STA 5377 Midterm Exam

Carson Slater

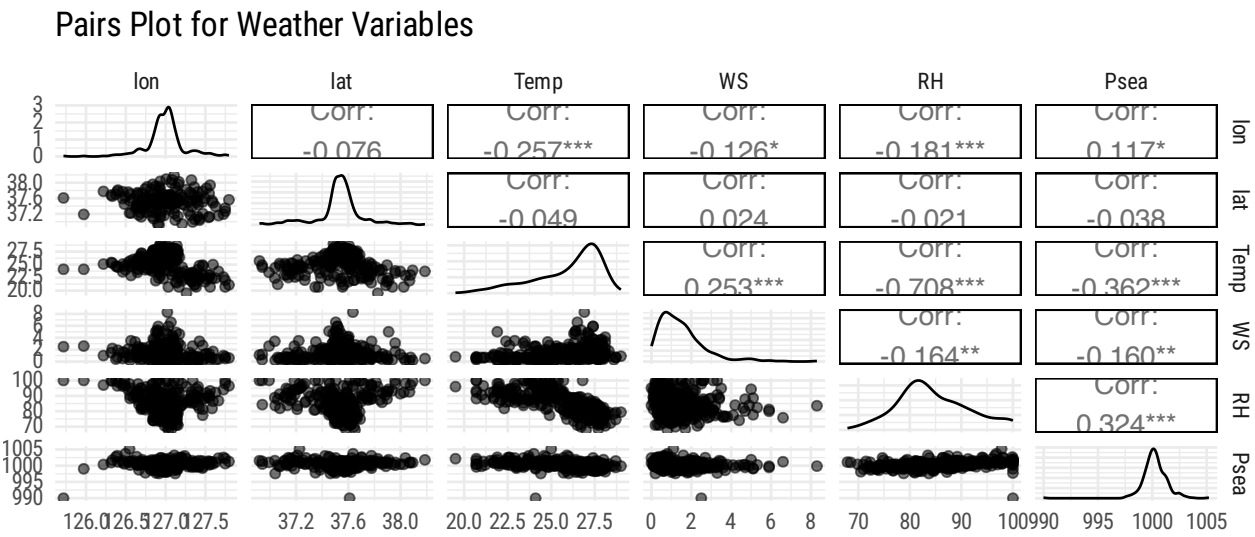
March 26, 2025

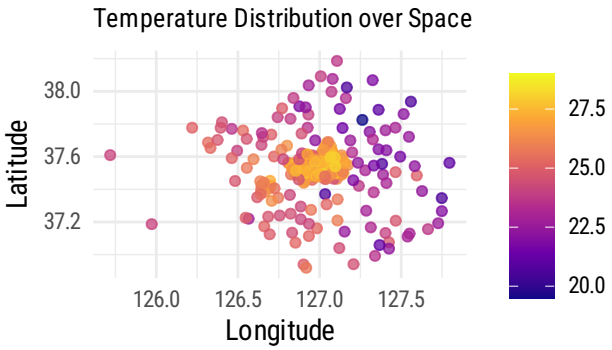
1

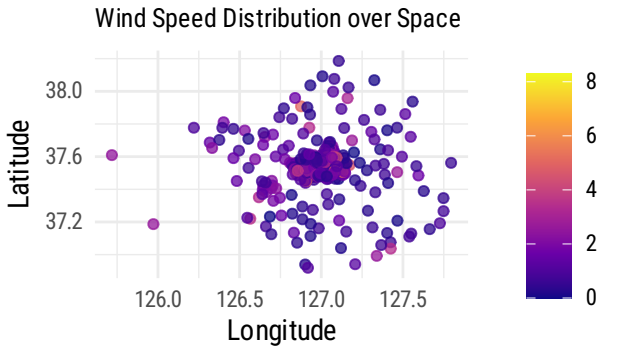
(10 pts) Perform exploratory data analysis and outlier detection for each weather variable. After addressing issues, such as missing values and outliers, generate a new data to be used in the subsequent analysis, if necessary.

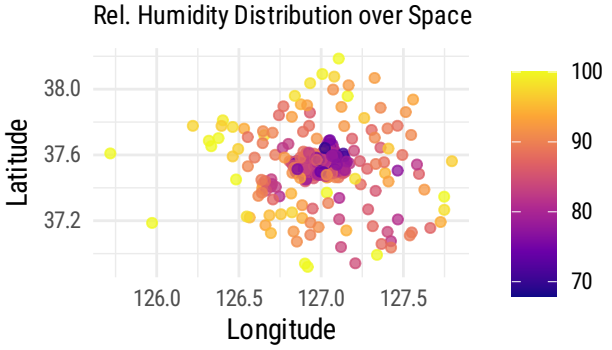
As stated in the exam instructions, *"The primary objective of the study is to determine how these weather variables are related each other, to study their spatial structures, and to choose an optimal interpolation technique of each variable."*

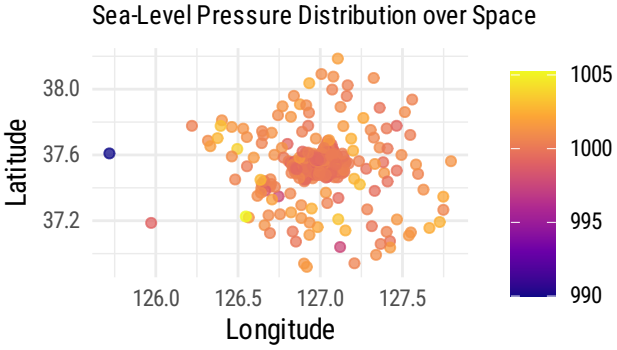
Before doing anything, we want to consider the shape of the data, and an easy way to do this is by using a pairs plot. Note that this is simply doing a similar thing to `plot.geodata()`, except that in this way `ggplot` is able to be used. There is no missing data in these observations, so no action must be taken to handle that.







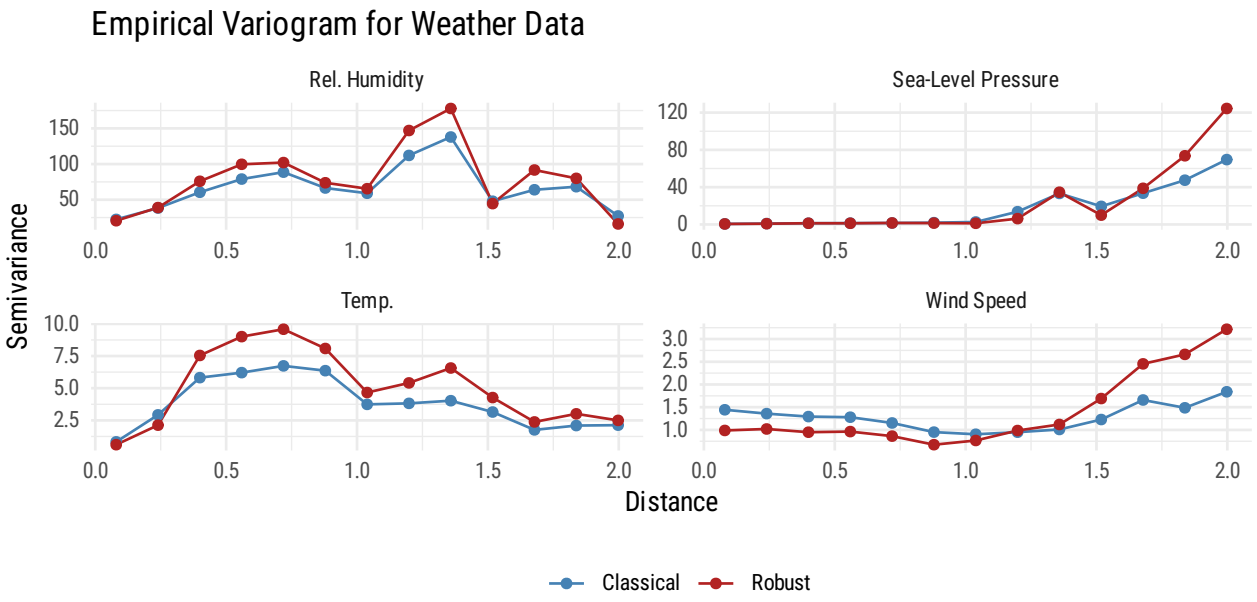




Taking a look at these data, there appears to be a few stations that are far west (Station IDs 11110 and 11176), that do not seem to be very close to the rest of the stations. The only interesting thing to note here is that these values do have 100% relative humidity, which is higher than all other stations in the dataset, otherwise thse values do not seem to be extreme. We opt to not consider those observations outliers.

As for the spatial distribution of temperature, there appears to be higher values in the center of the spatial process, with lower values strewn throughout the outer regions. The spatial distribution of wind speed seems to be very uniform across the process. The spatial distribution of humidity seems to be the inverse of temperature, with lower values in the middle and higher values on the outskirts. Lastly, sea-level pressure seems to be fairly consistent around 1000, with some values being higher around the outer ring of the spatial process.

Lastly, both the classical and robust empirical variograms shown below do seem to be sufficiently close enough to give confidence that there are no outliers confounding these data.



2

(15 pts) For this part of the problem only, ignore any possible spatial correlation in the data. Perform regression analysis to investigate potential trends in the data using the following models:

- $Temp \sim WS + RH + Psea$
- $WS \sim Temp + RH + Psea$
- $RH \sim WS + Temp + Psea$
- $Psea \sim WS + RH + Temp$

Examine the residuals and summarize your findings. The analysis will be helpful to model trend in the subsequent analysis.

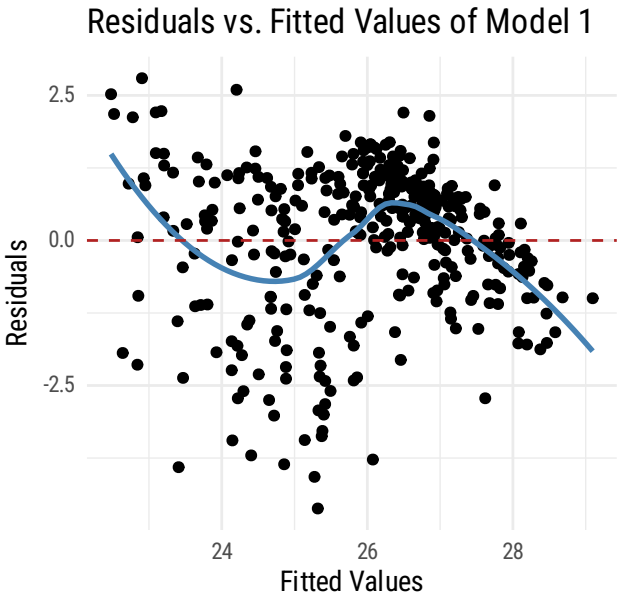
We proceed to fit the given models and explore the findings, ignoring the possibility of spatial correlations.

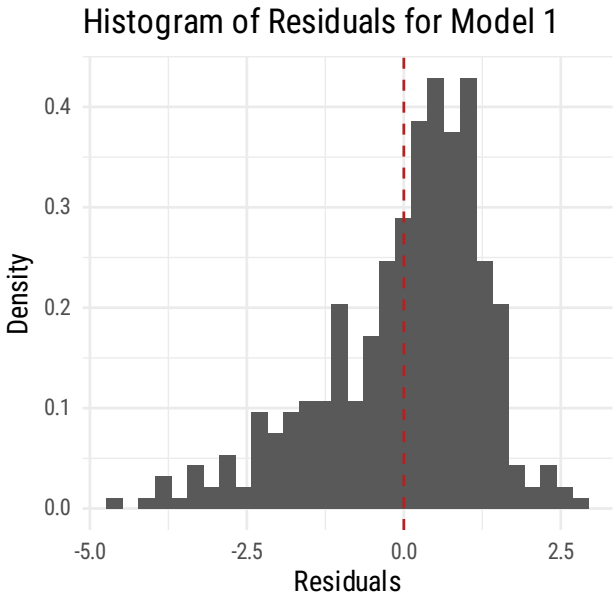
2.1 Model 1: $Temp \sim WS + RH + Psea$

Below is the table including the fitted model and also a residual diagnostic plot.

Characteristic	Beta	95% CI[1]	p-value
WS	0.21	0.09, 0.33	<0.001
RH	-0.18	-0.20, -0.16	<0.001
Psea	-0.22	-0.35, -0.10	<0.001

[1] CI = Confidence Interval





For this first model, $Temp \sim WS + RH + Psea$, all of the coefficients estimates are statistically significant. The coefficient estimate for wind speed is positive, while the ones for relative humidity and sea-level pressure are negative. Taking a look at diagnostic plots for the residuals, there seems to be some patterns in the data that are not being captured by all of the variables in the model. The distribution of the model residuals seems to be skewed left, and the Shapiro-Wilk normality test rejects the null hypotheses that these residuals come from a normal distribution (p-value = 6.72e-11). The model coefficients estimates, standard error estimates, and their p-values should be taken with a grain of salt, since there is reason to believe that key assumptions such as homoscedasticity are violated in this model. This would lead us to suspect that the p-values are a little too low, as the standard errors for the coefficient estimates are possibly underestimated. Lastly, all of the variance inflation factors are below two, so multicollinearity in not likely an issue.

2.2 Model 2: $WS \sim Temp + RH + Psea$

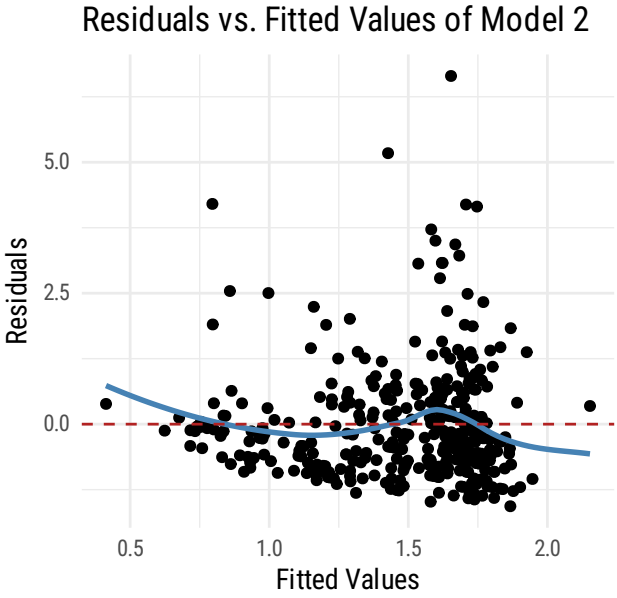
Below is the table including the fitted model and also a residual diagnostic plot.

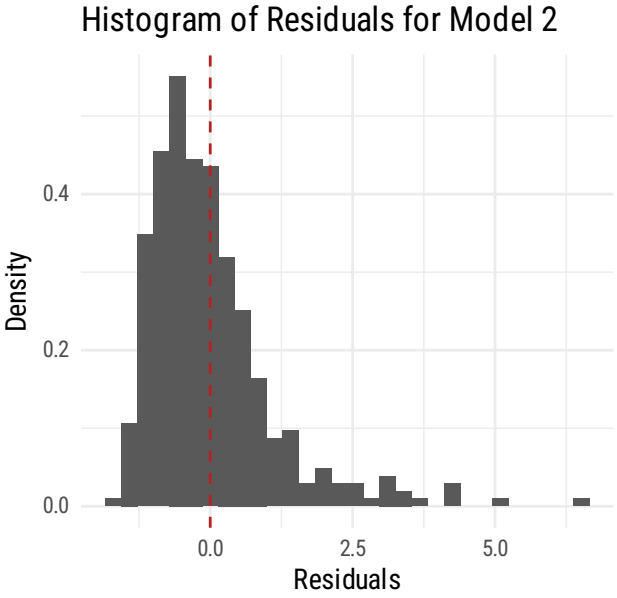
Characteristic	Beta	95% CI[1]	p-value
Temp	0.15	0.07, 0.24	<0.001
RH	0.01	-0.02, 0.03	0.6
Psea	-0.08	-0.19, 0.03	0.14

[1] CI = Confidence Interval

For this second model, $WS \sim Temp + RH + Psea$, only the temperature coefficient estimate is statistically significant, and positive. Taking a look at diagnostic plots for the residuals, there also seems to be some

patterns in the data that are not being captured by all of the variables in the model. The distribution of the model residuals seems to be skewed right, and the Shapiro-Wilk normality test rejects the null hypotheses that these residuals come from a normal distribution (p-value < 2.2e-16). There is an obvious reason to believe that key assumptions such as homoscedasticity are violated in this model, as the residuals versus fitted values plot shows a 'fanning out' of the residuals across the fitted values, lending to the idea of non-constant variance. Lastly, all of the variance inflation factors are right around two, so multicollinearity in not likely an issue in this model.



