Temporal autocorrelation

Eddie Wu

2024-07-03

```
library(dplyr)
library(tidyverse)
library(lme4)
library(nlme)
library(xts)
library(ModelMetrics)
library(zoo)
library(lubridate)
library(ggpubr)
library(performance) #for easy diagnostic plotting
library(imputeTS) #for check NAs in times series
```

```
cal.rmse <- function(out.list, fold) {

  # data frame to store the results
  r <- matrix(NA, nrow=fold, ncol=length(unique(out.list[[1]]$location)))
  colnames(r) <- unique(out.list[[1]]$location)

  # 10 iterations
  for (i in 1:fold){
   compare <- out.list[[i]]

   # calculate rmse
   for (loc in unique(compare$location)) {
     compare.now <- subset(compare, location == loc) %>% na.omit()
     r[i,loc] <- rmse(compare.now$obs, compare.now$preds)
    }
  }
  return(r)
}
cal.nsc <- function(out.list, fold) {

  # dataframe to store the results
  r <- matrix(NA, nrow=fold, ncol=length(unique(out.list[[1]]$location)))
  colnames(r) <- unique(out.list[[1]]$location)

  # 10 iterations
  for (i in 1:fold){
   compare <- out.list[[i]]

   # calculate nsc
   for (loc in unique(compare$location)) {
     compare.now <- subset(compare, location == loc) %>% na.omit()
```

```
    nsc <- 1 - sum((compare.now$obs - compare.now$preds)^2) / sum((compare.now$obs - mean(compare.now$
    r[i,loc] <- nsc
   }
  }
  return(r)
}
```

Data importing and cleaning

```
## Data import
airtemp <- read.csv("tributary air temperatures clean.csv")
watertemp <- read.csv("tributary water temperature/water_temperature_d.csv")
flow <- read.csv("tributary discharge.csv")

# Convert to date format
airtemp$date <- as.Date(airtemp$date, format = "%m/%d/%Y")
watertemp$date <- as.Date(watertemp$date, format = "%m/%d/%Y")
flow$date <- as.Date(flow$date, format = "%m/%d/%Y")


table(airtemp$station_name)
```

```
##
##               CAMERON FALLS (AUT)                          CLOQUET
##                              4324                             1461
##                         DELHI CS                       DELHI CS_PD
##                              4749                             1827
##               ELYRIA LORAIN CO AP                  GORE BAY CLIMATE
##                              1096                             1637
##               GREEN BAY BOTANICAL                  INDIANA DUNES NP
##                              1461                              731
##                  LONG POINT (AUT)                         MONETVILLE
##                              1461                             1462
##                ROCHESTER GTR INTL                         SAGINAW #3
##                              1096                             1096
##                TILLSONBURG NORTH TORONTO LESTER B. PEARSON INT'L A
##                              1096                             4912
```

```
table(watertemp$station_name)
```

```
##
##                                          CAMERON FALLS (AUT)
##                            1095                             3923
##                          CLOQUET                          DELHI CS
##                            1349                             4554
##                       DELHI CS_PD                ELYRIA LORAIN CO AP
##                            1825                             1095
##                  GORE BAY CLIMATE                GREEN BAY BOTANICAL
##                            1879                             1374
##                  INDIANA DUNES NP                   LONG POINT (AUT)
##                             730                             1883
##                        MONETVILLE                 ROCHESTER GTR INTL
##                            1246                             1095
##                        SAGINAW #3                  TILLSONBURG NORTH
```

```
##                              1095                                  919
## TORONTO LESTER B. PEARSON INT'L A
##                              5012
```

```
table(watertemp$location)
```

```
##
##              bigcreek   bigotter        fox    genesee     humber  longpoint
##       1095       4554        919       1374       1095       5012       1883
## mississagi    nipigon    portage  portdover    saginaw      still    stlouis
##       1879       3923        730       1825       1095       1246       1349
##  vermilion
##       1095
```

```
table(flow$location)
```

```
##
##   bigcreek   bigotter        fox    genesee     humber mississagi    nipigon
##       4749       1096       1461       1096       5113       1826       1096
##    portage    saginaw      still    stlouis  vermilion
##        731       1096       1462       1461       1096
```

```
## Calculate the MEAN airT for US locations
for (i in which(is.na(airtemp$mean_temp))) {
  airtemp$mean_temp <- (airtemp$max_temp + airtemp$min_temp) / 2
}
```

Check for imputed values

Use a 7-days rolling mean to check for possibly imputed temperatures. If the variance of a certain day's temperature is very close to zero, then it is likely that this particular data is imputed.

```
# make sure that no initial NA values in watertemp
which(is.na(watertemp$location))
```

```
## integer(0)
```

```
# Need to calculate the rolling mean for each location separately...
watertemp$location <- as.factor(watertemp$location)
unique_locations <- unique(watertemp$location)

for(loc in unique_locations) {
  sub <- watertemp[watertemp$location == loc,]
  sub$rolling_mean <- rollmean(sub$temp, k = 7, fill = NA, align = "right")
  watertemp[watertemp$location == loc, "rolling_mean"] <- sub$rolling_mean
}

# Calculate variance
watertemp$variance <- (watertemp$temp - watertemp$rolling_mean)^2

# Assign 999 when variance is very small
watertemp$temp[which(watertemp$variance < 1e-10)] <- NA
```

Data cleaning and combining

Now we want to combine airT, waterT, and flow into a master dataframe

```r
# Combine water temperature and discharge
water <- left_join(watertemp, flow, by = c("location","date"))

master.temp <- left_join(airtemp, water, by = c("station_name", "date")) %>%
  select(location, country, station_name, date, year, month = month.x,
         day = day.x, airT = mean_temp, waterT = temp, flow) %>%
  na.omit() %>%
  arrange(location, date) # arrange by location and date


## Check for duplicates
duplicates <- master.temp %>%
  group_by(location, date) %>%
  filter(n() > 1) # Duplicates should be NA...


## Check the time range for each location
time_range_per_location <- master.temp %>%
  group_by(location) %>%
  summarize(
    start_date = min(date),
    end_date = max(date)
  )

# Print the result
print(time_range_per_location)
```

```
## # A tibble: 12 x 3
##    location   start_date end_date
##    <chr>      <date>     <date>
##  1 bigcreek   2000-04-04 2014-09-24
##  2 bigotter   2012-03-21 2014-09-25
##  3 fox        2011-03-04 2014-12-31
##  4 genesee    2011-01-01 2013-12-31
##  5 humber     1998-04-09 2013-06-13
##  6 mississagi 2010-07-09 2014-06-12
##  7 nipigon    2008-01-01 2010-04-28
##  8 portage    2011-01-01 2012-12-30
##  9 saginaw    2012-01-01 2014-12-31
## 10 still      2002-05-02 2008-09-28
## 11 stlouis    2011-04-22 2014-12-31
## 12 vermilion  2012-01-01 2014-12-31
```

Get three years

We try to subset three years of data for each location, as closely possible as to 2014.

```r
## Get only three years for each location
start_end <- read.csv("start_end.csv")

# Pivot to long
start_end_long <- start_end %>%
  pivot_longer(cols = starts_with("year"),
               names_to = "rank", names_prefix = "year",
               values_to = "year") %>%
```

```
  filter(year != 0) # still river only has two years

# Semi-combine
master.temp <- master.temp %>%
  semi_join(start_end_long, by = c("location", "year"))

table(master.temp$location)
```

```
##
##   bigcreek   bigotter        fox    genesee     humber mississagi    nipigon
##       1066        914       1085       1089       1095        989        844
##    portage    saginaw      still    stlouis  vermilion
##        727       1013        731        951        954
```

```
## Check for complete records
# Generate a complete sequence of dates for each location/year
complete_dates <- master.temp %>%
  group_by(location, year) %>%
  summarize(start_date = min(date), end_date = max(date),
            .groups = "drop") %>%
  rowwise() %>%
  mutate(all_dates = list(seq.Date(from = start_date, to = end_date,
                                   by = "day"))) %>%
  select(location, all_dates) %>%
  unnest(all_dates)

# Join with the original dataframe to find missing dates and fill with NAs
master.temp.c <- complete_dates %>%
  left_join(master.temp, by = c("location", "all_dates" = "date")) %>%
  rename(date = all_dates)


table(master.temp.c$location)
```

```
##
##   bigcreek   bigotter        fox    genesee     humber mississagi    nipigon
##       1095        919       1095       1095       1096       1096        849
##    portage    saginaw      still    stlouis  vermilion
##        730       1096        903        981       1095
```

Get lagged days

```
## Get the time lag day variables
master.temp.c <- master.temp.c %>%
  group_by(location) %>%
  mutate(airT.lag1 = lag(airT, 1),
         airT.lag2 = lag(airT, 2),
         airT.lag3 = lag(airT, 3),
         airT.lag4 = lag(airT, 4),
         airT.lag5 = lag(airT, 5))


## Change the location into factors
master.temp.c$location <- as.factor(master.temp.c$location)
```

```r
table(master.temp.c$location)
```

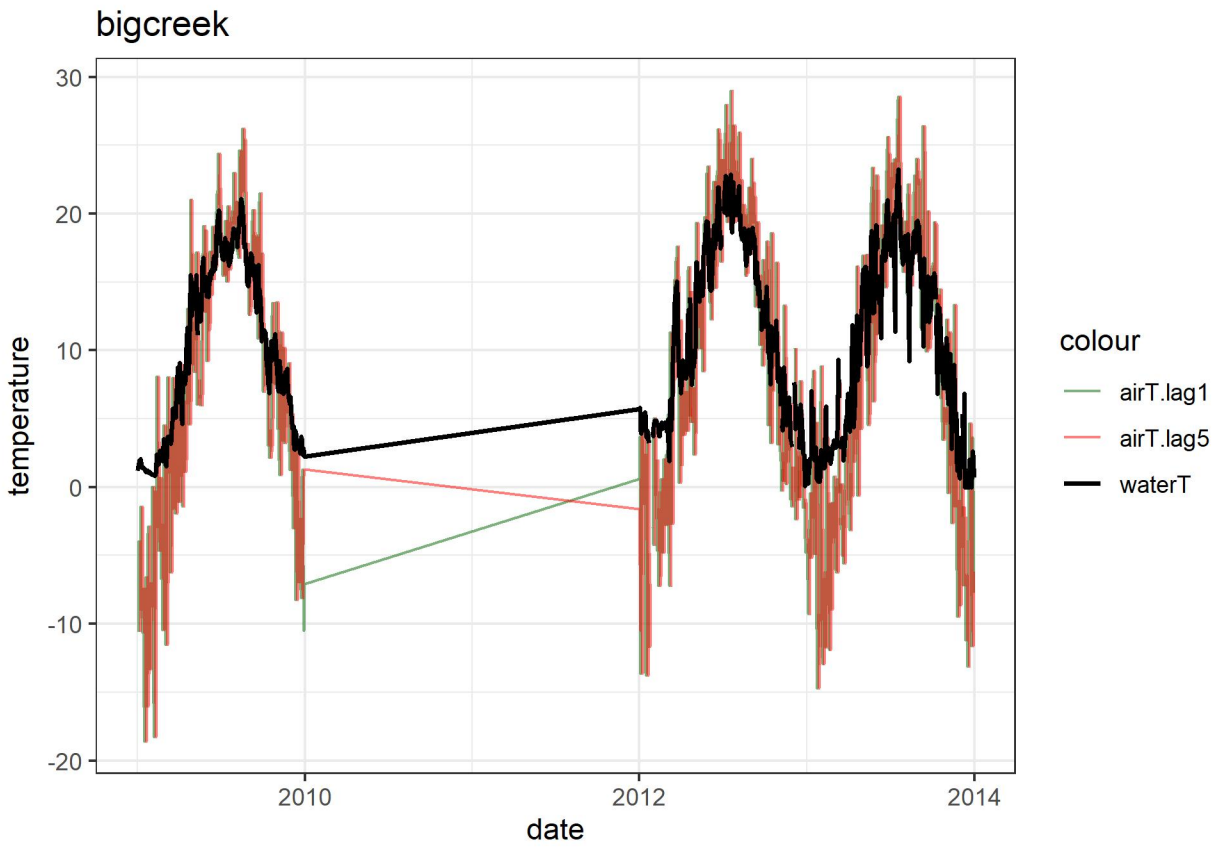
```
##
##   bigcreek   bigotter        fox    genesee     humber mississagi    nipigon
##       1095        919       1095       1095       1096       1096        849
##    portage    saginaw      still    stlouis  vermilion
##        730       1096        903        981       1095
```

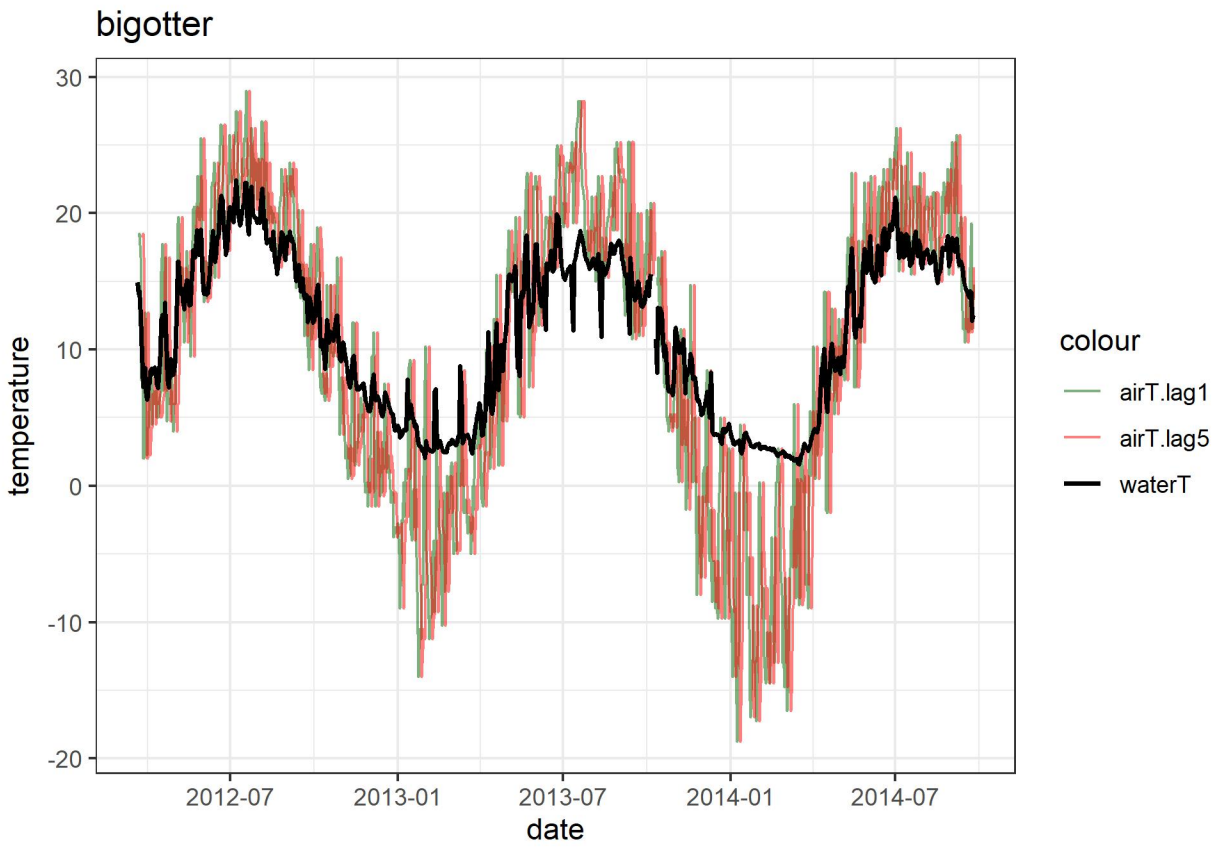
```r
master.temp.c$country <- as.factor(master.temp.c$country)

## Remove winter data:
#master.temp <- master.temp %>% filter(month != 12 & month != 1 & month != 2)
```

