Project report. Parametric modeling

Mélanie FOURNIER

2024-10-24

Carbon uptake project

In this project, we have a look at time series modeling. I chose this project for the time series aspect, as I will be taking the course in the second part of the semester.

In this project, we work with meterological data from the Integrated Carbon Observation System (ICOS). The meterogical station where the data(Mölder et al. (2024) and Heliasz et al. (2024)) is gathered is the Norunda station, located north of Uppsala, Sweden. From this station, variables such as temperature, radiation or precipitations are sampled every half hour. Then, the fluxes are calculated variables, which we can use as target variable in this project.

The target variable is chosen to be the Net Ecosystem Exchang(NEE), which is the net exchange of Co2 between the atmosphere and the soil. A negative NEE means that the earth absorbed CO2 from the atmosphere, and inversely for a positive NEE.

Preparing the data

The data is time series data that is collected every half hour, and most of the data is missing until december of 2018. We first need to prepare the data.

There are two dataset, one is called meteo, the other flux. We want to use the variable in the meteo file as predictors for the NEE, situated in the flux file.

Loading the data

First, let us load the data.

```
library(dplyr)
library(ggplot2)
library(lubridate)
library(tidyr)
library(forecast)
```

```
quarto <- TRUE
if(quarto){
    fluxes <- read.csv("ICOSETC_SE-Nor_FLUXES_L2.csv")
    meteo <- read.csv("ICOSETC_SE-Nor_METEO_L2.csv")
} else{
    fluxes <- read.csv("Project/ICOSETC_SE-Nor_FLUXES_L2.csv")
    meteo <- read.csv("Project/ICOSETC_SE-Nor_METEO_L2.csv")
}
```

The NA values are set to -9999 in the data. For practical reason, these are replaced with NAs.

```
meteo2 <- meteo %>% mutate(across(everything(), ~replace(., . == -9999, NA)))
fluxes2 <- fluxes %>% mutate(across(everything(), ~replace(., . == -9999, NA)))
```

We select only the timestamp at the start and the NEE for the flux before joining it to the main dataframe for further processing.

```
fluxes_reduced <- fluxes2 %>% select(c(TIMESTAMP_START, NEE))
```

```
df <- meteo2 %>% inner_join(fluxes_reduced, by = join_by(TIMESTAMP_START))
```

Then, in order to have a more human readable datetime format, the timestamps are converted to a date-time format. We also create another variable that only gives the month and the year as information for grouping purposes.

In addition, looking at the data directly reveals that while the first date is the first of January 2018, data is only available from the the middle of November 2018 onward. We thus filter the data to start on the first of December 2018 to only get complete months.

```
df <- df %>% mutate(TIMESTAMP_START = ymd_hm(TIMESTAMP_START)) %>%
    mutate(month = floor_date(TIMESTAMP_START, "month"))

df <- df %>% filter(month > "2018-12-01")
```

Removing highly correlated variables

The stations has 14 differents measurements of air temperature done at different altitude level, which have the name "TA_*". Each of these measurements are extremely correlated to each other with correlation of 0.98 at the mimimum. For this reason, only one of the variable is kept, namely the one which is not part of the vertical profile.

This reasoning is also repeated accross all variable where multiple measurements are available. Some of them have different amount of missing data, so we also elect to keep the measurements with the least amount of missing data in these cases.

The soil temperature also have an altitude profile measurements, which are done with multiple sensor. The standard deviation of such measurements is also available. We select the measurements which have the lowest standard deviation on average, that is TS_5.

Next is the GD variable, which corresponds to the soil heat flux. There are two different measurements and the second one is the one with the least missing values, so it is the one we keep.

Then, we have the soil water content (SWC) variable for which there are 4 levels. The first level uses 4 different sensors, while the other only use two. It is also the level with the least amount of missing data. For this reason, this is the level we keep.

Relative humidity (RH) also has measurements. We select the one with the least amount of missing data.

Shortwave incoming radiation (SW_IN) has two measurements, the second one having the least amount of missing data.

Vapour pressure deficit (VPD) also has multiple measurements, the second of which has the least amount of missing data.

Wind direction can also be removed, as it is not very relevant for the linear models we want to use to use wind direction as a numerical value.

```
#Command to get the correlation, for reproducibility
air_temp <- df %>% select(starts_with("TA")) %>% na.omit()
cor(air_temp)
```

```
df.reduced <- df %>% select(!starts_with("TA_")) #Remove the altitude profile part

soil_temp <- df.reduced %>% select("TS_5")
df.reduced <- df.reduced %>% select(!starts_with("TS")) %>% mutate("TS" = soil_temp$TS_5)

soil.heatflux <- df.reduced %>% select("G_2")
df.reduced <- df.reduced %>% select(!starts_with("G")) %>% mutate("G" = soil.heatflux$G_2)
```

```r
soil.wc <- df.reduced %>% select("SWC_1")
df.reduced <- df.reduced %>% select(!starts_with("SWC")) %>% mutate("SWC" = soil.wc$SWC_1)

rh <- df.reduced %>% select("RH_2")
df.reduced <- df.reduced %>% select(!starts_with("RH")) %>% mutate("RH" = rh$RH_2)

swin <- df.reduced %>% select("SW_IN_2")
df.reduced <- df.reduced %>% select(!starts_with("SW_IN")) %>% mutate("SW_IN" = swin$SW_IN_2)

vpd <- df.reduced %>% select("VPD_2")
df.reduced <- df.reduced %>% select(!starts_with("VPD")) %>% mutate("VPD" = vpd$VPD_2)

df.reduced <- df.reduced %>% select(!WD)
```

From hourly data to monthly

This allows us to work with the first visualisation, of monthly Net Ecosystem Exchange. Based on the available metadata, Net Ecosystem Exchange is calculated in $molCo2m^{-2}s^{-1}$. Using this, we can compute the cumulative NEE over each months.

