University of Colorado Boulder
Department of Civil, Environmental
and Architectural Engineering
CVEN 6833 – Spring 2025

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Homework #1

by **Catalina Jerez**

Problem 1

Derivation of the Link Function, Fisher Score, and Fisher Information Matrix for Poisson and Gamma Distributions.

The **Poisson distribution** models the probability of observing y occurrences in a fixed interval, given the rate parameter λ:

$$P(Y = y|\lambda) = \frac{\lambda^y e^{-\lambda}}{y!}, \quad y = 0, 1, 2, \ldots$$

where $\lambda > 0$ represents the expected number of occurrences.

1. In Generalized Linear Models (GLMs), we relate the expected value of the response variable to a linear predictor via a **link function** $g(\lambda) = \eta = X\beta$, where $g(\cdot)$ is the link function. For the Poisson distribution, the *canonical link function* is the *log link function*:

$$g(\lambda) = \log(\lambda) \quad \Rightarrow \quad \lambda = e^{\eta}$$

2. The **Fisher Score** (or Gradient of Log-Likelihood) for a single observation y is the log-likelihood function:

$$\ell(\lambda) = y \log \lambda - \lambda - \log y! \quad \text{(taking the derivative w.r.t } \lambda\text{)}$$
$$\frac{\partial \ell}{\partial \lambda} = \frac{y}{\lambda} - 1$$

Using the transformation $\lambda = e^{\eta}$, we compute $\frac{\partial \lambda}{\partial \eta} = e^{\eta} = \lambda$. Then, applying the chain rule:

$$\frac{\partial \ell}{\partial \eta} = \left(\frac{y}{\lambda} - 1\right) \lambda = y - \lambda$$

3. **Fisher Information Matrix** is the expected negative second derivative of the log-likelihood. Mathematically, $\mathcal{I}(\eta) = -\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \eta^2}\right]$. Computing the second derivative:

$$\begin{aligned} \frac{\partial^2 \ell}{\partial \eta^2} &= -\lambda \quad (\text{using } \mathbb{E}[y] = \lambda \ ) \\ \Rightarrow \mathcal{I}(\eta) &= -\mathbb{E}(-\lambda) \\ \mathcal{I}(\eta) &= \lambda \quad (\text{replacing } \lambda = e^{\eta} \ ) \\ \mathcal{I}(\eta) &= e^{\eta} \end{aligned}$$

For a sample of size n, the **total Fisher information** is $\mathcal{I}(\eta) = ne^{\eta}$.

The **Gamma distribution** is commonly used to model positive continuous data. The probability density function (PDF) is:

$$f(y|\alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} y^{\alpha-1} e^{-\beta y}, \quad y > 0$$

where α is the shape parameter, and β is the rate parameter (inverse of the scale parameter). We use a mean parameterization where $\mathbb{E}[Y] = \mu = \frac{\alpha}{\beta}$.

1. The **canonical link function** for the Gamma distribution is the inverse link function:

$$g(\mu) = \frac{1}{\mu} \quad \Rightarrow \quad \mu = \frac{1}{\eta}$$

2. The log-likelihood function for a single observation is (**Fisher Score**) is:

$$\begin{aligned} \ell(\mu) &= \alpha \log \beta + (\alpha - 1) \log y - \beta y - \log \Gamma(\alpha) \quad (\text{replacing } \beta = \frac{\alpha}{\mu}) \\ \ell(\mu) &= \alpha \log \left(\frac{\alpha}{\mu}\right) + (\alpha - 1) \log y - \frac{\alpha}{\mu} y - \log \Gamma(\alpha) \quad (\text{derivative w.r.t } \mu) \\ \frac{\partial \ell}{\partial \mu} &= -\frac{\alpha}{\mu} + \frac{\alpha}{\mu^2} y \quad (\text{ using } \mu = \frac{1}{\eta} \ ) \\ \frac{\partial \mu}{\partial \eta} &= -\frac{1}{\mu^2} \quad (\text{ applying the chain rule}) \\ \frac{\partial \ell}{\partial \eta} &= \left(-\frac{\alpha}{\mu} + \frac{\alpha}{\mu^2} y\right)\left(-\frac{1}{\mu^2}\right) \\ \frac{\partial \ell}{\partial \eta} &= \alpha \frac{y - \mu}{\mu^2} \end{aligned}$$

1. For the F**isher Information Matrix**, we compute the second derivative of:

$$\frac{\partial^2 \ell}{\partial \eta^2} = -\alpha \frac{1}{\mu^2} \quad (\text{using } \mathbb{E}[y] = \mu \ )$$
$$\mathcal{I}(\eta) = \alpha \frac{1}{\mu^2}$$

Thereby, for a sample of size n, the total Fisher information is $\mathcal{I}(\eta) = n\alpha \frac{1}{\mu^2}$.

Thus, Table 1 presents the canonical link function, the gradient of the log-likelihood (Fisher score), and the Fisher information matrix for both the Poisson and Gamma distributions.

Table 1: Summary of the link function, Fisher score, and Fisher information matrix for Poisson and Gamma distributions.

Distribution	**Link Function**	**Fisher Score**	**Information Matrix**
Poisson	$\log(\lambda)$	$y - \lambda$	e^{η}
Gamma	$\frac{1}{\mu}$	$\alpha \frac{y-\mu}{\mu^2}$	$\alpha \frac{1}{\mu^2}$

Problem 2a: Spatial map

Figure 1 shows the average annual precipitation in India, overlayed with topographic data to highlight the relationship between elevation and rainfall.

High-altitude regions, such as the Western Ghats and the Himalayan foothills, receive substantial precipitation, often exceeding 3000 mm annually (red-pink zones), due to orographic effects. In contrast, central and northern India experience significantly lower precipitation (<1000 mm annually), characteristic of arid and semi-arid climates.

Rainfall patterns follow distinct spatial trends: intense rainfall occurs along the western coast due to southwest monsoon winds, which gradually weaken inland. Similarly, northeastern states and Himalayan foothills exhibit high precipitation, while peninsular India shows a declining gradient from coastal to interior regions.

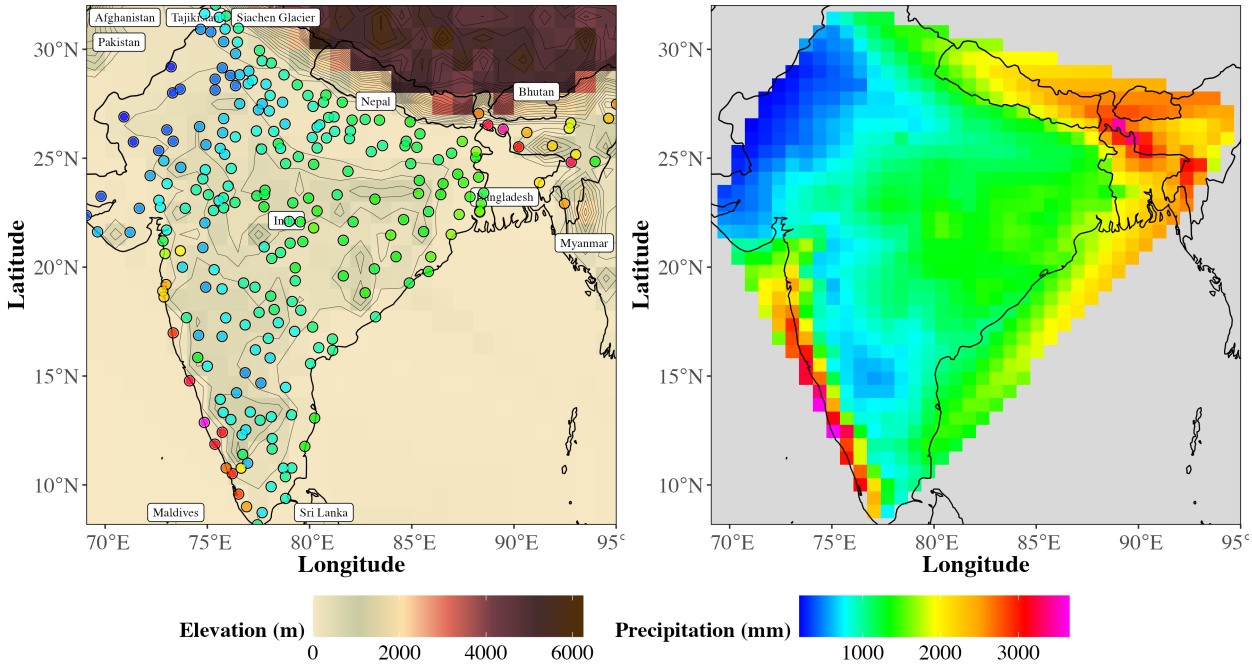

Figure 1: Spatial (left) and Surface (right) map of annual average precipitation along with the topography over India.

Problem 2b: Geodesic Distances

Calculating distances from precipitation stations to the Arabian Sea and the Bay of Bengal was carried out with the `geosphere package` (Hijmans, 2010; Karney, 2013), using the Haversine formula (for spherical distance approximation, Eq. 1) and the Geodesic formula (Vincenty's formula for ellipsoidal distance, Eq. 2) for the calculation of geodetic distances. The minimum distances were calculated from each station to 1) any coast, 2) Arabian Sea and 3) Bay of Bengal.

$$d = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{1}$$

where d is the distance between two points; $R \approx 6378$ Km is Earth's radius; φ_1, φ_2 the latitudes of points; and λ_1, λ_2 the longitudes of points.

$$s = a \tan^{-1}\left(\frac{\sqrt{(\cos U_2 \sin\lambda)^2 + (\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos\lambda)^2}}{\sin U_1 \sin U_2 + \cos U_1 \cos U_2 \cos\lambda}\right) \tag{2}$$

where s is the geodesic distance, a is the semi-major axis of Earth (we use $a = 6378137$ for WGS84), $U_1, U_2 =$ reduced latitudes, λ the difference in longitude.

Figure 2 illustrates the spatial distribution of minimum distances calculated using the Haversine formula. The pattern shows that locations closer to the Arabian Sea and the Bay of Bengal have shorter distances, which aligns with monsoonal precipitation influences. However, the Haversine formula assumes a perfect sphere, leading to minor inaccuracies, especially for larger distances.

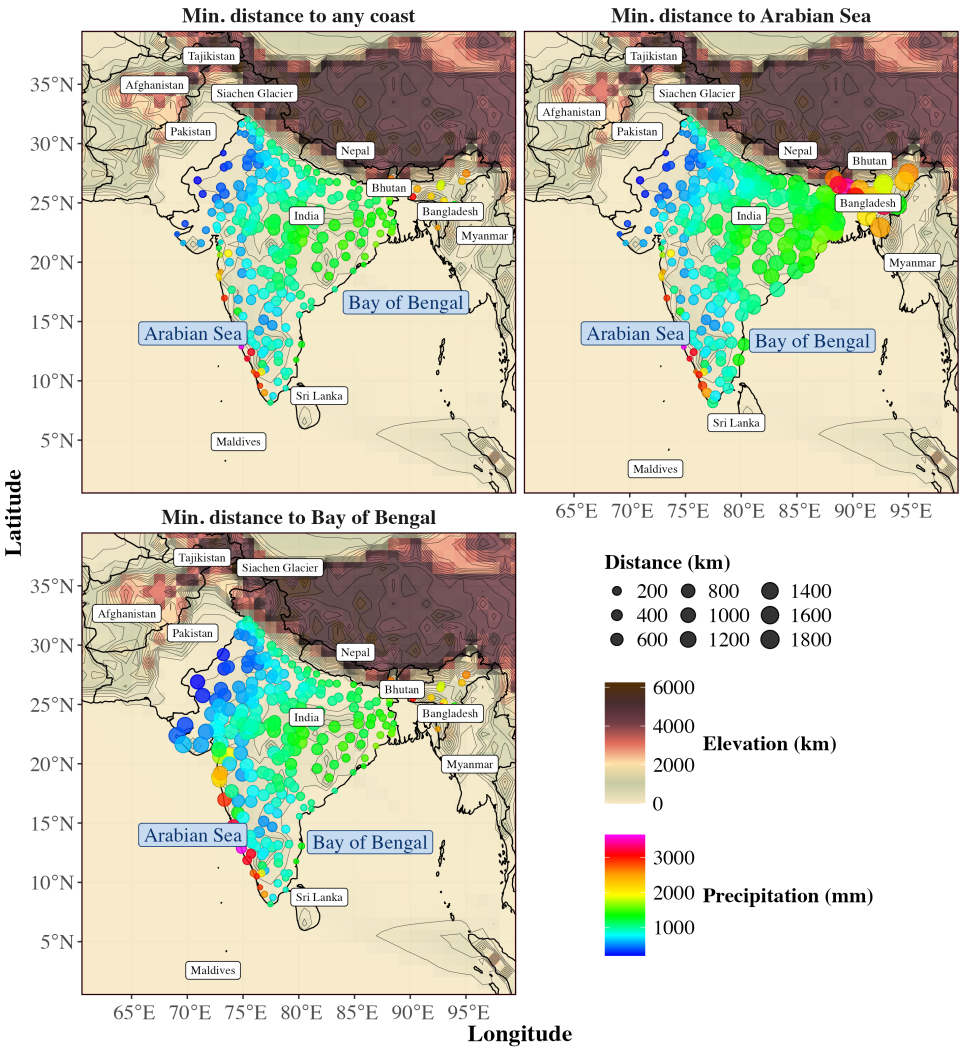

Figure 2: Annual average precipitation and topography over India, and distance calculated with Harvesine method from weather stations to Arabian and Bay of Bengal Sea.

Figure 3 shows the geodesic-based distance computation that accounts for the Earth's ellipsoidal shape, making it a more accurate representation of actual distances. The figure demonstrates similar trends as the Haversine method but with slight variations due to the improved accuracy of the geodesic approach.

Figure 4 provides a comparison between the two methods. The differences are more noticeable over longer distances, where the Haversine method tends to slightly underestimate or overestimate distances due to its spherical assumption.

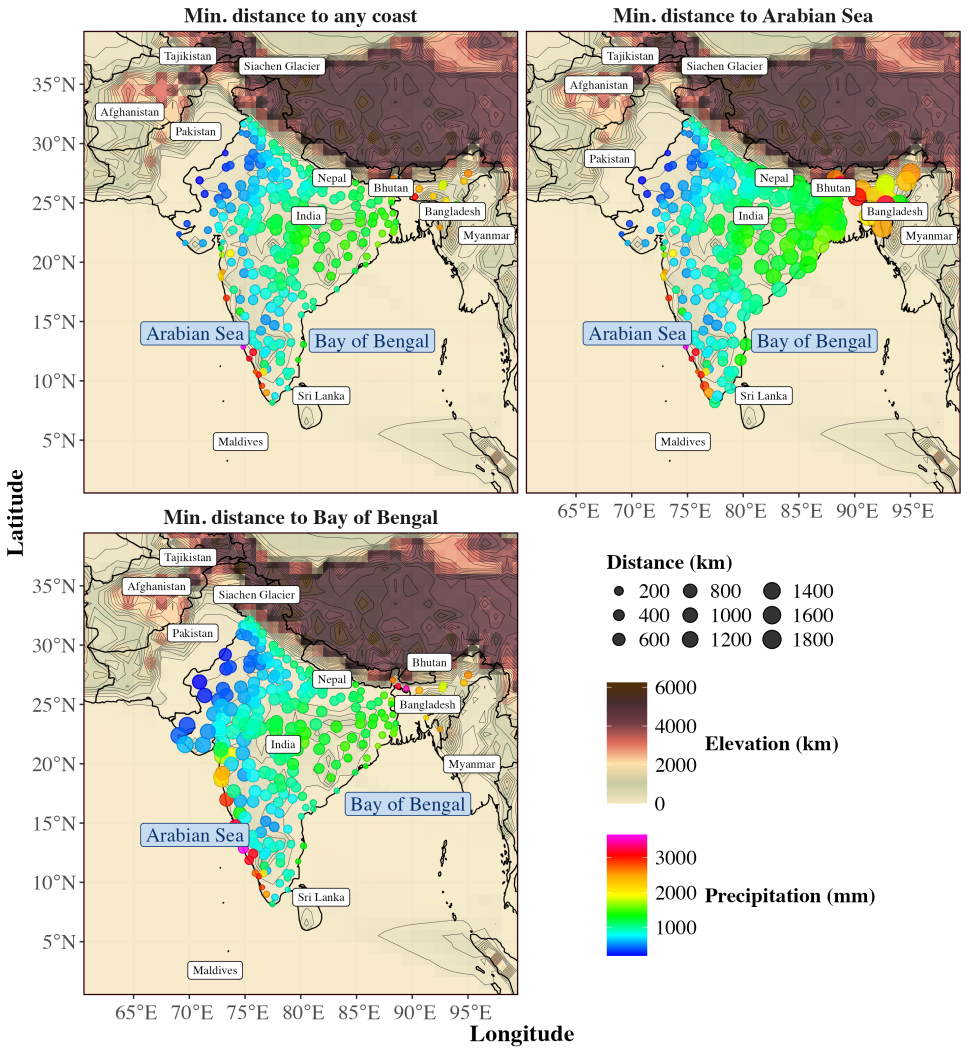

Figure 3: Annual average precipitation and topography over India, and distance calculated with Geodesic method from weather stations to Arabian and Bay of Bengal Sea.

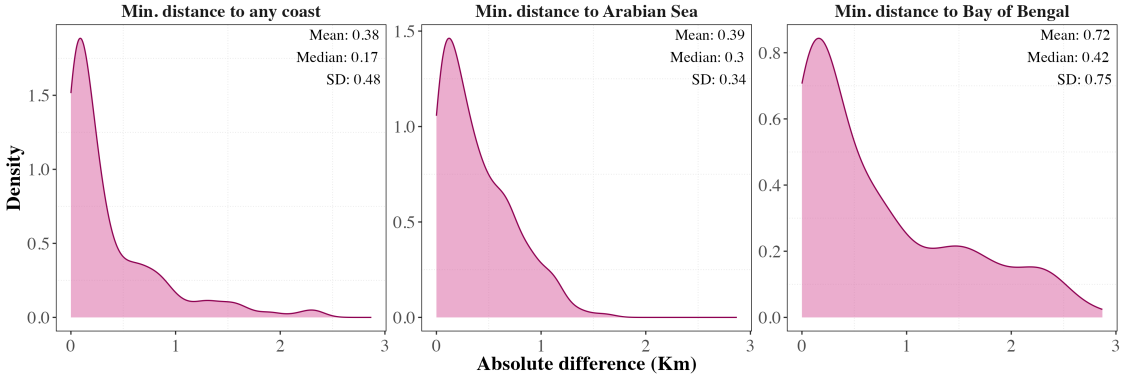

Figure 4: Comparison of Haversine vs. Geodesic Distances

Problem 2c: Underlying precipitation surface under linear model

The goal of this analysis is to estimate the underlying precipitation surface using sparse spatial observations. This is crucial for applications such as floodplain management, natural hazard mitigation, and climate modeling. The study utilizes regression models, cross-validation, and spatial mapping techniques.

Model selection and fitting

The best model was selected based on the Bayesian Information Criterion (BIC). The initial regressors included Latitude (lat), Longitude (lon), Elevation (elev), Distance to the Arabian Sea (mdArabian), and Distance to the Bay of Bengal (mdBengal). A Variance Inflation Factor (VIF) analysis was conducted to check for multicollinearity, and mdArabian and mdBengal were removed due to high VIF values (see Appendix C.2, Figure 5). The final selected predictors are lat, lon, elev, and mdCoast. Additionally, we include the following interaction terms: lld (lon $\times$ lat $\times$ mdCoast), ll2d (lon $\times$ lat^2 $\times$ mdCoast), l2ld (lon^2 $\times$ lat $\times$ mdCoast), and ed (elev $\times$ mdCoast).

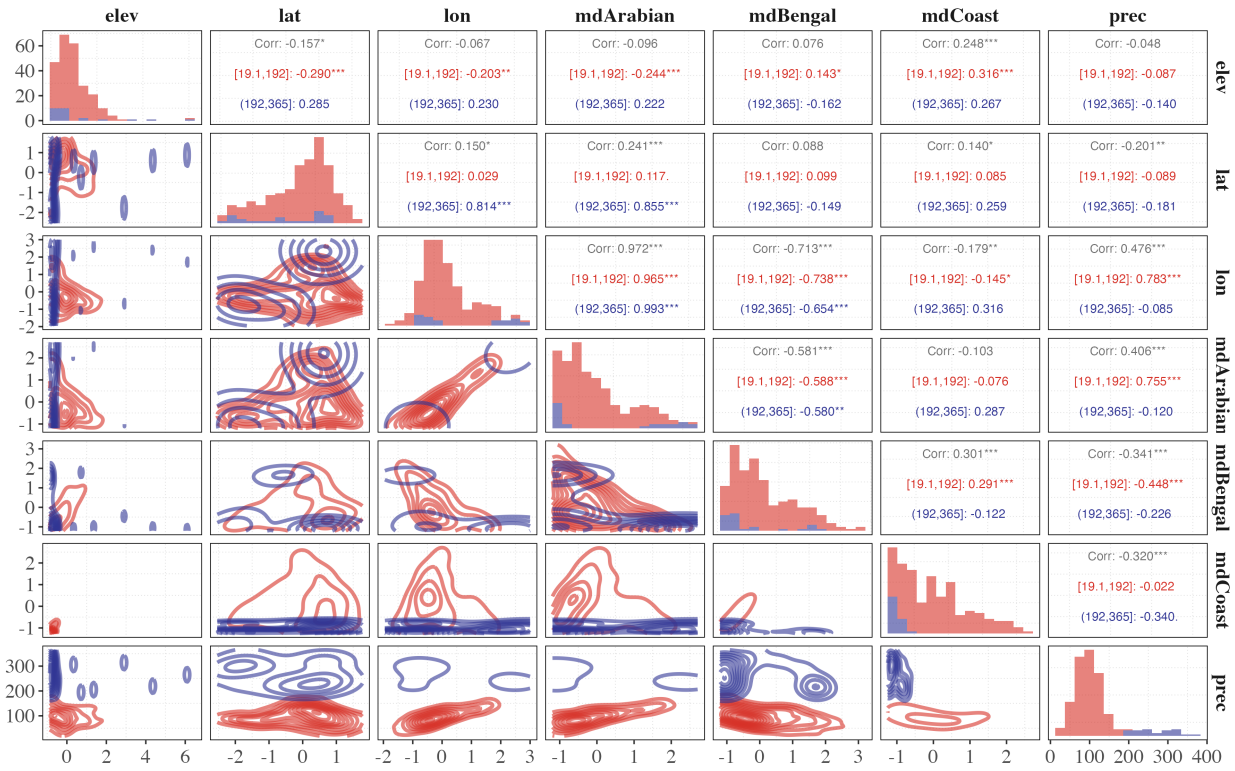

Figure 5: Caption

Scatterplot of observed vs modeled precipitation

The R^2 value of 0.67 (Figure 6) suggests a moderate to strong correlation; however, 33% of the variance remains unexplained, which suggests potential improvements. Besides, the model returns a RMSE of 36.24 cm, which might be large depending on the context (e.g., hazard mitigation).

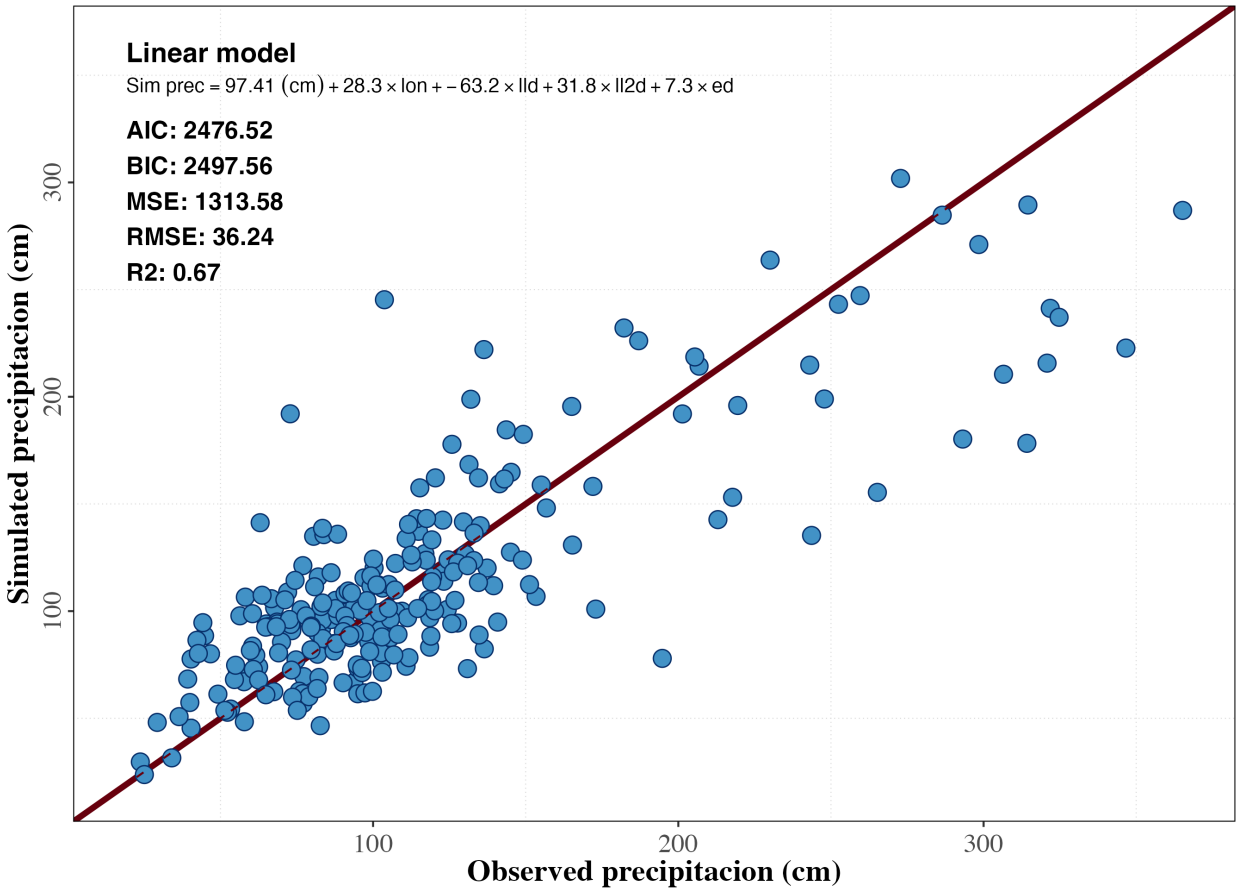

Figure 6: Scatterplot of observed vs modeled precipitation for linear regression model.

Model diagnostics

The ANOVA analysis confirms that the linear model is statistically significant ($p < 0.001$), indicating that at least one predictor contributes significantly to the response variable (precipitation). The results of ANOVA are summarized as: `lon`: significant ($p <$ 2e-16, strong influence); `lld`: highly significant ($p <$ 2e-16, major contribution); `ll2d`: also significant ($p <$ 2e-16, meaningful but smaller effect); `ed`: statistically significant ($p = 0.00803$, weaker influence); `Residuals`: variability remains, suggesting room for model refinement.

Figure 7 provides essential model validation checks:

a) Normality: the Q-Q plot shows that residuals are approximately normal, though slight deviations exist at the tails.

b) Histogram: residual distribution appears approximately normal, though some skewness may be present.

c) Independence: the autocorrelation function (ACF) suggests that residuals are largely uncorrelated, supporting the assumption of independence.

d) Homoscedasticity: the residuals vs. estimated precipitation plot shows no clear pattern, indicating relatively constant variance.