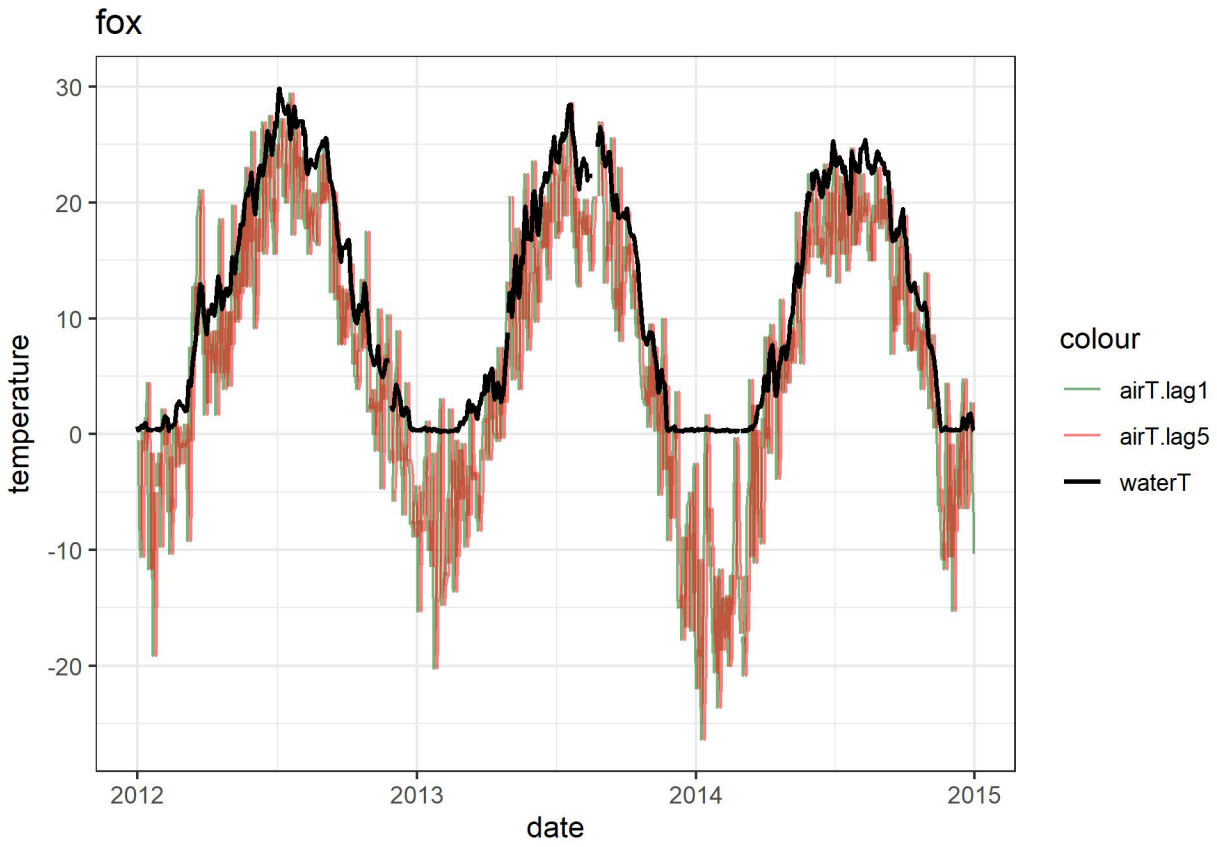
Data plots

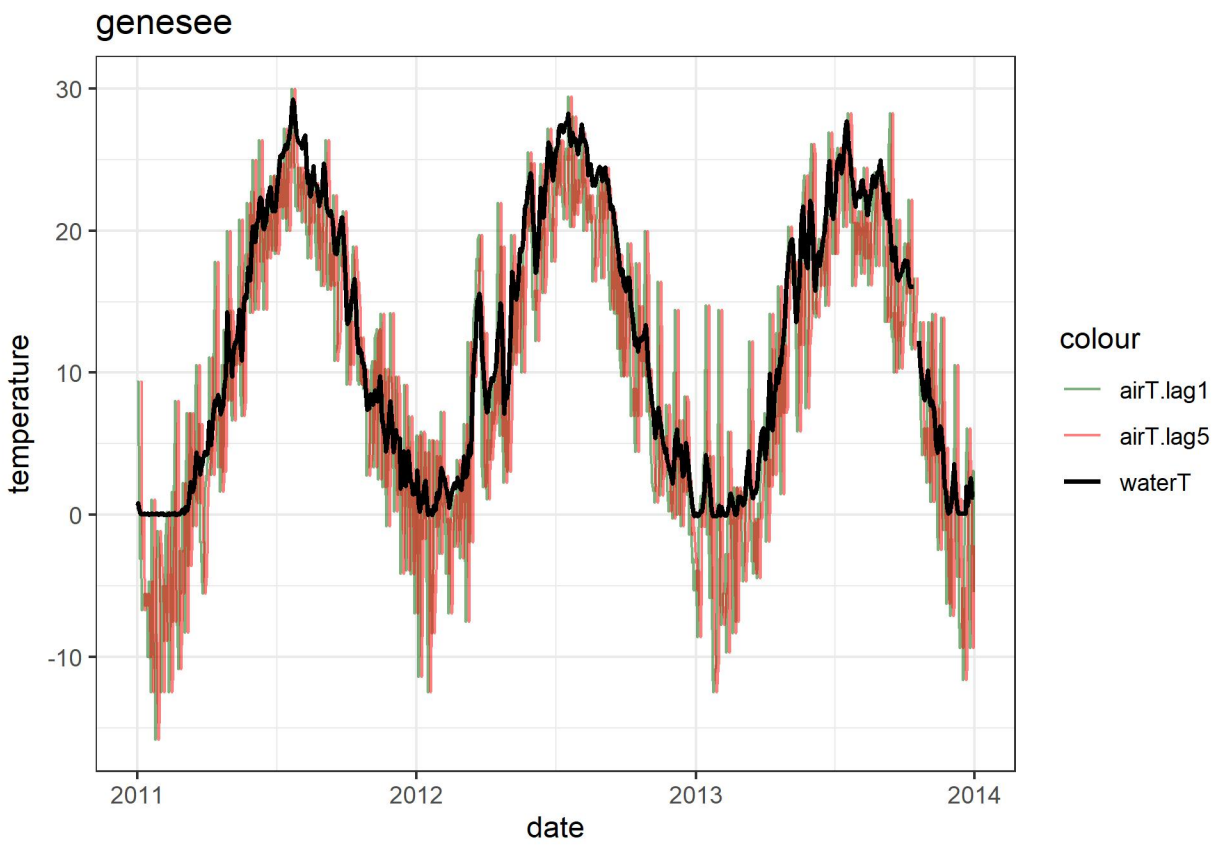
Before we run any analysis, we first want to check whether the air temperature and lags are corrected calculated.

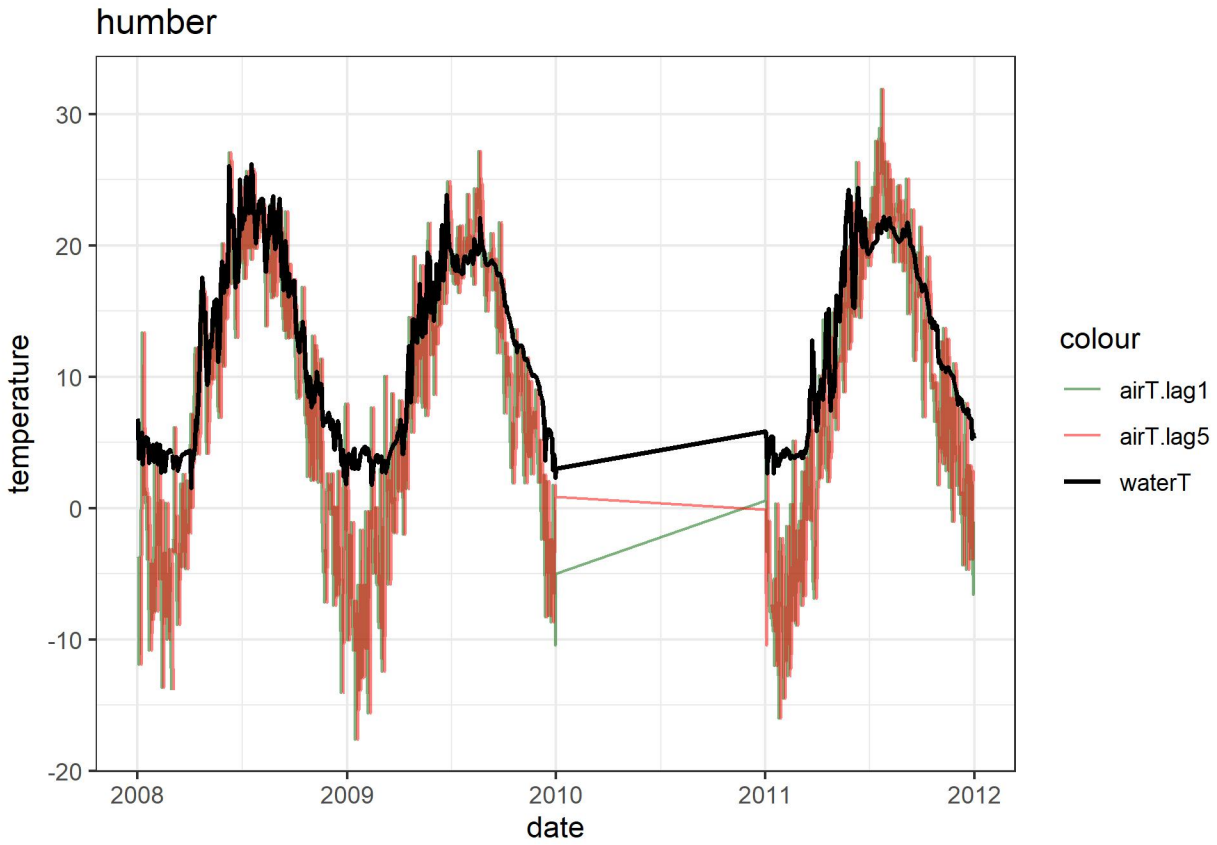
```r
colors <- c("airT.lag1" = "darkgreen", "airT.lag5" = "red", "waterT" = "black")

for (loc in levels(master.temp.c$location)) {

  plot.df <- master.temp.c %>% filter(location == loc)

  pl <- ggplot(plot.df, aes(x = date))+
          geom_line(aes(y = airT.lag1, color = "airT.lag1"), alpha = 0.5)+
          geom_line(aes(y = airT.lag5, color = "airT.lag5"), alpha = 0.5)+
          geom_line(aes(y = waterT, color = "waterT"), linewidth = 0.8)+
          labs(x="date", y="temperature")+
          ggtitle(paste(loc))+
          scale_color_manual(values = colors)+
          theme_bw()

  print(pl)
}
```

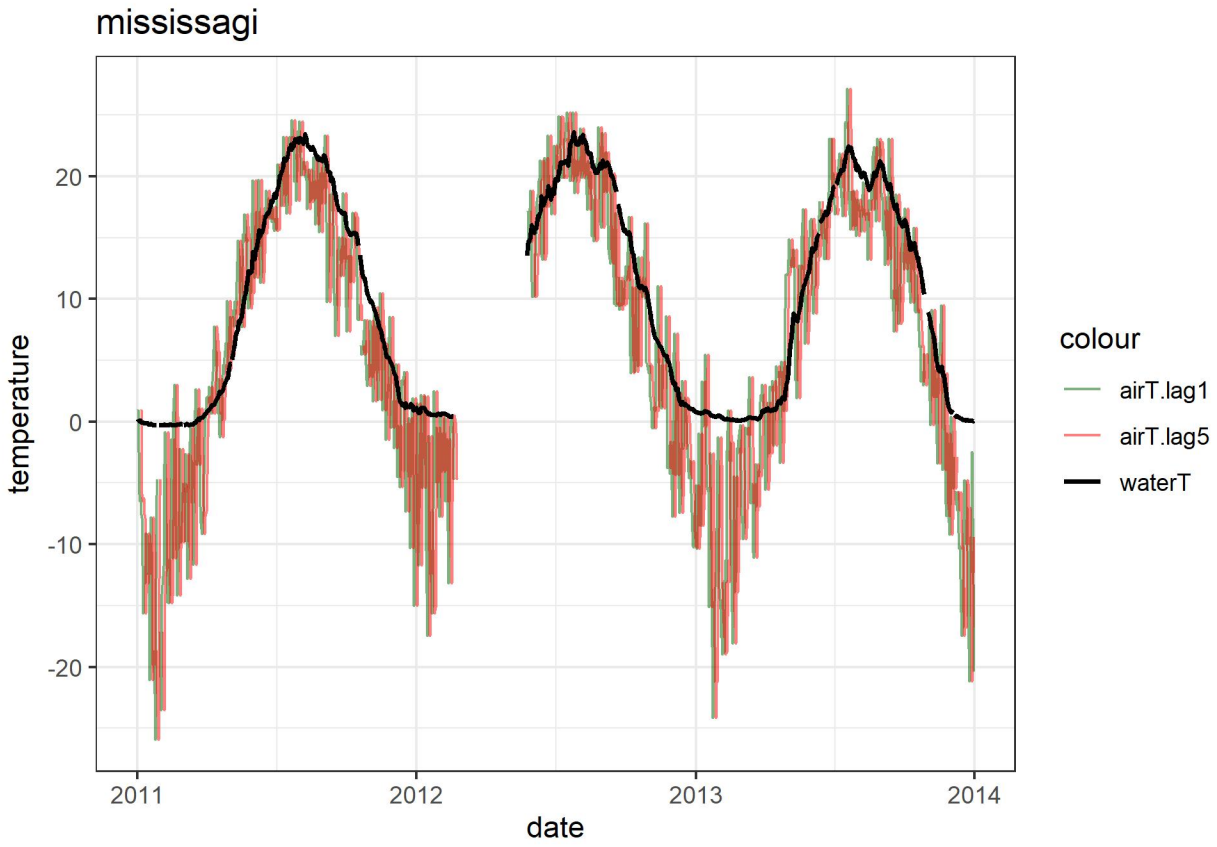

bigcreek
30
20
10
0
-10
-20
temperature
2010
2012
2014
date
colour
airT.lag1
airT.lag5
waterT

