 degrees North from late 2017 to early 2023, along with a 12-month rolling average to emphasise broader trends.

There is considerable short-term variability, with frequent fluctuations in monthly values ranging roughly between 10 and 22 units. This suggests high-frequency noise or rapid short-term changes in ocean circulation dynamics. The rolling average smooths out this noise, revealing a gradual declining trend in AMOC strength from 2017 through to mid 2021. This could point to a long-term weakening of circulation, though it would require further statistical confirmation.

From 2021 onwards, the circulation shows signs of recovery or stabilisation, with rolling averages possibly beginning to increase. However, this recent uptick is subtle and may reflect short-term variation rather than a reversal of the long-term trend. The cyclical nature of the monthly AMOC average seems to be spiking heavily through 2021 and 2022, reflecting earlier dramatic changes in 2017, which was then followed by a more consistent seasonal pattern until more recent years.

Overall, the graph appears to lack strong seasonal cycles, suggesting that any seasonal component is weak or absent. However, there are consistent fluctuations but they do not seem to follow any particular seasonal pattern.

```r
acf(MOC_df$value, main = "ACF of AMOC Monthly Data")
```

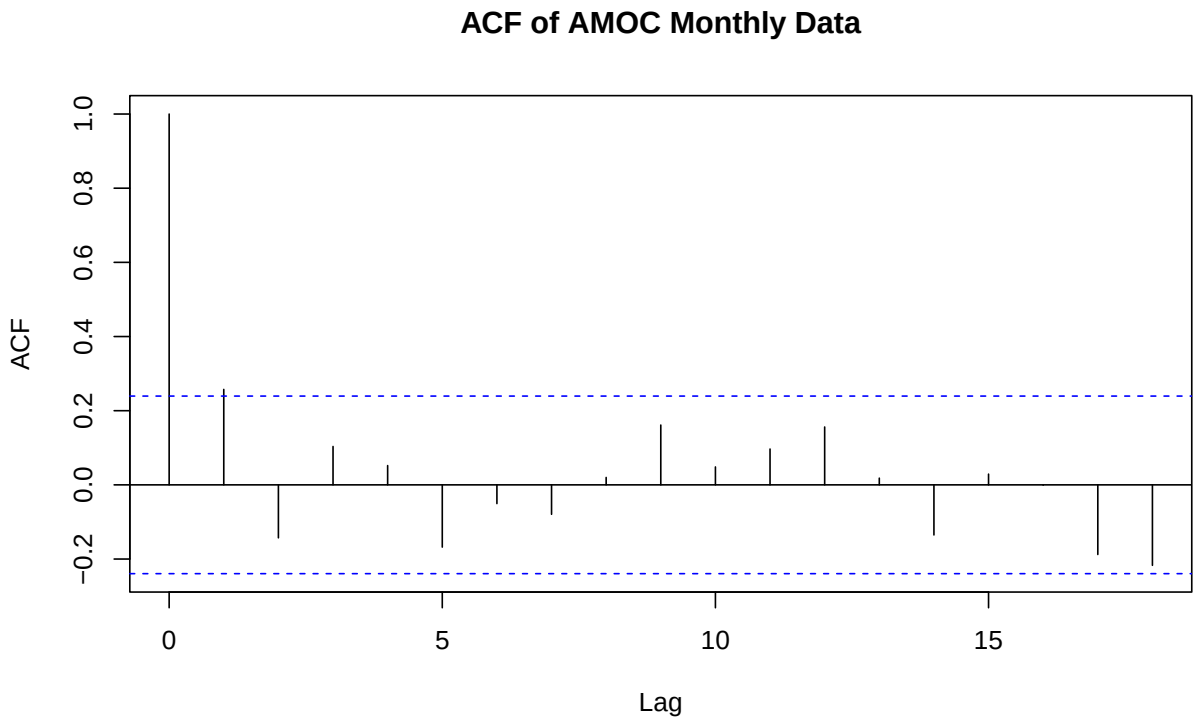


Figure 7: Autocorrelation Function Plot (ACF) to check seasonality

We can double check the seasonality using the plot above in figure-7 shows that beyond lag 1 all remaining lags are within the confidence bounds, including lag 12. Therefore, there is no clear evidence of strong seasonality in this data based on the ACF plot. If there was a yearly seasonal pattern, there would be a notable spike at lag 12, but this is not the case with this data. This supports the earlier assumption from the time series plot, that there may be some longer-term trends, but no dominant seasonal component.

(b)

Fitting ARMA and ARIMA models without seasonal components

```r
# Converting to ts
MOC_ts <- ts(MOC_df$value, start = c(2017, 8), frequency = 12)

# Get total length and cutt off the last 8 months
n <- length(MOC_ts)
```

```r
train_ts <- window(MOC_ts, end = time(MOC_ts)[n - 8])
test_ts <- window(MOC_ts, start = time(MOC_ts)[n - 7])

# Inspect the last few entries of training data
tail(train_ts)
```

```
          Jan      Feb      Mar      Apr      May      Jun
2022 10.26911 15.55530 16.72529 12.73420 13.49348 16.03536
```

```r
# Inspect the start of the test data (should be 8 months)
length(test_ts) # Last 8 months taken out
```

```
[1] 8
```

```r
start(test_ts) # First data point of the 8 months
```

```
[1] 2022    7
```

```r
end(test_ts) # Last data point of the 8 months
```

```
[1] 2023    2
```

Plotting ACF and PACF for the training set to guide ARMA/ARIMA model selection:

```r
par(mfrow = c(1, 2))
acf(train_ts, main = "ACF of Training Data")
pacf(train_ts, main = "PACF of Training Data")
```

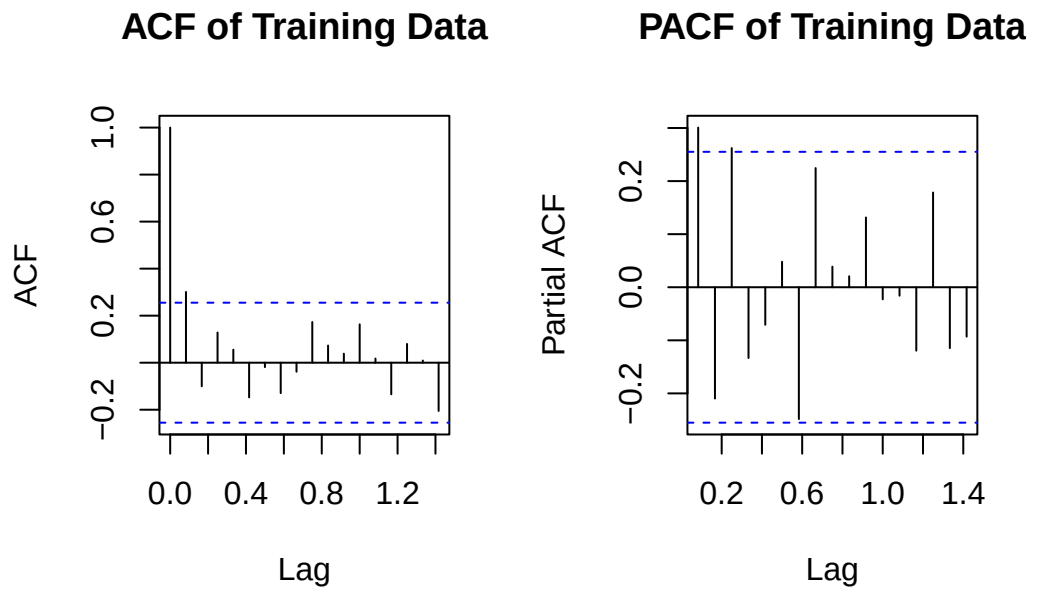


Figure 8: ACF and PACF of Training Data

The ACF plot suggests not much long-range autocorrelation, and possibly no need to difference the data implying that it may already be stationary. The PACF (Partial Autocorrelation Function) plot shows a strong spike at lag 1 and lag 3. The other lags are mostly insignificant. This plot shows the classic pattern of an AR(1) process.

Differencing the Data to Guide Model Selection

```
par(mfrow = c(1, 3))
# 1. Original series
plot(train_ts, main = "Original AMOC Series",
     ylab = "AMOC Strength",
     xlab = "Time",
     col = "black")

# 2. First difference
diff_train_ts <- diff(train_ts)
plot(diff_train_ts, main = "First Difference",
     ylab = "Differenced AMOC",
     xlab = "Time",
     col = "black")

# 3. Second difference
```

```r
diff2_train_ts <- diff(diff_train_ts)
plot(diff2_train_ts, main = "Second Difference",
     ylab = "2nd Differenced AMOC",
     xlab = "Time",
     col = "black")
```

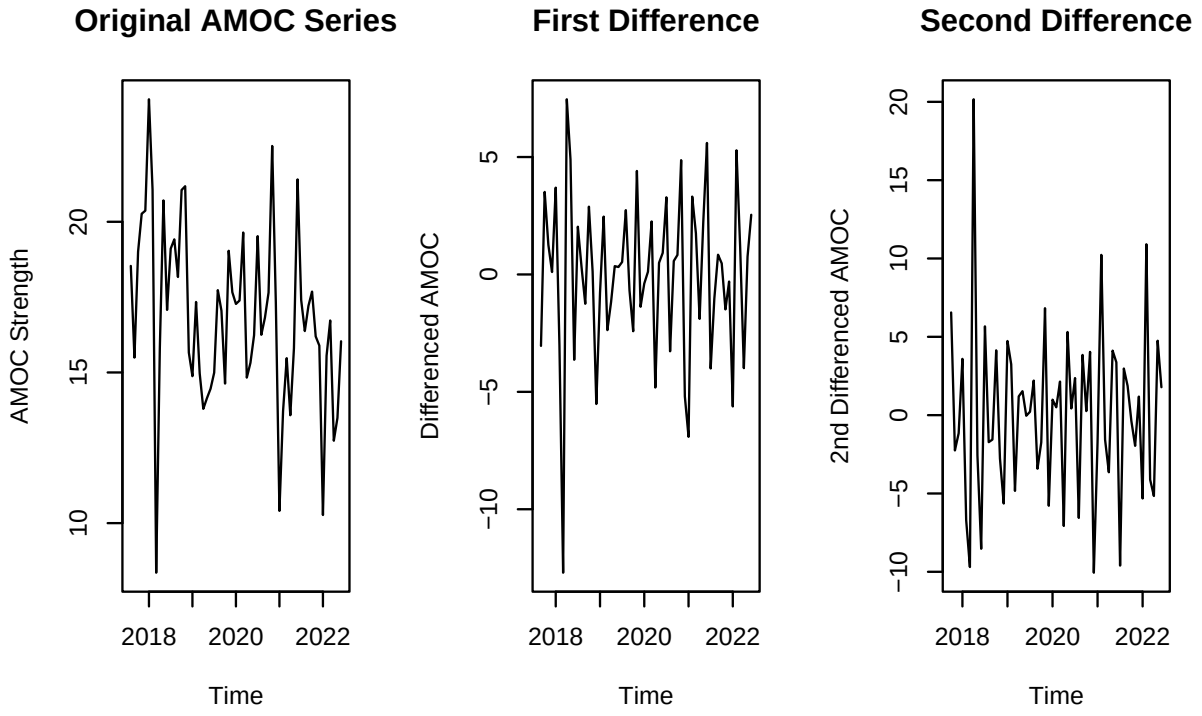


Figure 9: Differencing the AMOC Data

The original AMOC series as we have already seen shows clear non-stationary behaviour, with visible trends and changes in variability over time. This suggests that a stationary model like ARMA may not be appropriate without differencing.

After applying the first difference, the series appears approximately stationary with fluctuations centred around zero with no obvious long-term trend, and the variance looks roughly constant over time. This suggests that first differencing is sufficient to achieve stationarity.

The second difference, while still stationary, shows excessive noise and larger jumps, indicating that the second differencing is unnecessary and would over-difference the data, potentially removing important structure.

Therefore, based on these observations, we conclude that the data should be modelled with d = 1, and models such as ARIMA(p, 1, q) are appropriate. We will proceed to fit ARIMA models with d = 1, alongside other models for comparison.

AR(1)

```
# Fitting AR(1)
fit_ar1 <- arima(train_ts, order = c(1, 0, 0))
fit_ar1
```

```
Call:
arima(x = train_ts, order = c(1, 0, 0))

Coefficients:
         ar1  intercept
      0.2979    16.8329
s.e.  0.1233     0.5136

sigma^2 estimated as 7.781:  log likelihood = -144.29,  aic = 294.58
```

The AR(1) coefficient of 0.2979, indicates a moderate positive dependence on the previous month's AMOC value. This suggests that the AMOC does exhibit persistence, but its not very strong and therefore shocks to the system fade relatively quickly. The intercept at around 16.83 represents the baseline level the process reverts to when not influenced by recent values. The standard error of 0.123 shows the coefficient is around 2.4 standard errors from 0, which is generally statistically significant at the 5% level. The AIC value of 294.58 shows how best the model fits the data.

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

# Residual ACF
acf(residuals(fit_ar1), main = "ACF of Residuals")

# Residual PACF
pacf(residuals(fit_ar1), main = "PACF of Residuals")
```

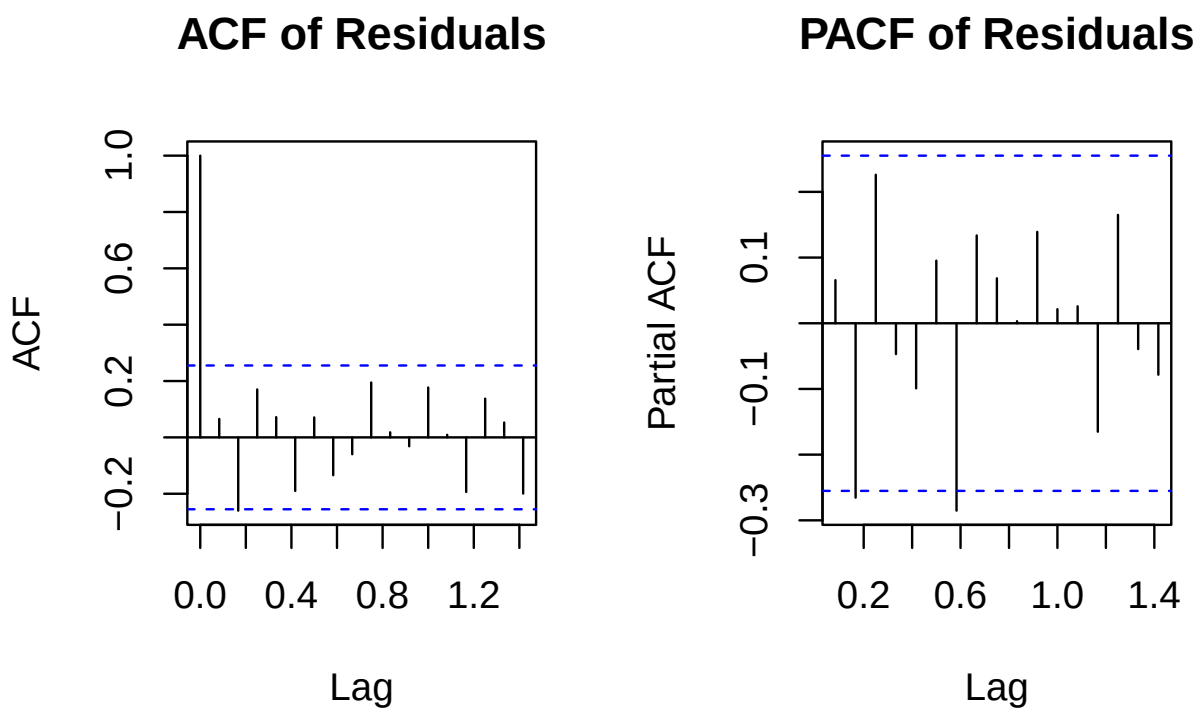


Figure 10: ACF and PACF of residuals for AR(1)

Looking at the residuals for AR(1), we can observe from the standardised residuals plot that the values randomly fluctuate around 0, which is good. There are a few large spikes, including some above +/- 5, which may suggest outliers or occasional volatility. There is no obvious time trend or persistent pattern, supporting mean stationarity. The ACF of residuals here show most autocorrelations despite the spike at lag 1 are within the blue significance bounds, meaning no strong remaining correlation. The PACF of residuals show several partial autocorrelations above the confidence bounds, including lag 1 and others up to lag 6. This suggests the residual structure is not entirely white noise. There may be potential under fitting if the model is too simple, as the residuals still contain predictable structure.

MA (1)

```
# Fitting MA(1)
fit_ma1 <- arima(train_ts, order = c(0, 1, 1))
fit_ma1
```

```
Call:
arima(x = train_ts, order = c(0, 1, 1))
```

```
Coefficients:
          ma1
      -0.8980
s.e.   0.0704

sigma^2 estimated as 8.63:  log likelihood = -145.62,  aic = 295.24
```

After fitting the MA(1) model, the moving average coefficient was -0.8980 indicating a strong negative dependence on the previous period's shock. This suggests that fluctuations in the differenced series are largely corrected by the model from one period to the next. The intercept was not estimated separately due to the differencing step.

Initially, both the AR(1) model and MA(1) model were fitted to the original AMOC series to provide a baseline assessment of autocorrelation structure. However, subsequent analysis revealed that the original series was non-stationary, requiring first differencing to achieve stationarity. This is because the non differenced original series has a slight negative trend. We can fit the AR(1) and MA(1) models to the differenced data but it would have the same outcome as fitting an ARIMA model. Consequently, fitting ARMA models such as AR(1) or MA(1) directly to the undifferenced data is not appropriate. Therefore, we proceed to focus on fitting ARIMA models with d = 1, specifically ARIMA(1, 1, 1), ARIMA (1, 1, 0) and ARIMA(0, 1, 1) for appropriate model selection and forecasting.

ARIMA (1, 1, 1)

```r
# ARIMA(1,1,1) - include this even if differencing isn't clearly needed
fit_arima111 <- arima(train_ts, order = c(1, 1, 1))
fit_arima111
```

```
Call:
arima(x = train_ts, order = c(1, 1, 1))

