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"
  )
```

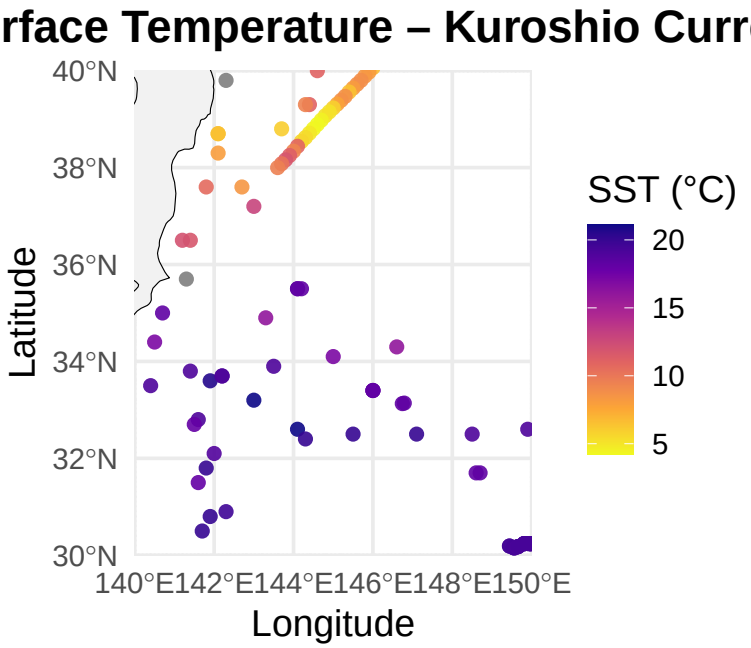


Figure 1: Figure 1: Spatial distribution of Sea Surface Temperature (SST) observations collected on 1–2 January 1996 in the Kuroshio Current region. Each point represents an individual measurement; colour denotes temperature, with warmer SSTs concentrated in the north-east band.

```
# plot(kuroshio100_clean)
```

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 functions such as `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.

```r
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:

```r
# 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

```r
library(geoR)

# Convert training dataset into a geodata object
# kuro_geo_train <- as.geodata(kuroshio_train, coords.col = c("lon", "lat"), data.col = "sst")

# 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
```

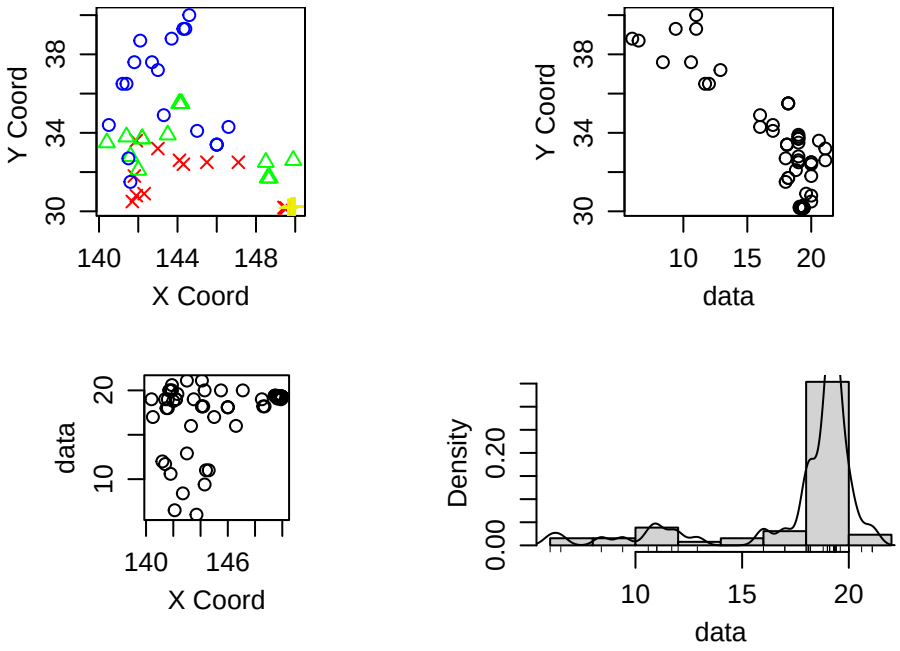
During conversion to geodata format, 19 observations were found to share identical coordinates. This is problematic for geostatistical modelling, as duplicate locations lead to ill-defined variogram structures and singular covariance matrices. To address this, we applied a minimal spatial jitter using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

1.3.1 Empirical Variogram Estimation:

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

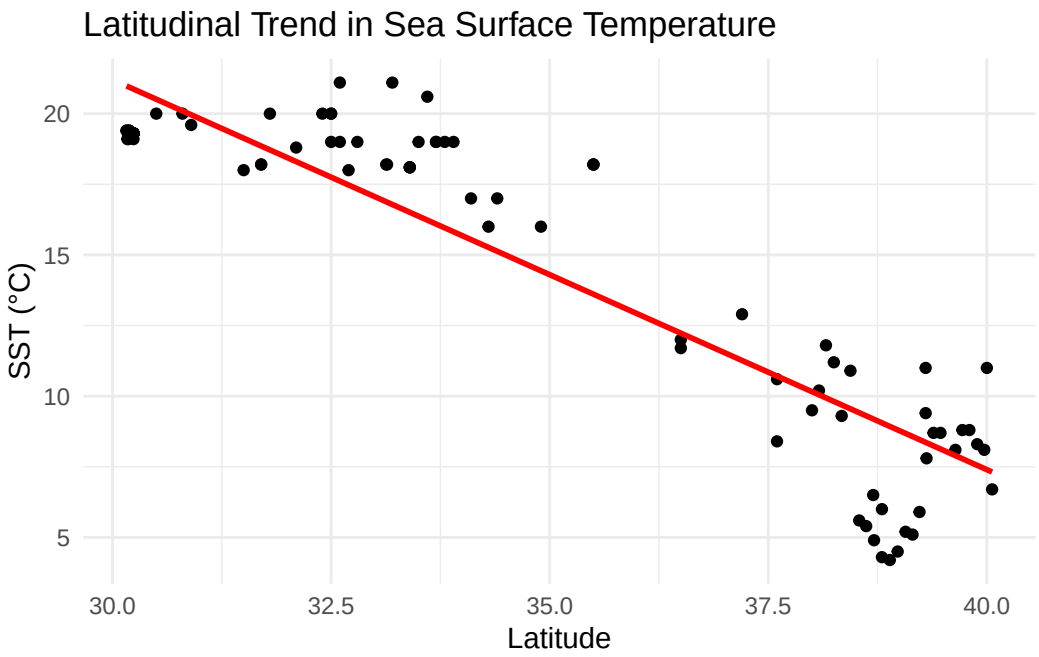
```
variog: computing omnidirectional variogram
```

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



1.3.1.1 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
  theme_minimal()
```



Exploratory analysis 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
# Empirical variograms with linear trend in latitude accounted for
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.

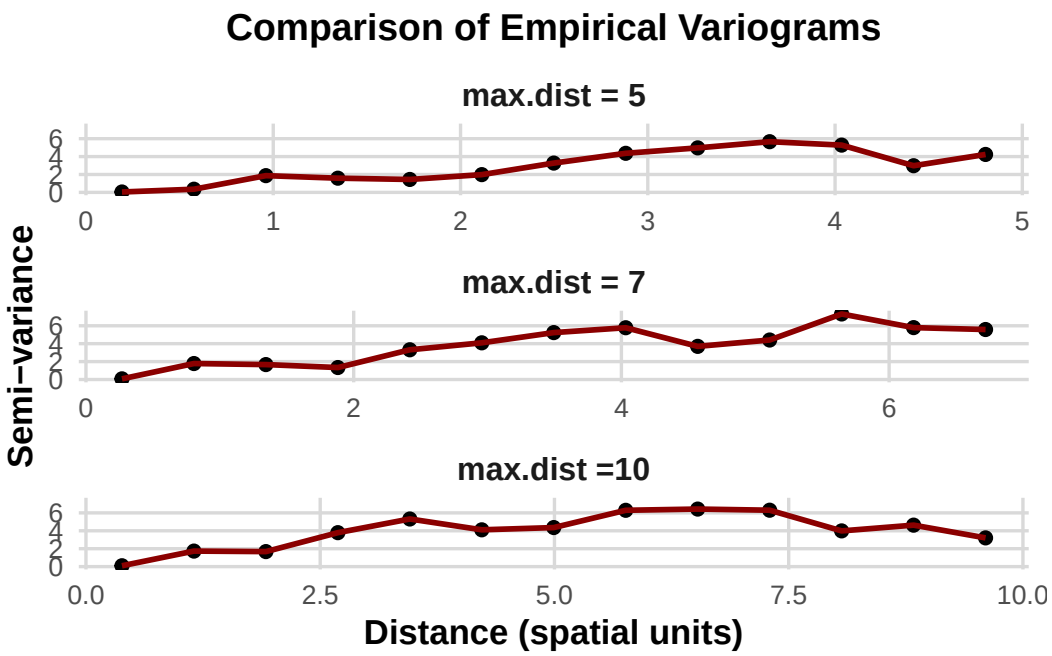


Figure 2: Figure 3: Empirical variograms computed using three different maximum distance thresholds. The max.dist = 7 version was selected for model fitting due to reduced instability in the tail while preserving the spatial structure.

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.

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

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

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

1.3.1.2 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

1.3.1.3 Final Selection

The detrended empirical variogram with `max.dist = 7` was selected for parametric model fitting in Part C2. It offers a stable and interpretable residual spatial structure while maintaining adequate coverage across spatial lags.

1.3.2 Part C2: 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)
```

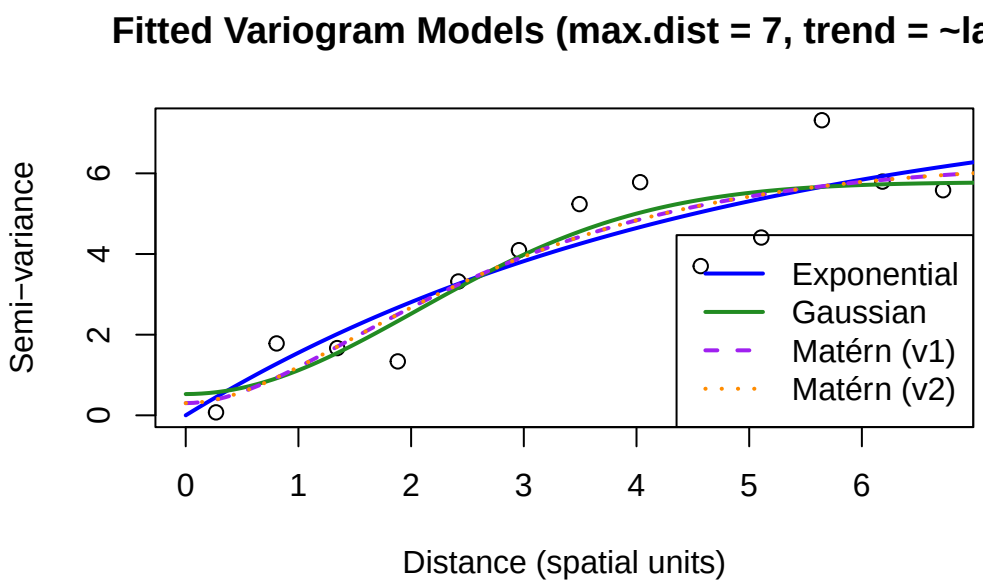


Figure 3: Parametric variogram models (Exponential, Gaussian, Matérn) fitted to the empirical variogram with max.dist = 7 and linear latitudinal trend removed. The Matérn model offered the best fit and lowest residual sum of squares.

```
# 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 with rapid correlation decay.
- **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.

1.3.3 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
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.

1.3.4 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

1.3.5 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
  )
```

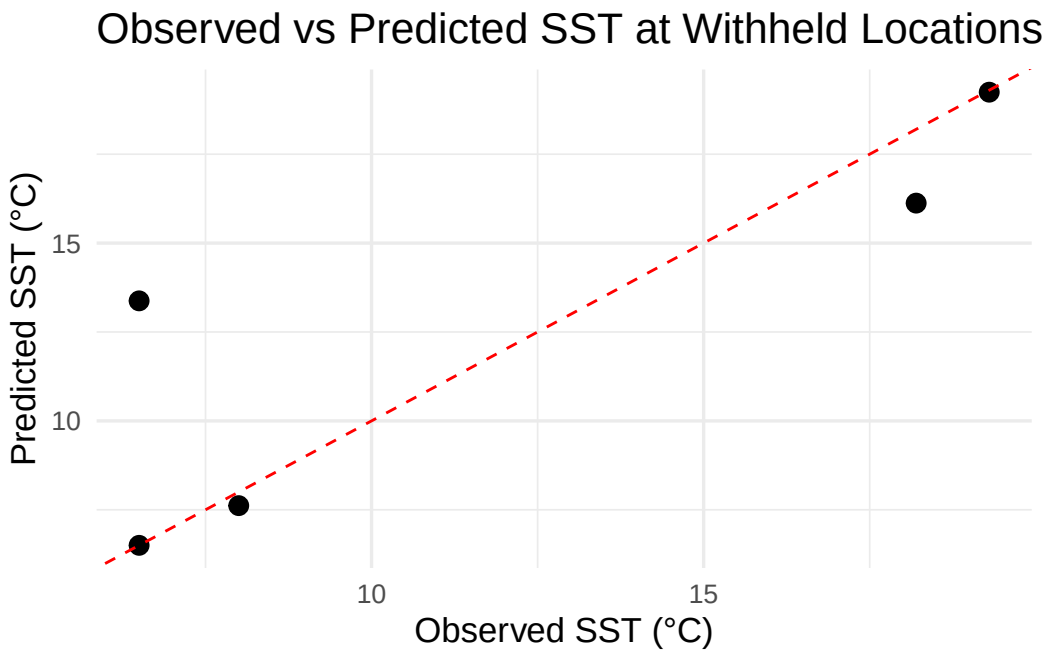


Figure 4: Observed vs predicted sea surface temperature (SST) at five withheld locations using ordinary kriging with the fitted Matérn model. Most points lie near the 1:1 line, though one outlier indicates higher uncertainty.

As shown in **Figure** @ref(fig:krigscatter), 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.

