Master temperature model comparison

Eddie Wu

2024-07-27

Introduction

This R markdown file is used to

1. Import and clean global water temperature model
2. Fit three candidate models on training and testing data for different time scales.
3. Compare the weekly performance of a non-linear model and the global futureStreams river temperature model on a different seasonal scales (spring, summer, fall, annual).

All model metrics are calculated on a weekly scale!!!

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

```
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
```

```r
        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)
}
```

SECTION 1: 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.

```r
# make sure that no initial NA values in awf water temperature
which(is.na(awf$temp))
```

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

```r
# 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

```r
## 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 cumulative air temp for past five days
awf <- awf %>%
```

```
  rowwise() %>%
  mutate(cair = (mean_temp+dmean_1+dmean_2+dmean_3+dmean_4+dmean_5)/6)

## 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, cair)

master.temp <- master.temp[complete.cases(master.temp[,8:18]),]
```

SECTION 2: 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 seasonal data to be greater than 60 days, annual data (winter removed) more than 180 days.

```
## $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"

## $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"

## $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"

## $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"
```

```
## $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)
```

SECTION 3: REGIONAL model comparison

We now run each of the three models ten 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

coef.lag5 <- data.frame(matrix(NA,50,7))
colnames(coef.lag5) <- c("a","b","c","d","e","f","g")
seasons <- rep(c(1,2,3,4,5), each = 10)
coef.lag5$season <- seasons

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){

    # 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))

    compare$week <- week(compare$date)
    compare$week[compare$week == 53] <- 52 #need to convert week 53 to 52

    # weekly comparison dataframe
```

```r
    compare.w <- merge(
      aggregate(preds.ar~week+location,FUN=mean,data=compare,
                na.action=na.omit),
      aggregate(water~week+location,FUN=mean,data=compare,
                na.action=na.omit),
      all=TRUE) %>%
      na.omit() %>%
      arrange(location, week)

    # store in list
    grand.lag5[[season]][[i]] <- compare.w
    coef.lag5[10*(season-1)+i,1:7] = coef(model.ar)[1,]
    #intercept only for bigcreek
  }
}
```

Seasonal residual

```r
coef.sto <- data.frame(matrix(NA,50,7))
colnames(coef.sto) <- c("a","b","c","beta1","beta2","beta3","d")
seasons <- rep(c(1,2,3,4,5), each = 10)
coef.sto$season <- seasons


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)
```

```r
# 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)

compare$week <- week(compare$date)
```

```r
    compare$week[compare$week == 53] <- 52 #need to convert week 53 to 52

    # weekly comparison dataframe
    compare.w <- merge(
      aggregate(preds.ar~week+location,FUN=mean,data=compare,
                na.action=na.omit),
      aggregate(water~week+location,FUN=mean,data=compare,
                na.action=na.omit),
      all=TRUE) %>%
      na.omit() %>%
      arrange(location, week)

    # store in a list
    grand.sto[[season]][[i]] <- compare.w
    coef.sto[10*(season-1)+i,1:3] = coef(annual.comp)
    coef.sto[10*(season-1)+i,4:6] = coef(residual.comp.ar)[1,2:4]
    coef.sto[10*(season-1)+i,7] = coef(residual.comp.ar)[1,1]
  }
}
```

