Global and regional temp model

Eddie Wu

2024-04-30

Introduction

This R markdown file is used to compare different water temperature models on a weekly timescale.

1. Weekly water temperature output from futureStream database
2. lag 5 days water temperature model - converted to weakly scale
3. ...

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

```
form.fixed <- waterT ~ airT.lag1 + airT.lag2 + airT.lag3 + airT.lag4 + airT.lag5

form.locrandom <- waterT ~ airT.lag1 + airT.lag2 + airT.lag3 + airT.lag4 + airT.lag5 + (1|location)


calculate.rmse <- function(fold, training, testing, form, type) {

  # lists to store all the outputs
  model.list <- vector("list", fold)
  out.list <- vector("list", fold)
  rmse.list <- matrix(NA, nrow = fold,
                      ncol = length(unique(training[[1]]$location)))
  colnames(rmse.list) <- unique(training[[1]]$location)

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

    compare <- NA

    # select current dataset, and all unique location levels
    current.training <- training[[i]]
    current.testing <- testing[[i]]
    loc.levels <- unique(current.training$location)

    if (type == "fixed") { # fixed effect model
      model <- lm(form, data = current.training)
```

```r
      preds <- predict(model, newdata = current.testing)}
    else{ # mixed effect model
      model <- lmer(form, data = current.training)
      preds <- predict(model, newdata = current.testing, re.form=(~1|location))}

    p <- as.data.frame(preds)

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

    for (loc in 1:length(loc.levels)) {
      compare.now <- compare[compare$location == loc.levels[loc],]
      rmse <- rmse(compare.now$obs, compare.now$preds)
      rmse.list[i,loc] <- rmse
    }

    model.list[[i]] <- model
    out.list[[i]] <- compare
  }
  return(list(rmse = rmse.list, model = model.list, out = out.list))
}
```

Data importing and cleaning

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

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
```

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

```
##
```

```
##                CAMERON FALLS (AUT)                               CLOQUET
##                               3923                                  1349
##                           DELHI CS                           DELHI CS_PD
##                               4554                                  1825
##                ELYRIA LORAIN CO AP                      GORE BAY CLIMATE
##                               1095                                  1879
##                GREEN BAY BOTANICAL                      INDIANA DUNES NP
##                               1374                                   730
##                   LONG POINT (AUT)                            MONETVILLE
##                               1883                                  1246
##                 ROCHESTER GTR INTL                            SAGINAW #3
##                               2190                                  1095
##                  TILLSONBURG NORTH TORONTO LESTER B. PEARSON INT'L A
##                                919                                  5012
```

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

```
##
##   bigcreek   bigotter        fox    genesee     humber  longpoint mississagi
##       4554        919       1374       2190       5012       1883       1879
##    nipigon    portage  portdover    saginaw      still    stlouis  vermilion
##       3923        730       1825       1095       1246       1349       1095
```

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

Get lagged days

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


## Get the time lag day variables
master.temp <- master.temp %>%
  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)) %>%
  na.omit()

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

```
##
##   bigcreek   bigotter        fox    genesee     humber  longpoint mississagi
##       4413        909       1352       2173       4429       1336       1317
##    nipigon    portage  portdover    saginaw      still    stlouis  vermilion
##       3796        722       1718       1008        970       1199        949
```

futureStream database

We import weekly modelled water temperature data from the 14 tributary locations.

```r
## Import all 14 files
bigcreek <-read.csv("weekly modelled water temperature/bigcreek_model.csv") %>%
  mutate(X = "bigcreek")
bigotter <- read.csv("weekly modelled water temperature/bigotter_model.csv") %>%
  mutate(X = "bigotter")
fox <- read.csv("weekly modelled water temperature/fox_model.csv") %>%
  mutate(X = "fox")
genesee <- read.csv("weekly modelled water temperature/genesee_model.csv") %>%
  mutate(X = "genesee")
humber <- read.csv("weekly modelled water temperature/humber_model.csv") %>%
  mutate(X = "humber")
longpoint <- read.csv("weekly modelled water temperature/lp_model.csv") %>%
  mutate(X = "longpoint")
mississagi <- read.csv("weekly modelled water temperature/mississagi_model.csv") %>%
  mutate(X = "mississagi")
nipigon <- read.csv("weekly modelled water temperature/nipigon_model.csv") %>%
  mutate(X = "nipigon")
portage <- read.csv("weekly modelled water temperature/pb_model.csv") %>%
  mutate(X = "portage")
portdover <- read.csv("weekly modelled water temperature/portdover_model.csv") %>%
  mutate(X = "portdover")
```