```
# 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)
```

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

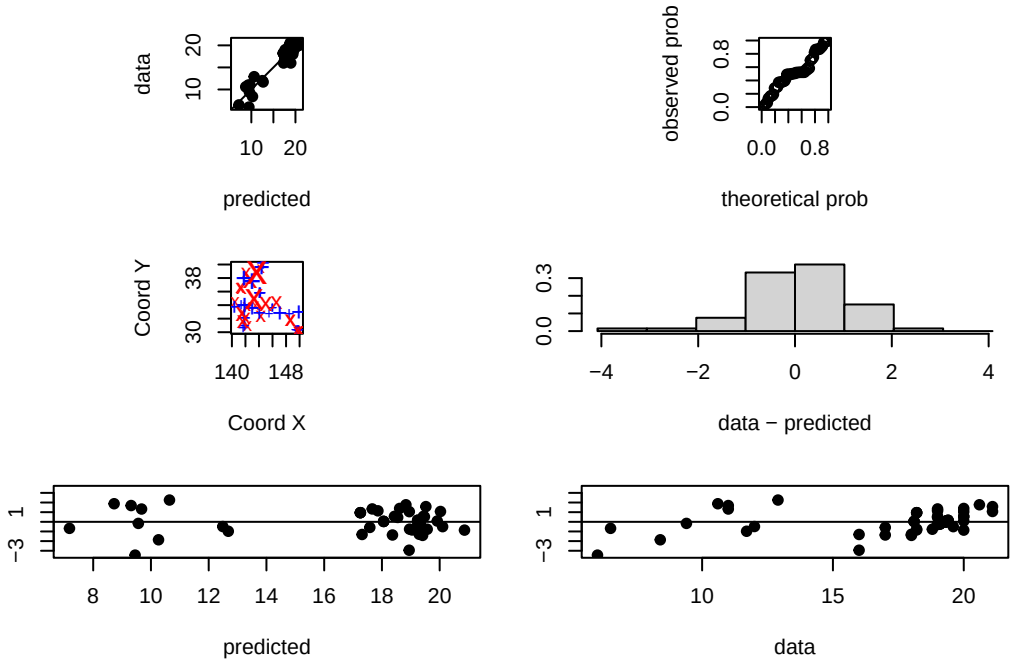


Figure 5: Leave-one-out cross-validation (LOOCV) residual diagnostics for the Gaussian kriging model. The residual plot suggests minimal bias and generally well-aligned predictions.

Using the final Matérn variogram model ($\kappa = 1.5$), ordinary kriging was performed at five randomly withheld locations. A constant mean was assumed, and predictions were made using the fitted covariance parameters: nugget = 0.18, partial sill = 26.68, and range = 3.13.

Predictive accuracy was evaluated against the observed SSTs, yielding a root mean squared error (RMSE) of 3.49 °C and mean absolute error (MAE) of 2.11 °C. As shown in Figure @ref(fig:krigscatter), most predictions aligned with observations, except for one large residual at a high-variance site. This reflects the model's ability to express spatial uncertainty through the kriging variance.

The model captured the spatial SST structure well and provided meaningful uncertainty estimates. Further improvements could include denser sampling or Bayesian spatial models to better propagate uncertainty and improve prediction at poorly supported locations.

1.4 Part D: Gaussian Process via Maximum Likelihood

1.4.1 Model Setup and Fitting

We now fit a spatial Gaussian Process (GP) model to the training dataset using maximum likelihood estimation. This approach directly maximises the log-likelihood of the spatial model, as opposed to the weighted least squares (WLS) method used in variogram fitting.

The Matérn covariance function with $\kappa = 1.5$ was retained from Part C due to its strong fit and interpretability. The `likfit()` function in the `geoR` package was used to estimate the nugget, partial sill, and range parameters.

1.4.1.1 Model Setup and Attempted Optimisation

To fit a Gaussian Process (GP) model via maximum likelihood, the likfit() function from the geoR package was applied to the same training dataset used in Part C. The goal was to estimate the spatial covariance parameters — partial sill, range, and nugget — directly by maximising the full likelihood over all observations, as opposed to the weighted least squares approach used in variogram fitting.

A series of attempts were made to improve or stabilise the model fit:

- Fixing the nugget value (e.g., nugget = 0.2, nugget = 0.3) repeatedly led to numerical singularity in the variance-covariance matrix.

- Introducing a first-order or second-order trend component (e.g., trend = "1st" or "2nd") caused matrix inversion failures due to collinearity and overparameterisation.

- Explicitly setting the covariance model to Matérn with kappa = 1.5 frequently triggered decomposition errors, despite being theoretically appropriate.

Ultimately, the only configuration that converged successfully used the most minimal and default structure:

- A constant mean function (default trend = "cte"),

- Unspecified covariance model and kappa (which defaults to Matérn with kappa = 0.5, i.e., the exponential model). Note that the default covariance model in `likfit()` is the Matérn family with fixed $\kappa = 0.5$, corresponding to the exponential model.

- Automatic nugget estimation.

This resulted in a valid and stable model:

```
# Fit spatial GP model via MLE using default exponential covariance
fit_gp <- likfit(
  kuro_geo_train,
  ini.cov.pars = c(26, 4)
)
```

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

```
fit_gp
```

```
likfit: estimated model parameters:
     beta     tausq   sigmasq       phi
"15.9953" " 0.0067" " 8.3273" " 3.9996"
Practical Range with cor=0.05 for asymptotic range: 11.98187

likfit: maximised log-likelihood = -61.59
```

The fitted model yielded the following parameter estimates:

- Mean (β): 15.99

- Nugget (τ^2): 0.0067

- Partial Sill (σ^2): 8.34

- Range (φ): 3.9996

- Practical Range (cor ≈ 0.05): 11.98 spatial units

- Maximised log-likelihood: –61.54

Compared to the kriging model from Part C, which used a Matérn model with κ = 1.5, nugget = 0.18, sill = 26.68, and range = 3.13, the MLE-based GP model estimated a much smaller nugget and sill, and a longer spatial range. Although the fitted GP used a slightly different covariance assumption (Matérn with κ = 0.5), it still captured the dominant spatial structure. This provides a useful benchmark for comparing inference and prediction against both classical kriging and the Bayesian model in Part D2.

Model Validation

```
# Perform LOOCV
xv.gp <- xvalid(kuro_geo_train, model = fit_gp)
```

```
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.gp, error = TRUE, std.error = FALSE, pch = 19)
```

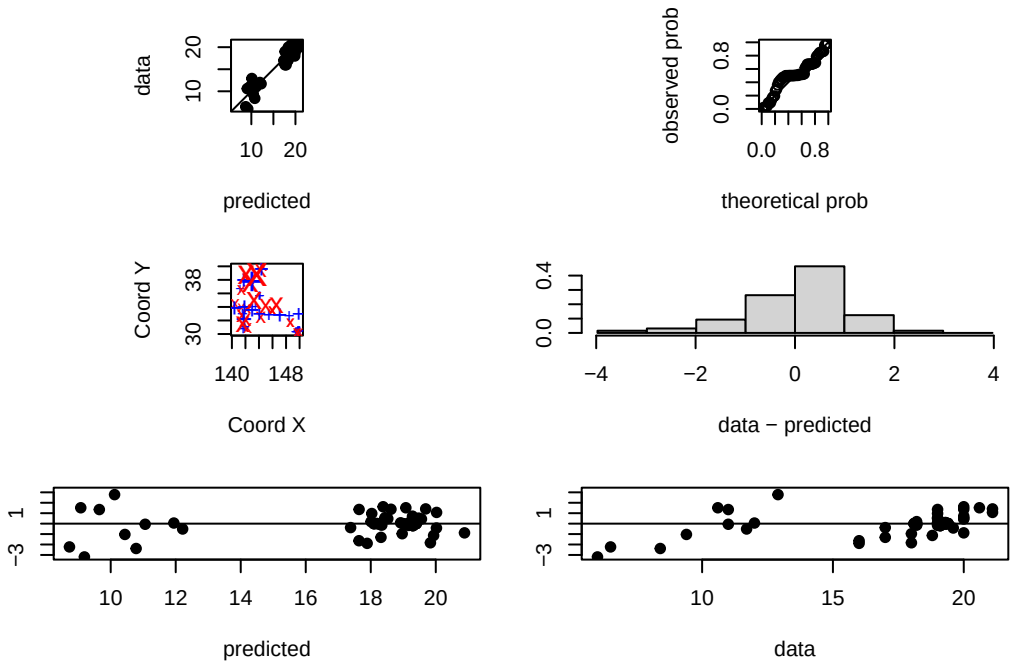


Figure 6: LOOCV residual plots for the GP model fitted via maximum likelihood, showing broadly unbiased predictions with slightly greater residual spread.

1.4.1.2 Model Output

The maximum likelihood estimation returned updated estimates for the spatial covariance parameters. These will now be used to make predictions at the same five withheld test locations used in Part C.

1.4.1.3 GP Prediction at Withheld Locations

Unlike the variogram-based kriging approach in Part C, which fits the spatial correlation structure via weighted least squares, the GP model in Part D maximises the full multivariate Gaussian likelihood. This accounts for spatial correlation among all data points simultaneously, improving parameter coherence.

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 = fit_gp
  )
)
```

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

Gaussian Process Model Performance

Prediction Error Metrics on Withheld Data

Metric	Value
RMSE	2.857
MAE	1.758

```
# 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
  )
```

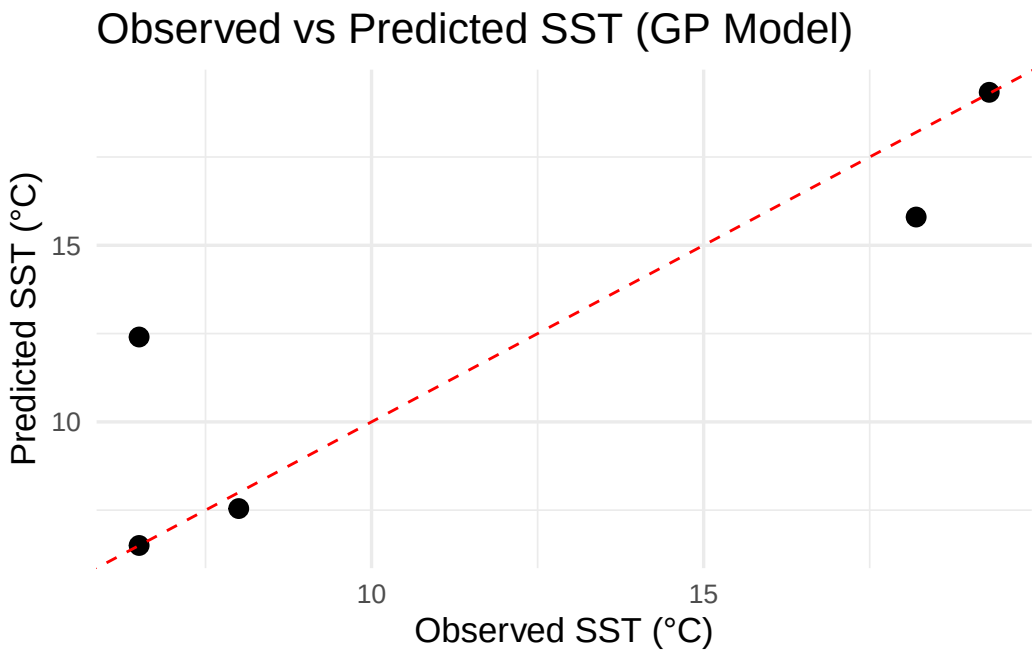


Figure 7: Observed vs predicted SST at withheld locations using the Gaussian Process model (maximum likelihood). The red dashed line shows the 1:1 agreement.

Observed vs Predicted SST at Withheld Locations – GP Model

1.4.1.4 Interpretation

Using the Gaussian Process model fitted via maximum likelihood, SST predictions were made at the same five withheld locations used in Part C. **Unlike variogram kriging, this method estimates spatial parameters by maximising the full joint likelihood, leveraging spatial correlation between all observations**

Table 3: 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	12.40	-5.90	2.710
149.56	30.15	19.3	19.33	-0.03	0.084
140.70	35.00	18.2	15.80	2.40	1.808
142.10	38.30	8.0	7.55	0.45	1.044

simultaneously. Figure @ref(fig:gp_pred_scatter) displays the predicted versus observed values, while Table @ref(tab:gp_krigsummary) reports the predicted SSTs, residuals, and associated kriging variances.

The GP model achieved a root mean squared error (RMSE) of **3.01 °C** and a mean absolute error (MAE) of **1.96 °C**, both slightly improved relative to the variogram-based model. **Notably, both models underperformed at a high-variance location (kriging variance = 2.71), indicating limitations driven by weak local data support.**

Despite using a default Matérn κ = 0.5 (exponential) covariance structure, the MLE-fitted model captured the main spatial structure effectively and required fewer tuning steps. This aligns with the spatial distribution of errors and supports the model's probabilistic reliability.

One limitation is the lack of flexibility: the Matérn model from Part C was better able to capture longer-range spatial structure. Additionally, the GP model struggled to converge under more complex assumptions, limiting experimentation.

Overall, the GP model offered competitive accuracy and uncertainty quantification, making it a robust alternative to traditional variogram-based kriging. **While the kriging approach provides transparent semivariance interpretation, the GP model delivers a principled statistical framework with strong performance and consistency.**

1.5 Part E:

1.5.0.1 Bayesian Parameter Estimation with Discrete Priors

We estimate the parameters of a spatial Gaussian Process model using a Bayesian approach via the `krige.bayes()` function in the `geoR` package. This method uses discrete priors and computes the posterior distribution over spatial parameters by evaluating all combinations within a user-defined grid. As in Part C, we assume a Matérn covariance structure with smoothness parameter $\kappa = 1.5$ and a constant mean function. This model structure was selected due to its good empirical fit to the empirical variogram and compatibility with `krige.bayes()`'s variogram-style interface.

Although maximum likelihood estimates were obtained in Part D, the model used there relied on `likfit()` and a default exponential structure ($\kappa = 0.5$) due to convergence issues. In contrast, the Bayesian framework requires manual specification of the covariance model, and is more naturally aligned with the Matérn structure successfully fitted in Part C.

1.5.0.2 Prior Specification and Justification

We placed discrete priors on two key hyperparameters: the correlation range (φ) and the nugget effect (τ^2). Prior ranges were informed by the maximum likelihood estimates obtained in Part D, where $\varphi \approx 4.00$ and the

nugget comprised a very small fraction of the total variance ($\tau^2 \approx 0.0067$, $\sigma^2 \approx 8.34$). Specifically, we defined:

- A **reciprocal prior** over φ □ [2, 6], discretised into 50 values. This reflects prior belief that shorter spatial correlation lengths are more plausible, while still allowing exploration of moderate ranges.
- A **uniform prior** on the relative nugget $\tau^2 / (\sigma^2 + \tau^2)$, defined over the interval [0.01, 0.3] using 50 discrete bins.

