Master temperature model comparison

Eddie Wu

2024-08-19

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(nlme)
library(xts)
library(zoo)
library(lubridate)
library(drc)
library(ggplot2)
```

```
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)
}
```

SECTION 1: Data importing and cleaning

```
## Data import
air <- read.csv("tributary air temperatures clean.csv", stringsAsFactors=F)
```

```r
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

```r
## 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),
         dmean_6 = lag(mean_temp, 6))


## 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+dmean_6)/7) %>%
  ungroup() %>%
  as.data.frame()
```

Get final master temp dataframe

```r
master.temp <- awf[complete.cases(cbind(awf$mean_temp,awf$temp)),] %>%
  dplyr::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, dmean_6, flow, cair)


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

SECTION 2: Get training and testing

Identify good year/month data

We want the seasonal data to be greater than 60 days, annual data (winter removed) more than 180 days.

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


## Seasonal (60 days)
fulspring <- good_year(master.spring, 60)
fulsum <- good_year(master.sum, 60)
fulfall <- good_year(master.fall, 60)
fulwin <- good_year(master.win, 60)
```

Get training and testing

Similarly, get training and testing for the specific season.

```r
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)
```

Simple get training and testing

```
ytrain = c(2006,2013,2013,2011,2001,2011,2008,2011,2014,2002,2011,2013)
ytest = c(2003,2012,2014,2013,2005,2013,2009,2012,2013,2005,2013,2014)
y = cbind(loc_seq,ytrain,ytest)

current.training <- merge(master.temp, y,
                          by.x = c("location", "year"),
                          by.y = c("loc_seq", "ytrain")) %>%
  arrange(location, date) %>% na.omit()

current.testing <- merge(master.temp, y,
                          by.x = c("location", "year"),
                          by.y = c("loc_seq", "ytest")) %>%
  arrange(location, date) %>% na.omit()


# breakpoint
current.training <-current.training %>%
  filter(yday(date)>91 & yday(date)<204)

current.testing <-current.testing %>%
  filter(yday(date)>91 & yday(date)<204)

ctrl = glsControl(opt='optim')


## dataframe to store all the results
rmse.r <- matrix(NA, ncol=4, nrow=12)
colnames(rmse.r) <- c("linear","seasonal","nonlinear","future")
rownames(rmse.r) <- loc_seq

bias.r <- matrix(NA, ncol=4, nrow=12)
colnames(bias.r) <- c("linear","seasonal","nonlinear","future")
rownames(bias.r) <- loc_seq

nsc.r <- matrix(NA, ncol=4, nrow=12)
colnames(nsc.r) <- c("linear","seasonal","nonlinear","future")
rownames(nsc.r) <- loc_seq
```

```
## list to store one of the weekly aggregated water temp
complist <- vector("list",12)
names(complist) <- loc_seq
```

SECTION 3: REGIONAL model comparison

We now run each of the three models only once.

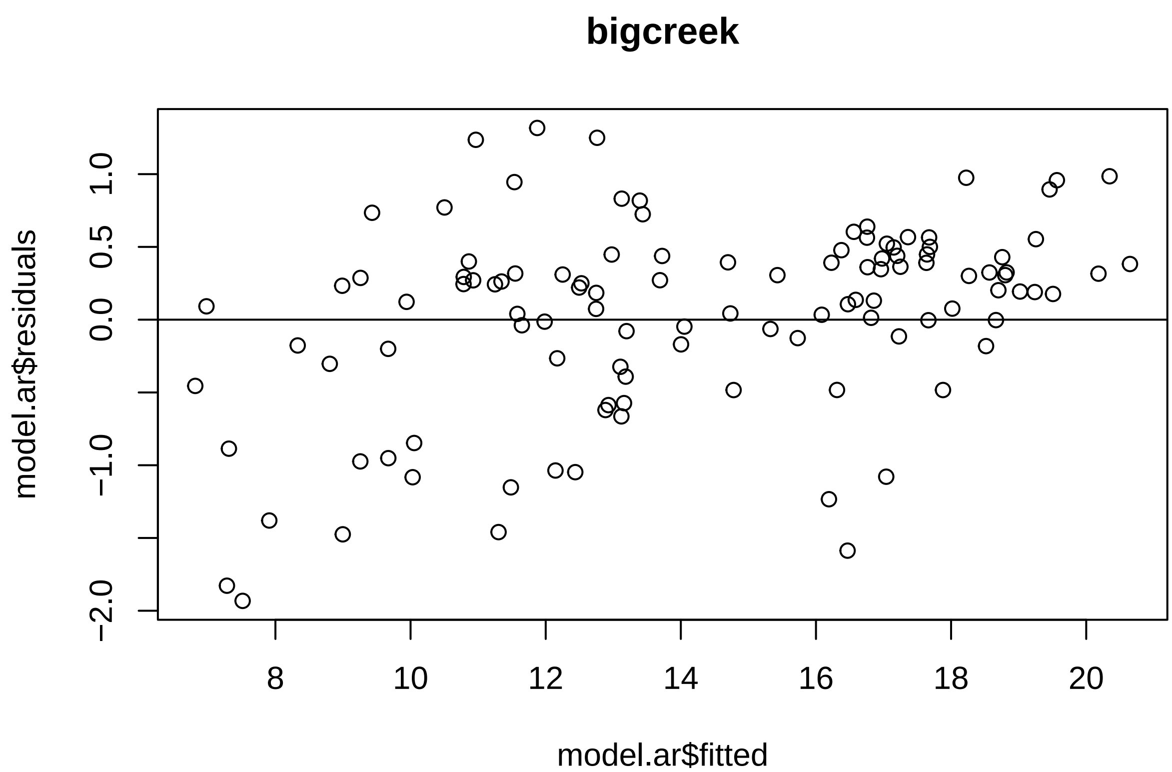
We run all three candidate models on 12 tributary locations separately. For each location, the model assumption is checked to ensure that the prediction is valid.

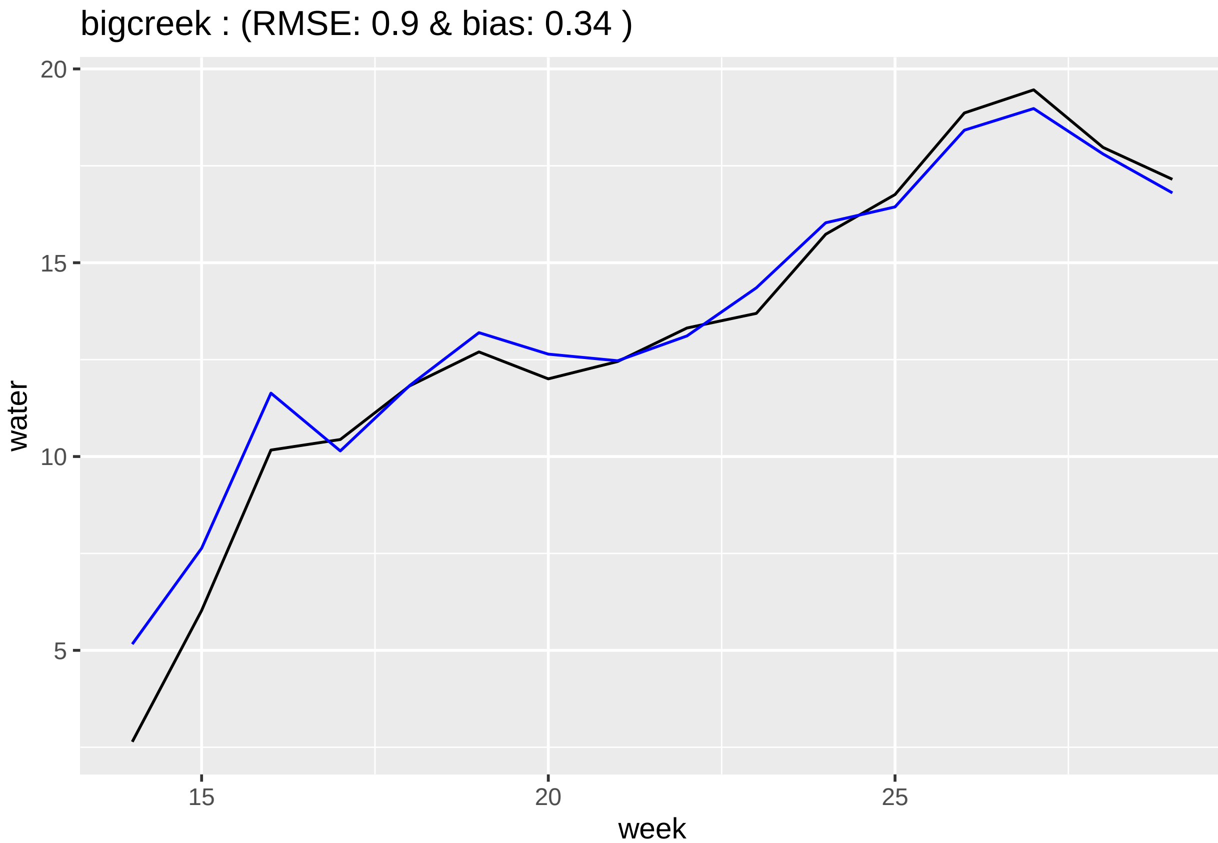
Model predictions are aggregated to weekly scale, and all performance metrics are calculated on a weekly scale to allow comparison to the global model.

