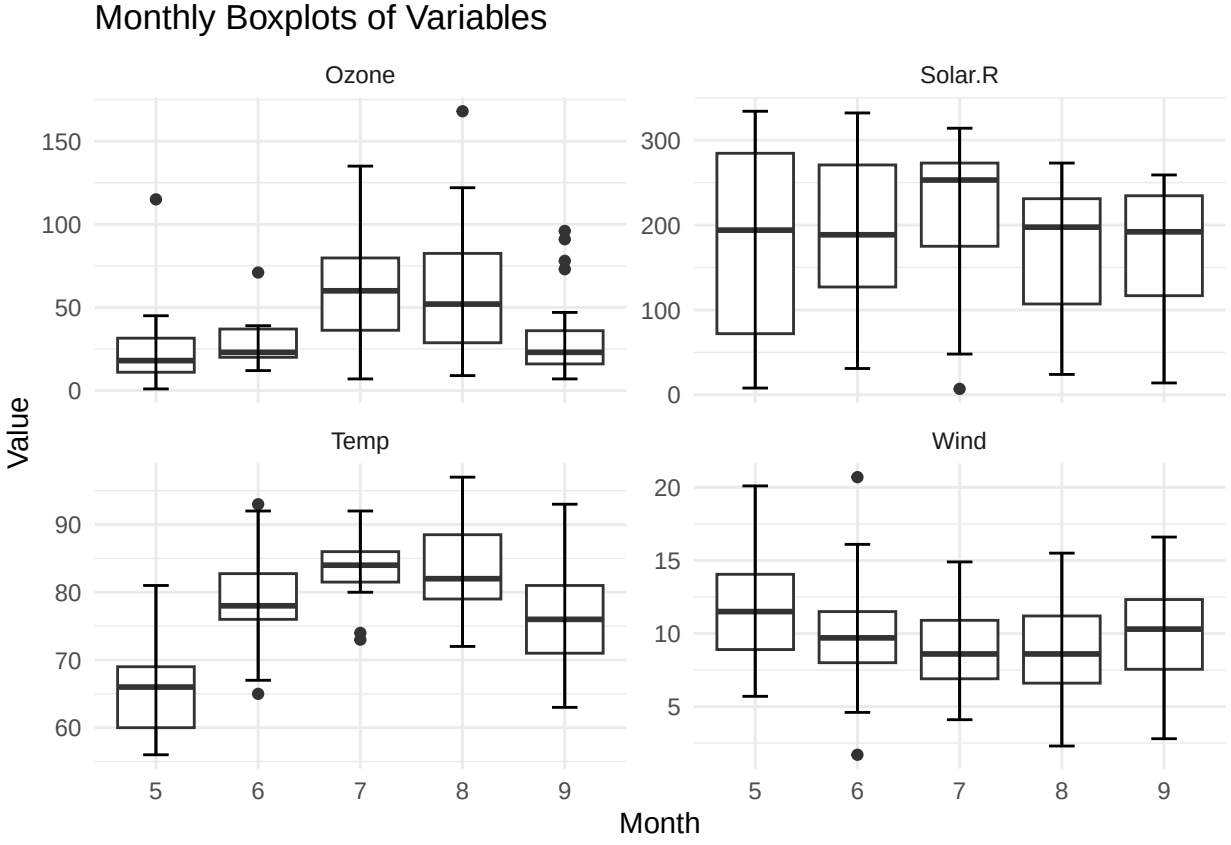
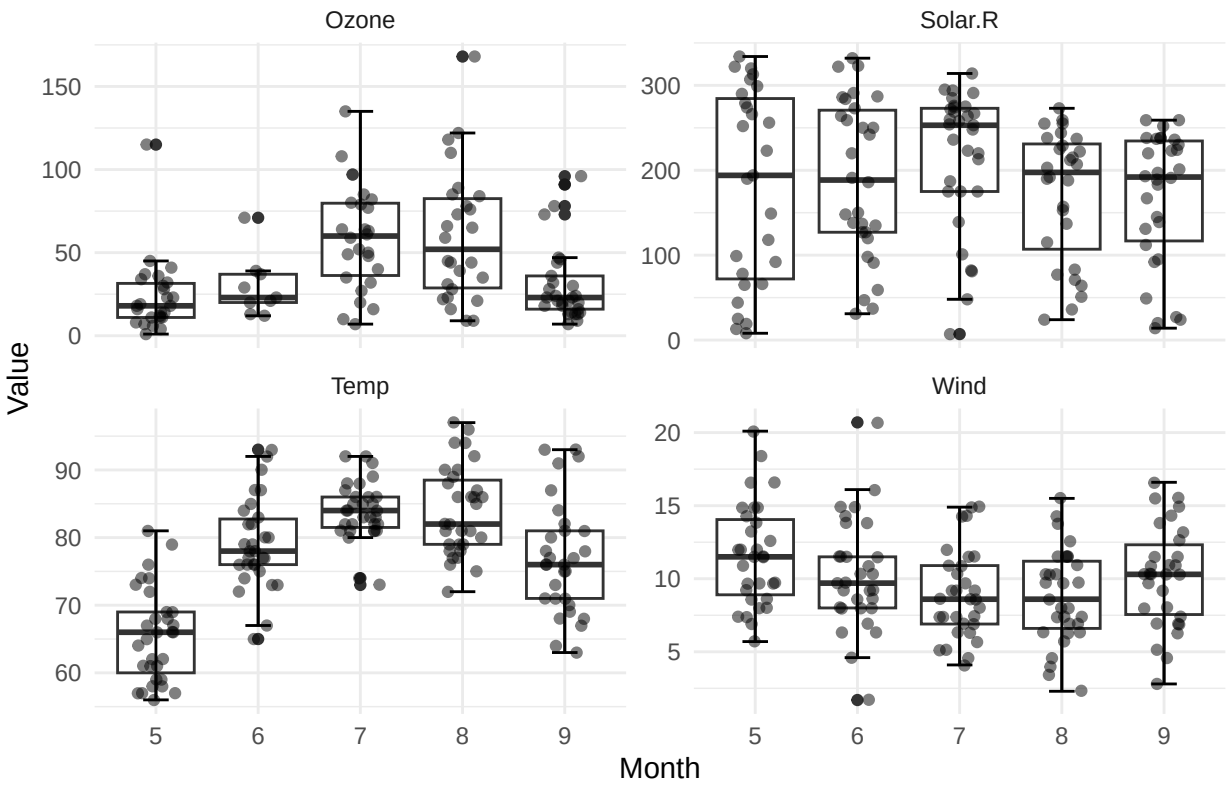
re2_getting creative with ozone concentration

keerthi

2025-03-10

```
## [1] "C:/Users/Keetz/OneDrive/Desktop/Economics/AGDS 1/agds_report_keerthi/data/RE2_ozone_data.csv"
```






```
## # A tibble: 5 x 5
##   Month mean_Ozone mean_Solar mean_Wind mean_Temp
##   <int>      <dbl>      <dbl>     <dbl>     <dbl>
## 1     5       23.6       182.     11.6       65.5
## 2     6       29.4       190.     10.3       79.1
## 3     7       59.1       216.      8.94      83.9
## 4     8       60.0       173.      8.79      84.0
## 5     9       31.4       167.     10.2       76.9
```

```
## # A tibble: 5 x 5
##   Month median_Ozone median_Solar median_Wind median_Temp
##   <int>        <dbl>        <dbl>       <dbl>       <dbl>
## 1     5           18         186.        11.5          66
## 2     6           23         188.         9.7          78
## 3     7           60         253          8.6          84
## 4     8           52         190          8.6          82
## 5     9           23         192         10.3          76
```

```
## # A tibble: 5 x 5
##   Month range_Ozone range_Solar range_Wind range_Temp
##   <int>       <int>       <dbl>      <dbl>      <int>
## 1     5         114         326       14.4         25
## 2     6          59         301       19           28
## 3     7         128         307       10.8         19
## 4     8         159         249       13.2         25
## 5     9          89         245       13.8         30
```

```
## # A tibble: 5 x 13
```

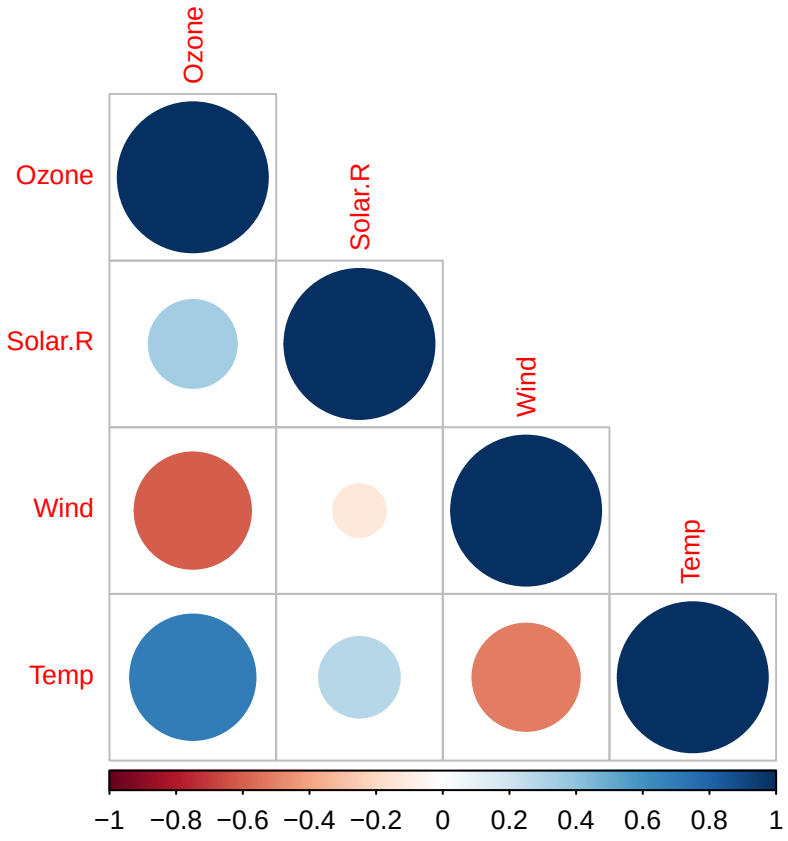
```
##   Month max_Ozone min_Ozone range_Ozone max_Solar min_Solar range_Solar max_Wind
##   <int>     <int>     <int>       <int>     <dbl>     <dbl>       <dbl>    <dbl>
## 1     5       115         1         114       334         8         326     20.1
## 2     6        71        12          59       332        31         301     20.7
## 3     7       135         7         128       314         7         307     14.9
## 4     8       168         9         159       273        24         249     15.5
## 5     9        96         7          89       259        14         245     16.6
## # i 5 more variables: min_Wind <dbl>, range_Wind <dbl>, max_Temp <int>,
## #   min_Temp <int>, range_Temp <int>
```

```
## [1] 0.3436702
```

```
## [1] -0.6015465
```

```
## [1] 0.6983603
```