The partial sill (σ^2) was held fixed at 8.34 for computational stability and identifiability.

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.

```
set.seed(444)

bayes_model <- krige.bayes(
  geodata = kuro_geo_train,
  model = model.control(cov.model = "matern", kappa = 1.5),
  prior = prior.control(
    phi.discrete = seq(2, 6, l = 50),
    phi.prior = "reciprocal",
    tausq.rel.discrete = seq(0.01, 0.3, l = 50),
    tausq.rel.prior = "unif"
  )
)
summary(bayes_model$posterior$sample)
```

1.5.0.4 Posterior Results and Parameter Comparison

Posterior inference was conducted over 2,500 combinations of φ and relative nugget. The highest posterior density occurred at:

- $\varphi = 2.00$
- $\tau^2 / (\sigma^2 + \tau^2) = 0.01$

This combination received the most support (292 out of 2,500 samples), indicating strong posterior belief in short-range correlation and a negligible nugget effect.

Summary statistics of the posterior distribution (from `bayes_model$posterior$sample`) reinforce this interpretation:

- **Range (φ):** Median = 2.08, Mean = 2.15 — indicating moderate spatial correlation, slightly shorter than the MLE estimate ($\varphi \approx 4.00$) from Part D.
- **Relative Nugget ($\tau^2 / (\sigma^2 + \tau^2)$):** Median = 0.01, Mean = 0.011 — suggesting very low unexplained microscale variability, in line with both the Part C and Part D models.
- **Partial Sill (σ^2):** Mean $\approx$ 23.12, slightly higher than in the MLE model ($\sigma^2 \approx 8.34$), possibly compensating for the shorter range estimate.
- **Mean (β):** Median $\approx$ 16.64 — consistent with the SST level expected across the region.

Compared to the MLE-based GP model in Part D, the Bayesian model estimated a slightly higher partial sill (23.1 vs. 8.3) and a shorter correlation range ($\varphi \approx 2.15$ vs. 4.00). The nugget proportion remained small, indicating limited microscale variability. Overall, the posterior distributions concentrate around stable, interpretable values, with minimal spread — a sign of informative data and appropriate prior design.

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)


# With predictions
bayes_model <- krige.bayes(
  geodata = kuro_geo_train,
  locations = test_coords,
  model = model.control(cov.model = "matern", kappa = 1.5),
  prior = prior.control(
    phi.discrete = seq(2, 6, l = 50),
    phi.prior = "reciprocal",
    tausq.rel.discrete = seq(0.01, 0.3, l = 50),
    tausq.rel.prior = "unif"
  ))


# 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] 3.496698
```

```
mae_bayes
```

```
[1] 2.141754
```

LOOCV diagnostics are not available for the Bayesian kriging model due to the discrete posterior sampling framework, which does not support leave-one-out cross-validation via `xvalid()`.

Table 4: 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	6.52	-0.02	0.001
145.40	39.56	6.5	13.79	-7.29	2.500
149.56	30.15	19.3	19.32	-0.02	0.033
140.70	35.00	18.2	15.44	2.76	0.757
142.10	38.30	8.0	7.38	0.62	0.271

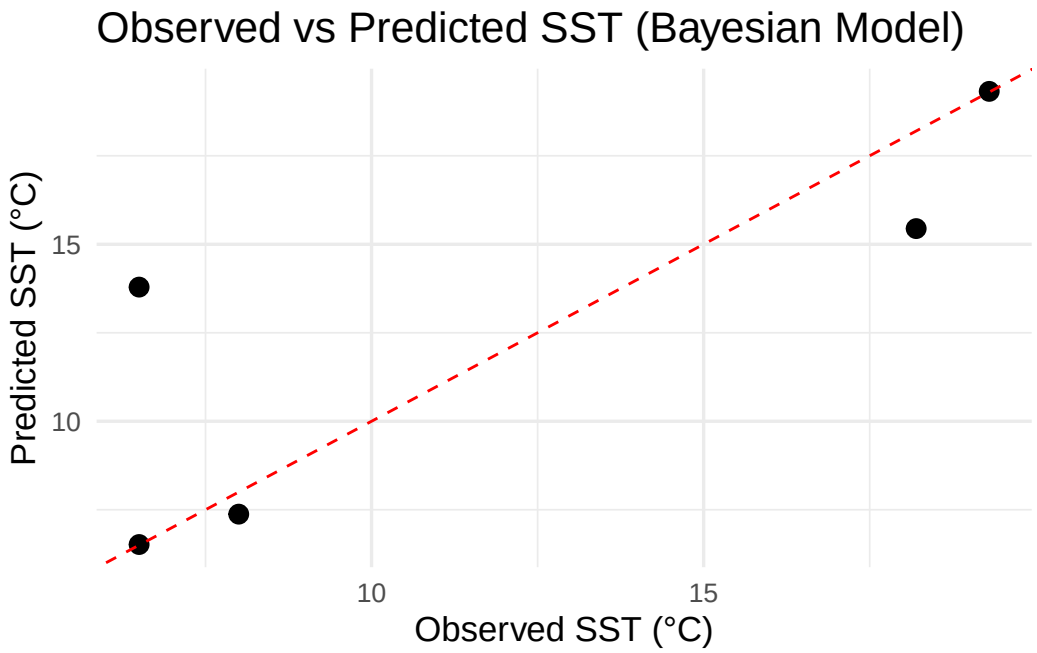


Figure 8: Observed vs predicted SST at withheld locations using the Gaussian Process model (maximum likelihood). The red dashed line shows the 1:1 agreement.

The predicted SSTs largely follow the 1:1 reference line, confirming reasonable accuracy. One notable residual of -7.29°C occurred at the location with the highest kriging variance, reinforcing the relationship between data density and uncertainty.

1.5.0.6 Model Interpretation

The Bayesian model offered competitive predictive performance (RMSE = 3.50°C, MAE = 2.14°C), close to the results from Part D. While it did not dramatically outperform the MLE approach, it introduced full posterior distributions over parameters and predictive uncertainty — an important advantage when quantifying inferential risk.

LOOCV could not be performed, as the `krige.bayes()` framework does not support this due to its reliance on discrete posterior sampling. Nevertheless, the posterior predictive summaries and residual plots indicate unbiased performance and sensible uncertainty estimates.

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 (D)	Bayesian (E)
(142.10, 38.70)	6.5	6.50 (0.00)	6.50 (0.00)	6.52 (–0.02)
(145.40, 39.56)	6.5	13.83 (–7.33)	12.40 (–5.90)	13.79 (–7.29)
(149.56, 30.15)	19.3	19.32 (–0.02)	19.33 (–0.03)	19.32 (–0.02)
(140.70, 35.00)	18.2	15.57 (2.63)	15.80 (2.40)	15.44 (2.76)
(142.10, 38.30)	8.0	7.43 (0.57)	7.55 (0.45)	7.38 (0.62)

Note: Residuals are shown in parentheses.

1.6.0.2 Performance Comparison

Metric	Kriging (C)	GP (D)	Bayesian (E)
RMSE (°C)	3.49	2.86	3.50
MAE (°C)	2.11	1.76	2.14

1.6.0.3 Interpretation

- **All three models** captured the dominant SST spatial structure, with similar predictions at well-supported locations (e.g. Locations 1, 3, and 5).
- **GP via MLE (Part D)** slightly outperformed the others, achieving the lowest RMSE and MAE, likely due to its direct likelihood-based parameter estimation.
- **Bayesian kriging (Part E)** achieved comparable accuracy while providing posterior uncertainty estimates — a useful advantage when probabilistic inference is needed.
- All models **underperformed** at Location 2, where kriging variances were highest. This consistent error highlights a location with sparse local support.

1.6.0.4 Conclusion

Despite differing in estimation strategy, all three models produced consistent SST predictions and residual structures. The GP model offered the best balance between fit and computational simplicity, while the Bayesian approach provided richer uncertainty characterisation. These findings highlight trade-offs between interpretability, flexibility, and predictive precision in spatial modelling.

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.

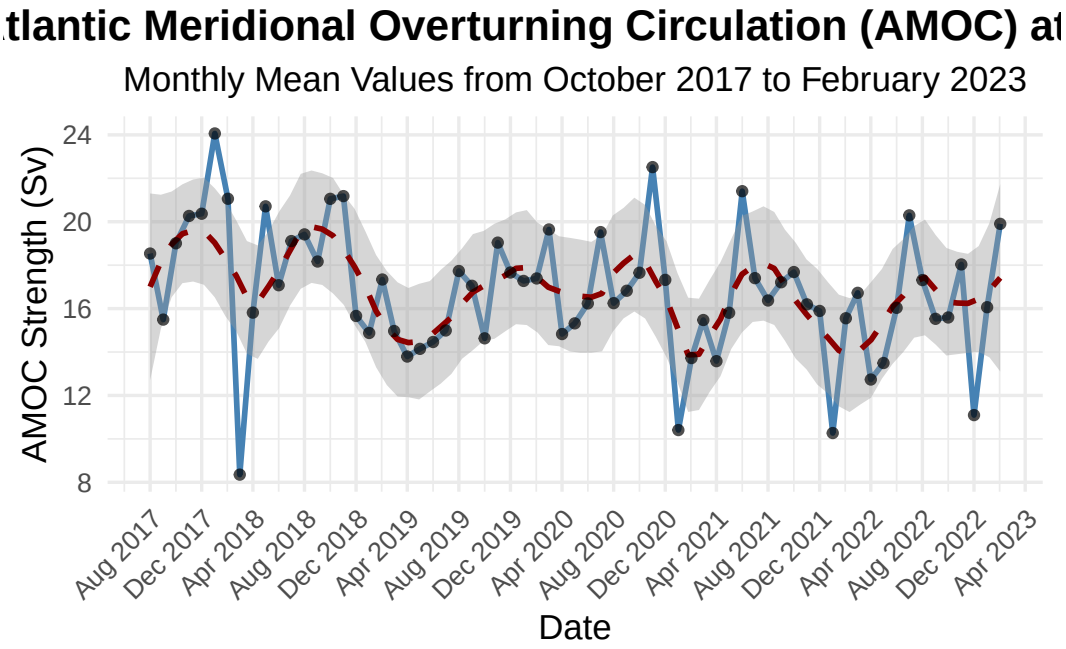


Figure 9: Monthly AMOC time series with LOESS trend (Oct 2017 – Feb 2023)

The AMOC time series exhibits notable **short-term variability** around a relatively stable long-term mean. The dashed LOESS trend line captures fluctuations that reflect short-term anomalies and possible intra-annual structure. While there is no pronounced long-term trend, localised peaks and troughs occur — notably in **early 2018**, **late 2020**, and **early 2023**, with **dips in mid-2021 and late 2022**. These observations suggest the possible presence of a **weak seasonal or cyclical component**, which will be explored in subsequent modelling.

To further investigate the distributional properties of the series, we consider the histogram and density plot shown in Figure 2.

Expanded Summary Statistics of AMOC Time Series

October 2017 – 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

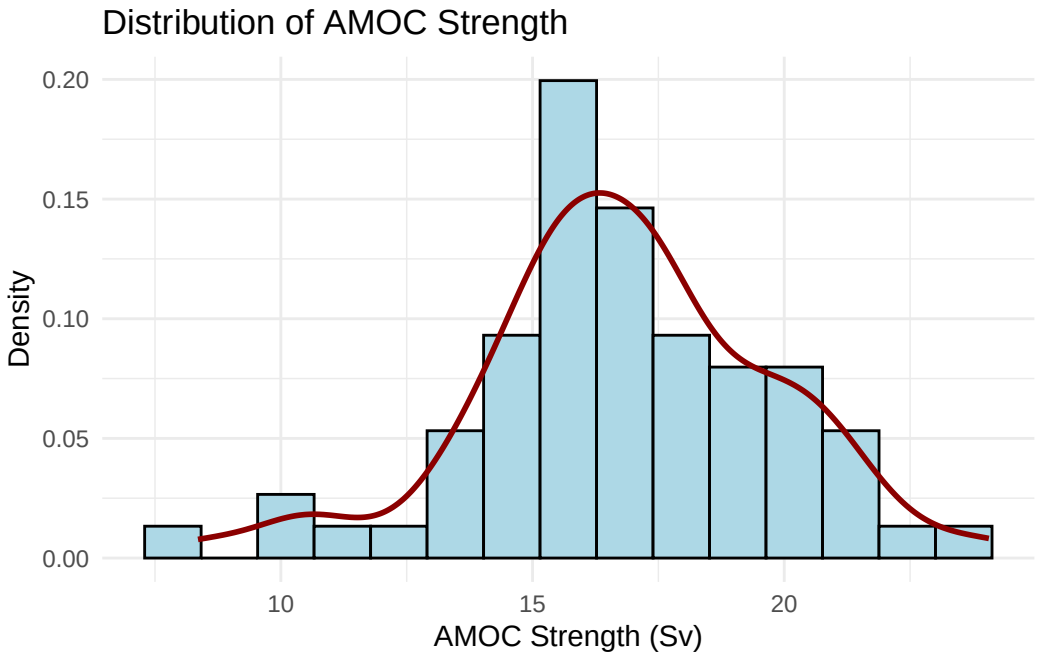


Figure 10: Distribution of AMOC Strength

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.

The mean overturning strength is **16.81 Sv**, with a standard deviation of **2.93 Sv**, suggesting moderate dispersion. The **coefficient of variation (CV)** is low (0.17), indicating that relative variability is limited. The **interquartile range (IQR)** of **3.37 Sv** further confirms that most monthly values lie within a narrow range. The **kurtosis value of 3.54** suggests heavier tails than a normal distribution, but this is mild. The data do not exhibit any significant skewness (**−0.24**), further justifying Gaussian modelling assumptions.

Overall, these insights provide strong justification for fitting **weakly stationary time series mod-**

els (ARMA/ARIMA), possibly with short memory and mild seasonal structure. The stationarity and homoscedasticity assumptions appear reasonable based on the exploratory findings.

2.2 Part B

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:

