

Temporal autocorrelation

Eddie Wu

2024-07-18

```
library(dplyr)
library(tidyverse)
library(lme4)
library(nlme)
library(xts)
library(ModelMetrics)
library(zoo)
library(lubridate)
library(ggpubr)
library(astsa) #for lag scatterplot

get.traintest <- function(master.df, year.list, fold) {

  # create output
  combined_training_list <- vector("list",fold)
  combined_testing_list <- vector("list",fold)

  ## Loop 10 folds
  for (f in 1:fold) {

    # dataframe to store the sampled years for each location
    dftrain = data.frame(location=character(),
                          year=integer(), stringsAsFactors = FALSE)
    dftrain$location=character()
    dftrain$year=integer(), stringsAsFactors = FALSE)

    for (i in 1:length(year.list)){ # i loops through each location
      smp = year.list[[i]] #train
      if (length(smp)>1){
        idx.train=(sample(smp, 1))
        ytrain=idx.train
      } else if (length(smp)==1){
        idx.train=smp
        ytrain=idx.train
      }

      smpt = year.list[[i]][year.list[[i]] != idx.train] #test
      if (length(smpt)>1){
        idx.test=(sample(smpt, 1))
        ytest=idx.test
      } else if (length(smpt)==1){
        idx.test=smpt
        ytest=idx.test
      }
    }
  }
}
```

```

    }

    dfind[i,] = c(names(year.list[i]), ytrain)
    dfindt[i,] = c(names(year.list[i]), ytest)
}

# subset the dataframe
awf_train <- merge(master.df, dfind, by=c("location", "year"))
awf_test <- merge(master.df, dfindt, by=c("location", "year"))

combined_training_list[[f]] <- awf_train
combined_testing_list[[f]] <- awf_test
}
return(list(combined_training_list, combined_testing_list))
}

good_year <- function(master.df, dayln) {
  # get table of sample years by location
  yr_out=table(master.df$location, master.df$year)

  # select sample years with more than X days (x=250)
  full_year=list()
  sind=vector()
  cnt=0
  ind=0

  for (i in seq_along(loc_seq)){
    rm(ind)
    if (ncol(yr_out)>=1) {
      ind=which(yr_out[row.names(yr_out)==loc_seq[i],]>dayln)

      if (length(ind)>0) {
        sind=c(sind,i)
        cnt=cnt+1
        full_year[[cnt]]=colnames(yr_out)[ind]
      }
    }
    full_year=setNames(full_year,loc_seq[sind])
    print(full_year)
  }
}

```

Data importing and cleaning

```

## Data import
air <- read.csv("tributary air temperatures clean.csv", stringsAsFactors=F)
water <- read.csv("tributary water temperature/water_temperature_d.csv",
                  stringsAsFactors=F)
flow <- read.csv("tributary discharge.csv", stringsAsFactors=F)

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

```

```

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

```

Data combining

Now we want to combine air, water, and flow into a master dataframe

```

## Combine water temperature and discharge
aw <- merge(air, water, by = c("station_name", "date"))
awf <- merge(aw, flow, by = c("location", "date"))
awf <- awf %>% arrange(location, date)

## Change into factor
awf$location <- as.factor(awf$location)
awf$station_name <- as.factor(awf$station_name)

# Get location sequence
loc_seq=levels(awf$location)

## Check if there are any duplicates
duplicates <- awf %>%
  group_by(location, date) %>%
  filter(n() > 1) # Duplicates should be NA...

## Print the result
table(awf$location)

##
##      bigcreek    bigotter       fox     genesee     humber mississagi    nipigon
##      4554          919      1374      1095      4446      1434        848
##      portage     saginaw      still    stlouis  vermilion
##      730          1095      1246      1349      1095

```

Check for imputed values in water temp

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 awf water temperature
which(is.na(awf$temp))

## integer(0)
# Need to calculate the rolling mean for each location separately...
for(loc in loc_seq) {
  sub <- awf[awf$location == loc,]
  sub$rolling_mean <- rollmean(sub$temp, k = 7, fill = NA, align = "right")
  awf[awf$location == loc, "rolling_mean"] <- sub$rolling_mean
}

# Calculate variance

```

```

awf$variance <- (awf$temp - awf$rolling_mean)^2

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

# Check results
awf %>%
  group_by(location) %>%
  summarise(na_count = sum(is.na(temp)))

## # A tibble: 12 x 2
##   location    na_count
##   <fct>        <int>
## 1 bigcreek      101
## 2 bigotter       0
## 3 fox            6
## 4 genesee        6
## 5 humber         12
## 6 mississagi     6
## 7 nipigon        0
## 8 portage         2
## 9 saginaw        17
## 10 still         226
## 11 stlouis       144
## 12 vermilion     137

```

Calculate lagged days

```

## Get the time lag day variables
awf <- awf %>%
  group_by(location) %>%
  mutate(dmean_1 = lag(mean_temp, 1),
         dmean_2 = lag(mean_temp, 2),
         dmean_3 = lag(mean_temp, 3),
         dmean_4 = lag(mean_temp, 4),
         dmean_5 = lag(mean_temp, 5))

```

Get final master temp dataframe

```

master.temp <- awf[complete.cases(cbind(awf$mean_temp,awf$temp)),] %>%
  select(location, date, station_name, country,
         year, month = month.x, day = day.x,
         water = temp, air = mean_temp, dmean_1, dmean_2, dmean_3,
         dmean_4, dmean_5, flow)

master.temp <- master.temp[complete.cases(master.temp[,8:15]),]

```

Get training and testing (for season and annual)

Subsetting seasonal scales

Now we want to subset four master dataframes that contains seasonal-scale data. We categorize the data into four different seasonal categories:

1. spring: 3,4,5
2. summer: 6,7,8
3. fall: 9,10,11
4. winter: 12,1,2

```
master.sum <- master.temp %>%
  filter(month == 6 | month == 7 | month == 8)

master.win <- master.temp %>%
  filter(month == 12 | month == 1 | month == 2)

master.spring <- master.temp %>%
  filter(month == 3 | month == 4 | month == 5)

master.fall <- master.temp %>%
  filter(month == 9 | month == 10 | month == 11)
```

Identify good years

Get years that contain more than 250 days of temperature record for each location.

```
## Annual (250 days)
fulyear <- good_year(master.temp, 250)

## $bigcreek
## [1] "2000" "2001" "2002" "2003" "2004" "2005" "2006" "2007" "2008" "2009"
## [11] "2012" "2013" "2014"
##
## $bigotter
## [1] "2012" "2013" "2014"
##
## $fox
## [1] "2012" "2013" "2014"
##
## $genesee
## [1] "2011" "2012" "2013"
##
## $humber
## [1] "1998" "1999" "2000" "2001" "2002" "2003" "2005" "2006" "2007" "2008"
## [11] "2009" "2011"
##
## $mississagi
## [1] "2011" "2012" "2013"
##
## $nipigon
## [1] "2008" "2009"
##
## $portage
## [1] "2011" "2012"
##
## $saginaw
## [1] "2014"
##
## $stlouis
## [1] "2012" "2013" "2014"
```

```

## 
## $vermillion
## [1] "2012" "2013" "2014"
## Seasonal (60 days)
fulspring <- good_year(master.spring, 60)

## $bigcreek
## [1] "2001" "2002" "2003" "2004" "2005" "2006" "2007" "2008" "2009" "2012"
## [11] "2013" "2014"
##
## $bigotter
## [1] "2012" "2013" "2014"
##
## $fox
## [1] "2011" "2012" "2013" "2014"
##
## $genesee
## [1] "2011" "2012" "2013"
##
## $humber
## [1] "1999" "2000" "2001" "2002" "2003" "2005" "2006" "2007" "2008" "2009"
## [11] "2011" "2013"
##
## $mississagi
## [1] "2011" "2013" "2014"
##
## $nipigon
## [1] "2008" "2009"
##
## $portage
## [1] "2011" "2012"
##
## $saginaw
## [1] "2013" "2014"
##
## $still
## [1] "2005" "2008"
##
## $stlouis
## [1] "2012" "2013"
##
## $vermillion
## [1] "2012" "2013" "2014"
fulsum <- good_year(master.sum, 60)

## $bigcreek
## [1] "2000" "2001" "2002" "2003" "2004" "2005" "2006" "2007" "2008" "2009"
## [11] "2012" "2013" "2014"
##
## $bigotter
## [1] "2012" "2013" "2014"
##
## $fox

```

```

## [1] "2011" "2012" "2013" "2014"
##
## $genesee
## [1] "2011" "2012" "2013"
##
## $humber
## [1] "1998" "1999" "2000" "2001" "2002" "2003" "2005" "2006" "2007" "2008"
## [11] "2009" "2011"
##
## $mississagi
## [1] "2011" "2012" "2013"
##
## $nipigon
## [1] "2008" "2009"
##
## $portage
## [1] "2011" "2012"
##
## $saginaw
## [1] "2014"
##
## $still
## [1] "2002" "2004" "2005" "2008"
##
## $stlouis
## [1] "2011" "2012" "2013" "2014"
##
## $vermilion
## [1] "2013" "2014"

fulfall <- good_year(master.fall, 60)

## $bigcreek
## [1] "2000" "2001" "2002" "2003" "2004" "2005" "2006" "2007" "2008" "2009"
## [11] "2012" "2013"
##
## $bigotter
## [1] "2012" "2013"
##
## $fox
## [1] "2011" "2012" "2013" "2014"
##
## $genesee
## [1] "2011" "2012" "2013"
##
## $humber
## [1] "1998" "1999" "2000" "2001" "2002" "2003" "2005" "2006" "2007" "2008"
## [11] "2009" "2011"
##
## $mississagi
## [1] "2010" "2011" "2012" "2013"
##
## $nipigon
## [1] "2008" "2009"
##

```

```

## $portage
## [1] "2011" "2012"
##
## $saginaw
## [1] "2012" "2014"
##
## $still
## [1] "2002" "2004"
##
## $stlouis
## [1] "2011" "2012" "2013" "2014"
##
## $vermilion
## [1] "2012" "2013" "2014"
fulwin <- good_year(master.win, 60)

## $bigcreek
## [1] "2001" "2002" "2004" "2005" "2006" "2007" "2008" "2009" "2012" "2013"
##
## $bigotter
## [1] "2013"
##
## $fox
## [1] "2012" "2013" "2014"
##
## $genesee
## [1] "2011" "2012" "2013"
##
## $humber
## [1] "1999" "2000" "2001" "2002" "2003" "2005" "2006" "2007" "2008" "2009"
## [11] "2011"
##
## $mississagi
## [1] "2011" "2012" "2013"
##
## $nipigon
## [1] "2008" "2009"
##
## $portage
## [1] "2011" "2012"
##
## $saginaw
## [1] "2012" "2014"
##
## $stlouis
## [1] "2012"
##
## $vermilion
## [1] "2012" "2013"

```

Get training and testing

Get training and testing for the specific season, and for annual. We store them in one master dataframe called grand_training and grand_testing, each is a list of five sub-temporal data.

```

fold = 10

## Get training and testing for annual
annual <- get.taintest(master.temp, fulyear, 10)

combined_training_list <- annual[[1]]
combined_testing_list <- annual[[2]]


## Get training and testing for each season
sum <- get.taintest(master.sum, fulsum, 10)
win <- get.taintest(master.win, fulwin, 10)
spr <- get.taintest(master.spring, fulspring, 10)
fall <- get.taintest(master.fall, fulfall, 10)

combined_training_list_sp <- spr[[1]]
combined_testing_list_sp <- spr[[2]]

combined_training_list_su <- sum[[1]]
combined_testing_list_su <- sum[[2]]

combined_training_list_fa <- fall[[1]]
combined_testing_list_fa <- fall[[2]]

combined_training_list_w <- win[[1]]
combined_testing_list_w <- win[[2]]


## Create a list to store all the combined training and testing lists
grand_training <- list(combined_training_list_sp, combined_training_list_su,
                      combined_training_list_fa, combined_training_list_w,
                      combined_training_list)

grand_testing <- list(combined_testing_list_sp, combined_testing_list_su,
                      combined_testing_list_fa, combined_testing_list_w,
                      combined_testing_list)

```

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

Data plots

Temperature plot

Before we run any analysis, we first want to check whether the air temperature and lags are correctly calculated. Here, we plot the lag1 air temperature, lag5 air temperature, and water temperature verus the times series for each location.

Notice here NA values are indicated by gaps in the plots.

```

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

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

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

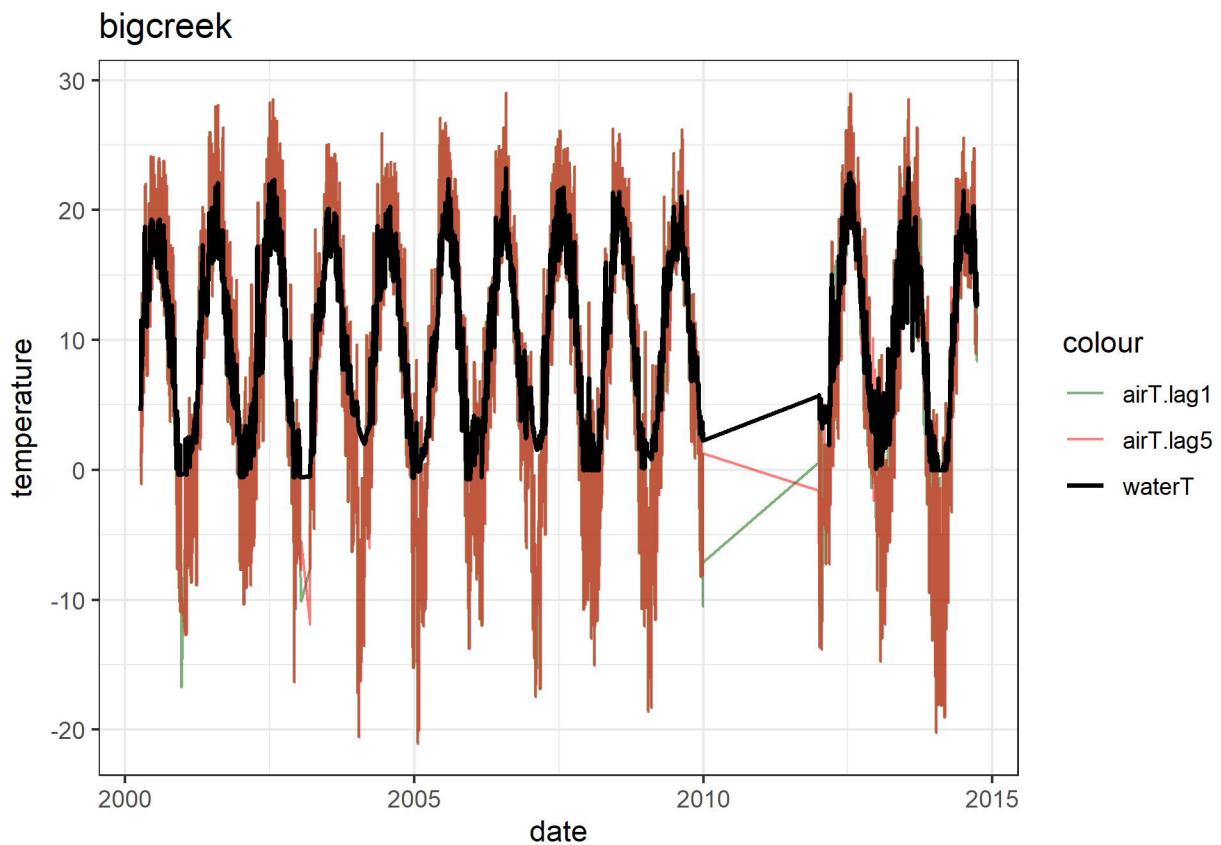
```

```

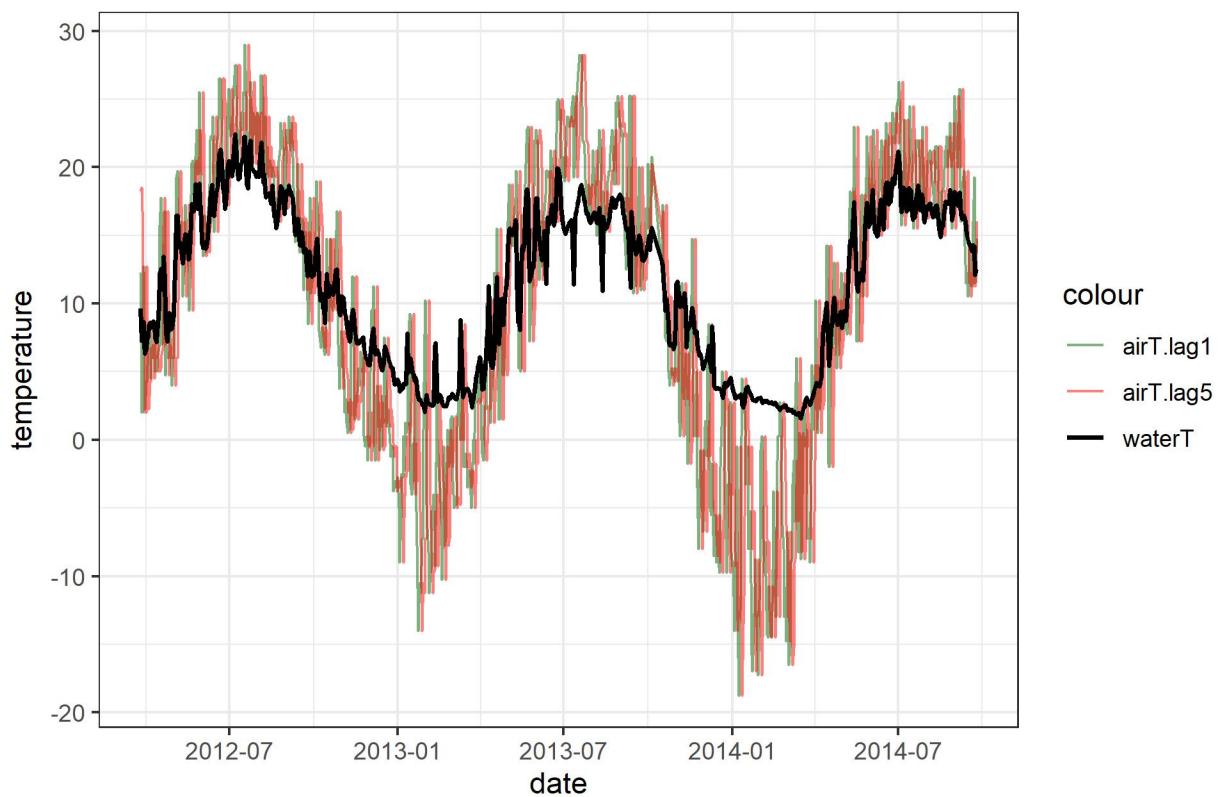
pl <- ggplot(plot.df, aes(x = date))+
  geom_line(aes(y = dmean_1, color = "airT.lag1"), alpha = 0.5)+
  geom_line(aes(y = dmean_5, color = "airT.lag5"), alpha = 0.5)+
  geom_line(aes(y = water, color = "waterT"), linewidth = 0.8)+
  labs(x="date", y="temperature")+
  ggtitle(paste(loc))+
  scale_color_manual(values = colors)+
  theme_bw()