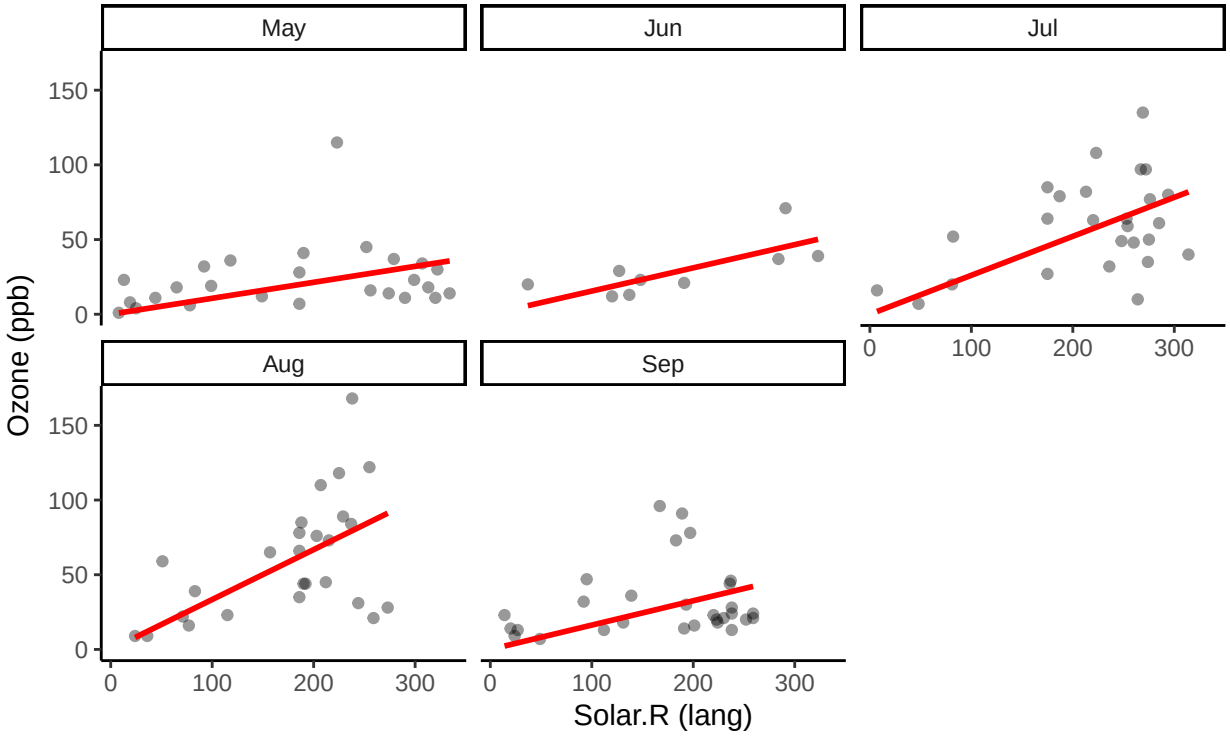
```
##              Ozone    Solar.R       Wind       Temp
## Ozone    1.0000000  0.3436702 -0.6015465  0.6983603
## Solar.R  0.3436702  1.0000000 -0.1240564  0.2890717
## Wind    -0.6015465 -0.1240564  1.0000000 -0.5110750
## Temp     0.6983603  0.2890717 -0.5110750  1.0000000
```



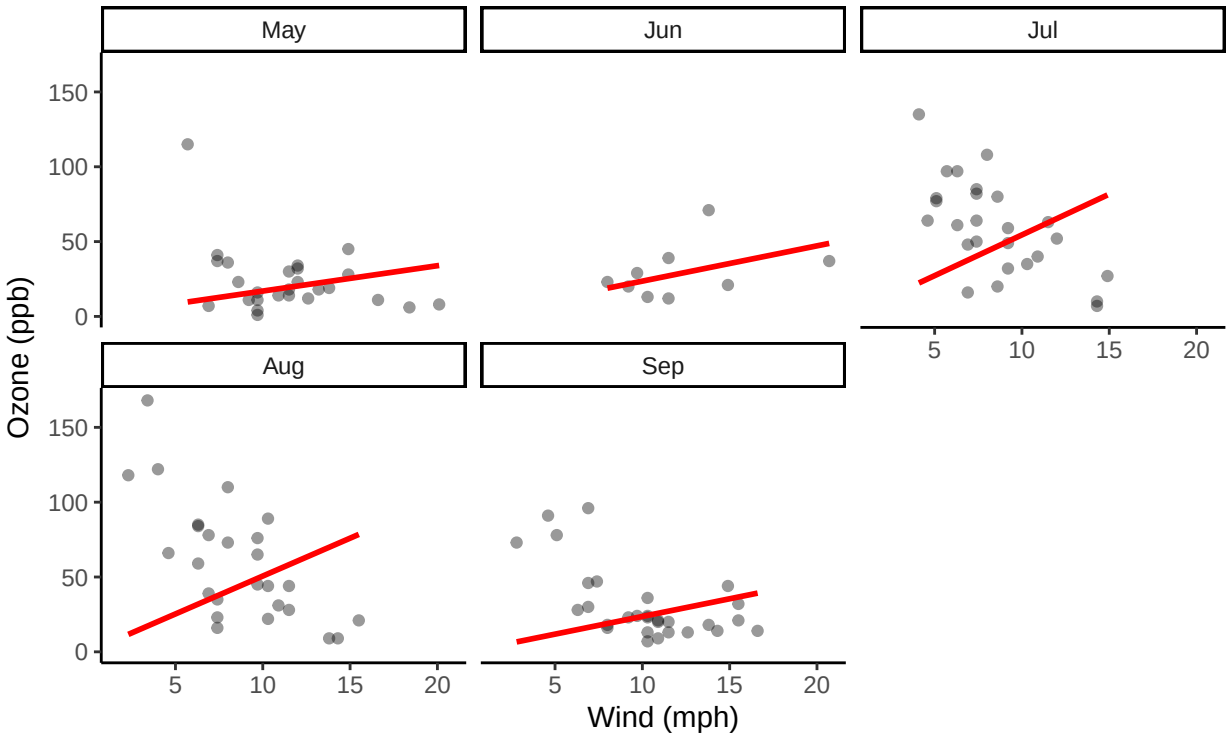

```
##
## Call:
## lm(formula = Ozone ~ Solar.R + Wind + Temp, data = ozone_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -39.767 -15.005  -3.047  10.016  97.384
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -68.21641   23.02746  -2.962  0.00373 **
## Solar.R       0.06047    0.02331   2.595  0.01074 *
## Wind         -3.10383    0.64731  -4.795 5.04e-06 ***
## Temp          1.66659    0.25287   6.591 1.49e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.32 on 112 degrees of freedom
## Multiple R-squared:  0.5932, Adjusted R-squared:  0.5823
## F-statistic: 54.43 on 3 and 112 DF,  p-value: < 2.2e-16
```

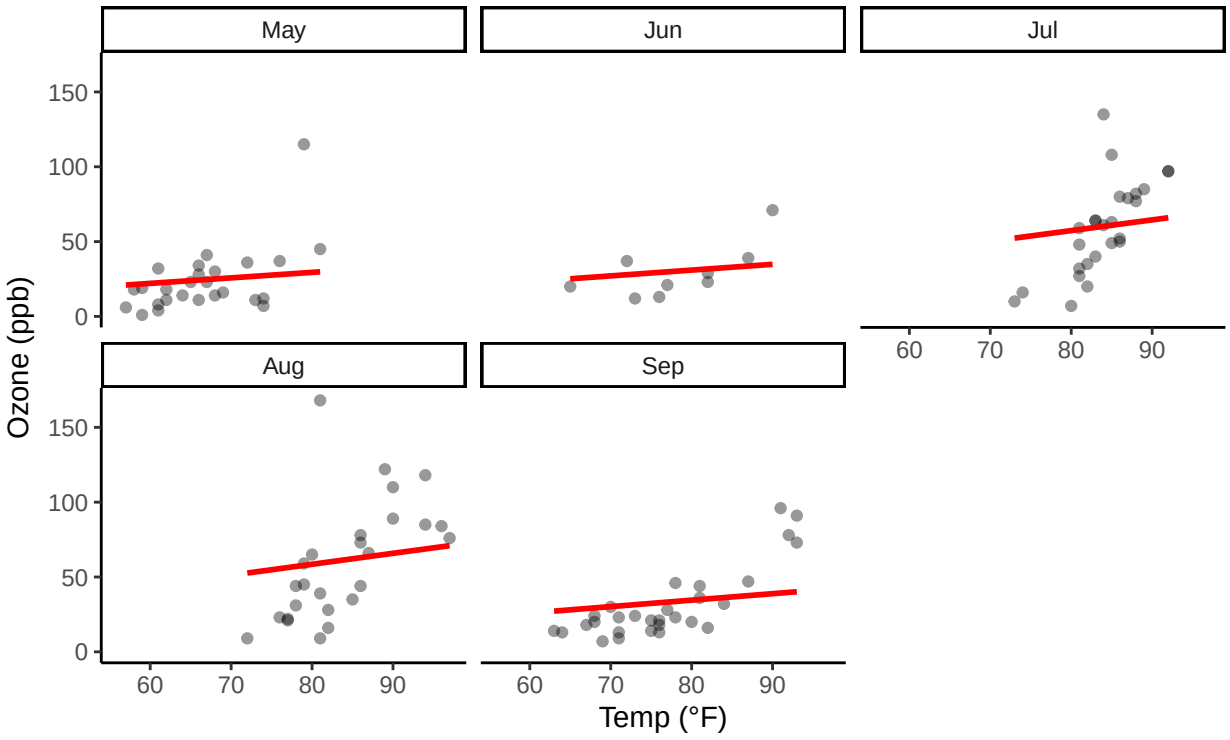
```
##  Solar.R     Wind     Temp
## 1.092086 1.354664 1.455435
```



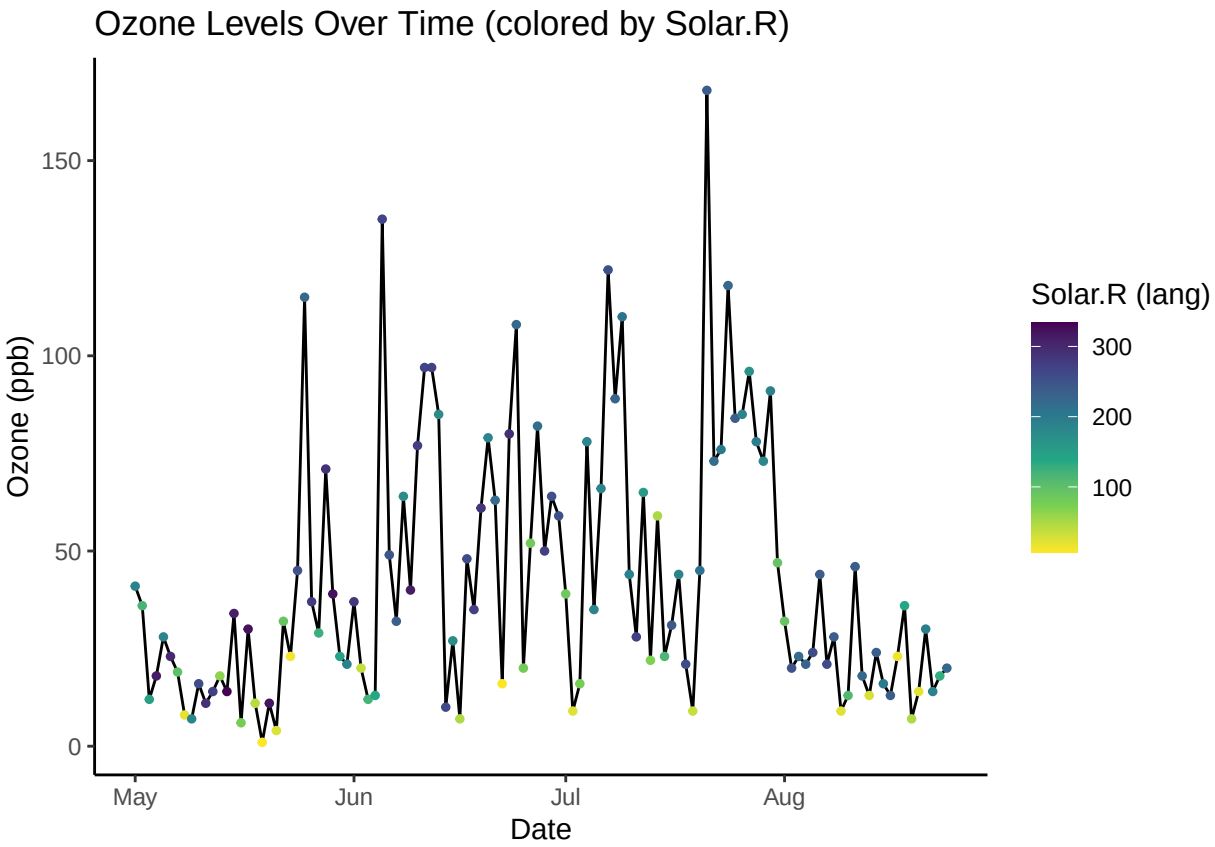
Scatter plot of Ozone vs Solar Radiation for each month from May to September. Red lines show linear regression fits with zero intercept.