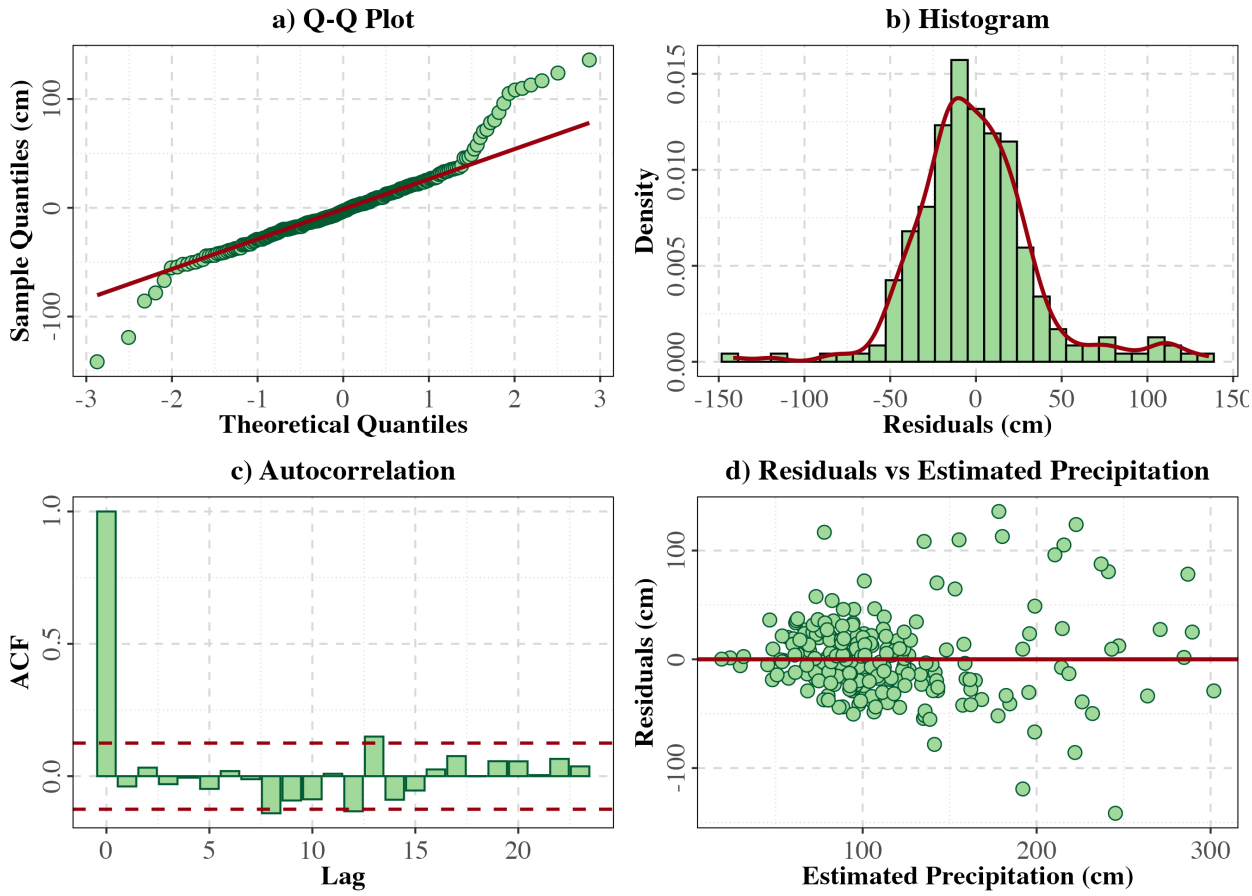

Figure 7: Model diagnostics for linear regression.

Cross-Validation Analysis

Leave-One-Out Cross-Validation (LOOCV) was performed to assess the model's generalization ability. The scatterplot in Figure 8 compares observed and simulated precipitation values from the cross-validated model, showing results consistent with the initial training model. However, the R^2 value (0.63) is slightly lower than the training model's R^2 of 0.67 (Figure 6), suggesting a small drop in predictive performance, which is expected in cross-validation due to reduced bias. In addition, the RMSE increase in 5% (from 36.24 to 38.09), indicating a minor generalization error.

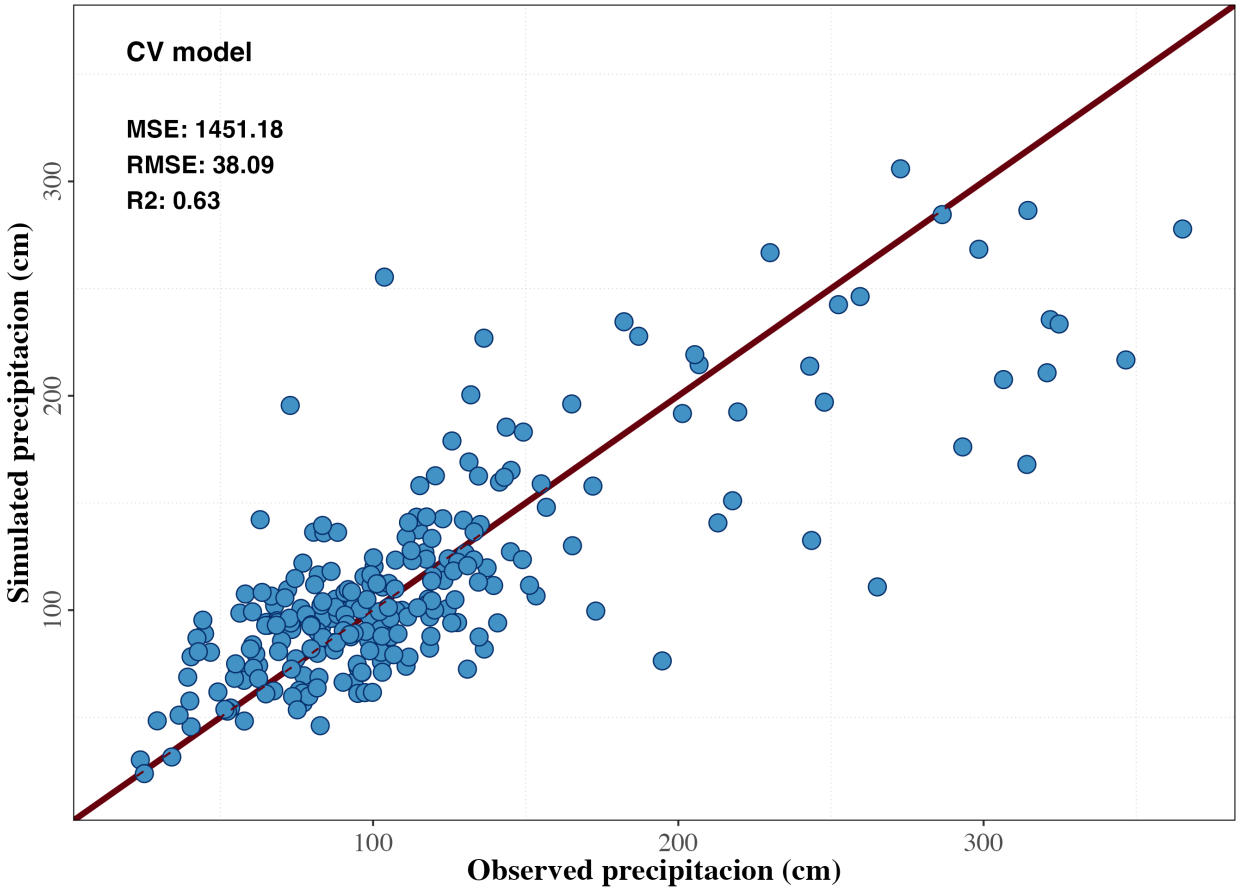

Figure 8: LOOCV Scatterplot.

Model robustness – cropping 10% of observations

To evaluate the robustness of the linear model, 10% of the observations were randomly removed, the model was re-fitted, and predictions were generated for the omitted data. This process assesses how well the model maintains stability when exposed to missing data.

The RMSE boxplot shows an expected increase in error after dropping 10% of the data (left panel in Figure 9), reflecting the model's slight sensitivity to data loss. Some outliers suggest that certain missing observations had a more significant impact on the prediction error.

The correlation between the observed and predicted values shows a wider spread and lower median correlation values (right panel in Figure 9). Many correlation values cluster around zero, suggesting that the model struggles to maintain predictive consistency when observations are removed. Some data points exhibit negative correlation, indicating possible overfitting in certain cases. Despite this, the overall trend remains relatively stable, though the model demonstrates reduced robustness to missing data.

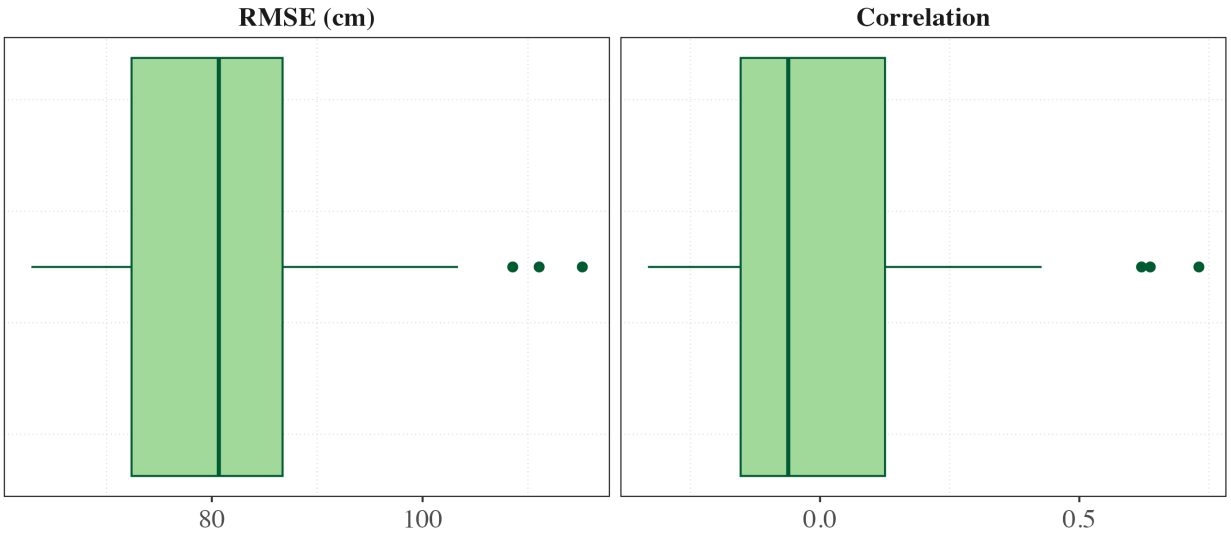

Figure 9: RMSE and correlation boxplots for model robustness analysis.

Spatial mapping

The map on the left of Figure 10 presents the predicted precipitation in India and the right map shows the standard error of predictions.

The predicted precipitation map shows higher precipitation in the northern and coastal regions, aligning with observed patterns (Figure 1). Lower precipitation is seen in central and western India, regions typically associated with drier conditions. The precipitation gradient follows a clear geographical trend, suggesting that the model captures regional climatic variations effectively.

Standard Error map (Figure 11) show lower standard errors are observed in densely sampled regions, indicating higher confidence in predictions. Higher standard errors appear in some inland and less-sampled areas, suggesting potential model uncertainty in regions with fewer observations.

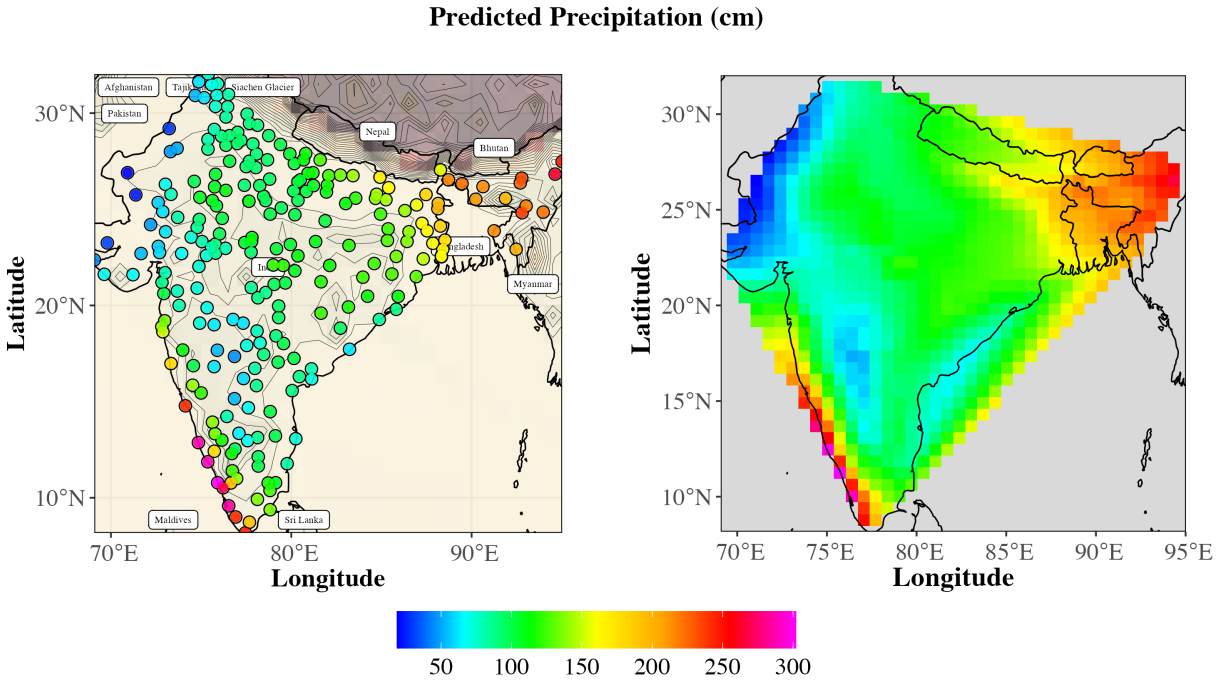

Figure 10: Predicted precipitation surface mapping.

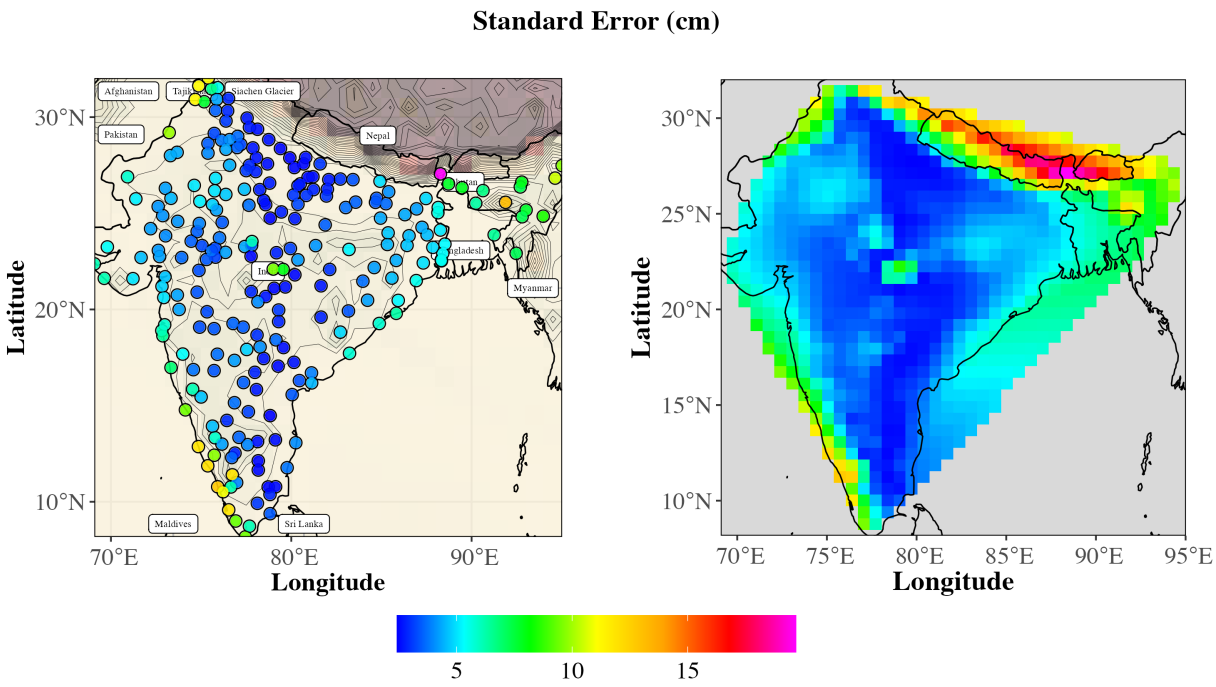

Figure 11: Standard error precipitation surface mapping.

Summary

The linear model was selected based on the lowest BIC, ensuring an optimal balance between predictive power and model complexity. Latitude and longitude emerged as the most significant predictors of precipitation, while elevation had a moderate influence. The distance to the coast (mdCoast) remained an important factor after removing collinear variables.

Model performance & diagnostics:

- The model demonstrated good predictive ability with an R^2 of 0.67 (Training) and 0.63 (LOOCV), indicating a moderate to strong correlation between observed and predicted precipitation.
- Residual analysis suggested that while the model assumptions largely hold, minor deviations in normality and homoscedasticity exist, which could be addressed through further refinements.

Cross-Validation & robustness:

- LOOCV results showed a slight increase in RMSE (from 36.24 to 38.09 cm), reflecting minor generalization error but overall stability.
- Robustness analysis (10% data removal) confirmed model reliability, though some sensitivity to high-leverage observations was observed.
- Correlation remained stable, but RMSE variations suggest that certain influential data points might impact predictions.

Spatial mapping:

- The predicted precipitation map captured key climatic trends, with higher rainfall in northern and coastal regions and lower precipitation in central/western regions.
- The standard error map highlighted higher uncertainty in inland regions, suggesting that improved data coverage in these regions could refine predictions.

Problem 3: Underlying precipitation surface under GLM

Scatterplot of observed vs modeled precipitation

The Generalized Linear Model (GLM) was fitted using the families Gamma, gaussian and binomial, the scatterplot in Figure 12 compares the observed vs. simulated values from the best fitted family.

The AIC and BIC values are slightly higher than the linear model, indicating no major improvement in model selection criteria. The R^2 value (0.67) indicates a strong correlation between observed and predicted precipitation, similar to the linear model. The RMSE of 36.24 cm suggests an error level comparable to the linear regression model. These values suggest that GLM does not provide a significant advantage over the linear approach. The scatterplot follows a strong linear trend, showing that the GLM effectively captures precipitation variations. However, variance increases at higher precipitation levels, which may indicate potential heteroscedasticity.

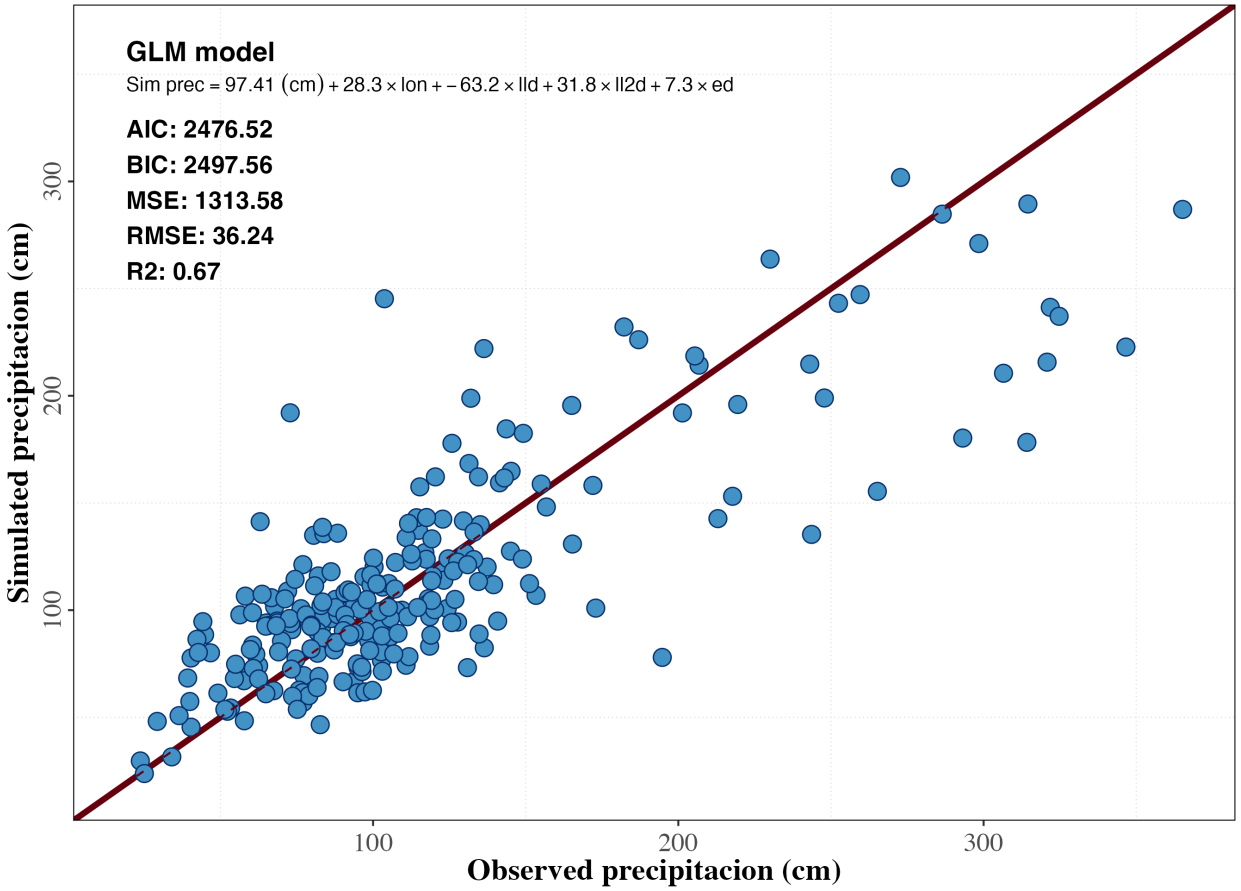

Figure 12: Scatterplot of observed vs modeled precipitation for Generalized Linear Model.

Model diagnostics

To evaluate the performance of the GLM, an ANOVA-based deviance analysis and residual diagnostics were conducted. Figure 13 presents key insights into the model's assumptions and fit quality. The GLM was fitted using a Gaussian distribution with an identity link function. The ANOVA analysis indicates that all predictors (lon, lld, ll2d, ed) are statistically significant ($p < 0.001$ or $p < 0.01$), confirming their contribution to precipitation modeling. Figure 13 provides essential model validation checks:

a) Normality: residuals in Q-Q plot are approximately normally distributed, but minor deviations at the tails suggest some skewness.

b) Histogram: the distribution appears roughly normal, though slight asymmetry and heavier tails suggest possible heteroscedasticity.

c) Independence: low autocorrelation values confirm that residuals are mostly independent, satisfying one of the key model assumptions.

d) Homoscedasticity: spread of residuals appears random, but a slight funnel shape suggests potential heteroscedasticity (variance increases at higher precipitation values).

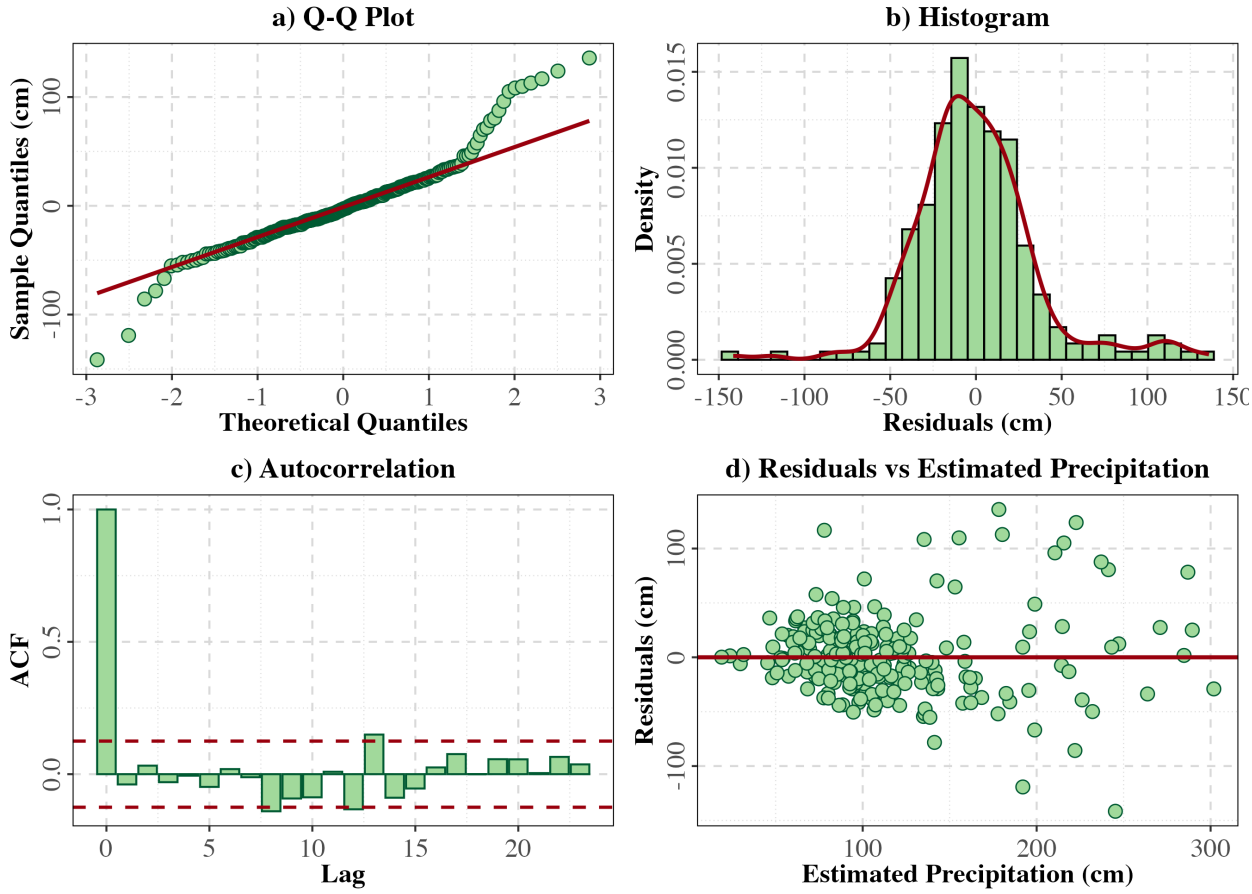

Figure 13: Model diagnostics for GLM regression.

Cross-Validation Analysis

The scatterplot (Figure 14) maintains a strong linear trend, reinforcing that the GLM is capturing overall precipitation patterns effectively. However, variance increases at higher precipitation values, suggesting that GLM predictions are less stable for extreme cases.

LOOCV results for GLM ($R^2 = 0.63$, RMSE = 38.38 cm) are nearly identical to the linear model's LOOCV performance ($R^2 = 0.63$, RMSE = 38.09 cm). This suggests that GLM does not provide a significant advantage over the linear model in terms of predictive accuracy.

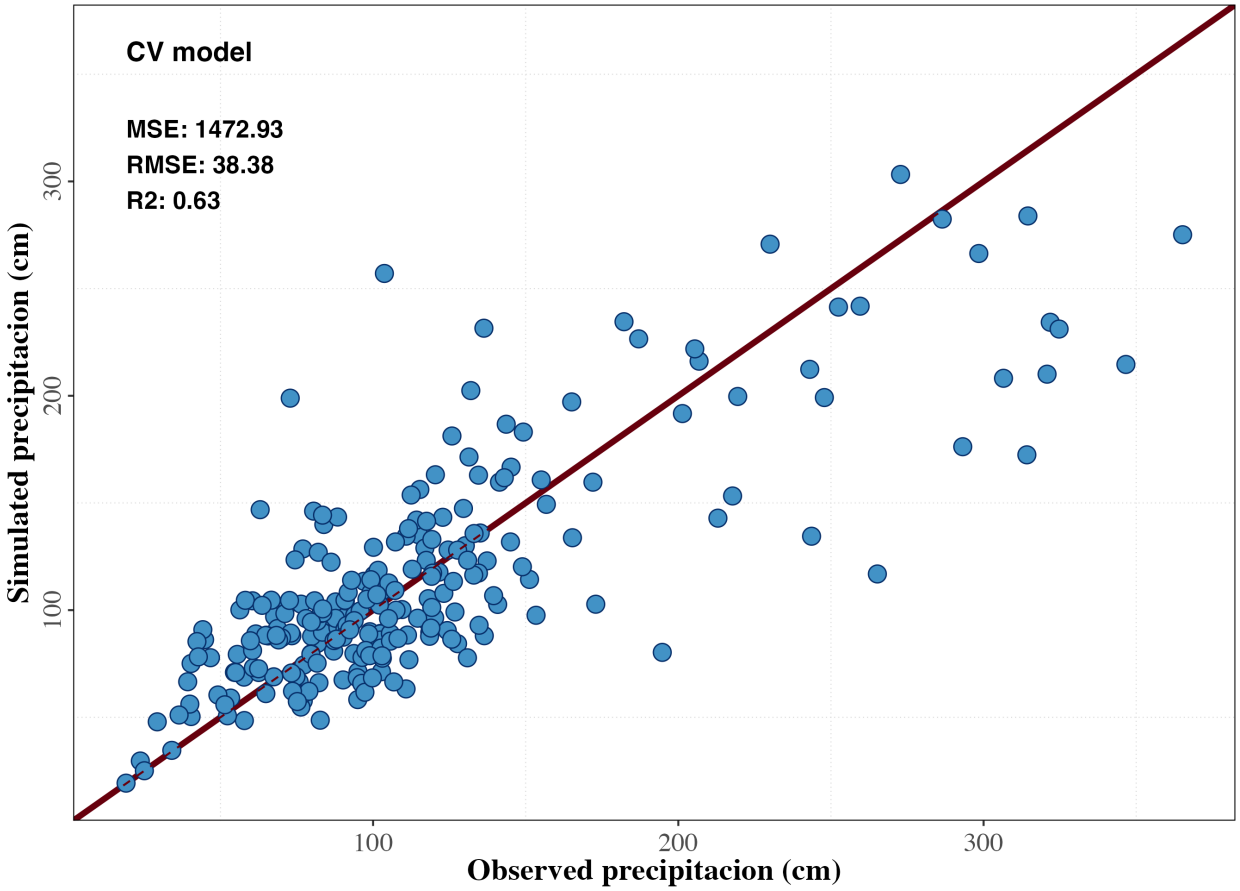

Figure 14: LOOCV Scatterplot.