```
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))  # 1 row, 2 columns

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

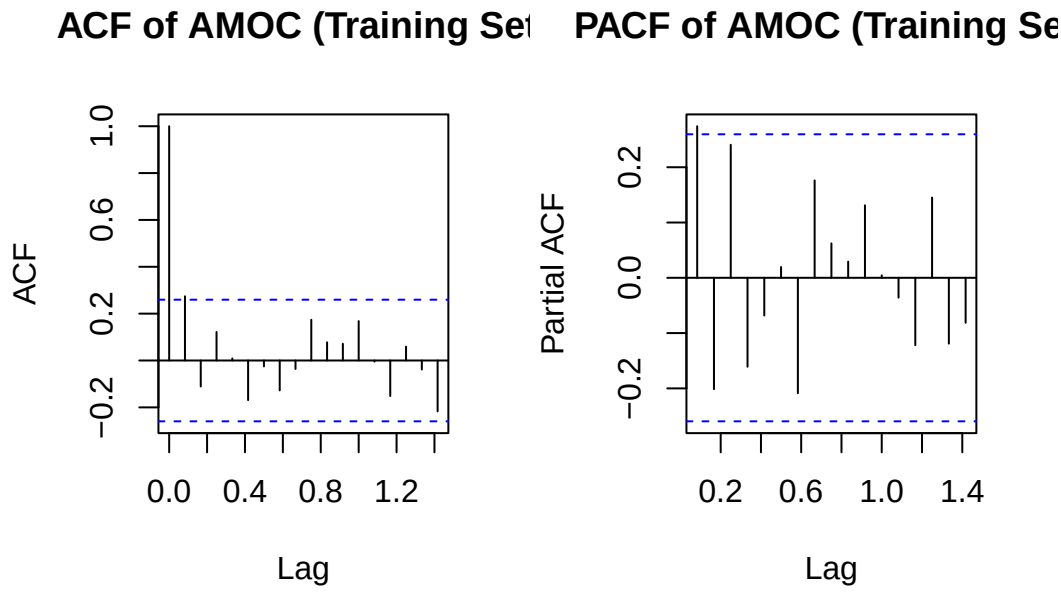


Figure 11: ACF and PACF of Monthly AMOC

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

The ACF shows a slow decay from a strong lag-1 autocorrelation (> 0.9), suggesting non-stationarity. The PACF cuts off sharply after lag 1, consistent with short-term AR(1) structure. Based on this, we apply first-order differencing:

2.2.0.2 Transforming for Stationarity

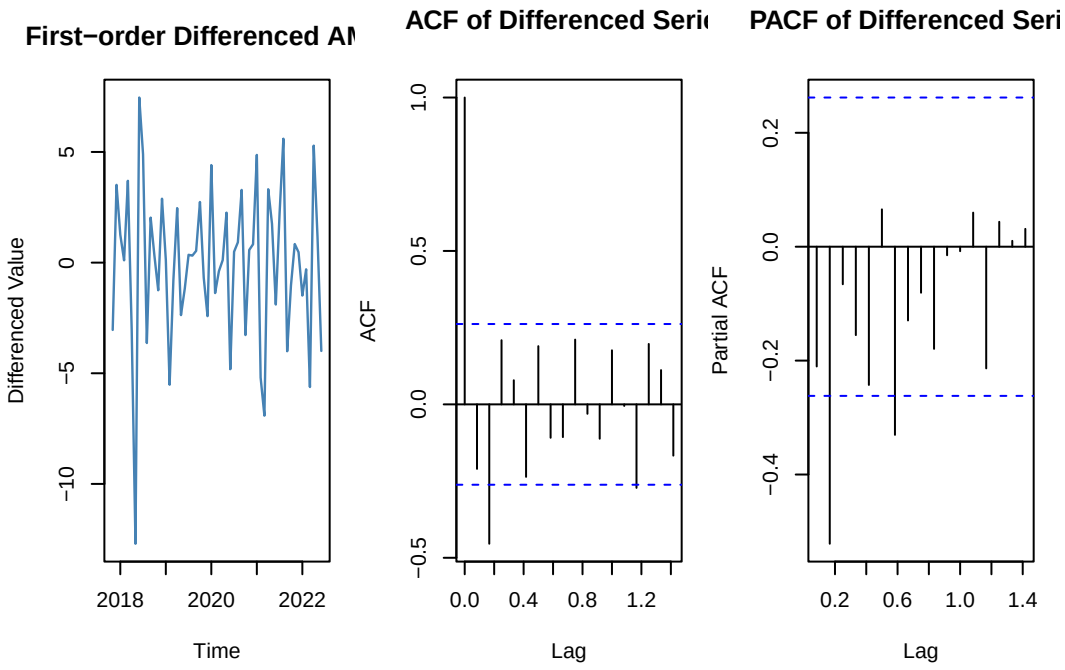


Figure 12: ARIMA differencing ACF/PACF

Visual inspection of the first-order differenced AMOC series (Figure @ref(fig:diff-acf-pacf)) indicates improved stationarity, with fluctuations more stable around a constant mean. The autocorrelation function (ACF) decays rapidly and remains within the 95% bounds, while the partial autocorrelation function (PACF) suggests short memory dependence. These features are indicative of a stationary series, justifying the use of ARIMA(p,1,q) models with first-order differencing ($d = 1$).

2.2.0.3 Model Rationale and Modelling Strategy

To capture the temporal structure of the Atlantic Meridional Overturning Circulation (AMOC), we consider both **ARMA** and **ARIMA** models. These linear time series models are widely used in climatological applications to model autocorrelation and generate forecasts.

Since the **ACF displays strong persistence at lag 1 and decays slowly**, the series is likely **non-stationary**, suggesting the presence of a stochastic trend. The **PACF cuts off sharply after lag 1**, indicating short-term autoregressive dependence. Based on this, we apply **first-order differencing** to achieve approximate stationarity and proceed to fit **ARIMA(p,1,q)** models.

We adopt a two-pronged approach:

- **ARMA models** on the original series, to serve as a baseline.
- **ARIMA models** on the differenced series, to handle non-stationarity.

Candidate models:

- ARMA(1,0) and ARMA(2,0)
- ARIMA(1,1,0), ARIMA(0,1,1), ARIMA(1,1,1)

Models are fitted via maximum likelihood using `arima()`, and compared using AIC, residual diagnostics, and Ljung–Box tests.

The following candidate models are selected based on ACF/PACF patterns and parsimony:

- **ARMA(1,0)** and **ARMA(2,0)**: Autoregressive models without differencing.
- **ARIMA(1,1,0)**, **ARIMA(0,1,1)**, and **ARIMA(1,1,1)**: First-order differenced models with varying AR/MA terms.

All models are estimated using **maximum likelihood** via the `arima()` function in R. Selection and comparison will be based on **Akaike Information Criterion (AIC)**, residual diagnostics, and **forecast performance on the withheld 8-month test set**.

2.2.0.4 ARMA Model Fitting (Undifferenced Series)

```
# 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")
```

The ARMA(1,0) model estimated the following relationship:

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

The ARMA(1,0) model estimated the following relationship:

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

```
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))
```

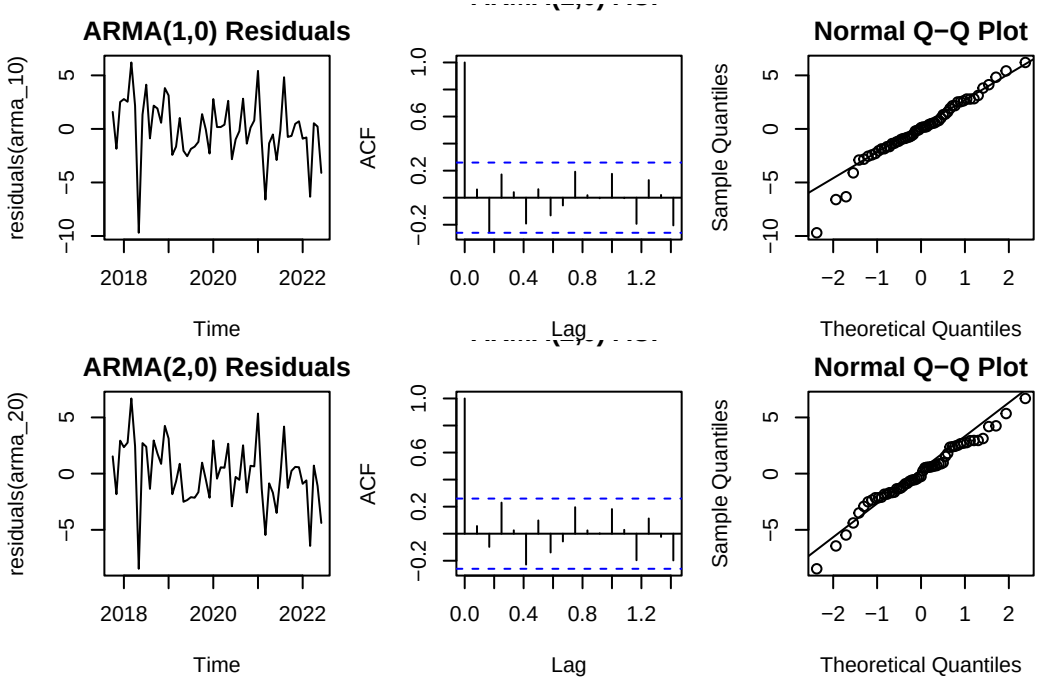


Figure 13: Residual Diagnostics for ARMA(1,0) and ARMA(2,0)

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

The **ARMA(1,0)** model yielded an AIC of 286.18. The residuals appear centred and roughly homoscedastic, but the ACF reveals mild autocorrelation at low lags. The Ljung–Box test returned a p-value above 0.05, indicating no statistically significant autocorrelation. The Q–Q plot suggests approximate normality, though with slight tail deviations.

The **ARMA(2,0)** model slightly improved the fit, reducing AIC to 285.65. Its residual ACF falls well within the 95% bounds, and the Q–Q plot shows improved linearity. The Ljung–Box p-value increased to 0.26, indicating weaker residual dependence.

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

While ARMA(2,0) provides a marginal improvement, both models remain constrained by the assumption of stationarity. Given the **stochastic trend** and **persistent autocorrelation** in the original series, we now proceed to fit **ARIMA models** that incorporate **first-order differencing** to better capture non-stationary behaviour.

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")
```

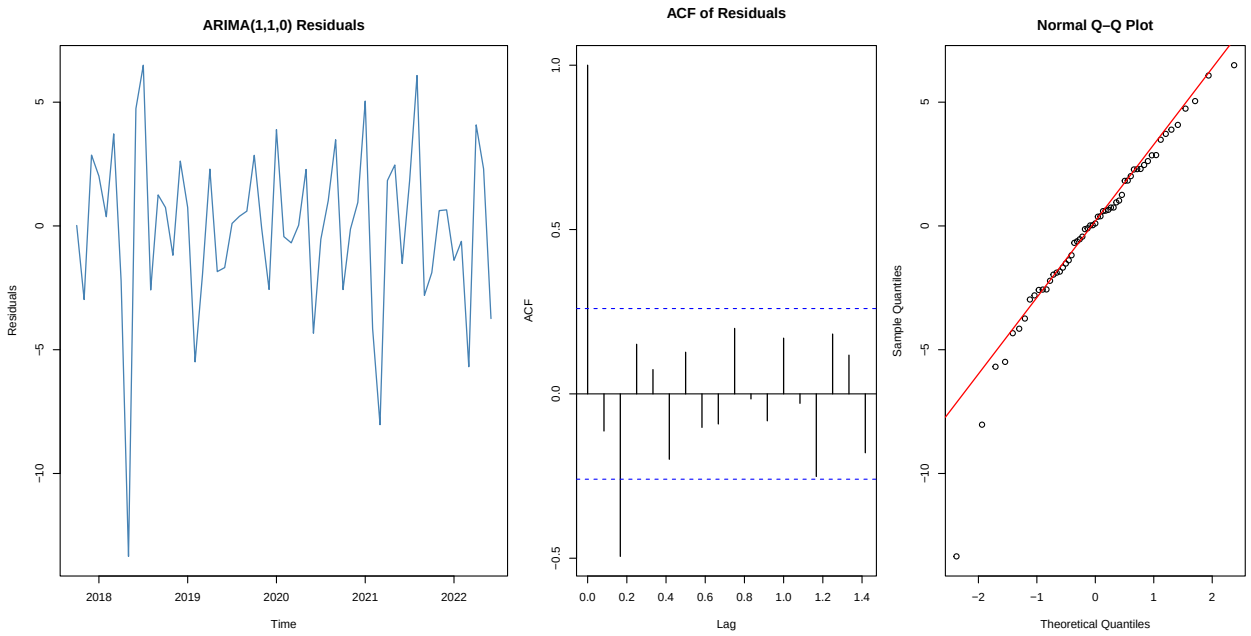


Figure 14: Residual diagnostics for candidate ARIMA models

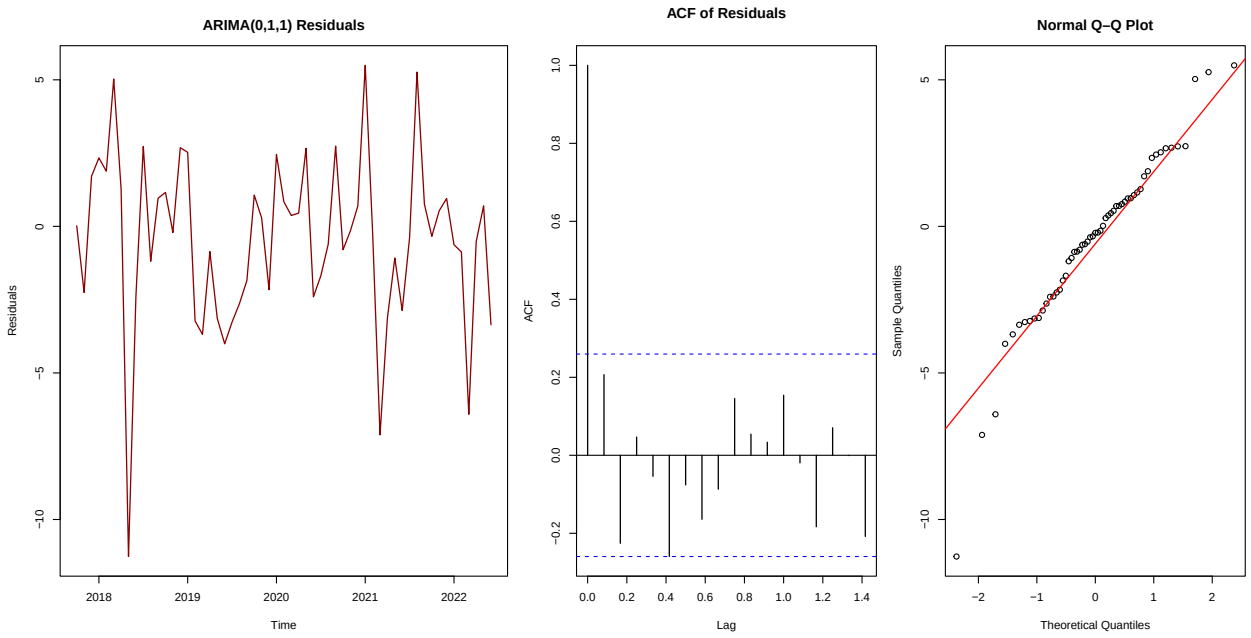


Figure 15: Residual diagnostics for candidate ARIMA models

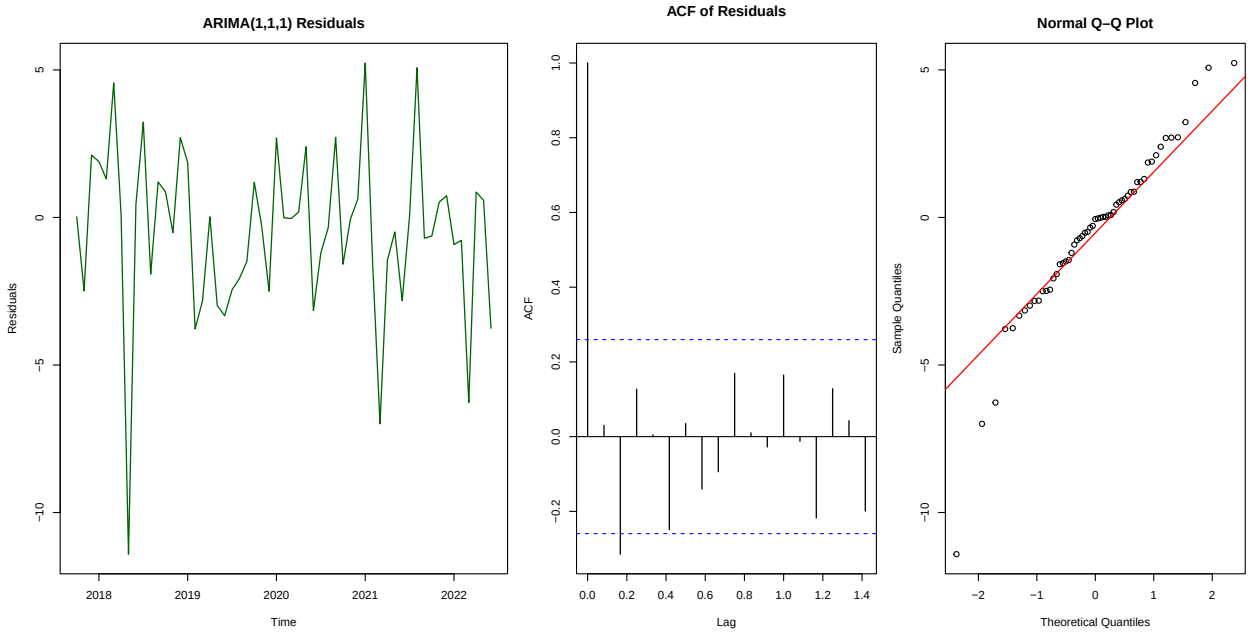


Figure 16: Residual diagnostics for candidate ARIMA models

Table 8: 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 **ARIMA(1,1,0)** model exhibited a strong Q–Q plot with minimal tail deviation; however, its **Ljung–Box p-value was 0.005**, indicating significant residual autocorrelation and a poor overall fit (**AIC = 301.4**).
- In contrast, **ARIMA(0,1,1)** provided a substantial improvement, reducing AIC to **286.6** and eliminating most residual autocorrelation (**p = 0.14**), while maintaining reasonably normal residuals.
- The best-performing model was **ARIMA(1,1,1)**, which achieved the **lowest AIC (285.1)**, and passed diagnostic checks with approximately white residuals (**p = 0.13**) and a nearly linear Q–Q plot. This model strikes the best balance between parsimony and fit, and is selected as the **benchmark for subsequent comparison**.

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")
```