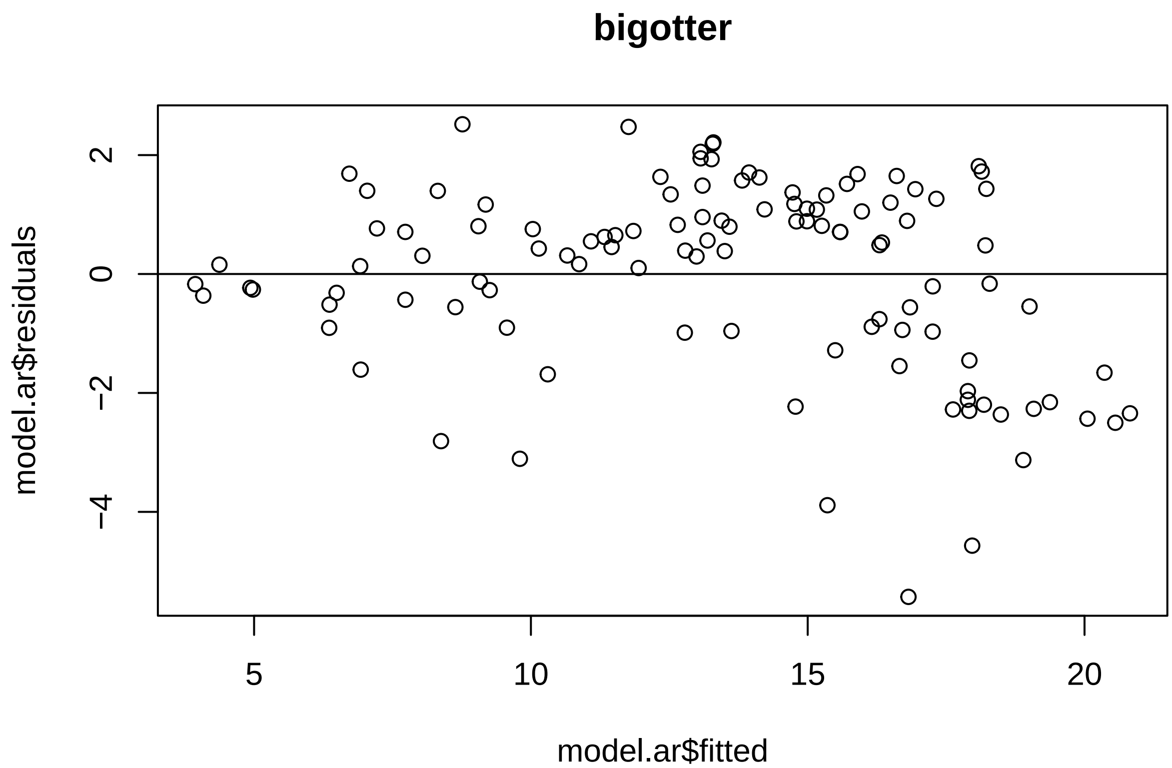
Linear lag5

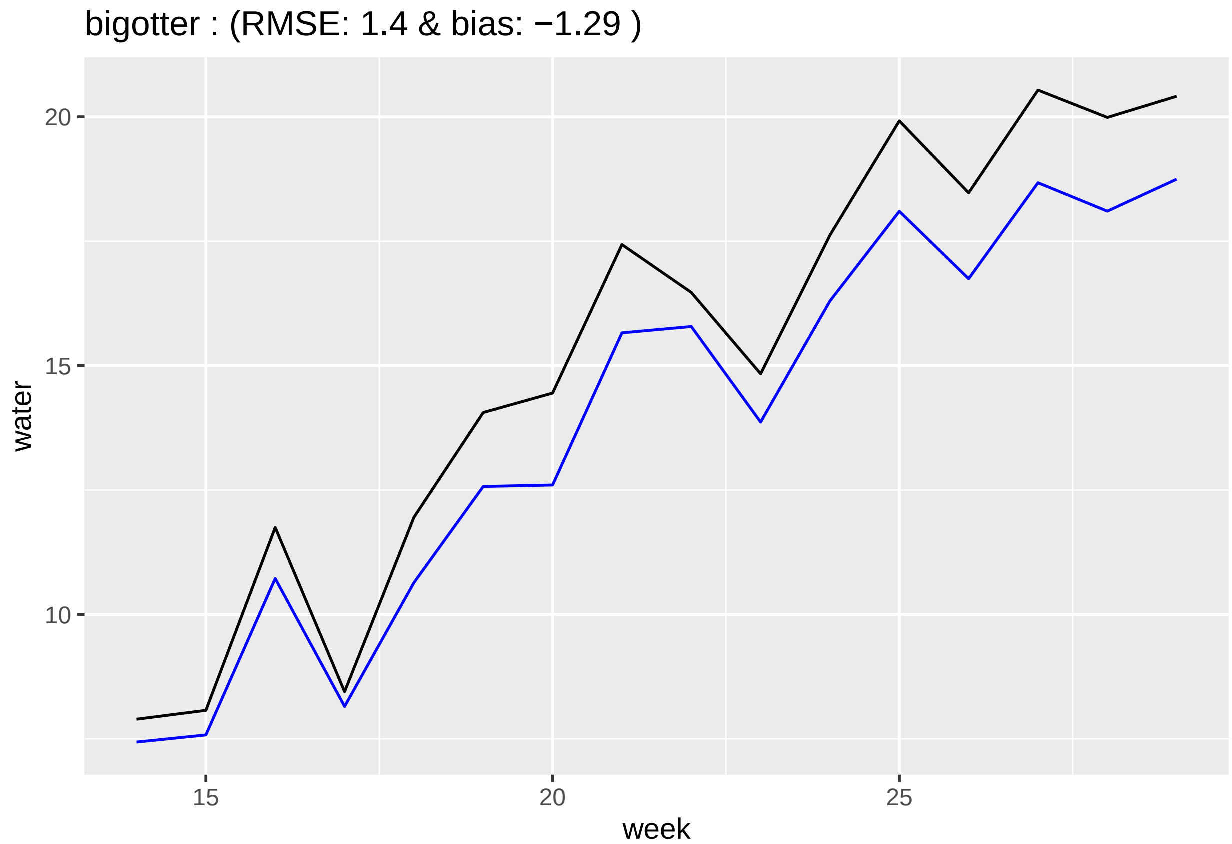
```
## Coefs
coef.lag6 <- matrix(NA, ncol=8, nrow=12)
colnames(coef.lag6) <- c("a","b1","b2","b3","b4","b5","b6","b7")
rownames(coef.lag6) <- loc_seq


## Forms
form.locrandom <- water ~ air + dmean_1 + dmean_2 + dmean_3 + dmean_4 + dmean_5 + dmean_6

# For each location
for (loc in loc_seq){

  dftrain = current.training[current.training$location == loc,]
  dftest = current.testing[current.testing$location == loc,]
  compare = dftest

  # model
  model.ar <- gls(form.locrandom, control = ctrl,
                  na.action = na.omit, data = dftrain,
                  correlation=corAR1())
  compare$preds.ar <- as.vector(predict(model.ar, newdata = dftest))
  compare$week <- ceiling(as.numeric(format(compare$date, "%j"))/7)

  # 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)

  complist[[loc]][[1]] <- compare.w

  # store in results dataframes
  rmse.r[loc,1]=round(sqrt(mean((compare.w$preds.ar-compare.w$water)^2)),2)
  bias.r[loc,1]=round(mean(compare.w$preds.ar-compare.w$water),2)
  nsc.r[loc,1]=round(1-sum((compare.w$water-compare.w$preds.ar)^2) / sum((compare.w$water-mean(compare.
```

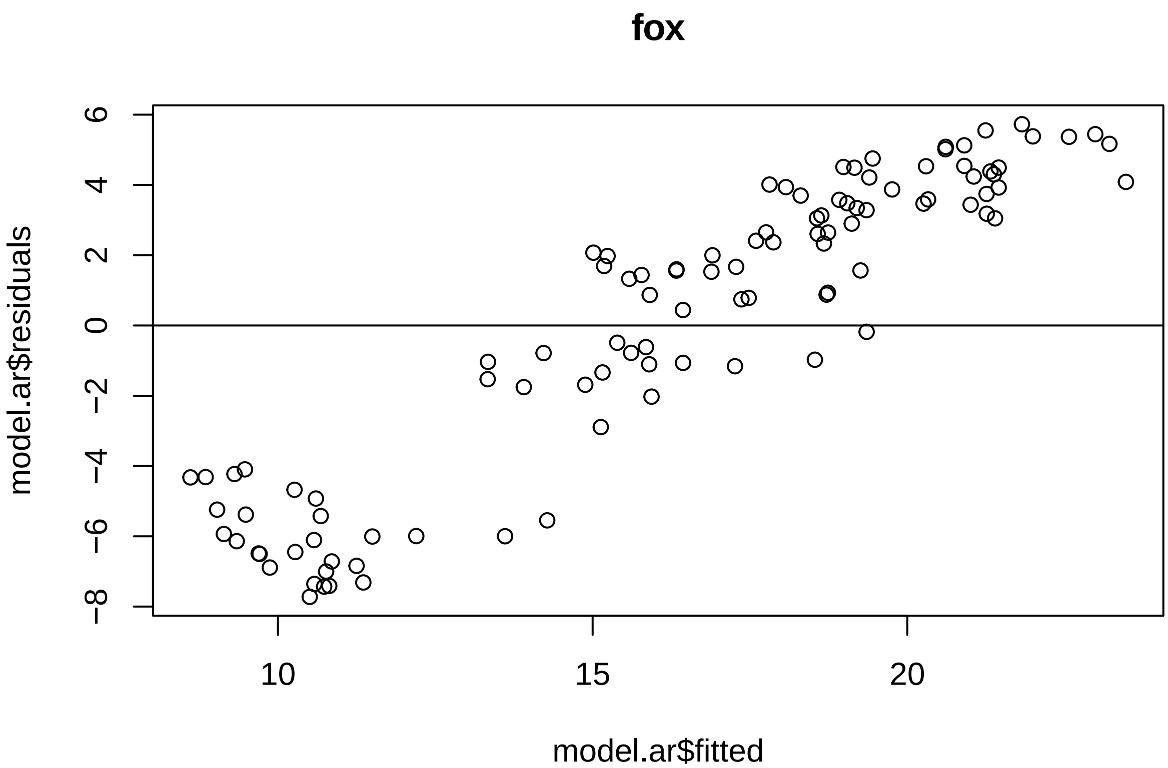
```r
## assumptions
plot(model.ar$residuals~model.ar$fitted, main=loc)
abline(h=0)


## prediction plots - weekly

pl <- ggplot(data=compare.w, aes(x=week))+
        geom_line(aes(x=week, y=water), color = "black")+
        geom_line(aes(x=week, y=preds.ar), color = "blue")+
        ggtitle(paste(
          loc, ": (RMSE:", rmse.r[loc,1], "&", "bias:",
          bias.r[loc,1], ")"))
print(pl)


## get coefs
dfcoef <- rbind(dftrain[,1:17], dftest[,1:17])
model.ar <- gls(form.locrandom, control = ctrl,
                na.action = na.omit, data = dfcoef,
                correlation=corAR1())
for (c in 1:8){coef.lag6[loc,c]=signif(coef(model.ar)[c],2)}
}
```