There are a quite a lot of measurement that are not available. For this reason, it convenes to impute these missing value by the average NEE during the month.

```r
avg.NEE <- df.reduced %>% select(c(month,NEE))  %>% na.omit() %>% group_by(month) %>% summari

df.reduced2 <- df.reduced %>% full_join(avg.NEE , by = join_by(month))

#For numerical stability reasons, divide by 1e6
cum.NEE <- df.reduced2 %>% group_by(month) %>% summarise(monthlyNEE = sum(avg.NEE*30*60/1e6))
```

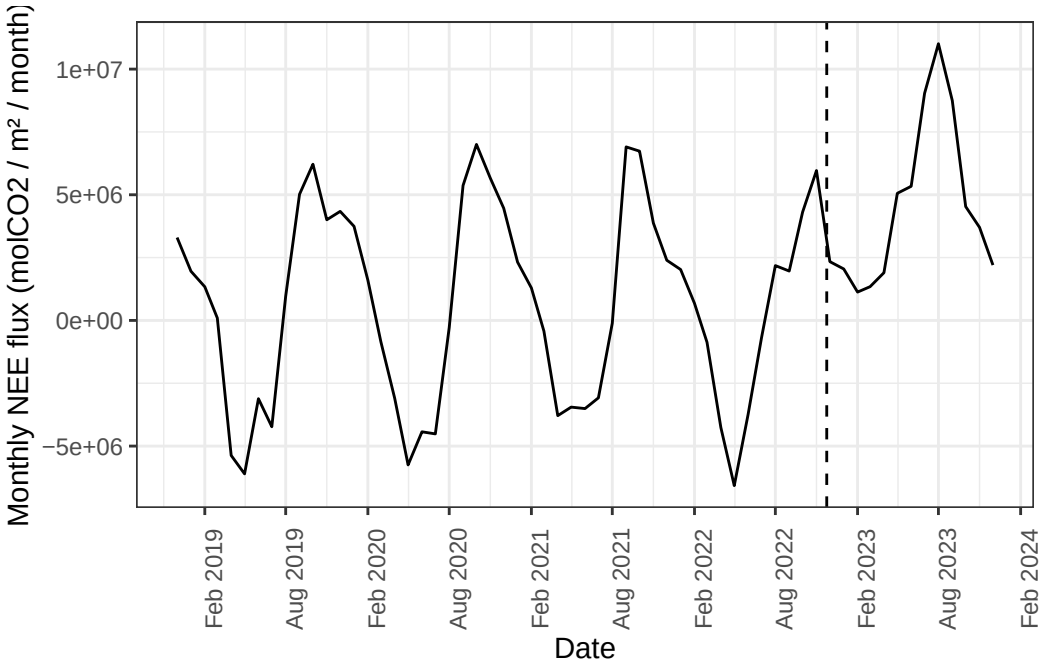


Figure 1: Monthly NEE over time. The NEE is high during autumn and low during spring. A slight upward trend is visible. The vertical line marks the date where part of the forest surronding the site was cut down.

While cumulative values make sense to measure for fluxes. It does not make sense to do so for non fluxes. For this reasons, radiation, soil and temperature measurements must be summarized. This is done with the usual summary variable such as mean, median, max, min, and other quantiles. Other summary variable of interest would number of days type variable, such as number of days with rainfall, or with snow on the soil.

In order to get the variables we will use as predictors, we remove the column we don't need first, and other columns with too many missing datapoints.

```
X <- df.reduced %>%
    select(!c(TIMESTAMP_START,TIMESTAMP_END,NEE,WTD,WTD_SD,ALB)) %>% select(!c(ends_with("_N
```

```
X_monthly <- X %>% group_by(month) %>%
    summarise_all(list(
                      ~ mean(.x,na.rm = TRUE),
                      ~ max(.x,na.rm = TRUE),
                      ~ min(.x,na.rm = TRUE),
                      ~ sd(.x, na.rm = TRUE)))
```

Additionally, we decide to remove any remaining column where there still are missing values. This mostly gets rid of variable related to pressure, humidity, and snow fall.

```
X_monthly2 <- X_monthly %>% select_if(~ !any(is.na(.)) & !any(is.infinite(.)))
```

```
X <- X_monthly2 %>% select(!month)
```

```
air.temp <-X_monthly2 %>% select(c(month,starts_with("TA"))) %>% select(!TA_sd)

air.temp %>%
    pivot_longer(
        cols = !month,
        names_to = "Temperature",
        values_to = "Air.Temperature") %>%
    ggplot(aes(x = month, y = Air.Temperature, color = Temperature)) + geom_line() +  scale_
    theme_bw() +
    theme(axis.text.x = element_text(angle = 90)) +
    xlab(element_blank()) + ylab("Air temperature") +
    scale_color_discrete(labels = c("Maximum","Mean","Minimum"))
```

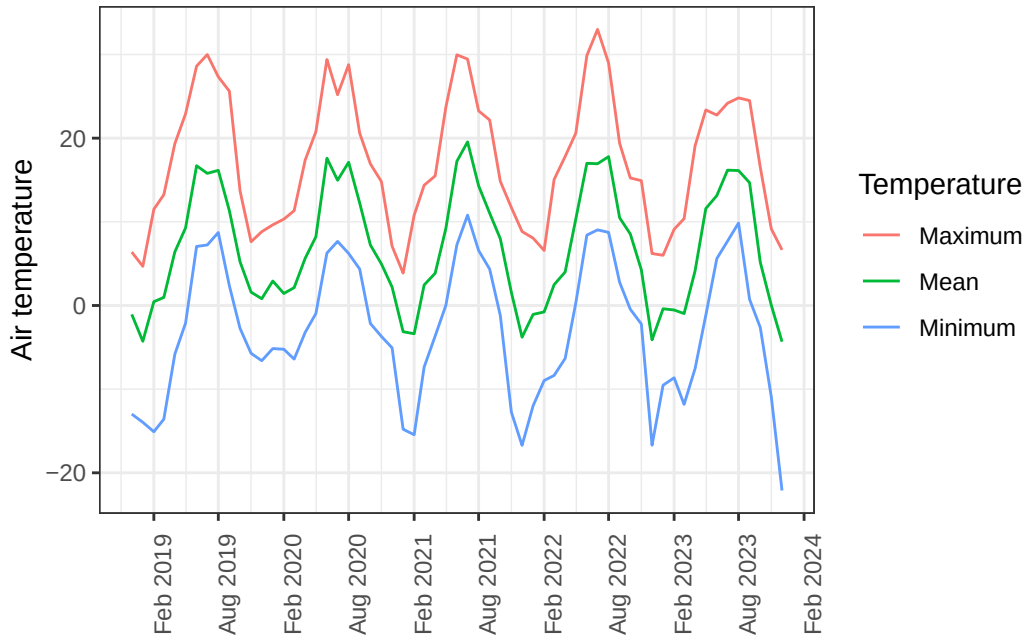


Figure 2: Air temperature at the first position in the Norunda research station.