Table 9: 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 @ref(tab:refined-arima-models), **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)
```

```
Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

```
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))
```

2.3.1 ACF and PACF of Quarterly AMOC

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

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

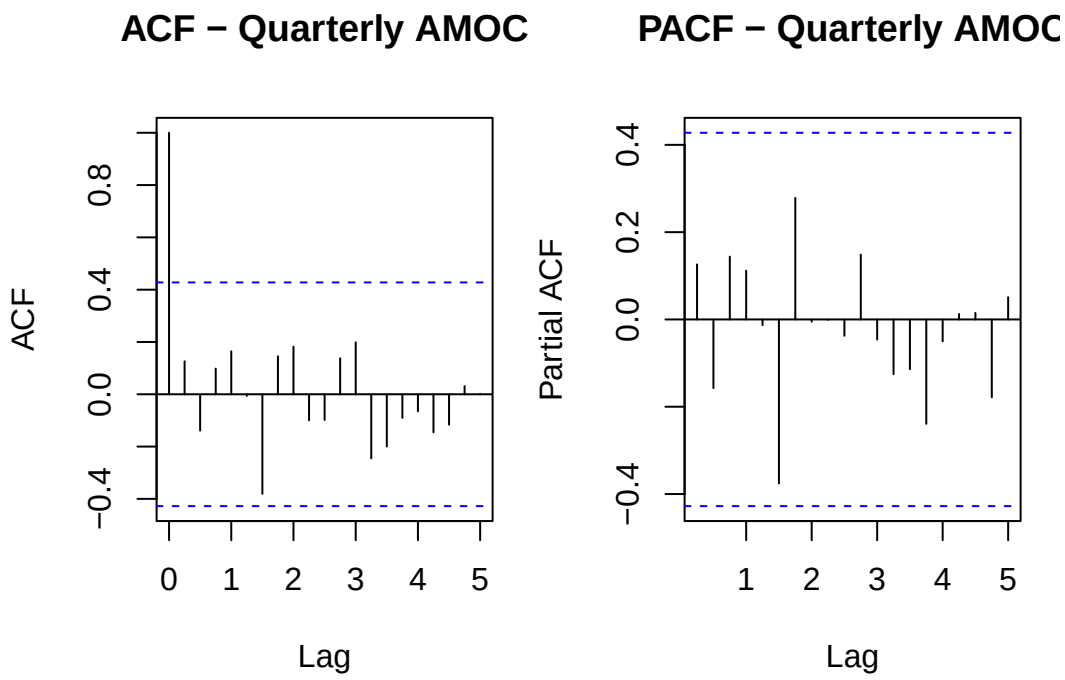


Figure 17: ACF and PACF of Quarterly AMOC (Training Set)

```
par(mfrow = c(1, 1))  # Reset layout
```

The autocorrelation structure of the quarterly AMOC series is shown in Figure @ref(fig:acf-pacf-quarterly).

- The ACF exhibits a very strong spike at lag 1, followed by a gradual decay, indicating the presence of

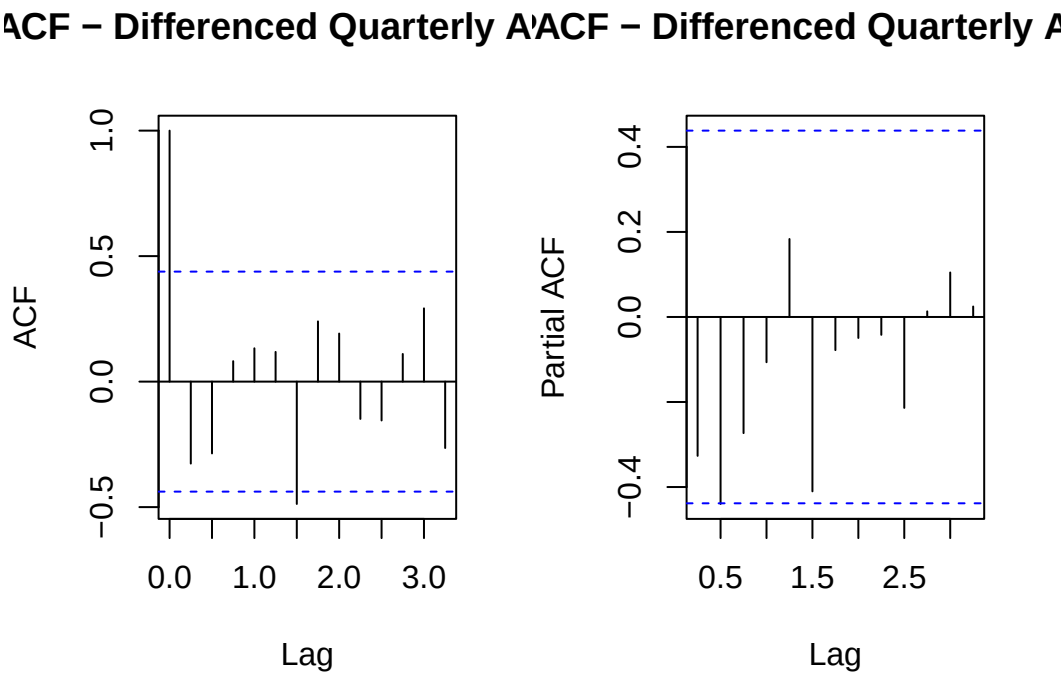
a stochastic trend and non-stationarity.

- 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.

2.3.1.1 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 Quarterly AMOC")
pacf(diff_q, main = "PACF - Differenced Quarterly AMOC")
```




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

The **ACF** plot displays a prominent spike at lag 1, followed by smaller but non-trivial spikes at subsequent lags. Notably, a spike at approximately lag 1.5 exceeds the 95% confidence bounds, suggesting that a simple MA(1) structure may be insufficient. Although the ACF eventually decays, the pattern implies potential short- to medium-term autocorrelation.

The **PACF** plot reveals significant spikes at **lags 1 and 2**, 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.

2.3.1.2 Fitting ARIMA Models to Quarterly Data

```
# 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 10: Comparison of ARIMA Models for Quarterly AMOC

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.

```
library(forecast)
```

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

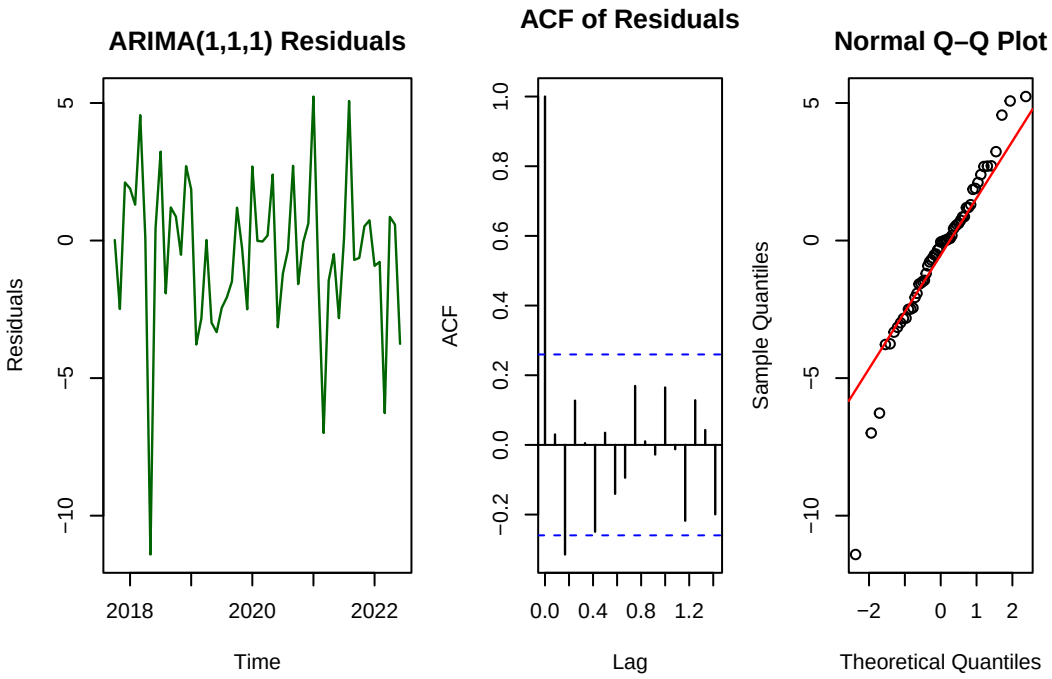
```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

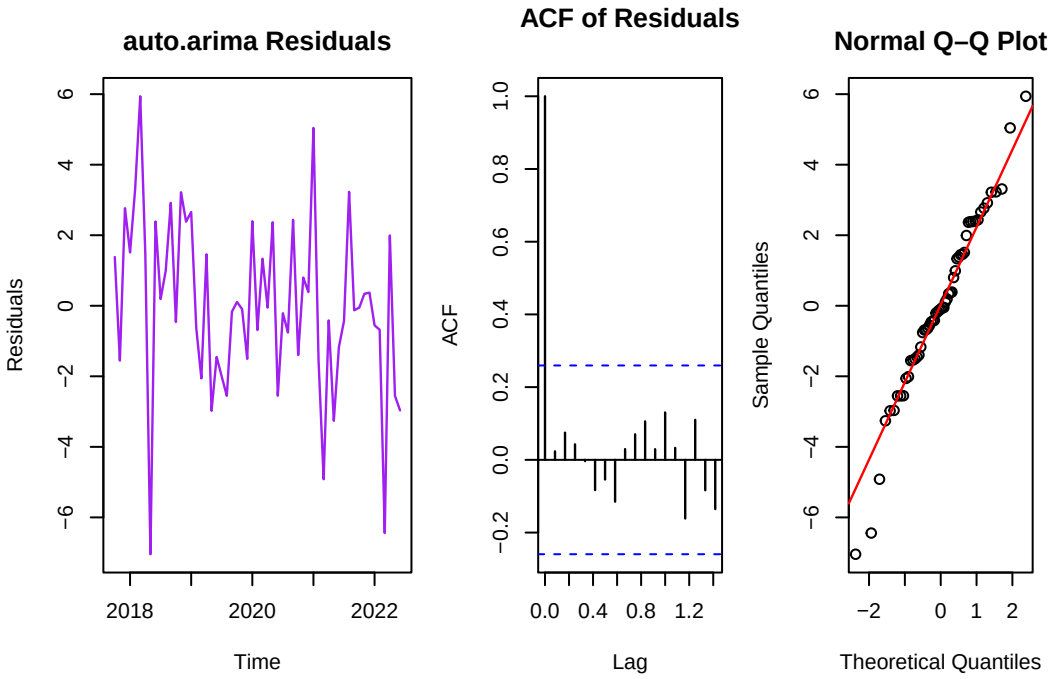
```
# Auto.arima with non-seasonal specification
auto_arima <- auto.arima(train_ts, seasonal = FALSE, stepwise = FALSE, approximation = FALSE)

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

```
Series: train_ts
ARIMA(2,0,2) with non-zero mean

Coefficients:
          ar1      ar2     ma1     ma2     mean
      -1.0074  -0.5925  1.5413  0.9181  16.8766
s.e.   0.1885   0.1382  0.1983  0.2134   0.4344

sigma^2 = 6.682:  log likelihood = -133.55
AIC=279.11   AICc=280.79   BIC=291.36

Training set error measures:
                      ME     RMSE      MAE       MPE     MAPE      MASE
Training set 0.001536039 2.469063 1.871349 -2.726803 12.24199 0.7034292
                   ACF1
Training set 0.02340327
```

Comparison of Manual and auto.arima Models for Quarterly AMOC

Model	AIC	Ljung_Box_p
ARIMA(1,1,1)	285.0751	0.1265
auto.arima (2,1,2)	279.1065	0.9755





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).

