# Seasonal river temperature model

Eddie Wu

2024-07-18

## Introduction

This R markdown file is used to

1. Fit models on a seasonal scale (all four seasons).
2. See how seasonal fits change compared to annual fit.

**All Models are mixed-effect with location as the random factor.*

```r
library(dplyr)
library(tidyverse)
library(lme4)
library(nlme)
library(xts)
library(ModelMetrics)
library(zoo)
library(lubridate)
library(nls2)
```

```r
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$location)) {
     compare.now <- subset(compare, location == loc) %>% na.omit()
     r[i,loc] <- ModelMetrics::rmse(compare.now$water, compare.now$preds.ar)
    }
  }
  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$location)) {
     compare.now <- subset(compare, location == loc) %>% na.omit()
     nsc <- 1 - sum((compare.now$water - compare.now$preds.ar)^2) / sum((compare.now$water - mean(compa
     r[i,loc] <- nsc
    }
  }
  return(r)
}
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
    dfind = data.frame(location=character(),
                       year=integer(), stringsAsFactors = FALSE)
    dfindt = data.frame(location=character(),
                       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){
        indx.train=(sample(smp, 1))
        ytrain=indx.train
      } else if (length(smp)==1){
        indx.train=smp
        ytrain=indx.train
      }

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

```
## 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

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 - how many imputed values are in each location
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
```

### Get 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),
         dflow_1 = lag(flow, 1))


## Get relative flow
awf <- awf %>% mutate(rqc = (flow - dflow_1)/flow)
```

### 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, dflow_1, rqc)

master.temp <- master.temp[complete.cases(master.temp[,8:17]),]
```

## Get training and testing

### Subsetting seasonal scales

Now we want to subset three 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)

master.annual <- master.temp %>%
  filter(month != 12 & month != 1 & month != 2)
```

### Identify good year/month data

We want the monthly data to be greater than 60 days, annual data (winter removed) more than 200 days.

```
## Annual (250 days)
fulyear <- good_year(master.annual, 180)
```

```
## $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" "2013"
##
## $nipigon
## [1] "2008" "2009"
##
## $portage
## [1] "2011" "2012"
##
## $saginaw
## [1] "2013" "2014"
##
## $still
## [1] "2002" "2004" "2005" "2008"
```

```
##
## $stlouis
## [1] "2011" "2012" "2013" "2014"
##
## $vermilion
## [1] "2012" "2013" "2014"
```

```r
## 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"
##
## $vermilion
## [1] "2012" "2013" "2014"
```

```r
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

Similarly, get training and testing for the specific season.

```
fold = 10

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

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


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

## Model Comparison

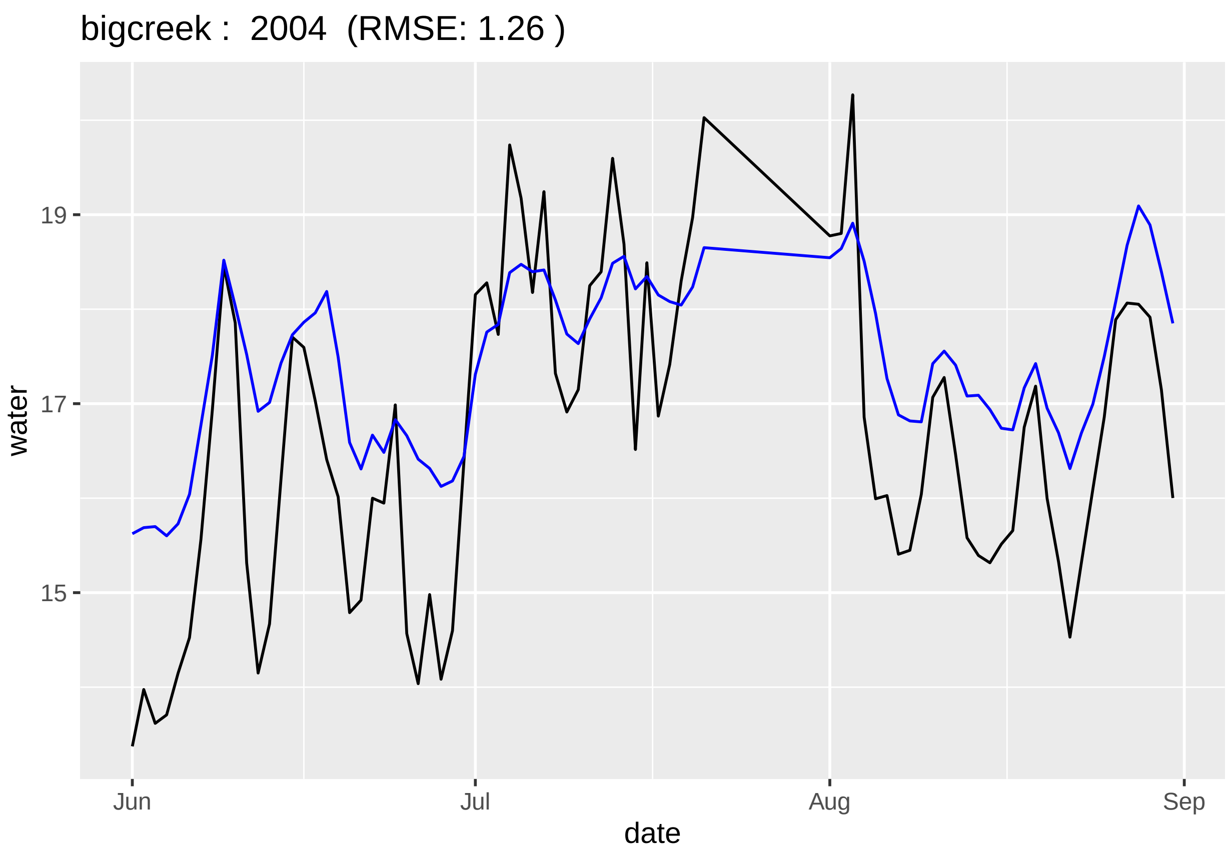
We now run each of the three models five times, once on each of the four seasons, and annual data.

**Important: Here we only look at the model output from models with corAR1() value fixed at 0.8. To check the ACF, please refer to the TEMP temporal autocorrelation document.**

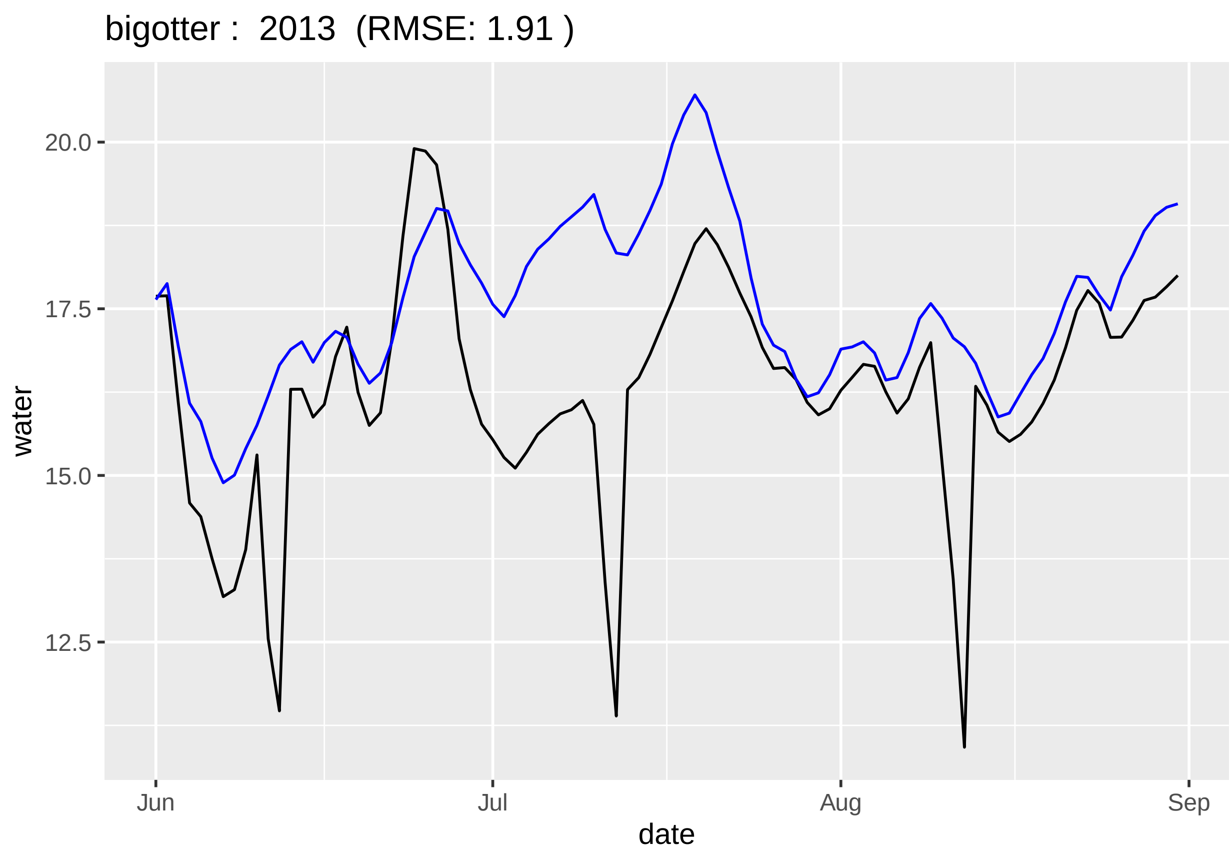
### Linear lag5

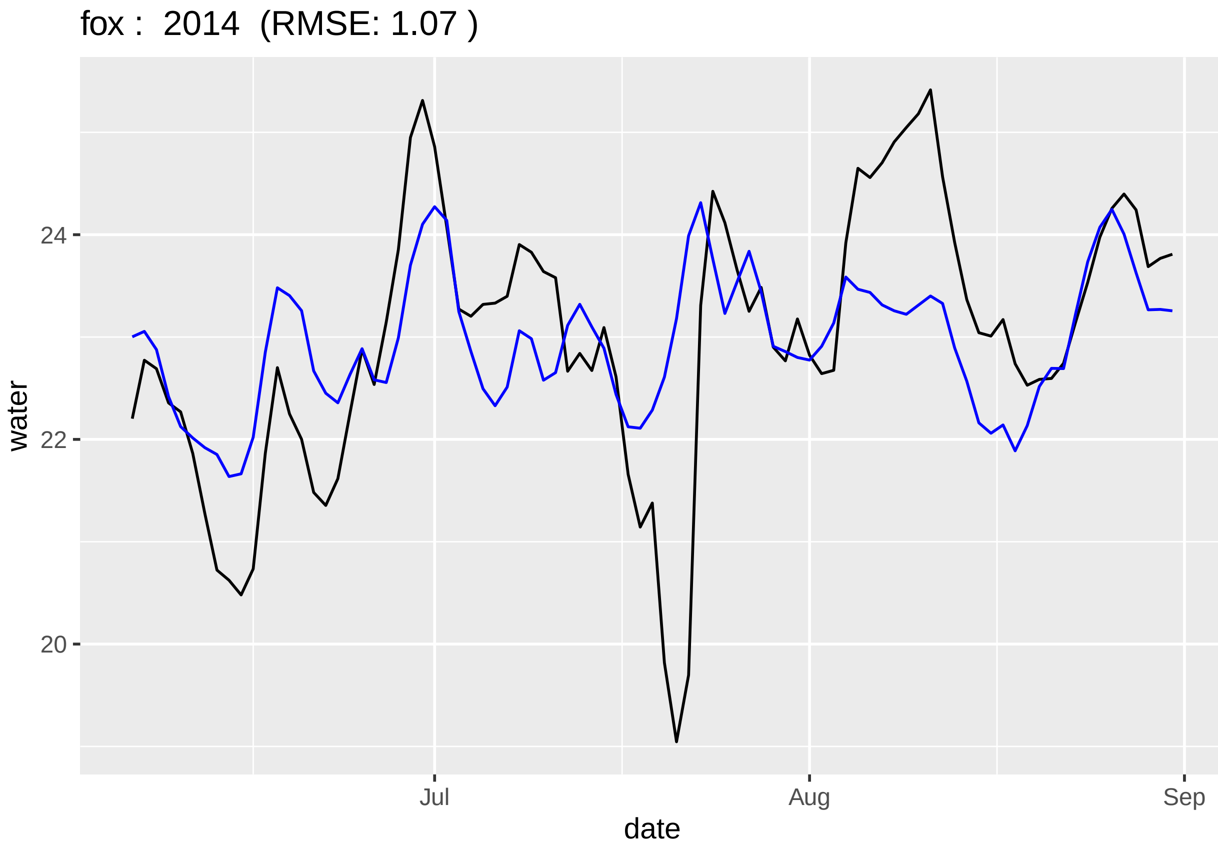
```
## Forms
ctrl = lmeControl(opt='optim')
form1 <- water ~ air + dmean_1 + dmean_2 + dmean_3 + dmean_4 + dmean_5

spring.r <- vector("list", fold)
```

```r
summer.r <- vector("list", fold)
fall.r <- vector("list", fold)
winter.r <- vector("list", fold)
annual.r <- vector("list", fold)

grand.lag5 <- list(spring.r, summer.r, fall.r, winter.r, annual.r)


## Iteration starts here
for (season in 1:5) {

  ## Get current season and its correspnding training/testing
  train <- grand_training[[season]]
  test <- grand_testing[[season]]

  ## 10 fold iteration starts here
  for (i in 1:fold){

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

    # model training and predicting
    model.ar <- lme(form1,
                    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))

    grand.lag5[[season]][[i]] <- compare
  }
}

View(grand.lag5[[4]][[1]])

## Plots
for (loc in loc_seq) {
  df <- grand.lag5[[2]][[1]]
  plot.df <- df[df$location == loc,]
  diff.ar <- round(sqrt(mean((plot.df$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.ar), color = "blue")+
          ggtitle(paste(
            loc, ": ", plot.df$year, " (RMSE:", diff.ar, ")"))
  print(pl)
}
```