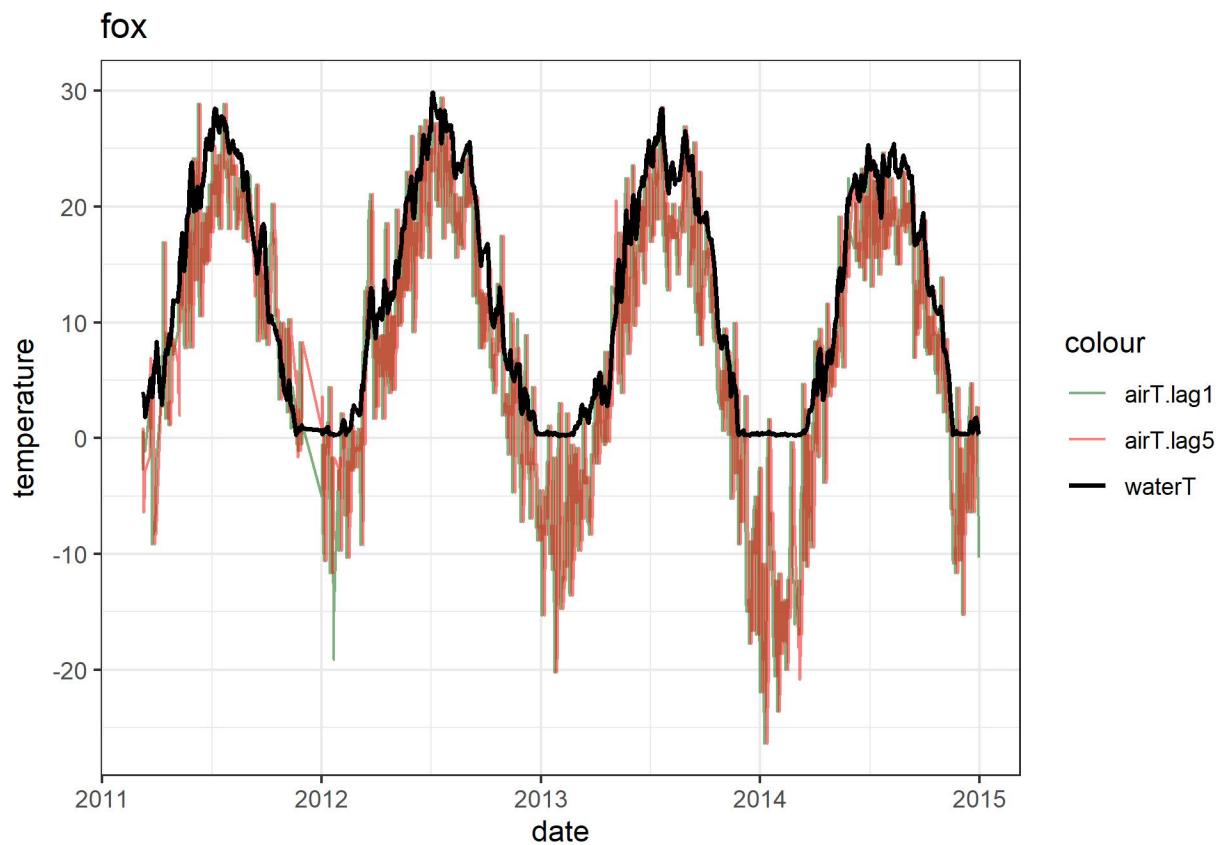
print(pl)
}

```

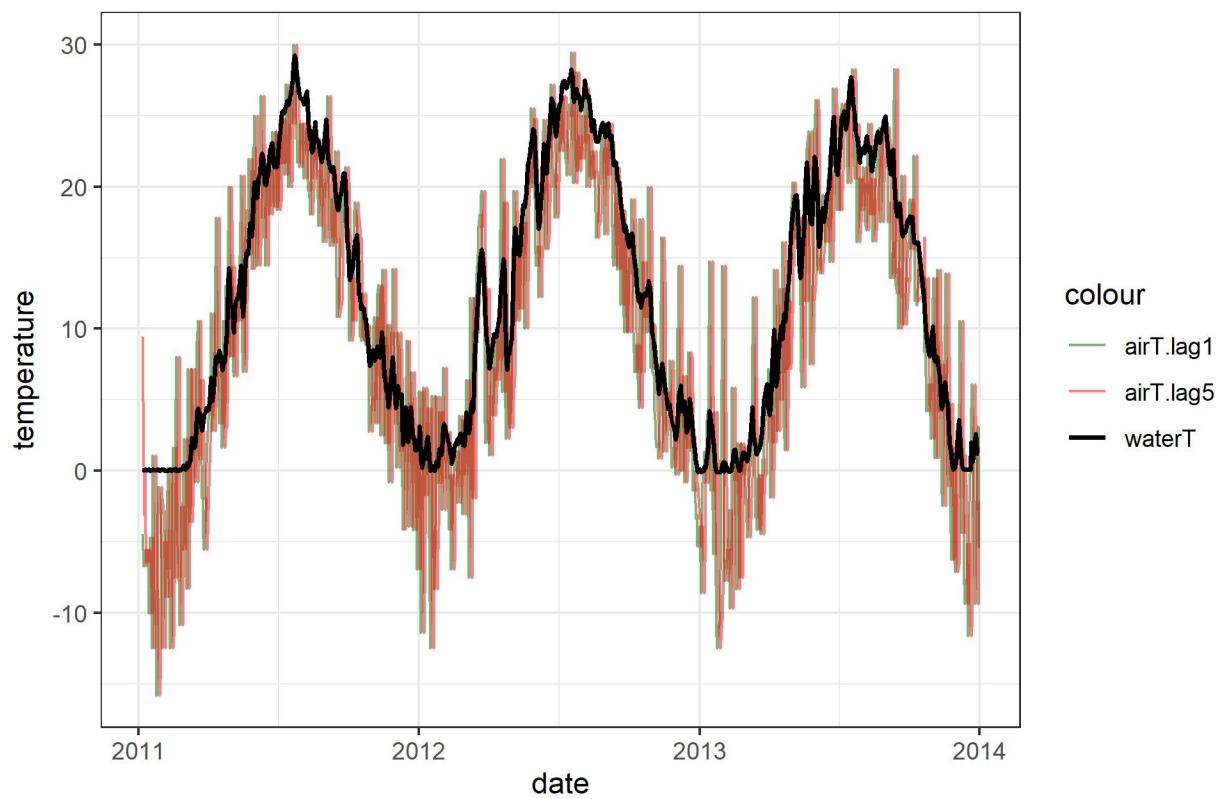


bigotter

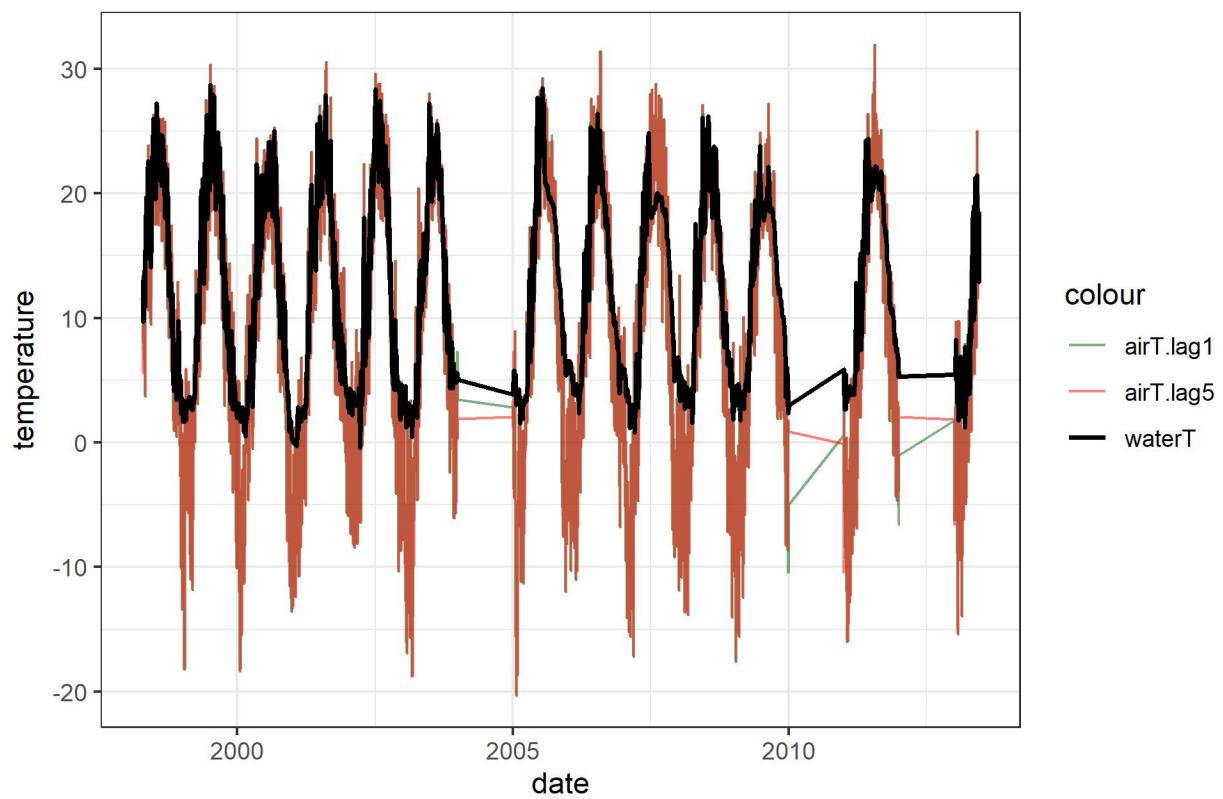




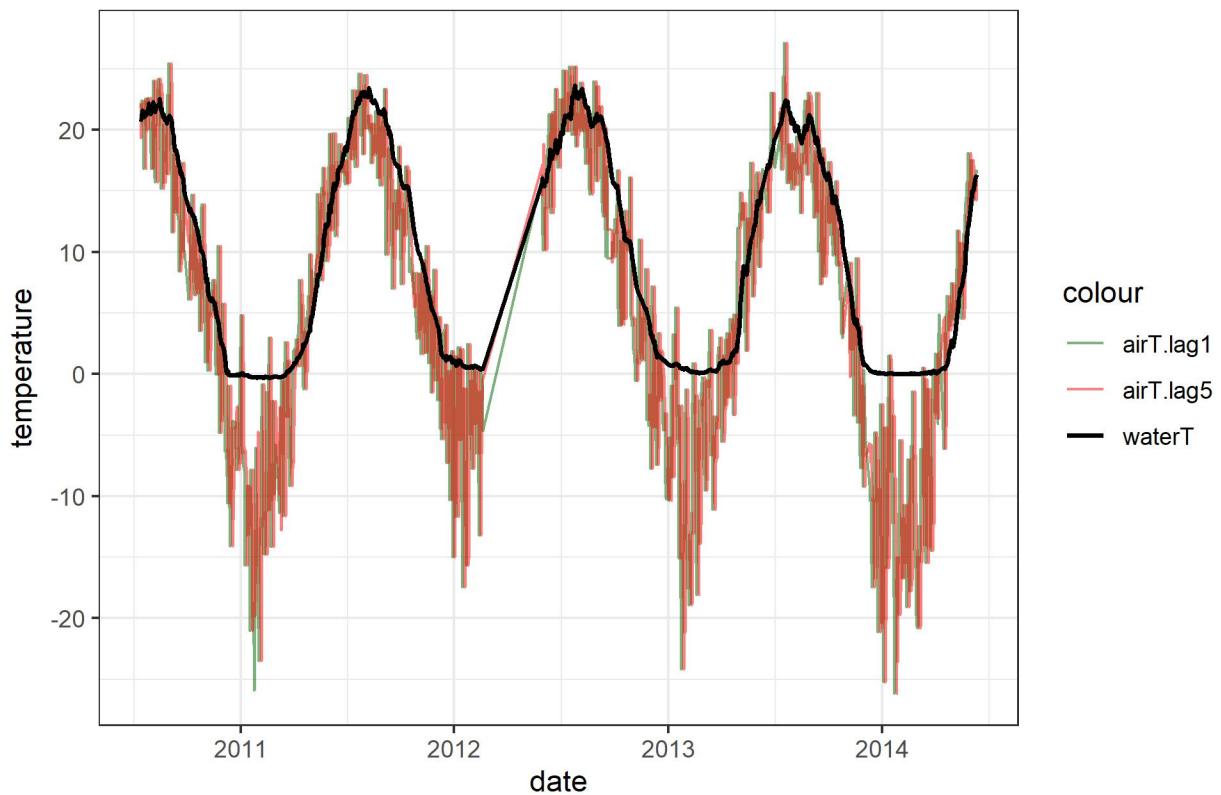
genesee



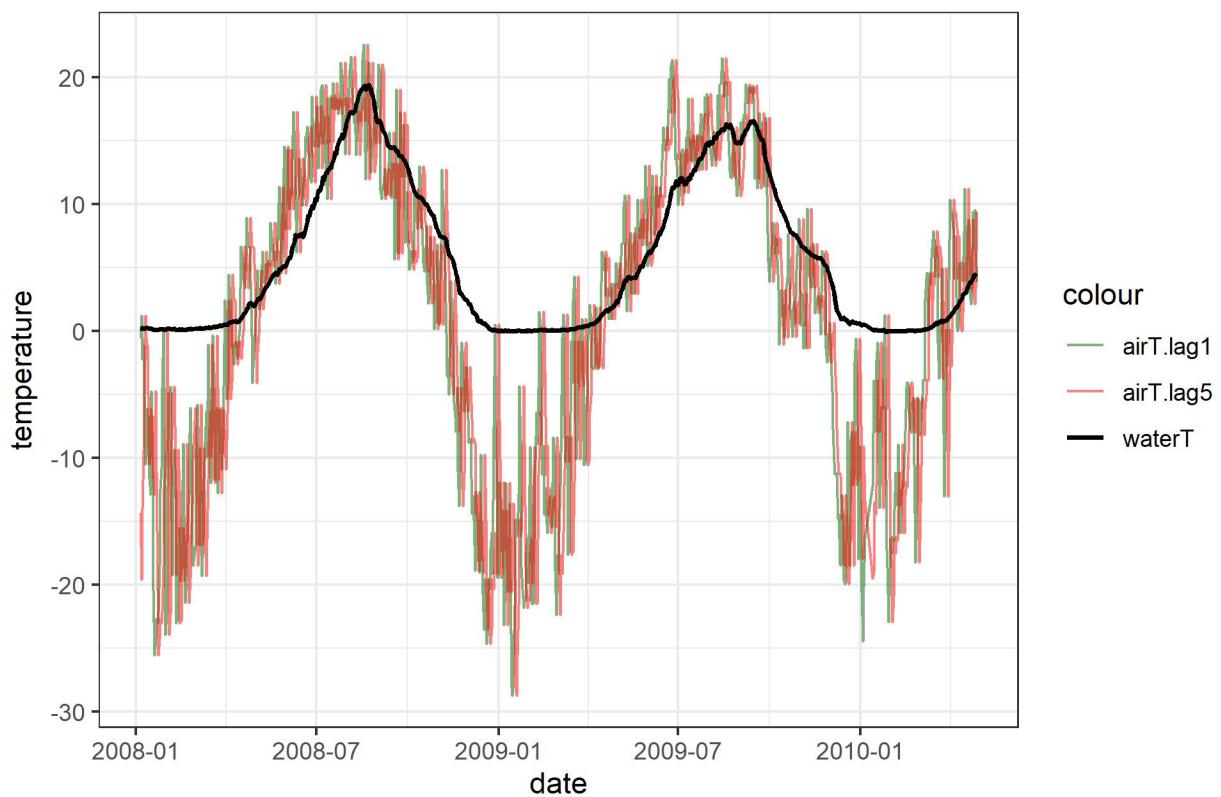
humber



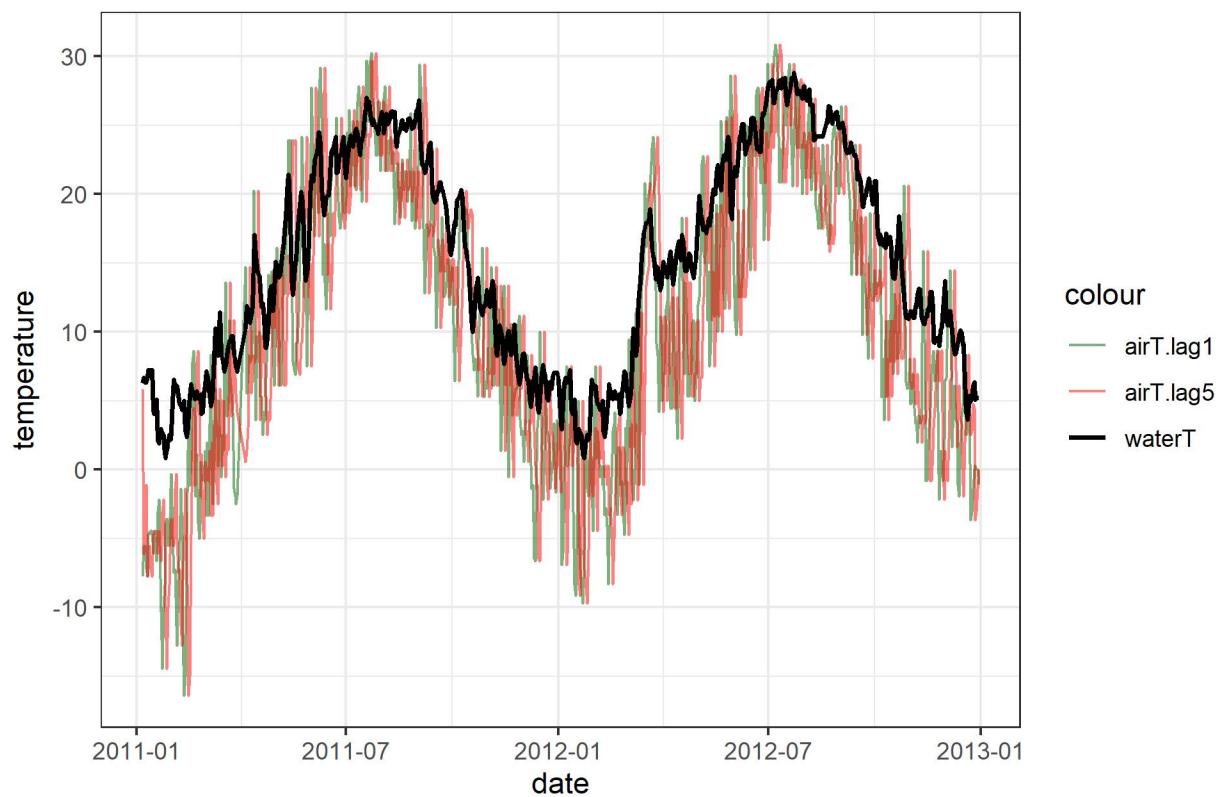
mississagi



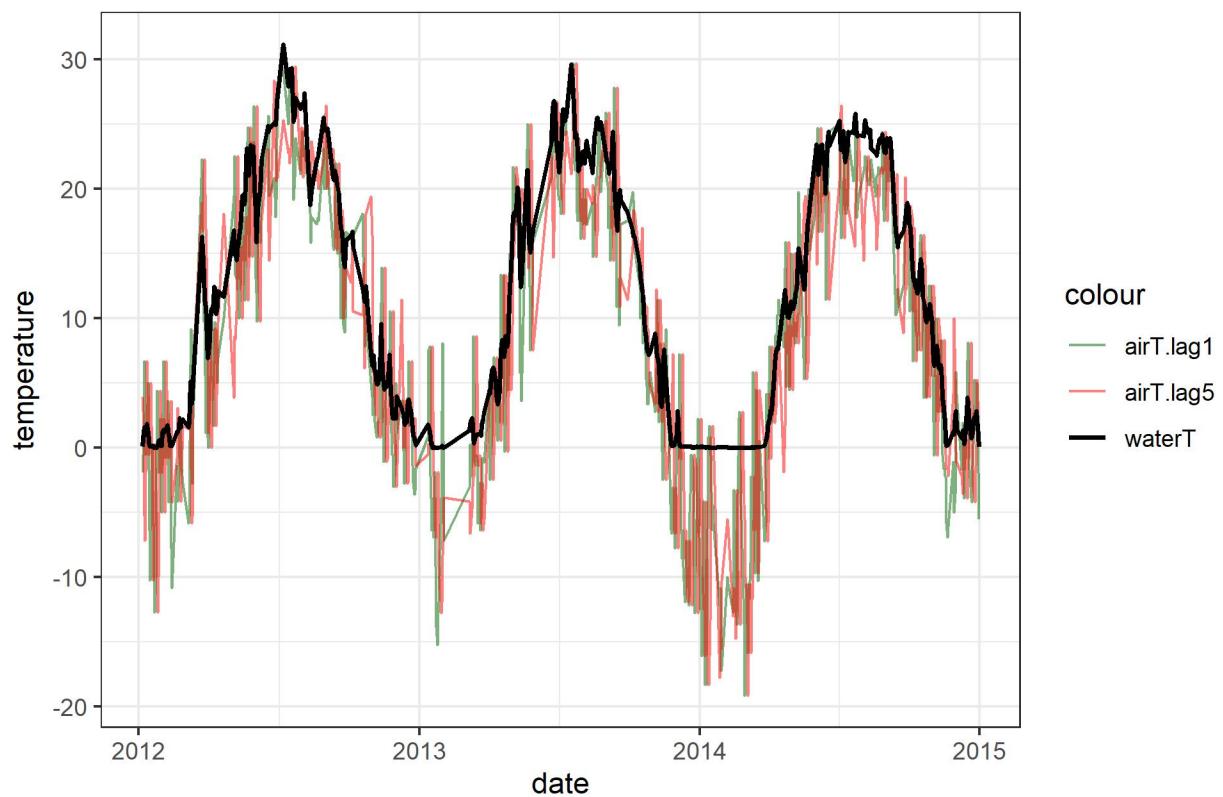
nipigon



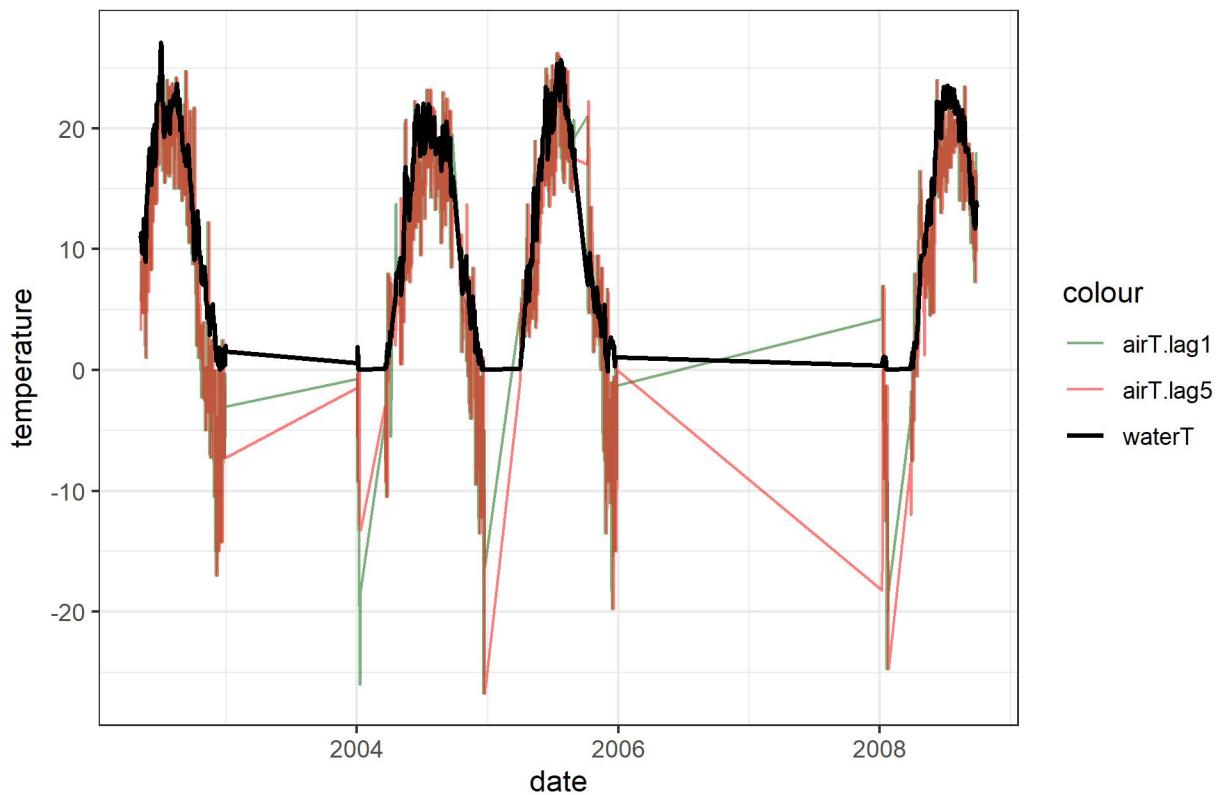
portage



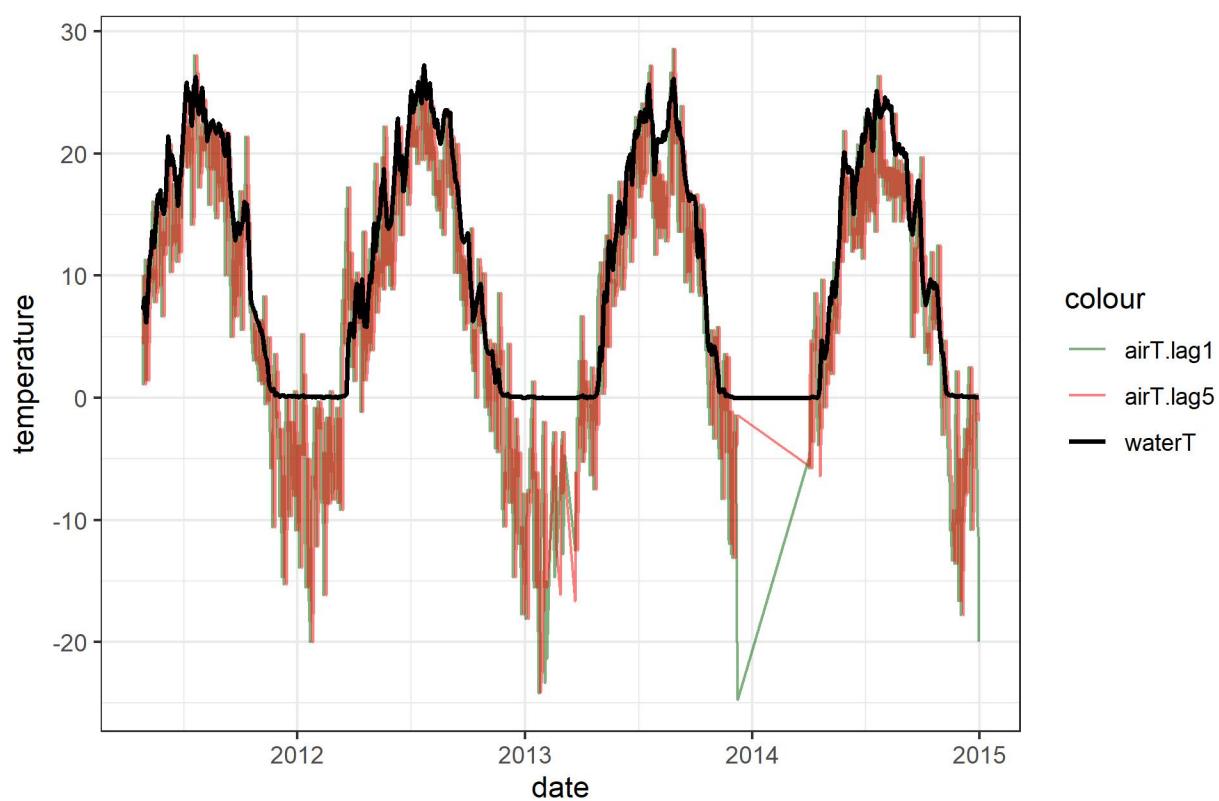
saginaw

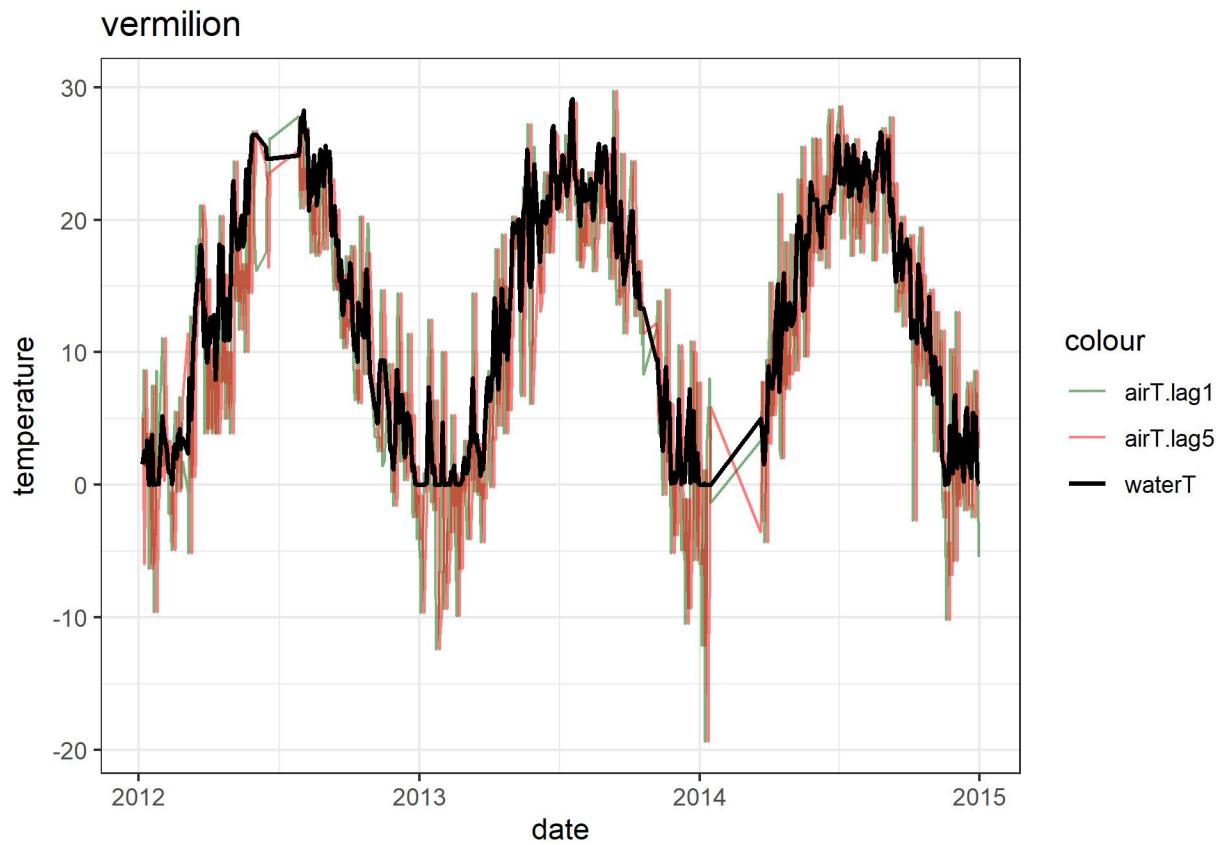


still



stlouis



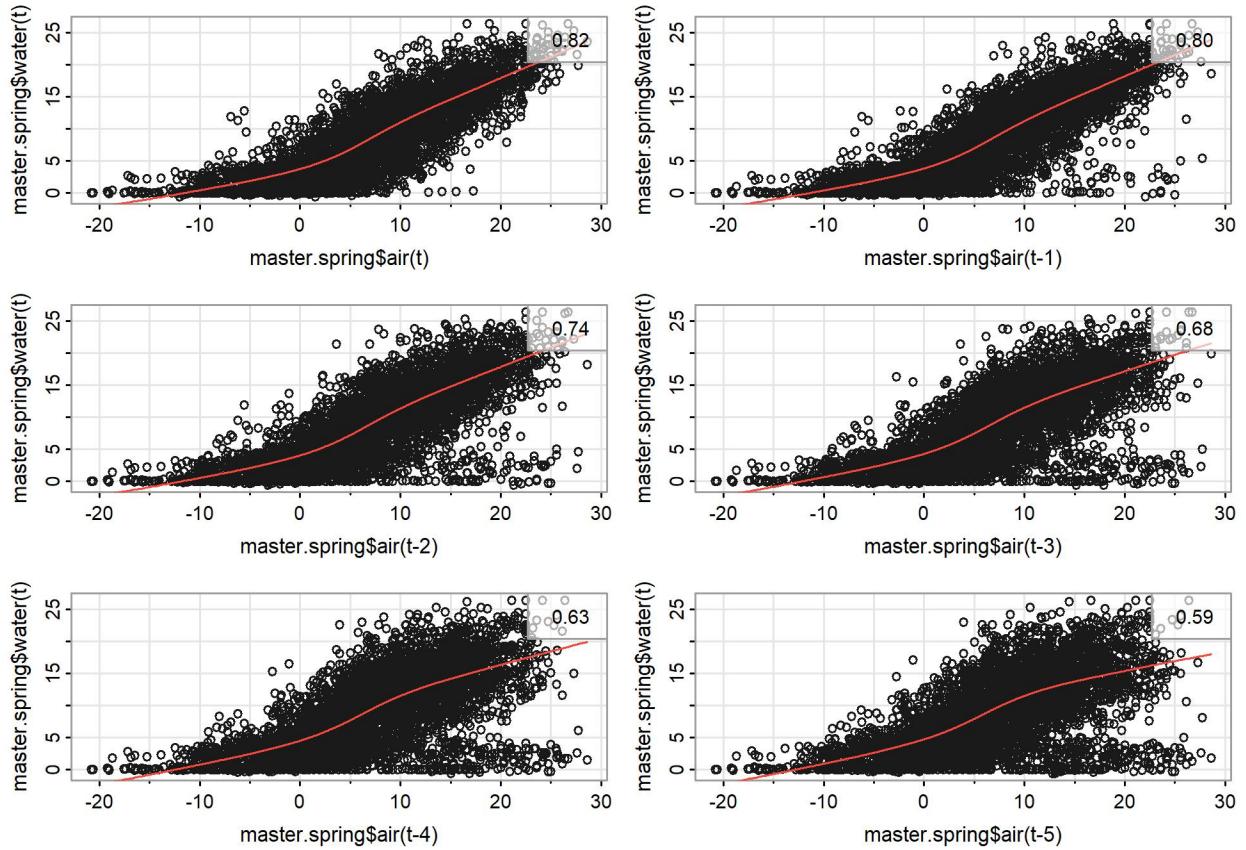


It seems that the lagged day temperatures are working fine.

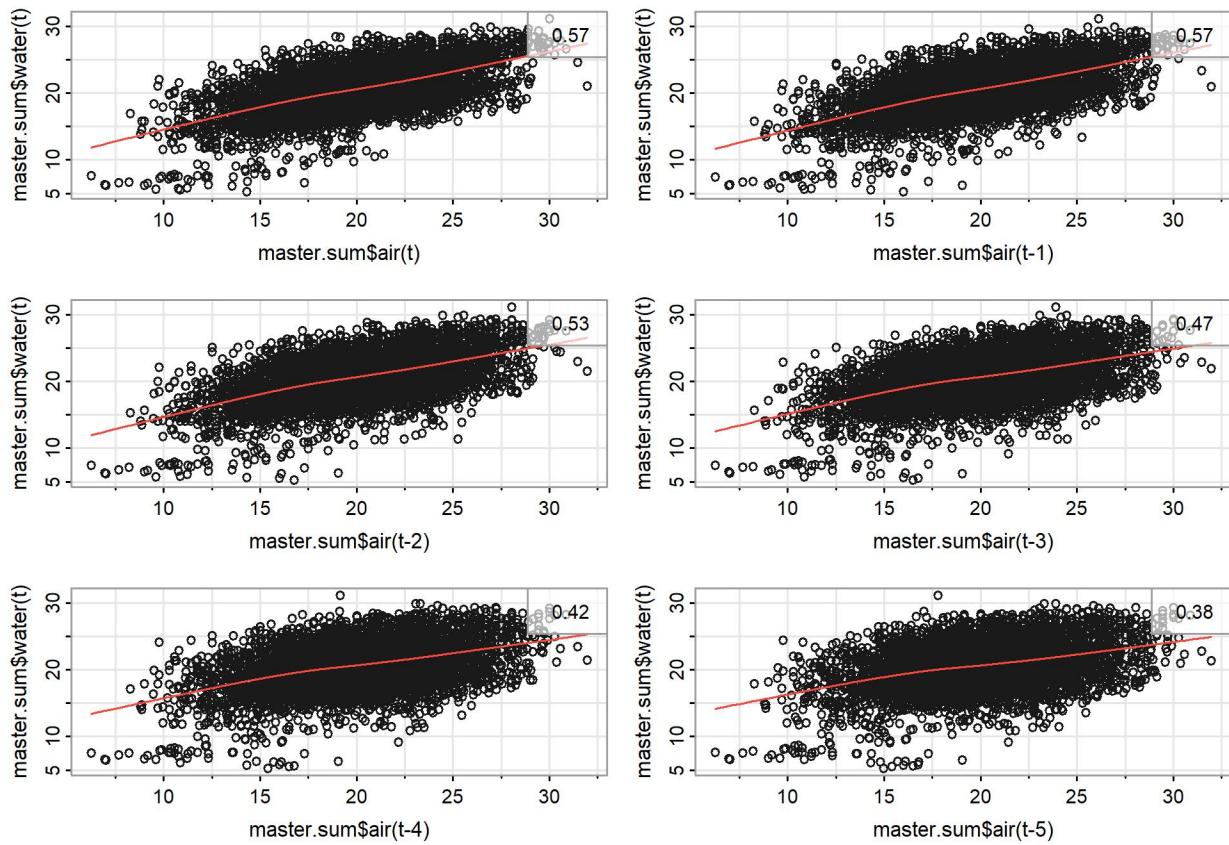
Lagged scatterplot

Then, we want to use the relationship between our response variables (water temperature), and our predictors (air temperature at various lags) for each season. All locations are plotted on one graph for easier demonstration.

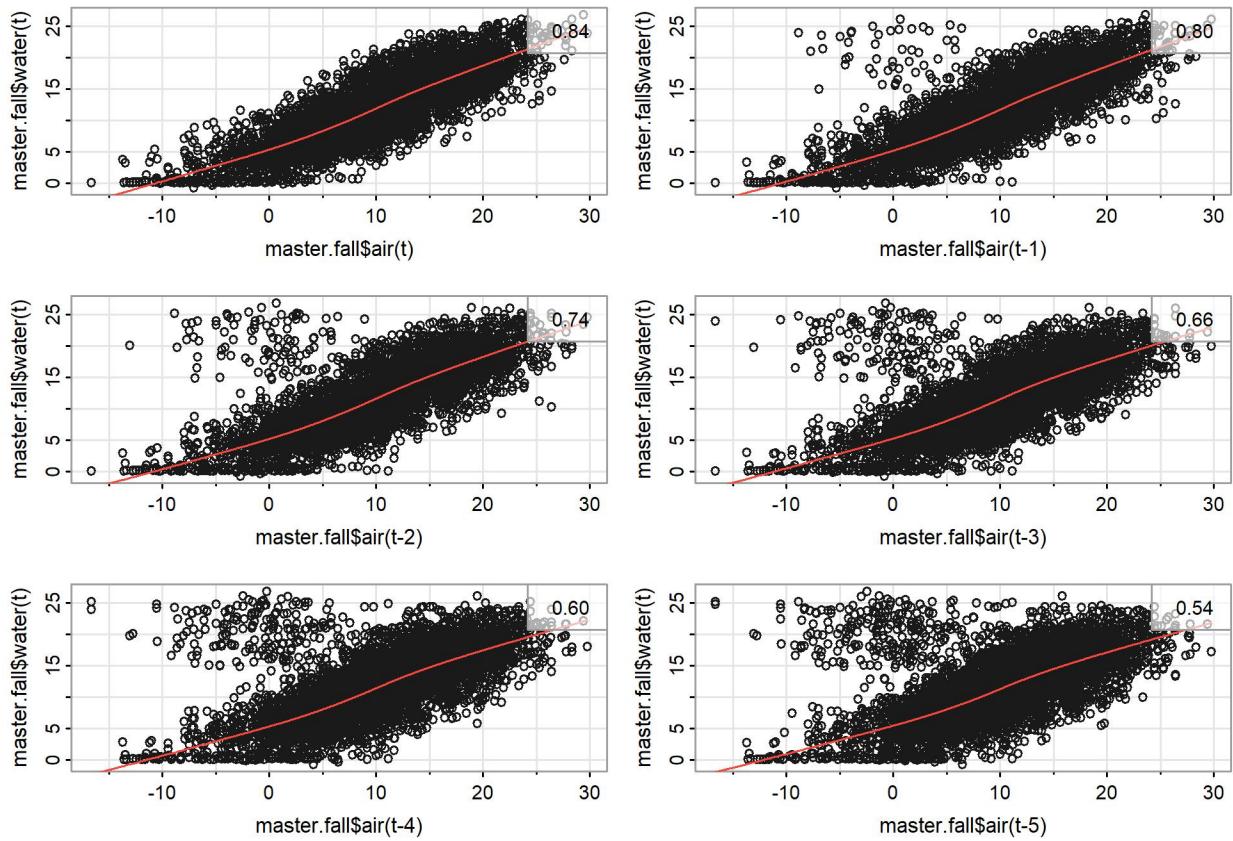
```
lag2.plot(master.spring$air, master.spring$water, 5)
```



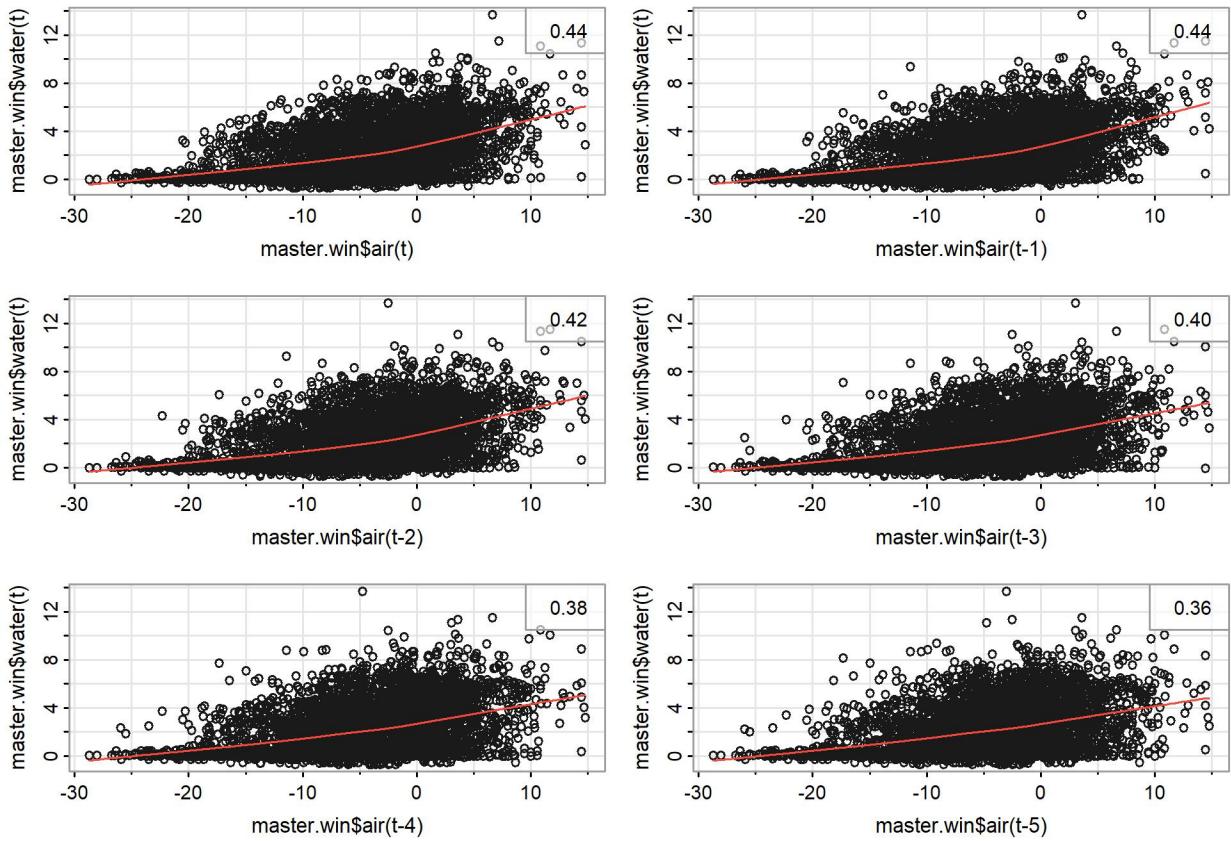
```
lag2.plot(master.sum$air, master.sum$water, 5)
```