Training-validation split.

Since we are working with correlated data, doing a random training-validation set split would lead to some significant bias as the validation data would look too much like the training one. The usual time series equivalent to cross validation would be to do some evaluation on a rolling forecasting origin@Hyndman_Athanasopoulos_2021, but this pose an issue regarding data leakage, as we will be looking to do dimensionality reduction later on.

Another issue is that the forest was cut down in the end of 2022, so setting the training-validation boundary prior to that date would lead to a model that cannot account for this change.

For this reason, we keep only the last four months as validation data. These data point will be used to check the model for any obviously bad generalization issue once we have chosen a model.

```
#Useful variables
n.val <- 4
n.all <- nrow(X_monthly2)
n.train <- n.all - n.val

X <- X_monthly2 %>% select(-month)
X.train <- X%>% slice_tail(n=-n.val)
X.val <- X %>% slice_tail(n = n.val)


y <- cum.NEE %>% select(-month)
y.train <- y %>% slice_tail(n = -n.val)
y.val <- y %>% slice_tail(n = n.val)
```

Dimensionality reduction.

We end up with a lot more variables than they are month in the observation. In fact, there are a few more variables than samples, leading to underdefined least square problem were we to use linear regression.

A standard approach would be to use PCA and select only a few principal components as predictors, which would work, but we would loose interpretability in the process.

There are only a few variable that we need to remove. We choose to manually do it by identifying set of highly correlated variables and removing one of them, as they provide the same information

```r
high.corr <- cor(X.train) > 0.97
```

```r
X.train2 <- X.train %>% select(!c(SW_IN_sd,NETRAD_max,PPFD_IN_mean,LW_OUT_mean,NETRAD_mean, I
high.corr <- cor(X.train2) > 0.97
```

Models

Additional variable

A variable of interest would be to account for the fact that the forest was cut in the end of 2022.

```r
n.months.before.cut <- nrow(X_monthly2 %>% filter(month < "2022-12-01"))
forest_status <- c(rep(0,n.months.before.cut),rep(1,n.train-n.months.before.cut))

X.val$forest_status <- 1
```

Model with no interaction

The first model is simply to forget that the data comes from a time series, and just consider a linear regression model. We work in both direction, with the upper model being the full model, and the lower model being the intercept only model. Once model is selected based on BIC, the other uses a stricter penalty term on the number of DoF for the regression.

```r
data <- cbind(X.train2,forest_status, y.train)

model.full <- lm(monthlyNEE ~ ., data = data)
model.empty <- lm(monthlyNEE ~ 1, data = data)


scope <- list("lower" = model.empty, "upper" = model.full)
model.reduced <- step(model.empty, direction = "both", scope = scope,k = log(n.train))
model.reduced2 <- step(model.empty, direction = "both", scope = scope,k = 1.35*log(n.train))
```

The summary of the model are printed below. The first model uses 8 regressors+intercept while the second one uses 6 regressors + intercept. While the usual metrics (AIC,BIC and adjusted R2) all favor the larger model, the reason two models are considered will be shown below

```r
summary(model.reduced)
```

```
Call:
lm(formula = monthlyNEE ~ forest_status + TA_sd + TS_mean + PA_mean +
    SW_IN_mean + G_max + SW_OUT_mean + SW_OUT_max, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7172 -0.8443 -0.0721  1.2972  2.3094

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.058e+02  4.019e+01  -2.633 0.011359 *
forest_status  2.707e+00  6.318e-01   4.284 8.77e-05 ***
TA_sd         -6.690e-01  2.534e-01  -2.640 0.011144 *
TS_mean        4.468e-01  7.633e-02   5.854 4.19e-07 ***
PA_mean        1.119e+00  4.031e-01   2.775 0.007847 **
SW_IN_mean    -1.006e-01  8.231e-03 -12.219 2.42e-16 ***
G_max          3.064e-01  7.980e-02   3.839 0.000361 ***
SW_OUT_mean    5.311e-01  6.929e-02   7.664 7.12e-10 ***
SW_OUT_max    -3.004e-02  5.832e-03  -5.152 4.80e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.362 on 48 degrees of freedom
Multiple R-squared:  0.9123,    Adjusted R-squared:  0.8977
F-statistic: 62.44 on 8 and 48 DF,  p-value: < 2.2e-16
```

```r
summary(model.reduced2)
```

```
Call:
lm(formula = monthlyNEE ~ forest_status + TA_sd + TS_mean + PA_mean +
    SW_IN_mean + LW_OUT_sd, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9798 -1.2212  0.0277  1.0554  3.2724

Coefficients:
```

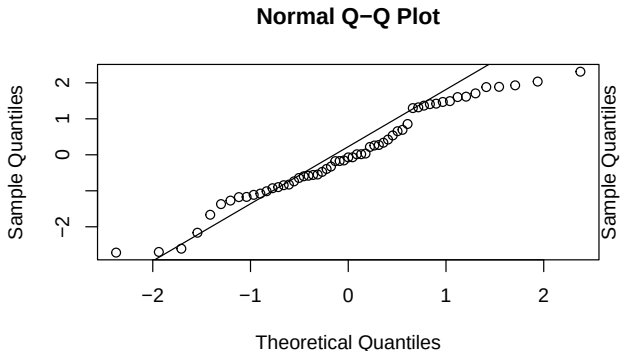
```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -116.34433   46.43488  -2.506  0.01553 *
forest_status    2.34202    0.79580   2.943  0.00492 **
TA_sd           -3.26652    0.48160  -6.783 1.31e-08 ***
TS_mean          0.47835    0.06927   6.906 8.43e-09 ***
PA_mean          1.20562    0.46672   2.583  0.01276 *
SW_IN_mean      -0.05287    0.00394 -13.418  < 2e-16 ***
LW_OUT_sd        0.55406    0.09113   6.080 1.65e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.585 on 50 degrees of freedom
Multiple R-squared:  0.8763,    Adjusted R-squared:  0.8614
F-statistic: 59.03 on 6 and 50 DF,  p-value: < 2.2e-16
```