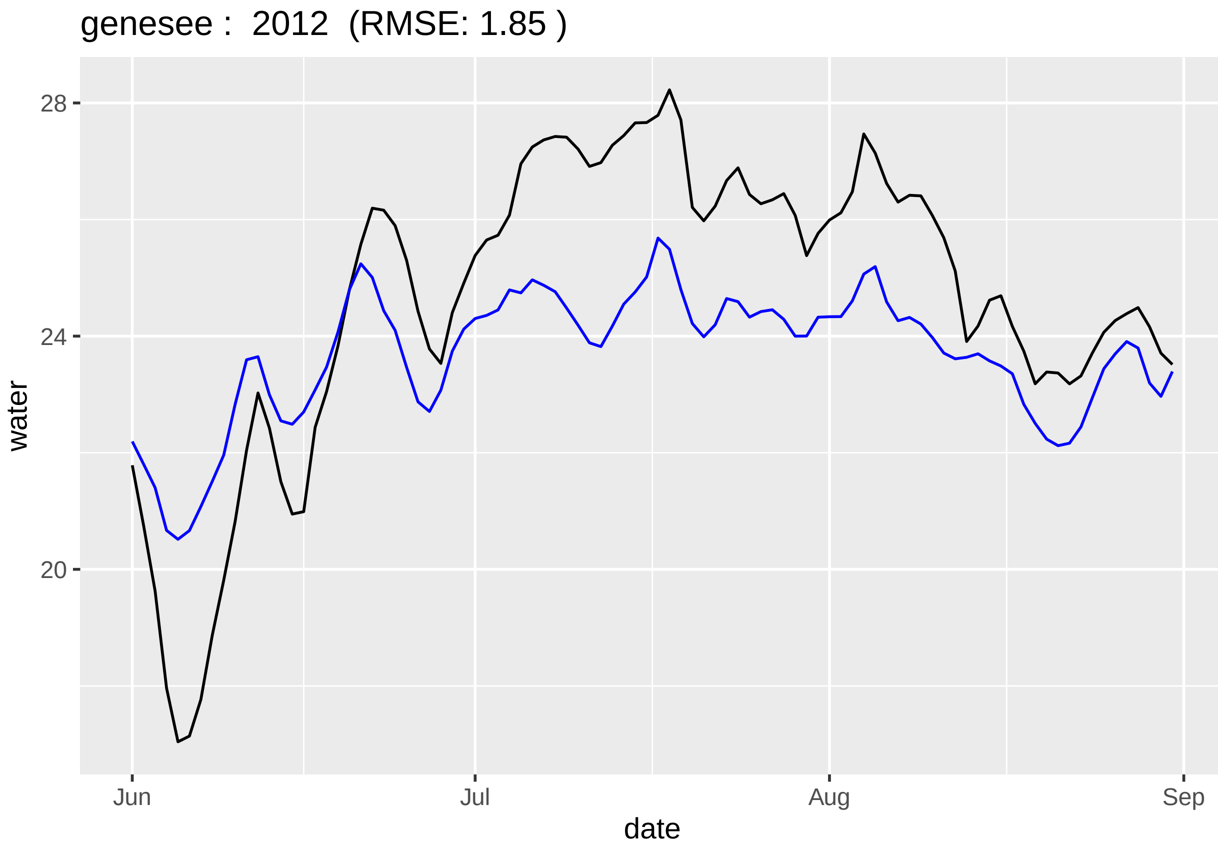


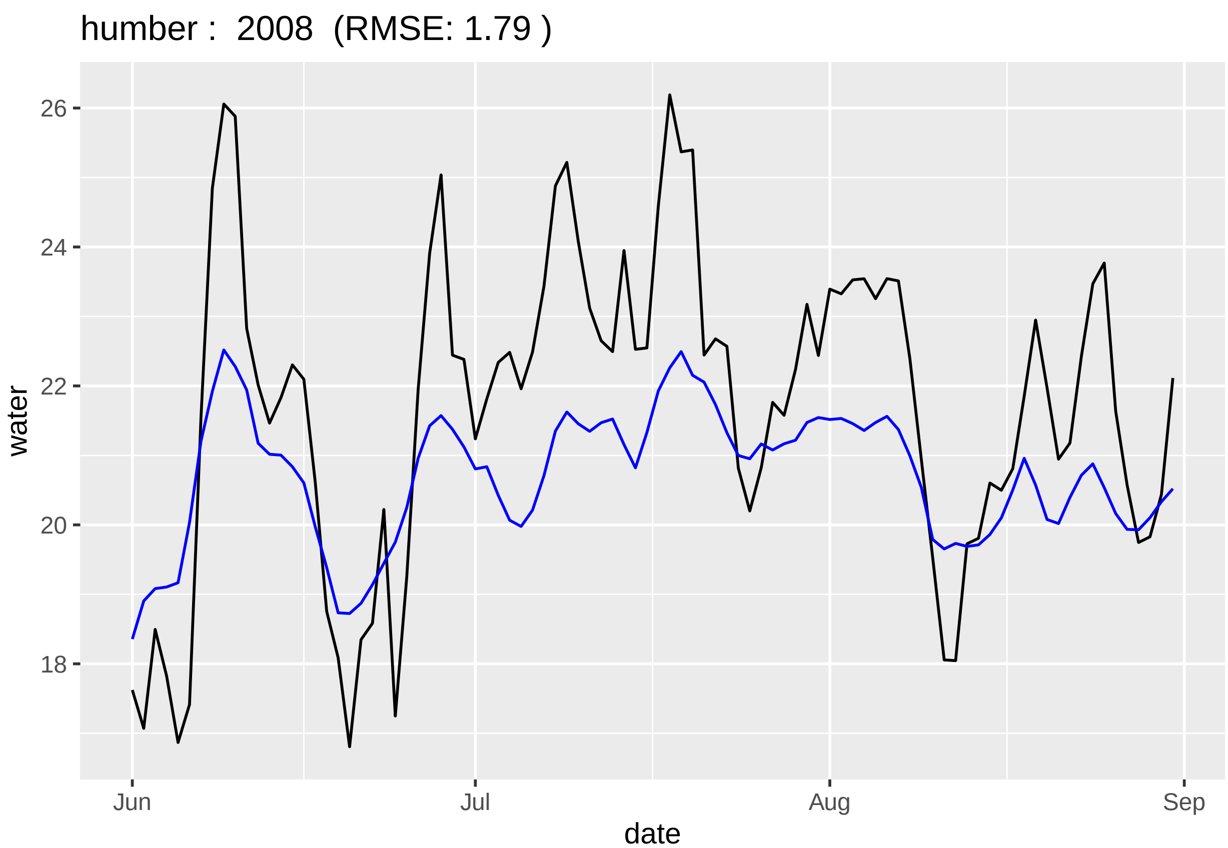
bigcreek :  2004  (RMSE: 1.26 )
water
19
17
15
Jun
Jul
Aug
Sep
date

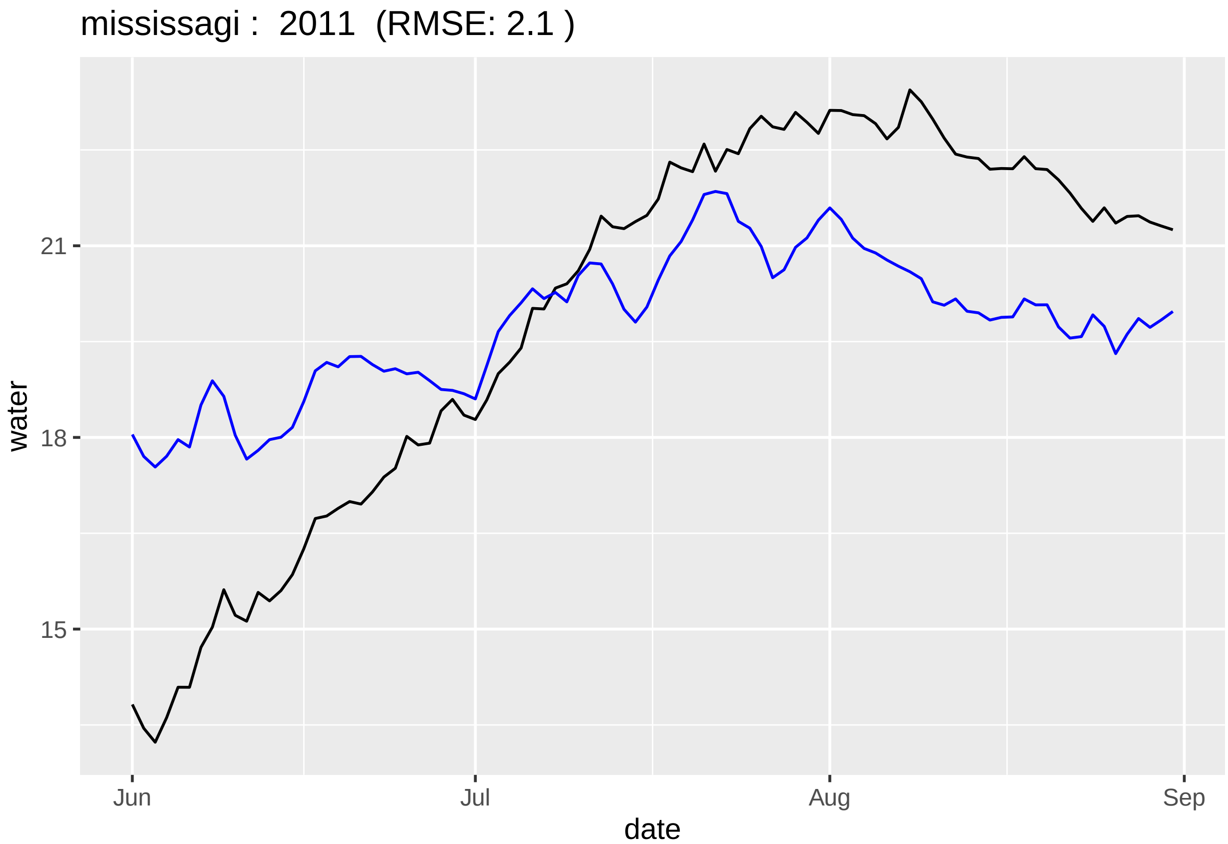

bigotter : 2013 (RMSE: 1.91 )
20.0
17.5
15.0
12.5
water
Jun
Jul
Aug
Sep
date

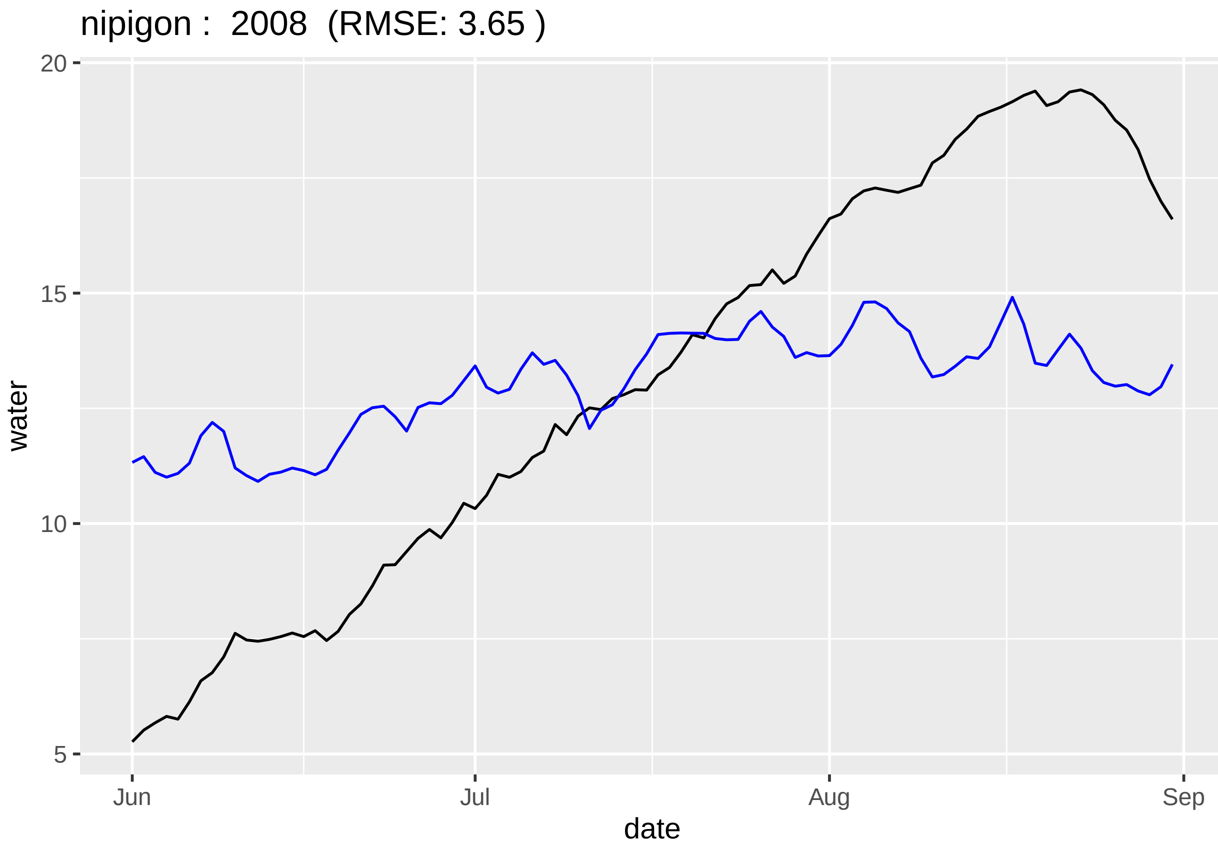


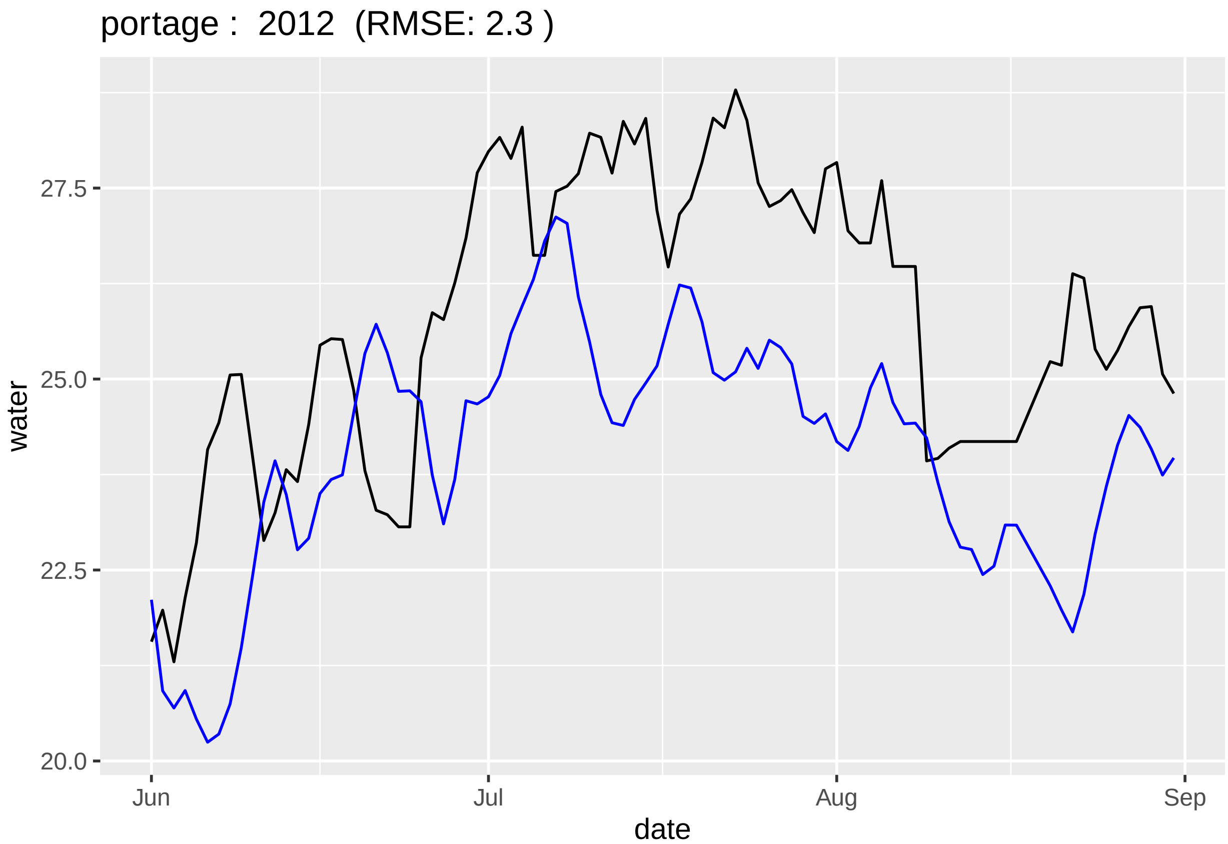
fox : 2014 (RMSE: 1.07 )
water
24
22
20
Jul
Aug
Sep
date


