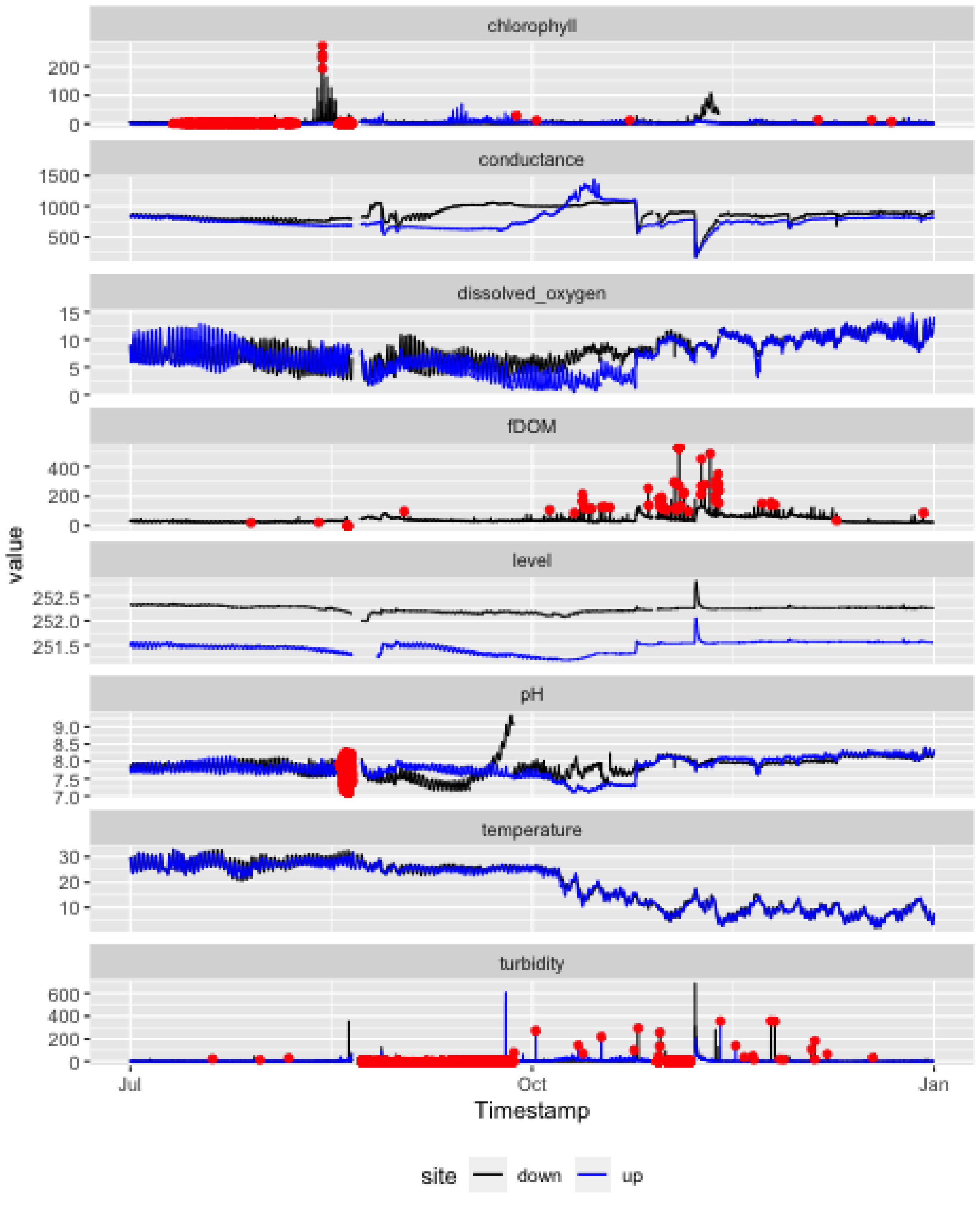
# Anomaly detection in data from PRIN (Pringle Creek TX) in Texas

## Load original data with outliers

```r
load(here::here("weird_river_data_paper", "data", "PRIN_5min_flagged.rda"))

PRIN_5min_flagged <- PRIN_5min_flagged %>%
  rename(
    Timestamp = roundedTimestamp,
    level = surfacewaterElevMean,
    conductance = specificConductance,
    dissolved_oxygen = dissolvedOxygen,
    temperature = surfWaterTempMean
  ) %>%
  mutate(Timestamp = ymd_hms(Timestamp))
```

### Visualization of data with outliers

```r
# Pivot data values from wide to long
data_val <- PRIN_5min_flagged %>%
  select(
    Timestamp, conductance, dissolved_oxygen,
    pH, turbidity, chlorophyll, fDOM,
    level, temperature, site
  ) %>%
  as_tsibble(index = Timestamp, key = site) %>%
  pivot_longer(cols = conductance:temperature)

# Pivot data flags from wide to long
data_flag <- PRIN_5min_flagged %>%
  select(
    Timestamp, conductanceAnomalyFlag,
    dissolvedOxygenAnomalyFlag,
    pHAnomalyFlag, turbidityAnomalyFlag,
    chlorophyllAnomalyFlag, fDOMAnomalyFlag,
    site
  ) %>%
  rename(
    conductance = conductanceAnomalyFlag,
    dissolved_oxygen = dissolvedOxygenAnomalyFlag,
    pH = pHAnomalyFlag,
    turbidity = turbidityAnomalyFlag,
    chlorophyll = chlorophyllAnomalyFlag,
    fDOM = fDOMAnomalyFlag
  ) %>%
  as_tsibble(index = Timestamp, key = site) %>%
  pivot_longer(cols = conductance:fDOM)

# Combine values and flag data

data <- left_join(data_val, data_flag,
  by =
    c("Timestamp", "site", "name")
) %>%
  rename(
    value = value.x,
    flag = value.y
  )

data_outliers <- data %>%
  filter(flag == 1)


p <- data %>%
  ggplot(aes(x = Timestamp, y = value, color = site)) +
  geom_line() +
  facet_wrap(~name, ncol = 1, scales = "free_y") +
  scale_color_manual(values = c("down" = "black", "up" = "blue")) +
  theme(legend.position = "bottom") +
  geom_point(data = data_outliers, aes(x = Timestamp, y = value), color = "red")

print(p)
```


chlorophyll
200
100
0
conductance
1500
1000
500
dissolved_oxygen
15
10
5
0
fDOM
400
200
0
level
252.5
252.0
251.5
pH
9.0
8.5
8.0
7.5
7.0
temperature
30
20
10
turbidity
600
400
200
0
value
Jul
Oct
Jan
Timestamp
site
down
up

## Visualization of upstream data with outliers

```r
data_outliers <- data %>%
  filter(flag == 1, site == "up")

p1 <- data %>%
  filter(site == "up") %>%
  ggplot(aes(x = Timestamp, y = value)) +
  geom_line() +
  facet_wrap(~name, ncol = 1, scales = "free_y") +
  theme(legend.position = "bottom") +
  geom_point(data = data_outliers, aes(x = Timestamp, y = value), color = "red")

print(p1)
```

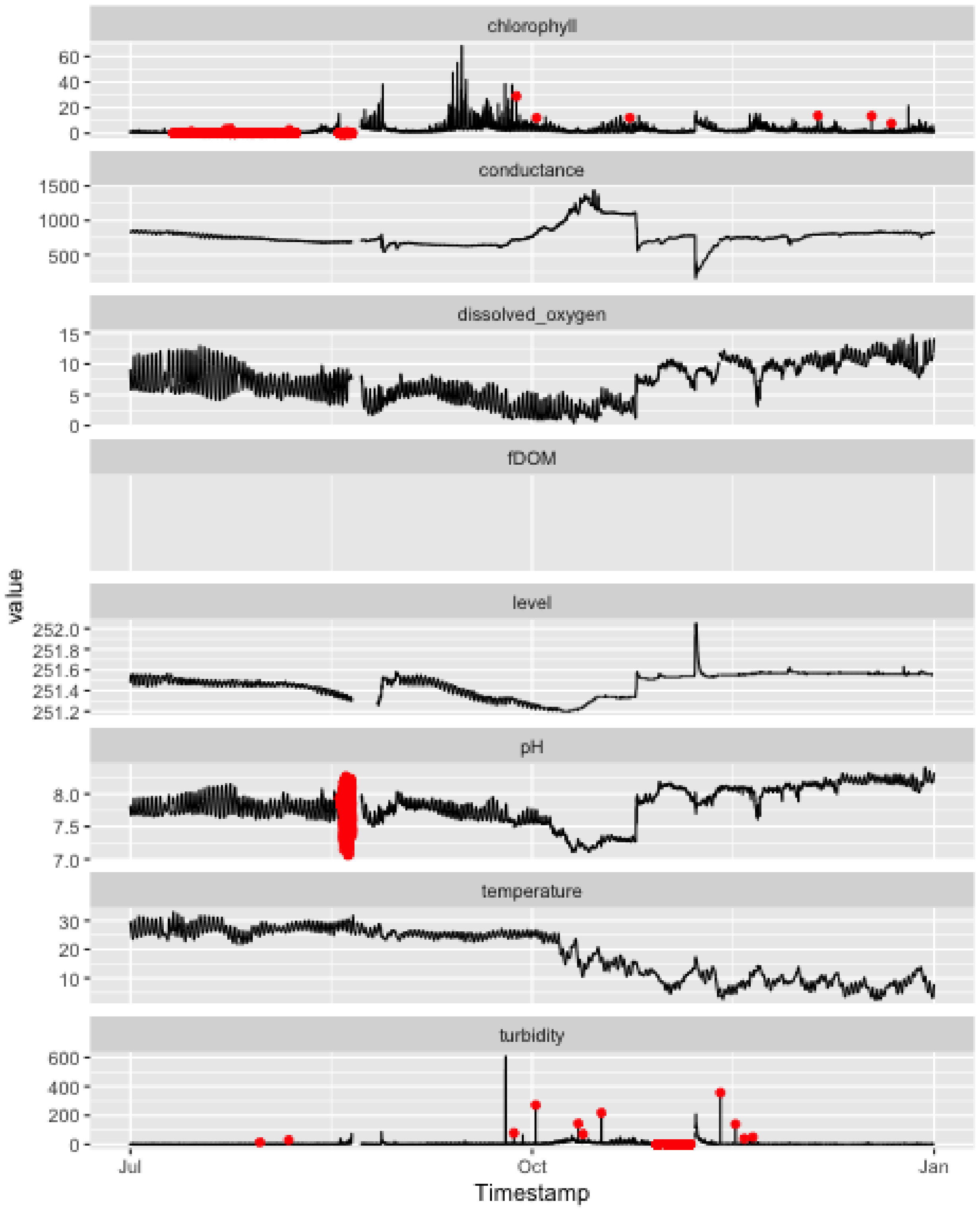


Figure 1: Upstream data

## Visualization of downstream data with outliers

```
data_outliers <- data %>%
  filter(flag == 1, site == "down")

p2 <- data %>%
  filter(site == "down") %>%
  ggplot(aes(x = Timestamp, y = value)) +
  geom_line() +
  facet_wrap(~name, ncol = 1, scales = "free_y") +
  theme(legend.position = "bottom") +
  geom_point(data = data_outliers, aes(x = Timestamp, y = value), color = "red")

print(p2)
```