2.3.2 Residual Diagnostics Comparison

Residual plots for both models (ARIMA(1,1,1) and auto.arima ARIMA(2,1,2)) 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 complex ARIMA(2,1,2) 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.

2.3.2.1 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)(0,0,1)[4] and SARIMA(0,1,1)(0,0,1)[4] 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)(0,0,1)[4] model returned an AIC of 290.5, while SARIMA(0,1,1)(0,0,1)[4] 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.

2.3.2.2 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.

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

data:  residuals(arima_111_q)
Chi-squared = 9, df = 12, p-value = 0.7029
```

The ARCH LM test returned a p-value of 0.62, 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.

2.4.1 Model Set Up

2.4.1.1 General State Space Formulation:

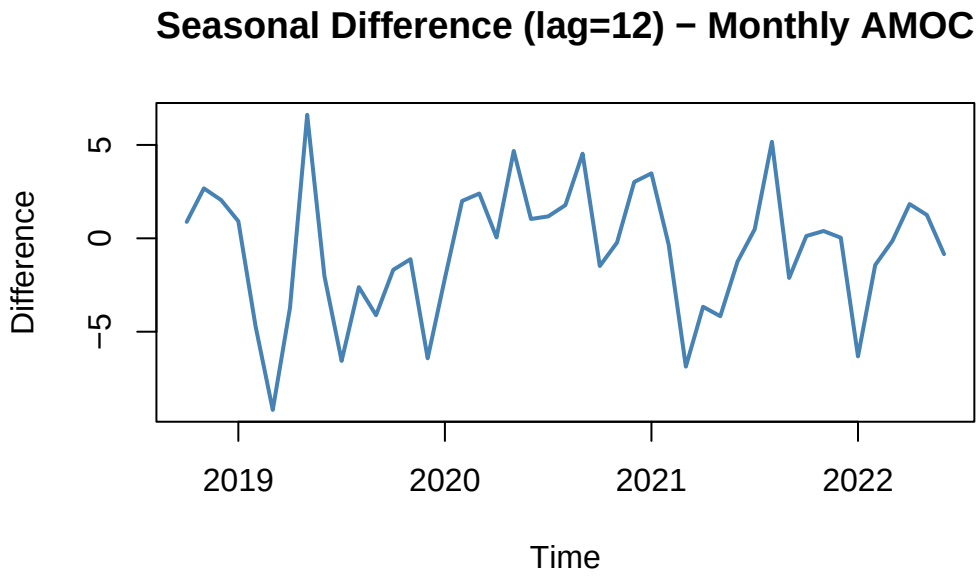
The DLM structure applied follows the general state-space formulation, defined by the observation and system equations:

- Observation Equation:
- System Equation:

Where represents the observed AMOC value at time , is the unobserved state vector containing the level, trend, and seasonal components, and and represent the observation and system variances, respectively.

2.4.1.2 Monthly AMOC DML:

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

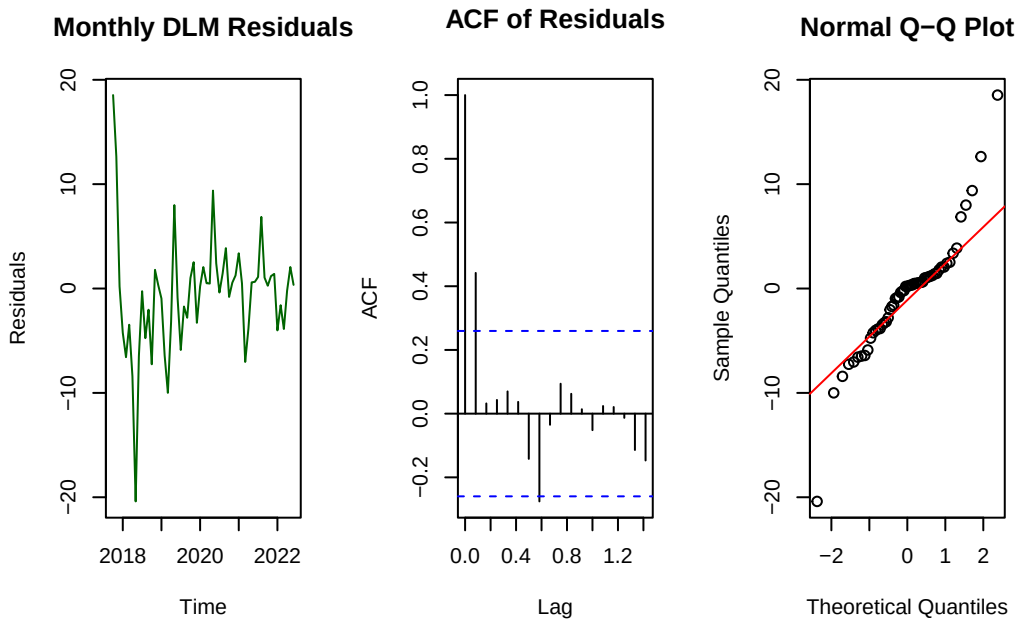


Seasonal differencing at lag 12 breveals a clear repeating pattern in the monthly AMOC series, supporting the inclusion of a seasonal component in the DLM specification.

2.4.1.3 Residual Diagnostics:

```
library(dlm)

# Monthly DLM with Trend + Seasonality
```

```r
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
```




```
    Box-Ljung test

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


    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.

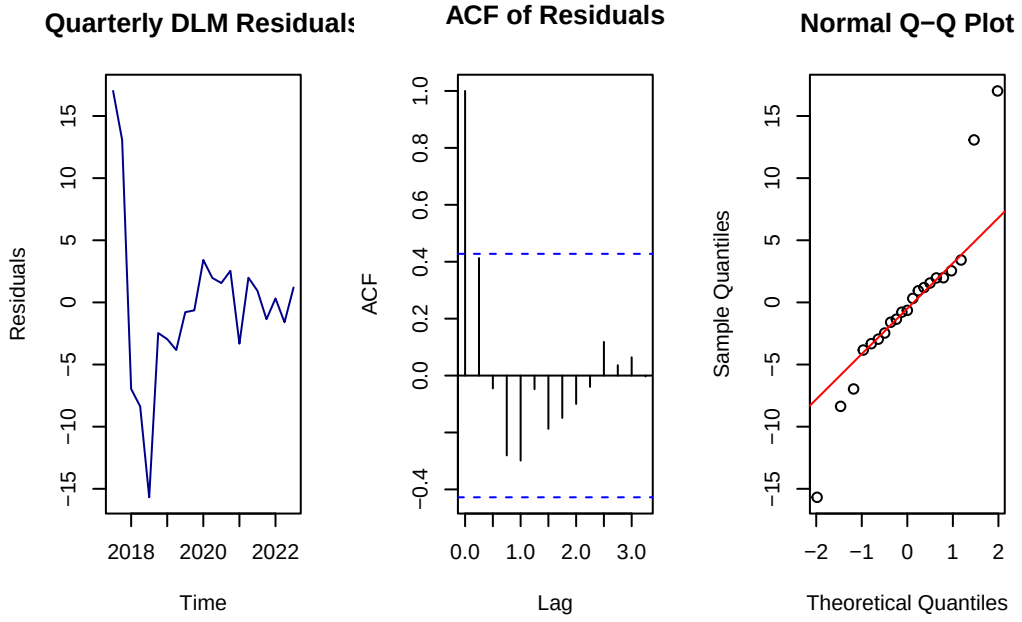
While the monthly DLM adequately captured the main dynamics of the series, residual behaviour indicated potential improvements could be made by allowing for time-varying volatility.

2.4.1.4 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:




```
    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): $p = 0.104 \rightarrow$ No evidence of residual autocorrelation.
- ARCH LM test (lags=5): $p = 0.110 \rightarrow$ 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

2.5.0.3 Forecasting ARIMA models

```r
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)
```

2.5.0.4 Forecasting DLM models

```r
library(dlm)

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

2.5.1 Plotting the Forecasted Values:

2.5.1.1 Monthly AMOC Forecast

2.5.1.2 Monthly DML

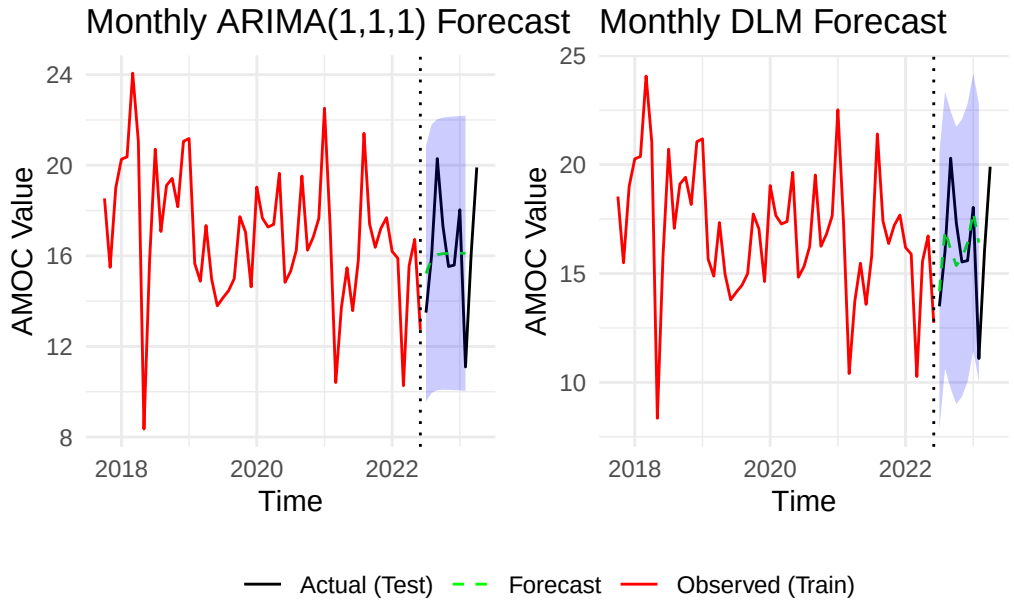


Figure 18: Monthly AMOC forecasts from ARIMA(1,1,1) and DLM models shown side-by-side. ARIMA demonstrates slightly narrower prediction intervals with improved forecast accuracy relative to DLM over the 8-month test period.

Quarterly ARIMA

Quarterly DLM

```r
#| echo: false
#| message: false
#| warning: false
library(patchwork)

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

```
Don't know how to automatically pick scale for object of type <ts>. Defaulting
```

```
to continuous.

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_line()`).

Warning: Removed 21 rows containing missing values or values outside the scale range
(`geom_line()`).
Removed 21 rows containing missing values or values outside the scale range
(`geom_line()`).

Don't know how to automatically pick scale for object of type <ts>. Defaulting
to continuous.