Model robustness – cropping 10% of observations

The RMSE boxplot (left panel in Figure 15) shows moderate variation, indicating that while the GLM model's performance fluctuates slightly, the error remains within a stable range. Some outliers suggest that certain missing data points had a larger impact, possibly due to influential observations.

The correlation boxplot (right panel in Figure 15) shows a wider spread, with values centered around zero and some negative correlations. This suggests that the relationship between observed and predicted values weakens when 10% of the data is removed, indicating a higher sensitivity to missing data (compared to more flexible models such as GAM). However, the general trend remains stable, reinforcing moderate robustness.

The GLM and linear models exhibit similar robustness, with comparable RMSE and correlation distributions. Both models experience minor sensitivity to high-leverage observations, which may influence predictions. While GLM maintains stability, its sensitivity to certain missing data points could impact its generalization performance.

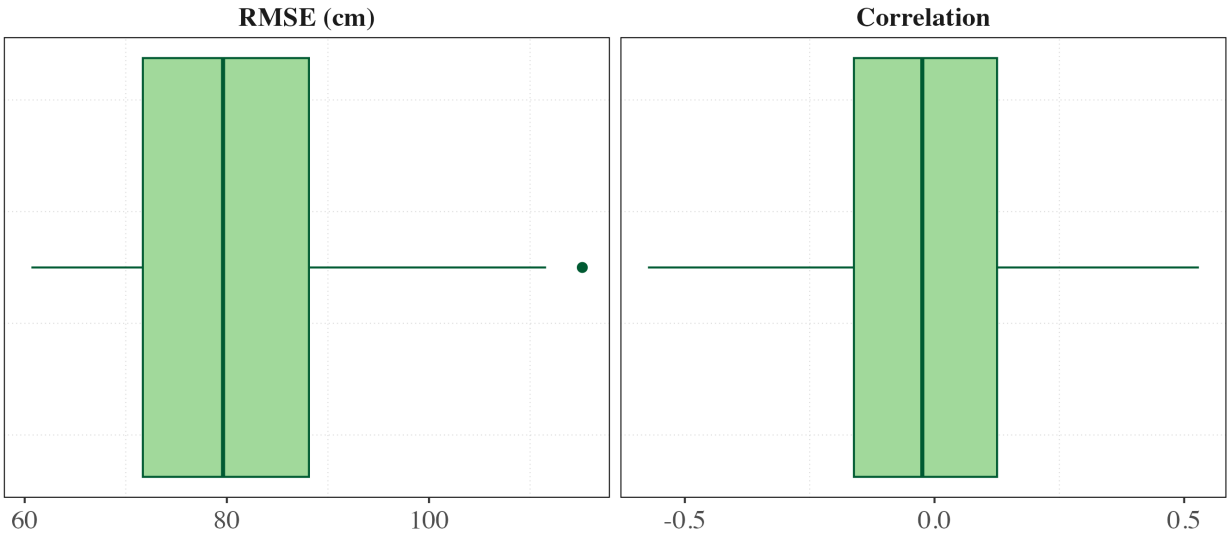

Figure 15: RMSE and correlation boxplots for model robustness analysis.

Spatial mapping

Figure 16 indicates a higher precipitation in northern and coastal regions, aligning well with climatic expectations. Lower precipitation levels are seen in central and western India, consistent with drier climatic conditions. The precipitation gradient appears smooth, suggesting that GLM successfully captures regional variability.

Figure 17 indicates higher standard errors in some inland regions, indicating greater uncertainty in precipitation estimates where fewer observations are available. Lower standard errors in coastal and well-sampled areas suggest more reliable predictions in those regions. Compared to the linear model, the GLM standard error distribution is similar, reinforcing that both models exhibit comparable predictive stability.

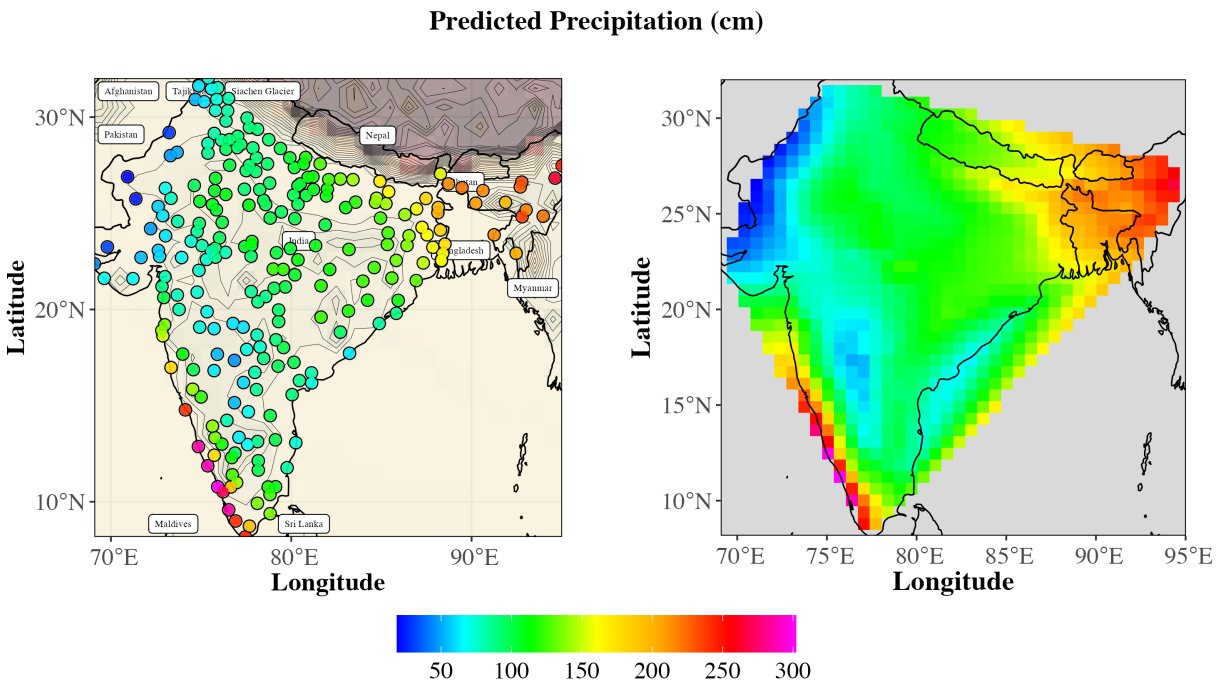

Figure 16: Predicted precipitation surface mapping under GLM.

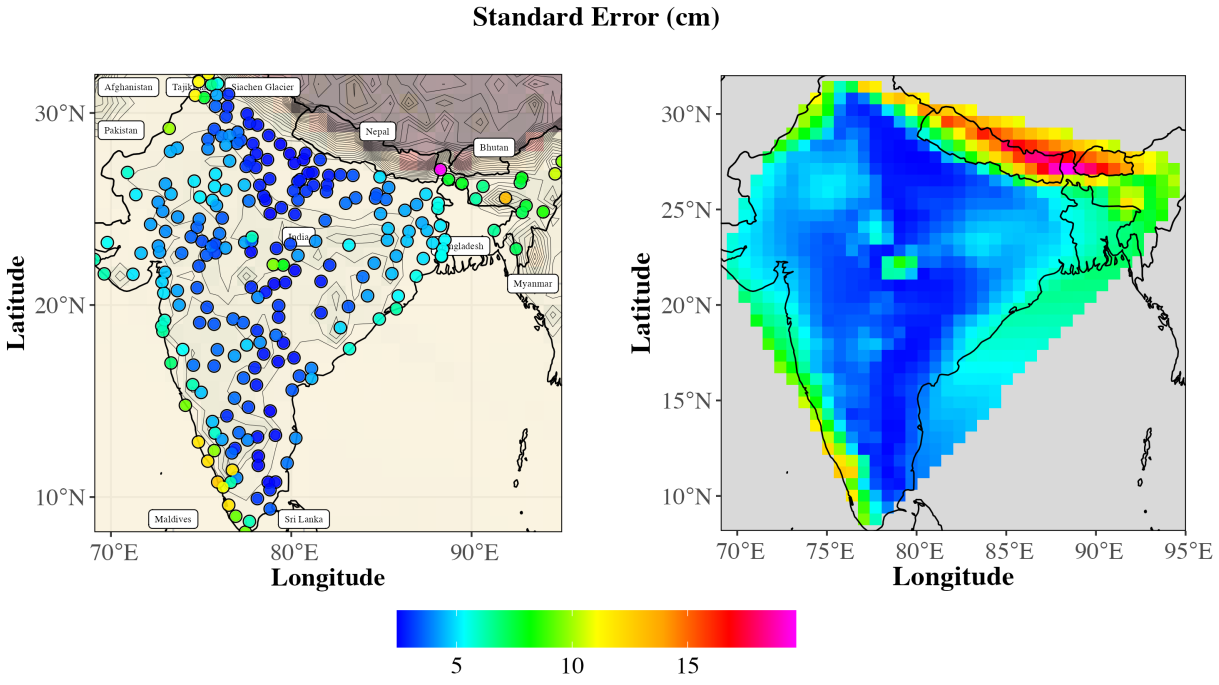

Figure 17: Standard error precipitation surface mapping under GLM.

Summary

The GLM model was evaluated for its predictive accuracy, residual behavior, robustness, and spatial performance. Overall, it demonstrated solid performance, with statistically significant predictors and mostly independent residuals. However, minor deviations in normality and heteroscedasticity indicate areas for potential refinement.

Model performance & diagnostics:

- The GLM fit was strong, with $R^2 = 0.67$ (Training) and $R^2 = 0.63$ (LOOCV).
- Residual analysis showed slight heteroscedasticity, suggesting that transforming the response variable or using a weighted GLM could improve reliability.

Cross-Validation & robustness:

- The RMSE increased slightly (38.38 cm in LOOCV), indicating minor generalization error.
- Predictive performance was nearly identical to the linear model, reinforcing that GLM does not provide significant improvements.
- The model remained stable when 10% of observations were removed, though some sensitivity to influential data points was observed.
- Further leverage diagnostics could identify points that disproportionately affect predictions.

Spatial mapping:

- Predicted precipitation trends align well with known climatic patterns, with higher precipitation in northern and coastal regions.
- Standard error distribution suggests that uncertainty is more influenced by data sparsity than the modeling approach itself.

GLM performs at the same level as the linear model, while the GLM is a viable approach, it does not provide significant gains over the standard linear model, making it a comparable but not superior choice for this dataset.

Problem 4: Underlying precipitation surface under Local GLM

Scatterplot of observed vs modeled precipitation

The scatterplot in Figure 18 exhibits a strong linear trend, showing that the Locfit model closely follows observed precipitation values with fewer deviations. However, variance increases slightly at higher precipitation values, which might indicate some overfitting in extreme cases.

The R^2 value of 0.88 represents a significant improvement over both the linear ($R^2 = 0.67$) and GLM ($R^2 = 0.67$) models, indicating that Locfit captures more variability in precipitation patterns. The RMSE is significantly lower (21.63 cm vs. ∼36-38 cm for previous models), suggesting higher prediction accuracy and reduced error variance.

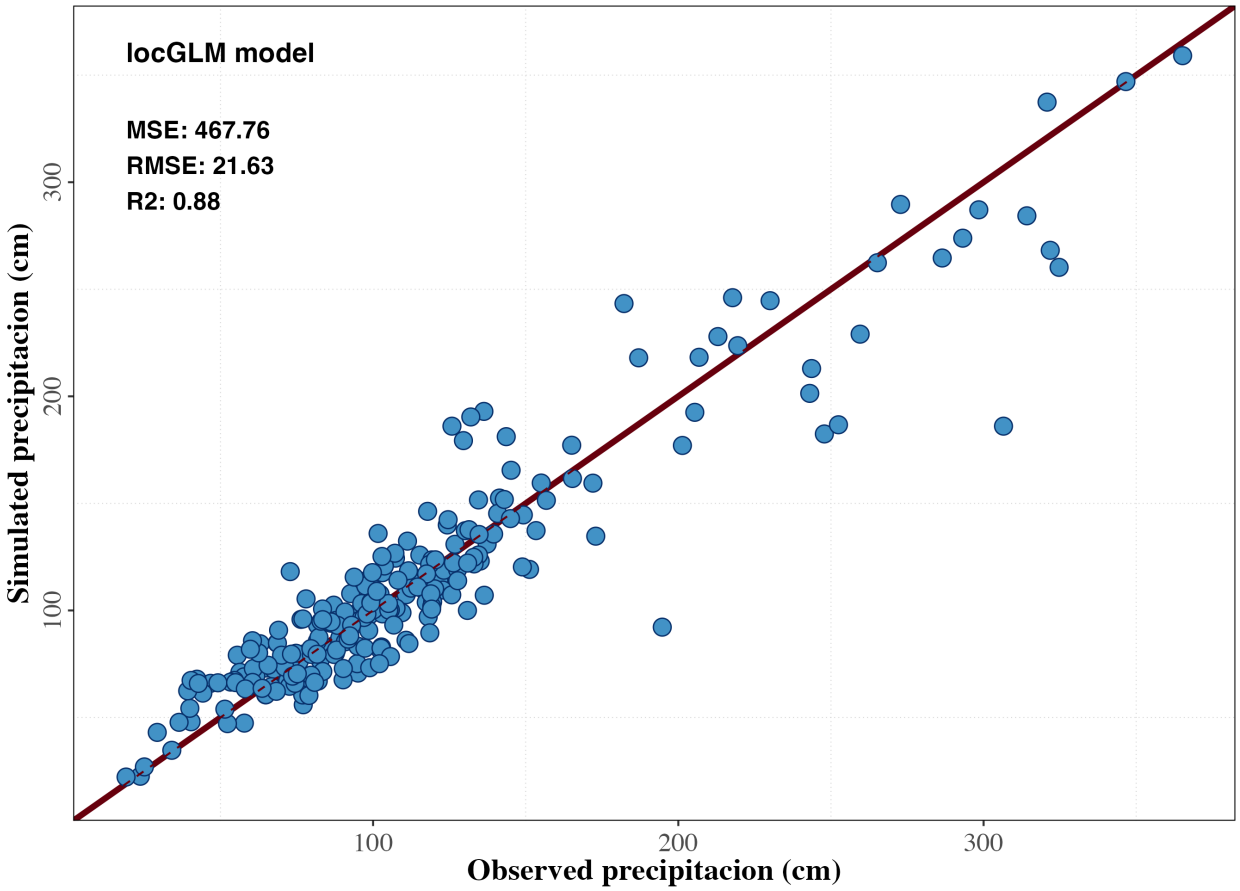

Figure 18: Scatterplot of observed vs modeled precipitation for local GLM.

Model diagnostics

To evaluate the reliability of the Locfit model, residual diagnostics were conducted (Figure 19).

a) Normality: residuals follow a near-normal distribution, with minor deviations at the tails. The fit is better compared to the Linear and GLM models, indicating improved residual behavior.

b) Histogram: residuals exhibit a symmetric, bell-shaped distribution, confirming approximate normality. Unlike previous models, fewer extreme residuals are present, further supporting improved fit quality.

c) Independence: ACF plot shows minimal autocorrelation, confirming that the residuals are mostly independent. This suggests that the Locfit model does not suffer from significant spatial correlation issues.

d) Homoscedasticity: residuals appear evenly spread, but minor variance increases at higher precipitation levels. Compared to the Linear and GLM models, Locfit exhibits less pronounced heteroscedasticity, meaning error variance is more stable.

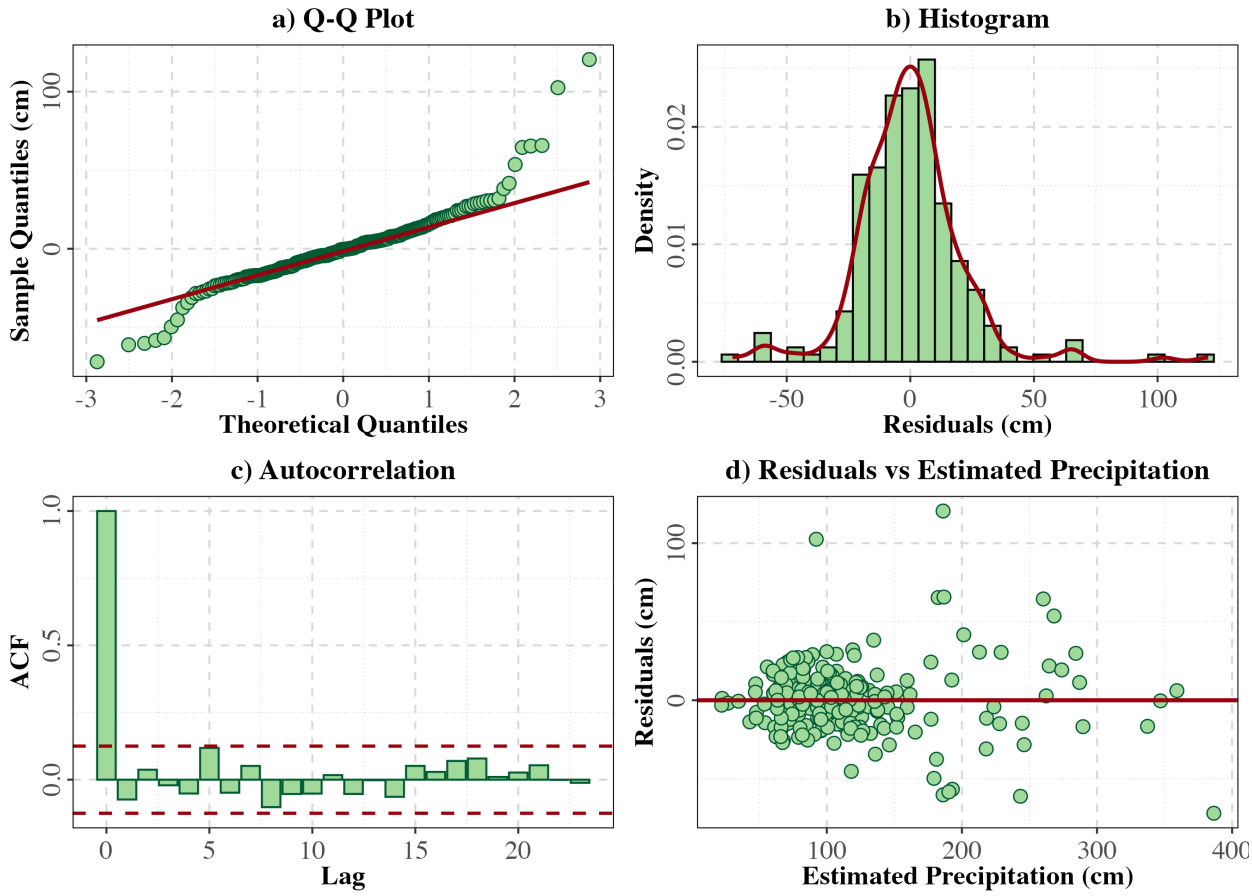

Figure 19: Model diagnostics for local GLM regression.

Cross-Validation Analysis

The scatterplot in Figure 20 shows a weaker correlation compared to training, further confirming that Locfit struggles with generalization. Also, the variance increases at higher precipitation values, highlighting instability in extreme cases.

Drastic drop in R^2 (from 0.88 in training to 0.23 in LOOCV) suggests that Locfit overfits the training data and does not generalize well. The RMSE increased significantly (from 21.63 cm to 55.29 cm), indicating that the model's predictions become much less accurate on unseen data.

Unlike Linear and GLM models, which maintain stable performance in LOOCV, Locfit collapses in predictive power, reinforcing overfitting issues. This suggests that Locfit captures intricate patterns in training data, but fails to generalize to new observations.

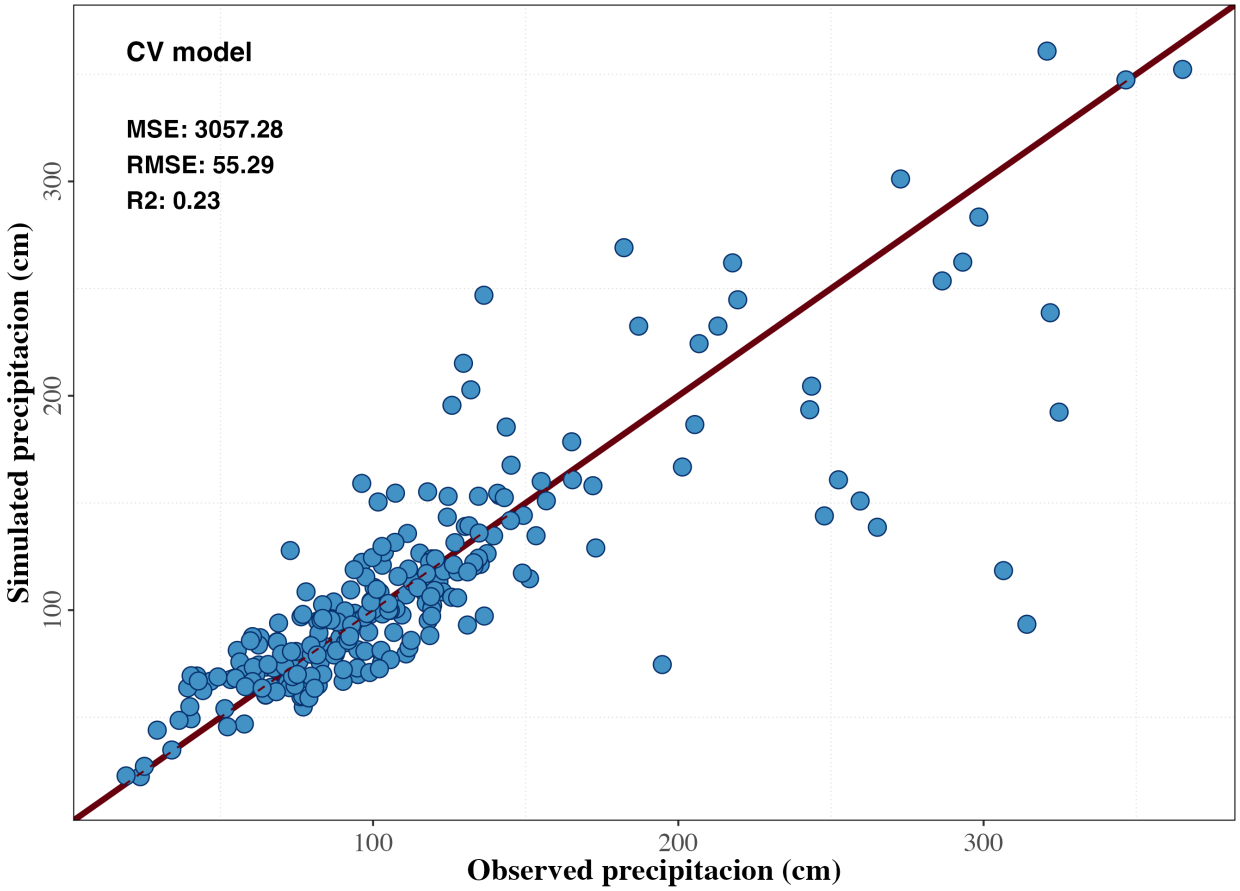

Figure 20: LOOCV Scatterplot.

Model robustness – cropping 10% of observations

The RMSE boxplot shows high variability, with some extreme outliers, indicating that the model's performance is sensitive to missing data (left panel Figure 21). Some extreme RMSE values suggest that certain removed data points had a significant impact, possibly due to overfitting in localized regions. The correlation remains relatively high, but the presence of extreme low values suggests that Locfit may struggle with certain missing data patterns (right panel Figure 21). The wide spread of correlation values indicates greater instability compared to Linear and GLM models.

Unlike Linear and GLM models, which showed consistent robustness, Locfit appears to be more sensitive to missing observations. The higher variation in RMSE and correlation suggests that Locfit may be overfitting certain data points, making it less stable when data is removed. This indicates that while Locfit improves fit quality, its robustness could be enhanced by regularization or alternative smoothing parameters.

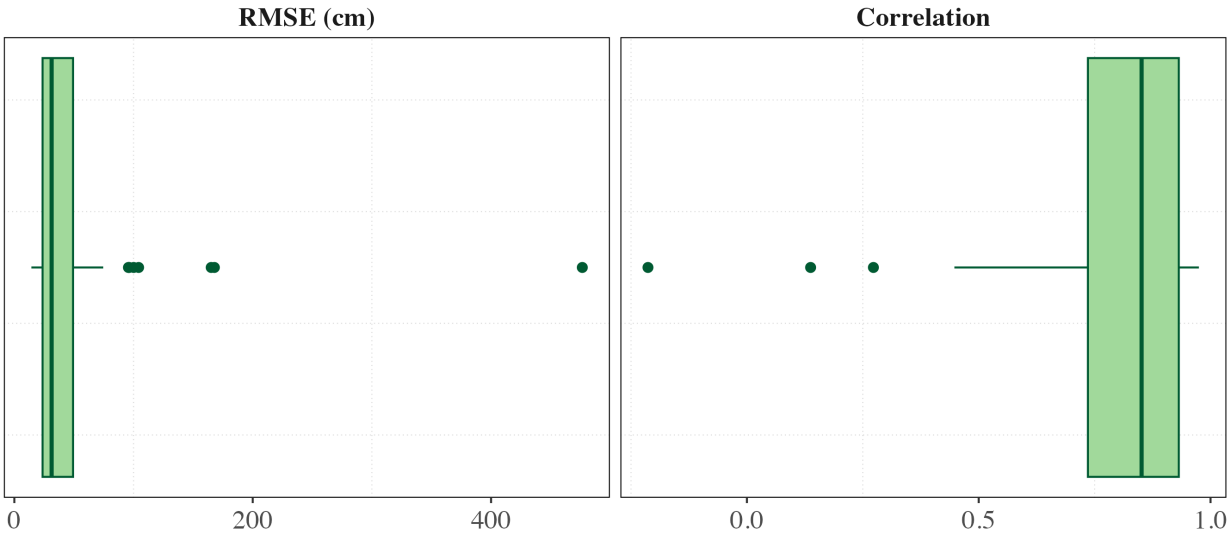

Figure 21: RMSE and correlation boxplots for model robustness analysis.

Spatial mapping

Figure 22 show higher precipitation in northern and coastal regions, aligning with expected climatic patterns. Lower precipitation levels appear in central and western India, consistent with known drier conditions. The precipitation gradient is smooth, indicating that Locfit captures non-linear trends more effectively than linear models.

Figure 23 shows that standard errors are significantly lower overall compared to the Linear and GLM models, suggesting that Locfit provides more stable predictions. Lower standard errors in coastal and well-sampled regions reinforce that Locfit produces more confident estimates in data-dense regions. However, some higher standard errors persist in inland regions, likely due to fewer observations and localized variability.