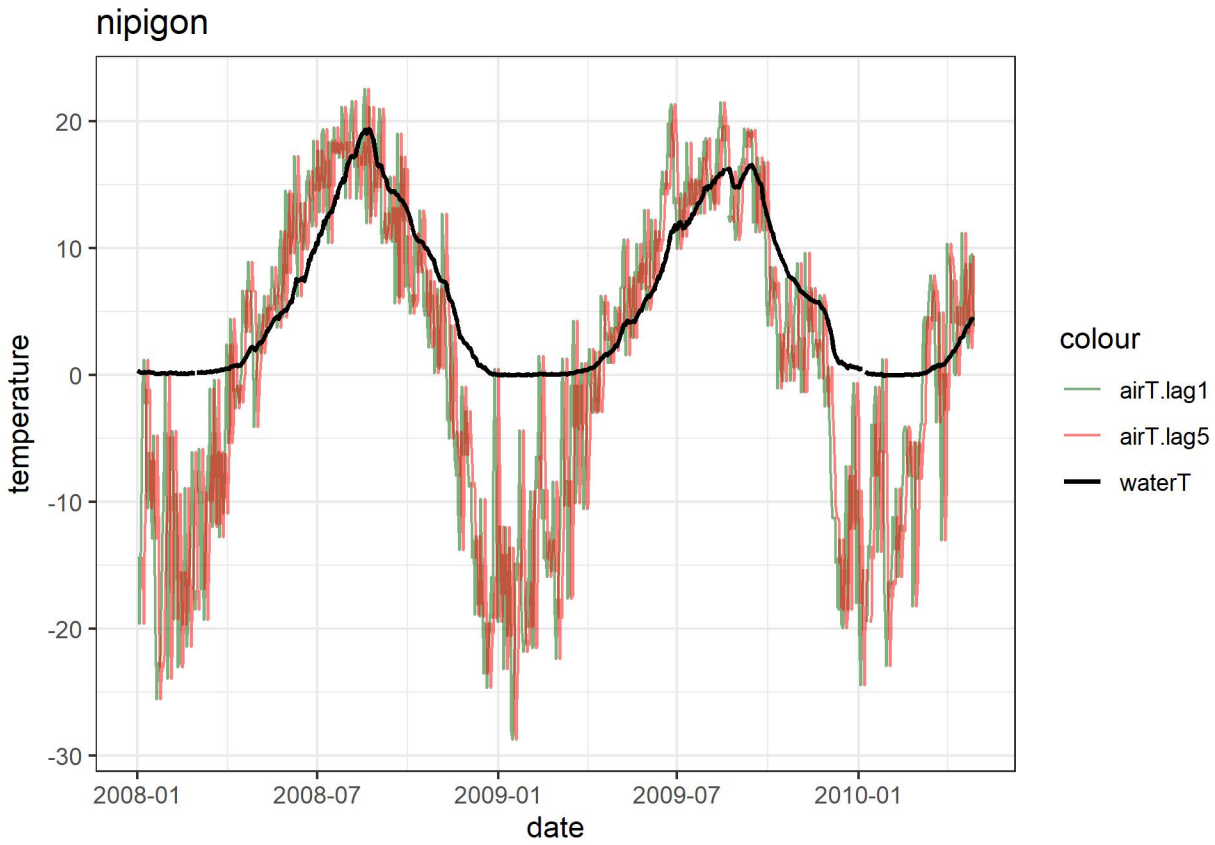

bigotter
30
20
10
0
-10
-20
temperature
2012-07
2013-01
2013-07
2014-01
2014-07
date
colour
airT.lag1
airT.lag5
waterT

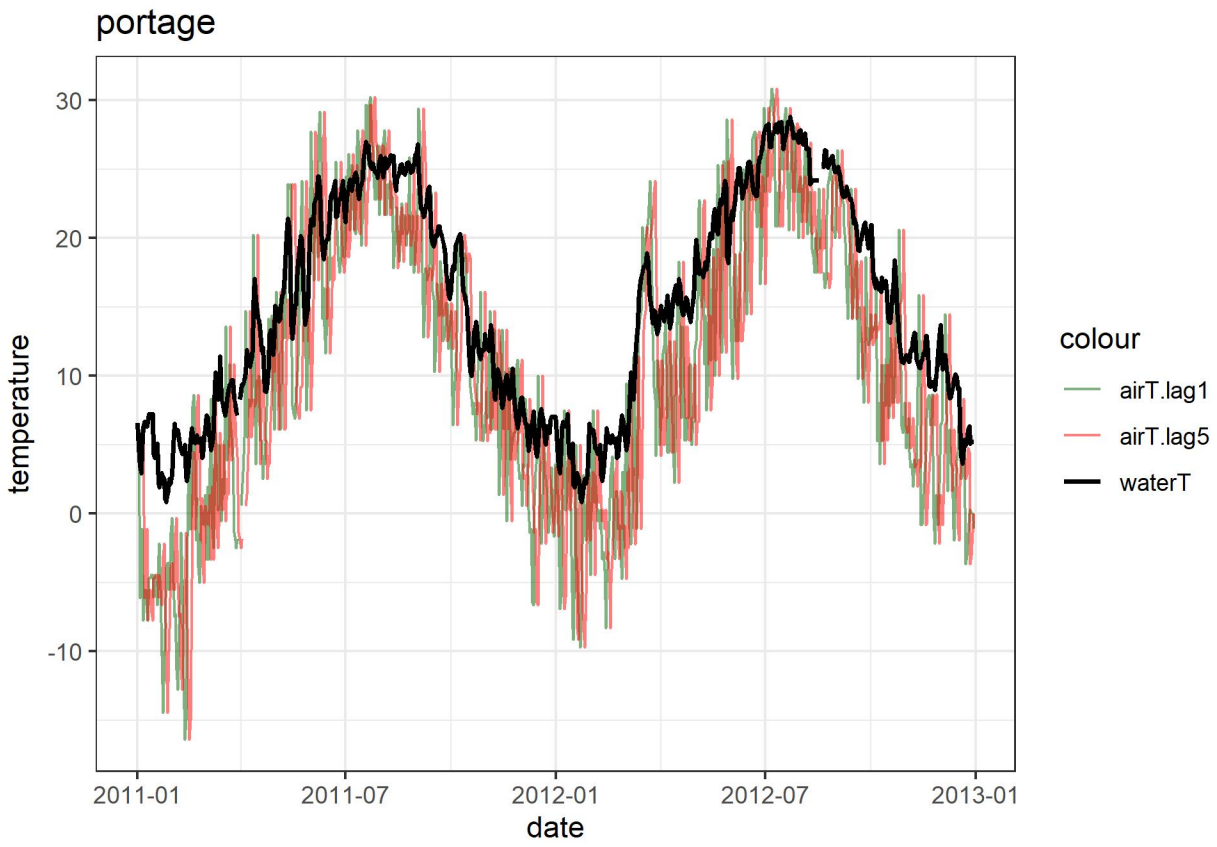

fox
30
20
10
0
-10
-20
temperature
2012
2013
2014
2015
date
colour
airT.lag1
airT.lag5
waterT

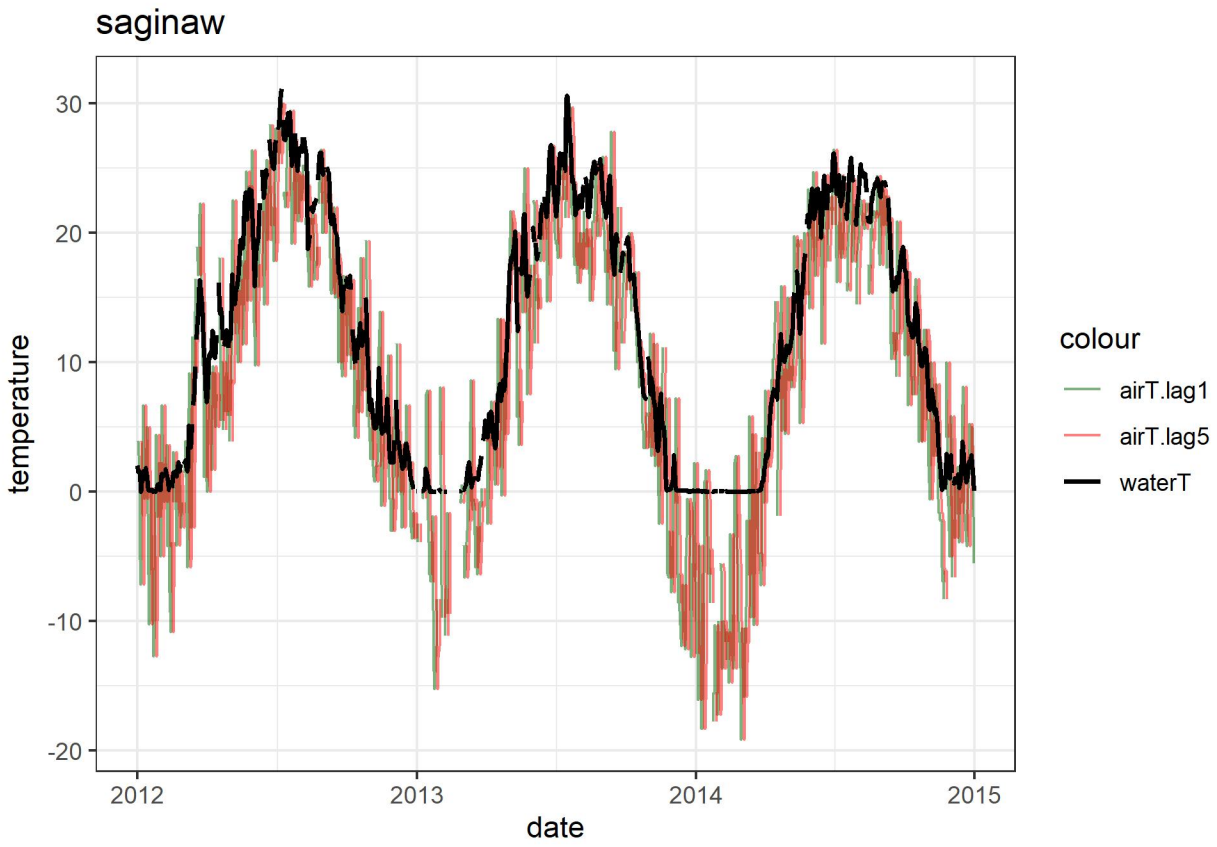

genesee
30
20
10
0
-10
temperature
2011
2012
2013
2014
date
colour
airT.lag1
airT.lag5
waterT

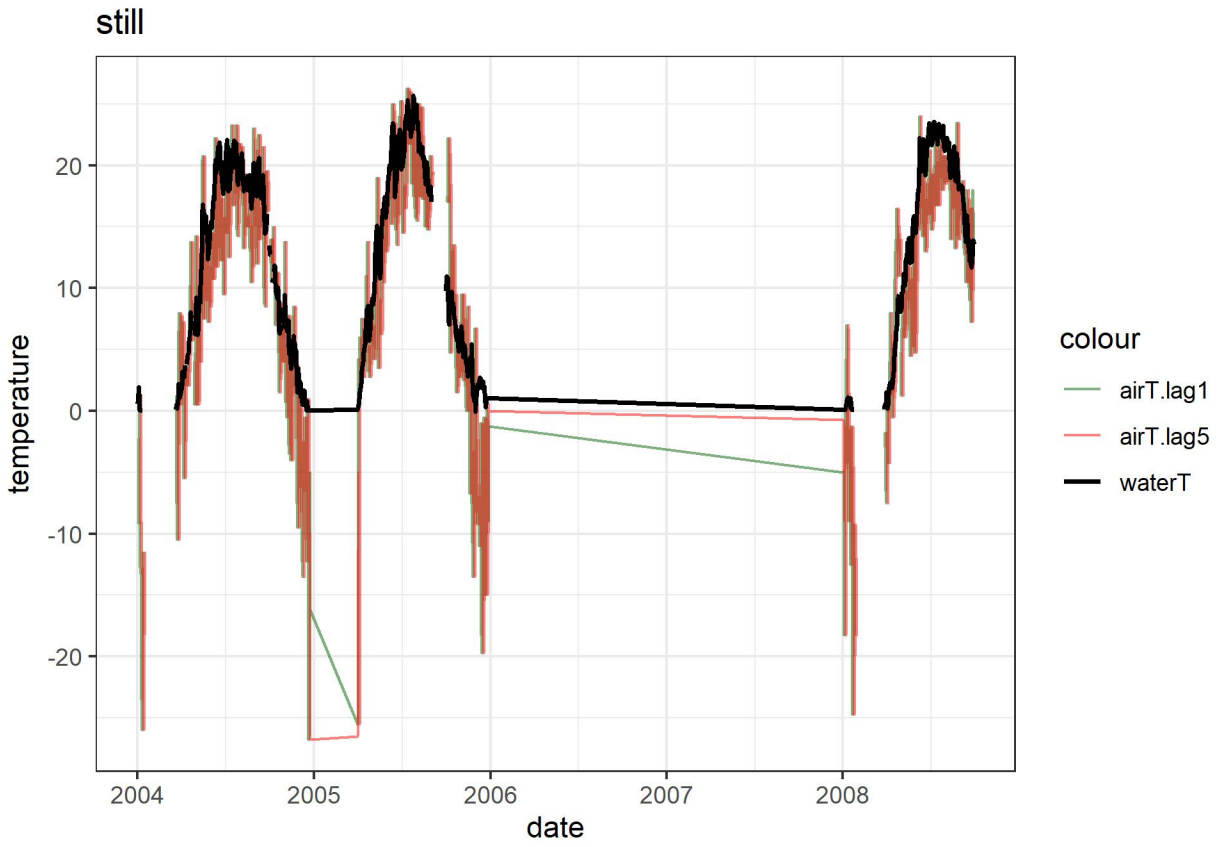

humber
temperature
30
20
10
0
-10
-20
2008
2009
2010
2011
2012
date
colour
airT.lag1
airT.lag5
waterT

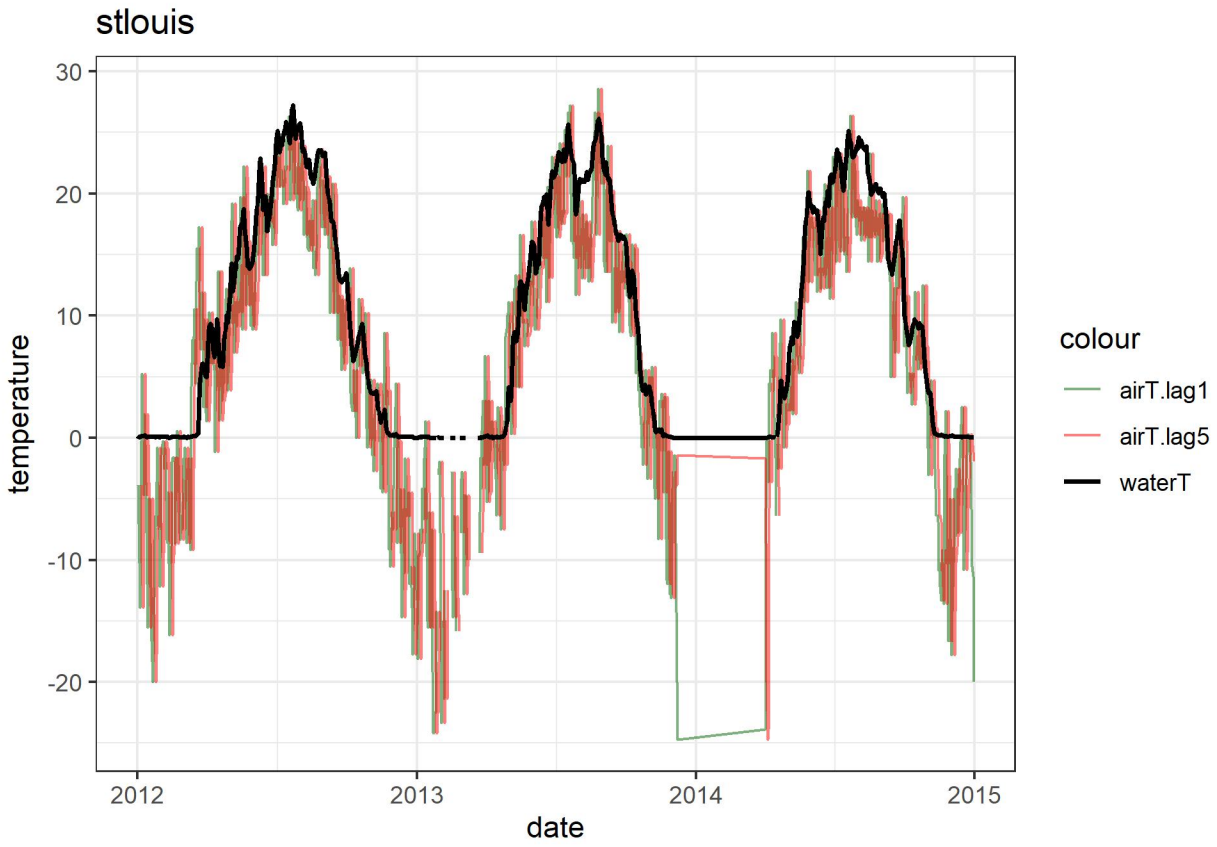

mississagi
temperature
20
10
0
-10
-20
2011
2012
2013
2014
date
colour
airT.lag1
airT.lag5
waterT

