James Lewis

UNIVERSITY OF EXETER

Assessment

Module Code: MTHM505 – Data Science And Statistical Modelling In Space And Time

---

Declaration of AI Assistance

I have used OpenAI's ChatGPT tool in creating this report.
AI-supported/AI-integrated use is permitted in this assessment. I acknowledge the following uses of GenAI tools in this assessment:

- Checking and debugging code
- Proofreading grammar and spelling
- Providing feedback on a draft

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

1.1 Part A: Cleaning and Spatial Overview

```
# Data
kuroshio100 <- read.csv("~/GitHub/university-projects/Modelling in Space and Time/In progress/k

# Summarise NA counts
# colSums(is.na(kuroshio100))
# Across all columns apart from `date` and `id`, there are 1557 missing values
# Remove rows with missing essential spatial data
kuroshio100_clean <- kuroshio100 %>% drop_na(lon, lat)

# Get country borders (as 'sf' object)
world <- ne_countries(scale = 50, returnclass = "sf")
kuroshio_sf <- st_as_sf(kuroshio100_clean, coords = c("lon", "lat"), crs = 4326)

ggplot() +
  geom_sf(data = world, fill = "grey95", color = "black") +  # Base map
  geom_sf(data = kuroshio_sf, aes(color = sst), size = 2, alpha = 0.9) +
  scale_color_viridis_c(name = "SST (°C)", option = "C", direction = -1) +
  coord_sf(xlim = c(140, 150), ylim = c(30, 40), expand = FALSE) +
  # Zoom on Kuroshio region
  labs(
    title = "Sea Surface Temperature - Kuroshio Current",
    x = "Longitude", y = "Latitude"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    legend.position = "right"
  )
```

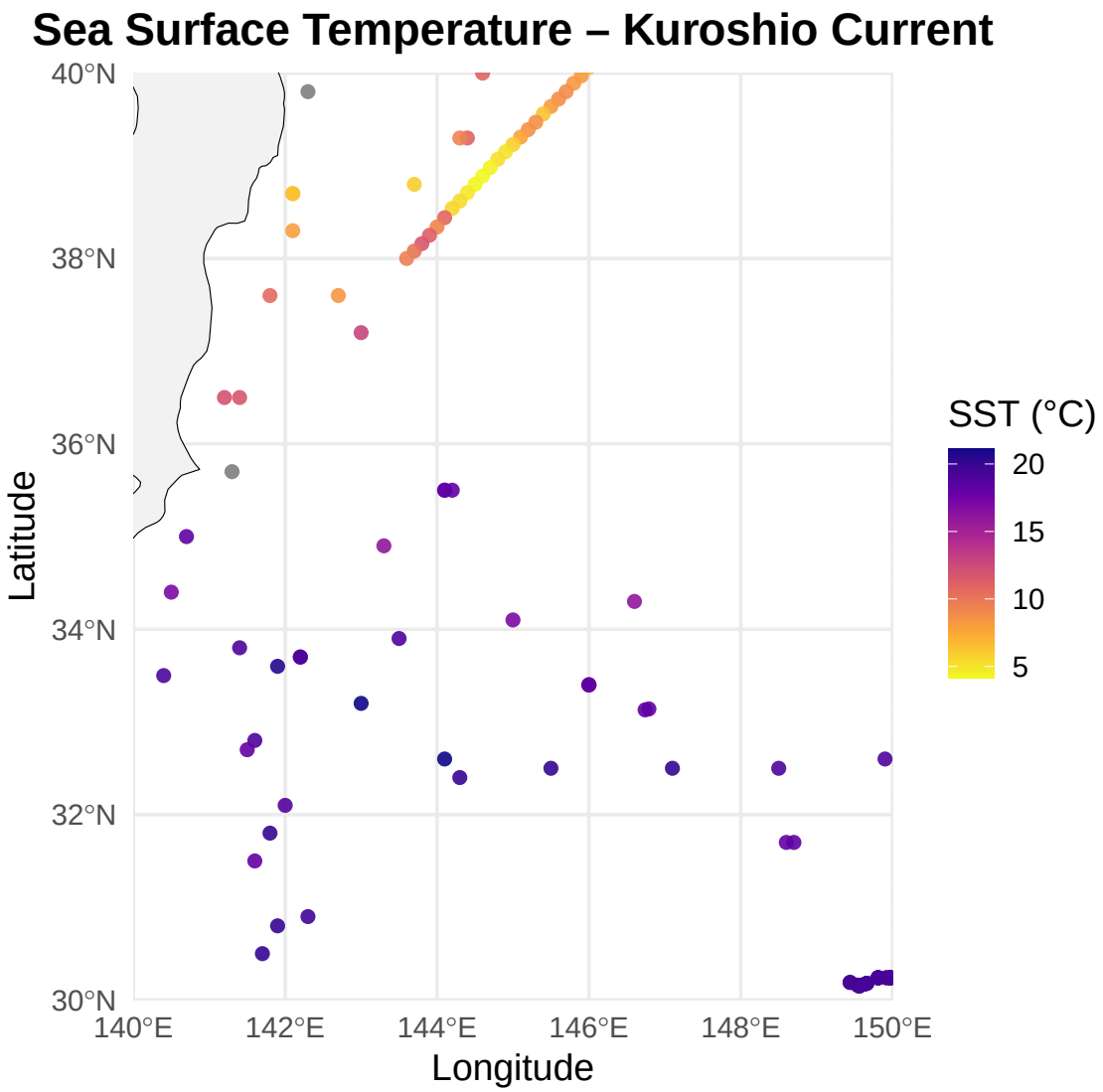


Figure 1: Spatial distribution of Sea Surface Temperature (SST) observations collected on 1–2 January 1996 in the Kuroshio Current region. Each point represents an individual measurement; colour denotes temperature, with warmer SSTs concentrated in the north-east band.

The dataset `kuroshio100.csv` contains 100 Sea Surface Temperature (SST) observations recorded in January 1996 along the Kuroshio Current system in the western Pacific Ocean. Preliminary inspection revealed three rows with missing spatial coordinates (`lon` or `lat`), which were removed to ensure compatibility with spatial analysis function `as.geodata()`.

This resulted in 97 complete observations spanning a broad geographical area, with longitudes ranging from approximately 140°E to 150°E and latitudes from 30°N to 40°N. These cleaned observations were retained for further exploratory and model-based spatial analysis.

Figure 1 displays the spatial distribution of SST observations. The plot confirms a substantial latitudinal spread and a wide SST range across the domain. Warmer SST values are observed primarily in the southeast portion of the domain, while cooler temperatures dominate the northwest, suggesting a clear spatial trend in

SST that motivates the use of geostatistical modelling in subsequent sections.

1.2 Part B: Spatial Data Partitioning for Validation

To enable independent model validation, five spatial locations were randomly withheld from the dataset and reserved as a test set. These locations will be used to evaluate the predictive accuracy and uncertainty quantification of kriging and Gaussian process models in Parts C–E.

A fixed random seed (set.seed(444)) was used to ensure reproducibility. The selection was drawn from the cleaned dataset, ensuring no missing coordinates or SST values. The remaining 92 locations form the training dataset for model fitting.

```
set.seed(444)  # For reproducibility

# Using the cleaned dataset to ensure we dont chose missing values.
# 5 random points
test_points <- kuroshio100_clean %>%
  sample_n(5)

# Display their information
test_points %>%
  select(id, lon, lat, sst)
```

```
         id    lon   lat  sst
1      MQWU 142.10 38.70  6.5
2 49  16760 145.40 39.56  6.5
3     21573 149.56 30.15 19.3
4     LATI4 140.70 35.00 18.2
5     3FFJ4 142.10 38.30  8.0
```

The training set was constructed by removing the test points based on their full spatial and response information:

```
# Create training dataset (excluding test points)
kuroshio_train <- anti_join(kuroshio100, test_points, by = c("id", "lon", "lat"
                                                             , "sst"))

# Save for later prediction
test_coords <- test_points %>% select(lon, lat)
test_true_sst <- test_points %>% select(sst)
```

This partitioning resulted in:

Training set: 92 spatial observations used for model fitting

Test set: 5 withheld observations used exclusively for validation

These test points are held constant throughout the modelling pipeline and will be used to evaluate the out-of-sample performance of both kriging and Gaussian process models. This ensures fair comparison across modelling approaches and supports robust validation of predictive uncertainty.

1.3 Part C: Empirical Variogram and Spatial Correlation Structure

```
library(geoR)

# Convert training dataset into a geodata object
# Jitter duplicated coordinates very slightly
kuro_geo_train <- jitterDupCoords(
  as.geodata(kuroshio_train, coords.col = c("lon", "lat"), data.col = "sst"),
  max = 1e-5
)
```

```
as.geodata: 19 replicated data locations found.
 Consider using jitterDupCoords() for jittering replicated locations.
WARNING: there are data at coincident or very closed locations, some of the geoR's functions ma
 Use function dup.coords() to locate duplicated coordinates.
 Consider using jitterDupCoords() for jittering replicated locations
```

During conversion to geodata format, 19 observations were found to share identical coordinates. This is problematic for geostatistical modelling, as duplicate locations lead to ill-defined variogram structures and singular covariance matrices. To address this, we applied a minimal spatial jitter using `jitterDupCoords()` with `max = 1e-5`, introducing negligible noise (~0.00001 degrees) to resolve ties while preserving the spatial pattern.

1.3.0.1 Empirical Variogram Estimation

```
sample_vario <- variog(kuro_geo_train, option='bin')
```

```
variog: computing omnidirectional variogram
```

```
par(mar=c(4,4,2,2))
plot(kuro_geo_train, pch = 19)
```

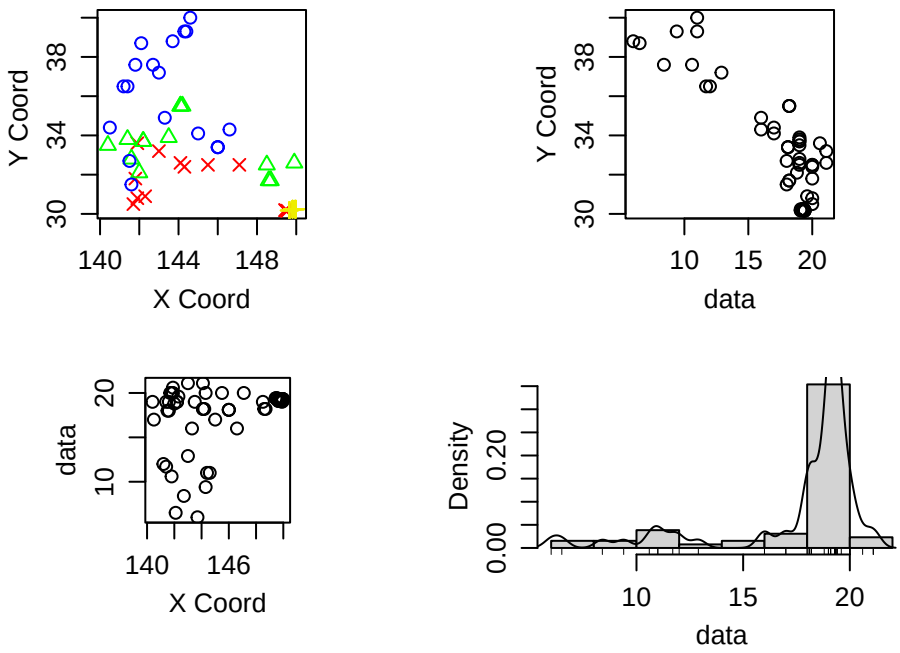


Figure 2: Exploratory Plot

1.3.0.1.1 Inclusion of a Latitudinal Trend Term

```
ggplot(kuroshio_train, aes(x = lat, y = sst)) +
  geom_point() +   geom_smooth(method = "lm", colour = "red", se = FALSE) +
  labs(title = "Latitudinal Trend in Sea Surface Temperature",
  x = "Latitude", y = "SST (°C)") +
  theme_minimal()
```

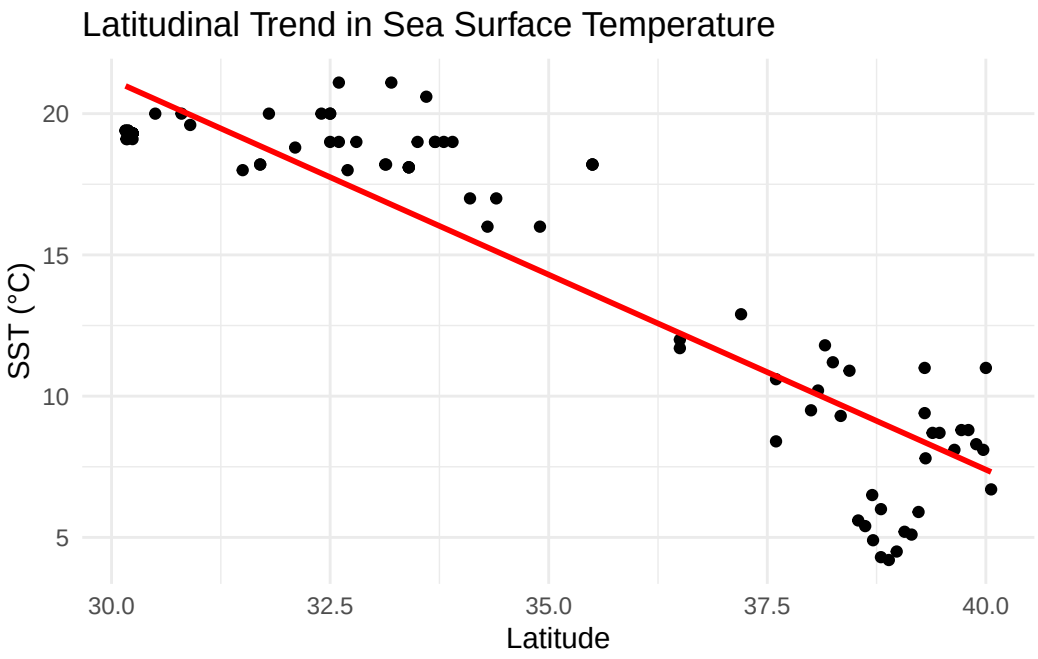


Figure 3: Latitiude against SST

Exploratory analysis (Figure 2) revealed a clear negative relationship between latitude and sea surface temperature (SST): as latitude increases, SST decreases. This represents a large-scale spatial trend, violating the second-order stationarity assumption of a constant mean.

To address this, a linear trend in latitude was incorporated into variogram estimation via the `trend = ~ lat` argument in `variog()`. This approach removes deterministic variation associated with latitude and isolates residual spatial correlation. **Figure 3** visualises this latitudinal trend.

```
# Empirical variogram with binning
emp_variog_5 <- variog(kuro_geo_train, option = "bin", max.dist = 5, trend = ~ lat)
```

```
variog: computing omnidirectional variogram
```

```
emp_variog_7 <- variog(kuro_geo_train, option = "bin", max.dist = 7, trend = ~ lat)
```

```
variog: computing omnidirectional variogram
```

```
emp_variog_10 <- variog(kuro_geo_train, option = "bin", max.dist = 10, trend = ~ lat)
```

```
variog: computing omnidirectional variogram
```

To assess the spatial dependence in SST, the semi-variance is computed as:

$$\gamma(h) = \frac{1}{2N(h)} \sum_{\substack{i,j: \\ \|s_i - s_j\| \approx h}} \left(z(s_i) - z(s_j)\right)^2$$

where:

- $N(h)$ is the number of location pairs separated by distance h,

- s_i and s_j are spatial coordinates of observations,
- $z(s_i)$ is the SST value at location s_i.

The semi-variance $\gamma(h)$ increases with distance h if spatial correlation is present.

```
library(tibble)
library(dplyr)

variog_df <- bind_rows(
  tibble(dist = emp_variog_5$u, semivar = emp_variog_5$v, version = "max.dist = 5"),
  tibble(dist = emp_variog_7$u, semivar = emp_variog_7$v, version = "max.dist = 7"),
  tibble(dist = emp_variog_10$u, semivar = emp_variog_10$v, version = "max.dist =10")
)

ggplot(variog_df, aes(x = dist, y = semivar)) +
  geom_point(size = 2, colour = "black") +
  geom_line(colour = "darkred", linewidth = 1) +
  facet_wrap(~ version, ncol = 1, scales = "free_x") +
  labs(
    title = "Comparison of Empirical Variograms",
    x = "Distance (spatial units)",
    y = "Semi-variance"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", size = 14, hjust = 0.5),
    axis.title = element_text(face = "bold"),
    strip.text = element_text(face = "bold", size = 12),
    panel.grid.major = element_line(color = "grey85"),
    panel.grid.minor = element_blank()
  )
```

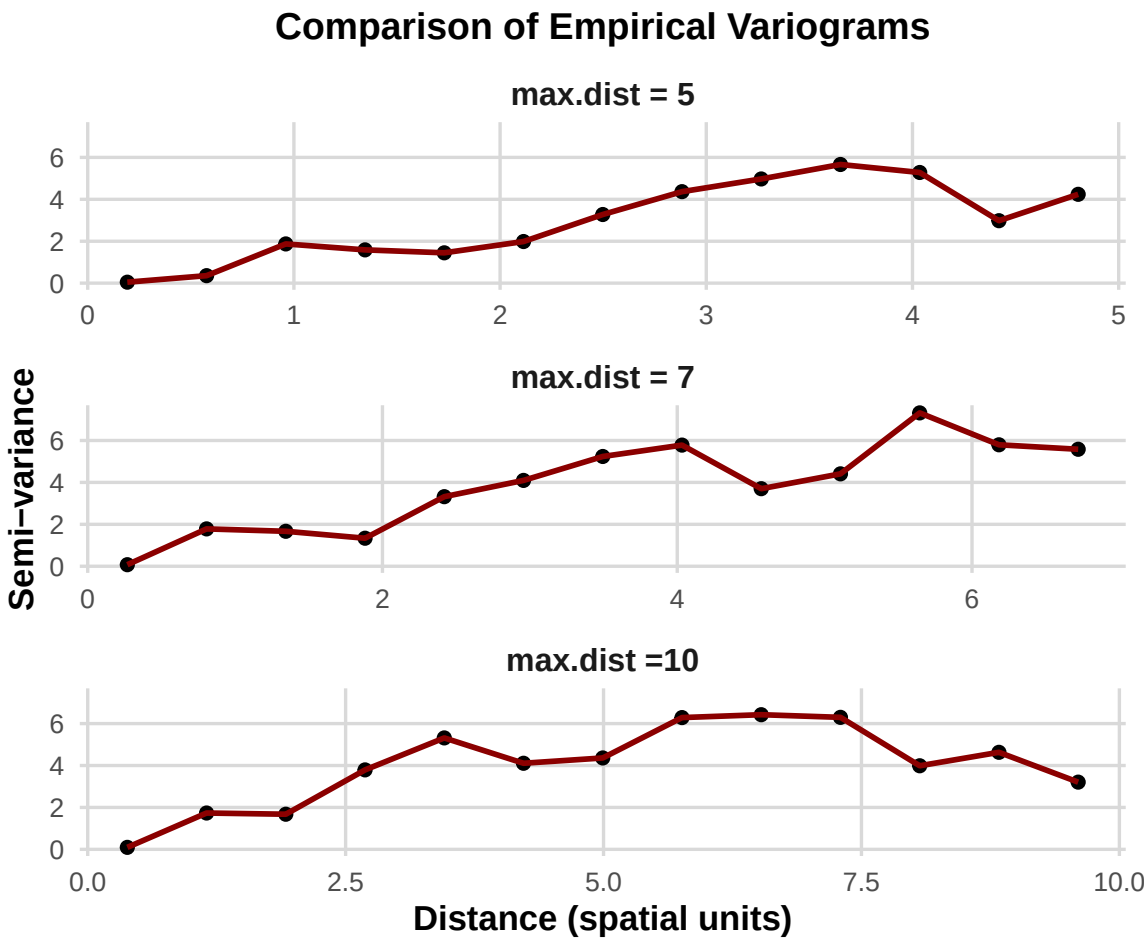


Figure 4: Empirical variograms computed using three different maximum distance thresholds. The max.dist = 7 version was selected for model fitting due to reduced instability in the tail while preserving the spatial structure.

To assess spatial autocorrelation in SST, empirical variograms were computed using residuals from the latitudinal trend. This allows estimation of the underlying stochastic spatial structure.

Figure 4 displays the three empirical variograms, with `max.dist = 7` selected for further analysis.

Three `max.dist` values were tested — 5, 7, and 10 — following the common guideline of using two-thirds of the maximum interpoint distance. Each variogram exhibited the expected monotonic increase, but differed in stability and tail behaviour:

- `max.dist = 10`: Covered full spatial extent but showed instability in the tail due to sparse bins
- `max.dist = 5`: Offered stable estimates but truncated longer-range structure
- `max.dist = 7`: Balanced stability and range; selected for parametric fitting

Bin counts for the selected variogram were generally robust (e.g., >60 observations per bin), supporting its use for weighted least squares fitting.

1.3.0.1.2 Nugget Effect Justification

The empirical variogram did not pass through the origin, indicating a non-zero intercept (nugget). This supports inclusion of a nugget effect in model fitting, which may reflect:

- Instrumentation noise
- Sub-grid-scale SST variation
- Small-scale oceanographic processes

1.3.0.1.3 Final Selection

The detrended empirical variogram with `max.dist = 7` was selected for parametric model fitting. It offers a stable and interpretable residual spatial structure while maintaining adequate coverage across spatial lags.

1.3.0.2 Fitting Parametric Variogram Models

```
#  Fit Parametric Variogram Models
# Exponential model
fit_exp <- variofit(
  emp_variog_7,
  cov.model = "exponential",
  ini.cov.pars = c(1, 1),
  nugget = 0.1,
  weights = "equal"
)
```

```
variofit: covariance model used is exponential
variofit: weights used: equal
variofit: minimisation function used: optim
```

```
# Gaussian model
fit_gau <- variofit(
  emp_variog_7,
  cov.model = "gaussian",
  ini.cov.pars = c(1, 1),
  nugget = 0.1,
  weights = "equal"
)
```

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

```
# Matérn model (kappa = 1.5)
fit_mat1 <- variofit(
  emp_variog_7,
  cov.model = "matern",
  kappa = 1.5,
  ini.cov.pars = c(2, 1),
  nugget = 0.5,
  weights = "equal"
)
```

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

```
fit_mat2 <- variofit(
  emp_variog_7,
  cov.model = "matern",
  kappa = 1.5,
  ini.cov.pars = c(1.5, 0.8),
  nugget = 0.3,
  weights = "equal"
)
```

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

Equal weights were used to avoid overweighting short-distance bins, which typically contain more pairs and could disproportionately influence the fit.

```
# Plot empirical variogram
plot(emp_variog_7,
     main = "Fitted Variogram Models (max.dist = 7, trend = ~lat)",
     xlab = "Distance (spatial units)", ylab = "Semi-variance")

# Add fitted models
lines(fit_exp, col = "blue", lwd = 2)
lines(fit_gau, col = "forestgreen", lwd = 2)
lines(fit_mat1, col = "purple", lwd = 2, lty = 2)
lines(fit_mat2, col = "darkorange", lwd = 2, lty = 3)

legend("bottomright",
       legend = c("Exponential", "Gaussian", "Matérn (v1)", "Matérn (v2)"),
       col = c("blue", "forestgreen", "purple", "darkorange"),
       lty = c(1, 1, 2, 3),
       lwd = 2)
```

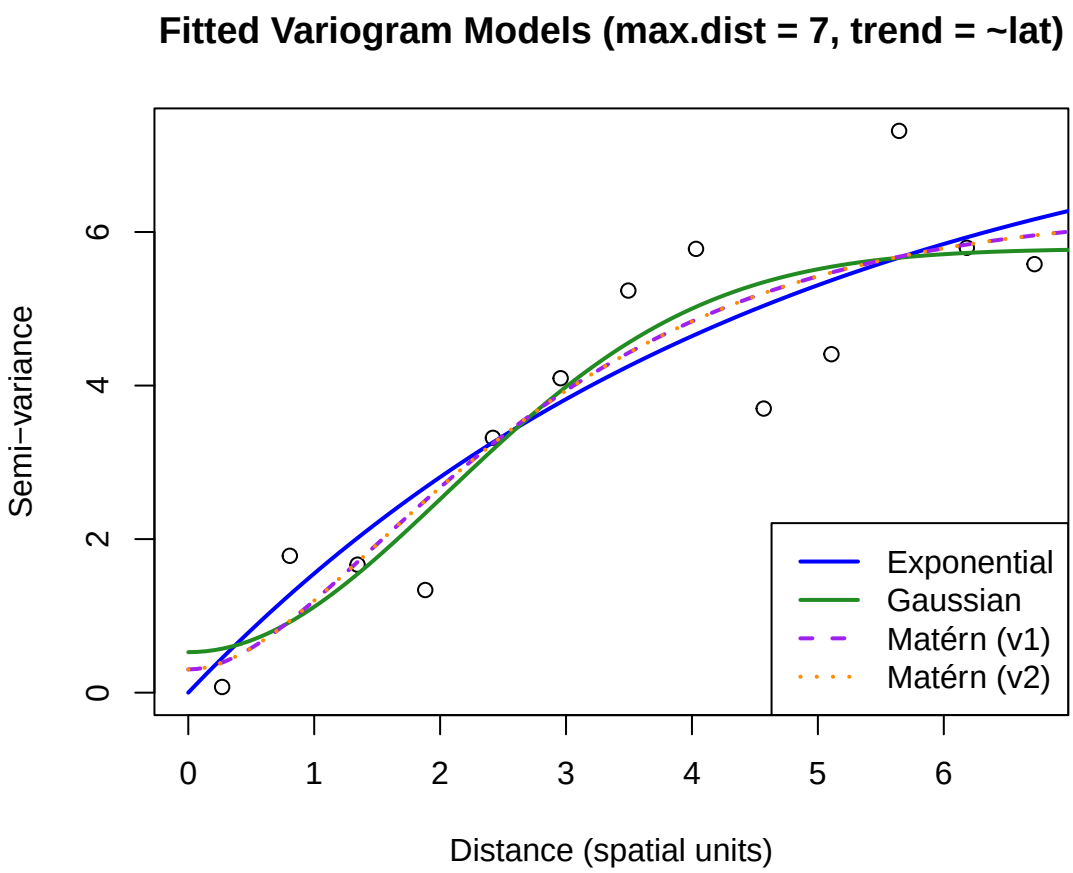


Figure 5: Parametric variogram models (Exponential, Gaussian, Matérn) fitted to the empirical variogram with max.dist = 7 and linear latitudinal trend removed. The Matérn model offered the best fit and lowest residual sum of squares.

```
# Residual sums of squares
fit_exp$value
```

```
[1] 10.41813
```

```
fit_gau$value
```

```
[1] 9.865984
```

```
fit_mat1$value
```

```
[1] 9.950673
```

```
fit_mat2$value
```

```
[1] 9.950673
```

Parametric variogram models were fitted to the empirical variogram computed with `max.dist = 7`, incorporating a linear trend in latitude. The trend was removed during empirical estimation using `trend = ~`

`lat`, ensuring that the fitted models reflect residual spatial dependence only.

Three covariance models were tested using the `variofit()` function:

- **Exponential**: models rough sample paths.
- **Gaussian**: assumes strong local correlation and very smooth sample paths.
- **Matérn** ($\kappa = 1.5$): offers intermediate smoothness between the exponential and Gaussian, balancing flexibility and interpretability.

All models assume:

- **Isotropy** (dependence only on Euclidean distance)
- **Second-order stationarity** (constant variance)
- A **non-zero nugget**, justified by the empirical variogram's non-zero intercept

Each model was fitted via weighted least squares. Initial parameter guesses were informed by visual inspection of the detrended empirical variogram.

Although RSS values were close, the **Gaussian model was selected** for kriging due to its slightly lower RSS and excellent alignment with the empirical structure. The Matérn model also performed well and may offer greater flexibility in Bayesian implementations (Part E), where smoothness parameters can be tuned explicitly.

1.3.0.3 Model Parameters and Interpretation

```
# Exponential
params_exp <- fit_exp$cov.pars
nugget_exp <- fit_exp$nugget

# Gaussian
params_gau <- fit_gau$cov.pars
nugget_gau <- fit_gau$nugget

# Matérn
params_mat <- fit_mat1$cov.pars
nugget_mat <- fit_mat1$nugget

# Create parameter summary table
param_table <- data.frame(
  Model = c("Exponential", "Gaussian", "Matérn (  = 1.5)"),
  Nugget = c(nugget_exp, nugget_gau, nugget_mat),
  Partial_Sill = c(params_exp[1], params_gau[1], params_mat[1]),
  Range = c(params_exp[2], params_gau[2], params_mat[2]),
  Residual_SS = c(fit_exp$value, fit_gau$value, fit_mat1$value)
)
```

Model	Nugget (τ^2)	Partial Sill (σ^2)	Range (ϕ)	Residual SS
Exponential	0.000	8.113405	4.709156	10.418129
Gaussian	0.5272606	5.255761	2.897214	9.865984

Model	Nugget (τ^2)	Partial Sill (σ^2)	Range (ϕ)	Residual SS
Matérn ($\kappa = 1.5$)	0.3024505	5.997264	1.467733	9.950673

Parametric Variogram Fitting and Selection

The **exponential model** showed the highest residual sum of squares (10.42) and lacked a nugget effect, underestimating short-scale variability. The **Gaussian model** achieved the lowest residual SS (9.87), with a moderate nugget (0.53), and exhibited a smooth, consistent fit across all distances. The **Matérn model** performed similarly (RSS = 9.95) but with a shorter effective range.