Coefficients:
         ar1      ma1
      0.2720  -0.9417
s.e.  0.1373   0.0552

sigma^2 estimated as 8.07:  log likelihood = -143.69,  aic = 293.37
```

The ARIMA model above was differenced once to achieve stationarity, with one autoregressive term and one moving average term. The AR(1) coefficient of 0.2720 suggests a modest positive dependence on the most recent lagged value of the differenced series. The MA(1) coefficient of -0.9417 shows is

a very strong negative correlation with the previous period's random shock (error term). Both coefficients are statistically significant. The model achieved an AIC value of 293.37, indicating the trade-off between fit and simplicity.

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

# 2. Residual ACF
acf(residuals(fit_arima111), main = "ACF of Residuals")

# 3. Residual PACF
pacf(residuals(fit_arima111), main = "PACF of Residuals")
```

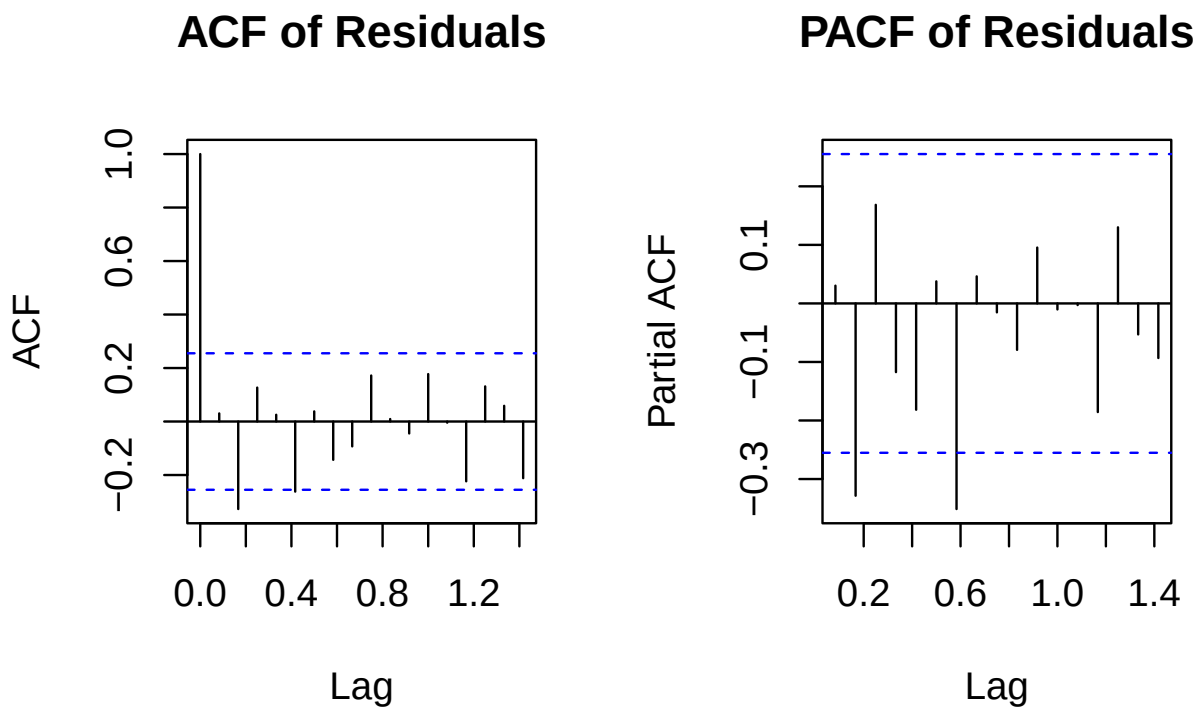


Figure 11: ACF and PACF residuals for ARIMA (1, 1, 1)

Looking at the ACF of residuals for the ARIMA(1, 1, 1) model, most autocorrelations are within the blue significant bounds, indicating that the model has successfully captured most of the temporal dependence in the data. The PACF of residuals is similar to the ACF with only two lags at lag 2 and lag 7 that are slightly outside the confidence interval. Therefore, the pattern is consistent with random noise, and there is no evidence of systematic autocorrelation.

ARIMA (1, 1, 0)

```
# Fitting ARIMA(1,1,0)
fit_arima110 <- arima(train_ts, order = c(1, 1, 0))
fit_arima110
```

```
Call:
arima(x = train_ts, order = c(1, 1, 0))

Coefficients:
          ar1
      -0.2108
s.e.   0.1286

sigma^2 estimated as 11.57:  log likelihood = -153.33,  aic = 310.66
```

Although an ARIMA (1, 1, 0) model was fitted to the differenced AMOC data, its AIC value of 310.66 was considerably higher than that of the ARIMA (1, 1, 1) model (AIC = 293.37). Since lower AIC values indicate a better trade-off between model complexity and goodness of fit, ARIMA (1, 1, 1) is the better choice of model in comparison.

ARIMA (0, 1, 1)

```
# Fitting ARIMA(0,1,1)
fit_arima011 <- arima(train_ts, order = c(0, 1, 1))
fit_arima011
```

```
Call:
arima(x = train_ts, order = c(0, 1, 1))

Coefficients:
          ma1
      -0.8980
s.e.   0.0704

sigma^2 estimated as 8.63:  log likelihood = -145.62,  aic = 295.24
```

The ARIMA (0, 1, 1) model provides an MA(1) coefficient of -0.8980 which is strongly negative. It suggests that current shocks correct almost completely for past shocks. The AIC of 295.24 is slightly higher than the ARIMA(1, 1, 1) but considerably lower than ARIMA (1, 1, 0), indicating a much better model fit. While ARIMA(0, 1, 1) improved over simpler structures, the ARIMA(1, 1, 1) model remains the preferred choice based on model selection criteria.

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

# ACF of residuals
acf(residuals(fit_arima011),
    main = "ACF of Residuals")

# PACF of residuals
pacf(residuals(fit_arima011),
     main = "PACF of Residuals")
```

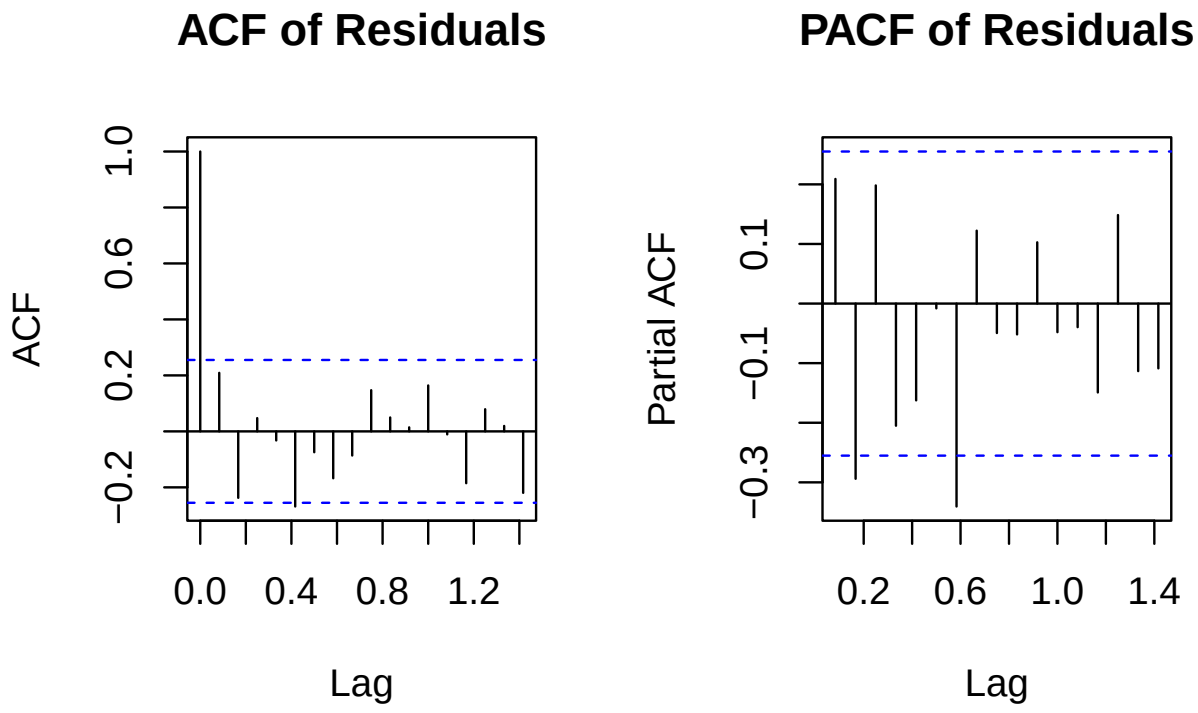


Figure 12: ACF and PACF residuals for ARIMA(0, 1, 1)

The ACF and PACF plots of the ARIMA(0,1,1) residuals show no significant autocorrelations, with nearly all lags falling within the 95% confidence bounds. This suggests that the residuals behave approximately as white noise, indicating that the ARIMA(0,1,1) model adequately captured the serial

dependence in the differenced AMOC data. However, as the model's AIC value is higher than that of ARIMA(1,1,1), it is not selected as the final model.

Model Selection Summary

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

# Setting lag range
lags <- 1:20

# AR(1)
lb_ar1 <- sapply(lags, function(l) Box.test(residuals(fit_ar1), lag = l,
                                            type = "Ljung-Box")$p.value)
plot(lags, lb_ar1, type = "b", ylim = c(0,1), col = "steelblue",
     main = "AR(1)", xlab = "Lag", ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)

# MA(1)
lb_ma1 <- sapply(lags, function(l) Box.test(residuals(fit_ma1), lag = l,
                                            type = "Ljung-Box")$p.value)
plot(lags, lb_ma1, type = "b", ylim = c(0,1), col = "steelblue",
     main = "MA(1)", xlab = "Lag", ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)

# ARIMA(0,1,1)
lb_arima011 <- sapply(lags, function(l) Box.test(residuals(fit_arima011),
                                                 lag = l,
                                                 type = "Ljung-Box")$p.value)
plot(lags, lb_arima011, type = "b", ylim = c(0,1), col = "steelblue",
     main = "ARIMA(0,1,1)", xlab = "Lag", ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)

# ARIMA(1,1,1)
lb_arima111 <- sapply(lags, function(l) Box.test(residuals(fit_arima111),
                                                 lag = l,
                                                 type = "Ljung-Box")$p.value)
plot(lags, lb_arima111, type = "b", ylim = c(0,1), col = "steelblue",
     main = "ARIMA(1,1,1)", xlab = "Lag", ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)
```

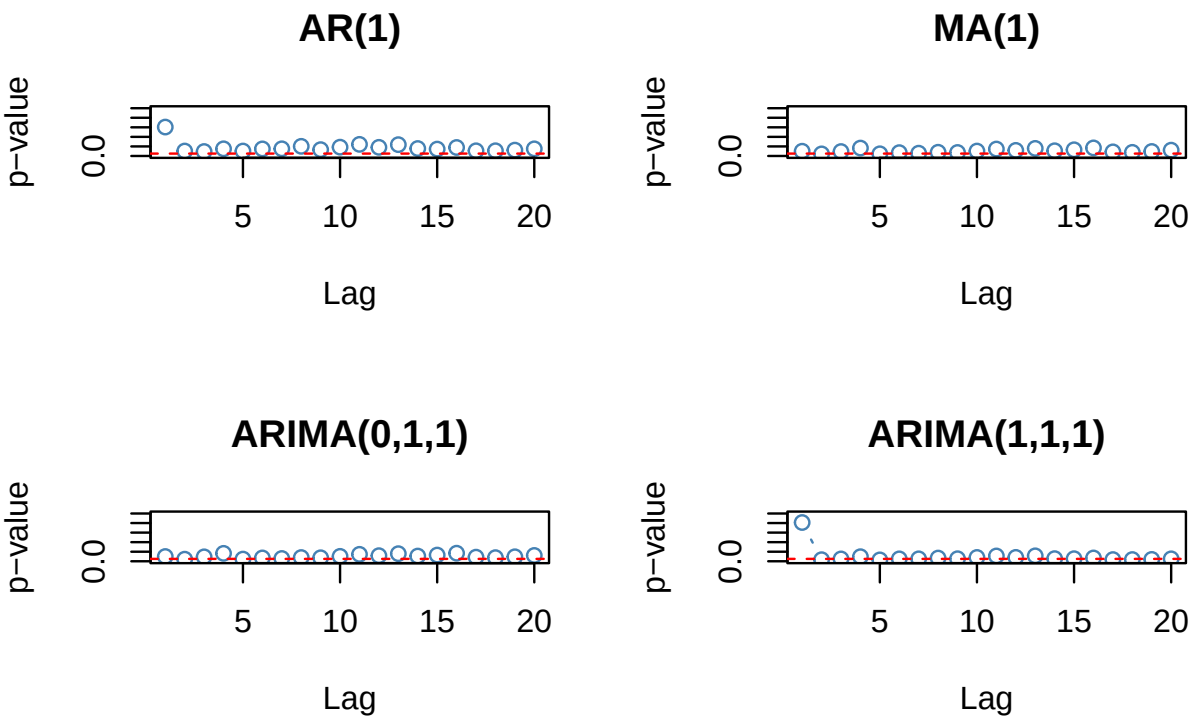


Figure 13: Ljung-box p-value plots

The Ljung-box plots above show p-values fort the Ljung-box test at lags 1-20. The red line is at p=0.05 (5% significance threshold). If the points are above the red line then the residuals are uncorrelated and if the residuals are below the red line then significant autocorrelation remains. None of the models achieve perfect white noise residuals but despite that, most points for all models fall above 0.05.

```
# Creating a data frame with model names and AICs
model_aic_summary <- data.frame(
  Model = c("AR(1)", "MA(1)", "ARIMA(1,1,0)", "ARIMA(0,1,1)", "ARIMA(1,1,1)"),
  AIC = c(AIC(fit_ar1), AIC(fit_ma1), AIC(fit_arima110), AIC(fit_arima011),
          AIC(fit_arima111))
)


kable(model_aic_summary, caption = "Model Comparison Based on AIC Values")
```

Table 3: Model Comparison Based on AIC Values

Model	AIC
AR(1)	294.5802
MA(1)	295.2403
ARIMA(1,1,0)	310.6605
ARIMA(0,1,1)	295.2403
ARIMA(1,1,1)	293.3743

After identifying that the original AMOC series was non-stationary, first differencing was applied to achieve stationarity. Consequently, models incorporating differencing (d = 1) were prioritised. Among the models fitted, the ARIMA(1, 1, 1) model achieved the lowest AIC value (293.37), indicating the best overall fit to the differenced series. Residual diagnostics for ARIMA(1, 1, 1) showed no significant autocorrelation and residuals behaving approximately as white noise, confirming the model's adequacy. In contrast, the AR(1) and MA(1) models, which were fitted to undifferenced data, had higher AIC values and showed evidence of residual autocorrelation. Therefore, ARIMA(1, 1, 1) was selected as the final model for forecasting the AMOC time series.

(c)

Averaging the Data into Quarterly Means and excluding the last 2 quarters

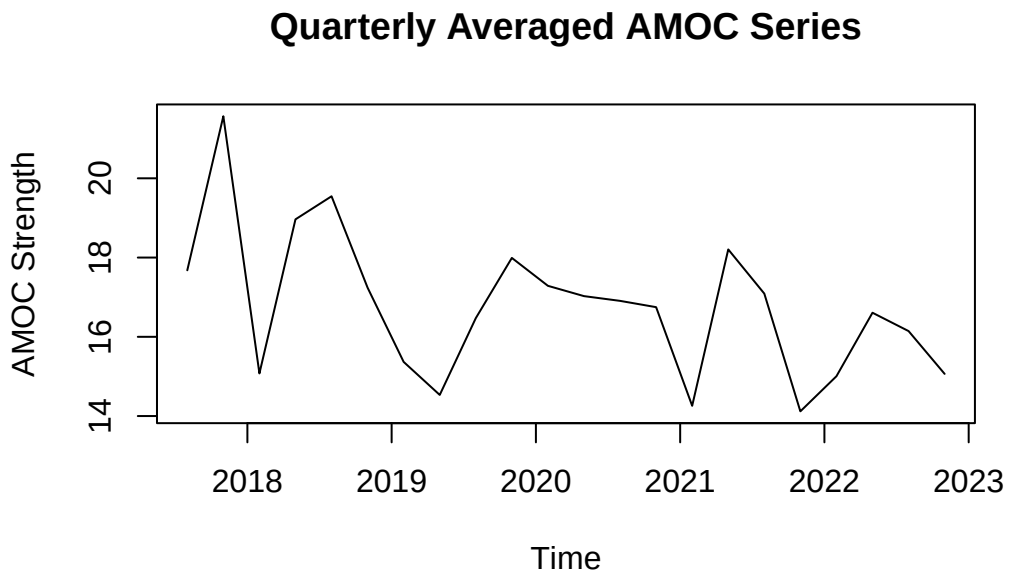
```
# Aggregate to quarterly means
MOC_quarterly <- aggregate(MOC_ts, nfrequency = 4, FUN = mean)

length(MOC_ts)    # How many months in original series
```

```
[1] 67
```

```
length(MOC_quarterly)   # How many quarters now
```

```
[1] 22
```