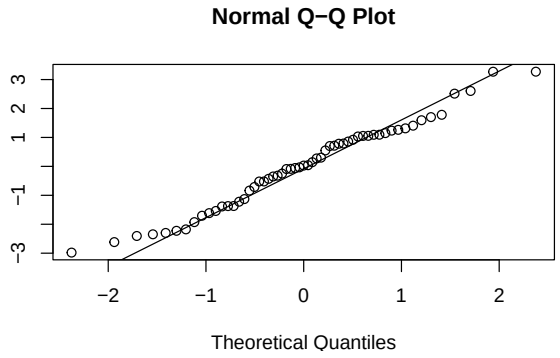
Residual analysis

In term of residual analysis, we can first look at the distribution of the residuals with a qqplot. The first model unfortunately does not respect the assumption that the residuals are normally distributed, while the second one looks acceptable.

```r
resid <- model.reduced$residuals
qqnorm(resid)
qqline(resid)
resid2 <- model.reduced2$residuals
qqnorm(resid2)
qqline(resid2)
```



(a) QQ plot of the residuals for the first model (8 regressors)



(8(a) QQ plot of the residuals for the second model (6 regressors)

Another assumption of the linear model is the assumption that the residuals are independent, and thus are not correlated. In that regards, the first model performs better than the second one. In the first models, there is still quite a bit of correlation left at early lags. In the second model, there is a peak in correlation at lag 12 and 13, indicating some seasonality is left in the data.

```
checkresiduals(model.reduced)
```

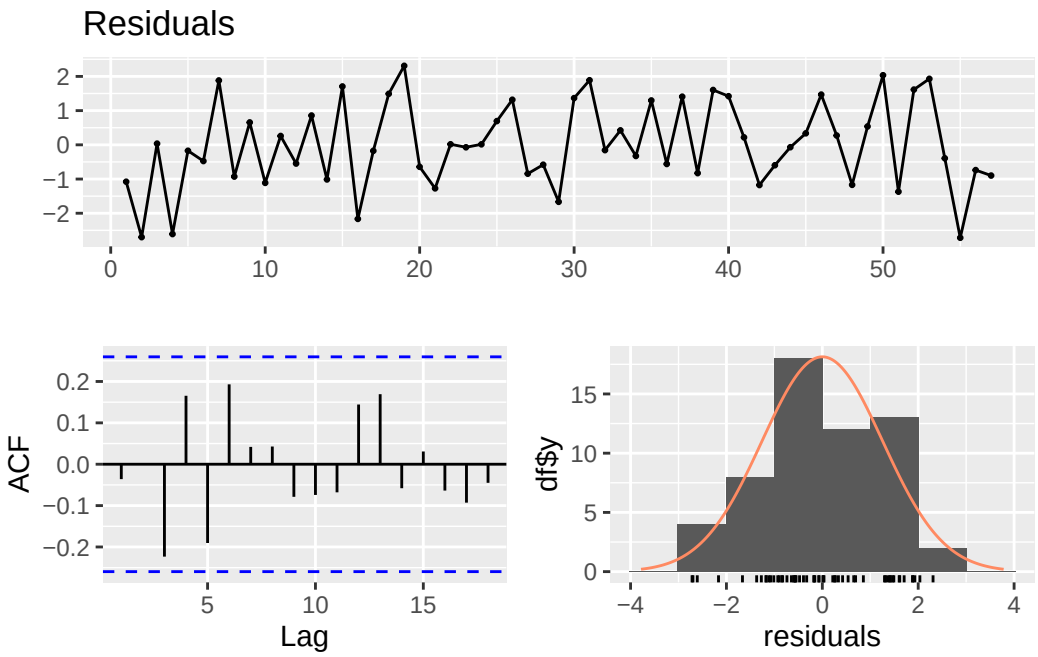


Figure 5: QQ plot of the residuals for the first model (8 regressors)

```
    Breusch-Godfrey test for serial correlation of order up to 12

data:  Residuals
LM test = 14.021, df = 12, p-value = 0.2994
```

```
checkresiduals(model.reduced2)
```

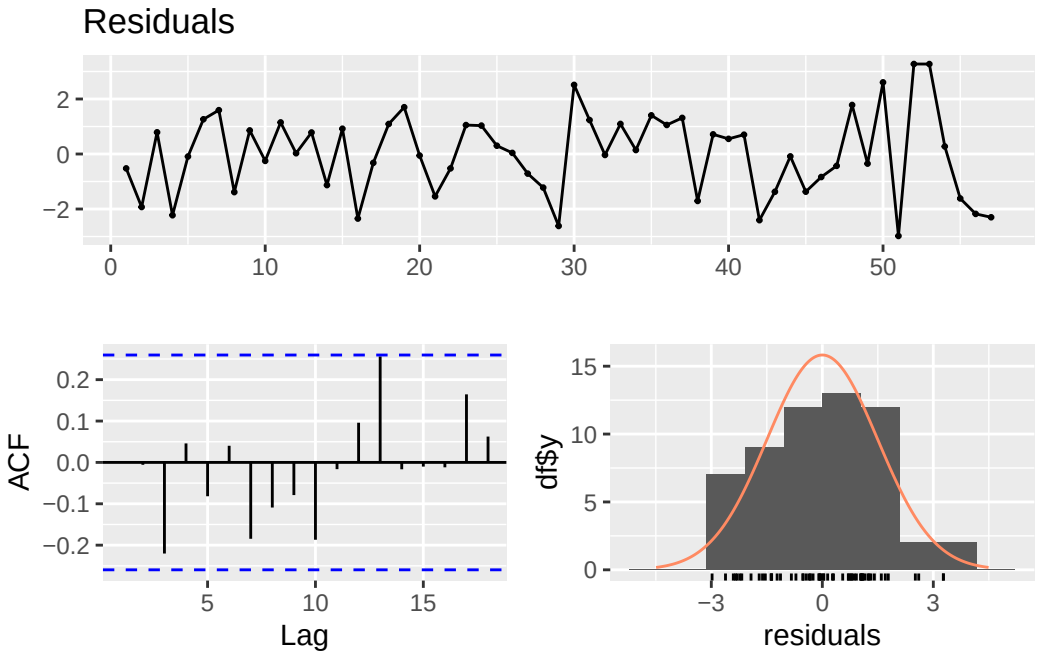


Figure 6: QQ plot of the residuals for the second model (6 regressors)

```
    Breusch-Godfrey test for serial correlation of order up to 10

data:  Residuals
LM test = 18.65, df = 10, p-value = 0.04493
```