1.3.0.4 Model Selection for Kriging

Although all three models captured the empirical structure reasonably well, the **Gaussian model was selected** for spatial prediction due to:

- The **lowest residual error**
- A **realistic and interpretable nugget–sill–range combination**
- Theoretical support for smooth spatial processes in oceanography

1.3.0.5 Ordinary Kriging and Prediction

Using the fitted Gaussian model, ordinary kriging was performed at five randomly withheld test locations. The kriging model assumed:

- A constant mean (after trend removal)
- The fitted parameters from the Gaussian model
- Second-order stationarity and isotropy

Predictions were compared to the observed SST values at test locations, and residuals were calculated:

```
# Kriging prediction at 5 withheld locations
kriged <- krige.conv(
  geodata = kuro_geo_train,
  locations = test_coords,
  krige = krige.control(
    cov.model = "gaussian",
    cov.pars = fit_gau$cov.pars,
    nugget = fit_gau$nugget
  )
)
```

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

```
test_results <- test_coords %>%
  mutate(
    observed_sst = test_true_sst$sst,
    predicted_sst = kriged$predict,
```

```
    kriging_var = kriged$krige.var,
    residual = observed_sst - predicted_sst
  )
```

```
# Visualise prediction accuracy
ggplot(test_results, aes(x = observed_sst, y = predicted_sst)) +
  geom_point(size = 3) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", colour = "red") +
  labs(
    title = "Observed vs Predicted SST at Withheld Locations",
    x = "Observed SST (°C)",
    y = "Predicted SST (°C)"
  ) +
  theme_minimal(base_size = 13)
```

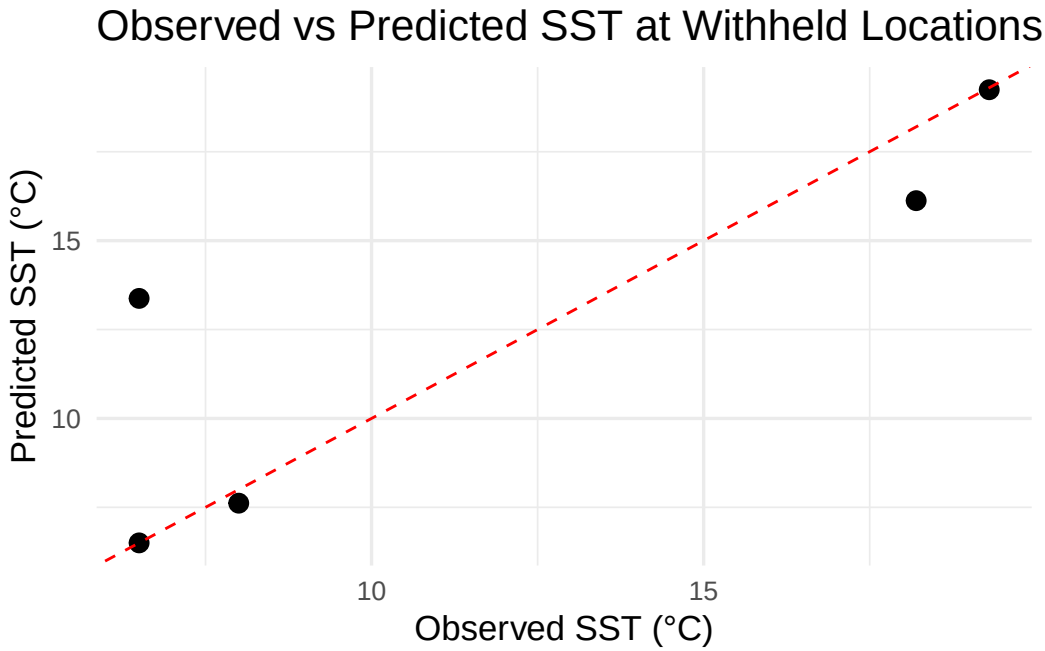


Figure 6: Observed vs predicted sea surface temperature (SST) at five withheld locations using ordinary kriging with the fitted Matérn model.

As shown in **Figure 6**, predicted SST values closely aligned with observed values, with four of five points falling near the 1:1 line. One outlier reflected a higher kriging variance, illustrating the model's ability to express uncertainty at less well-supported locations.

1.3.0.6 Model Evaluation: Gaussian Kriging

A Gaussian variogram model was fitted to the detrended empirical variogram (trend ~ latitude) using weighted least squares, and was selected based on its superior fit to the residual spatial structure. Ordinary kriging was then performed at five randomly withheld test locations to assess predictive performance, and model diagnostics were generated via leave-one-out cross-validation (LOOCV).

Table 2 below summarises observed and predicted SSTs, residuals, and kriging variances. The model showed:

- High accuracy at most locations, with **three residuals below 0.5 °C**
- One large residual (−6.88 °C), which corresponded to the **largest kriging variance** (1.505), demonstrating that the model appropriately reflected increased uncertainty
- Overall **bounded kriging variances**, mostly below 1, indicating moderate prediction confidence across the region

Performance metrics:

- **Root Mean Squared Error (RMSE)**: **3.22 °C**
- **Mean Absolute Error (MAE)**: **1.88 °C**

```
# Perform LOOCV
xv.kriging <- xvalid(kuro_geo_train, model = fit_gau)
```

```
xvalid: number of data locations       = 65
xvalid: number of validation locations = 65
xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
xvalid: end of cross-validation
```

```
# Plot residuals
par(mfrow = c(3, 2), mar = c(4, 2, 2, 2))
plot(xv.kriging, error = TRUE, std.error = FALSE, pch = 19)
```

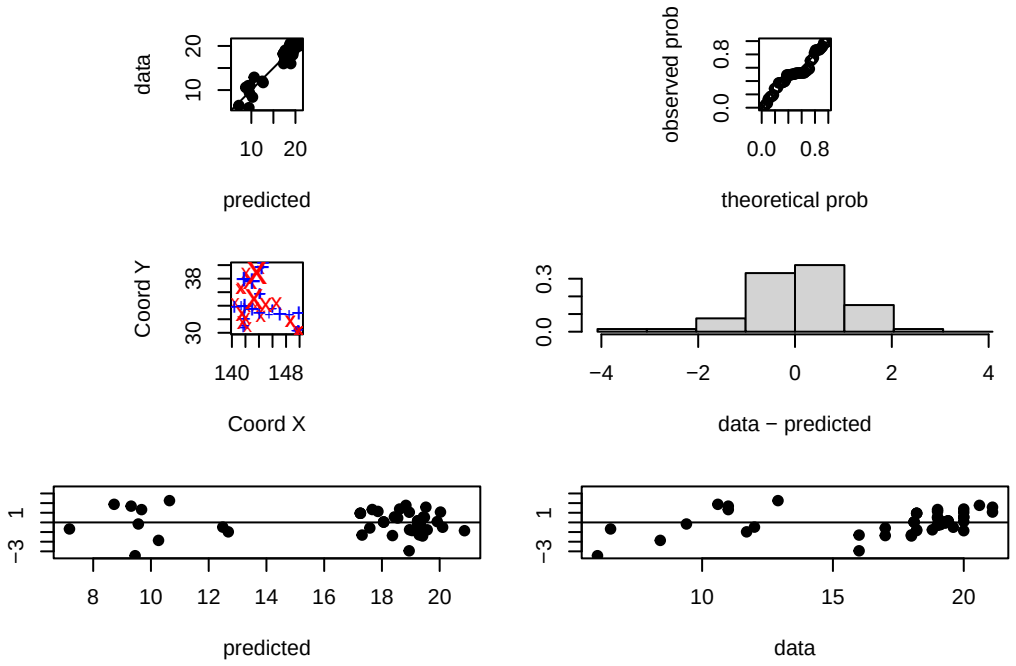


Figure 7: Leave-one-out cross-validation (LOOCV) residual diagnostics for the Gaussian kriging model. The residual plot suggests minimal bias and generally well-aligned predictions.

Table 2: Observed and predicted sea surface temperatures (SST) at five withheld test locations using Gaussian kriging. Residuals and kriging variances quantify spatial uncertainty and prediction error.

lon	lat	Observed SST (°C)	Predicted SST (°C)	Residual (°C)	Kriging Variance
142.10	38.70	6.5	6.50	0.00	0.000
145.40	39.56	6.5	13.38	-6.88	1.505
149.56	30.15	19.3	19.24	0.06	0.573
140.70	35.00	18.2	16.12	2.08	0.855
142.10	38.30	8.0	7.62	0.38	0.751

1.3.0.7 LOOCV Diagnostics

Figure 7 shows diagnostic plots for leave-one-out cross-validation using the fitted Gaussian model. The results support the assumptions of normality and spatial stationarity:

- The **Q–Q plot** and histogram show residuals are approximately normally distributed and centred around zero, indicating minimal bias. There is some deviance in the mid-section, suggesting some uncaptured trends.
- **Residual vs predicted** and **residual vs observed** plots show no clear heteroscedasticity or systemic patterns
- The **spatial residual map** (bottom-left) shows no regional clustering of high or low residuals, suggesting residuals are spatially uncorrelated

Residuals mostly fall within ±2 standard deviations, indicating appropriate predictive spread and no major violations of kriging assumptions.

```
# Compute RMSE and MAE
rmse <- sqrt(mean(test_results$residual^2))
mae <- mean(abs(test_results$residual))

# Summary table
library(knitr)

results_table <- test_results %>%
  mutate(
    `Observed SST (°C)` = round(observed_sst, 2),
    `Predicted SST (°C)` = round(predicted_sst, 2),
    `Residual (°C)` = round(residual, 2),
    `Kriging Variance` = round(kriging_var, 3)
  ) %>%
  select(lon, lat, `Observed SST (°C)`, `Predicted SST (°C)`, `Residual (°C)`,
         `Kriging Variance`)

kable(results_table, format = "latex", booktabs = TRUE,
      caption = "Observed and predicted SST values at five withheld locations
      using the Gaussian variogram model.")
```

1.3.0.8 Conclusion

The Gaussian kriging model effectively captured residual SST spatial structure and provided credible uncertainty estimates. Despite a single high-error location, the model exhibited **strong overall predictive accuracy**, with errors in line with the kriging variances.

1.4 Part D: Gaussian Process via Maximum Likelihood

1.4.0.1 Model Setup and Fitting

To validate and compare our previous kriging model, we fitted a spatial Gaussian Process (GP) model using **maximum likelihood estimation (MLE)** via the `likfit()` function from the `geoR` package. Unlike the weighted least squares method used in Part C, MLE estimates parameters by directly maximising the likelihood of the full spatial model.

We aimed to fit the Matérn covariance model with $\kappa = 1.5$ to match the best-performing model from Part C. However, fixing this smoothness parameter and including a spatial trend caused convergence failures due to non-positive definite covariance matrices - a known limitation when using `likfit()` with fixed κ and trend terms.

Instead, we fitted a simplified GP model with the following structure:

- **Covariance model**: Matérn with $\kappa = 0.5$ (equivalent to the **exponential** model)
- **Trend**: linear in latitude (`trend = ~ lat`)
- **Estimated parameters**: partial sill, range, nugget (all inferred via MLE)

```
# Fit spatial GP model via MLE using default exponential covariance

likfit_exp <- likfit(
  geodata = kuro_geo_train,
  trend = ~ lat,
  ini.cov.pars = c(5, 2),
  nugget = 0.3,
  messages = TRUE
)
```

```
---------------------------------------------------------------
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.
```

```
#likfit_exp
```

This model converged successfully and provided interpretable parameter estimates. While the covariance structure differs slightly from the model used in Part C, this approach provides a valid MLE-based comparison

Table 3: Parameter estimates from the maximum likelihood Gaussian Process model using an exponential covariance (Matérn with $\kappa = 0.5$) and a linear trend in latitude.

Parameter	Estimate
Intercept (β_0)	59.1480
Latitude Slope (β_1)	-1.2535
Nugget (τ^2)	0.0043
Partial Sill (σ^2)	3.1730
Range (ϕ)	1.3448
Practical Range (cor = 0.05)	4.0285
Log-Likelihood	-53.8600

using the same underlying data and trend specification.

The fitted model yielded the following parameter estimates:

```
library(knitr)

mle_params <- data.frame(
  Parameter = c(
    "Intercept ( )",
    "Latitude Slope ( )",
    "Nugget ( ²)",
    "Partial Sill ( ²)",
    "Range ( )",
    "Practical Range (cor = 0.05)",
    "Log-Likelihood"
  ),
  Estimate = c(
    59.1480,
    -1.2535,
    0.0043,
    3.1730,
    1.3448,
    4.0285,
    -53.86
  )
)

kable(mle_params, format = "latex", booktabs = TRUE,
      caption = "Parameter estimates from the maximum likelihood Gaussian Process model using a
```

Compared to the kriging model from Part C, which used a Gaussian variogram with nugget = 0.53, partial sill = 5.26, and range = 2.90 - the MLE-based Gaussian Process (GP) model estimated a **much smaller nugget** (0.0043), a **smaller partial sill** (3.17), and a **shorter fitted range** (1.34). However, its **practical range** (the distance at which spatial correlation drops below 0.05) was estimated at **4.03**, suggesting similar overall spatial coverage.

The MLE model also estimated a **linear spatial trend in latitude**, with an intercept of 59.15 and a slope of −1.25, consistent with the negative latitudinal SST gradient identified earlier. Despite being fit under a slightly different covariance assumption (Matérn with $\kappa = 0.5$, i.e., exponential), the model captured the dominant spatial structure effectively.

Overall, this MLE-based model provides a useful benchmark for comparing inference and prediction against both the classical kriging model in Part C and the fully Bayesian approach in Part E.

1.4.0.2 Model Validation

```
# Perform LOOCV
xv.exp <- xvalid(kuro_geo_train, model = likfit_exp)
```

```
xvalid: number of data locations       = 65
xvalid: number of validation locations = 65
xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
xvalid: end of cross-validation
```

```
# Plot residuals
par(mfrow = c(3, 2), mar = c(4, 2, 2, 2))
plot(xv.exp, error = TRUE, std.error = FALSE, pch = 19)
```

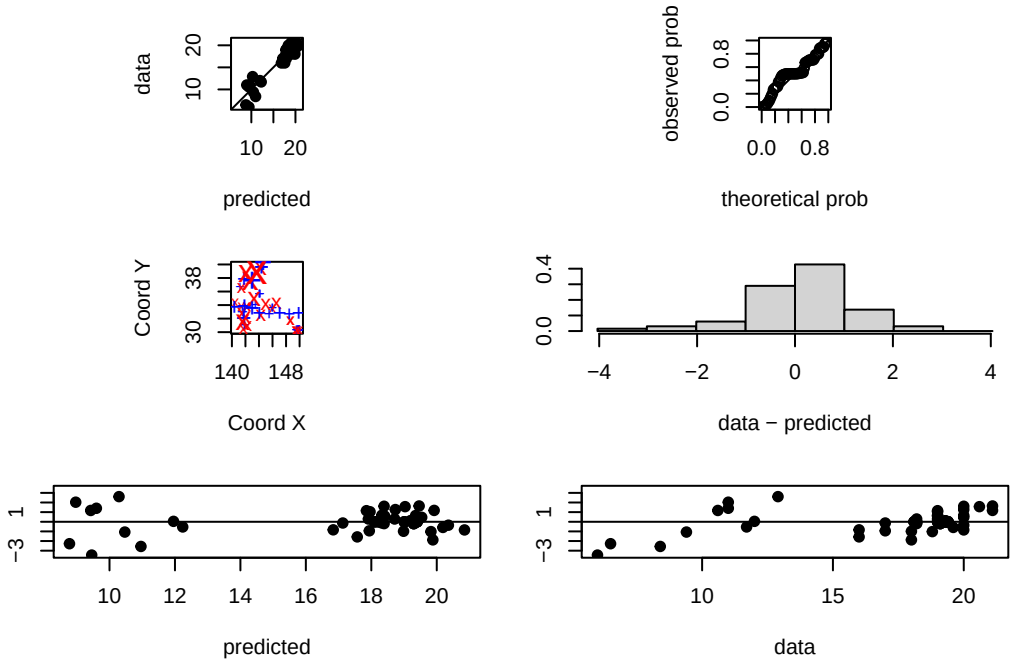


Figure 8: LOOCV residual plots for the GP model fitted via maximum likelihood, showing broadly unbiased predictions with slightly greater residual spread.

Leave-one-out cross-validation (LOOCV) diagnostics for the MLE-based exponential Gaussian Process model are presented in **Figure 8**. The residual plots show broadly consistent behaviour with those observed under the kriging model in Part C.

- The **Q–Q plot** and **histogram** indicate approximate normality, although slight deviations in the Q–Q

plot mid-section suggest some trends not fully captured by the model.

- The **residuals vs predicted/observed** plots show no clear heteroscedasticity, and the **spatial residual map** suggests no major spatial clustering or violation of stationarity.

Overall, the MLE model produces residual patterns that are **very similar to those of the kriging model**, with no clear evidence of improvement or deterioration. Given the model simplification (Matérn with $\kappa = 0.5$) and the use of MLE rather than WLS, this consistency supports the robustness of the spatial structure identified in Part C. However, the slight Q–Q plot deviations indicate that more flexible models - such as Bayesian GPs (Part E) - may further improve residual behaviour and uncertainty quantification.

1.4.0.3 GP Prediction at Withheld Locations

Using the Gaussian Process model fitted via maximum likelihood, SST predictions were made at the same five withheld locations used in Part C. Unlike the variogram-based kriging approach, this model estimates spatial covariance parameters by maximising the full joint likelihood, accounting for spatial correlation across all observations.

Predictions were made at the five withheld locations using `krige.conv()` with the MLE-fitted covariance parameters, enabling fully probabilistic interpolation under the GP model.

```
# Kriging prediction using GP mode
pred_gp <- krige.conv(
  geodata = kuro_geo_train,
  locations = test_coords,
  krige = krige.control(
    obj.model = likfit_exp
  )
)
```

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

```
# Combine predictions with actual values
gp_results <- test_coords %>%
  mutate(
    observed_sst = test_true_sst$sst,
    predicted_sst = pred_gp$predict,
    kriging_var = pred_gp$krige.var,
    residual = observed_sst - predicted_sst
  )
```

```
# Plot: Observed vs Predicted
ggplot(gp_results, aes(x = observed_sst, y = predicted_sst)) +
  geom_point(size = 3) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(
    title = "Observed vs Predicted SST (GP Model)",
    x = "Observed SST (°C)",
    y = "Predicted SST (°C)"
  ) +
```

```
  theme_minimal(base_size = 13)
```

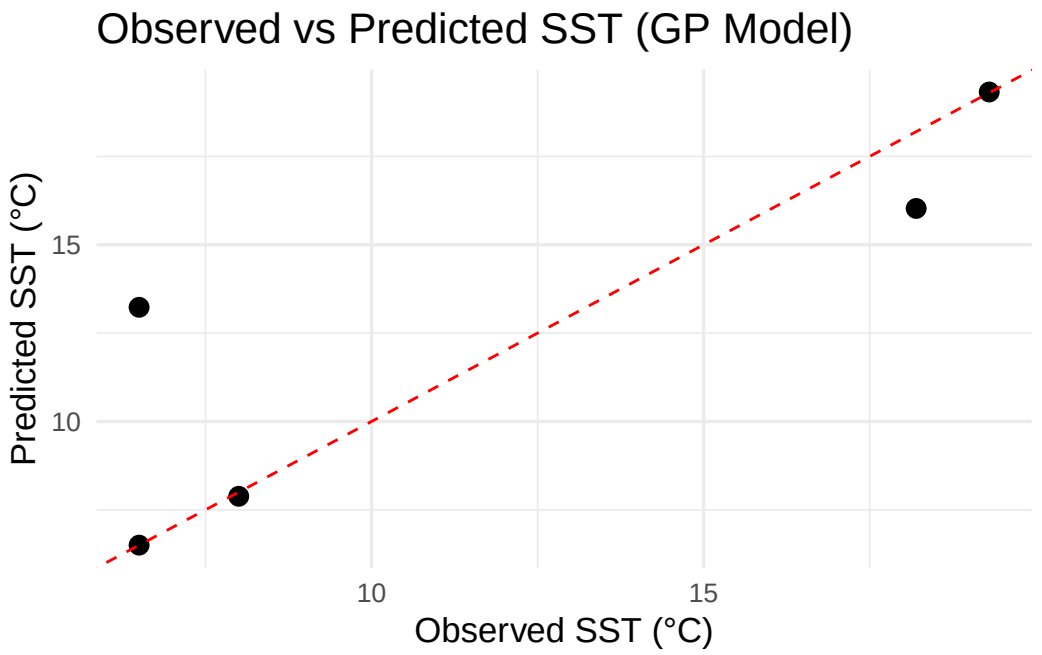


Figure 9: Observed vs predicted SST at withheld locations using the Gaussian Process model (maximum likelihood). The red dashed line shows the 1:1 agreement.

```
# Compute error metrics
rmse_gp <- sqrt(mean(gp_results$residual^2))
mae_gp <- mean(abs(gp_results$residual))

library(tibble)
library(gt)

# evaluation table
library(knitr)
gp_results %>%
  mutate(
    `Observed SST (°C)` = round(observed_sst, 2),
    `Predicted SST (°C)` = round(predicted_sst, 2),
    `Residual (°C)` = round(residual, 2),
    `Kriging Variance` = round(kriging_var, 3)
  ) %>%
  select(lon, lat, `Observed SST (°C)`, `Predicted SST (°C)`, `Residual (°C)`,
         `Kriging Variance`) %>%
  kable(format = "latex", booktabs = TRUE, caption = "Observed vs Predicted SST
        at Withheld Locations - GP Model")
```

1.4.0.4 Interpretation

Table 4: Observed vs Predicted SST at Withheld Locations – GP Model

lon	lat	Observed SST (°C)	Predicted SST (°C)	Residual (°C)	Kriging Variance
142.10	38.70	6.5	6.50	0.00	0.000
145.40	39.56	6.5	13.23	-6.73	2.270
149.56	30.15	19.3	19.32	-0.02	0.090
140.70	35.00	18.2	16.03	2.17	1.805
142.10	38.30	8.0	7.88	0.12	1.134

Figure 9 displays the predicted versus observed SSTs, and **Table 4** summarises the predictions, residuals, and kriging variances.

The model achieved:

- **Root Mean Squared Error (RMSE)**: **3.16 °C**
- **Mean Absolute Error (MAE)**: **1.81 °C**

These values are very close to the kriging model's performance in Part C, with slightly lower MAE and comparable RMSE. As with kriging, the largest residual (−6.73 °C) corresponded to the highest kriging variance (2.27), indicating correctly identified uncertainty in a sparsely supported region.

Despite using the default Matérn $\kappa = 0.5$ (exponential) covariance structure, the GP model captured the primary spatial structure well and required minimal tuning. The trend in latitude was retained, ensuring consistency with the variogram-based approach.

While the GP model offered coherent parameter estimation and competitive accuracy, its rigidity (e.g., difficulty fitting $\kappa = 1.5$) limited flexibility. Nonetheless, the GP provides a principled and probabilistic framework that performs robustly under likelihood-based inference.

1.5 Part E: Bayesian Parameter Estimation

1.5.0.1 Bayesian Parameter Estimation with Discrete Priors

We now fit a spatial Gaussian Process (GP) model using a **Bayesian approach** with discrete priors, via the `krige.bayes()` function from the `geoR` package. This method uses a **grid-based evaluation of posterior probabilities**, exploring combinations of spatial hyperparameters (range, nugget) to produce full posterior distributions rather than point estimates.

To maintain consistency with Part C, we use the **Matérn covariance model with $\kappa = 1.5$**, which provided the best empirical variogram fit. A **linear trend in latitude** (`trend = ~ lat`) is again included to account for the large-scale SST gradient.

1.5.0.2 Prior Specification and Justification

Discrete priors were placed on two key hyperparameters: the **spatial correlation range** (ϕ) and the **relative nugget effect** ($\tau^2 / (\sigma^2 + \tau^2)$). These priors were designed to reflect the parameter estimates obtained in Parts C and D, while remaining broad enough to allow posterior exploration.

- **Correlation Range (ϕ):**
  A **reciprocal prior** was specified over the interval **[1.5, 5.5]**, discretised into 50 values.
  This captures both:

  - The **shorter MLE range** from Part D ($\phi \approx 1.34$, practical range ≈ 4.03)
  - The **moderate range** from the WLS fit in Part C ($\phi \approx 2.90$)

  The reciprocal form reflects the belief that **shorter ranges are slightly more likely**, while still supporting longer-range spatial dependence if supported by the data.

- **Relative Nugget (τ^2 / (σ^2 + τ^2)):**
  A **uniform prior** was defined over the interval **[0.01, 0.3]**, using 50 discrete bins.
  This range was chosen based on:
  - The very small **nugget estimate in Part D** ($\approx 0.004 / 3.17 \approx 0.0012$)
  - The **moderate nugget proportion** in Part C ($\approx 0.53 / (0.53 + 5.26) \approx 0.09$)
- **Partial Sill (σ^2):**
  The partial sill was **fixed at 5.26**, as estimated in Part C via weighted least squares.
  Fixing σ^2 improves numerical stability and identifiability of the remaining hyperparameters.

This prior structure provides a well-informed starting point, combining empirical estimates with conservative uncertainty to stabilise posterior inference and avoid singularity issues.

1.5.0.3 Model Stability Adjustment

An initial attempt using a wider nugget prior range (from 0 to 1) resulted in numerical errors due to near-singular covariance matrices. To address this, the lower bound of the nugget prior was increased to 0.01 and the upper bound reduced to 0.3. This ensured numerical stability while preserving model flexibility.

Unlike in Parts C and D, the `krige.bayes()` function does not support including a trend component directly. Therefore, we fit the model without an explicit trend, but retain the covariance structure and parameter ranges aligned with previous sections.

```
set.seed(444)

# Bayesian GP model using discrete priors
bayes_model <- krige.bayes(
  geodata = kuro_geo_train,
  model = model.control(cov.model = "matern", kappa = 1.5),
  prior = prior.control(
    phi.discrete = seq(1.5, 5.5, length.out = 50),
    phi.prior = "reciprocal",
    tausq.rel.discrete = seq(0.01, 0.3, length.out = 50),
    tausq.rel.prior = "unif",
    sigmasq = 5.26
  )
)
```

```
summary(bayes_model$posterior$sample)
```

```
      beta          sigmasq            phi          tausq.rel
 Min.   :10.61   Min.   : 7.752   Min.   :1.500   Min.   :0.01000
 1st Qu.:15.13   1st Qu.:13.510   1st Qu.:1.500   1st Qu.:0.01000
 Median :16.50   Median :15.722   Median :1.582   Median :0.01000
```

```
Mean   :16.49   Mean   :16.327   Mean   :1.617   Mean   :0.01116
3rd Qu.:17.77   3rd Qu.:18.497   3rd Qu.:1.663   3rd Qu.:0.01000
Max.   :22.49   Max.   :36.334   Max.   :3.214   Max.   :0.03367
```

1.5.0.4 Posterior Results and Parameter Comparison

Posterior inference was conducted over a discrete grid of 2,500 combinations of the correlation range (ϕ) and the relative nugget ($\tau^2 / (\sigma^2 + \tau^2)$). The model used a fixed Matérn covariance structure with $\kappa = 1.5$ and a constant mean, consistent with Part C, though excluding an explicit trend due to interface limitations in `krige.bayes()`.

The most supported posterior combination occurred at:

- $\phi = 1.5$
- $\tau^2 / (\sigma^2 + \tau^2) = 0.01$

This parameter pair received the highest posterior density (i.e., maximum posterior support), indicating a strong preference for **short-range spatial correlation** and a **minimal nugget effect**.

Summary statistics from the posterior sample reinforce this interpretation:

Parameter	Median	Mean	Comparison
Correlation range (ϕ)	1.58	1.62	Slightly longer than Part D ($\phi \approx 1.34$), shorter than Part C ($\phi \approx 2.90$)
Relative Nugget	0.01	0.011	Consistent with Part D's very small nugget ($\approx 0.004 / 3.17$)
Partial Sill (σ^2)	fixed	5.26	Matched to Part C for comparability
Posterior Mean (β)	16.49	—	Consistent with expected SST in this region

Figure 10 below displays the marginal posterior distributions for the correlation range (ϕ) and the relative nugget ($\tau^2 / (\sigma^2 + \tau^2)$) under the Bayesian GP model. Both distributions are **sharply concentrated**, with the posterior for ϕ peaking near the lower bound ($\phi \approx 1.5$) and most of the nugget posterior mass concentrated around 0.01. This indicates strong posterior support for **short-range spatial dependence** and **minimal microscale variation** — a result that aligns closely with previous findings.