Non-linear

```r
coef.nonlinear <- data.frame(matrix(NA,50,3))
colnames(coef.nonlinear) <- c("alpha","gamma","beta")
seasons <- rep(c(1,2,3,4,5), each = 10)
coef.nonlinear$season <- seasons


## starting parameters
coef.spring <- c(alpha=30, gamma=0.05,beta=10)
coef.summer <- c(alpha=25, gamma=0.05,beta=10)
coef.fall <- c(alpha=30, gamma=0.05,beta=10)
coef.annual <- c(alpha=20, gamma=0.05,beta=9)

coef.list <- list(coef.spring, coef.summer, coef.fall, NA, 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]]
    compare <- current.testing

    # model training and predicting
    model <- nlme(water ~ alpha / (1 + exp(gamma * (beta - cair))),
                  fixed = list(alpha~1,gamma~1,beta~1),
                  random = gamma ~ 1|location,
                  start = coef.list[[season]],
                  data = current.training, na.action = na.omit,
                  control = list(msMaxIter = 200))

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

    compare$week <- week(compare$date)
    compare$week[compare$week == 53] <- 52 #need to convert week 53 to 52

    # weekly comparison dataframe
    compare.w <- merge(
      aggregate(preds.ar~week+location,FUN=mean,data=compare,
                na.action=na.omit),
      aggregate(water~week+location,FUN=mean,data=compare,
                na.action=na.omit),
      all=TRUE) %>%
      na.omit() %>%
      arrange(location, week)

    grand.nonlinear[[season]][[i]] <- compare.w
    coef.nonlinear[10*(season-1)+i,1:3] <- coef(model)
  }
}
```

Comparison at the end

Note that all these metric calculations are done on a weekly scale, in order to be comparable to the global water temperature model!

```
# new function - different from other files...
cal.rmse <- function(out.list, fold, switch) {

  # 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()

      if (switch=="r"){
        r[i,loc] <-
          round(sqrt(mean((compare.now$water-compare.now$preds.ar)^2)),2)
      } else if (switch=="g"){
        r[i,loc] <-
          round(sqrt(mean((compare.now$water-compare.now$preds.futureS)^2)),2)
      }
    }
  }
  return(r)
}

cal.nsc <- function(out.list, fold, switch) {

  # 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()

     if (switch=="r"){
       r[i,loc] <-
         1 - sum((compare.now$water - compare.now$preds.ar)^2) / sum((compare.now$water - mean(compare.n
     } else if (switch=="g"){
       r[i,loc] <-
         1 - sum((compare.now$water - compare.now$preds.futureS)^2) / sum((compare.now$water - mean(comp
     }
    }
  }
  return(r)
}
```

We first compare the RMSE of each model within one specific season.

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

colMeans(spring.compare)
```

```
##      lag5       sto nonlinear
##  2.981083  3.260250  2.041000
```

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

	lag5	sto	nonlinear
bigcreek	2.338	2.993	1.946
bigotter	1.562	2.915	1.564
fox	3.210	3.008	1.931
genesee	3.823	2.512	2.205
humber	2.937	2.908	1.830
mississagi	2.466	3.783	1.338
nipigon	2.116	4.877	0.534
portage	2.684	3.971	2.631
saginaw	3.346	2.465	1.916
still	3.775	1.735	3.333
stlouis	3.624	4.173	2.686
vermilion	3.892	3.783	2.578

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

colMeans(summer.compare)
```

```
##      lag5       sto nonlinear
##  1.777917  1.925750  1.798125
```

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

	lag5	sto	nonlinear
bigcreek	1.109	1.594	0.948
bigotter	1.162	2.009	1.307
fox	1.515	1.772	1.430
genesee	1.830	1.518	1.706
humber	1.590	1.746	1.611
mississagi	1.932	1.801	1.414
nipigon	3.058	2.927	3.564
portage	2.024	1.822	2.373
saginaw	2.725	3.050	3.022
still	1.337	1.289	1.570
stlouis	1.867	2.071	1.388
vermilion	1.186	1.510	1.244

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

colMeans(fall.compare)
```

```
##      lag5       sto nonlinear
##  2.597000  3.159417  2.113833
```

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

	lag5	sto	nonlinear
bigcreek	1.528	3.352	2.503
bigotter	0.798	3.910	1.279
fox	4.310	3.100	3.033
genesee	3.236	2.737	1.761
humber	2.748	3.359	2.296
mississagi	2.712	3.023	1.948
nipigon	1.444	3.291	1.248
portage	3.181	3.235	3.389
saginaw	3.142	3.109	1.412
still	2.033	3.194	2.318
stlouis	2.994	2.719	1.752
vermilion	3.038	2.884	2.427

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

colMeans(annual.compare)
```

```
##      lag5       sto nonlinear
##  3.012417  3.049000  2.229250
```