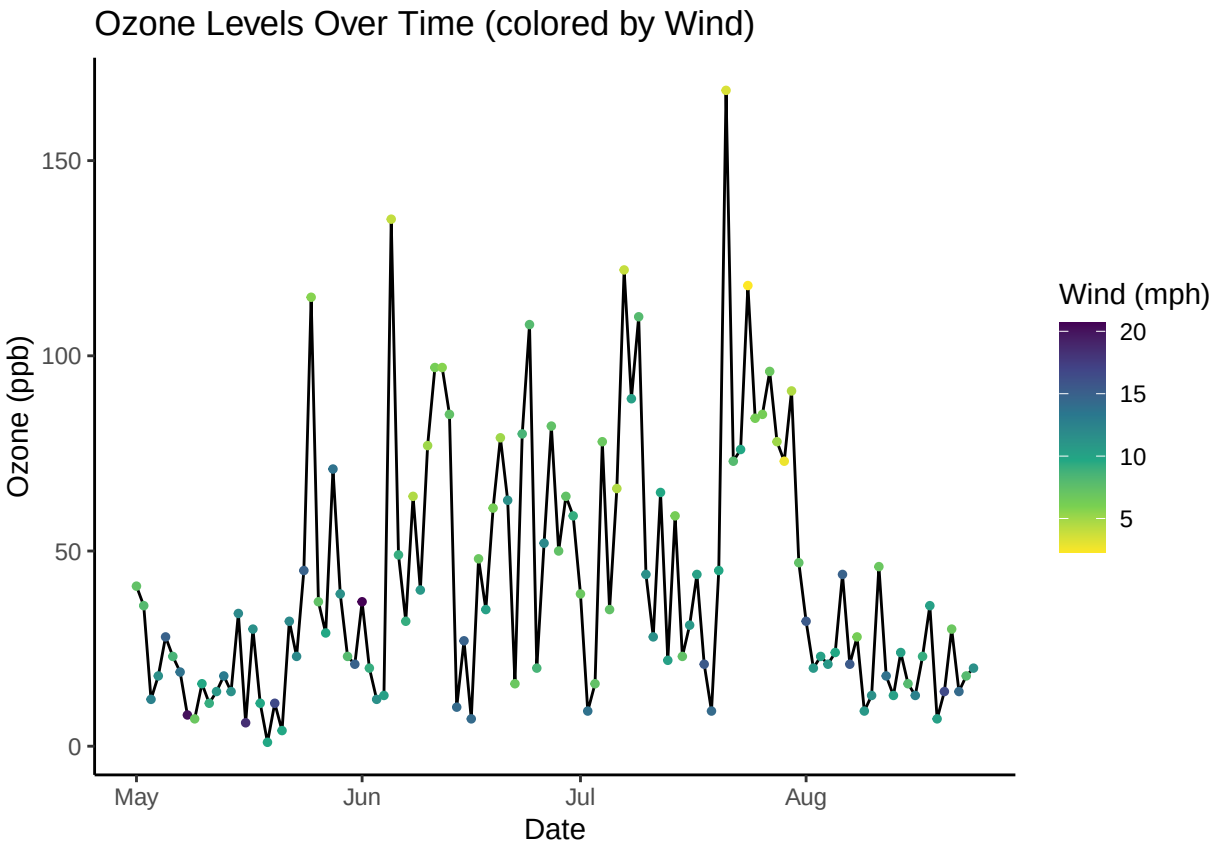


Scatter plot of Ozone vs Wind for each month from May to September.
Red lines show linear regression fits with zero intercept.



Scatter plot of Ozone vs Temperature for each month from May to September. Red lines show linear regression fits with zero intercept.


Ozone Levels Over Time (colored by Solar.R)
Ozone (ppb)
150
100
50
0
May
Jun
Jul
Aug
Date
Solar.R (lang)
300
200
100


Ozone Levels Over Time (colored by Wind)
Ozone (ppb)
150
100
50
0
May
Jun
Jul
Aug
Date
Wind (mph)
20
15
10
5

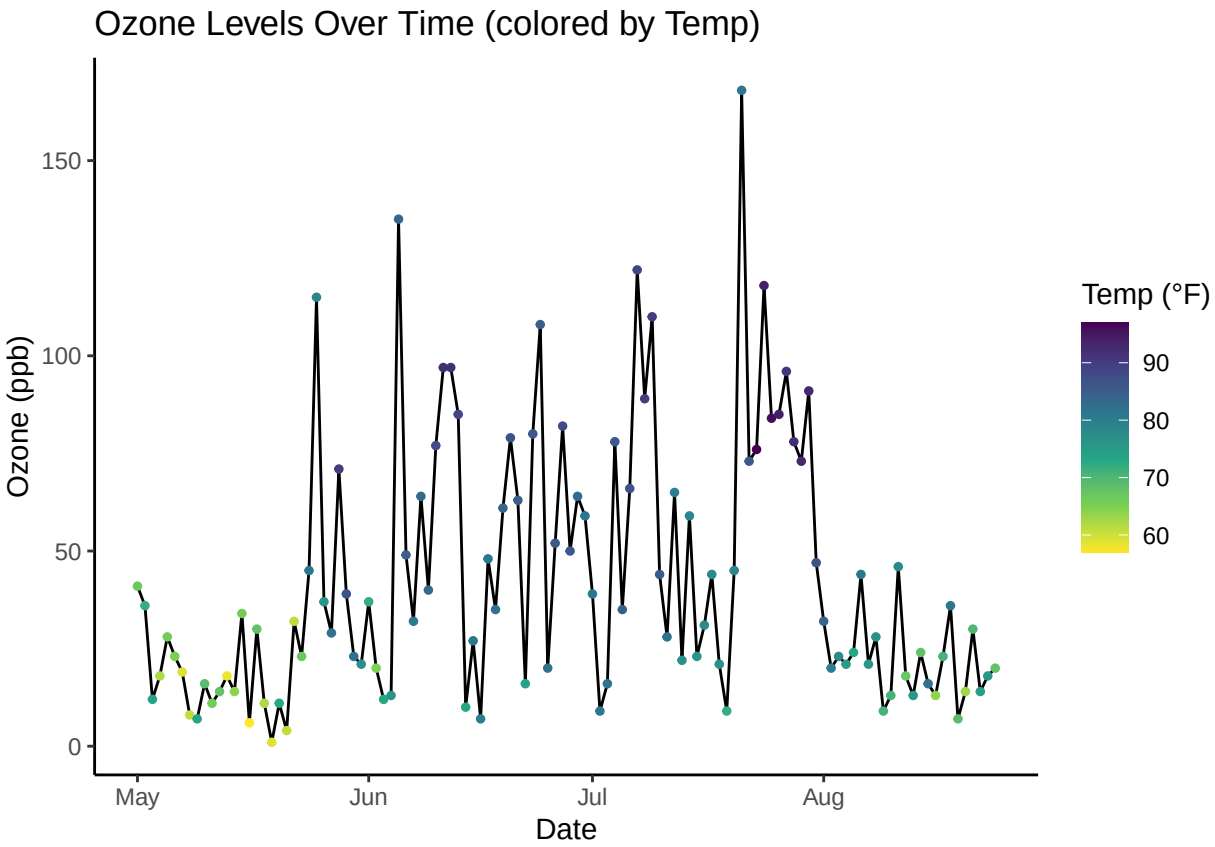


Table 1: Table 1: Monthly Mean of Ozone, Solar Radiation, Wind, and Temperature.

Month	mean_Ozone	mean_Solar	mean_Wind	mean_Temp
5	23.61538	181.8944	11.622581	65.54839
6	29.44444	190.1667	10.266667	79.10000
7	59.11538	216.4839	8.941935	83.90323
8	59.96154	173.2192	8.793548	83.96774
9	31.44828	167.4333	10.180000	76.90000

Table 2: Table 2: Monthly Median of Ozone, Solar Radiation, Wind, and Temperature.

Month	median_Ozone	median_Solar	median_Wind	median_Temp
5	18	185.9315	11.5	66
6	23	188.5000	9.7	78
7	60	253.0000	8.6	84
8	52	190.0000	8.6	82
9	23	192.0000	10.3	76

Table 3: Table 3: Monthly Ranges of Ozone, Solar Radiation, Wind, and Temperature.

Month	range_Ozone	range_Solar	range_Wind	range_Temp
5	114	326	14.4	25
6	59	301	19.0	28
7	128	307	10.8	19
8	159	249	13.2	25
9	89	245	13.8	30

Dataset Description- This report analyzes diurnal air quality measurements from New York across three sites—Roosevelt Island, Central Park, and LaGuardia Airport—recorded between May and September 1973 (153 observations). Variables include mean ozone levels (ppb), solar radiation (Langleys), average wind speed (mph), and maximum daily temperature (°F).