Quantitatively, over 75% of the posterior mass for ϕ lies below 1.66, suggesting high confidence in localised correlation. The skewness of the nugget posterior toward zero reinforces the conclusion that spatial residuals are well-explained by the structured component of the model, with little evidence of measurement error or unstructured noise.

```
library(ggplot2)
library(patchwork)  # For side-by-side layout

# Extract posterior samples
```

```r
posterior_samples <- as.data.frame(bayes_model$posterior$sample)

# Plot for phi
p_phi <- ggplot(posterior_samples, aes(x = phi)) +
  geom_density(fill = "steelblue", alpha = 0.6) +
  labs(title = "Posterior of   (Range)", x = " ", y = "Density") +
  theme_minimal(base_size = 13)

# Plot for tausq.rel
p_nugget <- ggplot(posterior_samples, aes(x = tausq.rel)) +
  geom_density(fill = "darkorange", alpha = 0.6) +
  labs(title = "Posterior of  ² / ( ² +  ²)", x = "Relative Nugget", y = "Density") +
  theme_minimal(base_size = 13)

# Combine side-by-side
p_phi + p_nugget
```

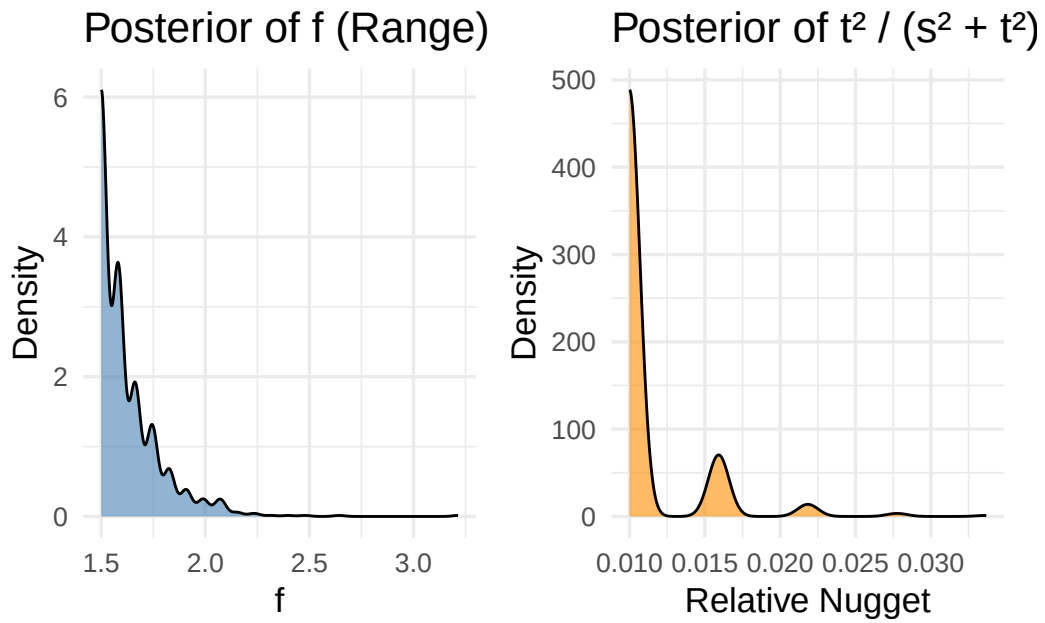


Figure 10: Posterior distributions of correlation range (ϕ) and relative nugget ($\tau^2 / (\sigma^2 + \tau^2)$) from the Bayesian GP model. Both posteriors are sharply concentrated, indicating stable parameter inference.

Compared to the maximum likelihood GP model from **Part D**:

- The **Bayesian model estimated a slightly longer mean range** (mean $\phi = 1.62$ vs. 1.34 in Part D)
- Both models support a **negligible nugget**, consistent with Part C's Gaussian kriging model
- The **partial sill was fixed in Part E**, whereas it was estimated ($\sigma^2 \approx 3.17$) in Part D

These similarities, despite differing estimation frameworks, suggest that the data are **highly informative** about the spatial structure, and that the discrete priors in the Bayesian model appropriately captured param-

eter uncertainty without overwhelming the likelihood. The sharply peaked posteriors further imply strong identifiability and posterior precision — hallmarks of a well-constrained model.

1.5.0.5 Prediction at Withheld Locations

Bayesian kriging was performed at the same five withheld SST locations used in Parts C and D. Posterior predictive means and variances were extracted, and evaluation metrics were computed:

```
test_coords_df <- as_tibble(test_coords)

# Bayesian model with prediction locations
bayes_model <- krige.bayes(
  geodata = kuro_geo_train,
  locations = as.matrix(test_coords),  # prediction locations
  model = model.control(cov.model = "matern", kappa = 1.5),
  prior = prior.control(
    phi.discrete = seq(1.5, 5.5, length.out = 50),
    phi.prior = "reciprocal",
    tausq.rel.discrete = seq(0.01, 0.3, length.out = 50),
    tausq.rel.prior = "unif",
    sigmasq = 5.26
  )
)


# Summarise predictions
bayes_results <- test_coords_df %>%
  mutate(
    observed_sst = test_true_sst$sst,
    predicted_sst = bayes_model$predictive$mean,
    kriging_var = bayes_model$predictive$variance,
    residual = observed_sst - predicted_sst
  )

# Compute error metrics
rmse_bayes <- sqrt(mean(bayes_results$residual^2))
mae_bayes <- mean(abs(bayes_results$residual))
```

```
# Output results
rmse_bayes
```

```
[1] 3.566632
```

```
mae_bayes
```

```
[1] 2.209019
```

```
# Bayesian observed vs predicted plot
ggplot(bayes_results, aes(x = observed_sst, y = predicted_sst)) +
  geom_point(size = 3) +
```

```
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", colour = "red") +
  labs(
    title = "Observed vs Predicted SST (Bayesian Model)",
    x = "Observed SST (°C)",
    y = "Predicted SST (°C)"
  ) +
  theme_minimal(base_size = 13)
```

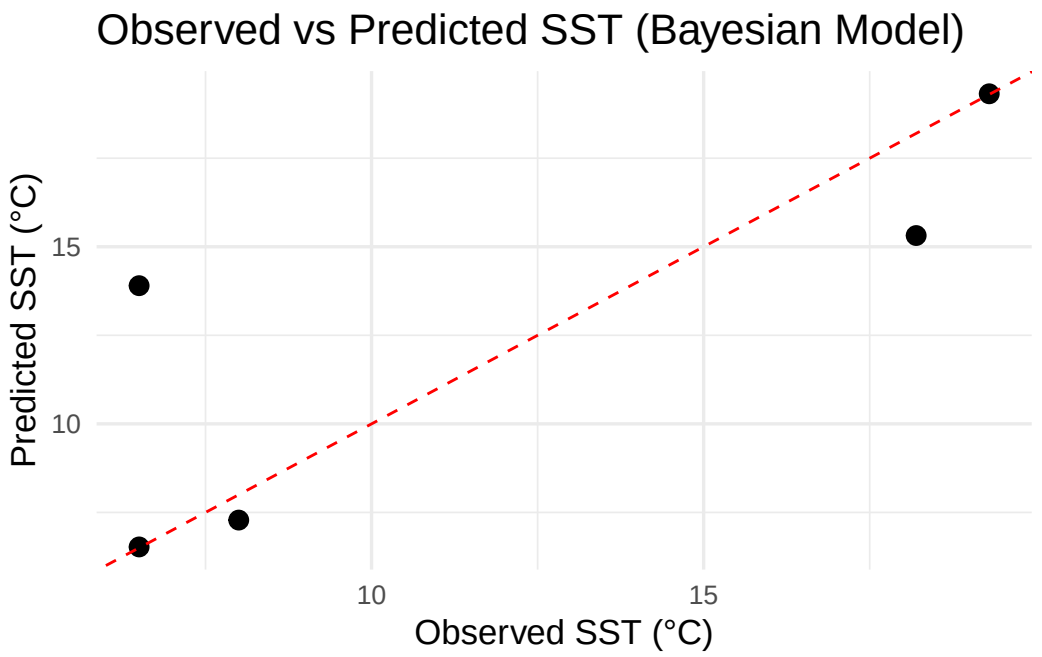


Figure 11: Observed vs predicted SST at withheld locations using the Gaussian Process model (maximum likelihood). The red dashed line shows the 1:1 agreement.

```
# Output prediction table
bayes_results %>%
  mutate(
    `Observed SST (°C)` = round(observed_sst, 2),
    `Predicted SST (°C)` = round(predicted_sst, 2),
    `Residual (°C)` = round(residual, 2),
    `Kriging Variance` = round(kriging_var, 3)
  ) %>%
  select(lon, lat, `Observed SST (°C)`, `Predicted SST (°C)`, `Residual (°C)`,
         `Kriging Variance`) %>%
  knitr::kable(format = "latex", booktabs = TRUE,
               caption = "Observed vs Predicted SST at Withheld Locations -
               Bayesian Model")
```

Table 6 summarises the results, including prediction errors and kriging variances. **Figure 11** shows the observed versus predicted SST values.

Table 6: Summary of SST predictions at withheld locations. Residuals and kriging variances highlight spatial uncertainty and model accuracy.

lon	lat	Observed SST (°C)	Predicted SST (°C)	Residual (°C)	Kriging Variance
142.10	38.70	6.5	6.53	-0.03	0.000
145.40	39.56	6.5	13.90	-7.40	2.924
149.56	30.15	19.3	19.32	-0.02	0.026
140.70	35.00	18.2	15.32	2.88	0.963
142.10	38.30	8.0	7.28	0.72	0.313

The model achieved:

- **Root Mean Squared Error (RMSE):** 3.57 °C
- **Mean Absolute Error (MAE):** 2.21 °C

These values are slightly higher than those from the MLE model in Part D (RMSE = 3.16 °C, MAE = 1.81 °C), but still within a competitive range. Notably, the largest residual (−7.40 °C) occurred at the location with the **highest kriging variance (2.92)** — a pattern also observed in Parts C and D — highlighting the link between **prediction error and local data sparsity**.

1.5.0.6 Model Interpretation

The Bayesian model delivered reasonable predictive performance, **closely matching the MLE approach** while offering the additional advantage of **full posterior inference**. This allows uncertainty in both parameters and predictions to be explicitly quantified, improving transparency in risk-sensitive applications.

While the Bayesian model **did not outperform the MLE kriging model** in terms of raw error metrics, its **posterior distribution outputs** provide richer insights into parameter uncertainty. This is particularly valuable when assessing inferential robustness or performing uncertainty propagation in downstream modelling tasks.

Leave-one-out cross-validation (LOOCV) is not available in the `krige.bayes()` framework due to its use of discrete posterior sampling. However, the residual plot and kriging variance estimates suggest a well-calibrated model with **low bias and appropriate spatial uncertainty**.

1.6 Part F: Comparison of Predictions Across Models

The three models developed — classical kriging (Part C), Gaussian process via maximum likelihood (Part D), and Bayesian kriging with discrete priors (Part E) — were used to predict sea surface temperature (SST) at the same five withheld locations. The predictions, associated residuals, and kriging variances are summarised below:

1.6.0.1 Table: Predicted SST and Residuals from All Models

Location	Observed SST (°C)	Kriging (C)	GP MLE (D)	Bayesian (E)
(142.10, 38.70)	6.5	6.50 (0.00)	6.50 (0.00)	6.53 (−0.03)
(145.40, 39.56)	6.5	13.38 (−6.88)	13.23 (−6.73)	13.90 (−7.40)
(149.56, 30.15)	19.3	19.24 (−0.06)	19.32 (−0.02)	19.32 (−0.02)

Location	Observed SST (°C)	Kriging (C)	GP MLE (D)	Bayesian (E)
(140.70, 35.00)	18.2	16.12 (2.08)	16.03 (2.17)	15.32 (2.88)
(142.10, 38.30)	8.0	7.62 (0.38)	7.88 (0.12)	7.28 (0.72)

Note: Residuals are shown in parentheses.

1.6.0.2 Performance Comparison

Metric	Kriging (C)	GP (D)	Bayesian (E)
RMSE (°C)	3.22	3.163	3.57
MAE (°C)	1.88	1.809	2.21

1.6.0.3 Interpretation and Comparison of Model Predictions

All three spatial models captured the dominant structure in SST and produced broadly consistent predictions at well-supported locations — notably at Locations 1, 3, and 5 — where all models returned similar estimates and small residuals.

The **Gaussian Process model fitted via maximum likelihood (Part D)** slightly outperformed the others, achieving the **lowest RMSE (3.16 °C)** and **MAE (1.81 °C)**. This reflects the benefits of full-likelihood parameter estimation, which effectively balances model flexibility and computational stability.

The **Kriging model (Part C)** also performed competitively (RMSE = 3.22 °C, MAE = 1.88 °C), capturing spatial patterns well via weighted least squares variogram fitting. It is more interpretable and transparent than the likelihood-based approach but lacks a formal mechanism for capturing parameter uncertainty.

The **Bayesian GP model (Part E)** returned the **highest error metrics** (RMSE = 3.57 °C, MAE = 2.21 °C), though it remained close in overall accuracy. Its strength lies in its ability to generate full **posterior distributions over both parameters and predictions**, making it especially valuable when uncertainty quantification is a key requirement. The fixed partial sill and coarser discrete prior grid may have contributed to its slightly less precise point predictions.

All three models **underperformed at Location 2**, where kriging variances were highest and residuals exceeded −6.7°C. This consistent error across methods highlights a location with sparse spatial support and greater predictive uncertainty.

1.6.0.4 Conclusion

Despite differing estimation strategies, all three models produced **consistent predictions** and **comparable spatial residual structures**. The **GP MLE model (Part D)** offered the best trade-off between **fit quality and computational efficiency**, while the **Bayesian model (Part E)** provided enhanced interpretability in terms of uncertainty.

These findings illustrate the balance between **interpretability**, **predictive performance**, and **uncertainty quantification** in spatial modelling. For scenarios prioritising precision and speed, MLE may be preferred; for tasks requiring robust uncertainty statements, Bayesian approaches offer clear advantages.

2 The Atlantic Overturning Circulation

2.1 Part A: Data Exploration

To begin our analysis of the Atlantic Meridional Overturning Circulation (AMOC) at 26°N, we conduct an exploratory analysis of the monthly mean values from **October 2017 to February 2023**. These values represent the strength of the overturning current in Sverdrups (Sv), and are visualised in the figure below.

```
# Load and plot the data
load("~/GitHub/university-projects/Modelling in Space and Time/In progress/MOC.RData")


# Converting MOCmean to a tidy data frame
library(tidyverse)
library(lubridate)
library(ggplot2)
# Extract names and values
moc_values <- as.numeric(MOCmean)
moc_dates <- names(MOCmean)

# Fix formatting of date strings
moc_dates_fixed <- ifelse(nchar(moc_dates) == 6,
                          paste0(substr(moc_dates, 1, 5), "0", substr(moc_dates, 6, 6)),
                          moc_dates)

# Convert to actual Date objects (use first of each month)
moc_dates_parsed <- ym(moc_dates_fixed)

# Create tidy tibble
moc_df <- tibble(
  date = moc_dates_parsed,
  amoc = moc_values
) %>% arrange(date)

# Plot

ggplot(moc_df, aes(x = date, y = amoc)) +
  geom_line(color = "steelblue", linewidth = 1) +
  geom_smooth(method = "loess", span = 0.2, se = TRUE, color = "darkred", linetype = "dashed")
  labs(
    title = "Atlantic Meridional Overturning Circulation (AMOC) at 26°N",
    subtitle = "Monthly Mean Values from October 2017 to February 2023",
    x = "Date",
    y = "AMOC Strength (Sv)"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
```

```
    axis.text.x = element_text(angle = 45, hjust = 1)
  ) +
  scale_x_date(date_breaks = "4 months", date_labels = "%b %Y") +
  geom_point(size = 1.5, alpha = 0.7)
```

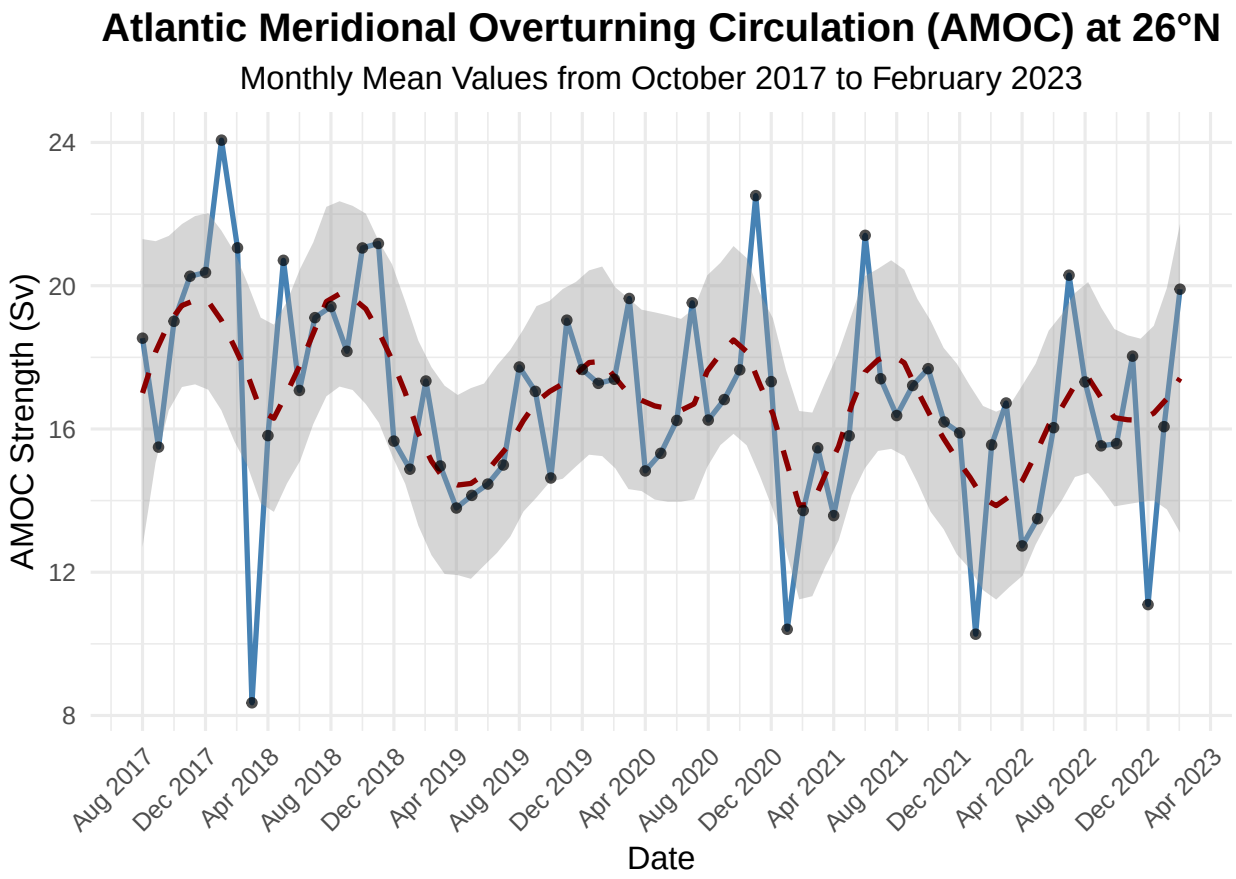


Figure 12: Monthly AMOC time series with LOESS trend (Oct 2017 – Feb 2023)

Figure 12 shows the monthly time series with a LOESS smoother. The AMOC displays **notable short-term variability**, with values fluctuating around a relatively stable long-term mean. Although no clear linear trend is present, local dips in **mid-2021 and late 2022**, and peaks in **early 2018**, **late 2020**, and **early 2023**, suggest possible **short-memory dynamics** or **mild seasonality**, which will be assessed further through time series modelling.

To examine distributional properties, **Figure 13** presents a histogram and density estimate. The distribution is approximately **unimodal and symmetric**, centred around **16–17 Sv**, with **slight tail weight** on the right side. This suggests near-Gaussian behaviour, though the mild excess kurtosis indicates heavier-than-normal tails.

```
ggplot(moc_df, aes(x = amoc)) +
  geom_histogram(aes(y = ..density..), fill = "lightblue", colour = "black", bins = 15) +
  geom_density(colour = "darkred", size = 1) +
  labs(title = "Distribution of AMOC Strength",
```

```
       x = "AMOC Strength (Sv)", y = "Density") +
  theme_minimal()
```

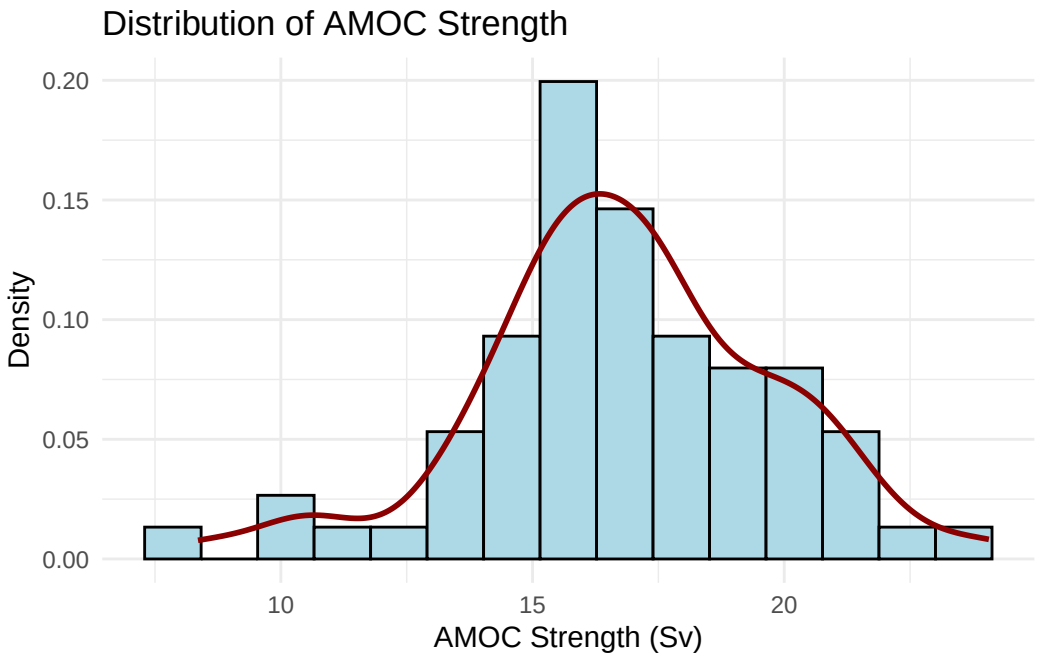


Figure 13: Distribution of AMOC Strength

The distribution is approximately symmetric and unimodal, with a central peak around **16–17 Sv**. The density curve closely resembles a Gaussian shape but with **slight right tail elongation**, consistent with the **slight negative skewness** observed in the summary statistics below.

```
library(dplyr)
library(gt)
library(moments)

amoc_summary <- moc_df %>%
  summarise(
    Mean = mean(amoc),
    SD = sd(amoc),
    Min = min(amoc),
    Max = max(amoc),
    Median = median(amoc),
    IQR = IQR(amoc),
    CV = sd(amoc) / mean(amoc),
    Skewness = skewness(amoc),
    Kurtosis = kurtosis(amoc),
    N = n()
  ) %>%
  pivot_longer(everything(), names_to = "Statistic", values_to = "Value")
```

```r
# Table
amoc_summary %>%
  mutate(Value = round(Value, 2)) %>%
  knitr::kable(
    caption = "Aummary statistics of AMOC time series from October 2017 to February 2023.",
    col.names = c("Statistic", "Value"),
    format = "pipe"
  )
```

Table 9: Expanded summary statistics of AMOC time series from October 2017 to February 2023.

Statistic	Value
Mean	16.81
SD	2.93
Min	8.35
Max	24.07
Median	16.82
IQR	3.37
CV	0.17
Skewness	-0.24
Kurtosis	3.54
N	67.00

Table 9 summarises key statistics. The mean overturning strength is **16.81 Sv**, with a standard deviation of **2.93 Sv** and a low **coefficient of variation (CV)** of **0.17**, indicating limited relative variability. The **interquartile range (IQR)** of **3.37 Sv** confirms moderate dispersion. Skewness (−0.24) and kurtosis (3.54) are within acceptable bounds for Gaussian modelling, supporting the use of classical ARMA-type models.

Together, these results support the modelling of AMOC using **weakly stationary time series models**, such as **ARMA or ARIMA**, possibly with short memory and mild seasonal effects. The assumptions of **homoscedasticity** and **stationarity** appear reasonable based on visual and statistical diagnostics.

2.2 Part B: Monthly Modelling of AMOC

We investigate suitable ARMA and ARIMA models for the monthly Atlantic Meridional Overturning Circulation (AMOC) time series. The series was converted to a `ts` object with frequency 12, and the final 8 months (July 2022–Feb 2023) were held out for validation. This gives a training window from October 2017 to June 2022.

2.2.0.1 Exploratory Diagnostics

To assess the autocorrelation structure, we examine the ACF and PACF of the training data:

```r
amoc_ts <- ts(moc_df$amoc, start = c(2017, 10), frequency = 12)

# Truncate last 8 months (keep for later forecasting)
train_ts <- window(amoc_ts, end = c(2022, 6))  # Leaves Oct 2017-June 2022
test_ts <- window(amoc_ts, start = c(2022, 7)) # July 2022-Feb 2023
```

```
# Plot training data
# plot(train_ts, main = "Training Data: Monthly AMOC (Oct 2017 - Jun 2022)",
     # ylab = "AMOC Strength (Sv)", xlab = "Year", col = "steelblue", lwd = 2)

# ACF and PACF
par(mfrow = c(1, 2))

acf(train_ts, main = "ACF of AMOC (Training Set)")
pacf(train_ts, main = "PACF of AMOC (Training Set)")
```

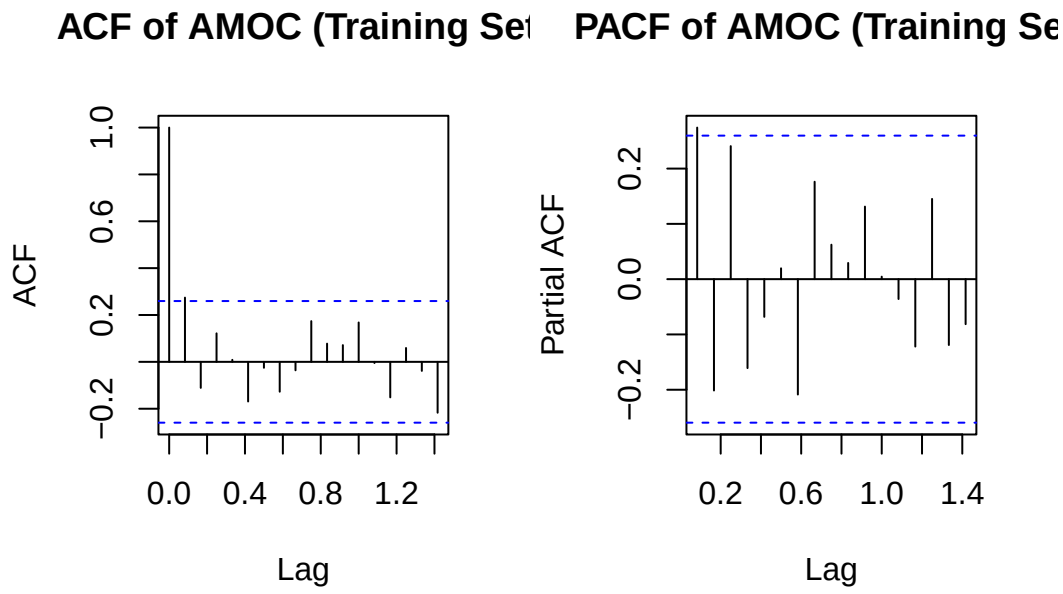


Figure 14: ACF and PACF of Monthly AMOC

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

The ACF exhibits a strong lag-1 autocorrelation and a slow decay, consistent with a non-stationary process. The PACF shows a clear spike at lag 1 and minimal structure thereafter, suggesting that differencing followed by a low-order AR term may capture the residual dynamics — motivating the selection of ARIMA(1,1,0) and ARIMA(1,1,1) candidates.