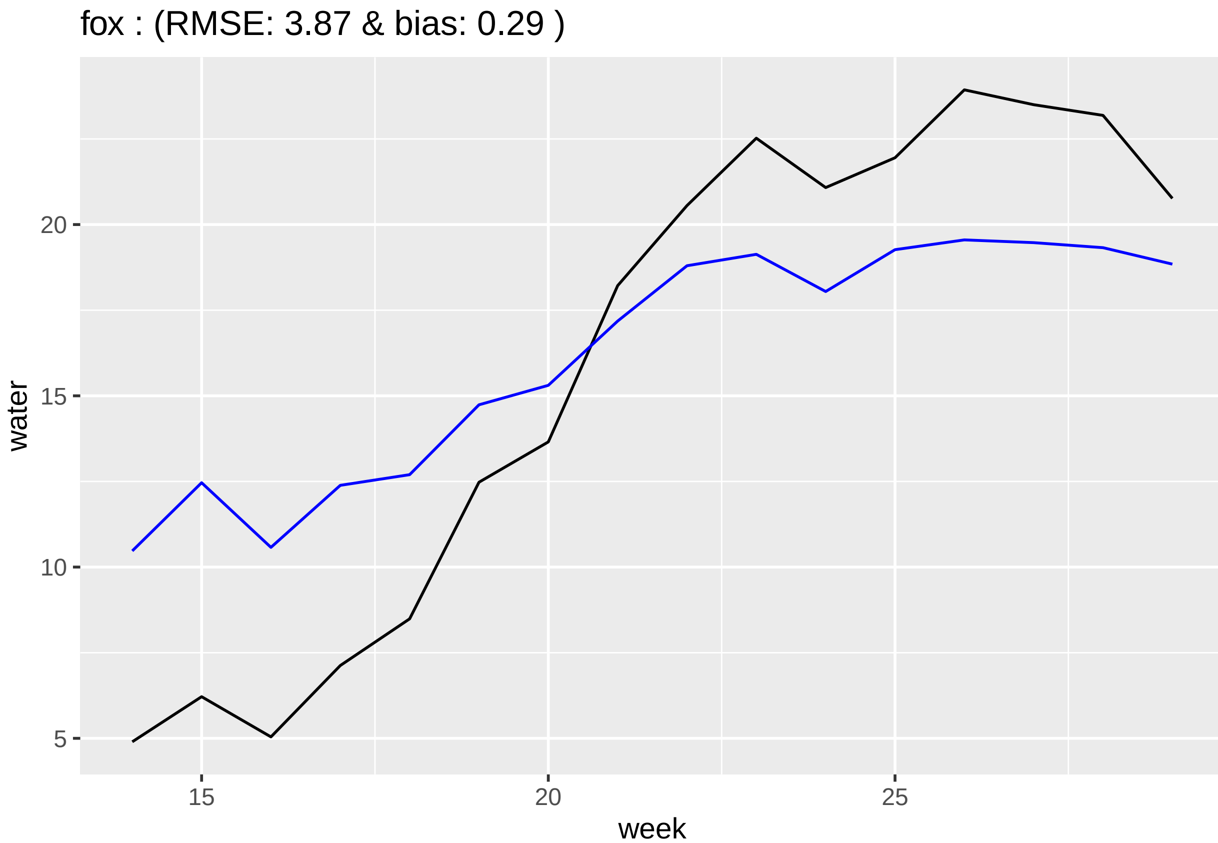


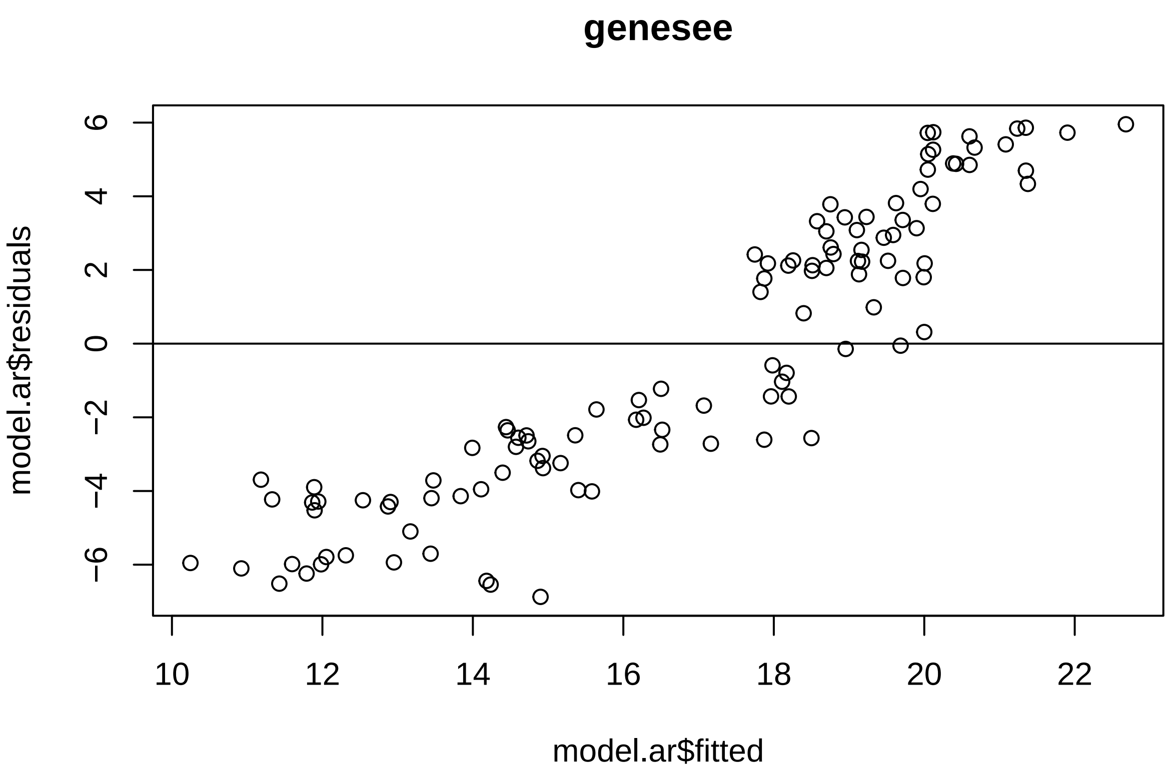

bigcreek : (RMSE: 0.9 & bias: 0.34)
20
15
10
5
water
15
20
25
week

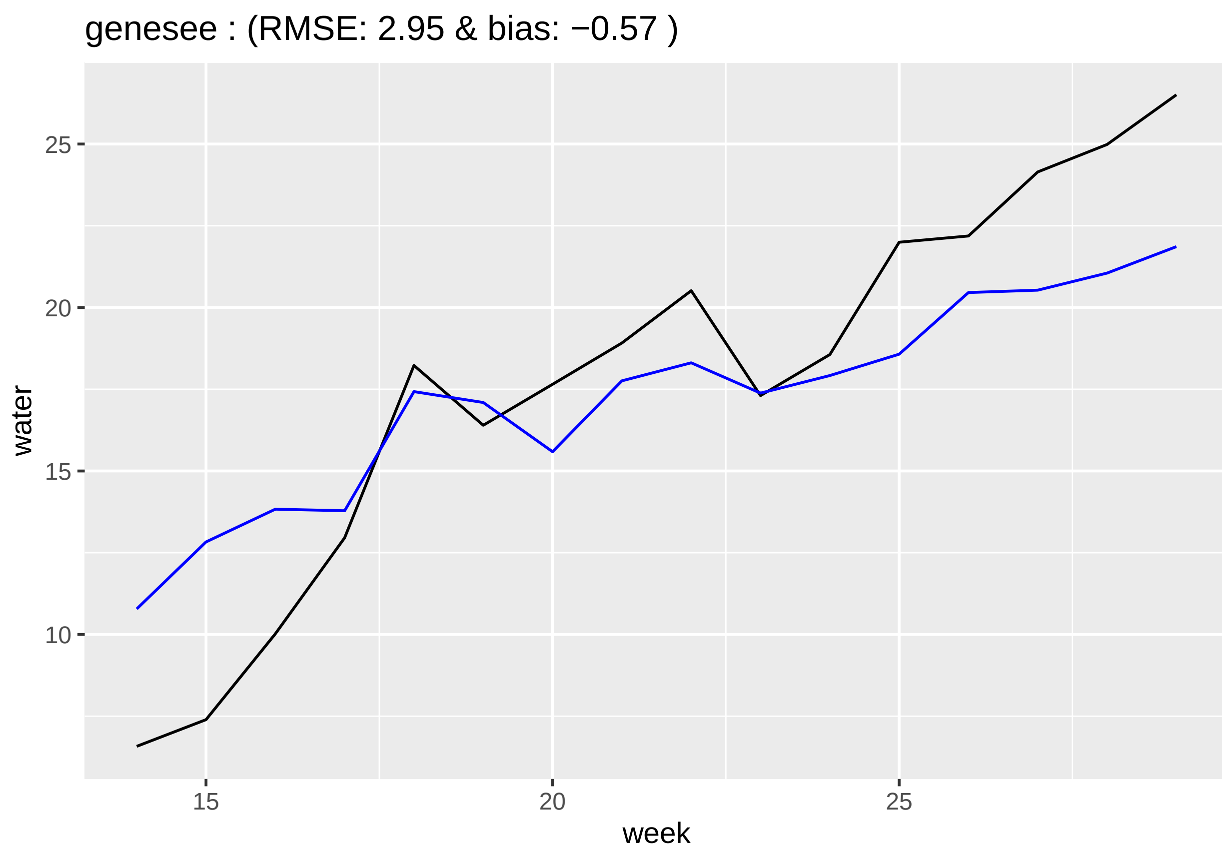

bigotter
model.ar$residuals
2
0
−2
−4
5
10
15
20
model.ar$fitted

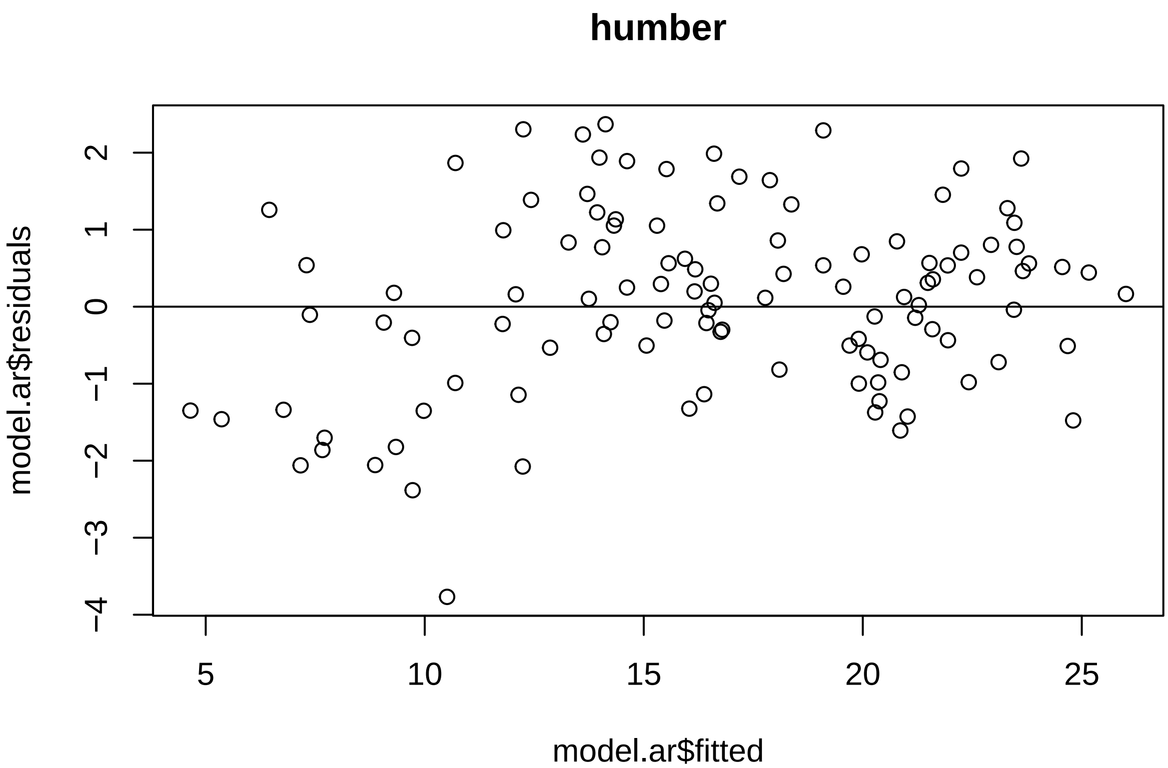

bigotter : (RMSE: 1.4 & bias: −1.29)
20
15
10
water
15
20
25
week

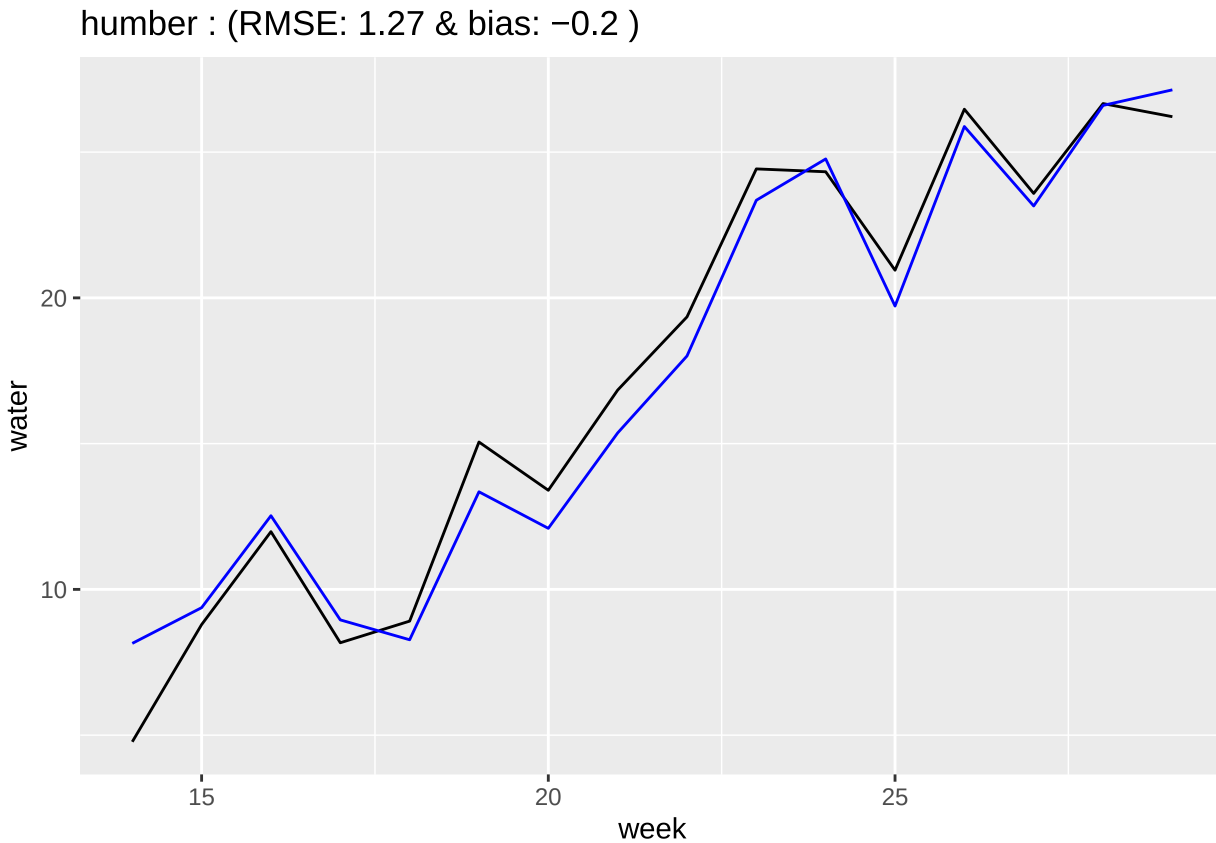