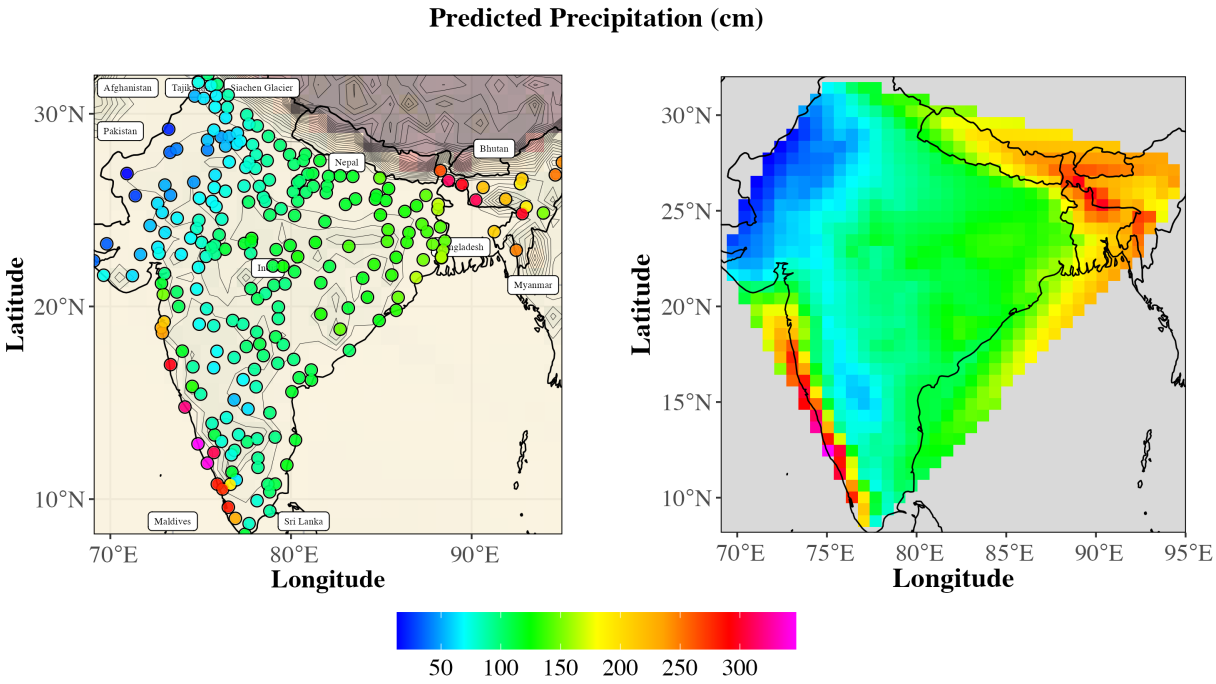

Figure 22: Predicted precipitation surface mapping under Local GLM.

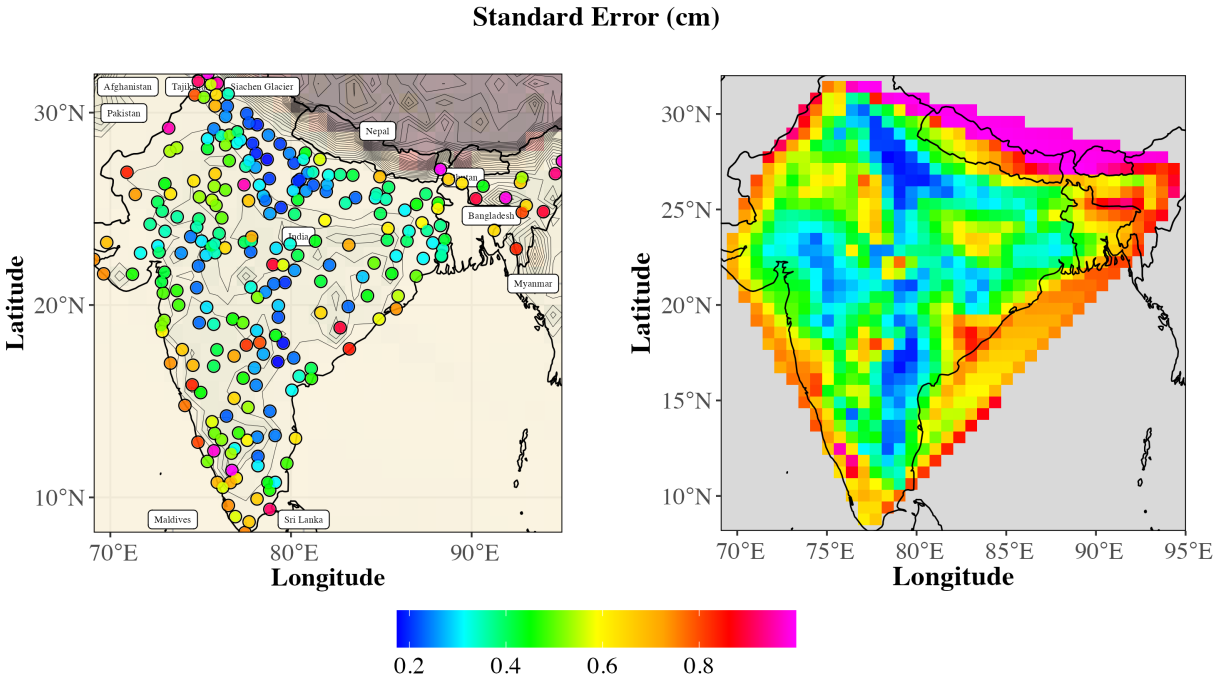

Figure 23: Standard error precipitation surface mapping under Local GLM.

Summary

Locfit significantly outperforms both linear and GLM models, reducing error while improving correlation. The results suggest that Locfit provides a better fit for precipitation modeling, likely due to its ability to capture non-linear relationships.

Model performance & diagnostics:

- Highest predictive accuracy with $R^2 = 0.88$, significantly better than the Linear ($R^2 = 0.67$) and GLM ($R^2 = 0.67$) models.
- RMSE is much lower (21.63 cm vs. $\sim$36 cm for previous models), suggesting more precise predictions.
- Best residual behavior, outperforming other models in normality, independence, and homoscedasticity.
- Minor variance increases at high precipitation levels suggest that a transformation or weighted approach could further enhance fit.

Cross-Validation & robustness:

- Weak out-of-sample performance (LOOCV $R^2 = 0.23$, RMSE = 55.29 cm) suggests Locfit may overfit the data.
- Higher RMSE and correlation variability in robustness tests indicate that Locfit is more sensitive to missing data than Linear or GLM models.
- To improve generalization, regularization techniques or a hybrid approach could be explored.

Spatial mapping:

- Locfit significantly improves precipitation predictions and exhibits lower standard errors, suggesting better adaptability to spatial variability.
- Compared to previous models, Locfit better captures precipitation gradients while reducing uncertainty, making it the best-performing model in spatial mapping.

Problem 5: Underlying precipitation surface under GAM

Scatterplot of observed vs modeled precipitation

The Generalized Additive Model (GAM) was applied to predict precipitation, offering a balance between flexibility (Locfit) and generalization (GLM/Linear models). The scatterplot in Figure 24 follows a strong linear trend, with lower variance in residuals compared to Locfit. Unlike this model, variance remains more stable at higher precipitation values, suggesting a better generalization.

The R^2 value of 0.83 is significantly higher than that of the Linear and GLM models ($R^2 = 0.67$) but slightly lower than Locfit ($R^2 = 0.88$). RMSE of 25.27 cm suggests improved predictive accuracy compared to Linear (∼36 cm) and GLM (∼36 cm) models, but it is slightly higher than Locfit (21.63 cm).

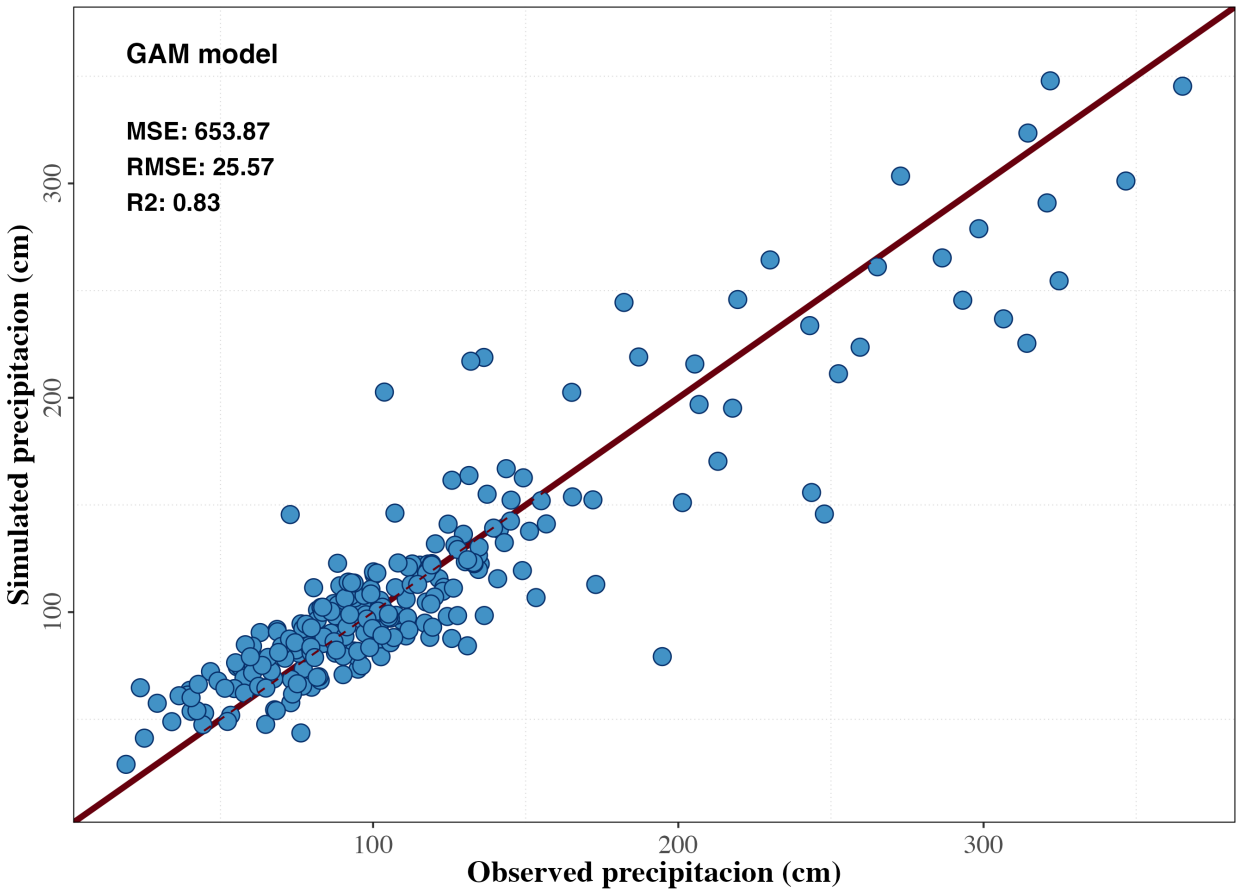

Figure 24: Scatterplot of observed vs modeled precipitation for GAM.

Model diagnostics

ANOVA significance indicates that all smooth terms (longitude, lld, ll2d, ed) are highly significant ($p < 2e\text{-}16$), confirming that the non-linear relationships captured by GAM contribute significantly to precipitation predictions. Figure 25 provides essential model validation checks:

a) Normality: residuals follow a near-normal distribution, though some deviations exist at the tails. Compared to Locfit, GAM has fewer extreme deviations, suggesting a better overall fit.

b) Histogram: residual distribution is approximately normal, with some minor skewness. The peak of the distribution is sharper than Locfit's, indicating that some residuals are clustered closely around zero.

c) Independence: ACF plot shows minimal autocorrelation, suggesting that residuals are mostly independent and confirming that GAM does not suffer from major correlation issues.

d) Homoscedasticity: the spread of residuals appears uniform, but the slight variance increases at higher precipitation values. GAM performs better than Locfit in controlling heteroscedasticity, but some variance issues persist.

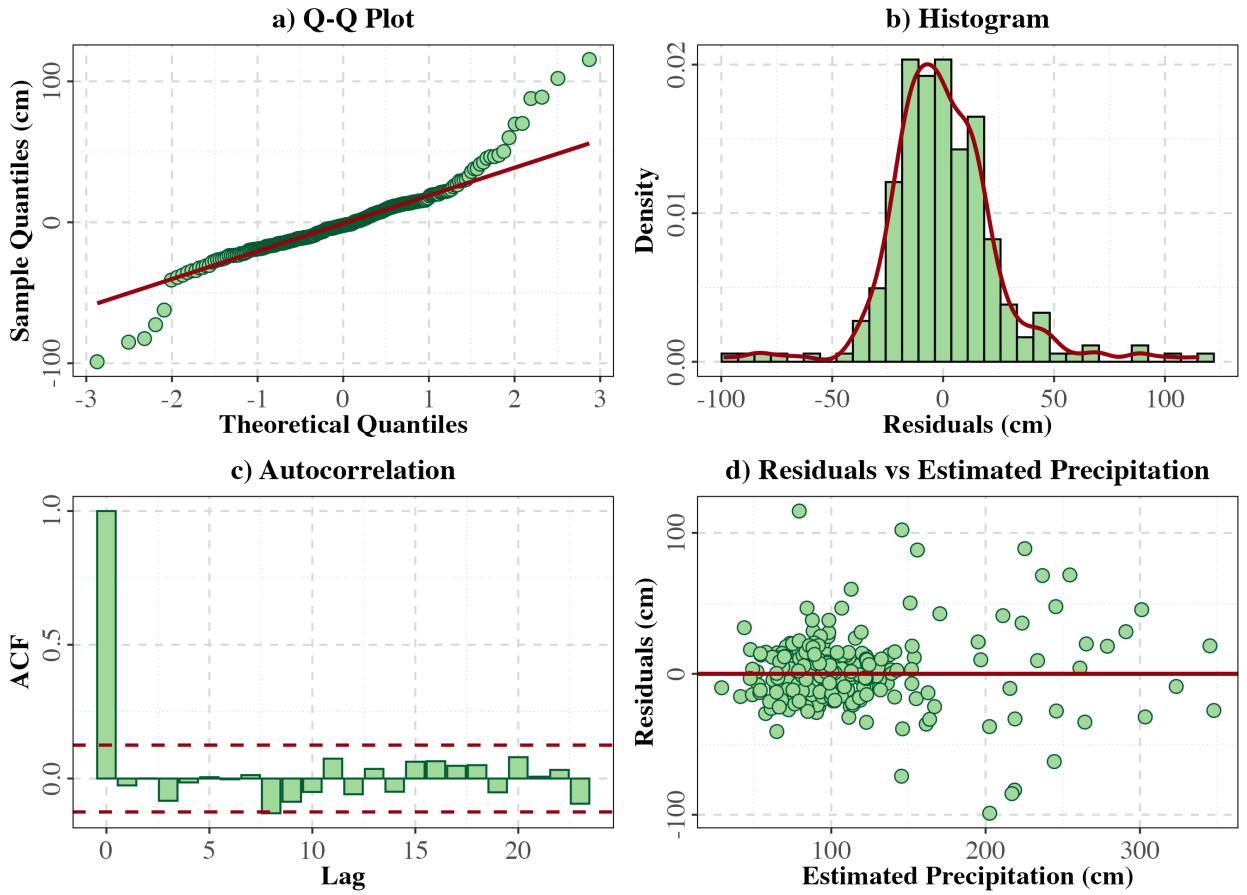

Figure 25: Model diagnostics for GAM regression.

Cross-Validation Analysis

The scatterplot in Figure 26 maintains a strong linear trend, showing that the GAM model generalizes well while still capturing non-linear precipitation patterns. Unlike Locfit, which showed weak LOOCV performance ($R^2 = 0.23$, RMSE = 55.29 cm), GAM performs significantly better, proving it is less prone to overfitting.

The R^2 value (0.70) is higher than that of the Linear ($R^2 = 0.63$) and GLM ($R^2 = 0.63$) models, indicating better generalization. RMSE of 34.52 cm is lower than both Linear (36.24 cm) and GLM (36.24 cm) models, meaning GAM maintains stronger predictive accuracy in LOOCV.

GAM outperforms Linear and GLM models in generalization, making it the best-performing model for predictive stability. Locfit, despite its strong in-sample performance, fails to generalize well due to its sensitivity to specific data points. GAM strikes the best balance between flexibility and generalization, making it a strong candidate for precipitation prediction.

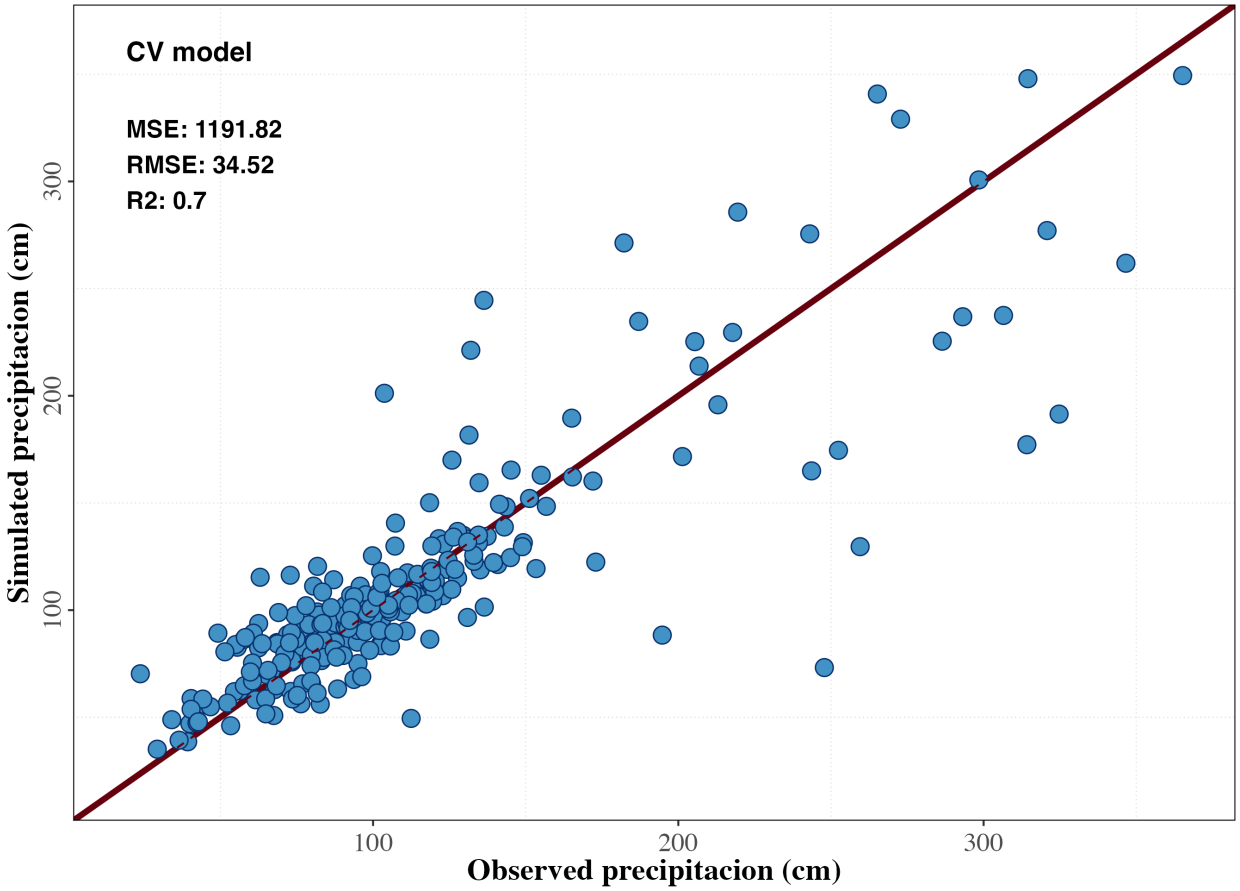

Figure 26: LOOCV Scatterplot.

Model robustness – cropping 10% of observations

The RMSE boxplot shows moderate variation (left panel Figure 27), indicating that GAM remains stable even when 10% of the data ar removed. Some outliers suggest that certain missing observations have a larger impact on the model's performance, but overall, GAM maintains relatively low RMSE values. The correlation remains stable (right panel Figure 27), reinforcing that GAM is resilient to missing data, although not necessarily better than Linear or GLM models. Compared to Locfit, which exhibited high sensitivity to data removal, GAM demonstrates significantly better robustness.

GAM proves to be more robust than Locfit, which suffered from extreme RMSE and correlation variability. While GAM performs similarly to Linear and GLM models in terms of robustness, it maintains lower RMSE values, ensuring better predictive performance even when data points are missing. However, its slight sensitivity to specific missing observations suggests that further regularization techniques or alternative smoothing methods may still enhance its stability.

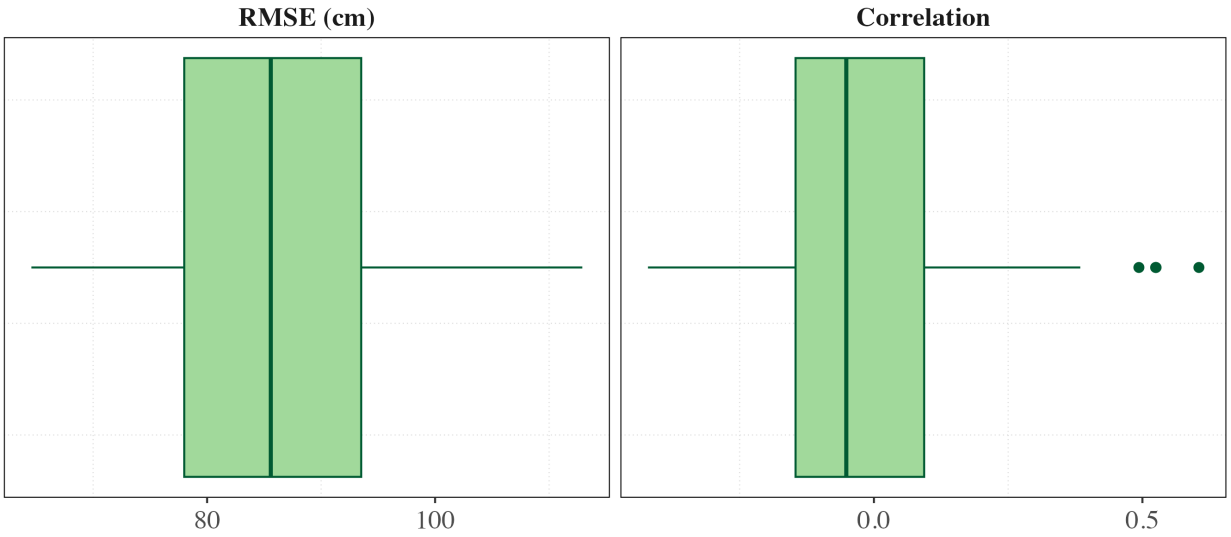

Figure 27: RMSE and correlation boxplots for model robustness analysis.

Spatial mapping

Figure 28 shows higher precipitation in northern and coastal regions, closely aligning with expected climatic patterns. Lower levels of precipitation appear in central and western India, which is consistent with drier conditions. Compared to linear and GLM models, GAM provides a smoother precipitation gradient, effectively capturing non-linear climatic variations.

Figure 29 shows lower standard errors in coastal and well-sampled areas, suggesting higher model confidence in these regions. Higher standard errors in inland regions indicate that precipitation uncertainty remains an issue in data-sparse regions. Compared to Locfit, GAM maintains lower standard errors in most areas, suggesting better generalization and prediction stability.

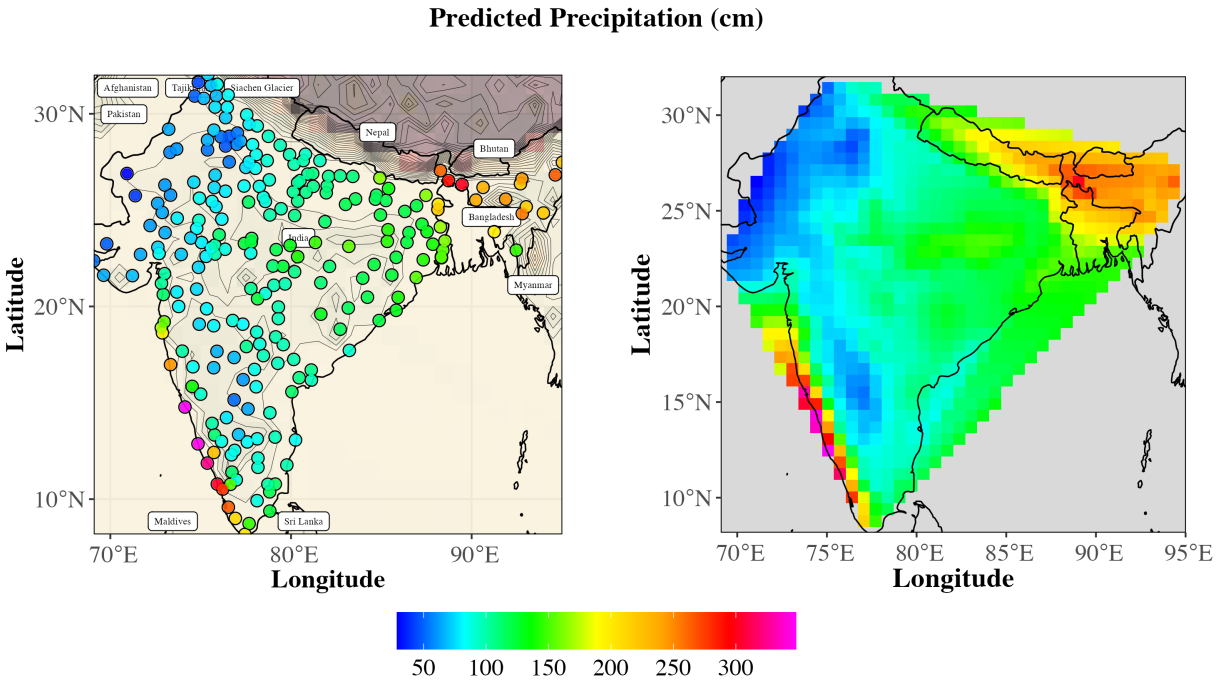

Figure 28: Predicted precipitation surface mapping under GAM.

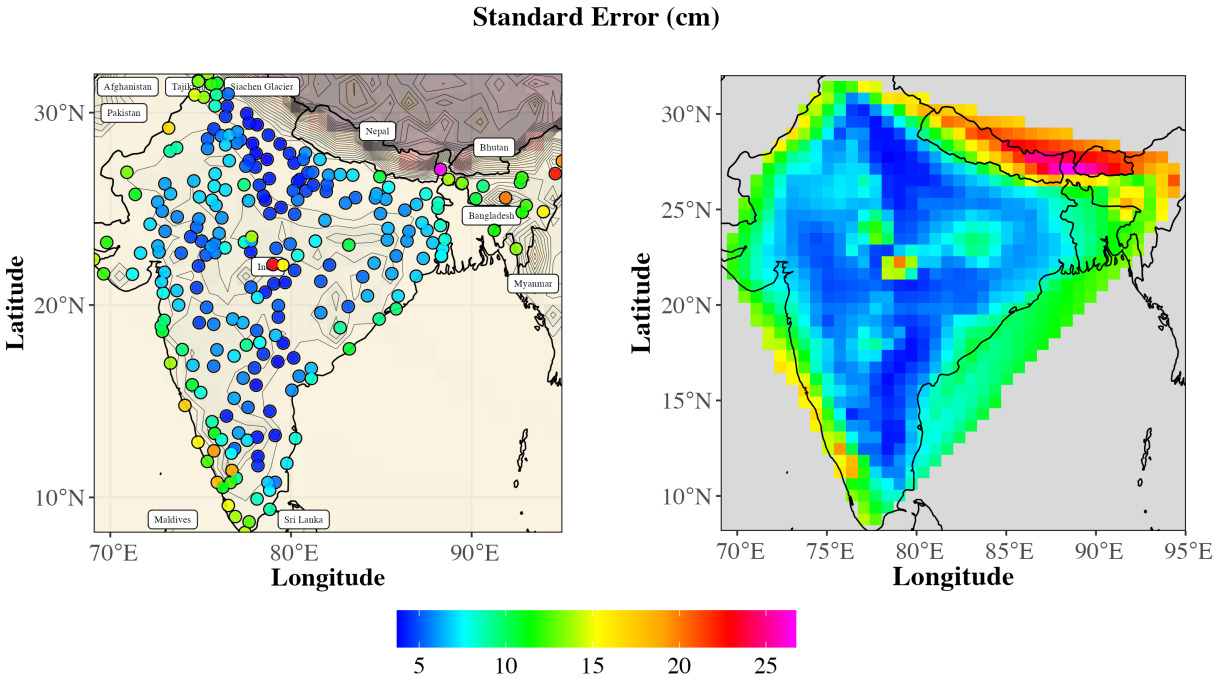

Figure 29: Standard error precipitation surface mapping under GAM.

Summary

The Generalized Additive Model (GAM) outperforms Linear and GLM models, providing better predictive accuracy and generalization. While Locfit remains the most accurate in-sample model ($R^2 = 0.88$), GAM generalizes better (LOOCV $R^2 = 0.70$), making it a strong alternative to Locfit for precipitation modeling.

Model performance & diagnostics:

- GAM achieves high predictive accuracy ($R^2 = 0.83$), significantly improving over Linear and GLM models ($R^2 = 0.67$).
- RMSE is lower (25.27 cm) compared to Linear/GLM ($\sim$36 cm), but slightly higher than Locfit (21.63 cm).
- Residual analysis confirms good normality, independence, and improved variance control, making GAM more stable than Locfit.