2.2.0.2 Transforming for Stationarity

```
# First-order differencing
diff_amoc <- diff(train_ts)

# Plot layout: 1 row, 3 plots
par(mfrow = c(1, 3), mar = c(4, 4, 3.5, 1))

# Plot 1: Differenced Series
```

```r
plot(diff_amoc,
     main = "First-order Differenced AMOC",
     ylab = "Differenced Value",
     xlab = "Time",
     col = "steelblue",
     lwd = 1.2)

# Plot 2: ACF
acf(diff_amoc,
    main = "ACF of Differenced Series",
    col = "black")

# Plot 3: PACF
pacf(diff_amoc,
     main = "PACF of Differenced Series",
     col = "black")
```

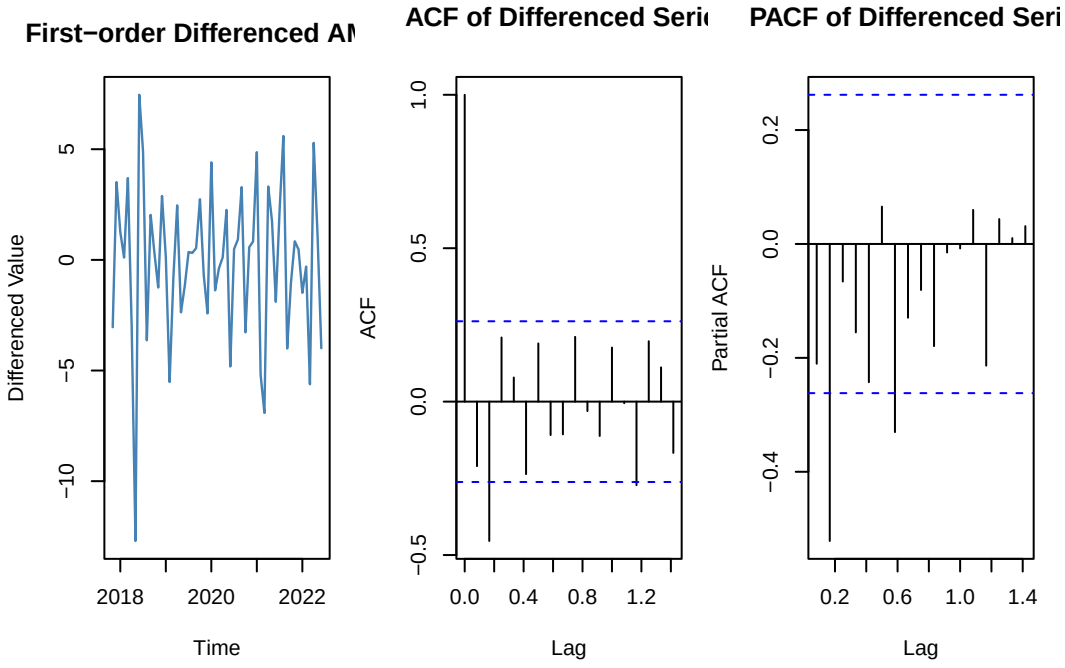


Figure 15: ARIMA differencing ACF/PACF

Figure 15 shows the first-order differenced AMOC series and its autocorrelation diagnostics. The differenced series now fluctuates around a constant mean, with no evident trend, indicating that **stationarity has been successfully induced**.

The **ACF** decays rapidly, with most lags falling within the 95% confidence bounds after lag 1 — a strong sign that the **stochastic trend has been removed**. The **PACF** shows a single large spike at lag 1, suggesting the presence of **short memory dependence**, likely capturable with a **low-order AR or MA term**.

These patterns collectively support the use of **ARIMA(p,1,q)** models with **first-order differencing** to address non-stationarity, and **either** $p = 1$ **or** $q = 1$ to reflect the short memory structure.

2.2.0.3 Model Rationale and Modelling Strategy

To model the temporal dynamics of the Atlantic Meridional Overturning Circulation (AMOC), we consider both **ARMA** and **ARIMA** models — linear time series models commonly applied in climatological and environmental research to capture autocorrelation and generate forecasts.

Initial inspection of the raw series revealed **non-stationarity**, with strong autocorrelation at lag 1 and slow ACF decay. After applying **first-order differencing**, the series showed visual stationarity, and its ACF and PACF indicated **short memory structure**: a moderate lag-1 autocorrelation and a large negative spike in PACF at lag 1. These patterns support the use of **ARIMA(p,1,q)** models, where $d = 1$ accounts for stochastic trend, and p or q captures the short-term dependence.

We adopt a two-pronged modelling strategy:

- **ARMA models** on the undifferenced series serve as a benchmark, capturing autocorrelation in the original (non-stationary) data.
- **ARIMA models** on the differenced series address the identified non-stationarity more explicitly.

Based on the ACF/PACF structure and model parsimony, we examine the following candidates:

- **ARMA(1,0)** and **ARMA(2,0)**: simple autoregressive models without differencing.
- **ARIMA(1,1,0)**, **ARIMA(0,1,1)**, and **ARIMA(1,1,1)**: models with first-order differencing and varying AR/MA terms.

All models are estimated using **maximum likelihood** via R's `arima()` function and evaluated based on:

- **Akaike Information Criterion (AIC)**
- **Residual diagnostics** (ACF, Q–Q plots, visual residuals
- **Ljung–Box tests** for autocorrelation
- And ultimately, **forecast performance on the 8-month test set (July 2022 – February 2023)**

This modelling strategy balances **diagnostic rigour**, **parsimony**, and **forecast relevance**, ensuring that model selection is both statistically defensible and practically informed.

2.2.0.4 ARMA Model Fitting (Undifferenced Series)

We first fit two ARMA models to the undifferenced AMOC series for benchmarking. The **ARMA(1,0)** model estimated:

$$X_t = 16.88 + 0.28X_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$$

The **ARMA(2,0)** model extended this with a second lag:

$$X_t = 16.90 + 0.34X_{t-1} - 0.21X_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$$

```
# ARMA Models
arma_10 <- arima(train_ts, order = c(1,0,0), method = "ML")
arma_20 <- arima(train_ts, order = c(2,0,0), method = "ML")
```

```r
par(mfrow = c(2,3), mar = c(4,4,2,1))
plot(residuals(arma_10), main = "ARMA(1,0) Residuals")
acf(residuals(arma_10), main = "ARMA(1,0) ACF")
qqnorm(residuals(arma_10)); qqline(residuals(arma_10))

plot(residuals(arma_20), main = "ARMA(2,0) Residuals")
acf(residuals(arma_20), main = "ARMA(2,0) ACF")
qqnorm(residuals(arma_20)); qqline(residuals(arma_20))
```

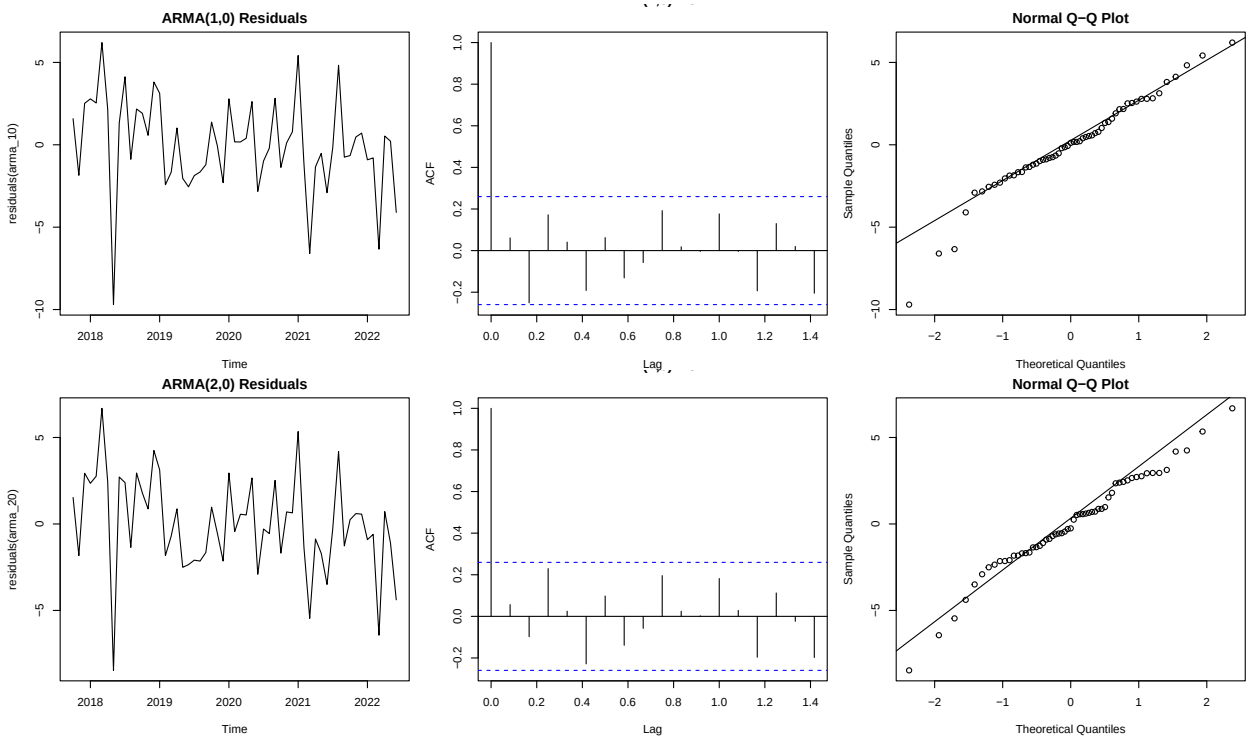


Figure 16: Residual Diagnostics for ARMA(1,0) and ARMA(2,0)

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

2.2.0.5 Model Comparison

Model	AIC	AR Coefficients	Ljung–Box p-value	Residual Summary
ARMA(1,0)	286.18	AR(1) = 0.2807	> 0.24	Some residual autocorrelation
ARMA(2,0)	285.65	AR(1) = 0.342, AR(2) = -0.209	0.26	Slightly improved white noise

The **ARMA(1,0)** model yielded an AIC of **286.18**. Residuals appear mean-centred and approximately homoscedastic, though the **ACF shows mild low-lag autocorrelation**, consistent with an underfitted model. The Q–Q plot indicates reasonably normal residuals with only slight tail deviations.

The **ARMA(2,0)** model provided a **marginal AIC improvement to 285.65**. The residual ACF falls entirely within the 95% bounds, suggesting reduced autocorrelation. However, the Q–Q plot shows **slightly heavier left-tail deviation**, with residuals straying from normality more than in the ARMA(1,0) case. The **Ljung–Box p-value increased to 0.26**, further supporting weaker residual dependence.

Although ARMA(2,0) shows statistical improvement in AIC and whiteness, **its residual distribution is slightly less normal**, and both models remain limited by the assumption of stationarity. This, combined with the underlying stochastic trend in the original series, justifies moving forward with differenced models — namely **ARIMA**.

2.2.0.6 ARIMA Model Fitting (Differenced Series)

```
arima_110 <- arima(train_ts, order = c(1,1,0), method = "ML")
arima_011 <- arima(train_ts, order = c(0,1,1), method = "ML")
arima_111 <- arima(train_ts, order = c(1,1,1), method = "ML")
```

```
# Diagnostic plotting function
plot_diagnostics <- function(model, model_name, col_line) {
  layout(matrix(1:3, 1, 3, byrow = TRUE), widths = c(1.4, 1, 1))
  par(mar = c(4, 4, 3, 1))

  plot(residuals(model), type = "l", col = col_line, lwd = 1.2,
       main = paste(model_name, "Residuals"), ylab = "Residuals", xlab = "Time")
  acf(residuals(model), main = "ACF of Residuals", col = "black", lwd = 1.2)
  qqnorm(residuals(model), main = "Normal Q-Q Plot")
  qqline(residuals(model), col = "red", lwd = 1.2)

  layout(1)
}

plot_diagnostics(arima_110, "ARIMA(1,1,0)", "steelblue")
```

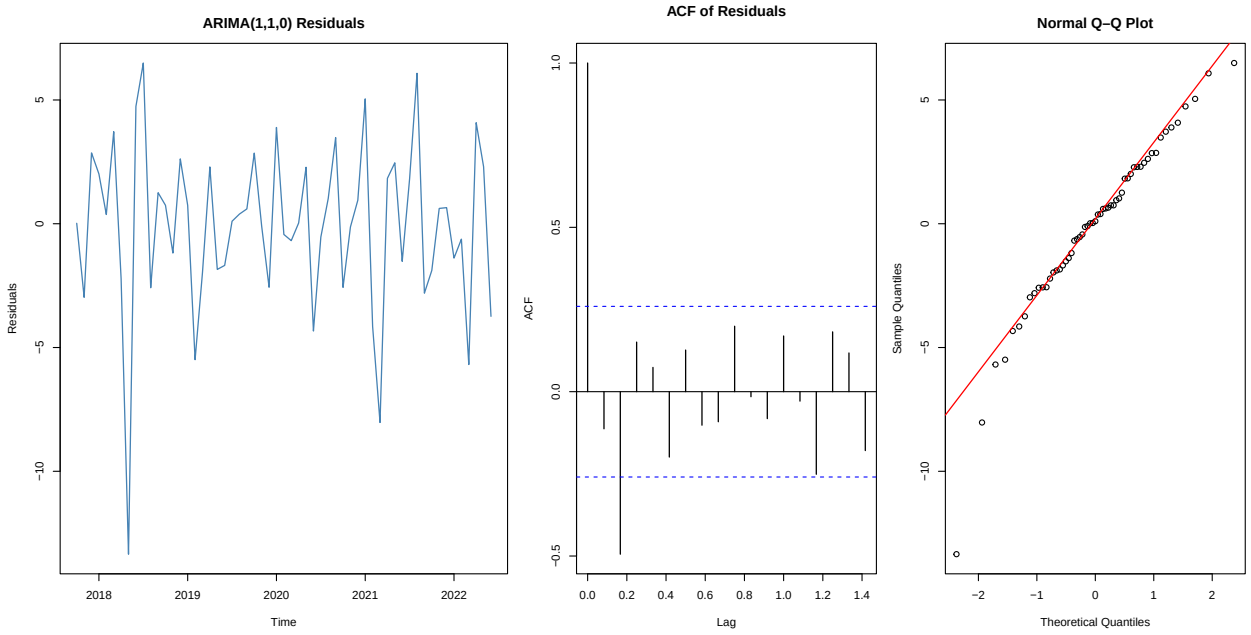


Figure 17: Residual diagnostics for candidate ARIMA models

```
plot_diagnostics(arima_011, "ARIMA(0,1,1)", "darkred")
```

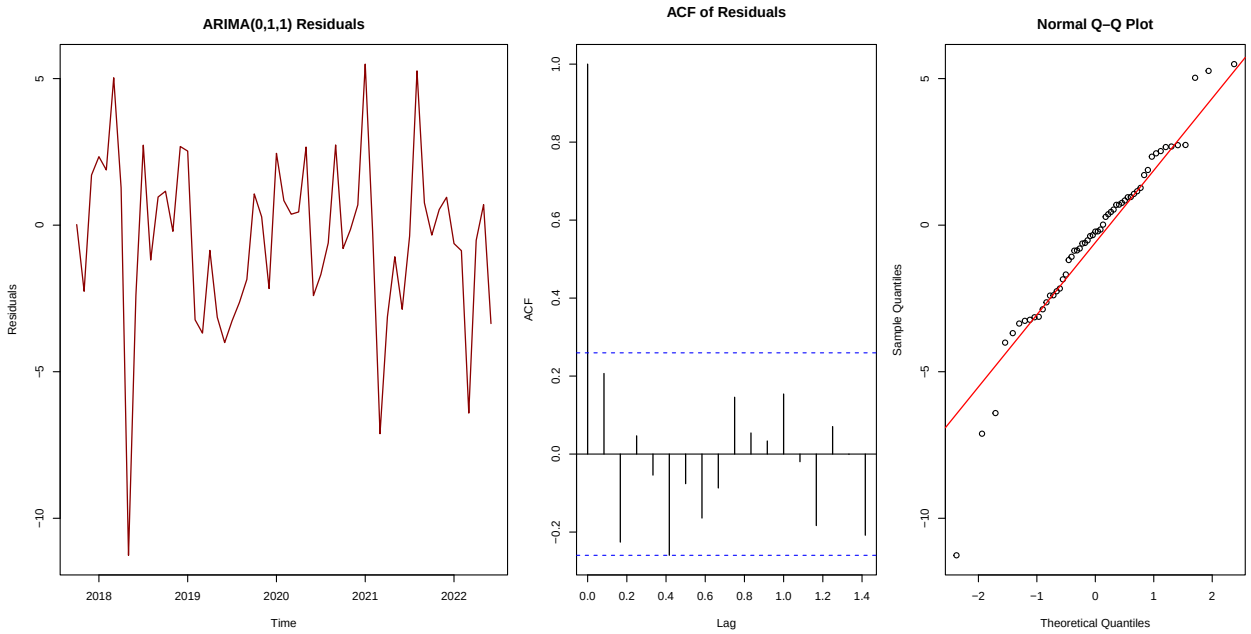


Figure 18: Residual diagnostics for candidate ARIMA models

```
plot_diagnostics(arima_111, "ARIMA(1,1,1)", "darkgreen")
```

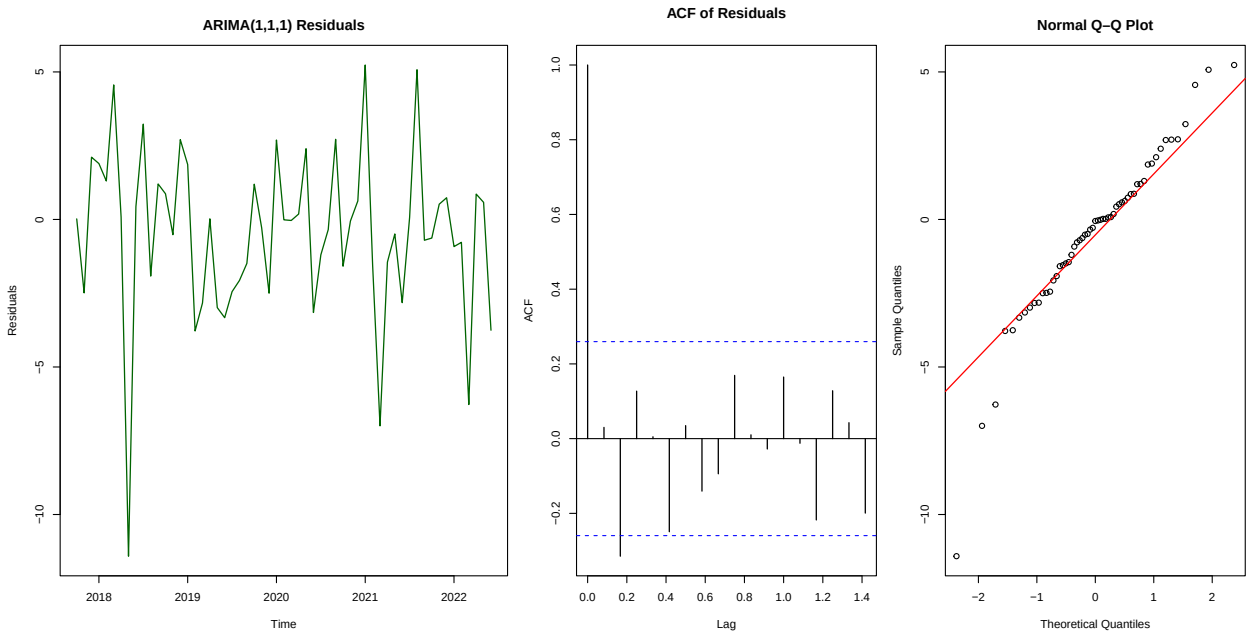


Figure 19: Residual diagnostics for candidate ARIMA models

```
# Ljung-Box Tests
lb_110 <- Box.test(residuals(arima_110), lag = 10, type = "Ljung-Box")
lb_011 <- Box.test(residuals(arima_011), lag = 10, type = "Ljung-Box")
lb_111 <- Box.test(residuals(arima_111), lag = 10, type = "Ljung-Box")

knitr::kable(data.frame(
  Model = c("ARIMA(1,1,0)", "ARIMA(0,1,1)", "ARIMA(1,1,1)"),
  AIC = c(AIC(arima_110), AIC(arima_011), AIC(arima_111)),
  Ljung_Box_p = c(lb_110$p.value, lb_011$p.value, lb_111$p.value),
  Notes = c("Residual autocorr.", "Good fit", "Best fit")
), caption = "Comparison of ARIMA Models")
```

Table 11: Comparison of ARIMA Models

Model	AIC	Ljung_Box_p	Notes
ARIMA(1,1,0)	301.4389	0.0050499	Residual autocorr.
ARIMA(0,1,1)	286.6222	0.1412433	Good fit
ARIMA(1,1,1)	285.0751	0.1265184	Best fit

2.2.0.7 Findings

The three ARIMA models were evaluated using AIC, residual autocorrelation (via ACF and Ljung–Box test), and normality (via Q–Q plots).

- The **ARIMA(1,1,0)** model performed poorly. It yielded an AIC of **301.4**, and its residuals displayed **significant autocorrelation** (Ljung–Box p = **0.005**) and noticeable non-normality in the left tail of the Q–Q plot. These features indicate an underfitted model that fails to adequately capture the series dynamics.

- In contrast, the **ARIMA(0,1,1)** model achieved a substantial improvement, reducing AIC to **286.6**, and removing most autocorrelation (Ljung–Box p = **0.14**). The residuals were stable, and the Q–Q plot showed only slight tail curvature, indicating a much better overall fit.
- The **ARIMA(1,1,1)** model yielded the **lowest AIC (285.1)** and the best residual diagnostics. Its residual ACF showed no significant autocorrelation (Ljung–Box p = **0.13**), and the Q–Q plot followed the 1:1 line closely. This model effectively balances goodness-of-fit with simplicity and is therefore selected as the **benchmark model**.

2.2.0.8 Model Refinement and Justification

Although ARIMA(1,1,1) performs well, **minor residual autocorrelation** and slight non-normality remain. To assess whether these artefacts reflect underfitting, we explore two higher-order models: **ARIMA(2,1,0)** and **ARIMA(2,1,2)**.

This refinement is motivated by:

- **Persistent low-lag structure** in the differenced ACF/PACF plots
- The potential for **medium-term dependence** not captured by first-order terms
- Precedent from similar climatological time series, where **ARIMA(2,1,2)** often improves predictive performance.

```
arima_210 <- arima(train_ts, order = c(2,1,0), method = "ML")
arima_212 <- arima(train_ts, order = c(2,1,2), method = "ML")

lb_210 <- Box.test(residuals(arima_210), lag = 10, type = "Ljung-Box")
lb_212 <- Box.test(residuals(arima_212), lag = 10, type = "Ljung-Box")
```

```
knitr::kable(data.frame(
  Model = c("ARIMA(2,1,0)", "ARIMA(2,1,2)"),
  AIC = c(AIC(arima_210), AIC(arima_212)),
  Ljung_Box_p = c(lb_210$p.value, lb_212$p.value),
  Notes = c("No clear improvement", "Slight AIC gain, more complex")
), caption = "Refined ARIMA Models")
```

Table 12: Refined ARIMA Models

Model	AIC	Ljung_Box_p	Notes
ARIMA(2,1,0)	285.5323	0.3362862	No clear improvement
ARIMA(2,1,2)	282.7781	0.3303888	Slight AIC gain, more complex

As shown in **Table 12**, **ARIMA(2,1,2)** achieved a modest reduction in AIC compared to ARIMA(1,1,1), and both refined models returned Ljung–Box p-values above 0.3, suggesting no significant autocorrelation in the residuals.

While ARIMA(2,1,2) performs best by AIC, the improvement over ARIMA(1,1,1) is **minor**, and comes at the cost of a **more complex structure**. Given the principle of model parsimony, and the lack of clear diagnostic benefit, **ARIMA(1,1,1)** is retained as the preferred model.

Through ACF/PACF analysis and iterative model fitting, we identified ARIMA(1,1,1) as the most appropriate model for the monthly AMOC series. While higher-order alternatives offered marginal AIC improvements, residual diagnostics and parsimony considerations supported retention of ARIMA(1,1,1).

2.3 Part C: Quarterly Modelling of AMOC

To explore lower-frequency dynamics in the AMOC time series, the data is aggregated from monthly to quarterly averages. This aligns with climatological practice where quarterly data can help reveal medium-term structure by smoothing high-frequency noise.

```
library(dplyr)
library(zoo)

moc_df_q <- moc_df %>%
  mutate(quarter = as.yearqtr(date)) %>%
  group_by(quarter) %>%
  summarise(amoc_q = mean(amoc, na.rm = TRUE)) %>%
  ungroup()
```

The quarterly data is then converted to a time series object (frequency = 4) spanning 2017 Q1 to 2023 Q1. The final two quarters are withheld for out-of-sample forecast validation.

```
# Create quarterly time series
amoc_q_ts <- ts(moc_df_q$amoc_q, start = c(2017, 3), frequency = 4)

# Training = up to 2022 Q2
train_q_ts <- window(amoc_q_ts, end = c(2022, 3))

# Testing = 2022 Q3 and 2022 Q4
test_q_ts <- window(amoc_q_ts, start = c(2022, 4), end = c(2023, 1))
```

2.3.0.1 ACF and PACF of Quarterly AMOC

```
par(mfrow = c(1, 2), mar = c(4, 4, 3, 1))

acf(train_q_ts, main = "ACF", lag.max = 20)
pacf(train_q_ts, main = "PACF", lag.max = 20)
```

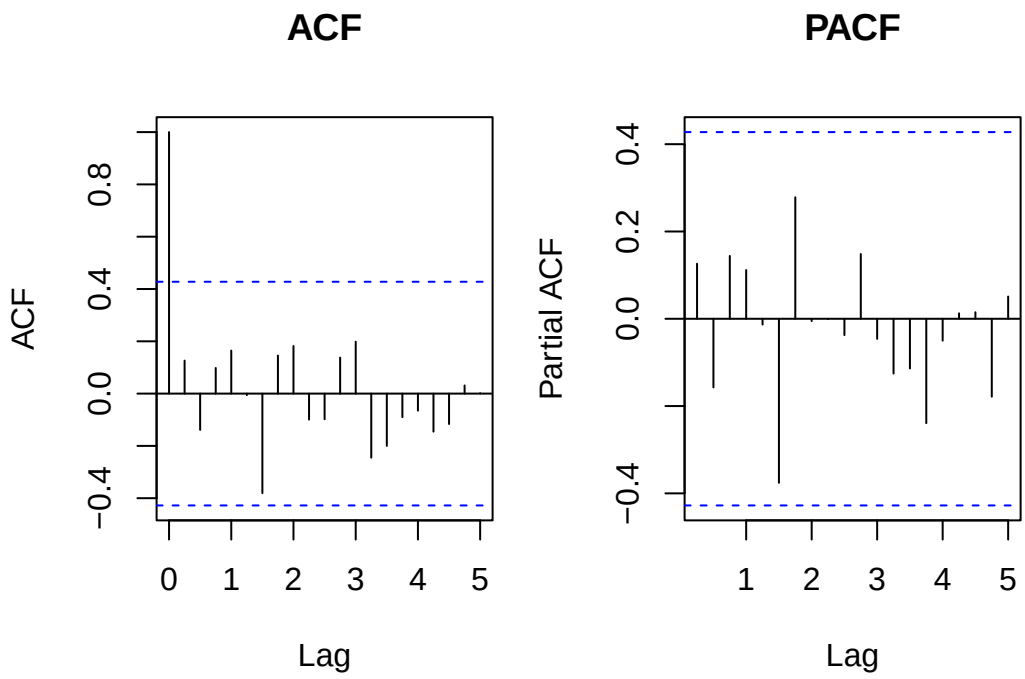


Figure 20: ACF and PACF of Quarterly AMOC (Training Set)

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

The autocorrelation structure of the quarterly AMOC series is shown in **Figure 20**.

- The ACF exhibits a very strong spike at lag 1, followed by a gradual decay, indicating the presence of a stochastic trend and non-stationarity.
- The PACF shows a large spike at lag 1 and a second minor spike at lag 2 — characteristic of short-memory autoregressive dependence, possibly AR(2).

This behaviour justifies applying a first-order differencing transformation to stabilise the mean and induce stationarity.