genesee : 2012 (RMSE: 1.85 )
28
24
20
water
Jun
Jul
Aug
Sep
date

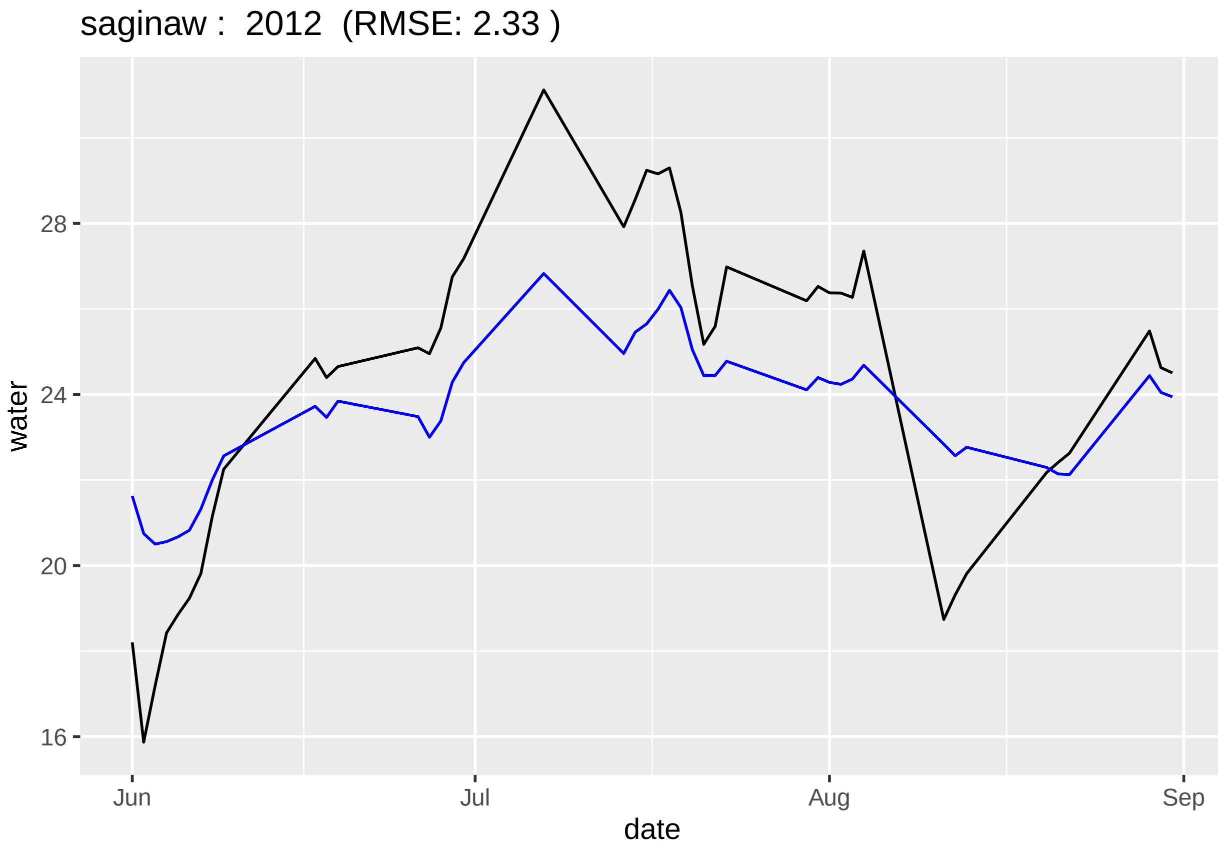

humber : 2008 (RMSE: 1.79 )
26
24
22
20
18
water
Jun
Jul
Aug
Sep
date

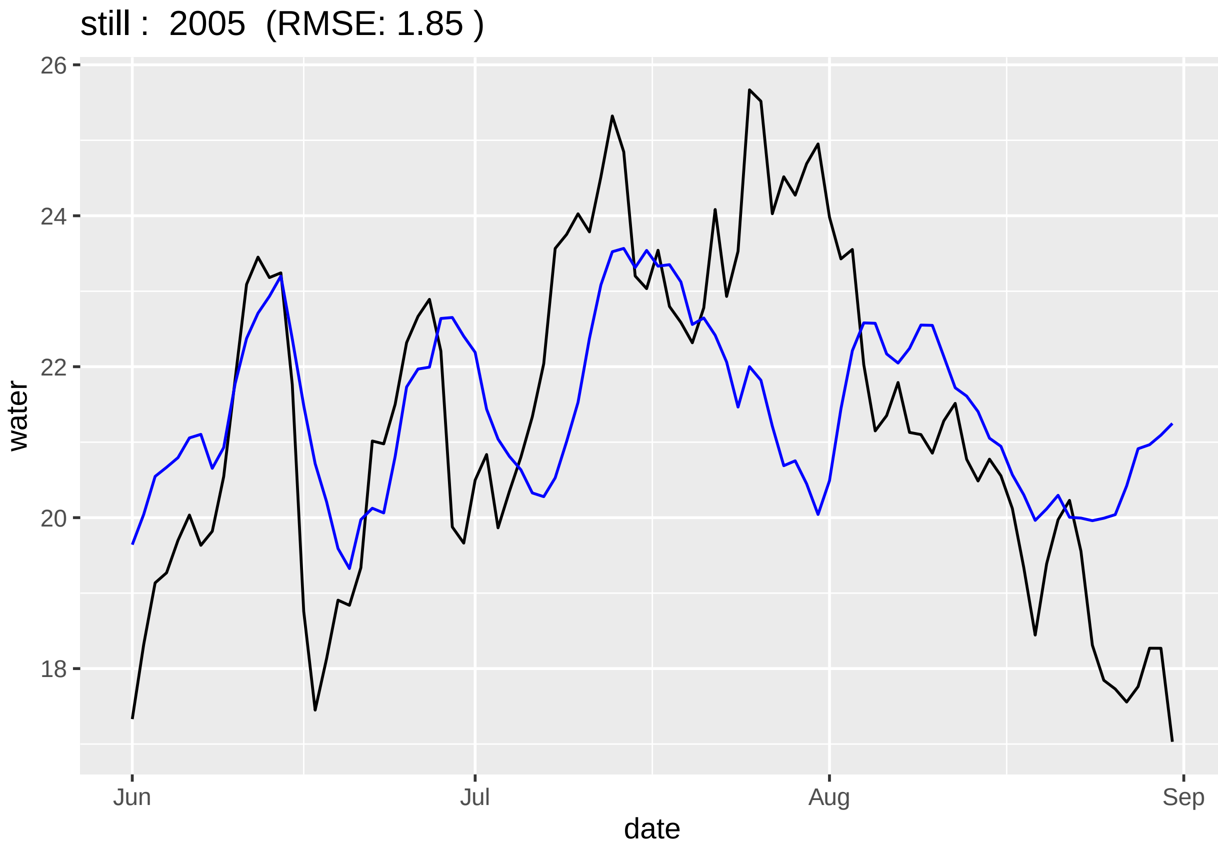

mississagi :  2011  (RMSE: 2.1 )
21
18
15
water
Jun
Jul
Aug
Sep
date

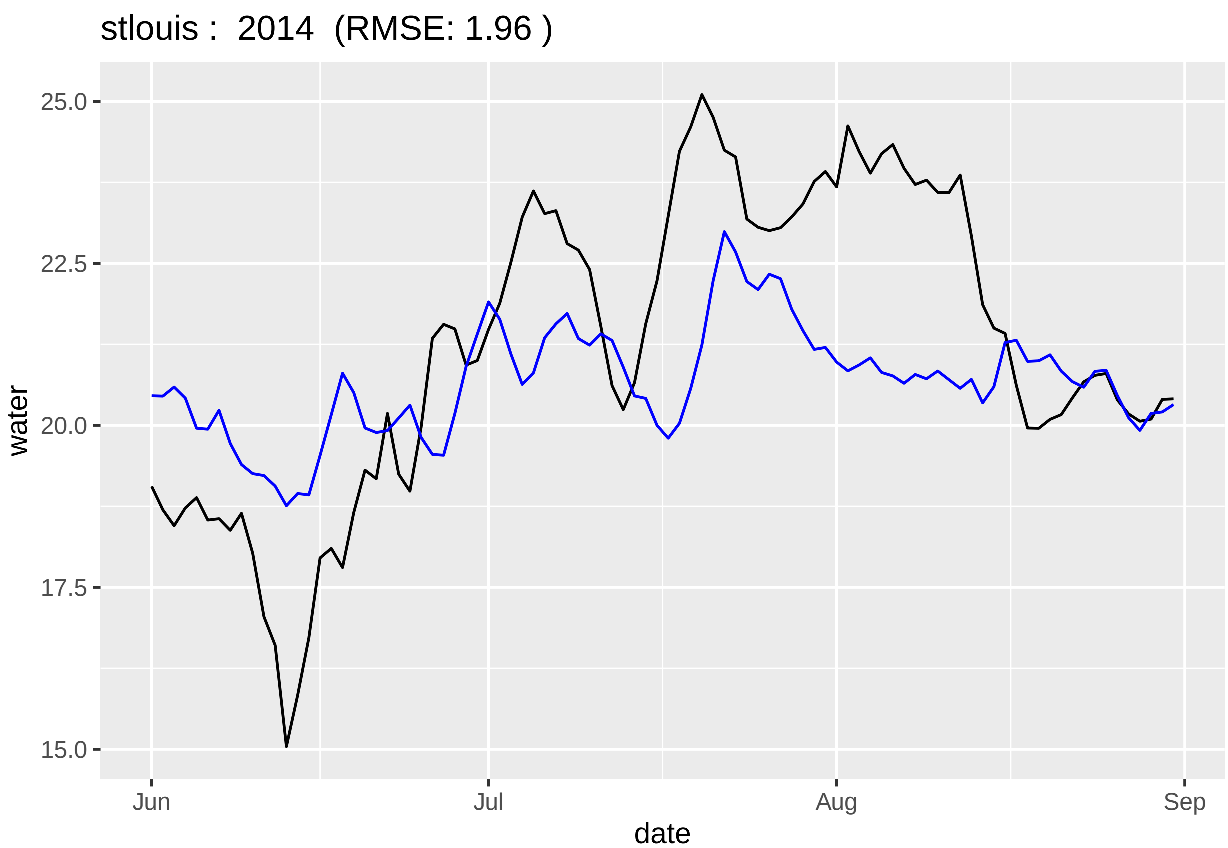

nipigon : 2008 (RMSE: 3.65 )
20
15
10
5
water
Jun
Jul
Aug
Sep
date



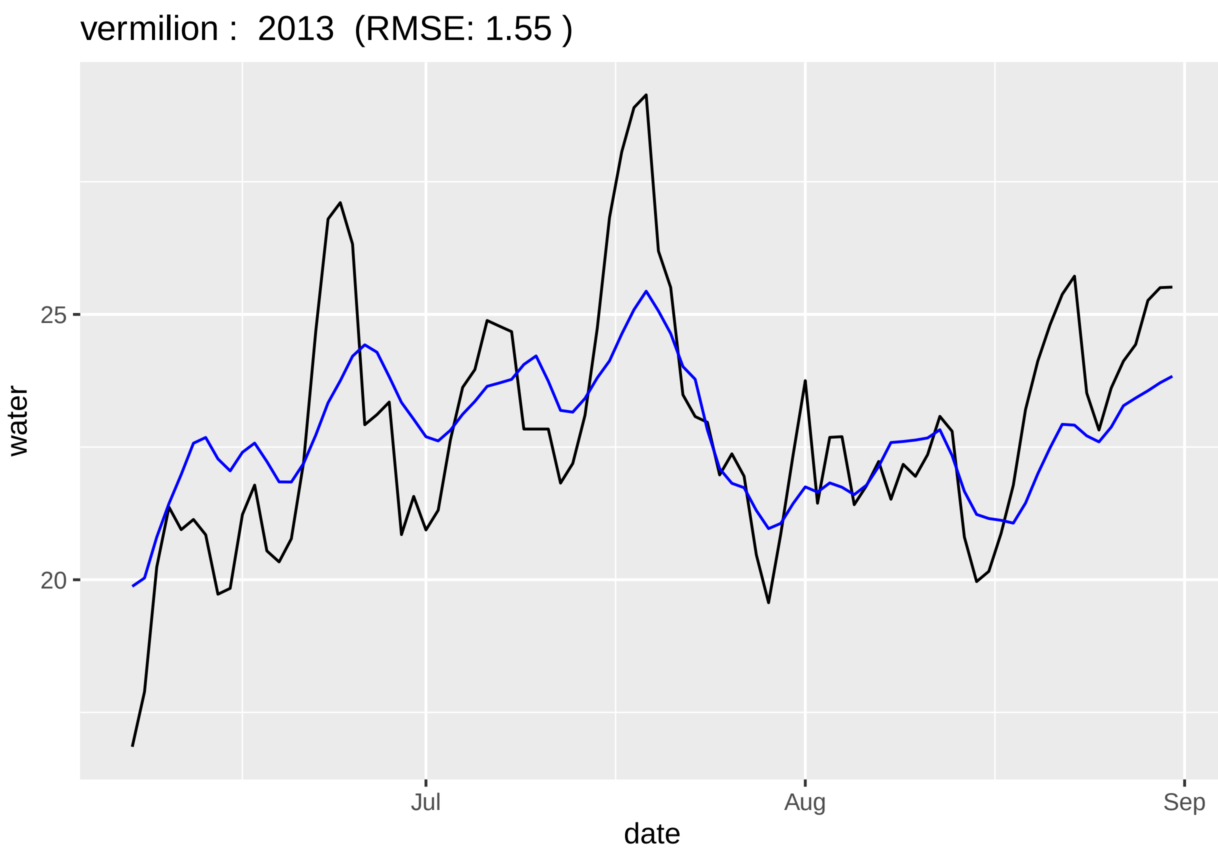
portage : 2012 (RMSE: 2.3 )
27.5
25.0
22.5
20.0
water
Jun
Jul
Aug
Sep
date