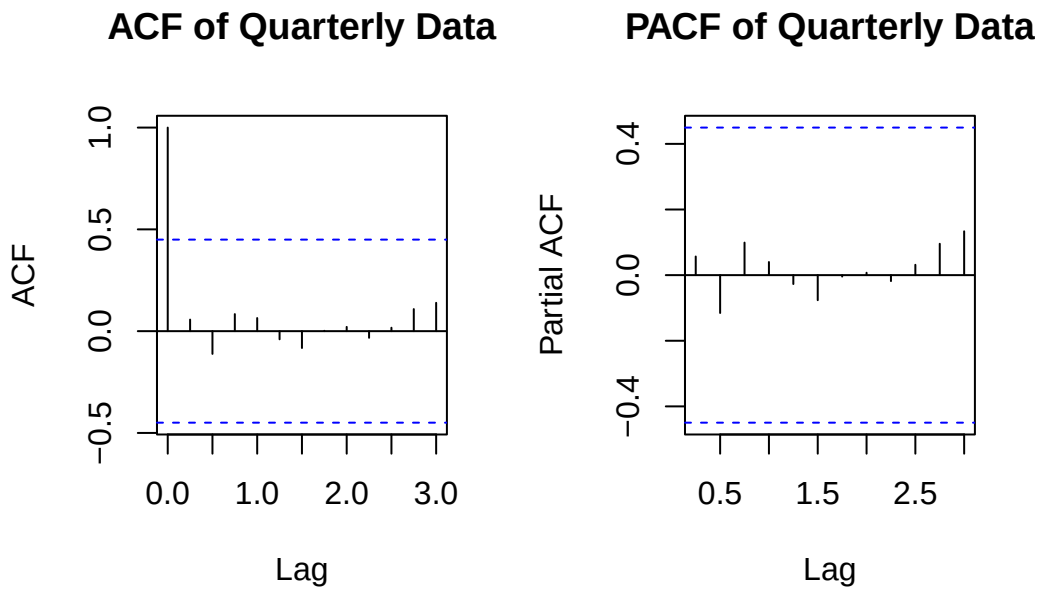
```
# Plot to check
plot(MOC_quarterly, main = "Quarterly Averaged AMOC Series",
     ylab = "AMOC Strength", xlab = "Time")
```



The original monthly AMOC time series was successfully averaged into quarterly means, reducing the dataset to 22 quarterly observations from mid-2017 to late 2022. We will now remove the last 2 quarters from the quarterly data to create a training set for modelling. As we can see, there is still a slight negative trend with the original data, so we will difference the data during modelling.

```
nq <- length(MOC_quarterly)
# Train up to 2022 Q2, test = Q3, Q4, Q1
train_quarterly <- window(MOC_quarterly, end = time(MOC_quarterly)[nq - 3])
test_quarterly  <- window(MOC_quarterly, start = time(MOC_quarterly)[nq - 2])
```

```
# Plotting ACF and PACF of the Quarterly Data
par(mfrow = c(1, 2))
acf(train_quarterly, main = "ACF of Quarterly Data")
pacf(train_quarterly, main = "PACF of Quarterly Data")
```



Looking at the ACF of the quarterly data, lag 1 has a very strong positive autocorrelation of around 0.8 to 0.9. Subsequent lags drop sharply towards zero with no obvious seasonal spikes. The PACF of the quarterly data displays small lags all within the significance bounds. This suggests an AR(1) could be applicable. No clear strong seasonality is observed, indicating that the SARIMA models may not be necessary. Based on these patterns, an ARMA(1, 0) model may be initially preferred. However, to ensure a comprehensive model comparison, ARIMA and SARIMA models will also be fitted and evaluated based on AIC and residual diagnosis.

ARIMA(0, 1, 1)

```
# Fitting ARIMA(0, 1, 1) model (with differencing)
fit_arima011_q <- arima(train_quarterly, order = c(0, 1, 1))
fit_arima011_q
```

```
Call:
arima(x = train_quarterly, order = c(0, 1, 1))

Coefficients:
          ma1
      -0.7936
s.e.   0.1586

sigma^2 estimated as 3.877:  log likelihood = -38.23,  aic = 80.46
```

The ARIMA (0, 1, 1) model fitted to the differenced quarterly AMOC data produced a moving average coefficient of -0.7936, indicating strong negative dependence on the previous shock. The AIC value of 80.46 refers to how well the model fits the data.

```
par(mfrow = c(1, 3))

# Standardized residuals
ts.plot(residuals(fit_arima011_q),
        main = "Standardized Residuals", col = "darkgray",
        ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)

# ACF
acf(residuals(fit_arima011_q), main = "ACF")

# PACF
pacf(residuals(fit_arima011_q), main = "PACF")
```

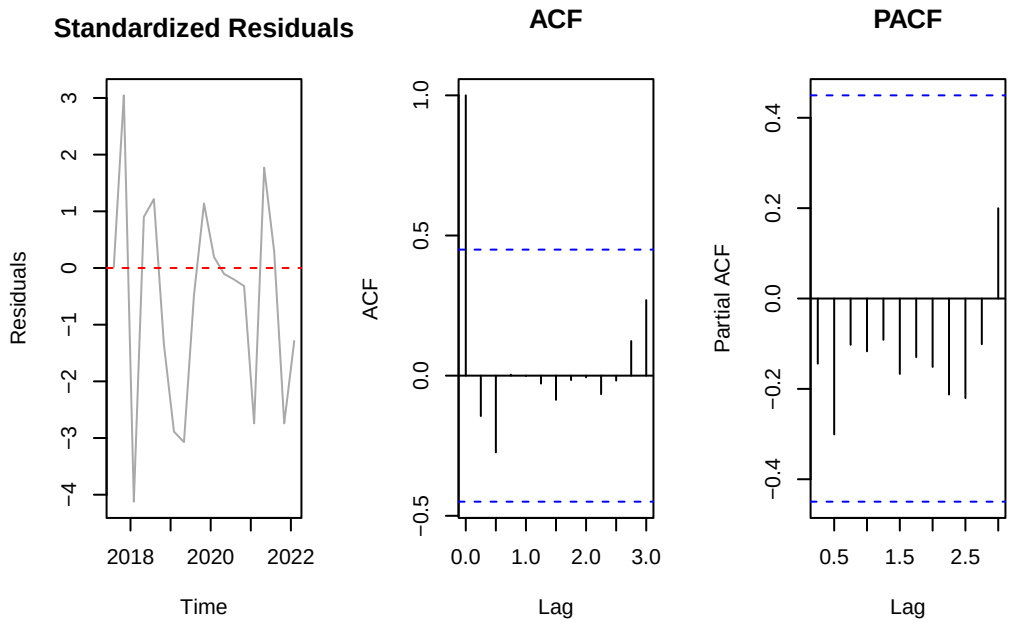


Figure 14: Residual diagnostics for ARIMA(0, 1, 1) model

The standardised residuals for ARIMA(0, 1, 1) model are randomly scattered around zero, with no obvious trend. The ACF and PACF plots of the residuals show no significant autocorrelation, as all spikes lie within the 95% confidence bounds. These diagnostics suggest that the residuals behave

approximately as white noise, supporting the adequacy of the ARIMA(0, 1, 1) model for the quarterly AMOC data.

ARIMA(1, 1, 0)

```
# Fitting ARIMA(1, 1, 0) model (with differencing)
fit_arima110_q <- arima(train_quarterly, order = c(1, 1, 0))
fit_arima110_q
```

```
Call:
arima(x = train_quarterly, order = c(1, 1, 0))

Coefficients:
          ar1
      -0.5011
s.e.   0.2089

sigma^2 estimated as 5.089:  log likelihood = -40.33,  aic = 84.66
```

The ARIMA(1, 1, 0) model fitted to the differenced quarterly AMOC data produced an autoregressive coefficient of -0.5011, indicating moderate negative dependence on the previous value. However, its AIC value of 84.66 was higher than several competing models, suggesting a less optimal fit compared to ARIMA(0, 1, 1). Therefore, residual diagnostics for this model were not pursued further, given its inferior performance relative to competing models.

SARIMA(1, 0, 0)(1, 0, 0)[4]

```
# Fitting SARIMA(1, 0, 0)(1, 0, 0)[4] model - with seasonal differencing
fit_sarima_q <- arima(train_quarterly, order = c(1, 1, 0),
                      seasonal = list(order = c(1, 1, 0), period = 4))
fit_sarima_q
```

```
Call:
arima(x = train_quarterly, order = c(1, 1, 0), seasonal = list(order = c(1,
    1, 0), period = 4))

Coefficients:
          ar1     sar1
```

```
      -0.5388  -0.4000
s.e.   0.2524   0.2903

sigma^2 estimated as 6.881:  log likelihood = -33.85,  aic = 73.71
```

The SARIMA(1, 1, 0)(1, 1, 0)[4] model fitted to the differenced quarterly AMOC data produced moderate negative autoregressive and seasonal autoregressive coefficients of -0.5388 and -0.4, respectively. The model produced an AIC of 73.71, it achieved the lowest AIC and highest log-likelihood among all models considered, suggesting that incorporating seasonal dynamics provided the best fit for the quarterly data.

```r
par(mfrow = c(1, 3))

# Standardized residuals
ts.plot(residuals(fit_sarima_q),
        main = "Standardized Residuals", col = "darkgray",
        ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)

# ACF
acf(residuals(fit_sarima_q), main = "ACF")

# PACF
pacf(residuals(fit_sarima_q), main = "PACF")
```

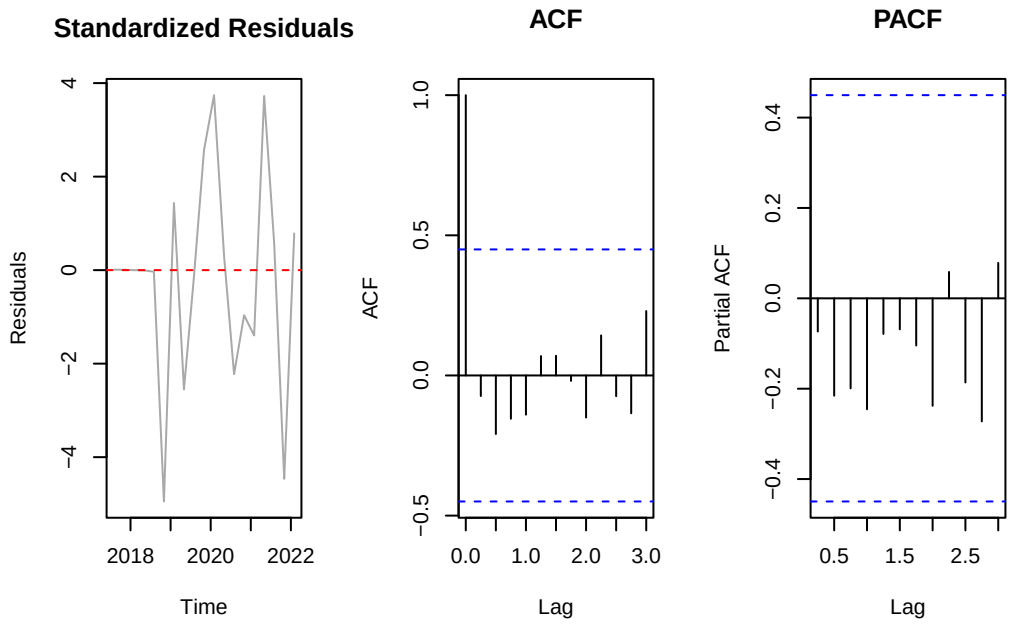


Figure 15: Residual diagnostics for SARIMA(1, 1, 0)(1, 1, 0)[4] model

Initially, the standardised residuals stay close to zero around 2018 for the SARIMA(1, 1, 0)(1, 1, 0)[4] model. However, after this point, the model fluctuates randomly around zero with no obvious trend or changing variance. The ACF and PACF plots show that the majority of autocorrelation values lie within the 95% confidence bounds, indicating no significant remaining autocorrelation. These results imply that the residuals are approximately white noise, validating the adequacy of the model for modelling the quarterly AMOC data.

Model Selection Summary

```
layout(matrix(c(1, 2, 3, 3), nrow = 2, byrow = TRUE))

lags <- 1:20

# ARIMA(0,1,1)
lb_arima011 <- sapply(lags, function(l) Box.test(residuals(fit_arima011_q),
                                                 lag = l,
                                                 type = "Ljung-Box")$p.value)
plot(lags, lb_arima011, type = "b", ylim = c(0, 1), col = "steelblue",
     main = "ARIMA(0,1,1)", xlab = "Lag", ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)

# ARIMA(1,1,0)
```

```r
lb_arima110 <- sapply(lags, function(l) Box.test(residuals(fit_arima110_q),
                                                 lag = l,
                                                 type = "Ljung-Box")$p.value)
plot(lags, lb_arima110, type = "b", ylim = c(0, 1), col = "steelblue",
     main = "ARIMA(1,1,0)", xlab = "Lag", ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)

# SARIMA(1,1,0)(1,1,0)[4]
lb_sarima <- sapply(lags, function(l) Box.test(residuals(fit_sarima_q), lag = l,
                                               type = "Ljung-Box")$p.value)
plot(lags, lb_sarima, type = "b", ylim = c(0, 1), col = "steelblue",
     main = "SARIMA(1,1,0)(1,1,0)[4]", xlab = "Lag",
     ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)
```

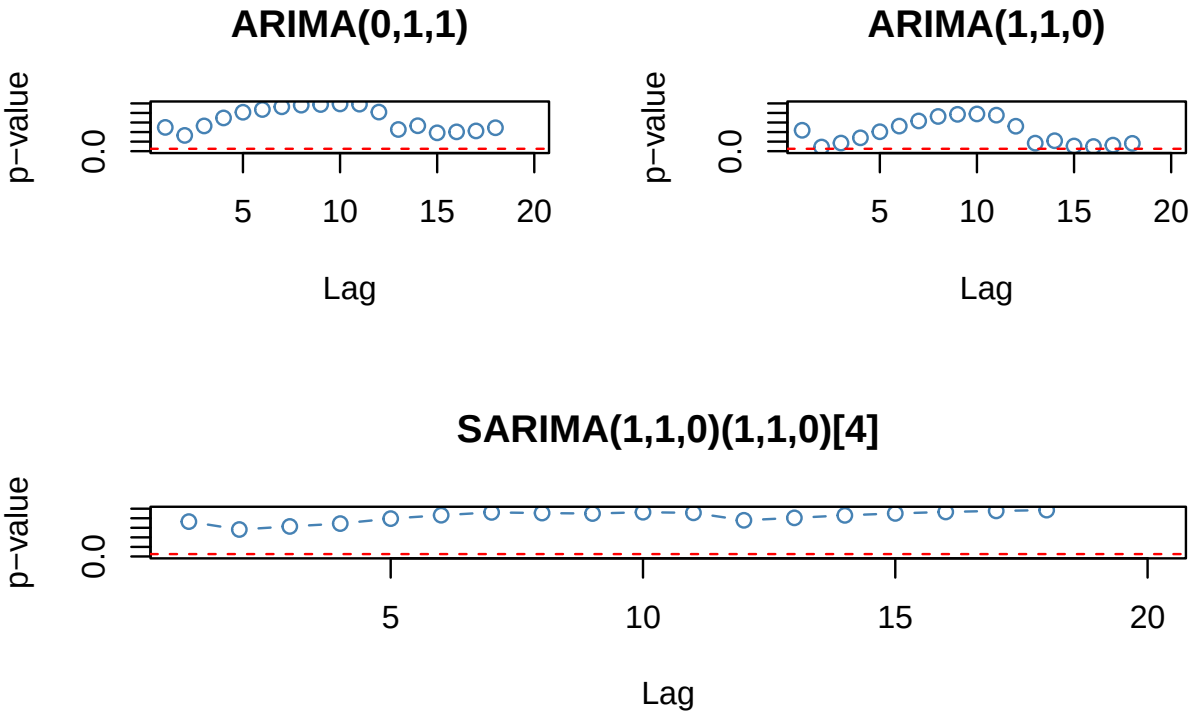


Figure 16: Ljung-Box Plots

The Ljung-Box p-value plots shown above display that the ARIMA(0, 1, 1) model exhibits all values above the 0.05 thresholds, with some inconsistency. This indicates there is no significant autocorrelation in the residuals. The ARIMA(1, 1, 0) model has a few values at lag 2 and later lags from lag 15 onwards that fall close to or below the 0.05 threshold. This suggests there is residual autocorrelation

that remains unaddressed in this model. In contrast, the SARIMA(1, 1, 0)(1, 1, 0)[4] model's p-values are consistently above 0.05 across all lags tested, indicating no significant autocorrelation in the residuals. These results further validate the SARIMA model as the best fitting model for the quarterly AMOC data.

```
# Summary table
model_aic_summary <- data.frame(
  Model = c("ARIMA(0,1,1)", "ARIMA(1,1,0)", "SARIMA(1,1,0)(1,1,0)[4]"),
  AIC = c(AIC(fit_arima011_q), AIC(fit_arima110_q), AIC(fit_sarima_q))
)


kable(model_aic_summary,
      caption = "Model Comparison Based on AIC Values - Quarterly Data")
```

Table 4: Model Comparison Based on AIC Values - Quarterly Data

Model	AIC
ARIMA(0,1,1)	80.46484
ARIMA(1,1,0)	84.65867
SARIMA(1,1,0)(1,1,0)[4]	73.70687

The AIC comparison table above shows that the SARIMA model achieved the lowest AIC value of 73.70, outperforming both the ARIMA(0, 1, 1) and the ARIMA(1, 1, 0) models, which had higher AIC values of 80.46 and 84.65 respectively. Therefore, incorporating seasonal dynamics for this quarterly data provided a better modelling fit for forecasting.

(d)

Dynamic Linear Modelling

A dynamic linear model is a state-space time series that allows the trend and seasonal patterns to change over time. It is more flexible than the previously fitted ARIMA models because the components like trend and seasonality can evolve.

The Dynamic Linear Model (DLM) is expressed in standard state-space form as:

$$x_t = \Phi x_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, Q)$$

$$y_t = A_t x_t + v_t, \quad v_t \sim \mathcal{N}(0, R)$$

where Φ is a $p \times p$ matrix, and A_t is a $q \times p$ matrix. In this setup, we set $q = 1$ and assume $A_t = A$ (constant over time).

First, we will build a DLM structure for the monthly data and then the quarterly data. To model both a dynamic level and a linear trend, a second-order polynomial model (order = 2) was used for the trend component of the Dynamic Linear Model (DLM). A seasonal component with period 12 (monthly data) or period 4 (quarterly data) is also included.