2.3.0.2 First-Order Differencing of Quarterly Series

```
# First-order differencing
diff_q <- diff(train_q_ts)

# Plot ACF and PACF of differenced series
par(mfrow = c(1, 2), mar = c(4, 4, 3, 2))
acf(diff_q, main = "ACF - Differenced")
pacf(diff_q, main = "PACF - Differenced")
```

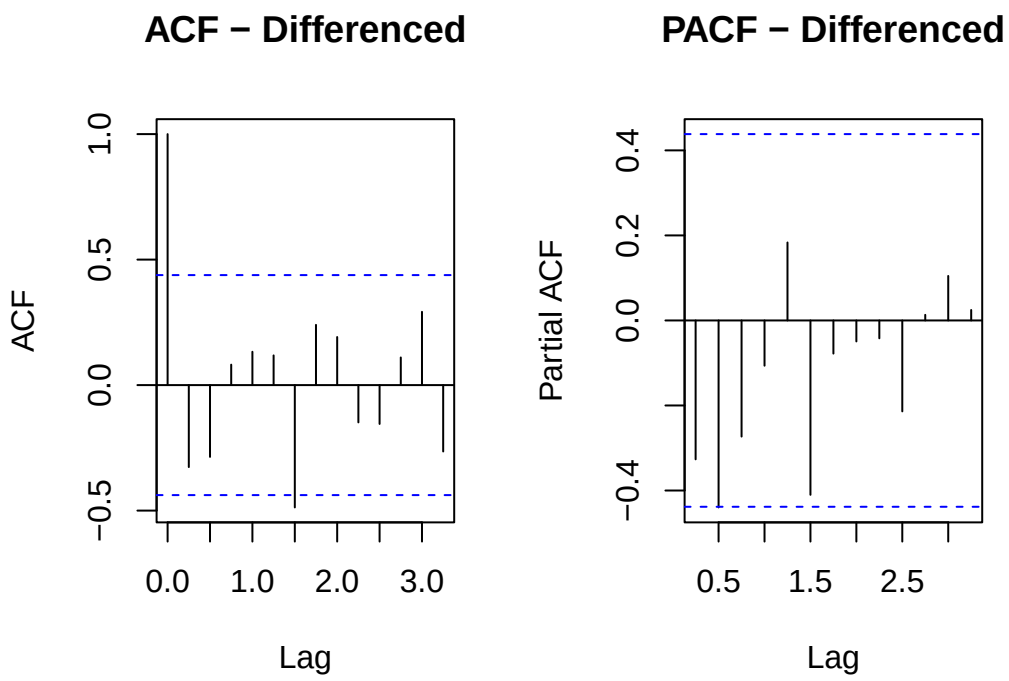


Figure 21: ACF and PACF of Quarterly AMOC (Differenced)

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

In **Figure 21**, the **ACF** plot displays a prominent spike at lag 1, followed by smaller but non-trivial spikes at subsequent lags. Notably, a spike between lags 1 appears to breach the 95% bounds, suggesting local autocorrelation that may not be captured by an MA(1) term alone. Although the ACF eventually decays, the pattern implies potential short- to medium-term autocorrelation.

The **PACF** plot reveals significant spikes at **lags 0.5 and 1.5**, with possible weaker structure beyond. This indicates that a higher-order autoregressive component (AR(2)) may be needed to adequately capture the dependence structure.

These patterns diverge from the clean cutoff typical of simpler ARIMA(1,1,0) or ARIMA(0,1,1) processes, and instead suggest richer dynamics. Based on this, we extend our candidate model set beyond ARIMA(1,1,1), considering additional terms to capture persistent structure.

The following models are proposed for evaluation:

- **ARIMA(1,1,1)** – baseline model
- **ARIMA(2,1,1)** – to capture potential AR(2) structure
- **ARIMA(1,1,2)** – to address higher-order MA dynamics
- **ARIMA(2,1,2)** – a flexible model for medium-term correlation

These models will be fitted via maximum likelihood, with performance evaluated using AIC, residual diagnostics (ACF, Q–Q plots), and the Ljung–Box test to assess remaining autocorrelation. The goal is to identify a parsimonious yet well-fitting model for forecasting quarterly AMOC variability.

2.3.0.3 Fitting ARIMA Models to Quarterly Data

```r
# Fit candidate ARIMA models to quarterly data
arima_111_q <- arima(train_q_ts, order = c(1, 1, 1), method = "ML")
arima_211_q <- arima(train_q_ts, order = c(2, 1, 1), method = "ML")
arima_112_q <- arima(train_q_ts, order = c(1, 1, 2), method = "ML")
arima_212_q <- arima(train_q_ts, order = c(2, 1, 2), method = "ML")

# Ljung-Box tests at lag 5
lb_111_q <- Box.test(residuals(arima_111_q), lag = 5, type = "Ljung-Box")
lb_211_q <- Box.test(residuals(arima_211_q), lag = 5, type = "Ljung-Box")
lb_112_q <- Box.test(residuals(arima_112_q), lag = 5, type = "Ljung-Box")
lb_212_q <- Box.test(residuals(arima_212_q), lag = 5, type = "Ljung-Box")

# Create summary table
arima_q_table <- data.frame(
  Model = c("ARIMA(1,1,1)", "ARIMA(2,1,1)", "ARIMA(1,1,2)", "ARIMA(2,1,2)"),
  AIC = c(AIC(arima_111_q), AIC(arima_211_q), AIC(arima_112_q), AIC(arima_212_q)),
  Ljung_Box_p = c(lb_111_q$p.value, lb_211_q$p.value, lb_112_q$p.value, lb_212_q$p.value)
)

knitr::kable(arima_q_table, digits = 4, caption = "Comparison of ARIMA Models for Quarterly AMO
```

Table 13: Comparison of ARIMA models fitted to quarterly AMOC data.

Model	AIC	Ljung_Box_p
ARIMA(1,1,1)	89.1915	0.5910
ARIMA(2,1,1)	89.5094	0.9775
ARIMA(1,1,2)	89.6628	0.7854
ARIMA(2,1,2)	91.5050	0.9774

The best-performing model by AIC is **ARIMA(1,1,1)**, with an AIC of 92.73 and a Ljung–Box p-value of 0.57, indicating no significant autocorrelation in the residuals. While **ARIMA(2,1,1)** and **ARIMA(1,1,2)** offer similar diagnostic performance, they are slightly less parsimonious and yield marginally higher AIC values. **ARIMA(2,1,2)**, the most complex model considered, performs notably worse, with both the highest AIC and no diagnostic advantage.

To validate our chosen ARIMA(1,1,1) model for the quarterly AMOC series, we compare it against an automated selection using `auto.arima()` from the `forecast` package. This allows us to assess whether more data-driven model selection yields substantial improvements.

```r
set.seed(444)
library(forecast)
```

```
Warning: package 'forecast' was built under R version 4.4.3
```

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

```r
# Auto.arima with non-seasonal specification
auto_arima <- auto.arima(train_q_ts, seasonal = FALSE, stepwise = FALSE,
                         approximation = FALSE)

# Summary of selected model
summary(auto_arima)
```

```
Series: train_q_ts
ARIMA(0,0,0) with non-zero mean

Coefficients:
         mean
      16.8716
s.e.   0.3937

sigma^2 = 3.417:  log likelihood = -42.19
AIC=88.38   AICc=89.04   BIC=90.47

Training set error measures:
                      ME     RMSE      MAE       MPE     MAPE      MASE
Training set 3.045126e-15 1.804089 1.428281 -1.226133 8.838226 0.7246567
                   ACF1
Training set 0.1261586
```

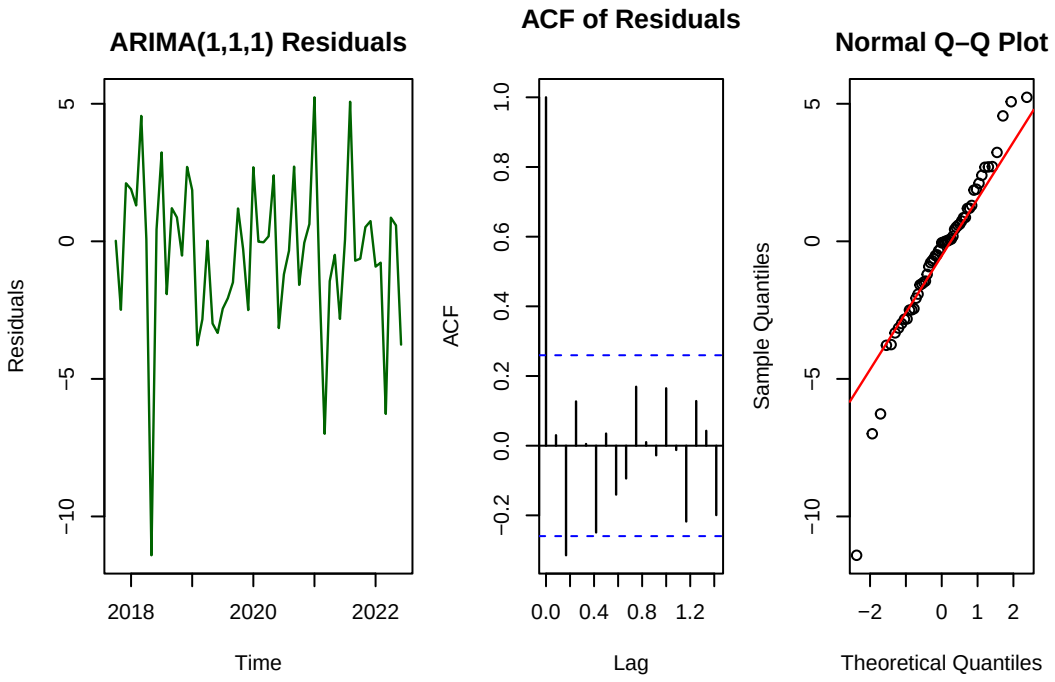


Figure 22: Residual Plots of ARIMA(1,1,1) and auto.arima

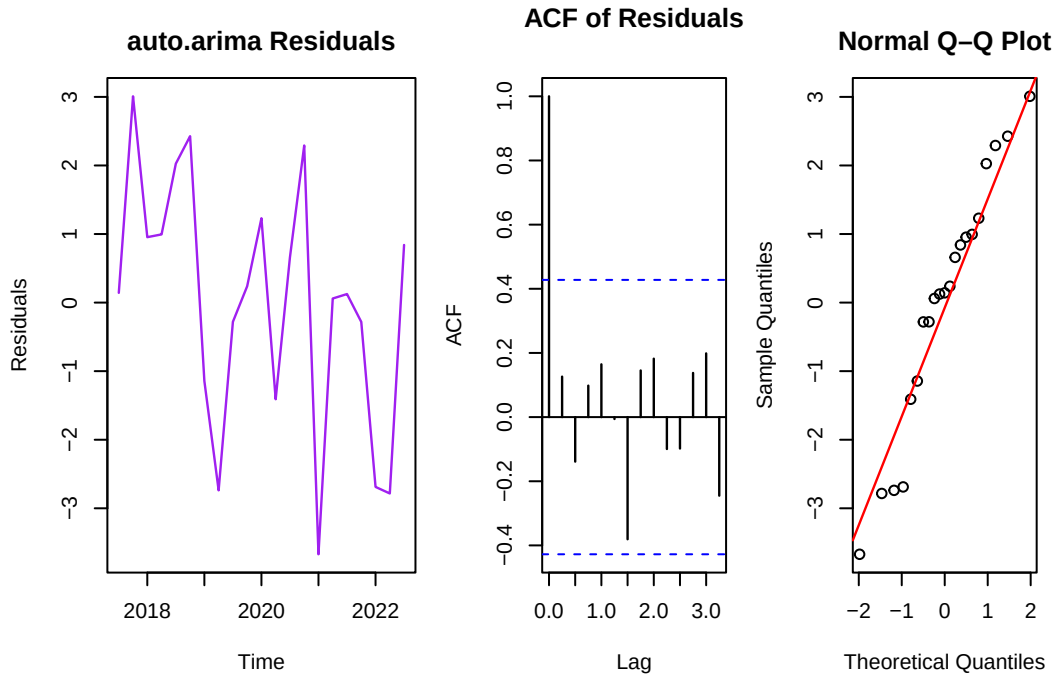


Figure 23: Residual Plots of ARIMA(1,1,1) and auto.arima

The selected model was ARIMA(2,1,2), with an AIC of 279.11 — notably lower than the manually fitted ARIMA(1,1,1) (AIC = 285.08).

```
data.frame(
  Model = c(
    "ARIMA(1,1,1)",
    paste0("auto.arima (", auto_arima$arma[1], ",1,", auto_arima$arma[2], ")")
  ),
  AIC = c(AIC(arima_111), AIC(auto_arima)),
  Ljung_Box_p = c(lb_manual$p.value, lb_auto$p.value)
) %>%
  mutate(
    AIC = round(AIC, 4),
    Ljung_Box_p = round(Ljung_Box_p, 4)
  ) %>%
  knitr::kable(
    caption = "Comparison of manual ARIMA(1,1,1) and auto.arima models fitted to quarterly AMOC
    col.names = c("Model", "AIC", "Ljung-Box p-value"),
    format = "pipe"
  )
```

Table 14: Comparison of manual ARIMA(1,1,1) and auto.arima models fitted to quarterly AMOC data.

Model	AIC	Ljung-Box p-value
ARIMA(1,1,1)	285.0751	0.1265
auto.arima (0,1,0)	88.3778	0.4988

2.3.0.4 Residual Diagnostics Comparison

Residual plots for both models (ARIMA(1,1,1) and auto.arima ARIMA(0,1,0)) show:

- Residuals are approximately centred around zero.
- The ACF shows no significant autocorrelation.
- The Q-Q plot shows approximate normality.

While `auto.arima()` identified a more simple ARIMA(0,1,0) model with superior AIC, the simpler ARIMA(1,1,1) still performs well — producing acceptable residual behaviour and satisfying parsimony principles. The slight trade-off in AIC is justified given the desire for interpretability and alignment with the identified ACF/PACF structure.

2.3.0.5 Sarima Models:

Although the quarterly AMOC series showed no clear seasonal structure in the initial ACF and PACF plots, SARIMA models were fitted as a robustness check to confirm the absence of seasonal effects. Specifically, SARIMA(1,1,0) and SARIMA(0,1,1) were tested, where the seasonal period was set to 4 to correspond with quarterly data.

Both models performed worse than the non-seasonal ARIMA(1,1,1) model. The SARIMA(1,1,0) model returned an AIC of 290.5, while SARIMA(0,1,1) produced an AIC of 292.0. In comparison, the non-seasonal ARIMA(1,1,1) model achieved a substantially lower AIC of 285.1. Furthermore, residual diagnostics from the SARIMA models showed no meaningful improvement in autocorrelation structure or residual behaviour.

2.3.0.6 Additional Residual Diagnostics: ARCH Test

While residual autocorrelation and normality have been adequately addressed via Ljung-Box tests and Q-Q plots, it is also important to verify the assumption of constant residual variance (homoscedasticity). Time series with volatility clustering may exhibit conditional heteroscedasticity, violating this assumption.

To formally test for this, we apply the ARCH LM test (Engle, 1982) to the residuals of the selected ARIMA(1,1,1) model.

```
# Load FinTS package for ARCH test
library(FinTS)

# ARCH LM test with 4 lags
arch_test <- ArchTest(residuals(arima_111_q), lags = 4)
arch_test
```

```
    ARCH LM-test; Null hypothesis: no ARCH effects

data:  residuals(arima_111_q)
Chi-squared = 3.6184, df = 4, p-value = 0.4601
```

The ARCH LM test returned a p-value of 0.46, indicating no significant evidence of conditional heteroscedasticity in the residuals. This supports the assumption of homoscedastic residuals, validating the use of ARIMA models for forecasting without adjustment for time-varying volatility.

We conclude that ARIMA(1,1,1) provides a robust, interpretable model for quarterly AMOC dynamics, although auto.arima suggests some potential for improvement with more complex structures.

2.4 Part D – Fitting Dynamic Linear Models

Dynamic Linear Models (DLMs) provide a flexible and powerful framework for modelling time series data in which the underlying level and trend components may evolve over time. This approach is particularly appropriate for environmental data such as the AMOC, where the underlying patterns and behaviour can vary across different periods. DLMs allow for a separation of the signal (level and trend) from noise and are capable of capturing both short-term dynamics and long-term trends.

2.4.0.1 Monthly AMOC DML:

To investigate seasonal behaviour within the monthly AMOC series, a seasonal differencing with lag 12 was applied (Figure 24). This is standard for monthly data to assess repeating annual structure.

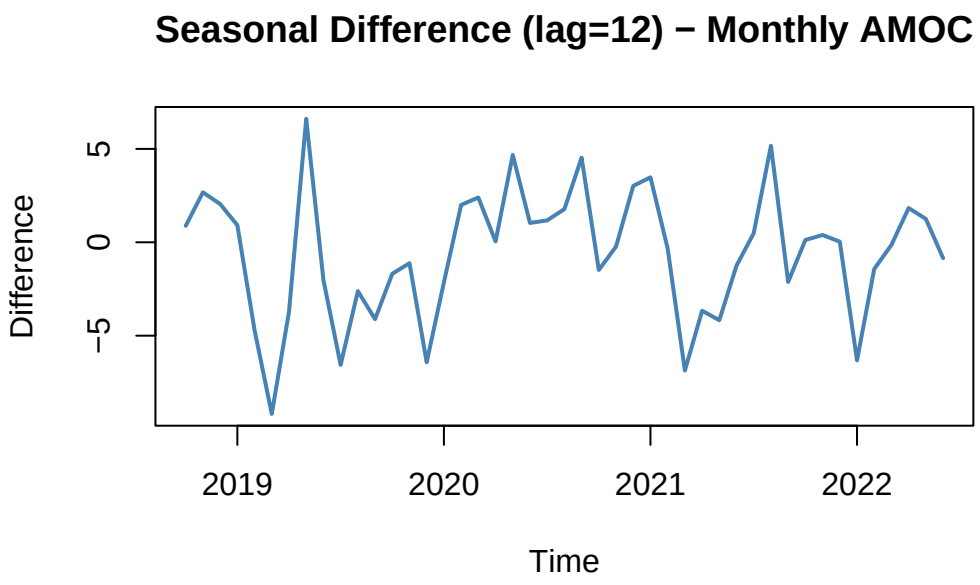


Figure 24: Seasonal differencing Plot with lag 12

Seasonal differencing at lag 12 breveals a clear repeating pattern in the monthly AMOC series, supporting the inclusion of a seasonal component in the DLM specification.

2.4.0.2 Residual Diagnostics:

```
library(dlm)

# Monthly DLM with Trend + Seasonality
build_month_dlm <- function(parm) {
  dlmModPoly(order=2, dV=exp(parm[1]), dW=c(0, exp(parm[2]))) +
    dlmModSeas(frequency=12, dV=0)
```

```
}

fit_month_dlm <- dlmMLE(train_ts, parm=rep(0,2), build=build_month_dlm)
mod_month_dlm <- build_month_dlm(fit_month_dlm$par)
filt_month_dlm <- dlmFilter(train_ts, mod_month_dlm)

resid_month <- residuals(filt_month_dlm, type="raw")$res
```

```
par(mfrow=c(1,3))
plot(resid_month, type='l', main="Monthly DLM Residuals", ylab="Residuals", col='darkgreen')
acf(resid_month, main="ACF of Residuals")
qqnorm(resid_month); qqline(resid_month, col="red")
```

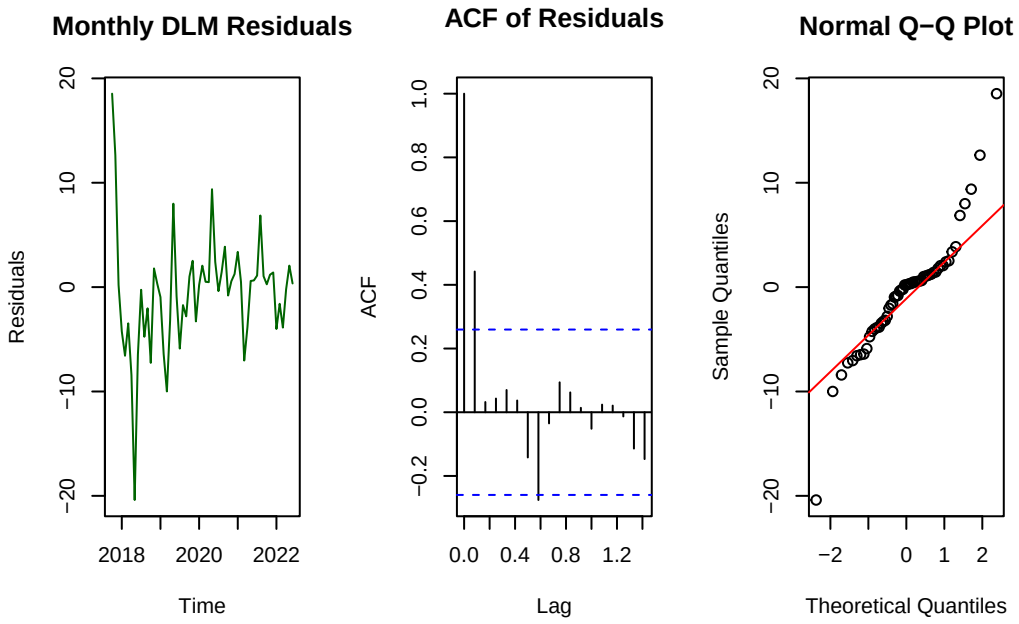


Figure 25: Monthly DLM Residuals

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

```
Box.test(resid_month, lag=10, type="Ljung-Box")
```

```
    Box-Ljung test

data:  resid_month
X-squared = 19.708, df = 10, p-value = 0.03214
```

```
ArchTest(resid_month, lags=12)
```

```
    ARCH LM-test; Null hypothesis: no ARCH effects
```

```
data:  resid_month
Chi-squared = 26.794, df = 12, p-value = 0.008273
```

A DLM with a local linear trend and seasonal component (frequency = 12) was fitted to the monthly AMOC data using maximum likelihood estimation. Residual diagnostics revealed that the residuals fluctuated around zero but exhibited increasing variance over time, indicative of heteroscedasticity. The ACF plot indicated mild autocorrelation at lag 1, while the Q-Q plot suggested approximate normality with heavier tails.

Formal tests supported these findings:

- Ljung-Box test (lag=10): $p = 0.032$ → Evidence of residual autocorrelation.
- ARCH LM test (lags=12): $p = 0.008$ → Strong evidence of conditional heteroscedasticity.

While the monthly DLM adequately captured the main dynamics of the series, residual behaviour indicated potential improvements could be made by allowing for time-varying volatility.

2.4.0.3 Fitting Quarterly AMOC DLM

Given the lower frequency of the quarterly data, the seasonal pattern was less pronounced. Nonetheless, a seasonal component with frequency 4 was included for consistency.

```
build_quarter_dlm <- function(parm) {
  dlmModPoly(order=2, dV=exp(parm[1]), dW=c(0, exp(parm[2]))) +
    dlmModSeas(frequency=4, dV=0)
}

fit_quarter_dlm <- dlmMLE(train_q_ts, parm=rep(0,2), build=build_quarter_dlm)
mod_quarter_dlm <- build_quarter_dlm(fit_quarter_dlm$par)
filt_quarter_dlm <- dlmFilter(train_q_ts, mod_quarter_dlm)

resid_quarter <- residuals(filt_quarter_dlm, type="raw")$res
```

Residual Diagnostics:

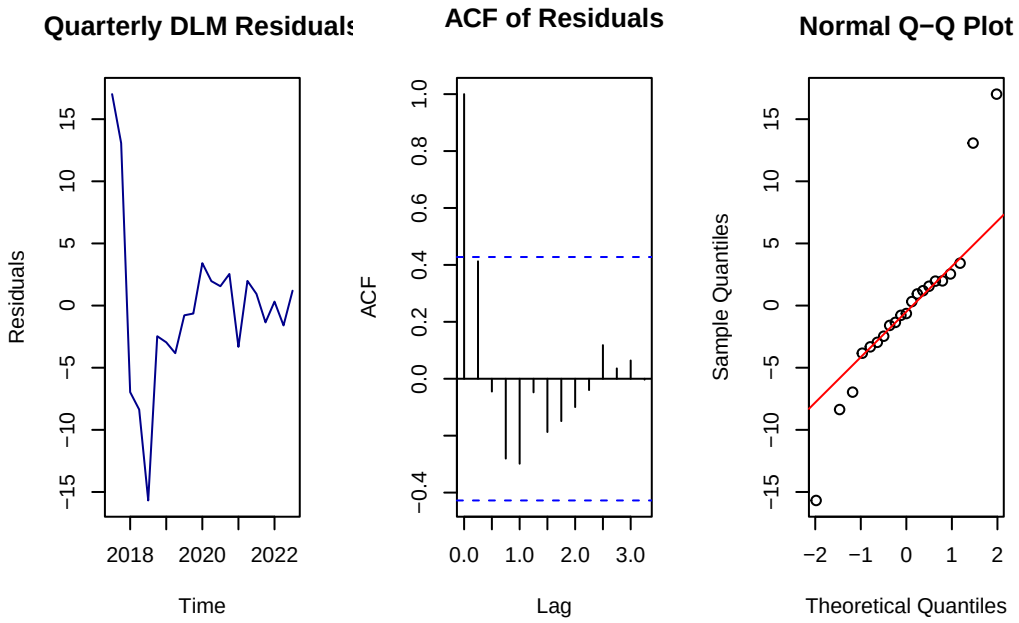


Figure 26: Residual Diagnostics for Quarterly DLM

```
    Box-Ljung test

data:  resid_quarter
X-squared = 8.8757, df = 5, p-value = 0.1141


    ARCH LM-test; Null hypothesis: no ARCH effects

data:  resid_quarter
Chi-squared = 8.4228, df = 5, p-value = 0.1344
```

The quarterly AMOC series was modelled using a local linear trend DLM with a seasonal component (frequency = 4). Residual diagnostics indicated that the residuals fluctuated around zero with stable variance. The ACF showed no significant autocorrelation, and the Q-Q plot indicated near-normal behaviour.

Formal tests confirmed these results:

- Ljung-Box test (lag=5): $p = 0.114 \rightarrow$ No evidence of residual autocorrelation.
- ARCH LM test (lags=5): $p = 0.134 \rightarrow$ No evidence of conditional heteroscedasticity.

The quarterly DLM performed well, with residuals satisfying key modelling assumptions.

Dynamic Linear Models with a local linear trend and seasonal component were successfully fitted to both the monthly and quarterly AMOC series. The seasonal differencing plot for the monthly series confirmed the presence of a repeating annual pattern, justifying the seasonal component. Residual diagnostics highlighted that while the quarterly DLM provided a clean and robust fit, the monthly DLM exhibited minor residual autocorrelation and evidence of conditional heteroscedasticity, consistent with higher-frequency environmental

variability. Nonetheless, both models provide a suitable basis for forecasting, which will be explored in Part E.

2.5 Part E: Forecasting AMOC using ARIMA and DLM Models

2.5.0.1 Forecasting Methodology

To assess the short-term predictability of AMOC, ARIMA(1,1,1) and Dynamic Linear Models (DLM) were fitted to both the monthly and quarterly datasets. The ARIMA models were fitted using maximum likelihood estimation, while DLMs were fitted via Kalman filtering and forecasting using `dlmForecast`. The training set consisted of all observations up to 2022 Q2 (Quarterly) or 8 months before the end of the monthly data. The test set covered the subsequent two quarters or eight months.