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

	lag5	sto	nonlinear
bigcreek	1.588	2.795	2.145
bigotter	1.331	3.483	1.968
fox	4.048	3.034	2.694
genesee	3.582	2.736	1.748
humber	2.503	2.634	1.685
mississagi	3.446	3.635	2.358
nipigon	3.060	4.966	3.412
portage	2.786	2.328	2.518
saginaw	3.445	2.612	1.737
still	3.338	2.319	2.526
stlouis	3.855	2.872	1.812
vermilion	3.167	3.174	2.148

Now, let's compare the NSC of each model within one specific season.

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

```r
colMeans(spring.compare)
```

```
##        lag5         sto   nonlinear
##  0.56739975 -0.05685539  0.81303307
```

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

	lag5	sto	nonlinear
bigcreek	0.740	0.485	0.797
bigotter	0.863	0.521	0.853
fox	0.700	0.700	0.881
genesee	0.641	0.770	0.880
humber	0.715	0.707	0.888
mississagi	0.640	0.137	0.893
nipigon	-0.425	-6.619	0.902
portage	0.576	0.024	0.592
saginaw	0.752	0.863	0.919
still	0.496	0.868	0.601
stlouis	0.574	0.422	0.762
vermilion	0.537	0.439	0.789

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

colMeans(summer.compare)
```

```
##        lag5         sto   nonlinear
##  0.34745629 -0.01529637  0.27235863
```

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

	lag5	sto	nonlinear
bigcreek	0.422	-0.856	0.559
bigotter	0.350	-1.212	0.171
fox	0.425	0.168	0.449
genesee	0.402	0.573	0.482
humber	-0.002	-0.022	-0.177
mississagi	0.502	0.502	0.722
nipigon	0.361	0.379	0.119
portage	-0.293	-0.490	-0.776
saginaw	0.553	0.318	0.452
still	0.365	0.017	0.054
stlouis	0.563	0.295	0.740
vermilion	0.521	0.144	0.473

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

```r
  nonlinear = colMeans(cal.nsc(grand.nonlinear[[3]], fold, "r")))

colMeans(fall.compare)
```

```
##      lag5       sto nonlinear
## 0.7852662 0.4773187 0.8247463
```

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

	lag5	sto	nonlinear
bigcreek	0.866	0.176	0.641
bigotter	0.946	-0.395	0.876
fox	0.583	0.704	0.782
genesee	0.768	0.791	0.929
humber	0.667	0.022	0.731
mississagi	0.778	0.664	0.883
nipigon	0.881	0.343	0.908
portage	0.648	0.413	0.573
saginaw	0.809	0.779	0.961
still	0.868	0.640	0.820
stlouis	0.826	0.827	0.937
vermilion	0.783	0.763	0.856

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

colMeans(annual.compare)
```

```
##      lag5       sto nonlinear
## 0.8151165 0.7492185 0.8746432
```

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

	lag5	sto	nonlinear
bigcreek	0.897	0.711	0.823
bigotter	0.922	0.434	0.830
fox	0.766	0.855	0.893
genesee	0.795	0.867	0.951
humber	0.858	0.817	0.926
mississagi	0.809	0.779	0.911
nipigon	0.728	0.281	0.661
portage	0.805	0.833	0.825
saginaw	0.844	0.910	0.959
still	0.774	0.877	0.866
stlouis	0.786	0.876	0.950
vermilion	0.796	0.751	0.901