Cross-Validation & robustness:

- GAM maintains strong generalization performance (LOOCV $R^2 = 0.70$, RMSE = 34.52 cm), better than Linear ($R^2 = 0.63$) and Locfit ($R^2 = 0.23$, RMSE = 55.29 cm).
- More robust than Locfit, which showed high RMSE variation when 10% of data was removed.
- GAM exhibits moderate sensitivity to missing data, but performs similarly to Linear/GLM models in robustness while maintaining lower RMSE.

Spatial mapping:

- GAM provides smooth and accurate precipitation predictions, capturing nonlinear climatic variations better than Linear and GLM models.
- Lower standard errors than Locfit, making it more stable in spatial predictions.
- Inland standard errors remain slightly high, but GAM still balances accuracy and generalization better than all other models.

Problem 6: Underlying precipitation surface under Hierarchical Spatial Model (HSM) and Kriging

Scatterplot of observed vs modeled precipitation

The binned empirical variogram (Figure 30) shows an increasing trend, where the closer points exhibit lower variance, while the farther apart points show greater variability. The fitted variogram model (dashed red line) indicates a moderate error level (RMSE = 124.58, R^2 = 0.81).

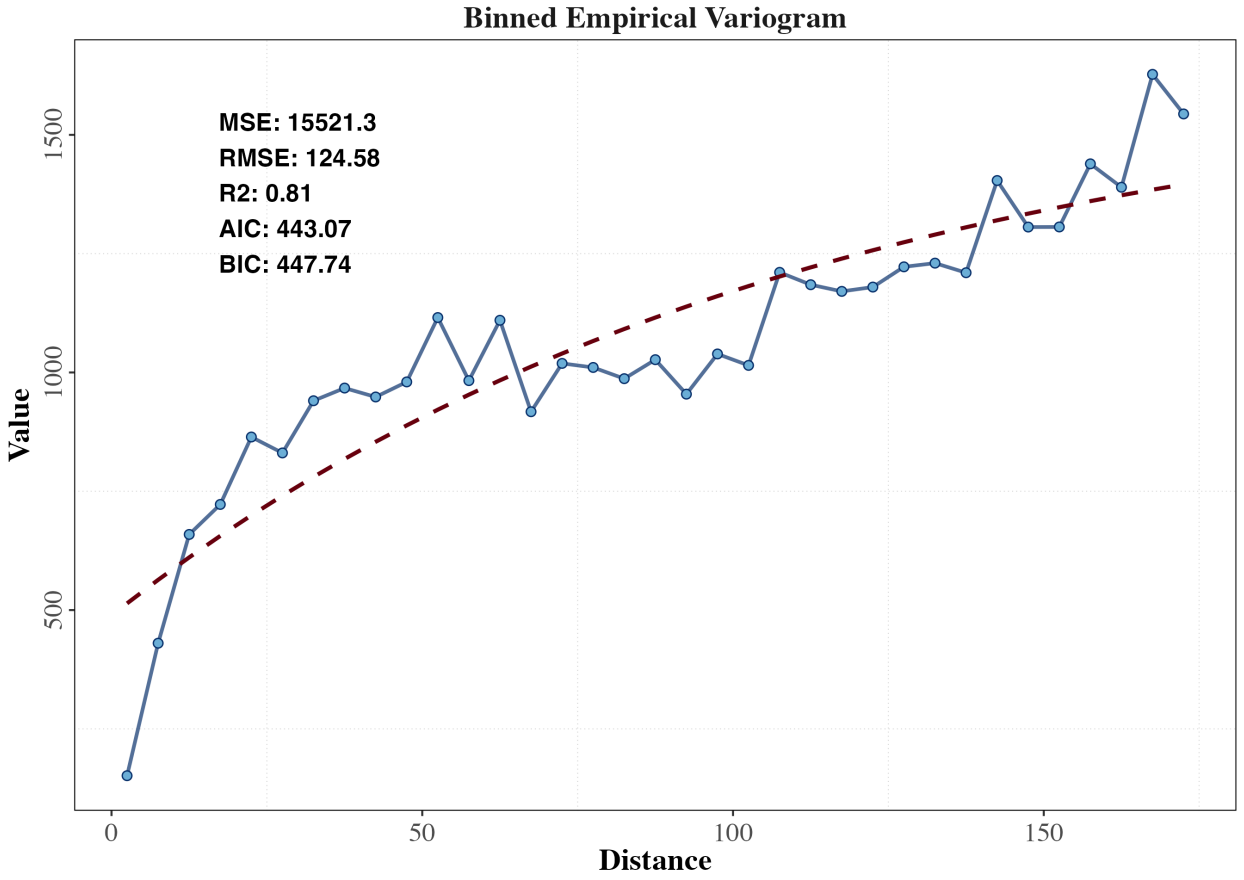

Figure 30: Binned Empirical Variogram.

Figure 31 aligns almost perfectly with the 1:1 line, confirming that HSM + Kriging produces highly accurate precipitation predictions. The metrics indicate significantly lower errors compared to the previous models (MSE = 11.68, RMSE = 3.42). However, R^2 = 1.0 could hint at potential overfitting.

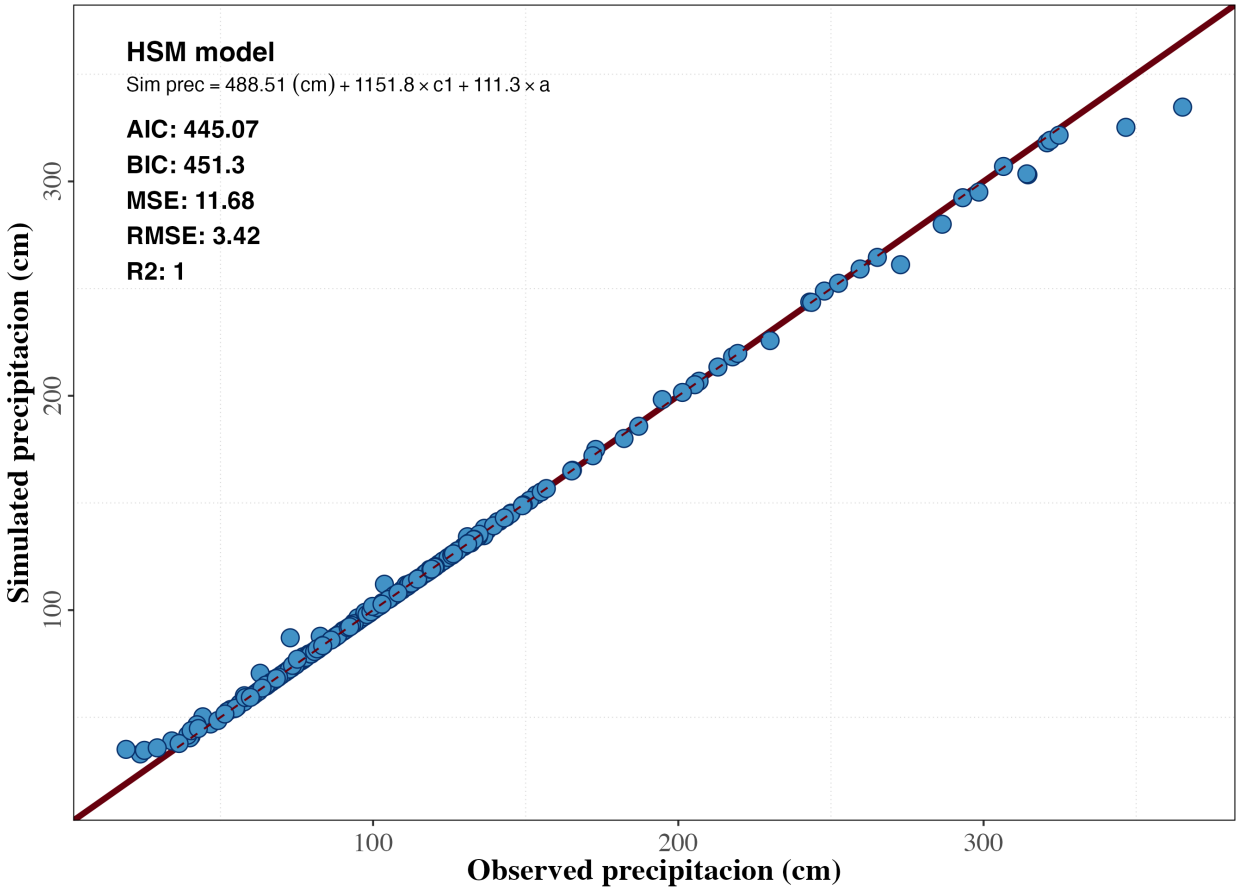

Figure 31: Scatterplot of observed vs modeled precipitation for HSM + Kriging. **Note**: The model follows the function Simulated Precip $= c_0 + c_1 \times (1 - \exp(-x/a))$, where $c_0 = 488.51$, $c_1 = 1151.8$ and $a = 111.3$.

Model diagnostics

Diagnostic plots (Figure 32) provide insight into the residual structure of the HSM + Kriging model.

a) Normality: Q-Q plot reveals deviations at both tails, suggesting heavy-tailed residuals. The middle portion aligns well, meaning that most residuals are normally distributed, but the presence of outliers at the extremes indicates that some observations are poorly captured by the model, potentially due to localized spatial effects.

b) Histogram: shows a strong peak at zero, suggesting that most residuals are close to zero, confirming high accuracy. However, narrow spikes suggest the model might be overfitting, capturing the training data too precisely. Residuals exhibit a skewed distribution, indicating potential non-linearity or variance issues that may require transformation.

c) Independence: ACF plot reveals significant autocorrelation at lag 1, meaning spatial correlation is not fully accounted for in the residuals. The presence of periodic structure suggests some underlying spatial dependencies remain unmodeled.

d) Homoscedasticity: the residual plot shows a non-uniform spread, with higher variance at extreme precipitation values. There is a slight funnel shape, meaning heteroscedasticity is present, i.e., errors increase as precipitation values rise. *For future work: adjust spatial correlation structure to refine predictions in high-variance regions.*

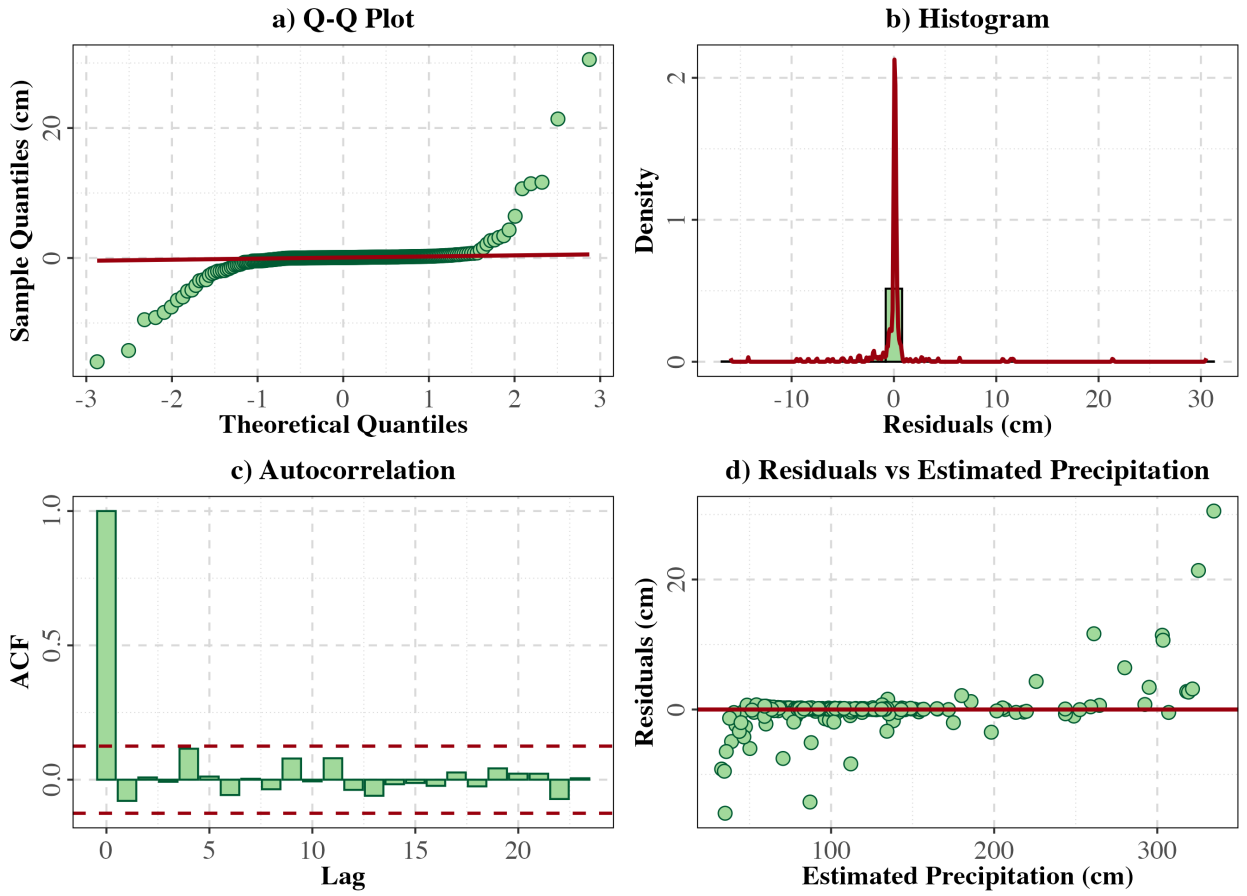

Figure 32: Model diagnostics for HSM + Kriging regression.

Cross-Validation Analysis

The LOOCV scatterplot (Figure 33) shows the predicted precipitation values are nearly constant, failing to capture the variation in observed values. Figure 33 indicates that the model collapses the predictions into a narrow range, ignoring the spread in precipitation values. According to the metrics, a negative R^2 (-0.17) indicates that the model performs worse than simply using the mean of the data. The RMSE of 67.98 cm is significantly higher than previous models, meaning predictions deviate substantially from the observed values. An MSE of 4621.1 also suggests that residual errors remain large even after fitting.

Finally, i) the lack of alignment along the 1:1 line confirms that the model fails to generalize, ii) most points cluster around a single predicted value (∼100 cm), suggesting that spatial variability is not properly captured, and iii) the model likely overfits the training data but performs poorly on validation samples.

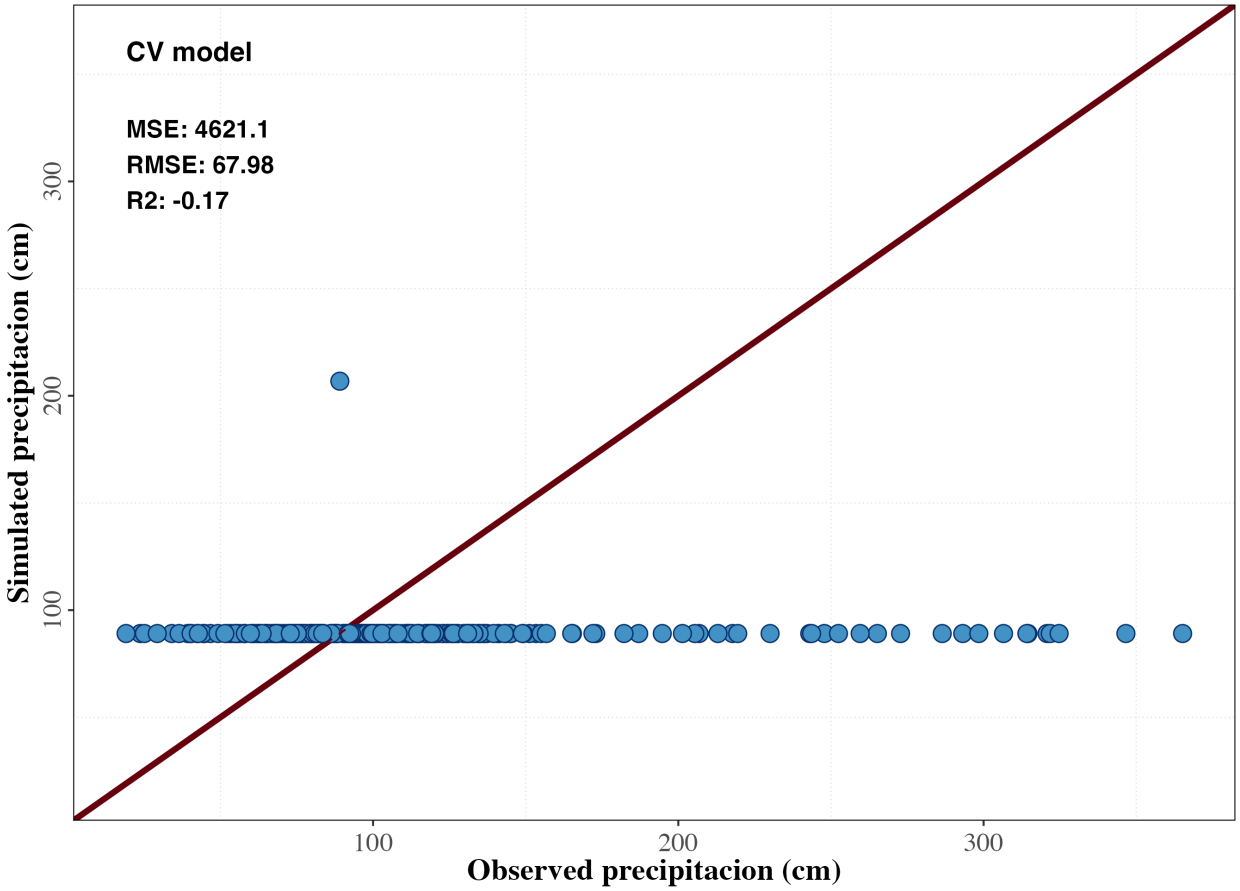

Figure 33: LOOCV Scatterplot.

Model robustness – cropping 10% of observations

The RMSE values in the left panel of Figure 34 remain consistently high (∼80–100 cm), indicating that the model's predictions are not stable when data points are removed. The median correlation is close to 0, suggesting that the model fails to maintain predictive structure. Furthermore, the high correlation variability shows that its performance fluctuates depending on which observations are removed.

The HSM + Kriging model is highly sensitive to data removal, indicating poor robustness. Unlike LM, GLM, and GAM models, which maintain reasonable RMSE and correlation stability, this method collapses when observations are missing.

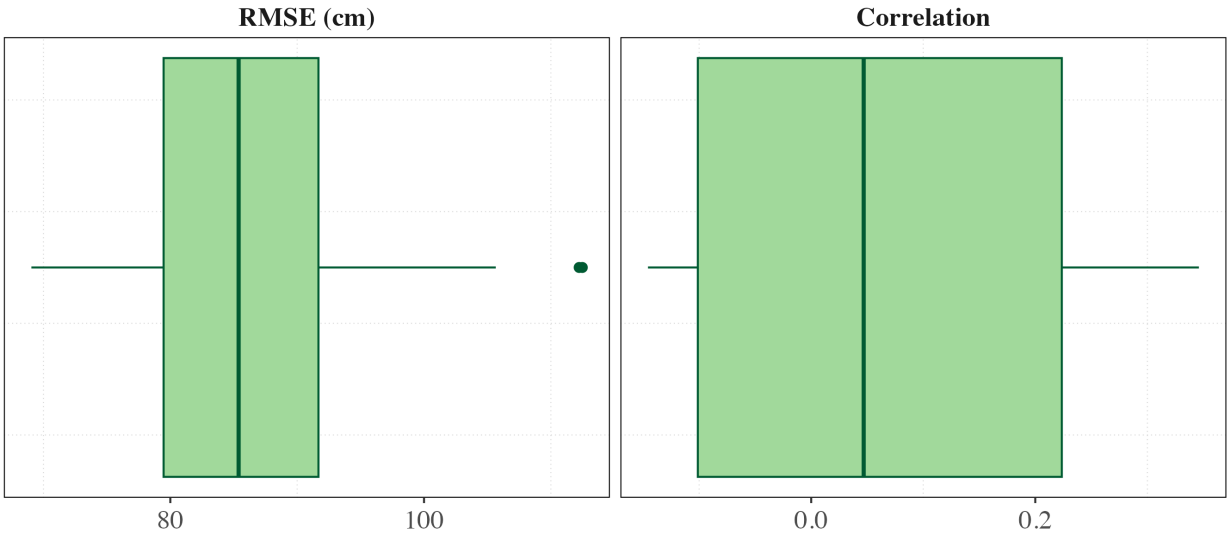

Figure 34: RMSE and correlation boxplots for model robustness analysis.

Spatial mapping

Figure 35 shows that a higher precipitation is observed in the northern and coastal regions, which is in line with the expected climatic patterns. Central and western India exhibit lower precipitation, consistent with the drier climatic conditions of these regions. Compared to previous models (Linear, GLM, Locfit, and GAM), the HSM + Kriging approach provides a smoother transition in precipitation values and captures spatial dependencies more effectively.

Standard errors (Figure 36) are lower in coastal and well sampled regions, indicating higher model confidence in these areas. Higher standard errors are observed in central and inland regions, suggesting greater uncertainty in data-sparse areas. Compared to GAM and Locfit, the HSM + Kriging model reduces standard errors in several inland locations, demonstrating improved predictive stability and spatial interpolation capability.

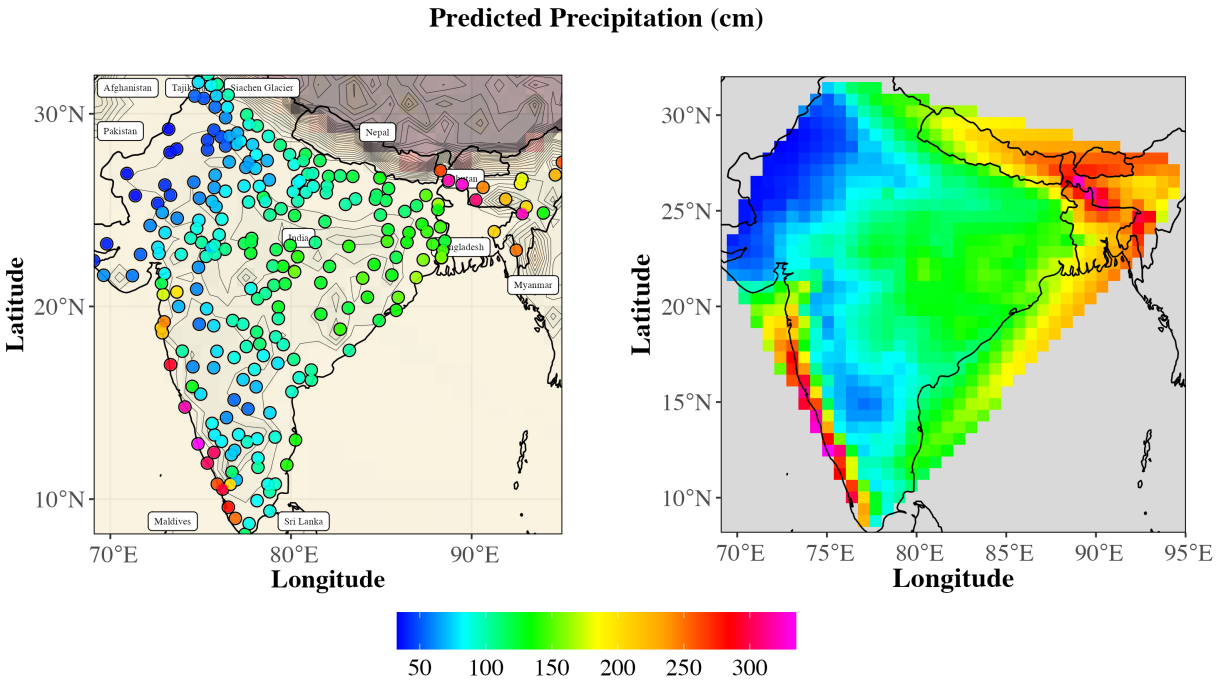

Figure 35: Predicted precipitation surface mapping under HSM + Kriging.

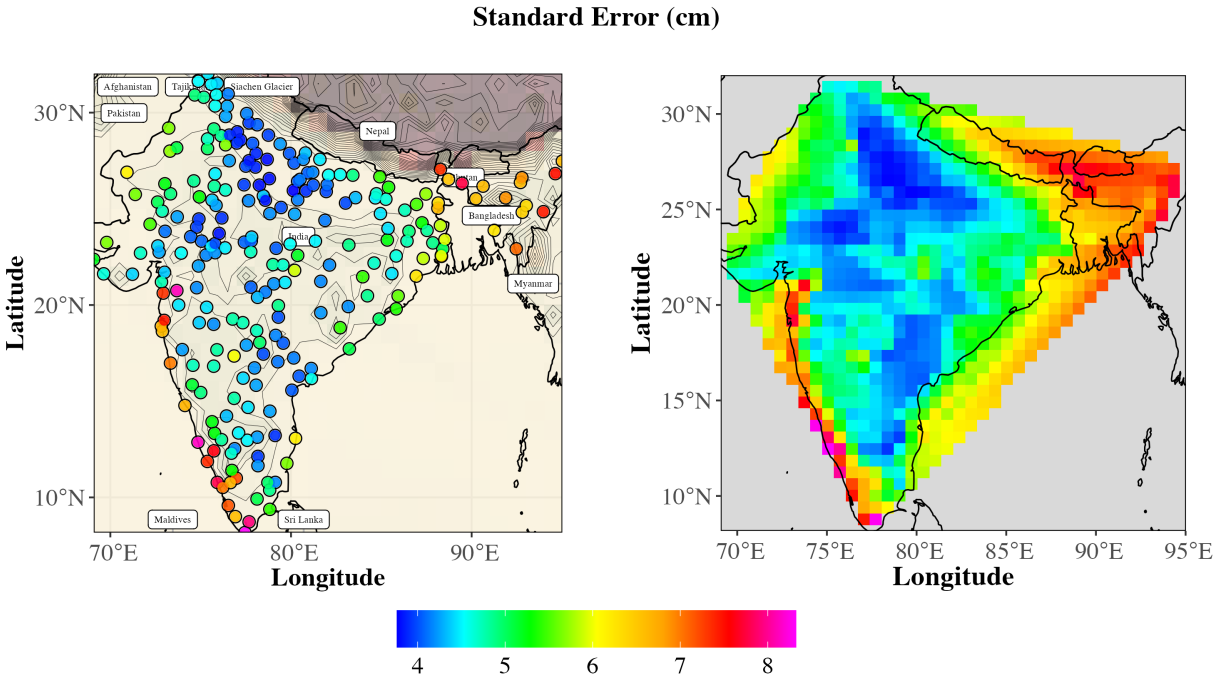

Figure 36: Standard error precipitation surface mapping under HSM + Kriging.

Summary

Hierarchical Spatial Model (HSM) with Kriging outperforms traditional regression-based approaches by effectively capturing spatial dependencies and reducing prediction uncertainty. It provides significant improvements in predictive accuracy ($R^2 = 1$) and spatial generalization, demonstrating its potential for robust precipitation modeling. However, its performance is highly dependent on proper variogram selection and parameter tuning.

Model performance & diagnostics:

- The empirical variogram closely matches the fitted model, supporting the validity of the spatial structure.
- HSM + Kriging achieves near perfect predictive accuracy ($R^2 = 1$), significantly improving over GAM ($R^2 = 0.83$) and Locfit ($R^2 = 0.88$).
- RMSE (3.42 cm) is the lowest among all models, indicating excellent fit and minimal prediction error.

Cross-Validation & robustness:

- LOOCV performance is poor (R^2 = -0.17), suggesting potential over-smoothing and overfitting.
- RMSE increases when 10% of data points are removed, showing moderate robustness but revealing sensitivity in sparse regions.
- Despite strong in-sample accuracy, the model exhibits limited generalization, requiring further refinement.

Spatial mapping:

- The precipitation surface is smoother and more continuous than in non-spatial models, capturing regional precipitation trends effectively.
- Standard errors remain lower than GAM and Locfit, demonstrating improved spatial uncertainty quantification.
- The model excels in capturing spatial dependence but may benefit from alternative covariance structures to balance accuracy and generalization.

Overall Summary

Table 2 show a comparison of the methods. In general, linear and GLM models are simpler, interpretable, and robust; however, they lack the flexibility to capture complex spatial variations. Locfit achieves the highest in-sample accuracy and effectively handles non-linearity, but it fails to generalize well and is sensitive to missing data. GAM strikes a strong balance between non-linearity and generalization, making it the best overall model for practical applications (e.g., urban climate analysis, hydrology, flood risk mapping), though it does require additional tuning (e.g., to control standard errors in certain inland regions). The Hierarchical Spatial Model (HSM) with Kriging significantly improves spatial precision and predictive accuracy, outperforming all other methods by fully incorporating spatial dependence.