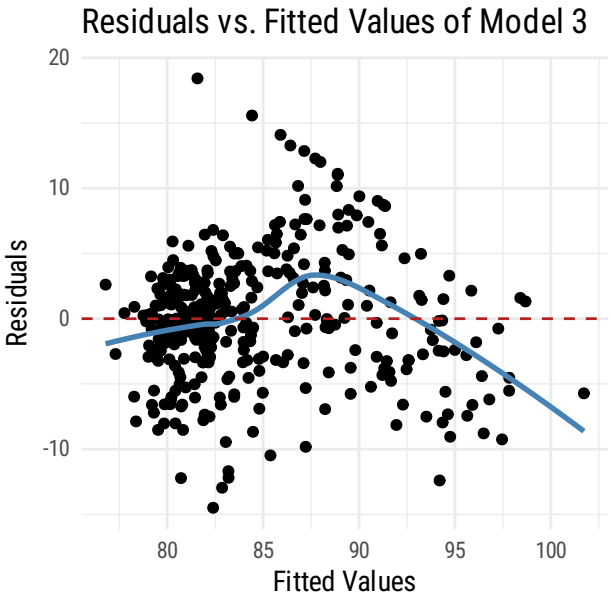


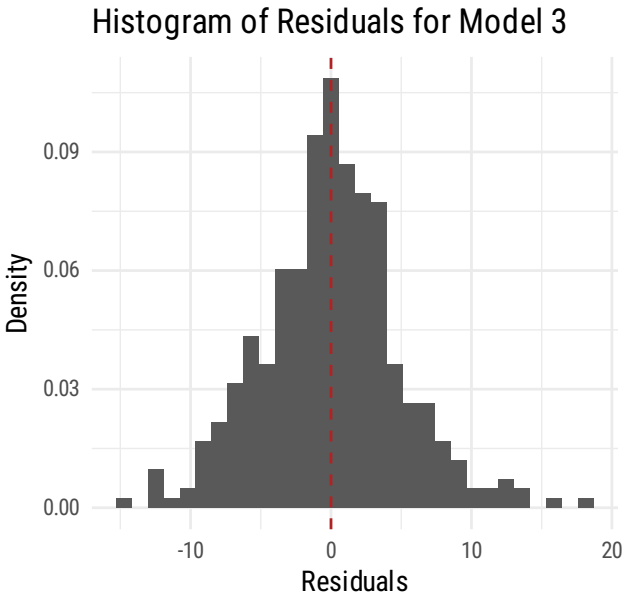
2.3 Model 3: $RH \sim WS + Temp + Psea$

Below is the table including the fitted model and also a residual diagnostic plot.

Characteristic	Beta	95% CI[1]	p-value
WS	0.13	-0.32, 0.59	0.6
Temp	-2.5	-2.8, -2.2	<0.001
Psea	0.49	0.01, 0.97	0.044

[1] CI = Confidence Interval





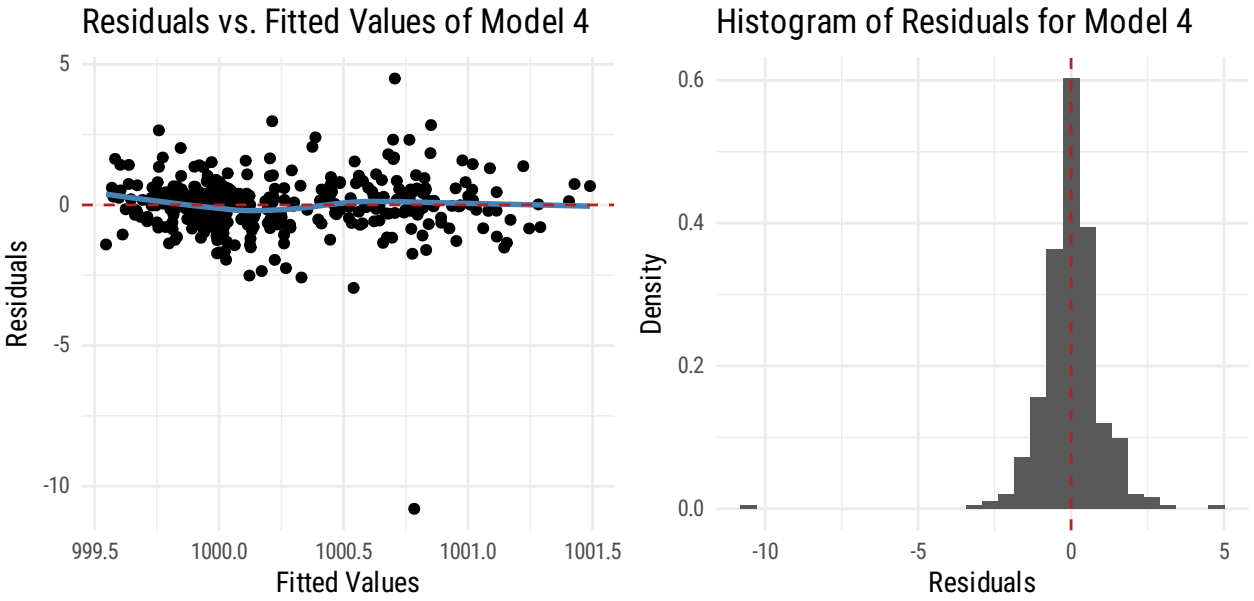
For this third model, $RH \sim WS + Temp + Psea$, only the temperature and sea-level pressure coefficient estimates are statistically significant. The coefficient estimate on temperature is negative and coefficient estimate on sea-level pressure is positive. Taking a look at diagnostic plots for the residuals, there also seems to be some patterns in the data that are not being captured by all of the variables in the model. The distribution of the model residuals seems to be shaped like a normal bell-curve; however the Shapiro-Wilk normality test also rejects the null hypotheses that these residuals come from a normal distribution (p-value < 0.01386). This test is known to be over-powered when the sample sizes are sufficiently large enough. Due to the shape of the histogram, we elect to conclude that the residuals reasonably resemble a normal distribution. It appears from the residual diagnostic plot that the assumption of homoscedasticity is violated in this model. There also appears to be patterns in the response that are not accounted for in the predictors, which could lend to the idea that the model is incorrectly specified. Lastly, all of the variance inflation factors are right around two, so multicollinearity in not likely an issue in this model.

2.4 Model 4: $Psea \sim WS + RH + Temp$

Below is the table including the fitted model and also a residual diagnostic plot.

Characteristic	**Beta**	**95% CI**[1]	**p-value**
WS	-0.07	-0.17, 0.02	0.14
RH	0.02	0.00, 0.04	0.044
Temp	-0.15	-0.23, -0.06	<0.001

[1] CI = Confidence Interval



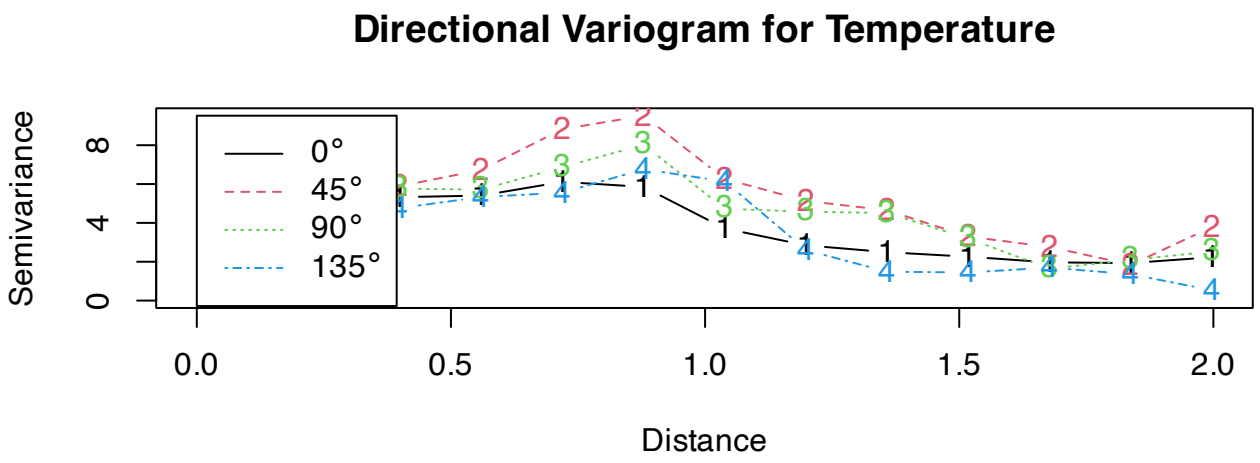
For this fourth model, $Psea \sim WS + RH + Temp$, only the temperature and sea-level pressure coefficient estimates are statistically significant, and positive. Taking a look at diagnostic plots for the residuals, this is the only model that seems to have residuals that do not exhibit any patterns in the data residuals. The distribution of the model residuals seems to be shaped like a normal bell-curve; however the Shapiro-Wilk normality test also rejects the null hypotheses that these residuals come from a normal distribution (p-value $<$ 2.2e-16). This test is known to be over-powered when the sample sizes are sufficiently large enough. Due to the shape of the histogram, we elect to conclude that the residuals reasonably resemble a normal distribution. Of all four models, this is the model that appears to have the nicest behaving residuals. Lastly, all of the variance inflation factors are right around two, so multicollinearity in not likely an issue in this model. Of all the models, this one likely has the most reliable coefficient estimates.

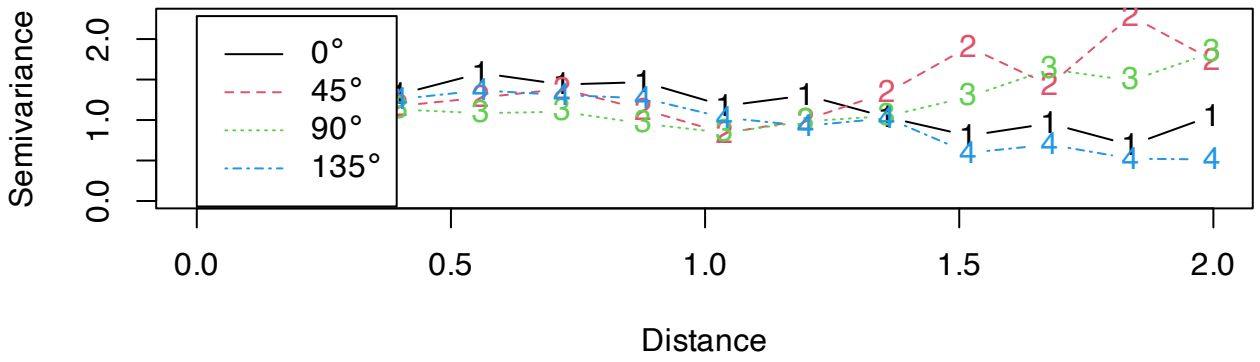
3

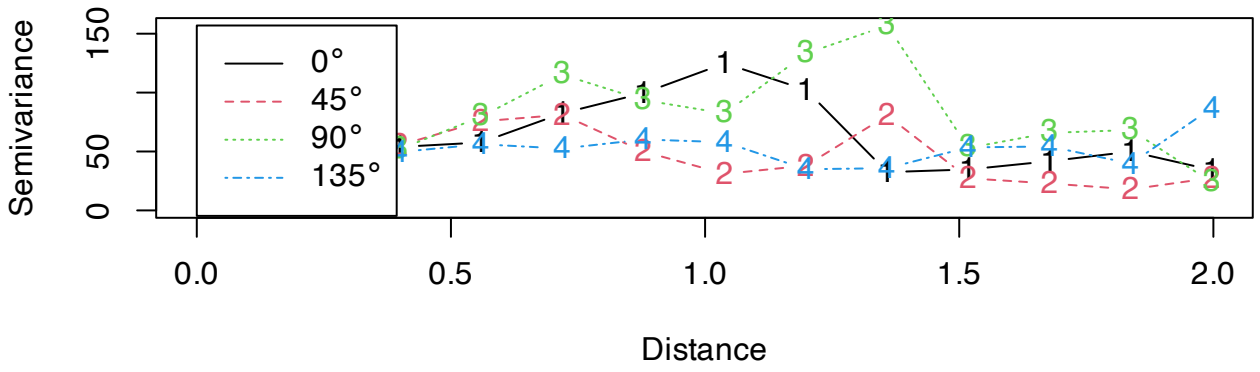
(20 pts) For each weather variable,

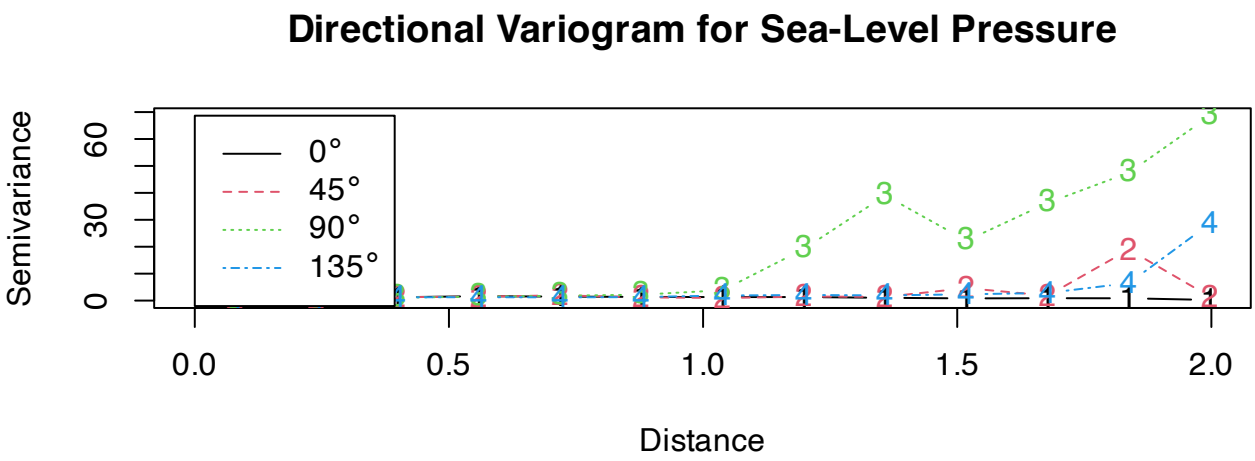
- **Check that the underlying spatial process for each variable is isotropy or anisotropy.**
- **Hereafter, for each variable, assume that the spatial process is isotropy and conduct variogram analysis to figure out spatial structure using appropriate variogram estimates and two theoretical variogram models (spherical and exponential).**

Despite an attempt to fit directional variograms and plot them in `ggplot2`, we broke graphics uniformity and used `base` plotting. For each variable in the data, we fit a directional variogram, as shown below.


Directional Variogram for Temperature
Semivariance
0
4
8
0°
45°
90°
135°
0.0
0.5
1.0
1.5
2.0
Distance


Directional Variogram for Wind Speed
Semivariance
0.0
1.0
2.0
0°
45°
90°
135°
0.0
0.5
1.0
1.5
2.0
Distance


Directional Variogram for Rel. Humidity
Semivariance
0
50
150
0°
45°
90°
135°
0.0
0.5
1.0
1.5
2.0
Distance



A good way to check for anisotropy is if directional empirical variograms are observed to look vastly different from one another. Generally speaking, the directional variograms seem to be following similar patterns for all variables. The range for each variable does not seem vastly different across directions. The one exception would be the west-ward variogram (90 degrees) for the `Psea` variable. It seems to jump up really high around the distance 1.2, and continues to drift upward unlike the rest of the other directional variograms for `Psea`. In light of this, we consider these data to be isotropic.

We now proceed to fit some variogram models to the data, assuming they are isotropic. There appears to be significant relationships with each weather variable and longtide and least, which would mean our data might not be second-order stationary. We do not exactly correct for this, as we were not sure how to when fitting our variogram models.