fox
model.ar$residuals
model.ar$fitted
6
4
2
0
−2
−4
−6
−8
10
15
20

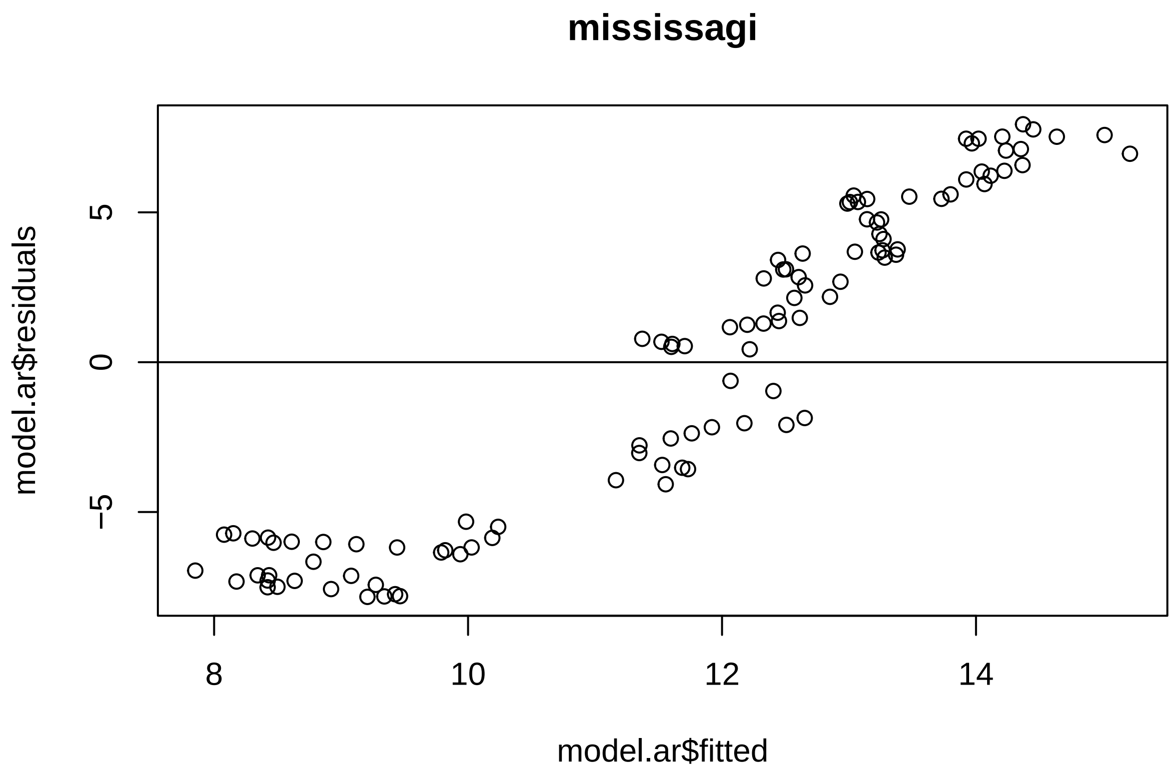

fox : (RMSE: 3.87 & bias: 0.29)
water
20
15
10
5
15
20
25
week

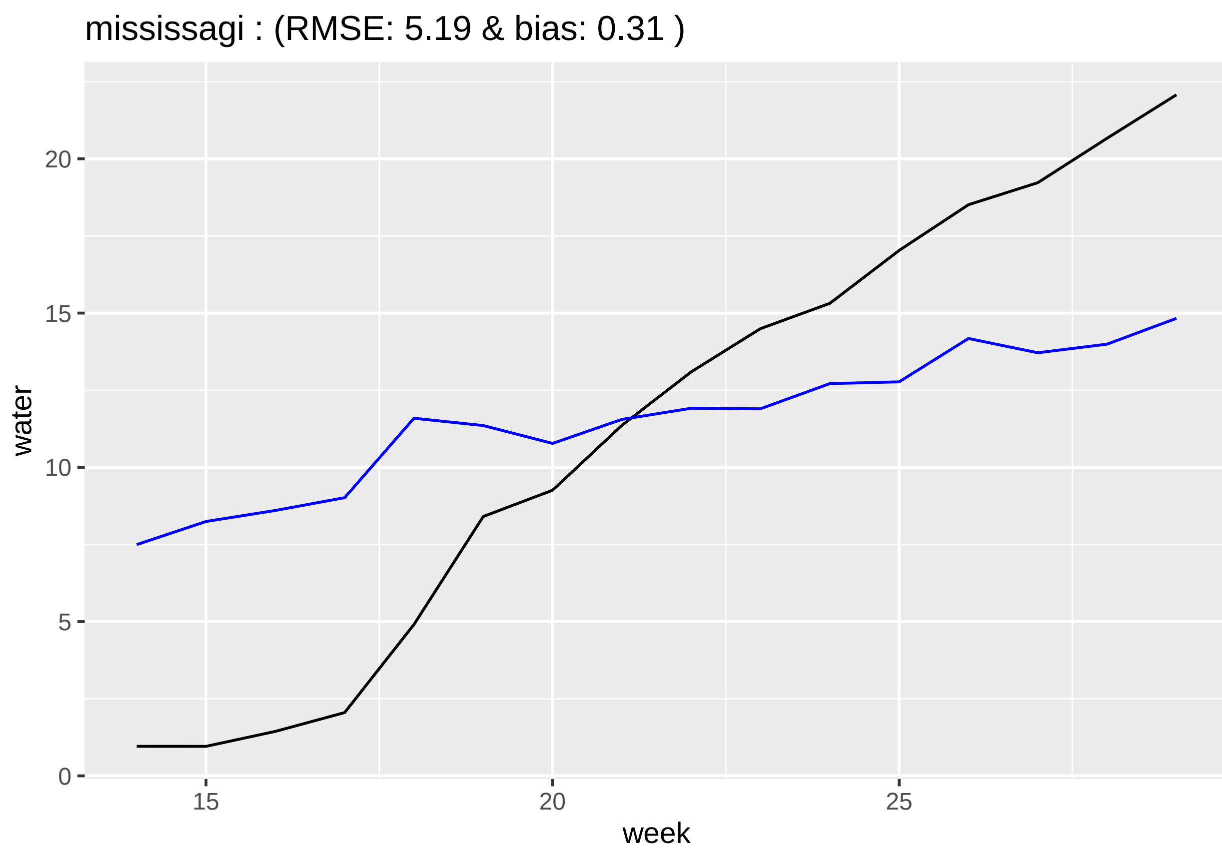

genesee
model.ar$residuals
6
4
2
0
−2
−4
−6
10
12
14
16
18
20
22
model.ar$fitted


genesee : (RMSE: 2.95 & bias: −0.57)
water
25
20
15
10
15
20
25
week