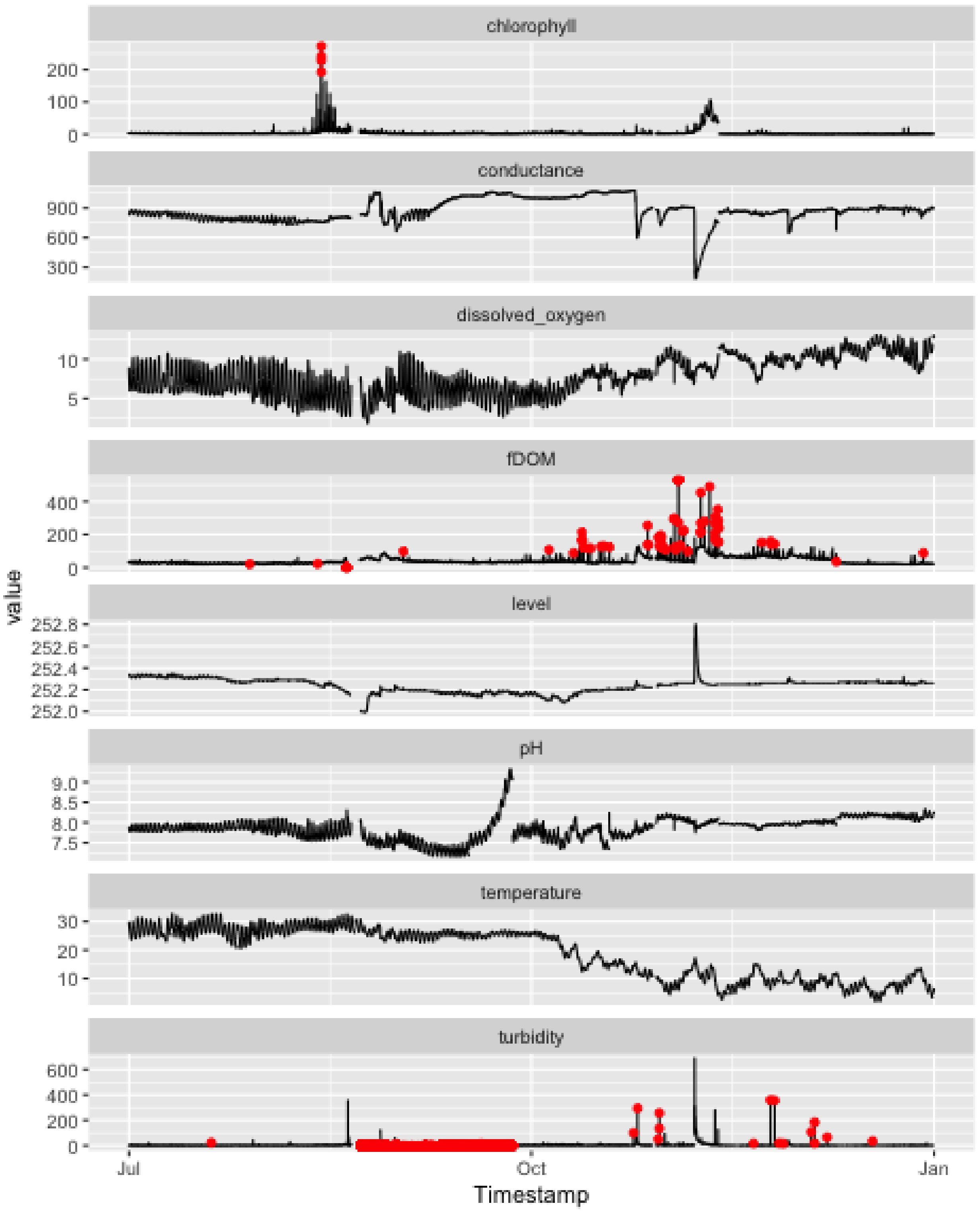


Figure 2: Downstream data

```
data_down <- data %>%
  filter(site == "down") %>%
  pivot_wider(names_from = name,
              values_from =
                c(value, flag)) %>%
  rename_all(paste0, "D") %>%
  rename(
    "Timestamp" = TimestampD
  )

data_up <- data %>%
  filter(site == "up") %>%
  pivot_wider(names_from = name,
              values_from =
                c(value, flag)) %>%
  rename_all(paste0, "U") %>%
  rename(
    "Timestamp" = TimestampU
  )

data_full <- left_join(data_up,data_down, by = "Timestamp") %>%
  mutate(flag_turbidityD = as_factor(flag_turbidityD))

p1 <- data_full %>%
  select(value_turbidityD,
         value_conductanceU,
         value_dissolved_oxygenU,
         value_pHU,
         value_chlorophyllU,
         value_turbidityU,
         value_fDOMU,
         value_levelU,
         value_temperatureU,
         flag_turbidityD) %>%
  pivot_longer(cols =
        c(value_conductanceU:value_temperatureU)) %>%
  filter(flag_turbidityD %in% c(0,1) ) %>%
  ggplot(aes(value, value_turbidityD,
             color =
            flag_turbidityD)) +
  geom_point() +
  facet_wrap(~name, scales = "free_x", ncol = 4)+
  theme(aspect.ratio = 1)

p2 <- data_full %>%
  select(value_turbidityD,
         value_conductanceD,
         value_dissolved_oxygenD,
         value_pHD,
         value_chlorophyllD,
         value_chlorophyllD,
         value_fDOMD,
         value_levelD,
         value_temperatureD,
         flag_turbidityD) %>%
 pivot_longer(cols =
        c(value_conductanceD:value_temperatureD)) %>%
   filter(flag_turbidityD %in% c(0,1) ) %>%
  ggplot(aes(value, value_turbidityD, color = flag_turbidityD)) +
  geom_point() +
  facet_wrap(~name, scales = "free_x", ncol = 4)+
  theme(aspect.ratio = 1)

p1/p2
```

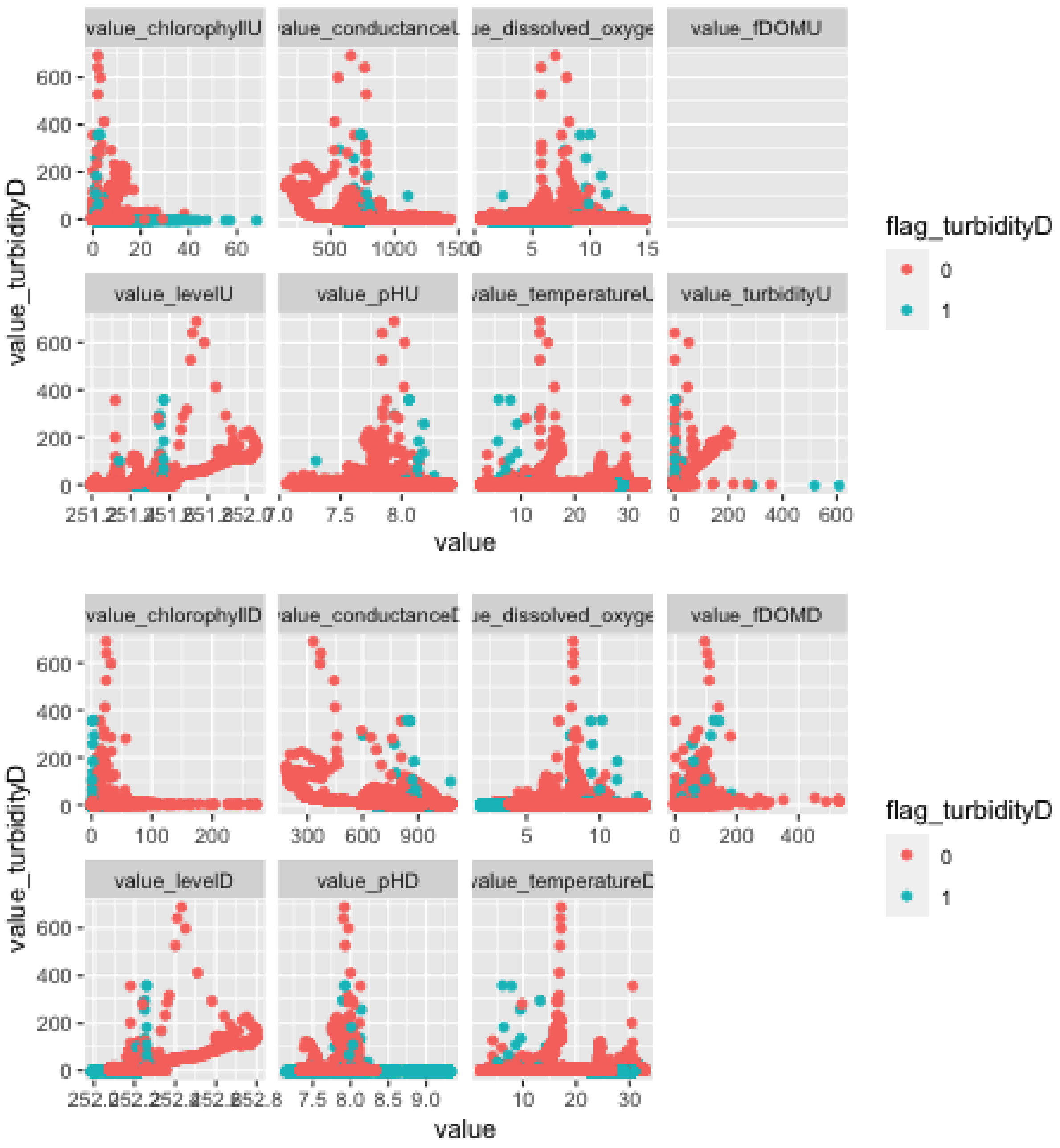


Figure 3: Relationships between turbidity downstream and other variables

- In Pringle Creek, the two locations are situated about 200m apart where a small tributary entering to the main creek between the two sensors.

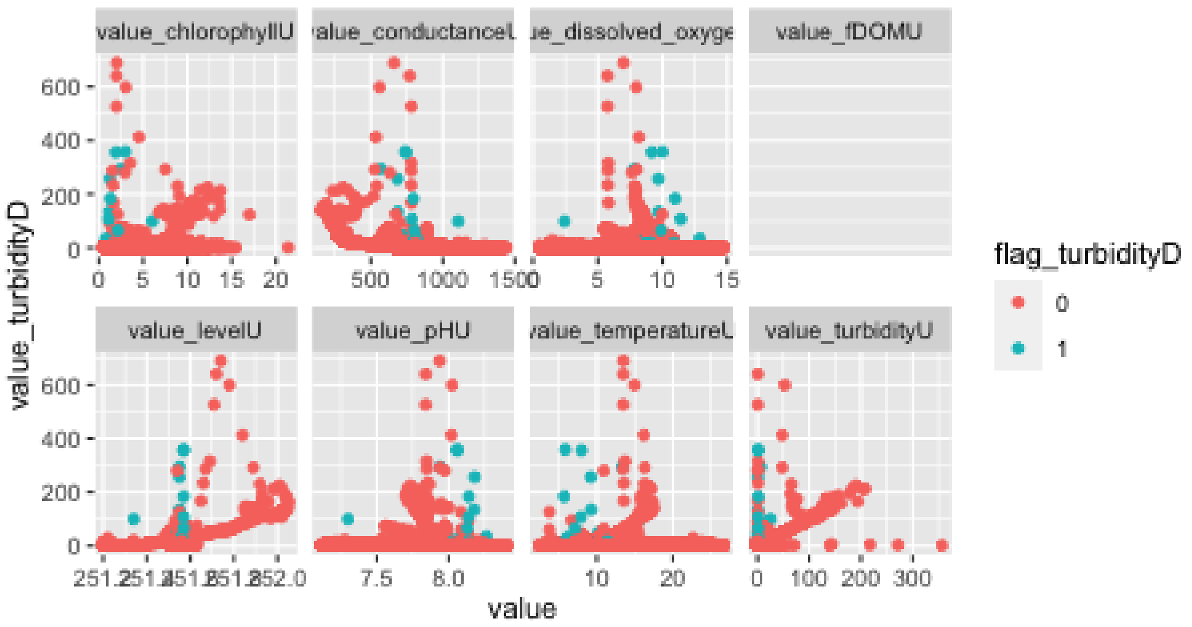
The analysis was restricted to the time span from 01-October-2019 to 31-December-2019.

This time span avoids the summer period in which surface pools of water disconnect and contain the least amount of missing observations after removing the anomalies.

- Time span - 01-October-2019 to 31-December-2019
- Frequency - 5 minutes
- This dataset contains the following water quality measurements
  - Surface water elevation (level)
  - dissolved oxygen
  - dissolved oxygen saturation
  - pH
  - fDOM
  - chlorophyll
  - specific conductance and
  - turbidity

## Time span - 01-October-2019 to 31-December-2019

```
p1 <- data_full %>%
  filter(Timestamp >=
           ymd("2019-10-01") &
           Timestamp <
           ymd("2020-01-01")) %>%
  select(value_turbidityD,
         value_conductanceU,
         value_dissolved_oxygenU,
         value_pHU,
         value_chlorophyllU,
         value_turbidityU,
         value_fDOMU,
         value_levelU,
         value_temperatureU,
         flag_turbidityD) %>%
  pivot_longer(cols =
        c(value_conductanceU:value_temperatureU)) %>%
  filter(flag_turbidityD %in% c(0,1) ) %>%
  ggplot(aes(value, value_turbidityD,
             color =
            flag_turbidityD)) +
  geom_point() +
  facet_wrap(~name, scales = "free_x", ncol = 4)+
  theme(aspect.ratio = 1)

p2 <- data_full %>%
   filter(Timestamp >=
           ymd("2019-10-01") &
           Timestamp <
           ymd("2020-01-01")) %>%
  select(value_turbidityD,
         value_conductanceD,
         value_dissolved_oxygenD,
         value_pHD,
         value_chlorophyllD,
         value_chlorophyllD,
         value_fDOMD,
         value_levelD,
         value_temperatureD,
         flag_turbidityD) %>%
 pivot_longer(cols =
        c(value_conductanceD:value_temperatureD)) %>%
   filter(flag_turbidityD %in% c(0,1) ) %>%
  ggplot(aes(value, value_turbidityD, color = flag_turbidityD)) +
  geom_point() +
  facet_wrap(~name, scales = "free_x", ncol = 4)+
  theme(aspect.ratio = 1)

p1/p2
```



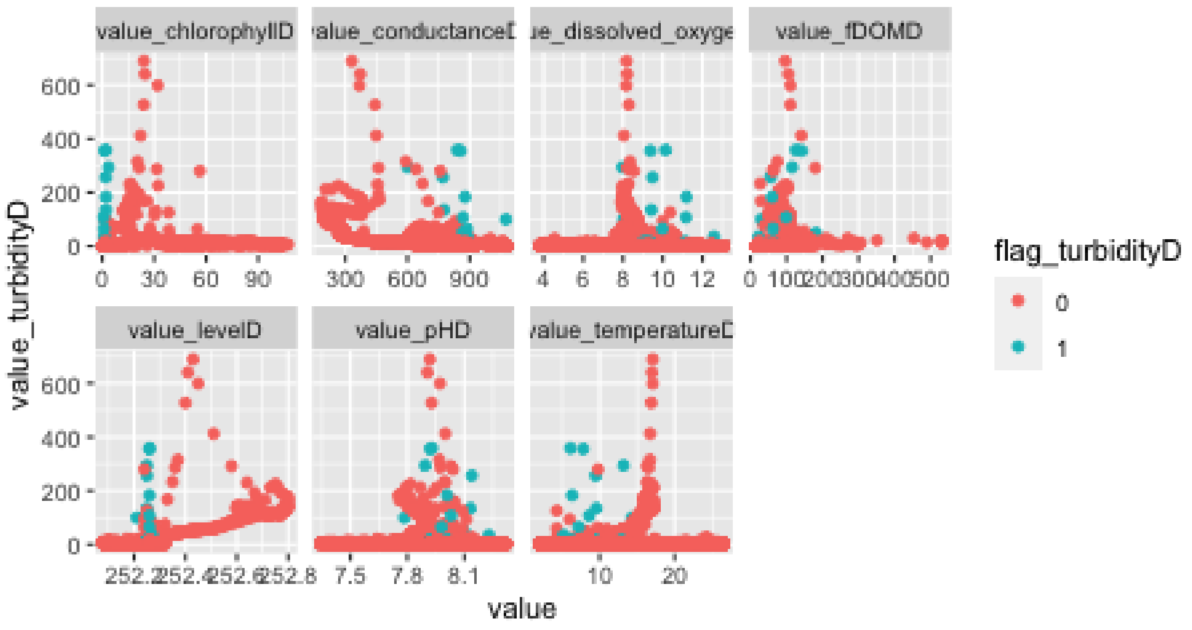


Figure 4: Relationships between turbidity downstream and other variables collected from 01-Octorber-2019 to 31-December-2019

## Analysis of Cleaned data

Robust methods are not suitable as the position of the outliers are not outside the region defined by the majority.

Therefore the cleaned dataset will be used for further analysis. As the focus of this study to identify the anomalies in the turbidity downstream data, the cleaned datasets will be updated with original data for turbidity downstream with outliers.

