EDA for Time Series Photovoltaic (PV) Data

Introduction

This document presents an exploratory data analysis (EDA) of Photovoltaic (PV) output data. Our goal is to understand how PV output changes over time, focusing on seasonality, trends, and variance.

Univariate Analysis

```
## [1] "Missing values in PV output: 0"
```

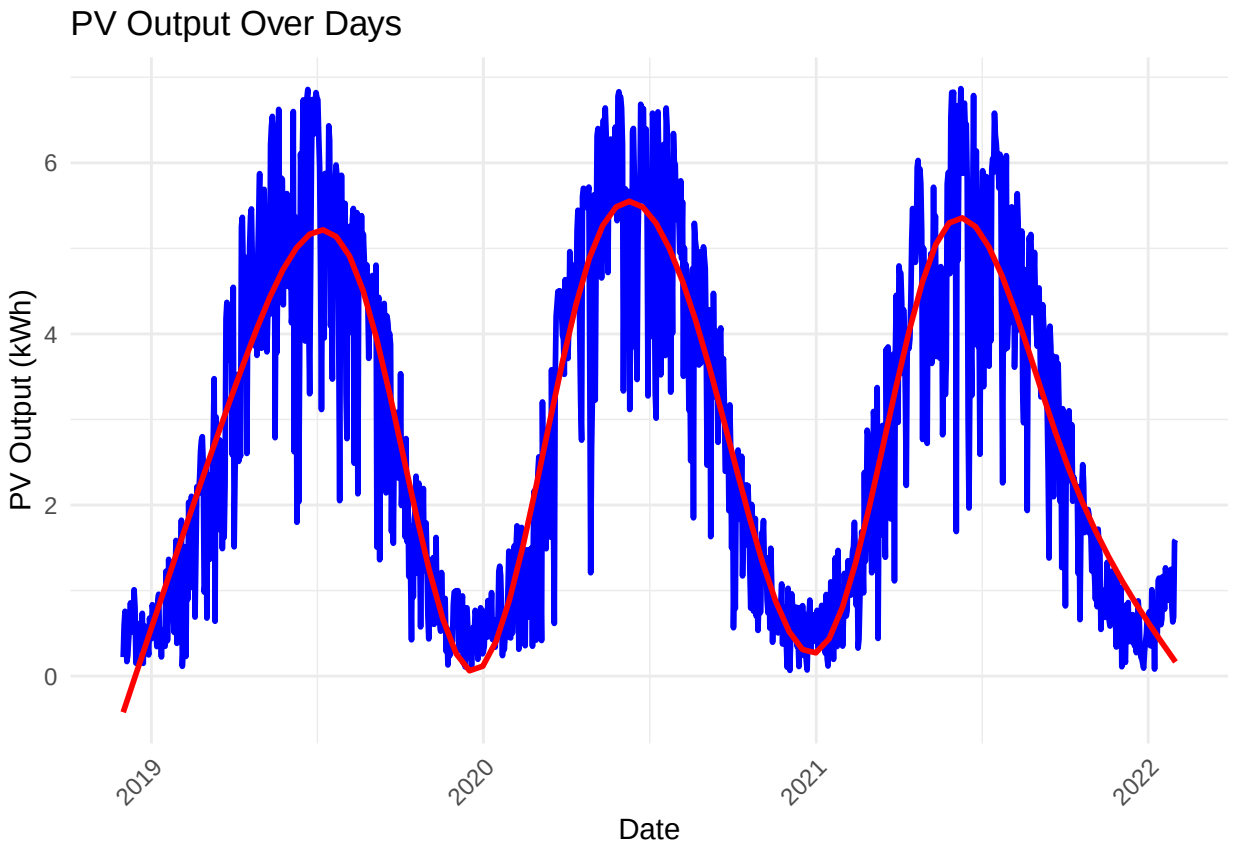
Visualizing PV Output Over Time

We conducted our UV analysis on photovoltaic (PV) output as it directly measures the performance and effectiveness of the current solar panel array. Before working on the expansion, it made sense to udnerstnad current energy production levels.

The following plot of PV output over the last 3 years shows visible seasonality with a regular sinusoidal pattern corresponding to a strong annual cycle. This makes sense due to the consistent seasonal variation in solar exposure due to the tilt of the Earth's axis. The red smoothing line filters out the noise of daily fluctuations to better highlight the seasonal pattern.

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



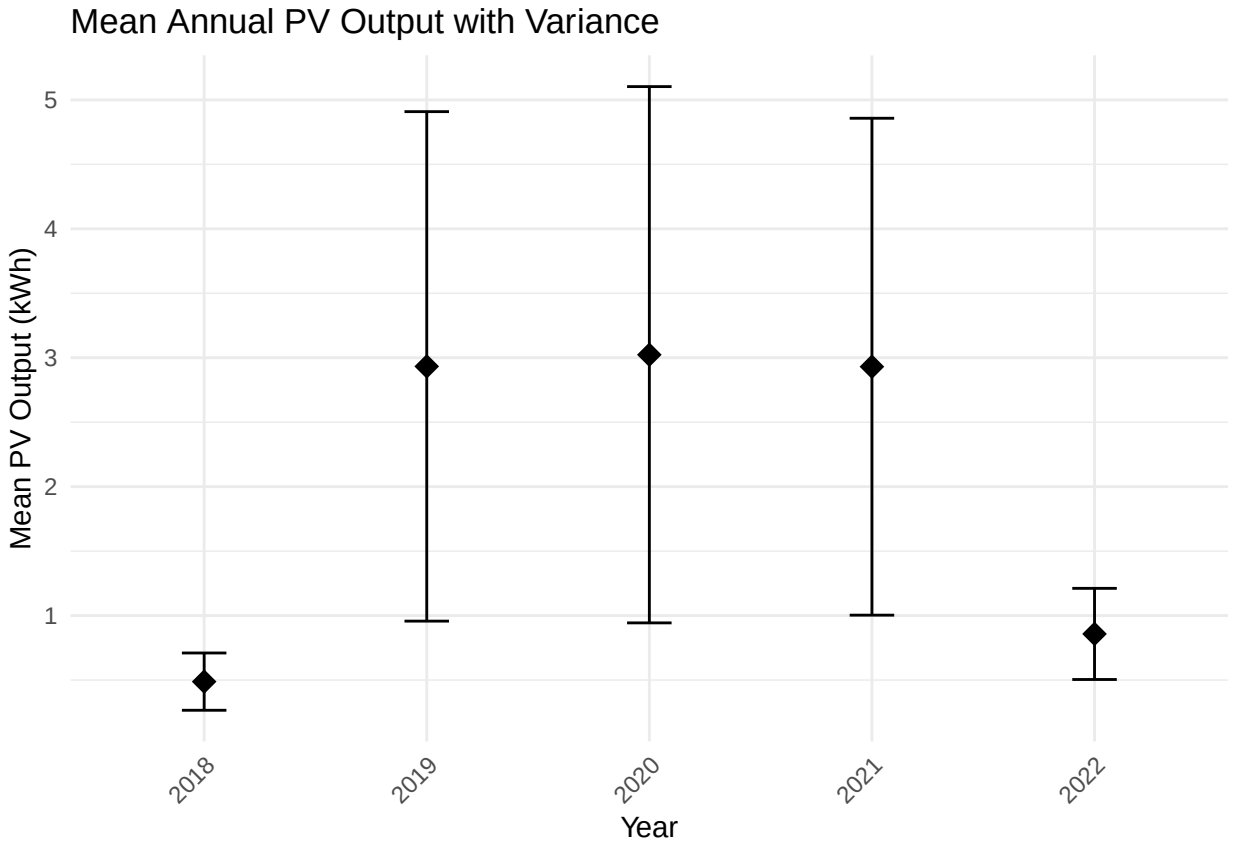
Annual PV Output Statistics

The box plot presents the mean annual photovoltaic (PV) output with corresponding variances for each year from 2018 to 2022. It helps us answer these basic questions: * How much power can we expect to get per year? * How much does PV output vary per year? * How much does the variance change across years?