```r
saginaw <- read.csv("weekly modelled water temperature/saginaw_model.csv") %>%
  mutate(X = "saginaw")
still <- read.csv("weekly modelled water temperature/still_model.csv") %>%
  mutate(X = "still")
stlouis <- read.csv("weekly modelled water temperature/st_louis_model.csv") %>%
  mutate(X = "stlouis")
vermilion <- read.csv("weekly modelled water temperature/vermilion_model.csv") %>%
  mutate(X = "vermilion")

# Combine into one dataframe
futurestream.temp <- rbind(bigcreek,bigotter,fox,genesee,humber,longpoint,
                           mississagi,nipigon,portage,portdover,saginaw,
                           still,stlouis,vermilion) %>%
  rename(location = X, week = weeks, preds.futureS = temperature.avg) %>%
  arrange(location)
```

SECTION 1: Lag5 day multiple linear regression

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.

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

## Subsetting
df_by_location <- split(master.temp, master.temp$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)
}
```

Linear lag 5 days multiple regression model (both fixed and mixed).

```
## Get results using 10-fold iterations
results.fixed <- calculate.rmse(fold,combined_training_list,
                                combined_testing_list,
                                form.fixed,"fixed")
results.mixed <- calculate.rmse(fold, combined_training_list,
                               combined_testing_list,
                               form.locrandom,"mixed")

# Get RMSE
rmse.lag5.f <- results.fixed[["rmse"]]
rmse.lag5.m <- results.mixed[["rmse"]]

colMeans(rmse.lag5.f)
```

```
##   bigcreek   bigotter        fox   genesee     humber  longpoint mississagi
##   2.543605   2.916283   3.622639  3.084740   2.797378   3.205437   2.908024
##    nipigon    portage  portdover   saginaw      still    stlouis  vermilion
##   4.681272   3.409683   2.985273  3.034091   2.898101   3.286049   3.044700
```

```
colMeans(rmse.lag5.m)
```

```
##   bigcreek   bigotter        fox   genesee     humber  longpoint mississagi
##   2.212807   2.636534   3.503543  3.071165   2.868046   3.146532   2.911232
##    nipigon    portage  portdover   saginaw      still    stlouis  vermilion
##   4.722085   2.799491   2.895952  3.021768   2.841109   3.261632   2.976130
```

SECTION 2: Compare lag5 and futureS on a weekly time scale

1. Convert the lag5 model daily water temperature predictions to weekly water temperature predictions.
2. Merge the weekly observations with predictions from lag5 model and predictions from futureStream model.
3. Calculate RMSE for both models and compare (for each location).

Note: Each is done through 10 iterations.

```
# Dataframe to store the results
rmse.lag5 <- matrix(NA, nrow = 10, ncol = length(unique_locations))
colnames(rmse.lag5) <- unique_locations

rmse.futureS <- matrix(NA, nrow = 10, ncol = length(unique_locations))
colnames(rmse.futureS) <- unique_locations

## Loop through 10 folds
for(i in 1:fold) {
  ## Convert lag5 model prediction to weekly timescale
  df.days <- results.mixed[["out"]][[i]]
  df.days$week <- week(df.days$date)
  df.days$week[df.days$week == 53] <- 52 #need to convert week 53 to 52

  df.week <- merge(
    aggregate(preds~week+location, FUN=mean, data=df.days, na.rm=TRUE),
    aggregate(obs~week+location, FUN=mean, data=df.days, na.rm=TRUE),
    all=TRUE)
```

```r
  # Merge together
  temp.compare <- merge(df.week, futurestream.temp) %>%
    rename(preds.lag5 = preds) %>%
    select(week, location, obs, preds.lag5, preds.futureS) %>%
    arrange(location)

  # get RMSE for each location
  for (loc in 1:length(unique_locations)) {
    compare <- temp.compare[temp.compare$location == unique_locations[loc],]
    rmse.lag5[i,loc] <- rmse(compare$obs, compare$preds.lag5)
    rmse.futureS[i,loc] <- rmse(compare$obs, compare$preds.futureS)
  }
}

colMeans(rmse.lag5)
```

```
##   genesee    stlouis    saginaw        fox    portage  vermilion   bigcreek
##  2.726278   3.063081   2.674251   3.289423   2.161299   2.482022   1.887438
##  bigotter      still mississagi    nipigon      humber  portdover  longpoint
##  2.331578   2.617732   2.749857   4.517677   2.576953   2.685474   2.983312
```

```r
colMeans(rmse.futureS)
```

```
##   genesee    stlouis    saginaw        fox    portage  vermilion   bigcreek
##  1.994712   2.242234   3.909068   3.365590   4.317717   2.366857   3.200598
##  bigotter      still mississagi    nipigon      humber  portdover  longpoint
##  3.799398   1.590286   1.480916   5.582345   3.154742   4.309248   2.061359
```