```
lag2.plot(master.fall$air, master.fall$water, 5)
```



```
lag2.plot(master.win$air, master.win$water, 5)
```



Specify the season

Here, we use this code to identify the specific season used to run model and check temporal autocorrelation.

Note: season 1-5: spring, summer, fall, winter, annual

```
i = 1
season = 1
ctrl = lmeControl(opt='optim')

# select current dataset, and all unique location levels
current.training <- grand_training[[season]][[i]] %>% arrange(location, date)
current.testing <- grand_testing[[season]][[i]] %>% arrange(location, date)

# option to subset annual training/testing without winter
#current.training <- combined_training_list[[i]] %>% arrange(location, date) %>% filter(month != 12 & month != 1)
#current.testing <- combined_testing_list[[i]] %>% arrange(location, date) %>% filter(month != 12 & month != 1)
```

Multiple linear regression models with lag

Linear regression lag 3

```
## Forms
form.locrandom <- water ~ air + dmean_1 + dmean_2
compare <- NA
```

```

# model training and predicting
model <- lme(form.locrandom,
              random = ~1 | location,
              na.action = na.omit, data = current.training)

current.testing$preds <- as.vector(predict(model, newdata = current.testing))
current.training$preds <- as.vector(predict(model))

# Include AR term
model.ar <- lme(form.locrandom,
                  random = ~1 | location,
                  na.action = na.omit, data = current.training,
                  correlation=corAR1(form=~1|location))

current.testing$preds.ar <- as.vector(predict(model.ar, newdata = current.testing, re.form = ~1|location))
current.training$preds.ar <- as.vector(predict(model.ar, re.form = ~1|location))

## compare dataframe output
comp.fit <- current.training %>%
  select(location, year, date, obs = water, preds, preds.ar)

comp.pred <- current.testing %>%
  select(location, year, date, obs = water, 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
for (loc in loc_seq) {
  plot.df <- comp.fit[comp.fit$location == loc,]
  diff <- round(sqrt(mean((plot.df$preds-plot.df$obs)^2)),2)
  diff.ar <- round(sqrt(mean((plot.df$preds.ar-plot.df$obs)^2)),2)

  pl <- 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")+
    ggtitle(paste(
      loc, ": ", plot.df$year, " (RMSE:", diff, "&", diff.ar, ")"))
  print(pl)
}

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

anova(model, model.ar)

```

Linear lag 3 model definitely does not perform better compared to the linear lag 5 model.

Linear regression lag 5

Now, let's use a multiple linear (lag 5) mixed model with location as random effect.