ARIMA model

In order to get rid of the autocorrelation of the residuals, we can try to model them as an ARMA process. For the first model, an ARIMA(2,0,0)(0,0,1) model with a seasonality of 12 months is the model retained after some trial and error. For the second one, there was no ARIMA model that yielded satisfying results

```
design.matrix <- as.matrix(model.reduced$model %>% select(!monthlyNEE))

season <- list("order" = c(0,0,1),"period" = 12)
arima.model <- Arima(y.train,xreg = design.matrix, order = c(2,0,0),seasonal = season)
#arima.model <- arima(y.train,xreg =as.matrix(X.pca.reduced), seasonal = season, order = c(3
summary(arima.model)
```

```
Series: y.train
Regression with ARIMA(2,0,0)(0,0,1)[12] errors

Coefficients:
         ar1     ar2    sma1  intercept  forest_status    TA_sd  TS_mean
      0.0810  0.2211  0.3490   -93.4457         2.9631  -0.4915   0.4550
s.e.  0.2991  0.3900  0.2205    40.7401         0.8789   0.3133   0.0918
      PA_mean  SW_IN_mean   G_max  SW_OUT_mean  SW_OUT_max
       0.9838     -0.0944  0.3354       0.4523     -0.0262
s.e.   0.4157      0.0146  0.0904       0.1637      0.0090

sigma^2 = 1.827:  log likelihood = -92.15
AIC=210.31   AICc=218.77   BIC=236.87

Training set error measures:
                     ME     RMSE       MAE      MPE     MAPE      MASE
Training set 0.05013698 1.200998 0.9367961 19.66947 47.93258 0.4666692
                   ACF1
Training set 0.0276864
```

Residual analysis and inverse AR roots

The residual analysis is good, the residuals are no longer autocorrelated nothing indicates that they are not normally distributed. A Ljung-Box test is also run with results that are not very conclusive. The inverse of the roots of the lag polynomials are also checked to be inside the unit circle so the model is stable.

```r
resid.arima <- arima.model$resid
qqnorm(resid.arima)
qqline(resid.arima)
```

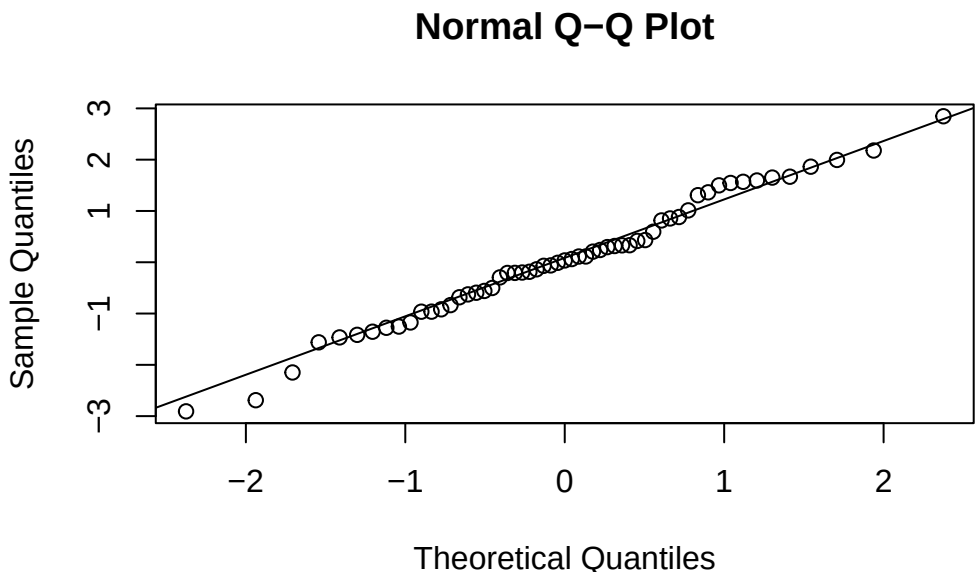


Figure 7: QQ plot for the ARIMA model

```
checkresiduals(arima.model)
```

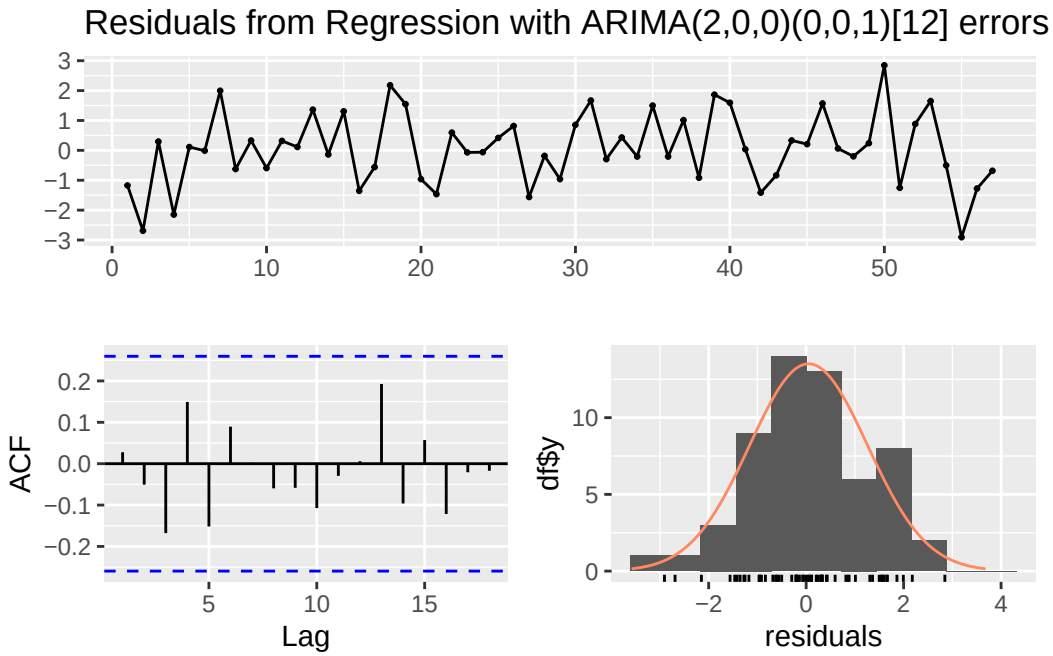


Figure 8: Residual plots for the ARIMA model.

```
    Ljung-Box test

data:  Residuals from Regression with ARIMA(2,0,0)(0,0,1)[12] errors
Q* = 6.691, df = 7, p-value = 0.4617

Model df: 3.   Total lags used: 10
```

```r
autoplot(arima.model)
```