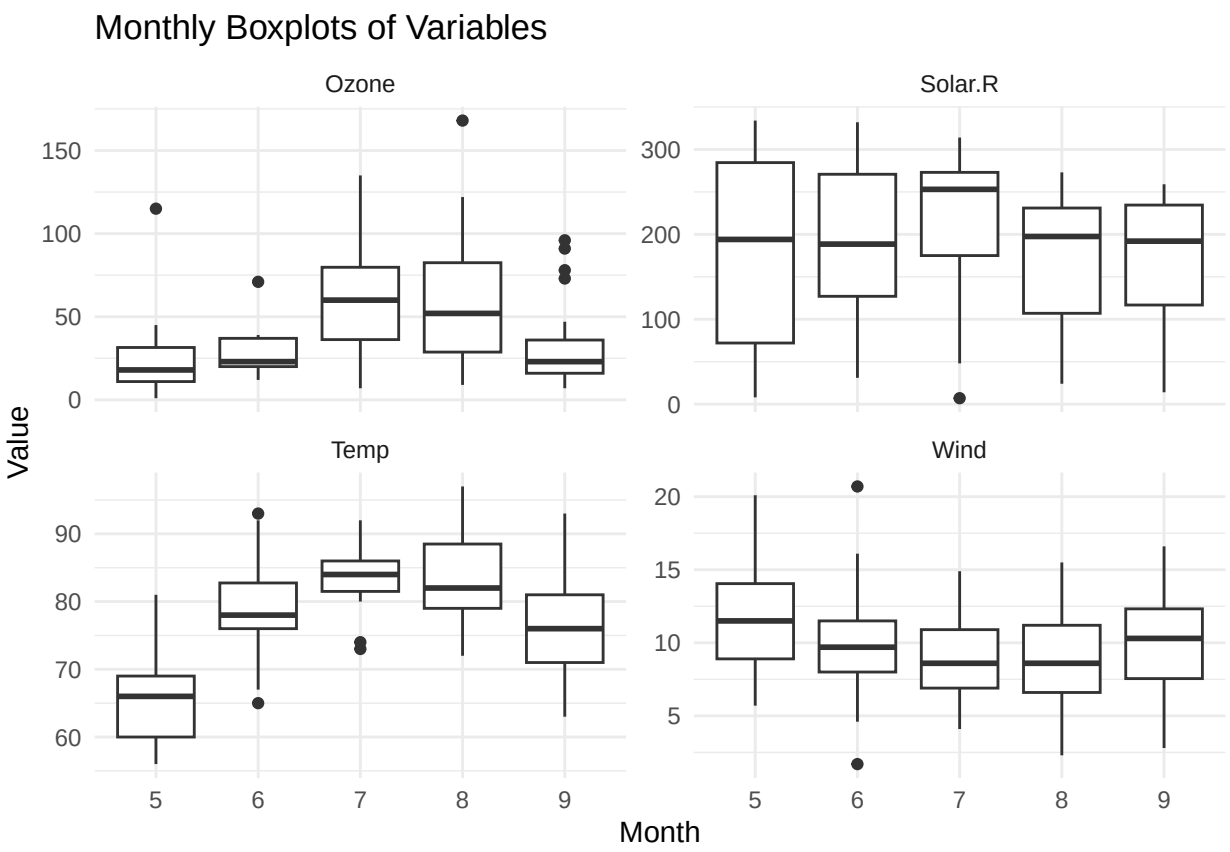
Data Distribution and Outliers- To understand variable distribution and detect outliers, monthly boxplots (visualisation 1) were created. Ozone showed right skewness with high-value outliers, except in July. Solar radiation displayed left skewness from July to September, with no outliers in August and September—indicating seasonal daylight decline. Median patterns in ozone and solar radiation, and later temperature, appeared aligned. Wind speeds had few outliers and showed a slight rise in September, consistent with early autumn. Summary statistics aligned well with these visual trends.

Following the distribution analysis, correlation analysis revealed strong positive correlation (visualisation 2) between ozone and temperature ($r = 0.69$), a moderate positive correlation with solar radiation ($r = 0.34$), and a moderate negative correlation with wind ($r = -0.60$). These findings guided further analysis.

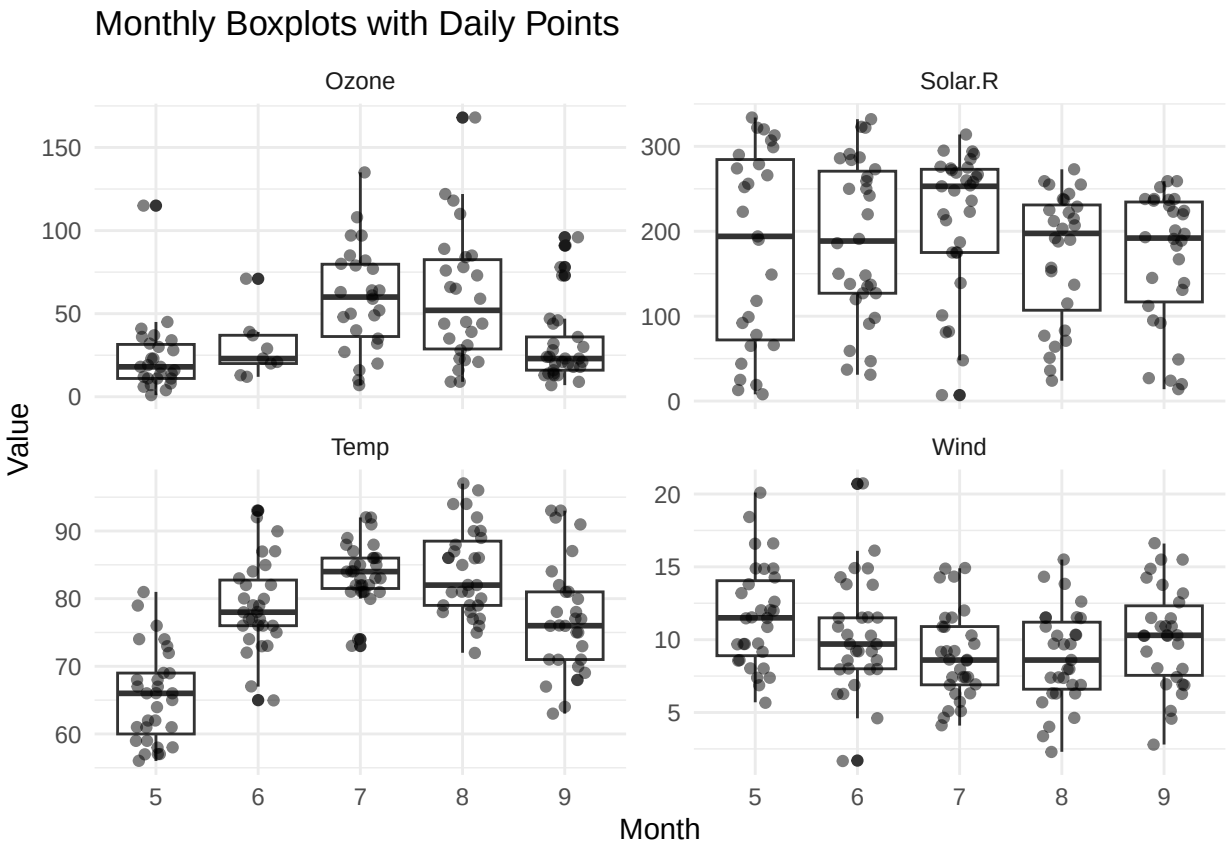
Regression Analysis- A regression model assessed the influence of solar radiation, wind, and temperature on ozone levels. Results indicated that a 1°F rise in temperature increases ozone by 1.66 ppb, and a unit rise in solar radiation by 0.06 ppb. These effects, though modest, were statistically significant ($p < 0.05$). Multicollinearity tests confirmed low inter-correlation among predictors, supporting model reliability. Solar radiation had minimal ozone-depleting influence, aligning with the historical context of CFC-related impacts.

Scatterplots (visualisation 3) supported earlier findings: ozone and temperature exhibited a positive linear trend; ozone and solar radiation showed a weak-to-moderate relationship—disturbed by outliers in August; ozone and wind lacked a clear pattern. These visuals validated the observed regression outcomes. Temporal analysis (visualisation 4) revealed irregular ozone fluctuations, peaking mid-to-late August—consistent with summer conditions. The relationship with solar radiation weakened over time, generally declining