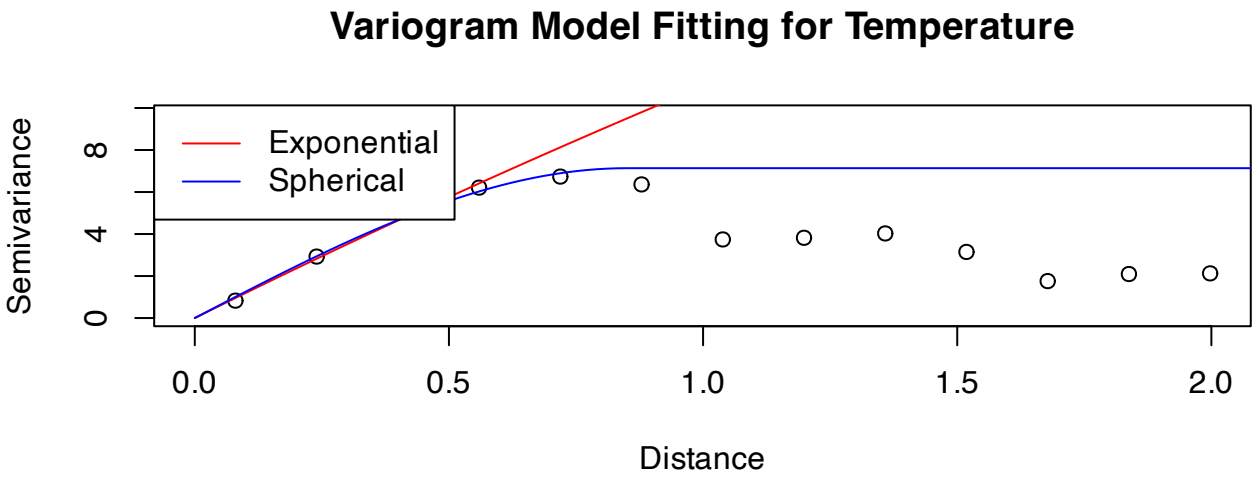


Table 5: Variogram fit estimates for Temperature

Model	Nugget	Partial Sill	Range	WSS
Exponential	0	68.728278	5.7136029	2357.149
Spherical	0	7.134591	0.8495602	1807.923

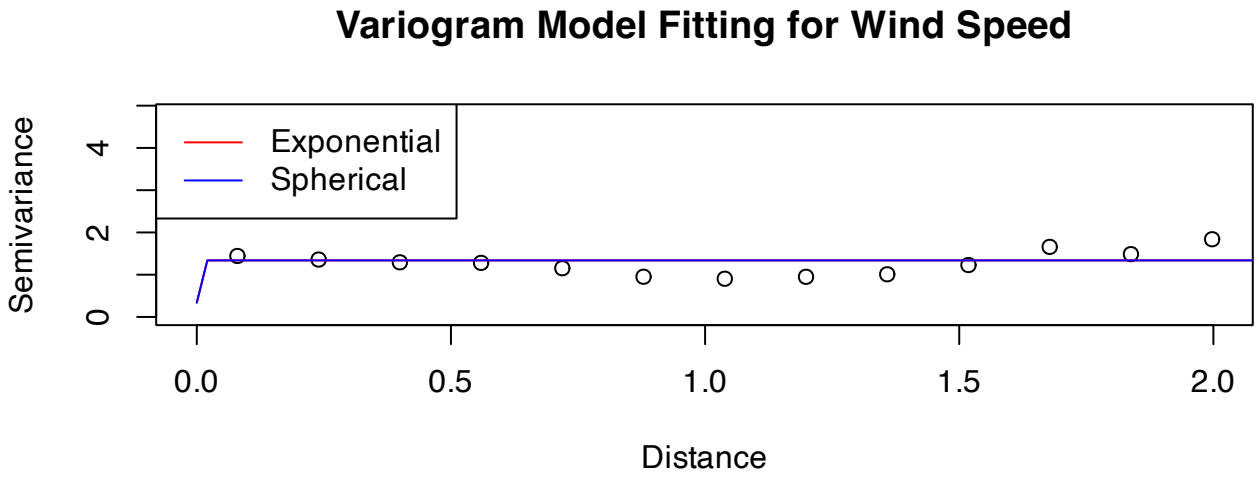


Table 6: Variogram fit estimates for Wind Speed

Model	Nugget	Partial Sill	Range	WSS
Exponential	1.338384	0	1.323686	569.7069
Spherical	1.338384	0	2.372928	569.7069

From the provided variogram model fits, here are some observations:

Temperature

- The Exponential model appears to capture the initial rise in semivariance well, while the Spherical model reaches a sill too quickly.
- The difference in WSS values suggests that the Spherical model might be a slightly better fit.

Wind Speed

- Both models show relatively low semivariance values, suggesting weak spatial dependence.
- The Spherical model appears to provide a smoother fit.

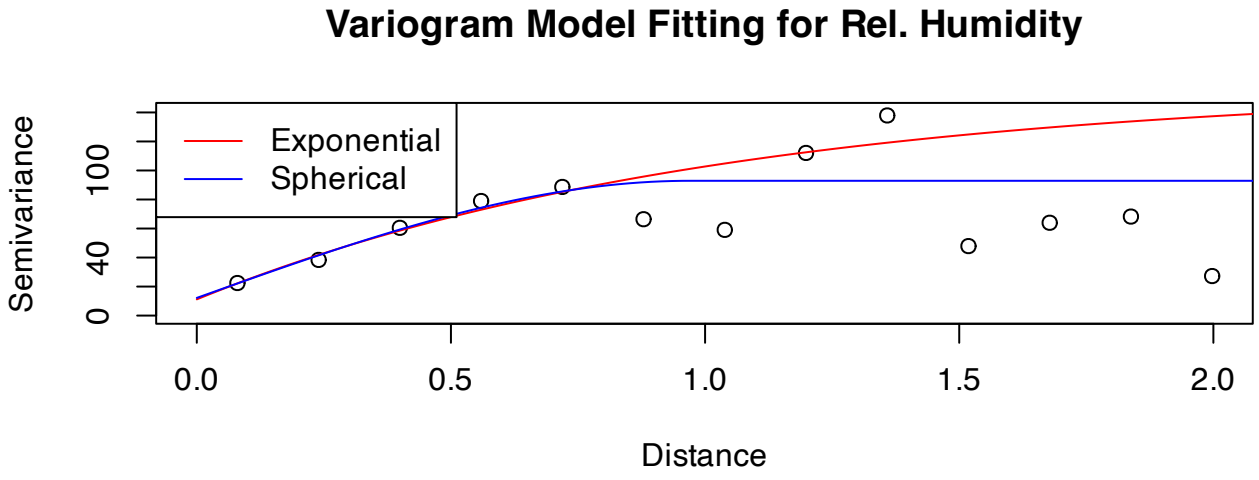


Table 7: Variogram fit estimates for Rel. Humidity

Model	Nugget	Partial Sill	Range	WSS
Exponential	11.17924	147.34067	1.0293740	561.3412
Spherical	12.17654	80.71117	0.9684011	484.2782

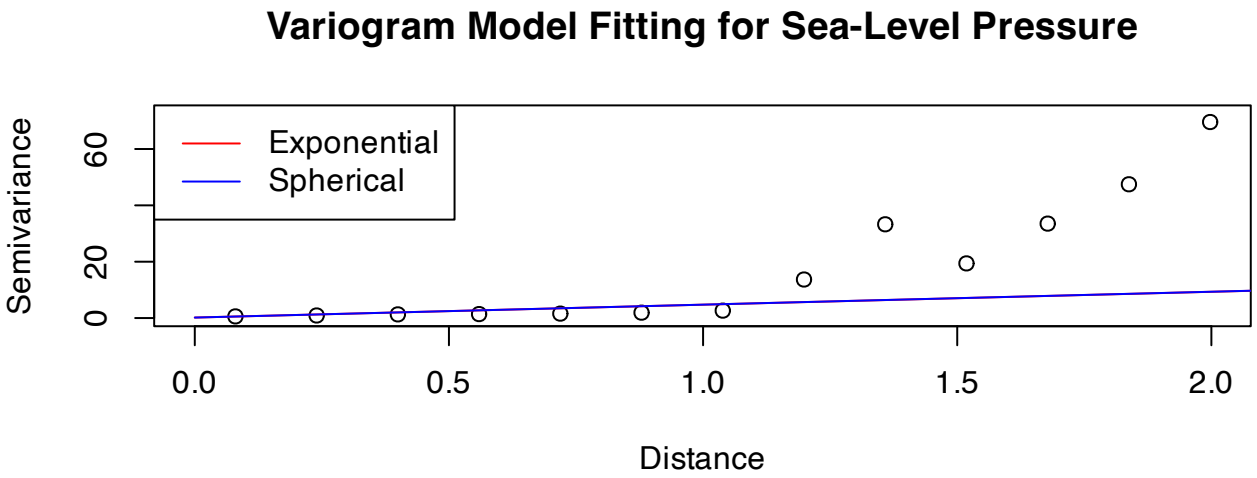


Table 8: Variogram fit estimates for Sea-Level Pressure

Model	Nugget	Partial Sill	Range	WSS
Exponential	0.1735140	55866.450	12210.66	13147.74
Spherical	0.1725412	5784.669	1892.23	13147.16

Relative Humidity

- The Spherical model seems to capture the flattening trend better than the Exponential model.
- The higher semivariance values suggest stronger spatial dependence in humidity compared to wind speed.

Sea-Level Pressure

- The Spherical model seems to align with the data trend more closely, while the Spherical model overshoots in some areas.
- The semivariance values indicate moderate spatial dependence.

General Comments:

- The Spherical model tends to reach the sill quickly, which works well when there is a clear range after which spatial dependence disappears. Assessing WSS values is key to determining which model is better for each variable.

4

(30 pts) Find an optimal interpolation technique for each weather variable.

For each variable, nine different models were fit and chosen to compare for spatial interpolation. Note that not all methods were able to describe the uncertainty among the estimates within the spatial structure, as some of the interpolation methods were deterministic (e.g. IDW). We also considered machine learning

approaches since we were encouraged to do so in our mid-exam meeting. Therefore the only metrics that were considered for model viability were mean-squared error (MSE) and mean absolute error (MAE). This is because the optimal interpolation method would minimize errors for which there are observed observations. This is also the reason no spatial heatmaps with predictions were included in this report. We elect to focus just on MSE and MAE.

First, a linear model with just the longitude and latitude were fit to model trend. This model was not considered for interpolation, but rather exploratory. The nine different models for each are described follows:

- **Spatial Regression:** A full linear model with longitude, latitude, and the remaining three predictors was fit and used to predict on all of the spatial points. MSE was computed without cross validation. This model was not expected to perform very well, and so care was not taken to perform leave-one-out cross validation (LOOCV). This model was meant to be a baseline model.
- **Inverse Distance Weighting:** Inverse Distance Weighting (IDW) is a spatial interpolation method that estimates the value of a function at an unobserved location based on weighted averages of nearby observations. The weight assigned to each observation decreases as the distance to the prediction location increases. IDW estimates the value $\hat{z}$ at an unknown location (x_0, y_0) using the formula:

  $$\hat{z}(x_0, y_0) = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}, \tag{1}$$

  where the weights w_i are given by

  $$w_i = \frac{1}{d_i^p}. \tag{2}$$

  Here, d_i is the Euclidean distance between the known point (x_i, y_i) and the prediction location (x_0, y_0), and p is a user-defined power parameter that controls the influence of distant points. For this analysis, we fit the models using LOOCV, for $p = 1, 2$, and 3.
- **Ordinary Kriging:** For ordinary kriging, we used LOOCV and specified no trend.
- **Universal Kriging:** For universal kriging, we also used LOOCV, except there was a trend specified.
- **Bayesian Ordinary Kriging:** For Bayesian kriging we took the posterior parameter estimates and using them as parameters in kriging. For Bayesian universal kriging, we faced issues with implementation.
- **Neural Network with Longitude and Latitude:** For a simple neural network, we fit a neural network with just the spatial values, using 10-fold cross validation. The hidden layers and L2 penalty were tuned for each variable. Five to fifty hidden layers were considered, and a penalty ranging from 0 to 0.1 was considered.
- **Neural Network with All Predictors:** For this neural network, we fit a neural network with all predictors, using 10-fold cross validation. Again, the hidden layers and L2 penalty were tuned for each variable. The hidden layers and L2 penalty were tuned for each variable. Five to fifty hidden layers were considered.

All processes were considered to be best modeled with the spherical variogram, and frequentist variogram parameter estimates were used for all variable modeling. Additionally, Bayesian universal kriging with cross validation was considered, but was unable to be implemented due to computational constraints.

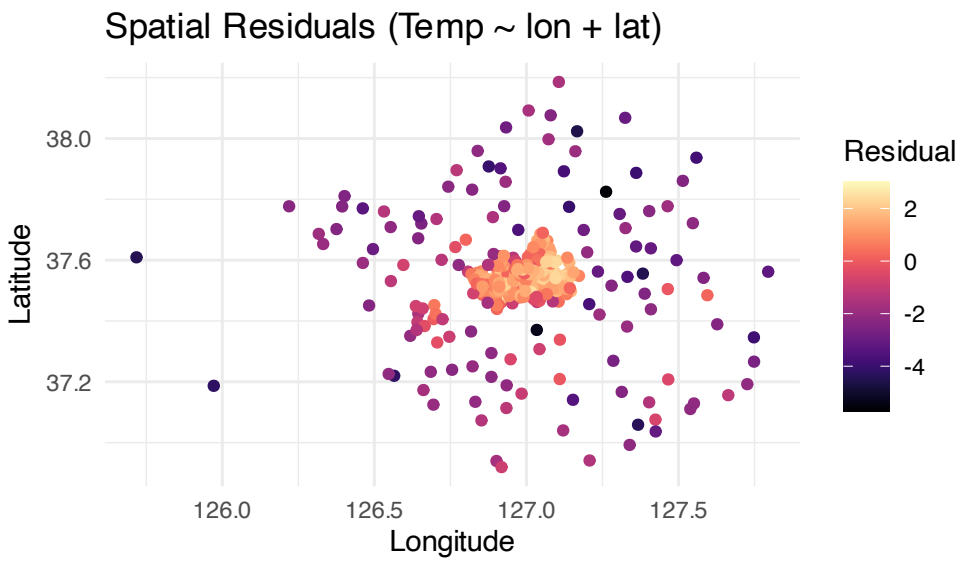
4.1 Temperature

From the temperature, we know that each of the predictors were yielded significant coefficient estimates in (2), but the model seemed to violate some key assumptions that could have yielded low p-values. Hence, we fit two models here, including one with just `lon` and `lat` to check for stationarity and trend, and one with all covariates.

If we fit a regression with longitude and latitude as predictors, we are essentially creating a trend surface model. This approach can help assess stationarity by examining whether there are significant spatial trends in your data.

Characteristic	Beta	95% CI[1]	p-value
lon	-2.1	-2.8, -1.3	<0.001
lat	-0.73	-1.8, 0.33	0.2

[1] CI = Confidence Interval



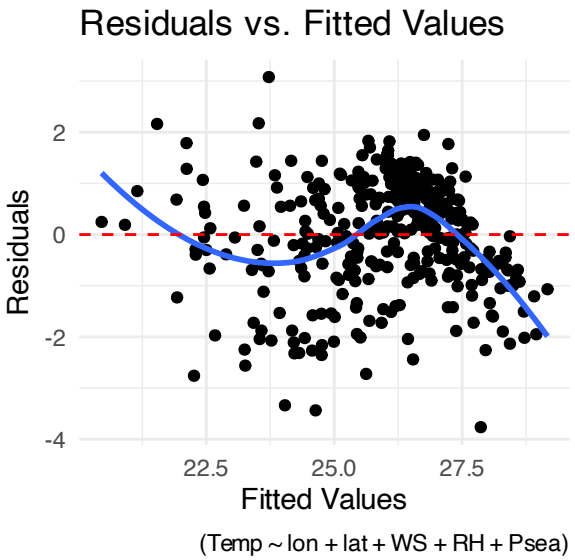
There definitely seems to be a negative trend across longitude, according to our fitted model to examine trend. This graphic visualizes the spatial distribution of residuals from a regression model predicting temperature based on longitude and latitude. Key observations include:

1. **Spatial Relationship of Residuals**: The residuals exhibit spatial structure, with noticeable clusters of positive (orange/yellow) and negative (purple/black) residuals, indicating potential model misspecification or unaccounted spatial trends.