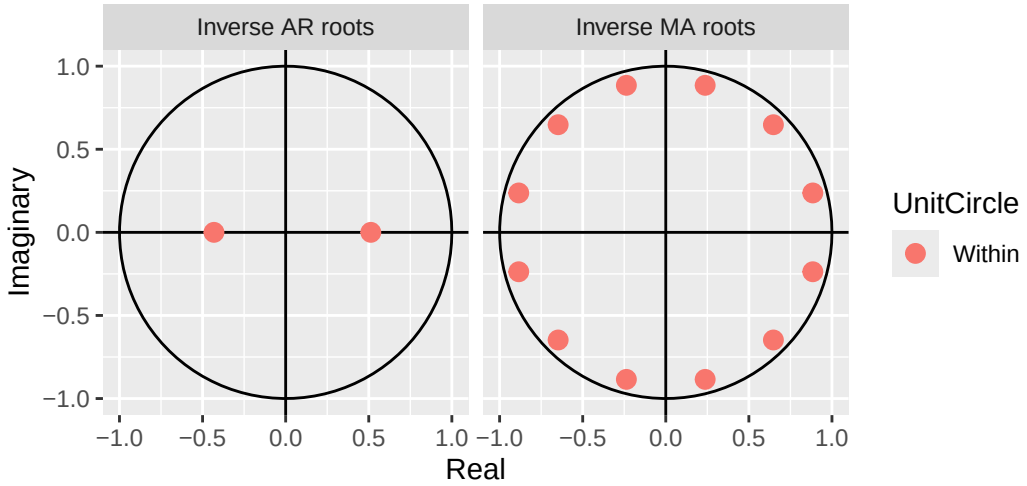


Figure 9: Inverse root of the ARIMA model coefficients. All the inverse roots are inside the unit circle, so the model is stable.

Validation

Due to the limited amount of data, the validation set is knowingly chosen to be quite small, so the use of the validation data is limited. We predict the

```r
pred.model1 <- predict(model.reduced, X.val)
pred.model2 <- predict(model.reduced2, X.val)

X.val.reduced <- X.val %>% select(forest_status, TA_sd, TS_mean, PA_mean,
    SW_IN_mean,G_max,SW_OUT_mean,SW_OUT_max)
pred.arima <- predict(arima.model, newxreg = X.val.reduced)
pred <- list(model1 = pred.model1,actual_value = y.val)

pred %>% as.data.frame() %>% knitr::kable()
```

Table 1: Predicted value against label in the validation dataset. The results are not exactly conclusive.

model1	monthlyNEE
9.139169	8.745927
7.246776	4.529627
4.439029	3.700888
3.100125	2.201651

The predictions are quite accurate.

Forecasting

In order to forecast, we retrain all the models but with all the data.

```
forest_status <- c(rep(0,n.months.before.cut),rep(1,n.all-n.months.before.cut))
X_monthly2$forest_status <- forest_status

X_monthly2_reduced <-
#model.reduced$call

data <- cbind(X_monthly2,y)
model.final <- lm(formula = monthlyNEE ~ forest_status + TA_sd + TS_mean + PA_mean +
    SW_IN_mean + G_max + SW_OUT_mean + SW_OUT_max, data = data)


design.matrix.final <- as.matrix(model.final$model %>% select(!monthlyNEE))
arima.model.final <- Arima(y,xreg = design.matrix.final, order = c(2,0,0),seasonal = season)
```

The regressor value also need to be extrapolated. One strategy would be to fit an ARIMA model to each of the regressor except the one with the status of the forest which we know will still be clear cut for the foreseeable future, then use pointwise prediction as a "scenario" for the future. Another approach is to repeat the values of the same month for the last year, and use them as regressor once again. The former approach introduce a lot of additional fitting and may not be the most accurate, while the latter approach has the advantage of having a scenario that we know possible.

```
fit.and.predict <- function(variable, n.months){
    model <- auto.arima(X_monthly2 %>% select(all_of(variable)))
    pred <- predict(model,n.months)
    return(as.vector(pred$pred))
}
```

```
future <- forecast(arima.model.final,xreg = as.matrix(newdata))
autoplot(future)
```