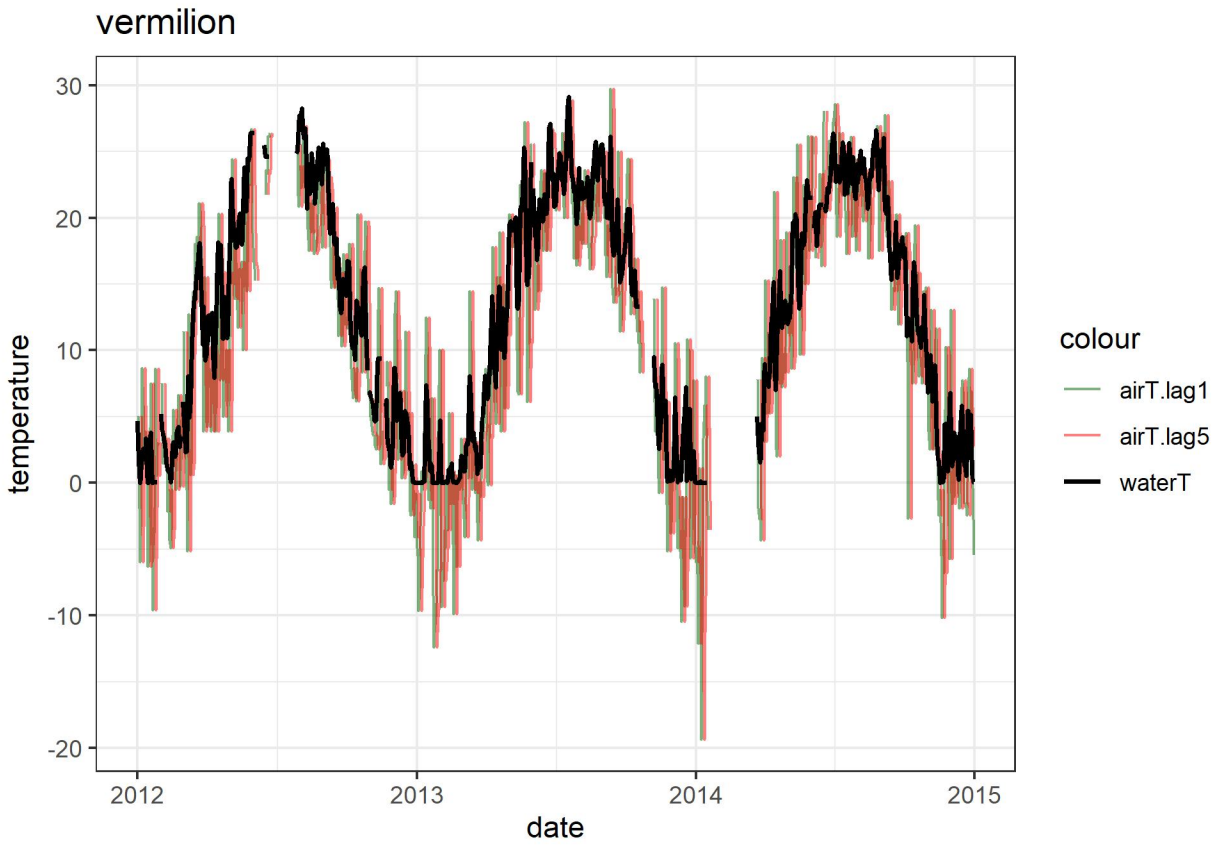

nipigon
temperature
20
10
0
-10
-20
-30
2008-01
2008-07
2009-01
2009-07
2010-01
date
colour
airT.lag1
airT.lag5
waterT


portage
30
20
10
0
-10
temperature
2011-01
2011-07
2012-01
2012-07
2013-01
date
colour
airT.lag1
airT.lag5
waterT


saginaw
temperature
30
20
10
0
-10
-20
2012
2013
2014
2015
date
colour
airT.lag1
airT.lag5
waterT


still
temperature
20
10
0
-10
-20
2004
2005
2006
2007
2008
date
colour
airT.lag1
airT.lag5
waterT


stlouis
30
20
10
0
-10
-20
temperature
2012
2013
2014
2015
date
colour
airT.lag1
airT.lag5
waterT



Get training and testing

First we need to get the training and testing data for 10 fold. Notice that if we want to compare between models, we need to use the same training and testing dataset for fitting all models and calculating RMSE.

```
## Subsetting
df_by_location <- split(master.temp.c, master.temp.c$location)
df_by_location <- df_by_location[sapply(df_by_location, function(x) !is.null(x) && nrow(x) > 0)]

fold = 10 #run 10 folds
combined_training_list <- vector("list",fold)
combined_testing_list <- vector("list",fold)

## Loop
for (f in 1:fold) {

  for (loc in 1:length(df_by_location)) {

    # Subset the current location data, and get train/test
    current <- df_by_location[[loc]]
    current_training <- current[current$year == sample(current$year, 1),]
    current_testing <- current[current$year == sample(current$year, 1),]

    # Add to the combined list
    combined_training_list[[f]] <- rbind(
      combined_training_list[[f]], current_training)
```

```
    combined_testing_list[[f]] <- rbind(
      combined_testing_list[[f]], current_testing)
  }

  combined_training_list[[f]]$year <- as.factor(combined_training_list[[f]]$year)
  combined_testing_list[[f]]$year <- as.factor(combined_testing_list[[f]]$year)
}


## Checks
table(combined_training_list[[1]]$location) #should contain roughly equal number of observations...
```

```
##
##   bigcreek   bigotter        fox    genesee     humber mississagi    nipigon
##        362        268        363        365        365        360        363
##    portage    saginaw      still    stlouis  vermilion
##        364        348        278          0        299
```

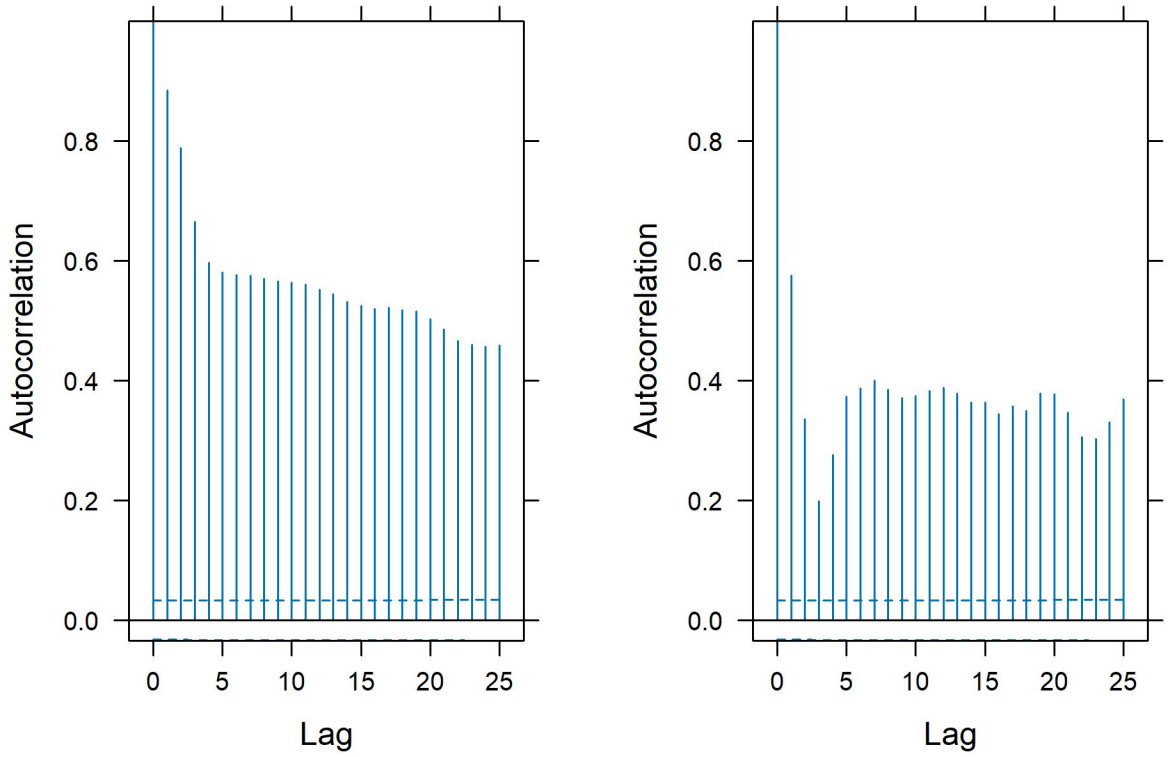
NOTE: All following models are done for one iteration only (using only i=1)!

Multiple linear regression models with lag

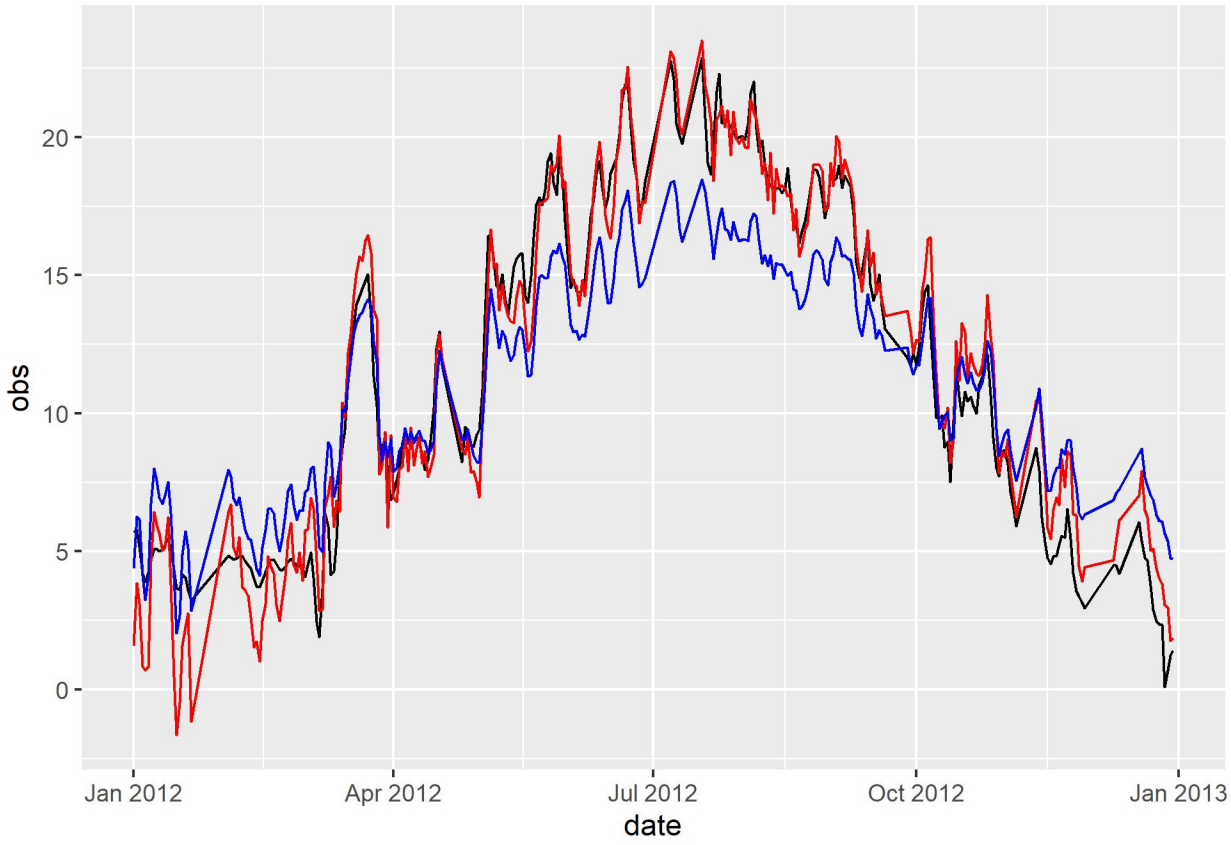
Linear regression lag 3

```
## Forms
form.locrandom <- waterT ~ airT.lag1 + airT.lag2 + airT.lag3

## Iteration starts here
i=1
compare <- NA
# select current dataset, and all unique location levels
current.training <- combined_training_list[[i]]
current.testing <- combined_testing_list[[i]] %>% na.omit()

# model training and predicting
model <- lme(fixed = form.locrandom,
             random = ~ 1 | location,
             na.action = na.omit, data = current.training)
preds <- predict(model, newdata = current.testing,
                 na.action = na.omit)
p <- as.data.frame(preds)

# Include AR term
model.ar <- lme(fixed = form.locrandom,
                random = ~ 1 | location,
                na.action = na.omit, data = current.training,
                correlation=corAR1(form=~1|location, value=0.8, fixed=T))
preds.ar <- predict(model.ar, newdata = current.testing,
                    na.action = na.omit)
p.ar <- as.data.frame(preds.ar)

# compare dataframe output
compare <- cbind(current.testing,preds=p$preds,preds.ar=p.ar$preds.ar) %>%
  select(location, date, obs = waterT, preds, preds.ar)
```

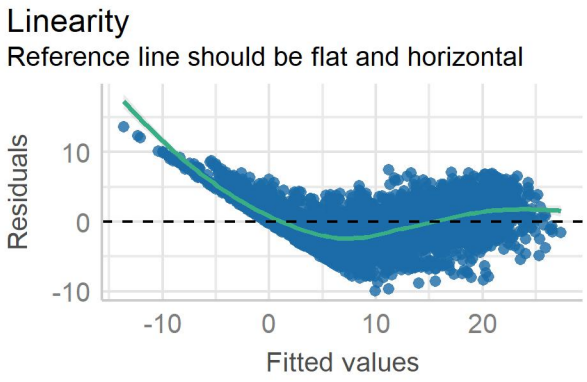
```
## Plot ACF
ggarrange(plot(ACF(model,resType="normalized"),alpha=0.05),
          plot(ACF(model.ar,resType="normalized"),alpha=0.05),
          nrow = 1)
```