Get regional coefficients

```r
# lag 5
for(s in 1:5) {
  c = coef.lag5[coef.lag5$season == s,]
  print(round(colMeans(c),2))
}
```

```
##      a      b      c      d      e      f      g season
##   5.41   0.12   0.11   0.09   0.07   0.04   0.04   1.00
##      a      b      c      d      e      f      g season
##   9.39   0.12   0.11   0.07   0.05   0.04   0.03   2.00
##      a      b      c      d      e      f      g season
##   4.55   0.12   0.12   0.12   0.09   0.07   0.07   3.00
##      a      b      c      d      e      f      g season
##   2.52   0.03   0.03   0.02   0.02   0.01   0.02   4.00
##      a      b      c      d      e      f      g season
##   5.16   0.15   0.13   0.12   0.10   0.07   0.06   5.00
```

```r
# seasonal residual
for(s in 1:5) {
  c = coef.sto[coef.lag5$season == s,]
  print(round(colMeans(c),2))
}
```

```
##      a      b      c  beta1  beta2  beta3      d season
##   4.68  13.92 -24.91   0.09   0.07   0.04   0.01   1.00
##      a      b      c  beta1  beta2  beta3      d season
##   7.96  13.36  68.89   0.10   0.09   0.05  -3.05   2.00
##       a       b       c   beta1   beta2   beta3       d  season
##    9.28    5.45 -343.34    0.07    0.06    0.04   -1.60    3.00
##       a       b       c   beta1   beta2   beta3       d  season
##    8.26   14.61 -116.65    0.02    0.02    0.01    0.00    4.00
##      a      b      c  beta1  beta2  beta3      d season
##   7.97  13.28  33.49   0.09   0.07   0.04  -1.25   5.00
```

```r
# non-linear
for(s in 1:5) {
  c = coef.nonlinear[coef.lag5$season == s,]
  print(round(colMeans(c),2))
}
```

```
##  alpha  gamma   beta season
##  37.01   0.12  17.35   1.00
##  alpha  gamma   beta season
##  25.86   0.10  11.36   2.00
##  alpha  gamma   beta season
##  26.80   0.12  11.41   3.00
##  alpha  gamma   beta season
##     NA     NA     NA      4
##  alpha  gamma   beta season
##  26.56   0.12  11.53   5.00
```

SECTION 4: Compare REGIONAL and GLOBAL

Global futureStream database

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

```
## Import all 14 files
bigcreek <-read.csv("weekly modeled water temperature_new/bigcreek_model.csv") %>%
  mutate(X = "bigcreek")
bigotter <- read.csv("weekly modeled water temperature_new/bigotter_model.csv") %>%
  mutate(X = "bigotter")
fox <- read.csv("weekly modeled water temperature_new/fox_model.csv") %>%
  mutate(X = "fox")
genesee <- read.csv("weekly modeled water temperature_new/genesee_model.csv") %>%
  mutate(X = "genesee")
humber <- read.csv("weekly modeled water temperature_new/humber_model.csv") %>%
  mutate(X = "humber")
longpoint <- read.csv("weekly modeled water temperature_new/lp_model.csv") %>%
  mutate(X = "longpoint")
mississagi <- read.csv("weekly modeled water temperature_new/mississagi_model.csv") %>%
  mutate(X = "mississagi")
nipigon <- read.csv("weekly modeled water temperature_new/nipigon_model.csv") %>%
  mutate(X = "nipigon")
portage <- read.csv("weekly modeled water temperature_new/pb_model.csv") %>%
  mutate(X = "portage")
portdover <- read.csv("weekly modeled water temperature_new/portdover_model.csv") %>%
  mutate(X = "portdover")
saginaw <- read.csv("weekly modeled water temperature_new/saginaw_model.csv") %>%
  mutate(X = "saginaw")
still <- read.csv("weekly modeled water temperature_new/still_model.csv") %>%
  mutate(X = "still")
stlouis <- read.csv("weekly modeled water temperature_new/st_louis_model.csv") %>%
  mutate(X = "stlouis")
vermilion <- read.csv("weekly modeled water temperature_new/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)
```

Combine REGIONAL and GLOBAL into one file

```
# new dataframe to store preds for regional and global
grand.week = grand.nonlinear

for (season in c(1,2,3,5)) {

  for(i in 1:fold) {
    ## Convert lag5 model prediction to weekly timescale
    df.week <- grand.nonlinear[[season]][[i]]
    comp.week <- merge(df.week, futurestream.temp,
                       by=c("location","week")) %>% arrange(location, week)
```

```r
    # store it back into the grand.nonlinear list
    grand.week[[season]][[i]] <- comp.week
  }
}
```