2. **Localized Patterns**: The central region near longitude ~127.0 appears to have a concentration of positive residuals, suggesting that the model underpredicts temperature in this area. Conversely, negative residuals are dispersed throughout other regions. This might hint a nonlinear trends, but these were not considered in this analysis due to software constraints.

Characteristic	Beta	95% CI[1]	p-value
lon	-3.0	-3.5, -2.5	<0.001
lat	-1.0	-1.7, -0.41	0.001
WS	0.12	0.02, 0.22	0.017
RH	-0.20	-0.22, -0.19	<0.001
Psea	-0.11	-0.22, -0.01	0.040

[1] CI = Confidence Interval



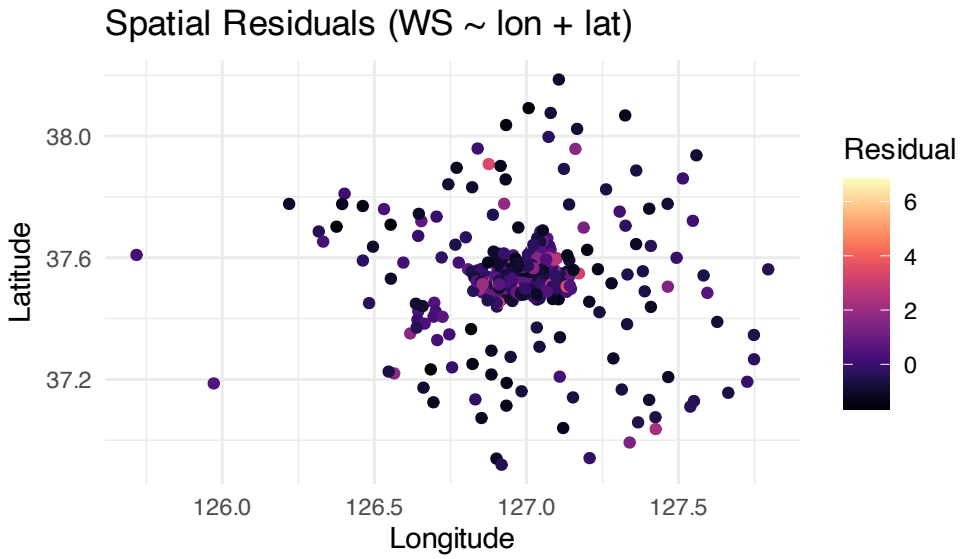
The full model residuals appear to demonstrate a non-linear relationship, which lends to the belief that there are relationships in the predictors that are not being accounted for in the model. This model might not be the most helpful for prediction.

For the universal kriging, we selected the trend to be `Temp ~ lon + lat + RH`, based on the coefficient estimates for the regression model. Additionally, for the simpler neural network, we found 23 hidden units with an L2 penalty of 1.17 to be most optimal. For the full neural network we found 21 hidden units with an L2 penalty parameter of 1.08. Results can be found in section (5).

4.2 Wind Speed

Characteristic	Beta	95% CI[1]	p-value
lon	-0.59	-1.1, -0.11	0.017
lat	0.09	-0.57, 0.76	0.8

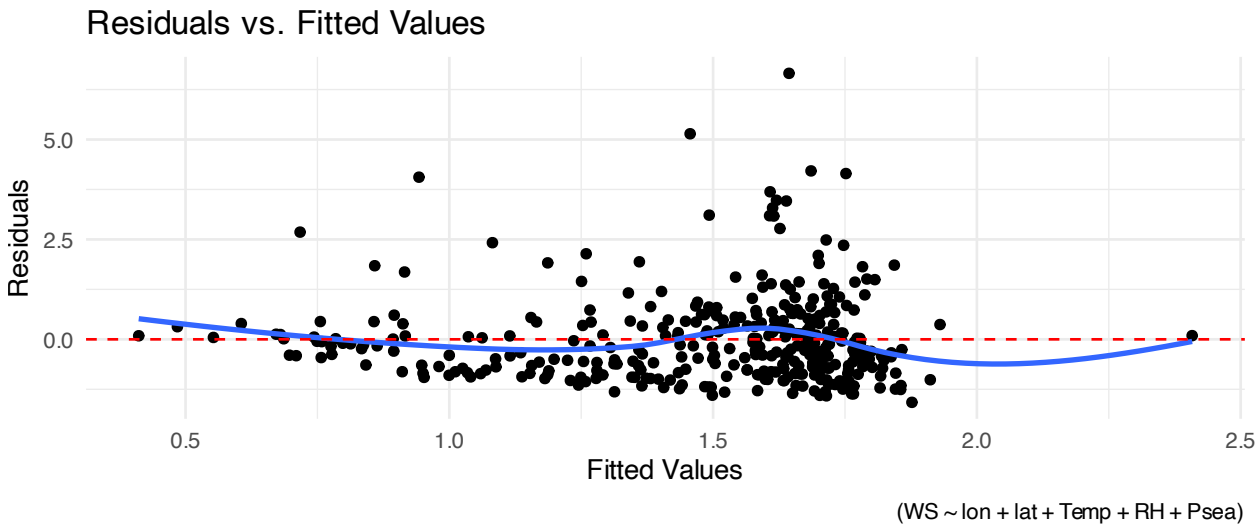
[1] CI = Confidence Interval



There not as much of a noticeable pattern of residual clustering. The residuals are mostly in the 0 to 6 range, with higher residuals (lighter colors) appearing near the central region. This suggests that the model might not fully capture spatial variations in wind speed.

Characteristic	Beta	95% CI[1]	p-value
lon	-0.29	-0.88, 0.30	0.3
lat	0.17	-0.48, 0.83	0.6
Temp	0.13	0.02, 0.23	0.017
RH	0.00	-0.03, 0.03	>0.9
Psea	-0.08	-0.19, 0.03	0.2

[1]CI = Confidence Interval



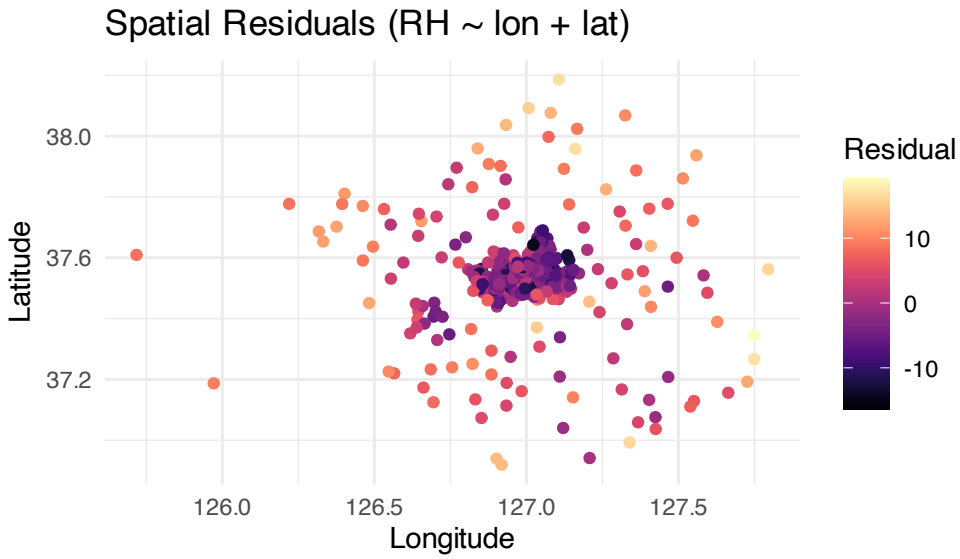
This full model seems to not have any blatant patterns in the residuals; however, there does seem to be some higher residuals toward the middle of the values.

For the universal kriging, we selected the trend to be `WS ~ lon + lat + Temp`, based on the coefficient estimates for the full regression model (see table above). Those estimates seemed to be likely not zero, so we used them to model the trend. Additionally, for the simpler neural network, we found 36 hidden units with an L2 penalty of 1.04 to be most optimal. For the full neural network we found 24 hidden units with an L2 penalty parameter of 1.24. Results can be found in section (5).

4.3 Relative Humidity

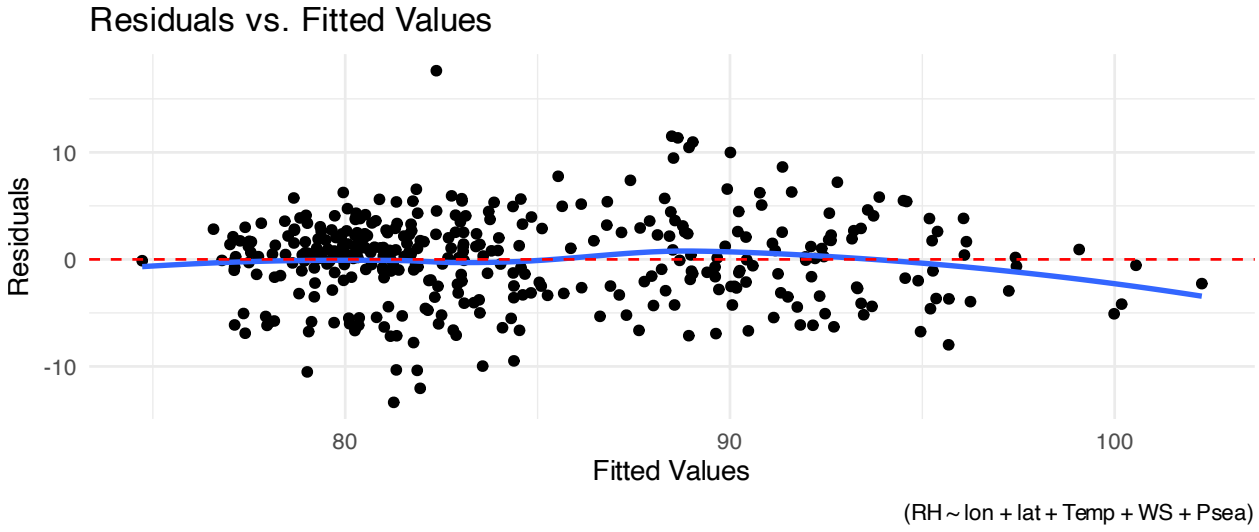
Characteristic	Beta	95% CI[1]	p-value
lon	-5.3	-8.2, -2.3	<0.001
lat	-1.4	-5.3, 2.6	0.5

[1]CI = Confidence Interval



Characteristic	Beta	95% CI[1]	p-value
lon	-11	-13, -9.6	<0.001
lat	-3.4	-5.8, -0.98	0.006
Temp	-2.9	-3.1, -2.6	<0.001
WS	0.00	-0.38, 0.38	>0.9
Psea	0.52	0.11, 0.92	0.012

[1] CI = Confidence Interval

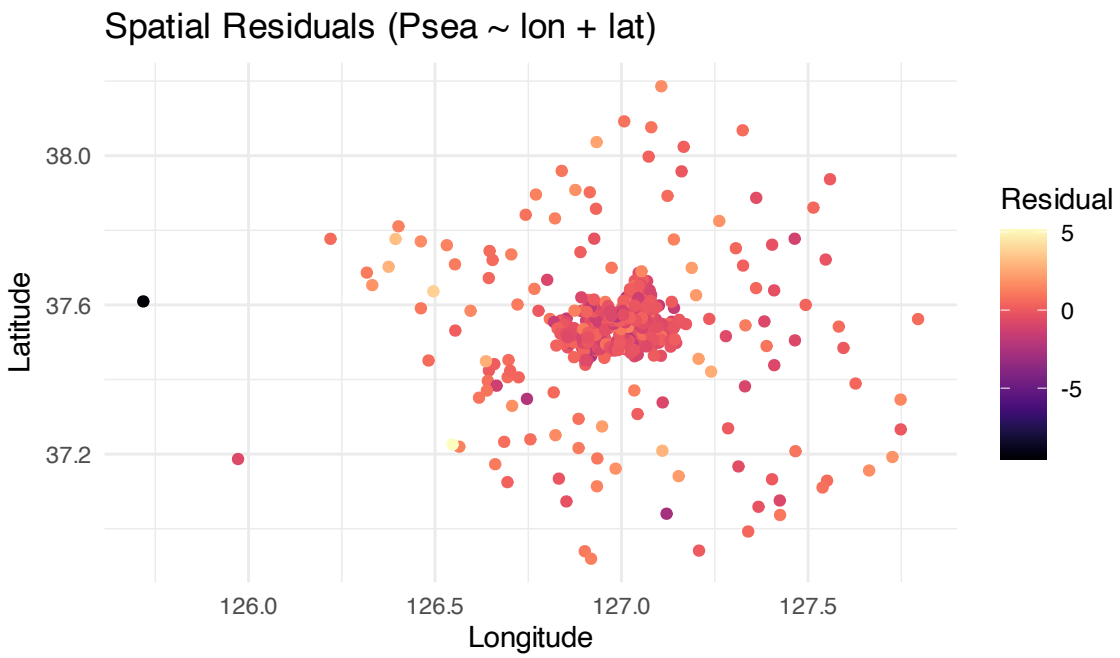


For the universal kriging, we selected the trend to be `RH ~ lon + lat + Temp + Psea`, based on the coefficient estimates for the full regression model (see table above). Those estimates seemed to be likely not zero, so we used them to model the trend. Additionally, for the simpler neural network, we found 37 hidden units with an L2 penalty of 1.03 to be most optimal. For the full neural network we, also ironically like for wind speed, found 24 hidden units with an L2 penalty parameter of 1.24. Results can be found in section (5).

4.4 Sea-Level Pressure

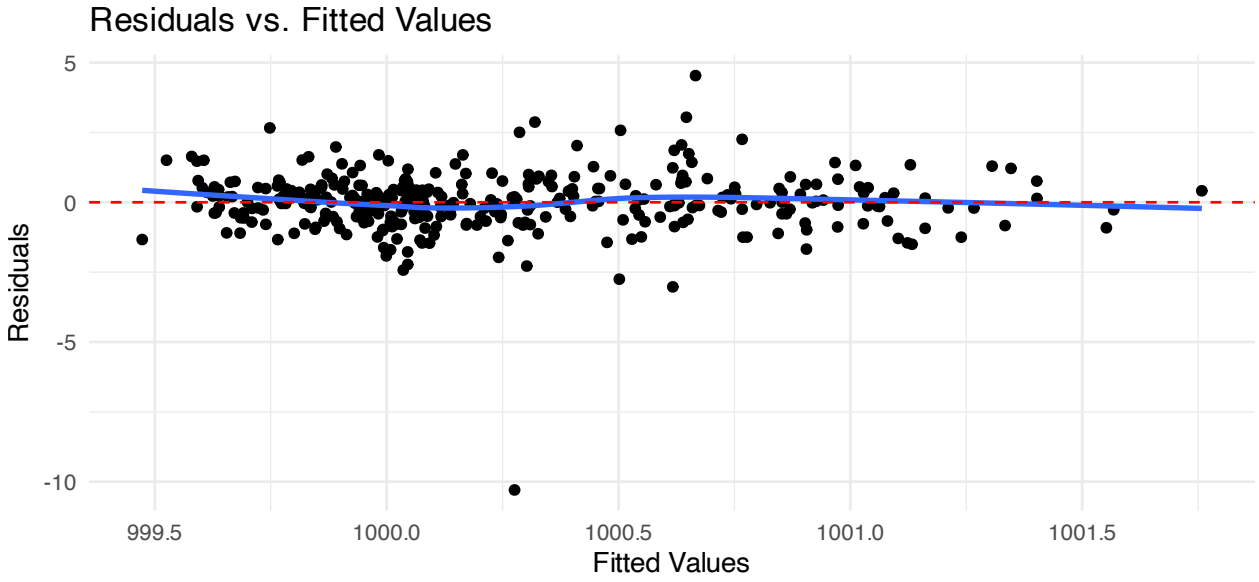
Characteristic	Beta	95% CI[1]	p-value
lon	0.54	0.06, 1.0	0.029
lat	-0.19	-0.84, 0.47	0.6

[1] CI = Confidence Interval



Characteristic	Beta	95% CI[1]	p-value
lon	0.46	-0.09, 1.0	0.10
lat	-0.21	-0.83, 0.41	0.5
Temp	-0.10	-0.20, 0.00	0.040
WS	-0.07	-0.16, 0.03	0.2
RH	0.03	0.01, 0.06	0.012

[1] CI = Confidence Interval



For the universal kriging, we selected the trend to be `Psea ~ lon + lat + Temp`, based on the coefficient estimates for the full regression model (see table above). Those estimates seemed to be likely not zero, so we used them to model the trend. Additionally, for the simpler neural network, we found 17 hidden units with an L2 penalty of 1.23 to be most optimal. For the full neural network we, also ironically like for wind speed, found 15 hidden units with an L2 penalty parameter of 1.07. Results can be found in section (5).