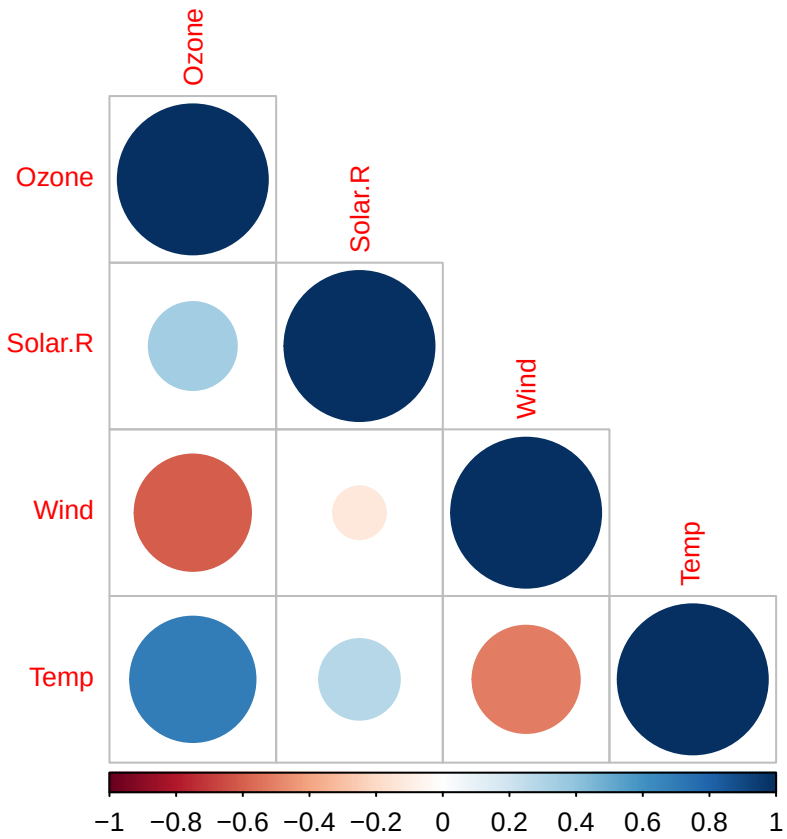
```
## (`stat_boxplot()`).
```




```
## Warning: Removed 44 rows containing non-finite outside the scale range
## (`stat_boxplot()`).

## Warning: Removed 44 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
## Warning: Removed 44 rows containing non-finite outside the scale range
## (`stat_boxplot()`).

## Warning: Removed 44 rows containing non-finite outside the scale range
## (`stat_boxplot()`).

## Warning: Removed 44 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
## function (corr, method = c("circle", "square", "ellipse", "number",
##     "shade", "color", "pie"), type = c("full", "lower", "upper"),
##     col = NULL, col.lim = NULL, is.corr = TRUE, bg = "white",
##     title = "", add = FALSE, diag = TRUE, outline = FALSE, mar = c(0,
##         0, 0, 0), addgrid.col = NULL, addCoef.col = NULL, addCoefasPercent = FALSE,
##     order = c("original", "AOE", "FPC", "hclust", "alphabet"),
##     hclust.method = c("complete", "ward", "ward.D", "ward.D2",
##         "single", "average", "mcquitty", "median", "centroid"),
##     addrect = NULL, rect.col = "black", rect.lwd = 2, tl.pos = NULL,
##     tl.cex = 1, tl.col = "red", tl.offset = 0.4, tl.srt = 90,
##     cl.pos = NULL, cl.length = NULL, cl.cex = 0.8, cl.ratio = 0.15,
##     cl.align.text = "c", cl.offset = 0.5, number.cex = 1, number.font = 2,
##     number.digits = NULL, addshade = c("negative", "positive",
##         "all"), shade.lwd = 1, shade.col = "white", transKeepSign = TRUE,
##     p.mat = NULL, sig.level = 0.05, insig = c("pch", "p-value",
##         "blank", "n", "label_sig"), pch = 4, pch.col = "black",
##     pch.cex = 3, plotCI = c("n", "square", "circle", "rect"),
##     lowCI.mat = NULL, uppCI.mat = NULL, na.label = "?", na.label.col = "black",
##     win.asp = 1, ...)
## {
##     method = match.arg(method)
##     type = match.arg(type)
##     order = match.arg(order)
##     hclust.method = match.arg(hclust.method)
##     addshade = match.arg(addshade)
##     insig = match.arg(insig)
```

```
##     plotCI = match.arg(plotCI)
##     if (win.asp != 1 && !(method %in% c("circle", "square"))) {
##         stop("Parameter 'win.asp' is supported only for circle and square methods.")
##     }
##     asp_rescale_factor = min(1, win.asp)/max(1, win.asp)
##     stopifnot(asp_rescale_factor >= 0 && asp_rescale_factor <=
##         1)
##     if (!is.matrix(corr) && !is.data.frame(corr)) {
##         stop("Need a matrix or data frame!")
##     }
##     if (is.null(addgrid.col)) {
##         addgrid.col = switch(method, color = NA, shade = NA,
##             "grey")
##     }
##     if (!is.corr & !transKeepSign & method %in% c("circle", "square",
##         "ellipse", "shade", "pie")) {
##         stop("method should not be in c('circle', 'square', 'ellipse', 'shade', 'pie') when transKeep
##     }
##     if (any(corr[!is.na(corr)] < col.lim[1]) || any(corr[!is.na(corr)] >
##         col.lim[2])) {
##         stop("color limits should cover matrix")
##     }
##     if (is.null(col.lim)) {
##         if (is.corr) {
##             col.lim = c(-1, 1)
##         }
##         else {
##             if (!diag) {
##                 diag(corr) = NA
##             }
##             col.lim = c(min(corr, na.rm = TRUE), max(corr, na.rm = TRUE))
##         }
##     }
##     SpecialCorr = 0
##     if (is.corr) {
##         if (min(corr, na.rm = TRUE) < -1 - .Machine$double.eps^0.75 ||
##             max(corr, na.rm = TRUE) > 1 + .Machine$double.eps^0.75) {
##             stop("The matrix is not in [-1, 1]!")
##         }
##         SpecialCorr = 1
##         if (col.lim[1] < -1 | col.lim[2] > 1) {
##             stop("col.lim should be within the interval [-1, 1]")
##         }
##     }
##     intercept = 0
##     zoom = 1
##     if (!is.corr) {
##         c_max = max(corr, na.rm = TRUE)
##         c_min = min(corr, na.rm = TRUE)
##         if ((col.lim[1] > c_min) | (col.lim[2] < c_max)) {
##             stop("Wrong color: matrix should be in col.lim interval!")
##         }
##         if (diff(col.lim)/(c_max - c_min) > 2) {
##             warning("col.lim interval too wide, please set a suitable value")
```

```
##         }
##         if (c_max <= 0 | c_min >= 0 | !transKeepSign) {
##             intercept = -col.lim[1]
##             zoom = 1/(diff(col.lim))
##         }
##         else {
##             stopifnot(c_max * c_min < 0)
##             stopifnot(c_min < 0 && c_max > 0)
##             intercept = 0
##             zoom = 1/max(abs(col.lim))
##             SpecialCorr = 1
##         }
##         corr = (intercept + corr) * zoom
##     }
##     col.lim2 = (intercept + col.lim) * zoom
##     int = intercept * zoom
##     if (is.null(col) & is.corr) {
##         col = COL2("RdBu", 200)
##     }
##     if (is.null(col) & !is.corr) {
##         if (col.lim[1] * col.lim[2] < 0) {
##             col = COL2("RdBu", 200)
##         }
##         else {
##             col = COL1("YlOrBr", 200)
##         }
##     }
##     n = nrow(corr)
##     m = ncol(corr)
##     min.nm = min(n, m)
##     ord = 1:min.nm
##     if (order != "original") {
##         ord = corrMatOrder(corr, order = order, hclust.method = hclust.method)
##         corr = corr[ord, ord]
##         if (!is.null(p.mat)) {
##             p.mat = p.mat[ord, ord]
##         }
##     }
##     if (is.null(rownames(corr))) {
##         rownames(corr) = 1:n
##     }
##     if (is.null(colnames(corr))) {
##         colnames(corr) = 1:m
##     }
##     apply_mat_filter = function(mat) {
##         x = matrix(1:n * m, nrow = n, ncol = m)
##         switch(type, upper = mat[row(x) > col(x)] <- Inf, lower = mat[row(x) <
##             col(x)] <- Inf)
##         if (!diag) {
##             diag(mat) = Inf
##         }
##         return(mat)
##     }
##     getPos.Dat = function(mat) {
```

```
##         tmp = apply_mat_filter(mat)
##         Dat = tmp[is.finite(tmp)]
##         ind = which(is.finite(tmp), arr.ind = TRUE)
##         Pos = ind
##         Pos[, 1] = ind[, 2]
##         Pos[, 2] = -ind[, 1] + 1 + n
##         PosName = ind
##         PosName[, 1] = colnames(mat)[ind[, 2]]
##         PosName[, 2] = rownames(mat)[ind[, 1]]
##         return(list(Pos, Dat, PosName))
##     }
##     getPos.NAs = function(mat) {
##         tmp = apply_mat_filter(mat)
##         ind = which(is.na(tmp), arr.ind = TRUE)
##         Pos = ind
##         Pos[, 1] = ind[, 2]
##         Pos[, 2] = -ind[, 1] + 1 + n
##         return(Pos)
##     }
##     testTemp = getPos.Dat(corr)
##     Pos = getPos.Dat(corr)[[1]]
##     PosName = getPos.Dat(corr)[[3]]
##     if (any(is.na(corr)) && is.character(na.label)) {
##         PosNA = getPos.NAs(corr)
##     }
##     else {
##         PosNA = NULL
##     }
##     AllCoords = rbind(Pos, PosNA)
##     n2 = max(AllCoords[, 2])
##     n1 = min(AllCoords[, 2])
##     nn = n2 - n1
##     m2 = max(AllCoords[, 1])
##     m1 = min(AllCoords[, 1])
##     mm = max(1, m2 - m1)
##     expand_expression = function(s) {
##         ifelse(grepl("^[:=$]", s), parse(text = substring(s,
##             2)), s)
##     }
##     newrownames = sapply(rownames(corr)[(n + 1 - n2):(n + 1 -
##         n1)], expand_expression)
##     newcolnames = sapply(colnames(corr)[m1:m2], expand_expression)
##     DAT = getPos.Dat(corr)[[2]]
##     len.DAT = length(DAT)
##     rm(expand_expression)
##     assign.color = function(dat = DAT, color = col, isSpecialCorr = SpecialCorr) {
##         if (isSpecialCorr) {
##             newcorr = (dat + 1)/2
##         }
##         else {
##             newcorr = dat
##         }
##         newcorr[newcorr <= 0] = 0
##         newcorr[newcorr >= 1] = 1 - 1e-16
```

```
##         color[floor(newcorr * length(color)) + 1]
##     }
##     col.fill = assign.color()
##     isFALSE = function(x) identical(x, FALSE)
##     isTRUE = function(x) identical(x, TRUE)
##     if (isFALSE(tl.pos)) {
##         tl.pos = "n"
##     }
##     if (is.null(tl.pos) || isTRUE(tl.pos)) {
##         tl.pos = switch(type, full = "lt", lower = "ld", upper = "td")
##     }
##     if (isFALSE(cl.pos)) {
##         cl.pos = "n"
##     }
##     if (is.null(cl.pos) || isTRUE(cl.pos)) {
##         cl.pos = switch(type, full = "r", lower = "b", upper = "r")
##     }
##     if (isFALSE(outline)) {
##         col.border = col.fill
##     }
##     else if (isTRUE(outline)) {
##         col.border = "black"
##     }
##     else if (is.character(outline)) {
##         col.border = outline
##     }
##     else {
##         stop("Unsupported value type for parameter outline")
##     }
##     oldpar = par(mar = mar, bg = par()$bg)
##     on.exit(par(oldpar), add = TRUE)
##     if (!add) {
##         plot.new()
##         xlabwidth = max(strwidth(newrownames, cex = tl.cex))
##         ylabwidth = max(strwidth(newcolnames, cex = tl.cex))
##         laboffset = strwidth("W", cex = tl.cex) * tl.offset
##         for (i in 1:50) {
##             xlim = c(m1 - 0.5 - laboffset - xlabwidth * (grepl("l",
##                 tl.pos) | grepl("d", tl.pos)), m2 + 0.5 + mm *
##                 cl.ratio * (cl.pos == "r") + xlabwidth * abs(cos(tl.srt *
##                 pi/180)) * grepl("d", tl.pos))
##             ylim = c(n1 - 0.5 - nn * cl.ratio * (cl.pos == "b") -
##                 laboffset, n2 + 0.5 + laboffset + ylabwidth *
##                 abs(sin(tl.srt * pi/180)) * grepl("t", tl.pos) +
##                 ylabwidth * abs(sin(tl.srt * pi/180)) * (type ==
##                   "lower") * grepl("d", tl.pos))
##             plot.window(xlim, ylim, asp = 1, xaxs = "i", yaxs = "i")
##             x.tmp = max(strwidth(newrownames, cex = tl.cex))
##             y.tmp = max(strwidth(newcolnames, cex = tl.cex))
##             laboffset.tmp = strwidth("W", cex = tl.cex) * tl.offset
##             if (max(x.tmp - xlabwidth, y.tmp - ylabwidth, laboffset.tmp -
##                 laboffset) < 0.001) {
##                 break
##             }
```

```
##             xlabwidth = x.tmp
##             ylabwidth = y.tmp
##             laboffset = laboffset.tmp
##             if (i == 50) {
##                 warning(c("Not been able to calculate text margin, ",
##                   "please try again with a clean new empty window using ",
##                   "{plot.new(); dev.off()} or reduce tl.cex"))
##             }
##         }
##         if (.Platform$OS.type == "windows") {
##             grDevices::windows.options(width = 7, height = 7 *
##                 diff(ylim)/diff(xlim))
##         }
##         xlim = xlim + diff(xlim) * 0.01 * c(-1, 1)
##         ylim = ylim + diff(ylim) * 0.01 * c(-1, 1)
##         plot.window(xlim = xlim, ylim = ylim, asp = win.asp,
##             xlab = "", ylab = "", xaxs = "i", yaxs = "i")
##     }
##     laboffset = strwidth("W", cex = tl.cex) * tl.offset
##     symbols(Pos, add = TRUE, inches = FALSE, rectangles = matrix(1,
##         len.DAT, 2), bg = bg, fg = bg)
##     if (method == "circle" && plotCI == "n") {
##         symbols(Pos, add = TRUE, inches = FALSE, circles = asp_rescale_factor *
##             0.9 * abs(DAT)^0.5/2, fg = col.border, bg = col.fill)
##     }
##     if (method == "ellipse" && plotCI == "n") {
##         ell.dat = function(rho, length = 99) {
##             k = seq(0, 2 * pi, length = length)
##             x = cos(k + acos(rho)/2)/2
##             y = cos(k - acos(rho)/2)/2
##             cbind(rbind(x, y), c(NA, NA))
##         }
##         ELL.dat = lapply(DAT, ell.dat)
##         ELL.dat2 = 0.85 * matrix(unlist(ELL.dat), ncol = 2, byrow = TRUE)
##         ELL.dat2 = ELL.dat2 + Pos[rep(1:length(DAT), each = 100),
##             ]
##         polygon(ELL.dat2, border = col.border, col = col.fill)
##     }
##     if (is.null(number.digits)) {
##         number.digits = switch(addCoefasPercent + 1, 2, 0)
##     }
##     stopifnot(number.digits%%1 == 0)
##     stopifnot(number.digits >= 0)
##     if (method == "number" && plotCI == "n") {
##         x = (DAT - int) * ifelse(addCoefasPercent, 100, 1)/zoom
##         text(Pos[, 1], Pos[, 2], font = number.font, col = col.fill,
##             labels = format(round(x, number.digits), nsmall = number.digits),
##             cex = number.cex)
##     }
##     NA_LABEL_MAX_CHARS = 2
##     if (is.matrix(PosNA) && nrow(PosNA) > 0) {
##         stopifnot(is.matrix(PosNA))
##         if (na.label == "square") {
##             symbols(PosNA, add = TRUE, inches = FALSE, squares = rep(1,
```

```
##                 nrow(PosNA)), bg = na.label.col, fg = na.label.col)
##         }
##         else if (nchar(na.label) %in% 1:NA_LABEL_MAX_CHARS) {
##             symbols(PosNA, add = TRUE, inches = FALSE, squares = rep(1,
##                 nrow(PosNA)), fg = bg, bg = bg)
##             text(PosNA[, 1], PosNA[, 2], font = number.font,
##                 col = na.label.col, labels = na.label, cex = number.cex,
##                 ...)
##         }
##         else {
##             stop(paste("Maximum number of characters for NA label is:",
##                 NA_LABEL_MAX_CHARS))
##         }
##     }
##     if (method == "pie" && plotCI == "n") {
##         symbols(Pos, add = TRUE, inches = FALSE, circles = rep(0.5,
##             len.DAT) * 0.85, fg = col.border)
##         pie.dat = function(theta, length = 100) {
##             k = seq(pi/2, pi/2 - theta, length = 0.5 * length *
##                 abs(theta)/pi)
##             x = c(0, cos(k)/2, 0)
##             y = c(0, sin(k)/2, 0)
##             cbind(rbind(x, y), c(NA, NA))
##         }
##         PIE.dat = lapply(DAT * 2 * pi, pie.dat)
##         len.pie = unlist(lapply(PIE.dat, length))/2
##         PIE.dat2 = 0.85 * matrix(unlist(PIE.dat), ncol = 2, byrow = TRUE)
##         PIE.dat2 = PIE.dat2 + Pos[rep(1:length(DAT), len.pie),
##             ]
##         polygon(PIE.dat2, border = "black", col = col.fill)
##     }
##     if (method == "shade" && plotCI == "n") {
##         symbols(Pos, add = TRUE, inches = FALSE, squares = rep(1,
##             len.DAT), bg = col.fill, fg = addgrid.col)
##         shade.dat = function(w) {
##             x = w[1]
##             y = w[2]
##             rho = w[3]
##             x1 = x - 0.5
##             x2 = x + 0.5
##             y1 = y - 0.5
##             y2 = y + 0.5
##             dat = NA
##             if ((addshade == "positive" || addshade == "all") &&
##                 rho > 0) {
##                 dat = cbind(c(x1, x1, x), c(y, y1, y1), c(x,
##                   x2, x2), c(y2, y2, y))
##             }
##             if ((addshade == "negative" || addshade == "all") &&
##                 rho < 0) {
##                 dat = cbind(c(x1, x1, x), c(y, y2, y2), c(x,
##                   x2, x2), c(y1, y1, y))
##             }
##             return(t(dat))
```