SECTION 3: Stochastic model

Simple stochastic model with a annual component and a short-term residual component on the daily water and air temperature data. Model formulation is:

$$T_W(t) = T_A(t) + R_W(t)$$

Annual component is calculated by optimizing a sine function to the times series.

$$T_A(t) = a + b * sin[\frac{2\pi}{365}(t + t_0)]$$

, where a, b, and t0 are estimated coefficients.

Short-term residual component is based on multiple regression lagged air temperature residuals (based on Kothandaraman, 1971). (Note that there are other ways of representing the residual components…)

$$R_W(t) = \beta_1 R_A(t) + \beta_2 R_A(t-1) + \beta_3 R_A(t-2)$$

, where β_1, β_2, β_3 are regression coefficients, and $R_A(t)$, $R_A(t-1)$, $R_A(t-2)$ are air temperature residuals at time t, t-1, and t-2.

AirT residuals as short-term component

We first try a stochastic model with no random effects.

```r
out <- vector("list", fold)
rmse <- matrix(NA, nrow=fold,
               ncol=length(unique(combined_training_list[[1]]$location)))
colnames(rmse) <- unique(combined_training_list[[1]]$location)

# Use a list to store all the outputs
results.sto.f <- list(rmse = rmse, out = out)


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

  compare <- NA
  # select current dataset, and all unique location levels
  current.training <- combined_training_list[[i]]
  current.testing <- combined_testing_list[[i]]
  loc.levels <- unique(current.training$location)

  ## 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)
  current.training$airT - fitted(annual.comp)

  # get the air temperature residuals
  res <- as.data.frame(matrix(NA, ncol = 4,
                             nrow = length(current.training$airT)))
                             # dataframe to store the residuals
  colnames(res) <- c("res.t", "res.t1", "res.t2", "res.w")
  res[,"res.t"] <- as.vector(residuals(annual.comp))
  res[,"res.t1"][-1] <- res[,"res.t"][-nrow(res)]
  res[,"res.t2"][-1] <- res[,"res.t1"][-nrow(res)]
  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 <- lm(res.w ~ res.t + res.t1 + res.t2,
                     data = res, na.action = na.omit)

  ## TESTING
  preds.annual <- predict(annual.comp, newdata=current.testing)

  res <- as.data.frame(matrix(NA, ncol = 3,
                           nrow = length(current.testing$airT))) #residuals
  colnames(res) <- c("res.t", "res.t1", "res.t2")
  res[,"res.t"] <- current.testing$airT - preds.annual
  res[,"res.t1"][-1] <- res[,"res.t"][-nrow(res)]
  res[,"res.t2"][-1] <- res[,"res.t1"][-nrow(res)]
  preds.residuals <- predict(residual.comp, newdata=res)

  # add up both components
  p <- as.data.frame(preds.annual + preds.residuals)
```

```r
  ## Calculate RMSE
  compare <- cbind(current.testing,
                   preds = p$`preds.annual + preds.residuals`) %>%
  select(location, date, obs = waterT, preds)

  for (loc in 1:length(loc.levels)) {
    compare.now <- compare[compare$location == loc.levels[loc],] %>% na.omit()
    results.sto.f[["rmse"]][i,loc] <-
      rmse(compare.now$obs, compare.now$preds)
  }

  results.sto.f[["out"]][[i]] <- compare
}


## Get the results
rmse.sto.f <- results.sto.f[["rmse"]]
colMeans(rmse.sto.f)
```

```
##   bigcreek   bigotter        fox    genesee     humber  longpoint mississagi
##   3.476815   4.278987   4.282903   3.769455   4.977361   3.248575   3.863901
##    nipigon    portage  portdover    saginaw      still    stlouis  vermilion
##   5.498926   6.370953   3.861145   3.643659   2.966415   3.189976   3.475506
```