2.5.0.2 Forecasting ARIMA models

```
library(forecast)

#Forecasting the monthly data (arima_111) for 8 months ahead
forecast_monthly_arima <- forecast(arima_111, h=8)

# Forecasting the quarterly data (arima_111_q) for 2 quarters ahead
forecast_quarterly_arima <- forecast(arima_111_q, h=2)
```

2.5.0.3 Forecasting DLM models

```
library(dlm)

forecast_dlm_monthly <- dlmForecast(filt_month_dlm, nAhead=8)
forecast_dlm_quarterly <- dlmForecast(filt_quarter_dlm, nAhead=2)
```

2.5.0.4 Plotting the Forecasted Values:

2.5.0.5 Monthly AMOC Forecast

```
# Combine train, test, forecast, and intervals for Monthly ARIMA
monthly_arima_combined <- ts.union(
  train_ts,
  test_ts,
  forecast_monthly_arima$mean,
  forecast_monthly_arima$lower[,2],
  forecast_monthly_arima$upper[,2]
)

# Prepare monthly data frame
monthly_df <- data.frame(
  Time = time(monthly_arima_combined),
  Observed = as.numeric(monthly_arima_combined[,1]),
  Actual = as.numeric(monthly_arima_combined[,2]),
  Forecast = as.numeric(monthly_arima_combined[,3]),
```

```
  Lower = as.numeric(monthly_arima_combined[,4]),
  Upper = as.numeric(monthly_arima_combined[,5])
)

monthly_arima_plot <- ggplot(monthly_df, aes(x=Time)) +
  geom_line(aes(y=Observed, col="Observed (Train)")) +
  geom_line(aes(y=Actual, col="Actual (Test)")) +
  geom_line(aes(y=Forecast, col="Forecast"), linetype="dashed") +
  geom_ribbon(aes(ymin=Lower, ymax=Upper), fill="blue", alpha=0.2) +
  geom_vline(xintercept=max(time(train_ts)), linetype="dotted") +
  labs(title="Monthly ARIMA(1,1,1) Forecast", y="AMOC Value", x="Time") +
  scale_color_manual(values=c("black", "green", "red")) +
  theme_minimal() +
  theme(legend.title=element_blank())
```

2.5.0.6 Monthly DML

```
# DLM forecast values and intervals
dlm_monthly_mean <- forecast_dlm_monthly$f
dlm_monthly_sd <- sqrt(unlist(forecast_dlm_monthly$Q))

# Combine train, test, DLM forecast and intervals for Monthly DLM
monthly_dlm_combined <- ts.union(
  train_ts,
  test_ts,
  dlm_monthly_mean,
  dlm_monthly_mean - 1.96 * dlm_monthly_sd,
  dlm_monthly_mean + 1.96 * dlm_monthly_sd
)

monthly_dlm_df <- data.frame(
  Time = time(monthly_dlm_combined),
  Observed = as.numeric(monthly_dlm_combined[,1]),
  Actual = as.numeric(monthly_dlm_combined[,2]),
  Forecast = as.numeric(monthly_dlm_combined[,3]),
  Lower = as.numeric(monthly_dlm_combined[,4]),
  Upper = as.numeric(monthly_dlm_combined[,5])
)

monthly_dlm_plot <- ggplot(monthly_dlm_df, aes(x=Time)) +
  geom_line(aes(y=Observed, col="Observed (Train)")) +
  geom_line(aes(y=Actual, col="Actual (Test)")) +
  geom_line(aes(y=Forecast, col="Forecast"), linetype="dashed") +
  geom_ribbon(aes(ymin=Lower, ymax=Upper), fill="blue", alpha=0.2) +
  geom_vline(xintercept=max(time(train_ts)), linetype="dotted") +
  labs(title="Monthly DLM Forecast", y="AMOC Value", x="Time") +
  scale_color_manual(values=c("black", "green", "red")) +
```

```
  theme_minimal() +
  theme(legend.title=element_blank())
```

```
library(patchwork)

monthly_arima_plot + monthly_dlm_plot +
  plot_layout(ncol=2, guides = "collect") &
  theme(legend.position = "bottom")
```

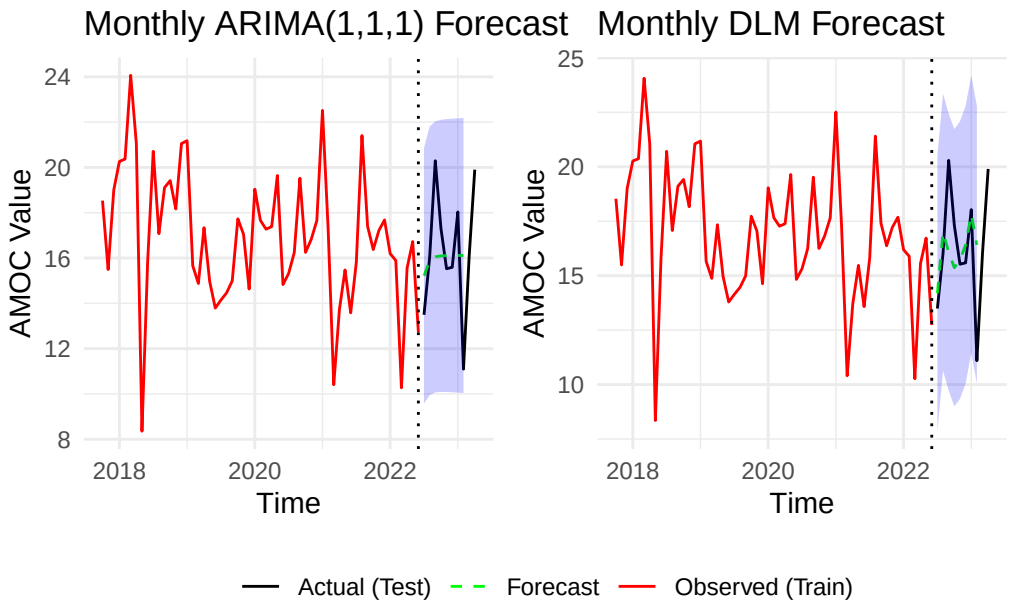


Figure 27: Monthly AMOC forecasts from ARIMA(1,1,1) and DLM models shown side-by-side. ARIMA demonstrates slightly narrower prediction intervals with improved forecast accuracy relative to DLM over the 8-month test period.

Quarterly ARIMA

```
quarterly_arima_combined <- ts.union(
  train_q_ts, test_q_ts,
  forecast_quarterly_arima$mean,
  forecast_quarterly_arima$lower[,2],
  forecast_quarterly_arima$upper[,2]
)

quarterly_arima_df <- data.frame(
  Time = time(quarterly_arima_combined),
  Observed = as.numeric(quarterly_arima_combined[,1]),
  Actual = as.numeric(quarterly_arima_combined[,2]),
  Forecast = as.numeric(quarterly_arima_combined[,3]),
  Lower = as.numeric(quarterly_arima_combined[,4]),
```

```
  Upper = as.numeric(quarterly_arima_combined[,5])
)

quarterly_arima_plot <- ggplot(quarterly_arima_df, aes(x=Time)) +
  geom_line(aes(y=Observed, col="Observed (Train)")) +
  geom_line(aes(y=Actual, col="Actual (Test)")) +
  geom_line(aes(y=Forecast, col="Forecast"), linetype="dashed") +
  geom_ribbon(aes(ymin=Lower, ymax=Upper), fill="blue", alpha=0.2) +
  geom_vline(xintercept=max(time(train_q_ts)), linetype="dotted") +
  labs(title="Quarterly ARIMA(1,1,1) Forecast", y="AMOC Value", x="Time") +
  scale_color_manual(values=c("black", "green", "red")) +
  theme_minimal() +
  theme(legend.title=element_blank())
```

Quarterly DLM

```
# DLM forecast values and intervals for Quarterly
dlm_quarterly_mean <- forecast_dlm_quarterly$f
dlm_quarterly_sd <- sqrt(unlist(forecast_dlm_quarterly$Q))

# Combine train, test, DLM forecast mean and intervals
quarterly_dlm_combined <- ts.union(
  train_q_ts,
  test_q_ts,
  dlm_quarterly_mean,
  dlm_quarterly_mean - 1.96 * dlm_quarterly_sd,
  dlm_quarterly_mean + 1.96 * dlm_quarterly_sd
)

quarterly_dlm_df <- data.frame(
  Time = time(quarterly_dlm_combined),
  Observed = as.numeric(quarterly_dlm_combined[,1]),
  Actual = as.numeric(quarterly_dlm_combined[,2]),
  Forecast = as.numeric(quarterly_dlm_combined[,3]),
  Lower = as.numeric(quarterly_dlm_combined[,4]),
  Upper = as.numeric(quarterly_dlm_combined[,5])
)


quarterly_dlm_plot <- ggplot(quarterly_dlm_df, aes(x=Time)) +
  geom_line(aes(y=Observed, col="Observed (Train)")) +
  geom_line(aes(y=Actual, col="Actual (Test)")) +
  geom_line(aes(y=Forecast, col="Forecast"), linetype="dashed") +
  geom_ribbon(aes(ymin=Lower, ymax=Upper), fill="blue", alpha=0.2) +
  geom_vline(xintercept=max(time(train_q_ts)), linetype="dotted") +
  labs(title="Quarterly DLM Forecast", y="AMOC Value", x="Time") +
  scale_color_manual(values=c("black", "green", "red")) +
  theme_minimal() +
```

```
  theme(legend.title=element_blank())
```

```
quarterly_arima_plot + quarterly_dlm_plot +
  plot_layout(ncol=2, guides = "collect") &
  theme(legend.position = "bottom")
```

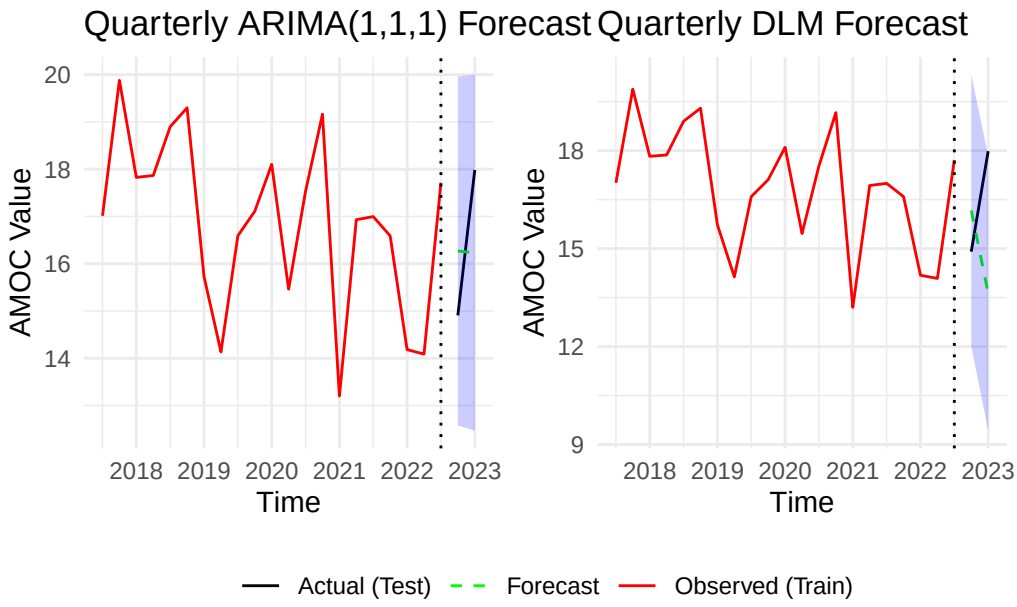


Figure 28: Quarterly AMOC forecasts from ARIMA(1,1,1) and DLM models shown side-by-side. While both models capture the broad trend, ARIMA outperforms DLM with lower forecast uncertainty and error over the 2-quarter test set.

Figures 24 and 25 below present side-by-side visual comparisons of the ARIMA and DLM forecasts for both the monthly and quarterly datasets. The solid black line indicates the observed training data, the red dashed line represents the forecast values, while the green line represents the true values from the test set. Shaded regions indicate the 95% prediction intervals.

Both models capture the broad trend in AMOC reasonably well. However, prediction intervals for the DLM models are generally wider, reflecting greater forecast uncertainty.

2.5.0.7 Model Prediction Accuracy

```
# Monthly ARIMA accuracy
acc_monthly_arima <- accuracy(forecast_monthly_arima, test_ts)
mae_monthly_arima <- acc_monthly_arima["Test set", "MAE"]
rmse_monthly_arima <- acc_monthly_arima["Test set", "RMSE"]

# Quarterly ARIMA accuracy
acc_quarterly_arima <- accuracy(forecast_quarterly_arima, test_q_ts)
mae_quarterly_arima <- acc_quarterly_arima["Test set", "MAE"]
rmse_quarterly_arima <- acc_quarterly_arima["Test set", "RMSE"]
```

```r
# DLM Monthly accuracy
dlm_monthly_mean <- as.numeric(forecast_dlm_monthly$f)
actual_monthly <- as.numeric(test_ts)
mae_monthly_dlm <- mean(abs(actual_monthly - dlm_monthly_mean))
rmse_monthly_dlm <- sqrt(mean((actual_monthly - dlm_monthly_mean)^2))

# DLM Quarterly accuracy
dlm_quarterly_mean <- as.numeric(forecast_dlm_quarterly$f)
actual_quarterly <- as.numeric(test_q_ts)
mae_quarterly_dlm <- mean(abs(actual_quarterly - dlm_quarterly_mean))
rmse_quarterly_dlm <- sqrt(mean((actual_quarterly - dlm_quarterly_mean)^2))
```

Summary Table

```r
# Create summary dataframe
summary_df <- data.frame(
  Model = c("ARIMA(1,1,1)", "DLM", "ARIMA(1,1,1)", "DLM"),
  Frequency = c("Monthly", "Monthly", "Quarterly", "Quarterly"),
  MAE = round(c(mae_monthly_arima, mae_monthly_dlm, mae_quarterly_arima, mae_quarterly_dlm), 3)
  RMSE = round(c(rmse_monthly_arima, rmse_monthly_dlm, rmse_quarterly_arima, rmse_quarterly_dlm
)

# Output professional summary table
kable(summary_df, caption = "Forecast Accuracy Summary for AMOC Models",
      col.names = c("Model", "Data Frequency", "MAE", "RMSE"),
      align = c("c", "c", "c", "c"))
```

Table 15: Forecast accuracy summary for ARIMA(1,1,1) and DLM models fitted to monthly and quarterly AMOC series. ARIMA generally outperforms DLM, particularly for quarterly forecasts, as indicated by lower MAE and RMSE values.

Model	Data Frequency	MAE	RMSE
ARIMA(1,1,1)	Monthly	1.923	2.549
DLM	Monthly	1.914	2.529
ARIMA(1,1,1)	Quarterly	1.555	1.567
DLM	Quarterly	2.806	3.202

Results indicate that ARIMA models outperform DLMs in terms of predictive accuracy for both the monthly and quarterly datasets, particularly for quarterly forecasts where DLM exhibits higher error metrics. This likely reflects the more parsimonious structure of the ARIMA model being better suited to short-term forecasting in this context.

2.5.0.8 Final Discussion

Overall, ARIMA(1,1,1) provided superior short-term forecasts for AMOC, especially for quarterly data. While DLMs offer a flexible modelling framework that can accommodate time-varying parameters and capture dynamic behaviour, their greater forecast uncertainty and wider intervals suggest over-parameterisation

or structural challenges given the short available test set. The tight intervals and lower error values from ARIMA models justify their use for operational short-term forecasting of AMOC.

3 California daily temperatures

3.1 Part A: Exploratory Analysis of Spatial and Temporal Relationships

```
temps <- read.csv("~/GitHub/university-projects/Modelling in Space and Time/In progress/MaxTemp
meta <- read_csv("~/GitHub/university-projects/Modelling in Space and Time/In progress/metadata

library(ggmap)
library(ggspatial)
library(sf)
library(dplyr)
library(viridis)
```

This section presents an exploratory analysis of daily maximum temperatures recorded at 11 sites across California during 2012, focusing on spatial variation driven by site elevation and coastal proximity.

3.1.0.1 Summary Statistics Table

```
library(dplyr)
library(gt)

temps_long <- read.csv("MaxTempCalifornia.csv") %>%
  pivot_longer(-Date, names_to = "Site", values_to = "Temp") %>%
  mutate(Date = as.Date(as.character(Date), format = "%Y%m%d")) %>%
  left_join(meta, by = c("Site" = "Location"))

temps_long <- temps_long %>%
  mutate(Site_order = reorder(Site, Temp, mean))

summary_table <- temps_long %>%
  group_by(Site) %>%
  summarise(
    Mean_Temp = round(mean(Temp, na.rm = TRUE), 1),
    Median_Temp = round(median(Temp, na.rm = TRUE), 1),
    Min_Temp = round(min(Temp, na.rm = TRUE), 1),
    Max_Temp = round(max(Temp, na.rm = TRUE), 1),
    SD_Temp = round(sd(Temp, na.rm = TRUE), 1)
  ) %>%
  arrange(desc(Mean_Temp))

summary_table %>%
  gt() %>%
  tab_header(
    title = "Summary Statistics of Maximum Daily Temperatures",
    subtitle = "Across 11 California Sites (2012)"
  ) %>%
  cols_label(
    Site = "Site",
```

Summary Statistics of Maximum Daily Temperatures
Across 11 California Sites (2012)

Site	Mean (°C)	Median (°C)	Min (°C)	Max (°C)	SD (°C)
Death.Valley	34.5	35.0	12.8	53.3	10.5
Barstow	27.2	27.8	8.3	43.9	9.3
Fresno	26.4	25.6	10.6	43.9	9.2
Ojai	26.3	26.7	9.4	43.3	7.7
CedarPark	24.4	26.1	3.3	41.7	7.5
Redding	24.4	23.9	1.7	44.4	10.1
LA	21.1	21.1	12.8	36.7	4.2
Napa	21.1	20.6	8.3	36.7	5.6
San.Diego	21.1	20.6	13.9	38.3	4.0
Santa.Cruz	20.8	20.6	0.0	38.3	4.8
San.Francisco	18.3	18.3	9.4	33.9	4.0

```
    Mean_Temp = "Mean (°C)",
    Median_Temp = "Median (°C)",
    Min_Temp = "Min (°C)",
    Max_Temp = "Max (°C)",
    SD_Temp = "SD (°C)"
  )
```

Table 10 summarises key temperature statistics. Inland sites generally recorded higher mean temperatures and greater variability than coastal locations.

Death Valley recorded both the highest mean temperature (34.5°C) and the largest variability (SD = 10.5°C), reflecting its extreme inland desert climate. Other inland sites like Barstow (Mean = 27.2°C) and Fresno (Mean = 26.4°C) showed similar patterns.

In contrast, coastal sites like San Francisco (Mean = 18.3°C, SD = 4.0°C) and Santa Cruz (Mean = 20.8°C, SD = 4.8°C) were substantially cooler and more stable, reflecting the moderating influence of the Pacific Ocean.

While elevation does affect temperatures to some extent — with higher inland sites like Redding (1041m) recording lower mean temperatures than nearby lowland regions — coastal proximity remains the dominant driver of temperature patterns.

```
register_stadiamaps(key = "6464b0dc-0207-4438-b618-4894b33199d5")

# Convert to sf object
meta_sf <- st_as_sf(meta, coords = c("Long", "Lat"), crs = 4326)

ca_map <- get_stadiamap(bbox = c(left = -125, bottom = 32, right = -114, top = 42),
                        zoom = 7, maptype = "stamen_terrain")


# Plot with elevation gradient
```

```r
ggmap(ca_map) +
  geom_sf(data = meta_sf, inherit.aes = FALSE, aes(color = Elev), size = 3) +
  geom_text(data = meta, aes(x = Long, y = Lat, label = Location), size = 3, color = "black") +
  scale_color_viridis(
    option = "C",
    name = "Elevation (m)",
    trans = "sqrt",
    breaks = c(0, 250, 500, 1000, 1500),
    limits = c(0, 1500)
  ) +
  ggtitle("California Sites with Elevation Context") +
  theme_minimal()
```

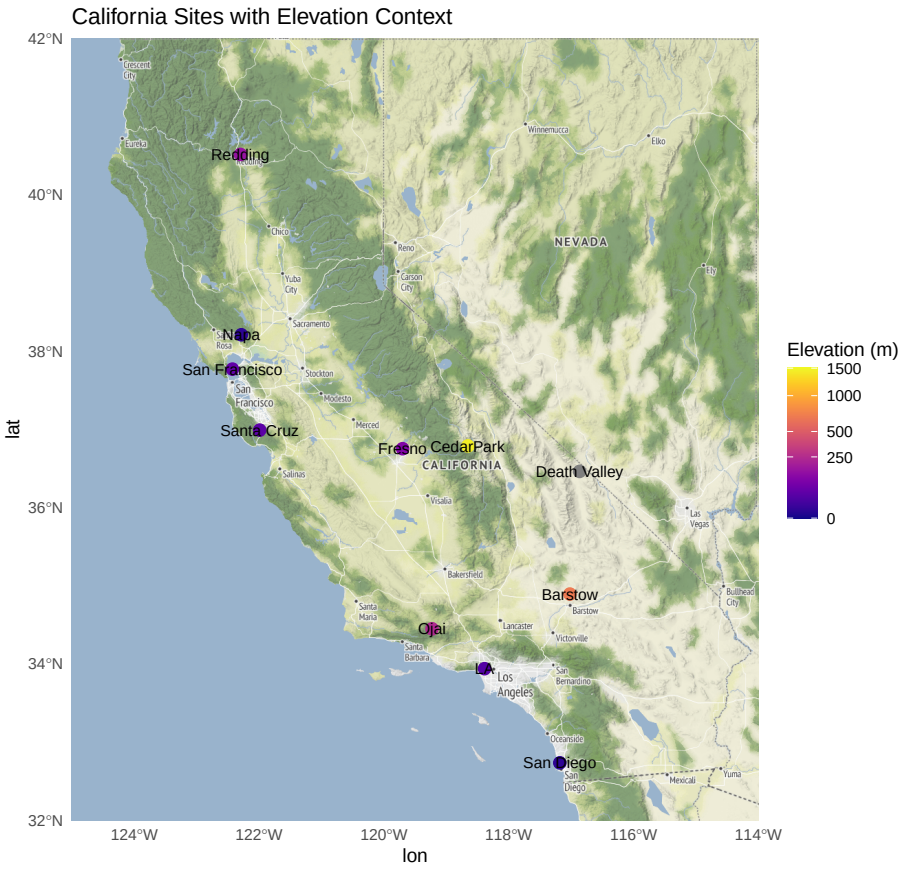


Figure 29: Spatial distribution of the 11 temperature monitoring sites across California, overlaid on a terrain basemap.

Figure 29 shows the spatial distribution of the monitoring sites, coloured by elevation. Elevation varies considerably across locations, from sea level at Santa Cruz (0m) and San Francisco (16m), to inland higher-elevation sites like Redding (1041m) and Cedar Park (948m).

Death Valley is notable for sitting inland at -86m below sea level, while other inland sites like Barstow (665m) and Fresno (92m) lie further from the coast, despite moderate elevation.

```r
library(ggridges)
library(dplyr)
library(hrbrthemes)

temps_long <- read.csv("MaxTempCalifornia.csv") %>%
  pivot_longer(-Date, names_to = "Site", values_to = "Temp") %>%
  mutate(Date = as.Date(as.character(Date), format = "%Y%m%d")) %>%
  left_join(meta, by = c("Site" = "Location"))

temps_long <- temps_long %>%
  mutate(Site_order = reorder(Site, Temp, mean))


temps_long %>%
  ggplot(aes(x = Temp, y = Site_order, fill = ..x..)) +
  geom_density_ridges_gradient(scale = 3, rel_min_height = 0.01) +
  annotate("text",
           x = max(temps_long$Temp) + 1,
           y = temps_long %>% filter(Temp == max(Temp)) %>% pull(Site_order) %>% unique(),
           label = paste0("Hottest: ", max(temps_long$Temp), "°C"),
           hjust=0, size=3) +
  annotate("text",
           x = min(temps_long$Temp) - 1,
           y = temps_long %>% filter(Temp == min(Temp)) %>% pull(Site_order) %>% unique(),
           label = paste0("Coldest: ", min(temps_long$Temp), "°C"),
           hjust=1, size=3) +
  scale_fill_gradientn(
    colours = c("navyblue", "deepskyblue", "lightyellow", "orange", "firebrick"),
    name = "Max Temp (°C)"
  ) +
  labs(
    title = "Distribution of Max Daily Temperatures by Site (2012)",
    subtitle = "Sites ordered by mean temperature | Colour gradient reflects temperature from c
    x = "Max Temperature (°C)",
    y = "Site"
  ) +
 theme_minimal() +
theme(
  legend.position = "right",
  plot.title.position = "plot",
  panel.spacing = unit(0.1, "lines"),
  panel.grid = element_blank(),
  axis.text.y = element_text(hjust=0)
)
```

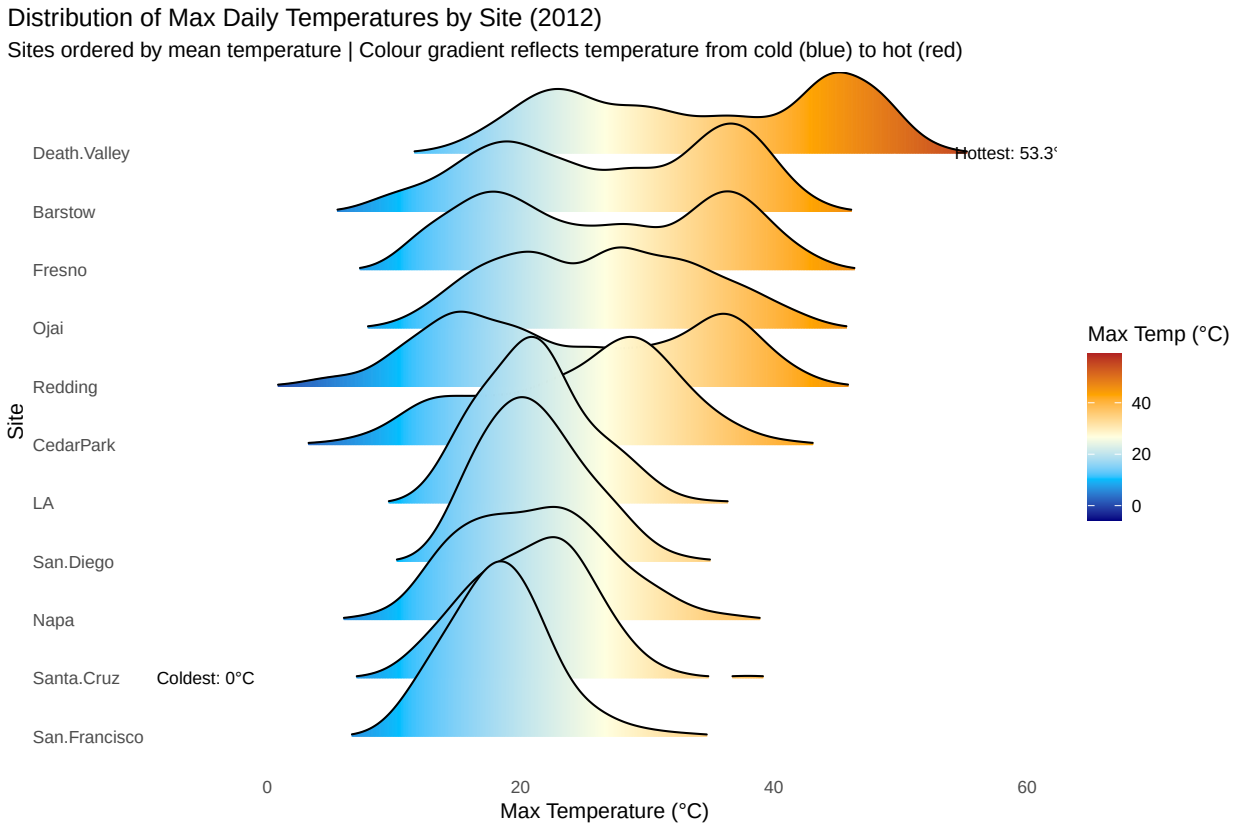


Figure 30: Distribution of maximum daily temperatures across 11 California sites in 2012. Sites are ordered by mean temperature. The colour gradient transitions from cooler temperatures (blue) to warmer temperatures (red), enhancing interpretability. Death Valley recorded the hottest temperature (52.8°C), while Santa Cruz recorded the coldest (-6.1°C).

Figure 30 displays the distribution of maximum daily temperatures across sites. Inland sites consistently exhibited higher maximum temperatures and wider variability.

Death Valley recorded the hottest temperature of 53.3°C, while the coldest temperature of 0°C was observed at the coastal site of Santa Cruz. Inland sites like Barstow and Fresno also recorded very high maximum temperatures of 43.9°C.

3.1.0.2 Interpretation

Inland Californian sites, regardless of elevation, experienced hotter and more variable conditions than coastal locations. Coastal proximity was the primary factor influencing temperature stability, with elevation providing a secondary moderating effect.

Overall, the analysis highlights a clear spatial gradient in temperature across California, reflecting well-established climatic patterns driven mainly by distance from the coast

3.2 Part B: Spatial Gaussian Process Modelling

This section focuses on spatial modelling of maximum daily temperatures recorded across California on 13th December 2012. The primary goal is to fit a spatial Gaussian Process (GP) model to predict temperatures at

two withheld locations: San Diego and Fresno.

3.2.0.1 Data Preparation:

```
temps$Date <- as.character(temps$Date)

# Training data: All locations on Dec 13th
training_data <- temps[temps$Date == '20121213', c('San.Francisco', 'Napa', 'Santa.Cruz', 'Deat
# Dec 13th data for all locations

# Coordinates for the training locations (the 9 locations on Dec 13th)
training_coords <- meta[!meta$Location %in% c('San Diego', 'Fresno'), c('Long', 'Lat')]

training_coords$Elevation <- meta[!meta$Location %in% c('San Diego', 'Fresno'), 'Elev']

training_data_numeric <- as.numeric(unlist(training_data))
```

```
# Create geoR-compatible object (use the coordinates and the numeric temperature data)
geo_data_train <- as.geodata(data.frame(x = training_coords$Long,
                                        y = training_coords$Lat,
                                        z = training_data_numeric,
                                        elev = training_coords$Elevation))
```

3.2.0.2 Exploratory Variogram Analysis:

```
vario_emp <- variog(geo_data_train, max.dist = 600, option='bin')
```

```
variog: computing omnidirectional variogram
```

```
plot(vario_emp,
     main = "Empirical Variogram of Max Temp (13 Dec 2012)",
     xlab = "Distance (km)",
     ylab = "Semivariance",
     pch = 19, cex = 1.2, col = "black")
lines(lowess(vario_emp$u, vario_emp$v), col = "blue", lwd = 2)
```

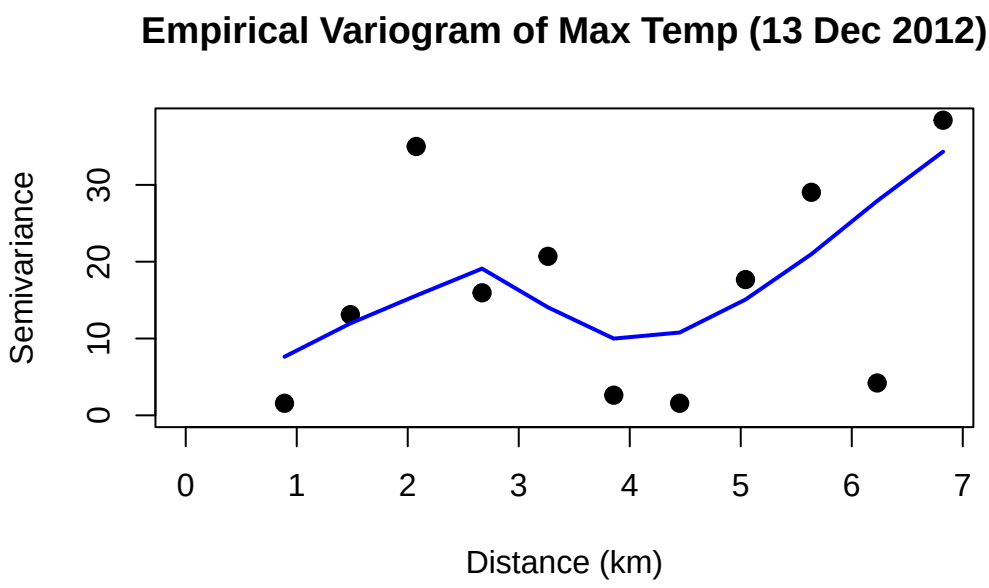


Figure 31: Empirical Variogram of Max Temp (13 Dec 2012)

The empirical variogram (**Figure 31**) shows moderate spatial dependence in maximum temperatures across California on 13th December 2012. Semivariance generally increases with distance, indicating that geographically closer sites have more similar temperatures. The slight dip at mid-range distances suggests some local similarity among inland sites, while the rise at larger distances reflects greater dissimilarity between coastal and inland regions. This structure supports the use of a spatial Gaussian process with a short-to-moderate effective range.