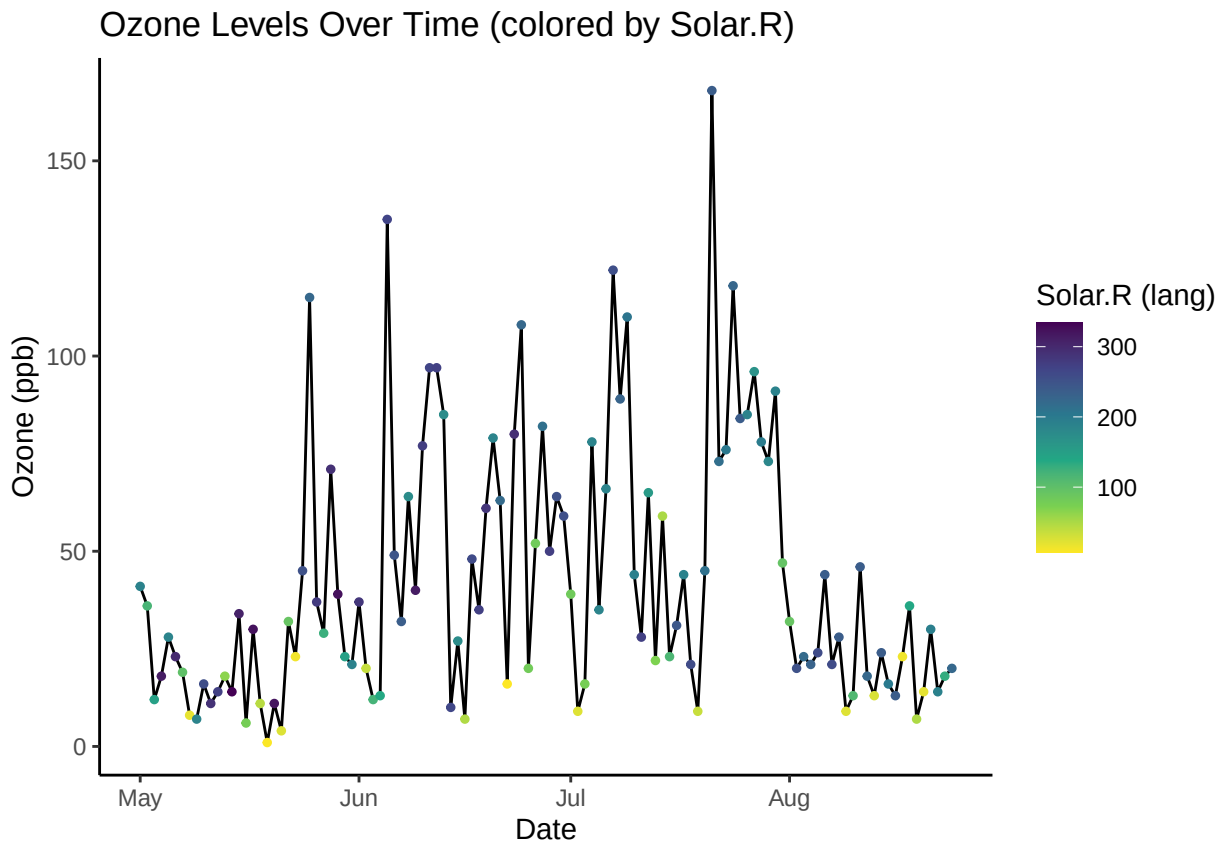
```
##         }
##         pos_corr = rbind(cbind(Pos, DAT))
##         pos_corr2 = split(pos_corr, 1:nrow(pos_corr))
##         SHADE.dat = matrix(na.omit(unlist(lapply(pos_corr2, shade.dat))),
##             byrow = TRUE, ncol = 4)
##         segments(SHADE.dat[, 1], SHADE.dat[, 2], SHADE.dat[,
##             3], SHADE.dat[, 4], col = shade.col, lwd = shade.lwd)
##     }
##     if (method == "square" && plotCI == "n") {
##         draw_method_square(Pos, DAT, asp_rescale_factor, col.border,
##             col.fill)
##     }
##     if (method == "color" && plotCI == "n") {
##         draw_method_color(Pos, col.border, col.fill)
##     }
##     draw_grid(AllCoords, addgrid.col)
##     if (plotCI != "n") {
##         if (is.null(lowCI.mat) || is.null(uppCI.mat)) {
##             stop("Need lowCI.mat and uppCI.mat!")
##         }
##         if (order != "original") {
##             lowCI.mat = lowCI.mat[ord, ord]
##             uppCI.mat = uppCI.mat[ord, ord]
##         }
##         pos.lowNew = getPos.Dat(lowCI.mat)[[1]]
##         lowNew = getPos.Dat(lowCI.mat)[[2]]
##         pos.uppNew = getPos.Dat(uppCI.mat)[[1]]
##         uppNew = getPos.Dat(uppCI.mat)[[2]]
##         k1 = (abs(uppNew) > abs(lowNew))
##         bigabs = uppNew
##         bigabs[which(!k1)] = lowNew[!k1]
##         smallabs = lowNew
##         smallabs[which(!k1)] = uppNew[!k1]
##         sig = sign(uppNew * lowNew)
##         color_bigabs = col[ceiling((bigabs + 1) * length(col)/2)]
##         color_smallabs = col[ceiling((smallabs + 1) * length(col)/2)]
##         if (plotCI == "circle") {
##             symbols(pos.uppNew[, 1], pos.uppNew[, 2], add = TRUE,
##                 inches = FALSE, circles = 0.95 * abs(bigabs)^0.5/2,
##                 bg = ifelse(sig > 0, col.fill, color_bigabs),
##                 fg = ifelse(sig > 0, col.fill, color_bigabs))
##             symbols(pos.lowNew[, 1], pos.lowNew[, 2], add = TRUE,
##                 inches = FALSE, circles = 0.95 * abs(smallabs)^0.5/2,
##                 bg = ifelse(sig > 0, bg, color_smallabs), fg = ifelse(sig >
##                   0, col.fill, color_smallabs))
##         }
##         if (plotCI == "square") {
##             symbols(pos.uppNew[, 1], pos.uppNew[, 2], add = TRUE,
##                 inches = FALSE, squares = abs(bigabs)^0.5, bg = ifelse(sig >
##                   0, col.fill, color_bigabs), fg = ifelse(sig >
##                   0, col.fill, color_bigabs))
##             symbols(pos.lowNew[, 1], pos.lowNew[, 2], add = TRUE,
##                 inches = FALSE, squares = abs(smallabs)^0.5,
##                 bg = ifelse(sig > 0, bg, color_smallabs), fg = ifelse(sig >
```