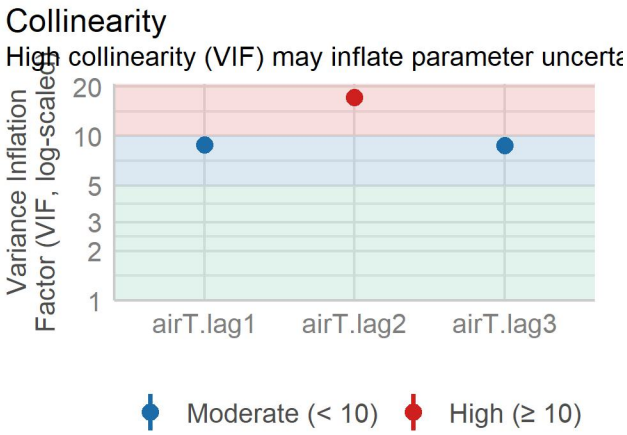


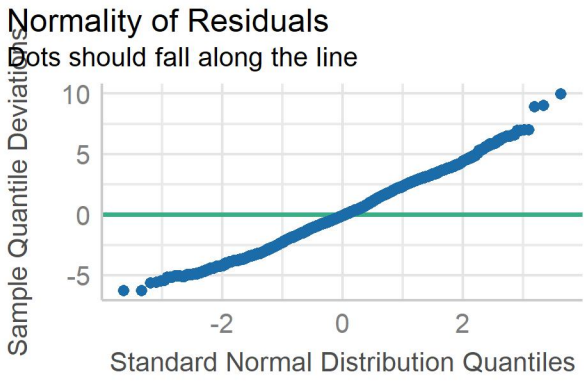

```
## Plots
plot.df <- compare %>% filter(location == "bigcreek")

ggplot(data=plot.df, aes(x=date))+
  geom_line(aes(x=date, y=obs), color = "black")+
  geom_line(aes(x=date, y=preds), color = "red")+
  geom_line(aes(x=date, y=preds.ar), color = "blue")
```

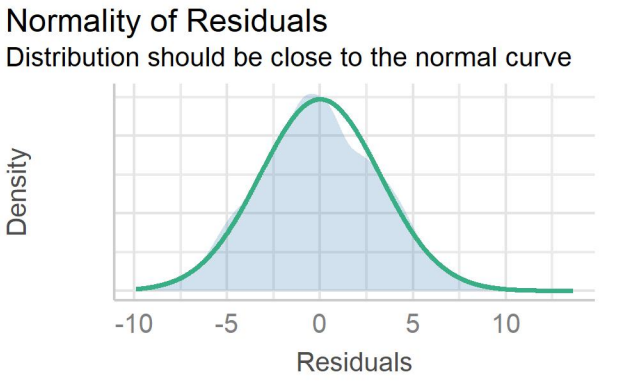



```
check_model(model)
```

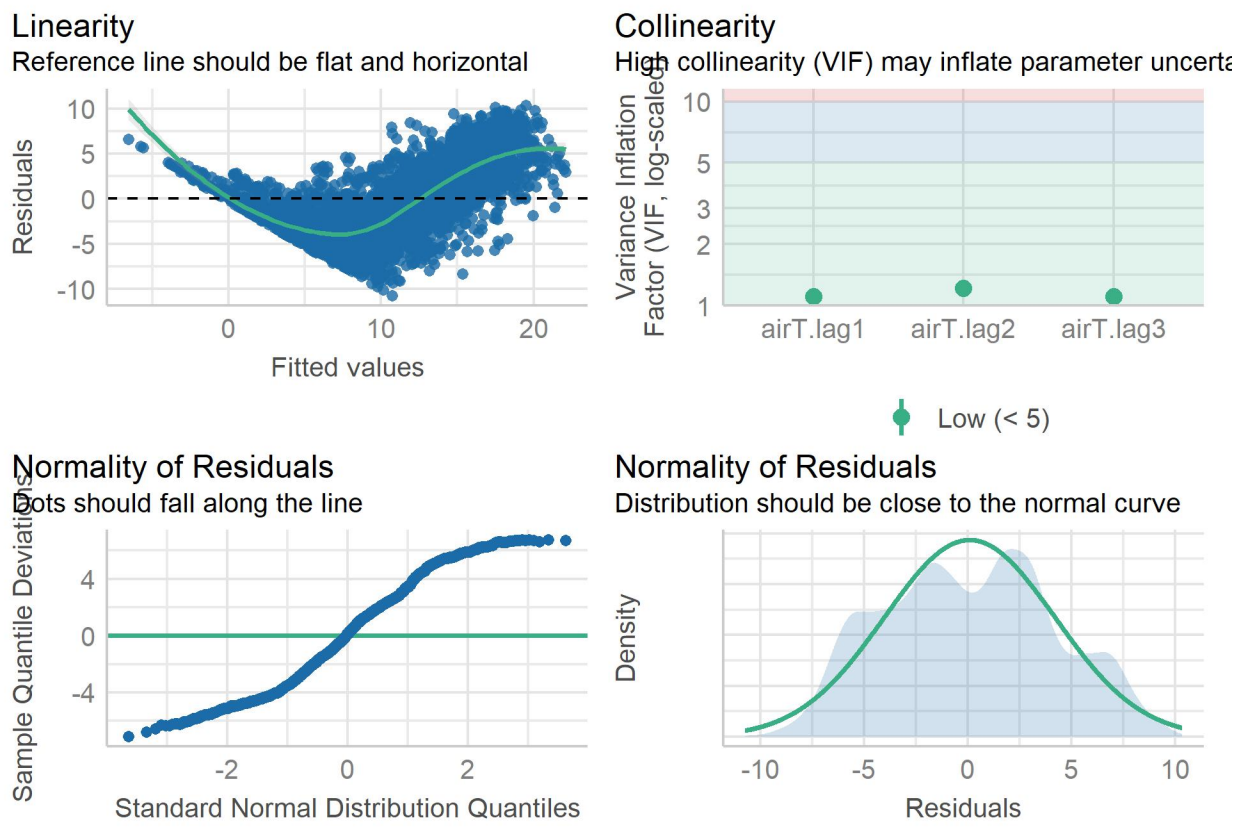









```
check_model(model.ar)
```




```
## Metrics calculation
ModelMetrics::rmse(compare$preds, compare$obs)
```

```
## [1] 3.329388
```

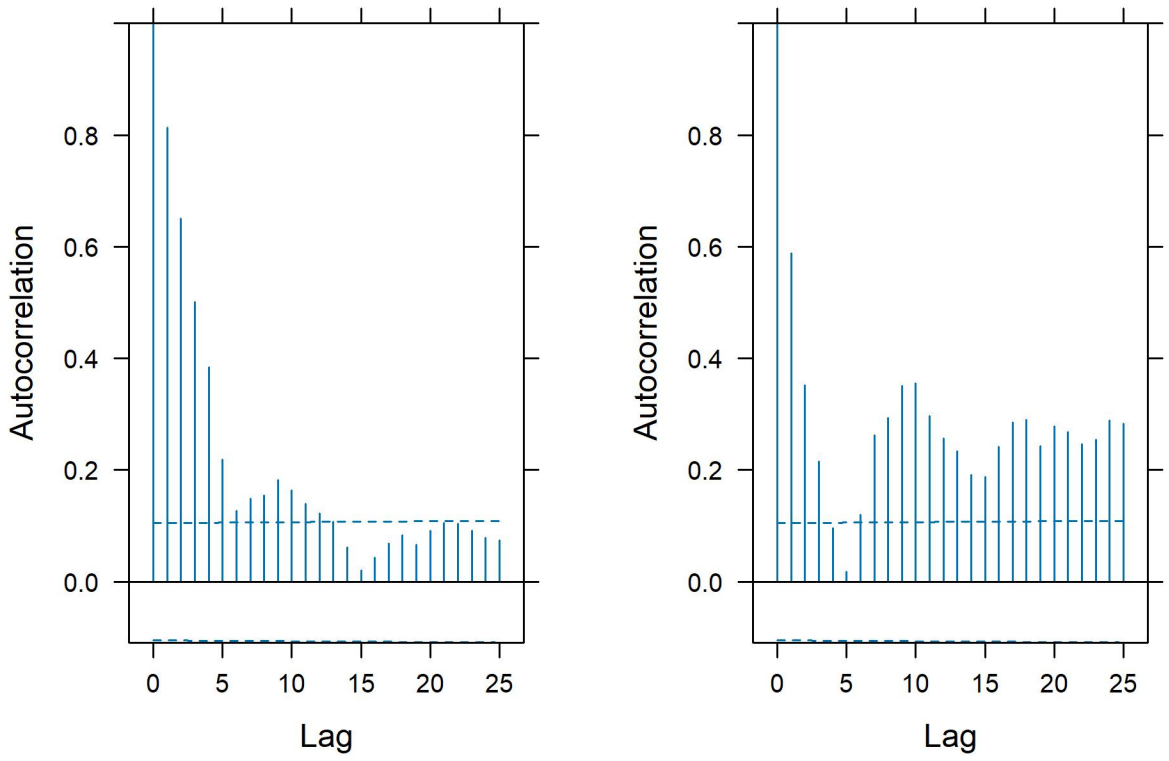
```
ModelMetrics::rmse(compare$preds.ar, compare$obs)
```

```
## [1] 4.225604
```

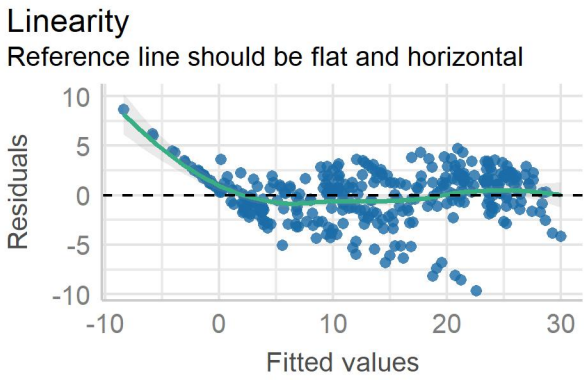
Linear regression lag 5

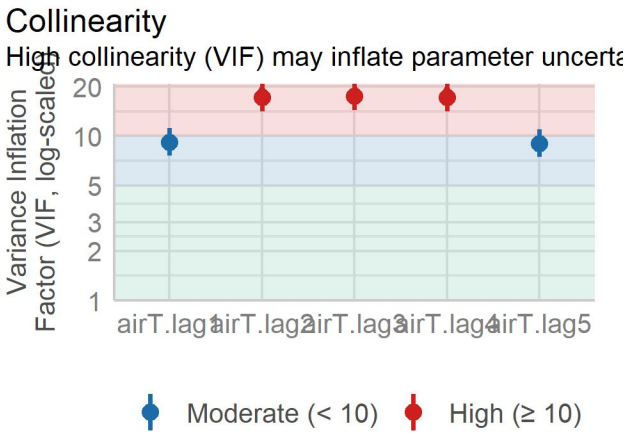
```
## Forms
form.locrandom <- waterT ~ airT.lag1 + airT.lag2 + airT.lag3 + airT.lag4 + airT.lag5

## Iteration starts here
i=1
compare <- NA
# select current dataset, and all unique location levels
current.training <- combined_training_list[[i]] %>%
  filter(location == "fox")
current.testing <- combined_testing_list[[i]] %>%
  filter(location == "fox") %>% na.omit()

# model training and predicting
model <- gls(form.locrandom, na.action = na.omit, data = current.training)
preds <- predict(model, newdata = current.testing,
                 na.action = na.omit)
```

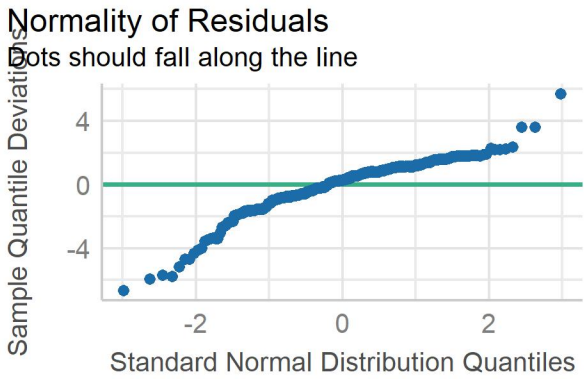
```
p <- as.data.frame(preds)

# Include AR term
model.ar <- gls(form.locrandom,
                na.action = na.omit, data = current.training,
                correlation=corAR1(form=~1|location, value=0.8, fixed=T))
preds.ar <- predict(model.ar, newdata = current.testing,
                    na.action = na.omit)
p.ar <- as.data.frame(preds.ar)

# compare dataframe output
compare <- cbind(current.testing,preds=p$preds,preds.ar=p.ar$preds.ar) %>%
  select(location, date, obs = waterT, preds, preds.ar)


## Plot ACF
ggarrange(plot(ACF(model,resType="normalized"),alpha=0.05),
        plot(ACF(model.ar,resType="normalized"),alpha=0.05),
        nrow = 1)
```