DLM for Monthly AMOC data:

```
# Monthly Data DLM
build_monthly_dlm <- function(x) {
  dlmModPoly(order = 2, dV = exp(x[1]), dW = c(0, exp(x[2]))) +
  dlmModSeas(frequency = 12, dV = 0, dW = c(exp(x[3]), rep(0, 10)))
}
```

```
Monthly_DLM <- dlmMLE(train_ts, parm = c(0, 0, 0), build = build_monthly_dlm)
Monthly_DLM$par
```

```
[1]   1.7362495 -20.9496979  -0.2368276
```

```
# Building final monthly DLM with estimated parameters
model_monthly_dlm <- build_monthly_dlm(Monthly_DLM$par)
V(model_monthly_dlm)
```

```
         [,1]
[1,] 5.676015
```

```
# Filtering
filtered_monthly <- dlmFilter(train_ts, model_monthly_dlm)

# Calculation of residuals
monthly_residuals <- drop(as.numeric(filtered_monthly$y)) - dropFirst(filtered_monthly$m[,1])


par(mfrow = c(1, 3))

# Standardized residuals
ts.plot(monthly_residuals, main = "Standardized Residuals",
        col = "darkgray", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)
```

```r
# ACF
acf(monthly_residuals, main = "ACF")

# PACF
pacf(monthly_residuals, main = "PACF")
```

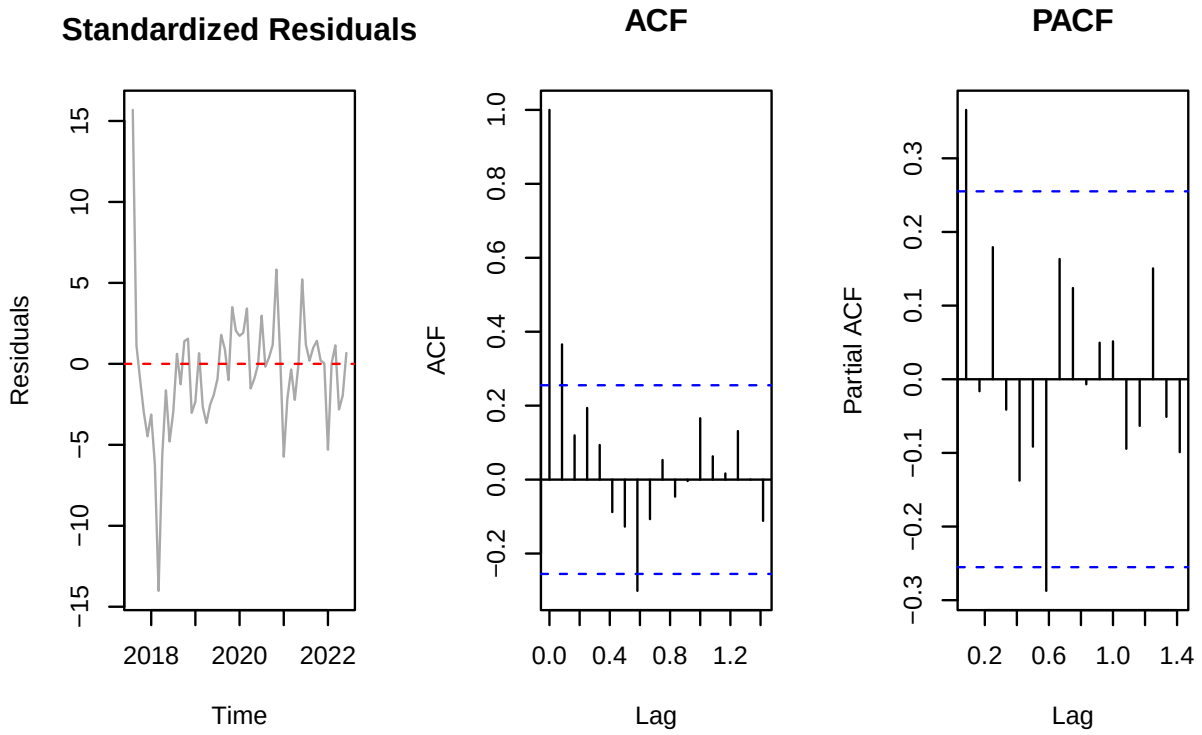


Figure 17: Monthly DLM Residuals Diagnostics

The standardised residuals from the monthly DLM appear to be centred around zero but show periods of higher variability. The ACF plot shows significant autocorrelation at lag 1, suggesting some short term structure still remains in the residuals. The PACF shows a few spikes such as at lag 7, indicating the model may have not accounted for all autocorrelation. However, all other points are within the confidence bounds.

DLM for Quarterly AMOC data:

```r
# Quarterly Data DLM
build_quarterly_dlm <- function(x) {
  dlmModPoly(order = 2, dV = exp(x[1]), dW = c(0, exp(x[2]))) +
    dlmModSeas(frequency = 4, dV = 0, dW = c(exp(x[3]), rep(0, 2)))
}
```

```r
Quarterly_DLM <- dlmMLE(train_quarterly, parm = c(0, 0, 0),
                        build = build_quarterly_dlm)
Quarterly_DLM$par
```

```
[1]   0.8312919 -19.5607447  -1.3946895
```

```r
# Building final quarterly dlm with estimated parameters
model_quarterly_dlm <- build_quarterly_dlm(Quarterly_DLM$par)
V(model_quarterly_dlm)
```

```
         [,1]
[1,] 2.296283
```

```r
# Filtering the quarterly model
filtered_quarterly <- dlmFilter(train_quarterly, model_quarterly_dlm)

# Calculating residuals: observed minus predicted trend
quarterly_residuals <- filtered_quarterly$y - filtered_quarterly$m[-1, 1]

par(mfrow = c(1, 3))

# Standardized residuals
ts.plot(quarterly_residuals, main = "Standardized Residuals",
        col = "darkgray", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)

# ACF
acf(quarterly_residuals, main = "ACF")

# PACF
pacf(quarterly_residuals, main = "PACF")
```

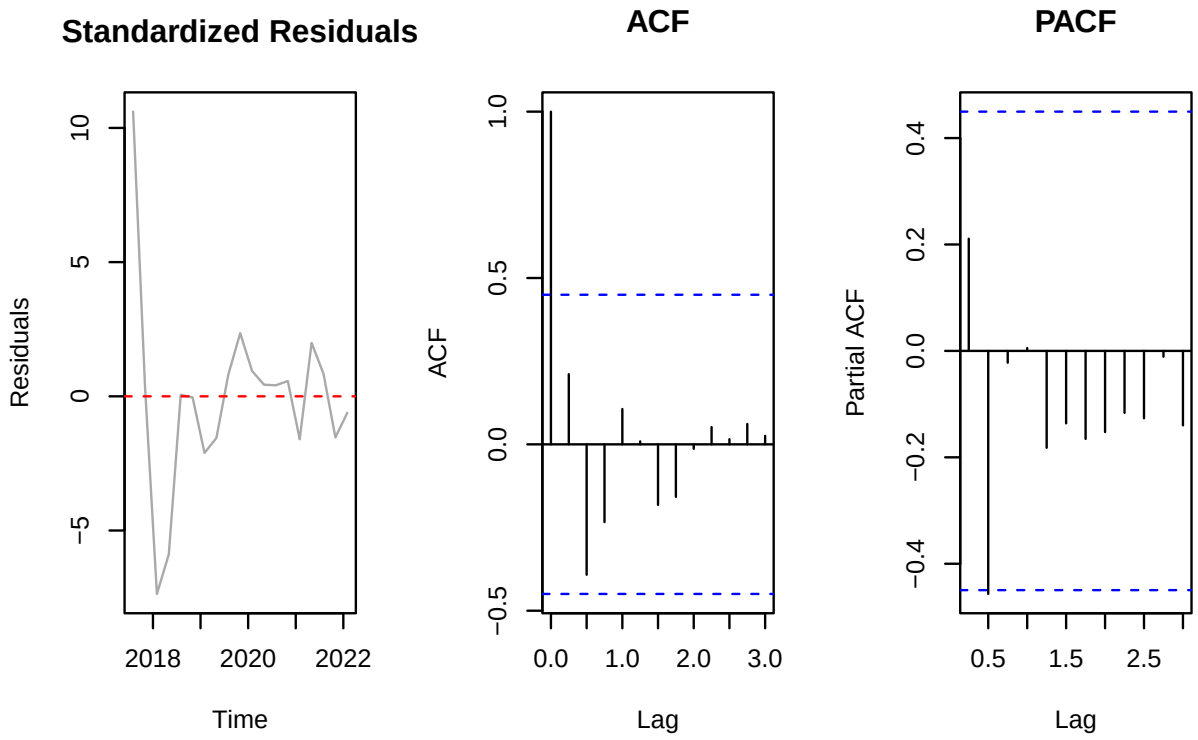


Figure 18: Quarterly DLM residual diagnostics

The standardised residuals from the quarterly DLM are centred around zero with no apparent trend. The ACF plot shows a significant spike at lag 1 and the PACF plot shows a significant spike at lag 2, implying that there is still some autocorrelation shown in the model. Despite this, the ACF and PACF plots still show that most of the autocorrelations lie within the 95% confidence bounds, suggesting that the residuals behave approximately as white noise. This indicates that the quarterly DLM model adequately captures the underlying structure.

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

lags <- 1:20

# Monthly DLM
lb_monthly <- sapply(lags, function(l) Box.test(monthly_residuals, lag = l,
                                                type = "Ljung-Box")$p.value)
plot(lags, lb_monthly, type = "b", ylim = c(0, 1), col = "blue",
     main = "Ljung-Box p-values - Monthly DLM Residuals", xlab = "Lag",
     ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)
```

```r
# Quarterly DLM
lb_quarterly <- sapply(lags, function(l) Box.test(quarterly_residuals, lag = l,
                                                  type = "Ljung-Box")$p.value)
plot(lags, lb_quarterly, type = "b", ylim = c(0, 1), col = "blue",
     main = "Ljung-Box p-values - Quarterly DLM Residuals", xlab = "Lag",
     ylab = "p-value")
abline(h = 0.05, col = "red", lty = 2)
```

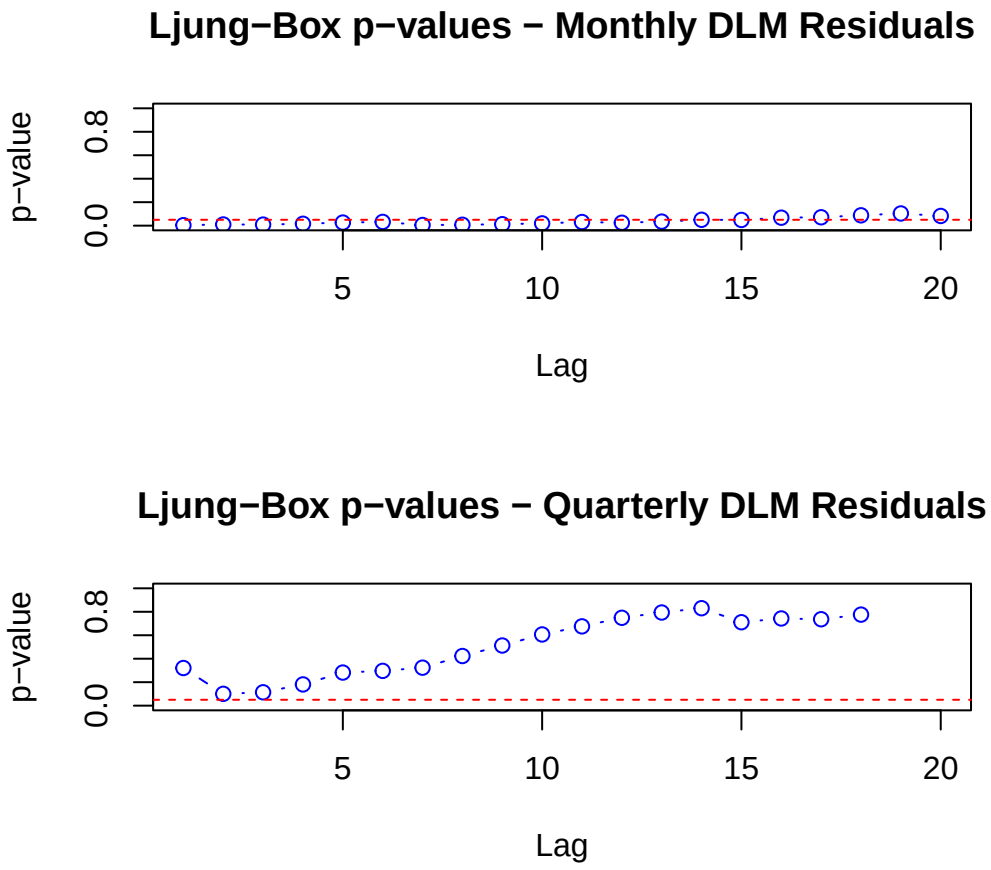


Figure 19: Ljung-Box Plots for DLMs

The Ljung-Box p-value plots show that the residuals from the quarterly DLM model consistently remain above the 0.05 significance threshold across all lags, suggesting there is no significant residual autocorrelation. In contrast, the monthly DLM residuals show many of its p-values near or below 0.05, indicating mild autocorrelation and a less adequate model fit compared to the quarterly DLM.

(e)

Forecast ARIMA, SARIMA and DLM models

First we will forecast our ARIMA (1, 1, 1) model using our testing data that was set aside earlier on. We will be forecasting the last 8 months (or 2 quarters). We will then forecast our SARIMA model and DLM model, creating a comparison table to see how the forecasted values compare to the actual values from the testing data.

```
# 1. Forecasts (nAhead = 3)
arima_pred <- round(forecast(fit_arima111, h = 3)$mean, 2)
sarima_pred <- round(forecast(fit_sarima_q, h = 3)$mean, 2)

# DLM forecast
dlm_forecast_q <- dlmForecast(filtered_quarterly, nAhead = 3)
dlm_pred <- round(drop(dlm_forecast_q$f), 2)

# 2. True values
true_vals <- round(as.numeric(test_quarterly), 2)


# 3. Absolute errors
arima_err <- round(abs(arima_pred - true_vals), 2)
sarima_err <- round(abs(sarima_pred - true_vals), 2)
dlm_err <- round(abs(dlm_pred - true_vals), 2)

# 4. Quarters
quarters <- c("2022 Q3", "2022 Q4", "2023 Q1")

# 5. Build table
forecast_tbl <- data.frame(
  Quarter = rep(quarters, 2),
  Type = rep(c("Prediction", "A. Error"), each = 3),
  ARIMA = c(arima_pred, arima_err),
  SARIMA = c(sarima_pred, sarima_err),
  DLM = c(dlm_pred, dlm_err),
  `True Value` = rep(true_vals, 2)
)

# 6. Output nicely
kable(forecast_tbl,
      caption = "Forecasted vs True Mean Atlantic Meridional Overturning Circulation")
```

Table 5: Forecasted vs True Mean Atlantic Meridional Overturning Circulation

Quarter	Type	ARIMA	SARIMA	DLM	True.Value
2022 Q3	Prediction	16.02	15.83	15.98	16.61
2022 Q4	Prediction	16.01	15.89	16.00	16.14
2023 Q1	Prediction	16.01	13.63	15.05	15.06
2022 Q3	A. Error	0.59	0.78	0.63	16.61
2022 Q4	A. Error	0.13	0.25	0.14	16.14
2023 Q1	A. Error	0.95	1.43	0.01	15.06

Table 5 above compares the forecasted AMOC values from the ARIMA(1, 1, 1), SARIMA(1, 1, 0)(1, 1, 0)[4], and DLM models against the actual observations for the last 8 months or 2 quarters. 3 quarters are shown since the data gives values up until February 2023. All three models produced forecasts that closely approximated the true values, with absolute errors remaining below 1.5 in every case. The DLM and SARIMA models demonstrated slightly strong performance in 2022 Q3 and Q4, while ARIMA showed the lowest error in Q4 of 0.13. The SARIMA model undestimated the true value in 2023 Q1 with an absolute error of 1.43. Notably, the DLM model exhibited the lowest accumulative error across all three quarters, suggesting that it provided the most consistent and accurate short-term forecast in this context. Overall, all models captured the trend well, with no extreme under or overestimation.