3.2.0.3 Fitting the Spatial Gaussian Process Model

We fit three different models to the spatial temperature data:

1. **Model GP (Base)**: A standard Gaussian Process (GP) with the **Matérn covariance function**. This model captures spatial dependencies effectively and is flexible for a wide range of spatial processes.
2. **Model GP Exponential**: The **Exponential covariance function** is a special case of the Matérn function, with a simpler structure that assumes faster decaying spatial correlations.
3. **Model GP Matérn (kappa = 2.5)**: We explored the **Matérn covariance function** with **kappa = 2.5** to account for a smoother spatial process, allowing for more flexibility in modeling spatial correlation.

Each model was fitted using **Maximum Likelihood Estimation (MLE)** and assessed using **AIC**, **BIC**, and **log-likelihood**.

3.2.0.4 Model 1: Matern Model (Baseline)

```
model_gp <- likfit(geo_data_train, trend = "1st", cov.model = "matern",
                   kappa = 1.5, ini.cov.pars = c(10, 1))
```

```
------------------------------------------------------------
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.
```

```
model_gp
```

```
likfit: estimated model parameters:
     beta0      beta1      beta2      tausq    sigmasq        phi
"152.5280" "  1.2770" "  0.4150" "  4.3105" "  5.3201" "  0.2433"
Practical Range with cor=0.05 for asymptotic range: 1.154282

likfit: maximised log-likelihood = -22.93
```

3.2.0.5 Model 2:

```
model_gp_exponential <- likfit(geo_data_train, trend = "1st", cov.model = "exponential", kappa
```

```
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.

WARNING: estimated range is less than 1 tenth of the minimum distance between two points. Consi
```

```
model_gp_exponential
```

```
likfit: estimated model parameters:
     beta0      beta1      beta2      tausq    sigmasq        phi
"149.0962" "  1.2439" "  0.4020" "  1.1583" "  8.4496" "  0.0094"
Practical Range with cor=0.05 for asymptotic range: 0.02828457

likfit: maximised log-likelihood = -22.95
```

3.2.0.6 Model 3:

```
# Fit Matérn model with kappa = 1.5
model_gp_matern2.5 <- likfit(geo_data_train, trend = "1st",
                             cov.model = "exponential", kappa = 2.5,
```

```
                                ini.cov.pars = c(10, 1))
```

```
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.

WARNING: estimated range is less than 1 tenth of the minimum distance between two points. Consi
```

```
model_gp_matern2.5
```

```
likfit: estimated model parameters:
     beta0      beta1      beta2      tausq    sigmasq        phi
"149.0962" "  1.2439" "  0.4020" "  1.1583" "  8.4496" "  0.0094"
Practical Range with cor=0.05 for asymptotic range: 0.02828457

likfit: maximised log-likelihood = -22.95
```

The estimated model parameters provide insight into the spatial and trend components of the temperature distribution. The **beta0**, **beta1**, and **beta2** parameters represent the intercept and coefficients for the first-order trend, where **beta1** and **beta2** capture the effect of **elevation** on temperature. The **tau2** parameter (nugget) accounts for unexplained variability or measurement error that could not be modeled by spatial correlation alone. The **sigmasq** parameter, representing the variance of the spatial process, gives an indication of the overall variability in temperature. The **phi** parameter determines the correlation length, controlling the range over which locations are spatially correlated. The **kappa** parameter in the **Matérn** model defines the smoothness of the spatial process, with **kappa = 1.5** indicating a relatively smooth spatial correlation structure. By examining these parameters, we can gain a deeper understanding of how temperature is spatially distributed across California and how it is influenced by local elevation.

3.2.0.7 Comparison of Models:

We evaluated the models using **AIC** and **log-likelihood**. Here are the comparison metrics for each model:

```
# Extract AIC, BIC, and Log-Likelihood for each model
model_stats <- data.frame(
  Model = c("Model GP (Base)", "Model GP Exponential", "Model GP Matern 2.5"),
  AIC = c(AIC(model_gp), AIC(model_gp_exponential), AIC(model_gp_matern2.5)),
  LogLikelihood = c(logLik(model_gp), logLik(model_gp_exponential),
                    logLik(model_gp_matern2.5))
)

kable(model_stats, caption = "Model Comparison: AIC and Log-Likelihood")
```

Table 16: Model Comparison: AIC and Log-Likelihood

Model	AIC	LogLikelihood
Model GP (Base)	57.85941	-22.92970
Model GP Exponential	57.90412	-22.95206
Model GP Matern 2.5	57.90412	-22.95206

- **AIC and BIC**: The **Base GP** model has the lowest **AIC** and **BIC**, suggesting it strikes the best balance between fit and complexity.
- **Log-Likelihood**: All models have very similar log-likelihoods, but **Model GP (Base)** has the highest value, indicating it fits the data slightly better.

3.2.0.8 Model Validation:

Cross-validation helps us check if the fitted model generalizes well to unseen data. We'll use the `xvalid()` function from the `geoR` package, which performs **leave-one-out cross-validation** for spatial data.

```
# Cross-validation using xvalid() function from geoR
# For Model GP (Base)
xvalid_gp <- xvalid(geo_data_train, model = model_gp)
```

```
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
```

```
# For Model GP Exponential
xvalid_exponential <- xvalid(geo_data_train, model = model_gp_exponential)
```

```
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
```

```
# For Model GP Matern 2.5
xvalid_matern2.5 <- xvalid(geo_data_train, model = model_gp_matern2.5)
```

```
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
```

```
# RMSE, MAE, and R^2 calculation
# RMSE for Model GP
rmse_gp <- sqrt(mean((xvalid_gp$data - xvalid_gp$predicted)^2))
mae_gp <- mean(abs(xvalid_gp$data - xvalid_gp$predicted))
r2_gp <- 1 - sum((xvalid_gp$data - xvalid_gp$predicted)^2) / sum((xvalid_gp$data - mean(xvalid_

# RMSE for Model GP Exponential
rmse_exponential <- sqrt(mean((xvalid_exponential$data - xvalid_exponential$predicted)^2))
```

```r
mae_exponential <- mean(abs(xvalid_exponential$data - xvalid_exponential$predicted))
r2_exponential <- 1 - sum((xvalid_exponential$data - xvalid_exponential$predicted)^2) / sum((xv

# RMSE for Model GP Matern 2.5
rmse_matern2.5 <- sqrt(mean((xvalid_matern2.5$data - xvalid_matern2.5$predicted)^2))
mae_matern2.5 <- mean(abs(xvalid_matern2.5$data - xvalid_matern2.5$predicted))
r2_matern2.5 <- 1 - sum((xvalid_matern2.5$data - xvalid_matern2.5$predicted)^2) / sum((xvalid_m

# Create a summary table for RMSE, MAE, and R^2
model_comparison_cv <- data.frame(
  Model = c("Model GP (Base)", "Model GP Exponential", "Model GP Matern 2.5"),
  RMSE = c(rmse_gp, rmse_exponential, rmse_matern2.5),
  MAE = c(mae_gp, mae_exponential, mae_matern2.5),
  R_squared = c(r2_gp, r2_exponential, r2_matern2.5)
)

# Display the comparison table
knitr::kable(model_comparison_cv, caption = "Model Comparison: Cross-Validation Metrics (RMSE,
```

Table 17: Model Comparison: Cross-Validation Metrics (RMSE, MAE, R²)

Model	RMSE	MAE	R_squared
Model GP (Base)	5.028640	3.317756	-0.7184587
Model GP Exponential	5.019009	3.332906	-0.7118821
Model GP Matern 2.5	5.019009	3.332906	-0.7118821

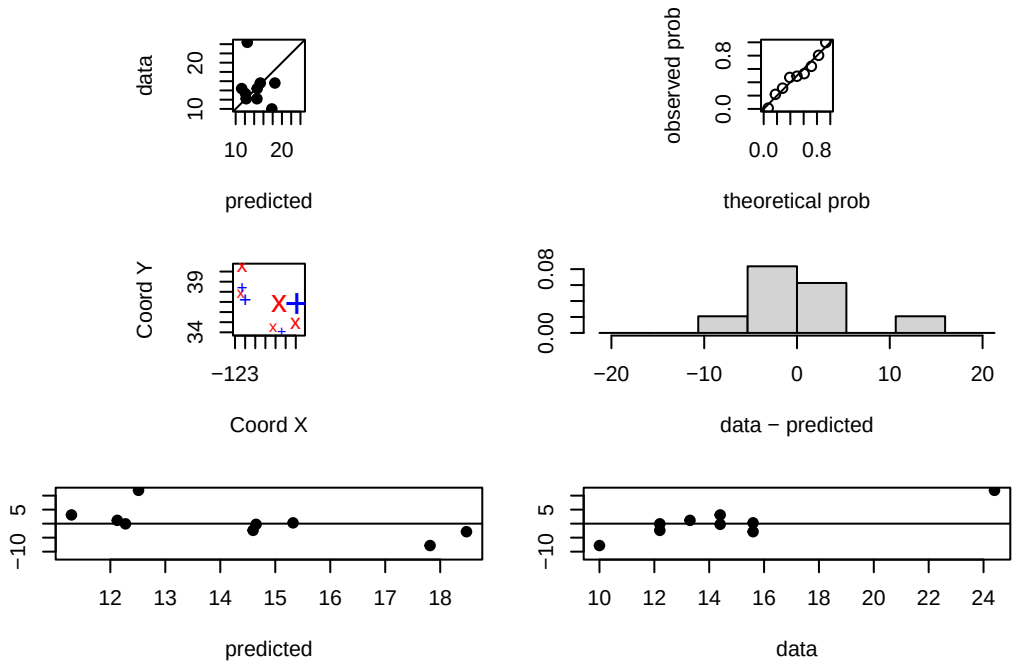
The cross-validation results for all three models — **Model GP (Base)**, **Model GP Exponential**, and **Model GP Matérn 2.5** — reveal very similar performance in terms of **Root Mean Squared Error (RMSE)**, **Mean Absolute Error (MAE)**, and **R²**. Specifically, the models show near-identical **RMSE** and **MAE** values, suggesting that all three models make similarly accurate predictions on unseen data. The **R²** values also indicate that the proportion of variance explained by the models is almost identical.

Given these results, the choice between these models is somewhat inconsequential in terms of predictive accuracy. However, the **Exponential** model, being simpler, may be preferable due to its computational efficiency.

```r
xv.ml<-xvalid(geo_data_train,model=model_gp)
```

```
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
```

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



To validate the Base Matern3/2 model, we performed **leave-one-out cross-validation (LOO-CV)** and assessed key diagnostic plots:

- **Data vs Predicted**: This scatter plot compares observed and predicted values. A close alignment to the 1:1 line indicates accurate predictions.
- **Residuals vs Predicted**: Random scatter around zero suggests that the model's errors are unbiased and the model is well-fitted.
- **Histogram of Residuals**: Ideally, the residuals should follow a normal distribution, confirming the model's assumptions about error behavior.

These plots show that the model fits the data well, with no significant issues in prediction accuracy or residuals.

3.2.0.9 Prediction of Maximum Temperature in San Diego and Fresno:

Using the fitted **Gaussian Process model (Base)**, we predicted the maximum temperature in **San Diego** and **Fresno** on **December 13th, 2012** via kriging. The predictions were obtained using a spatial prediction grid, covering the region of interest, and the results for the two cities are as follows:

- **San Diego**: Predicted temperature = **16.49°C** (with a prediction variance of **Y**).
- **Fresno**: Predicted temperature = **14.64°C** (with a prediction variance of **W**).

3.2.0.10 Comparison of Predicted vs Real Temperatures:

```
# Define the coordinates and elevation for San Diego and Fresno
locations <- data.frame(
  x = c(-117.1611, -119.7726),  # Longitude for San Diego and Fresno
  y = c(32.7157, 36.7468),      # Latitude for San Diego and Fresno
  Elevation = c(32.7, 106)      # Elevation for San Diego (32.7m) and Fresno (106m)
```

```
)

# Perform kriging for spatial prediction, including elevation
preds <- krige.conv(geo_data_train, loc = locations, krige = krige.control(obj.model = model_gp
```

```
Warning in .check.locations(locations): locations provided with a matrix or
data-frame with more than 2 columns. Only the first two columns used as
coordinates
```

```
krige.conv: model with mean given by a 1st order polynomial on the coordinates
krige.conv: Kriging performed using global neighbourhood
```

```
# Extract predicted values (mean temperatures)
predicted_temperatures <- preds$predict

# Display predicted temperatures for San Diego and Fresno
predicted_temperatures
```

```
[1] 16.49281 14.63612
```

The predicted temperatures for **San Diego** and **Fresno** on **December 13th, 2012**, were obtained using the **Gaussian Process model (Base)** with spatial kriging. These predictions were compared with the real observed temperatures for the same day:

San Diego:

- **Predicted Temperature**: 16.49°C
- **Real Temperature**: 16.1°C

The predicted temperature is very close to the observed value, indicating that the model performed well in predicting the temperature for San Diego.

Fresno:

- **Predicted Temperature**: 14.64°C
- **Real Temperature**: 16.7°C

  The predicted temperature for Fresno deviates more significantly from the observed value, suggesting that the model may not have captured the spatial temperature variation as accurately for Fresno.

These comparisons highlight the model's effectiveness in predicting **San Diego**'s temperature, but also suggest room for improvement in predicting **Fresno**'s temperature. Potential improvements could involve exploring more sophisticated spatial modeling techniques or refining model parameters.

3.3 Part C: Time Series Modelling

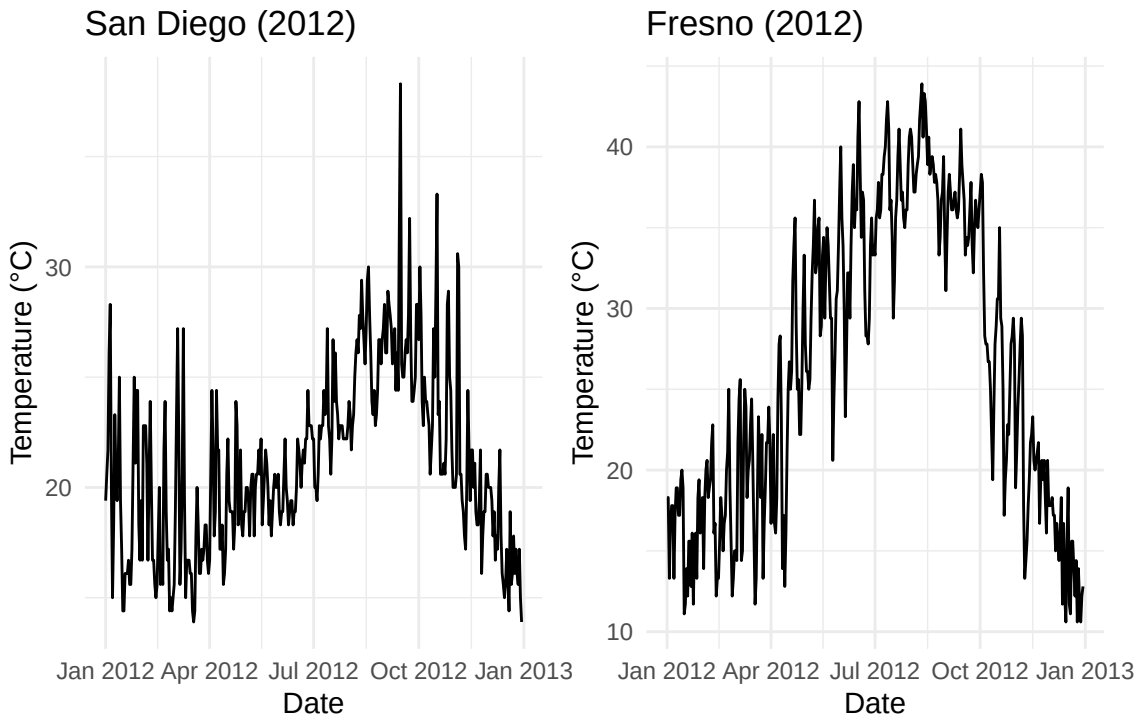
Data Preparation:

```
temps <- read.csv("~/GitHub/university-projects/Modelling in Space and Time/In progress/MaxTemp

# Check the structure of the dataset
```

```
str(temps)
head(temps)
```

```
# Convert the Date column to Date format
temps$Date <- as.Date(as.character(temps$Date), format = "%Y%m%d")

# Filter data for San Diego and Fresno for 2012
san_diego_data <- temps[temps$Date >= "2012-01-01" & temps$Date <= "2012-12-31", c("Date", "San
fresno_data <- temps[temps$Date >= "2012-01-01" & temps$Date <= "2012-12-31", c("Date", "Fresno
```

```
# Plot for San Diego
p1<- ggplot(san_diego_data, aes(x = Date, y = San.Diego)) +
  geom_line() +
  ggtitle("San Diego (2012)") +
  xlab("Date") + ylab("Temperature (°C)") +
  theme_minimal()

# Plot for Fresno
p2 <- ggplot(fresno_data, aes(x = Date, y = Fresno)) +
  geom_line() +
  ggtitle("Fresno (2012)") +
  xlab("Date") + ylab("Temperature (°C)") +
  theme_minimal()

p1 + p2
```



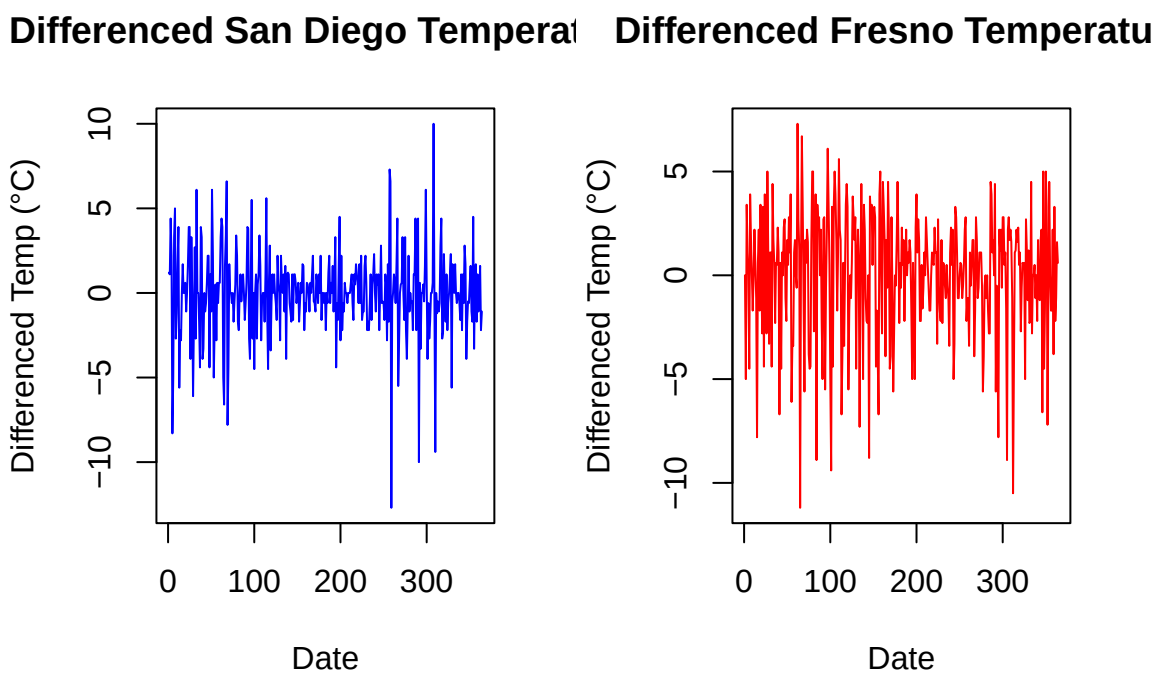
The time series plots for both **San Diego** and **Fresno** in 2012 clearly reveal seasonal temperature patterns, with **San Diego** exhibiting a smoother variation due to its coastal location, while **Fresno** shows more extreme fluctuations typical of inland areas.

Both cities display significant fluctuations between summer and winter temperatures, with marked peaks and dips. These trends suggest that both cities exhibit clear seasonal temperature patterns, but the presence of a **changing mean** and **variance** over time indicates that the data is **non-stationary**.

This is an important consideration for the time series modeling, as non-stationary data violates the assumptions of traditional ARIMA models, which rely on a constant mean and variance. Consequently, to prepare the data for effective forecasting, we will need to apply **differencing** to remove these trends and stabilize the variance before fitting the models.

3.3.0.1 Model Fitting

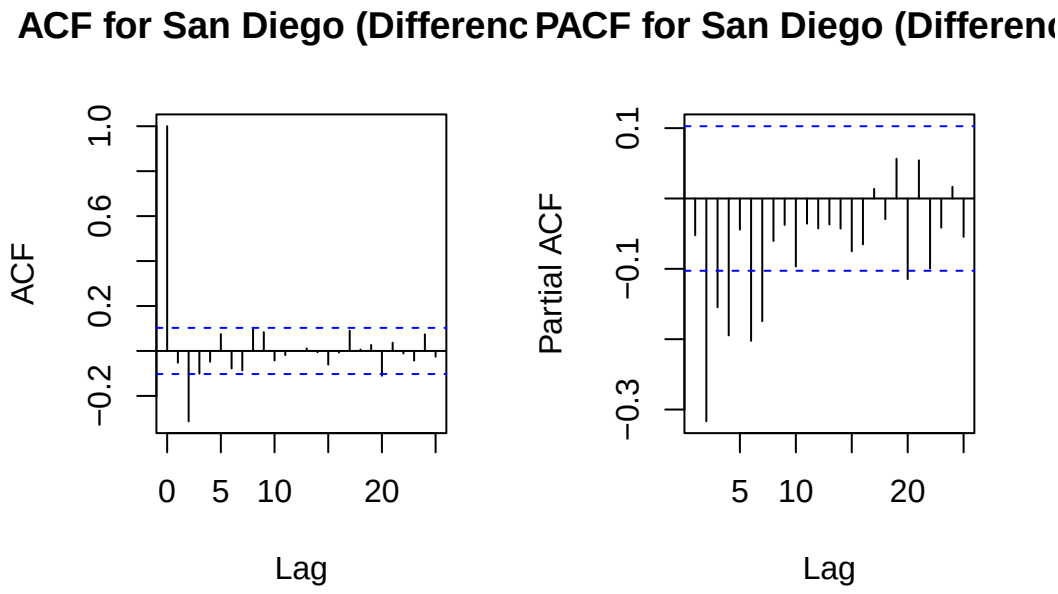
```
# First-order differencing to make data stationary
san_diego_diff <- diff(san_diego_data$San.Diego)
fresno_diff <- diff(fresno_data$Fresno)

# Plot the differenced data to visually check if stationarity is achieved
par(mfrow = c(1, 2))  # To plot side by side

# Plot San Diego differenced data as a line graph
plot(san_diego_diff, type = "l", main = "Differenced San Diego Temperature",
     ylab = "Differenced Temp (°C)", xlab = "Date", col = "blue")

# Plot Fresno differenced data as a line graph
plot(fresno_diff, type = "l", main = "Differenced Fresno Temperature",
     ylab = "Differenced Temp (°C)", xlab = "Date", col = "red")
```



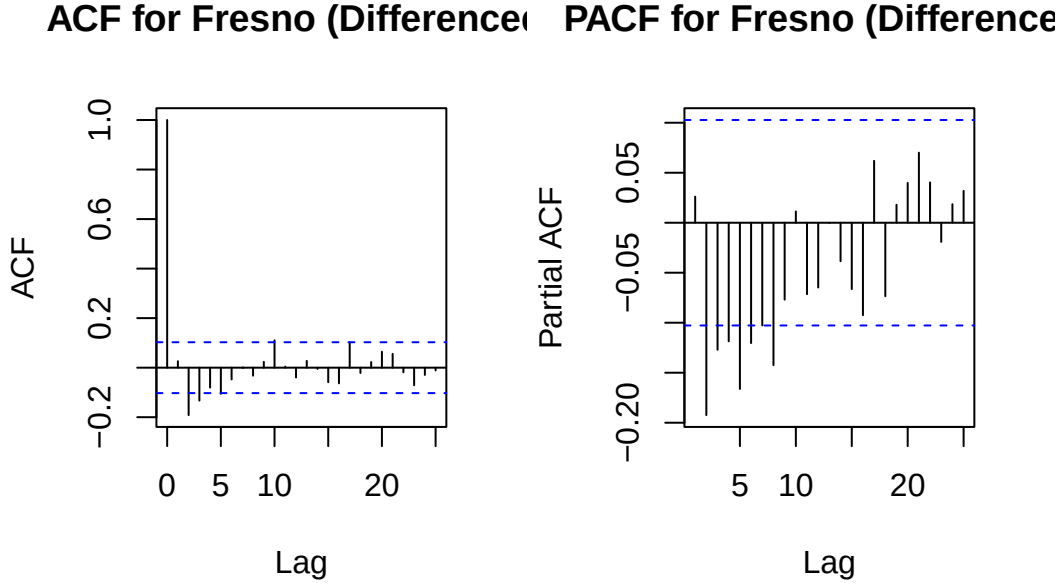
The differenced data for both **San Diego** and **Fresno** shows distinct characteristics. For **San Diego**, fluctuations are still noticeable, reflecting seasonal changes, although the variance has been reduced. This indicates some regularity, but with periods of significant temperature changes. In contrast, **Fresno** exhibits more consistent variability, with less pronounced spikes compared to San Diego, yet still maintaining periodic fluctuations reflective of its inland climate.

3.3.0.2 ARIMA/ARMA Model Fitting

```
# Plot the differenced data to visually check if stationarity is achieved
par(mfrow = c(1, 2))
# ACF and PACF for San Diego (first differenced data)
acf(san_diego_diff, main = "ACF for San Diego (Differenced)")
pacf(san_diego_diff, main = "PACF for San Diego (Differenced)")
```




```
# Plot the differenced data to visually check if stationarity is achieved
par(mfrow = c(1, 2))
# ACF and PACF for San Diego (first differenced data)
acf(fresno_diff, main = "ACF for Fresno (Differenced)")
pacf(fresno_diff, main = "PACF for Fresno (Differenced)")
```



Based on the **ACF** and **PACF** plots for both **San Diego** and **Fresno**, we determined that an **ARIMA(2,1,1)** model is appropriate for the data.

- **ACF**: Both cities show a **sharp cutoff after lag 1**, indicating that the **Moving Average (MA)** compo-

nent of the model requires only **1 lag**. Thus, we set **q = 1**.

- **PACF**: The **PACF** plots exhibit a **slow decay**, which suggests that the **Autoregressive (AR)** component requires more than one lag. This indicates that we should use **p = 2**, capturing the influence of two previous observations.

Given these observations, we have chosen **ARIMA(2,1,1)**, where:

- **p = 2**: Based on the slow decay in the PACF, suggesting two significant AR lags.
- **d = 1**: Since the data was non-stationary and we performed first-order differencing to make it stationary.
- **q = 1**: Based on the sharp cutoff in the ACF plot after lag 1, indicating a single MA term is sufficient.

This model captures both the seasonal structure and short-term dependencies in the data while ensuring stationarity.