```
## Forms
form.locrandom <- water ~ air + dmean_1 + dmean_2 + dmean_3 + dmean_4 + dmean_5
compare <- current.testing

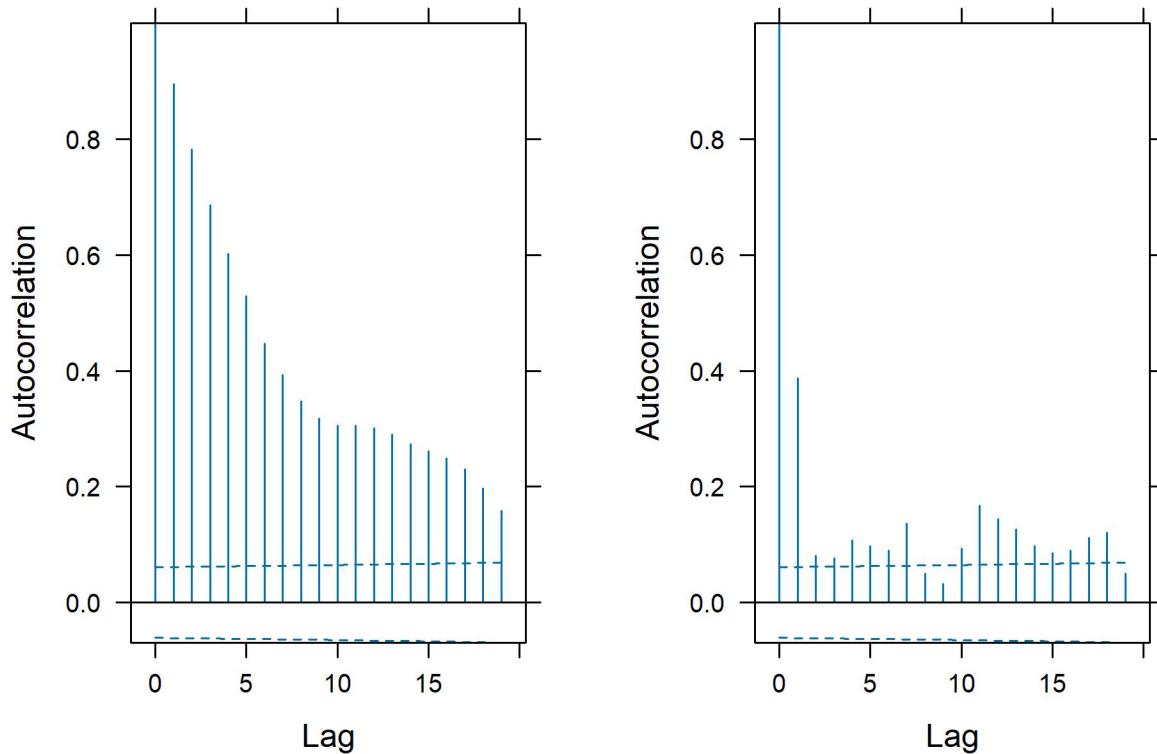
# model training and predicting
model <- lme(form.locrandom,
              random = ~1 | location,
              na.action = na.omit, data = current.training)

compare$preds <- as.vector(predict(model, newdata = current.testing))

# Include AR term
model.ar <- lme(form.locrandom,
                  random = ~1 | location, control = ctrl,
                  na.action = na.omit, data = current.training,
                  correlation=corAR1(form=~1|location,0.85,fixed=T))

compare$preds.ar <- as.vector(predict(model.ar, newdata = current.testing, re.form = ~1|location))

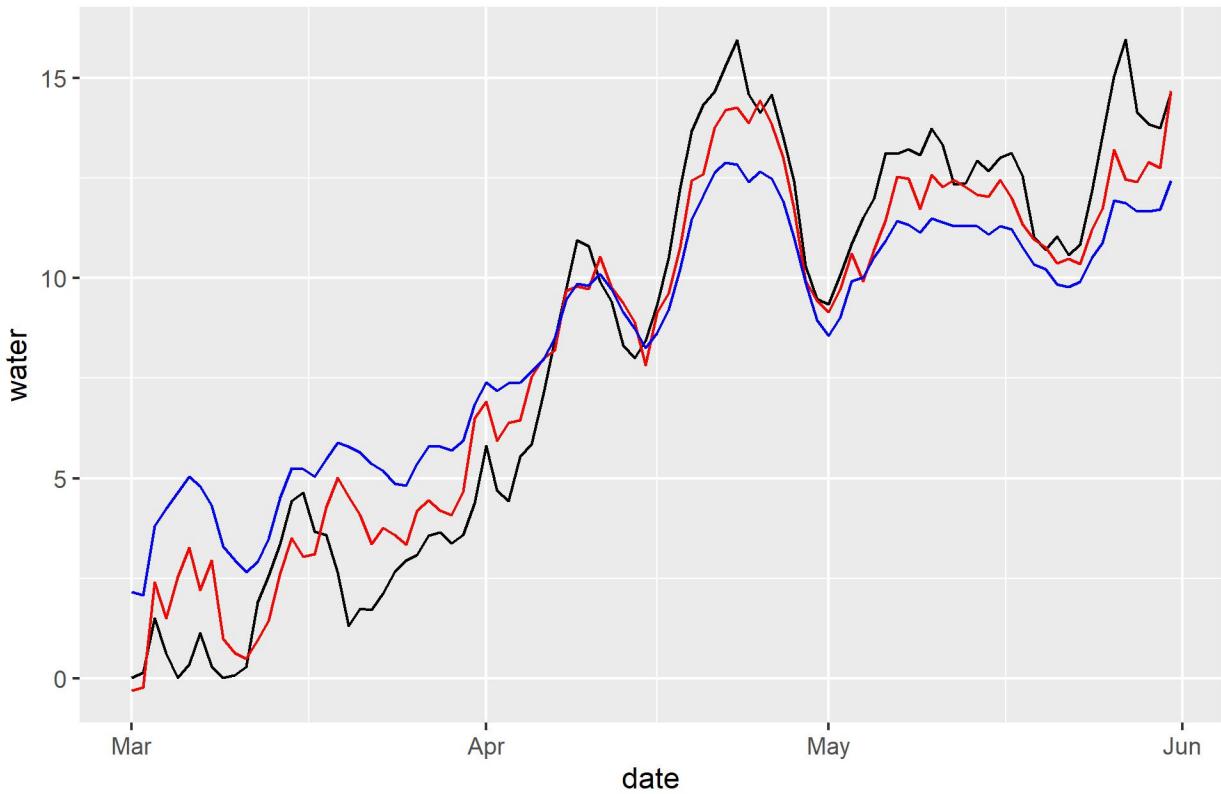
## Plot ACF
ggarrange(plot(ACF(model,resType="normalized"),alpha=0.05),
          plot(ACF(model.ar,resType="normalized"),alpha=0.05),
          nrow = 1)
```



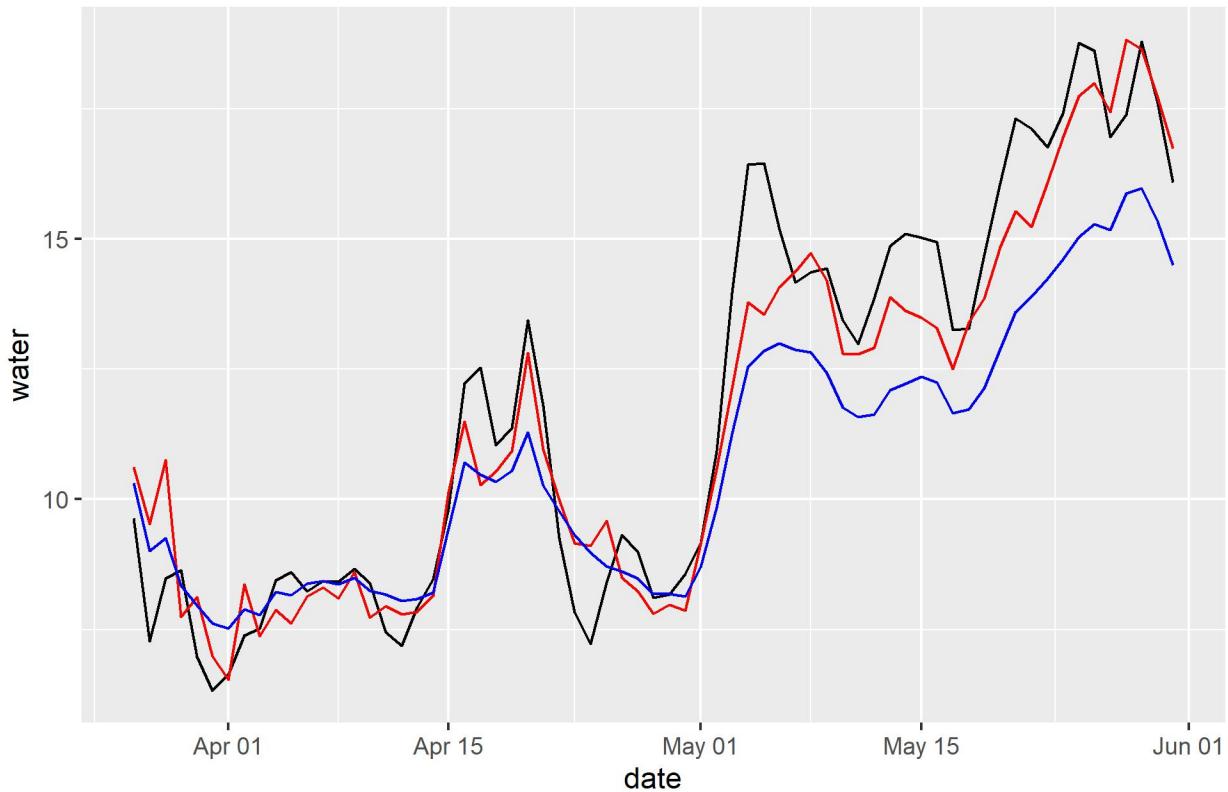
```
## Plots
for (loc in loc_seq) {
  plot.df <- compare[compare$location == loc,]
  diff <- round(sqrt(mean((plot.df$preds - plot.df$water)^2)),2)
  diff.ar <- round(sqrt(mean((plot.df$preds.ar - plot.df$water)^2)),2)

  pl <- ggplot(data=plot.df, aes(x=date))+
    geom_line(aes(x=date, y=water), color = "black")+
    geom_line(aes(x=date, y=preds), color = "red")+
    geom_line(aes(x=date, y=preds.ar), color = "blue")+
    ggtitle(paste(
      loc, ":", plot.df$year, " (RMSE:", diff, "&", diff.ar, ")"))
  print(pl)
}
```

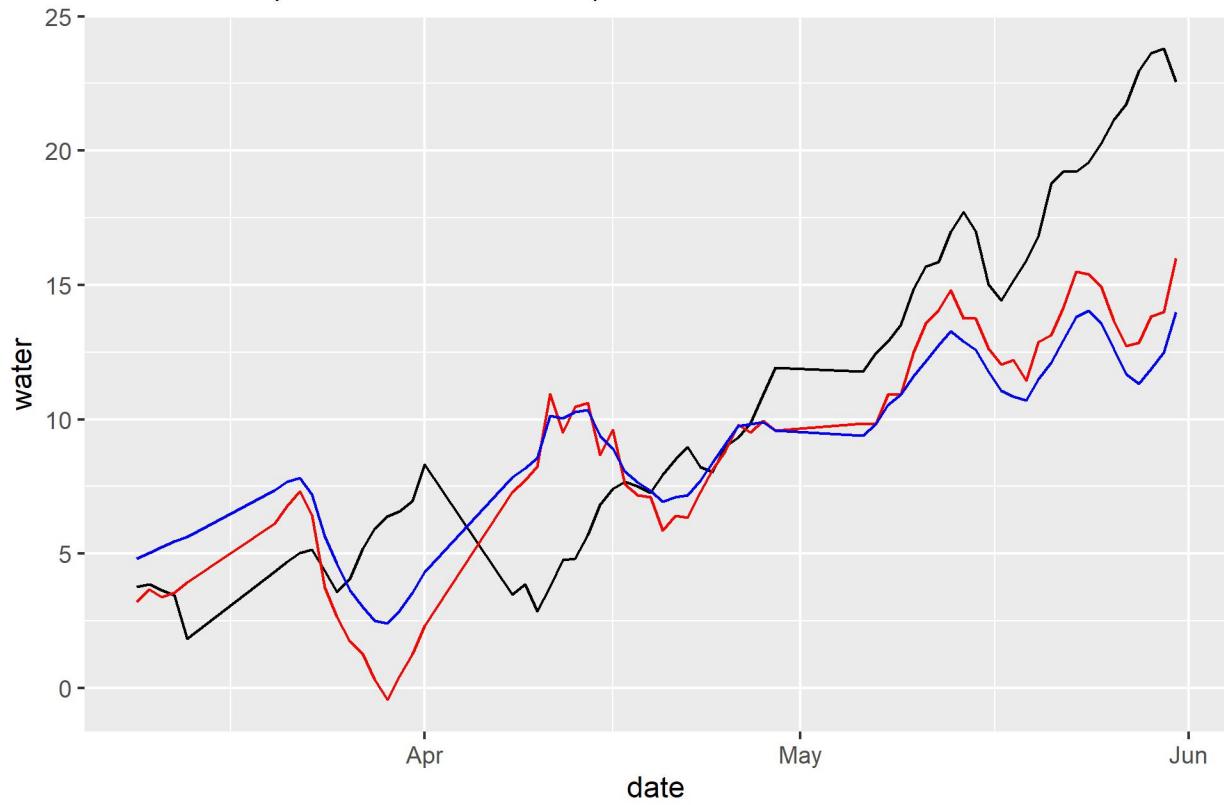
bigcreek : 2008 (RMSE: 1.24 & 2.17)



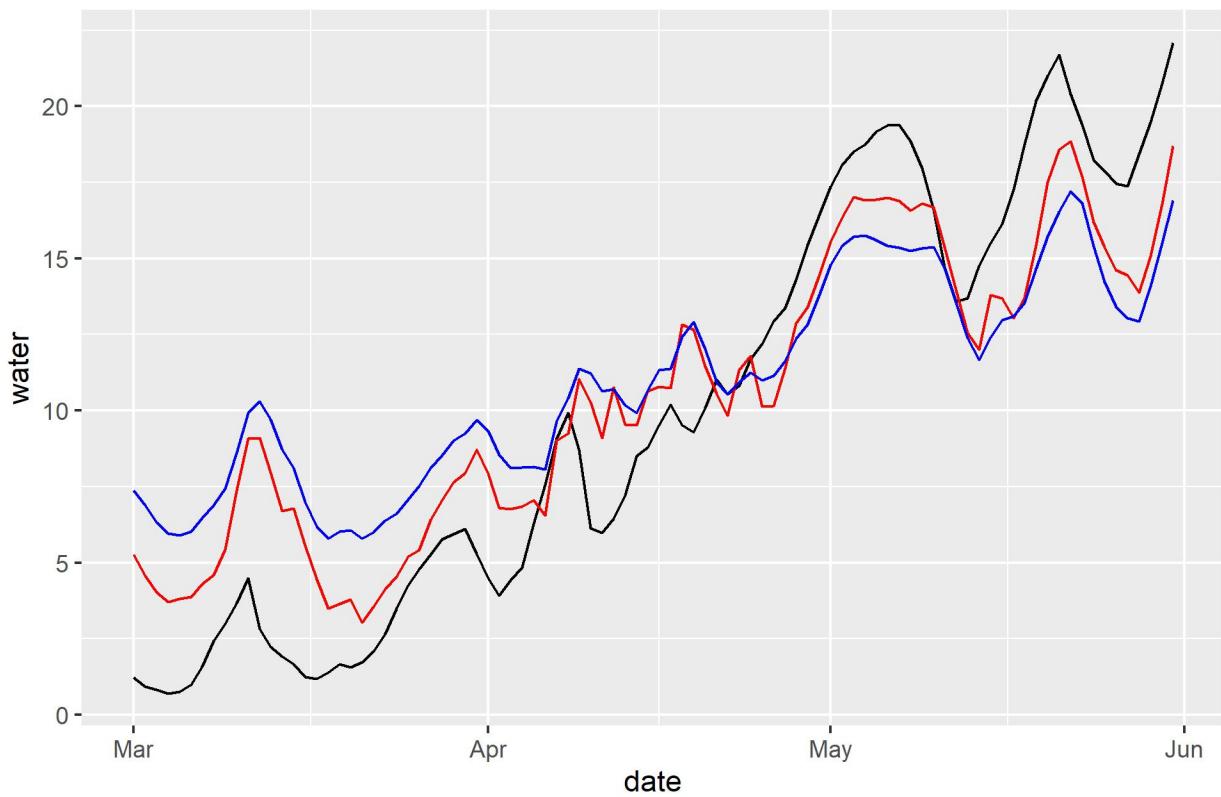
bigotter : 2012 (RMSE: 1.08 & 1.84)



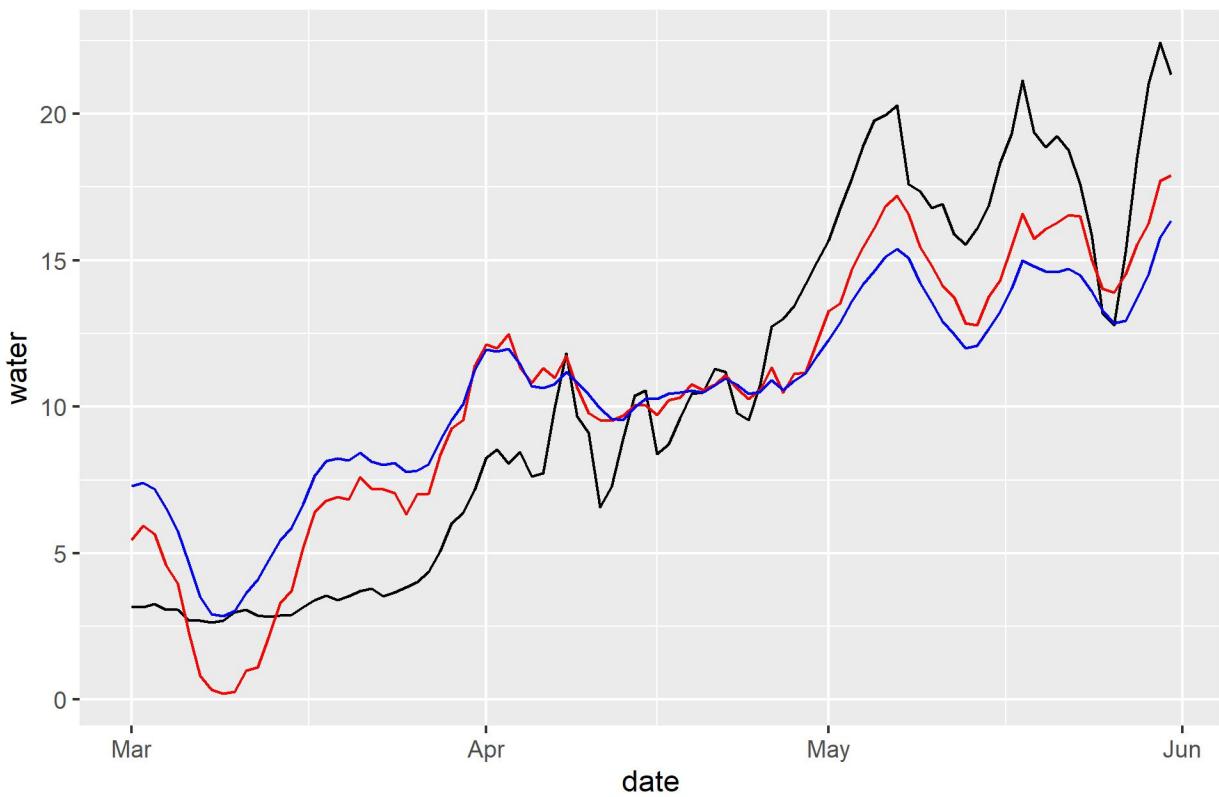
fox : 2011 (RMSE: 4.19 & 4.59)



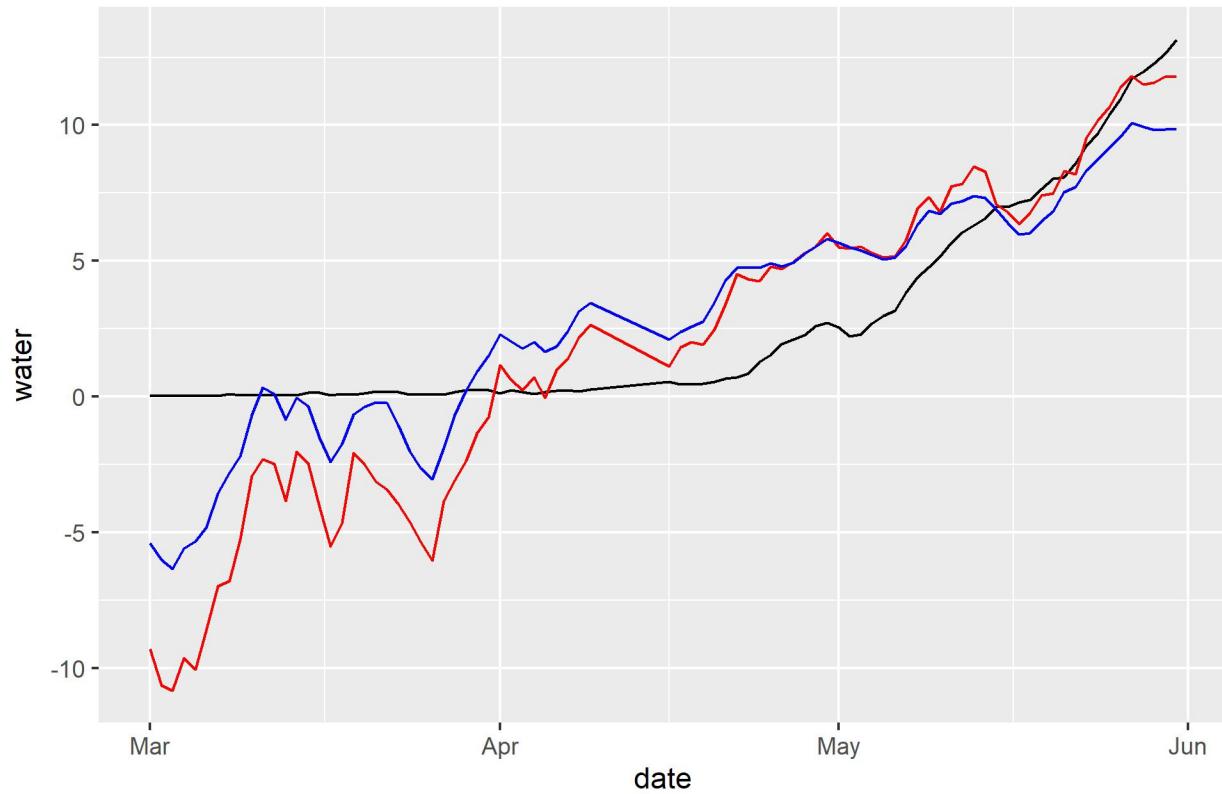
genesee : 2013 (RMSE: 2.76 & 3.9)



number : 1999 (RMSE: 2.62 & 3.41)



mississagi : 2014 (RMSE: 3.76 & 2.52)



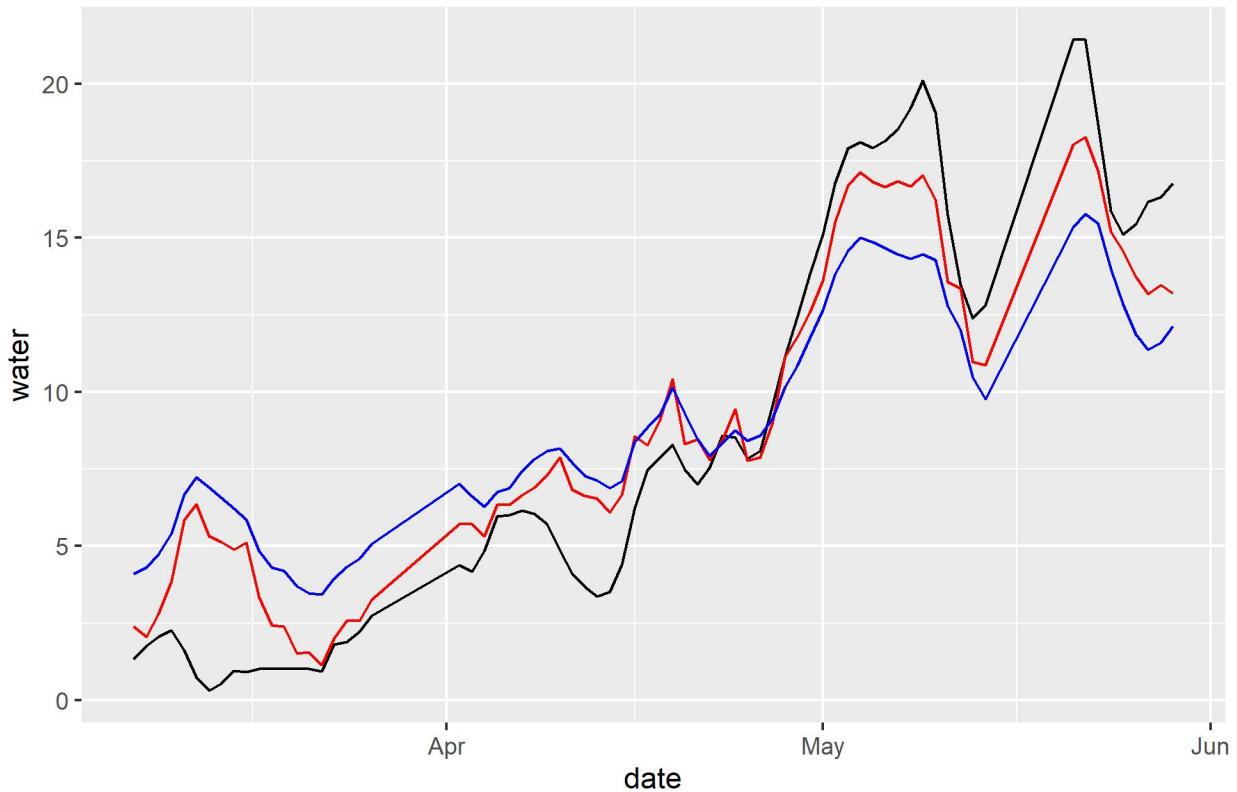
nipigon : 2009 (RMSE: 2.93 & 2.01)



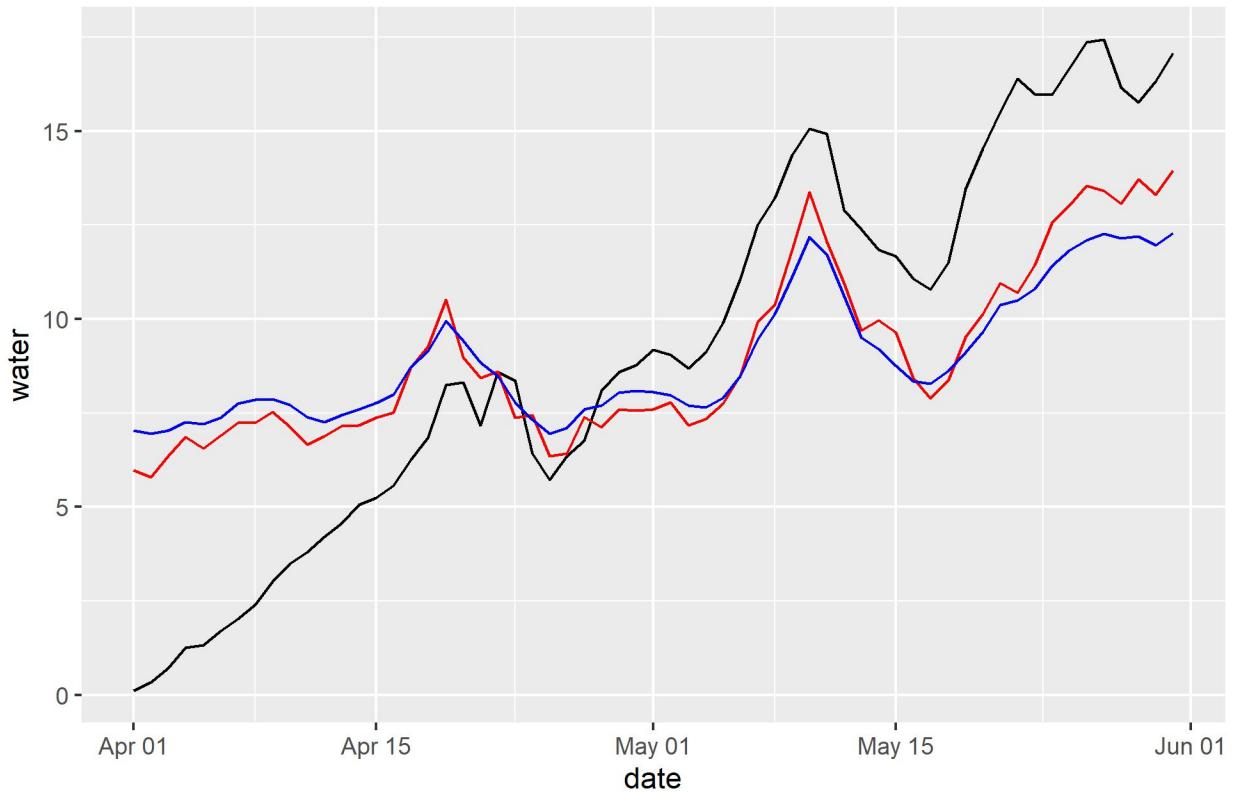
portage : 2012 (RMSE: 3.01 & 3.44)



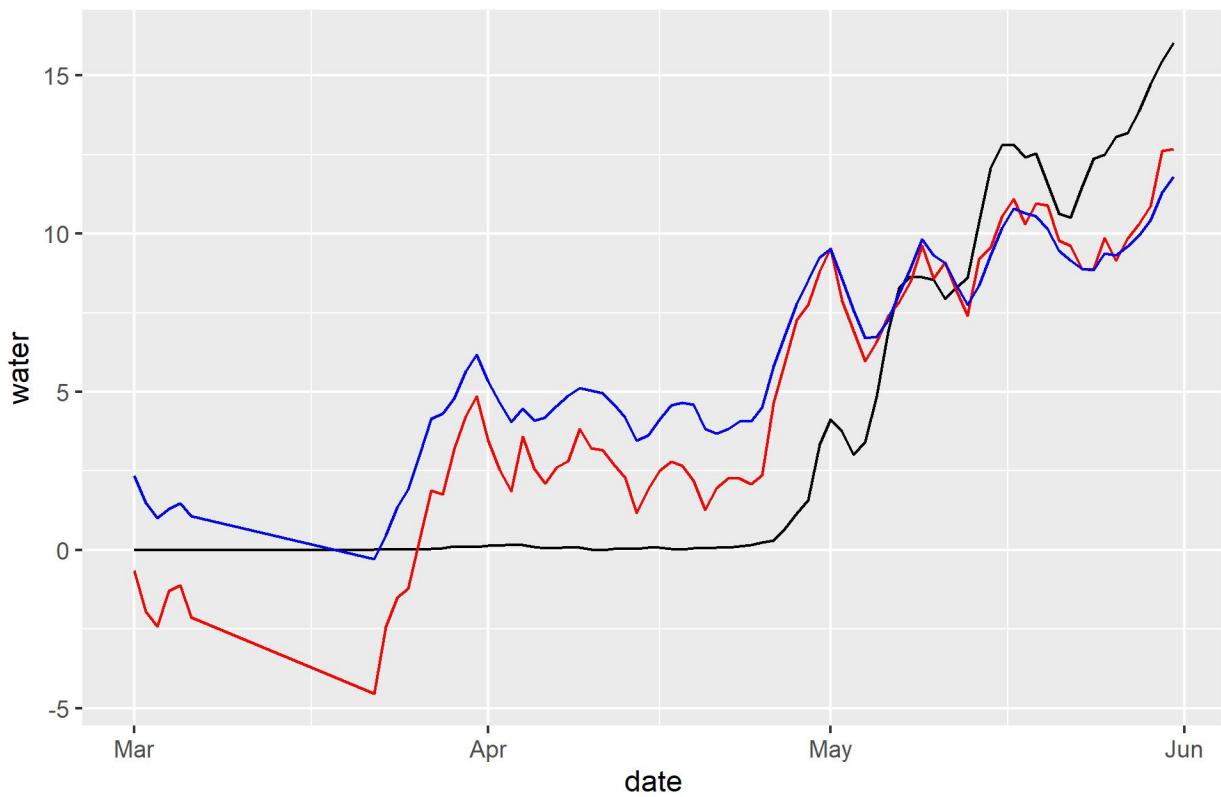
saginaw : 2013 (RMSE: 2.11 & 3.27)



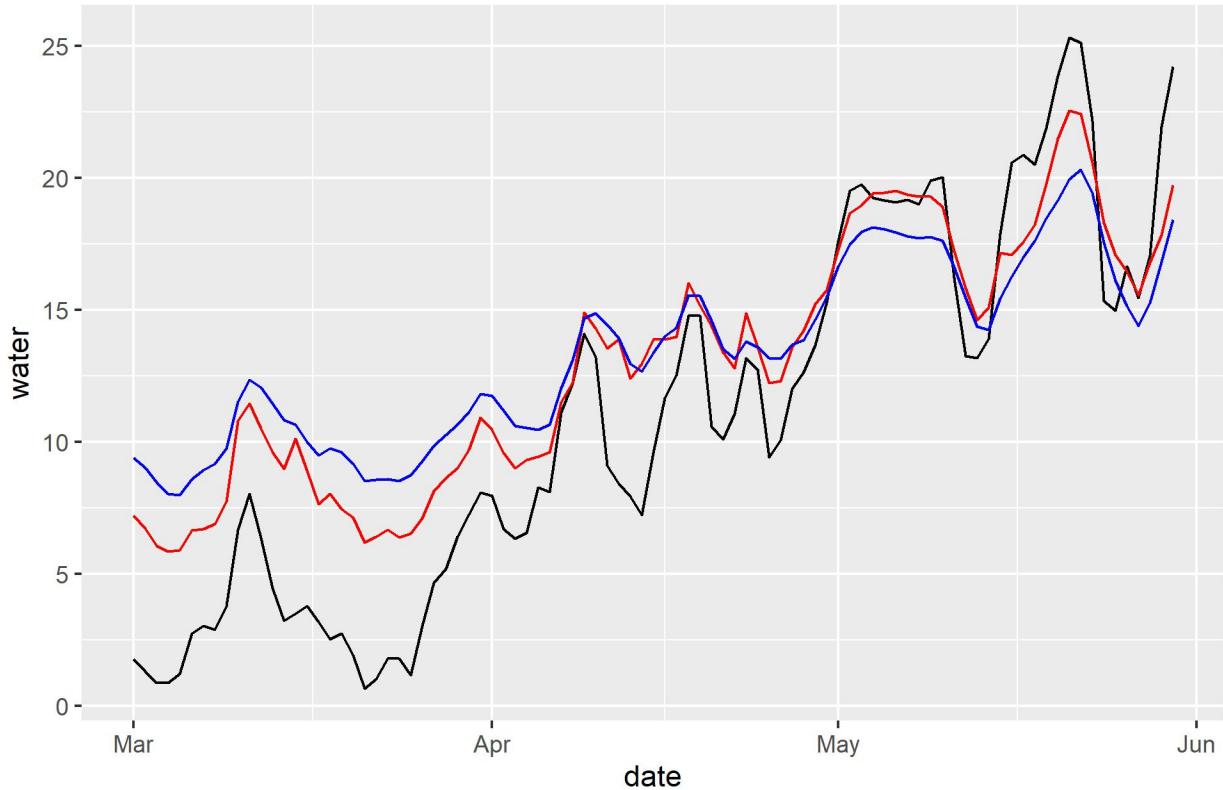
still : 2005 (RMSE: 3.19 & 3.68)



stlouis : 2013 (RMSE: 2.81 & 3.72)



vermillion : 2013 (RMSE: 3.31 & 4.46)



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

anova(model, model.ar)
```

```
##           Model df      AIC      BIC    logLik
## model       1 9 5054.700 5099.101 -2518.350
## model.ar   2 9 3077.309 3121.710 -1529.654
```