```
# MAE
mae <- function(pred, actual) {
  mean(abs(pred - actual))
}
# RMSE
rmse <- function(pred, actual) {
  sqrt(mean((pred - actual)^2))
}
# Bias (mean error)
bias <- function(pred, actual) {
  mean(pred - actual)
}
# Compute for each model
metrics_tbl <- data.frame(
  Metric = c("MAE", "RMSE", "Bias"),
  ARIMA = c(mae(arima_pred, true_vals),
            rmse(arima_pred, true_vals),
            bias(arima_pred, true_vals)),
  SARIMA = c(mae(sarima_pred, true_vals),
             rmse(sarima_pred, true_vals),
             bias(sarima_pred, true_vals)),
```

```
  DLM = c(mae(dlm_pred, true_vals),
          rmse(dlm_pred, true_vals),
          bias(dlm_pred, true_vals))
)
# Round for display
metrics_tbl[ , 2:4] <- round(metrics_tbl[ , 2:4], 2)

# Show table
knitr::kable(metrics_tbl, caption = "Predictive Performance Metrics for Models")
```

Table 6: Predictive Performance Metrics for Models

Metric	ARIMA	SARIMA	DLM
MAE	0.56	0.82	0.26
RMSE	0.65	0.95	0.37
Bias	0.08	-0.82	-0.26

Table 6 displays the predictive performance of the ARIMA, SARIMA, and DLM models over the final 8 months of the monthly AMOC data. Among the three, the DLM model performed best, achieving the lowest MAE of 0.26 and lowest RMSE of 0.37, indicating that it produced the most accurate and stable predictions overall. The ARIMA model also performed reasonably well, with moderate error and near-zero bias of 0.08, suggesting balanced forecasts. In contrast, the SARIMA model exhibited the highest error and a noticeable negative bias of -0.82, suggesting a consistent underestimation of the AMOC values. Overall, the DLM was the most reliable model for forecasting the last 2 quarters or 8 months of the AMOC data.

3. California Daily Temperatures

(a)

```
metadata <- read_csv("C:/Users/joshh/OneDrive - University of Exeter/MSc Applied Data Science

temps <- read_csv("C:/Users/joshh/OneDrive - University of Exeter/MSc Applied Data Science ar

# Convert date column and reshape temperature data to long format
temps_long <- temps %>%
  rename(Date = 1) %>%  # First column is date
  mutate(Date = ymd(Date)) %>%
  pivot_longer(-Date, names_to = "Location", values_to = "Temperature")

# Merge with metadata to attach lat/long/elev
full_data <- temps_long %>%
  left_join(metadata, by = "Location")
```

Spatial Plot of California Daily Temperature Data

```
# Get USA map data
usa <- ne_states(country = "United States of America", returnclass = "sf")

# Focus only on California
california <- usa %>% filter(name == "California")

mean_temps <- full_data %>%
  group_by(Location) %>%
  summarise(mean_temp = mean(Temperature, na.rm = TRUE)) %>%
  left_join(metadata, by = "Location") %>%
  st_as_sf(coords = c("Long", "Lat"), crs = 4326)

# Convert to regular data frame with coordinates
mean_temps_coords <- mean_temps %>%
  mutate(lon = st_coordinates(.)[,1],
         lat = st_coordinates(.)[,2])

# Computing average max temperature per location
mean_temps <- full_data %>%
  group_by(Location) %>%
```

```
  summarise(mean_temp = mean(Temperature, na.rm = TRUE)) %>%
  left_join(metadata, by = "Location") %>%
  st_as_sf(coords = c("Long", "Lat"), crs = 4326)


ggplot() +
  geom_sf(data = california, fill = "gray95", color = "black") +
  geom_sf(data = mean_temps, aes(color = mean_temp), size = 4) +
  geom_text_repel(data = mean_temps_coords,
                  aes(x = lon, y = lat, label = Location),
                  size = 3.5,
                  max.overlaps = Inf,
                  min.segment.length = 0,
                  seed = 123) +
  scale_color_viridis_c(name = "Mean Max Temp (°C)", option = "viridis") +
  coord_sf(xlim = c(-125, -113), ylim = c(32, 43), expand = FALSE) +
  labs(title = "Mean Max Daily Temperature by Location (2012)") +
  theme_minimal(base_size = 14)
```

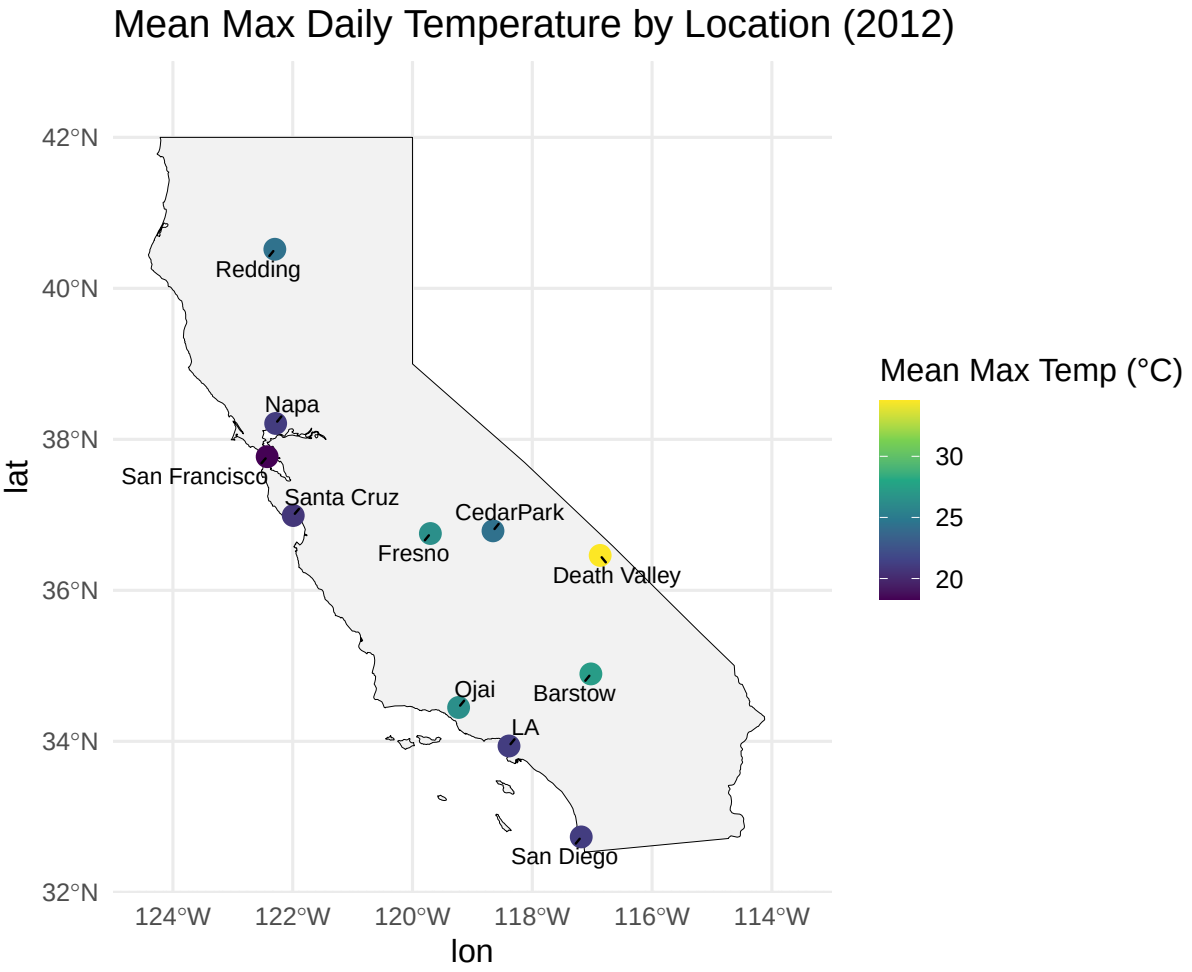


Figure 20: California Weather Station Locations 2012

The map in Figure-20 shows the locations of 11 weather stations across California, coloured by mean maximum temperature in 2012. Stations span a range of geographic regions - from coastal cities such as Santa Cruz, San Francisco to inland desert areas such as Death Valley, and high-elevation sites like Cedar Park and Barstow.

Temperature differences are clearly visualised using a colour gradient: warmer colours indicate higher average maximum temperatures. Death Valley recorded the highest mean maximum temperature, while coastal stations such as San Diego, Santa Cruz and San Francisco show lower values, likely due to the moderating influence of the Pacific Ocean. Cedar Park and Barstow appear to be moderately warm despite higher elevations, suggesting other climatic factors may be at play.

The map offers an intuitive spatial context for interpreting daily and seasonal temperature trends across the state.

Time Series Plot of California Daily Temperature Data

```
ggplot(full_data, aes(x = Date, y = Temperature, color = Location)) +
  geom_smooth(se = FALSE, method = "loess", linewidth = 1.2, alpha = 0.9) +
  scale_x_date(date_breaks = "1 month", date_labels = "%b") +
  labs(title = "Smoothed Daily Max Temperatures Across California (2012)",
       x = "Month", y = "Max Temperature (°C)") +
  theme_minimal(base_size = 14) +
 theme(
  legend.position = "right", legend.title = element_blank(), legend.text = element_text(size
  legend.spacing.x = unit(0.5, "cm"), legend.box = "horizontal"
)
```

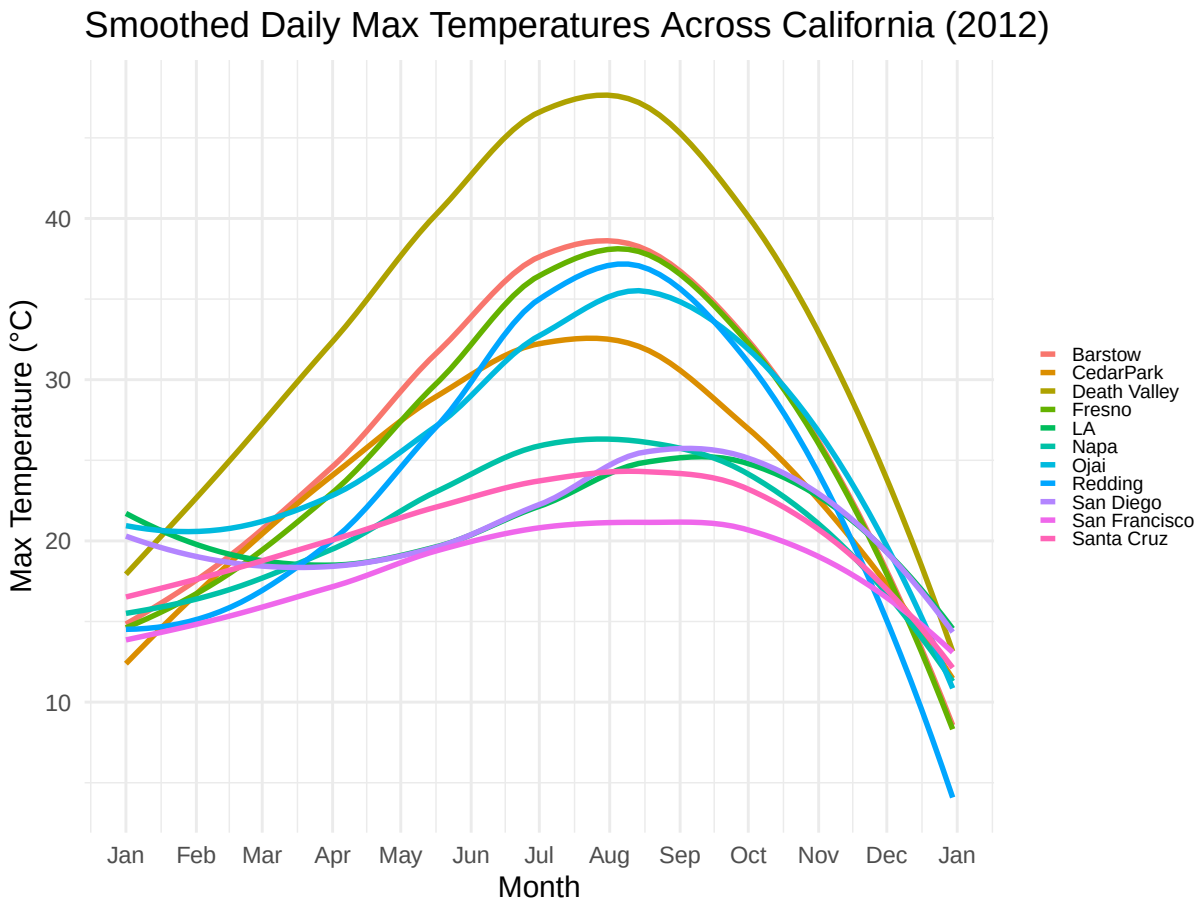


Figure 21: Daily Max Temperatures Across California 2012

The plot above presents smoothed daily maximum temperatures for all 11 California weather stations in 2012. All stations show a clear seasonal pattern, with temperatures peaking between July and August

and reaching their lowest levels in January ad December. This is consistent with a typical Northern Hemisphere summer-winter cycle.

Death Valley stands out dramatically, recording the highest peak temperatures, regularly exceeding 45 degrees celsius in summer. In contrast, coastal stations such as San Francisco, Santa Cruz, and San Diego as seen in the spatial plot, display the lowest and most stable temperatures ranging from around 15 to 25 degrees celsius.

Inland stations such as Barstow, Fresno and Redding exhibit steeper seasonal gradients and warmer summers compared to coastal locations. Some higher-elevation sites such as CedarPark show relatively subdued warming, highlighting the role of elevation in suppressing extreme daytime temperatures.

Overall, this time series plot complements the spatial analysis by revealing how geographic location - particularly proximity to the coast, elevation, and desert vs. urban setting - governs temporal temperature dynamics across California.

(b)

Gaussian Process Model using Maximum Likelihood

```
# First filtering data for Decemeber 13th 2012
dec13_data <- full_data %>%
  filter(Date == as.Date("2012-12-13")) %>%
  mutate(Location = as.character(Location))

# Exclude San Diego and Fresno
train_data <- dec13_data %>%
  filter(!(Location %in% c("San Diego", "Fresno")))

# Prediction locations: only San Diego and Fresno
pred_data <- dec13_data %>%
  filter(Location %in% c("San Diego", "Fresno"))
```

```
# Creating geodata object and plotting empirical variogram
geo_train <- as.geodata(train_data,
                        coords.col = c("Long", "Lat"),
                        data.col = "Temperature",
                        covar.col = "Elev")


vario <- variog(geo_train, option = "bin")
```

```
variog: computing omnidirectional variogram
```

```r
plot(vario, pch = 19, main = "Sample Variogram")
```

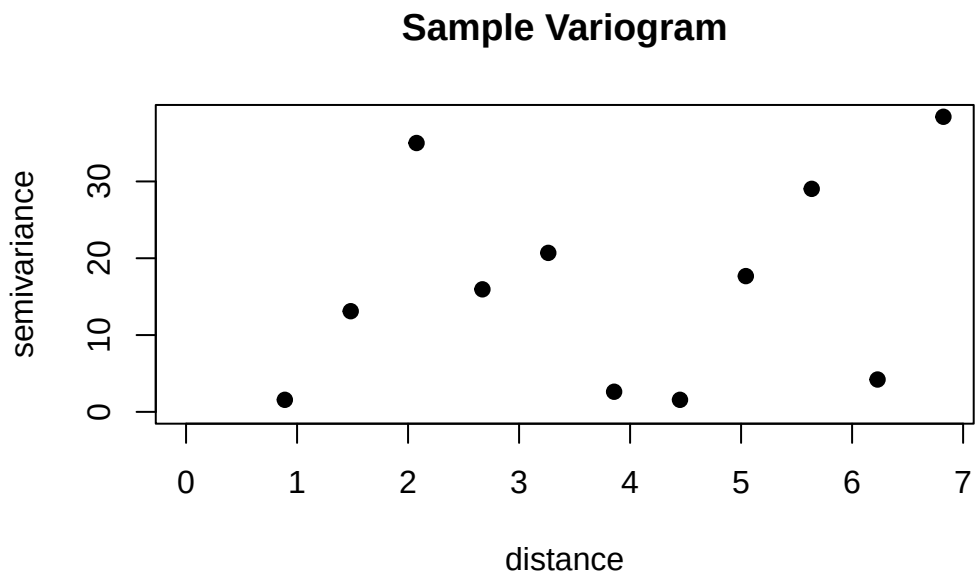


Figure 22: Sample Variogram for December 13th Data

The sample variogram above for December 13th shows us there is no strong spatial correlation and lots of noise, which is expected with only 9 stations. This suggests limited spatial dependence in the observed maximum temperatures. There is no smooth increasing trend in semivariance with distance. The points fluctuate unpredictably, and semivariance actually drops at mid-ranges (distances 4-5). This irregularity suggests very weak or noisy spatial structure in the data. As we proceed to fit a spatial Gaussian process model using maximum likelihood, we note that the model fit and resulting predictions are likely to be uncertain and should be interpreted with caution.

We will assume a constant mean function (i.e. no trend). We will use an exponential covariance model which is appropriate for short-range dependence and allows some smoothness. A small nugget will be estimated using `fix.nugget = FALSE`. We will choose initial values with a partial sill of 30 since the height of the semivariance from the variogram was roughly 30 and we will choose a range parameter of 3, which shows where the semivariance starts to level off. In fitting the Gaussian process model, we included a spatial trend based on elevation, specified as `trend = ~ Elev`. This allows the model to account for large-scale variation in temperature attributable to changes in altitude, with the residual variation modelled as a zero-mean spatially correlated Gaussian process.