5

(10 pts) Report the best spatial interpolator for each variable with the performance table for each variable.

5.1 Temperature Interpolation Performance

Method	MSE	MAE
NN (Lon/Lat Only)	0.5261744	0.5378614
LOOCV UK	0.6007225	0.5652206
NN All Preds.	0.6479467	0.5761237
LOOCV OK Bayes	0.6562829	0.5872261
LOOCV OK	0.6701764	0.6019297
Spatial Regression	1.1750160	0.8732776
IDW (p = 3)	1.3120009	0.7633802
IDW (p = 2)	1.9697153	0.9430535
IDW (p = 1)	2.7430922	1.1797662

Surprisingly, the simpler neural network yielded the lowest MSE among all interpolation models.

5.2 Wind Speed Interpolation Performance

Method	MSE	MAE
NN All Preds.	0.8461487	0.6611832

Method	MSE	MAE
NN (Lon/Lat Only)	0.8921029	0.6662810
Spatial Regression	1.2253981	0.7839901
LOOCV OK Bayes	1.2322500	0.7821390
LOOCV UK	1.2544615	0.7958264
IDW (p = 1)	1.2926357	0.8388596
IDW (p = 2)	1.3219298	0.8463043
LOOCV OK	1.3305579	0.8384540
IDW (p = 3)	1.4911370	0.8732137

For wind speed, the best interpolator was the neural network with all of the predictors.

5.3 Relative Humidity Interpolation Performance

Method	MSE	MAE
LOOCV UK	14.71868	2.865960
NN (Lon/Lat Only)	15.62949	3.156036
Spatial Regression	16.59929	3.127453
NN All Preds.	16.62422	2.909625
LOOCV OK	18.08475	3.174946
LOOCV OK Bayes	18.23677	3.180243
IDW (p = 3)	23.78904	3.740163
IDW (p = 2)	29.18238	4.166864
IDW (p = 1)	38.98403	4.761323

For relative humidity, the universal kriging yielded the lowest MSE.

5.4 Sea-Level Pressure Interpolation Performance

Method	MSE	MAE
LOOCV UK	1.083316	0.7066052
LOOCV OK Bayes	1.084181	0.7002600
LOOCV OK	1.092724	0.7063446
Spatial Regression	1.107210	0.6779361
IDW (p = 2)	1.169930	0.7193979
IDW (p = 1)	1.212961	0.7287618
IDW (p = 3)	1.242006	0.7485461
NN All Preds.	4.975260	1.5880504
NN (Lon/Lat Only)	34.443691	3.6630367

For Sea-Level Pressure, the universal kriging also yielded the lowest MSE, and hence was the best interpolator for this variable.

6

(15 pts) Summarize your choice and overall findings. Discuss about the limitations of your analysis and potential solutions for them.

The interpolation methods varied in effectiveness depending on the variable. For temperature, the neural network (NN) using only longitude and latitude was the best performer, with an MSE of 0.53 and an MAE of 0.54. A clear west-to-east cooling trend was evident, highlighting the spatial dependency of temperature. Wind speed showed less pronounced spatial patterns, but NN with all predictors outperformed other methods, yielding an MSE of 0.85 and an MAE of 0.66. Relative humidity displayed a strong longitudinal trend, best captured by Universal Kriging with leave-one-out cross-validation (LOOCV), where the estimated trend coefficient was -5.3. For sea-level pressure, Universal Kriging also provided the best results (MSE = 1.08, MAE = 0.71), though this variable exhibited minimal spatial structure except for an anomalous westward variogram.

Regression analysis revealed that Model 4 (`Psea ~ WS + RH + Temp`) had the most well-behaved residuals. Spherical variograms generally provided a better fit than exponential ones. While isotropy was reasonable for most variables, directional variograms suggested variations, particularly for sea-level pressure at 90°. The presence of outlier stations (11110 and 11176) with extreme relative humidity values warranted further scrutiny, though they did not significantly distort the overall models. These findings highlight the importance of model selection and validation techniques in ensuring reliable spatial interpolation.

Several limitations suggest directions for future research. The isotropy assumption, while simplifying, overlooks directional effects that may be influential. Residual plots hint at potential non-linear relationships that linear models fail to capture, and empirical variograms do not perfectly align with theoretical expectations at larger distances. While LOOCV provided useful cross-validation, k-fold validation could yield more robust performance metrics. Computational constraints prevented the implementation of Bayesian Universal Kriging with cross-validation, but this remains a promising avenue. Future work should explore anisotropy modeling for sea-level pressure, advanced neural network architectures, locally adaptive methods, higher-order trend surfaces, and robust estimation techniques for variables prone to outliers like relative humidity. Despite these challenges, this analysis establishes a strong foundation for spatial interpolation while acknowledging the complexities inherent in weather data.