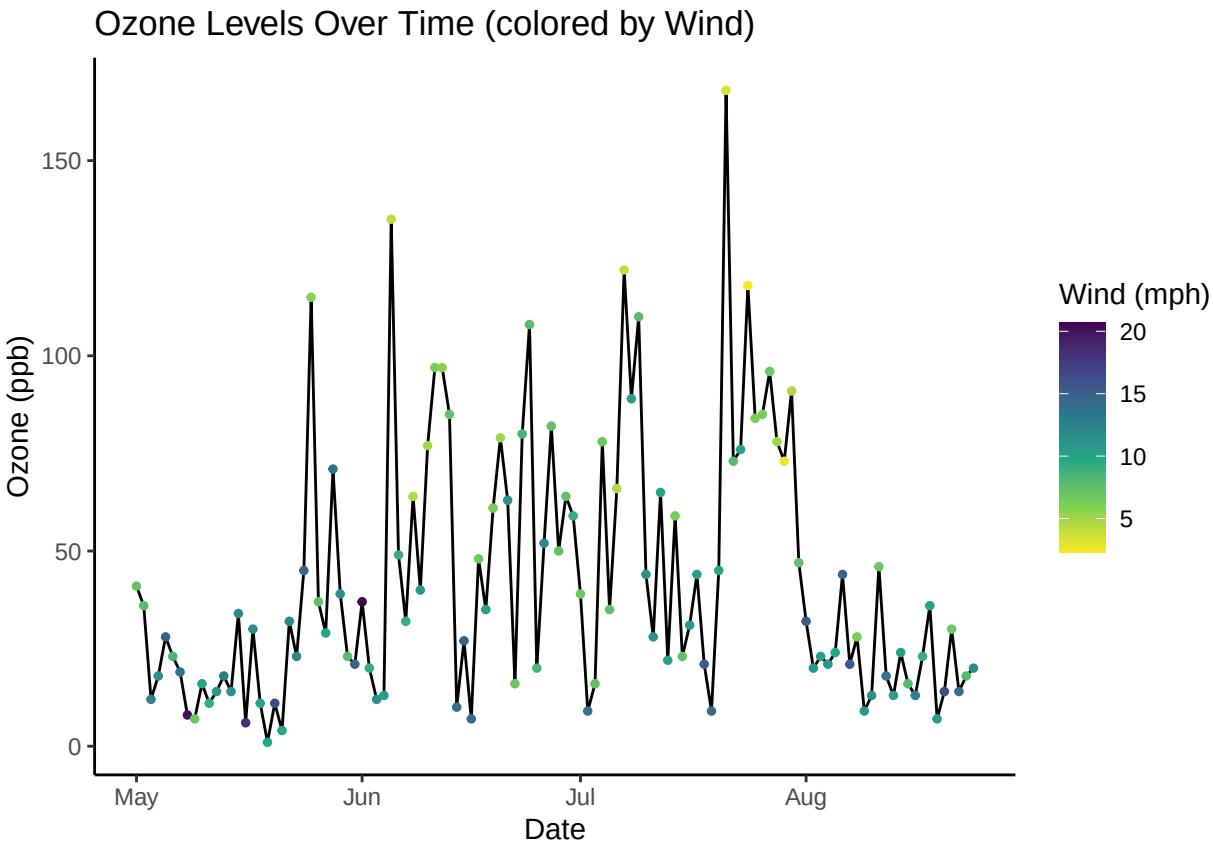
```
##                   0, col.fill, color_smallabs))
##         }
##         if (plotCI == "rect") {
##             rect.width = 0.25
##             rect(pos.uppNew[, 1] - rect.width, pos.uppNew[, 2] +
##                 smallabs/2, pos.uppNew[, 1] + rect.width, pos.uppNew[,
##                 2] + bigabs/2, col = col.fill, border = col.fill)
##             segments(pos.lowNew[, 1] - rect.width, pos.lowNew[,
##                 2] + DAT/2, pos.lowNew[, 1] + rect.width, pos.lowNew[,
##                 2] + DAT/2, col = "black", lwd = 1)
##             segments(pos.uppNew[, 1] - rect.width, pos.uppNew[,
##                 2] + uppNew/2, pos.uppNew[, 1] + rect.width,
##                 pos.uppNew[, 2] + uppNew/2, col = "black", lwd = 1)
##             segments(pos.lowNew[, 1] - rect.width, pos.lowNew[,
##                 2] + lowNew/2, pos.lowNew[, 1] + rect.width,
##                 pos.lowNew[, 2] + lowNew/2, col = "black", lwd = 1)
##             segments(pos.lowNew[, 1] - 0.5, pos.lowNew[, 2],
##                 pos.lowNew[, 1] + 0.5, pos.lowNew[, 2], col = "grey70",
##                 lty = 3)
##         }
##     }
##     if (!is.null(addCoef.col) && method != "number") {
##         text(Pos[, 1], Pos[, 2], col = addCoef.col, labels = format(round((DAT -
##             int) * ifelse(addCoefasPercent, 100, 1)/zoom, number.digits),
##             nsmall = number.digits), cex = number.cex, font = number.font)
##     }
##     if (!is.null(p.mat)) {
##         pos.pNew = getPos.Dat(p.mat)[[1]]
##         pNew = getPos.Dat(p.mat)[[2]]
##     }
##     if (!is.null(p.mat) && insig != "n") {
##         if (!is.null(rownames(p.mat)) | !is.null(rownames(p.mat))) {
##             if (!all(colnames(p.mat) == colnames(corr)) | !all(rownames(p.mat) ==
##                 rownames(corr))) {
##                 warning("p.mat and corr may be not paired, their rownames and colnames are not totall
##             }
##         }
##         if (insig == "label_sig") {
##             if (!is.character(pch))
##                 pch = "*"
##             place_points = function(sig.locs, point) {
##                 text(pos.pNew[, 1][sig.locs], pos.pNew[, 2][sig.locs],
##                   labels = point, col = pch.col, cex = pch.cex,
##                   lwd = 2)
##             }
##             if (length(sig.level) == 1) {
##                 place_points(sig.locs = which(pNew < sig.level),
##                   point = pch)
##             }
##             else {
##                 l = length(sig.level)
##                 for (i in seq_along(sig.level)) {
##                   iter = l + 1 - i
##                   pchTmp = paste(rep(pch, i), collapse = "")
```

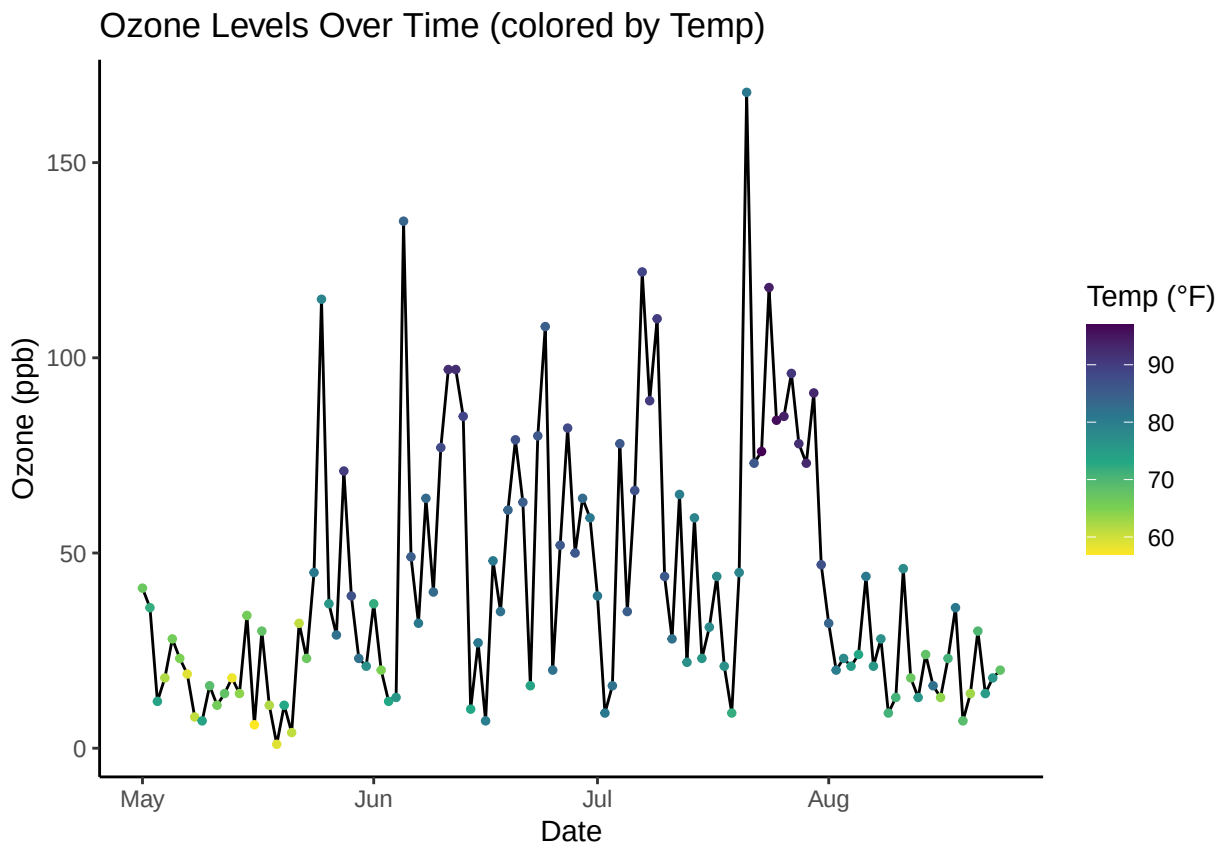
```
##                   if (i == length(sig.level)) {
##                     locs = which(pNew < sig.level[iter])
##                     if (length(locs)) {
##                       place_points(sig.locs = locs, point = pchTmp)
##                     }
##                   }
##                   else {
##                     locs = which(pNew < sig.level[iter] & pNew >
##                       sig.level[iter - 1])
##                     if (length(locs)) {
##                       place_points(sig.locs = locs, point = pchTmp)
##                     }
##                   }
##                 }
##             }
##         }
##         else {
##             ind.p = which(pNew > sig.level)
##             p_inSig = length(ind.p) > 0
##             if (insig == "pch" && p_inSig) {
##                 points(pos.pNew[, 1][ind.p], pos.pNew[, 2][ind.p],
##                   pch = pch, col = pch.col, cex = pch.cex, lwd = 2)
##             }
##             if (insig == "p-value" && p_inSig) {
##                 text(pos.pNew[, 1][ind.p], pos.pNew[, 2][ind.p],
##                   round(pNew[ind.p], number.digits), col = pch.col)
##             }
##             if (insig == "blank" && p_inSig) {
##                 symbols(pos.pNew[, 1][ind.p], pos.pNew[, 2][ind.p],
##                   inches = FALSE, squares = rep(1, length(pos.pNew[,
##                     1][ind.p])), fg = addgrid.col, bg = bg, add = TRUE)
##             }
##         }
##     }
##     if (cl.pos != "n") {
##         colRange = assign.color(dat = col.lim2)
##         ind1 = which(col == colRange[1])
##         ind2 = which(col == colRange[2])
##         colbar = col[ind1:ind2]
##         if (is.null(cl.length)) {
##             cl.length = ifelse(length(colbar) > 20, 11, length(colbar) +
##                 1)
##         }
##         labels = seq(col.lim[1], col.lim[2], length = cl.length)
##         if (cl.pos == "r") {
##             vertical = TRUE
##             xlim = c(m2 + 0.5 + mm * 0.02, m2 + 0.5 + mm * cl.ratio)
##             ylim = c(n1 - 0.5, n2 + 0.5)
##         }
##         if (cl.pos == "b") {
##             vertical = FALSE
##             xlim = c(m1 - 0.5, m2 + 0.5)
##             ylim = c(n1 - 0.5 - nn * cl.ratio, n1 - 0.5 - nn *
##                 0.02)
```

```
##         }
##         colorlegend(colbar = colbar, labels = round(labels, 2),
##             offset = cl.offset, ratio.colbar = 0.3, cex = cl.cex,
##             xlim = xlim, ylim = ylim, vertical = vertical, align = cl.align.text)
##     }
##     if (tl.pos != "n") {
##         pos.xlabel = cbind(m1:m2, n2 + 0.5 + laboffset)
##         pos.ylabel = cbind(m1 - 0.5, n2:n1)
##         if (tl.pos == "td") {
##             if (type != "upper") {
##                 stop("type should be 'upper' if tl.pos is 'dt'.")
##             }
##             pos.ylabel = cbind(m1:(m1 + nn) - 0.5, n2:n1)
##         }
##         if (tl.pos == "ld") {
##             if (type != "lower") {
##                 stop("type should be 'lower' if tl.pos is 'ld'.")
##             }
##             pos.xlabel = cbind(m1:m2, n2:(n2 - mm) + 0.5 + laboffset)
##         }
##         if (tl.pos == "d") {
##             pos.ylabel = cbind(m1:(m1 + nn) - 0.5, n2:n1)
##             pos.ylabel = pos.ylabel[1:min(n, m), ]
##             symbols(pos.ylabel[, 1] + 0.5, pos.ylabel[, 2], add = TRUE,
##                 bg = bg, fg = addgrid.col, inches = FALSE, squares = rep(1,
##                   length(pos.ylabel[, 1])))
##             text(pos.ylabel[, 1] + 0.5, pos.ylabel[, 2], newcolnames[1:min(n,
##                 m)], col = tl.col, cex = tl.cex, ...)
##         }
##         else {
##             if (tl.pos != "l") {
##                 text(pos.xlabel[, 1], pos.xlabel[, 2], newcolnames,
##                   srt = tl.srt, adj = ifelse(tl.srt == 0, c(0.5,
##                     0), c(0, 0)), col = tl.col, cex = tl.cex,
##                   offset = tl.offset, ...)
##             }
##             text(pos.ylabel[, 1], pos.ylabel[, 2], newrownames,
##                 col = tl.col, cex = tl.cex, pos = 2, offset = tl.offset,
##                 ...)
##         }
##     }
##     title(title, ...)
##     if (type == "full" && plotCI == "n" && !is.null(addgrid.col)) {
##         rect(m1 - 0.5, n1 - 0.5, m2 + 0.5, n2 + 0.5, border = addgrid.col)
##     }
##     if (!is.null(addrect) && order == "hclust" && type == "full") {
##         corrRect.hclust(corr, k = addrect, method = hclust.method,
##             col = rect.col, lwd = rect.lwd)
##     }
##     corrPos = data.frame(PosName, Pos, DAT)
##     colnames(corrPos) = c("xName", "yName", "x", "y", "corr")
##     if (!is.null(p.mat)) {
##         corrPos = cbind(corrPos, pNew)
##         colnames(corrPos)[6] = c("p.value")
```

```
##     }
##     corrPos = corrPos[order(corrPos[, 3], -corrPos[, 4]), ]
##     rownames(corrPos) = NULL
##     res = list(corr = corr, corrPos = corrPos, arg = list(type = type))
##     invisible(res)
## }
## <bytecode: 0x0000023e24a22fc8>
## <environment: namespace:corrplot>
```