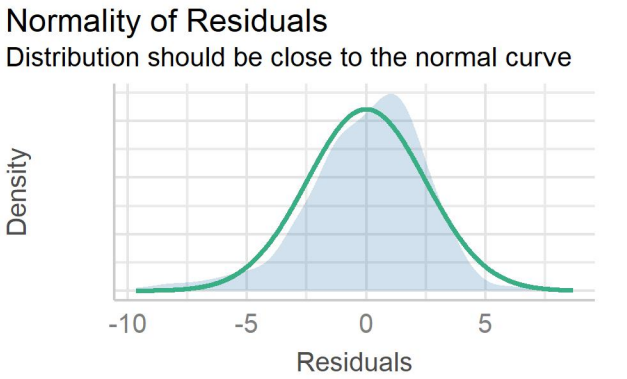



```
check_model(model)
```

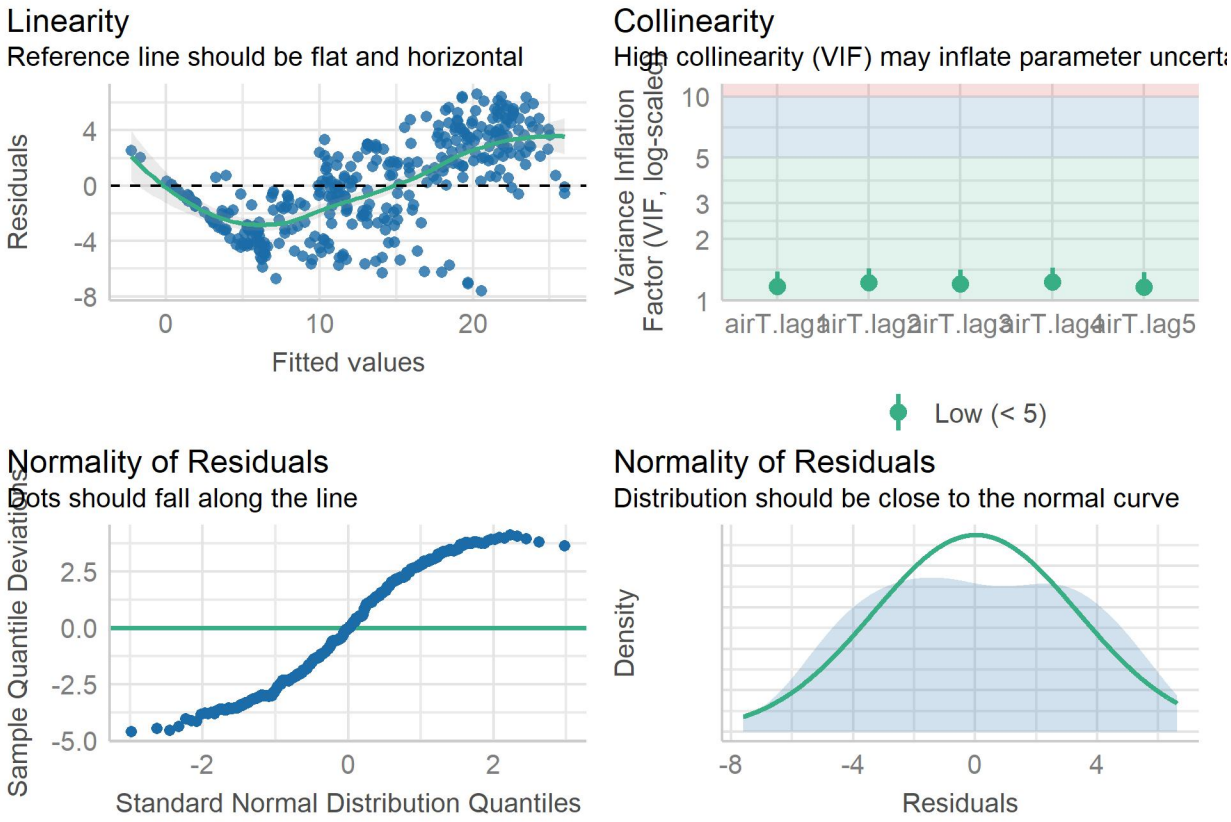




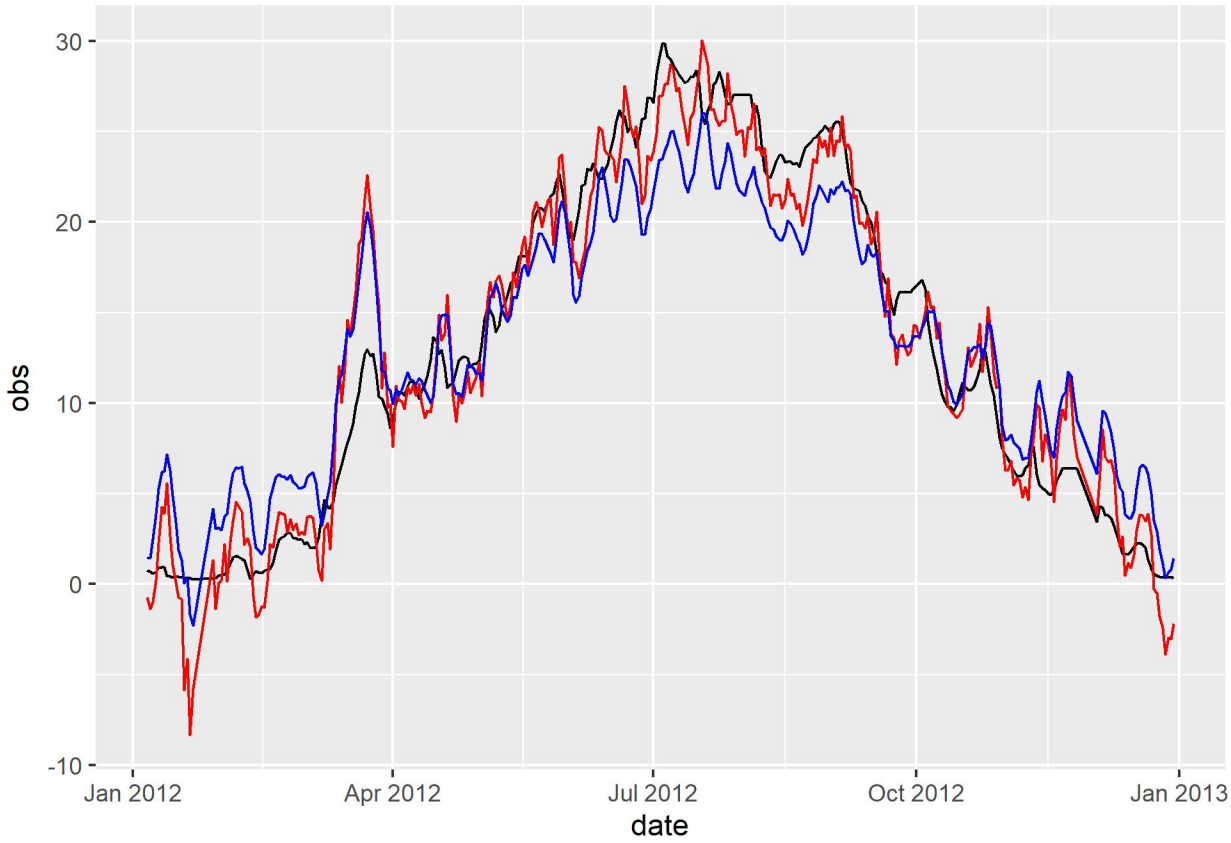





```
check_model(model.ar)
```




```
## Plots
plot.df <- compare %>% filter(location == "fox")

ggplot(data=plot.df, aes(x=date))+
  geom_line(aes(x=date, y=obs), color = "black")+
  geom_line(aes(x=date, y=preds), color = "red")+
  geom_line(aes(x=date, y=preds.ar), color = "blue")
```




```
## Metrics calculation
ModelMetrics::rmse(compare$preds, compare$obs)
```

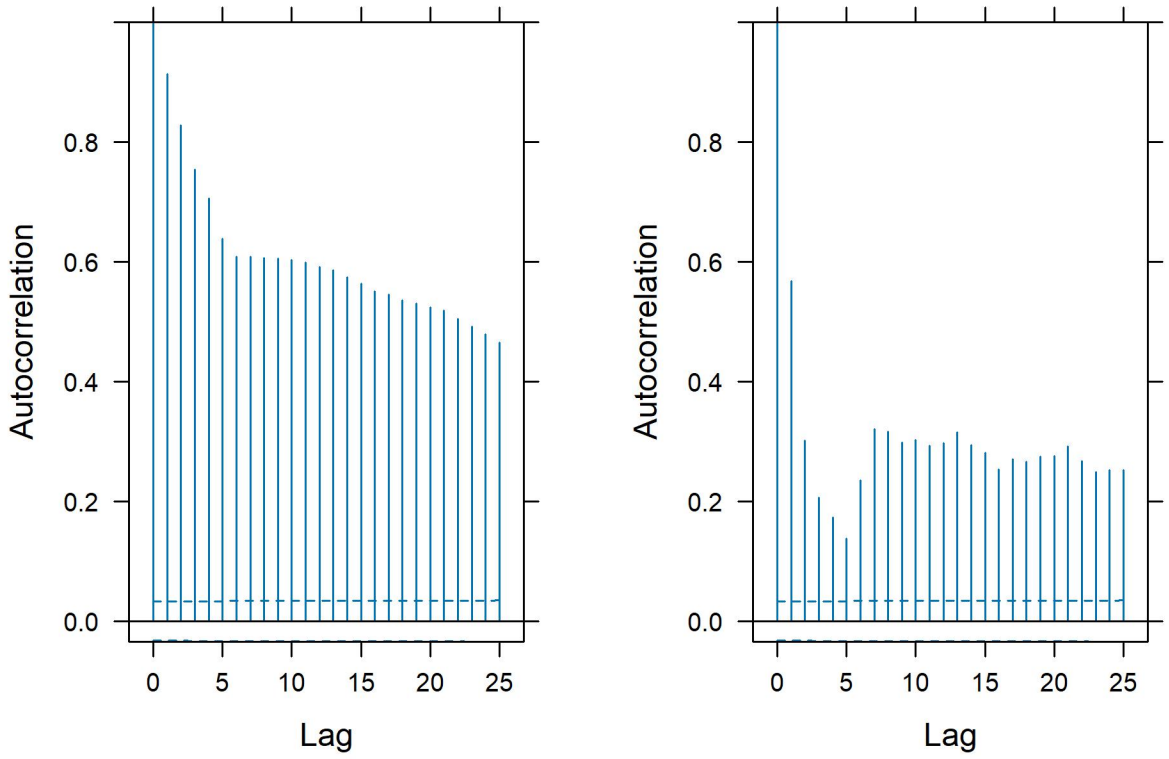
```
## [1] 2.481928
```

```
ModelMetrics::rmse(compare$preds.ar, compare$obs)
```

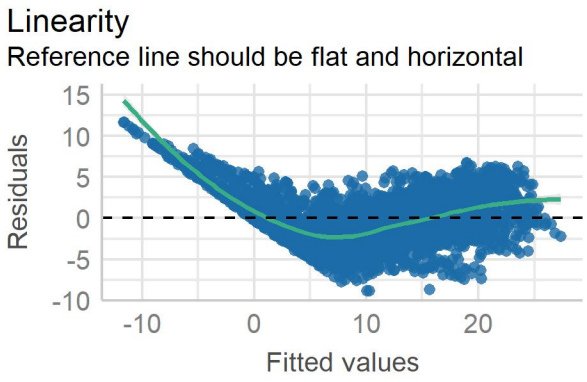
```
## [1] 3.352488
```

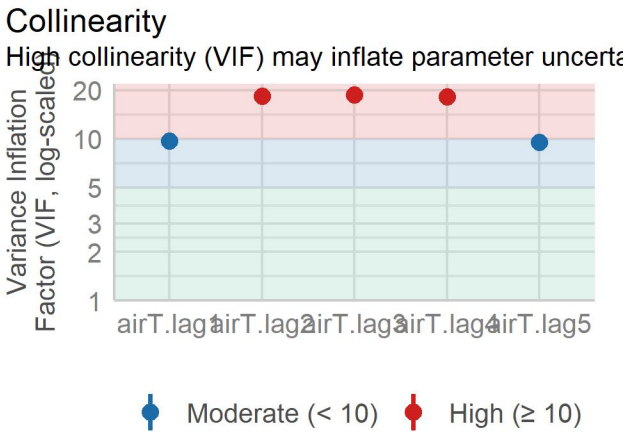
```
## Forms
form.locrandom <- waterT ~ airT.lag1 + airT.lag2 + airT.lag3 + airT.lag4 + airT.lag5

## Iteration starts here
i=1
compare <- NA
# select current dataset, and all unique location levels
current.training <- combined_training_list[[i]]
current.testing <- combined_testing_list[[i]] %>% na.omit()

# model training and predicting
model <- lme(fixed = form.locrandom,
             random = ~1 | location,
             na.action = na.omit, data = current.training)
preds <- predict(model, newdata = current.testing,
                 na.action = na.omit)
p <- as.data.frame(preds)

# Include AR term
```

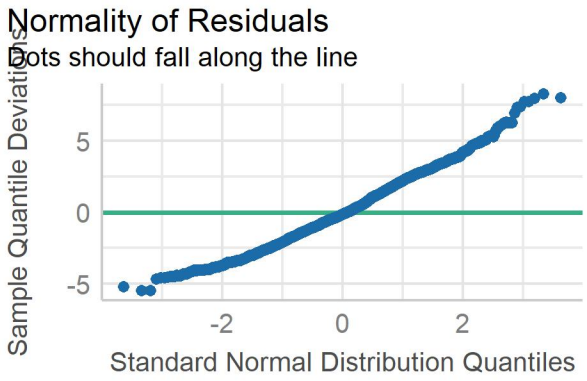
```
model.ar <- lme(fixed = form.locrandom,
                random = ~1 | location,
                na.action = na.omit, data = current.training,
                correlation=corAR1(form=~1|location, value = 0.8, fixed=T))
preds.ar <- predict(model.ar, newdata = current.testing,
                    na.action = na.omit)
p.ar <- as.data.frame(preds.ar)

# compare dataframe output
compare <- cbind(current.testing,preds=p$preds,preds.ar=p.ar$preds.ar) %>%
  select(location, date, obs = waterT, preds, preds.ar)


## Plot ACF
ggarrange(plot(ACF(model,resType="normalized"),alpha=0.05),
        plot(ACF(model.ar,resType="normalized"),alpha=0.05),
        nrow = 1)
```