1. Temporal autocorrelation: The autocorrelation in summer and winter is reduced to a decent amount. In spring and fall, it has some degree of autocorrelation at day 6-10, but low at day 5.
2. The winter prediction is relatively good, but worse than simply using the mean. Also, it does not capture large daily variation. Meanwhile, residuals at two ends seems not normally distributed.

Reasonal residual

```
compare <- NA
current.testing <- grand_testing[[season]][[i]] %>% arrange(location, date)

## TRAINING
# get the model annual component
annual.comp <- nls(air ~ 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$air))))
# dataframe to store the residuals
colnames(res) <- c("res.t", "location")
res[,"location"] <- na.omit(current.training$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(water ~ 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,
                      control = ctrl)

residual.comp.ar <- lme(fixed = res.w ~ res.t + res.t1 + res.t2,
                        random = ~ 1|location,
                        correlation = corAR1(form=~1|location,0.85,fixed=T),
                        data = res, na.action = na.omit,
                        control = ctrl)

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

# Residuals data
res <- as.data.frame(matrix(NA, ncol = 3,
                           nrow = length(current.testing$air))) #residuals
colnames(res) <- c("res.t", "location", "date")
res[,"location"] <- current.testing$location
res[,"date"] <- current.testing$date
res[,"res.t"] <- current.testing$air - preds.annual$preds.annual
res <- res %>% group_by(location) %>%
  mutate(res.t1 = lag(res.t, 1),
        res.t2 = lag(res.t, 2))

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

```

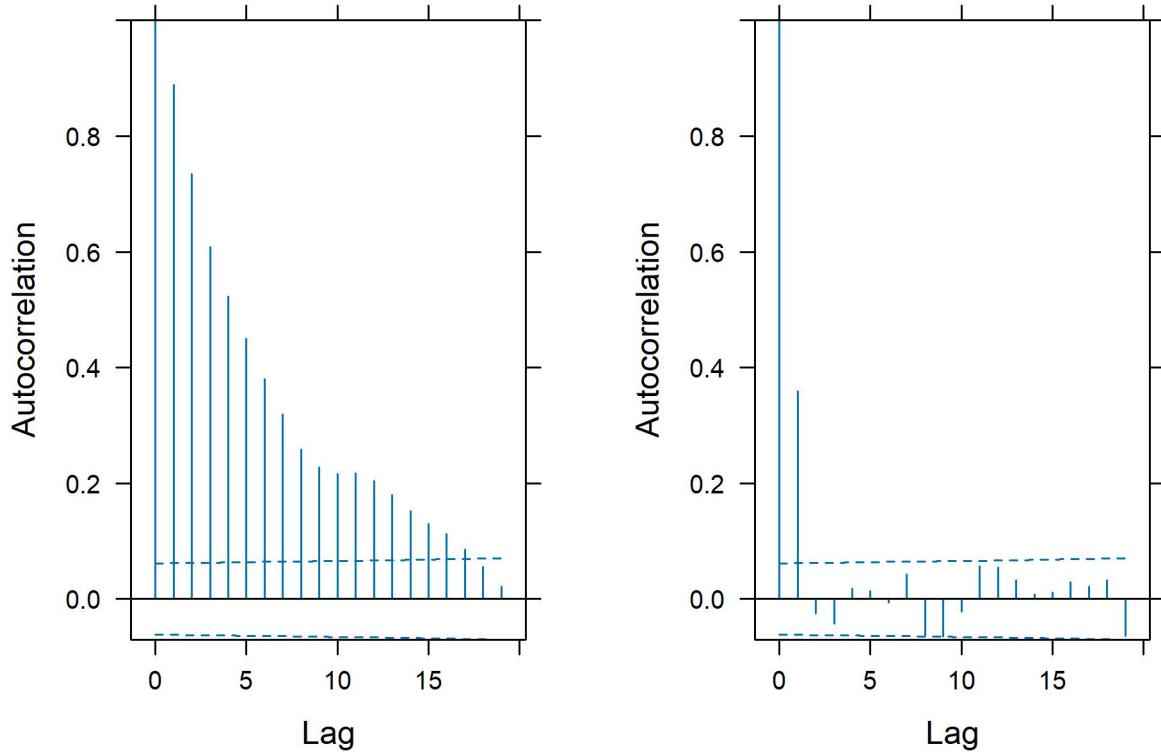
```

# Combine and add up both components
p <- merge(preds.annual, preds.residuals, by=c("location","date"))
p[, "preds"] <- p$preds.annual + p$pres
p[, "preds.ar"] <- p$preds.annual + p$pres.ar

compare <- merge(current.testing, p, by=c("location","date")) %>%
  select(location, year, date, water, preds, preds.ar, preds.annual)

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

```



```

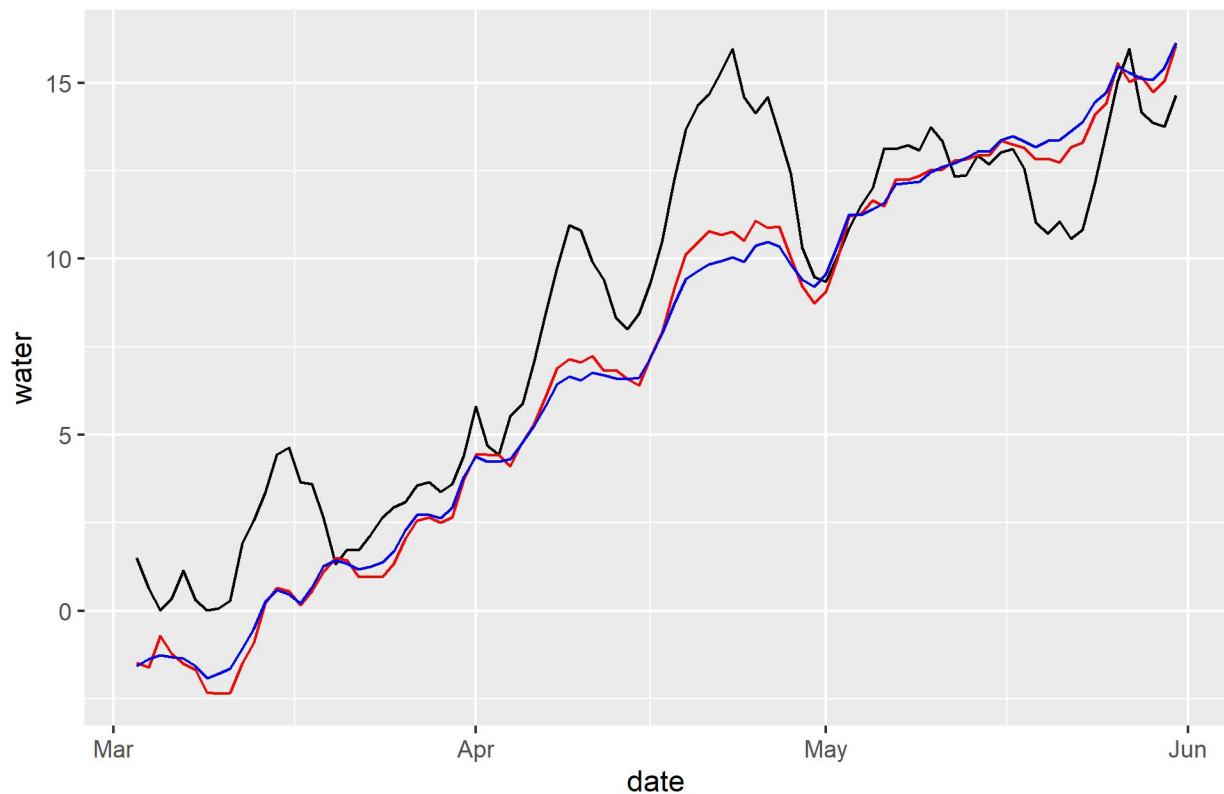
## Plots
for (loc in loc_seq) {
  plot.df <- compare[compare$location == loc,]
  diff <- round(sqrt(mean((plot.df$preds-plot.df$water)^2)),2)
  diff.ar <- round(sqrt(mean((plot.df$preds.ar-plot.df$water)^2)),2)

  pl <- ggplot(data=plot.df, aes(x=date))+
    geom_line(aes(x=date, y=water), color = "black")+
    geom_line(aes(x=date, y=preds), color = "red")+
    geom_line(aes(x=date, y=preds.ar), color = "blue")+
    ggtitle(paste(
      loc, ":", plot.df$year, " (RMSE:", diff, "&", diff.ar, ")"))
}

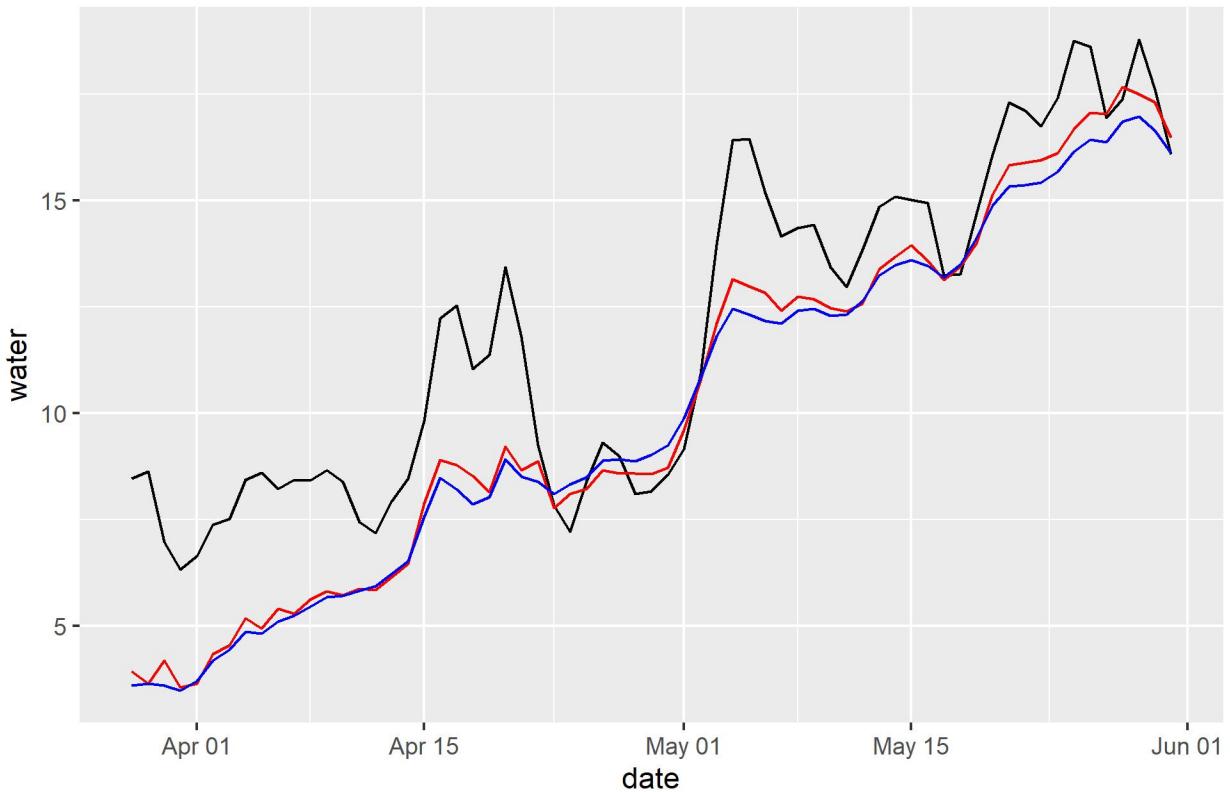
```

```
    print(p1)
}
```