## Load Cleaned data

```
# Pringle Creek in Texas
load(here::here("weird_river_data_paper", "data", "PRIN_5min_cleaned.rda"))
PRIN_5min_cleaned <- PRIN_5min_cleaned %>%
  rename(
    Timestamp = roundedTimestamp,
    level = surfacewaterElevMean,
    conductance = specificConductance,
    dissolved_oxygen = dissolvedOxygen,
    temperature = surfWaterTempMean
  ) %>%
  mutate(Timestamp = ymd_hms(Timestamp))
```

Refer paper titled "Conditional normalization in time series analysis" for the data cleaning process (Section 4.2.1 , 4.2.1 and Appendix B )

## Visualization of cleaned data

```
# Pivot data values from wide to long
data_cleaned <- PRIN_5min_cleaned %>%
  select(Timestamp, conductance, dissolved_oxygen,
       pH, turbidity, chlorophyll, fDOM,
       level, temperature, site ) %>%
  as_tsibble(index = Timestamp, key = site) %>%
  pivot_longer(cols = conductance:temperature)

p <- data_cleaned %>%
  ggplot(aes(x = Timestamp, y = value, color = site)) +
  geom_line() +
  facet_wrap(~name, ncol = 1, scales = "free_y") +
  scale_color_manual(values = c("down" = "black", "up" = "blue")) +
  theme(legend.position='bottom')

print(p)
```

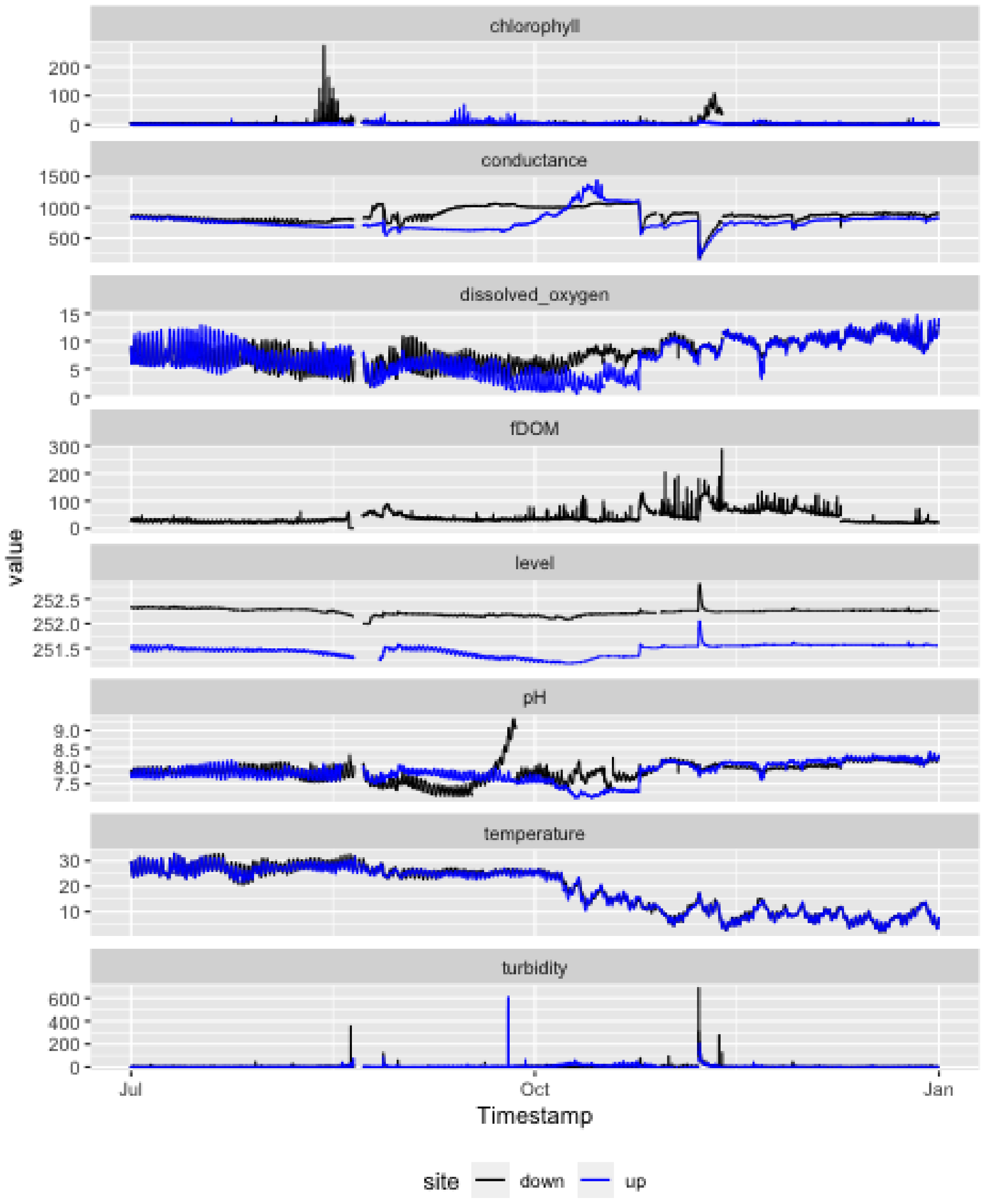


Figure 5: Cleaned data

## Preprocessing of cleaned data for the analysis

Robust methods are not suitable as the position of the outliers are not outside the region defined by the majority.

Therefore the cleaned data set will be used for further analysis. As the focus of this study to identify the anomalies in the turbidity downstream data, the cleaned datasets will be updated with original data for turbidity downstream with outliers.

```r
dataclean_up <- PRIN_5min_cleaned %>%
  filter(site == "up") %>%
  select(Timestamp, conductance,
         dissolved_oxygen, pH, turbidity,
         chlorophyll, fDOM, level,
         temperature) %>%
  rename(
    "conductance_up" = conductance,
    "DO_up" = dissolved_oxygen,
    "pH_up" = pH,
    "turbidity_up" = turbidity,
    "chlorophyll_up" = chlorophyll,
    "fDOM_up" = fDOM,
    "level_up" = level,
    "temperature_up" = temperature
  )


dataclean_down <- PRIN_5min_cleaned %>%
  filter(site == "down") %>%
  select(Timestamp, conductance,
         dissolved_oxygen, pH, turbidity,
         chlorophyll, fDOM, level,
         temperature) %>%
  rename(
    "conductance_down" = conductance,
    "DO_down" = dissolved_oxygen,
    "pH_down" = pH,
    "turbidity_down" = turbidity,
    "chlorophyll_down" = chlorophyll,
    "fDOM_down" = fDOM,
    "level_down" = level,
    "temperature_down" = temperature
  )

dataclean_full <- left_join(dataclean_up,dataclean_down, by = "Timestamp" )


data_turb_flag <- PRIN_5min_flagged %>%
  filter(site == "down") %>%
  select(Timestamp, turbidity, turbidityAnomalyFlag) %>%
  rename(
    "turbidity_down" = turbidity ,
    "turb_Dflag" = turbidityAnomalyFlag
  )
```

```
dataclean_full <- dataclean_full %>%
  select(-turbidity_down)

dataclean_full <- left_join(dataclean_full,data_turb_flag, by = "Timestamp" )

dataclean_full <- dataclean_full %>%
   mutate(turb_Dflag = as.factor(turb_Dflag)) %>%
  filter(Timestamp>= ymd("2019-10-01") & Timestamp< ymd("2020-01-01"))
```

## Predicting time delay

In this analysis we are trying to identify technical anomalies in the turbidity in downstream sensor by modeling turbidity downstream using related upstream variables. To build this model, we want to compute the lagged values of the upstream variables by incorporating the time it takes to flow from upstream to the downstream sensor. Hence the first step is to estimate the time delay between the two sensors.

**Method 1: Using the cross correlation between normalized upstream and downstream turbidity series**

### Cross Correlation

`ccf` in `stats` package

In R, the sample CCF is defined as the set of sample correlations between and for $h = 0, \pm1, \pm2, \pm3$, and so on. A negative value for is a correlation between the $x$-variable at a time before and the $y$-variable at time . For instance, consider $h = -2$. The CCF value would give the correlation between $x_{t-2}$ and $y_t$ .

- When one or more $x_{t+h}$ , with $h$ negative, are predictors of $y_t$ , it is sometimes said that $x$ **leads** $y$.
- When one or more $x_{t+h}$ , with $h$ positive, are predictors of $y_t$ , it is sometimes said that $x$ **lags** $y$.

`CCF` in `feasts` package follows the opposite.

**Mean and Correlation Functions**

**Mean Function of a Random Process**

For a random process $\{X(t), t \in J\}$, the **mean function** $\mu_X(t) : J \to \Re$, is defined as

$$\mu_X(t) = E[X(t)]$$

The mean function gives us an idea about how the random process behaves on average as time evolves.

**Autocorrelation and Autocovariance:**

The mean function $\mu_X(t)$ gives us the expected value of $X(t)$ at time $t$, but it does not give us any information about how $X(t_1)$ and $X(t_2)$ are related. To get some insight on the relation between $X(t_1)$ and $X(t_2)$, we define correlation and covariance functions.

For a random process $\{X(t), t \in J\}$, the **autocorrelation function** or, simply, the **correlation function**, $R_X(t_1, t_2)$, is defined by

$$R_X(t_1, t_2) = E[X(t_1)X(t_2)], \text{ for } t_1, t_2 \in J$$

.

For a random process $\{X(t), t \in J\}$, the **autocovariance function** or, simply, the **covariance function**, $C_X(t_1, t_2)$, is defined by

$$C_X(t_1, t_2) = Cov(X(t_1), X(t_2))$$

$$= R_X(t_1, t_2) - \mu_X(t_1)\mu_X(t_2), \quad for \quad t_1, t_2 \in J.$$

In particular, note that

$$C_X(t_1, t_2) = E[(X(t_1) - E[X(t_1)])(X(t_2) - E[X(t_2)])].$$

Intuitively, $C_X(t_1, t_2)$ shows how $X(t_1)$ and $X(t_2)$ move relative to each other. If large values of $X(t_1)$ tend to imply large values of $X(t_2)$, then $(X(t_1) - E[X(t_1)])(X(t_2) - E[X(t_2)])$ is positive on average. In this case, $C_X(t_1, t_2)$ is positive, and we say $X(t_1)$ and $X(t_2)$ are positively correlated. On the other hand, if large values of $X(t_1)$ imply small values of $X(t_2)$, then $(X(t_1) - E[X(t_1)])(X(t_2) - E[X(t_2)])$ is negative on average, and we say $X(t_1)$ and $X(t_2)$ are negatively correlated. If $C_X(t_1, t_2) = 0$, then $X(t_1)$ and $X(t_2)$ are uncorrelated.

## Conditional normalisation

Reference: From `conduits` package vignette

Let $y_t$ is a variable observed at times $t = 1, ..., T$ and $\mathbf{z}_t = (z_{1,t}, ..., z_{p,t})$ be a $p$ dimensional vector of variables measured at the same time points. Assuming the expectation and the variance of $y_t$ are functions of $\mathbf{z}_t$, i.e., $\mathrm{E}(y_t|\mathbf{z}_t) = m(\mathbf{z}_t)$ and $\mathrm{V}(y_t|\mathbf{z}_t) = v(\mathbf{z}_t)$, we normalize $y_t$ conditional on $\mathbf{z}_t$ as,

$$y_t^* = \frac{y_t - \hat{m}(\mathbf{z}_t)}{\sqrt{\hat{v}(\mathbf{z}_t)}}$$

To estimate $m(\mathbf{z}_t)$ we fit Generalized Additive Models (GAMs) to $y_t$ using $\mathbf{z}_t$ as predictors. i.e., we fit the following model.

$$y_t = \alpha_0 + \sum_{i=1}^{p} f_i(z_{i,t}) + \varepsilon_t$$

where, $f_i(.)$ are smooth functions and $\varepsilon_1, \varepsilon_2, ..., \varepsilon_t$ have mean 0 and variance $v(\mathbf{z}_t)$. This gives,

$$\hat{m}(\mathbf{z}_t) = \hat{\alpha}_0 + \sum_{i=1}^{p} \hat{f}_i(z_{i,t}).$$

Next, to estimate $v(\mathbf{z}_t)$, we fit GAMs to the squared errors from the previous conditional mean model, i.e., $[y_t - \hat{m}(\mathbf{z}_t)]^2$, using $\mathbf{z}_t$ as predictors. i.e., we fit the following model.