In 2018, the mean PV output we had the lowest average output of 0.488 kWh with a variance of 0.0493 kWh^2, indicating a relatively stable output with minimal daily fluctuation. The years 2019 to 2021 show substantially higher mean outputs with much larger variances, indicating significant variability in daily output. In 2022, the mean output and variance declined significantly. This box plot underscored the variability in PV output, with particularly high yearly differences in both mean and variance during the period of 2019-2021.

```
## # A tibble: 5 x 3
##    Year  Mean Variance
##   <dbl> <dbl>    <dbl>
## 1  2018 0.488   0.0493
## 2  2019 2.93    3.90
## 3  2020 3.02    4.33
## 4  2021 2.93    3.71
## 5  2022 0.858   0.125
```

```
## [1] "Average variance across all years: 4.03511646687356"
```



Testing for Stationarity

Stationarity is crucial for time-series analysis and we use the Augmented Dickey-Fuller test to check for stationarity which will guide the rest of our EDA.

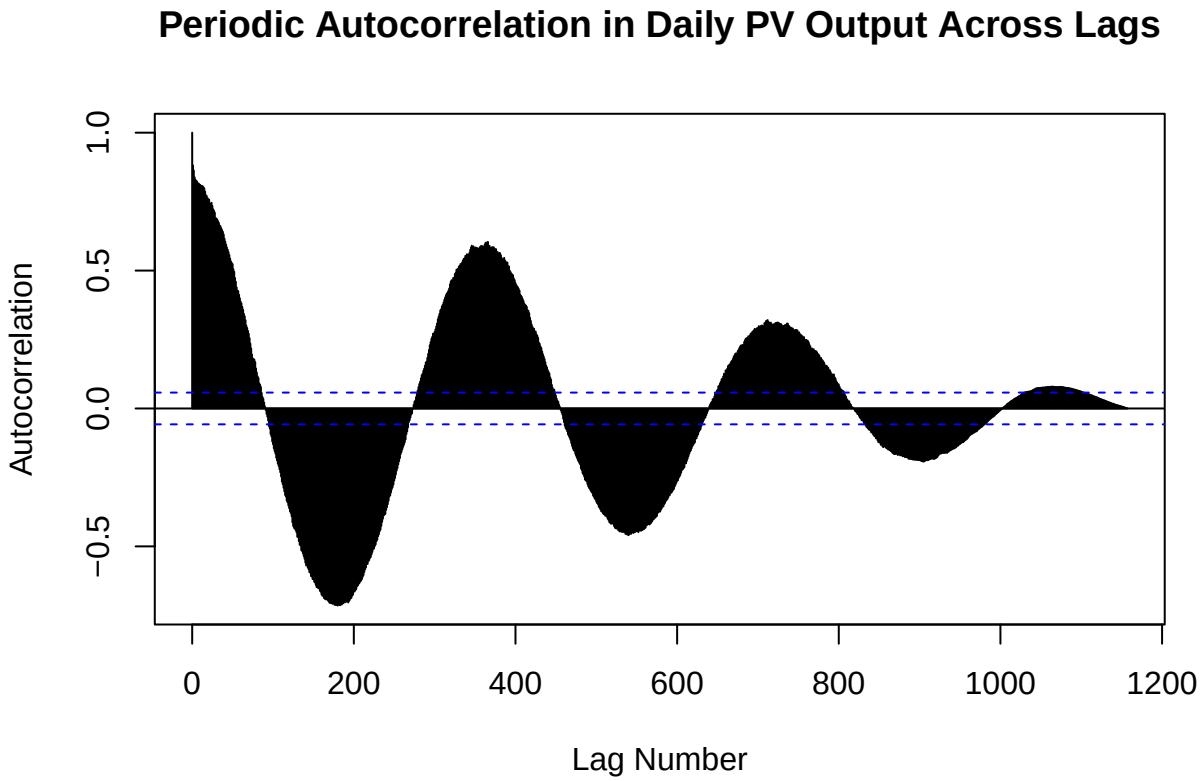
```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$PV.output
## Dickey-Fuller = -2.0806, Lag order = 10, p-value = 0.5442
## alternative hypothesis: stationary
```

Periodic and Partial Autocorrelation

The graph is a Periodic Autocorrelation Function (ACF) plot, which displays the autocorrelation of daily photovoltaic (PV) output in daily lags for 4 years.

The two characteristics that are specifically interesting are the weakening correlation over time and the difference in peak sizes across the years.

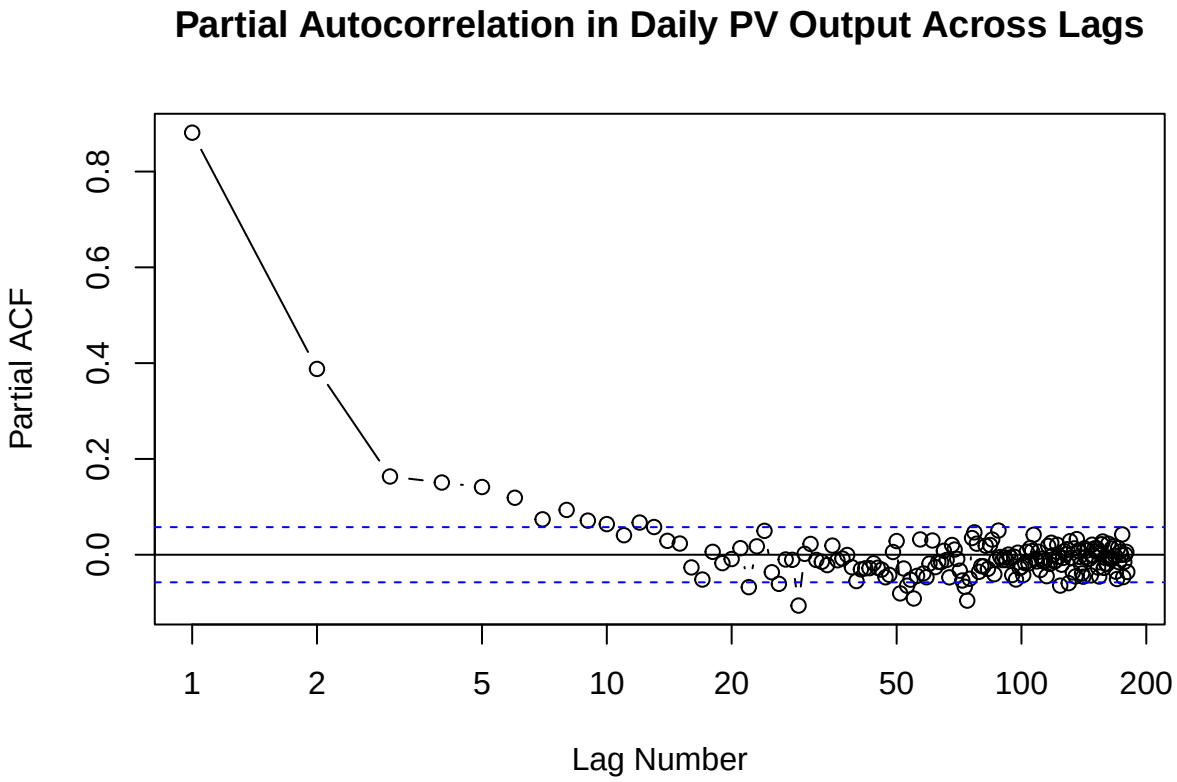
```
acf(data$PV.output, lag.max=1158, main="Periodic Autocorrelation in Daily PV Output Across Lags", xlab=
```



After further analysis into these periodic ACF properties, we came up with the following hypotheses.

- **Weakening of Correlation Over Time**:
  - *Natural Attenuation*: Autocorrelation tends to weaken as the lag increases due to the natural decrease in linear dependencies over time. This is expected even when there is a strong seasonal component present in the data.
  - *Accumulating Variability*: The further the time points are from each other, the more external factors and random noise can accumulate, resulting in reduced similarity between these distant data points.
- **Variation in Peak Sizes**:
  - *Changing Seasonal Intensity*: The solar output's intensity can fluctuate annually due to variable factors such as unusual weather patterns, leading to changes in the strength of the seasonal pattern.
  - *Nonlinear Seasonal Effects*: Seasonal patterns are influenced by the complex, nonlinear interactions within the climate system, which can lead to variations in the size of peaks observed in autocorrelation functions.
  - *Intervention Effects*: Operational changes like maintenance or system upgrades can cause variations in the solar output data, which may be reflected as changes in autocorrelation peaks.
  - *Yearly Anomalies*: Specific events or anomalies, such as volcanic activity or extreme weather conditions, can deviate from the expected seasonal patterns, leading to fluctuations in autocorrelation peaks.