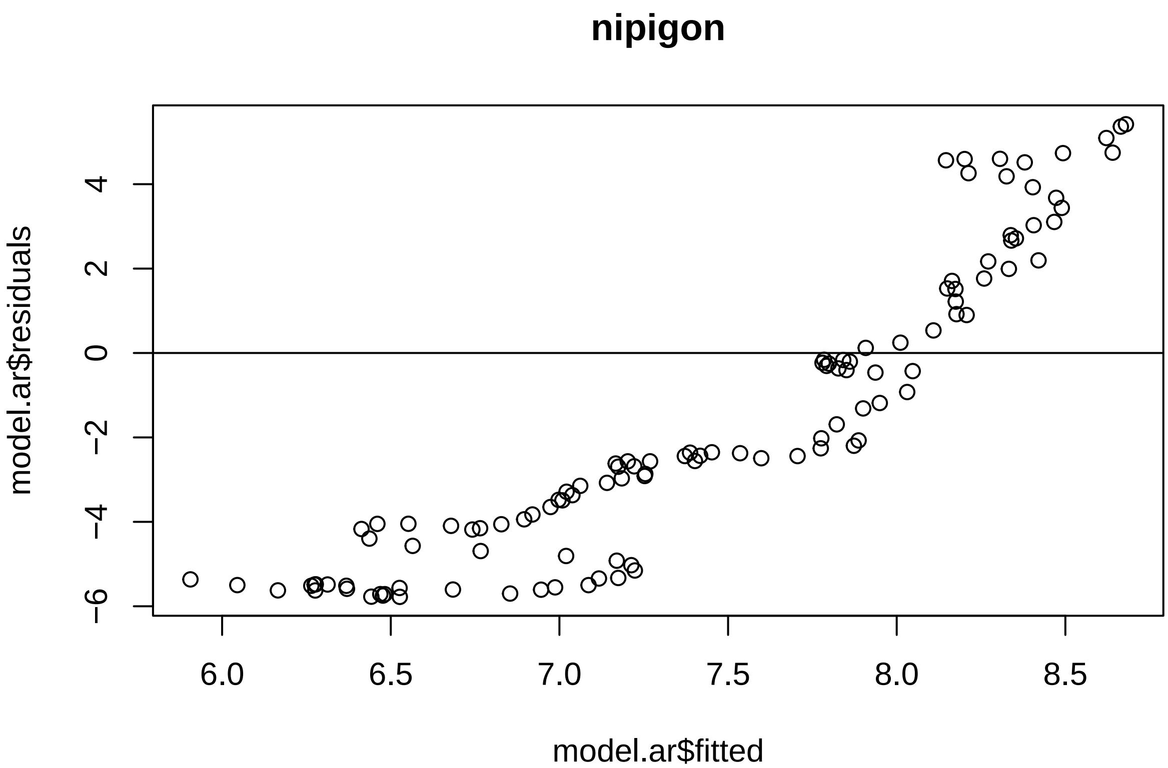

humber
model.ar$residuals
model.ar$fitted
−4
−3
−2
−1
0
1
2
5
10
15
20
25

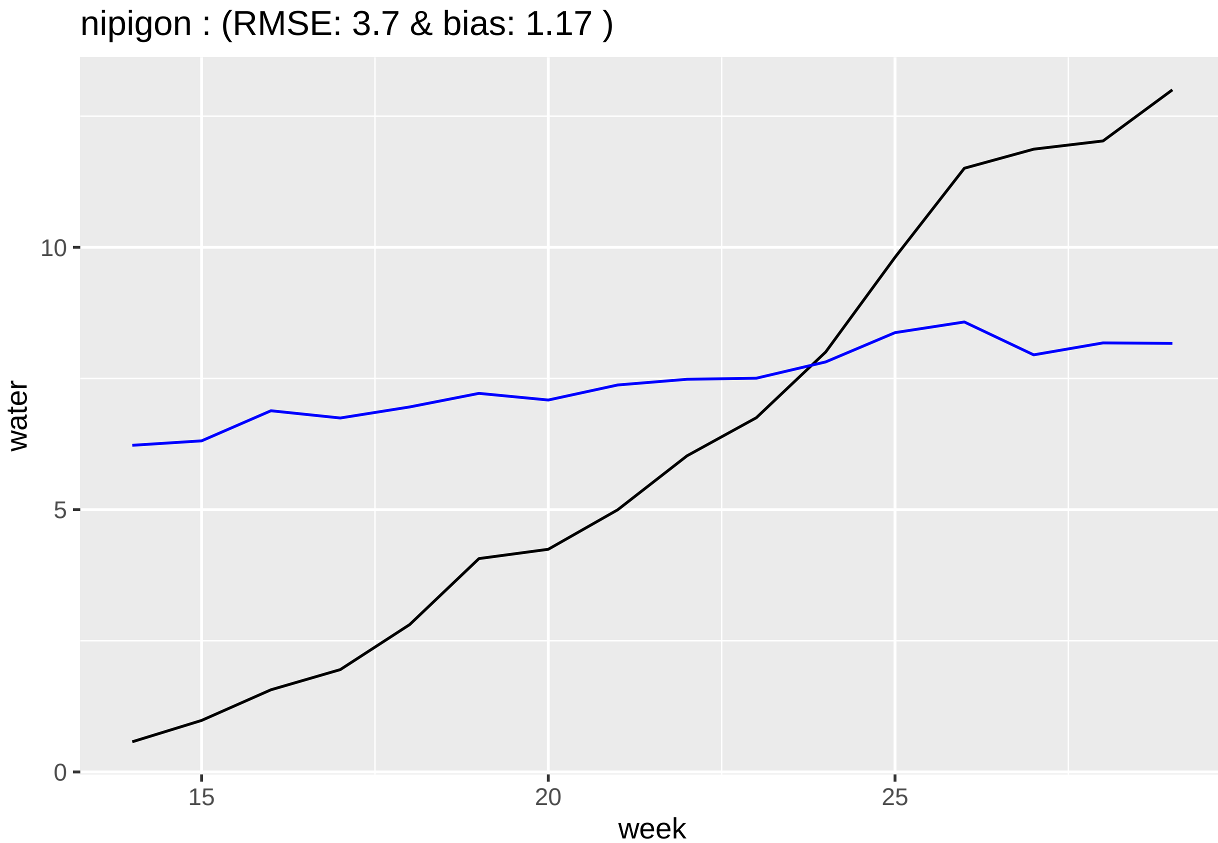

humber : (RMSE: 1.27 & bias: −0.2)
20
10
water
15
20
25
week

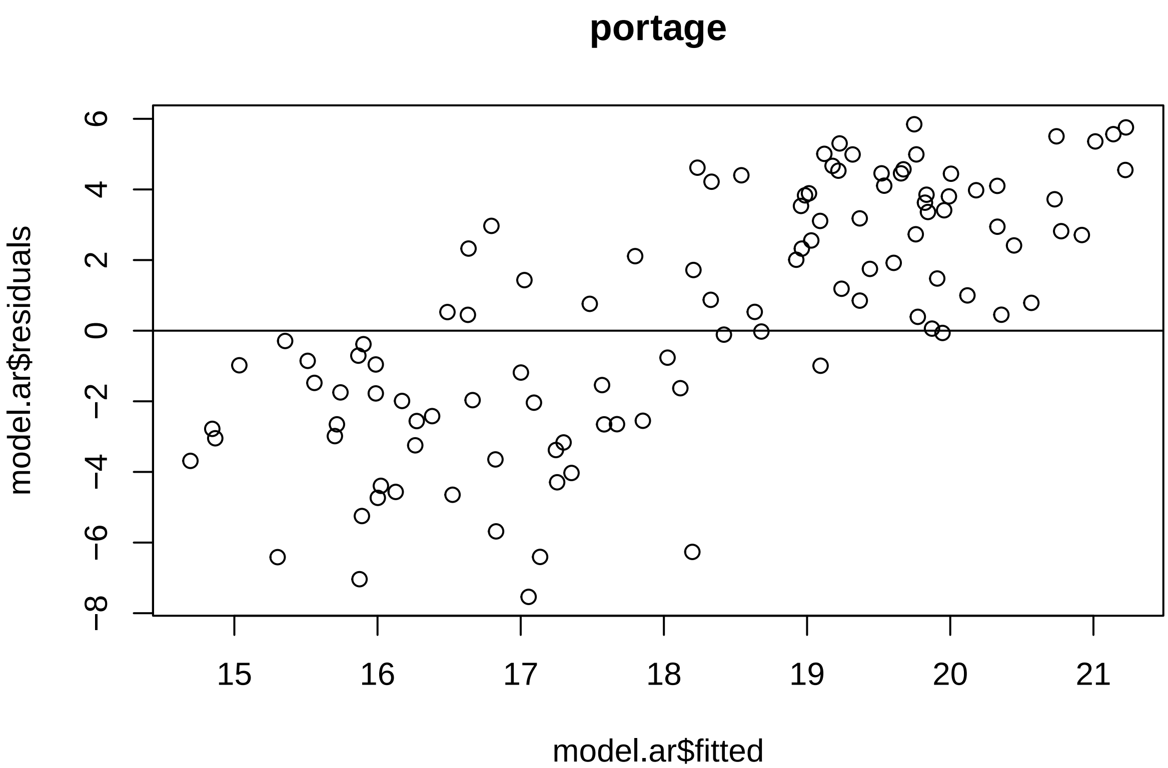

mississagi
model.ar$residuals
model.ar$fitted
5
0
−5
8
10
12
14

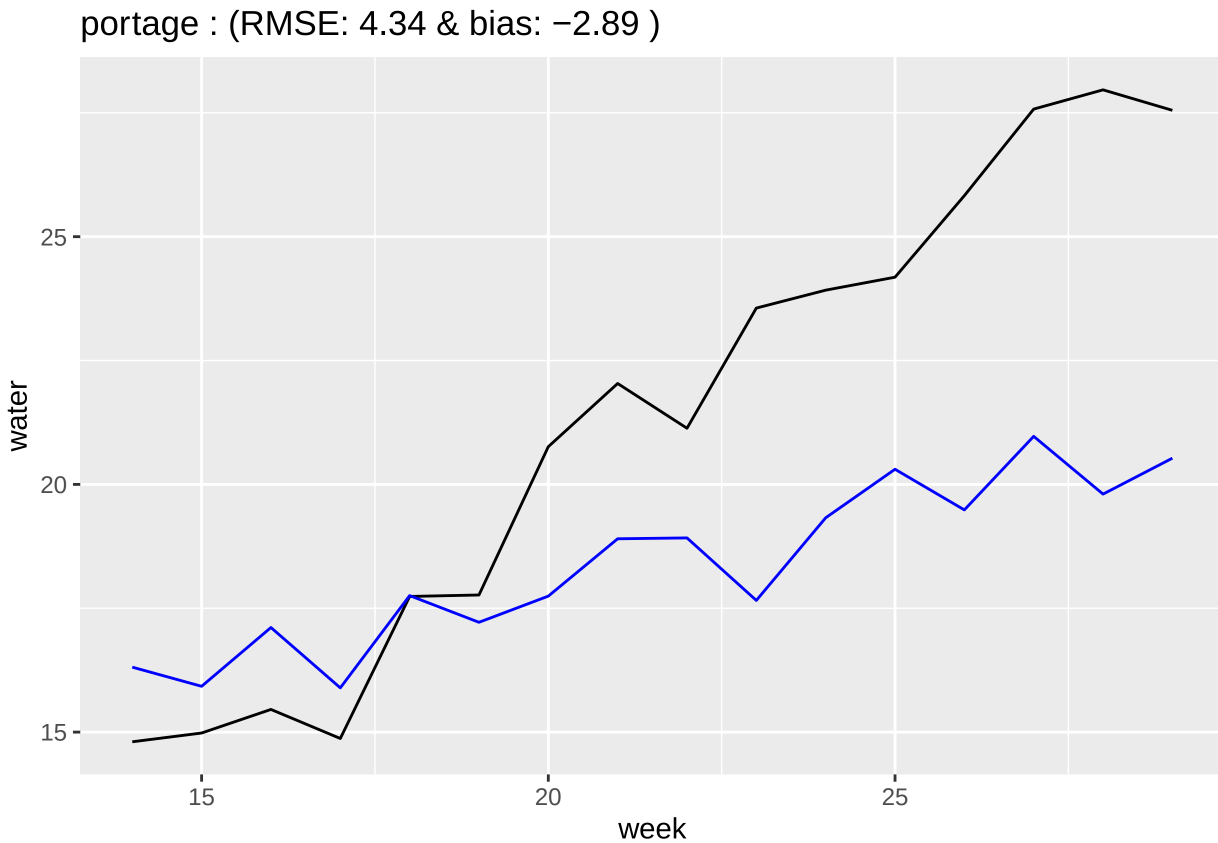


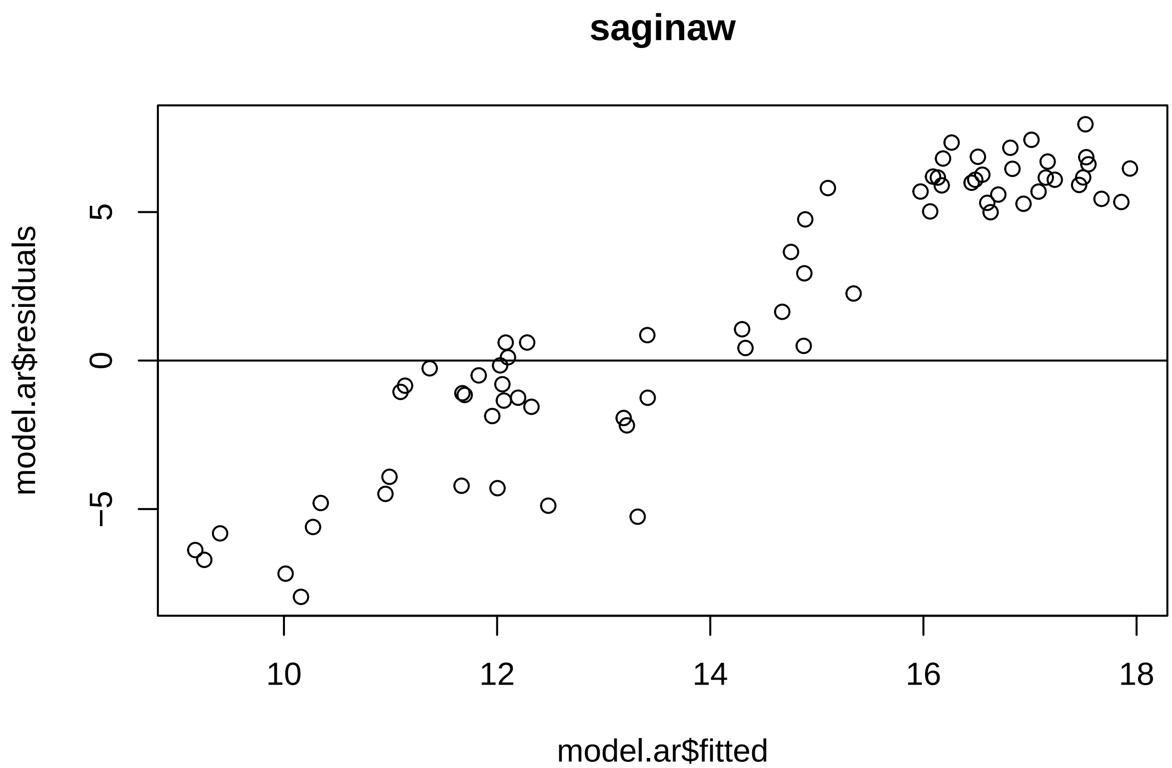