saginaw : 2012 (RMSE: 2.33 )
water
28
24
20
16
Jun
Jul
Aug
Sep
date


still : 2005 (RMSE: 1.85 )
26
24
22
20
18
water
Jun
Jul
Aug
Sep
date



stlouis : 2014 (RMSE: 1.96 )
25.0
22.5
20.0
17.5
15.0
water
Jun
Jul
Aug
Sep
date

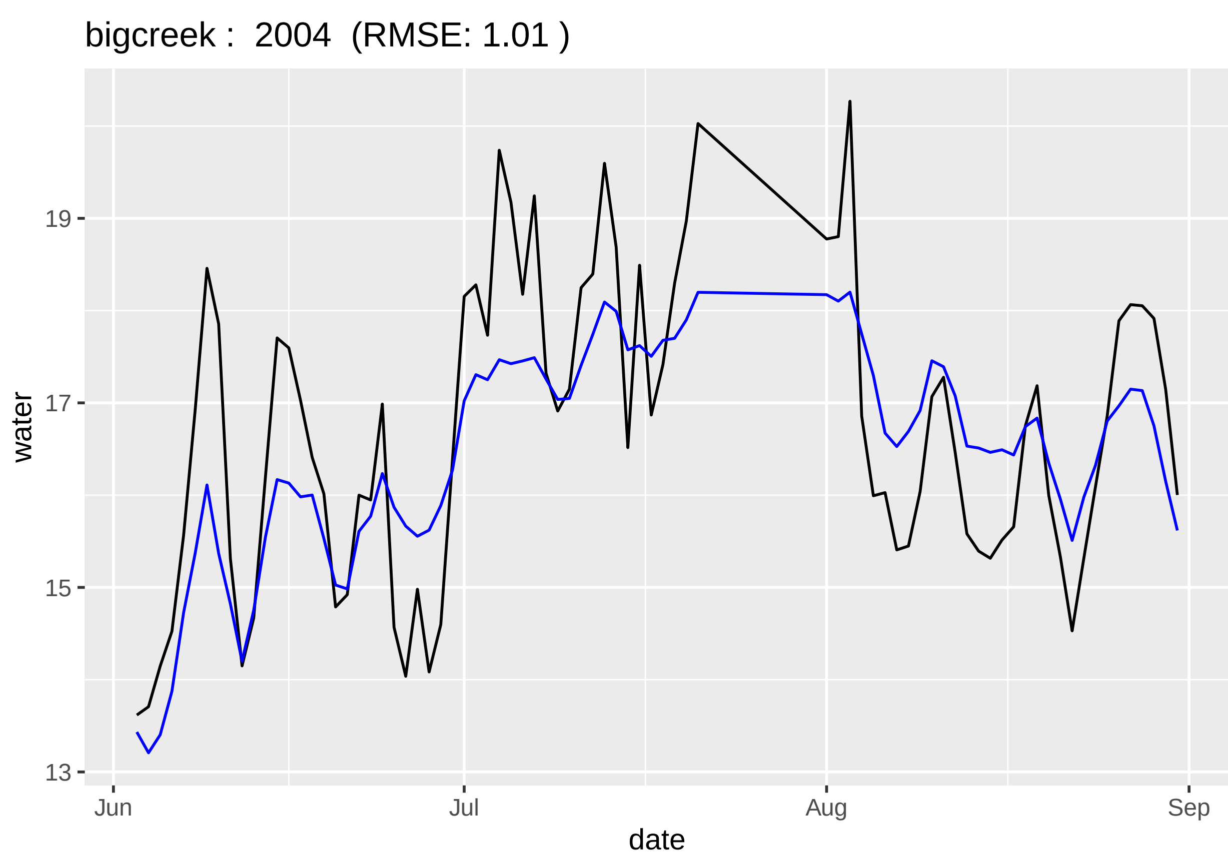


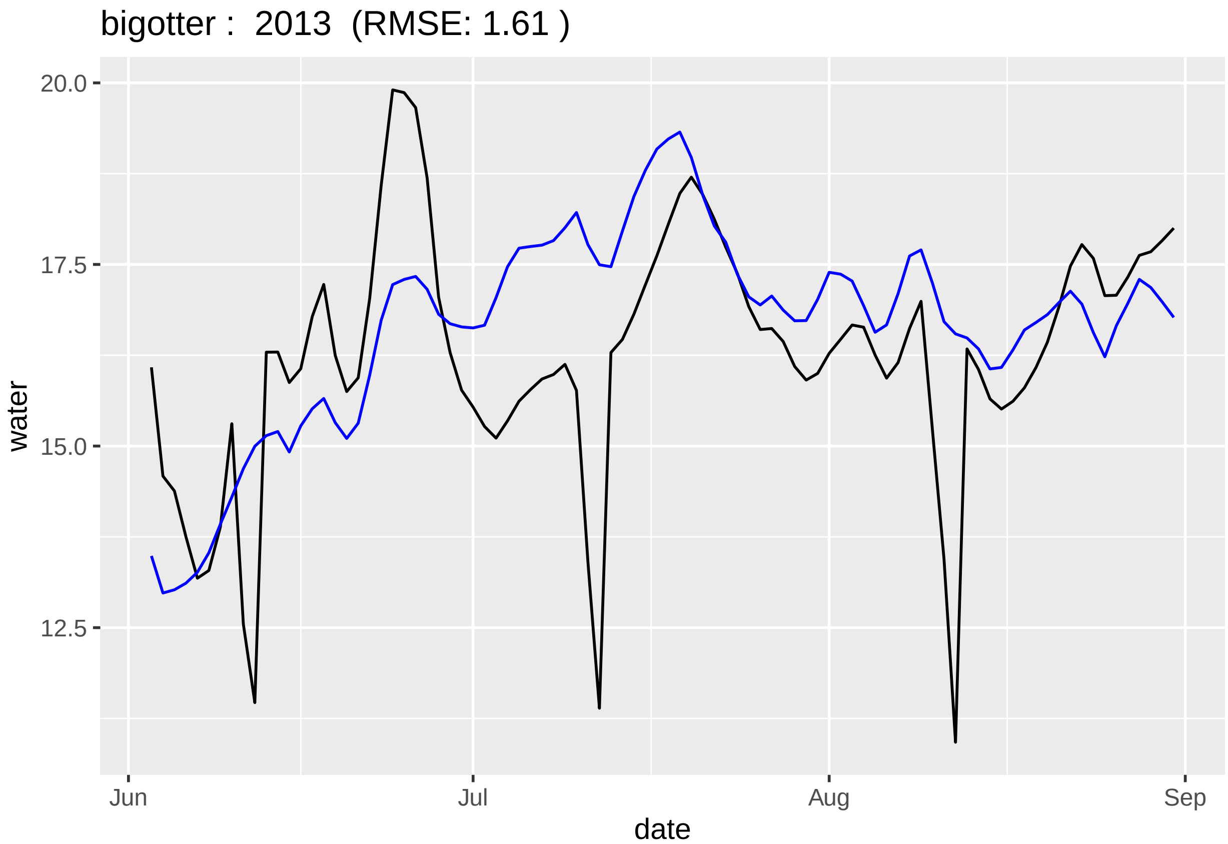

**Seasonal residual**

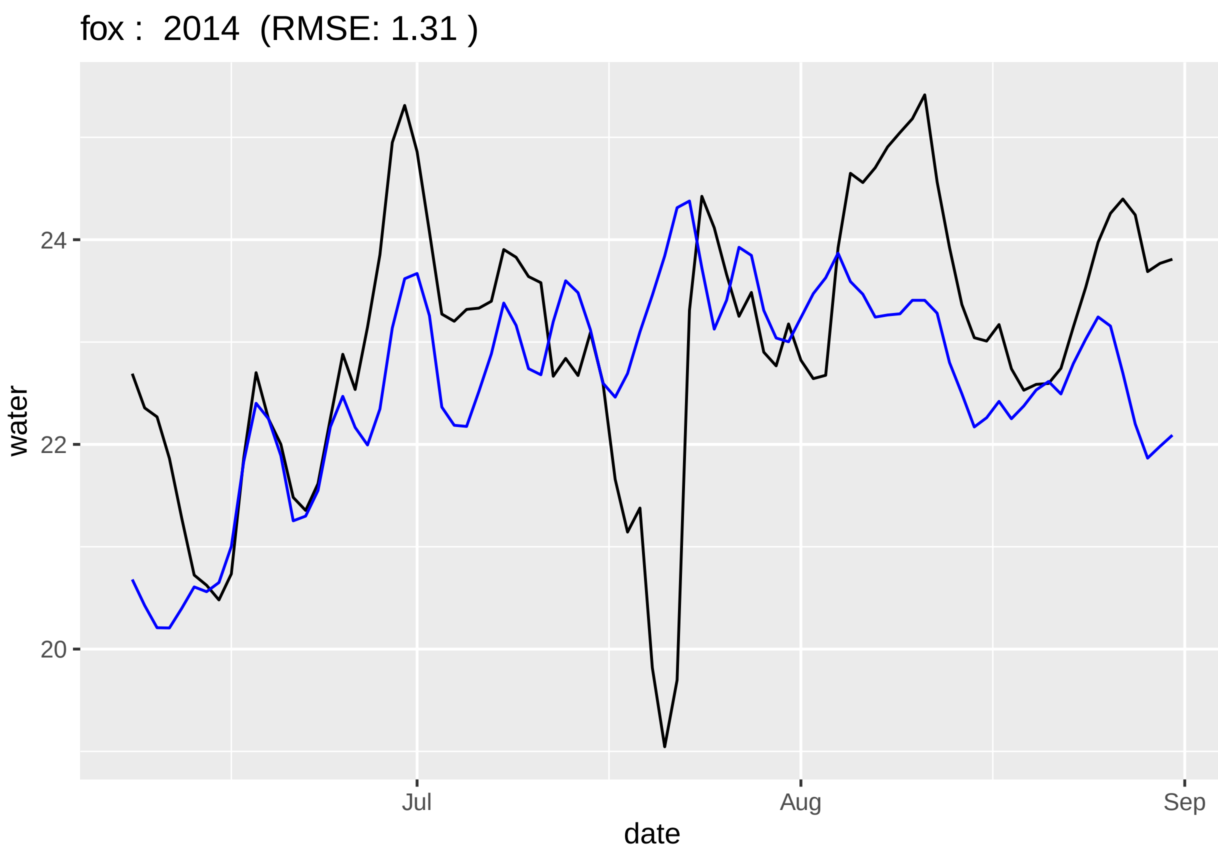
```
## Forms
spring.r <- vector("list", fold)
summer.r <- vector("list", fold)
fall.r <- vector("list", fold)
winter.r <- vector("list", fold)
annual.r <- vector("list", fold)

grand.sto <- list(spring.r, summer.r, fall.r, winter.r, annual.r)


## Iteration starts here
for (season in 1:5) {

  ## Get current season and its correspnding training/testing
  train <- grand_training[[season]]
  test <- grand_testing[[season]]

  # Each iteration
  for (i in 1:fold) {

    compare <- NA
    # select current dataset, and all unique location levels
    current.training <- train[[i]] %>% arrange(location, date)
```

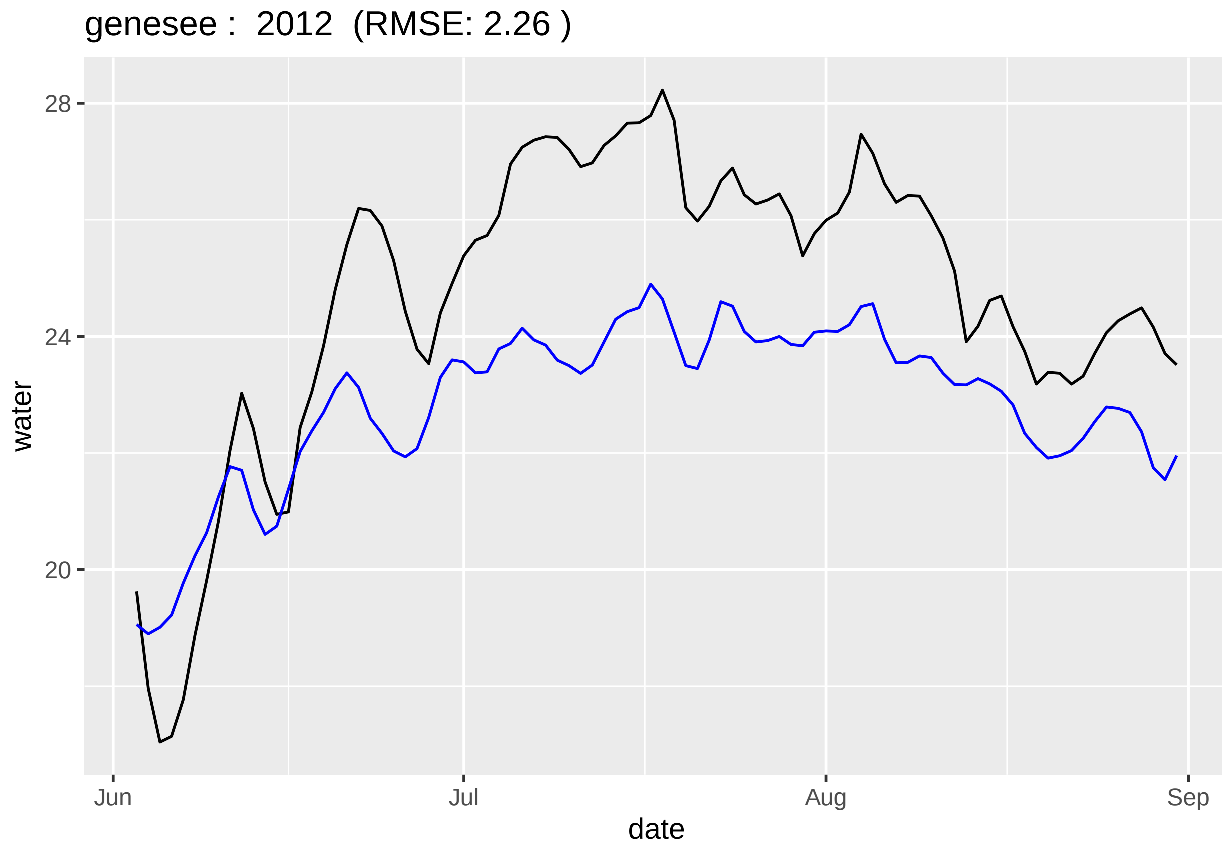
```
current.testing <- test[[i]] %>% arrange(location, date)

## 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.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
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))

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.ar))

# add up both components
```

```
    p <- merge(preds.annual, preds.residuals, by=c("location","date"))
    p[,"preds.ar"] <- p$preds.annual + p$pres.ar

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

    grand.sto[[season]][[i]] <- compare
  }
}


## Plots
for (loc in loc_seq) {
  df <- grand.sto[[2]][[1]]
  plot.df <- df[df$location == loc,]
  diff.ar <- round(sqrt(mean((plot.df$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.ar), color = "blue")+
          ggtitle(paste(
            loc, ": ", plot.df$year, " (RMSE:", diff.ar, ")"))
  print(pl)
}
```