7 Code Appendix

Below is all of the code used for this assignment. Commented out sections were run and objects from those were saved and loaded later for computational efficiency. Syntax highlighting proved difficult for rendering purposes, but code has been provided in a `verbatim environment`.

```
## ----setup, include=FALSE-------------------------------------------------------------------
knitr::opts_chunk$set(dev = "cairo_pdf",
                      echo = FALSE,
                      message = FALSE,
                      warning = FALSE,
                      results = 'hide',
                      fig.align = 'center',
                      fig.height = 3.5,
                      fig.width = 7)
library("tidymodels"); library("patchwork"); library("glue")
library("scales", warn.conflicts = FALSE); library("extrafont")
library("tinytex"); library("ggmap"); library("geoR");
library("ggnewscale"); library("gtsummary"); library("car")
theme_set(theme_minimal(base_family = "Roboto Condensed"))


## -------------------------------------------------------------------------------------------
load(here::here("Midterm", "weather.RData"))

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

multi_geodata <- as.geodata(weather, coords.col = c("lon", "lat"),
                            data.col = c("Temp", "WS", "RH", "Psea"))

all_geodata <- list(
  "temp_geodata" = as.geodata(weather, coords.col = c("lon", "lat"),
                              data.col = c("Temp")),
  "WS_geodata" = as.geodata(weather, coords.col = c("lon", "lat"),
                            data.col = c("WS")),
  "RH_geodata" = as.geodata(weather, coords.col = c("lon", "lat"),
                            data.col = c("RH")),
  "Psea_geodata" = as.geodata(weather, coords.col = c("lon", "lat"),
                              data.col = c("Psea"))
)


## ----fig.height=3.2-------------------------------------------------------------------------
# 1 ---------------------------------
# EDA

weather |>
  select(-c(Sid)) |>
  GGally::ggpairs(aes(alpha = 0.2), title = "Pairs Plot for Weather Variables")

eda1 <- weather |>
  ggplot(aes(lon, lat, color = Temp)) +
  geom_point(alpha = 0.75) +
  scale_color_viridis_c(option = "plasma") +
```

```
  labs(
    title = "Temperature Distribution over Space",
    x = "Longitude",
    y = "Latitude",
    color = ""
  ) +
  coord_equal() +
  theme(plot.title = element_text(size = 10))

eda2 <- weather |>
  ggplot(aes(lon, lat, color = WS)) +
  geom_point(alpha = 0.75) +
  scale_color_viridis_c(option = "plasma") +
  labs(
    title = "Wind Speed Distribution over Space",
    x = "Longitude",
    y = "Latitude",
    color = ""
  ) +
  coord_equal() +
  theme(plot.title = element_text(size = 10))

eda3 <- weather |>
  ggplot(aes(lon, lat, color = RH)) +
  geom_point(alpha = 0.75) +
  scale_color_viridis_c(option = "plasma") +
  labs(
    title = "Rel. Humidity Distribution over Space",
    x = "Longitude",
    y = "Latitude",
    color = ""
  ) +
  coord_equal() +
  theme(plot.title = element_text(size = 10))

eda4 <- weather |>
  ggplot(aes(lon, lat, color = Psea)) +
  geom_point(alpha = 0.75) +
  scale_color_viridis_c(option = "plasma") +
  labs(
    title = "Sea-Level Pressure Distribution over Space",
    x = "Longitude",
    y = "Latitude",
    color = ""
  ) +
  coord_equal() +
  theme(plot.title = element_text(size = 10))

## ----fig.height=5-----------------------------------------------------------------------------
(eda1 + eda2) / (eda3 + eda4)


## ---------------------------------------------------------------------------------------------
summary(weather[,4:7]) |> knitr::kable(
```

```
  caption = "Summary statistics for each variable of interest within the data."
)
weather |> filter(lon < 126) |>  select(-c(lon, lat)) |> knitr::kable(
  caption = "Observed values of further weather stations for variables of interest within the data."
)


## ------------------------------------------------------------------------------------------
multi_variog <- variog(multi_geodata)
robust_multi_variog <- variog(multi_geodata, estimator.type = "modulus")

vg_plot_data <- tibble(
  "Distance" = rep(multi_variog$u, times = 2),
  "Temp." = c(multi_variog$v[,1], robust_multi_variog$v[,1]),
  "Wind Speed" = c(multi_variog$v[,2], robust_multi_variog$v[,2]),
  "Rel. Humidity" = c(multi_variog$v[,3],  robust_multi_variog$v[,3]),
  "Sea-Level Pressure" = c(multi_variog$v[,4], robust_multi_variog$v[,4]),
  "Variogram" = c(rep("Classical", times = 13), rep("Robust", times = 13))
) |>
  pivot_longer(
    cols = -c(Distance, Variogram),
    names_to = "Variable",
    values_to = "Semivariance"
  )

ggplot() +
  geom_line(data = vg_plot_data, aes(Distance, Semivariance, colour = Variogram)) +
  geom_point(data = vg_plot_data, aes(Distance, Semivariance, colour = Variogram)) +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("steelblue", "firebrick")) +
  labs(
    title = "Empirical Variogram for Weather Data",
    color = ""
  ) +
  facet_wrap(Variable ~ ., ncol = 2, scales = "free")


## ----results='asis'-----------------------------------------------------------------------
# 2 ---------------------------------
# Regression Analysis
reg <- function(formula, data = weather, n = 1, log = FALSE) {

  formula <- as.formula(formula)



  m <- lm(formula, data = data)

  tib <- tibble(
    "resid" = residuals(m),
    "fitted" = fitted(m)
  )

  plt1 <- tib |>
```

```
    ggplot(aes(fitted, resid)) +
    geom_point() +
    geom_smooth(se = FALSE, color = "steelblue") +
    geom_hline(yintercept = 0, linetype = "dashed", color = "firebrick") +
    labs(
      title = paste0("Residuals vs. Fitted Values of Model ", n),
      x = "Fitted Values",
      y = "Residuals"
    )

  plt2 <- tib |>
    ggplot(aes(resid)) +
    geom_histogram(aes(y = after_stat(density))) +
    geom_vline(xintercept = 0, linetype = "dashed", color = "firebrick") +
    labs(
      title = paste0("Histogram of Residuals for Model ", n),
      x = "Residuals",
      y = "Density"
    )

  t <- shapiro.test(tib$resid)

  list(model = m, plot = (plt1 + plt2), test = t, vif = car::vif(m))
}

m1 <- reg("Temp ~ WS + RH + Psea")
m1$model |> tbl_regression()
m1$plot
# m1$t


## ----results='asis'-----------------------------------------------------------
m2 <- reg("WS ~ Temp + RH + Psea", n = 2)
m2$model |> tbl_regression()


## -----------------------------------------------------------------------------
m2$plot
# m2$test
# m2$vif


## ----results='asis'-----------------------------------------------------------
m3 <- reg("RH ~ WS + Temp + Psea", n = 3)
m3$model |> tbl_regression()
m3$plot
# m3$test
# m3$vif


## ----results='asis'-----------------------------------------------------------
m4 <- reg("Psea ~ WS + RH + Temp", n = 4)
m4$model |> tbl_regression()
m4$plot
```

```
# m4$test
# m4$vif


## ----fig.height=10.7, eval=FALSE-----------------------------------------------------------------
## # 3 ---------------------------------
## # Analyzing Isotropy
##
## # ggplot graphics were not that good here...
##
## impute_with_previous <- function(vec) {
##   # Check if input is a vector
##   if (!is.vector(vec)) {
##     stop("Input must be a vector")
##   }
##
##   # Initialize result vector
##   result <- vec
##
##   # If the first element is NA, we can't impute it with a previous value
##   # (could modify this behavior if needed)
##
##   # Loop through the vector starting from the second element
##   for (i in 2:length(vec)) {
##     if (is.na(result[i])) {
##       result[i] <- result[i-1]
##     }
##   }
##
##   result
## }
##
## multi_variog4 <- variog4(multi_geodata)
##
##
##
## # plots <- list()
##
## vg_plot_data_dir <- list()
##
## for (i in 1:4) {
##   dvg <- multi_variog4[[i]]
##
##   angle <- names(multi_variog4[i])
##
##   v <- dvg$v[(which(!is.nan(dvg$v)))]
##
##   vg_plot_data_dir[[i]] <- tibble(
##     "Distance" = dvg$u,
##     "Temp." = v[1:13] |> impute_with_previous(),
##     "Wind Speed" = v[14:26] |> impute_with_previous(),
##     "Rel. Humidity" = v[27:39] |> impute_with_previous(),
##     "Sea-Level Pressure" = v[40:52] |> impute_with_previous(),
##     "Angle" = rep(angle, 13) |> as.factor()
```

```
##   )
##
##   # plots[[i]] <- ggplot() +
##   #   geom_line(data = vg_plot_data_dir, aes(Distance, Semivariance, color = Variable)) +
##   #   geom_point(data = vg_plot_data_dir, aes(Distance, Semivariance, color = Variable)) +
##   #   theme(legend.position = "none") +
##   #   scale_color_viridis_d() +
##   #   labs(
##   #     title = paste("Directional Empirical Variogram for Weather Data at ", angle, " Degrees"),
##   #     color = ""
##   #   ) +
##   #   facet_wrap(Variable ~ ., ncol = 2, scales = "free")
##
## }
##
## plt_dir_data <- reduce(vg_plot_data_dir, bind_rows) |>
##   pivot_longer(
##     cols = -c(Distance, Angle),
##     names_to = "Variable",
##     values_to = "Semivariance"
##   )
##
## ggplot() +
##     geom_line(data = plt_dir_data, aes(Distance, Semivariance, color = Angle)) +
##     geom_point(data = plt_dir_data, aes(Distance, Semivariance, color = Angle)) +
##     theme(legend.position = "bottom") +
##     scale_color_viridis_d() +
##     labs(
##       title = paste("Multi-Directional Empirical Variograms for Weather Data"),
##       color = "Angle"
##     ) +
##     facet_wrap(Variable ~ ., ncol = 1, scales = "free")


## ----fig.height=3.5-------------------------------------------------------------------------

# Assuming all_geodata is a list with each element corresponding to a variable
variable_names <- c("Temperature", "Wind Speed", "Rel. Humidity", "Sea-Level Pressure")

lapply(seq_along(all_geodata), function(i) {
  geodata <- all_geodata[[i]]

  # Get the corresponding variable name for this geodata element
  var_name <- variable_names[i]

  # Create plot with proper title
  plot(variog4(geodata), same = TRUE, type = "b",
       xlab = "Distance", ylab = "Semivariance",
       main = paste("Directional Variogram for", var_name),
       cex.main = 0.8)
  title(main = paste("Directional Variogram for", var_name))
})
```

```
## ----results='asis'-------------------------------------------------------------------------
vg_m <- function(i) {
  geodata <- all_geodata[[i]]
  var_name <- variable_names[i]

  vario <- variog(geodata, messages = FALSE)

  # Calculate initial parameter estimates
  # A common approach is to use:
  # nugget: minimum semivariance value
  # partial sill: maximum semivariance minus nugget
  # range: distance at which semivariogram reaches ~95% of sill

  nugget_ini <- min(vario$v)
  sill_ini <- max(vario$v)
  partial_sill_ini <- sill_ini - nugget_ini

  # For range, find distance at ~95% of sill
  sill_95 <- nugget_ini + 0.95 * partial_sill_ini
  range_index <- which(vario$v >= sill_95)[1]
  range_ini <- ifelse(!is.na(range_index), vario$u[range_index], max(vario$u)/3)

  # Create initial parameter vectors
  # Order is typically: nugget, partial sill, range (phi)
  ini_exp <- c(nugget_ini, partial_sill_ini, range_ini/3)  # Exponential model needs smaller range
  ini_sph <- c(nugget_ini, partial_sill_ini, range_ini)

  # Fit variogram models with initial parameters
  v_exp <- variofit(vario, cov.model = "exponential",
                    weights = "cressie",
                    ini.cov.pars = c(partial_sill_ini, range_ini/3),
                    nugget = nugget_ini,
                    fix.nugget = FALSE,  # Set to TRUE if you want to fix the nugget
                    messages = FALSE)

  v_sph <- variofit(vario, cov.model = "spherical",
                    weights = "cressie",
                    ini.cov.pars = c(partial_sill_ini, range_ini),
                    nugget = nugget_ini,
                    fix.nugget = FALSE,
                    messages = FALSE)

  # Continue with plotting as before
  plot(vario, main = paste0("Variogram Model Fitting for ", var_name), xlab = "Distance", ylab = "Semiva
       ylim = c(0, max(vario$v + 3)))
  lines(v_exp, col = "red")
  lines(v_sph, col = "blue")
  legend("topleft", legend = c("Exponential", "Spherical"),
         col = c("red", "blue"), lty = 1)

  # Add model parameter information to output
  tibble(
    "Model" = c("Exponential", "Spherical"),
    "Nugget" = c(v_exp$nugget, v_sph$nugget),
```

```
    "Partial Sill" = c(v_exp$cov.pars[1], v_sph$cov.pars[1]),
    "Range" = c(v_exp$cov.pars[2], v_sph$cov.pars[2]),
    "WSS" = c(v_exp$value, v_sph$value)
  ) |>
    knitr::kable(caption = paste0("Variogram fit estimates for ", var_name))
}

## ----fig.height=3, results='asis'-----------------------------------------------------------
vg_m(1)
vg_m(2)

## ----results='asis'-------------------------------------------------------------------------
vg_m(3)
vg_m(4)


## ----fig.height=3, results='asis'-----------------------------------------------------------
# 4 ---------------------------------
# Temperature Prediction


# Regression ------------------------
spatial_trend <- lm(Temp ~ lon + lat, data = weather)
spatial_trend |> tbl_regression()

# Visualize residuals spatially
weather$spatial_residuals <- residuals(spatial_trend)
ggplot(weather, aes(x = lon, y = lat, color = spatial_residuals)) +
  geom_point() +
  scale_color_viridis(option = "magma") +
  labs(title = "Spatial Residuals (Temp ~ lon + lat)",
       x = "Longitude", y = "Latitude", color = "Residual") +
  theme_minimal() +
  coord_equal()



## ----fig.height=3, results='asis'-----------------------------------------------------------
expanded_model <- lm(Temp ~ lon + lat + WS + RH + Psea, data = weather)
expanded_model |> tbl_regression()

weather$expanded_residuals <- residuals(expanded_model)
ggplot(weather, aes(x = fitted(expanded_model), y = expanded_residuals)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values", y = "Residuals",
       caption = "(Temp ~ lon + lat + WS + RH + Psea)") +
  theme_minimal() +
  coord_equal()


## -------------------------------------------------------------------------------------------
```

```
# Oridnary/Universal Kriging -------------------------
library("gstat")
library("sp")

# Convert wells data to a spatial data frame
coordinates(weather) <- ~ lon + lat

ok_cv <- krige.cv(Temp ~ 1, weather,
                  model = vgm(psill = 7.13459,
                              model = "Sph",
                              range = 0.8495602,
                              nugget = 0))

uk_cv <- krige.cv(Temp ~ lon + lat + RH, weather,
                  model = vgm(psill = 7.13459,
                              model = "Sph",
                              range = 0.8495602,
                              nugget = 0))

mse <- \(residuals) mean(residuals^2)
mae <- \(residuals) mean(abs(residuals))
# mse(ok_cv$residual)
# mae(ok_cv$residual)
# mse(uk_cv$residual)
# mae(uk_cv$residual)


## ------------------------------------------------------------------------------------------------
temp_bayes_ok <- read_rds("temp_bayes_ok.rds")
ok_cv_bayes <- krige.cv(Temp ~ 1, weather,
                  model = vgm(psill = temp_bayes_ok$posterior$sigmasq$summary[[1]],
                              model = "Sph",
                              range = temp_bayes_ok$posterior$sigmasq$summary[[1]],
                              nugget = temp_bayes_ok$posterior$tausq.rel$summary[[1]])
)


## ----eval=FALSE--------------------------------------------------------------------------------
## # Bayesian Kriging -------------------------
##
## temp <- all_geodata$temp_geodata
##
## locations <- as.matrix(weather.dat |> select(lon, lat))
##
## bayes_mod_ok <- model.control(
##   trend.d = "cte",
##   cov.model = "spherical"
## )
##
## bayes_prior <- prior.control(
##   beta.prior = "flat",
##   sigmasq.prior = "reciprocal",
##   tausq.rel.prior = "uniform",
##   tausq.rel.discrete = seq(from = 0, to = 1, by = 0.01)
```

```
## )
##
## temp_bayes_ok <- krige.bayes(
##   temp,
##   locations = locations[,1:2],
##   model = bayes_mod_ok,
##   prior = bayes_prior
## )
##
## ok_cv_bayes <- krige.cv(Temp ~ 1, weather,
##                   model = vgm(psill = temp_bayes_ok$posterior$sigmasq$summary[[1]],
##                               model = "Sph",
##                               range = temp_bayes_ok$posterior$sigmasq$summary[[1]],
##                               nugget = temp_bayes_ok$posterior$tausq.rel$summary[[1]])
## )
##
## mse(ok_cv_bayes$residual)
## mae(ok_cv_bayes$residual)
##
## # temp <- all_geodata$temp_geodata
## #
## # # Ensure RH, lon, and lat are properly included in temp$data
## # temp$data <- weather.dat |> select(lon, lat, RH)
## #
## # # Ensure locations match the number of observations
## # locations <- temp$data |> select(lon, lat) |> as.matrix()
## #
## # attr(temp$coords, "dimnames")[[1]] <- as.character(1:365)
## #
## # # Define the trend formula
## # trend_formula <- data ~ coords[,1] + coords[,2]
## #
## # # Define the Bayesian universal kriging model
## # bayes_mod_uk <- model.control(
## #   trend.d = trend_formula,
## #   trend.l = trend_formula,
## #   cov.model = "spherical",
## #   kappa = 0.5
## # )
## #
## # # Set the Bayesian priors
## # bayes_prior <- prior.control(
## #   beta.prior = "flat",
## #   sigmasq.prior = "reciprocal",
## #   tausq.rel.prior = "uniform",
## #   tausq.rel.discrete = seq(from = 0, to = 1, by = 0.01)
## # )
## #
## # # Perform Bayesian kriging
## # temp_bayes_uk <- krige.bayes(
## #   temp,
## #   locations = locations[,1:2],
## #   model = bayes_mod_uk,
## #   prior = bayes_prior
```

```
## # )
##


## ------------------------------------------------------------------------------------------------
# Inverse Distance Weighting -------------------------

loocv <- function(data, method, ...) {
  n <- nrow(data)
  pred <- numeric(n)

  for (i in 1:n) {
    train_data <- data[-i, ]
    test_point <- data[i, ]

    if (method == "idw") {
      # IDW method
      power <- list(...)$power
      coords <- train_data[, c("lon", "lat")]
      values <- train_data$Temp
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(Temp = values))
      idw_model <- gstat::idw(Temp ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat
                              idp = power)
      pred[i] <- idw_model$var1.pred
    }
    else if (method == "regression") {
      # Regression method
      formula <- list(...)$formula
      model <- lm(formula, data = train_data)
      pred[i] <- predict(model, newdata = test_point)
    }
    else if (method == "log_regression") {
      # Log-transformed regression
      formula <- list(...)$formula
      model <- lm(formula, data = train_data)
      pred[i] <- exp(predict(model, newdata = test_point))
    }
    else if (method == "kriging") {
      # Ordinary kriging
      coords <- train_data[, c("lon", "lat")]
      values <- train_data$Temp
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(Temp = values))

      # Fit variogram
      vgm <- variogram(Temp ~ 1, data = sp_data)
      fit_vgm <- fit.variogram(vgm, vgm("Sph"))

      # Kriging prediction
      kriging_result <- krige(Temp ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat
                              model = fit_vgm)
      pred[i] <- kriging_result$var1.pred
    }