To delve deeper into these patterns, one would segment the solar energy output data by year and potentially by season within each year, performing a spectral analysis on each segment to discern how the frequency and amplitude of the cycles shift over time. Observing consistent wave patterns in the ACF reinforces the presence of a strong annual periodic component in the data. Additionally, examining the variability in peak sizes can shed light on the annual variability of the PV system's output and how it may be influenced by other factors that change over time.



The pattern shown in the Partial Autocorrelation Function (PACF) plot indicates that the PV output of a given day is predominantly influenced by the immediate previous day, and not by earlier days. Specifically, we have an 80% correlation between today and tomorrow's weather which drops to 40% at lag 2, then 20% at lag 3, at 1 week we are at 15%, and then at 2 weeks we are random – which makes sense.

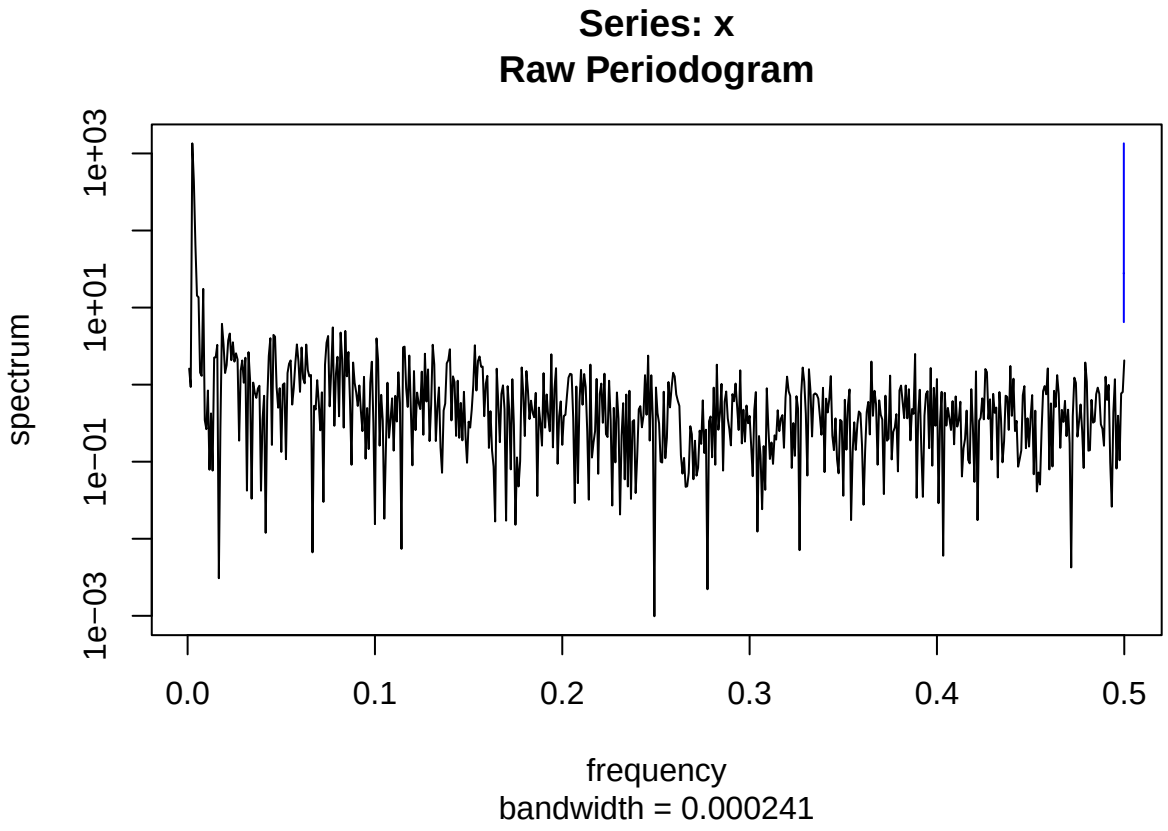
We truncated the the analysis at 180 as there were no meaningful correlation beyond this point, corroborating the short-term predictive power of the time series. We also removed the lag at 0 as each day autocorrelates perfectly with iself. We applied a logarithmic transformation to the lag scale to spread out the lower lag correlations.

The PACF plot emphasizes the non-persistent nature of autocorrelation in daily PV output, which aligns with the stochastic characteristics commonly observed in meteorological time series data influencing solar energy generation.

Spectral Analysis

The raw periodogram has the max spectrum value at f= 0.0025, suggesting that the frequency f=0.0025 has the largest contribution to the data, meaning there is likely a periodicity around 1/f = 400. This estimate supports the fact that we are looking at data with an annual seasonality.

```
# Plot raw periodogram.
spec <- spectrum(data$PV.output)
```




```r
max_index <- which.max(abs(spec$spec))
highest_frequency <- spec$freq[max_index]
print(highest_frequency)
```

```
## [1] 0.0025
```

```r
1/highest_frequency
```

```
## [1] 400
```

Seasonal Decomposition and Outlier Detection

Decomposing the PV output to identify seasonal patterns and detecting outliers based on the residuals.

```r
pv_ts <- ts(data$PV.output, frequency=365) # Adjust frequency based on data's nature
decomposed_data <- stl(pv_ts, s.window="periodic")
residuals <- na.omit(decomposed_data$time.series[, "remainder"])
outliers <- boxplot.stats(residuals)$out
print(paste("Number of outliers detected:", length(outliers)))
```

```
## [1] "Number of outliers detected: 62"
```

Transition to frequency domain

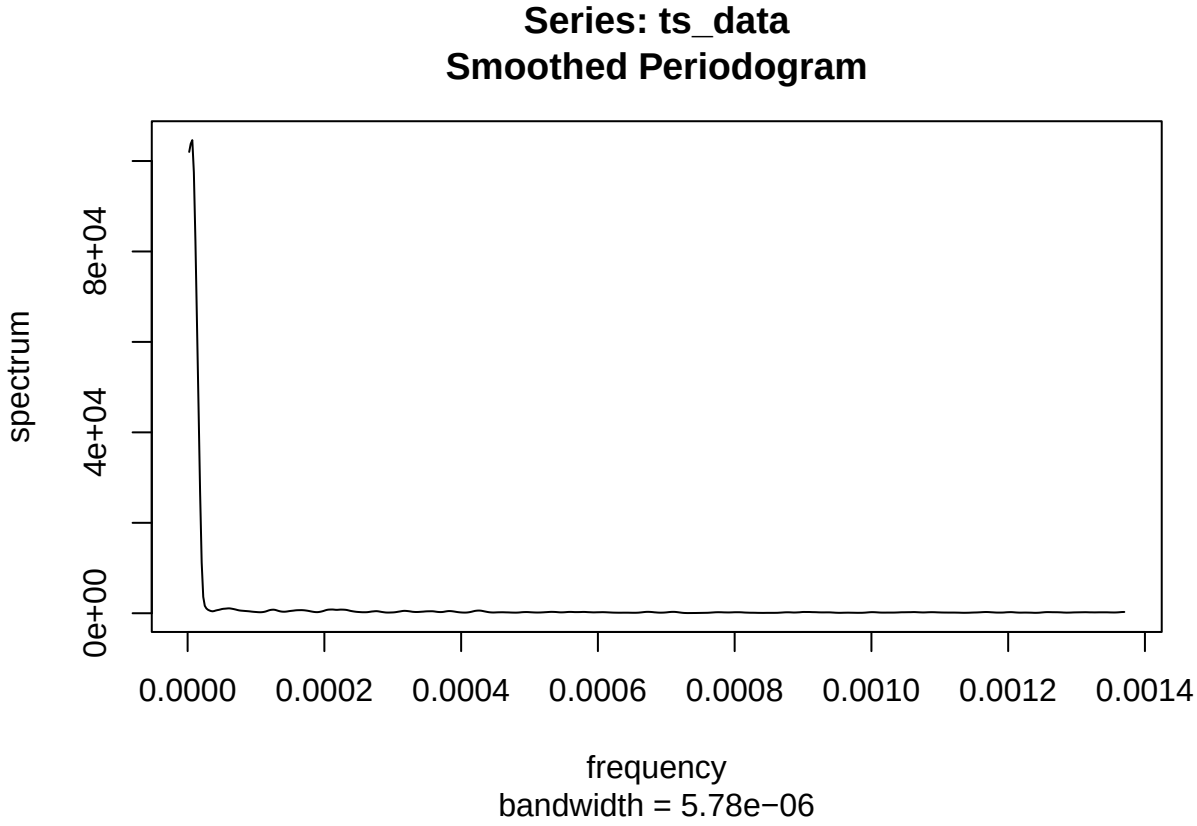
This clear seasonality in the data and it's non-stationary, which we concluded after conducting the Augmented Dickey-Fuller (ADF) test and analysis of the the Autocorrelation Function (ACF), we ceased exploring the data structure in the time domain, and instead looked at it in the frequency domain which we obtained through Fast Fourier Transform. This is possible given that a time series x(t) can be represented as a sum of sinusoids which is mathematically expressed as the following.