Then, we try a stochastic model with mixed effect (location as the random factor).

```r
out <- vector("list", fold)
rmse <- matrix(NA, nrow=fold,
               ncol=length(unique(combined_training_list[[1]]$location)))
colnames(rmse) <- unique(combined_training_list[[1]]$location)

# Use a list to store all the outputs
results.sto.m <- list(rmse = rmse, out = out)


for (i in 1:fold){

  compare <- NA
  # select current dataset, and all unique location levels
  current.training <- combined_training_list[[i]]
  current.testing <- combined_testing_list[[i]]
  loc.levels <- unique(current.training$location)

  ## 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 = 5,
                              nrow = length(current.training$airT)))
                              # dataframe to store the residuals
  colnames(res) <- c("res.t", "res.t1", "res.t2", "res.w", "location")
  res[,"res.t"] <- as.vector(residuals(annual.comp))
```

```r
  res[,"res.t1"][-1] <- res[,"res.t"][-nrow(res)]
  res[,"res.t2"][-1] <- res[,"res.t1"][-nrow(res)]
  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))
  res[,"location"] <- current.training$location

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

  ## TESTING
  preds.annual <- predict(annual.comp, newdata=current.testing)

  res <- as.data.frame(matrix(NA, ncol = 4,
                              nrow = length(current.testing$airT))) #residuals
  colnames(res) <- c("res.t", "res.t1", "res.t2", "location")
  res[,"res.t"] <- current.testing$airT - preds.annual
  res[,"res.t1"][-1] <- res[,"res.t"][-nrow(res)]
  res[,"res.t2"][-1] <- res[,"res.t1"][-nrow(res)]
  res[,"location"] <- current.testing$location
  preds.residuals <- predict(residual.comp, newdata=res,
                             re.form=(~1|location))

  # add up both components
  p <- as.data.frame(preds.annual + preds.residuals)

  ## Calculate RMSE
  compare <- cbind(current.testing,
                  preds = p$`preds.annual + preds.residuals`) %>%
  select(location, date, obs = waterT, preds)

  for (loc in 1:length(loc.levels)) {
    compare.now <- compare[compare$location == loc.levels[loc],] %>% na.omit()
    results.sto.m[["rmse"]][i,loc] <- rmse(compare.now$obs, compare.now$preds)
  }

  results.sto.m[["out"]][[i]] <- compare
}


## Get the results
rmse.sto.m <- results.sto.m[["rmse"]]
colMeans(rmse.sto.m)
```

```
##   bigcreek   bigotter        fox    genesee     humber  longpoint mississagi
##   4.209479   4.649439   3.503283   3.232909   4.406338   3.327063   4.498905
##    nipigon    portage  portdover    saginaw      still    stlouis  vermilion
##   5.798437   3.822877   4.234537   3.253695   3.238497   3.888686   3.246384
```

Actual airT as short-term component

Some literatures (Rabi et al., 2015) use actual air temperatures (instead of residuals) to represent the residual component. In this case, the residual component is:

$$R_W(t) = \beta_1 T_A(t) + \beta_2 T_A(t-1) + \beta_3 T_A(t-2)$$

We first start with a fixed model.

```
out <- vector("list", fold)
rmse <- matrix(NA, nrow=fold,
               ncol=length(unique(combined_training_list[[1]]$location)))
colnames(rmse) <- unique(combined_training_list[[1]]$location)

# Use a list to store all the outputs
results.sto.fa <- list(rmse = rmse, out = out)


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

  compare <- NA
  # select current dataset, and all unique location levels
  current.training <- combined_training_list[[i]]
  current.testing <- combined_testing_list[[i]]
  loc.levels <- unique(current.training$location)

  ## 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 water temperature residual component
  residual.comp <- lm(waterT ~ airT + airT.lag1 + airT.lag2,
                      data = current.training)

  ## TESTING
  preds.annual <- predict(annual.comp, newdata=current.testing)
  preds.residuals <- predict(residual.comp, newdata=current.testing)
  p <- as.data.frame(preds.annual + preds.residuals)

  ## Calculate RMSE
  compare <- cbind(current.testing,
                  preds = p$`preds.annual + preds.residuals`) %>%
  select(location, date, obs = waterT, preds)

  for (loc in 1:length(loc.levels)) {
    compare.now <- compare[compare$location == loc.levels[loc],] %>% na.omit()
    results.sto.fa[["rmse"]][i,loc] <-
      rmse(compare.now$obs, compare.now$preds)
  }

  results.sto.fa[["out"]][[i]] <- compare
}


## Get the results
rmse.sto.fa <- results.sto.fa[["rmse"]]
```

```r
colMeans(rmse.sto.fa)
```

```
##   bigcreek   bigotter        fox    genesee     humber  longpoint mississagi 
##   14.55430   16.08952   11.21202   11.41809   12.78300   11.45808   13.10092 
##    nipigon    portage  portdover    saginaw      still    stlouis  vermilion 
##   14.71802   10.96822   13.78149   11.45514   14.19240   12.94891   12.23475
```

Now a mixed model with location as a random effect

```r
out <- vector("list", fold)
rmse <- matrix(NA, nrow=fold,
               ncol=length(unique(combined_training_list[[1]]$location)))
colnames(rmse) <- unique(combined_training_list[[1]]$location)

# Use a list to store all the outputs
results.sto.ma <- list(rmse = rmse, out = out)


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

  compare <- NA
  # select current dataset, and all unique location levels
  current.training <- combined_training_list[[i]]
  current.testing <- combined_testing_list[[i]]
  loc.levels <- unique(current.training$location)

  ## 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 water temperature residual component
  residual.comp <- lmer(waterT ~ airT + airT.lag1 + airT.lag2 + (1|location),
                        data = current.training)

  ## TESTING
  preds.annual <- predict(annual.comp, newdata=current.testing)
  preds.residuals <- predict(residual.comp, newdata=current.testing,
                             re.form=(~1|location))
  p <- as.data.frame(preds.annual + preds.residuals)

  ## Calculate RMSE
  compare <- cbind(current.testing,
                   preds = p$`preds.annual + preds.residuals`) %>%
  select(location, date, obs = waterT, preds)

  for (loc in 1:length(loc.levels)) {
    compare.now <- compare[compare$location == loc.levels[loc],] %>% na.omit()
    results.sto.ma[["rmse"]][i,loc] <-
      rmse(compare.now$obs, compare.now$preds)
  }

  results.sto.ma[["out"]][[i]] <- compare
}
```

```r
## Get the results
rmse.sto.ma <- results.sto.ma[["rmse"]]
colMeans(rmse.sto.ma)
```

```
##   bigcreek   bigotter        fox    genesee     humber  longpoint mississagi
##   13.56266   15.26097   12.12649   11.25672   13.16136   11.13404   13.08909
##    nipigon    portage  portdover    saginaw      still    stlouis  vermilion
##   14.67902   12.25324   13.26060   11.63462   14.62839   13.22161   11.80771
```

Both models with actual air temperature seem to perform very poorly... So we should not use this form...

SECTION 4: Overall Comparison and graphs

```r
rmse.lag5.f
rmse.lag5.m
rmse.sto.f
rmse.sto.m

# regional comparison
data.frame(lag5_fixed = colMeans(rmse.lag5.f),
           lag5_mixed = colMeans(rmse.lag5.m),
           sto_fixed = colMeans(rmse.sto.f),
           sto_mixed = colMeans(rmse.sto.m))

# global and regional
a <- data.frame(regional_lag5 = colMeans(rmse.lag5),
                global = colMeans(rmse.futureS))

View(a)

max(a[,"regional_lag5"])
max(a[,"global"])
```

```r
a <- coef(annual.comp)[1]
b <- coef(annual.comp)[2]
t0 <- coef(annual.comp)[3]

form <- function(x) a+b*sin(2*pi/365*(x+t0))

curve(expr = form, from=1, to=1000)


pp.c <- compare %>% filter(location == "mississagi")

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