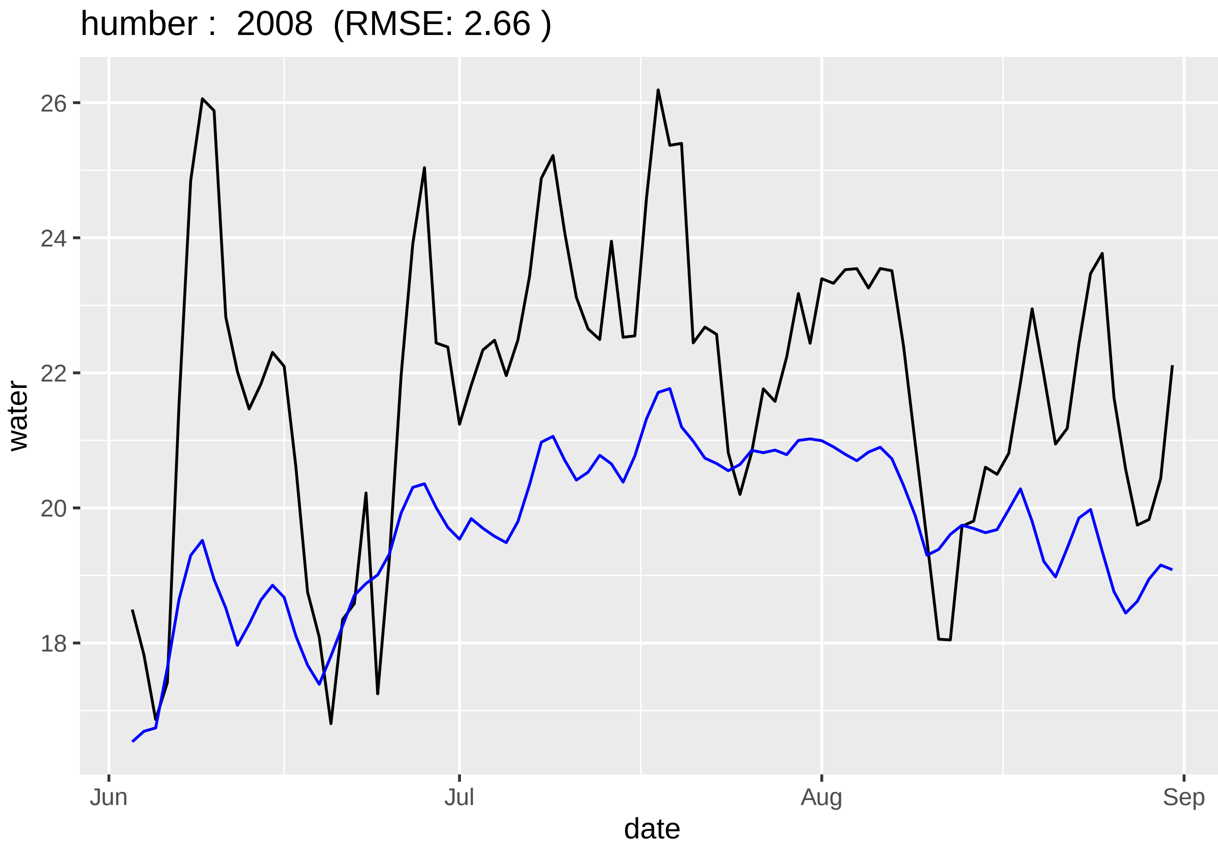


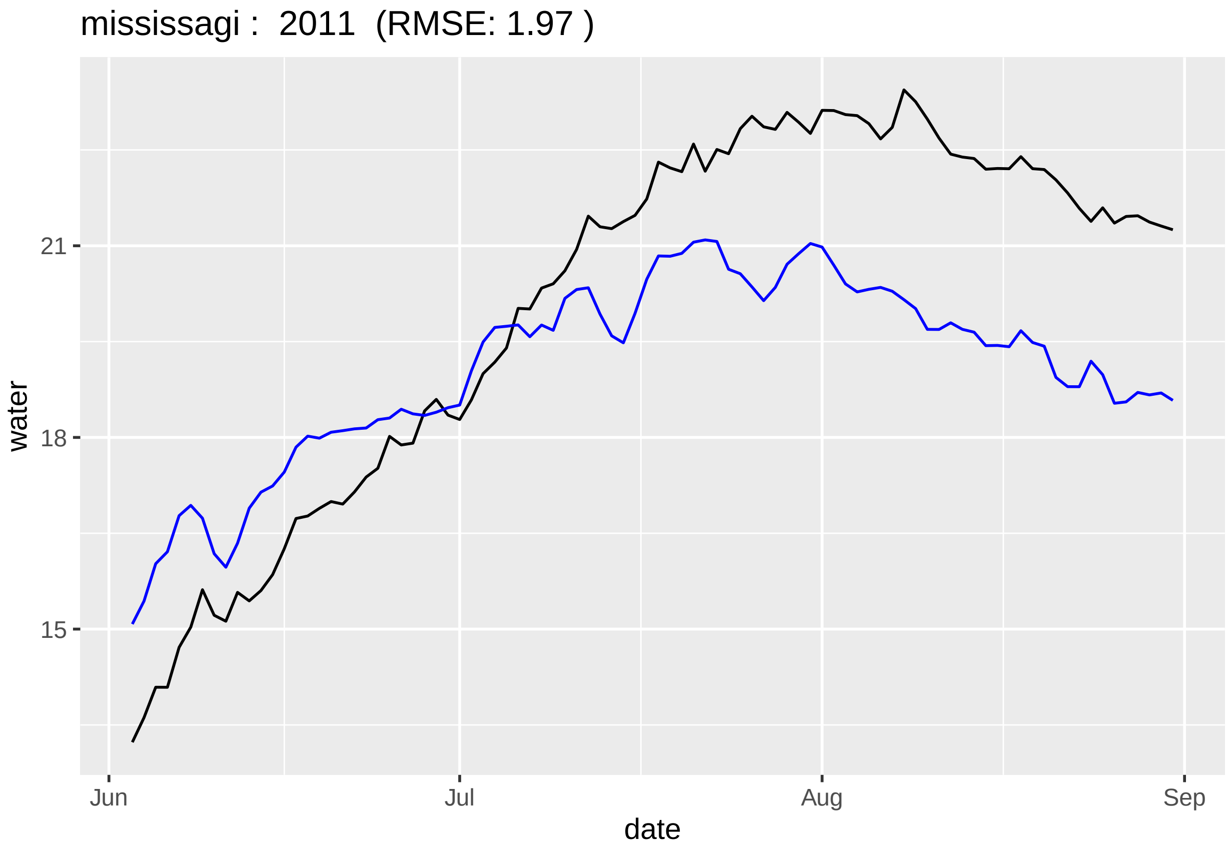

bigotter :  2013  (RMSE: 1.61 )
20.0
17.5
15.0
12.5
water
Jun
Jul
Aug
Sep
date

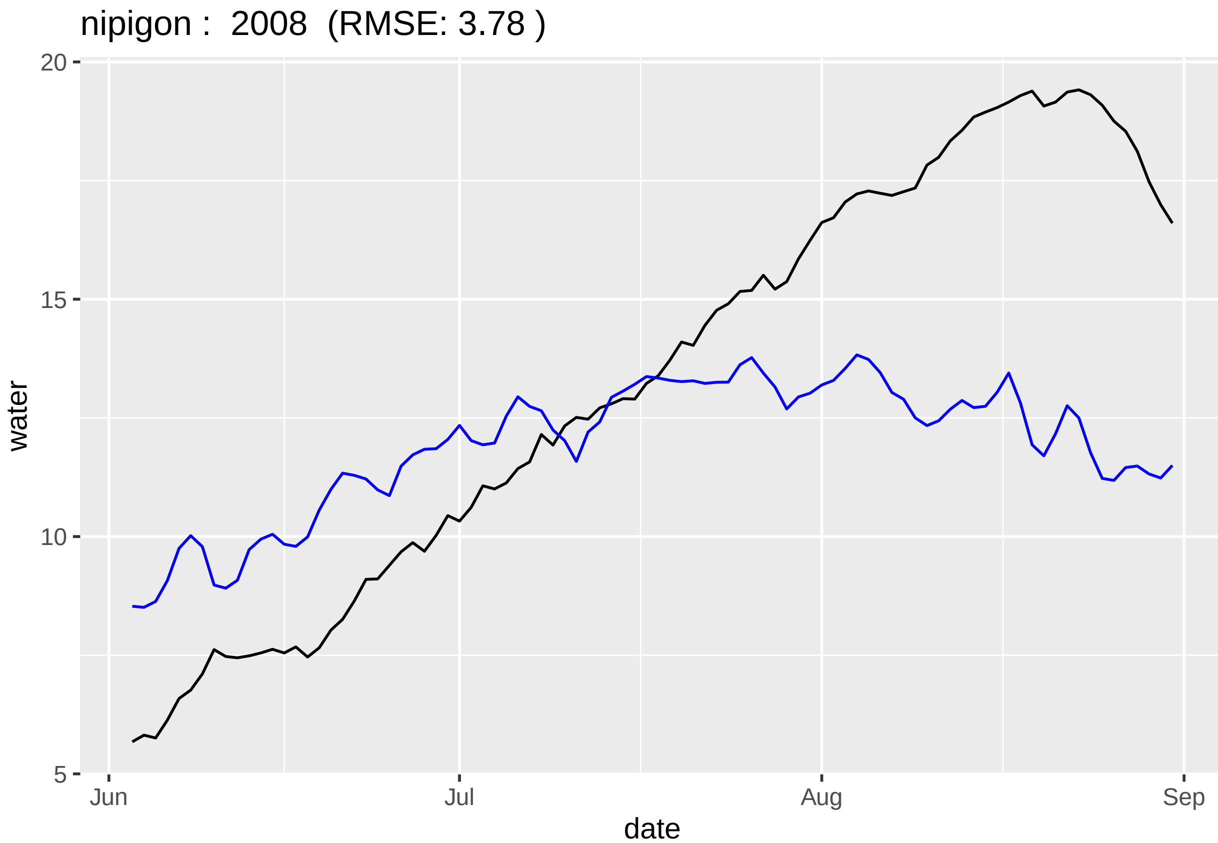

fox : 2014 (RMSE: 1.31 )
24
22
20
water
Jul
Aug
Sep
date

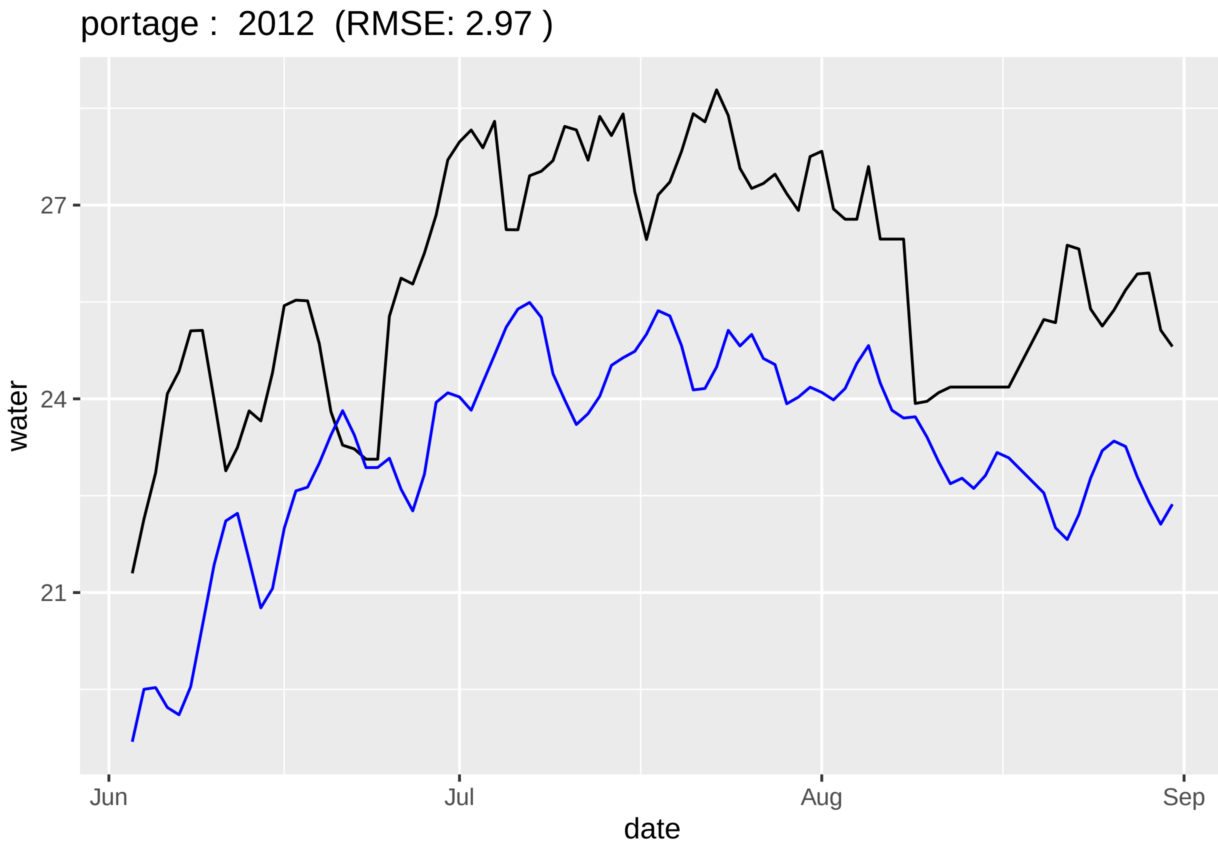


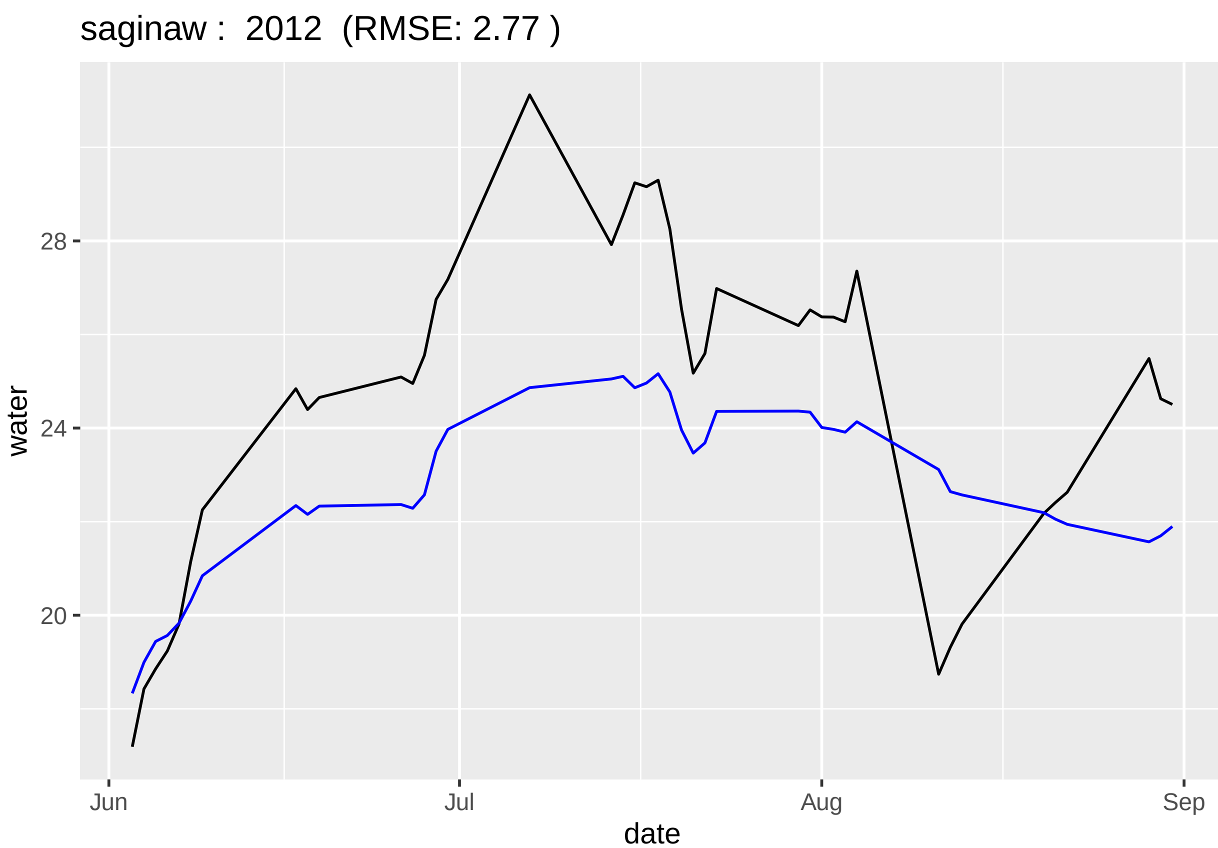
genesee : 2012 (RMSE: 2.26 )
28
24
20
water
Jun
Jul
Aug
Sep
date