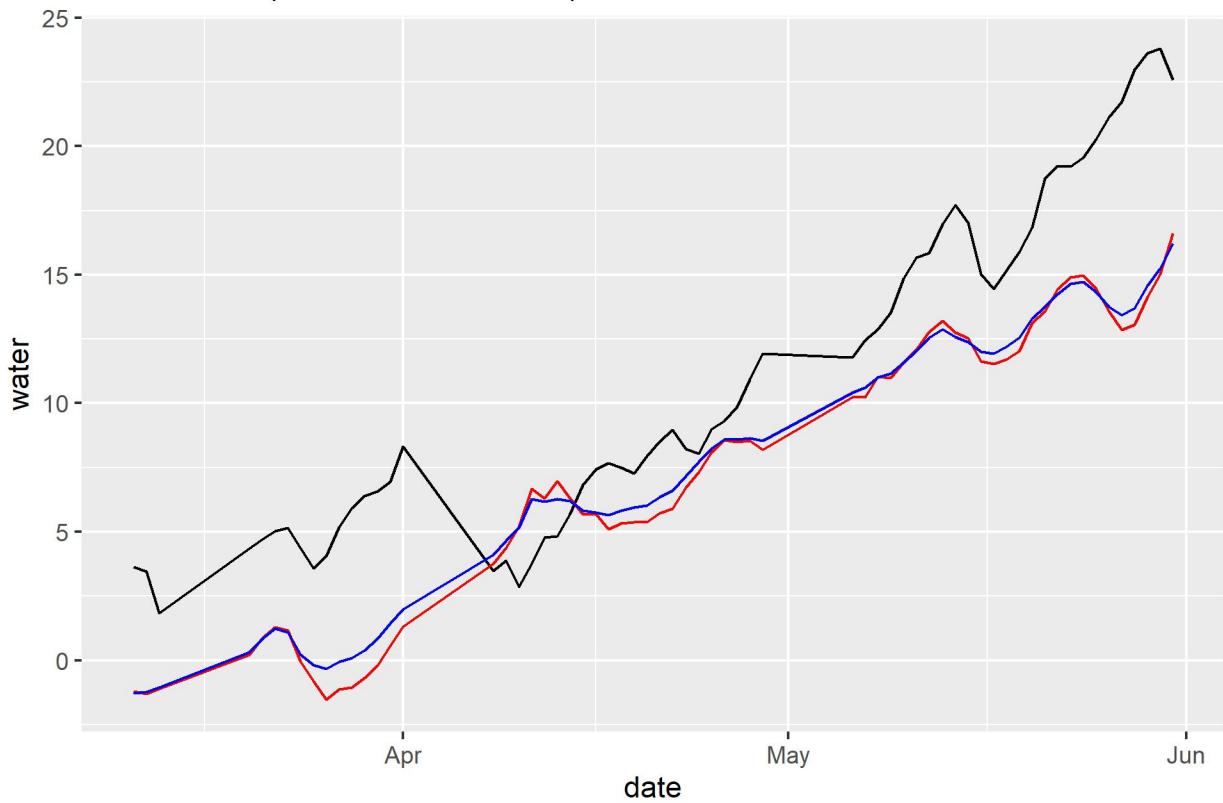
bigcreek : 2008 (RMSE: 2.18 & 2.36)



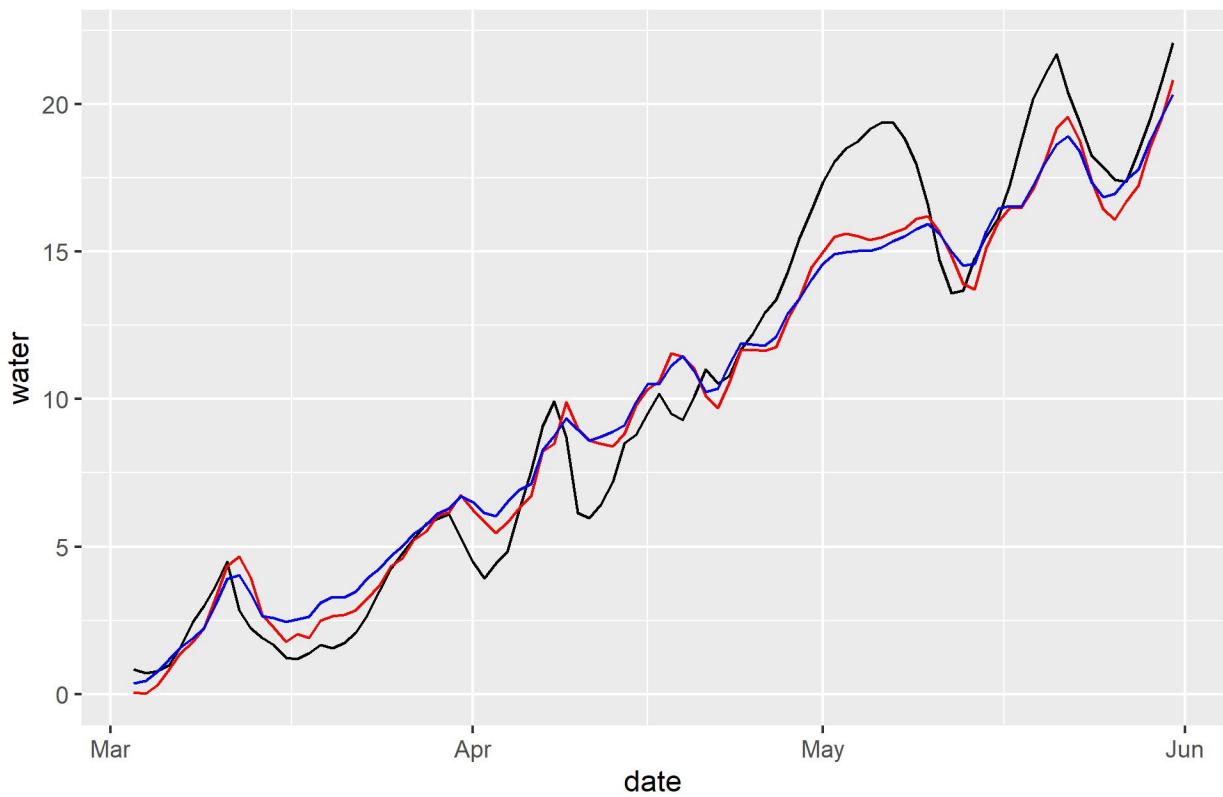
bigotter : 2012 (RMSE: 2.17 & 2.4)



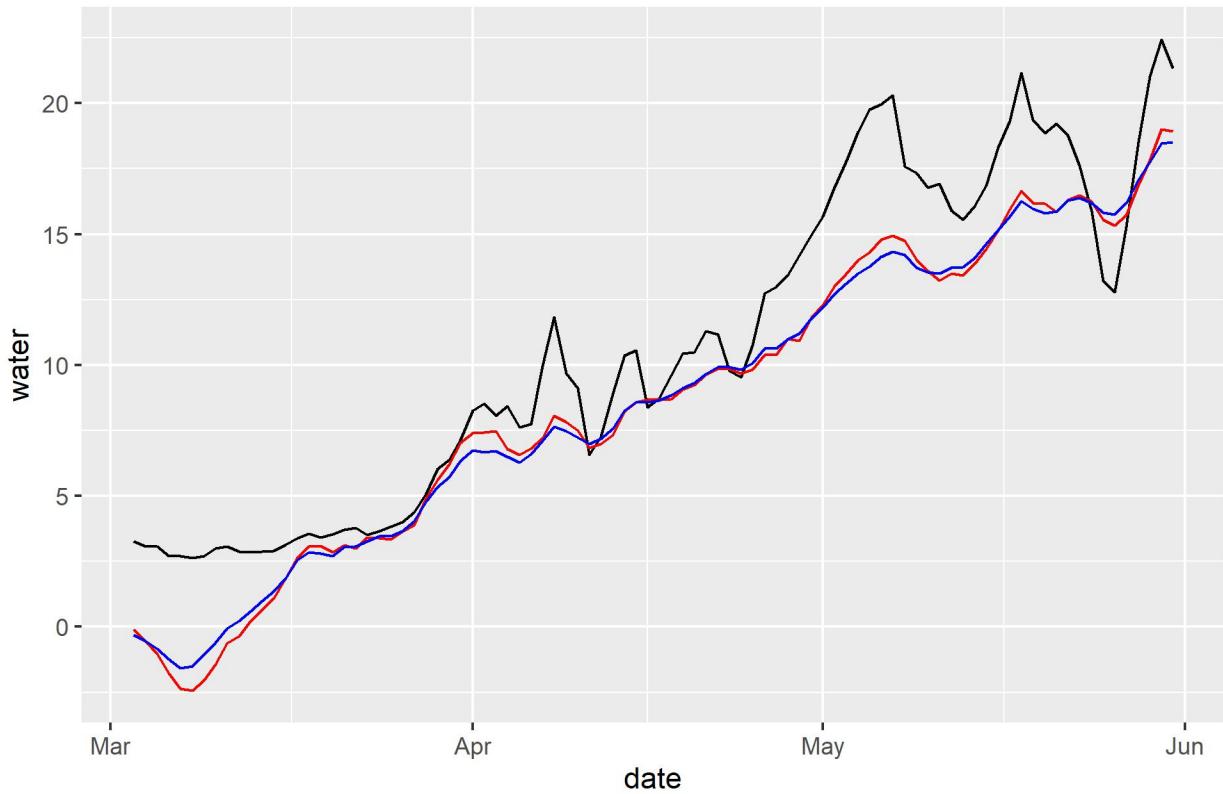
fox : 2011 (RMSE: 4.47 & 4.2)



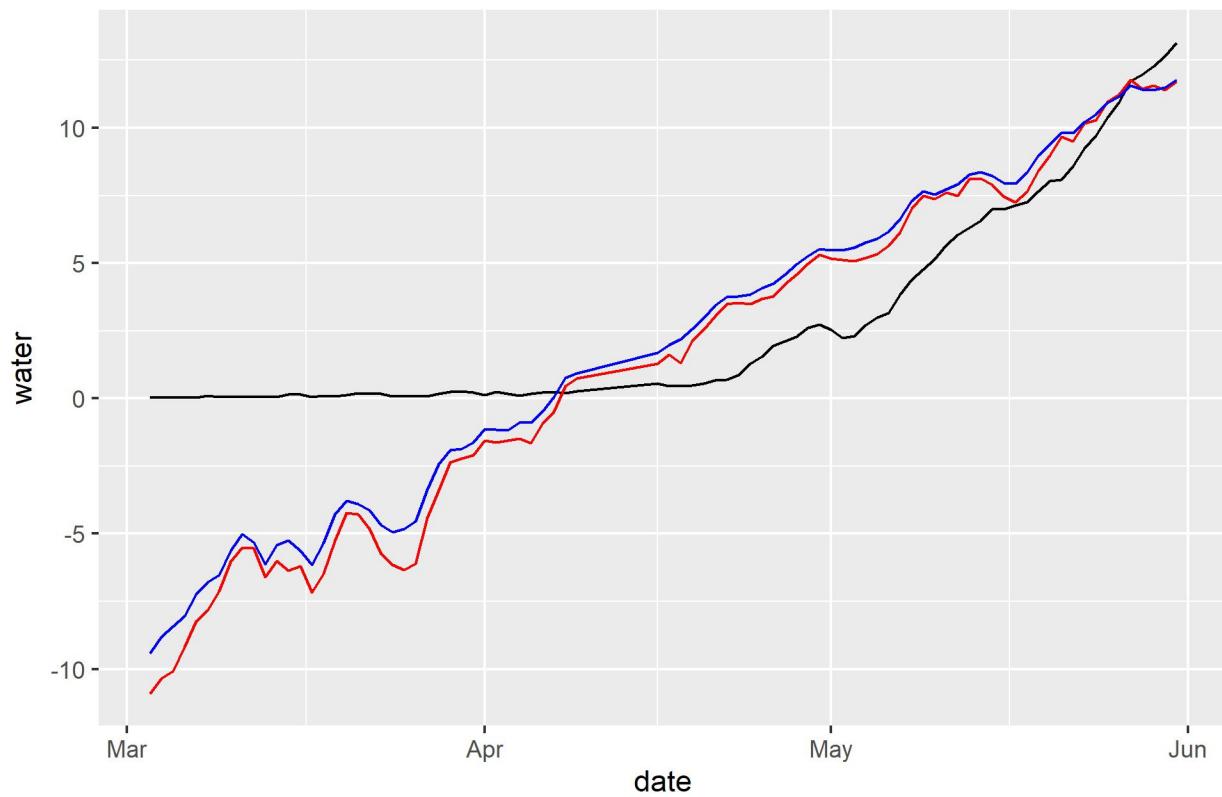
genesee : 2013 (RMSE: 1.54 & 1.66)



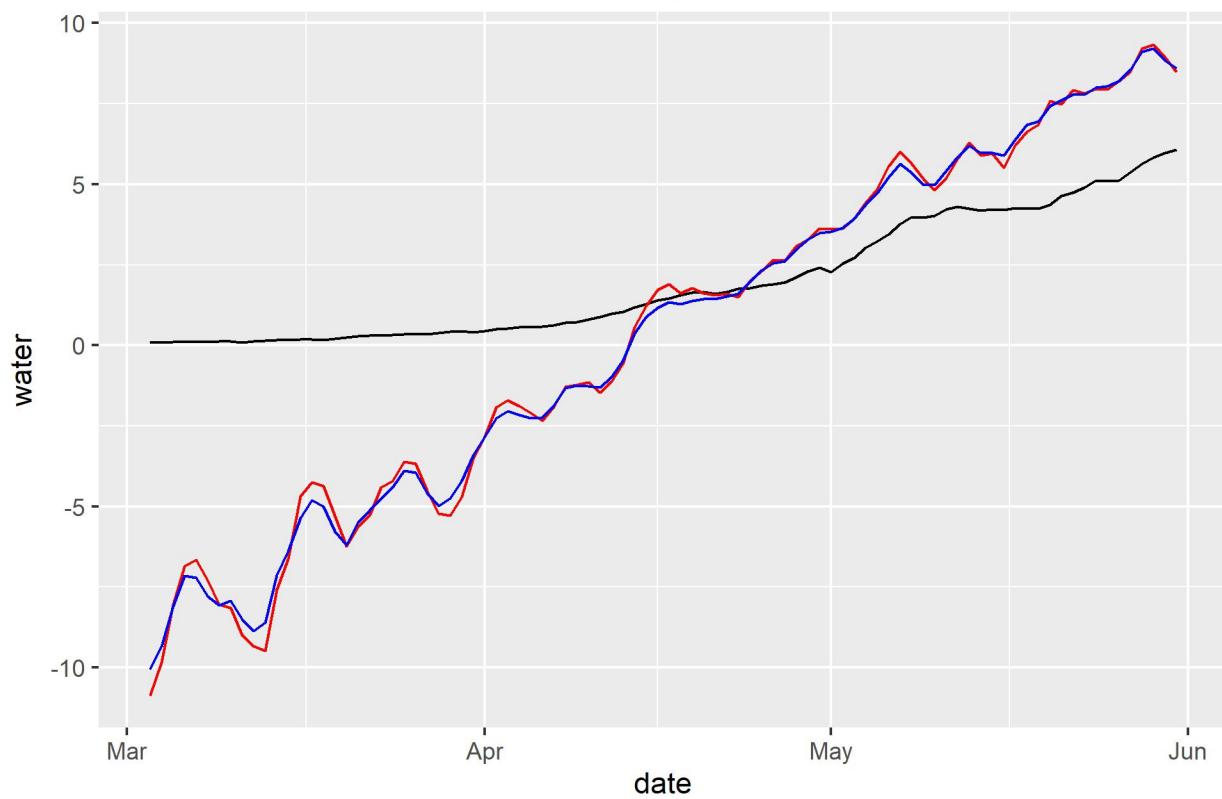
number : 1999 (RMSE: 2.65 & 2.68)



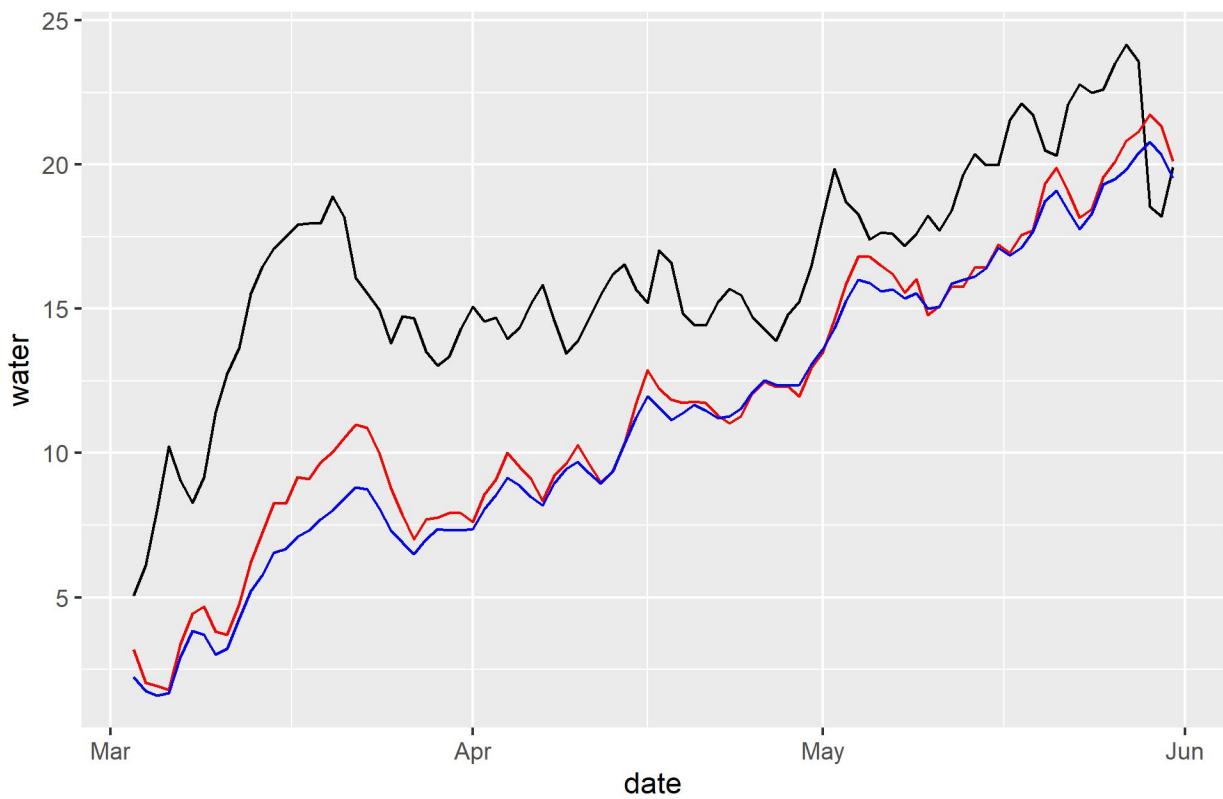
mississagi : 2014 (RMSE: 4.11 & 3.69)



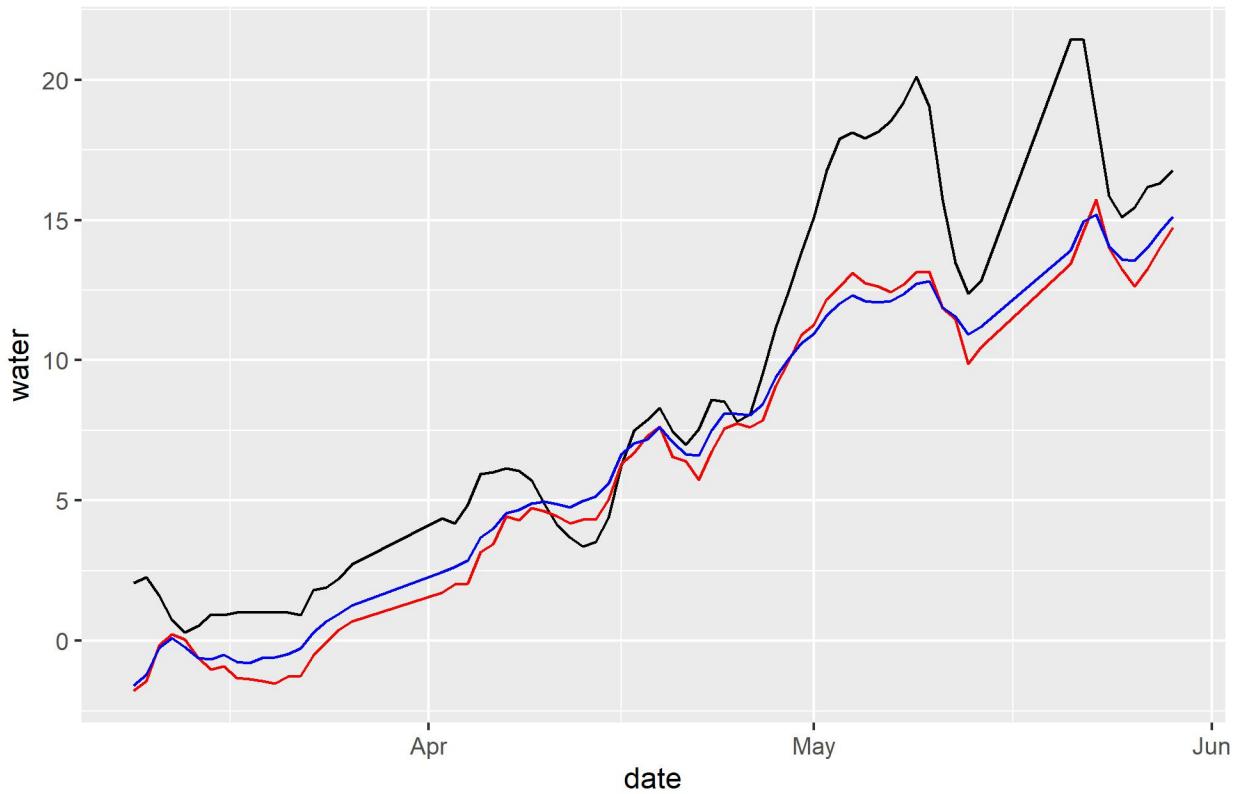
nipigon : 2009 (RMSE: 4.21 & 4.16)