Why did we do this? * The decomposition enables the easy identification of seasonal patterns as peaks in the frequency spectrum * It simplifies the application of filters for detrending and noise reduction * Enhances the understanding of the data's non-stationarity by analyzing the power spectral density. * If we had a larger dataset, due to minute by minute sensor data, it would also improve the computational efficiency as the complexity of calculations is reduced from $O(N^2)$ to $O(N \log N)$,

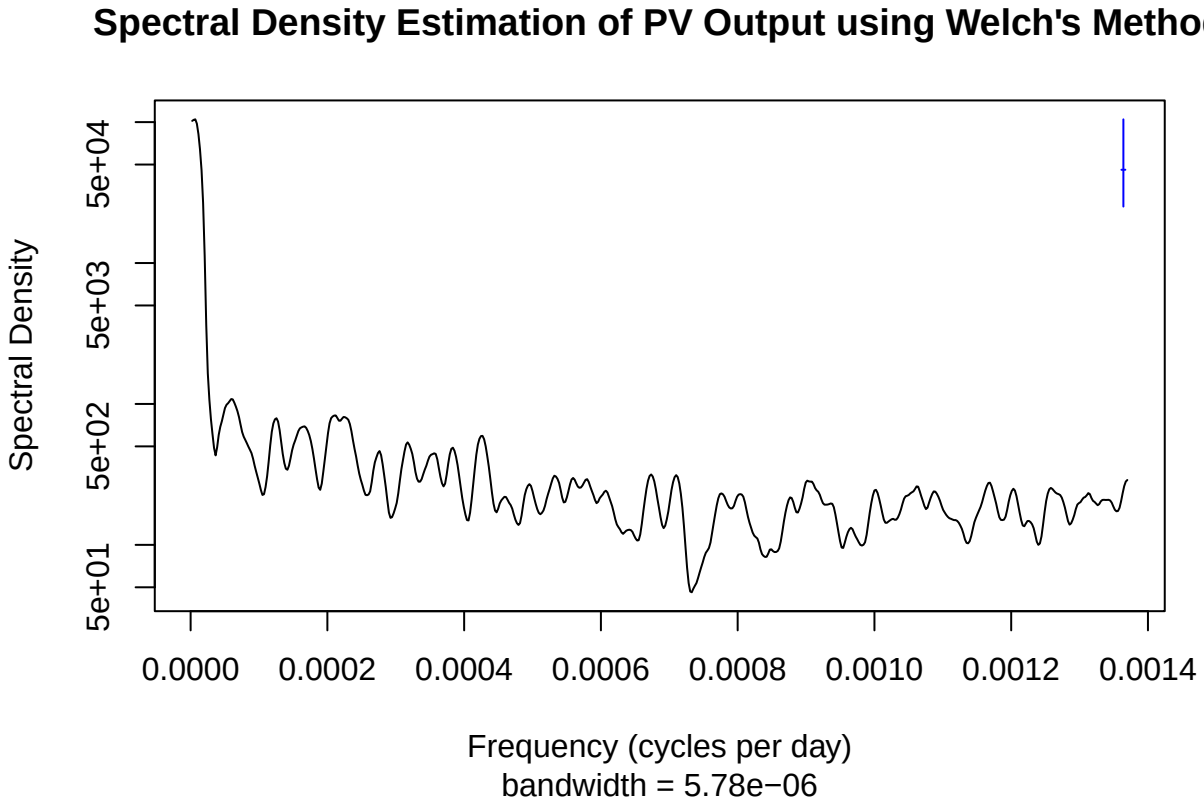
When constructing the graph, Nyquist's theorem played a role. We had a sampling frequency of 1 sample per day. Nyquist's theorem says that for us, the highest frequency that can be accurately sampled without aliasing is 0.5. The Nyquist frequency for a daily sampling rate would allow you to capture periodic components with a period of two days or longer. For an annual cycle, this is more than adequate.

```
ts_data <- ts(data$PV.output, frequency=1/365)
spec_result <- spec.pgram(ts_data, spans = c(7,7), taper = 0.1, overlap = 0.5, log = "no")
```

```
## Warning in plot.window(...): "overlap" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "overlap" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "overlap" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "overlap" is not a
## graphical parameter

## Warning in box(...): "overlap" is not a graphical parameter

## Warning in title(...): "overlap" is not a graphical parameter
```

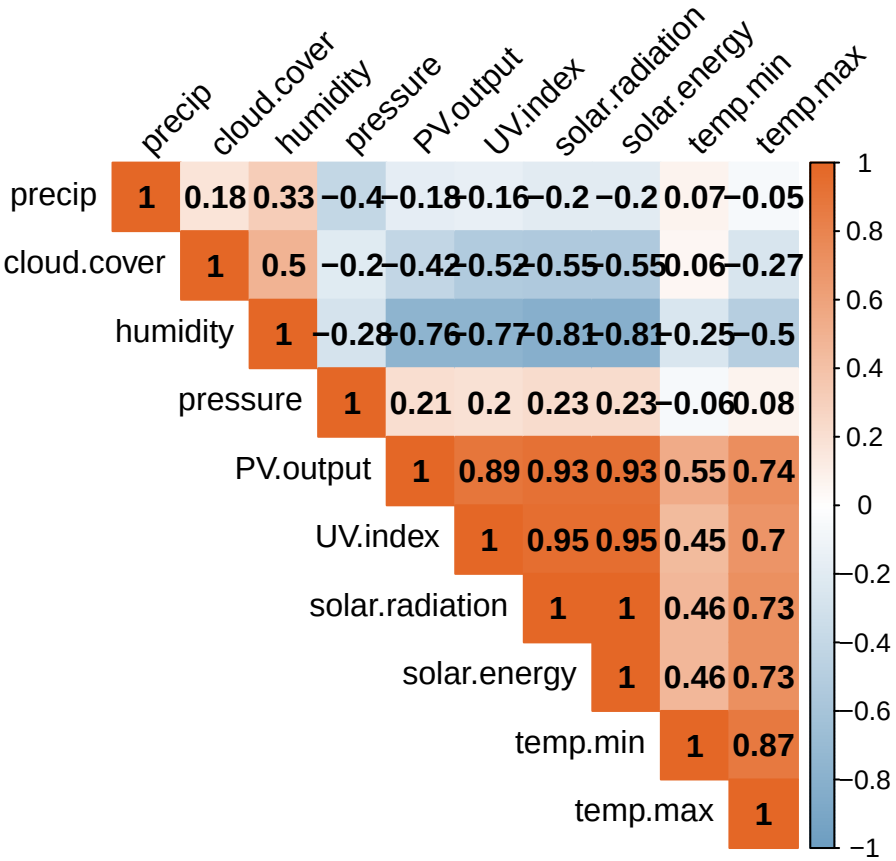



```
plot(spec_result, main="Spectral Density Estimation of PV Output using Welch's Method",
     xlab="Frequency (cycles per day)", ylab="Spectral Density")
```



Multivariate Analysis

```r
# Calculate the correlation matrix
# Read the data
pv_data <- read.csv("IC_SolarPower_38m.csv")
# Calculate the correlation matrix
cor_matrix <- cor(pv_data[,-1], use = "pairwise.complete.obs")
# Create a heatmap
corrplot(cor_matrix, method = 'color', type = 'upper',
         order = 'hclust', addCoef.col = 'black',
         tl.col = 'black', tl.srt = 45,
         col = colorRampPalette(c("#6D9EC1", "white", "#E46726"))(200))
cor_matrix <- cor(pv_data[,-1], use = "pairwise.complete.obs")
# Create a heatmap
corrplot(cor_matrix, method = 'color', type = 'upper',
         order = 'hclust', addCoef.col = 'black',
         tl.col = 'black', tl.srt = 45,
         col = colorRampPalette(c("#6D9EC1", "white", "#E46726"))(200))
```