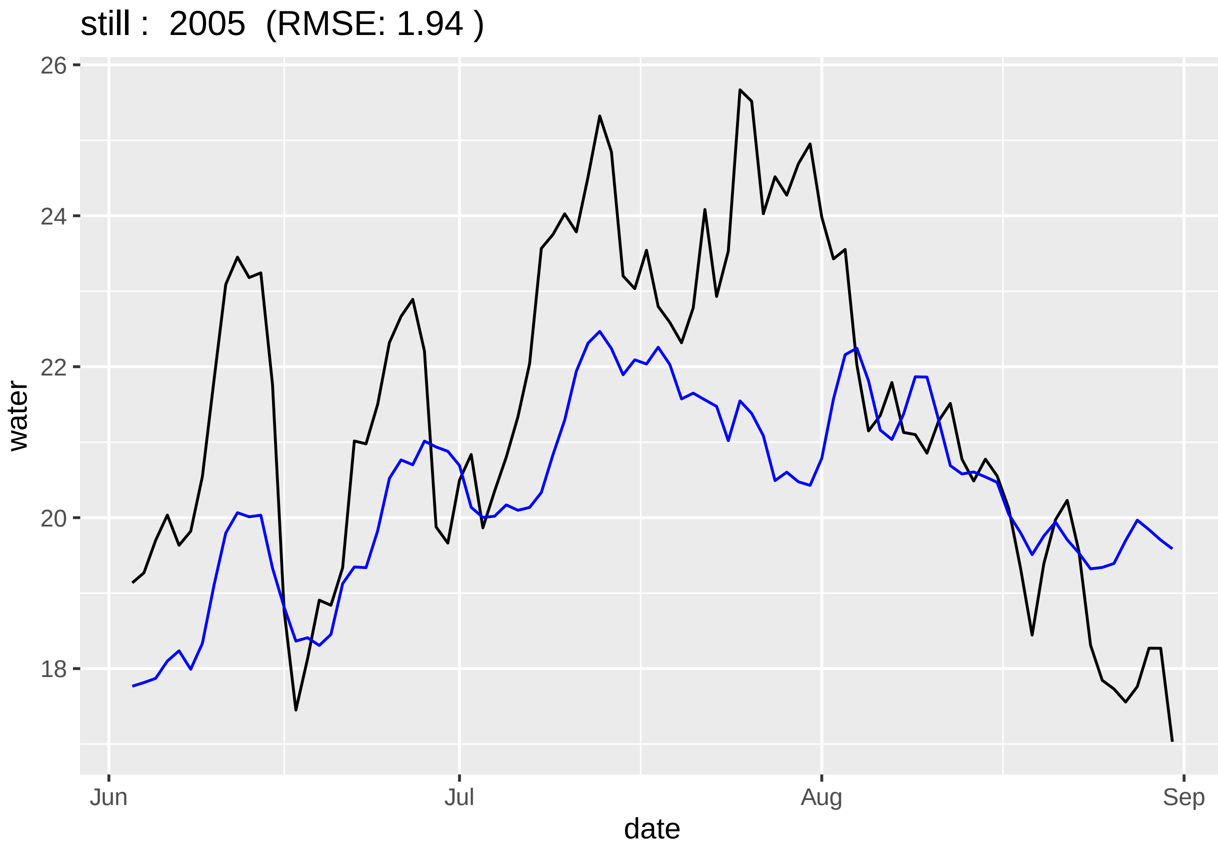

humber : 2008 (RMSE: 2.66 )
water
26
24
22
20
18
Jun
Jul
Aug
Sep
date

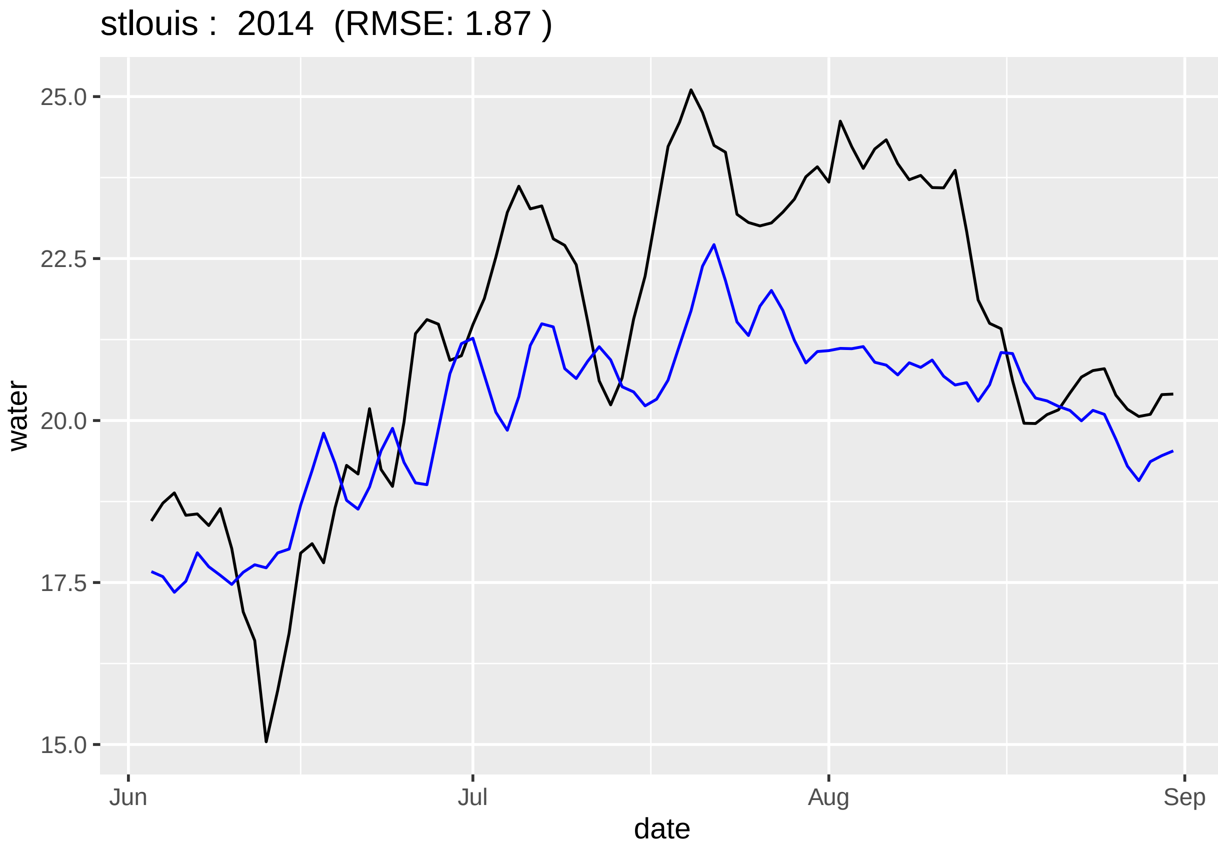

mississagi : 2011 (RMSE: 1.97 )
water
21
18
15
Jun
Jul
Aug
Sep
date

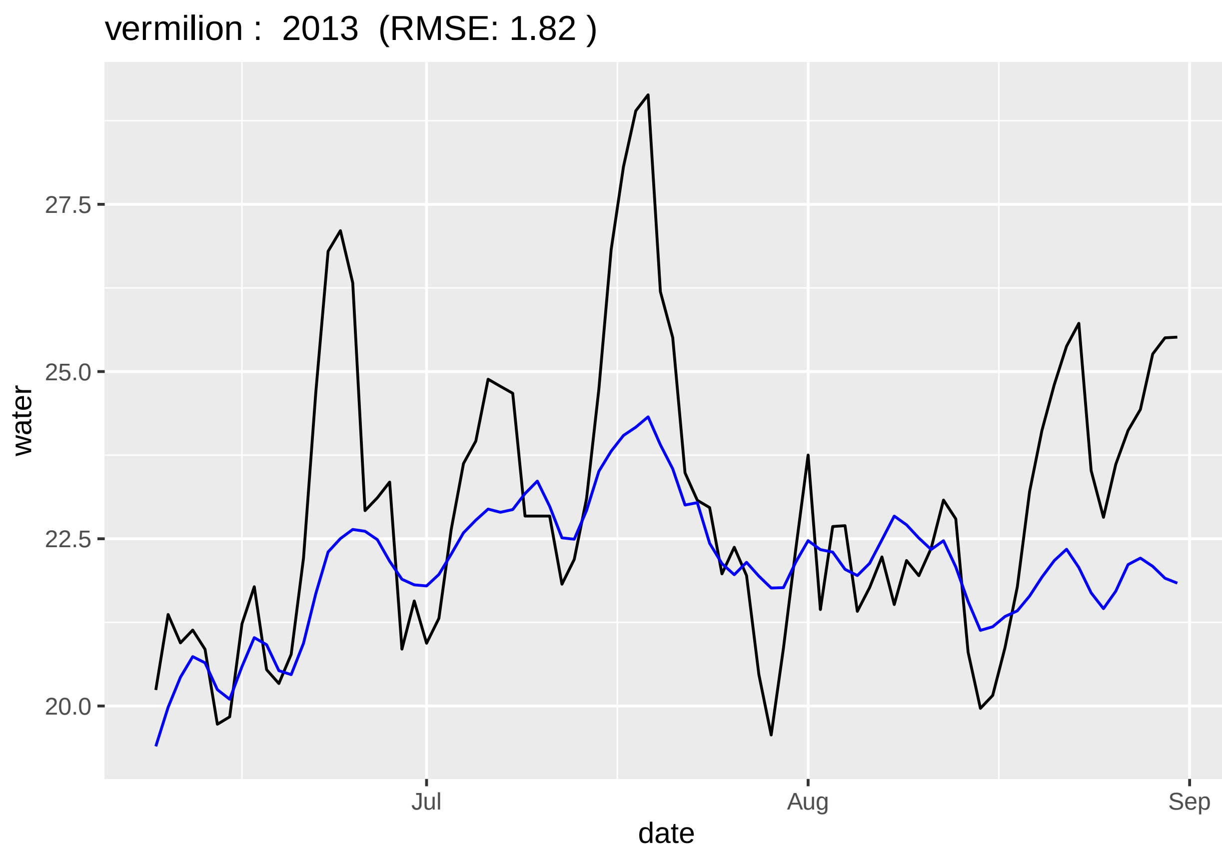

nipigon : 2008 (RMSE: 3.78 )
20
15
10
5
water
Jun
Jul
Aug
Sep
date


portage : 2012 (RMSE: 2.97 )
water
27
24
21
Jun
Jul
Aug
Sep
date


saginaw : 2012 (RMSE: 2.77 )
water
28
24
20
Jun
Jul
Aug
Sep
date


still : 2005 (RMSE: 1.94 )
26
24
22
20
18
water
Jun
Jul
Aug
Sep
date


stlouis : 2014 (RMSE: 1.87 )
water
25.0
22.5
20.0
17.5
15.0
Jun
Jul
Aug
Sep
date




**Non-linear**

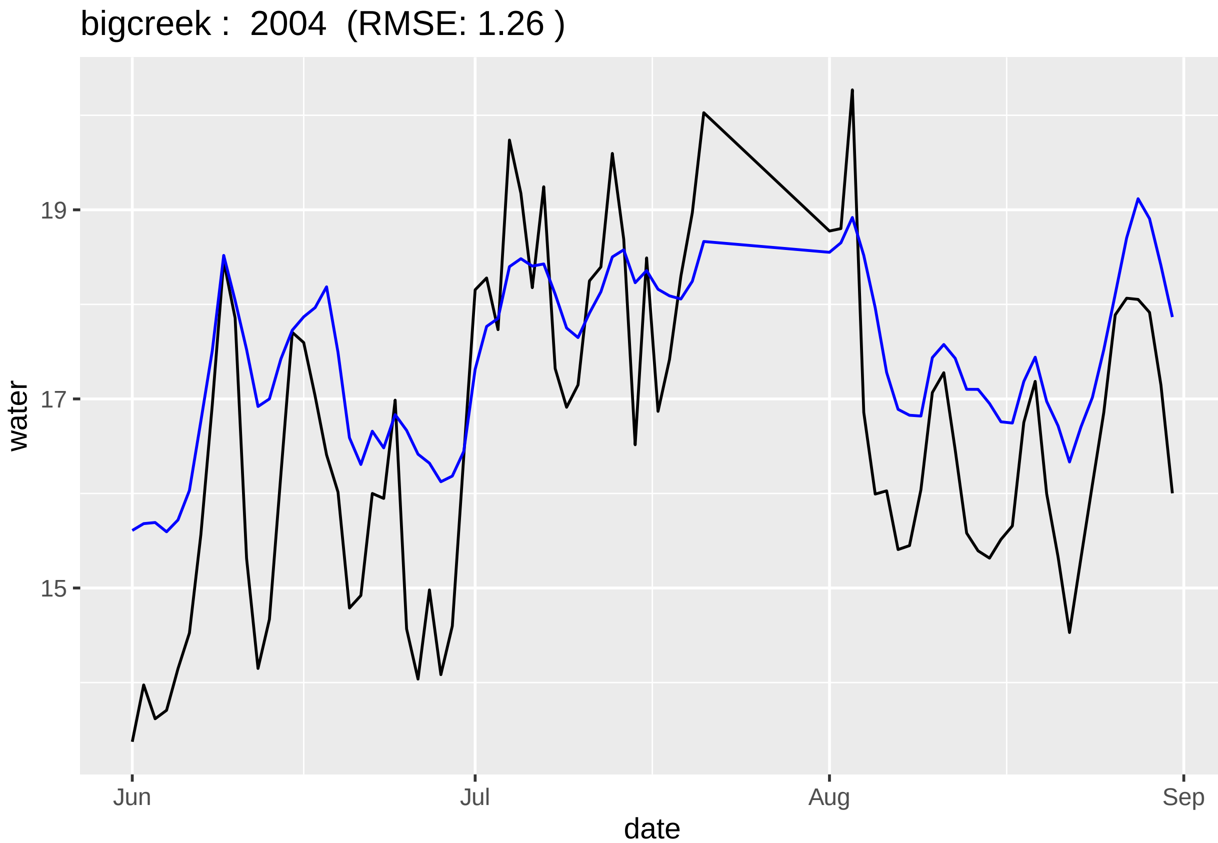
```
## starting parameters
coef.spring <- c(alpha=20, gamma=5,beta=9)
coef.summer <- c(alpha=25, gamma=7,beta=10)
coef.fall <- c(alpha=27, gamma=8,beta=10)
coef.winter <- c(alpha=11, gamma=8,beta=12)
coef.annual <- c(alpha=20, gamma=5,beta=9)

coef.list <- list(coef.spring, coef.summer, coef.fall, coef.winter, coef.annual)


spring.r <- vector("list", fold)
summer.r <- vector("list", fold)
fall.r <- vector("list", fold)
winter.r <- vector("list", fold)
annual.r <- vector("list", fold)

grand.nonlinear <- list(spring.r, summer.r, fall.r, winter.r, annual.r)


## Iteration starts here
for (season in c(1,2,3,5)) {

  ## Get current season and its correspnding training/testing
```

```
  train <- grand_training[[season]]
  test <- grand_testing[[season]]

  ## 10 fold iteration starts here
  for (i in 1:fold){

    compare <- NA
    # select current dataset, and all unique location levels
    current.training <- train[[i]]
    current.testing <- test[[i]]

    # model training and predicting
    model <- nlme(waterT ~ alpha / (1 + exp(gamma * (beta - airT))),
                  fixed = list(alpha~1,gamma~1,beta~1),
                  random = gamma ~ 1|location,
                  start = coef.list[[season]],
                  data=current.training,
                  control = list(maxIter = 1000, tolerance = 1e-02))

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

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

    grand.nonlinear[[season]][[i]] <- compare
  }
}

View(grand.nonlinear[[2]])
```

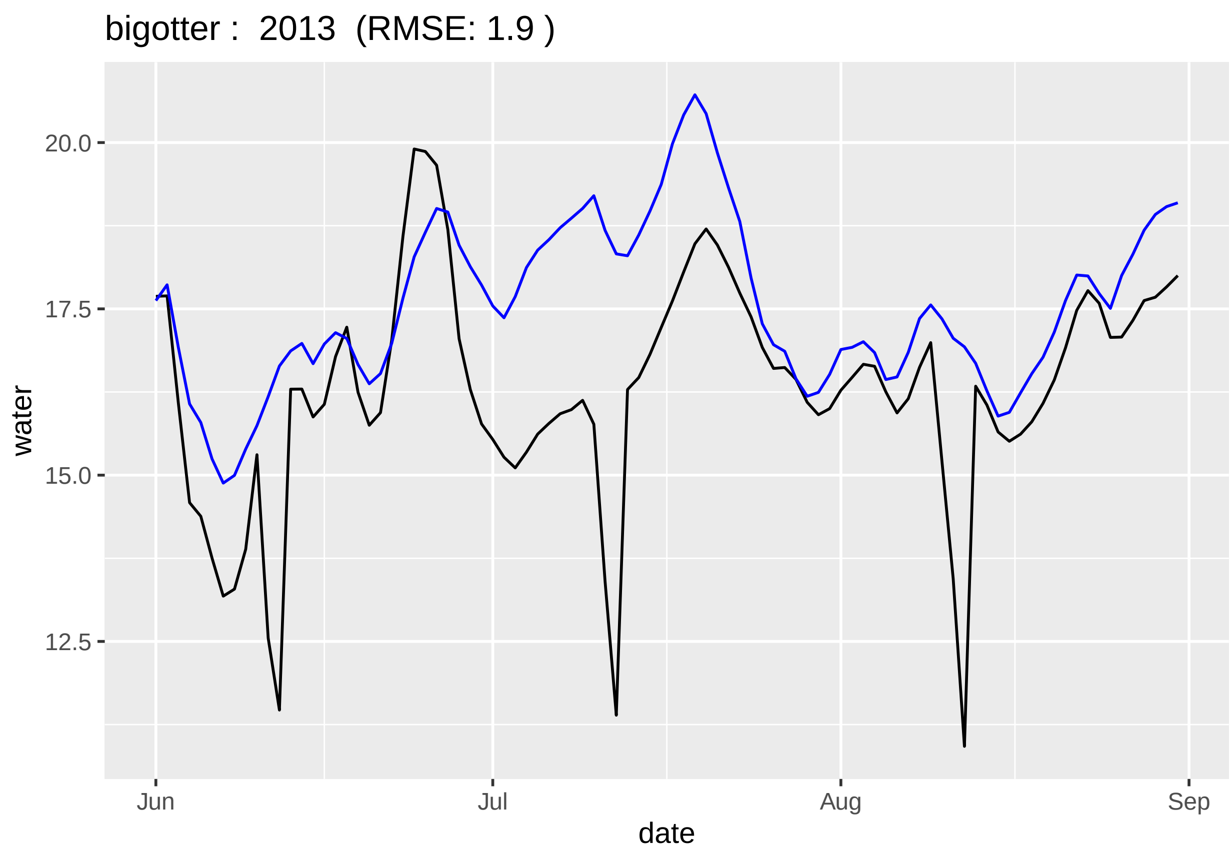
**lag5 with flow**

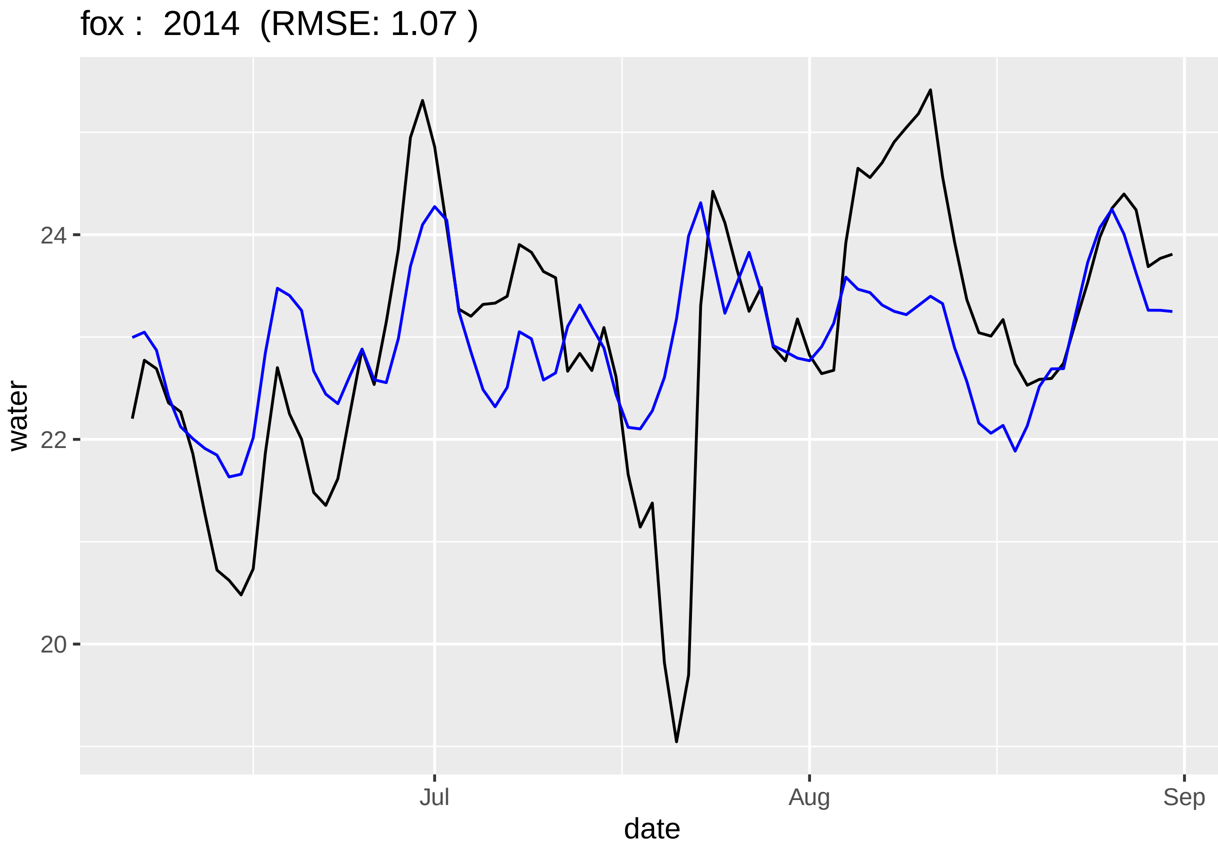
```
## Forms
ctrl = lmeControl(opt='optim')
form3 <- water ~ air + dmean_1 + dmean_2 + dmean_3 + dmean_4 + dmean_5 + I(1/flow)

spring.r <- vector("list", fold)
summer.r <- vector("list", fold)
fall.r <- vector("list", fold)
winter.r <- vector("list", fold)
annual.r <- vector("list", fold)

grand.flow <- list(spring.r, summer.r, fall.r, winter.r, annual.r)


## Iteration starts here
for (season in 1:5) {

  ## Get current season and its correspnding training/testing
  train <- grand_training[[season]]
  test <- grand_testing[[season]]
```

```r
  ## 10 fold iteration starts here
  for (i in 1:fold){

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

    # model training and predicting
    model.ar <- lme(form3,
                    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))

    grand.flow[[season]][[i]] <- compare
  }
}

View(grand.flow[[4]][[1]])

## Plots
for (loc in loc_seq) {
  df <- grand.flow[[2]][[1]]
  plot.df <- df[df$location == loc,]
  diff.ar <- round(sqrt(mean((plot.df$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.ar), color = "blue")+
          ggtitle(paste(
            loc, ": ", plot.df$year, " (RMSE:", diff.ar, ")"))
  print(pl)
}
```