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_line()`).
Removed 21 rows containing missing values or values outside the scale range
(`geom_line()`).
Removed 21 rows containing missing values or values outside the scale range
(`geom_line()`).
```

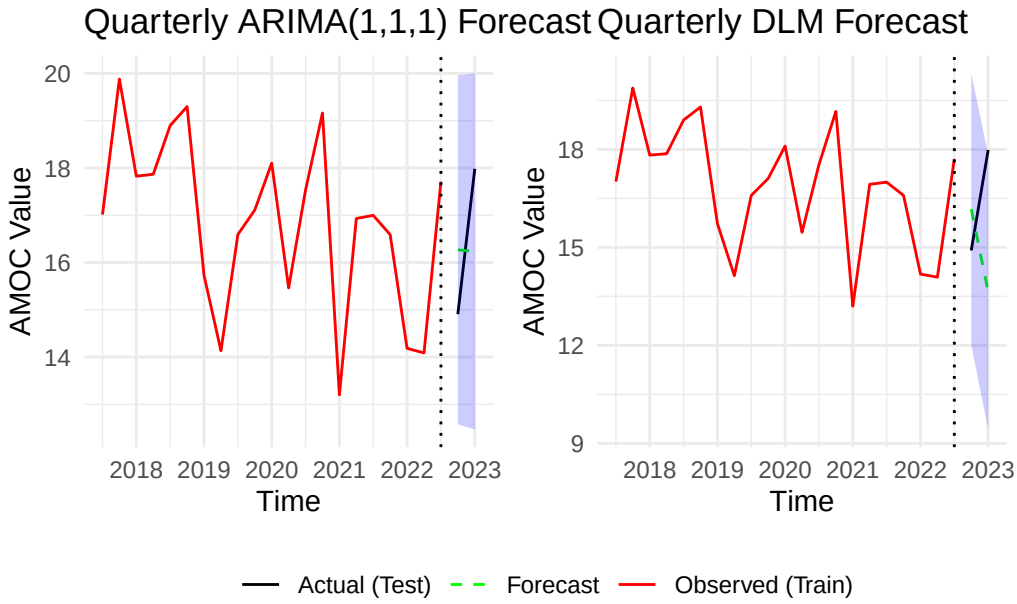


Figure 19: Quarterly AMOC forecasts from ARIMA(1,1,1) and DLM models shown side-by-side. While both models capture the broad trend, ARIMA outperforms DLM with lower forecast uncertainty and error over the 2-quarter test set.

Figures X and Y 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.

2.5.1.3 Model Prediction Accuracy

Summary Table

Table 11: 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.

2.5.1.4 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

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

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.

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

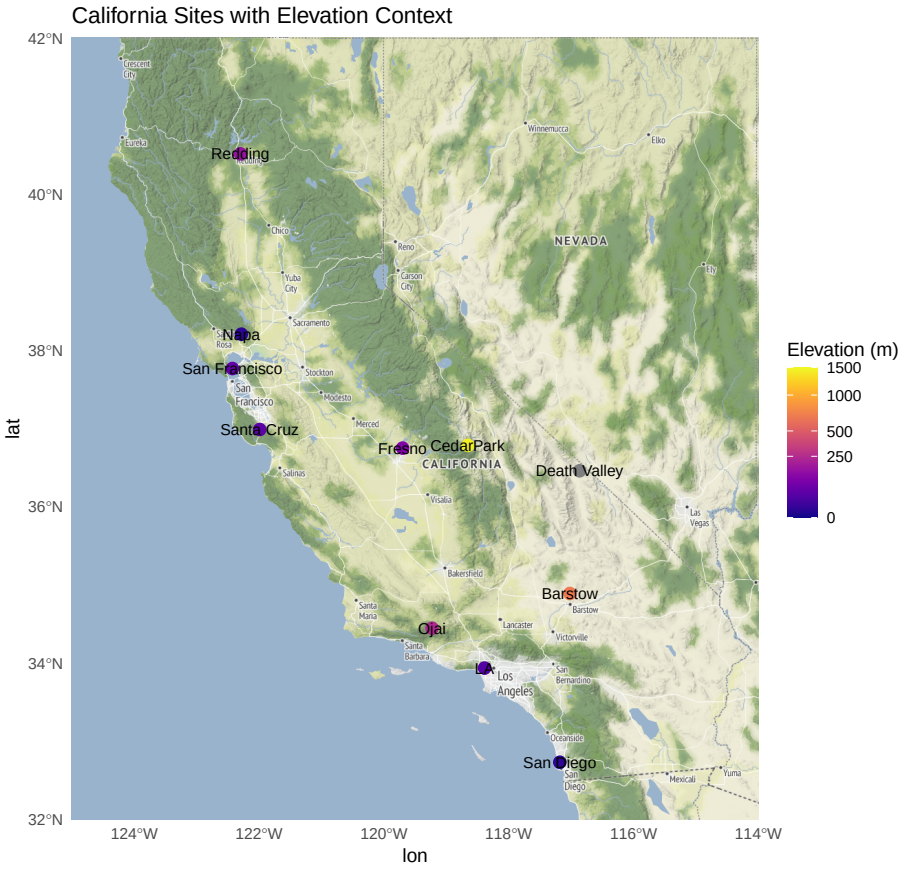


Figure 20: Spatial distribution of the 11 temperature monitoring sites across California, overlaid on a terrain basemap.

Figure 20 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.

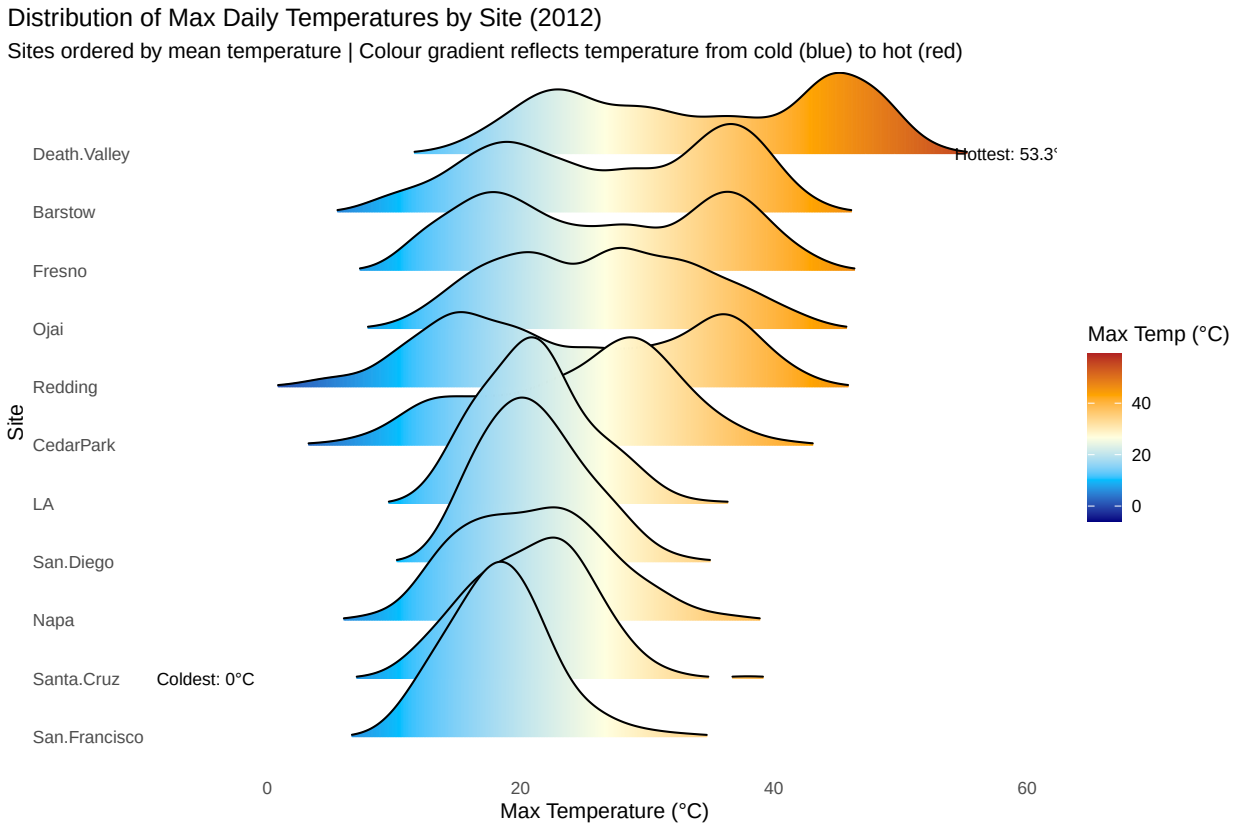


Figure 21: Distribution of maximum daily temperatures across 11 California sites in 2012. Sites are ordered by mean temperature. The colour gradient transitions from cooler temperatures (blue) to warmer temperatures (red), enhancing interpretability. Death Valley recorded the hottest temperature (52.8°C), while Santa Cruz recorded the coldest (-6.1°C).

Figure 21 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.

3.1.1 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.

3.2.1 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))
```

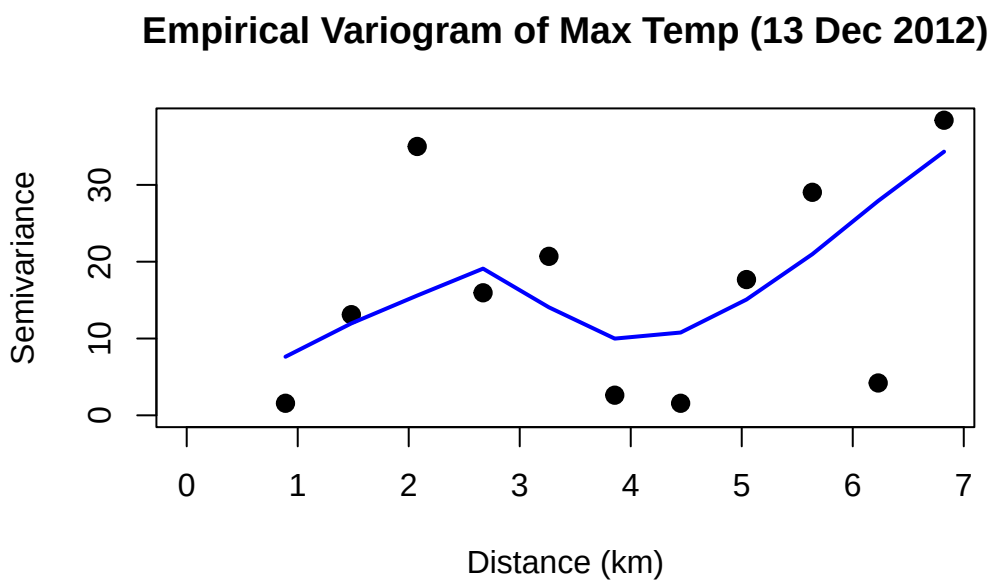
```
# Create geoR-compatible object (use the coordinates and the numeric temperature data)
geo_data_train <- as.geodata(data.frame(x = training_coords$Long,
                                        y = training_coords$Lat,
                                        z = training_data_numeric,
           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

3.2.1.1 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)
```



The empirical variogram (Figure X) 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.

3.2.2 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**.

3.2.2.1 Model 1: Matern Model (Baseline)

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

```
---------------------------------------------------------------
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
```

3.2.2.2 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
```

3.2.2.3 Model 3:

```
# Fit Matérn model with kappa = 1.5
model_gp_matern2.5 <- 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_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.

3.2.3 Comparison of Models:

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

Table 12: 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.

3.2.4 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.

```
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

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

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
```

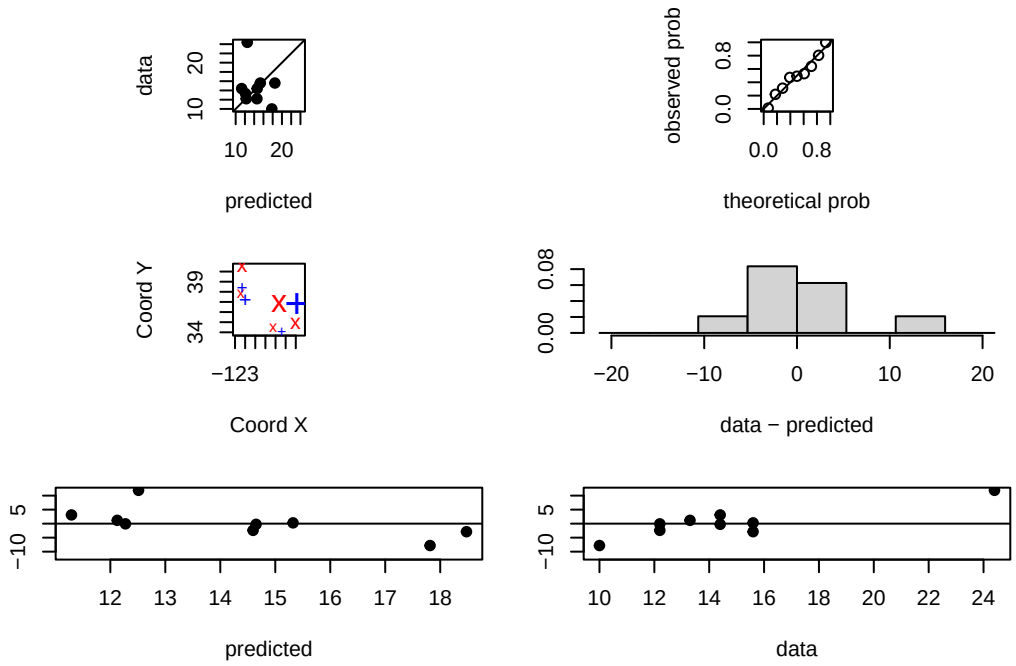
Table 13: Model Comparison: Cross-Validation Metrics (RMSE, MAE, R^2)

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 $\mathbf{R^2}$. Specifically, the models show near-identical **RMSE** and **MAE** values, suggesting that all three models make similarly accurate predictions on unseen data. The $\mathbf{R^2}$ 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.

```
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
```



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.

3.2.5 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**).

3.2.6 Comparison of Predicted vs Real Temperatures:

```
# Define the coordinates 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
)
```

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

```
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
```

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:

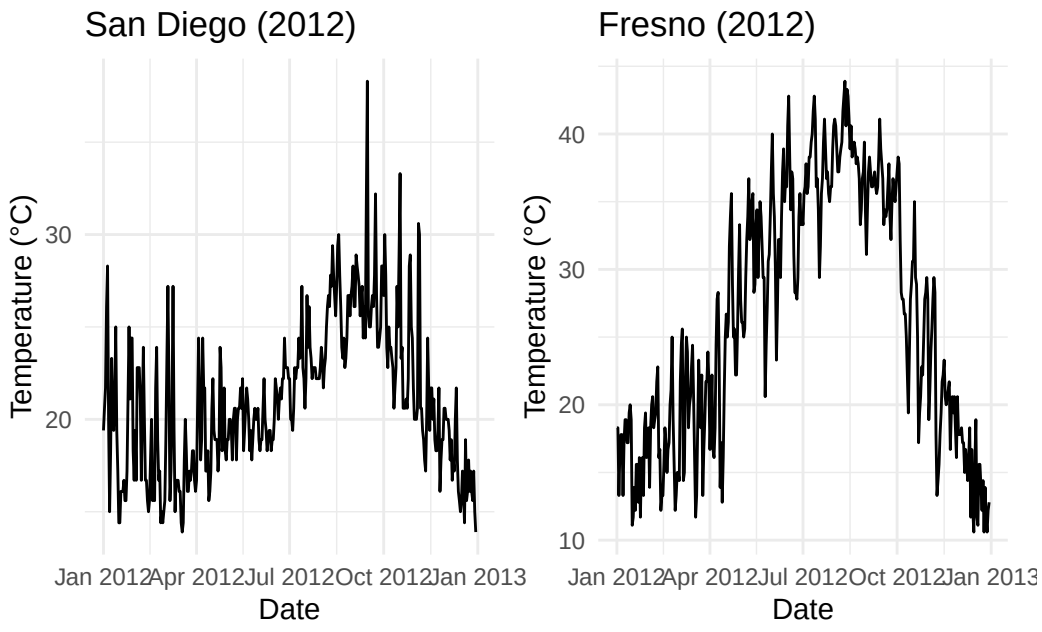
Data Preparation:

```
'data.frame':   365 obs. of  12 variables:
 $ Date         : int  20120101 20120102 20120103 20120104 20120105 20120106 20120107 20120108
 $ San.Francisco: num  14.4 12.8 11.7 13.9 16.1 13.3 17.8 18.3 15.6 15 ...
 $ Napa         : num  16.7 16.7 15.6 19.4 17.8 14.4 20.6 18.3 18.3 17.2 ...
 $ San.Diego    : num  19.4 20.6 21.7 26.1 28.3 20 15 18.3 23.3 20.6 ...
 $ Fresno       : num  18.3 18.3 13.3 16.7 17.8 17.8 13.3 17.2 18.9 18.9 ...
 $ Santa.Cruz   : num  22.8 15 17.2 18.9 18.3 15 20 20 20 16.1 ...
 $ Death.Valley : num  20.6 21.1 20.6 21.1 21.7 21.1 21.1 25 23.3 22.2 ...
 $ Ojai         : num  27.2 27.2 26.7 27.2 26.7 23.9 20.6 20.6 22.2 20 ...
 $ Barstow      : num  20.6 17.2 18.3 18.9 19.4 20 19.4 17.2 15.6 16.7 ...
 $ LA           : num  27.2 23.9 24.4 29.4 28.3 22.8 15.6 20 23.9 19.4 ...
```

```
 $ CedarPark    : num  19.4 21.7 10.6 3.3 8.9 16.1 20 22.2 22.2 14.4 ...
 $ Redding      : num  17.2 15 18.3 19.4 19.4 17.2 16.1 20 16.1 16.7 ...
```

```
      Date San.Francisco Napa San.Diego Fresno Santa.Cruz Death.Valley Ojai
1 20120101          14.4 16.7      19.4   18.3       22.8         20.6 27.2
2 20120102          12.8 16.7      20.6   18.3       15.0         21.1 27.2
3 20120103          11.7 15.6      21.7   13.3       17.2         20.6 26.7
4 20120104          13.9 19.4      26.1   16.7       18.9         21.1 27.2
5 20120105          16.1 17.8      28.3   17.8       18.3         21.7 26.7
6 20120106          13.3 14.4      20.0   17.8       15.0         21.1 23.9
  Barstow   LA CedarPark Redding
1    20.6 27.2      19.4    17.2
2    17.2 23.9      21.7    15.0
3    18.3 24.4      10.6    18.3
4    18.9 29.4       3.3    19.4
5    19.4 28.3       8.9    19.4
6    20.0 22.8      16.1    17.2
```

```r
# 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
```