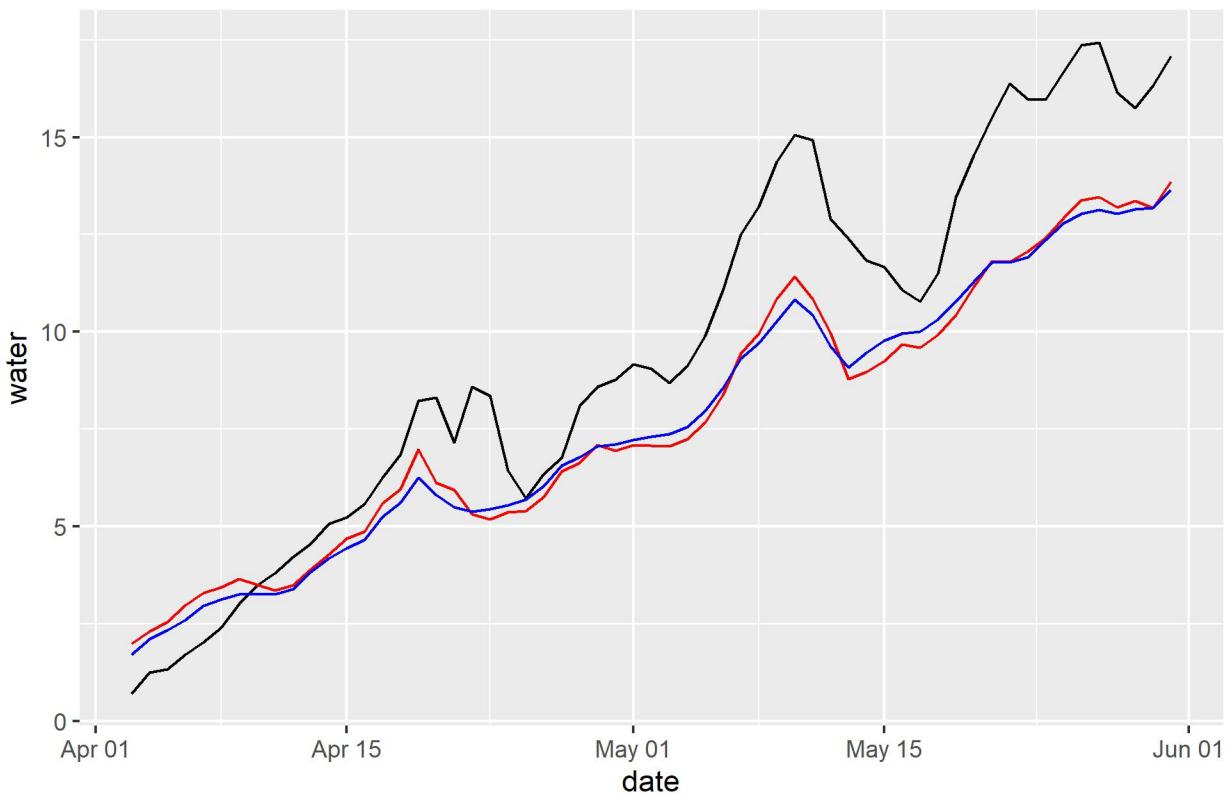
portage : 2012 (RMSE: 5.17 & 5.8)



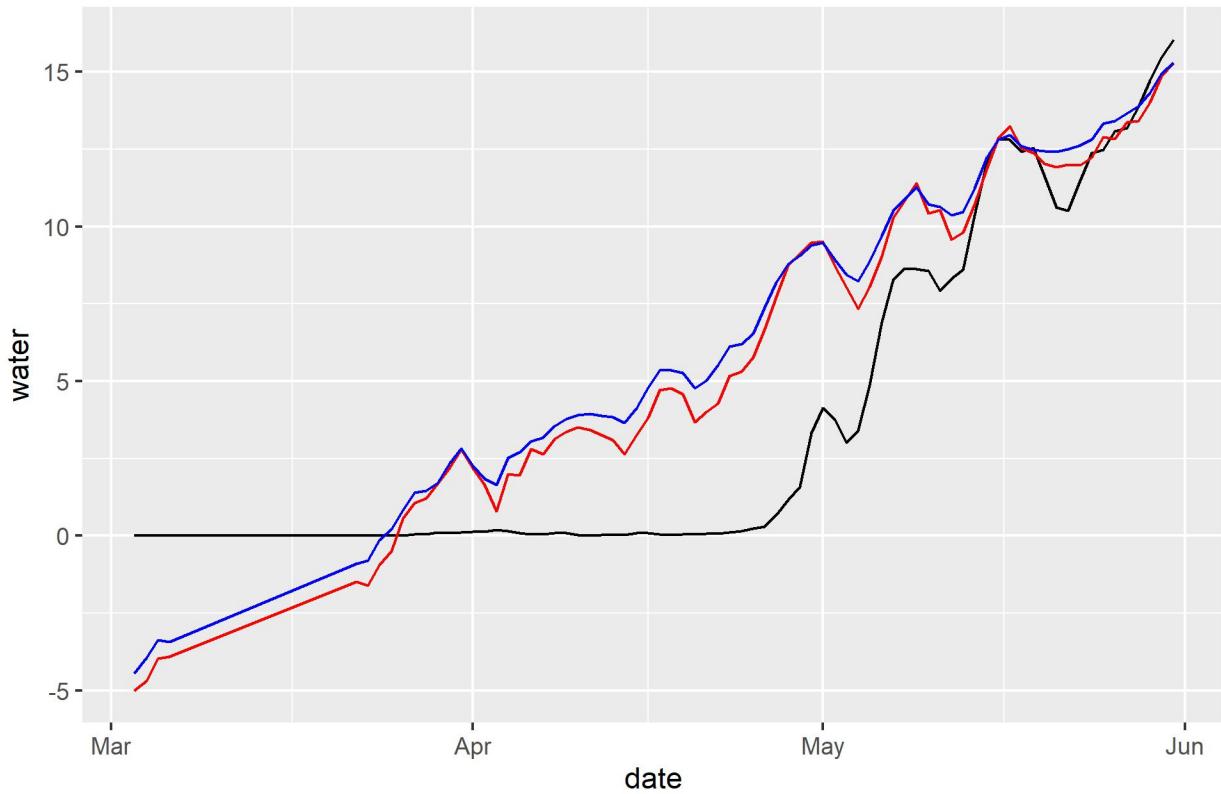
saginaw : 2013 (RMSE: 3.06 & 3)



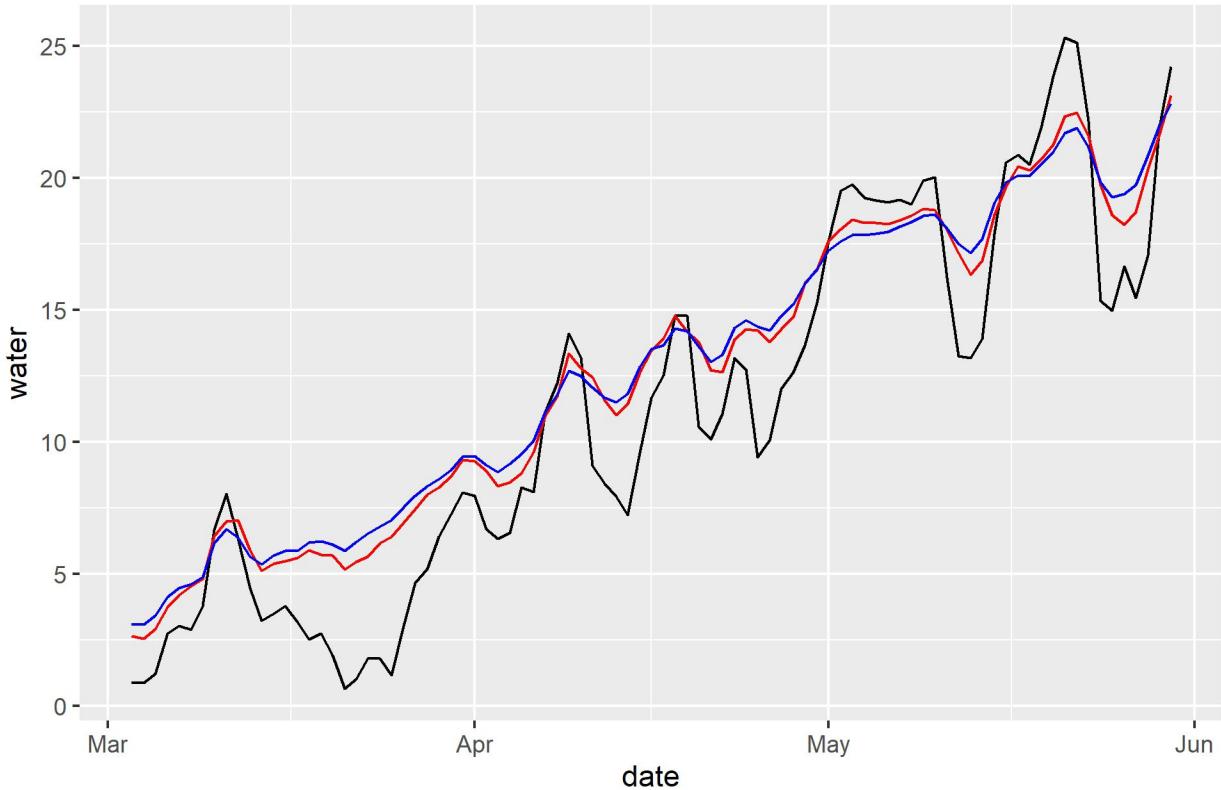
still : 2005 (RMSE: 2.42 & 2.46)



stlouis : 2013 (RMSE: 3.28 & 3.59)



vermillion : 2013 (RMSE: 2.36 & 2.71)



```

## Ploting only the annual component
# for (loc in loc_seq) {
#   plot.df <- compare[compare$location == loc,]
#
#   pl <- ggplot(data=plot.df, aes(x=date))+
#     geom_line(aes(x=date, y=water), color = "black")+
#     geom_line(aes(x=date, y=preds.annual), color = "blue")+
#     ggtitle(paste(loc, ": ", plot.df$year))
#   print(pl)
# }

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

anova(residual.comp, residual.comp.ar)

##          Model df      AIC      BIC    logLik
## residual.comp      1 6 4458.480 4487.956 -2223.240
## residual.comp.ar   2 6 2786.611 2816.087 -1387.305

```

1. Temporal autocorrelation: Set phi=0.85 largely reduced temporal autocorrelation to an acceptable level for all seasons.
2. Gives good estimation for spring, summer, and fall. In late fall and winter, the model tends to underestimate the temperature.

3. In general, this model cannot capture the detailed daily variation when set phi at a reasonably high level.

Lag model with flow variable

We determined in the “TEMP simple regression with flow.R” that including inverse flow as input is the best model with AIC selection.

So here, let's try the multiple linear regression lag 5 model with daily inverse flow as a separate input.

```
## Forms
form.locreandom <- water ~ air + dmean_1 + dmean_2 + dmean_3 + dmean_4 + dmean_5 + I(1/flow)
compare <- current.testing

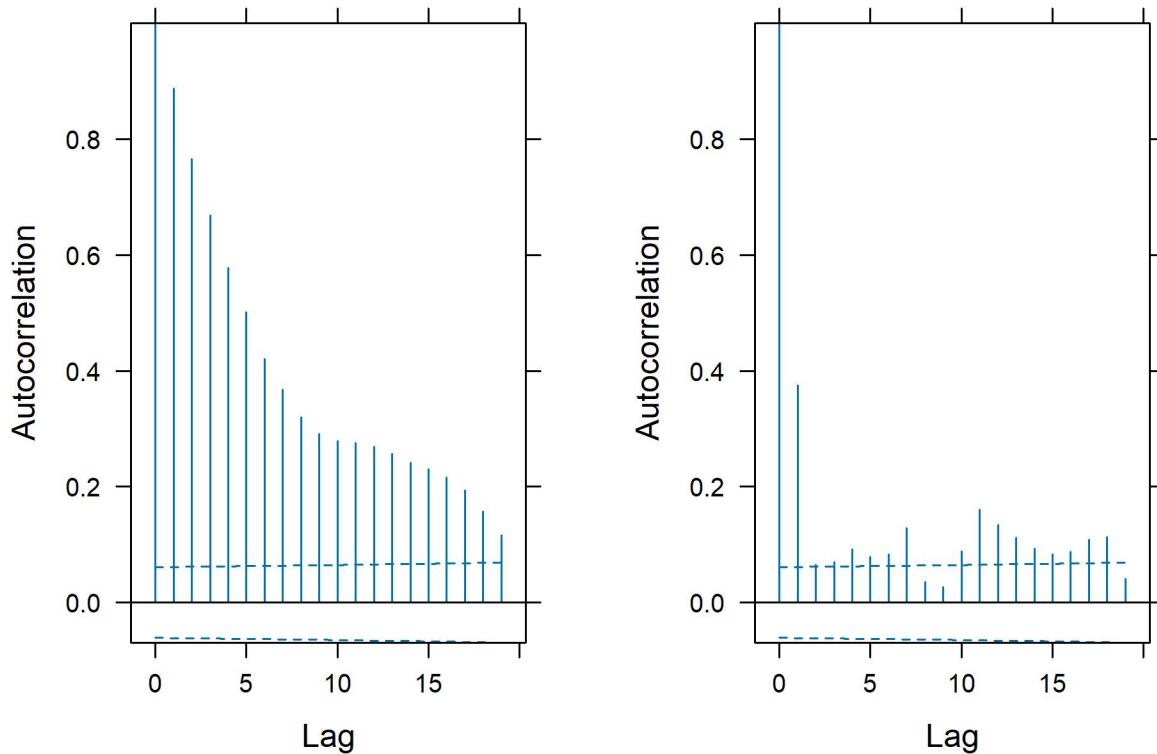
# model training and predicting
model <- lme(form.locreandom,
               random = ~1 | location,
               na.action = na.omit, data = current.training)

compare$preds <- as.vector(predict(model, newdata = current.testing))

# Include AR term
model.ar <- lme(form.locreandom,
                  random = ~1 | location, control = ctrl,
                  na.action = na.omit, data = current.training,
                  correlation=corAR1(form=~1|location,0.85,fixed=T))

compare$preds.ar <- as.vector(predict(model.ar, newdata = current.testing, re.form = ~1|location))

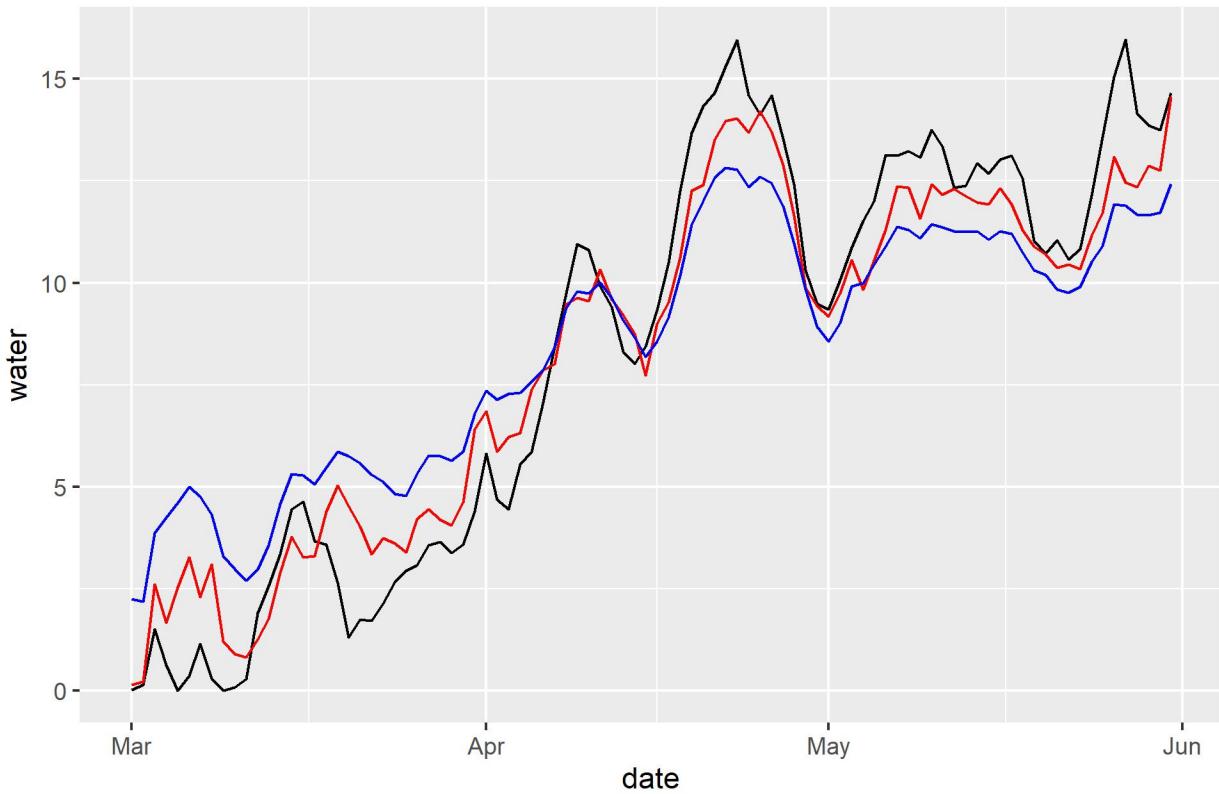
## Plot ACF
ggarrange(plot(ACF(model,resType="normalized"),alpha=0.05),
          plot(ACF(model.ar,resType="normalized"),alpha=0.05),
          nrow = 1)
```



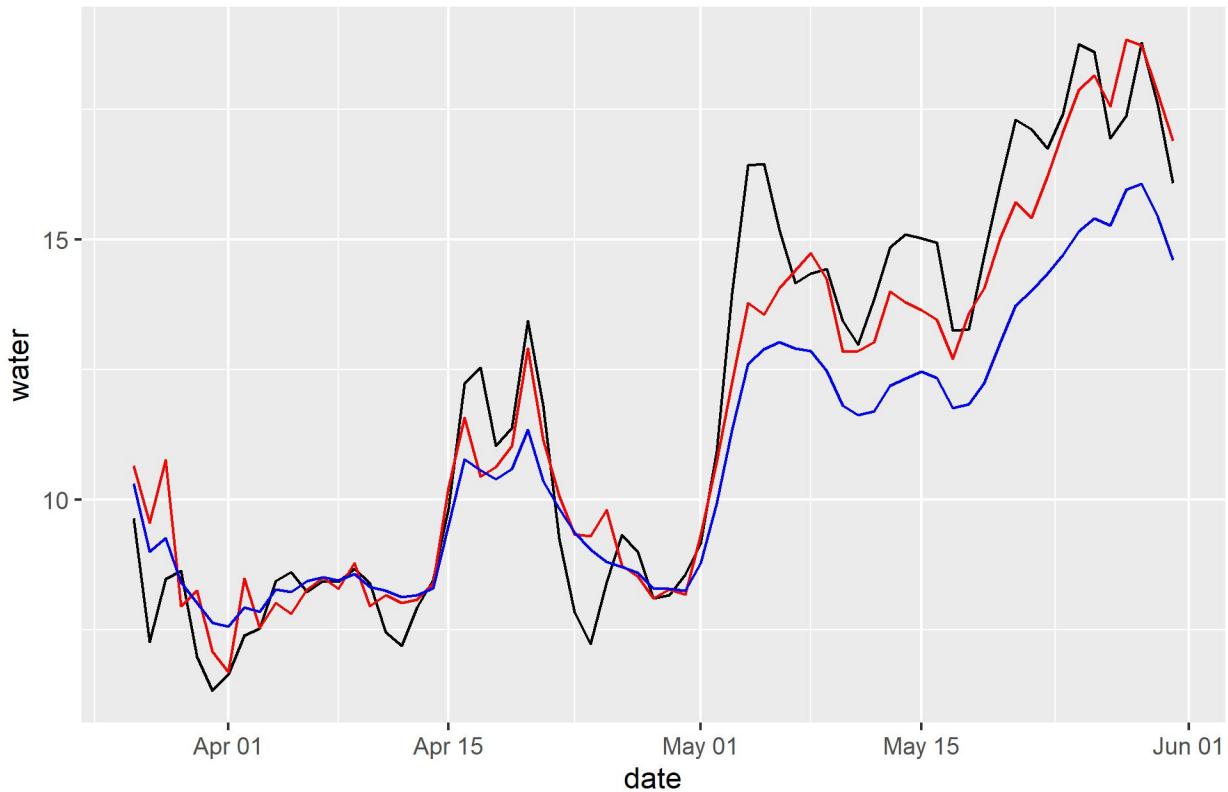
```
## Plots
for (loc in loc_seq) {
  plot.df <- compare[compare$location == loc,]
  diff <- round(sqrt(mean((plot.df$preds - plot.df$water)^2)),2)
  diff.ar <- round(sqrt(mean((plot.df$preds.ar - plot.df$water)^2)),2)

  pl <- ggplot(data=plot.df, aes(x=date))+
    geom_line(aes(x=date, y=water), color = "black")+
    geom_line(aes(x=date, y=preds), color = "red")+
    geom_line(aes(x=date, y=preds.ar), color = "blue")+
    ggtitle(paste(
      loc, ":", plot.df$year, " (RMSE:", diff, "&", diff.ar, ")"))
  print(pl)
}
```

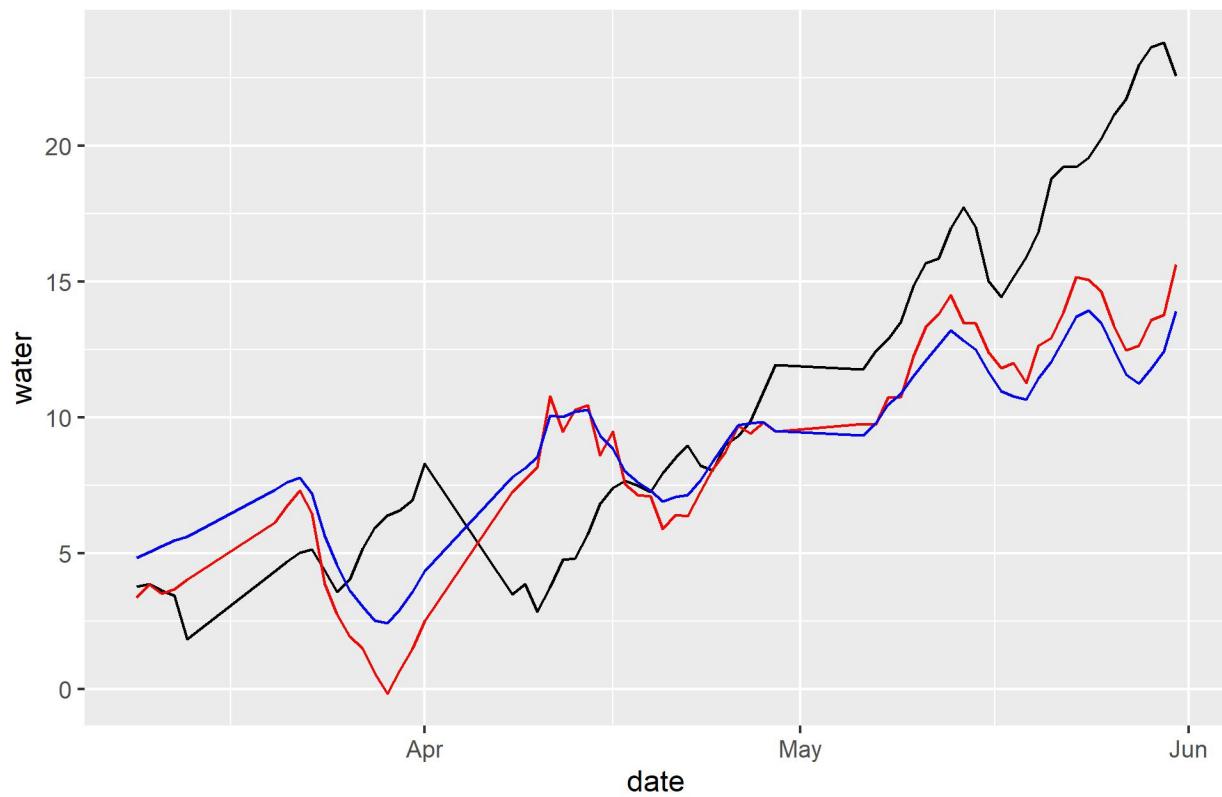
bigcreek : 2008 (RMSE: 1.27 & 2.17)