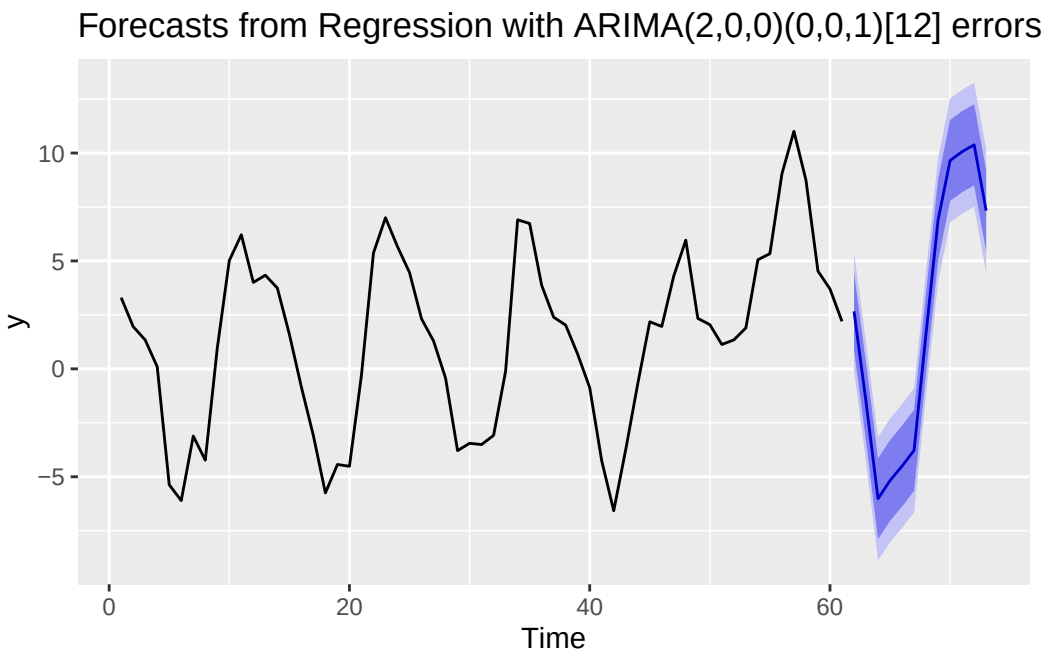


Figure 10: 12 months forecast using the arima model, conditional on the values given by the auto arima model for the regressor.

```
future <- forecast(arima.model.final,xreg = as.matrix(newdata2))
autoplot(future)
```

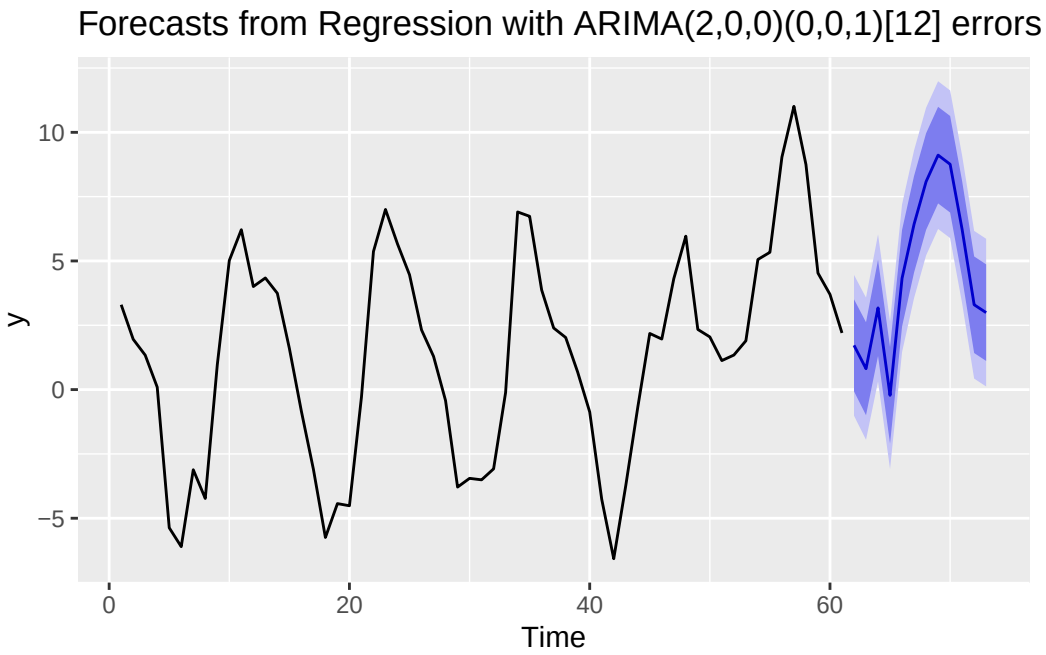


Figure 11: 12 months forecast using the arima model, conditional on a repeat of the values of the regressor from the same month last year.

The ARIMA fit scenario is more optimistic and predicts that the earth will absorb some carbon from the atmosphere during the winter-spring months, while the "repeat last year" scenario is less so. Time will tell if any of the two prediction is correct, my guess is on the pessimistic one.

Misc - Visualization of the missing values

The missing values in NEE were imputed, but before imputing, it was interesting to check if there was a pattern in the missing values or if it was truly random. For example, some sensors may undergo maintenance at regular intervals. For these cases, it would not be appropriate to simply impute the way it was done during the project, as this would most likely bias the results.

```
test <- fluxes_reduced %>% filter(is.na(NEE)) %>% mutate(datetime = ymd_hm(TIMESTAMP_START))
```

```
test %>% group_by(hour = hour(datetime)) %>% count() %>%
    ggplot(aes(x = hour, y = n)) + geom_col()
```