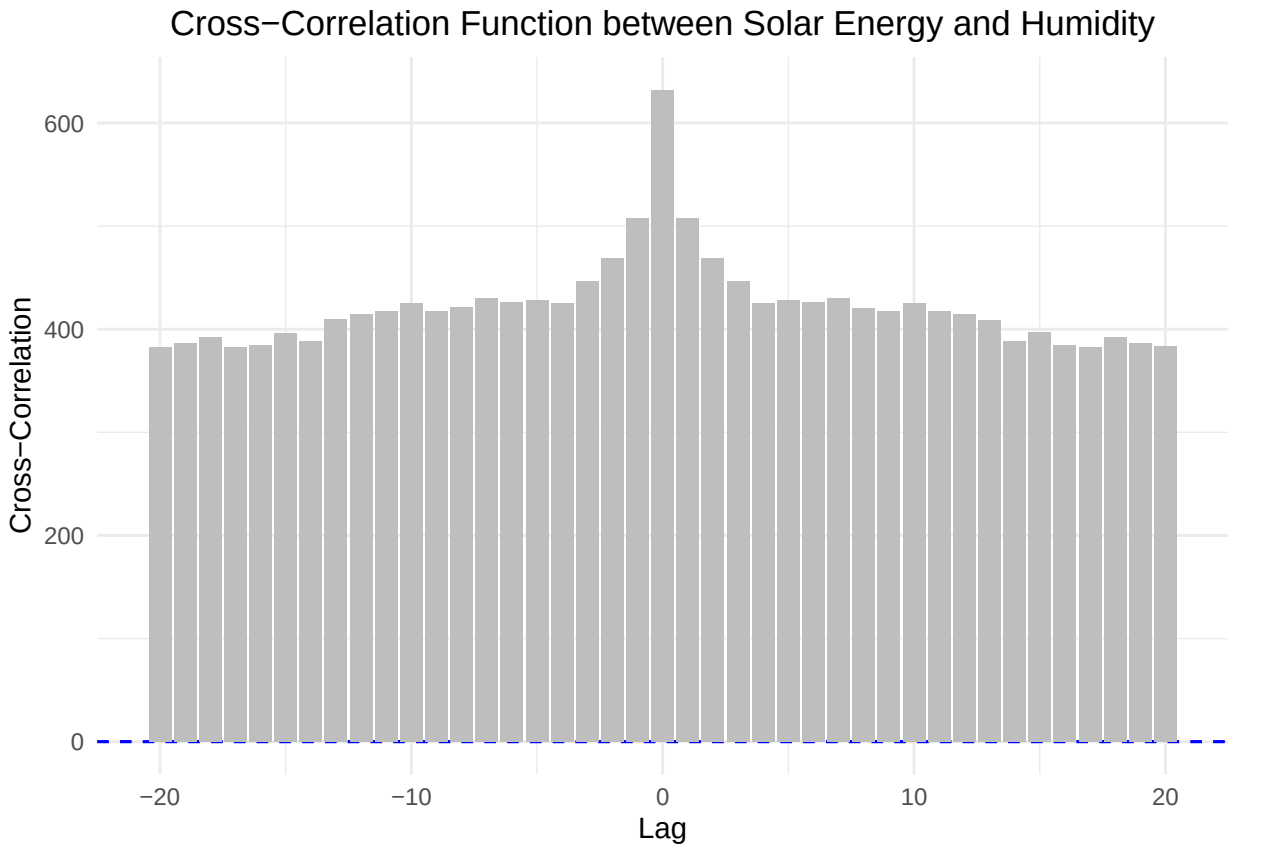


The heatmap illustrates the correlation coefficients between several meteorological variables and photovoltaic (PV) output. It indicates a very strong positive correlation between PV output and both the UV index (r = 0.93) and solar radiation (r = 0.93), implying that as UV intensity and solar radiation increase, so does the energy output of PV systems. Conversely, there's a very strong negative correlation between PV output and humidity (r = -0.77), suggesting that higher humidity levels are associated with lower PV efficiency or energy capture. Additionally, there are substantial positive correlations between PV output and temperature metrics, with temp.min (r = 0.74) and temp.max (r = 0.87), reinforcing the understanding that PV output tends to increase with temperature within the observed range.

Cross Covariance between solar energy and solar radiation

```
sol_e = pv_data$solar.energy
s_rad = pv_data$solar.radiation
ccf_crosscov <- ccf(sol_e, s_rad, lag.max = 20, type = "covariance", plot = FALSE)

# Create a dataframe for plotting
ccf_df <- data.frame(Lag = as.numeric(ccf_crosscov$lag), ACF = as.numeric(ccf_crosscov$acf))

# Calculate confidence intervals (95% by default)
conf_int <- qnorm((1 + 0.95) / 2) / sqrt(length(sol_e))

# Plot the CCF with ggplot2
ggplot(ccf_df, aes(x = Lag, y = ACF)) +
  geom_hline(yintercept = c(-conf_int, conf_int), linetype = "dashed", color = "blue") +
  geom_bar(stat = "identity", fill = "grey") +
  theme_minimal() +
  labs(title = "Cross-Correlation Function between Solar Energy and Humidity",
```

```
       x = "Lag",
       y = "Cross-Correlation") +
  theme(plot.title = element_text(hjust = 0.5))
```



Vector Autoregression on Solar Energy and Humidity

```
pv_data <- read.csv("IC_SolarPower_38m.csv")
solar_energy <- pv_data$solar.energy
humidity <- pv_data$`humidity`

# Install and load the necessary library for ADF test
if (!require(tseries)) {
  install.packages("tseries")
}
library(tseries)

# Install and load the necessary library for VAR analysis
if (!require(vars)) {
  install.packages("vars")
}
library(vars)

# Check for stationarity
adf_test_solar <- adf.test(solar_energy, alternative = "stationary")
adf_test_humidity <- adf.test(humidity, alternative = "stationary")
```

```
## Warning in adf.test(humidity, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
# Print the ADF test results
print(adf_test_solar)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  solar_energy
## Dickey-Fuller = -2.8818, Lag order = 10, p-value = 0.205
## alternative hypothesis: stationary
```

```
print(adf_test_humidity)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  humidity
## Dickey-Fuller = -4.1841, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
# If the Solar Energy series is non-stationary, we difference it
diff_solar_energy <- diff(solar_energy)

# Check stationarity of the differenced Solar Energy series again
adf_test_diff_solar <- adf.test(diff_solar_energy, alternative = "stationary")
```

```
## Warning in adf.test(diff_solar_energy, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
print(adf_test_diff_solar)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_solar_energy
## Dickey-Fuller = -15.993, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Combine the two stationary series into a multivariate time series object
data_var <- cbind(diff_solar_energy, humidity)
```

```
## Warning in cbind(diff_solar_energy, humidity): number of rows of result is not
## a multiple of vector length (arg 1)
```

```
# Determine the appropriate lag order for the VAR model using AIC
lag_selection <- VARselect(data_var, lag.max = 15, type = "both")
selected_lags <- as.integer(lag_selection$selection["AIC"])

if (!is.na(selected_lags)){selected_lags <- 1}

# Fit the VAR model with the selected lag order
var_model <- VAR(data_var, p = 1, type = "both")

# Display the results of the VAR model
summary(var_model)
```

```
##
```

```
## VAR Estimation Results:
## ========================
## Endogenous variables: diff_solar_energy, humidity
## Deterministic variables: both
## Sample size: 1157
## Log Likelihood: -7018.082
## Roots of the characteristic polynomial:
## 0.7805 0.3286
## Call:
## VAR(y = data_var, p = 1, type = "both")
##
##
## Estimation results for equation diff_solar_energy:
## ================================================
## diff_solar_energy = diff_solar_energy.l1 + humidity.l1 + const + trend
##
##                        Estimate Std. Error t value Pr(>|t|)
## diff_solar_energy.l1 -0.3948975  0.0279993 -14.104  < 2e-16 ***
## humidity.l1           0.0818829  0.0119100   6.875 1.01e-11 ***
## const                -5.9148274  0.9026572  -6.553 8.50e-11 ***
## trend                -0.0004545  0.0003809  -1.193    0.233
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 4.28 on 1153 degrees of freedom
## Multiple R-Squared: 0.1544,  Adjusted R-squared: 0.1522
## F-statistic: 70.18 on 3 and 1153 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation humidity:
## =========================================
## humidity = diff_solar_energy.l1 + humidity.l1 + const + trend
##
##                        Estimate Std. Error t value Pr(>|t|)
## diff_solar_energy.l1 -0.9509213  0.0387964 -24.511   <2e-16 ***
## humidity.l1           0.8467022  0.0165027  51.307   <2e-16 ***
## const                11.1740268  1.2507408   8.934   <2e-16 ***
## trend                 0.0006613  0.0005278   1.253     0.21
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 5.93 on 1153 degrees of freedom
## Multiple R-Squared: 0.7123,  Adjusted R-squared: 0.7115
## F-statistic: 951.4 on 3 and 1153 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                   diff_solar_energy humidity
## diff_solar_energy            18.315    1.806
## humidity                      1.806   35.164
##
```

```
## Correlation matrix of residuals:
##                   diff_solar_energy humidity
## diff_solar_energy           1.00000  0.07118
## humidity                    0.07118  1.00000
```