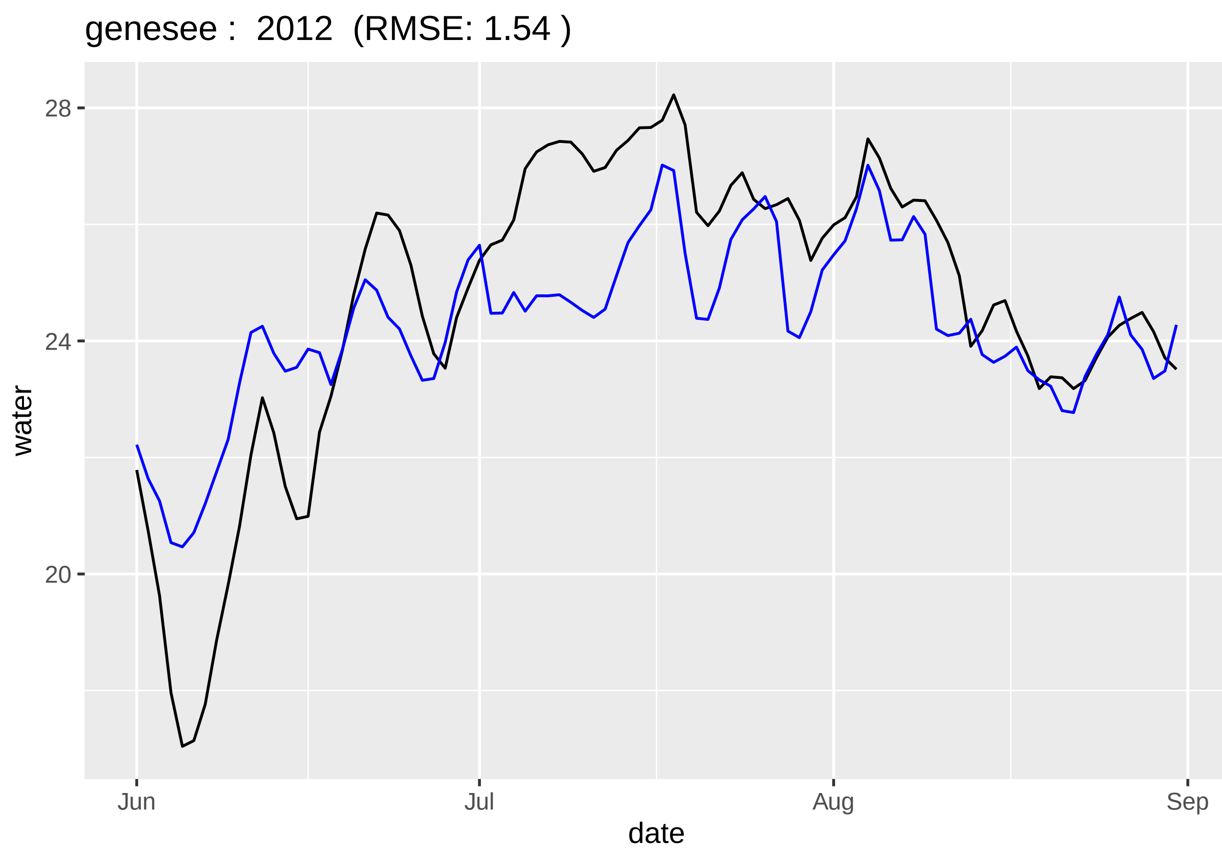


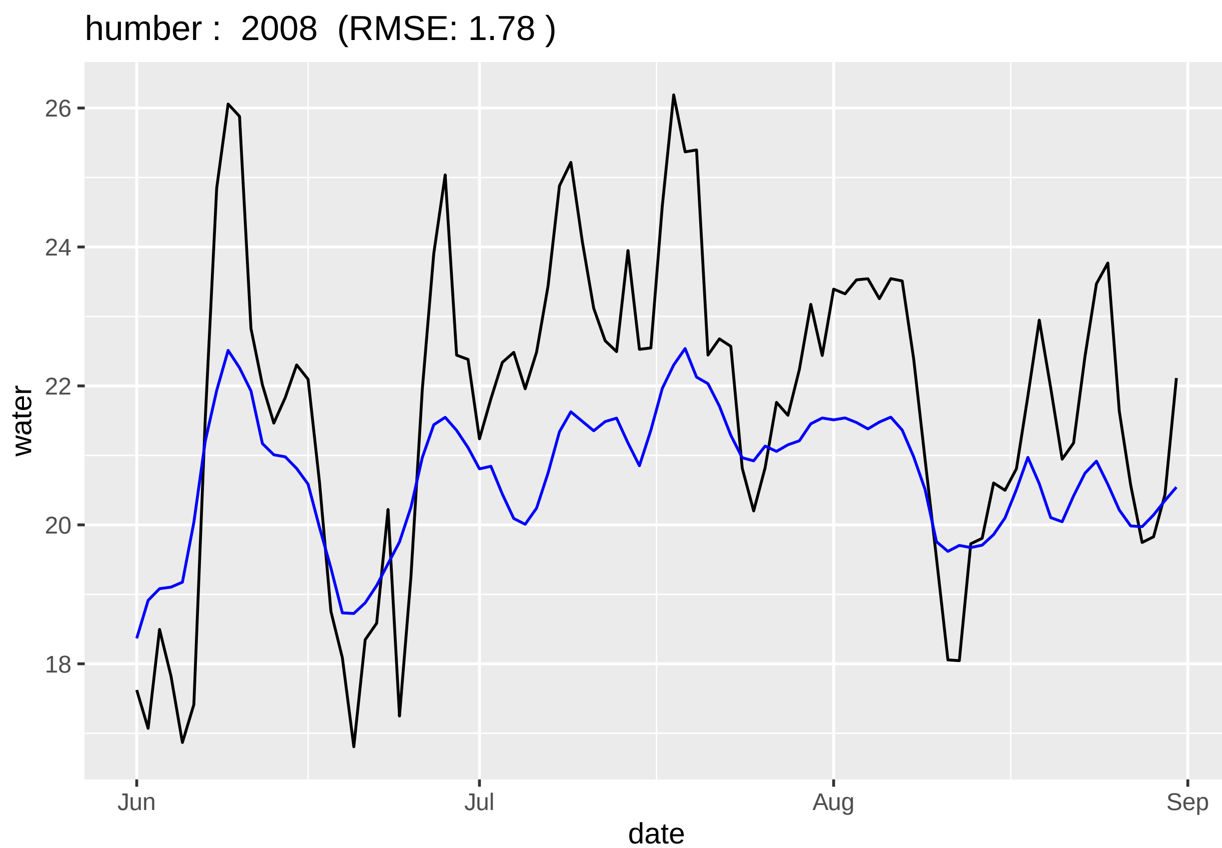
bigcreek : 2004 (RMSE: 1.26 )
water
19
17
15
Jun
Jul
Aug
Sep
date

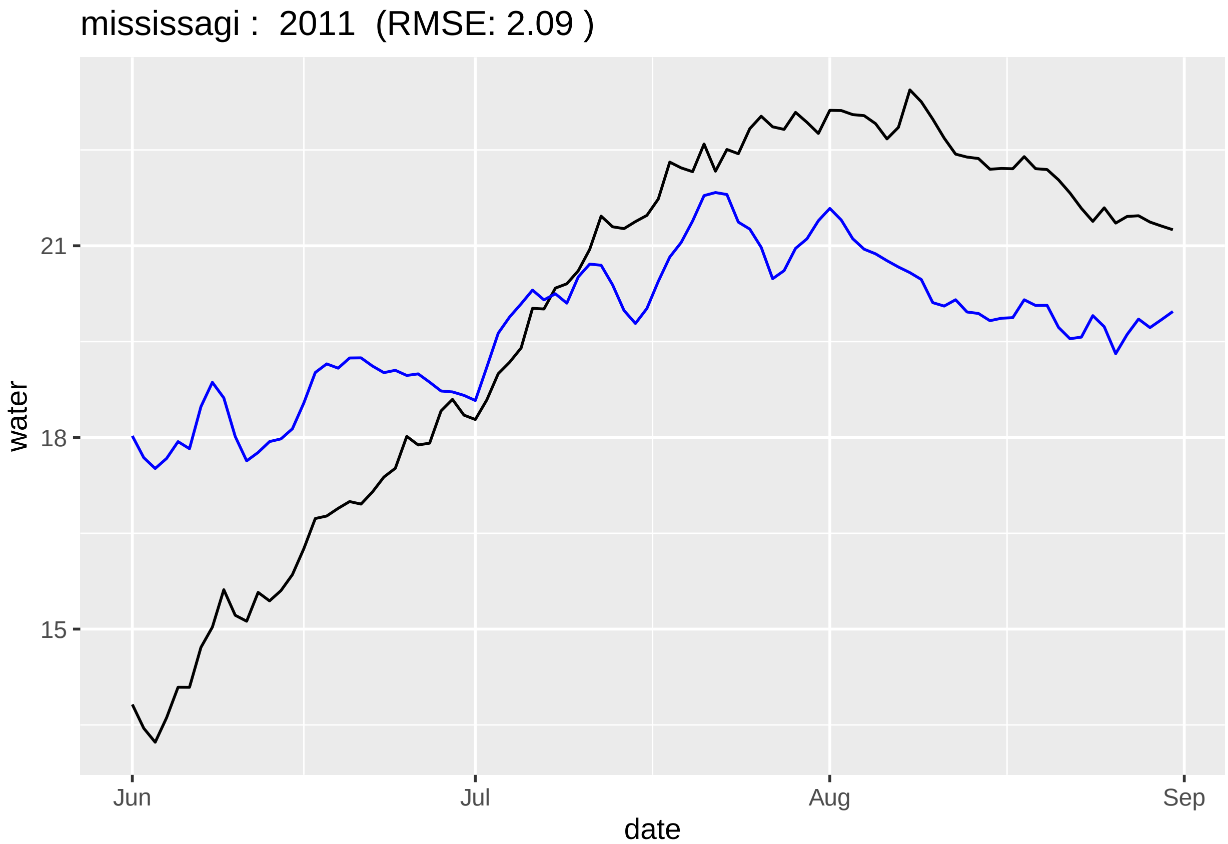

bigotter : 2013 (RMSE: 1.9 )
water
20.0
17.5
15.0
12.5
Jun
Jul
Aug
Sep
date



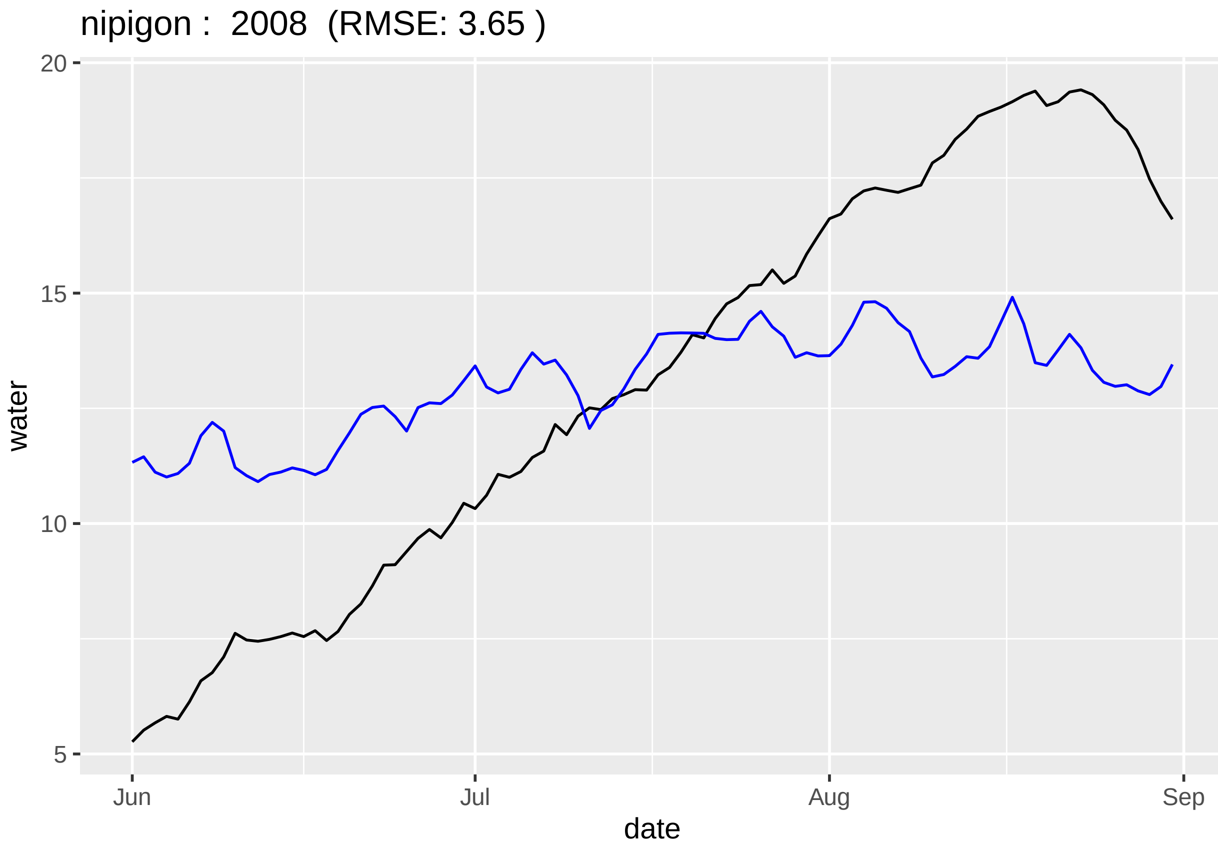
fox : 2014 (RMSE: 1.07 )
water
24
22
20
Jul
Aug
Sep
date

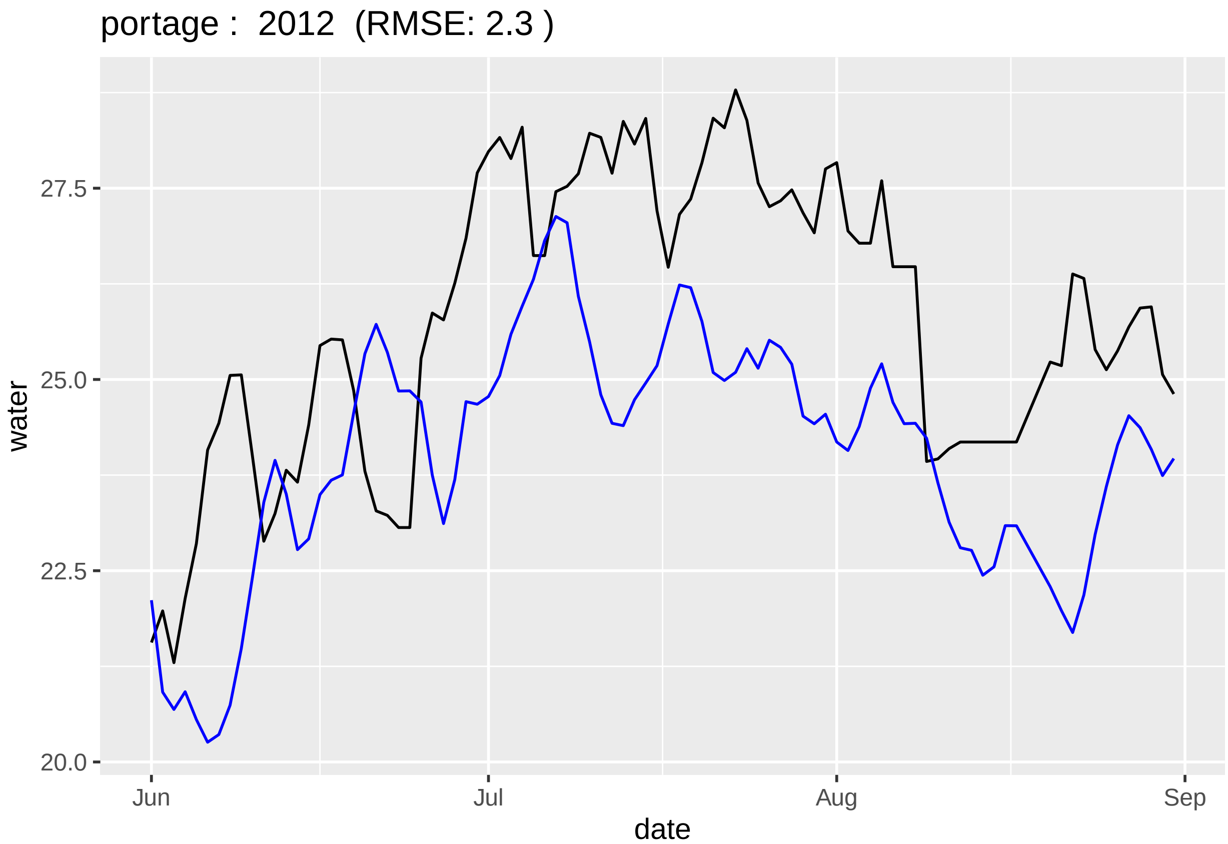

genesee : 2012 (RMSE: 1.54 )
28
24
20
water
Jun
Jul
Aug
Sep
date


humber :  2008  (RMSE: 1.78 )
26
24
22
20
18
water
Jun
Jul
Aug
Sep
date

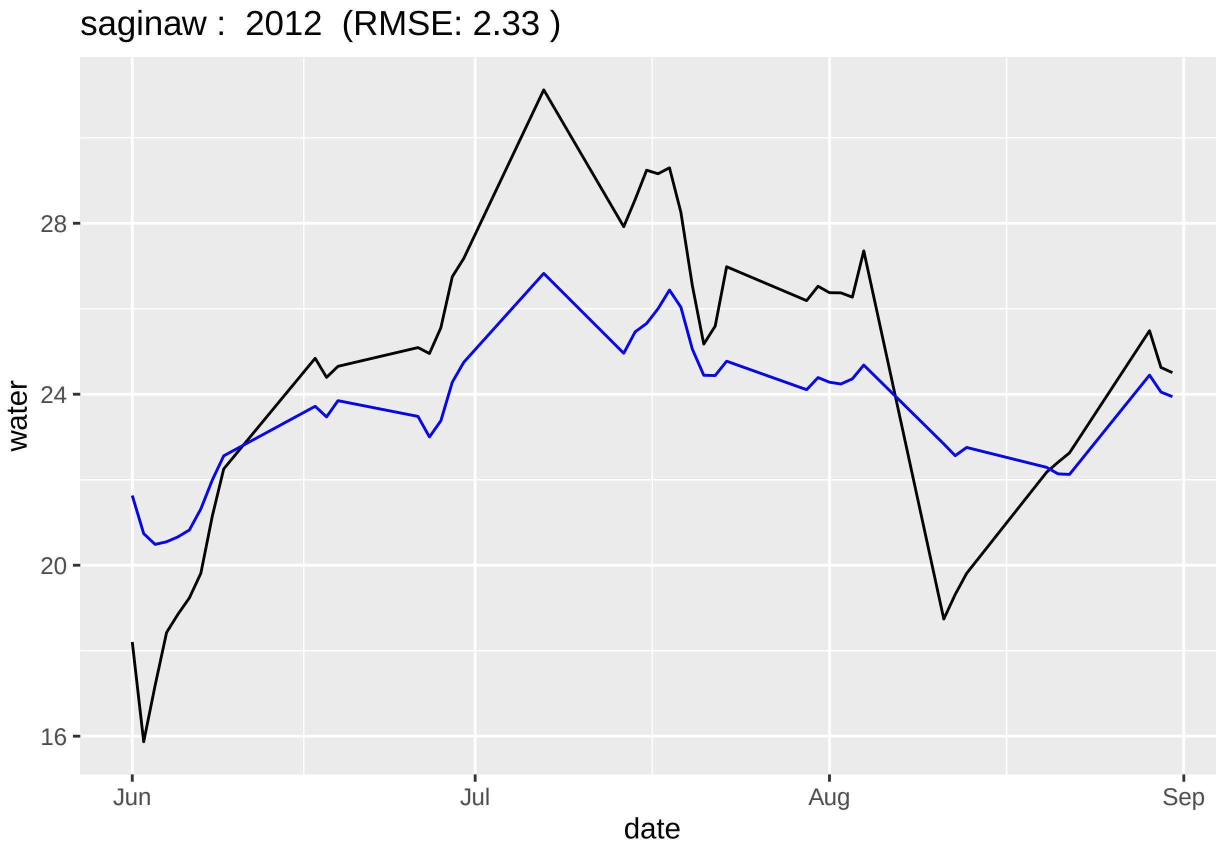


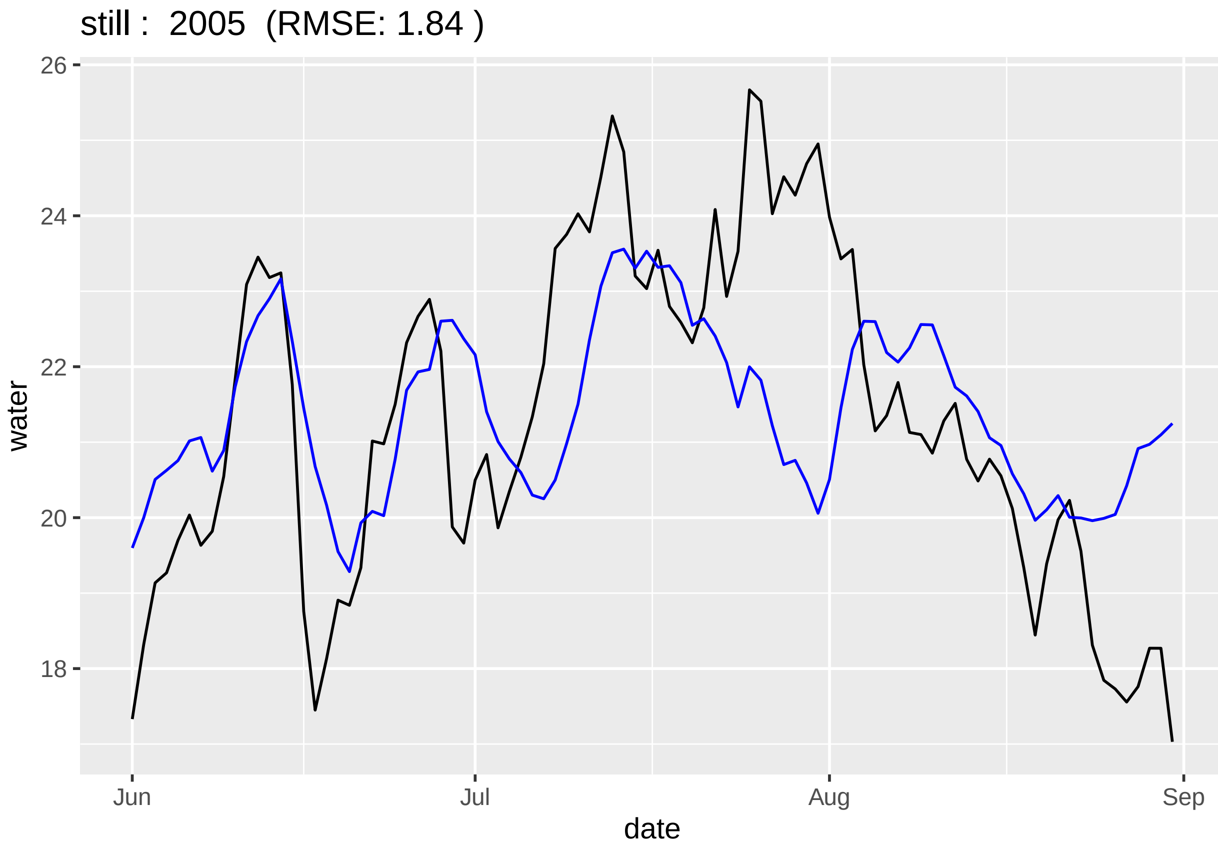
mississagi : 2011 (RMSE: 2.09 )
water
21
18
15
Jun
Jul
Aug
Sep
date