```r
# Fitting the Gaussian process model using ML
gp_fit <- likfit(geo_train,
                 cov.model = "exponential",
                 ini.cov.pars = c(30, 3),
```

```
                  fix.nugget = FALSE,
                 kappa = 1.99,
                 trend = ~ Elev,
                 lik.method = "ML")
```

```
kappa not used for the exponential correlation function
---------------------------------------------------------------
likfit: likelihood maximisation using the function optim.
likfit: Use control() to pass additional
         arguments for the maximisation function.
        For further details see documentation for optim.
likfit: It is highly advisable to run this function several
        times with different initial values for the parameters.
likfit: WARNING: This step can be time demanding!
---------------------------------------------------------------
likfit: end of numerical maximisation.
```

```
print(gp_fit)
```

```
likfit: estimated model parameters:
    beta0     beta1     tausq   sigmasq       phi
"16.4939" "-0.0068" " 0.0000" "17.5345" " 6.3884"
Practical Range with cor=0.05 for asymptotic range: 19.13788

likfit: maximised log-likelihood = -21.22
```

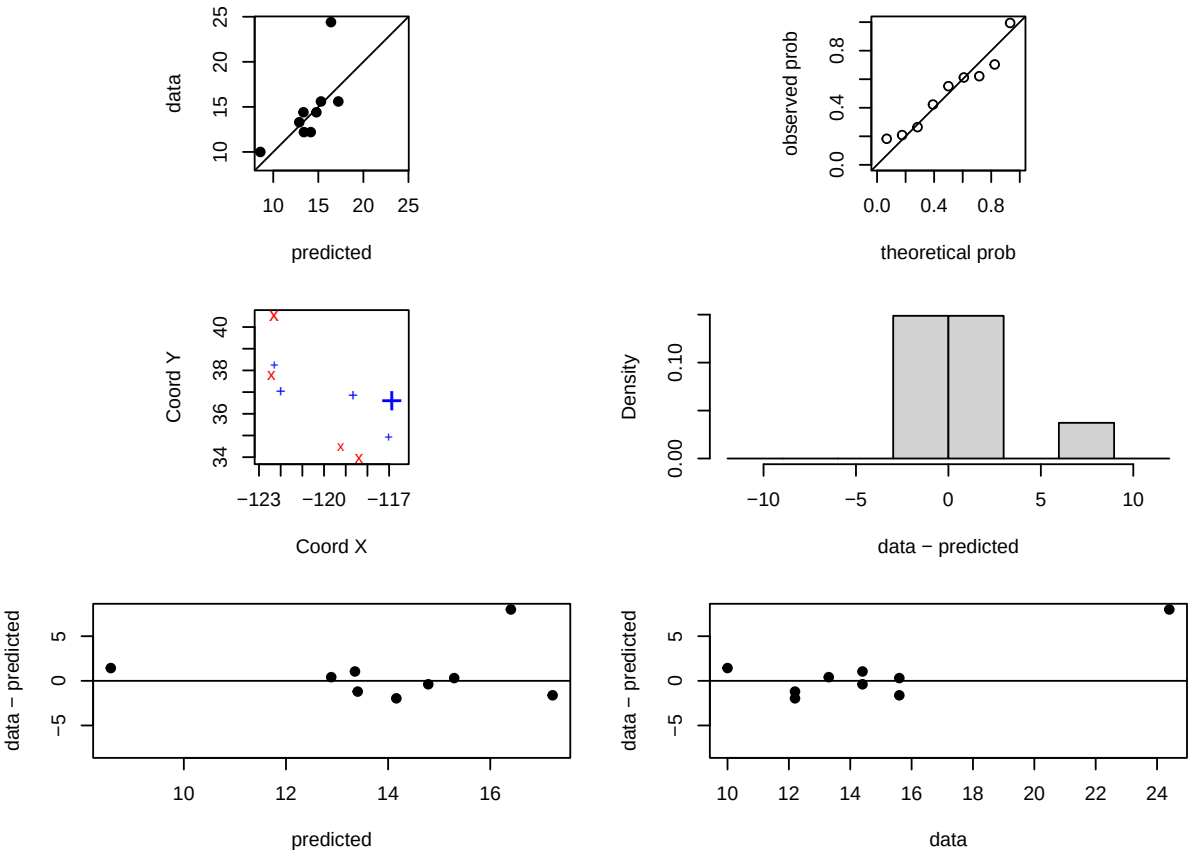
The Gaussian Process Model using maximum likelihood converged successfully using an exponential covariance function and a linear elevation trend. The estimated spatial variance (σ^2) was 17.53, and the range parameter (ϕ) was 6.39, suggesting moderate spatial dependence in maximum temperatures across the region. Elevation had a small but negative effect on temperature (-0.0068 degrees celsius per metre), consistent with physical expectations. No nugget effect was estimated, indicating that the model attributes nearly all variation to spatial structure and the linear trend.

Cross Validation for the Gaussian Process Model using Maximum Likelihood

```
# Leave-one-out cross-validation for the Gaussian Process Model using Maximum Likelihood
xv_gp_fit <- xvalid(geo_train, model = gp_fit)
```

```
xvalid: number of data locations       = 9
xvalid: number of validation locations = 9
xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9,
xvalid: end of cross-validation
```

```
par(mfrow = c(3, 2), mar = c(4, 4, 2, 2))
plot(xv_gp_fit, error = TRUE, std.error = FALSE, pch = 19)
```



The cross-validation diagnostics provide a visual and quantitative check on the predictive performance of the Gaussian process model fitted using maximum likelihood. The top-left scatterplot shows moderate alignement between predicted and observed temperatures. However, a few points fall noticeably off the 1:1 line, indicating some over-or under prediction.

The QQ plot (top-right) shows a close alignment between theoretical and observed quantiles, supporting the assumption that residuals are approximately normally distributed.

The residual histogram (middle-right) is roughly symmetric and centred near zero, indicating minimal systematic bias.

The residual vs. predicted and residual vs. observed plots show no strong trend or patterns, which supports the assumption of homoscedasticity (constant variance). Most residuals fall within +/- 5 degrees celsius, which is acceptable given the natural variability in temperature.

Overall, the model appears to offer reliable spatial predictions, with well-behaved residuals and no major signs of misspecification. The residual spread may reflect natural spatial variability rather than model inadequacy.

Predicting Maximum Temperature in San Diego and San Fresno on December 13th 2012

```
# Preparing data for predictions

# Filter from full dataset for prediction locations
pred_data <- full_data %>%
  filter(Date == as.Date("2012-12-13")) %>%
  filter(Location %in% c("San Diego", "Fresno")) %>%
  mutate(Location = as.character(Location))

# Kiring Model
pred <- krige.conv(geo_train, locations = as.matrix(pred_data[,c("Long", "Lat")]),
                   krige = krige.control(type.krige = "ok",
                  cov.model = "exponential",
                  obj.model = gp_fit))
```

```
krige.conv: model with constant mean
krige.conv: Kriging performed using global neighbourhood
```

```
# Print prediction results
results <- pred_data %>%
  mutate(pred.temp = pred$predict, pred.sd = sqrt(pred$krige.var)
         )

print(results)
```

```
# A tibble: 2 x 8
  Date       Location  Temperature  Long   Lat  Elev pred.temp pred.sd
  <date>     <chr>           <dbl> <dbl> <dbl> <dbl>     <dbl>   <dbl>
1 2012-12-13 San Diego        16.1 -117.  32.7   4.6      16.0    2.63
2 2012-12-13 Fresno           16.7 -120.  36.8 100        11.4    1.92
```

The kriging model predicted maximum daily temperatures of 16 degrees celsius for San Diego and 11.4 degrees celsius for Fresno on December 13th, 2012. These predicted are close to the observed values (16.1 and 16.7 respectively), although the model slightly underestimates Fresno. The prediction standard deviations of 2.62 degrees celsius and 1.92 degrees celsius respectively, reflect the level of uncertainty in these estimates due to spatial distance from training stations and the underlying variability in the temperature field. The higher uncertainty for San Diego suggests it may be less spatially supported by nearby observations compared to Fresno.

(c)

Time Series Modelling

Extracting Time Series for both stations

To make our time series forecast, we will first filter the data up to but not including the start of the forecast window. Therefore the training window will end on December 8th.

```
# Full data for San Diego and Fresno
station_data <- full_data %>%
  filter(Location %in% c("San Diego", "Fresno")) %>%
  mutate(Location = as.character(Location))

# Training data: up to Dec 8 (for Dec 9-13 forecast)
training_data <- station_data %>%
  filter(Date <= as.Date("2012-12-08"))

# Testing data: Dec 9-13 (for evaluating first forecast)
testing_data_1 <- station_data %>%
  filter(Date >= as.Date("2012-12-09"),
         Date <= as.Date("2012-12-13"))

# Separate time series data for training
san_diego_ts <- ts(training_data %>%
                     filter(Location == "San Diego") %>%
                     pull(Temperature),
                   start = c(2012, 1), frequency = 365)
```

```r
fresno_ts <- ts(training_data %>%
                  filter(Location == "Fresno") %>%
                  pull(Temperature),
                start = c(2012, 1), frequency = 365)
```

Plotting ACF and PACF of training data (before differencing)

```r
par(mfrow = c(1, 2))
acf(san_diego_ts, main = "ACF of Training Data (Raw)")
pacf(san_diego_ts, main = "PACF of Training Data (Raw)")
```

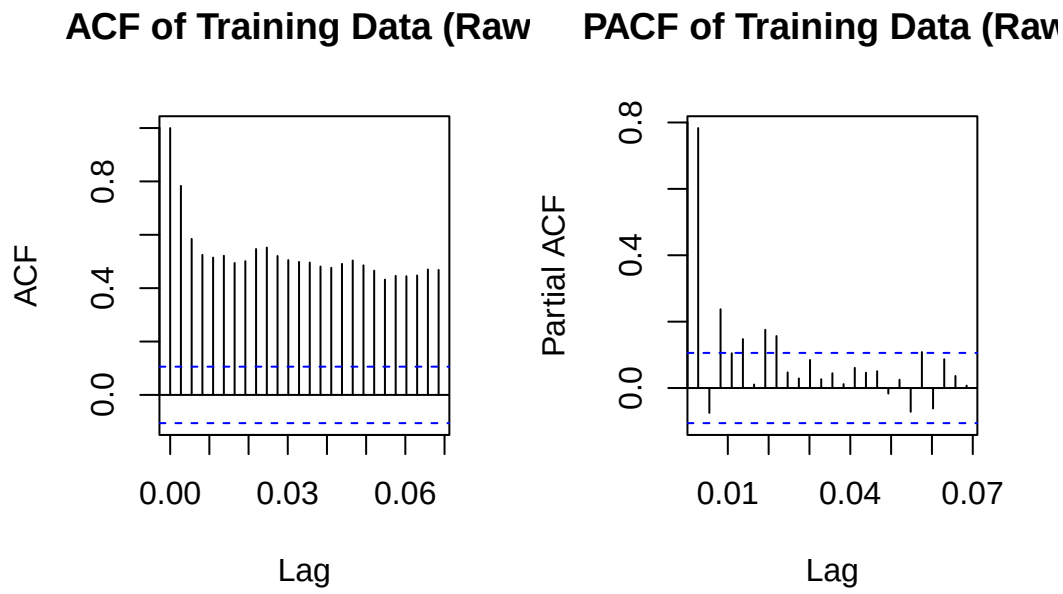


Figure 23: ACF and PACF of Training Data for San Diego

The ACF plot for San Diego shows a slow decay, indicating strong persistence and non-stationarity in the original temperature series. All lags remain outside of the confidence bounds, a sign of a unit root process. The PACF plot displays a sharp-cut off after lag 1 suggesting that an autoregressive component may be appropriate after differencing. These patterns support the need for first-order differencing (d=1).

```r
par(mfrow = c(1, 2))
acf(fresno_ts, main = "ACF of Training Data (Raw)")
pacf(fresno_ts, main = "PACF of Training Data (Raw)")
```

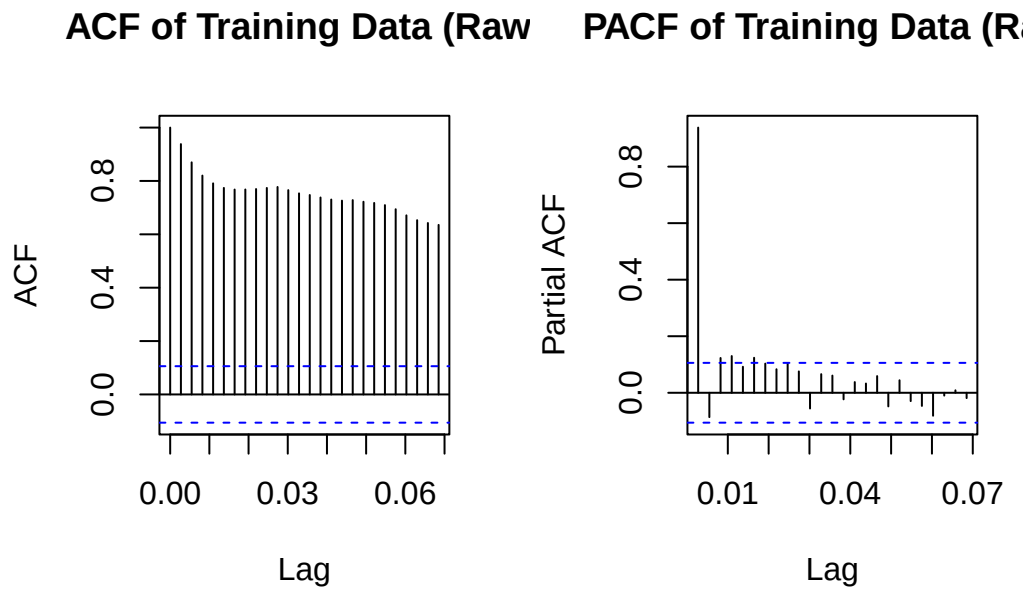


Figure 24: ACF and PACF of Training Data for Fresno

The ACF and PACF plots for Fresno is extremely similar to that of San Diego with almost identical patterns, further supporting the need for first-order differencing for both training data sets.

```
par(mfrow = c(1, 3))
# 1. Original series
plot(san_diego_ts, main = "Original San Diego",
     ylab = "Temp",
     xlab = "Time",
     col = "black")

# 2. First difference
diff_san_diego_ts <- diff(san_diego_ts)
plot(diff_san_diego_ts, main = "First Difference",
     ylab = "Differenced Temp",
     xlab = "Time",
     col = "black")

# 3. Second difference
diff2_san_diego_ts <- diff(diff_san_diego_ts)
plot(diff2_san_diego_ts, main = "Second Difference",
     ylab = "2nd Differenced Temp",
     xlab = "Time",
     col = "black")
```

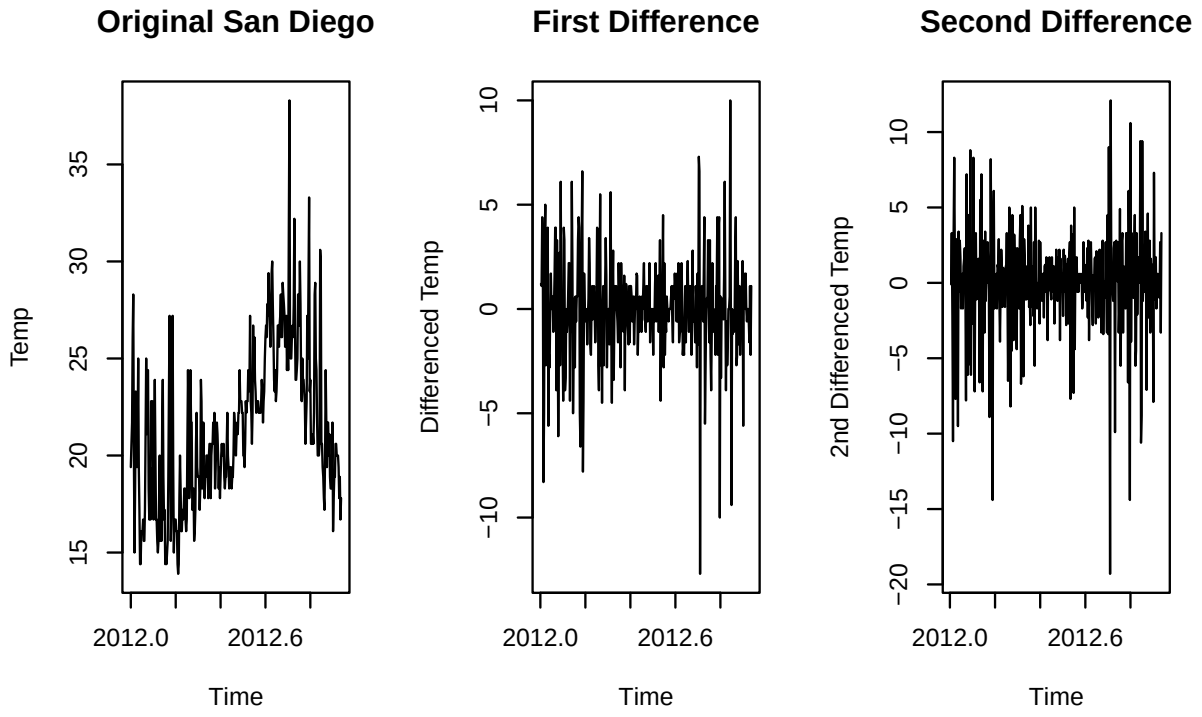


Figure 25: Differencing the Training Data for San Diego

The original San Diego temperature series shows a clear cyclical, indicating non-stationarity. After first-order differencing, the trend is removed and the series fluctuates around a constant mean, suggesting stationarity has been achieved. Second-order differencing appears to over-difference the data, introducing excessive noise. Therefore, a first-order difference (d=1) is appropriate to stabilise the mean and ensure the validity of ARIMA modelling.

```
par(mfrow = c(1, 3))
# 1. Original series
plot(fresno_ts, main = "Original Fresno",
     ylab = "Temp",
     xlab = "Time",
     col = "black")

# 2. First difference
diff_fresno_ts <- diff(fresno_ts)
plot(diff_fresno_ts, main = "First Difference",
     ylab = "Differenced Temp",
     xlab = "Time",
     col = "black")
```

```r
# 3. Second difference
diff2_fresno_ts <- diff(diff_fresno_ts)
plot(diff2_fresno_ts, main = "Second Difference",
     ylab = "2nd Differenced Temp",
     xlab = "Time",
     col = "black")
```

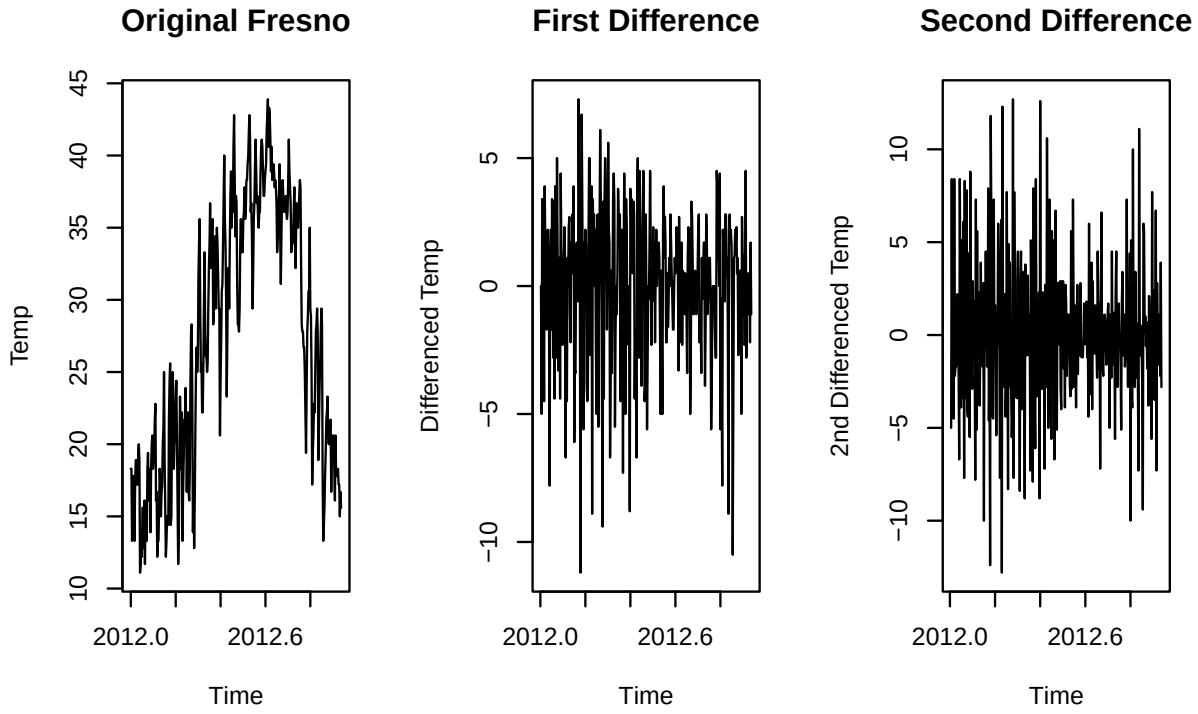


Figure 26: Differencing the Training Data for Fresno

The original Fresno temperature series shows a strong seasonal trend, indicating non-stationarity. First order differencing effectively removes this trend and stabilises the variance, producing a series that fluctuates around a constant. Again, a first-order difference (d=1) is appropriate to stabilise the mean and ensure the validity of ARIMA modelling in this case.

Plotting ACF and PACF of training data (after differencing)