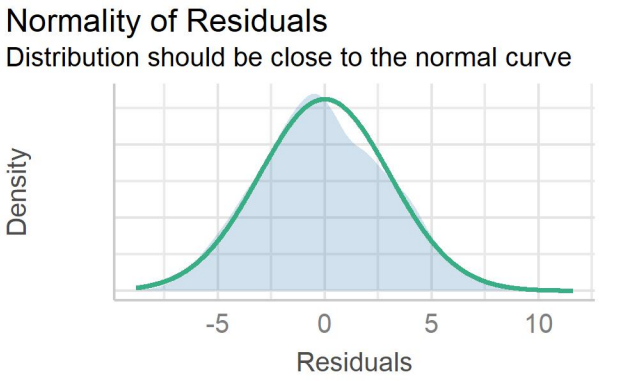



```
check_model(model)
```

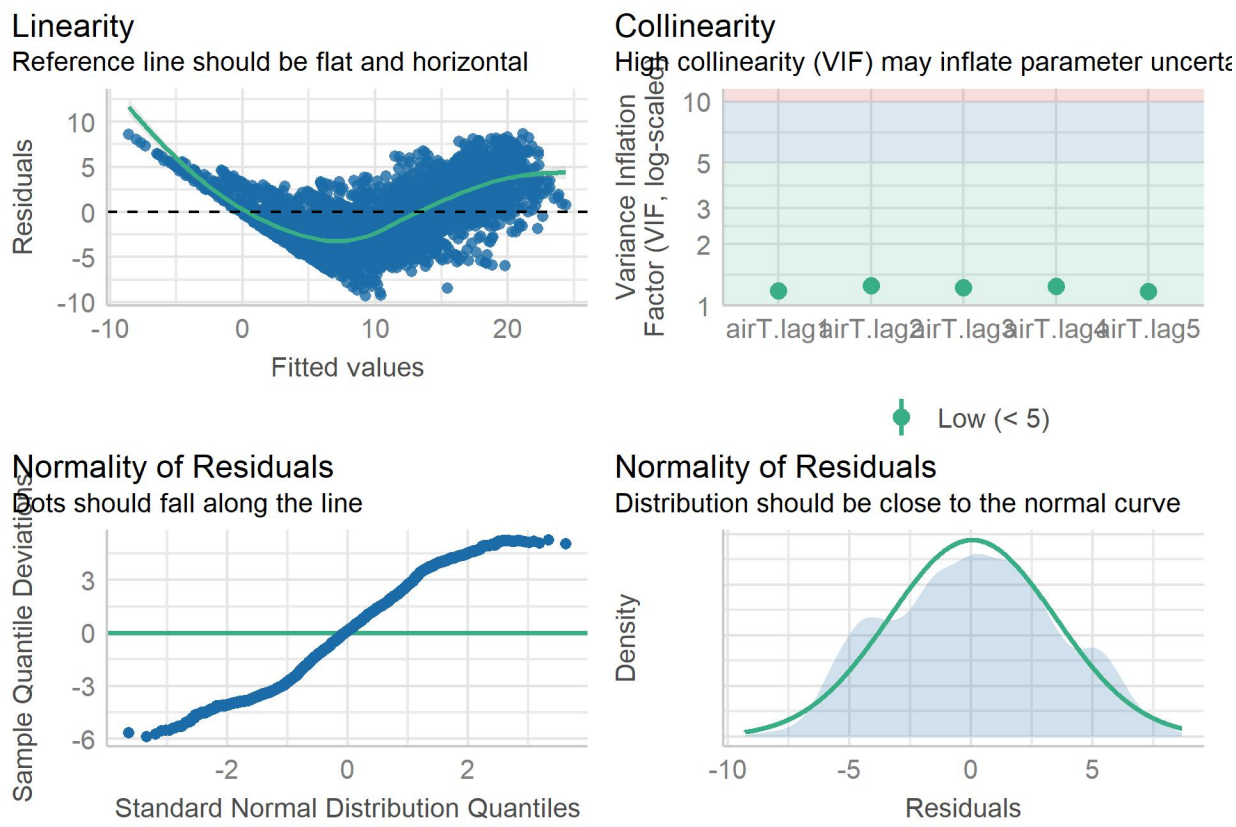




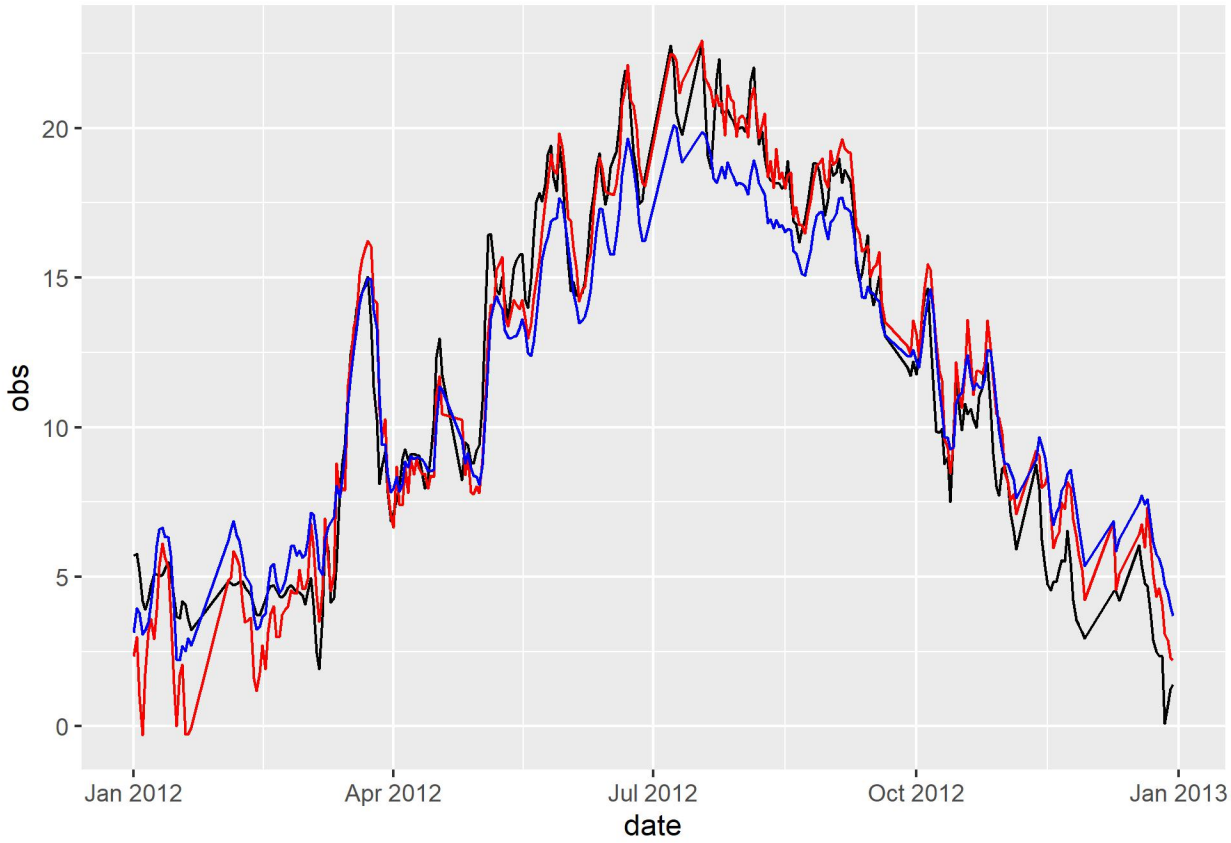





```
check_model(model.ar)
```




```
## Plots
plot.df <- compare %>% filter(location == "bigcreek")

ggplot(data=plot.df, aes(x=date))+
  geom_line(aes(x=date, y=obs), color = "black")+
  geom_line(aes(x=date, y=preds), color = "red")+
  geom_line(aes(x=date, y=preds.ar), color = "blue")
```




```
## Metrics calculation
ModelMetrics::rmse(compare$preds, compare$obs)
```

```
## [1] 3.170756
```

```
ModelMetrics::rmse(compare$preds.ar, compare$obs)
```

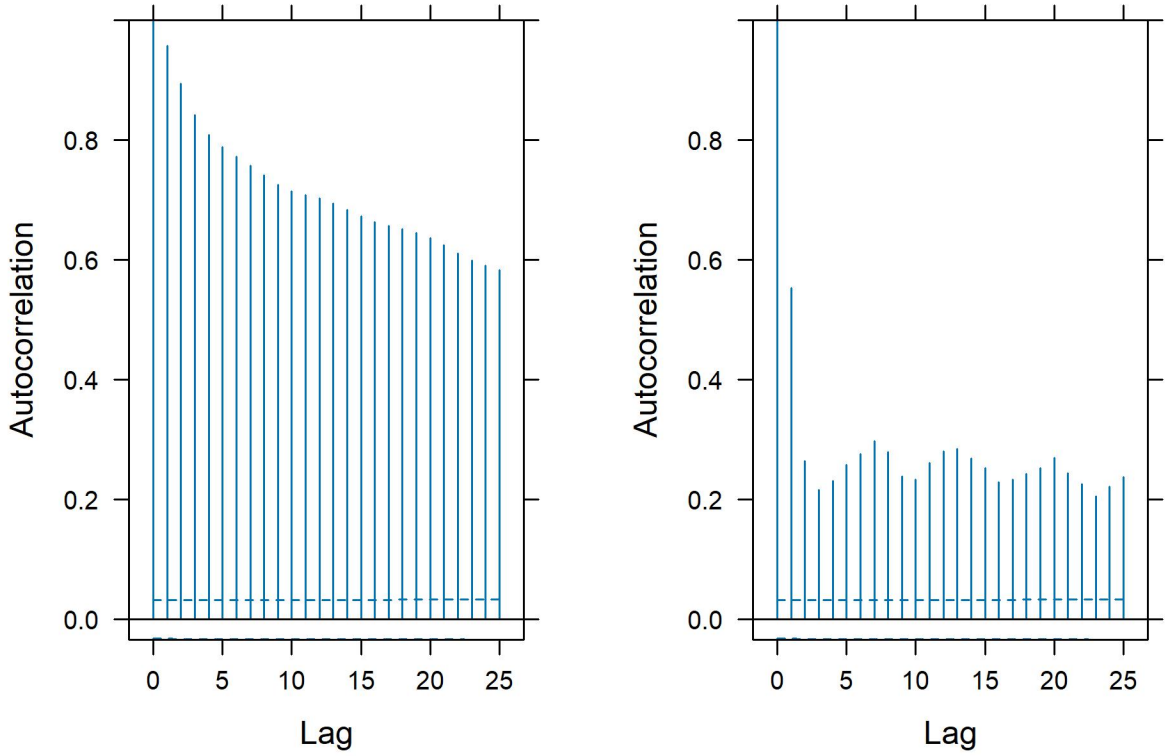
```
## [1] 3.552429
```

Reasonal residual

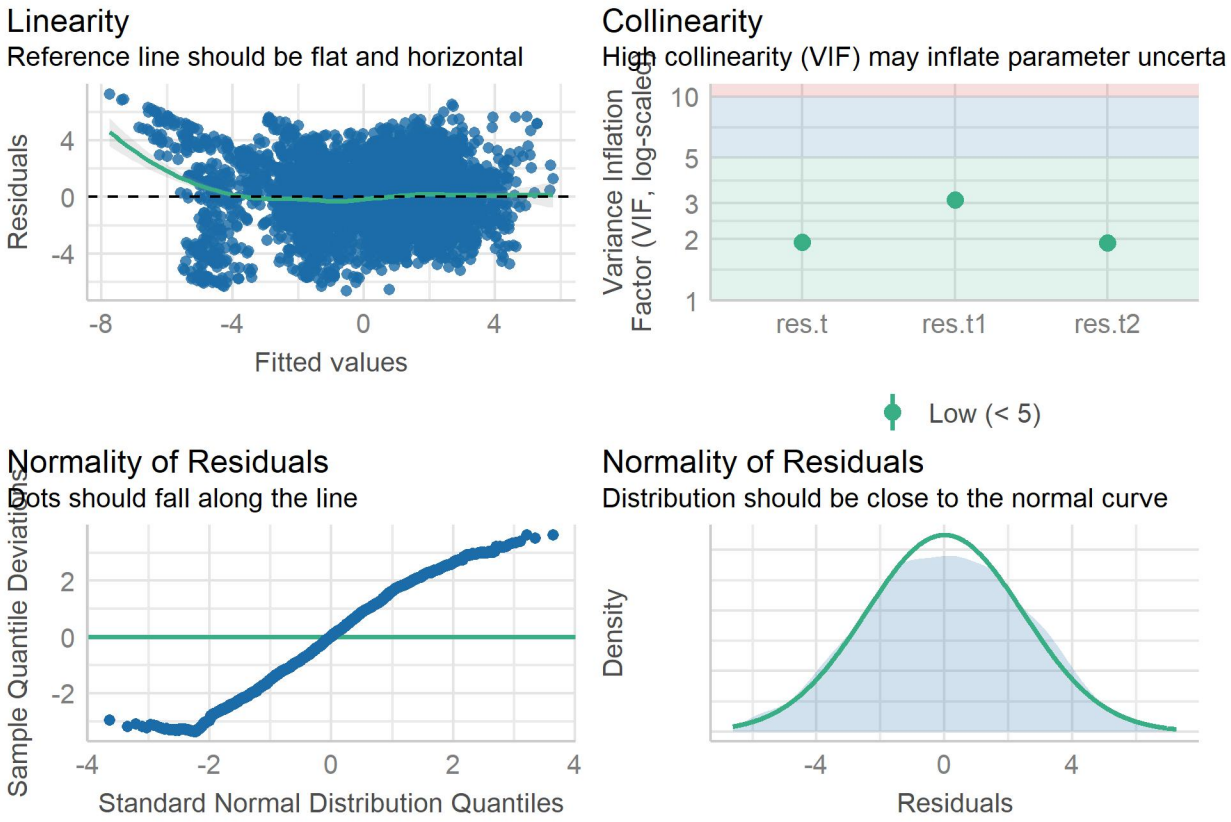
```
i = 1
compare <- NA
# select current dataset, and all unique location levels
current.training <- combined_training_list[[i]]
current.testing <- combined_testing_list[[i]] %>% na.omit()


## TRAINING
# get the model annual component
annual.comp <- nls(airT ~ a+b*sin(2*pi/365*(yday(date)+t0)),
                   start = list(a=0.05, b=5, t0=-26),
                   data=current.training)

# get the air temperature residuals
res <- as.data.frame(matrix(NA, ncol = 2,
                            nrow = length(na.omit(current.training$airT))))
```

```r
# dataframe to store the residuals
colnames(res) <- c("res.t", "location")
res[,"location"] <- na.omit(current.training$location)
res[,"res.t"] <- as.vector(residuals(annual.comp))
res <- res %>% group_by(location) %>%
  mutate(res.t1 = lag(res.t, 1),
         res.t2 = lag(res.t, 2))
res[,"res.w"] <- residuals(nls(waterT ~ a+b*sin(2*pi/365*(yday(date)+t0)),
                               start = list(a=0.05, b=5, t0=-26),
                               data = current.training))

# get the water temperature residual component
residual.comp <- lme(fixed = res.w ~ res.t + res.t1 + res.t2,
                     random = ~ 1|location,
                     data = res, na.action = na.omit)

residual.comp.ar <- lme(fixed = res.w ~ res.t + res.t1 + res.t2,
                        random = ~ 1|location,
                        correlation = corAR1(form=~1|location,value=0.8, fixed=TRUE),
                        data = res, na.action = na.omit)


## TESTING
# Annual
preds.annual <- as.data.frame(predict(annual.comp, newdata=current.testing))
preds.annual <- cbind(preds.annual, current.testing$location)
colnames(preds.annual) <- c("preds.annual", "location")

# Residuals
res <- as.data.frame(matrix(NA, ncol = 2,
                            nrow = length(current.testing$airT))) #residuals
colnames(res) <- c("res.t", "location")
res[,"location"] <- current.testing$location
res[,"res.t"] <- current.testing$airT - preds.annual$preds.annual
res <- res %>% group_by(location) %>%
  mutate(res.t1 = lag(res.t, 1),
         res.t2 = lag(res.t, 2))

# Residuals
preds.residuals <- predict(residual.comp, newdata=res, na.action=na.omit,
                           re.form=~(1|location))
preds.residuals <- cbind(as.data.frame(preds.residuals), na.omit(res)[,"location"])
preds.residuals.ar <- predict(residual.comp.ar,
                              newdata=res, na.action=na.omit,
                              re.form=~(1|location))
preds.residuals.ar <- cbind(as.data.frame(preds.residuals.ar), na.omit(res)[,"location"])


# Combine and add up both components
preds.annual.short <- preds.annual %>%
  group_by(location) %>% filter(row_number()>2) #remove first two rows for each locations

p <- cbind(preds.annual.short, preds.residuals,preds.residuals.ar) %>%
```

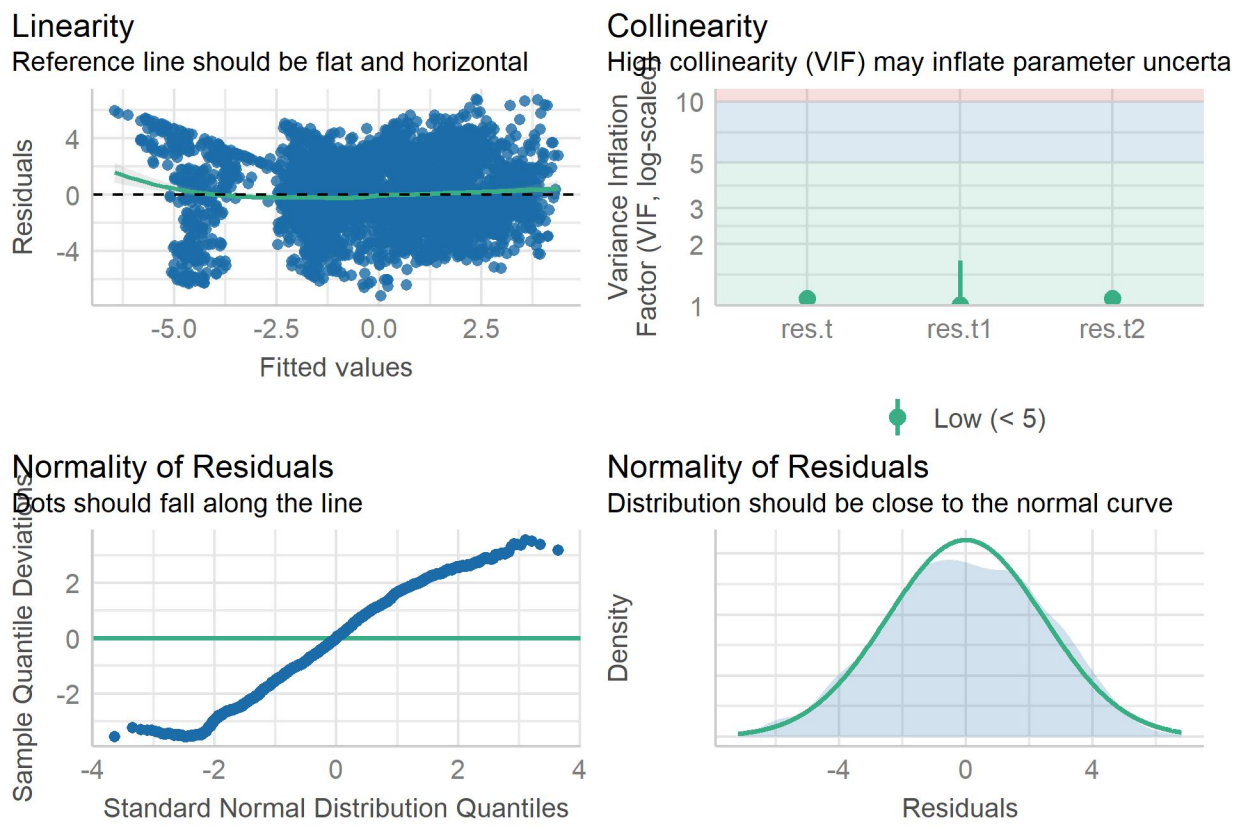
```
  mutate(preds = preds.annual + preds.residuals,
         preds.ar = preds.annual + preds.residuals.ar)

# Create a comparison dataframe
current.testing <- current.testing %>%
  group_by(location) %>% filter(row_number()>2) #remove first two rows for each locations

compare <- cbind(current.testing, p) %>%
  select(location = location...17, date, obs = waterT, preds, preds.ar)


## Plot ACF
ggarrange(plot(ACF(residual.comp,resType="normalized"),alpha=0.05),
          plot(ACF(residual.comp.ar,resType="normalized"),alpha=0.05),
        nrow = 1)
```