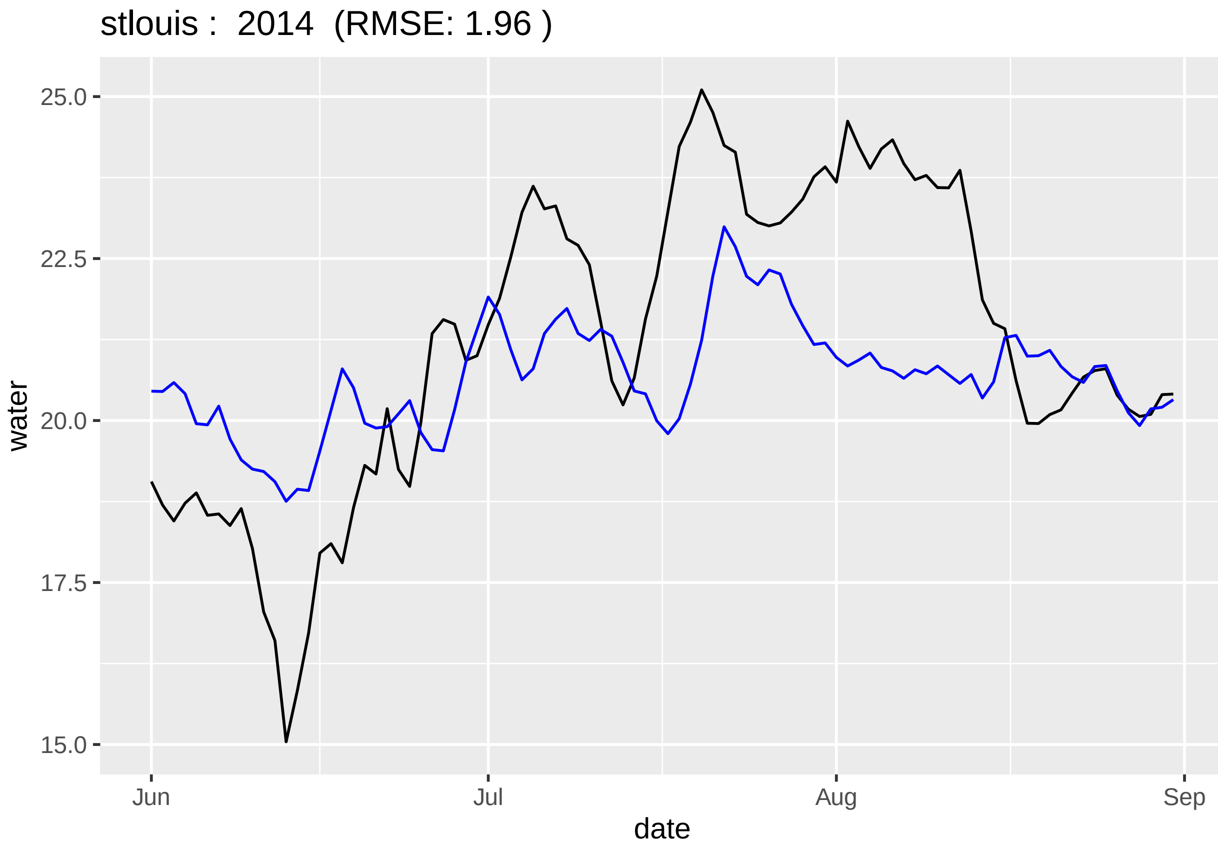

nipigon : 2008 (RMSE: 3.65 )
20
15
10
5
water
Jun
Jul
Aug
Sep
date

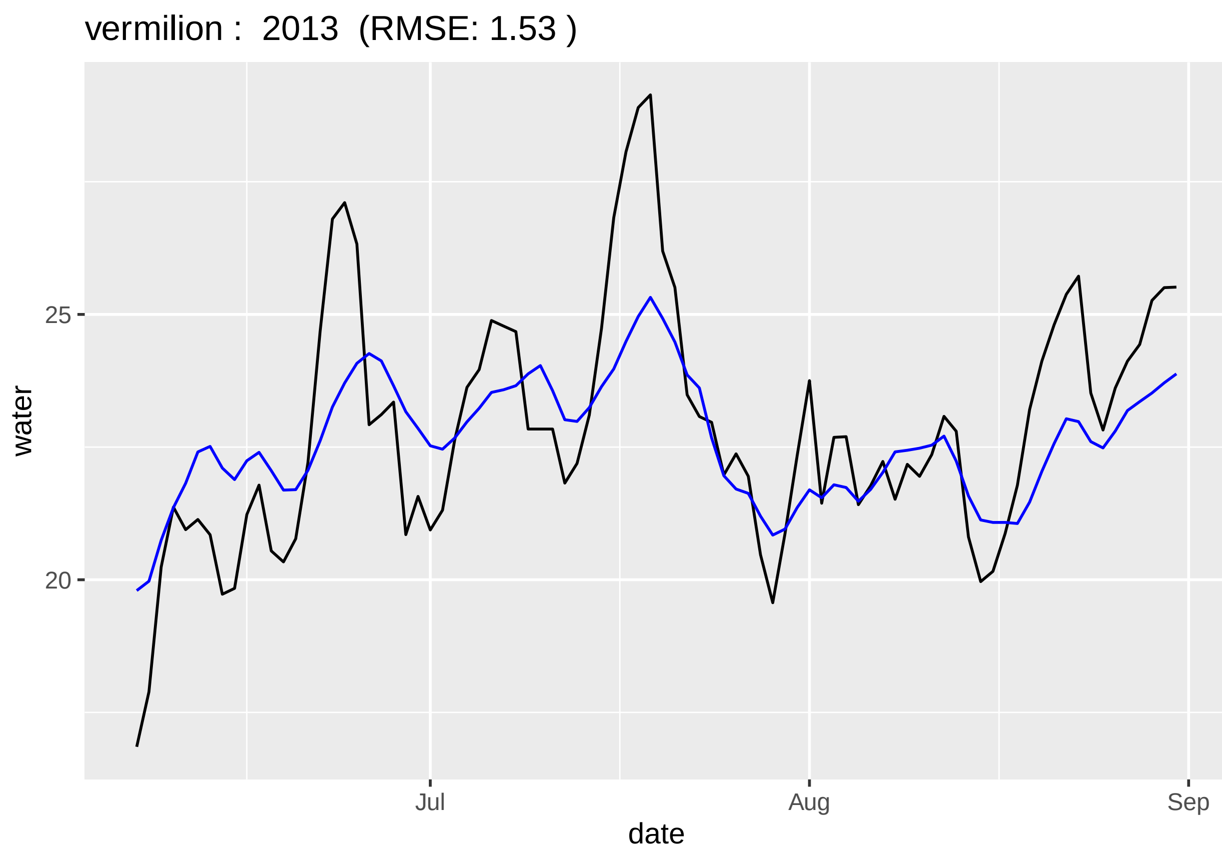


portage : 2012 (RMSE: 2.3 )
water
27.5
25.0
22.5
20.0
Jun
Jul
Aug
Sep
date


saginaw : 2012 (RMSE: 2.33 )
water
28
24
20
16
Jun
Jul
Aug
Sep
date


still : 2005 (RMSE: 1.84 )
26
24
22
20
18
water
Jun
Jul
Aug
Sep
date


stlouis : 2014 (RMSE: 1.96 )
25.0
22.5
20.0
17.5
15.0
water
Jun
Jul
Aug
Sep
date




### Comparison at the end

We first compare each model within one specific season.

```
## Spring::
spring.compare <- data.frame(
  lag5 = colMeans(cal.rmse(grand.lag5[[1]], fold)),
  sto = colMeans(cal.rmse(grand.sto[[1]], fold)),
  #nonlinear = colMeans(cal.rmse(grand.nonlinear[[1]], fold)),
  flow = colMeans(cal.rmse(grand.flow[[1]], fold)))

colMeans(spring.compare)
```

```
##    lag5      sto     flow
## 3.082264 3.377733 2.976439
```

```
knitr::kable(spring.compare, digits = 3)
```

| | lag5 | sto | flow |
|---|---|---|---|
| bigcreek | 2.149 | 3.539 | 2.126 |
| bigotter | 1.830 | 2.701 | 1.656 |
| fox | 3.786 | 3.852 | 3.824 |
| genesee | 3.599 | 2.961 | 2.978 |
| humber | 3.384 | 3.089 | 3.306 |
| mississagi | 2.556 | 3.472 | 2.505 |
| nipigon | 2.164 | 4.462 | 2.118 |
| portage | 2.857 | 4.048 | 2.848 |

| | lag5 | sto | flow |
|---|---|---|---|
| saginaw | 3.135 | 2.529 | 3.165 |
| still | 3.673 | 1.923 | 3.681 |
| stlouis | 3.917 | 4.492 | 3.962 |
| vermilion | 3.937 | 3.465 | 3.548 |

```
## Summer::
summer.compare <- data.frame(
  lag5 = colMeans(cal.rmse(grand.lag5[[2]], fold), na.rm=T),
  sto = colMeans(cal.rmse(grand.sto[[2]], fold), na.rm=T),
  #nonlinear = colMeans(cal.rmse(grand.nonlinear[[2]], fold)),
  flow = colMeans(cal.rmse(grand.flow[[2]], fold), na.rm=T))

colMeans(summer.compare)
```

```
##     lag5      sto     flow
## 1.886360 2.052192 1.884769
```

```
knitr::kable(summer.compare, digits = 3)
```

| | lag5 | sto | flow |
|---|---|---|---|
| bigcreek | 1.169 | 1.358 | 1.156 |
| bigotter | 1.477 | 1.767 | 1.431 |
| fox | 1.640 | 1.843 | 1.637 |
| genesee | 2.130 | 2.231 | 2.265 |
| humber | 2.103 | 2.573 | 2.033 |
| mississagi | 1.786 | 1.584 | 1.773 |
| nipigon | 2.845 | 2.683 | 2.843 |
| portage | 2.319 | 2.711 | 2.307 |
| saginaw | 2.267 | 2.596 | 2.262 |
| still | 1.550 | 1.754 | 1.553 |
| stlouis | 1.834 | 1.799 | 1.829 |
| vermilion | 1.517 | 1.728 | 1.528 |

```
## Fall::
fall.compare <- data.frame(
  lag5 = colMeans(cal.rmse(grand.lag5[[3]], fold)),
  sto = colMeans(cal.rmse(grand.sto[[3]], fold)),
  #nonlinear = colMeans(cal.rmse(grand.nonlinear[[3]], fold)),
  flow = colMeans(cal.rmse(grand.flow[[3]], fold)))

colMeans(fall.compare)
```

```
##     lag5      sto     flow
## 2.581101 3.221028 2.562034
```

```
knitr::kable(fall.compare, digits = 3)
```

| | lag5 | sto | flow |
|---|---|---|---|
| bigcreek | 1.511 | 3.517 | 1.505 |
| bigotter | 1.297 | 4.003 | 1.290 |
| fox | 3.783 | 3.288 | 3.805 |

| | lag5 | sto | flow |
|---|---|---|---|
| genesee | 3.147 | 2.582 | 2.989 |
| humber | 2.348 | 3.717 | 2.347 |
| mississagi | 2.722 | 2.762 | 2.728 |
| nipigon | 1.635 | 3.254 | 1.636 |
| portage | 3.481 | 3.726 | 3.471 |
| saginaw | 2.834 | 2.981 | 2.855 |
| still | 2.233 | 3.218 | 2.231 |
| stlouis | 3.084 | 2.530 | 3.098 |
| vermilion | 2.900 | 3.073 | 2.790 |

```
## Annual
annual.compare <- data.frame(
  lag5 = colMeans(cal.rmse(grand.lag5[[5]], fold)),
  sto = colMeans(cal.rmse(grand.sto[[5]], fold)),
  #nonlinear = colMeans(cal.rmse(grand.nonlinear[[5]], fold)),
  flow = colMeans(cal.rmse(grand.flow[[5]], fold)))

colMeans(annual.compare)
```

```
##     lag5      sto     flow
## 3.040102 3.150102 3.005924
```

```
knitr::kable(annual.compare, digits = 3)
```

| | lag5 | sto | flow |
|---|---|---|---|
| bigcreek | 1.960 | 3.110 | 1.965 |
| bigotter | 1.393 | 3.356 | 1.386 |
| fox | 3.813 | 3.203 | 3.830 |
| genesee | 3.340 | 2.541 | 3.105 |
| humber | 2.656 | 3.288 | 2.657 |
| mississagi | 3.524 | 3.454 | 3.531 |
| nipigon | 2.870 | 4.505 | 2.866 |
| portage | 3.079 | 2.922 | 3.081 |
| saginaw | 3.454 | 2.554 | 3.473 |
| still | 3.142 | 2.305 | 3.140 |
| stlouis | 3.955 | 3.031 | 3.973 |
| vermilion | 3.295 | 3.531 | 3.065 |

We then compare the results of different seasonal scales on each individual model.

```
## Lag5::
lag5.compare <- data.frame(
  spring = colMeans(cal.rmse(grand.lag5[[1]], fold)),
  summer = colMeans(cal.rmse(grand.lag5[[2]], fold)),
  fall = colMeans(cal.rmse(grand.lag5[[3]], fold)),
  winter = colMeans(cal.rmse(grand.lag5[[4]], fold)),
  annual = colMeans(cal.rmse(grand.lag5[[5]], fold)))

colMeans(lag5.compare)
knitr::kable(lag5.compare, digits = 3)
```

```r
## Stochastic::
sto.compare <- data.frame(
  spring = colMeans(cal.rmse(grand.sto[[1]], fold)),
  summer = colMeans(cal.rmse(grand.sto[[2]], fold)),
  fall = colMeans(cal.rmse(grand.sto[[3]], fold)),
  winter = colMeans(cal.rmse(grand.sto[[4]], fold)),
  annual = colMeans(cal.rmse(grand.sto[[5]], fold)))

colMeans(sto.compare)
knitr::kable(sto.compare, digits = 3)


## Flow::
nonlinear.compare <- data.frame(
  spring = colMeans(cal.rmse(grand.nonlinear[[1]], fold)),
  summer = colMeans(cal.rmse(grand.nonlinear[[2]], fold)),
  fall = colMeans(cal.rmse(grand.nonlinear[[3]], fold)),
  annual = colMeans(cal.rmse(grand.nonlinear[[5]], fold)))

colMeans(nonlinear.compare)
knitr::kable(flow.compare, digits = 3)

## Flow::
flow.compare <- data.frame(
  spring = colMeans(cal.rmse(grand.flow[[1]], fold)),
  summer = colMeans(cal.rmse(grand.flow[[2]], fold)),
  fall = colMeans(cal.rmse(grand.flow[[3]], fold)),
  winter = colMeans(cal.rmse(grand.flow[[4]], fold)),
  annual = colMeans(cal.rmse(grand.flow[[5]], fold)))

colMeans(flow.compare)
knitr::kable(flow.compare, digits = 3)
```