Calculate RMSE for each seasonal scale

```r
## Annual
annual.compare <- data.frame(
  regional = colMeans(cal.rmse(grand.week[[5]], fold, "r"), na.rm=T),
  global = colMeans(cal.rmse(grand.week[[5]], fold, "g"), na.rm=T))

colMeans(annual.compare)
```

```
## regional   global
## 2.229250 3.316083
```

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

	regional	global
bigcreek	2.145	2.739
bigotter	1.968	3.803
fox	2.694	3.722
genesee	1.748	1.891
humber	1.685	3.628
mississagi	2.358	2.086
nipigon	3.412	4.600
portage	2.518	3.784
saginaw	1.737	3.806
still	2.526	2.475
stlouis	1.812	3.892
vermilion	2.148	3.367

```r
## Spring
spring.compare <- data.frame(
  regional = colMeans(cal.rmse(grand.week[[1]], fold, "r"), na.rm=T),
  global = colMeans(cal.rmse(grand.week[[1]], fold, "g"), na.rm=T))

colMeans(spring.compare)
```

```
## regional   global
##   2.0410   3.0825
```

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

	regional	global
bigcreek	1.946	2.264
bigotter	1.564	2.208
fox	1.931	2.438
genesee	2.205	2.310
humber	1.830	2.921
mississagi	1.338	1.152

	regional	global
nipigon	0.534	4.046
portage	2.631	5.360
saginaw	1.916	3.110
still	3.333	4.254
stlouis	2.686	2.383
vermilion	2.578	4.544

```r
## Summer
summer.compare <- data.frame(
  regional = colMeans(cal.rmse(grand.week[[2]], fold, "r"), na.rm=T),
  global = colMeans(cal.rmse(grand.week[[2]], fold, "g"), na.rm=T))

colMeans(summer.compare)
```

```
## regional   global
## 1.798125 3.546417
```

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

	regional	global
bigcreek	0.948	3.453
bigotter	1.307	4.552
fox	1.430	3.966
genesee	1.706	1.360
humber	1.611	4.258
mississagi	1.414	1.882
nipigon	3.564	5.618
portage	2.373	2.506
saginaw	3.022	5.990
still	1.570	1.978
stlouis	1.388	4.919
vermilion	1.244	2.075

```r
## Fall
fall.compare <- data.frame(
  regional = colMeans(cal.rmse(grand.week[[3]], fold, "r"), na.rm=T),
  global = colMeans(cal.rmse(grand.week[[3]], fold, "g"), na.rm=T))

colMeans(fall.compare)
```

```
## regional   global
## 2.113833 2.748500
```

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

	regional	global
bigcreek	2.503	2.301
bigotter	1.279	2.304
fox	3.033	3.790
genesee	1.761	1.546
humber	2.296	2.410

	regional	global
mississagi	1.948	2.047
nipigon	1.248	3.520
portage	3.389	4.660
saginaw	1.412	3.554
still	2.318	1.490
stlouis	1.752	2.986
vermilion	2.427	2.374

Calculate NSC for each seasonal scale

```
## Annual
annual.compare <- data.frame(
  regional = colMeans(cal.nsc(grand.week[[5]], fold, "r"), na.rm=T),
  global = colMeans(cal.nsc(grand.week[[5]], fold, "g"), na.rm=T))

colMeans(annual.compare)
```

```
##  regional    global
## 0.8746432 0.7175326
```