$$[y_t - \hat{m}(\mathbf{z}_t)]^2 \sim \mathrm{Gamma}(v(\mathbf{z}_t), r),$$
$$\log(v(\mathbf{z}_t)) = \beta_0 + \sum_{i=1}^{p} g_i(z_{i,t}),$$

where, $g_i(.)$ are smooth functions. This gives,

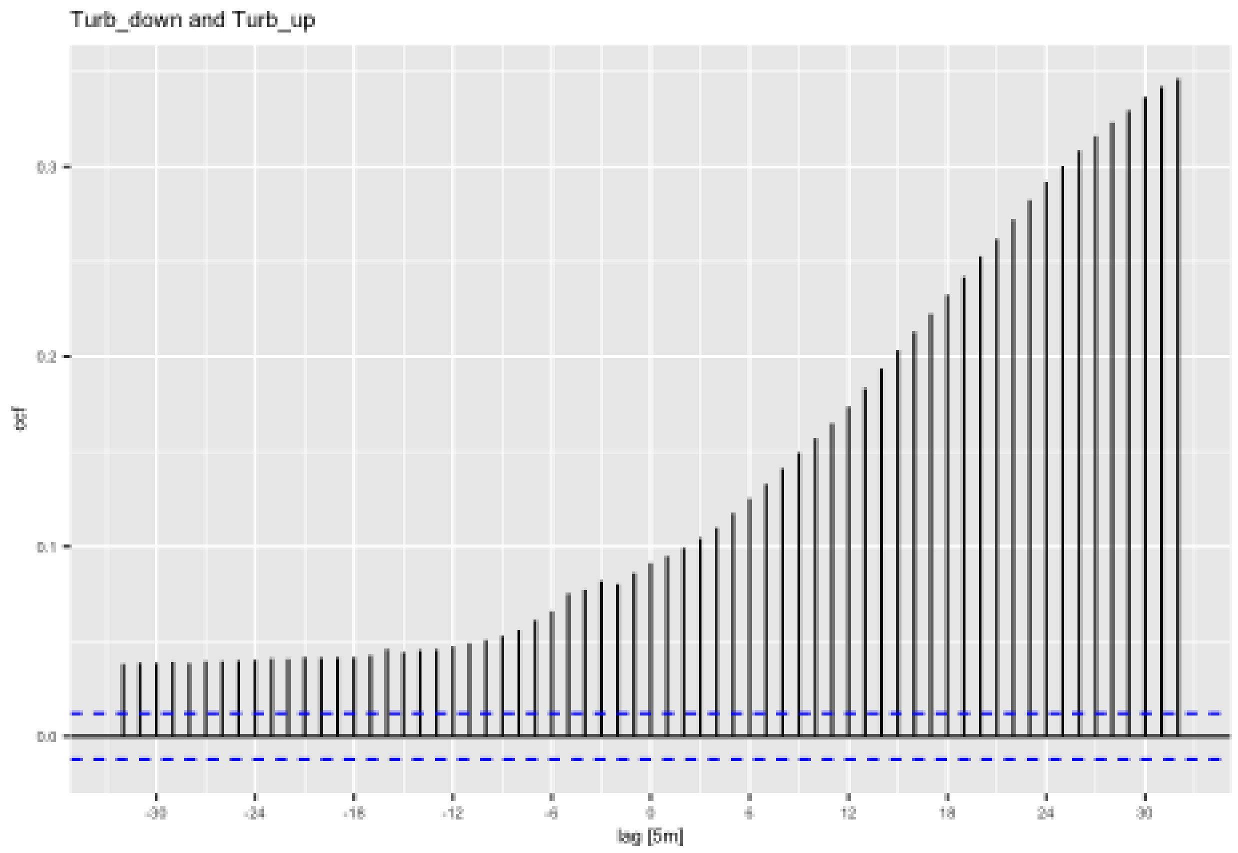
$$\hat{v}(\mathbf{z}_t) = \exp\left(\hat{\beta}_0 + \sum_{i=1}^{p} \hat{g}_i(z_{i,t})\right)$$

We use gam function from mgcv package to fit the GAMs in the above conditional mean and variance models. ### Conditional normalization of turbidity data from sensors in Pringle Creek upstream

Now we use the conditional_moments function to normalize turbidity from the upstream site in the Pringle Creek. We use water level, temperature and conductance measured at the same upstream site as predictors. Therefore the normalisation is done conditionally to these predictors.

**CCF of original data (for comparison)**

```
# the correlation between $x[t+k]$ and $y[t]$
c2 <- dataclean_full %>%
  as_tsibble(index= Timestamp) %>%
  CCF(turbidity_down, turbidity_up)
# keep upstream fixed
# Now I should slide downstream backward to find a
# match (Should take the maximum of Negative lags)
p2 <- c2 %>% autoplot() + ggtitle("Turb_down and Turb_up")+ theme(text = element_text(size=7))
print(p2)
```



```
c2 %>% filter(lag <=0)%>% slice_max(ccf)
```

```
## # A tsibble: 1 x 2 [5m]
##     lag    ccf
##   <lag>  <dbl>
## 1    0m 0.0913
```

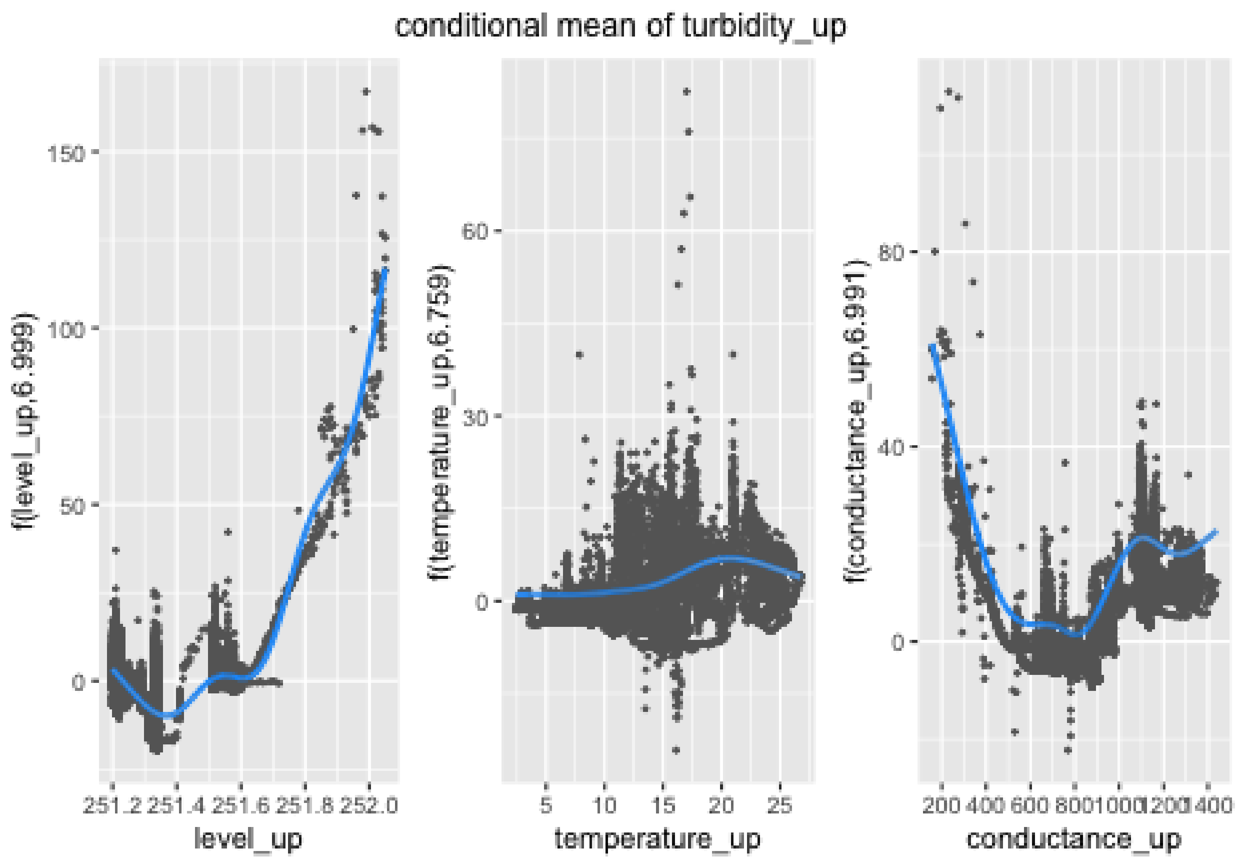
### conditional Normalization

```
turb_up_norm <- dataclean_full %>%
  conditional_moments(x =turbidity_up,
  z_numeric = c(level_up,
                temperature_up,
                conductance_up),
  knots_mean = c(8,8,8),
  knots_variance = c(7,7,7))
head(turb_up_norm$data_conditional_moments)
```

```
## # A tibble: 6 x 20
##   Timestamp           conductance_up DO_up pH_up turbidity_up chlorophyll_up
##   <dttm>                       <dbl> <dbl> <dbl>        <dbl>          <dbl>
## 1 2019-10-01 00:00:00            761  5.14  7.74         2.08           1.96
## 2 2019-10-01 00:05:00            761.  5.16  7.75         1.68           1.85
## 3 2019-10-01 00:10:00            761.  5.34  7.76         1.67           1.91
## 4 2019-10-01 00:15:00            760.  5.29  7.77         1.54           1.85
## 5 2019-10-01 00:20:00            761.  5.35  7.77         1.48           1.75
## 6 2019-10-01 00:25:00            761.  5.35  7.77         1.38           1.90
## # ... with 14 more variables: fDOM_up <dbl>, level_up <dbl>,
## #   temperature_up <dbl>, conductance_down <dbl>, DO_down <dbl>, pH_down <dbl>,
## #   chlorophyll_down <dbl>, fDOM_down <dbl>, level_down <dbl>,
## #   temperature_down <dbl>, turbidity_down <dbl>, turb_Dflag <fct>,
## #   E_turbidity_up <dbl>, Var_turbidity_up <dbl>
```

To visualize the fitted models for conditional means and variances, we can use the `autoplot` method written for `conditional_moments` functions.

```
autoplot(turb_up_norm, type = "mean")
```



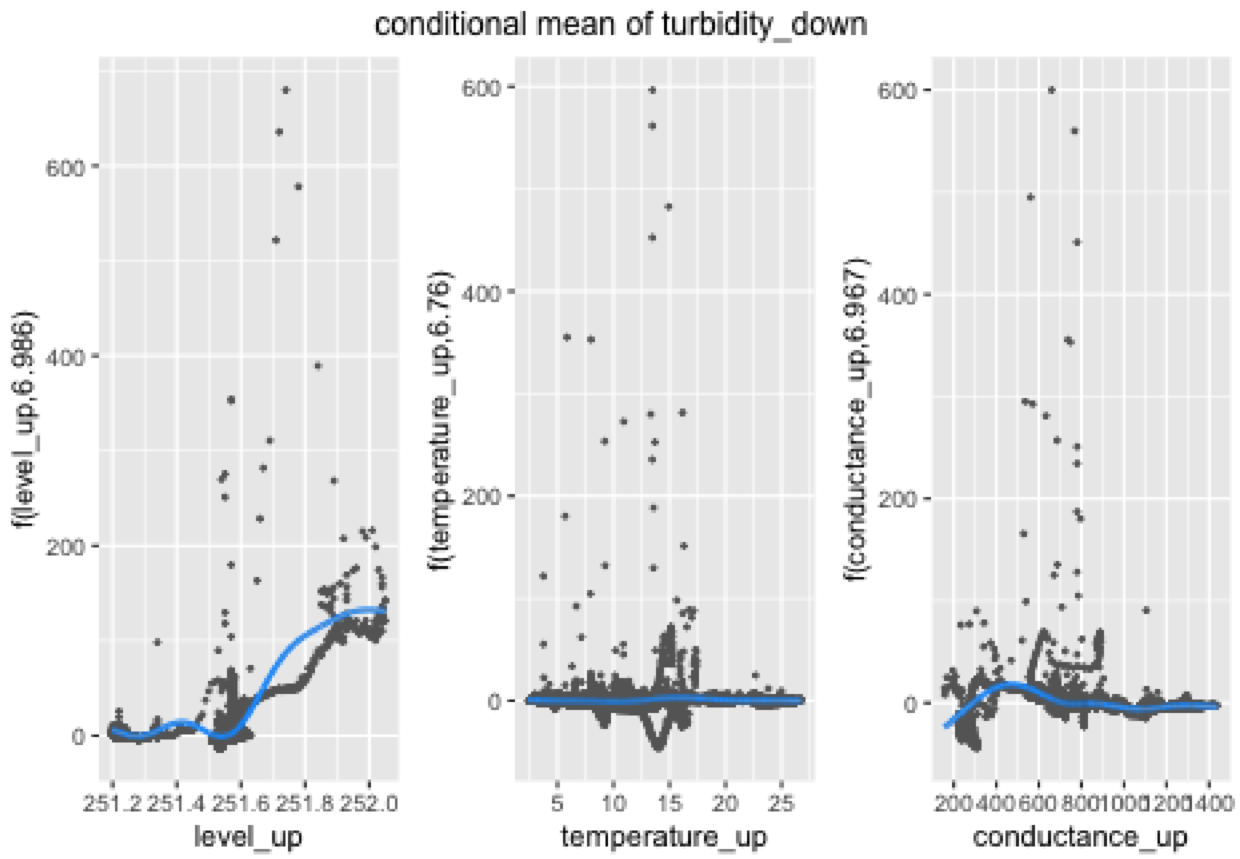
```
#autoplot(cond_moments_x, type = "variance")
```

**Conditional normalization of turbidity data from sensors in Pringle Creek downstream**

```
turb_down_norm <- dataclean_full %>%
  conditional_moments(x = turbidity_down,
                      z_numeric = c(level_up,
                                    temperature_up,
                                    conductance_up),
                      knots_mean = c(8,8,8),
                      knots_variance = c(7,7,7))
head(turb_down_norm$data_conditional_moments)
```