    else if (method == "regression_kriging") {
      # Regression kriging
```

```
      formula <- list(...)$formula

      # Fit regression model
      reg_model <- lm(formula, data = train_data)

      # Get residuals
      train_data$residuals <- residuals(reg_model)

      # Predict trend
      trend_pred <- predict(reg_model, newdata = test_point)

      # Kriging on residuals
      coords <- train_data[, c("lon", "lat")]
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(residuals = train_data$residuals))

      # Fit variogram on residuals
      vgm <- variogram(residuals ~ 1, data = sp_data)
      fit_vgm <- fit.variogram(vgm, vgm("Sph"))

      # Kriging prediction of residuals
      kriging_result <- krige(residuals ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon",
                              model = fit_vgm)

      # Combine trend and residual prediction
      pred[i] <- trend_pred + kriging_result$var1.pred
    }
  }

  # Calculate error metrics
  mse <- mean((data$Temp - pred)^2)
  mae <- mean(abs(data$Temp - pred))

  return(list(predictions = pred, mse = mse, mae = mae))
}

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

idw_results <- list()
powers <- c(1, 2, 3)

for (p in powers) {
  idw_results[[paste("power", p, sep = "_")]] <- loocv(weather, method = "idw", power = p)
}

# # Print IDW results
# for (p in powers) {
#   cat("IDW with power =", p, ":\n")
#   cat("MSE:", idw_results[[paste("power", p, sep = "_")]]$mse, "\n")
#   cat("MAE:", idw_results[[paste("power", p, sep = "_")]]$mae, "\n\n")
# }


## ----eval=FALSE-------------------------------------------------------------------------------
```

```
## # Neural Network -------------------------
##
## weather <- weather.dat |>
##   select(Sid, lon, lat, Temp, WS, RH, Psea)
##
## # Split data into training and testing
## set.seed(123)
## data_split <- initial_split(weather, prop = 0.8)
## train_data <- training(data_split)
## test_data <- testing(data_split)
##
## # Create cross-validation folds
## set.seed(123)
## cross_val <- vfold_cv(train_data, v = 10)
##
## # Define recipe
## weather_recipe <- recipe(Temp ~ lon + lat + WS + RH + Psea, data = train_data) |>
##   step_normalize(all_numeric_predictors())
##
## # Define model with tuning parameters
## nn_model <- mlp(
##   hidden_units = tune(),
##   penalty = tune()
## ) |>
##   set_engine("nnet") |>
##   set_mode("regression")
##
## # Create workflow
## nn_workflow <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(weather_recipe)
##
## # Define tuning grid
## nn_grid <- grid_random(
##   hidden_units(range = c(5, 50)),
##   penalty(range = c(0, 0.1)),
##   size = 50
## )
##
## # Tune model
## set.seed(123)
## results <- tune_grid(
##   nn_workflow,
##   resamples = cross_val,
##   grid = nn_grid
## )
##
## # Select best model
## best_params <- select_best(results, metric = "rmse")
##
## # Finalize workflow
## final_nn <- finalize_workflow(nn_workflow, best_params)
##
## # Fit final model on training data
```

```
## final_fit <- fit(final_nn, data = train_data)
##
## # Predict on test data
## test_preds <- predict(final_fit, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## metrics <- metrics(test_preds, truth = Temp, estimate = .pred)
##
## metrics[1, 1] <- "mse"
## metrics[1, 3] <- metrics[1, 3]^2
##
## # Neural Network with lon/lat Only -------------------------
##
## weather_recipe2 <- recipe(Temp ~ lon + lat, data = train_data) |>
##   step_normalize(all_numeric_predictors())
##
## # Create workflow
## nn_workflow2 <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(weather_recipe2)
##
## # Tune model
## set.seed(123)
## results2 <- tune_grid(
##   nn_workflow,
##   resamples = cross_val,
##   grid = nn_grid
##   )
##
## # Select best model
## best_params2 <- select_best(results2, metric = "rmse")
##
## # Finalize workflow
## final_nn2 <- finalize_workflow(nn_workflow2, best_params)
##
## # Fit final model on training data
## final_fit2 <- fit(final_nn2, data = train_data)
##
## # Predict on test data
## test_preds2 <- predict(final_fit2, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## metrics2 <- metrics(test_preds2, truth = Temp, estimate = .pred)
##
## metrics2[1, 1] <- "mse"
## metrics2[1, 3] <- metrics2[1, 3]^2


## ------------------------------------------------------------------------------------------------
load("nn_results.RData")

idw_mse <- numeric(3)
```

```
idw_mae <- numeric(3)

for (p in 1:3) {
  idw_mse[p] <- idw_results[[paste("power", p, sep = "_")]]$mse
  idw_mae[p] <- idw_results[[paste("power", p, sep = "_")]]$mae
}

temp_interp_metrics <- tibble(
  "Method" = c("Spatial Regression", "LOOCV OK", "LOOCV UK", "LOOCV OK Bayes",
               paste0("IDW (p = ", 1:3, ")"), "NN (Lon/Lat Only)", "NN All Preds."),
  "MSE" = c(mean(expanded_model$residuals^2), mse(ok_cv$residual), mse(uk_cv$residual),
            mse(ok_cv_bayes$residual), idw_mse, metrics2[1, 3] |> unlist(), metrics[1, 3] |> unlist()),
  "MAE" = c(mean(abs(expanded_model$residuals)), mae(ok_cv$residual), mae(uk_cv$residual),
            mae(ok_cv_bayes$residual), idw_mae, metrics2[3, 3] |> unlist(), metrics[3, 3] |> unlist())
) |> arrange(MSE)

# write_rds(temp_interp_metrics, "temp_interp_metrics.rds")


## ----results='asis', fig.height=3-----------------------------------------------------------
# 4 ---------------------------------
# Wind Prediction

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

# Regression ------------------------
WS_spatial_trend <- lm(WS ~ lon + lat, data = weather)
WS_spatial_trend |> tbl_regression()

# Visualize residuals spatially
weather$WS_spatial_residuals <- residuals(WS_spatial_trend)
ggplot(weather, aes(x = lon, y = lat, color = WS_spatial_residuals)) +
  geom_point() +
  scale_color_viridis(option = "magma") +
  labs(title = "Spatial Residuals (WS ~ lon + lat)",
       x = "Longitude", y = "Latitude", color = "Residual") +
  theme_minimal() +
  coord_equal()


## ----results='asis', fig.height=3-----------------------------------------------------------

expanded_model <- lm(WS ~ lon + lat + Temp + RH + Psea, data = weather)
expanded_model |> tbl_regression()

weather$expanded_residuals <- residuals(expanded_model)
ggplot(weather, aes(x = fitted(expanded_model), y = expanded_residuals)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values", y = "Residuals",
       caption = "(WS ~ lon + lat + Temp + RH + Psea)") +
```

```
  theme_minimal()


## ------------------------------------------------------------------------------------------------
# Oridnary/Universal Kriging -------------------------
library("gstat")
library("sp")

# Convert wells data to a spatial data frame
coordinates(weather) <- ~ lon + lat

ws_ok_cv <- krige.cv(WS ~ 1, weather,
                  model =  vgm(psill = 0,
                               model = "Sph",
                               range = 2.372928,
                               nugget = 1.338384))

ws_uk_cv <- krige.cv(WS ~ lon + lat + Temp, weather,
                  model =  vgm(psill = 0,
                               model = "Sph",
                               range = 2.372928,
                               nugget = 1.338384))

# mse <- \(residuals) mean(residuals^2)
# mae <- \(residuals) mean(abs(residuals))
# mse(ok_cv$residual)
# mae(ok_cv$residual)
# mse(uk_cv$residual)
# mae(uk_cv$residual)


## ------------------------------------------------------------------------------------------------
wind_bayes_ok <- read_rds("wind_bayes_ok.rds")
wind_ok_cv_bayes <- krige.cv(WS ~ 1, weather,
                  model = vgm(psill = wind_bayes_ok$posterior$sigmasq$summary[[1]],
                               model = "Sph",
                               range = wind_bayes_ok$posterior$sigmasq$summary[[1]],
                               nugget = wind_bayes_ok$posterior$tausq.rel$summary[[1]])
)


## ----eval=FALSE--------------------------------------------------------------------------------
## # Bayesian Kriging -------------------------
##
## wind <- all_geodata$WS_geodata
##
## locations <- as.matrix(weather.dat |> select(lon, lat))
##
## wind_bayes_mod_ok <- model.control(
##   trend.d = "cte",
##   cov.model = "spherical"
## )
##
## wind_bayes_ok <- krige.bayes(
```

```
##   wind,
##   locations = locations[,1:2],
##   model = wind_bayes_mod_ok,
##   prior = bayes_prior
## )
##
## wind_ok_cv_bayes <- krige.cv(WS ~ 1, weather,
##                   model = vgm(psill = wind_bayes_ok$posterior$sigmasq$summary[[1]],
##                               model = "Sph",
##                               range = wind_bayes_ok$posterior$sigmasq$summary[[1]],
##                               nugget = wind_bayes_ok$posterior$tausq.rel$summary[[1]])
## )
##
## mse(wind_ok_cv_bayes$residual)
## mae(wind_ok_cv_bayes$residual)
##


## ------------------------------------------------------------------------------------------------
# Inverse Distance Weighting -------------------------

loocv_wind <- function(data, method, ...) {
  n <- nrow(data)
  pred <- numeric(n)

  for (i in 1:n) {
    train_data <- data[-i, ]
    test_point <- data[i, ]

    if (method == "idw") {
      # IDW method
      power <- list(...)$power
      coords <- train_data[, c("lon", "lat")]
      values <- train_data$WS
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(WS = values))
      idw_model <- gstat::idw(WS ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat")]
                              idp = power)
      pred[i] <- idw_model$var1.pred
    }
    else if (method == "regression") {
      # Regression method
      formula <- list(...)$formula
      model <- lm(formula, data = train_data)
      pred[i] <- predict(model, newdata = test_point)
    }
    else if (method == "log_regression") {
      # Log-transformed regression
      formula <- list(...)$formula
      model <- lm(formula, data = train_data)
      pred[i] <- exp(predict(model, newdata = test_point))
    }
    else if (method == "kriging") {
      # Ordinary kriging
      coords <- train_data[, c("lon", "lat")]
```

```
      values <- train_data$WS
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(WS = values))

      # Fit variogram
      vgm <- variogram(WS ~ 1, data = sp_data)
      fit_vgm <- fit.variogram(vgm, vgm("Sph"))

      # Kriging prediction
      kriging_result <- krige(WS ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat")]
                              model = fit_vgm)
      pred[i] <- kriging_result$var1.pred
    }
    else if (method == "regression_kriging") {
      # Regression kriging
      formula <- list(...)$formula

      # Fit regression model
      reg_model <- lm(formula, data = train_data)

      # Get residuals
      train_data$residuals <- residuals(reg_model)

      # Predict trend
      trend_pred <- predict(reg_model, newdata = test_point)

      # Kriging on residuals
      coords <- train_data[, c("lon", "lat")]
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(residuals = train_data$residuals))

      # Fit variogram on residuals
      vgm <- variogram(residuals ~ 1, data = sp_data)
      fit_vgm <- fit.variogram(vgm, vgm("Sph"))

      # Kriging prediction of residuals
      kriging_result <- krige(residuals ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon",
                              model = fit_vgm)

      # Combine trend and residual prediction
      pred[i] <- trend_pred + kriging_result$var1.pred
    }
  }

  # Calculate error metrics
  mse <- mean((data$WS - pred)^2)
  mae <- mean(abs(data$WS - pred))

  return(list(predictions = pred, mse = mse, mae = mae))
}

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

idw_results <- list()
powers <- c(1, 2, 3)
```

```
for (p in powers) {
  idw_results[[paste("power", p, sep = "_")]] <- loocv_wind(weather, method = "idw", power = p)
}

# Print IDW results
for (p in powers) {
  cat("IDW with power =", p, ":\n")
  cat("MSE:", idw_results[[paste("power", p, sep = "_")]]$mse, "\n")
  cat("MAE:", idw_results[[paste("power", p, sep = "_")]]$mae, "\n\n")
}
```

```
## ----eval=FALSE-------------------------------------------------------------------------------
## # Wind Neural Networks -------------------------
##
## weather <- weather.dat |>
##   select(Sid, lon, lat, Temp, WS, RH, Psea)
##
## # Split data into training and testing
## set.seed(123)
## data_split <- initial_split(weather, prop = 0.8)
## train_data <- training(data_split)
## test_data <- testing(data_split)
##
## # Create cross-validation folds
## set.seed(123)
## cross_val <- vfold_cv(train_data, v = 10)
##
## # Define recipe
## wind_recipe <- recipe(WS ~ lon + lat + Temp + RH + Psea, data = train_data) |>
##   step_normalize(all_numeric_predictors())
##
## # Define model with tuning parameters
## nn_model <- mlp(
##   hidden_units = tune(),
##   penalty = tune()
## ) |>
##   set_engine("nnet") |>
##   set_mode("regression")
##
## # Create workflow
## ws_nn_workflow <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(wind_recipe)
##
## # Define tuning grid
## ws_nn_grid <- grid_random(
##   hidden_units(range = c(5, 50)),
##   penalty(range = c(0, 0.1)),
##   size = 50
## )
##
```

```
## ws_metrics_set <- metric_set(rmse, mae)
##
## # Tune model
## set.seed(123)
## ws_results <- tune_grid(
##   ws_nn_workflow,
##   resamples = cross_val,
##   grid = ws_nn_grid,
##   metrics = ws_metrics_set
## )
##
## # Select best model
## ws_best_params <- select_best(ws_results, metric = "rmse")
##
## # Finalize workflow
## ws_final_nn <- finalize_workflow(ws_nn_workflow, ws_best_params)
##
## # Fit final model on training data
## ws_final_fit <- fit(ws_final_nn, data = train_data)
##
## # Predict on test data
## ws_test_preds <- predict(ws_final_fit, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## ws_metrics <- metrics(ws_test_preds, truth = WS, estimate = .pred)
##
## ws_metrics[1, 1] <- "mse"
## ws_metrics[1, 3] <- ws_metrics[1, 3]^2
##
## # Neural Network with lon/lat Only -------------------------
##
## wind_recipe2 <- recipe(WS ~ lon + lat, data = train_data) |>
##   step_normalize(all_numeric_predictors())
##
## # Create workflow
## ws_nn_workflow2 <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(wind_recipe2)
##
## # Tune model
## set.seed(123)
## ws_results2 <- tune_grid(
##   ws_nn_workflow2,
##   resamples = cross_val,
##   grid = ws_nn_grid,
##   metrics = ws_metrics_set
## )
##
## # Select best model
## ws_best_params2 <- select_best(ws_results2, metric = "rmse")
##
## # Finalize workflow
## ws_final_nn2 <- finalize_workflow(ws_nn_workflow2, ws_best_params2)
```

```
##
## # Fit final model on training data
## ws_final_fit2 <- fit(ws_final_nn2, data = train_data)
##
## # Predict on test data
## ws_test_preds2 <- predict(ws_final_fit2, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## ws_metrics2 <- metrics(ws_test_preds2, truth = WS, estimate = .pred)
##
## ws_metrics2[1, 1] <- "mse"
## ws_metrics2[1, 3] <- ws_metrics2[1, 3]^2


## ------------------------------------------------------------------------------------------------
load("ws_nn_results.RData")

idw_mse <- numeric(3)
idw_mae <- numeric(3)

for (p in 1:3) {
  idw_mse[p] <- idw_results[[paste("power", p, sep = "_")]]$mse
  idw_mae[p] <- idw_results[[paste("power", p, sep = "_")]]$mae
}

wind_interp_metrics <- tibble(
  "Method" = c("Spatial Regression", "LOOCV OK", "LOOCV UK", "LOOCV OK Bayes",
               paste0("IDW (p = ", 1:3, ")"), "NN (Lon/Lat Only)", "NN All Preds."),
  "MSE" = c(mean(expanded_model$residuals^2), mse(ws_ok_cv$residual), mse(ws_uk_cv$residual),
            mse(wind_ok_cv_bayes$residual), idw_mse, ws_metrics[1, 3] |> unlist(), ws_metrics2[1, 3] |>
  "MAE" = c(mean(abs(expanded_model$residuals)), mae(ws_ok_cv$residual), mae(ws_uk_cv$residual),
            mae(wind_ok_cv_bayes$residual), idw_mae, ws_metrics[3, 3] |> unlist(), ws_metrics2[3, 3] |>
) |> arrange(MSE)

# write_rds(wind_interp_metrics, "wind_interp_metrics.rds")


## ----results='asis', fig.height=3----------------------------------------------------------------
# 4 ---------------------------------
# Humidity Prediction

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

# Regression ------------------------
RH_spatial_trend <- lm(RH ~ lon + lat, data = weather)
RH_spatial_trend |> tbl_regression()

# Visualize residuals spatially
weather$RH_spatial_residuals <- residuals(RH_spatial_trend)
ggplot(weather, aes(x = lon, y = lat, color = RH_spatial_residuals)) +
  geom_point() +
  scale_color_viridis(option = "magma") +
```

```
  labs(title = "Spatial Residuals (RH ~ lon + lat)",
       x = "Longitude", y = "Latitude", color = "Residual") +
  theme_minimal() +
  coord_equal()


## ----results='asis', fig.height=3-----------------------------------------------------------
expanded_model <- lm(RH ~ lon + lat + Temp + WS + Psea, data = weather)
expanded_model |> tbl_regression()

weather$expanded_residuals <- residuals(expanded_model)
ggplot(weather, aes(x = fitted(expanded_model), y = expanded_residuals)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values", y = "Residuals",
       caption = "(RH ~ lon + lat + Temp + WS + Psea)") +
  theme_minimal()


## -------------------------------------------------------------------------------------------
# Oridnary/Universal Kriging -------------------------
library("gstat")
library("sp")
weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

# Convert wells data to a spatial data frame
coordinates(weather) <- ~ lon + lat

rh_ok_cv <- krige.cv(RH ~ 1, weather,
                  model = vgm(psill = 80.71117,
                              model = "Sph",
                              range = 0.9684011,
                              nugget = 12.17654))

rh_uk_cv <- krige.cv(RH ~ lon + lat + Temp + Psea, weather,
                  model = vgm(psill = 80.71117,
                              model = "Sph",
                              range = 0.9684011,
                              nugget = 12.17654))


# mse <- \(residuals) mean(residuals^2)
# mae <- \(residuals) mean(abs(residuals))
# mse(ok_cv$residual)
# mae(ok_cv$residual)
# mse(uk_cv$residual)
# mae(uk_cv$residual)


## -------------------------------------------------------------------------------------------
load("rh_bayes_krig.RData")
```

```
## ----eval=FALSE-----------------------------------------------------------------------------
## # Bayesian Kriging -------------------------
## hum <- all_geodata$RH_geodata
##
## locations <- as.matrix(weather.dat |> select(lon, lat))
##
## hum_bayes_mod_ok <- model.control(
##   trend.d = "cte",
##   cov.model = "spherical"
## )
##
## bayes_prior <- prior.control(
##   beta.prior = "flat",
##   sigmasq.prior = "reciprocal",
##   tausq.rel.prior = "uniform",
##   tausq.rel.discrete = seq(from = 0, to = 1, by = 0.01)
## )
##
## hum_bayes_ok <- krige.bayes(
##   hum,
##   locations = locations[,1:2],
##   model = hum_bayes_mod_ok,
##   prior = bayes_prior
## )
##
## hum_ok_cv_bayes <- krige.cv(RH ~ 1, weather,
##                             model = vgm(psill = hum_bayes_ok$posterior$sigmasq$summary[[1]],
##                                         model = "Sph",
##                                         range = hum_bayes_ok$posterior$sigmasq$summary[[1]],
##                                         nugget = hum_bayes_ok$posterior$tausq.rel$summary[[1]])
## )
##
## mse(ok_cv_bayes$residual)
## mae(ok_cv_bayes$residual)
##


## -------------------------------------------------------------------------------------------
# Inverse Distance Weighting -------------------------
powers <- 1:3