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

	regional	global
bigcreek	0.823	0.723
bigotter	0.830	0.381
fox	0.893	0.778
genesee	0.951	0.935
humber	0.926	0.666
mississagi	0.911	0.930
nipigon	0.661	0.384
portage	0.825	0.616
saginaw	0.959	0.809
still	0.866	0.874
stlouis	0.950	0.774
vermilion	0.901	0.741

```
## Spring
spring.compare <- data.frame(
  regional = colMeans(cal.nsc(grand.week[[1]], fold, "r"), na.rm=T),
  global = colMeans(cal.nsc(grand.week[[1]], fold, "g"), na.rm=T))

colMeans(spring.compare)
```

```
##  regional    global
## 0.8130331 0.1576901
```

```
colMeans(spring.compare[-7,]) #remove nipigon river
```

```
##  regional    global
## 0.8049803 0.5469866
```

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

	regional	global
bigcreek	0.797	0.727
bigotter	0.853	0.728
fox	0.881	0.745
genesee	0.880	0.840
humber	0.888	0.699
mississagi	0.893	0.920
nipigon	0.902	-4.125
portage	0.592	-0.738
saginaw	0.919	0.787
still	0.601	0.359
stlouis	0.762	0.733
vermilion	0.789	0.218

```r
## Summer
summer.compare <- data.frame(
  regional = colMeans(cal.nsc(grand.week[[2]], fold, "r"), na.rm=T),
  global = colMeans(cal.nsc(grand.week[[2]], fold, "g"), na.rm=T))

colMeans(summer.compare)
```

```
##   regional     global
##  0.2723586 -2.7430405
```

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

	regional	global
bigcreek	0.559	-5.930
bigotter	0.171	-12.563
fox	0.449	-3.091
genesee	0.482	0.626
humber	-0.177	-5.722
mississagi	0.722	0.500
nipigon	0.119	-1.300
portage	-0.776	-1.014
saginaw	0.452	-1.151
still	0.054	-0.823
stlouis	0.740	-2.024
vermilion	0.473	-0.424

```r
## Fall
fall.compare <- data.frame(
  regional = colMeans(cal.nsc(grand.week[[3]], fold, "r"), na.rm=T),
  global = colMeans(cal.nsc(grand.week[[3]], fold, "g"), na.rm=T))

colMeans(fall.compare)
```

```
##  regional    global
## 0.8247463 0.6860235
```

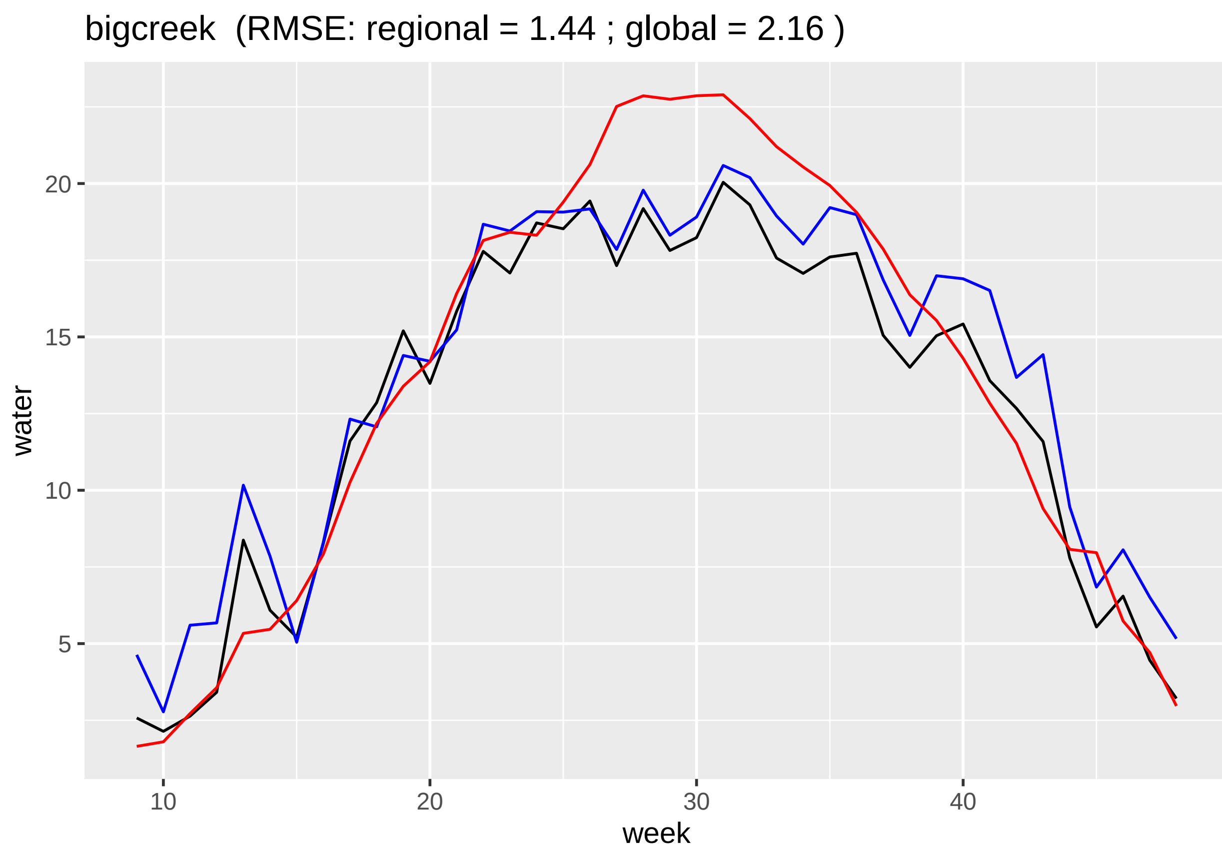
```r
knitr::kable(fall.compare, digits = 3)
```