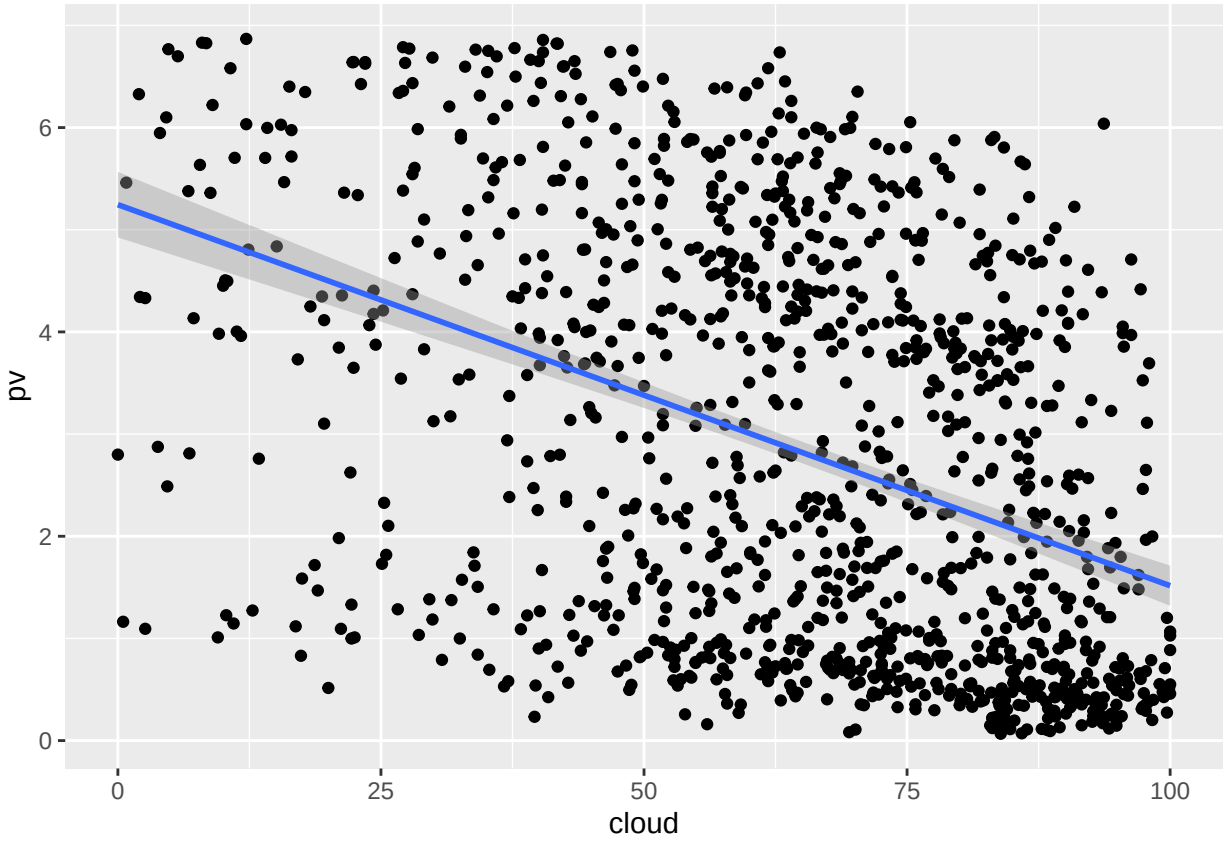
```
# Check the residuals for correlation
residuals_correlation <- cor(resid(var_model))
print(residuals_correlation)
```

```
##                   diff_solar_energy   humidity
## diff_solar_energy        1.00000000 0.07118121
## humidity                 0.07118121 1.00000000
```

Analysis of Photovoltaic Output vs Cloud Cover and UV Index

```
pv <- data$PV.output
cloud <- data$cloud.cover
df <- data.frame(pv, cloud)
# Plot suggests more cloud cover predicts lower PV output.
df %>%
  ggplot(aes(cloud, pv)) + geom_point() +
  geom_smooth(method = "lm") # draw the best-fitted line with OLS
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



While the general trend is negative correlation, we want to uncover any potential Simpson's paradox. Namely, even if it's very cloudy, if the UV index is high, can we still expect a higher PV output?

The plot confirms that the correlation between pv output and cloud is qualitatively different after controlling

for UV index. For example, when UV level is between 1 and 5, more cloud actually predicts more PV output, where as the relationship is negative when the UV level is above 5.