loocv_rh <- function(data, method, ...) {
  n <- nrow(data)
  pred <- numeric(n)

  for (i in 1:n) {
    train_data <- data[-i, ]
    test_point <- data[i, ]

    if (method == "idw") {
      # IDW method
      power <- list(...)$power
```

```
  coords <- train_data[, c("lon", "lat")]
  values <- train_data$RH
  sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(RH = values))
  idw_model <- gstat::idw(RH ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat")]
                          idp = power)
  pred[i] <- idw_model$var1.pred
}
else if (method == "regression") {
  # Regression method
  formula <- list(...)$formula
  model <- lm(formula, data = train_data)
  pred[i] <- predict(model, newdata = test_point)
}
else if (method == "log_regression") {
  # Log-transformed regression
  formula <- list(...)$formula
  model <- lm(formula, data = train_data)
  pred[i] <- exp(predict(model, newdata = test_point))
}
else if (method == "kriging") {
  # Ordinary kriging
  coords <- train_data[, c("lon", "lat")]
  values <- train_data$RH
  sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(RH = values))

  # Fit variogram
  vgm <- variogram(RH ~ 1, data = sp_data)
  fit_vgm <- fit.variogram(vgm, vgm("Sph"))

  # Kriging prediction
  kriging_result <- krige(RH ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat")]
                          model = fit_vgm)
  pred[i] <- kriging_result$var1.pred
}
else if (method == "regression_kriging") {
  # Regression kriging
  formula <- list(...)$formula

  # Fit regression model
  reg_model <- lm(formula, data = train_data)

  # Get residuals
  train_data$residuals <- residuals(reg_model)

  # Predict trend
  trend_pred <- predict(reg_model, newdata = test_point)

  # Kriging on residuals
  coords <- train_data[, c("lon", "lat")]
  sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(residuals = train_data$residuals))

  # Fit variogram on residuals
  vgm <- variogram(residuals ~ 1, data = sp_data)
  fit_vgm <- fit.variogram(vgm, vgm("Sph"))
```

```
      # Kriging prediction of residuals
      kriging_result <- krige(residuals ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon",
                              model = fit_vgm)

      # Combine trend and residual prediction
      pred[i] <- trend_pred + kriging_result$var1.pred
    }
  }

  # Calculate error metrics
  mse <- mean((data$RH - pred)^2)
  mae <- mean(abs(data$RH - pred))

  return(list(predictions = pred, mse = mse, mae = mae))
}

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, RH, RH, Psea)

rh_idw_results <- list()

for (p in powers) {
  rh_idw_results[[paste("power", p, sep = "_")]] <- loocv_rh(weather, method = "idw", power = p)
}

# # Print IDW results
# for (p in powers) {
#   cat("IDW with power =", p, ":\n")
#   cat("MSE:", rh_idw_results[[paste("power", p, sep = "_")]]$mse, "\n")
#   cat("MAE:", rh_idw_results[[paste("power", p, sep = "_")]]$mae, "\n\n")
# }



## ----eval=FALSE-----------------------------------------------------------------------------------
## # Humidity Neural Networks -------------------------
##
## weather <- weather.dat |>
##   select(Sid, lon, lat, Temp, WS, RH, Psea)
##
## # Split data into training and testing
## set.seed(123)
## data_split <- initial_split(weather, prop = 0.8)
## train_data <- training(data_split)
## test_data <- testing(data_split)
##
## # Create cross-validation folds
## set.seed(123)
## cross_val <- vfold_cv(train_data, v = 10)
##
## # Define recipe
## hum_recipe <- recipe(RH ~ lon + lat + Temp + WS + Psea, data = train_data) |>
##   step_normalize(all_numeric_predictors())
```

```
##
## # Define model with tuning parameters
## nn_model <- mlp(
##   hidden_units = tune(),
##   penalty = tune()
## ) |>
##   set_engine("nnet") |>
##   set_mode("regression")
##
## # Create workflow
## hum_nn_workflow <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(hum_recipe)
##
## # Define tuning grid
## hum_nn_grid <- grid_random(
##   hidden_units(range = c(5, 50)),
##   penalty(range = c(0, 0.1)),
##   size = 50
## )
##
## # Tune model
## set.seed(123)
## hum_results <- tune_grid(
##   hum_nn_workflow,
##   resamples = cross_val,
##   grid = hum_nn_grid
## )
##
## # Select best model
## hum_best_params <- select_best(hum_results, metric = "rmse")
##
## # Finalize workflow
## hum_final_nn <- finalize_workflow(hum_nn_workflow, hum_best_params)
##
## # Fit final model on training data
## hum_final_fit <- fit(hum_final_nn, data = train_data)
##
## # Predict on test data
## hum_test_preds <- predict(hum_final_fit, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## hum_metrics <- metrics(hum_test_preds, truth = RH, estimate = .pred)
##
## hum_metrics[1, 1] <- "mse"
## hum_metrics[1, 3] <- hum_metrics[1, 3]^2
##
## # Neural Network with lon/lat Only -------------------------
##
## hum_recipe2 <- recipe(RH ~ lon + lat, data = train_data) |>
##   step_normalize(all_numeric_predictors())
##
## # Create workflow
```

```
## hum_nn_workflow2 <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(hum_recipe2)
##
## # Tune model
## set.seed(123)
## hum_results2 <- tune_grid(
##   hum_nn_workflow2,
##   resamples = cross_val,
##   grid = hum_nn_grid
## )
##
## # Select best model
## hum_best_params2 <- select_best(hum_results2, metric = "rmse")
##
## # Finalize workflow
## hum_final_nn2 <- finalize_workflow(hum_nn_workflow2, hum_best_params2)
##
## # Fit final model on training data
## hum_final_fit2 <- fit(hum_final_nn2, data = train_data)
##
## # Predict on test data
## hum_test_preds2 <- predict(hum_final_fit2, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## hum_metrics2 <- metrics(hum_test_preds2, truth = RH, estimate = .pred)
##
## hum_metrics2[1, 1] <- "mse"
## hum_metrics2[1, 3] <- hum_metrics2[1, 3]^2


## ------------------------------------------------------------------------------------------------
load("rh_nn_results.RData")

idw_mse <- numeric(3)
idw_mae <- numeric(3)

for (p in 1:3) {
  idw_mse[p] <- rh_idw_results[[paste("power", p, sep = "_")]]$mse
  idw_mae[p] <- rh_idw_results[[paste("power", p, sep = "_")]]$mae
}

coordinates(weather) <- ~ lon + lat

hum_ok_cv_bayes <- krige.cv(RH ~ 1, weather,
                            model = vgm(psill = hum_bayes_ok$posterior$sigmasq$summary[[1]],
                                        model = "Sph",
                                        range = hum_bayes_ok$posterior$sigmasq$summary[[1]],
                                        nugget = hum_bayes_ok$posterior$tausq.rel$summary[[1]])
)

RH_interp_metrics <- tibble(
  "Method" = c("Spatial Regression", "LOOCV OK", "LOOCV UK", "LOOCV OK Bayes",
```

```
              paste0("IDW (p = ", 1:3, ")"), "NN (Lon/Lat Only)", "NN All Preds."),
  "MSE" = c(mean(expanded_model$residuals^2), mse(rh_ok_cv$residual), mse(rh_uk_cv$residual),
            mse(hum_ok_cv_bayes$residual), idw_mse, hum_metrics[1, 3] |> unlist(), hum_metrics2[1, 3] |>
  "MAE" = c(mean(abs(expanded_model$residuals)), mae(rh_ok_cv$residual), mae(rh_uk_cv$residual),
            mae(hum_ok_cv_bayes$residual), idw_mae, hum_metrics[3, 3] |> unlist(), hum_metrics2[3, 3] |>
) |> arrange(MSE)



## ----results='asis'-----------------------------------------------------------
# 4 --------------------------------
# Sea-Level Pressure Prediction

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

# Regression ------------------------
Psea_spatial_trend <- lm(Psea ~ lon + lat, data = weather)
Psea_spatial_trend |> tbl_regression()

# Visualize residuals spatially
weather$Psea_spatial_residuals <- residuals(Psea_spatial_trend)
ggplot(weather, aes(x = lon, y = lat, color = Psea_spatial_residuals)) +
  geom_point() +
  scale_color_viridis(option = "magma") +
  labs(title = "Spatial Residuals (Psea ~ lon + lat)",
       x = "Longitude", y = "Latitude", color = "Residual") +
  theme_minimal() +
  coord_equal()


## ----results='asis'-----------------------------------------------------------
expanded_model <- lm(Psea ~ lon + lat + Temp + WS + RH, data = weather)
expanded_model |> tbl_regression()

weather$expanded_residuals <- residuals(expanded_model)
ggplot(weather, aes(x = fitted(expanded_model), y = expanded_residuals)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values", y = "Residuals",
       caption = "(Psea ~ lon + lat + Temp + WS + Psea)") +
  theme_minimal()


## -----------------------------------------------------------------------------
# Oridnary/Universal Kriging -------------------------
library("gstat")
library("sp")
weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

# Convert wells data to a spatial data frame
```

```
coordinates(weather) <- ~ lon + lat

Psea_ok_cv <- krige.cv(Psea ~ 1, weather,
                  model = vgm(psill = 5784.669,
                              model = "Sph",
                              range = 1892.23,
                              nugget = 0.1725412))

Psea_uk_cv <- krige.cv(Psea ~ lon + lat + Temp, weather,
                  model = vgm(psill = 5784.669,
                              model = "Sph",
                              range = 1892.23,
                              nugget = 0.1725412))


# mse <- \(residuals) mean(residuals^2)
# mae <- \(residuals) mean(abs(residuals))
# mse(ok_cv$residual)
# mae(ok_cv$residual)
# mse(uk_cv$residual)
# mae(uk_cv$residual)


## ----eval=FALSE-------------------------------------------------------------------------------
## # Bayesian Kriging -------------------------
## # Bayes
## psea <- all_geodata$Psea_geodata
##
## locations <- as.matrix(weather.dat |> select(lon, lat))
##
## psea_bayes_mod_ok <- model.control(
##   trend.d = "cte",
##   cov.model = "spherical"
## )
##
## bayes_prior <- prior.control(
##   beta.prior = "flat",
##   sigmasq.prior = "reciprocal",
##   tausq.rel.prior = "uniform",
##   tausq.rel.discrete = seq(from = 0, to = 1, by = 0.01)
## )
##
## psea_bayes_ok <- krige.bayes(
##   psea,
##   locations = locations[,1:2],
##   model = psea_bayes_mod_ok,
##   prior = bayes_prior
## )
##
## psea_ok_cv_bayes <- krige.cv(Psea ~ 1, weather,
##                              model = vgm(psill = psea_bayes_ok$posterior$sigmasq$summary[[1]],
##                                          model = "Sph",
##                                          range = psea_bayes_ok$posterior$sigmasq$summary[[1]],
##                                          nugget = psea_bayes_ok$posterior$tausq.rel$summary[[1]])
```

```
## )


## ------------------------------------------------------------------------------------------------
load("Psea_bayes_krig.RData")


## ------------------------------------------------------------------------------------------------
# Inverse Distance Weighting -------------------------

powers <- 1:3

loocv_Psea <- function(data, method, ...) {
  n <- nrow(data)
  pred <- numeric(n)

  for (i in 1:n) {
    train_data <- data[-i, ]
    test_point <- data[i, ]

    if (method == "idw") {
      # IDW method
      power <- list(...)$power
      coords <- train_data[, c("lon", "lat")]
      values <- train_data$Psea
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(Psea = values))
      idw_model <- gstat::idw(Psea ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat
                              idp = power)
      pred[i] <- idw_model$var1.pred
    }
    else if (method == "regression") {
      # Regression method
      formula <- list(...)$formula
      model <- lm(formula, data = train_data)
      pred[i] <- predict(model, newdata = test_point)
    }
    else if (method == "log_regression") {
      # Log-transformed regression
      formula <- list(...)$formula
      model <- lm(formula, data = train_data)
      pred[i] <- exp(predict(model, newdata = test_point))
    }
    else if (method == "kriging") {
      # Ordinary kriging
      coords <- train_data[, c("lon", "lat")]
      values <- train_data$Psea
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(Psea = values))

      # Fit variogram
      vgm <- variogram(Psea ~ 1, data = sp_data)
      fit_vgm <- fit.variogram(vgm, vgm("Sph"))

      # Kriging prediction
      kriging_result <- krige(Psea ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon", "lat
```

```
                            model = fit_vgm)
      pred[i] <- kriging_result$var1.pred
    }
    else if (method == "regression_kriging") {
      # Regression kriging
      formula <- list(...)$formula

      # Fit regression model
      reg_model <- lm(formula, data = train_data)

      # Get residuals
      train_data$residuals <- residuals(reg_model)

      # Predict trend
      trend_pred <- predict(reg_model, newdata = test_point)

      # Kriging on residuals
      coords <- train_data[, c("lon", "lat")]
      sp_data <- sp::SpatialPointsDataFrame(coords, data.frame(residuals = train_data$residuals))

      # Fit variogram on residuals
      vgm <- variogram(residuals ~ 1, data = sp_data)
      fit_vgm <- fit.variogram(vgm, vgm("Sph"))

      # Kriging prediction of residuals
      kriging_result <- krige(residuals ~ 1, sp_data, newdata = sp::SpatialPoints(test_point[, c("lon",
                            model = fit_vgm)

      # Combine trend and residual prediction
      pred[i] <- trend_pred + kriging_result$var1.pred
    }
  }

  # Calculate error metrics
  mse <- mean((data$Psea - pred)^2)
  mae <- mean(abs(data$Psea - pred))

  return(list(predictions = pred, mse = mse, mae = mae))
}

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

Psea_idw_results <- list()

for (p in powers) {
  Psea_idw_results[[paste("power", p, sep = "_")]] <- loocv_Psea(weather, method = "idw", power = p)
}

# Print IDW results
# for (p in powers) {
#   cat("IDW with power =", p, ":\n")
#   cat("MSE:", Psea_idw_results[[paste("power", p, sep = "_")]]$mse, "\n")
#   cat("MAE:", Psea_idw_results[[paste("power", p, sep = "_")]]$mae, "\n\n")
```

```
# }



## ----eval=FALSE-------------------------------------------------------------------------------
## # Humidity Neural Networks -------------------------
##
## weather <- weather.dat |>
##   select(Sid, lon, lat, Temp, WS, RH, Psea)
##
## # Split data into training and testing
## set.seed(123)
## data_split <- initial_split(weather, prop = 0.8)
## train_data <- training(data_split)
## test_data <- testing(data_split)
##
## # Create cross-validation folds
## set.seed(123)
## cross_val <- vfold_cv(train_data, v = 10)
##
## # Define recipe
## Psea_recipe <- recipe(Psea ~ lon + lat + Temp + WS + Psea, data = train_data) |>
##   step_normalize(all_numeric_predictors())
##
## # Define model with tuning parameters
## nn_model <- mlp(
##   hidden_units = tune(),
##   penalty = tune()
## ) |>
##   set_engine("nnet") |>
##   set_mode("regression")
##
## # Create workflow
## Psea_nn_workflow <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(Psea_recipe)
##
## # Define tuning grid
## Psea_nn_grid <- grid_random(
##   hidden_units(range = c(5, 50)),
##   penalty(range = c(0, 0.1)),
##   size = 50
## )
##
## # Tune model
## set.seed(123)
## Psea_results <- tune_grid(
##   Psea_nn_workflow,
##   resamples = cross_val,
##   grid = Psea_nn_grid
## )
##
## # Select best model
## Psea_best_params <- select_best(Psea_results, metric = "rmse")
```

```
##
## # Finalize workflow
## Psea_final_nn <- finalize_workflow(Psea_nn_workflow, Psea_best_params)
##
## # Fit final model on training data
## Psea_final_fit <- fit(Psea_final_nn, data = train_data)
##
## # Predict on test data
## Psea_test_preds <- predict(Psea_final_fit, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## Psea_metrics <- metrics(Psea_test_preds, truth = Psea, estimate = .pred)
##
## Psea_metrics[1, 1] <- "mse"
## Psea_metrics[1, 3] <- Psea_metrics[1, 3]^2
##
## # Neural Network with lon/lat Only -------------------------
##
## Psea_recipe2 <- recipe(Psea ~ lon + lat, data = train_data) |>
##   step_normalize(all_numeric_predictors())
##
## # Create workflow
## Psea_nn_workflow2 <- workflow() |>
##   add_model(nn_model) |>
##   add_recipe(Psea_recipe2)
##
## # Tune model
## set.seed(123)
## Psea_results2 <- tune_grid(
##   Psea_nn_workflow2,
##   resamples = cross_val,
##   grid = Psea_nn_grid
## )
##
## # Select best model
## Psea_best_params2 <- select_best(Psea_results2, metric = "rmse")
##
## # Finalize workflow
## Psea_final_nn2 <- finalize_workflow(Psea_nn_workflow2, Psea_best_params2)
##
## # Fit final model on training data
## Psea_final_fit2 <- fit(Psea_final_nn2, data = train_data)
##
## # Predict on test data
## Psea_test_preds2 <- predict(Psea_final_fit2, test_data) |>
##   bind_cols(test_data)
##
## # Evaluate model
## Psea_metrics2 <- metrics(Psea_test_preds2, truth = Psea, estimate = .pred)
##
## Psea_metrics2[1, 1] <- "mse"
## Psea_metrics2[1, 3] <- Psea_metrics2[1, 3]^2
```

```
## ----------------------------------------------------------------------------------------------
load("Psea_results.RData")

idw_mse <- numeric(3)
idw_mae <- numeric(3)

for (p in 1:3) {
  idw_mse[p] <- Psea_idw_results[[paste("power", p, sep = "_")]]$mse
  idw_mae[p] <- Psea_idw_results[[paste("power", p, sep = "_")]]$mae
}

weather <- weather.dat |>
  select(Sid, lon, lat, Temp, WS, RH, Psea)

coordinates(weather) <- ~ lon + lat

Psea_ok_cv_bayes <- krige.cv(Psea ~ 1, weather,
                             model = vgm(psill = psea_bayes_ok$posterior$sigmasq$summary[[1]],
                                         model = "Sph",
                                         range = psea_bayes_ok$posterior$sigmasq$summary[[1]],
                                         nugget = psea_bayes_ok$posterior$tausq.rel$summary[[1]]))

Psea_interp_metrics <- tibble(
  "Method" = c("Spatial Regression", "LOOCV OK", "LOOCV UK", "LOOCV OK Bayes",
               paste0("IDW (p = ", 1:3, ")"), "NN (Lon/Lat Only)", "NN All Preds."),
  "MSE" = c(mean(expanded_model$residuals^2), mse(Psea_ok_cv$residual), mse(Psea_uk_cv$residual),
            mse(Psea_ok_cv_bayes$residual), idw_mse, Psea_metrics[1, 3] |> unlist(), Psea_metrics2[1, 3]
  "MAE" = c(mean(abs(expanded_model$residuals)), mae(Psea_ok_cv$residual), mae(Psea_uk_cv$residual),
            mae(Psea_ok_cv_bayes$residual), idw_mae, Psea_metrics[3, 3] |> unlist(), Psea_metrics2[3, 3]
) |> arrange(MSE)

# write_rds(Psea_interp_metrics, "Psea_interp_metrics.rds")


## ----temp metrics, results='asis'--------------------------------------------------------------
# Tables ----------------------
# These table data frames were made in (4)
temp_interp_metrics |> knitr::kable()


## ----ws metrics, results='asis'----------------------------------------------------------------
wind_interp_metrics |> knitr::kable()


## ----rh metrics, results='asis'----------------------------------------------------------------
RH_interp_metrics |> knitr::kable()


## ----psea metrics, results='asis'--------------------------------------------------------------
Psea_interp_metrics |> knitr::kable()
```