	regional	global
bigcreek	0.641	0.650
bigotter	0.876	0.604
fox	0.782	0.667
genesee	0.929	0.946
humber	0.731	0.621
mississagi	0.883	0.861
nipigon	0.908	0.299
portage	0.573	0.200
saginaw	0.961	0.758
still	0.820	0.928
stlouis	0.937	0.827
vermilion	0.856	0.869

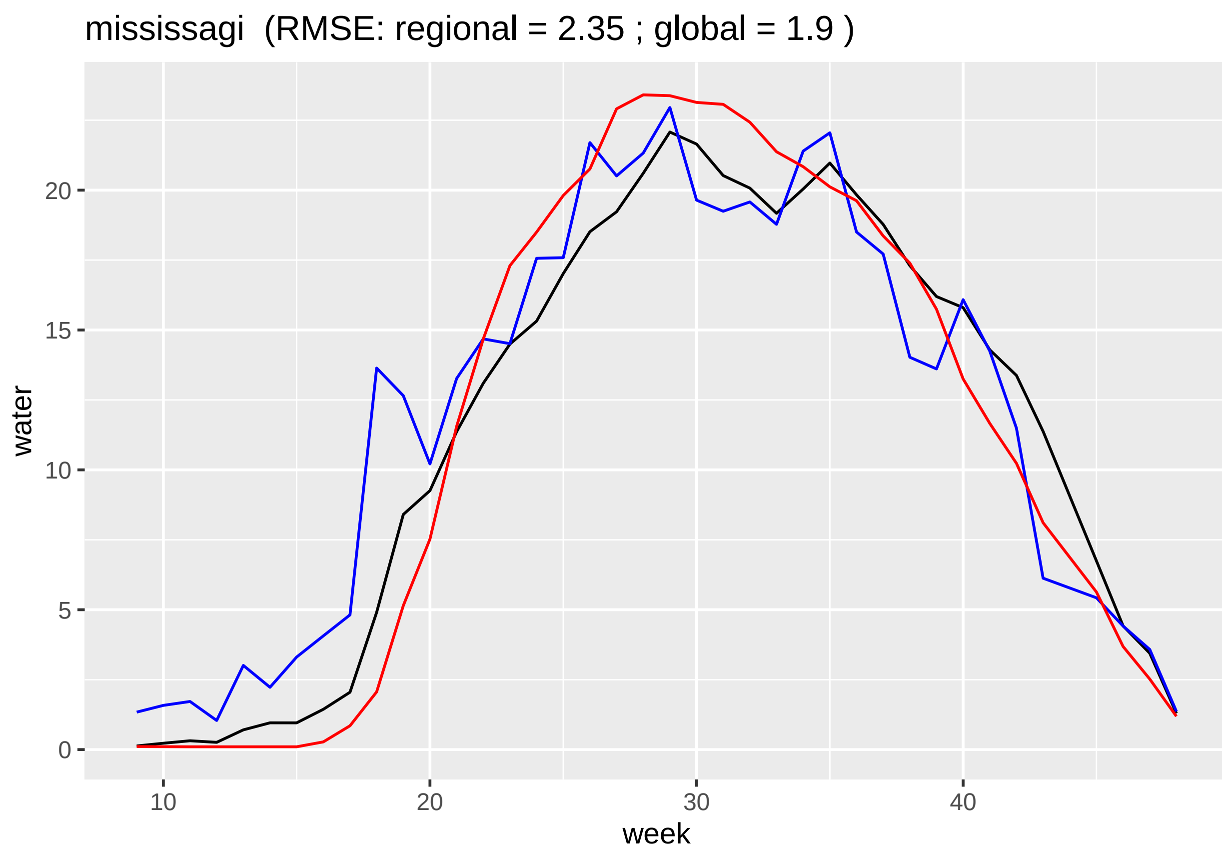
Create comparison plot for tributaries

Mississagi River, Stlouis River…

```r
df <- grand.week[[5]][[2]]

plot.df1 <- df[df$location == "bigcreek",]
plot.df2 <- df[df$location == "mississagi",]

diff1.r <- round(sqrt(mean((plot.df1$preds.ar-plot.df1$water)^2)),2)
diff1.g <- round(sqrt(mean((plot.df1$preds.futureS-plot.df1$water)^2)),2)

diff2.r <- round(sqrt(mean((plot.df2$preds.ar-plot.df2$water)^2)),2)
diff2.g <- round(sqrt(mean((plot.df2$preds.futureS-plot.df2$water)^2)),2)


mis <- ggplot(data=plot.df1, aes(x=week))+
        geom_line(aes(x=week, y=water), color = "black")+
        geom_line(aes(x=week, y=preds.ar), color = "blue")+
        geom_line(aes(x=week, y=preds.futureS), color = "red")+
        ggtitle(paste(
          plot.df1$location, " (RMSE: regional =", diff1.r, ";",
          "global =", diff1.g, ")"))

big <- ggplot(data=plot.df2, aes(x=week))+
        geom_line(aes(x=week, y=water), color = "black")+