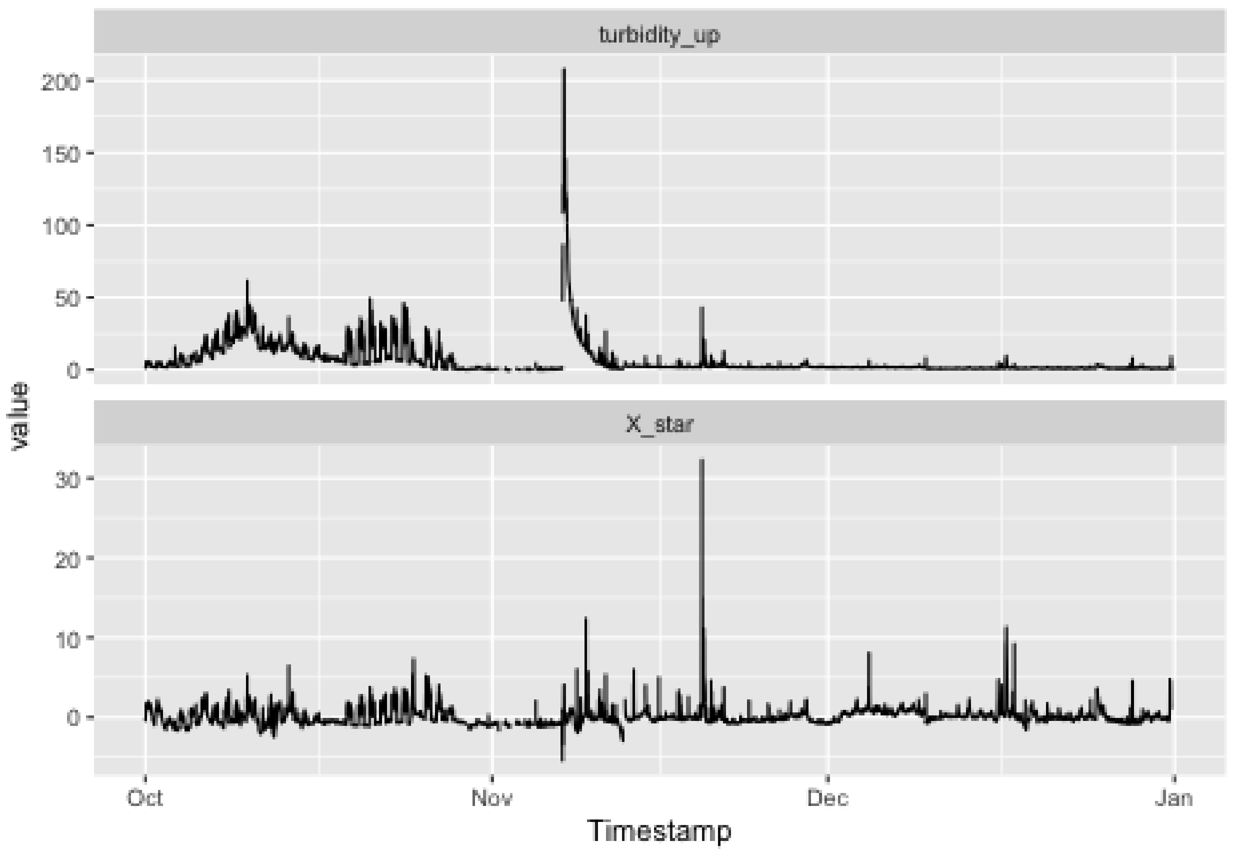
```
## # A tibble: 6 x 20
##   Timestamp           conductance_up DO_up pH_up turbidity_up chlorophyll_up
##   <dttm>                       <dbl> <dbl> <dbl>        <dbl>          <dbl>
## 1 2019-10-01 00:00:00           761   5.14  7.74         2.08           1.96
## 2 2019-10-01 00:05:00           761.  5.16  7.75         1.68           1.85
## 3 2019-10-01 00:10:00           761.  5.34  7.76         1.67           1.91
## 4 2019-10-01 00:15:00           760.  5.29  7.77         1.54           1.85
## 5 2019-10-01 00:20:00           761.  5.35  7.77         1.48           1.75
## 6 2019-10-01 00:25:00           761.  5.35  7.77         1.38           1.90
## # ... with 14 more variables: fDOM_up <dbl>, level_up <dbl>,
## #   temperature_up <dbl>, conductance_down <dbl>, DO_down <dbl>, pH_down <dbl>,
## #   chlorophyll_down <dbl>, fDOM_down <dbl>, level_down <dbl>,
## #   temperature_down <dbl>, turbidity_down <dbl>, turb_Dflag <fct>,
## #   E_turbidity_down <dbl>, Var_turbidity_down <dbl>
```

```
autoplot(turb_down_norm, type = "mean")
```

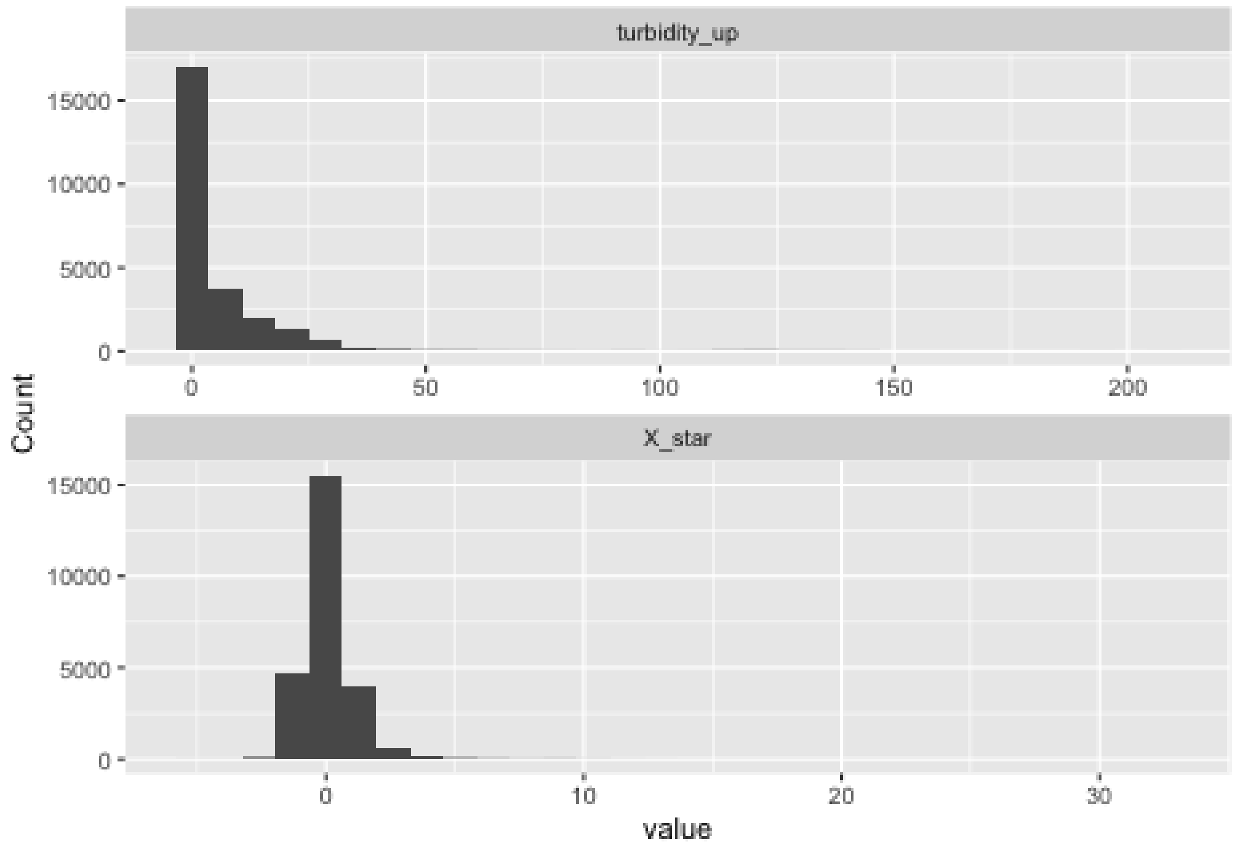


We now plot the conditionally normalised turbidity series.

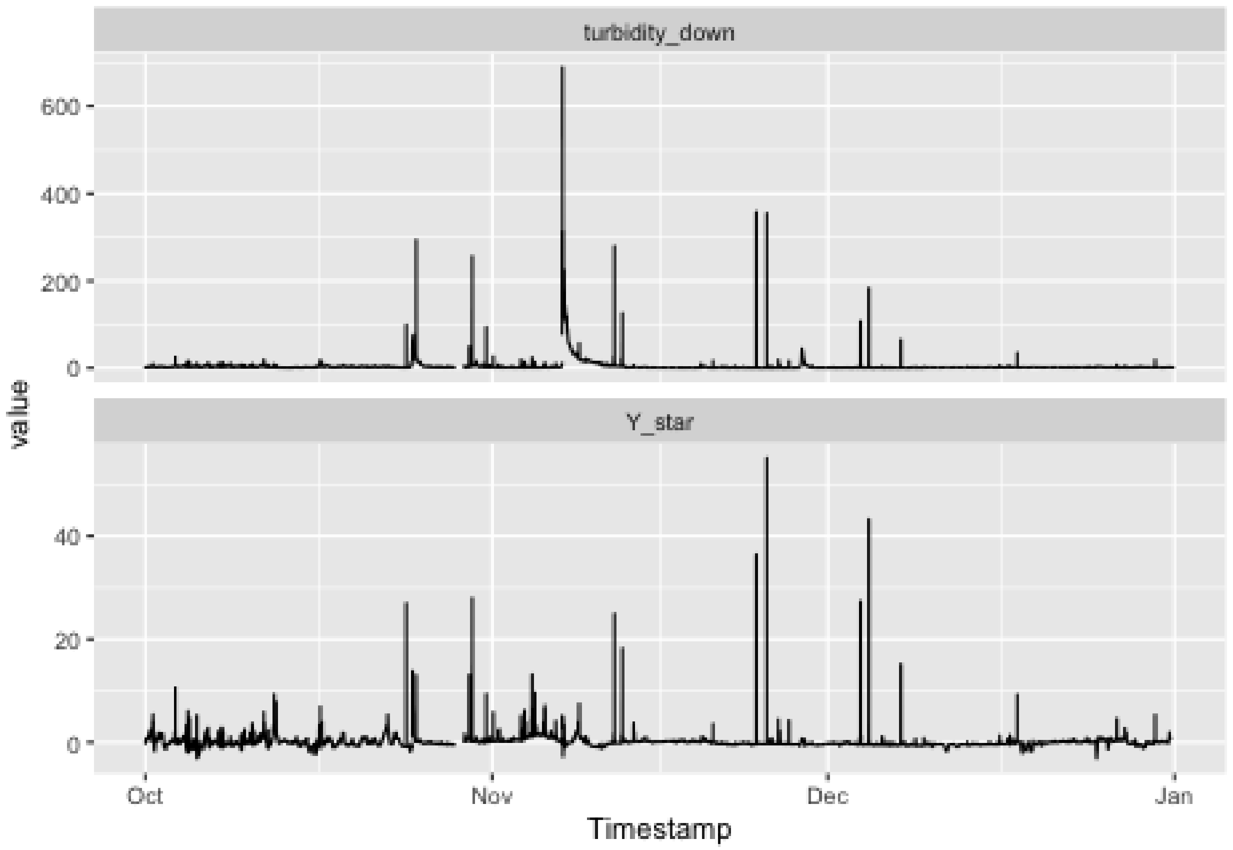
```
### Upstream data
normalised_x <- turb_up_norm$data_conditional_moments %>%
  mutate(X_star = (turbidity_up - E_turbidity_up)
         /sqrt(Var_turbidity_up)) %>%
  select(Timestamp, X_star, turbidity_up)

X_timeplot <-  normalised_x %>%
  pivot_longer(-Timestamp) %>%
  ggplot(aes(Timestamp, value)) +
  geom_line() +
  facet_wrap(~name, ncol = 1, scales = "free_y")
print(X_timeplot)
```



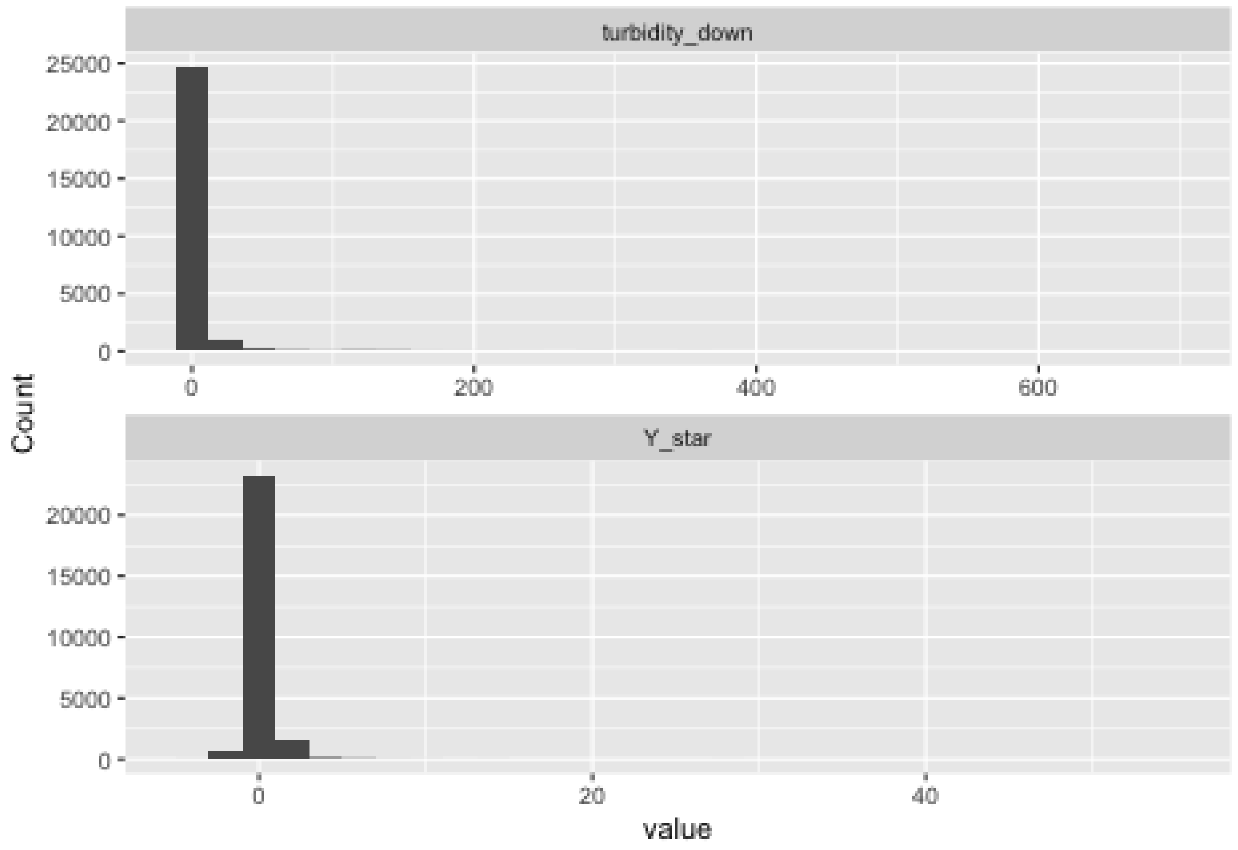
```r
X_hist <- normalised_x %>%
  pivot_longer(-Timestamp) %>%
  ggplot(aes(value)) +
  geom_histogram() +
  facet_wrap(~name, ncol = 1, scales = "free") +
  ylab("Count")
print(X_hist)
```



```r
### Downstream data
normalised_y <- turb_down_norm$data_conditional_moments %>%
  mutate(Y_star = (turbidity_down - E_turbidity_down)
         /sqrt(Var_turbidity_down)) %>%
  select(Timestamp, Y_star, turbidity_down)

Y_timeplot <-  normalised_y %>%
  pivot_longer(-Timestamp) %>%
  ggplot(aes(Timestamp, value)) +
  geom_line() +
  facet_wrap(~name, ncol = 1, scales = "free_y")
print(Y_timeplot)
```