mississagi : (RMSE: 5.19 & bias: 0.31)
water
20
15
10
5
0
15
20
25
week


nipigon
model.ar$residuals
model.ar$fitted
4
2
0
−2
−4
−6
6.0
6.5
7.0
7.5
8.0
8.5

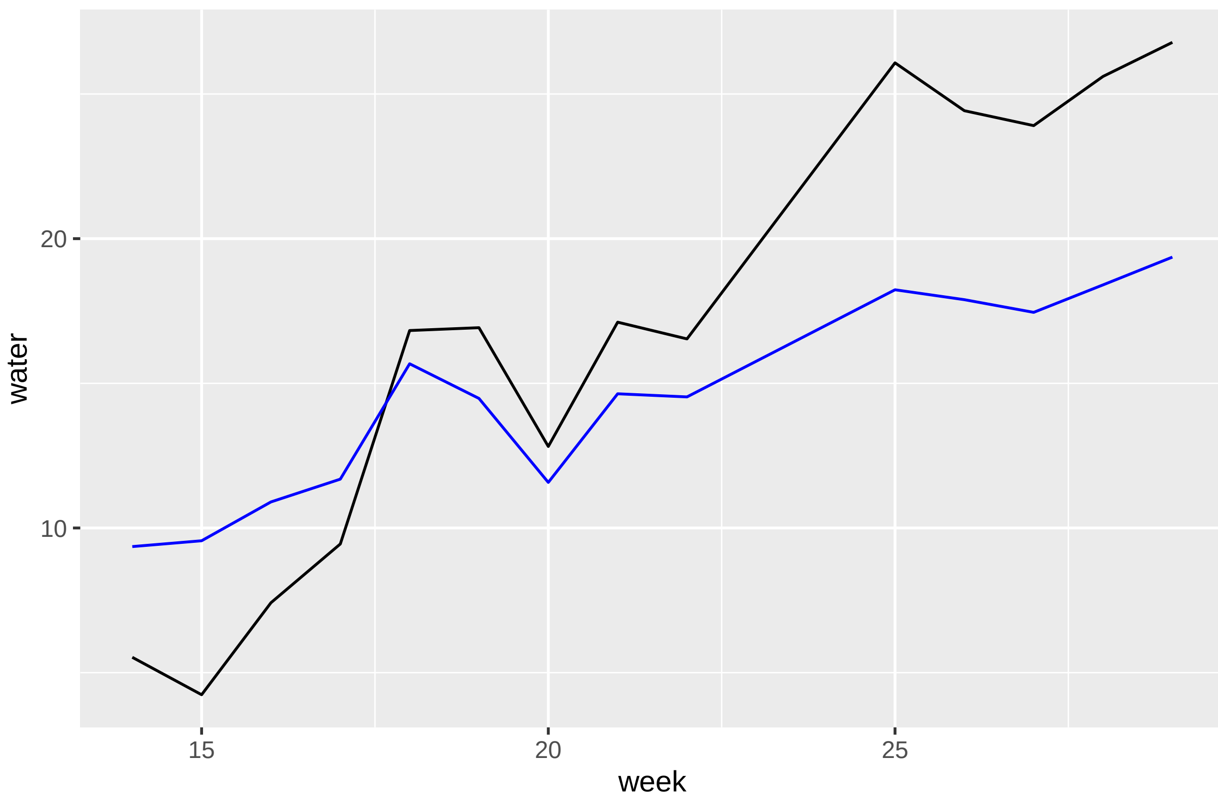

nipigon : (RMSE: 3.7 & bias: 1.17)
water
10
5
0
15
20
25
week

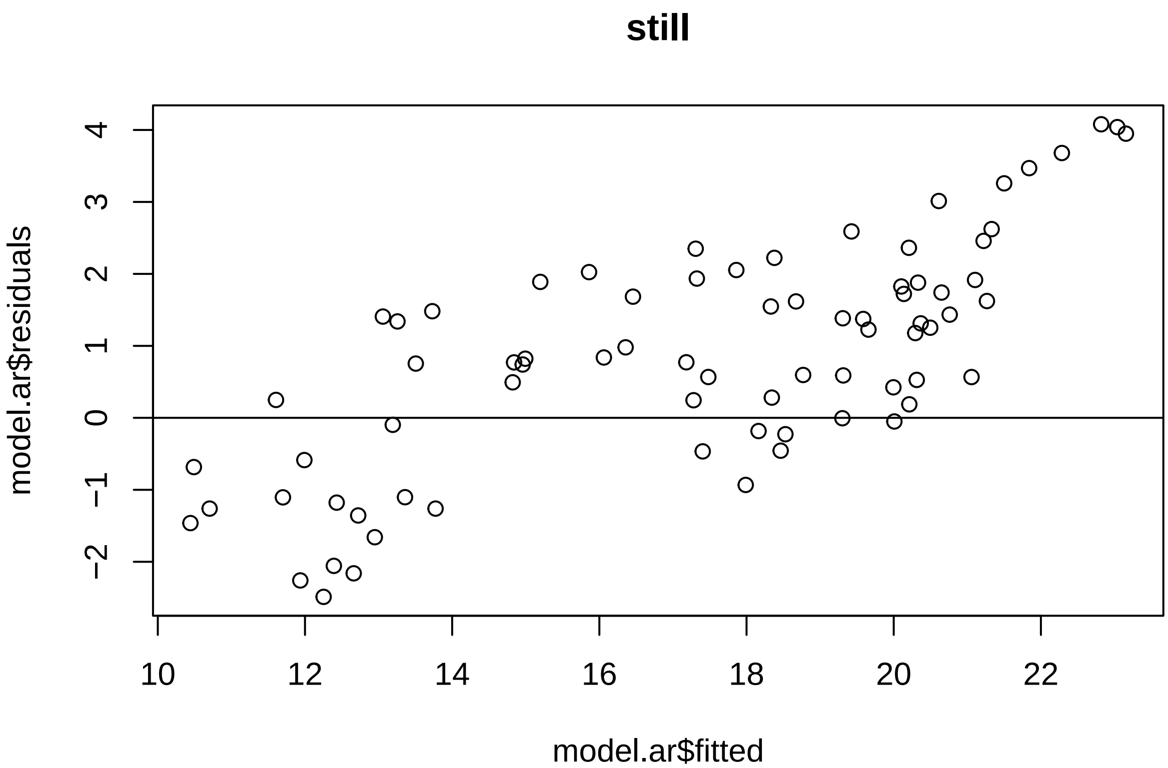

portage
model.ar$residuals
6
4
2
0
−2
−4
−6
−8
15
16
17
18
19
20
21
model.ar$fitted