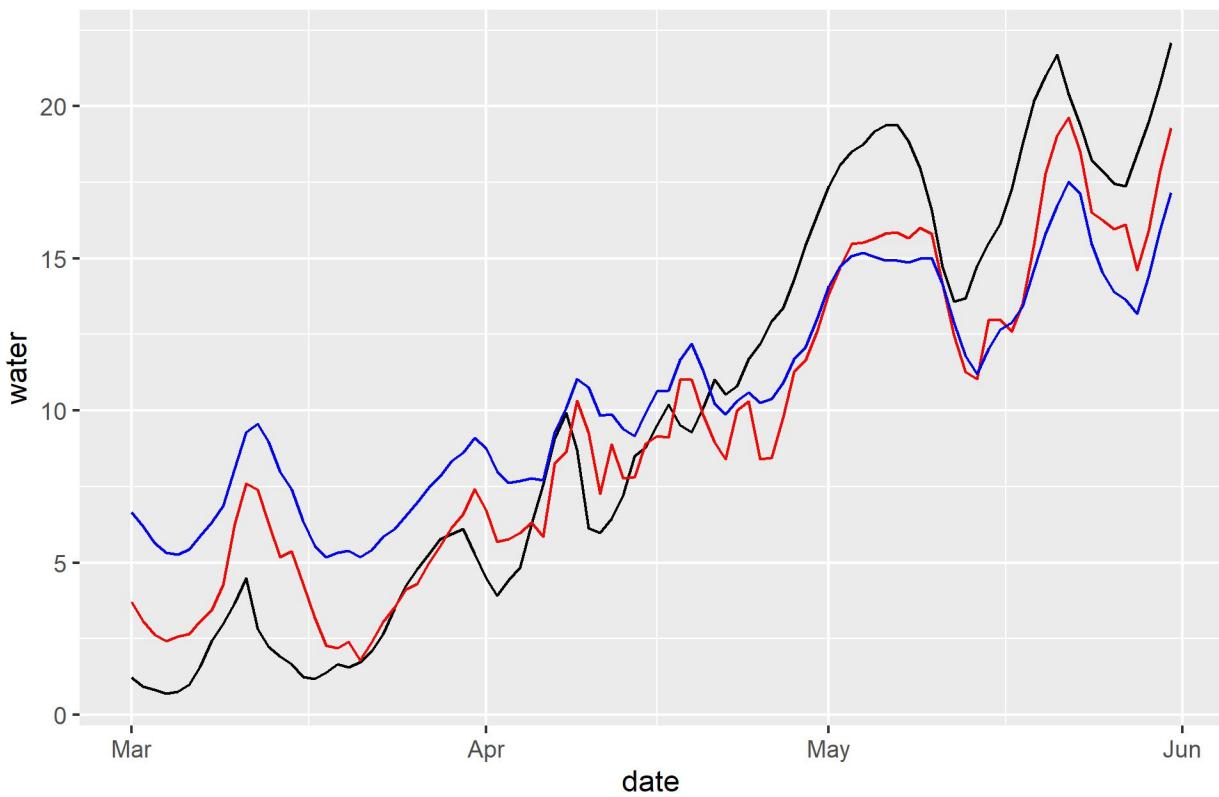
bigotter : 2012 (RMSE: 1.04 & 1.78)



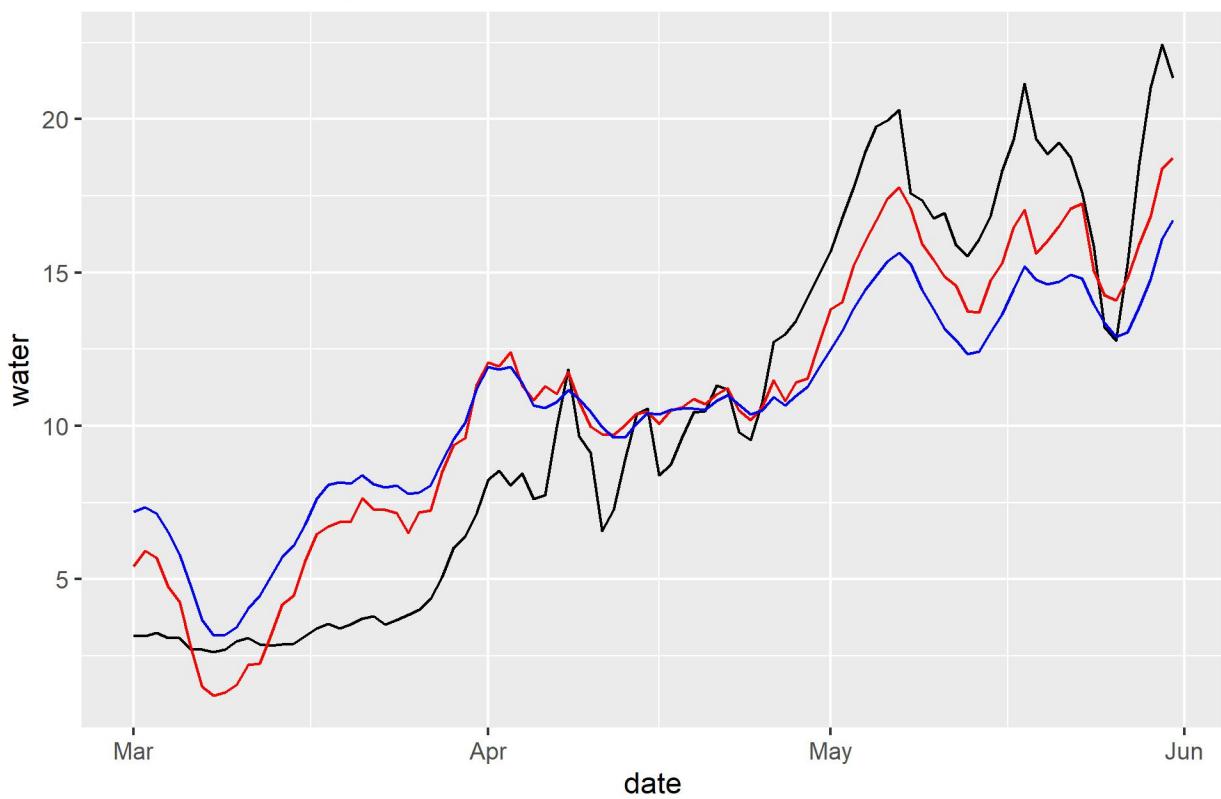
fox : 2011 (RMSE: 4.25 & 4.62)



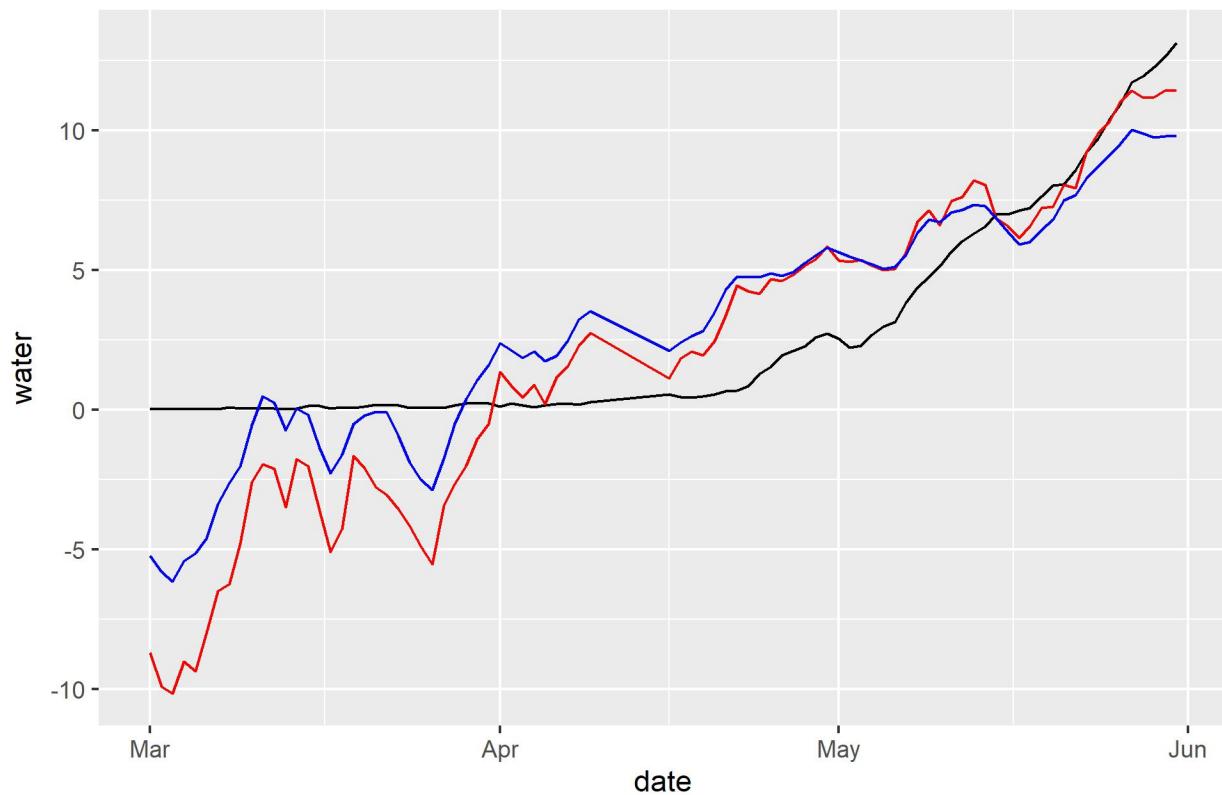
genesee : 2013 (RMSE: 2.43 & 3.62)



number : 1999 (RMSE: 2.4 & 3.31)



mississagi : 2014 (RMSE: 3.49 & 2.48)



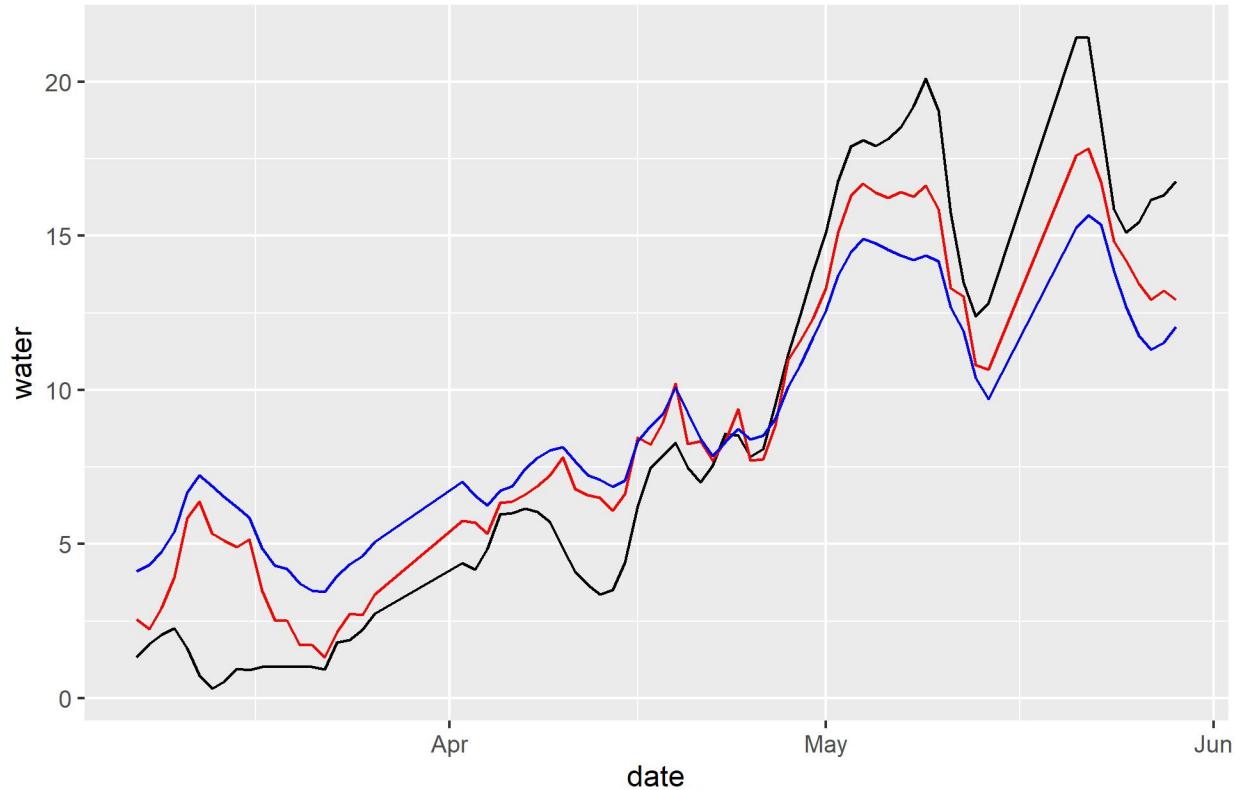
nipigon : 2009 (RMSE: 2.78 & 1.99)



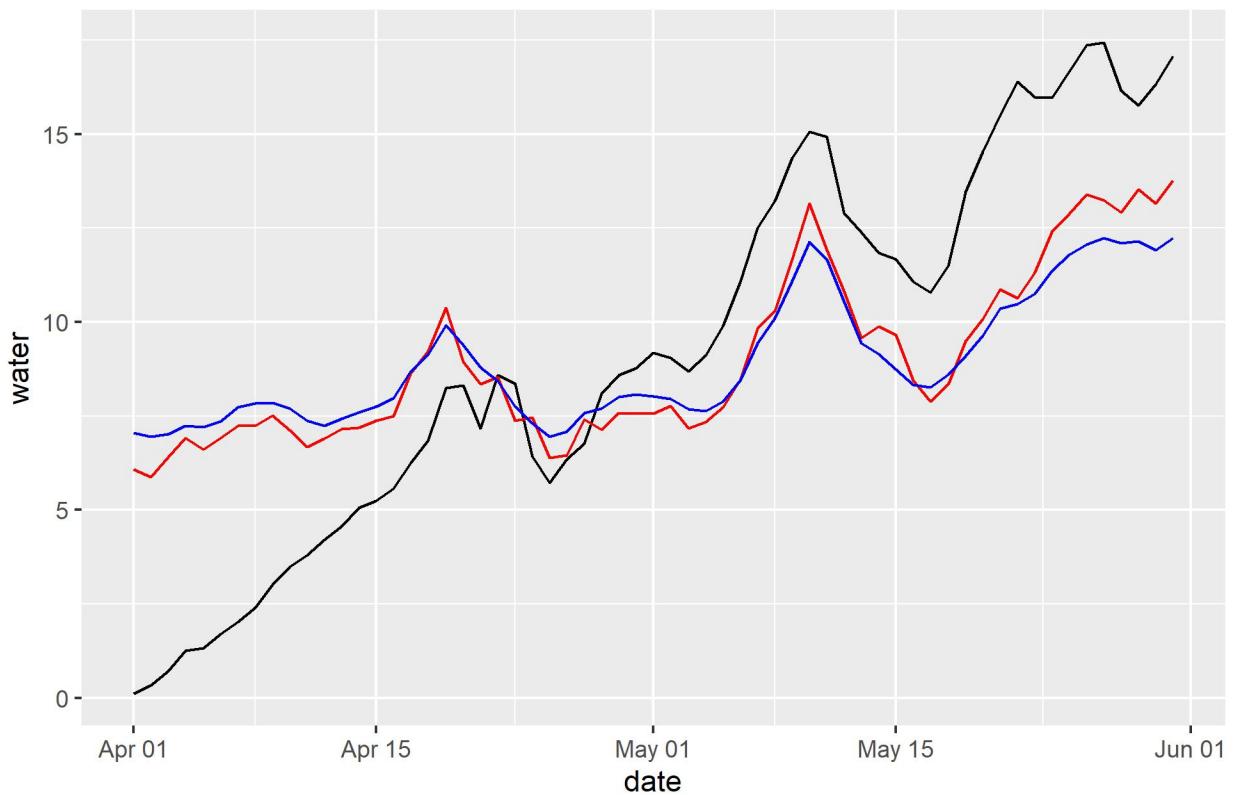
portage : 2012 (RMSE: 3.01 & 3.48)



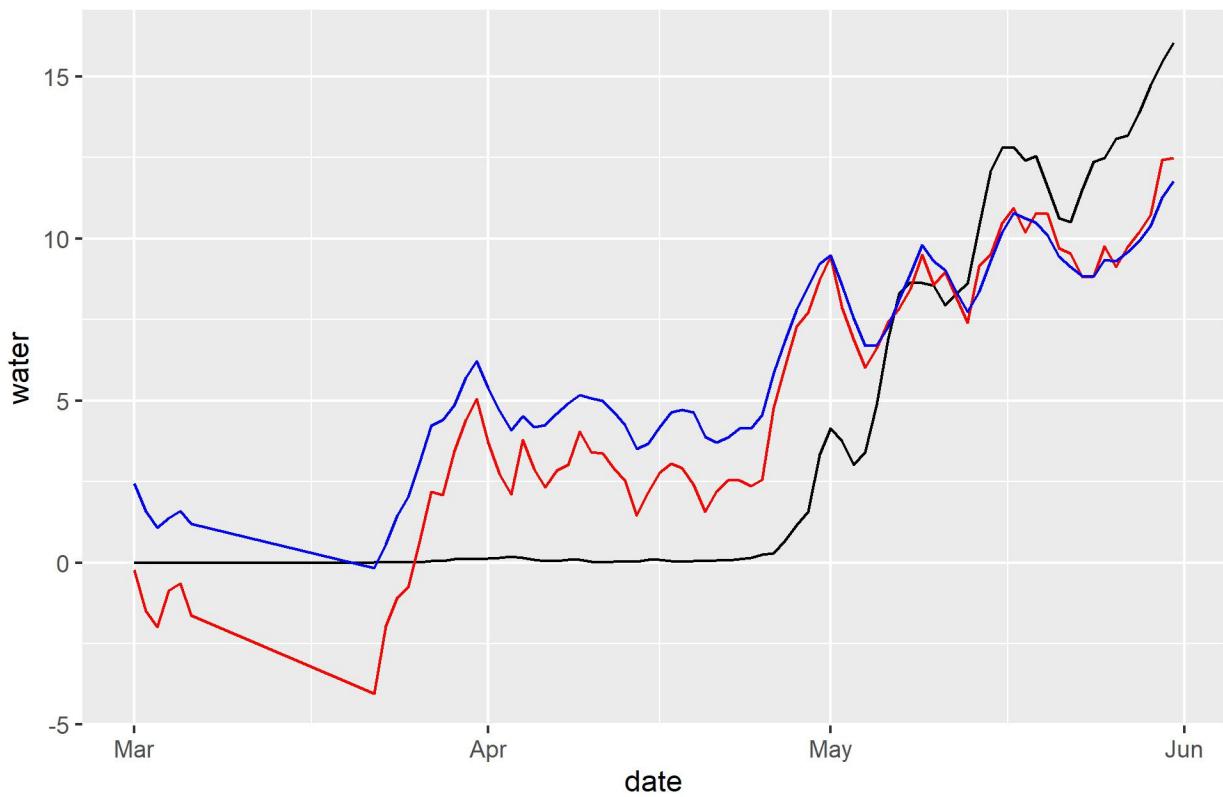
saginaw : 2013 (RMSE: 2.22 & 3.3)



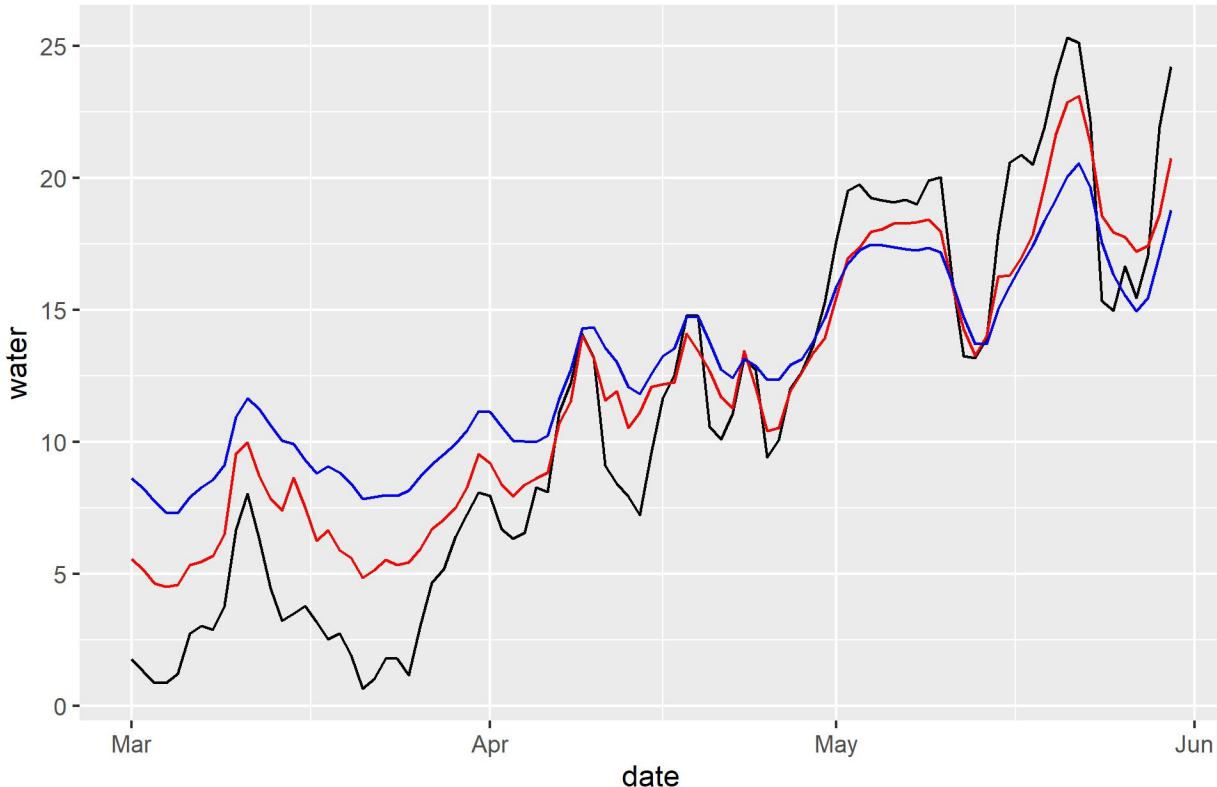
still : 2005 (RMSE: 3.24 & 3.69)



stlouis : 2013 (RMSE: 2.88 & 3.75)



vermillion : 2013 (RMSE: 2.44 & 4.02)



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

anova(model, model.ar)
```

```
##          Model df      AIC      BIC logLik
## model      1 10 4952.336 5001.660 -2466.168
## model.ar   2 10 3063.769 3113.093 -1521.884
```

1. Overall, including a temporal autocorrelation AR1 correction at $\phi=0.85$ at seasonal scale yields relatively good prediction and low ACF.
2. At annual scale (winter removed), we are unable to obtain a good predictive model while fully reducing temporal autocorrelation under significant level. Yet, it is sensible to use one (still set $\phi=0.85$) solely for the purpose of prediction.