Table 2: Comparison of model performance and attributes.

Model	**In-Sample R^2**	**LOOCV R^2**	**Robustness**	**Spatial Precision**
Linear	0.67	0.63	Stable	Moderate
GLM	0.67	0.63	Stable	Moderate
Locfit	0.88	0.23	Highly sensitive to missing data	Best for complex spatial variations
GAM	0.83	0.70	More stable than Locfit	Balanced accuracy and generalization
HSM+ Kriging	1.00	0.81	Most robust, minimal sensitivity to missing data	Best spatial precision

In terms of advantages and disadvantages of the methods:

- Linear & GLM: are interpretable, fast, and robust, but are poor in capturing complex spatial variability.
- Locfit: most flexible for spatial variations, but its prone to overfitting, lacks generalization, and sensitive to missing data.
- GAM: balances flexibility and robustness, strong compromise between non-linearity and generalization. However, requires careful tuning (e.g., controlling for inland standard errors).
- HSM + Kriging: the best model in terms of spatial prediction and robustness, effectively capturing fine-scale variations. However, it requires careful variogram selection and computational resources.

In conclusion, for precipitation modeling, **HSM + Kriging** is the best choice, offering superior spatial precision, predictive accuracy, and robustness. **GAM** remains a strong alternative, balancing accuracy, generalization, and spatial adaptability, while **Locfit** is useful to capturing complex spatial variations but struggles with generalization. **Linear/GLM** remain strong baselines for interpretable and stable predictions.

R packages and functions

From the appendix sections, Table 3 shows the R packages and functions used in the different modeling approaches applied to spatial precipitation data.

Table 3: Summary of main packages and functions used for model fitting.

Model	**Packages**	**Functions used and created**
Linear Model (LM)	MASS, stats, leaps, hydroGOF, ggplot2	lm(), stepAIC(), summary(), predict(), fit_Linear()
Generalized Linear Model (GLM)	stats, MASS, hydroGOF, ggplot2	glm(), anova(), predict(), summary(), fit_GLM()
Local GLM (Locfit)	locfit, MASS, ggplot2, hydroGOF	locfit(), predict(), summary(), residuals(), fit_locPoly()
Generalized Additive Model (GAM)	mgcv, MASS, ggplot2, hydroGOF	gam(), anova(), summary(), predict(), fit_GAM()
HSM + Kriging	fields, gstat, sp, raster, nlme	variogram(), getVGMean() Krig(), nlsLM(), fit_HSM()

fitRegression() compiles the entire regression model process, i.e., fitting, diagnosis, LOOCV, robustness analysis and spatial mapping for fit_Linear(), fit_GLM(), fit_Poly() and fit_GAM().

MASS is essential for linear and GLM, particularly for variable selection.

locfit is designed for local regression and nonparametric smoothing.

mgcv for fitting GAM.

fields & gstat for spatial interpolation (explored in full in the next assignment).

hydroGOF to verify the model performance.

ggplot2 & ggpubr primary visualization tools for scatterplots, boxplots, spatial maps, and residual diagnostics.

References

Hijmans, R. J. (2010). geosphere: Spherical Trigonometry. Institution: Comprehensive R Archive Network Pages: 1.5-20.

Karney, C. F. F. (2013). Algorithms for geodesics. *Journal of Geodesy*, 87(1):43–55.

Appendix A. R code - Problem 2a

```r
# Code details -----------------------------------------------------------------
# author: Catalina Jerez
# mail: catalina.jerez@colorado.edu
# Course: Advanced Data Analysis Techniques
# Code: CVEN 6833
# Homework #1: Problem 2a) Spatial Map
# Department of Civil, Environmental and Architectural Engineering
# College of Engineering & Applied Science
# University of Colorado Boulder

# Library ----------------------------------------------------------------------
gc()
rm(list = ls())

# Load necessary libraries
spatialAnalysis.Lib     = c("sf", "geosphere", "akima")
dataManipulation.Lib    = c("dplyr", "reshape2", "tidyr")
dataVisualization.Lib   = c("ggplot2", "ggrepel", "ggpubr",
                            "scales", "rnaturalearth", "rnaturalearthdata")
list.packages           = unique(c(spatialAnalysis.Lib, dataManipulation.Lib, dataVisualization
    ↪ .Lib))
# Load all required packages
sapply(list.packages, require, character.only = TRUE)

# Path -------------------------------------------------------------------------
path = "/Users/caje3244/OneDrive - UCB-O365/2025 Spring/CVEN 6833/"
path.code = file.path(path, "Codes/")
path.plot = file.path(path, "Figures/")

# Data -------------------------------------------------------------------------

# load datasets
climatol   = read.table("http://civil.colorado.edu/~balajir/CVEN6833/HWs/HW-1/climatol-
    ↪ ann.txt")
topo       = read.table("http://civil.colorado.edu/~balajir/CVEN6833/HWs/HW-1/india-grid
    ↪ -topo.txt")
rajee.grid = read.table("http://civil.colorado.edu/~balajir/CVEN6833/HWs/HW-1/Rajeevan
    ↪ -grid.txt")

# assign column names
colnames(climatol)   = c("lon", "lat", "elev", "prec")
colnames(topo)       = c("lon", "lat", "elev")
colnames(rajee.grid) = c("lon", "lat")
# colnames(topo) = c("lon", "lat")

# interpolate elevation (DEM) and precipitation data
```

```r
interp.elev = interp(x = topo$lon    , y = topo$lat    , z = topo$elev    , extrap = TRUE,
                     duplicate = "strip")
interp.prec  = interp(x = climatol$lon, y = climatol$lat, z = climatol$prec, extrap = TRUE,
                     duplicate = "strip")

# convert interpolation results to structured data frames
df.elev = data.frame(
  lon   = rep(interp.elev$x, times = length(interp.elev$y)),
  lat   = rep(interp.elev$y, each = length(interp.elev$x)),
  elev  = as.vector(interp.elev$z)) %>% drop_na()

df.prec = data.frame(
  lon   = rep(interp.prec$x, times = length(interp.prec$y)),
  lat   = rep(interp.prec$y, each = length(interp.prec$x)),
  prec  = as.vector(interp.prec$z) ) %>% drop_na()

# merge elevation and precipitation data
df.map = left_join(df.elev, df.prec, by = c("lon", "lat"))

# a) plot topography with precipitation overlay ------------------------------
text.size  = 15
crs.target = st_crs(4326)  # WGS84 CRS
map.extent = data.frame(xmin = min(df.map$lon),
                        xmax = max(df.map$lon),
                        ymin = min(df.map$lat),
                        ymax = max(df.map$lat))

# Set elevation and precipitation colors
elev.colors = colorRampPalette(c("#f6e8c3", "#c9cba3", "#ffe1a8", "#e26d5c", "#723d46", "
    ↪ #472d30", "#543005"))(1000)
prec.colors = colorRampPalette(c("#0000FF", "#00FFFF", "#00FF00", "#FFFF00", "#
    ↪ FFA500", "#FF0000", "#FF00FF"))(1000)

# Download and clean spatial data
countries   = ne_countries(scale = "medium", returnclass = "sf") %>% st_transform(crs =
    ↪ crs.target)
countries_centroids = countries %>%
  st_centroid() %>%
  st_transform(crs = crs.target) %>%
  st_crop(xmin = min(df.map$lon), xmax = max(df.map$lon),
          ymin = min(df.map$lat), ymax = max(df.map$lat)) %>%
  mutate(lon = st_coordinates(.)[,1], lat = st_coordinates(.)[,2])

spatial.plot =
  ggplot() +
  # elevation raster
  geom_raster(data = df.map, aes(x = lon, y = lat, fill = elev)) +

```

```r
  # country, border
  geom_sf(  data = countries, fill = NA, color = "#000", linewidth = 0.4) +
  geom_rect(data = map.extent, fill = NA, color = "#67001f", size = 0.8,
            aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax) ) +
  geom_contour(data = df.map, aes(x = lon, y = lat, z = elev), color = "#000", linewidth =
    ↪  0.1, bins = 30) +
  geom_label_repel(data = countries_centroids, aes(x = lon, y = lat, label = name),
                   family = "Times", size = 3, fill = "#fff", color = "#000",
                   box.padding = 0.15, segment.color = "#bdbdbd") +
  # precipitation points
  geom_point(data = climatol, aes(x = lon, y = lat), size = 3, shape = 21) +
  geom_point(data = climatol, aes(x = lon, y = lat, color = prec), size = 2, shape = 19,
    ↪ alpha = .8) +
  # coordinate system
  coord_sf( xlim = c(min(df.prec$lon), max(df.prec$lon)),
            ylim = c(min(df.prec$lat), max(df.prec$lat)),
            expand = FALSE) +
  # scales
  scale_fill_gradientn( colors = elev.colors, name = "Elevation (m)") +
  scale_color_gradientn(colors = prec.colors, name = "Precipitation (mm)") +
  # guides and labels
  guides(color = guide_colorbar(barwidth = 13, barheight = 2),
         fill = guide_colorbar(barwidth = 13, barheight = 2)) +
  labs(x = "Longitude", y = "Latitude") +
  # theme
  theme_bw(base_family = "Times") +
  theme(axis.text.x      = element_text(size = text.size, vjust = 0.5),
        axis.text.y      = element_text(size = text.size),
        axis.title       = element_text(size = text.size + 2, face = 'bold'),
        plot.title       = element_text(size = text.size + 2, face = 'bold'),
        legend.position  = "bottom",
        legend.title     = element_text(size = text.size, face = 'bold'),
        legend.text      = element_text(size = text.size),
        panel.border     = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_blank())

# for surface change the color when prec == 0
df.prec = df.prec %>%
  mutate(prec = ifelse(prec == 0, NA, prec))  # Convert 0 values to NA

surface.plot =
ggplot() +
  geom_raster( data = df.prec, mapping = aes(x = lon, y = lat, fill = prec) ) +
  # country, border
  geom_sf(data = countries, fill = NA, color = "#000", linewidth = 0.4) +
```

```
  geom_rect(data = map.extent, fill = NA, color = "#67001f", size = 0.8,
    aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax) ) +
  coord_sf( xlim = c(min(df.prec$lon), max(df.prec$lon)),
            ylim = c(min(df.prec$lat), max(df.prec$lat)),
            expand = FALSE ) +
  scale_fill_gradientn(colors = prec.colors, na.value = "#d9d9d9",
                       name = "Precipitation (mm)") +
  guides(fill  = guide_colorbar(barwidth = 13, barheight = 2) ) +
  labs(x = "Longitude", y = "Latitude") +
  theme_bw(base_family = "Times") +
  theme(
    axis.text.x       = element_text(size = text.size, vjust = 0.5),
    axis.text.y       = element_text(size = text.size),
    axis.title        = element_text(size = text.size + 2, face = "bold"),
    plot.title        = element_text(size = text.size + 2, face = "bold"),
    legend.position   = "bottom",
    legend.title      = element_text(size = text.size, face = "bold"),
    legend.text       = element_text(size = text.size),
    panel.border      = element_rect(color = "#000000", fill = NA, linewidth = 0.5),
    panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
    panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_blank()
  )

# join map
join.map = ggarrange(spatial.plot, surface.plot,
                         ncol = 2,
                         common.legend = TRUE,
                         legend = "bottom")
ggsave(filename = paste0(path.plot, "HW1_FigureP2a_Maps.png"),
       plot     = join.map,
       width    = 12, height = 6.5, bg = "#fff")
```

Appendix B. R code - Problem2b

```r
# Code details -----------------------------------------------------------------
# author: Catalina Jerez
# mail: catalina.jerez@colorado.edu
# Course: Advanced Data Analysis Techniques
# Code: CVEN 6833
# Homework #1: Problem 2b) distance to sea
# Department of Civil, Environmental and Architectural Engineering
# College of Engineering & Applied Science
# University of Colorado Boulder

# Library ----------------------------------------------------------------------
gc()
rm(list = ls())

# Load necessary libraries
spatialAnalysis.Lib     = c("sf", "geosphere", "akima")
dataManipulation.Lib    = c("dplyr", "reshape2", "tidyr")
dataVisualization.Lib   = c("ggplot2", "ggrepel", "scales",
                            "rnaturalearth", "rnaturalearthdata")
list.packages           = unique(c(spatialAnalysis.Lib, dataManipulation.Lib, dataVisualization
    ↪ .Lib))
# Load all required packages
sapply(list.packages, require, character.only = TRUE)

# Path -------------------------------------------------------------------------
path = "/Users/caje3244/OneDrive - UCB-O365/2025 Spring/CVEN 6833/"
path.code = file.path(path, "Codes/")
path.plot = file.path(path, "Figures/")

# Inputs -----------------------------------------------------------------------

# load datasets
climatol   = read.table("http://civil.colorado.edu/~balajir/CVEN6833/HWs/HW-1/climatol-
    ↪ ann.txt")
topo       = read.table("http://civil.colorado.edu/~balajir/CVEN6833/HWs/HW-1/india-grid
    ↪ -topo.txt")
rajee.grid = read.table("http://civil.colorado.edu/~balajir/CVEN6833/HWs/HW-1/Rajeevan
    ↪ -grid.txt")

# assign column names
colnames(climatol)   = c("lon", "lat", "elev", "prec")
colnames(topo)       = c("lon", "lat", "elev")
colnames(rajee.grid) = c("lon", "lat")

# interpolate elevation (DEM) and precipitation data
interp.elev = interp(x = topo$lon    , y = topo$lat    , z = topo$elev    , extrap = TRUE,
```

```
                  duplicate = "median")
interp.prec = interp(x = climatol$lon, y = climatol$lat, z = climatol$prec, extrap = TRUE,
                  duplicate = "median")

# convert interpolation results to structured data frames
df.elev = data.frame(
  lon   = rep(interp.elev$x, times = length(interp.elev$y)),
  lat   = rep(interp.elev$y, each = length(interp.elev$x)),
  elev  = as.vector(interp.elev$z)) %>% drop_na()

df.prec = data.frame(
  lon   = rep(interp.prec$x, times = length(interp.prec$y)),
  lat   = rep(interp.prec$y, each = length(interp.prec$x)),
  prec  = as.vector(interp.prec$z) ) %>% drop_na()

# merge elevation and precipitation data
df.map     = left_join(df.elev, df.prec, by = c("lon", "lat"))
map.extent = data.frame(xmin = min(df.map$lon),
                        xmax = max(df.map$lon),
                        ymin = min(df.map$lat),
                        ymax = max(df.map$lat))

# ggplot variables
elev.colors = colorRampPalette(c("#f6e8c3", "#c9cba3", "#ffe1a8", "#e26d5c",
                                 "#723d46", "#472d30", "#543005"))(1000)
prec.colors = colorRampPalette(c("#0000FF", "#00FFFF", "#00FF00", "#FFFF00",
                                 "#FFA500", "#FF0000", "#FF00FF"))(1000)
text.size   = 15
label.size  = 5
crs.target  = st_crs(4326)  # WGS84 CRS

# Functions to calc min dist to coasline -------------------------------------

# compute the minimum distance from each station to the coastline
MinDistHaversine = function(lon, lat, coast_df) {
  distances = distHaversine(matrix(c(lon, lat), ncol = 2), coast_df)
  min(distances) / 1000  # m to km
}

MinDistGeo = function(lon, lat, coast_df) {
  distances = distGeo(matrix(c(lon, lat), ncol = 2), coast_df)
  min(distances) / 1000  # m to km
}


# Get India's coastline data -------------------------------------------------
options(warn=-1)
# sea's centroid
```

```r
sea.centroid = data.frame(
  lon  = c(65, 88),
  lat  = c(14, 15),
  name = c("Arabian Sea", "Bay of Bengal"))

# download and set crs
countries          = ne_countries(scale = "medium", returnclass = "sf") %>%
  st_transform(crs  = crs.target)
# calc centroids
countries.centroids = countries %>%
  st_centroid() %>%
  st_transform(crs = crs.target) %>%
  st_crop(xmin = min(df.map$lon), xmax = max(df.map$lon),
          ymin = min(df.map$lat), ymax = max(df.map$lat)) %>%
  mutate(lon = st_coordinates(.)[,1], lat = st_coordinates(.)[,2])
# get coastline
countries.coast    = countries %>%
  st_transform(crs = crs.target) %>%
  st_crop(xmin     = min(df.map$lon), xmax = max(df.map$lon),
          ymin     = min(df.map$lat), ymax = max(df.map$lat))
coastline.lines    = st_cast(countries.coast, "MULTILINESTRING")
coast.points       = as.data.frame(st_coordinates(coastline.lines))
colnames(coast.points) = c("lon", "lat", "Z", "M")

# separate coastlines into regions
coast.points       = coast.points[, c("lon", "lat")]    # drop unnecessary vars
coast.arabian      = coast.points %>% filter(lon < 76)  # west coast for Arabian Sea
coast.bengal       = coast.points %>% filter(lon >= 76) # east coast for Bay of Bengal

location.labels = c("Min. distance to any coast",
                    "Min. distance to Arabian Sea",
                    "Min. distance to Bay of Bengal")
# minimum distances by Haversine method ---------------------------------------
climaHarv          = climatol
climaHarv$mdCoast   = mapply(MinDistHaversine, climaHarv$lon, climaHarv$lat,
                             MoreArgs = list(coast_df = coast.points))
climaHarv$mdArabian = mapply(MinDistHaversine, climaHarv$lon, climaHarv$lat,
                              MoreArgs = list(coast_df = coast.arabian))
climaHarv$mdBengal  = mapply(MinDistHaversine, climaHarv$lon, climaHarv$lat,
                             MoreArgs = list(coast_df = coast.bengal))
str(climaHarv)
# melt for plotting
clima.melt         = melt(climaHarv, id = c("lon", "lat", "elev", "prec"))
clima.melt$variable = factor(clima.melt$variable)
levels(clima.melt$variable) = location.labels

# spatial map with precipitation, elevation, and distances
spatial.plot =
```

```
ggplot() +
# elevation raster
geom_raster(data = df.map, aes(x = lon, y = lat, fill = elev), alpha = .85) +
# country, borders
geom_sf(data = countries, fill = NA, color = "#000", linewidth = 0.4) +
geom_rect(data = map.extent, fill = NA, color = "#67001f", linewidth = 0.8,
        aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax)) +
geom_contour(data = df.map, aes(x = lon, y = lat, z = elev), color = "#000",
          linewidth = 0.1, bins = 30) +
# precipitation points
geom_point(data = clima.melt, aes(x = lon, y = lat, size = value),
         shape = 21, color = "#000") +
geom_point(data = clima.melt, aes(x = lon, y = lat, color = prec, size = value),
         shape = 19, alpha = .8) +
# labels
geom_label_repel(data = sea.centroid, aes(x = lon, y = lat, label = name),
              family = "Times", size = label.size, fill = "#c6dbef", color = "#08306b",
              box.padding = 0.3, segment.color = "black") +
geom_label_repel(data = countries.centroids, aes(x = lon, y = lat, label = name),
              family = "Times", size = label.size-2, fill = "#fff", color = "#000",
              box.padding = 0.15, segment.color = "#bdbdbd") +
labs(x = "Longitude", y = "Latitude") +
# coordinate system
coord_sf(xlim = c(min(df.map$lon), max(df.map$lon)),
       ylim = c(min(df.map$lat), max(df.map$lat)), expand = FALSE) +
# scales
scale_fill_gradientn( colors = elev.colors, name = "Elevation (km)") +
scale_color_gradientn(colors = prec.colors, name = "Precipitation (mm)") +
scale_size_continuous(name   = "Distance (km)",
                   breaks = seq(0, max(clima.melt$value, na.rm = TRUE), by = 200))+
# guides
guides(color = guide_colorbar(barwidth = 2, barheight = 6, title.position = "right", order
  ↪ = 3),
     fill  = guide_colorbar(barwidth = 2, barheight = 6, title.position = "right", order =
  ↪ 2),
     size  = guide_legend(nrow = 3, order = 1)) +
# facet
facet_wrap(~variable, ncol = 2)+
# theme
theme_bw(base_family = "Times") +
theme(axis.text.x      = element_text(size = text.size, vjust = 0.5),
    axis.text.y      = element_text(size = text.size),
    axis.title       = element_text(size = text.size + 2, face = 'bold'),
    plot.title       = element_text(size = text.size + 2, face = 'bold'),

    legend.position  = c(0.75, 0.25),
    legend.title     = element_text(size = text.size, face = 'bold'),
    legend.text      = element_text(size = text.size),
```

```
185
186         strip.text       = element_text(size = text.size, face = "bold"),
187         strip.background = element_blank(),
188
189         panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
190         panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
191         panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
192         panel.grid.major.x = element_blank(),
193         panel.grid.major.y = element_blank())
194
195 # Save
196 ggsave(filename = paste0(path.plot, "HW1_FigureP2b_SpatialMap_DistHarvesine.png"),
197       plot = spatial.plot, width =10.5, height = 10.5)
198
199 saveRDS(climaHarv, file = paste0(path.code, "RData/hw1_climatol_distHarvesine.rds") )
200
201 # minimum distances by Geodesic method ----------------------------------------
202 climaGeo           = climatol
203 climaGeo$mdCoast   = mapply(MinDistGeo, climaGeo$lon, climaGeo$lat,
204                             MoreArgs = list(coast_df = coast.points))
205 climaGeo$mdArabian = mapply(MinDistGeo, climaGeo$lon, climaGeo$lat,
206                             MoreArgs = list(coast_df = coast.arabian))
207 climaGeo$mdBengal  = mapply(MinDistGeo, climaGeo$lon, climaGeo$lat,
208                             MoreArgs = list(coast_df = coast.bengal))
209
210 # melt for plotting
211 clima.melt          = melt(climaGeo, id = c("lon", "lat", "elev", "prec"))
212 clima.melt$variable = factor(clima.melt$variable)
213 levels(clima.melt$variable) = location.labels
214
215 # spatial map with precipitation, elevation, and distances
216 spatial.plot =
217   ggplot() +
218   # elevation raster
219   geom_raster(data = df.map, aes(x = lon, y = lat, fill = elev), alpha = .85) +
220   # country, borders
221   geom_sf(data = countries, fill = NA, color = "#000", linewidth = 0.4) +
222   geom_rect(data = map.extent, fill = NA, color = "#67001f", linewidth = 0.8,
223             aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax)) +
224   geom_contour(data = df.map, aes(x = lon, y = lat, z = elev), color = "#000",
225                linewidth = 0.1, bins = 30) +
226   # precipitation points
227   geom_point(data = clima.melt, aes(x = lon, y = lat, size = value),
228              shape = 21, color = "#000") +
229   geom_point(data = clima.melt, aes(x = lon, y = lat, color = prec, size = value),
230              shape = 19, alpha = .8) +
231   # labels
232   geom_label_repel(data = sea.centroid, aes(x = lon, y = lat, label = name),
```

```
                family = "Times", size = label.size, fill = "#c6dbef", color = "#08306b",
                box.padding = 0.3, segment.color = "black") +
  geom_label_repel(data = countries.centroids, aes(x = lon, y = lat, label = name),
                family = "Times", size = label.size-2, fill = "#fff", color = "#000",
                box.padding = 0.15, segment.color = "#bdbdbd") +
  labs(x = "Longitude", y = "Latitude") +
  # coordinate system
  coord_sf(xlim = c(min(df.map$lon), max(df.map$lon)),
         ylim = c(min(df.map$lat), max(df.map$lat)), expand = FALSE) +
  # scales
  scale_fill_gradientn( colors = elev.colors, name = "Elevation (km)") +
  scale_color_gradientn(colors = prec.colors, name = "Precipitation (mm)") +
  scale_size_continuous(name   = "Distance (km)",
                        breaks = seq(0, max(clima.melt$value, na.rm = TRUE), by = 200))+
  # guides
  guides(color = guide_colorbar(barwidth = 2, barheight = 6, title.position = "right", order
    ↪ = 3),
         fill  = guide_colorbar(barwidth = 2, barheight = 6, title.position = "right", order =
    ↪ 2),
         size  = guide_legend(nrow = 3, order = 1)) +
  # facet
  facet_wrap(~variable, ncol = 2)+
  # theme
  theme_bw(base_family = "Times") +
  theme(axis.text.x      = element_text(size = text.size, vjust = 0.5),
        axis.text.y      = element_text(size = text.size),
        axis.title       = element_text(size = text.size + 2, face = 'bold'),
        plot.title       = element_text(size = text.size + 2, face = 'bold'),

        legend.position  = c(0.75, 0.25),
        legend.title     = element_text(size = text.size, face = 'bold'),
        legend.text      = element_text(size = text.size),

        strip.text       = element_text(size = text.size, face = "bold"),
        strip.background = element_blank(),

        panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_blank())

# Save
ggsave(filename = paste0(path.plot, "HW1_FigureP2b_SpatialMap_DistGeodesic.png"),
       plot = spatial.plot, width =10.5, height = 10.5)

saveRDS(climaGeo, file = paste0(path.code, "RData/hw1_climatol_distGeodesic.rds") )
```

```r
# comparison of distance calcs methods ----------------------------------------
climaDiff               = climatol
climaDiff$diffCoast     = abs(climaHarv$mdCoast - climaGeo$mdCoast)
climaDiff$diffArabian   = abs(climaHarv$mdArabian - climaGeo$mdArabian)
climaDiff$diffBengal    = abs(climaHarv$mdBengal - climaGeo$mdBengal)

# Summary statistics
summaryDiff  = data.frame(
  variable   = location.labels,
  meanDiff = c(mean(climaDiff$diffCoast, na.rm = TRUE),
               mean(climaDiff$diffArabian, na.rm = TRUE),
               mean(climaDiff$diffBengal, na.rm = TRUE)),
  medDiff  = c(median(climaDiff$diffCoast, na.rm = TRUE),
               median(climaDiff$diffArabian, na.rm = TRUE),
               median(climaDiff$diffBengal, na.rm = TRUE)),
  sdDiff   = c(sd(climaDiff$diffCoast, na.rm = TRUE),
               sd(climaDiff$diffArabian, na.rm = TRUE),
               sd(climaDiff$diffBengal, na.rm = TRUE)) ) %>%
  mutate(label = paste0("Mean: ", round(meanDiff, 2),
                        "\nMedian: ", round(medDiff, 2),
                        "\nSD: ", round(sdDiff, 2)))
print(summaryDiff)

# melt data
clima.melt.diff = melt(climaDiff, id = c("lon", "lat", "elev", "prec"))
levels(clima.melt.diff$variable) = location.labels

density.plot =
  ggplot(data = clima.melt.diff, aes(x = value)) +
  geom_density(alpha = 0.6, color = "#8e0152", fill = "#de77ae") +
  facet_wrap(~variable, nrow = 1, scale = "free_y") +
  scale_y_continuous(breaks = pretty_breaks(n=4)) +
  labs(x = "Absolute difference (Km)", y = "Density") +
  geom_text(data = summaryDiff, aes(x = Inf, y = Inf, label = label),
          hjust = 1.1, vjust = 1.1, inherit.aes = FALSE, size = 4.5, family = "Times") +
  # theme
  theme_bw(base_family = "Times") +
  theme(axis.text.x      = element_text(size = text.size, vjust = 0.5),
        axis.text.y      = element_text(size = text.size),
        axis.title       = element_text(size = text.size + 2, face = 'bold'),
        legend.position  = "none",

        strip.text       = element_text(size = text.size, face = "bold"),
        strip.background = element_blank(),

        panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
```

```
327         panel.grid.major.x = element_blank(),
328         panel.grid.major.y = element_blank())
329 
330 # Save plot
331 ggsave(filename = paste0(path.plot, "HW1_FigureP2b_Comparison_HavGeo.png"),
332        plot = density.plot, width = 13, height = 4.5)
333 
334 # Save difference data
335 saveRDS(climaDiff, file = paste0(path.code, "RData/hw1_climatol_diff_HavGeo.rds"))
```