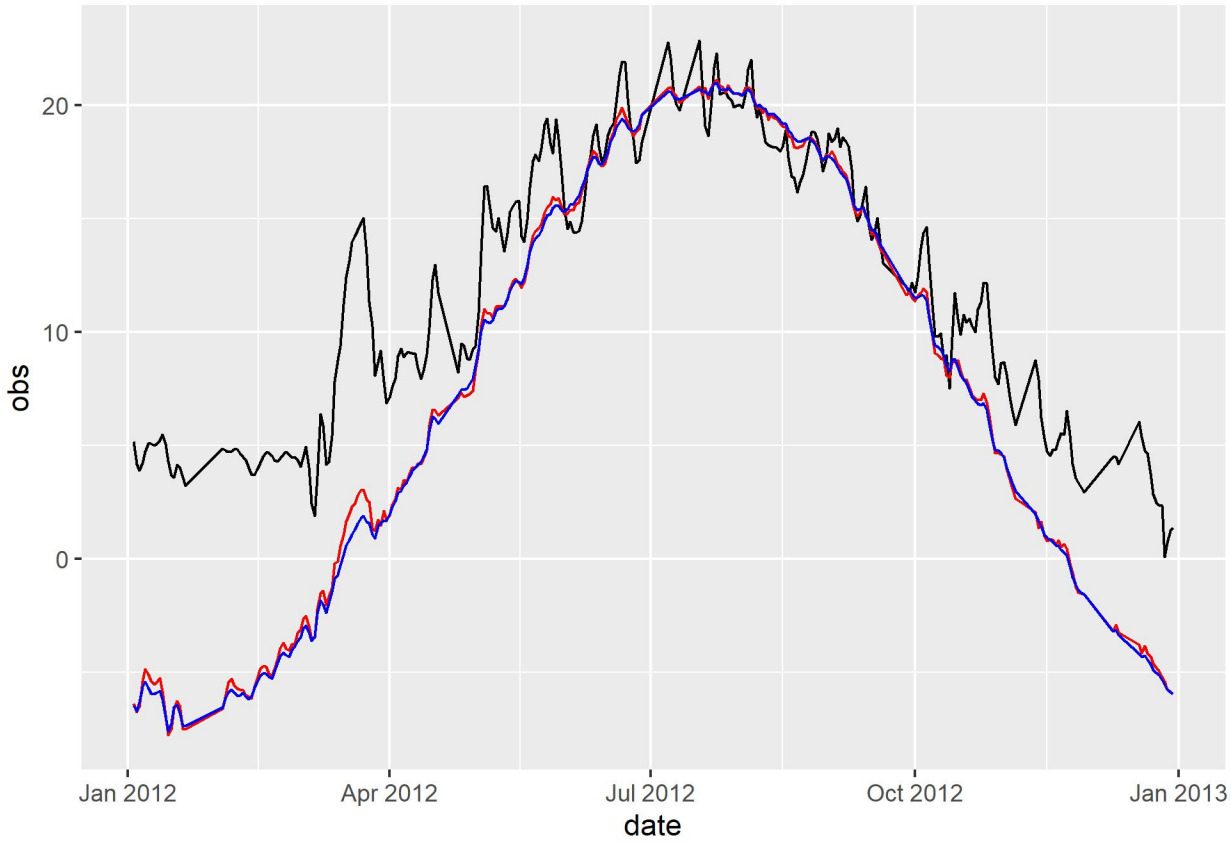

```
check_model(residual.comp)
```




```
check_model(residual.comp.ar)
```




```
## Plots
plot.df <- compare %>% filter(location == "bigcreek")

ggplot(data=plot.df, aes(x=date))+
  geom_line(aes(x=date, y=obs), color = "black")+
  geom_line(aes(x=date, y=preds), color = "red")+
  geom_line(aes(x=date, y=preds.ar), color = "blue")
```




```
## Metrics calculation
ModelMetrics::rmse(compare$preds, compare$obs)
```

```
## [1] 4.966473
```

```
ModelMetrics::rmse(compare$preds.ar, compare$obs)
```

```
## [1] 5.025352
```

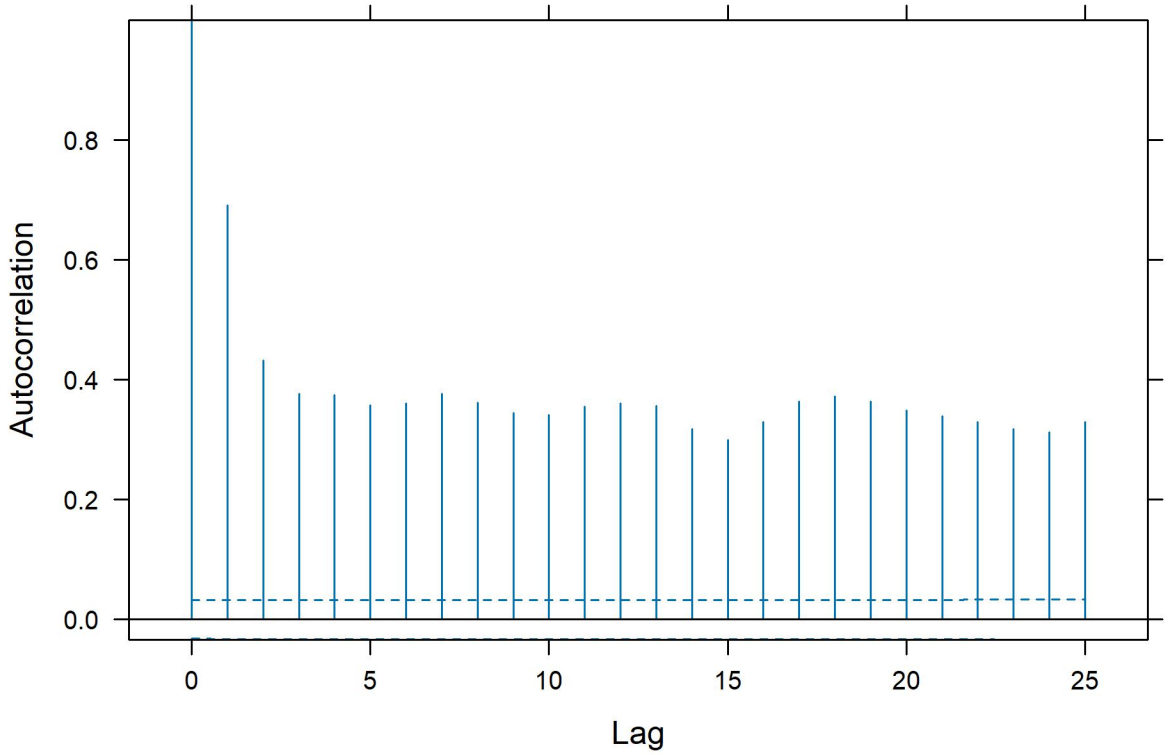
So, the seasonal residual model captures the annual variation very well. Even with a very high temporal autocorrelation correction, it stills gives a relatively good result.

Non-linear model

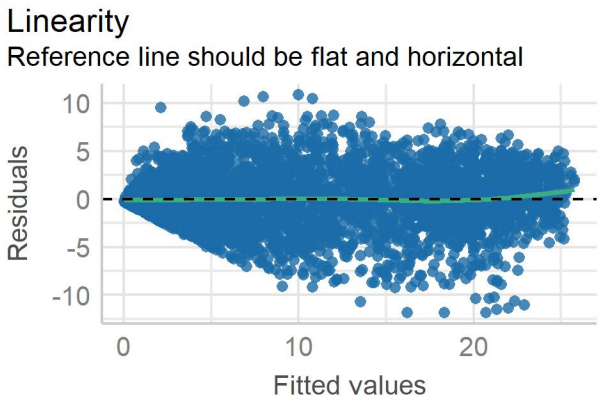
No AR1 structure in this model.

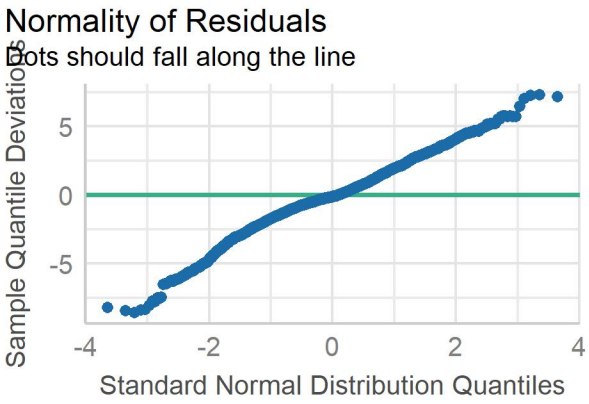
```
## Iteration starts here
i=1
compare <- NA
# select current dataset, and all unique location levels
current.training <- combined_training_list[[i]]
current.testing <- combined_testing_list[[i]] %>% na.omit()

model <- nlme(waterT ~ alpha / (1 + exp(gamma * (beta - airT))),
              fixed = list(alpha~1,gamma~1,beta~1),
              random = gamma ~ 1|location,
              start = c(alpha=30, gamma=0.2, beta=0.1),
              data=current.training, na.action = na.omit)
```

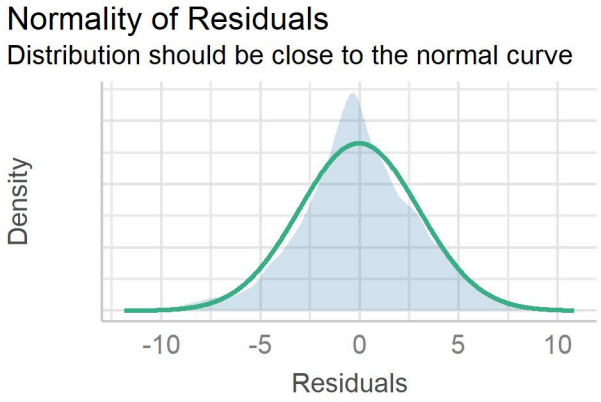
```
preds <- predict(model, newdata = current.testing, level = 1,
                 na.action = na.omit)
p <- as.data.frame(preds)


# compare dataframe output
compare <- cbind(current.testing,preds=p$preds) %>%
  select(location, date, obs = waterT, preds)


## Plot ACF
plot(ACF(model,resType="normalized"),alpha=0.05)
```



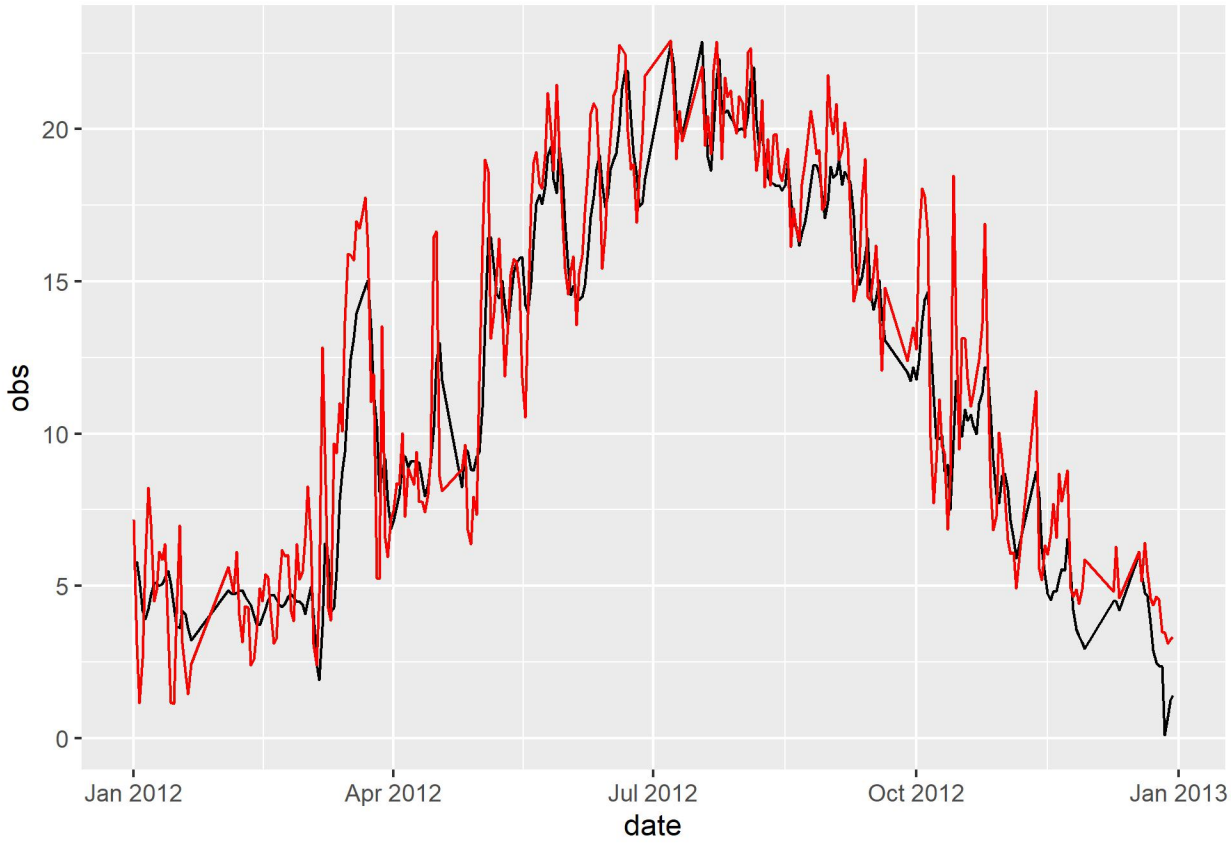

```
check_model(model)
```








```
## Plots
plot.df <- compare %>% filter(location == "bigcreek")

ggplot(data=plot.df, aes(x=date))+
  geom_line(aes(x=date, y=obs), color = "black")+
  geom_line(aes(x=date, y=preds), color = "red")
```




```
## Metrics calculation
ModelMetrics::rmse(compare$preds, compare$obs)
```

```
## [1] 3.167867
```