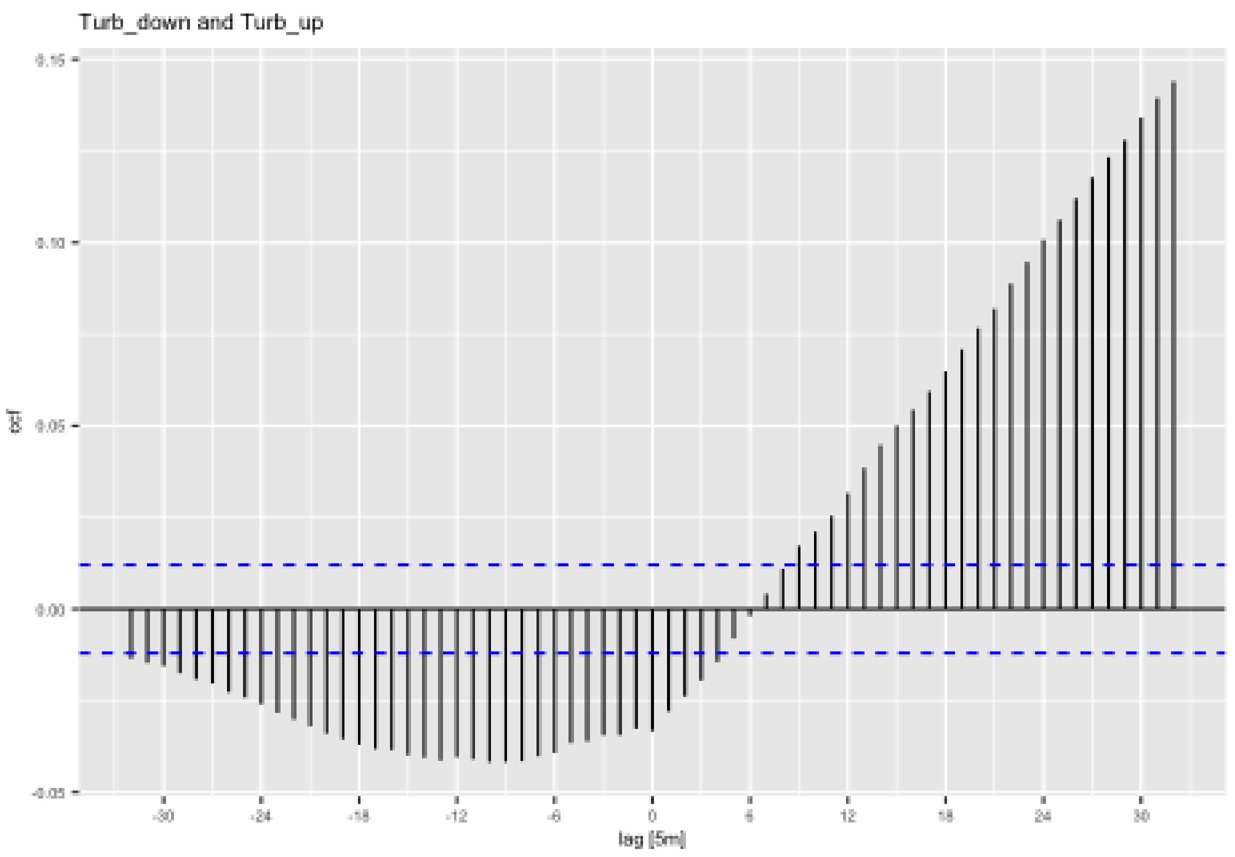
```
Y_hist <- normalised_y %>%
  pivot_longer(-Timestamp) %>%
  ggplot(aes(value)) +
  geom_histogram() +
  facet_wrap(~name, ncol = 1, scales = "free") +
  ylab("Count")
print(Y_hist)
```



**Cross Correlation between normalized series turbidity upstream and turbidity downstream**

Note: in `stat` package the lag $k$ value returned by $ccf(x, y)$ estimates the correlation between $x[t + k]$ and $y[t]$. The function `feasts::CCF()` improves the `stats::ccf()` function.

```r
normalized_data <- left_join(normalised_x, normalised_y, by = "Timestamp") %>%
  as_tsibble(index = Timestamp)

# Xstar - turb_upstream
# Ystar -  turb downstream
# the correlation between $x[t+k]$ and $y[t]$
c1 <- normalized_data %>%
  CCF( Y_star, X_star) # keep upstream fixed

# Now I should slide downstream backward to find a
# match (Should take the maximum of Negative lags)


p1 <- c1 %>% autoplot() +
  ggtitle("Turb_down and Turb_up")+
  theme(text = element_text(size=7))
print(p1)
```



```
c1 %>% filter(lag <=0)%>% slice_max(abs(ccf))
```

```
## # A tsibble: 1 x 2 [5m]
##     lag     ccf
##   <lag>   <dbl>
## 1  -50m -0.0418
```

**Calculate Time delay, dt between upstream and downstream**

```
dt <- c1 %>% filter(lag <=0)%>% slice_max(abs(ccf)) %>% first()
dt
```

```
## <lag[1]>
## [1] -50m
```

```
lags <- as.numeric(abs(dt)) #frequency is five
lags
```

```
## [1] 10
```

In Pringle Creek, the two locations are situated about 200 m apart where a small tributary entering to the main creek between the two sensors.

According to the NEON (National Ecological Observatory Network) field experts, the approximate time it takes water to travel from upstream to downstream locations at Pringle Creek is typically about 45 – 60 minutes. The time will be shorter than 45 minutes during high flows and greater than 60 minutes during low flows.

**The calculated $d_t$ is consistent with the actual conditions.**

However, the above $d_t$ calculation gives us an **average time delay** between the two sensor locations. The time delay can vary in a wide range due to many seasonal and environmental factors throughout the period. For example, at one extreme, during summer, surface flow tends to cease, creating disconnected pools of surface water along the channel. (however, we did not include summer data for our analysis)

### Conditional Cross Correlation between normalized series turbidity upstream and turbidity downstream

**Our objective is to allow the time delay to vary for different periods of the year**

Considering the information about the actual time delay (45 to 60 minutes delay), we choose to compute the cross-correlations up to 60 lags, which will allow for a maximum of five hours of travel time, given the frequency of the water-quality data we analyse is 5 minutes.

We compute conditional cross-correlations between turbidity at two sites conditional on upstream variables.

First we fit GAM models for the normalised $x_t y_{t+k}$ with upstream variables at lags $k = 1, ..., 60$.

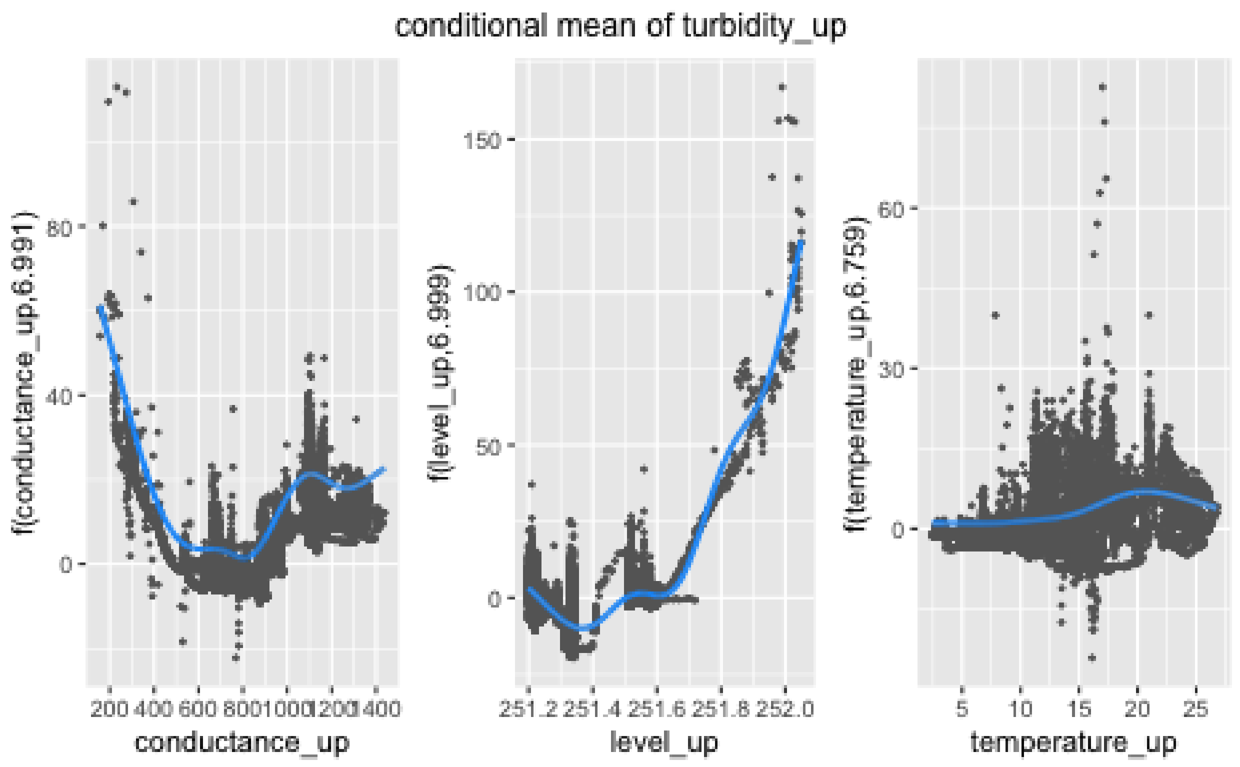
We fit the following three different models at lags $k = 1, ..., 60$.

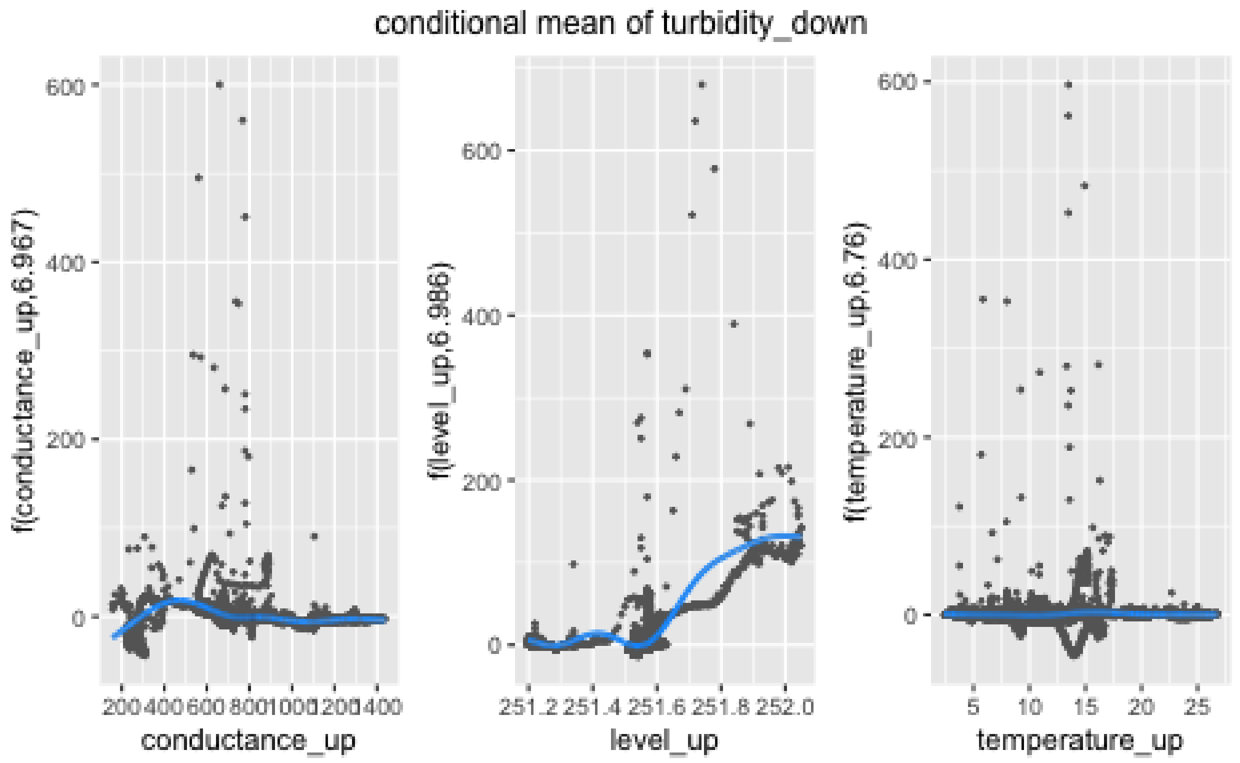
$$x_t^* y_{t+k}^* \sim s(level\_up_t) + s(conductance\_up_t) + s(temperature\_up_t) + \epsilon_t$$