```
# Plotting ACF and PACF after first differencing
par(mfrow = c(1, 2))


acf(diff(san_diego_ts), main = "ACF After First Differencing")
pacf(diff(san_diego_ts), main = "PACF After First Differencing")
```

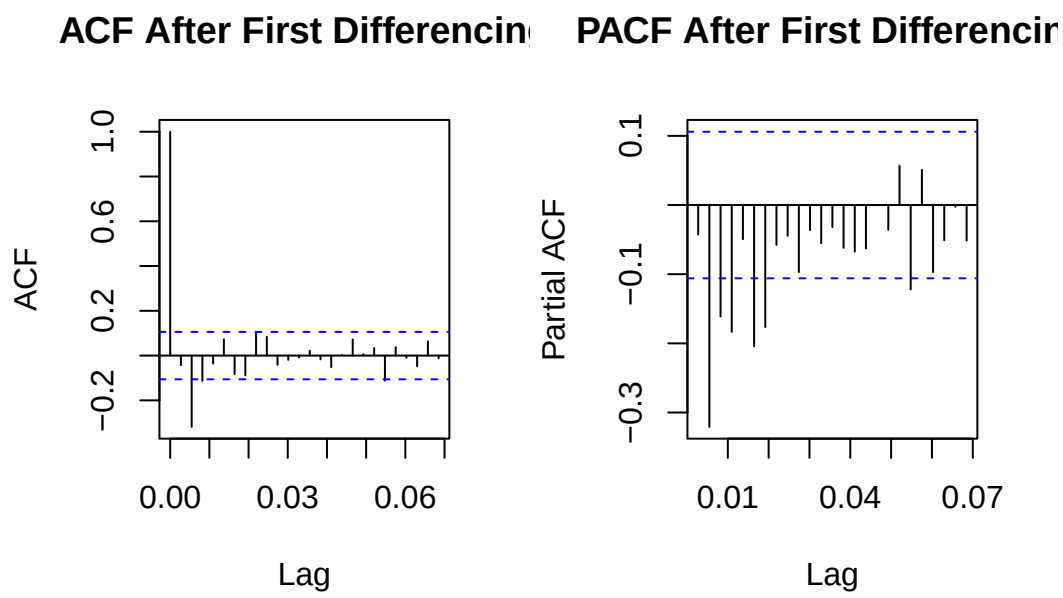


Figure 27: ACF and PACF of Training Data for San Diego

After first-order differencing, the ACF plot for San Diego shows a significant drop-off after lag 1, and subsequent lags mostly fall within the confidence bounds, suggesting that autocorrelation has been largely removed. The PACF plot exhibits a sharp cut off after lag 1. These patterns show that the differenced series is now stationary, and an ARIMA model with one autoregressive term (AR) and one differencing step (d=1) is suitable for modelling the data.

```
# Plotting ACF and PACF after first differencing
par(mfrow = c(1, 2))


acf(diff(fresno_ts), main = "ACF After First Differencing")
pacf(diff(fresno_ts), main = "PACF After First Differencing")
```

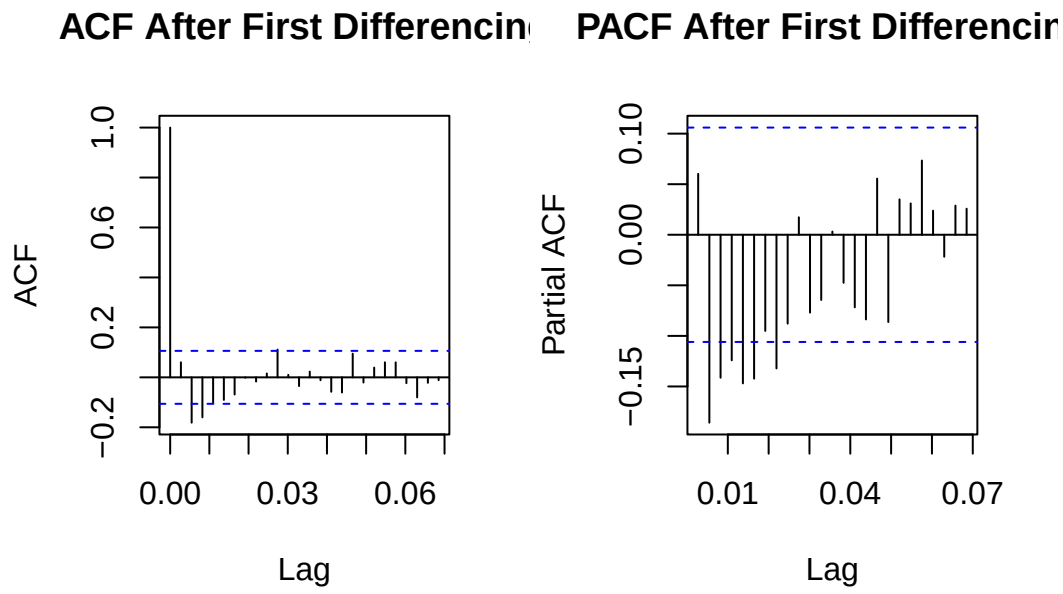


Figure 28: ACF and PACF of Training Data for Fresno

After first-order differencing, the ACF and PACF plots for Fresno training data are similar to San Diego's. The patterns show that the differenced series is now stationary, and an ARIMA model with one autoregressive term (AR) and one differencing step (d=1) is suitable for modelling this data.

Fitting ARIMA Models

```
# Fitting ARIMA (1, 1, 1) for San Diego
san_diego_fit1 <- Arima(san_diego_ts, order = c(1, 1, 1))
san_diego_fit1
```

```
Series: san_diego_ts
ARIMA(1,1,1)

Coefficients:
         ar1      ma1
      0.5471  -0.9282
s.e.  0.0526   0.0191

sigma^2 = 5.545:  log likelihood = -777.64
AIC=1561.29   AICc=1561.36   BIC=1572.79
```

The fitted ARIMA(1, 1, 1) model for San Diego has an AR(1) coefficient of 0.5471, indicating a moderate positive correlation with the previous day's differenced temperature. The MA(1) coefficient of -0.9282 suggests a strong correction for recent forecast errors, effectively capturing short-term shocks. The relatively low AIC (1561.29) and BIC (1572.79) indicate reasonably good model fit given the complexity.

```
# Model diagnostics for ARIMA (1, 1, 1) for San Diego
tsdiag(san_diego_fit1)
```

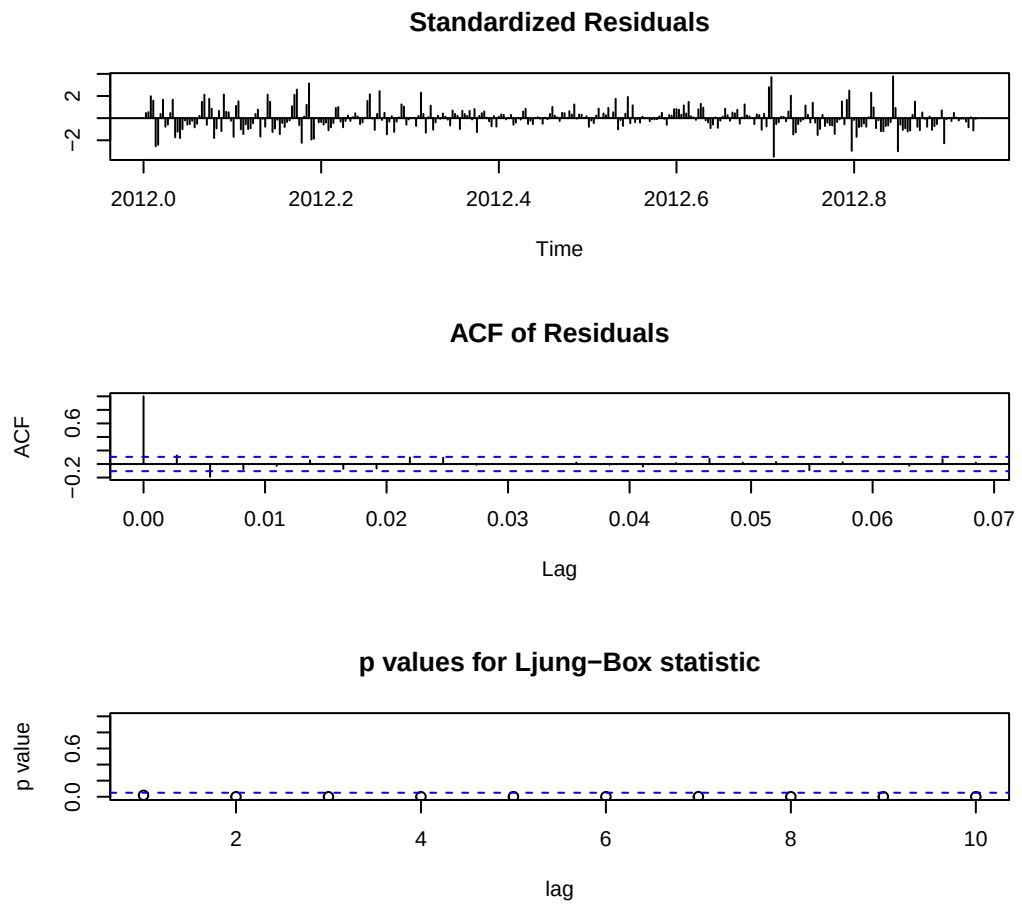


Figure 29: ARIMA (1, 1, 1) San Diego Diagnostics

Diagnostics show that the p values in the Ljung Box statistic test are on or below the 0.05 threshold which indicates that there is still some residual autocorrelation present within this model.

```r
# Fitting ARIMA (1, 1, 1) for Fresno
fresno_fit1 <- Arima(fresno_ts, order = c(1, 1, 1))
fresno_fit1
```

```
Series: fresno_ts
ARIMA(1,1,1)

Coefficients:
          ar1     ma1
      -0.4364  0.5372
s.e.   0.2752  0.2564

sigma^2 = 9.155:  log likelihood = -862.93
AIC=1731.87   AICc=1731.94   BIC=1743.37
```

The fitted ARIMA (1, 1, 1) model for Fresno has an AR(1) coefficient of -0.4364 and an MA(1) coefficient of 0.5372. These indicate a negative autoregressive influence from the previous day's differenced temperature, and a positive adjustment based on the previous forecast error. This model has an AIC value of value of 1731.87 and a BIC value of 1743.37.

```r
# Model diagnostics for ARIMA (1, 1, 1) for Fresno
tsdiag(fresno_fit1)
```

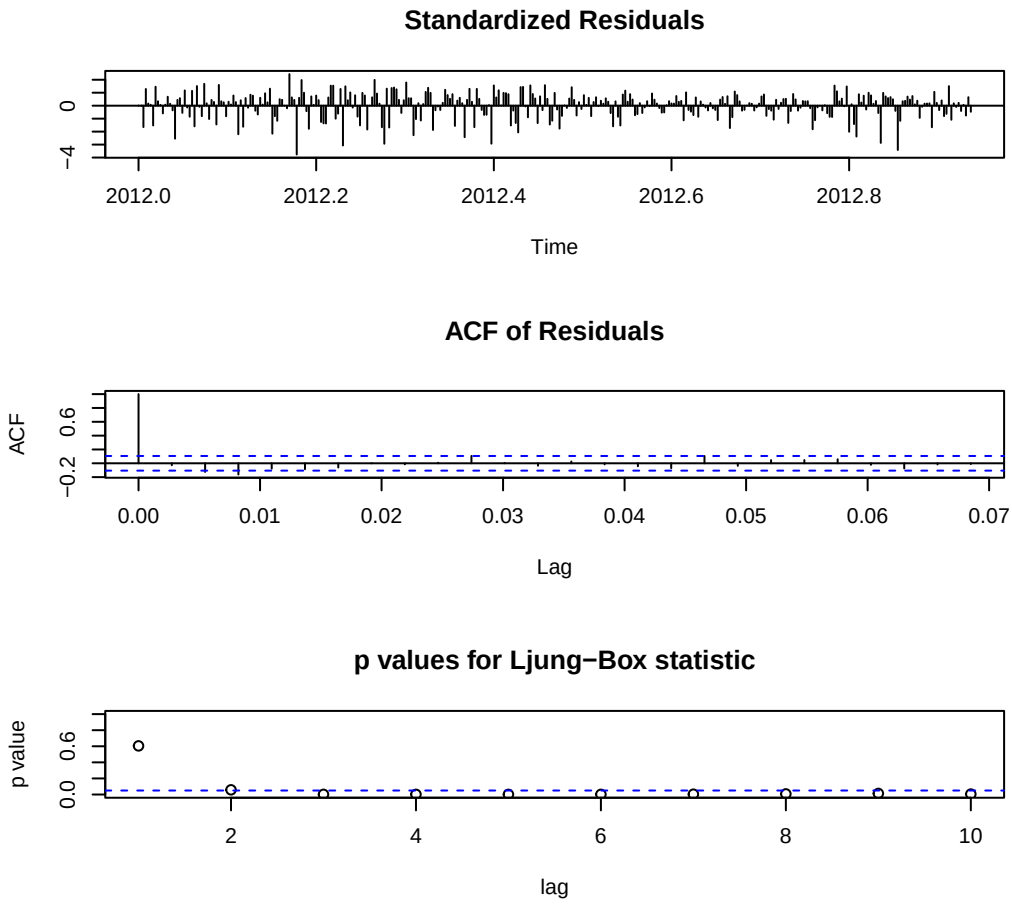


Figure 30: ARIMA (1, 1, 1) Fresno Diagnostics

The residuals diagnostics suggest this model does not adequately capture the structure of the data. The Ljung box test shows that most p-values are below 0.05, strongly suggesting significant autocorrelation remains in the residuals. This means that the model fails to fully explain the time series structure and lacks predictive adequacy.

```
# Fitting ARIMA (1, 1, 2) manually for San Diego
san_diego_fit2 <- Arima(san_diego_ts, order = c(1, 1, 2))
san_diego_fit2
```

```
Series: san_diego_ts
ARIMA(1,1,2)

Coefficients:
         ar1      ma1      ma2
```

```
      0.2149  -0.4490  -0.4200
s.e.  0.0979   0.0914   0.0749

sigma^2 = 5.199:  log likelihood = -766.21
AIC=1540.41   AICc=1540.53   BIC=1555.75
```

This ARIMA (1, 1, 2) model for San Diego displays an AR(1) of 0.214 referring to the current temperature being positively influenced by the previous day's value. The MA(2) value refers to the two lagged forecast errors which are used to adjust the model, capturing irregularities and shocks in the temperature data. This model outputs an AIC value of 1540.41 which is the lowest of all models tested and therefore is the best fitting model. The model has a residual variance (σ^2) of 5.199.

```r
# Model diagnostics for ARIMA (1, 1, 2) for San Diego
tsdiag(san_diego_fit2)
```

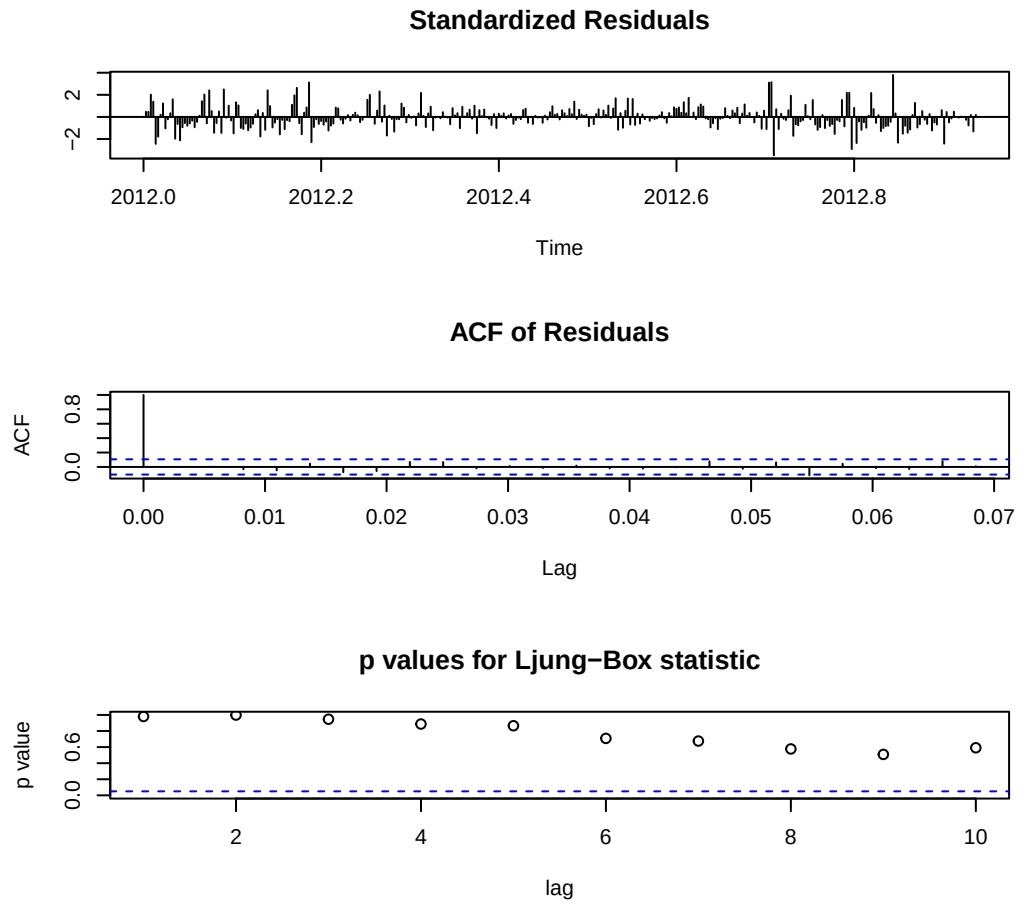


Figure 31: ARIMA (1, 1, 2) San Diego Diagnostics

The diagnostic residuals for the ARIMA (1, 1, 2) for San Diego show randomly scattered standardised residuals around zero with no obvious pattern, suggesting constant variance and no autocorrelation. The ACF of residuals shows all autocorrelations lying within the confidence bounds, indicating that the residuals behave like white noise. The Ljung-box test shows all p-values are above 0.05 which means there is no autocorrelation of residuals. These diagnostics confirm that the model captures the structure of the data well and is suitable for forecasting.

