Seasonal river temperature model

Eddie Wu

2024-05-17

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.*

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

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

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

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

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

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

  # 10 iterations
  for (i in 1:fold){
```

```
   compare <- out.list[[i]]

   # calculate nsc
   for (loc in unique(compare$location)) {
     compare.now <- subset(compare, location == loc) %>% na.omit()
     nsc <- 1 - sum((compare.now$obs - compare.now$preds)^2) / sum((compare.now$obs - mean(compare.now$
     r[i,loc] <- nsc
    }
  }
  return(r)
}
get.traintest <- function(master.df, fold) {
  ## Subsetting
  df_by_location <- split(master.df, master.df$location)
  df_by_location <- df_by_location[sapply(df_by_location, function(x)
    !is.null(x) && nrow(x) > 0)]

  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]]
      loc_training <- current[current$year == sample(current$year, 1),]
      loc_testing <- current[current$year == sample(current$year, 1),]
      # Add to the combined list
      combined_training_list[[f]] <- rbind(
        combined_training_list[[f]], loc_training)
      combined_testing_list[[f]] <- rbind(
        combined_testing_list[[f]], loc_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)
  }
  return(list(combined_training_list, combined_testing_list))
}
```

Data importing and cleaning

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

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

```r
flow$date <- as.Date(flow$date, format = "%m/%d/%Y")


table(airtemp$station_name)
```

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

```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
```

```r
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
```

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

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

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

Check for imputed values

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

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

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

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

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

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

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

Data cleaning and combining

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

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

Get lagged days

```
## 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()

master.temp <- master.temp %>%
```

```
  group_by(location) %>%
  mutate(flow.lag1 = lag(flow, 1),
         flow.lag2 = lag(flow, 2),
         flow.lag3 = lag(flow, 3),
         flow.lag4 = lag(flow, 4),
         flow.lag5 = lag(flow, 5)) %>%
  na.omit()


## Get relative flow and cumulative flow
master.temp <- master.temp %>%
  mutate(rqc = (flow - flow.lag1)/flow)

master.temp <- master.temp %>%
  mutate(cumflow = flow.lag1 + flow.lag2 + flow.lag3 + flow.lag4 + flow.lag5)


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

```
##
##   bigcreek   bigotter        fox    genesee     humber mississagi    nipigon
##       4408        904       1347       2168       4424       1312        834
##    portage    saginaw      still    stlouis  vermilion
##        717       1003        965       1194        944
```

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. Growing season (Jun-Aug)
2. Winter

```
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)
```

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.temp, 10)
```

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


## Get training and testing for each season
sum <- get.traintest(master.sum, 10)
win <- get.traintest(master.win, 10)
spr <- get.traintest(master.spring, 10)
fall <- get.traintest(master.fall, 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 four times, once on each season.

Linear lag5

```
## Forms
form1 <- waterT ~ airT.lag1 + airT.lag2 + airT.lag3 + airT.lag4 + airT.lag5 + (1|location)

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.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){

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

    # model training and predicting
    model <- lmer(form1, 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)

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

Stochastic

```
## 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]]
    current.testing <- test[[i]]

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

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

lag5 with flow

```
## Forms
form2 <- waterT ~ airT.lag1 + airT.lag2 + airT.lag3 + airT.lag4 + airT.lag5 + cumflow + (1|location)

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.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]]

  ## Iterations for each fold 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 <- lmer(form2, 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)

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

Comparison at the end

We first compare each model within one specific season.

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

colMeans(spring.compare)
```

```
##     lag5      sto     flow
## 2.809226 3.168302 2.785718
```

```r
## Summer::
summer.compare <- data.frame(
  lag5 = colMeans(cal.rmse(grand.lag5[[2]], fold)),
```

```
  sto = colMeans(cal.rmse(grand.sto[[2]], fold)),
  flow = colMeans(cal.rmse(grand.flow[[2]], fold)))

colMeans(summer.compare)
```

```
##     lag5      sto     flow
## 1.750424 2.046760 1.770621
```

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

colMeans(fall.compare)
```

```
##     lag5      sto     flow
## 2.457526 3.101649 2.432441
```

```
## Winter::
winter.compare <- data.frame(
  lag5 = colMeans(cal.rmse(grand.lag5[[4]], fold)),
  sto = colMeans(cal.rmse(grand.sto[[4]], fold)),
  flow = colMeans(cal.rmse(grand.flow[[4]], fold)))

colMeans(winter.compare)
```

```
##     lag5      sto     flow
## 1.152675 6.554349 1.168630
```

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

colMeans(annual.compare)
```

```
##     lag5      sto     flow
## 3.156671 4.212904 3.140803
```

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)
```

```
##   spring   summer     fall   winter   annual
## 2.809226 1.750424 2.457526 1.152675 3.156671
```

```
## 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)
```

```
##   spring   summer     fall   winter   annual
## 3.168302 2.046760 3.101649 6.554349 4.212904
```

```
## 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)
```

```
##   spring   summer     fall   winter   annual
## 2.785718 1.770621 2.432441 1.168630 3.140803
```