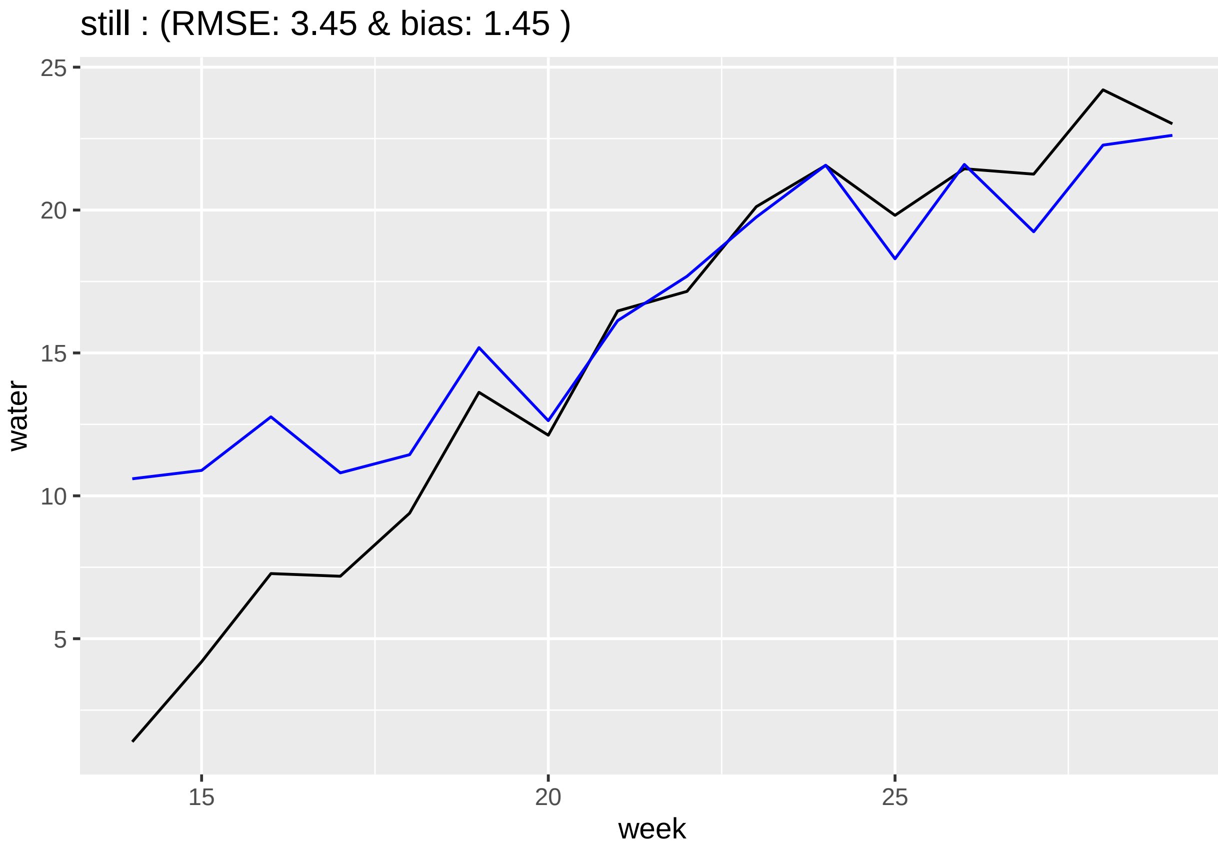

portage : (RMSE: 4.34 & bias: −2.89)
25
20
15
water
15
20
25
week

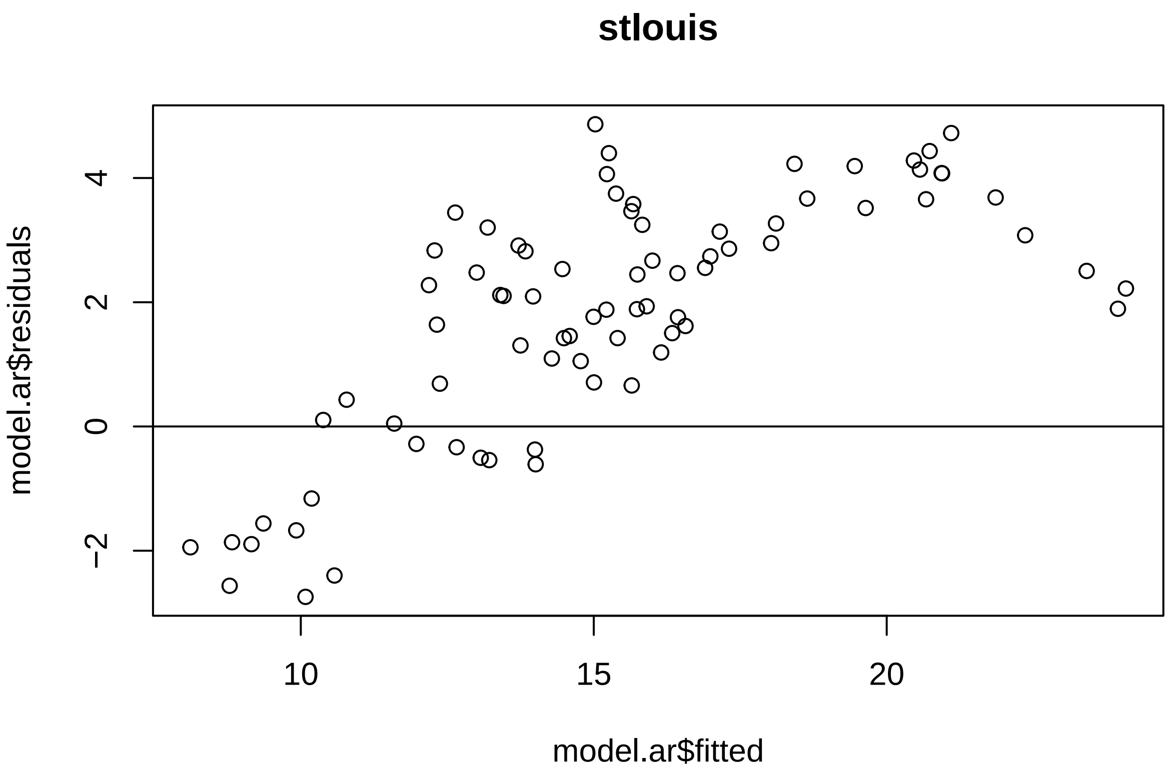

saginaw
model.ar$residuals
model.ar$fitted
5
0
−5
10
12
14
16
18

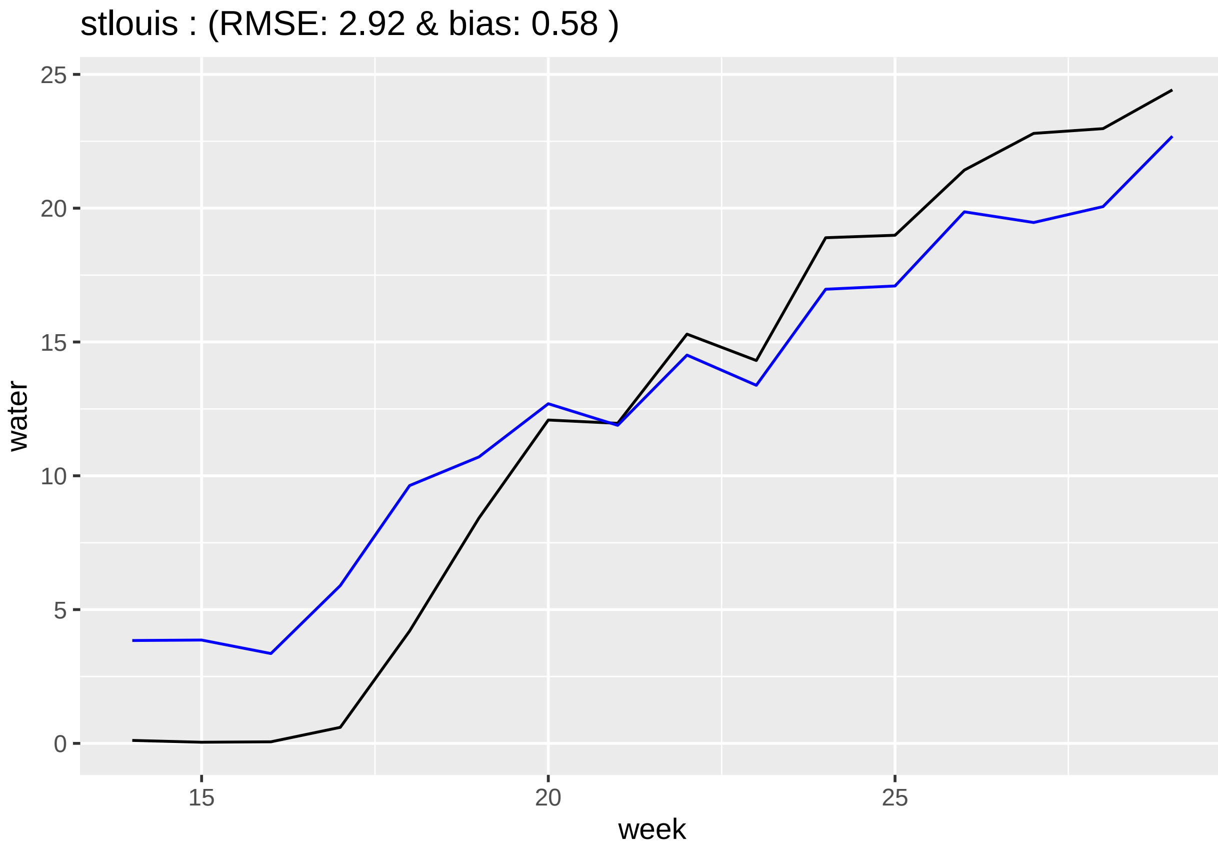
saginaw : (RMSE: 4.87 & bias: −2.14)