```
# Check for missing values in UV index and handle them if they exist
summary(data$UV.index)
```

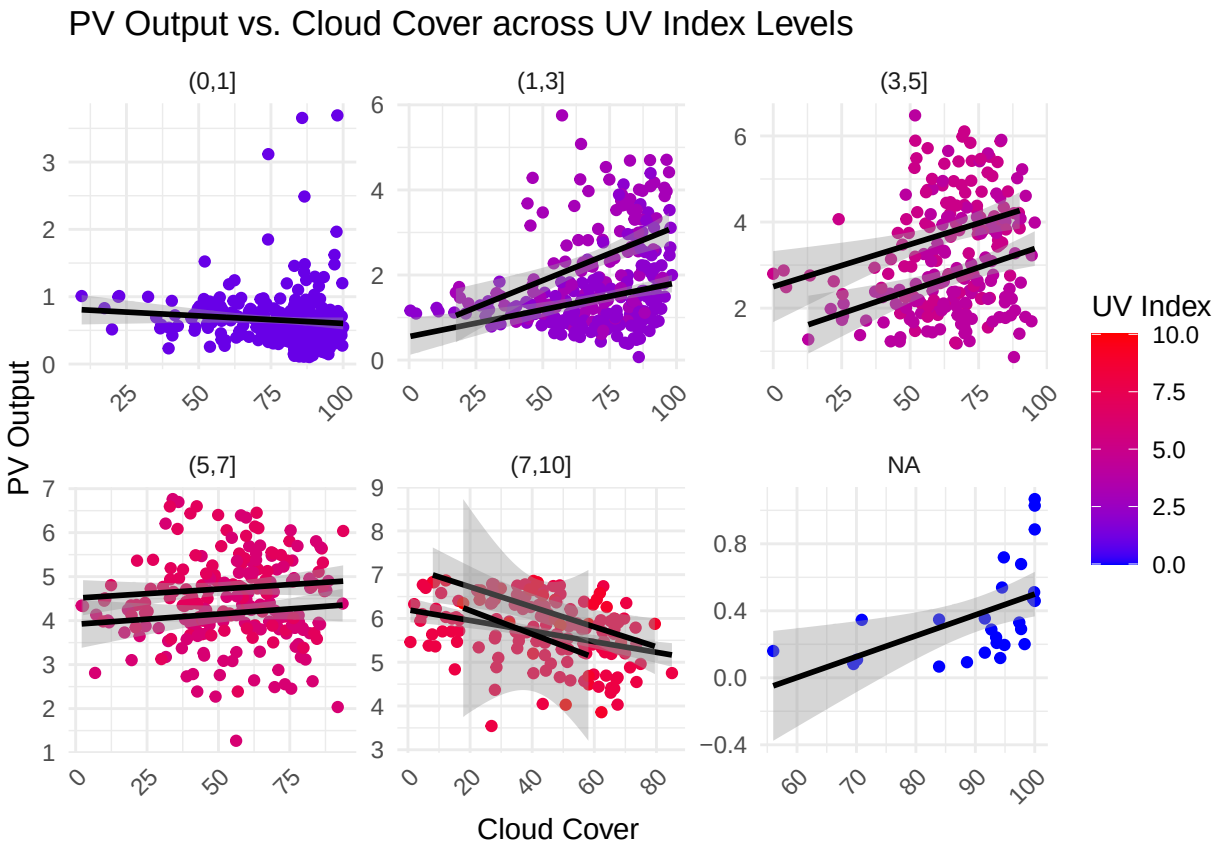
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   4.000   4.018   6.000  10.000
```

```
missing_values <- sum(is.na(data$UV.index))
print(paste("Missing values in UV index:", missing_values))
```

```
## [1] "Missing values in UV index: 0"
```

```
# Handle missing values
if(missing_values > 0){
  data$UV.index[is.na(data$UV.index)] <- mean(data$UV.index, na.rm = TRUE)
  print("Missing UV index values replaced with mean.")
}

# Add UV index to the dataframe for analysis
data$uv <- data$UV.index

# Enhanced plotting with a consistent color scale and facets if the UV index has a wide range
ggplot(data, aes(x = cloud, y = pv)) +
  geom_point(aes(col = uv)) + # Color points by UV index
  scale_colour_gradient(low = "blue", high = "red") + # Use an intuitive color scale
  geom_smooth(aes(group = uv), method = 'lm', color = "black") + # Add regression lines
  facet_wrap(~ cut(uv, breaks = quantile(uv, probs = 0:5 / 5)), scales = "free") + # Facet by UV index
  labs(title = "PV Output vs. Cloud Cover across UV Index Levels",
       x = "Cloud Cover",
       y = "PV Output",
       color = "UV Index") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) # Improve readability
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



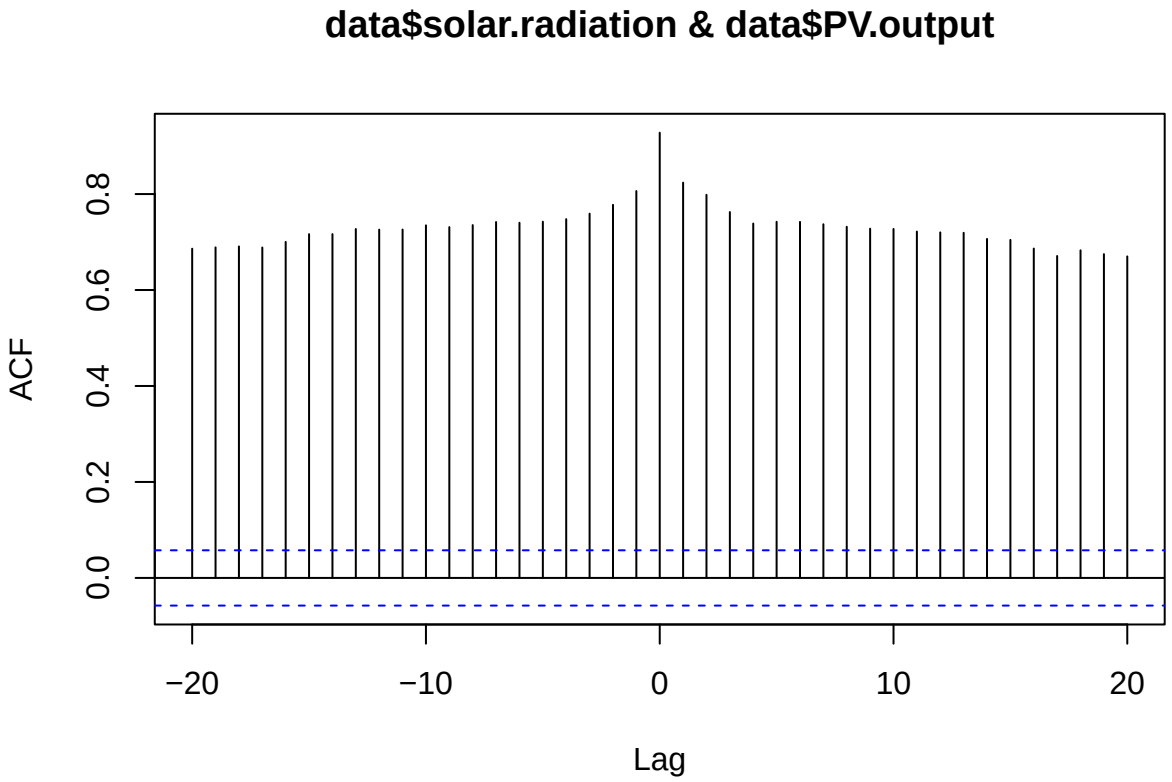