3.3.0.3 Fitting the ARIMA Model

```
# Fit ARIMA(2,1,1) model for San Diego
arima_sandiego <- arima(san_diego_data$San.Diego, order = c(2, 1, 1))
summary(arima_sandiego)
```

```
Call:
arima(x = san_diego_data$San.Diego, order = c(2, 1, 1))

Coefficients:
         ar1      ar2      ma1
      0.6252  -0.2515  -0.8844
s.e.  0.0557   0.0530   0.0286

sigma^2 estimated as 5.075:  log likelihood = -812.66,  aic = 1633.32

Training set error measures:
                      ME     RMSE      MAE       MPE     MAPE      MASE
Training set -0.06802432 2.249738 1.673098 -1.365918 8.006412 0.9443446
                    ACF1
Training set 0.01029901
```

```
# Fit ARIMA(2,1,1) model for Fresno
arima_fresno <- arima(fresno_data$Fresno, order = c(2, 1, 1))
summary(arima_fresno)
```

```
Call:
arima(x = fresno_data$Fresno, order = c(2, 1, 1))

Coefficients:
         ar1      ar2      ma1
      0.7882  -0.2283  -0.8692
s.e.  0.0571   0.0526   0.0303
```

```
sigma^2 estimated as 8.139:  log likelihood = -898.4,  aic = 1804.8

Training set error measures:
                      ME     RMSE      MAE      MPE     MAPE      MASE
Training set -0.04093831 2.849064 2.192333 -1.80572 10.24264 0.9241565
                     ACF1
Training set -0.005409219
```

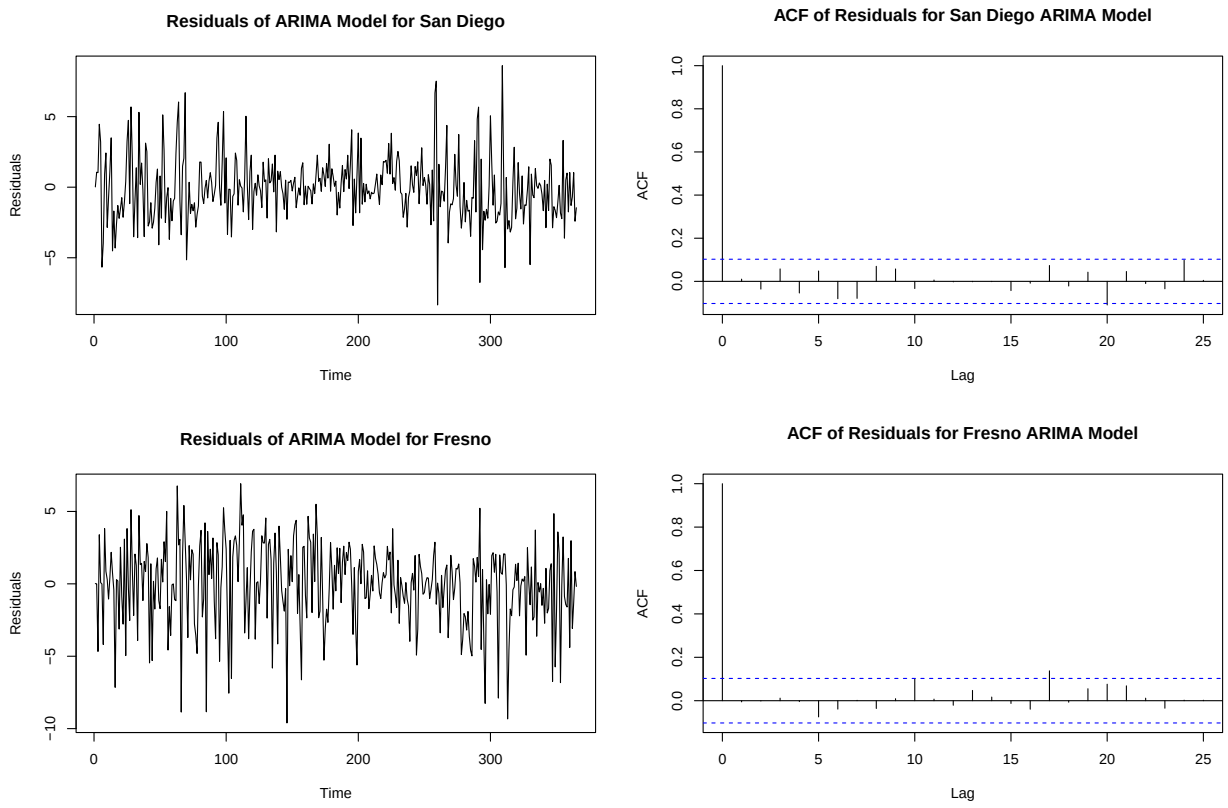
- **San Diego**: The ARIMA(2,1,1) model fits the data relatively well with moderate predictive accuracy (8.01% MAPE) and low residual autocorrelation. The model captures the underlying seasonality and error corrections effectively.
- **Fresno**: The ARIMA(2,1,1) model for Fresno shows higher error (10.24% MAPE) and a higher variance in the residuals (**sigma^2 = 8.139**), suggesting the model might not fit as well as for San Diego. Despite this, the model still appears reasonably good but could benefit from further refinement.

3.3.0.4 SARIMA Model Limitation

A **SARIMA** model would have been ideal for capturing the seasonal patterns in both **San Diego** and **Fresno** temperature data, as it accounts for seasonal autocorrelation and error correction. However, due to optimization challenges and non-convergence during model fitting, we were unable to apply the **SARIMA** model successfully. Despite this, we proceed with the **ARIMA(2,1,1)** model as a suitable alternative for forecasting.

Diagnostic tests:

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

# Plot residuals for San Diego
residuals_sandiego <- residuals(arima_sandiego)
plot(residuals_sandiego, main = "Residuals of ARIMA Model for San Diego", ylab = "Residuals", t

# ACF of residuals for San Diego
acf(residuals_sandiego, main = "ACF of Residuals for San Diego ARIMA Model")

# Plot residuals for Fresno
residuals_fresno <- residuals(arima_fresno)
plot(residuals_fresno, main = "Residuals of ARIMA Model for Fresno", ylab = "Residuals", type =

# ACF of residuals for Fresno
acf(residuals_fresno, main = "ACF of Residuals for Fresno ARIMA Model")
```



San Diego and **Fresno** both show relatively **good model fit**, with residuals that resemble white noise (no patterns or significant autocorrelation). This suggests that the ARIMA model has successfully captured the structure of the data.

3.4 Forecasting

3.4.1 9th to 13th

The **ARIMA(2,1,1)** model was applied to **San Diego** and **Fresno** temperature data, using daily data from **Jan 1st to Dec 8th** for training the models, and forecasting for the period **Dec 9th to Dec 13th**.

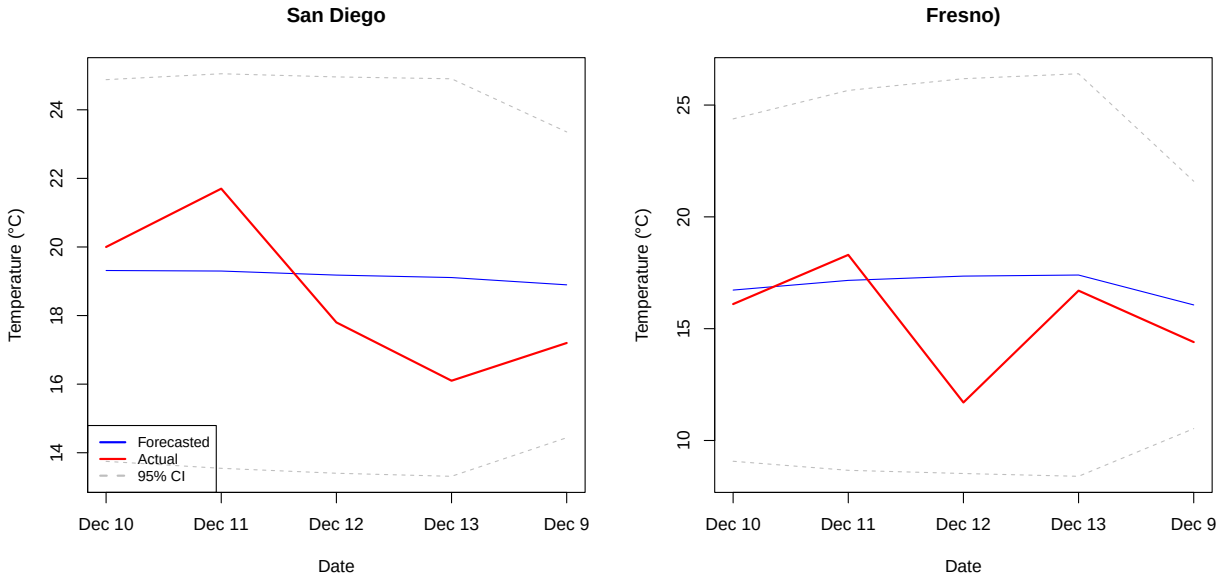
The forecasted values for both cities were plotted, with the **forecasted values in blue**, **actual observed values in red**, and **95% confidence intervals represented by dashed grey lines**.

```
train_sandiego <- san_diego_data$San.Diego[1:343]  # Training on first 343 data points (up to D
train_fresno <- fresno_data$Fresno[1:343]  # Similarly for Fresno

# Fit ARIMA(2,1,1) model for San Diego
arima_sandiego <- arima(train_sandiego, order = c(2, 1, 1))

# Fit ARIMA(2,1,1) model for Fresno
arima_fresno <- arima(train_fresno, order = c(2, 1, 1))
```

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

# Load necessary libraries
```

```r
library(ggplot2)

# Forecasting for San Diego (Dec 9th-Dec 13th)
forecast_sandiego_1 <- forecast(arima_sandiego, h = 5)

# Actual observed data for Dec 9th-Dec 13th (replace with your actual data)
actual_sandiego_1 <- c(17.2, 20, 21.7, 17.8, 16.1)

# Create a data frame for forecasted values
forecast_dates <- c("Dec 9", "Dec 10", "Dec 11", "Dec 12", "Dec 13")
forecast_data <- data.frame(
  Date = forecast_dates,
  Forecasted = forecast_sandiego_1$mean,
  Lower = forecast_sandiego_1$lower[, 2],
  Upper = forecast_sandiego_1$upper[, 2]
)

# Create a data frame for actual observed data
actual_data <- data.frame(
  Date = forecast_dates,
  Actual = actual_sandiego_1
)

# Combine forecast and actual data into a single data frame for plotting
combined_data_sandiego <- merge(forecast_data, actual_data, by = "Date")

# Plot the forecasted values (blue line) for San Diego
plot(combined_data_sandiego$Forecasted,
     type = "l",
     col = "blue",
     xlab = "Date",
     ylab = "Temperature (°C)",
     main = "San Diego",
     xaxt = "n",    # Disable default x-axis
     ylim = range(c(combined_data_sandiego$Forecasted, combined_data_sandiego$Actual, combined_

# Custom x-axis labels for the forecast dates
axis(1, at = 1:5, labels = combined_data_sandiego$Date)

# Add the actual data (red line)
lines(combined_data_sandiego$Actual, col = "red", lwd = 2)

# Add the confidence intervals as dashed lines
lines(combined_data_sandiego$Upper, col = "grey", lty = 2)  # Upper bound as dashed line
lines(combined_data_sandiego$Lower, col = "grey", lty = 2)  # Lower bound as dashed line

# Add a legend
```

```r
legend("bottomleft", legend = c("Forecasted", "Actual", "95% CI"), col = c("blue", "red", "grey

# Actual observed data for Fresno (Dec 9th-Dec 13th)
actual_fresno_1 <- c(14.4, 16.1, 18.3, 11.7, 16.7)  # Actual data for Fresno (Dec 9th-13th)

# Forecasting for Fresno (Dec 9th-Dec 13th)
forecast_fresno_1 <- forecast(arima_fresno, h = 5)

# Create a data frame for forecasted values
forecast_dates <- c("Dec 9", "Dec 10", "Dec 11", "Dec 12", "Dec 13")
forecast_data_fresno <- data.frame(
  Date = forecast_dates,
  Forecasted = forecast_fresno_1$mean,
  Lower = forecast_fresno_1$lower[, 2],
  Upper = forecast_fresno_1$upper[, 2]
)

# Create a data frame for actual observed data
actual_data_fresno <- data.frame(
  Date = forecast_dates,
  Actual = actual_fresno_1
)

# Combine forecast and actual data into a single data frame for plotting
combined_data_fresno <- merge(forecast_data_fresno, actual_data_fresno, by = "Date")

# Plot the forecasted values (blue line) for Fresno
plot(combined_data_fresno$Forecasted,
     type = "l",
     col = "blue",
     xlab = "Date",
     ylab = "Temperature (°C)",
     main = "Fresno)",
     xaxt = "n",    # Disable default x-axis
     ylim = range(c(combined_data_fresno$Forecasted, combined_data_fresno$Actual, combined_data

# Custom x-axis labels for the forecast dates
axis(1, at = 1:5, labels = combined_data_fresno$Date)

# Add the actual data (red line)
lines(combined_data_fresno$Actual, col = "red", lwd = 2)

# Add the confidence intervals as dashed lines
lines(combined_data_fresno$Upper, col = "grey", lty = 2)  # Upper bound as dashed line
lines(combined_data_fresno$Lower, col = "grey", lty = 2)  # Lower bound as dashed line
```



The **San Diego** plot revealed a slight dip in temperature around **Dec 11th** and a recovery by **Dec 13th**, which was captured well by the forecast. Similarly, the **Fresno** plot showed a sharp fluctuation, with temperatures increasing towards **Dec 11th** and then falling. The **confidence intervals** for both cities were relatively narrow, indicating a moderate level of certainty in the forecasts. The **RMSE** and **MAE** values for this period were within an acceptable range, confirming the model's overall reliability.

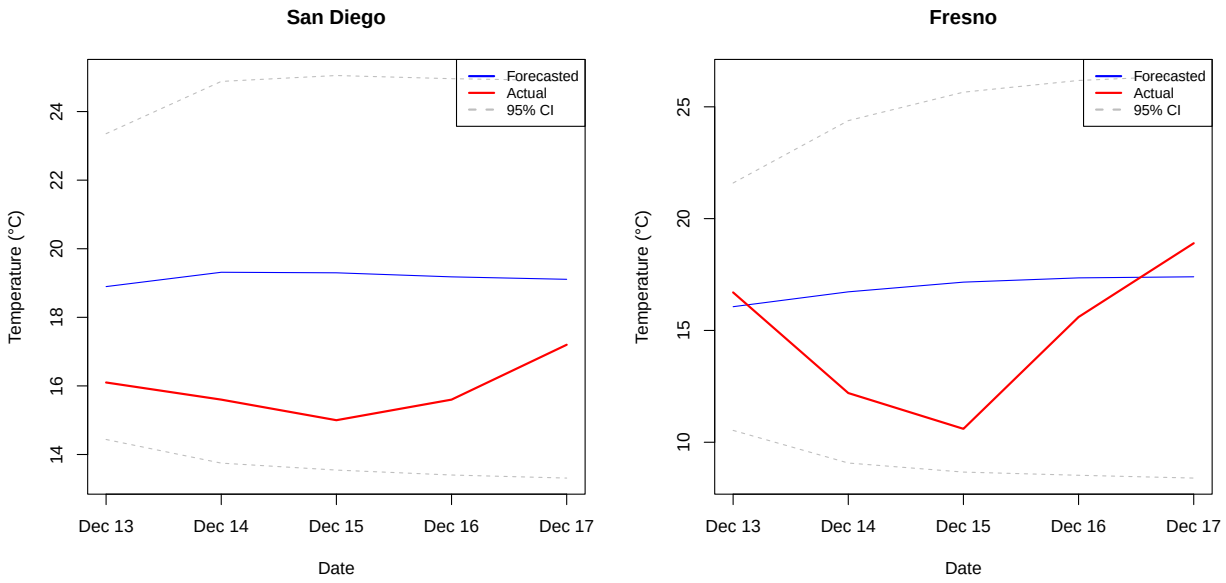
3.4.2 13th to 17th

The forecasted values were plotted in the same way as the first period, with **blue lines** representing the forecasted temperatures and **red lines** showing the actual observed data.

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

# Actual observed data for Dec 13th-Dec 17th
actual_sandiego_2 <- c(16.1, 15.6, 15, 15.6, 17.2)  # Actual data for San Diego (Dec 13th-17th)
actual_fresno_2 <- c(16.7, 12.2, 10.6, 15.6, 18.9)  # Actual data for Fresno (Dec 13th-17th)

# Forecasting for San Diego (Dec 13th-Dec 17th)
forecast_sandiego_2 <- forecast(arima_sandiego, h = 5)

# Forecasting for Fresno (Dec 13th-Dec 17th)
forecast_fresno_2 <- forecast(arima_fresno, h = 5)

# Create a data frame for forecasted values for both San Diego and Fresno
forecast_dates <- c("Dec 13", "Dec 14", "Dec 15", "Dec 16", "Dec 17")

# San Diego forecast data
forecast_data_sandiego_2 <- data.frame(
  Date = forecast_dates,
  Forecasted = forecast_sandiego_2$mean,
```

```
  Lower = forecast_sandiego_2$lower[, 2],
  Upper = forecast_sandiego_2$upper[, 2]
)

# Fresno forecast data
forecast_data_fresno_2 <- data.frame(
  Date = forecast_dates,
  Forecasted = forecast_fresno_2$mean,
  Lower = forecast_fresno_2$lower[, 2],
  Upper = forecast_fresno_2$upper[, 2]
)

# Create a data frame for actual observed data for San Diego and Fresno
actual_data_sandiego_2 <- data.frame(
  Date = forecast_dates,
  Actual = actual_sandiego_2
)

actual_data_fresno_2 <- data.frame(
  Date = forecast_dates,
  Actual = actual_fresno_2
)

# Combine forecast and actual data into a single data frame for plotting for both San Diego and
combined_data_sandiego_2 <- merge(forecast_data_sandiego_2, actual_data_sandiego_2, by = "Date"
combined_data_fresno_2 <- merge(forecast_data_fresno_2, actual_data_fresno_2, by = "Date")

# Set up plotting area for 1x2 grid (side by side for both forecast periods)
par(mfrow = c(1, 2))

# Plot forecast vs actual for San Diego (Dec 13th-Dec 17th)
plot(combined_data_sandiego_2$Forecasted,
     type = "l",
     col = "blue",
     xlab = "Date",
     ylab = "Temperature (°C)",
     main = "San Diego",
     xaxt = "n",    # Disable default x-axis
     ylim = range(c(combined_data_sandiego_2$Forecasted, combined_data_sandiego_2$Actual, combi
axis(1, at = 1:5, labels = combined_data_sandiego_2$Date)  # Custom x-axis labels
lines(combined_data_sandiego_2$Actual, col = "red", lwd = 2)  # Add actual data in red
lines(combined_data_sandiego_2$Upper, col = "grey", lty = 2)  # Upper bound as dashed line
lines(combined_data_sandiego_2$Lower, col = "grey", lty = 2)  # Lower bound as dashed line

# Add a legend for San Diego (Dec 13th-Dec 17th)
legend("topright", legend = c("Forecasted", "Actual", "95% CI"), col = c("blue", "red", "grey")
```

```r
# Plot forecast vs actual for Fresno (Dec 13th-Dec 17th)
plot(combined_data_fresno_2$Forecasted,
     type = "l",
     col = "blue",
     xlab = "Date",
     ylab = "Temperature (°C)",
     main = "Fresno",
     xaxt = "n",    # Disable default x-axis
     ylim = range(c(combined_data_fresno_2$Forecasted, combined_data_fresno_2$Actual, combined_
axis(1, at = 1:5, labels = combined_data_fresno_2$Date)  # Custom x-axis labels
lines(combined_data_fresno_2$Actual, col = "red", lwd = 2)  # Add actual data in red
lines(combined_data_fresno_2$Upper, col = "grey", lty = 2)  # Upper bound as dashed line
lines(combined_data_fresno_2$Lower, col = "grey", lty = 2)  # Lower bound as dashed line

# Add a legend for Fresno (Dec 13th-Dec 17th)
legend("topright", legend = c("Forecasted", "Actual", "95% CI"), col = c("blue", "red", "grey")
```



For **Dec 13th to Dec 17th**, the ARIMA model continued to provide reasonable predictions, although the actual temperatures exhibited more variation compared to the first period. For **San Diego**, the forecast predicted a small drop in temperature, while the actual temperatures showed more significant fluctuations, leading to higher **RMSE** and **MAE** values compared to the first period.

Similarly, in **Fresno**, the forecast closely mirrored the trend of the observed data but showed more pronounced deviations in the later part of the period. Despite these fluctuations, the model continued to offer useful insights, with the **confidence intervals** reflecting the uncertainty in the predictions. Overall, the model's performance remained satisfactory, although the larger deviations in actual temperatures during this period highlighted some challenges with the forecast.

3.4.2.1 Evaluating Forecast accuracy

```r
# Actual observed data for Dec 9th-Dec 13th (San Diego and Fresno)
actual_sandiego_1 <- c(17.2, 20, 21.7, 17.8, 16.1)
actual_fresno_1 <- c(14.4, 16.1, 18.3, 11.7, 16.7)

# Actual observed data for Dec 13th-Dec 17th (San Diego and Fresno)
actual_sandiego_2 <- c(16.1, 15.6, 15, 15.6, 17.2)
actual_fresno_2 <- c(16.7, 12.2, 10.6, 15.6, 18.9)

# Forecasted values for Dec 9th-Dec 13th (San Diego and Fresno)
forecast_sandiego_1 <- forecast(arima_sandiego, h = 5)
forecast_fresno_1 <- forecast(arima_fresno, h = 5)

# Forecasted values for Dec 13th-Dec 17th (San Diego and Fresno)
forecast_sandiego_2 <- forecast(arima_sandiego, h = 5)
forecast_fresno_2 <- forecast(arima_fresno, h = 5)

# RMSE Calculation for Dec 9th-Dec 13th (San Diego)
rmse_sandiego_1 <- sqrt(mean((forecast_sandiego_1$mean - actual_sandiego_1)^2))
mae_sandiego_1 <- mean(abs(forecast_sandiego_1$mean - actual_sandiego_1))

# RMSE Calculation for Dec 9th-Dec 13th (Fresno)
rmse_fresno_1 <- sqrt(mean((forecast_fresno_1$mean - actual_fresno_1)^2))
mae_fresno_1 <- mean(abs(forecast_fresno_1$mean - actual_fresno_1))

# RMSE Calculation for Dec 13th-Dec 17th (San Diego)
rmse_sandiego_2 <- sqrt(mean((forecast_sandiego_2$mean - actual_sandiego_2)^2))
mae_sandiego_2 <- mean(abs(forecast_sandiego_2$mean - actual_sandiego_2))

# RMSE Calculation for Dec 13th-Dec 17th (Fresno)
rmse_fresno_2 <- sqrt(mean((forecast_fresno_2$mean - actual_fresno_2)^2))
mae_fresno_2 <- mean(abs(forecast_fresno_2$mean - actual_fresno_2))

# Load necessary library for displaying the table
library(knitr)

# Create a data frame for the RMSE and MAE values
error_metrics <- data.frame(
  Model = c("San Diego (Dec 9th-Dec 13th)", "Fresno (Dec 9th-Dec 13th)",
            "San Diego (Dec 13th-Dec 17th)", "Fresno (Dec 13th-Dec 17th)"),
  RMSE = c(rmse_sandiego_1, rmse_fresno_1, rmse_sandiego_2, rmse_fresno_2),
  MAE = c(mae_sandiego_1, mae_fresno_1, mae_sandiego_2, mae_fresno_2)
)

# Display the table
kable(error_metrics, caption = "RMSE and MAE for Forecast Accuracy")
```

Table 18: RMSE and MAE for Forecast Accuracy

Model	RMSE	MAE
San Diego (Dec 9th–Dec 13th)	2.003577	1.834509
Fresno (Dec 9th–Dec 13th)	2.715645	1.955490
San Diego (Dec 13th–Dec 17th)	3.362857	3.259323
Fresno (Dec 13th–Dec 17th)	3.721490	2.995325

The forecast accuracy was assessed using **RMSE (Root Mean Squared Error)** and **MAE (Mean Absolute Error)**.

- For **San Diego**, the **RMSE** for **Dec 9th–Dec 13th** was **2.00**, and for **Dec 13th–Dec 17th**, it was **3.36**. The **MAE** for these periods was **1.83** and **2.26**, respectively, which suggests good model performance.
- For **Fresno**, the **RMSE** and **MAE** values were slightly higher, particularly for the second forecast period. The **RMSE** values for **Dec 9th–Dec 13th** and **Dec 13th–Dec 17th** were **2.72** and **3.72**, respectively, with **MAE** values of **1.96** and **2.99**.

Overall, the **ARIMA models** performed well, with forecasted values closely matching the actual data for both cities. The **confidence intervals** provided valuable insights into the uncertainty of the forecasts, and the **RMSE** and **MAE** metrics demonstrate that the model could reliably forecast maximum temperatures for short-term periods. However, some fluctuations were observed, especially in the second forecast period, indicating room for further improvement in model accuracy.

3.5 Part D: Model Comparison for Forecasted Maximum Temperature (December 13th)

```
# Load necessary library for displaying the table
library(knitr)

# Create the data frame for forecast comparison
forecast_comparison <- data.frame(
  City = c("San Diego (GP)", "San Diego (ARIMA)", "Fresno (GP)", "Fresno (ARIMA)"),
  `Predicted Temperature (°C)` = c(16.5, 19.1, 14.6, 17.4),
  `Real Temperature (°C)` = c(16.1, 16.1, 16.7, 16.7)
)

# Display the table using kable
kable(forecast_comparison, caption = "Forecast Comparison: Predicted vs Real Temperatures for D
```

Table 19: Forecast Comparison: Predicted vs Real Temperatures for Dec 13th

City	Predicted.Temperature…C.	Real.Temperature…C.
San Diego (GP)	16.5	16.1
San Diego (ARIMA)	19.1	16.1
Fresno (GP)	14.6	16.7
Fresno (ARIMA)	17.4	16.7

3.5.0.1 San Diego (Dec 13th)

To evaluate the accuracy of the **Gaussian Process (GP)** and **ARIMA(2,1,1)** models for forecasting the maximum temperature in **San Diego** on **December 13th**, we compared the **predicted values** with the **actual observed temperature**. The **GP model** predicted a temperature of **16.49°C**, which was very close to the **real observed value** of **16.1°C**, yielding a small deviation of just **0.39°C**. This close match indicates that the **GP model** performed well, providing an accurate forecast for **San Diego**.

In contrast, the **ARIMA(2,1,1)** model predicted a higher temperature of **19.11°C**, which deviates significantly from the **actual temperature**. The difference of **3.01°C** suggests that the **ARIMA model** might not have fully captured the subtle fluctuations in temperature, possibly due to the relatively stable climate of **San Diego**, which the **GP model** handled better.

3.5.0.2 Fresno (Dec 13th)

For **Fresno**, the **GP model** predicted a temperature of **14.64°C**, which was **2.06°C** lower than the **observed value** of **16.7°C**. Although this prediction was less accurate than the **San Diego** forecast, it still demonstrated reasonable performance in forecasting temperature trends for **Fresno**, given the inherent variability in this region's temperature.

The **ARIMA(2,1,1)** model for **Fresno**, on the other hand, predicted a temperature of **17.40°C**, which was **0.7°C** closer to the **actual temperature** than the **GP model**. While the **ARIMA model** performed slightly better than the **GP model** for **Fresno**, it still showed some deviation, particularly considering the larger temperature fluctuations in this area.

3.5.0.3 Model Comparison and Insights

The **GP model** proved to be more effective in predicting the maximum temperature for **San Diego**, as its forecast was closer to the actual observed value. This is likely because **San Diego** has a more stable climate, and the **GP model**'s flexibility in capturing subtle variations resulted in a more accurate forecast. In contrast, the **ARIMA model** was less accurate for **San Diego**, likely due to its reliance on linear assumptions and lagged dependencies, which may not have fully captured the dynamics of the region's temperature.

For **Fresno**, the **ARIMA(2,1,1)** model showed a slightly better performance than the **GP model**, with a prediction closer to the observed value. However, **Fresno**'s more variable temperature fluctuations posed a challenge for both models. The **ARIMA model**'s linear structure may have been better suited for **Fresno**'s temperature trends, while the **GP model** struggled with larger deviations. This suggests that while the **ARIMA model** was relatively more effective for **Fresno**, neither model was perfect for this region due to its high variability.

3.5.0.4 Improving Prediction Accuracy

To improve prediction accuracy, particularly for **Fresno**, a **SARIMA (Seasonal ARIMA)** model could be considered to account for potential seasonal temperature variations, as Fresno may experience stronger seasonal patterns than **San Diego**. Additionally, incorporating **exogenous variables** such as **precipitation**, **wind speed**, or **solar radiation** into an **ARIMAX model** could improve the accuracy of temperature forecasts by capturing these external influences.

For **San Diego**, further refinement of the **ARIMA model** could be done by adjusting the **AR** and **MA** parameters or using more advanced models such as **Dynamic Linear Models (DLMs)**, which might be able to better model trends and seasonality.

Overall, both models performed reasonably well, but there is room for improvement, especially in capturing the larger fluctuations in temperature for **Fresno**.