**Setting** $k_{max} = 60$

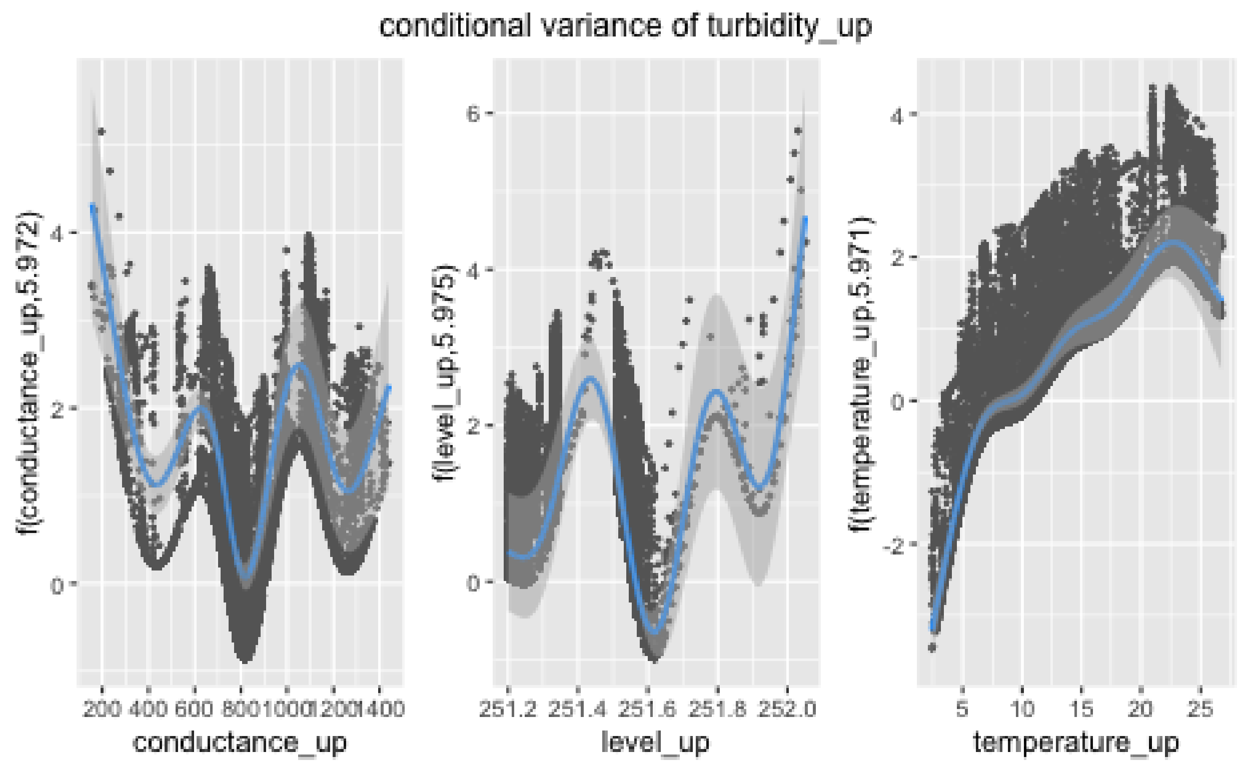
```r
# This function computes cross correlation between
# $x_t$ and $y_t+k$ at $k = 1,2,...$ conditional on a set of
# time series $z_t$
cond_ccf <- dataclean_full %>%
  conduits::conditional_ccf(
    x = turbidity_up,
    y = turbidity_down,
    z_numeric = c(conductance_up,
                  level_up,
                  temperature_up),
    k = 1:60,
    knots_mean = list(x = c(8,8,8),
                      y = c(8,8,8)),
    knots_variance = list(x = c(7,7,7),
                          y = c(7,7,7)),
    df_correlation = c(4,2,4))
```

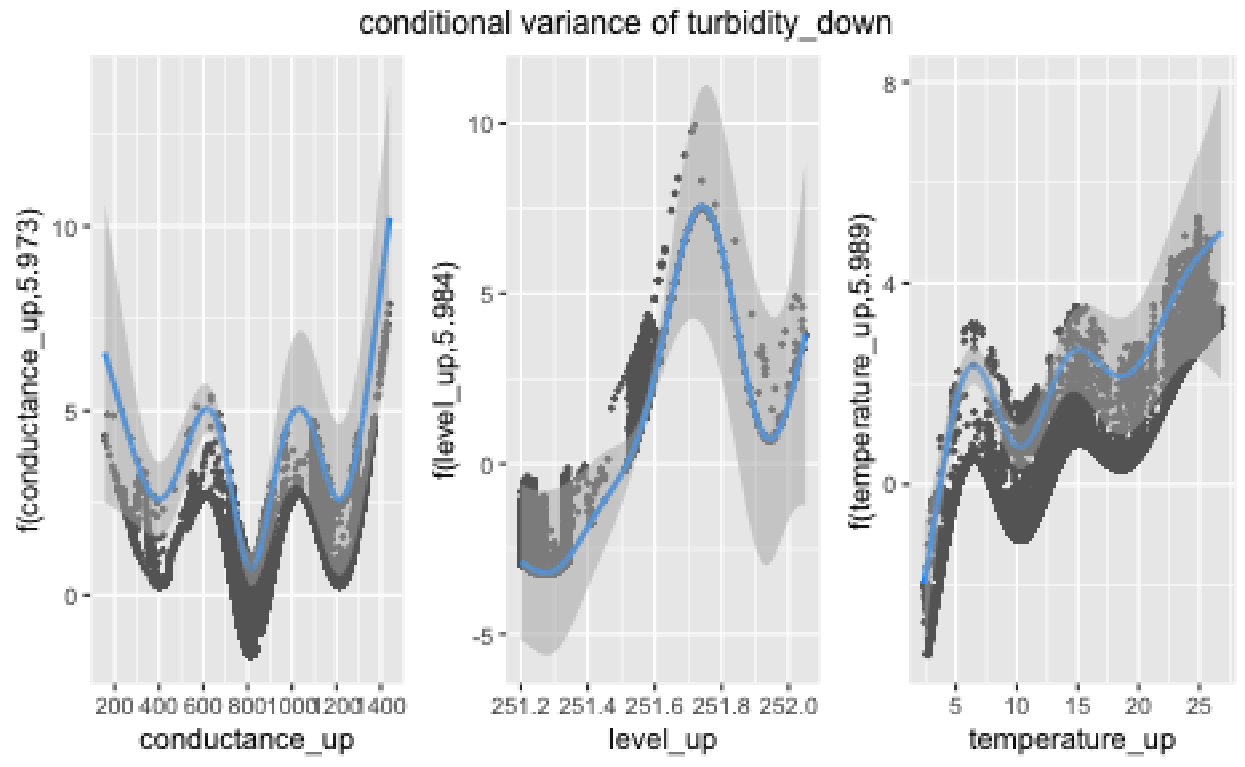
```r
# returns plots visualising fitted gam models for
# conditional mean or variance vs each predictor.
autoplot(cond_ccf, type = "mean" )
```





```
autoplot(cond_ccf, type = "variance" )
```





**Estimating the time delay, $d_t$, from the conditional cross-correlations**

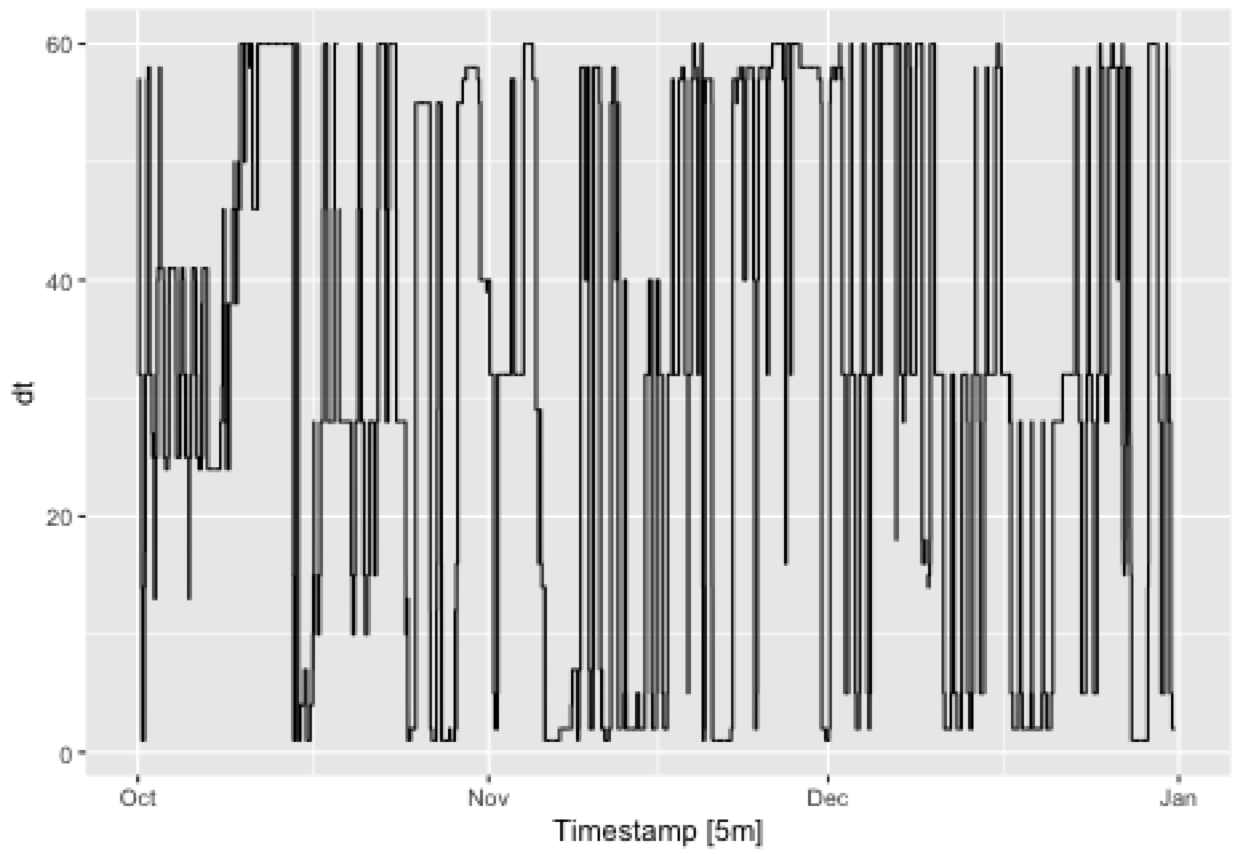
We estimate the time delay between sensors, $d_t$, as the *lag that gives maximum cross-correlation conditional on upstream variables*

Now let us visualize $d_t$ with each covariate at upstream site.

```
Estimate_dt <- cond_ccf %>%
  estimate_dt(new_data = dataclean_full, k_min = 1, k_max = 60)


summary(Estimate_dt$data_dt$dt)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00    5.00   32.00   32.42   58.00   60.00     315
```

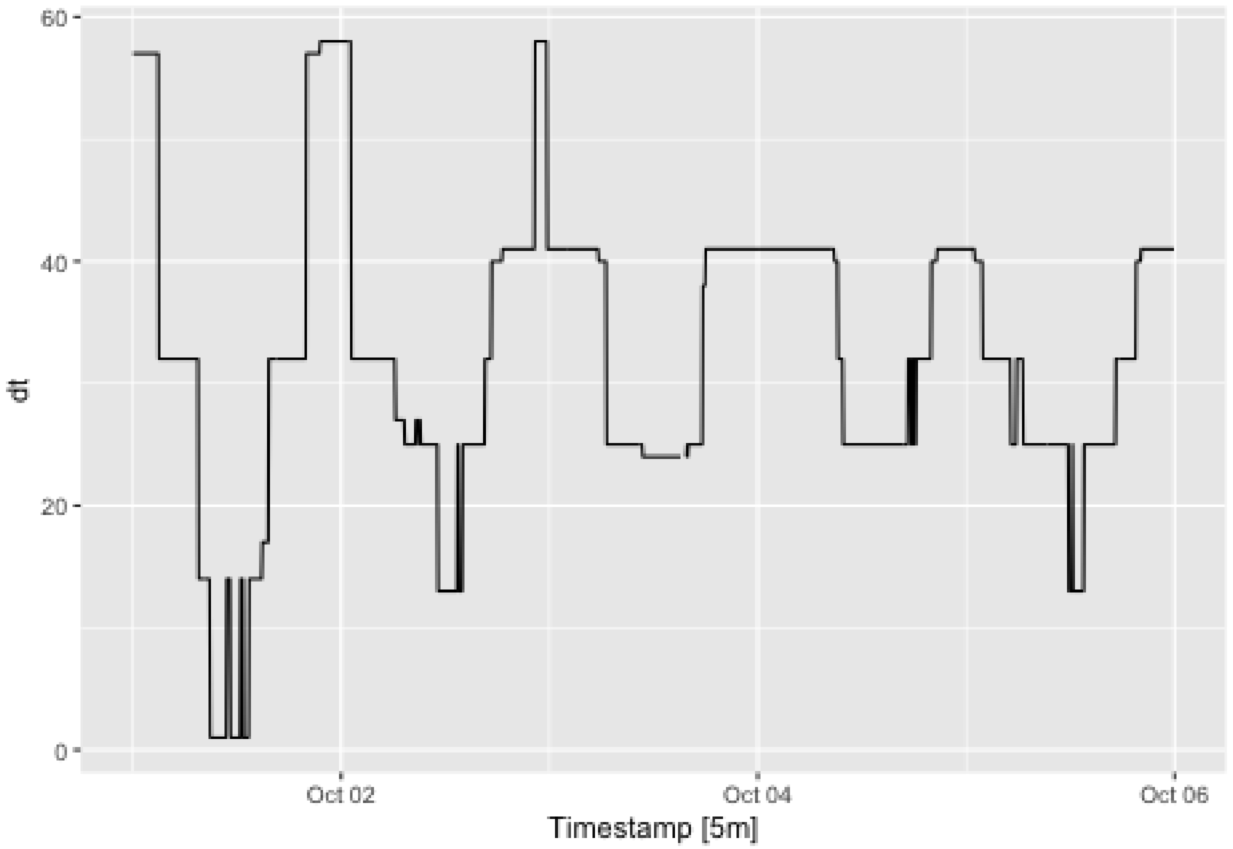
```
dt_table <- Estimate_dt$data_dt %>%
  as_tsibble(index = Timestamp)

p <- dt_table %>%
  select(Timestamp, dt) %>% autoplot()
p
```