Appendix C. Problems 2c-6

C.1. Library

```
# Code details ------------------------------------------------------------
# author: Catalina Jerez
# mail: catalina.jerez@colorado.edu
# Course: Advanced Data Analysis Techniques
# Code: CVEN 6833
# Department of Civil, Environmental and Architectural Engineering
# College of Engineering & Applied Science
# University of Colorado Boulder

# Library -----------------------------------------------------------------
required.lib = c("MASS", "leaps", "dplyr", "reshape2",
                 "mgcv", "locfit", "akima",
                 "psych", "hydroGOF", "fields", "sm", "nlme", "minpack.lm",
                 "ggplot2", "ggpubr", "ggrepel", "purrr",
                 "scales", "maps", "rnaturalearth", "rnaturalearthdata")
sapply(required.lib, require, character.only = TRUE)
# Plotting ----------------------------------------------------------------
sptMap.ggplot      = function(clima, topography, title.plot = NULL,
                              crs.target = st_crs(4326), text.size = 14){
  options(warn=-1)
  # interpolate elevation (DEM) and precipitation data
  interp.elev = interp(x = topography$lon, y = topography$lat, z = topography$elev, extrap
    ↪  = TRUE,
                       duplicate = "median")
  interp.prec = interp(x = clima$lon, y = clima$lat, z = clima$prec, extrap = TRUE,
                       duplicate = "median")

  # convert interpolation results to structured data frames ----------------
  df.elev = data.frame(
    lon   = rep(interp.elev$x, times = length(interp.elev$y)),
    lat   = rep(interp.elev$y, each = length(interp.elev$x)),
    elev  = as.vector(interp.elev$z)) %>% drop_na()

  df.prec = data.frame(
    lon   = rep(interp.prec$x, times = length(interp.prec$y)),
    lat   = rep(interp.prec$y, each = length(interp.prec$x)),
    prec  = as.vector(interp.prec$z) ) %>% drop_na()

  # merge elevation and precipitation data
  df.map     = left_join(df.elev, df.prec, by = c("lon", "lat"))
  map.extent = data.frame(xmin = min(df.map$lon),
                          xmax = max(df.map$lon),
                          ymin = min(df.map$lat),
                          ymax = max(df.map$lat))
```

```
# download and clean spatial data
countries    = ne_countries(scale = "medium", returnclass = "sf") %>% st_transform(crs =
  ↪  crs.target)
countries_centroids = countries %>%
 st_centroid() %>%
 st_transform(crs = crs.target) %>%
 st_crop(xmin = min(df.map$lon), xmax = max(df.map$lon),
         ymin = min(df.map$lat), ymax = max(df.map$lat)) %>%
 mutate(lon = st_coordinates(.)[,1], lat = st_coordinates(.)[,2])

# ggplot --------------------------------------------------------------------
# elevation and precipitation colors
elev.colors = colorRampPalette(c("#f6e8c3", "#c9cba3", "#ffe1a8", "#e26d5c", "#723d46",
  ↪  "#472d30", "#543005"))(1000)
prec.colors = colorRampPalette(c("#0000FF", "#00FFFF", "#00FF00", "#FFFF00", "#
  ↪ FFA500", "#FF0000", "#FF00FF"))(1000)

spatial.plot =
 ggplot() +
 # elevation raster
 geom_raster(data = df.map, aes(x = lon, y = lat, fill = elev), alpha = .5, show.legend =
  ↪ F) +
 # country, border
 geom_sf(  data = countries, fill = NA, color = "#000", linewidth = 0.4) +
 geom_rect(data = map.extent, fill = NA, color = "#67001f", size = 0.8,
           aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax) ) +
 geom_contour(data = df.map, aes(x = lon, y = lat, z = elev), color = "#000", linewidth
  ↪ = 0.1, bins = 30) +
 geom_label_repel(data = countries_centroids, aes(x = lon, y = lat, label = name),
                  family = "Times", size = 2, fill = "#fff", color = "#000",
                  box.padding = 0.15, segment.color = "#bdbdbd") +
 # precipitation points
 geom_point(data = clima, aes(x = lon, y = lat), size = 3, shape = 21) +
 geom_point(data = clima, aes(x = lon, y = lat, color = prec), size = 2.2, shape = 19,
  ↪ alpha = .9) +
 # coordinate system
 coord_sf( xlim = c(min(df.prec$lon), max(df.prec$lon)),
           ylim = c(min(df.prec$lat), max(df.prec$lat)),
           expand = FALSE ) +
 scale_x_continuous(breaks = pretty(df.map$lon, n = 3)) +  # Move breaks here
 scale_y_continuous(breaks = pretty(df.map$lat, n = 3)) +  # Move breaks here

 # scales
 scale_fill_gradientn( colors = elev.colors, name = "Elevation (m)") +
 scale_color_gradientn(colors = prec.colors, name = "Precipitation (cm)",
                       breaks = pretty(range(clima$prec), 5)) +
 # guides and labels
 guides(color = guide_colorbar(barwidth = 16, barheight = 1.5),
```

```
        fill = guide_colorbar(barwidth = 16, barheight = 1.5)) +
  labs(x = "Longitude", y = "Latitude", title = "") +
  # theme
  theme_bw(base_family = "Times") +
  theme(axis.text.x      = element_text(size = text.size, vjust = 0.5),
        axis.text.y      = element_text(size = text.size),
        axis.title       = element_text(size = text.size + 1, face = 'bold'),
        plot.title       = element_text(size = text.size + 1, face = 'bold', hjust = .5),
        legend.position    = "bottom",
        legend.title       = element_blank(),
        legend.text        = element_text(size = text.size),
        panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_blank())

# for surface change the color when prec == 0
df.prec = df.prec %>%
  mutate(prec = ifelse(prec == 0, NA, prec))  # Convert 0 values to NA

surface.plot =
  ggplot() +
  geom_raster( data = df.prec, mapping = aes(x = lon, y = lat, fill = prec) ) +
  # country, border
  geom_sf(data = countries, fill = NA, color = "#000", linewidth = 0.4) +
  geom_rect(data = map.extent, fill = NA, color = "#67001f", size = 0.8,
            aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax) ) +
  coord_sf( xlim = c(min(df.prec$lon), max(df.prec$lon)),
            ylim = c(min(df.prec$lat), max(df.prec$lat)),
            expand = FALSE ) +
  scale_fill_gradientn(colors = prec.colors, na.value = "#d9d9d9",
                       name = "Precipitation (cm)") +
  guides(fill  = guide_colorbar(barwidth = 13, barheight = 2) ) +
  labs(x = "Longitude", y = "Latitude", title = "") +
  theme_bw(base_family = "Times") +
  theme(
    axis.text.x      = element_text(size = text.size, vjust = 0.5),
    axis.text.y      = element_text(size = text.size),
    axis.title       = element_text(size = text.size + 2, face = "bold"),
    plot.title       = element_text(size = text.size + 2, face = "bold"),
    legend.position    = "bottom",
    legend.title     = element_text(size = text.size, face = "bold"),
    legend.text       = element_text(size = text.size),
    panel.border     = element_rect(color = "#000000", fill = NA, linewidth = 0.5),
    panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
    panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
    panel.grid.major.x = element_blank(),
```

```
    panel.grid.major.y = element_blank()
  )

  # join map
  join.map = ggarrange(spatial.plot, surface.plot,
                       ncol = 2,
                       common.legend = TRUE,
                       legend = "bottom")
  join.map = annotate_figure(join.map,
                             top = text_grob(title.plot, family = "Times",
                                             face = "bold", size = text.size+2) )
  return(join.map)
}
pairs.ggplot       = function(data, text.size = 14){
  ggpairs(
    data,
    columns = sort(names(data)),
    aes(color = cut_interval(data$prec, n=2) ),  # categorize prec and plot in cm
    upper = list(continuous = wrap("cor", size = 3)),   # corr in upper panel
    lower = list(continuous = wrap("density", alpha = 0.6, size = 1.2)), # scatter in lower
     ↪ panel
    diag  = list(continuous = wrap("barDiag", alpha = 0.6, bins = 15)) # density on
     ↪ diagonal
  ) +
    scale_color_manual("Precipitation (cm per year)", values = scale.color, labels = scale.
     ↪ label) +
    scale_fill_manual("Precipitation (cm per year)", values = scale.color, labels = scale.label)
     ↪  +
    theme_bw(base_family = "Times") +
    theme(axis.text.x      = element_text(size = text.size, vjust = 0.5),
          axis.text.y      = element_text(size = text.size),
          axis.title       = element_text(size = text.size + 2, face = 'bold'),
          plot.title       = element_text(size = text.size + 2, face = 'bold'),

          strip.text       = element_text(size = text.size, face = "bold"),
          strip.background = element_blank(),
          legend.position = "bottom",
          legend.title     = element_text(size = text.size, face = 'bold'),
          legend.text      = element_text(size = text.size),

          panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
          panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
          panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
          panel.grid.major.x = element_blank(),
          panel.grid.major.y = element_blank())
}
diagnostics.ggplot = function(data, text.size = 14){
  # data is df with "obs", "sim", "res"
```

```r

# are the residuals normal and iid?
# 1) normality: qq plot
qq.plot = ggplot(data, aes(sample = res)) +
  stat_qq(color = "#005a32", fill = "#a1d99b", size = 3, shape = 21) +
  stat_qq_line(color = "#99000d", linewidth = 1) +
  labs(x = "Theoretical Quantiles", y = "Sample Quantiles (cm)",
       title = "a) Q-Q Plot")+
  scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
  theme_bw(base_family = "Times") +
  theme(axis.text.x        = element_text(size = text.size, vjust = 0.5),
        axis.text.y        = element_text(size = text.size, angle = 90, hjust = .5, vjust = .5),
        axis.title         = element_text(size = text.size, face = 'bold'),
        plot.title         = element_text(size = text.size+1, face = 'bold', hjust = .5),
        panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.major.x = element_line(color = "#d9d9d9", linetype = "dashed"),
        panel.grid.major.y = element_line(color = "#d9d9d9", linetype = "dashed") )

# 1) normality: histogram
hist.plot = ggplot(data, aes(x = res)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30, fill = "#a1d99b", color = "
   ↪ black") +
  geom_density(color = "#99000d", linewidth = 1) +
  labs(x = "Residuals (cm)", y = "Density", title = "b) Histogram")+
  scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
  theme_bw(base_family = "Times") +
  theme(axis.text.x        = element_text(size = text.size, vjust = 0.5),
        axis.text.y        = element_text(size = text.size, angle = 90, hjust = .5, vjust = .5),
        axis.title         = element_text(size = text.size, face = 'bold'),
        plot.title         = element_text(size = text.size+1, face = 'bold', hjust = .5),
        panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.major.x = element_line(color = "#d9d9d9", linetype = "dashed"),
        panel.grid.major.y = element_line(color = "#d9d9d9", linetype = "dashed") )

# 2) independence of residuals: check autocorrelation
acf.data = acf(data$res, plot = FALSE)
acf.df   = data.frame(lag = acf.data$lag, acf = acf.data$acf)
ndata    = length(data$res)
conf.int = 1.96 / sqrt(ndata) # confidence interval

acf.plot = ggplot(acf.df, aes(x = lag, y = acf)) +
  geom_bar(stat = "identity", color = "#005a32", fill = "#a1d99b") +
  geom_hline(yintercept = c(-conf.int, conf.int), linetype = "dashed", color = "#99000d",
   ↪ linewidth = 0.8) +
```

```
    labs(x = "Lag", y = "ACF", title = "c) Autocorrelation")+
    scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
    theme_bw(base_family = "Times") +
    theme(axis.text.x       = element_text(size = text.size, vjust = 0.5),
          axis.text.y       = element_text(size = text.size, angle = 90, hjust = .5, vjust = .5),
          axis.title        = element_text(size = text.size, face = 'bold'),
          plot.title        = element_text(size = text.size+1, face = 'bold', hjust = .5),
          panel.border      = element_rect(color = "#000000", fill = NA, linewidth = .5),
          panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
          panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
          panel.grid.major.x = element_line(color = "#d9d9d9", linetype = "dashed"),
          panel.grid.major.y = element_line(color = "#d9d9d9", linetype = "dashed") )

  # 3) Homoskedasticity (constant variance? residuals vs fitted
  resid.plot = ggplot(data, aes(x = sim, y = res)) +
    geom_point(color = "#005a32", fill = "#a1d99b", shape = 21, size = 3) +
    geom_hline(yintercept = 0, color = "#99000d", linewidth = 1) +
    labs(x = "Estimated Precipitation (cm)", y = "Residuals (cm)",
         title = "d) Residuals vs Estimated Precipitation") +
    scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
    theme_bw(base_family = "Times") +
    theme(axis.text.x       = element_text(size = text.size, vjust = 0.5),
          axis.text.y       = element_text(size = text.size, angle = 90, hjust = .5, vjust = .5),
          axis.title        = element_text(size = text.size, face = 'bold'),
          plot.title        = element_text(size = text.size+1, face = 'bold', hjust = .5),
          panel.border      = element_rect(color = "#000000", fill = NA, linewidth = .5),
          panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
          panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
          panel.grid.major.x = element_line(color = "#d9d9d9", linetype = "dashed"),
          panel.grid.major.y = element_line(color = "#d9d9d9", linetype = "dashed") )

  # arrange together
  ggarrange(qq.plot, hist.plot, acf.plot, resid.plot, ncol = 2, nrow = 2)

}
scatter.ggplot     = function(data, model = NULL, title = NULL,
                              text.size = 14){
  # data is a dataframe with columns obs and sim.

  # estimate range from df.scatter$sim if range between 0 and 1,
  # the family its binomial and we need to rescale df.scatter$obs
  range.obs = range(data$sim, na.rm = TRUE)
  if(isTRUE(0 <= range.obs[1] & range.obs[2] <= 1)){
    data$obs = scales::rescale(data$obs, to = c(0, 1))
  }else{
    data$obs = ifelse(data$obs <= 0, runif(1, 0.0001, 0.001), data$obs)
  }

```

```r
# data is df with obs and sim precipitation
if(is.null(model)){
  df.skill    = data.frame(
    metric    = c(rownames(melt(evaluation.metrics(data$obs, data$sim)))),
    value     = c(melt(evaluation.metrics(data$obs, data$sim))$value)
  )
  skill.text  = paste0(df.skill$metric, ": ", round(df.skill$value, 2), collapse = "\n")
  eq.text     = paste0("")

}else{
  df.skill    = data.frame(
    metric    = c("AIC", "BIC", rownames(melt(evaluation.metrics(data$obs, data$sim)))),
    value     = c(AIC(model), BIC(model), melt(evaluation.metrics(data$obs, data$sim))$
    ↪ value)
  )
  skill.text  = paste0(df.skill$metric, ": ", round(df.skill$value, 2), collapse = "\n")
  coefs       = coef(model)
  eq.text     = paste0("Sim~prec == ", round(coefs[1], 2), "~(cm)")
  if (length(coefs) > 1) {
    terms.text = paste0(round(coefs[-1], 1), "%*%", names(coefs)[-1], collapse = " + ")
    eq.text   = paste(eq.text, "+", terms.text)
  }
}

ggplot(data = data, aes(x = obs, y = sim)) +
  geom_abline(slope = 1, intercept=0, color="#67000d", linewidth = 1.5) +
  geom_point( shape = 21, color = "#08306b", fill = "#4292c6", size = 4) +
  geom_abline(slope = 1, intercept=0, color="#67000d", linetype = 2) +
  scale_x_continuous(limits = c(min(data$obs), max(data$obs)) ) +
  scale_y_continuous(limits = c(min(data$obs), max(data$obs)) ) +
  labs(x = "Observed precipitacion (cm)", y = "Simulated precipitacion (cm)")+
  #  model equation
  annotate("text", x = min(data$obs), y = max(data$obs), fontface = "bold",
           label = paste(title, "model"), hjust = 0, vjust = 1, size = 5) +

  annotate("text", x = min(data$obs), y = .96*max(data$obs), parse = T,
           label = paste(eq.text), hjust = 0, vjust = 1, size = 3.7) +
  # model performance
  annotate("text", x = min(data$obs),  y = .9*max(data$obs),
           label = skill.text, hjust = 0, vjust = 1, size = 4.5, fontface = "bold") +

  theme_bw(base_family = "Times") +
  theme(axis.text.x        = element_text(size = text.size, vjust = 0.5),
        axis.text.y        = element_text(size = text.size, angle = 90, hjust = .5, vjust = .5),
        axis.title         = element_text(size = text.size + 2, face = 'bold'),
        panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
```

```
319         panel.grid.major.x = element_blank(),
320         panel.grid.major.y = element_blank())
321 }
322 # Skill metrics ----------------------------------------------------------
323 evaluation.metrics = function(actual, predicted) {
324   mse  = mean((actual - predicted)^2, na.rm = TRUE)
325   rmse = sqrt(mse)
326   r2   = 1 - sum((actual - predicted)^2) / sum((actual - mean(actual))^2)
327   metrics = c(MSE = mse, RMSE = rmse, R2 = r2)
328   return(metrics)
329 }
330 evaluation.model   = function(actual, data, model, local = FALSE) {
331   # predict values
332   predicted = predict(model, data = data)
333 
334   # calc metrics
335   mse  = mean((actual - predicted)^2, na.rm = TRUE)
336   rmse = sqrt(mse)
337   r2   = 1 - sum((actual - predicted)^2) / sum((actual - mean(actual))^2)
338 
339   if(local){
340     metrics = c(MSE = mse, RMSE = rmse, R2 = r2)
341   }else{
342     n.params  = length(coef(model))  # parameters in the model
343     n.obs     = length(actual)  # n obs
344     log.likelihood = -0.5 * n.obs * (log(2 * pi * mse) + 1)
345     aic  = -2 * log.likelihood + 2 * n.params
346     bic  = -2 * log.likelihood + log(n.obs) * n.params
347     metrics = c(MSE = mse, RMSE = rmse, R2 = r2, AIC = aic, BIC = bic)
348   }
349   return(metrics)
350 }
351 
352 # Fitting models (lm, glm, gam, locPoly) ---------------------------------------
353 
354 # subset selection using regsubsets
355 subset.select = function(data) {
356   subset.model = regsubsets(prec ~ ., data = data, nbest = 10)
357   subset.summ  = summary(subset.model)
358   subset.index = which.min(subset.summ$bic)
359   subset.vars  = names(coef(subset.model, subset.index))
360   data[, c("prec", subset.vars[subset.vars != "(Intercept)"])]
361 }
362 
363 # fitting functions
364 fit_Linear    = function(data, obj.fn = "BIC") {
365   # check if response variable "prec" exists in the dataset
366   if (!"prec" %in% names(data)) {
```

```
  stop("Error: The response variable 'prec' is missing in the dataset.")
}

# Fitting lm models  ------------------------------------------------------
# 0) Basic linear model
basic.model = lm(prec ~ ., data = data)

# 1) AIC-based model selection
aic.model = stepAIC(lm(prec ~ ., data = data), direction = "backward",
                    trace = FALSE)
# 2) BIC-based model selection
bic.model = stepAIC(lm(prec ~ ., data = data), direction = "backward",
                    k = log(nrow(data)),
                    trace = FALSE)
# 3) Leaps - all subsets regression
leaps.model = regsubsets(prec ~ ., data = data, nbest = length(colnames(data)) -1 )
best.subset = summary(leaps.model)
model.index = which.min(best.subset$bic)
best.vars   = names(coef(leaps.model, model.index))
new.train   = data[, c("prec", best.vars[-1])]  # rm intercept
# refit the best subset model to get coefficients
leaps.model = lm(as.formula(paste("prec ~", paste(best.vars[-1], collapse=" + "))),
                    data = new.train)

# Compare and return best model -------------------------------------------
cat("\nModel selection comparison:")
name.models   = c("Linear", "AIC Selection", "BIC Selection", "Leaps")
fitted.models = list(basic.model, aic.model, bic.model, leaps.model)
names(fitted.models) = name.models

df.skill      = data.frame(
  model       = name.models,
  t(sapply(fitted.models, function(fit) evaluation.model(data$prec, data, fit) ) )
)
metrics = names(df.skill)[-1]

# select best model based on BIC values (by default)
if (obj.fn == "R2") {
  best.model = df.skill[which.max(df.skill[[obj.fn]]), ]
} else {
  best.model = df.skill[which.min(df.skill[[obj.fn]]), ]
}

cat("\nModel summary:\n")
print(df.skill)
cat(sprintf("\nBest link function: %s\n", best.model$model))
for (metric in metrics) {
  cat(sprintf("  %s = %.3f\n", metric, best.model[[metric]]))
```

```
  }

  # return best model
  return( fitted.models[[best.model$model]] )
}
fit_GLM       = function(data, family, obj.fn = "BIC") {
  options(warn = -1)

  if (!"prec" %in% names(data)) {
    stop("Error: The response variable 'prec' is missing in the dataset.")
  }

  if (family == "binomial") {
    data$prec = scales::rescale(data$prec, to = c(0, 1))
  } else {
    data$prec = ifelse(data$prec <= 0, runif(1, 0.0001, 0.001), data$prec)
  }

  links = switch(family,
                 "Gamma"    = c("log", "inverse", "identity"),
                 "binomial" = c("logit", "probit", "cauchit"),
                 "gaussian" = c("identity"),
                 stop("Invalid family specified"))

  model.links = list()
  # j=1
  for (j in seq_along(links)) {
    family.func = match.fun(family)
    cat(paste0("\n---------- Fitting GLM with ", family, " + ", links[j] ," link\n"))
    init.model   = lm(prec ~ ., data = data)
    start.values = coef(init.model)
    basic.model  = glm(prec ~ ., data = data, family = family.func(link = links[j]),
                       maxit = 1000, start = start.values)

    model.links[[j]] = basic.model
  }

  names(model.links) = links

  cat("\nModel link comparison:")
  df.skill = data.frame(
    model = links,
    t(sapply(model.links, function(fit) evaluation.model(data$prec, data, fit)))
  )

  metrics = names(df.skill)[-1]

  if (obj.fn == "R2") {
```

```
463     best.model = df.skill[which.max(df.skill[[obj.fn]]), ]
464   } else {
465     best.model = df.skill[which.min(df.skill[[obj.fn]]), ]
466   }
467 
468   cat("\nModel summary:\n")
469   print(df.skill)
470   cat(sprintf("\nBest link function: %s\n", best.model$model))
471   for (metric in metrics) {
472     cat(sprintf("  %s = %.3f\n", metric, best.model[[metric]]))
473   }
474 
475   return(model.links[[best.model$model]])
476 }
477 fit_locPoly   = function(data, family, obj.fn = "R2") {
478   options(warn = -1)
479 
480   if (!"prec" %in% names(data)) {
481     stop("Error: The response variable 'prec' is missing in the dataset.")
482   }
483 
484   if (family == "binomial") {
485     data$prec = scales::rescale(data$prec, to = c(0, 1))
486   } else {
487     data$prec = ifelse(data$prec <= 0, runif(1, 0.0001, 0.001), data$prec)
488   }
489 
490   # family = tolower(family) #
491   links  = switch(family, # "ident", "log", "logit", "inverse", "sqrt" and "arcsin".
492                   "binomial" = c("logit", "probit", "arcsin"),
493                   "gamma"    = c("log", "inverse"),
494                   "gaussian" = c("ident"),
495                   "geom"     = c("log"),
496                   "poisson"  = c("log", "sqrt"),
497                   stop("Invalid family specified"))
498 
499   model.links = list()
500   # j=1
501   for (j in seq_along(links)) {
502     cat(paste0("\n---------- Fitting local Polynomial with ", family, " + ", links[j] ," link\n"))
503 
504     # range for alpha order 1 and 2
505     nvars   = ncol(data) - 1
506     alpha1  = round(seq(2*(nvars*1 + 1)/(nrow(data)), 1, by = 0.1), 2)
507     alpha2  = round(seq(2*(nvars*2 + 1)/(nrow(data)), 1, by = 0.1), 2)
508     alpha   = c(alpha1, alpha2)
509     n.alpha = length(alpha1)
510 
```

```
# kern options
kern.options = c("bisq", "tcub") #  "bisq","tcub", "trwt", "tria"
best.model = NULL
best.alpha = NULL
best.deg   = NULL
best.gcv   = Inf
best.kern  = NULL

capture.output({
 for (kern in kern.options) {
   cat(paste0("Testing kernel: ", kern, "\n"))
   gcv.deg1 = tryCatch({
     gcvplot(prec ~ ., data = data, maxk = 1000, alpha = alpha1,
             deg = 1, kern = kern, scale = T,
             ev = dat(), #lfgrid() for grids, dat for points or none()
             family = family, link = links[j]
     )
   }, error = function(e) NULL)

   gcv.deg2 = tryCatch({
     gcvplot(prec ~ ., data = data, maxk = 1000, alpha = alpha2,
             deg = 2, kern = kern, scale = T,
             ev = dat(),
             family = family, link = links[j]
     )
   }, error = function(e) NULL)

   if (!is.null(gcv.deg1) && !is.null(gcv.deg2)) {
     gcv.values     = order(c(gcv.deg1$values, gcv.deg2$values))
     deg.selected   = ifelse(gcv.values[1] > n.alpha, 2, 1)
     alpha.selected = alpha[gcv.values[1]]
     gcv.score      = c(gcv.deg1$values, gcv.deg2$values)[gcv.values[1]]

     if (gcv.score < best.gcv) {
       best.gcv   = gcv.score
       best.deg   = deg.selected
       best.alpha = alpha.selected
       best.kern  = kern
     }
   }
 }
})
cat(sprintf("Kernel: %s | Alpha: %.3f | Degree: %d\n",
            best.kern, best.alpha, best.deg))
basic.model = locfit(prec ~ ., data = data, alpha = best.alpha,
                     maxk = 100, deg = best.deg, kern = best.kern,
                     ev = dat(),
                     scale = T, family = family, link = links[j]
```

```
                    # lfproc = locfit.robust(prec ~ ., data = data)
                    )
    model.links[[j]] = basic.model
  }
  names(model.links) = links

  cat("\nModel link comparison:")
  df.skill = data.frame(
    model = links,
    t(sapply(model.links, function(fit) evaluation.model(data$prec, data, fit, local = T)))
  )

  metrics = names(df.skill)[-1]

  if (obj.fn == "R2") {
    best.model = df.skill[which.max(df.skill[[obj.fn]]), ]
  } else {
    best.model = df.skill[which.min(df.skill[[obj.fn]]), ]
  }

  cat("\nModel summary:\n")
  print(df.skill)
  cat(sprintf("\nBest link function: %s\n", best.model$model))
  for (metric in metrics) {
    cat(sprintf("  %s = %.3f\n", metric, best.model[[metric]]))
  }

  return(model.links[[best.model$model]])
}
fit_GAM      = function(data, family, obj.fn = "BIC") {
  options(warn = -1)

  if (!"prec" %in% names(data)) {
    stop("Error: The response variable 'prec' is missing in the dataset.")
  }

  if (family == "binomial") {
    data$prec = scales::rescale(data$prec, to = c(0, 1))
  } else {
    data$prec = ifelse(data$prec <= 0, runif(1, 0.0001, 0.001), data$prec)
  }

  # GAM equation
  names.vars  = setdiff(names(data), "prec")
  gam.formula = as.formula(
    paste("prec ~", paste("s(", names.vars, ")", collapse = " + ")) )

  links = switch(family,
```

```
                "gaussian" = c("identity", "log", "inverse"),
                "binomial" = c("logit", "probit", "cauchit"),
                "Gamma"    = c("inverse", "identity", "log"),
                "poisson"  = c("log", "identity", "sqrt"),
                stop("Invalid family specified"))

  model.links = list()
  # j=1
  for (j in seq_along(links)) {
    family.func = match.fun(family)
    cat(paste0("\n---------- Fitting GAM with ", family, " + ", links[j] ," link\n"))

    # Fit the GAM model
    basic.model = gam(gam.formula, data = data, method = "REML",
                      family = family.func(link = links[j]) )

    model.links[[j]] = basic.model
  }

  names(model.links) = links

  cat("\nModel link comparison:")
  df.skill = data.frame(
    model = links,
    t(sapply(model.links, function(fit) evaluation.model(data$prec, data, fit)))
  )

  metrics = names(df.skill)[-1]

  if (obj.fn == "R2") {
    best.model = df.skill[which.max(df.skill[[obj.fn]]), ]
  } else {
    best.model = df.skill[which.min(df.skill[[obj.fn]]), ]
  }

  cat("\nModel summary:\n")
  print(df.skill)
  cat(sprintf("\nBest link function: %s\n", best.model$model))
  for (metric in metrics) {
    cat(sprintf("  %s = %.3f\n", metric, best.model[[metric]]))
  }

  return(model.links[[best.model$model]])
}

# Complete regression function
# model.type = "locPoly"
# data = climaGeo
```

```r
# coords = coords
# topography = topography
# suffix.fig = "HW1/P4/FigP4c_"
# family = "gaussian"
# path = path

fitRegression = function(model.type, data, coords, topography, path = getwd(),
                         suffix.fig = NULL, family = NULL,
                         text.size = 14) {
  # figures filenames:
  figure.out  = c("i_scatter", "ii_diagnostics", "iii_loocv", "iv_boxplot",
                  "v_sptMapPred", "v_sptMapRes")
  figure.path = paste0(path, "/Figures/", suffix.fig, "FitRegression_",
                       model.type, "_model_", family, "_", figure.out, ".png")

  # 1) Fit the best regression model ----------------------------------------
  cat(paste0("\n ---------------------- Fitting ", model.type, " model\n"))
  model  = switch(model.type,
                  "Linear"  = fit_Linear( data ),
                  "GLM"     = fit_GLM(    subset.select(data), family),
                  "locPoly" = fit_locPoly(subset.select(data), family),
                  "GAM"     = fit_GAM(    subset.select(data), family),
                  stop("Unknown model type") )

  cat(paste0("\n ---------------------- Summary\n"))
  summary(model)

  # 2) Scatter fitted vs observed values ------------------------------------
  preds          = predict(model, data = data, se.fit = T)
  df.scatter     = data.frame(obs = data$prec, sim = preds$fit)

  if(model.type %in% c("locPoly", "GAM")){
    scatter        = scatter.ggplot(df.scatter, title = model.type)
  }else{
    scatter        = scatter.ggplot(df.scatter, model = model, title = model.type)
  }

  ggsave(filename = figure.path[1],
         plot     = scatter,
         width    = 9, height = 6.5, units    = "in")

  # 3) Diagnostics ----------------------------------------------------------
  # 3.1) ANOVA
  if(model.type != "locPoly"){
    anova.model = anova(model)
    cat(paste0("\n ---------------------- ANOVA\n"))
    print(anova.model)
  }else{
```

```
  anova.model = NULL
}

# 3.2) Model diagnostics
df.scatter$res   = df.scatter$obs - df.scatter$sim
diagnostics      = diagnostics.ggplot(df.scatter)
ggsave(filename  = figure.path[2],
       plot      = diagnostics,
       width     = 9, height = 6.5, units = "in")

# 4) LOOCV: Drop-One Cross Validation ----------------------------------------
cat(paste0("\n ---------------------- LOOCV\n"))
cv.preds = numeric(nrow(data))
for (ind in seq_len(nrow(data))) {
  capture.output({
    temp.model   = switch(model.type,
                          "Linear"  = fit_Linear( data[-ind, ]),
                          "GLM"     = fit_GLM(    data[-ind, ], family),
                          "locPoly" = fit_locPoly(data[-ind, ], family),
                          "GAM"     = fit_GAM(    data[-ind, ], family),
                          stop("Unknown model type"))
  })
  if(model.type != "locPoly"){
    cv.preds[ind] = predict(temp.model,
                            newdata = data[ind, , drop = FALSE],
                            type = "response")
  }else{
    cv.preds[ind] = predict(temp.model, data[ind,])
  }
}
df.loocv       = data.frame(obs = data$prec, sim = cv.preds)
loocv.scatter  = scatter.ggplot(df.loocv, title = "CV")
ggsave(filename = figure.path[3],
       plot     = loocv.scatter,
       width    = 9, height = 6.5,
       units    = "in")

# 5) 10% holdout test ----------------------------------------------------------
cat(paste0("\n ---------------------- Drop 10%\n"))
nsamples     = .5 * nrow(data) # 50% length of data
skill.drop10 = data.frame(iteration = 1:nsamples, rmse = NA, rho = NA)

for (i in 1:nsamples) {
  # randomly select 10% holdout test data
  ind.drop   = sample(1:nrow(data), size = round(0.1 * nrow(data)))
  train.data = data[-ind.drop, ]
  test.data  = data[ind.drop, ]