```
# Fitting ARIMA (1, 1, 2) model manually for Fresno
fresno_fit2 <- Arima(fresno_ts, order = c(1, 1, 2))
fresno_fit2
```

```
Series: fresno_ts
ARIMA(1,1,2)

Coefficients:
         ar1      ma1      ma2
      0.5273  -0.5855  -0.2738
s.e.  0.0752   0.0769   0.0589

sigma^2 = 8.111:  log likelihood = -841.98
AIC=1691.96   AICc=1692.08   BIC=1707.3
```

The ARIMA (1, 1, 2) model for Fresno shows that the current maximum temperature is influenced by both the previous day's value and the two most recent forecast errors. The AR(1) term of 0.527 indicates a positive relationship with the previous day's temperature, suggesting some persistence in temperature trends. This model yields an AIC of 1691.96, which was the lowest among the candidate models tested for Fresno. The relatively higher residual variance (σ^2) of 8.11 compared to San Diego suggests that Fresno experiences more daily temperature fluctuations.

```
# Model diagnostics for ARIMA (1, 1, 2) for Fresno
tsdiag(fresno_fit2)
```

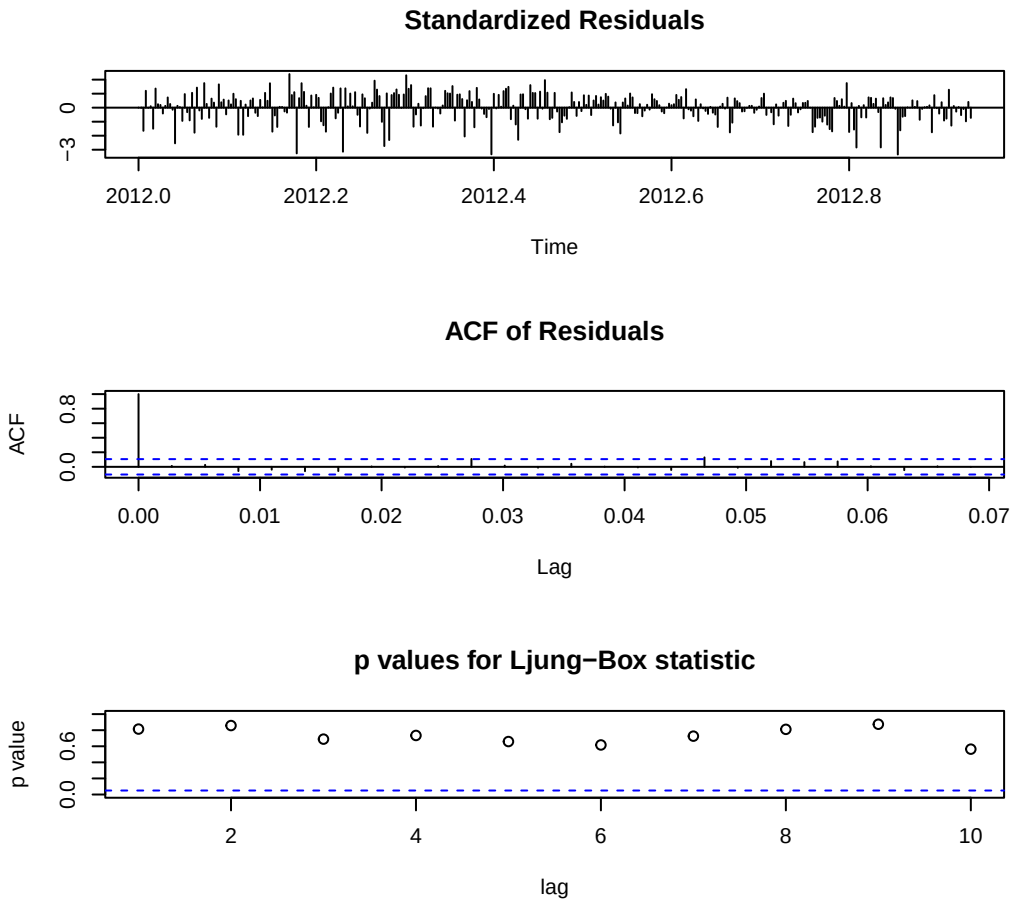


Figure 32: ARIMA (1, 1, 2) Fresno Diagnostics

The diagnostic residuals for the ARIMA (1, 1, 2) for Fresno are very similar to San Diego, showing constant variance and no autocorrelation. The ACF of residuals shows all autocorrelations lying between the confidence intervals, indicating that the residuals behave approximately like white noise. The Ljung-box test shows all p-values are above 0.05 which means there is no autocorrelation of residuals. These diagnostics confirm that the model captures the structure of the data well and is suitable for forecasting.

1. Forecasting for December 9th to December 13th

San Diego

```
# Forecasting Max Temp between Dec 9 - Dec 13 for San Diego
forecast_san_diego <- forecast(san_diego_fit2, h = 5)

# Dates for Dec 9-13
dates <- seq(ymd("2012-12-09"), ymd("2012-12-13"), by = "day")

# Actual values for comparison
actual_values <- full_data %>%
  filter(Location == "San Diego", Date %in% dates) %>%
  arrange(Date) %>%
  pull(Temperature)

# Table for Results
forecast_table <- tibble(
  Date = dates,
  Forecast = as.numeric(forecast_san_diego$mean),
  Actual = actual_values,
  Absolute_Error = abs(Forecast - Actual)
)

kable(forecast_table, caption = "San Diego Forecast vs Actual (Dec 9-13)", digits = 2)
```

Table 7: San Diego Forecast vs Actual (Dec 9–13)

Date	Forecast	Actual	Absolute_Error
2012-12-09	19.11	17.2	1.91
2012-12-10	19.23	20.0	0.77
2012-12-11	19.25	21.7	2.45
2012-12-12	19.25	17.8	1.45
2012-12-13	19.26	16.1	3.16

The table above compares the ARIMA (1, 1, 2) forecasted maximum temperatures for San Diego with the actual observed values over the period December 9th to 13th, 2012. The forecast peforms reasonably well, with absolute errors ranging from 0.77 to 3.16 degrees celsius. The smallest error occurs on December 10th, suggesting the model captured short-term dynamics effectively. The largest error on December 13th may indicate that the model shows reduced accuracy towards the end of the forecast horizon. The forecasted values remain around 19.2 degrees celsius with an increasing pattern because the ARIMA model has been extrapolated off the previous trend.

Fresno

```
# Forecasting Max Temp between Dec 9 - Dec 13 for Fresno
forecast_fresno <- forecast(fresno_fit2, h = 5)

# Dates for Dec 9-13
dates <- seq(ymd("2012-12-09"), ymd("2012-12-13"), by = "day")

# Actual values for comparison
actual_values <- full_data %>%
  filter(Location == "Fresno", Date %in% dates) %>%
  arrange(Date) %>%
  pull(Temperature)

# Table for Results
forecast_table <- tibble(
  Date = dates,
  Forecast = as.numeric(forecast_fresno$mean),
  Actual = actual_values,
  Absolute_Error = abs(Forecast - Actual)
)

kable(forecast_table, caption = "Fresno Forecast vs Actual (Dec 9-13)", digits = 2)
```

Table 8: Fresno Forecast vs Actual (Dec 9–13)

Date	Forecast	Actual	Absolute_Error
2012-12-09	15.92	14.4	1.52
2012-12-10	16.65	16.1	0.55
2012-12-11	17.04	18.3	1.26
2012-12-12	17.24	11.7	5.54
2012-12-13	17.35	16.7	0.65

The table above compares the ARIMA (1, 1, 2) forecasted maximum temperatures for Fresno with the observed values from December 9th to 13th, 2012. Overall, the model performs reasonably well, with absolute errors mostly below 1.5 degrees except for a significant spike on December 12th, where the forecasted value overestimates by over 5 degrees celsius. This large error suggests that the model struggled to capture a sharp dip in local temperature, likely due to a short-term anomaly not reflected in past trends. The forecasted values follow a smooth upward pattern, consistent with the model extrapolating recent behaviour - similar to the San Diego forecasts.

2. Forecasting for December 13th to December 17th

In order to model for December 13th to December 17th, we must first change the training data to include the data up until the 12th December and then refit the models for forecasting.

```
# Training data: up to Dec 12 (for Dec 13-17 forecast)
training_data2 <- station_data %>%
  filter(Date <= as.Date("2012-12-12"))

# Testing data: Dec 9-13 (for evaluating first forecast)
testing_data_2 <- station_data %>%
  filter(Date >= as.Date("2012-12-13"),
         Date <= as.Date("2012-12-17"))

# Separate time series data for training
san_diego_ts2 <- ts(training_data2 %>%
                      filter(Location == "San Diego") %>%
                      pull(Temperature),
                    start = c(2012, 1), frequency = 365)

fresno_ts2 <- ts(training_data2 %>%
                   filter(Location == "Fresno") %>%
                   pull(Temperature),
                 start = c(2012, 1), frequency = 365)
```

Refitting the models with new training data

```
# Fitting ARIMA (1, 1, 2) manually for San Diego
san_diego_fit3 <- Arima(san_diego_ts2, order = c(1, 1, 2))
san_diego_fit3
```

```
Series: san_diego_ts2
ARIMA(1,1,2)

Coefficients:
         ar1      ma1      ma2
      0.2097  -0.4471  -0.4210
s.e.  0.0978   0.0911   0.0747

sigma^2 = 5.19:  log likelihood = -774.89
AIC=1557.78   AICc=1557.9   BIC=1573.17
```

The refitted ARIMA (1, 1, 2) model for San Diego shows a slight insignificant change in parameter estimates from the previous fit. The AIC value has slightly increased to 1557.78, this is likely due to the added uncertainty of the extra 4 days of data entries. The model still however maintains good fit characteristics and is expected to capture recent temporal patterns in the updated data well.

```
# Fitting ARIMA (1, 1, 2) model manually for Fresno
fresno_fit3 <- Arima(fresno_ts2, order = c(1, 1, 2))
fresno_fit3
```

```
Series: fresno_ts2
ARIMA(1,1,2)

Coefficients:
         ar1      ma1      ma2
      0.5202  -0.5801  -0.2754
s.e.  0.0761   0.0778   0.0592

sigma^2 = 8.161:  log likelihood = -852.92
AIC=1713.84   AICc=1713.96   BIC=1729.23
```

The refitted ARIMA (1, 1, 2) model for Fresno displays consistent structure with a slight insignificant change in parameter estimates. The AIC value has slightly increased to 1713.84, reflecting the uncertainty from including the extra 4 days of data values. The model still maintains good fit characteristics and is suitable for forecasting.

San Diego

```
# Forecasting Max Temp between Dec 13 - Dec 17 for San Diego
forecast_san_diego2 <- forecast(san_diego_fit3, h = 5)

# Dates for Dec 9-13
dates <- seq(ymd("2012-12-13"), ymd("2012-12-17"), by = "day")

# Actual values for comparison
actual_values <- full_data %>%
  filter(Location == "San Diego", Date %in% dates) %>%
  arrange(Date) %>%
  pull(Temperature)

# Table for Results
forecast_table2 <- tibble(
  Date = dates,
```

```
  Forecast = as.numeric(forecast_san_diego2$mean),
  Actual = actual_values,
  Absolute_Error = abs(Forecast - Actual)
)

kable(forecast_table2, caption = "San Diego Forecast vs Actual (Dec 13-17)", digits = 2)
```

Table 9: San Diego Forecast vs Actual (Dec 13-17)

Date	Forecast	Actual	Absolute_Error
2012-12-13	17.65	16.1	1.55
2012-12-14	18.77	15.6	3.17
2012-12-15	19.00	15.0	4.00
2012-12-16	19.05	15.6	3.45
2012-12-17	19.06	17.2	1.86

The forecast results for San Diego from December 13th to 17th show that the ARIMA (1, 1, 2) model continues to perform reasonably well, though with slightly larger absolute errors compared to the earlier forecast window. Absolute errors range from 1.55 to 4.00 degrees celsius, with the largest discrepancy occurring on December 15th, suggesting that the model struggled to adjust to short-term fluctuations in this period. The forecasted values generally follow the actual trend, maintaining values close to 19 degrees celsius, reflecting the model projecting forward the established pattern of the training data.

Fresno

```
# Forecasting Max Temp between Dec 13 - Dec 17 for Fresno
forecast_fresno2 <- forecast(fresno_fit3, h = 5)

# Dates for Dec 9-13
dates <- seq(ymd("2012-12-13"), ymd("2012-12-17"), by = "day")

# Actual values for comparison
actual_values <- full_data %>%
  filter(Location == "Fresno", Date %in% dates) %>%
  arrange(Date) %>%
  pull(Temperature)

# Table for Results
forecast_table2 <- tibble(
  Date = dates,
  Forecast = as.numeric(forecast_fresno2$mean),
```

```
  Actual = actual_values,
  Absolute_Error = abs(Forecast - Actual)
)

kable(forecast_table2, caption = "Fresno Forecast vs Actual (Dec 13-17)", digits = 2)
```

Table 10: Fresno Forecast vs Actual (Dec 13-17)

Date	Forecast	Actual	Absolute_Error
2012-12-13	11.75	16.7	4.95
2012-12-14	13.62	12.2	1.42
2012-12-15	14.60	10.6	4.00
2012-12-16	15.10	15.6	0.50
2012-12-17	15.36	18.9	3.54

The Fresno forecast from December 13th to 17th using ARIMA (1, 1, 2) model shows mixed performance, with absolute errors ranging from 0.5 degrees celsius to 4.95 degrees celsius. The forecast for December 13th significantly underestimates the actual temperature, producing the largest error, which suggests that the model did not capture a short-term spike. Similarly, the model underestimated temperatures again on December 17th and overestimates temperatures on December 15th. Extreme weather changes could have caused this short-term fluctuation which the model cannot predict through trends. The forecast on December 16th is highly accurate, with only a 0.5 degree celsius deviation. Overall, the model captures the trend well but appears less reliable for sharp or irregular temperature changes for this forecast window.

(d)

Comparing Predictions for December 13th

```
# Building comparison table for December 13th predictions
comparison_table <- tibble(
  Location = rep(c("San Diego", "Fresno"), each = 3),
  Model = rep(c("Kriging", "ARIMA (to Dec 8)", "ARIMA (to Dec 12)"), times = 2),
  Prediction = c(16.00891, 19.26, 17.65, 11.42300, 17.35, 11.75),
  Actual_Temp = c(16.1, 16.1, 16.1, 16.7, 16.7, 16.7),
  Absolute_Error = round(abs(Prediction - Actual_Temp), 2)
)

# Display table
kable(comparison_table, caption = "Model Comparison for December 13th, 2012", digits = 2)
```

Table 11: Model Comparison for December 13th, 2012

Location	Model	Prediction	Actual_Temp	Absolute_Error
San Diego	Kriging	16.01	16.1	0.09
San Diego	ARIMA (to Dec 8)	19.26	16.1	3.16
San Diego	ARIMA (to Dec 12)	17.65	16.1	1.55
Fresno	Kriging	11.42	16.7	5.28
Fresno	ARIMA (to Dec 8)	17.35	16.7	0.65
Fresno	ARIMA (to Dec 12)	11.75	16.7	4.95

The table above displays a comparison of the model's predictive performances in forecasting maximum temperature in San Diego and Fresno on December 13th, 2012. For San Diego, the Kriging model outperformed both ARIMA variants, likely due to its use of spatial and elevation covariates, which are relevant in a coastal location where geography significantly affects temperature. The Kriging model had an absolute error of 0.09 which was significantly better than both ARIMA models that forecasted San Diego's temperature on December 13th with an absolute error of 3.16 and 1.55 respectively.

On the other hand, Fresno was best forecasted by the ARIMA model trained up to December 8th which achieved the lowest absolute error of 0.65. This suggests that local temporal patterns were more predictive than spatial features, possibly due to Fresno's temperature being more stable and less influenced by spatial variance compared to coastal areas like San Diego, making the time series model a better fit. Interestingly, the ARIMA model trained up to December 12th, which included more recent data, performed worse with an absolute error of 4.95. This may reflect overfitting or sensitivity to short-term fluctuations in the additional data.

In order to improve future predictions, hybrid models could be integrated that include both spatial (e.g. Kriging) and temporal (e.g. ARIMA) components in order to capture a broader range of variability. The inclusion of additional covariates such as humidity, wind speed or cloud cover could enhance the model's explanatory power if this data can be gathered. For the ARIMA models - incorporating data from 2011 would improve the predictive performance of the model because seasonal patterns, especially in December would be more informed. In this model, only data up until December 8th or 12th was used to inform predictions and trends. Fitting models on multiple years would give ARIMA better structure for learning long-term trends and periodicity. Switching to a Seasonal ARIMA (SARIMA) model could then be more appropriate in order to account for patterns repeating over fixed intervals such as 12 months or 365 days.