**Zoom in (Five days period)**

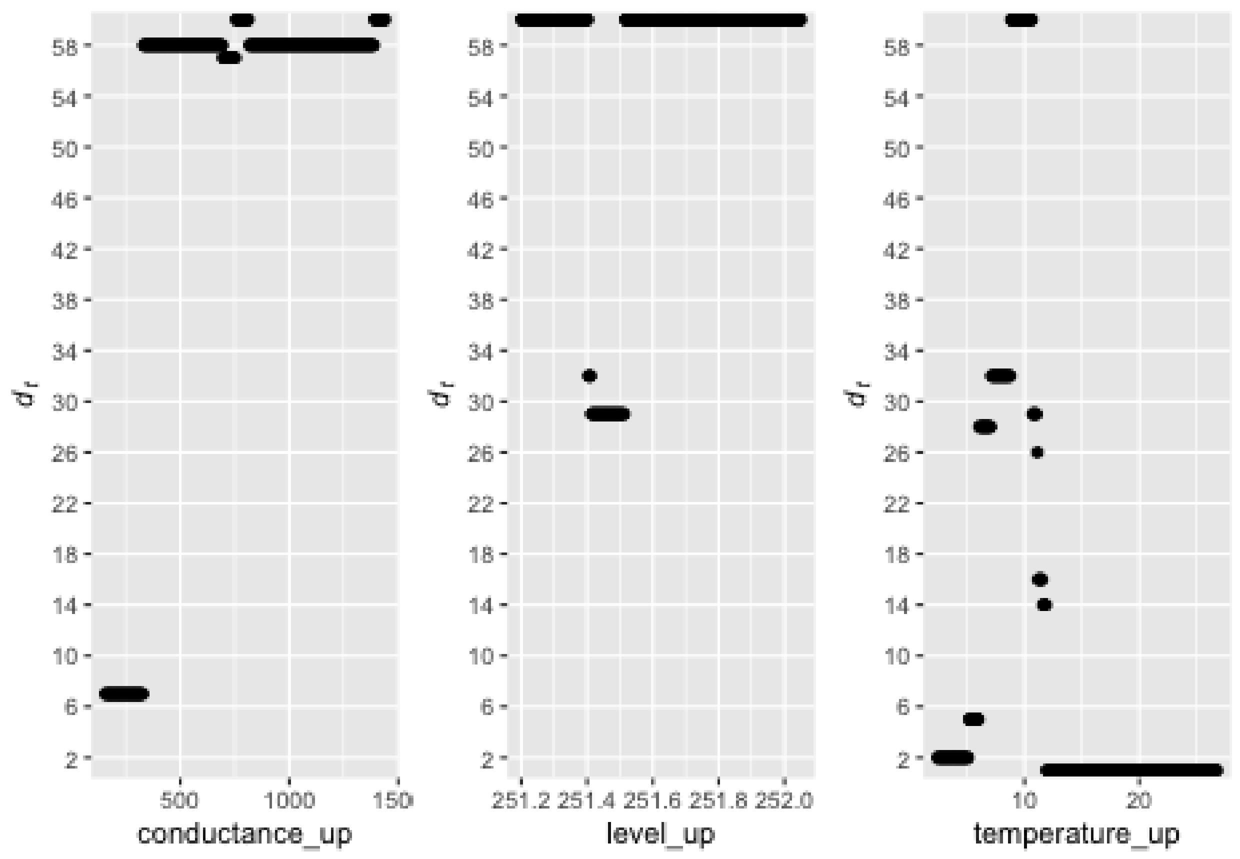
```
dt_table_week <- Estimate_dt$data_dt %>%
  slice(1:1440) %>%  # five days period
  as_tsibble(index = Timestamp)
```

```r
p <- dt_table_week %>%
  select(Timestamp, dt) %>% autoplot()
p
```



We can also visualize the estimated $d_t$ with each predictor used in the conditional cross-correlation models. For that, we can use `autoplot` method written for the `estimate_dt` function as follows. Setting `interval = TRUE` will compute prediction intervals that were obtained using the Sieve bootstrap method which will take more computational time.

```r
# set interval=TRUE to compute prediction intervals
# (This will take more computational time)
autoplot(object = Estimate_dt, interval = FALSE)
```



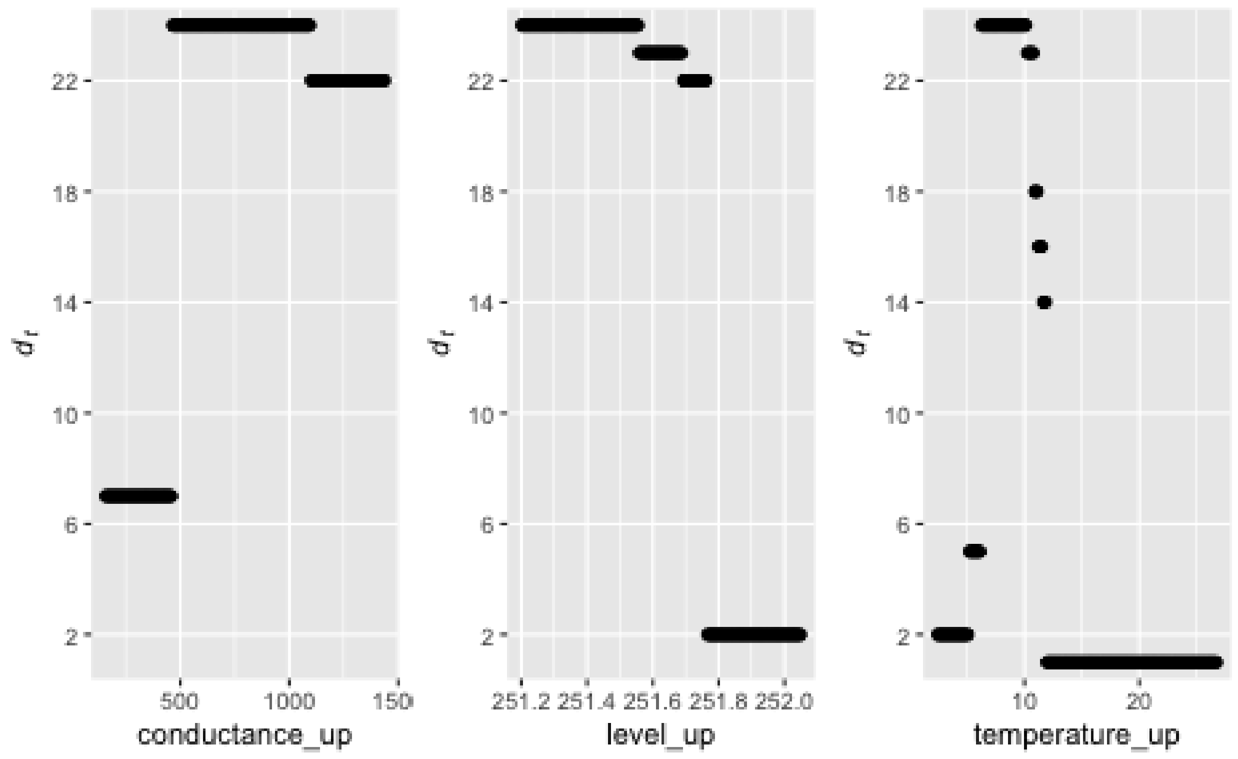
**Setting** $k_{max} = 24$

```
kmax <-  24

cond_ccf <- dataclean_full %>%
  conduits::conditional_ccf(
    x = turbidity_up,
    y = turbidity_down,
    z_numeric = c(conductance_up,
                  level_up,
                  temperature_up),
    k = 1:kmax,
    knots_mean = list(x = c(8,8,8),
                      y = c(8,8,8)),
    knots_variance = list(x = c(7,7,7),
                          y = c(7,7,7)),
    df_correlation = c(4,2,4))


#autoplot(cond_ccf_c_24, type = "mean" )

# autoplot(cond_ccf_c_24, type = "variance" )

Estimate_dt <- cond_ccf %>%
  estimate_dt(new_data = dataclean_full, k_min = 1, k_max = kmax)
```

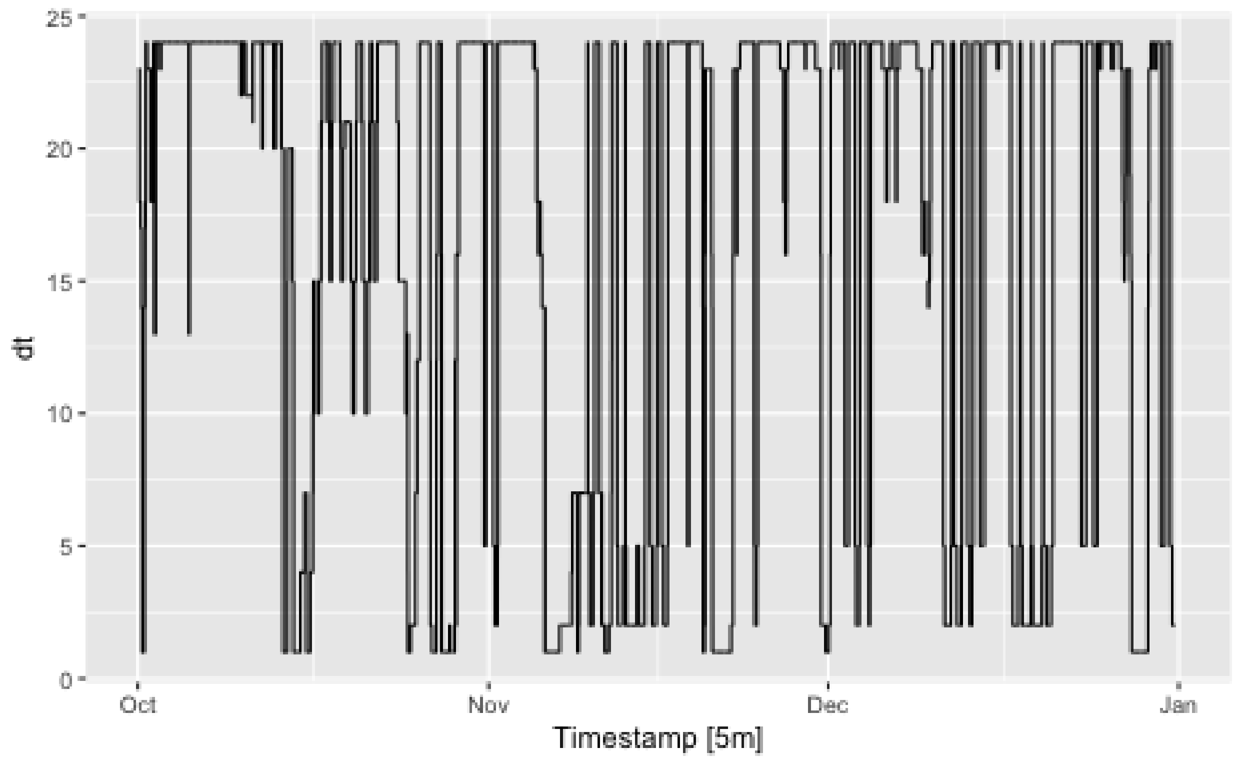
```r
summary(Estimate_dt$data_dt$dt)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00    5.00   24.00   16.71   24.00   24.00     315
```

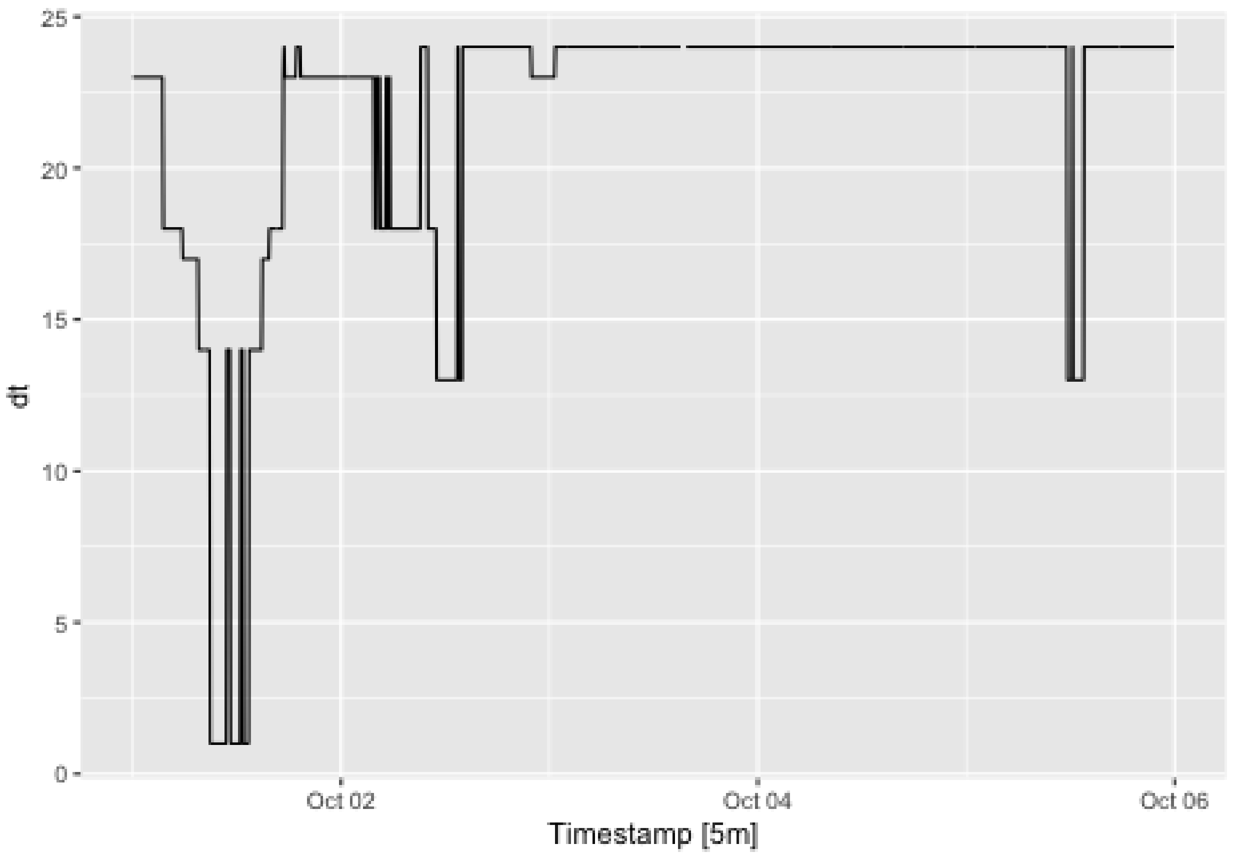
```r
autoplot(object = Estimate_dt, interval = FALSE)
```



```r
p <- Estimate_dt$data_dt %>%
  as_tsibble(index = Timestamp)%>%
  select(Timestamp, dt) %>% autoplot()
print(p)
```



```
dt_table_week <- Estimate_dt$data_dt %>%
  slice(1:1440) %>%  # five days period
  as_tsibble(index = Timestamp)

p <- dt_table_week %>%
  select(Timestamp, dt) %>% autoplot()
p
```


25
20
15
10
5
0
dt
Oct 02
Oct 04
Oct 06
Timestamp [5m]