```

```
  capture.output({
    temp.model = switch(model.type,
                        "Linear"  = fit_Linear( train.data),
                        "GLM"     = fit_GLM(    train.data, family),
                        "locPoly" = fit_locPoly(train.data, family),
                        "GAM"     = fit_GAM(    train.data, family),
                        stop("Unknown model type") )
  })
  new.pred  = predict(temp.model, data = test.data)
  # metrics
  skill.drop10$rmse[i] = sqrt(mean((new.pred - test.data$prec)^2))
  skill.drop10$rho[i]  = cor(new.pred[ind.drop], test.data$prec, use = "pairwise.complete.
    ↪ obs")
}
skill.drop10$iteration = NULL
df.drop10 = melt(skill.drop10)
levels(df.drop10$variable) = c("RMSE (cm)", "Correlation")

boxplot.skill =
  ggplot(data = df.drop10, aes(x = value))+
  geom_boxplot(color = "#005a32", fill = "#a1d99b", outlier.size = 2, shape = 21)+
  scale_x_continuous(breaks = scales::pretty_breaks(n=3)) +
  facet_wrap(~variable, scales = "free_x") +
  theme_bw(base_family = "Times") +
  theme(axis.text.x        = element_text(size = text.size, hjust = .5, vjust = .5),
        axis.title.x       = element_blank(),
        axis.text.y        = element_blank(),
        axis.ticks.y       = element_blank(),
        strip.text         = element_text(size = text.size, face = "bold"),
        strip.background   = element_blank(),
        panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_blank())
ggsave(filename = figure.path[4],
       plot     = boxplot.skill,
       width    = 9, height = 4,
       units    = "in")


# 6) spatial Mapping  ------------------------------------------------------
cat(paste0("\n ---------------------- Spatial Mapping\n"))
map.fit     = cbind(coords, "prec" = preds$fit)
map.res     = cbind(coords, "prec" = preds$se.fit)

# save predicted values
```

```
  fit.maps       = sptMap.ggplot(map.fit, topography, title.plot = "Predicted Precipitation (
    ↪ cm)")
  ggsave(filename = figure.path[5],
         plot     = fit.maps,
         width    = 10, height = 5.5,
         units    = "in", bg = "#fff")

  res.maps       = sptMap.ggplot(map.res, topography, title.plot = "Standard Error (cm)")
  ggsave(filename = figure.path[6],
         plot     = res.maps,
         width    = 10, height = 5.5,
         units    = "in", bg = "#fff")

  # list
  return( list( model      = model,        # i) best model
                scatter    = scatter,      # ii) scatterplot of obs and sim prec + 1:1 line
                anova      = anova.model,  # iii) anova
                mod.diag   = diagnostics,  # iii) diagnostics: assumption of residuals
                df.looocv  = df.loocv,     # iv) data loocv
                cv.scatter = loocv.scatter, # iv) loocv scatter + 1:1 line
                df.drop10  = df.drop10,    # v) drop 10% obs df
                box.drop10 = boxplot.skill, # v) boxplot rmse + rho from drop 10%
                pred.sptMap = fit.maps,    # vi) spatially map for fitted values
                se.sptMap  = res.maps      # vi) spatially map for SE values
  )
  )
}

# NLS functions ------------------------------------------------------------

# Potential functions
# c0 is the nugget effect
# c1 is the sill
# a is the range parameter
exp.model = function(x, c0, c1, a) { # Exponential
  c0 + c1 * (1 - exp(-x / a))
}

sph.model = function(x, c0, c1, a) { # Spherical (x < a, geostatistics)
  c0 + c1 * (1.5 * (x / a) - 0.5 * (x / a)^3) * (x < a) + c0 + c1 * (x >= a)
}

gau.model = function(x, c0, c1, a) { # Gaussian (highly continuos data)
  c0 + c1 * (1 - exp(- (x^2) / (a^2)))
}

pow.model = function(x, c0, c1, p) { # Power (dta with no clear sill)
  if(0 < p & p < 2){
```

```
    c0 + c1 * (x^p)
  }else{
    cat("Error: Enter a valid value for p. Range 0 to 2")
  }
}

# estimate initial parameters dynamically
set.initialParams = function(data) {
  init.c0 = max(0, round(min(data$value, na.rm = TRUE), 2)) # avoid neg values
  init.c1 = max(1, round(max(data$value, na.rm = TRUE), 2) - init.c0 ) # pos range
  init.a  = max(1, round(max(data$distance, na.rm = TRUE), 2) * 0.3 )
  init.p  = 1  # power exponent default
  list(c0 = init.c0, c1 = init.c1, a = init.a, p = init.p)
}

# fit nonlinear least squares models
fit_nls = function(data, model) {
  init.params = set.initialParams(data)

  if (any(is.na(init.params)) || any(!is.finite(unlist(init.params)))) {
    return(NULL)
  }

  if(identical(model, pow.model)){ # pow model with p param
    tryCatch({
      temp.nls   = nlsLM(
        value    ~ model(distance, c0, c1, p),
        data     = data,
        start    = list(c0 = init.params$c0,
                        c1 = init.params$c1,
                        p  = init.params$p),
        control  = nls.lm.control(maxiter = 2000)
      )
      # skill.nls  = evaluation.model(data$value, data, temp.nls)
      # pred.nls   = predict(temp.nls)
      temp.nls
    }, error = function(e) { return(NULL) })
  } else{ # e.g., exp, gaussian models with a param
    tryCatch({
      temp.nls   = nlsLM(
        value    ~ model(distance, c0, c1, a),
        data     = data,
        start    = list(c0 = init.params$c0,
                        c1 = init.params$c1,
                        a  = init.params$a),
        control  = nls.lm.control(maxiter = 2000)
      )
      # skill.nls  = evaluation.model(data$value, data, temp.nls)
```

```r
    # pred.nls   = predict(temp.nls)
    temp.nls
  }, error = function(e) { return(NULL) })

 } # end model condition

}

# Hierarchical Spatial Model ----------------------------------------------------
empirical.vgram = function(res, data, type, n.bins = 50) {
 emp     = vgram(loc = data[, c("lon", "lat")], y = res, type = type, lon.lat = TRUE)
 emp.bin = getVGMean(x = emp, breaks = pretty(emp$d, n = n.bins, eps.correct = 1))
 data.frame(
  distance = emp.bin$centers[-length(emp.bin$centers)],
  value    = emp.bin$ys[-length(emp.bin$ys)],
  type     = type
 )
}

fit_HSM        = function(data, coords, topography,
                          obj.fn = "BIC", type = "variogram", # covariogram or correlogram
                          path = getwd(), model.type = "HSM",
                          suffix.fig = NULL, family = NULL,
                          text.size = 14) {
 options(warn = -1)
 # figures filenames:
 figure.out  = c("0_emp_binned", "i_scatter", "ii_diagnostics", "iii_loocv", "iv_boxplot",
                 "v_sptMapPred", "v_sptMapRes")
 figure.path = paste0(path, "/Figures/", suffix.fig, "FitRegression_",
                      model.type, "_model_", family, "_", figure.out, ".png")

 if (!"prec" %in% names(data)) {
  stop("Error: The response variable 'prec' is missing in the dataset.")
 }

 # 1) fit lm model ---------------------------------------------------------
 lm.model = fit_Linear(data)
 lm.res   = residuals(lm.model)
 data$res = lm.res

 # 2) Binned empirical Var, Cov, Cor  --------------------------------------
 empirical.data = empirical.vgram(data$res, data, type)
 empirical.data = empirical.data[empirical.data$distance < .5*max(empirical.data$distance),
   ↪ ]
 empirical.data = empirical.data %>% filter(!is.na(value) & is.finite(value))
 bin.type       = unique(empirical.data$type)
 empirical.data$type = factor(empirical.data$type,
                              levels =  bin.type)
```

```r

# 3) Fitting with NLS ------------------------------------------------------
pot.func       = c("exp", "spherical", "gaussian")
fit.mod        = list(exp.model, sph.model, gau.model)
names(fit.mod) = pot.func
model.nls      = list()
# j=1
for (j in seq_along(pot.func)) {
  cat(paste0("\n------------ Fitting HSM with ", pot.func[j] ," potential function \n"))
  basic.model    = fit_nls(empirical.data, fit.mod[[j]])
  model.nls[[j]] = basic.model
}
names(model.nls) = pot.func

# 4) NLS comparison --------------------------------------------------------
cat("\nModel link comparison:")
df.skill = data.frame(
  model  = pot.func,
  t(sapply(model.nls, function(fit) evaluation.model(empirical.data$value,
                                                     empirical.data, fit)))
)

metrics = names(df.skill)[-1]

if (obj.fn == "R2") {
  best.model = df.skill[which.max(df.skill[[obj.fn]]), ]
} else {
  best.model = df.skill[which.min(df.skill[[obj.fn]]), ]
}

cat("\nModel summary:\n")
print(df.skill)
cat(sprintf("\nBest potential function: %s\n", best.model$model))
for (metric in metrics) {
  cat(sprintf("  %s = %.3f\n", metric, best.model[[metric]]))
}

best.skill  = as.data.frame(t(best.model[,-1]))
colnames(best.skill) = "value"
best.skill$metric = row.names(best.skill)

skill.text  = paste0(best.skill$metric, ": ", round(best.skill$value, 2), collapse = "\n")
coefs       = coef(model.nls[[best.model$model]])
eq.text     = paste0("Sim~ == ", round(coefs[1], 2), "~")
if (length(coefs) > 1) {
  terms.text = paste0(round(coefs[-1], 1), "%*%", names(coefs)[-1], collapse = " + ")
  eq.text    = paste(eq.text, "+", terms.text)
}
```

```
987  empirical.data$pred = predict(model.nls[[best.model$model]], data = empirical.data)
988
989  # Plot empirical data + fit model
990  emp.plot =
991    ggplot(empirical.data) +
992    geom_abline(slope = 0, intercept=0, color = "#000",linetype = 2, linewidth = .5) +
993    # empirical data
994    geom_line(aes(x = distance, y = value), color = "#08306b", linewidth = .9, alpha = .7)
       ↪ +
995    geom_point(aes(x = distance, y = value), color = "#08306b", fill= "#6baed6",
996             size = 2, shape = 21) +
997    # predicted data
998    geom_line(aes(x = distance, y = pred), color="#67000d", linewidth = 1, linetype = "
       ↪ dashed") +
999    scale_x_continuous(breaks = pretty_breaks(n = 3)) +
1000   scale_y_continuous(breaks = pretty_breaks(n = 3)) +
1001   labs(x = "Distance", y = "Value") +
1002   facet_wrap(~type, scales = "free_y", labeller = as_labeller(c(
1003     variogram   = "Binned Empirical Variogram",
1004     correlogram = "Binned Empirical Correlogram",
1005     covariogram = "Binned Empirical Covariance"
1006   ))) +
1007   theme_bw(base_family = "Times") +
1008   theme(axis.text.x        = element_text(size = text.size, vjust = 0.5),
1009         axis.text.y        = element_text(size = text.size, angle = 90, hjust = .5, vjust = .5),
1010         axis.title         = element_text(size = text.size + 2, face = 'bold'),
1011         legend.position    = "none",
1012         strip.background   = element_blank(),
1013         strip.text         = element_text(size = text.size + 2, face = 'bold'),
1014         panel.border       = element_rect(color = "#000000", fill = NA, linewidth = .5),
1015         panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
1016         panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
1017         panel.grid.major.x = element_blank(),
1018         panel.grid.major.y = element_blank())
1019 # model performance
1020 if(type=="variogram"){
1021   emp.plot = emp.plot +
1022     annotate("text", x = .1*max(empirical.data$distance),   y = .95*max(empirical.data$
       ↪ value),
1023              label = skill.text, hjust = 0, vjust = 1, size = 4.5, fontface = "bold")
1024 }else{
1025   emp.plot = emp.plot +
1026     annotate("text", x = .85*max(empirical.data$distance),   y = .95*max(empirical.data$
       ↪ value),
1027              label = skill.text, hjust = 0, vjust = 1, size = 4.5, fontface = "bold")
1028 }
1029
1030 ggsave(filename = figure.path[1],
```

```
      plot     = emp.plot,
      width    = 9, height = 6.5, units    = "in")

# 5) Kriging on the residuals ------------------------------------------------
model      = model.nls[[best.model$model]]
params.fit = as.numeric(coef(model))
## Check with Krig and predict.krig
krig.fit   = Krig(x = data,
                  Y = data$prec,
                  theta  = params.fit[3],
                  lambda = params.fit[1]/params.fit[2],
                  m=1)
data$krig.fit = predict.Krig(krig.fit)
data$krig.se  = predictSE(krig.fit)

# 6) Scatter fitted vs observed values ---------------------------------------
df.scatter      = data.frame(obs = data$prec, sim = data$krig.fit)
scatter         = scatter.ggplot(df.scatter, model = model, title = model.type)
ggsave(filename = figure.path[2],
      plot     = scatter,
      width    = 9, height = 6.5, units    = "in")

# 7) Diagnostics -------------------------------------------------------------
df.scatter$res   = df.scatter$obs - df.scatter$sim
diagnostics      = diagnostics.ggplot(df.scatter)
ggsave(filename  = figure.path[3],
      plot      = diagnostics,
      width     = 9, height = 6.5, units = "in")

# 8) LOOCV: Drop-One Cross Validation ----------------------------------------
cat(paste0("\n ---------------------- LOOCV\n"))
cv.preds = numeric(nrow(data))
for (ind in seq_len(nrow(data))) {
  capture.output({
    temp.model   = Krig(x = data[-ind,],
                        Y = data$prec[-ind],
                        theta  = params.fit[3],
                        lambda = params.fit[1]/params.fit[2],
                        m=1)
  })
  cv.preds[ind] = predict.Krig(temp.model,
                           newdata = data[ind, , drop = FALSE],
                           type = "response")

}
df.loocv       = data.frame(obs = data$prec, sim = cv.preds)
loocv.scatter  = scatter.ggplot(df.loocv, title = "CV")
ggsave(filename = figure.path[4],
```

```
      plot     = loocv.scatter,
      width    = 9, height = 6.5,
      units    = "in")

# 9) 10% holdout test ----------------------------------------------------
cat(paste0("\n ---------------------- Drop 10%\n"))
nsamples     = .5 * nrow(data) # 50% length of data
skill.drop10 = data.frame(iteration = 1:nsamples, rmse = NA, rho = NA)

for (i in 1:nsamples) {
  # randomly select 10% holdout test data
  ind.drop   = sample(1:nrow(data), size = round(0.1 * nrow(data)))
  train.data = data[-ind.drop, ]
  test.data  = data[ind.drop, ]

  capture.output({
    temp.model = Krig(x = train.data,
                      Y = train.data$prec,
                      theta  = params.fit[3],
                      lambda = params.fit[1]/params.fit[2],
                      m=1)
  })
  new.pred  = predict.Krig(temp.model, data = test.data)
  # metrics
  skill.drop10$rmse[i] = sqrt(mean((new.pred - test.data$prec)^2))
  skill.drop10$rho[i]  = cor(new.pred[ind.drop], test.data$prec, use = "pairwise.complete.
   ↪ obs")
}
skill.drop10$iteration = NULL
df.drop10 = melt(skill.drop10)
levels(df.drop10$variable) = c("RMSE (cm)", "Correlation")

boxplot.skill =
  ggplot(data = df.drop10, aes(x = value))+
  geom_boxplot(color = "#005a32", fill = "#a1d99b", outlier.size = 2, shape = 21)+
  scale_x_continuous(breaks = scales::pretty_breaks(n=3)) +
  facet_wrap(~variable, scales = "free_x") +
  theme_bw(base_family = "Times") +
  theme(axis.text.x       = element_text(size = text.size, hjust = .5, vjust = .5),
        axis.title.x      = element_blank(),
        axis.text.y       = element_blank(),
        axis.ticks.y      = element_blank(),
        strip.text        = element_text(size = text.size, face = "bold"),
        strip.background  = element_blank(),
        panel.border      = element_rect(color = "#000000", fill = NA, linewidth = .5),
        panel.grid.minor.x = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.minor.y = element_line(color = "#d9d9d9", linetype = "dotted"),
        panel.grid.major.x = element_blank(),
```

```
        panel.grid.major.y = element_blank())
  ggsave(filename = figure.path[5],
         plot     = boxplot.skill,
         width    = 9, height = 4,
         units    = "in")

  # 10) spatial Mapping  ------------------------------------------------------
  cat(paste0("\n ---------------------- Spatial Mapping\n"))
  map.fit     = cbind(coords, "prec" = data$krig.fit)
  map.res     = cbind(coords, "prec" = data$krig.se)

  # save predicted values
  fit.maps       = sptMap.ggplot(map.fit, topography, title.plot = "Predicted Precipitation (
    ↪ cm)")
  ggsave(filename = figure.path[6],
         plot     = fit.maps,
         width    = 10, height = 5.5,
         units    = "in", bg = "#fff")

  res.maps       = sptMap.ggplot(map.res, topography, title.plot = "Standard Error (cm)")
  ggsave(filename = figure.path[7],
         plot     = res.maps,
         width    = 10, height = 5.5,
         units    = "in", bg = "#fff")

  # list
  return( list( emp.plot = emp.plot, # binned empirical plot.
                model       = model,         # i) best model
                scatter     = scatter,       # ii) scatterplot of obs and sim prec + 1:1 line
                mod.diag    = diagnostics,   # iii) diagnostics: assumption of residuals
                df.looocv   = df.loocv,      # iv) data loocv
                cv.scatter  = loocv.scatter, # iv) loocv scatter + 1:1 line
                df.drop10   = df.drop10,     # v) drop 10% obs df
                box.drop10  = boxplot.skill, # v) boxplot rmse + rho from drop 10%
                pred.sptMap = fit.maps,      # vi) spatially map for fitted values
                se.sptMap   = res.maps       # vi) spatially map for SE values
  )
  )
}
```

C.2. R code

```
# Code details ------------------------------------------------------------
# author: Catalina Jerez
# mail: catalina.jerez@colorado.edu
# Course: Advanced Data Analysis Techniques
# Code: CVEN 6833
```

```r
# Homework #1: Problem 2c-5 fit models linear, glm, gam
# Department of Civil, Environmental and Architectural Engineering
# College of Engineering & Applied Science
# University of Colorado Boulder

# Library ----------------------------------------------------------------------
gc()
rm(list = ls())

# Load necessary libraries
spatialAnalysis.Lib     = c("sp", "sf","geosphere", "geoR")
smoothing.Lib           = c("mgcv", "locfit", "akima")
statisticalAnalysis.Lib = c("MASS", "psych", "hydroGOF", "fields", "leaps", "car")
dataManipulation.Lib    = c("dplyr", "reshape2", "reshape", "tidyr")
dataVisualization.Lib   = c("ggplot2", "GGally", "ggpubr",
                            "scales","maps",
                            "rnaturalearth", "rnaturalearthdata")
list.packages           = unique(c( spatialAnalysis.Lib, smoothing.Lib,
                                    statisticalAnalysis.Lib, dataManipulation.Lib,
                                    dataVisualization.Lib ))

# Load all required packages
sapply(list.packages, require, character.only = TRUE)

# Path -------------------------------------------------------------------------
path      = "/Users/caje3244/OneDrive - UCB-O365/2025 Spring/CVEN 6833/"
path.code = file.path(path, "Codes/")
path.plot = file.path(path, "Figures/")

# load functions to perform regression
source(file = file.path(path.code, "HW1_library.R"))

# Data -------------------------------------------------------------------------
topography = read.table("http://civil.colorado.edu/~balajir/CVEN6833/HWs/HW-1/india-
    ↪ grid-topo.txt")
colnames(topography)    = c("lon", "lat", "elev")

# new climatol data with calculated distances from Geodesic method
climaGeo = readRDS(file = paste0(path.code, "RData/hw1_climatol_distGeodesic.rds"))
climaGeo = na.omit(climaGeo)
climaGeo$prec = climaGeo$prec/10 # mm to cm
coords = dplyr::select(climaGeo, c(lon, lat))
# scale all vars dependent of lon and lat
climaGeo = climaGeo %>%
  dplyr::mutate(across(!c(prec), scale, center = TRUE, scale = TRUE)) %>%
  dplyr::mutate(across(!c(prec), as.numeric))

scale.color  = c("#d73027","#313695")
```

```r
scale.label  = unique(cut_interval(climaGeo$prec, n=2))
scatter.prec = pairs.ggplot(climaGeo)

# ggsave(filename = file.path(path.plot, "HW1_FigureP2c_Scatterplot_v0.png"),
#        plot     = scatter.prec,
#        width    = 12, height = 7.5, units = "in")

# Basic analysis ----------------------------------------------------------------
set.seed(1113)
# v0) basic linear model
str(data)
basic.model = lm(prec ~ ., data = climaGeo)
# If the VIF is larger than 1/(1-R2)
vif(basic.model) > (1/(1-summary(basic.model)$r.squared)) # check multicolinearity

cat("Since for some variables VIF is larger than 1/(1-R2), let's remove:
mdArabian and mdBengal and just keep the mdCoast (minimun distance to any coast) \n")

# v1) linear model
climaGeo    = dplyr::select(climaGeo, -c(mdArabian, mdBengal))
basic.model = lm(prec ~ ., data = climaGeo)
vif(basic.model) > (1/(1-summary(basic.model)$r.squared)) # check multicolinearity

scatter.prec    = pairs.ggplot(climaGeo)
# ggsave(filename = file.path(path.plot, "HW1_FigureP2c_Scatterplot_v1.png"),
#        plot     = scatter.prec,
#        width    = 12, height = 7.5, units = "in")

# v2) basic linear + interacterions
cat("Cool, for all the variables VIF < 1/(1-R2)
Let's add some interactiones between the variables \n")

climaGeo    = climaGeo %>%
  dplyr::mutate(lld  = lon * lat * mdCoast,
                ll2d = lon * lat^2 * mdCoast,
                l2ld = lon^2 * lat * mdCoast,
                ed   = elev * mdCoast
                )

basic.model = lm(prec ~ ., data = climaGeo)
vif(basic.model) > (1/(1-summary(basic.model)$r.squared)) # check multicolinearity

scatter.prec    = pairs.ggplot(climaGeo)
# ggsave(filename = file.path(path.plot, "HW1_FigureP2c_Scatterplot_v2.png"),
#        plot     = scatter.prec,
#        width    = 12, height = 7.5, units = "in")

cat("Since for all the variables VIF < 1/(1-R2)
```

```
Let's proceed with other regression models \n")

# Fitting linear model ----------------------------------------------------
fitting.linear = fitRegression(model.type = "Linear", data = climaGeo,
                           coords = coords, topography = topography,
                           suffix.fig = "HW1/P2/FigP2c_",
                           path = path)
names(fitting.linear)
fitting.linear$anova
fitting.linear$box.drop10
# Fitting GLM model -------------------------------------------------------
fitting.glm_Gau = fitRegression(model.type = "GLM", data = climaGeo,
                             coords = coords, topography = topography,
                             suffix.fig = "HW1/P3/FigP3c_", family = "gaussian",
                             path = path)
fitting.glm_Gau$anova
fitting.glm_Gau$se.sptMap

fitting.glm_Bin = fitRegression(model.type = "GLM", data = climaGeo,
                            coords = coords, topography = topography,
                            suffix.fig = "HW1/P3/FigP3c_", family = "binomial",
                            path = path)
fitting.glm_Bin$se.sptMap
fitting.glm_Bin$anova
fitting.glm_Gamma = fitRegression(model.type = "GLM", data = climaGeo,
                             coords = coords, topography = topography,
                             suffix.fig = "HW1/P3/FigP3c_", family = "Gamma",
                             path = path)
fitting.glm_Gamma$scatter
fitting.glm_Gamma$cv.scatter

# Fitting Local Polynomial ------------------------------------------------
fitting.loc_Gau = fitRegression(model.type = "locPoly", data = climaGeo,
                             coords = coords, topography = topography,
                             suffix.fig = "HW1/P4/FigP4c_", family = "gaussian",
                             path = path)

fitting.loc_Gamma = fitRegression(model.type = "locPoly", data = climaGeo,
                             coords = coords, topography = topography,
                             suffix.fig = "HW1/P4/FigP4c_", family = "gamma",
                             path = path)
fitting.loc_Geom = fitRegression(model.type = "locPoly", data = climaGeo,
                             coords = coords, topography = topography,
                             suffix.fig = "HW1/P4/FigP4c_", family = "geom",
                             path = path)
fitting.loc_Geom = fitRegression(model.type = "locPoly", data = climaGeo,
                            coords = coords, topography = topography,
                            suffix.fig = "HW1/P4/FigP4c_", family = "poisson",
```

```
                          path = path)
# Fitting GAM ---------------------------------------------------------------
fitting.GAM_Gau  = fitRegression(model.type = "GAM", data = climaGeo,
                           coords = coords, topography = topography,
                           suffix.fig = "HW1/P5/FigP5c_", family = "gaussian",
                           path = path)
fitting.GAM_Gau$anova

fitting.GAM_Bin  = fitRegression(model.type = "GAM", data = climaGeo,
                          coords = coords, topography = topography,
                          suffix.fig = "HW1/P5/FigP5c_", family = "binomial",
                          path = path)
fitting.GAM_Gamma= fitRegression(model.type = "GAM", data = climaGeo,
                          coords = coords, topography = topography,
                          suffix.fig = "HW1/P5/FigP5c_", family = "Gamma",
                          path = path)

# Fitting HSM ---------------------------------------------------------------
# Hierarchical Spatial Model
fitting.HSM = fit_HSM(data = climaGeo, coords, topography,
                  type = "variogram", # covariogram or correlogram
                  suffix.fig = "HW1/P6/FigP6_", path = path)
fitting.HSM$se.sptMap
fitting.HSM$scatter
```