**Explanation of R codes**

**Step 1 Reading NetCDF Files**

fname <- paste0("D:/Kitaplar/METU-PHD/Thesis/IsmailHocandanAldim\_Aksoy\_27092023/",

"aswout/wrfout\_d01\_2004-08-11\_00\_00\_00")

nc\_data <- nc\_open(fname)

This code reads a NetCDF file (**fname**) using the **nc\_open** function from the **ncdf4** package. NetCDF is a file format commonly used for storing multidimensional scientific data.

**Step 2: Extracting Data from NetCDF Files**

long <- ncvar\_get(nc\_data, "XLONG")

lat <- ncvar\_get(nc\_data, "XLAT", verbose = FALSE)

temp <- ncvar\_get(nc\_data, "T2")

t <- ncvar\_get(nc\_data, "Times")

Here, the code extracts specific variables (longitude, latitude, temperature, and time) from the NetCDF file using the **ncvar\_get** function.

**Step 3: Creating Raster Objects**

raster\_temp <- list()

for (i in 1:dim(temp)[3]) {

raster\_temp[[i]] <- raster(t(temp[, , i] - 273.15),

xmn = min(long), xmx = max(long),

ymn = min(lat), ymx = max(lat),

crs = CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no\_defs+ towgs84=0,0,0"))

}

This code creates a list of raster objects for each time step from the extracted temperature data. Raster objects are used for working with gridded spatial data.

**Step 4: Data Visualization**

temp\_df <- as.data.frame(raster\_temp[[length(t)]], xy = TRUE)

world <- rnaturalearth::ne\_countries(scale='medium', returnclass = 'sf')

ggplot(data = world) + geom\_sf(fill = "white") +

coord\_sf(crs = st\_crs(4326), xlim = c(10, 55), ylim = c(30,50)) +

geom\_raster(data = temp\_df, aes(x, y, fill = layer), alpha=0.6) +

scale\_fill\_viridis\_c(limits = c(0, 35)) +

labs(x="", y="", fill= expression(degree\*C)) +

ggtitle("Coverage of Domain 1") + theme(legend.key.height = unit(1, "cm"))

This code uses the **ggplot2** package to create a map visualization. It overlays the temperature data onto a map of countries, setting up appropriate coordinate systems and color scales.

**Step 5: Reading and Processing Data from Another NetCDF File**

fname2 <- paste0("D:/Kitaplar/METU-PHD/Thesis/IsmailHocandanAldim\_Aksoy\_27092023/",

"aswout/wrfout\_d02\_2004-08-11\_00\_00\_00")

nc\_data2 <- nc\_open(fname2)

long\_2 <- ncvar\_get(nc\_data2, "XLONG")

lat\_2 <- ncvar\_get(nc\_data2, "XLAT", verbose = FALSE)

temp\_2 <- ncvar\_get(nc\_data2, "T2")

Similar to Step 1 and 2, this code reads and extracts data from another NetCDF file.

**Step 6: Reading Meteorological Station Data**

df\_gauges <- read.delim("D:/Kitaplar/METU-PHD/COURSES/3-TERM/STAT570/STAT\_570\_FINAL\_PROJECT\_MAKSOY-SAKIL/gauges.txt", sep="|")

This code reads data from a delimited text file containing information about meteorological stations.

**Step 7: Data Wrangling and Visualization**

df\_gauges <- df\_gauges[,-c(3,4)]

colnames(df\_gauges) <- c("Station","Province","Latitude","Longitude","Altitude")

df\_gauges$Province <- tolower(df\_gauges$Province) |> str\_to\_title()

df\_gauges <- df\_gauges |> arrange(Station)

df\_gauges |> gt()

This code performs data wrangling and cleaning on the meteorological station data, including renaming columns, converting province names, and arranging the data. It uses the **dplyr** and **gt** packages for data manipulation and table creation.

**Step 8: Extracting Temperature Observations**

temp\_obs <- read\_excel("D:/Kitaplar/METU-PHD/COURSES/3-TERM/STAT570/STAT\_570\_FINAL\_PROJECT\_MAKSOY-SAKIL/df\_2023122096C0-Saatlik\_Sicaklik.xlsx")

This code reads temperature observations from an Excel

**Step 9: Extracting Temperature Values for Meteorological Stations**

raster\_temp\_stack <- stack(raster\_temp)

raster\_temp\_value <- raster::extract(raster\_temp\_stack, centroids)

This code stacks raster layers and extracts temperature values for meteorological station locations.

**Step 10: Merging Data Frames and Reshaping**

new\_df\_list <- list()

for (i in 1:length(variable)) {

new\_df\_list[[i]] <- data\_list[[i]] |> distinct(Station, .keep\_all = TRUE) |> pivot\_longer(...)

}

This code merges and reshapes data frames for assimilated and non-assimilated predictions using the **dplyr** package.

**Step 11: Error Analysis**

This part calculates various error metrics, including bias, mean squared error (MSE), root mean squared error (RMSE), normalized RMSE (NRMSE), and correlation coefficients for the predictions.

**Step 12: Displaying Error Statistics**

error\_table |> gt()

Finally, this code displays error statistics tables using **the gt** package. These steps collectively demonstrate a comprehensive workflow for loading, processing, and analyzing meteorological data, including visualization and error analysis.