        geom_line(aes(x=week, y=preds.ar), color = "blue")+
        geom_line(aes(x=week, y=preds.futureS), color = "red")+
        ggtitle(paste(
          plot.df2$location, " (RMSE: regional =", diff2.r, ";",
          "global =", diff2.g, ")"))

print(mis)
```


bigcreek (RMSE: regional = 1.44 ; global = 2.16)
water
20
15
10
5
10
20
30
40
week

```
print(big)
```




```
#ggarrange(mis, big, nrow=1)
```

NOT NEEDED (previous code): Convert into weekly output

```
# new dataframe to store all weekly results
grand.week = grand.nonlinear

for (season in c(1,2,3,5)) {

  for(i in 1:fold) {
    ## Convert lag5 model prediction to weekly timescale
    df.days <- grand.nonlinear[[season]][[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.ar~week+location,FUN=mean,data=df.days,na.action=na.omit),
      aggregate(water~week+location,FUN=mean,data=df.days,na.action=na.omit),
      all=TRUE) %>%
      na.omit() %>%
      arrange(location, week)

    comp.week <- merge(df.week, futurestream.temp, by=c("location","week")) %>%
      arrange(location, week)

    # store it back into the grand.nonlinear list
```

```
    grand.week[[season]][[i]] <- comp.week
  }
}
```