```
# Execute the plot
print(ggplot)
```

```
## function (data = NULL, mapping = aes(), ..., environment = parent.frame())
## {
##     UseMethod("ggplot")
## }
## <bytecode: 0x111379e48>
## <environment: namespace:ggplot2>
```

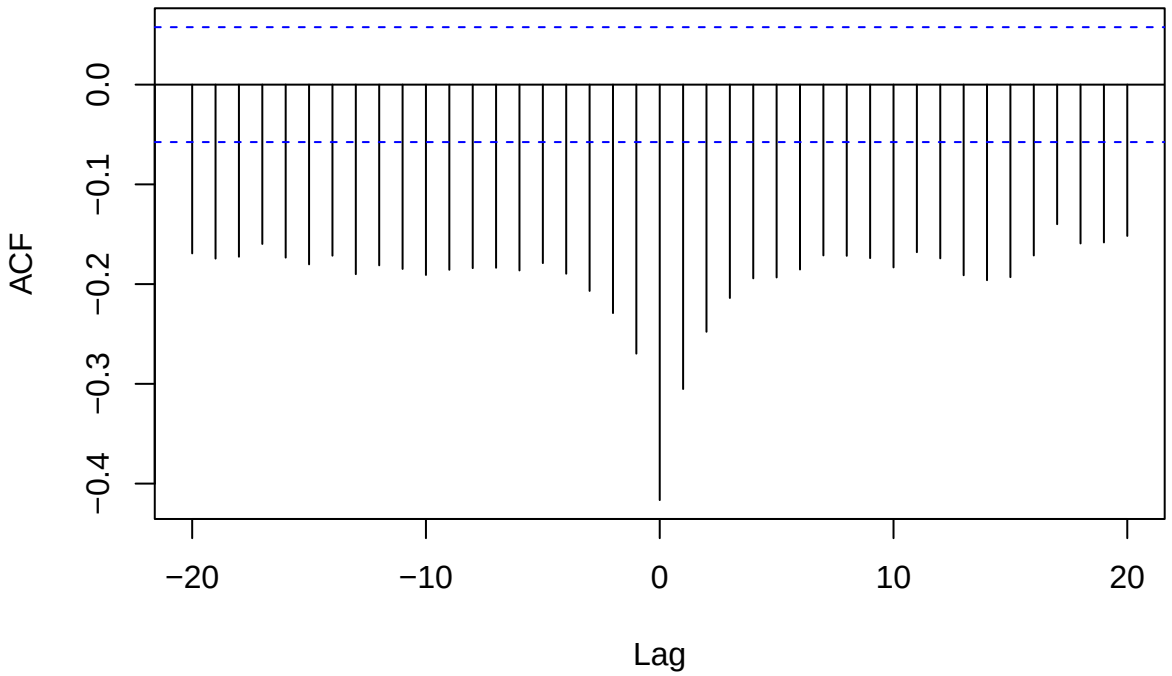
Cross Covariance, Cross Correlation and Granger Causality Tests for Solar Radiation and Cloud Cover

Here we conduct cross-correlation, cross-covariance, and Granger Causality tests to statistically investigate the relationships between solar radiation, cloud cover, and PV output. Cross-correlation and cross-covariance analyses determine the degree and lead-lag relationship between the variables across different times. Granger Causality tests further explore whether solar radiation and cloud cover have a significant predictive influence on PV output. These tests are crucial for identifying potential predictors for PV output, which can be instrumental for forecasting models and enhancing the efficiency of solar energy production systems. The goal is to establish a statistical basis for causality and correlation that may inform operational strategies and understand the dynamics affecting solar power generation.

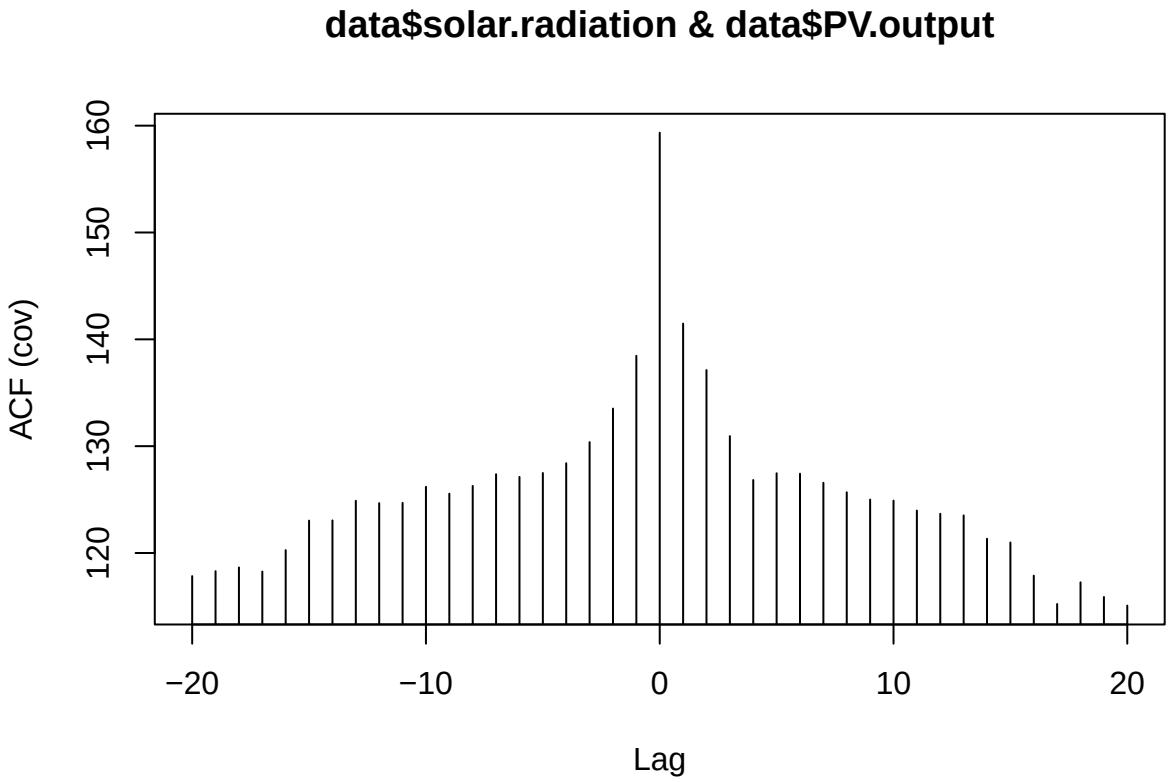
```
# Cross-correlation
ccf_solar_pv <- ccf(data$solar.radiation, data$PV.output, lag.max = 20, plot = TRUE)
```



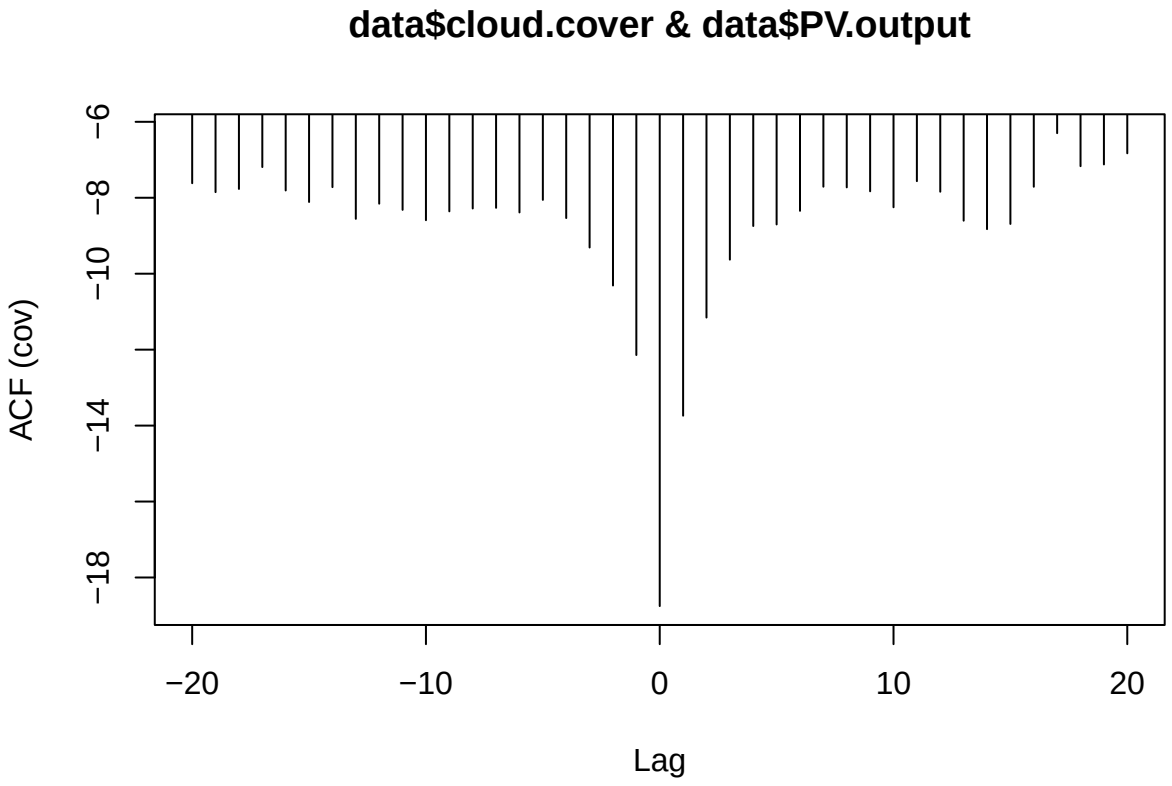

```r
ccf_cloud_pv <- ccf(data$cloud.cover, data$PV.output, lag.max = 20, plot = TRUE)
```




```r
# Cross-covariance
cov_solar_pv <- ccf(data$solar.radiation, data$PV.output, lag.max = 20, type = "covariance", plot = TRU
```




```
cov_cloud_pv <- ccf(data$cloud.cover, data$PV.output, lag.max = 20, type = "covariance", plot = TRUE)
```




```
# Granger Causality Test
# The VAR model should be fitted with a suitable lag order first
# Combine the variables into a new data frame for the VAR model
```

```r
var_data <- cbind(data$solar.radiation, data$cloud.cover, data$PV.output)

# Convert to a time series object
ts_data <- ts(var_data)

# Fit VAR model - choose appropriate lag based on your data
lag_order <- VARselect(ts_data, lag.max = 10, type = "both")$selection["AIC"]
var_model <- VAR(ts_data, p = 2, type = "const")
print(names(coef(var_model)))
```

```
## [1] "Series.1" "Series.2" "Series.3"
```

```r
# Perform Granger Causality tests
# Series 1 is Solar Radiation
# Series 2 is Cloud Cover
granger_test_solar <- causality(var_model, cause = "Series.1")
granger_test_cloud <- causality(var_model, cause = "Series.2")

# Print the results
print(granger_test_solar)
```

```
## $Granger
##
##  Granger causality H0: Series.1 do not Granger-cause Series.2 Series.3
##
## data:  VAR object var_model
## F-Test = 1.1384, df1 = 4, df2 = 3447, p-value = 0.3365
##
##
## $Instant
##
##  H0: No instantaneous causality between: Series.1 and Series.2 Series.3
##
## data:  VAR object var_model
## Chi-squared = 458.03, df = 2, p-value < 2.2e-16
```

```r
print(granger_test_cloud)
```

```
## $Granger
##
##  Granger causality H0: Series.2 do not Granger-cause Series.1 Series.3
##
## data:  VAR object var_model
## F-Test = 14.734, df1 = 4, df2 = 3447, p-value = 6.129e-12
##
##
## $Instant
##
##  H0: No instantaneous causality between: Series.2 and Series.1 Series.3
##
## data:  VAR object var_model
## Chi-squared = 323.86, df = 2, p-value < 2.2e-16
```