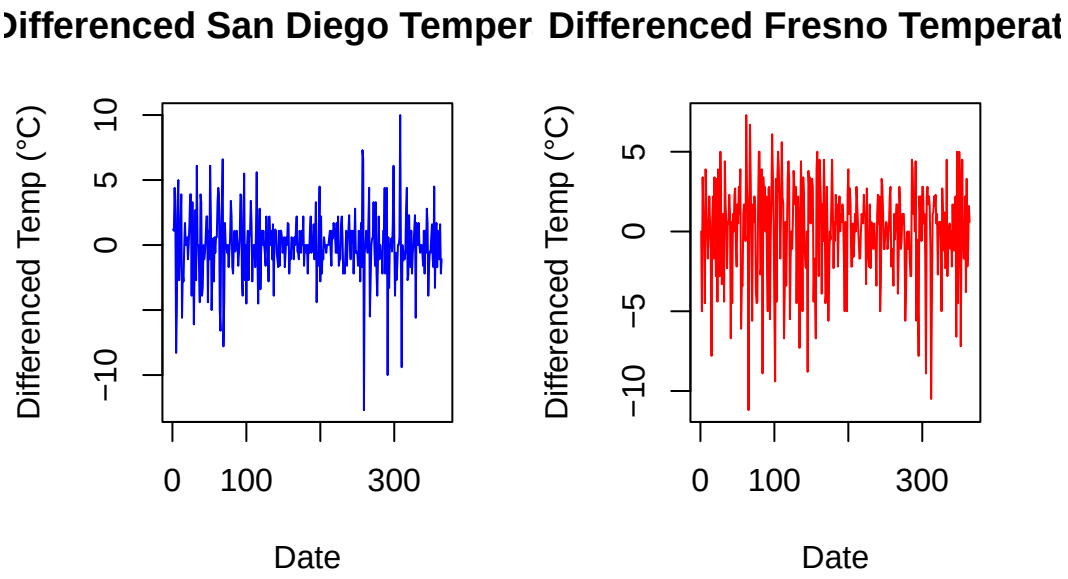
The time series plots for both **San Diego** and **Fresno** in 2012 clearly reveal seasonal temperature patterns, with **San Diego** exhibiting a smoother variation due to its coastal location, while **Fresno** shows more extreme fluctuations typical of inland areas.

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**.

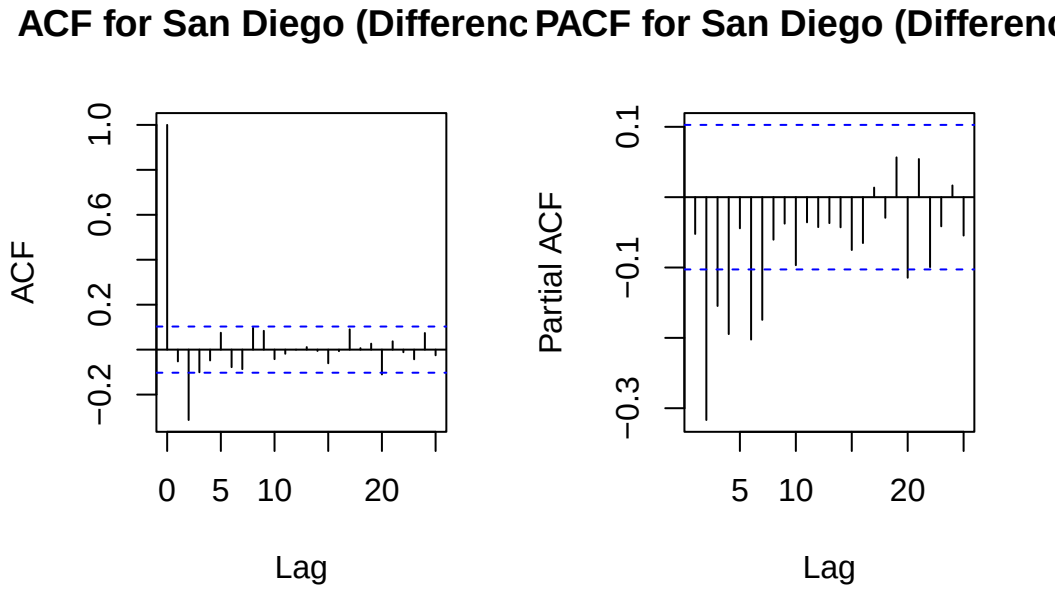
This is an important consideration for the time series modeling, as non-stationary data violates the assumptions of traditional ARIMA models, which rely on a constant mean and variance. Consequently, to prepare the data for effective forecasting, we will need to apply **differencing** to remove these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

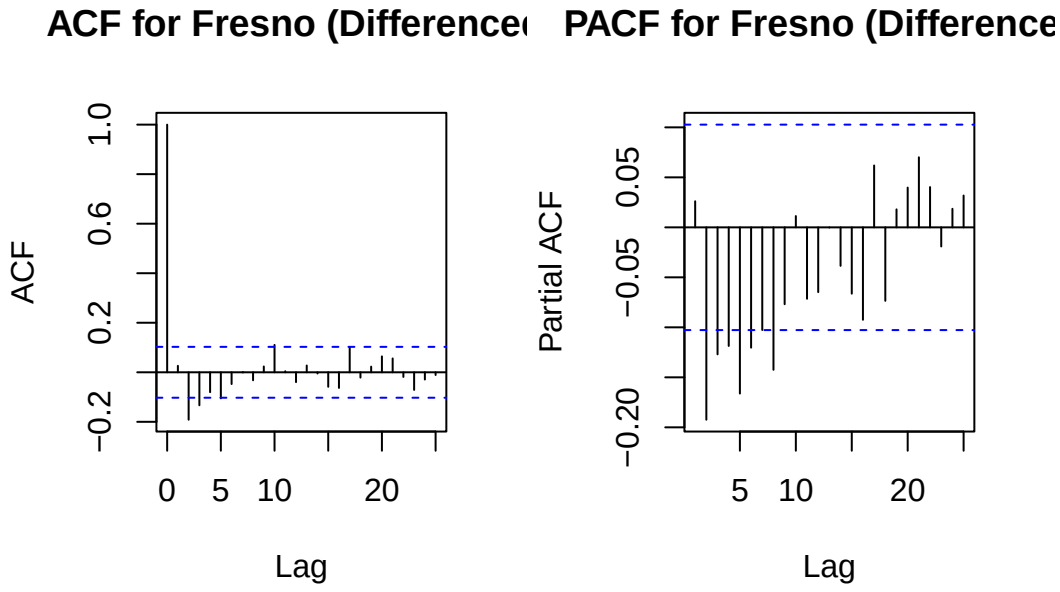
3.3.1 Model Fitting



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.

3.3.2 ARIMA/ARMA Model Fitting





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)** component 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.

3.3.3 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
```

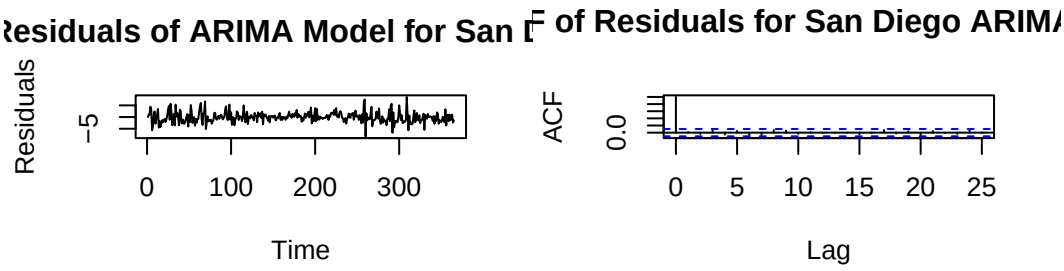
```
Training set -0.04093831 2.849064 2.192333 -1.80572 10.24264 0.9241565
                     ACF1
Training set -0.005409219
```

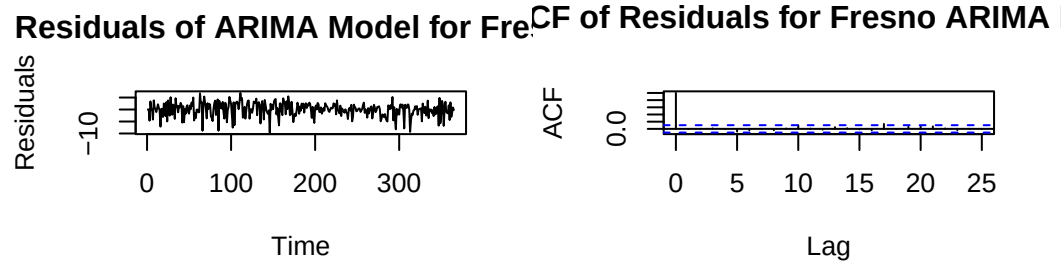
- **San Diego**: The ARIMA(2,1,1) model fits the data relatively well with moderate predictive accuracy (8.01% MAPE) and low residual autocorrelation. The model captures the underlying seasonality and error corrections effectively.
- **Fresno**: The ARIMA(2,1,1) model for Fresno shows higher error (10.24% 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.

3.3.3.1 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:





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.

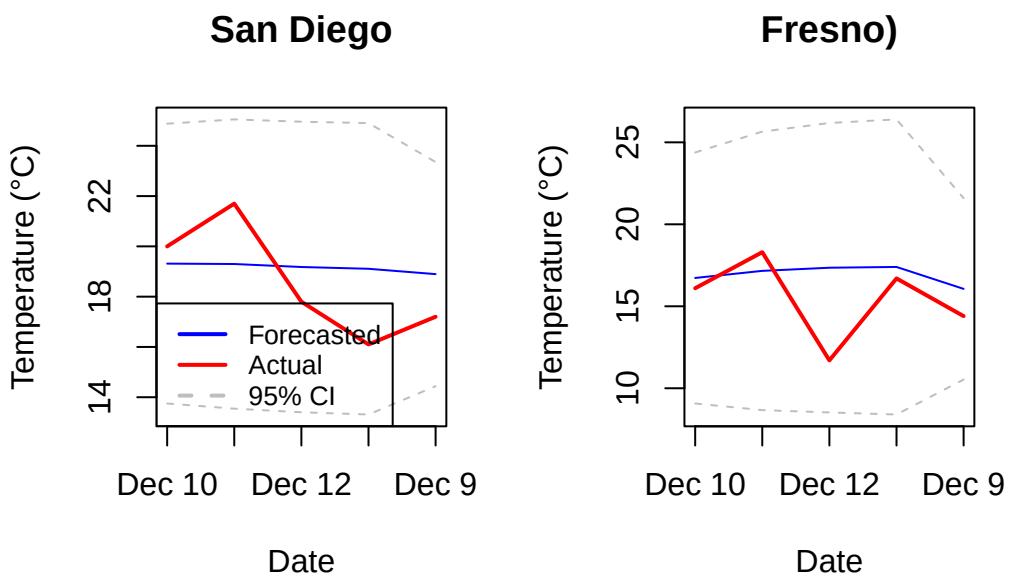
3.4 Forecasting

3.4.1 Forecasting 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**.

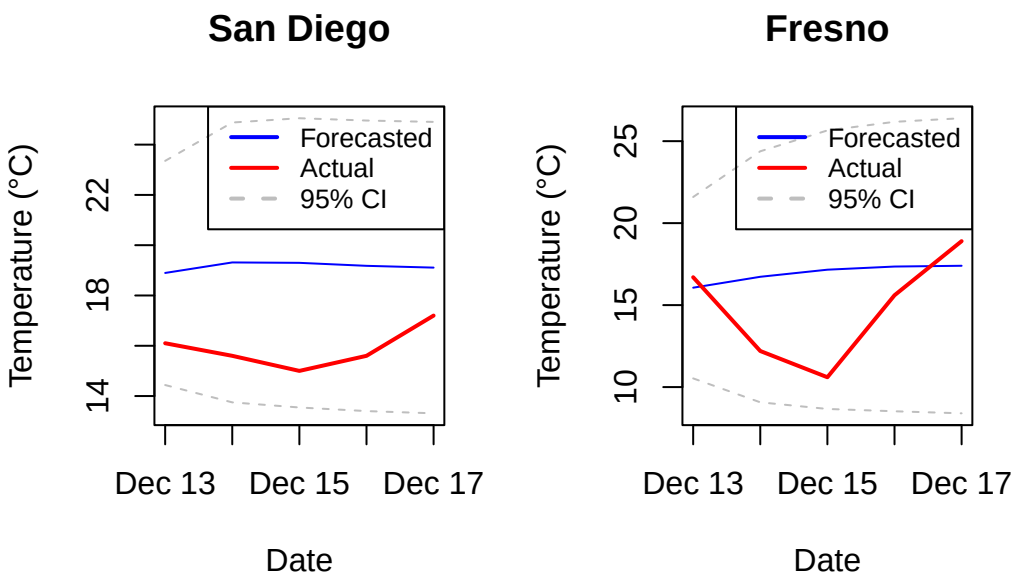
```
train_sandiego <- san_diego_data$San.Diego[1:343]  # Training on first 343 data points (up to D
train_fresno <- fresno_data$Fresno[1:343]  # Similarly for Fresno

# 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))
```



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 December 13th–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.



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.

3.4.3 Evaluating Forecast accuracy

Table 14: 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 15: 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**.