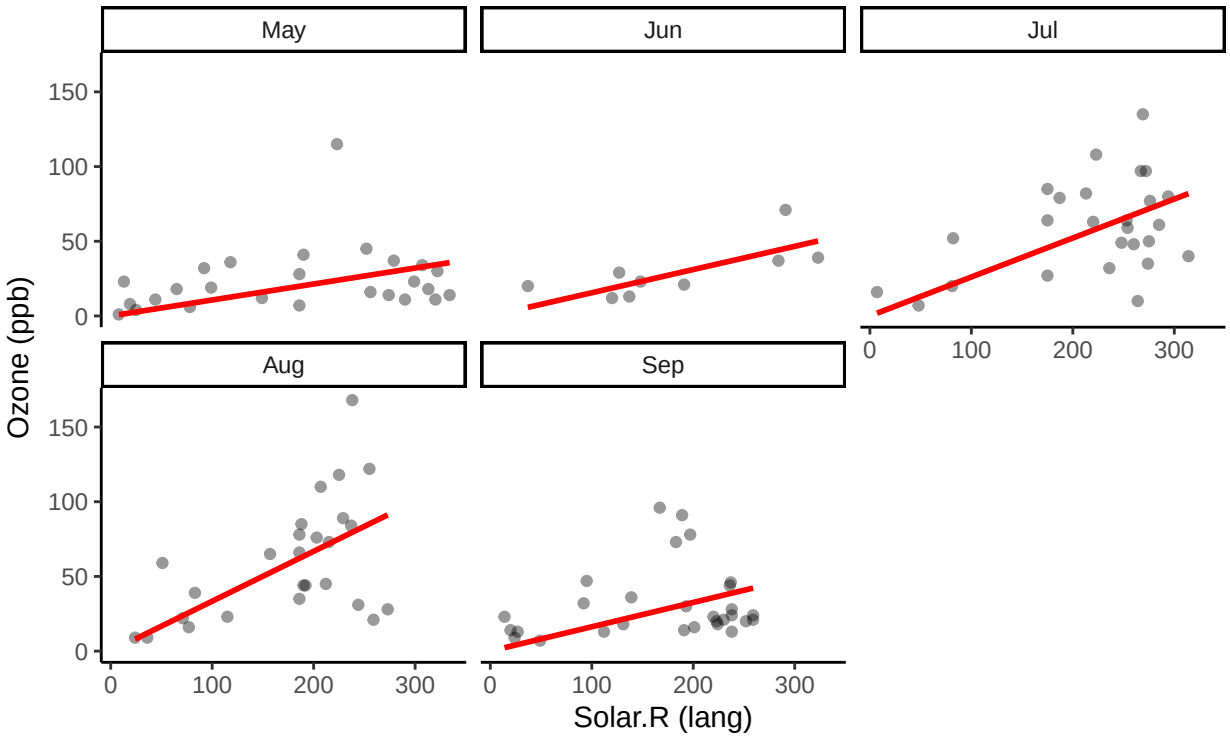



Ozone Levels Over Time (colored by Wind)
Ozone (ppb)
150
100
50
0
May
Jun
Jul
Aug
Date
Wind (mph)
20
15
10
5



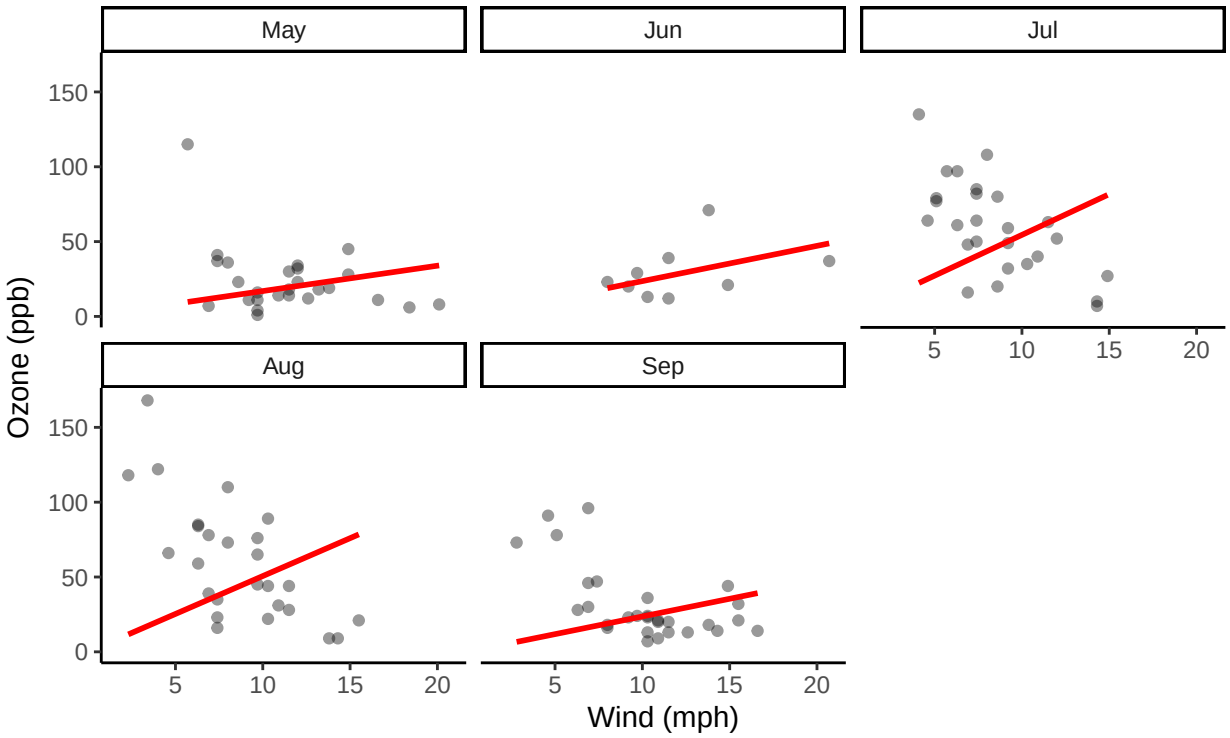

```
## Warning: Removed 37 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 37 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



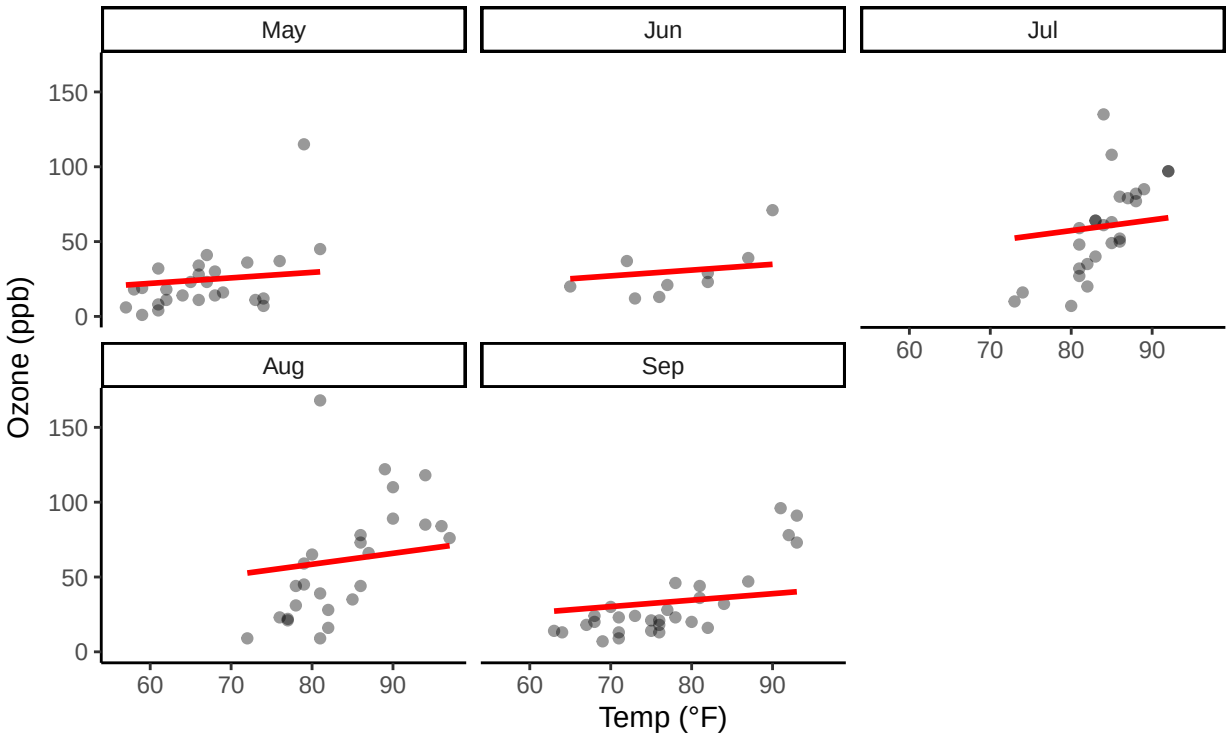
Scatter plot of Ozone vs Solar Radiation for each month from May to September.
Red lines show linear regression fits with zero intercept.

```
## Warning: Removed 37 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 37 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 37 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 37 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Scatter plot of Ozone vs Wind for each month from May to September.
Red lines show linear regression fits with zero intercept.

```
## Warning: Removed 37 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 37 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 37 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 37 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Scatter plot of Ozone vs Temperature for each month from May to September.
Red lines show linear regression fits with zero intercept.

```
## Warning: Removed 37 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 37 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.