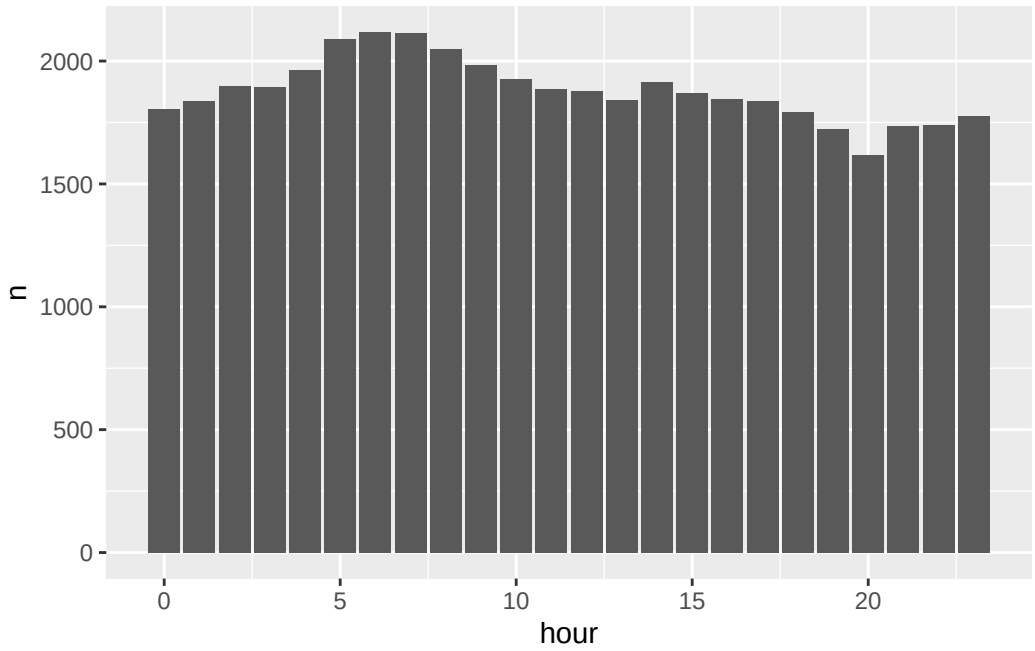
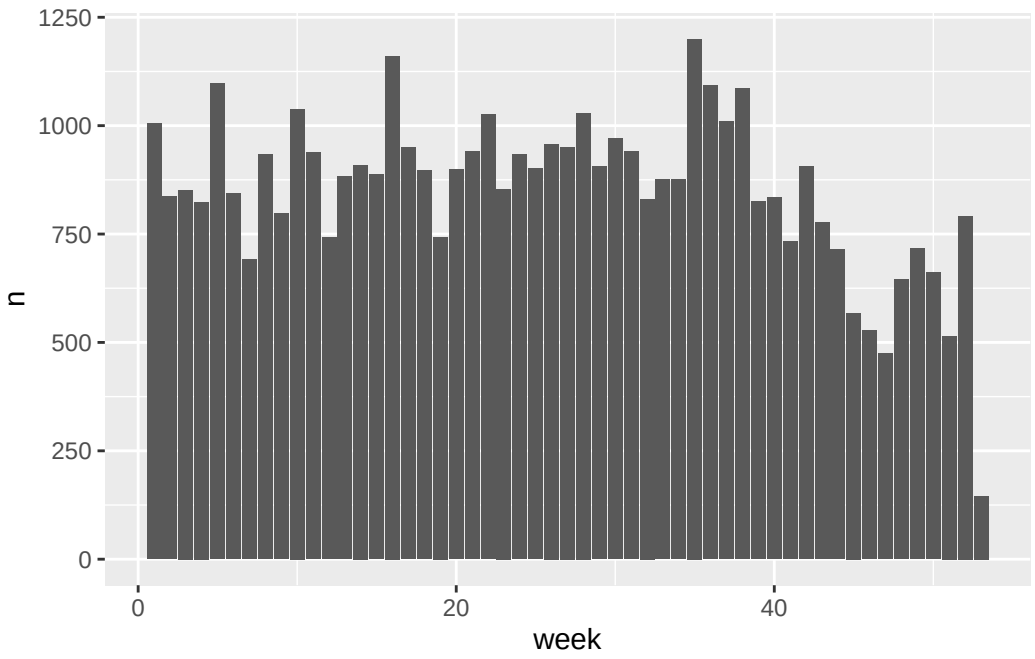
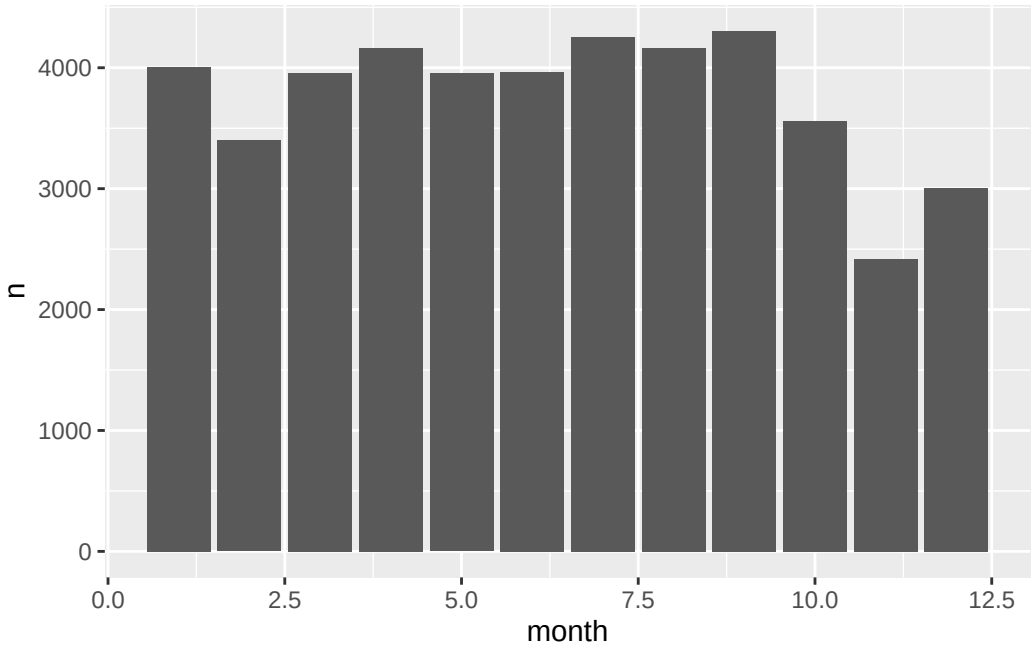


Figure 12: There does not seem to be a time of the day where values are missing in particular.

```
test %>% group_by(week = week(datetime)) %>% count() %>%
    ggplot(aes(x = week, y = n)) + geom_col()
```



```
test %>% group_by(month = month(datetime)) %>% count() %>%
    ggplot(aes(x = month, y = n)) + geom_col()
```



There is less missing values during the later months, but from a purely subjective viewpoint, this is not enough to consider this a pattern.

Heliasz, Michal, Natascha Kljun, Tobias Biermann, Jutta Holst, Thomas Holst, Maj-Lena Linderson, Meelis Mölder, and Janne Rinne. 2024. "ETC L2 Fluxes, Hyltemossa, 2017-12-31–2023-12-31." Ecosystem Thematic Centre. https://hdl.handle.net/11676/XESY-JxtNIMxXKat66tMUD9y.

Mölder, Meelis, Natascha Kljun, Irene Lehner, Gunnar Bergström, Anders Båth, Jutta Holst, and Maj-Lena Linderson. 2024. "ETC L2 Meteo, Norunda, 2017-12-31–2023-12-31." Ecosystem Thematic Centre. https://hdl.handle.net/11676/G6RuDZf-v0-Kym20ToJrmwf5.