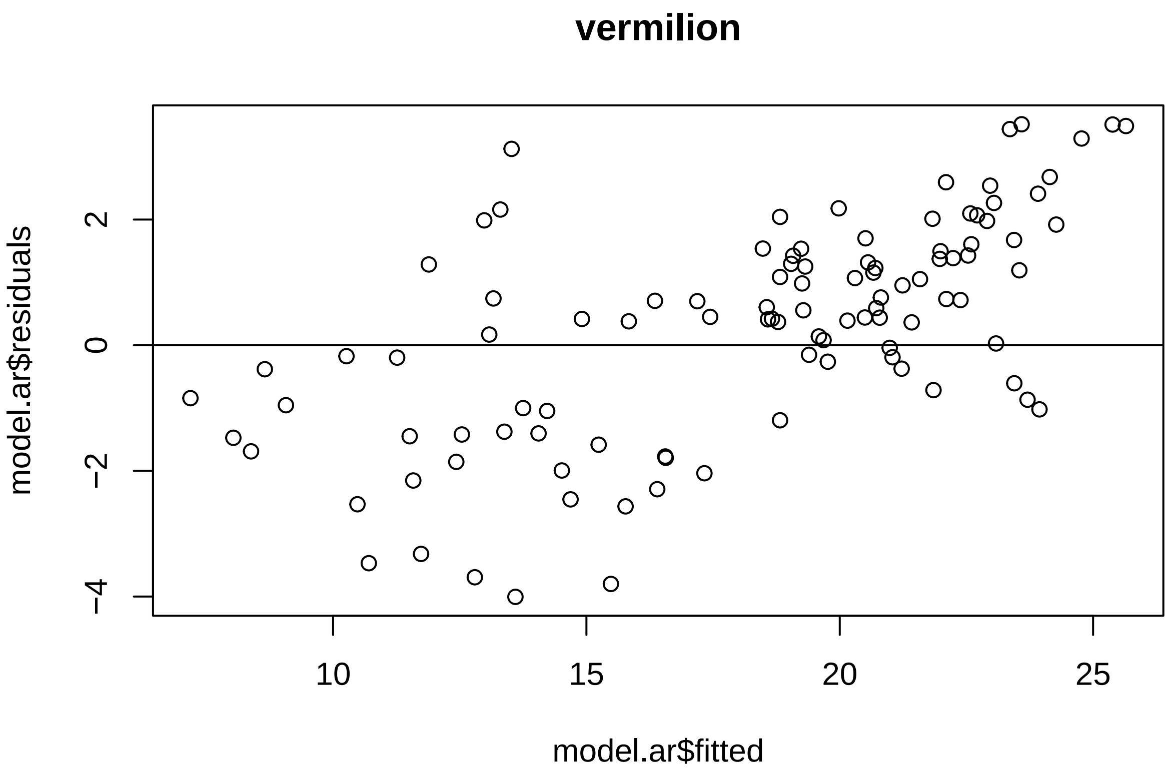



still
model.ar$residuals
model.ar$fitted
4
3
2
1
0
−1
−2
10
12
14
16
18
20
22

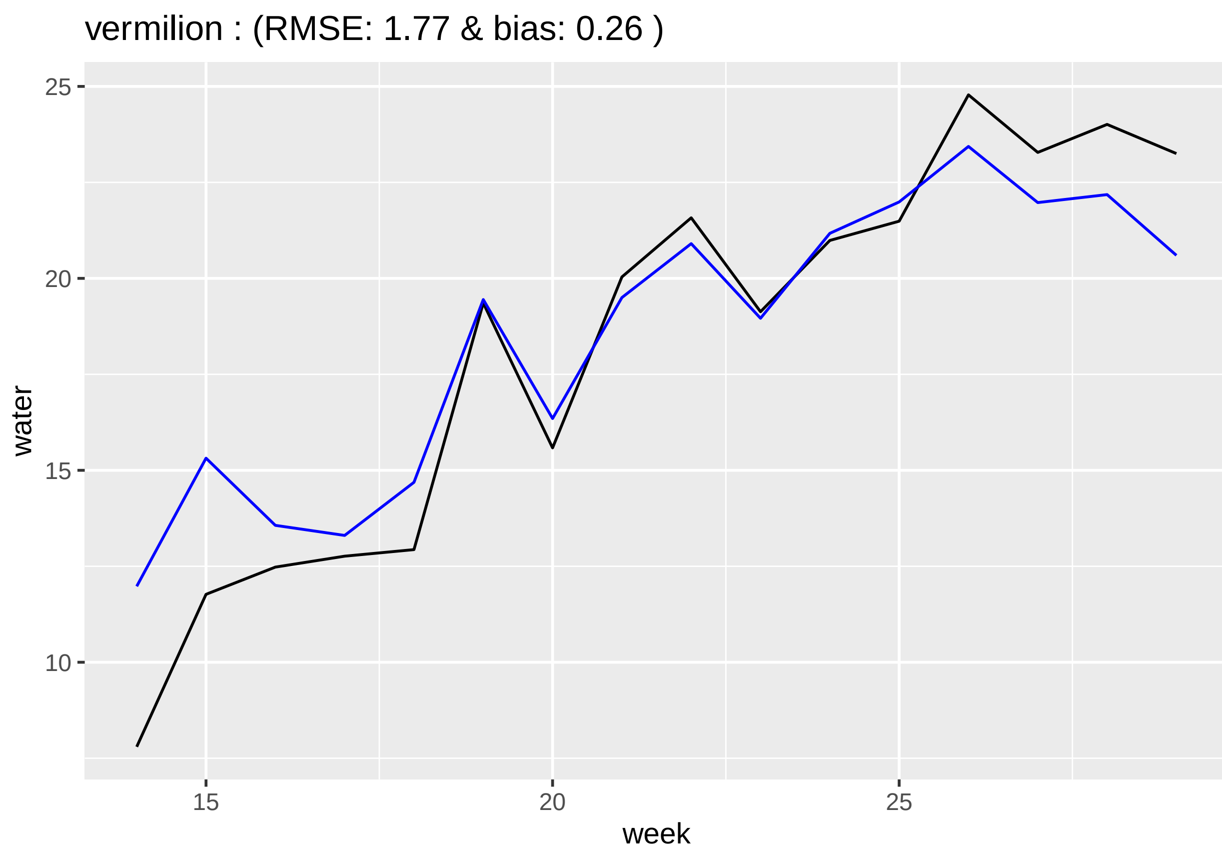

still : (RMSE: 3.45 & bias: 1.45)
25
20
15
10
5
water
15
20
25
week


stlouis
model.ar$residuals
−2
0
2
4
10
15
20
model.ar$fitted


stlouis : (RMSE: 2.92 & bias: 0.58)
25
20
15
10
5
0
water
15
20
25
week



vermilion
model.ar$residuals
model.ar$fitted
2
0
−2
−4
10
15
20
25

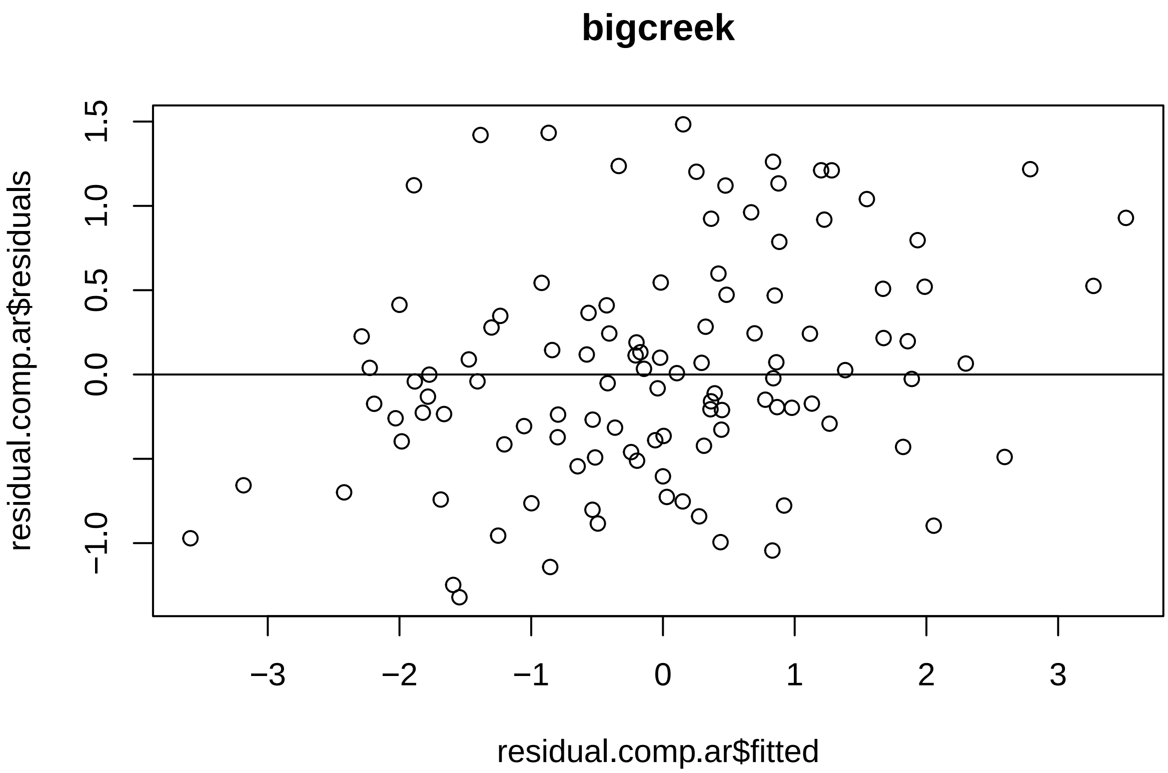


Seasonal residual

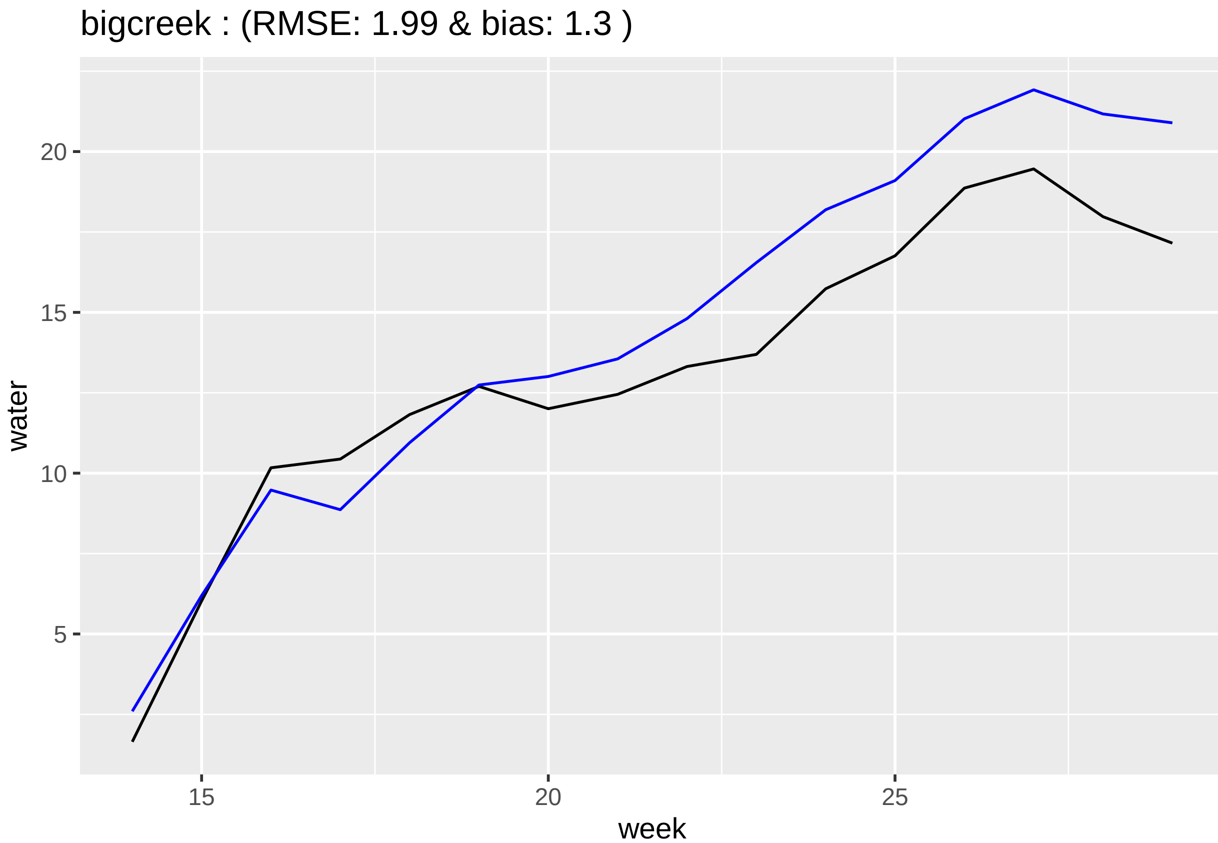
```
## Coefs
coef.season <- matrix(NA, ncol=9, nrow=12)
colnames(coef.season) <- c("a","b","c","d","f","g","beta1","beta2","beta3")
rownames(coef.season) <- loc_seq


# For each location
for (loc in loc_seq){

  dftrain = current.training[current.training$location == loc,]
  dftest = current.testing[current.testing$location == loc,]
  compare <- dftest

  # model
  annual.comp <- nls(air ~ a+b*sin(2*pi/365*(yday(date)+t0)),
                     start = list(a=0.05, b=5, t0=-26),
                     data=dftrain)

  # get the air temperature residuals
  res <- as.data.frame(matrix(NA, ncol = 2,
                              nrow = length(na.omit(dftrain$air))))
  # dataframe to store the residuals
  colnames(res) <- c("res.t", "location")
```

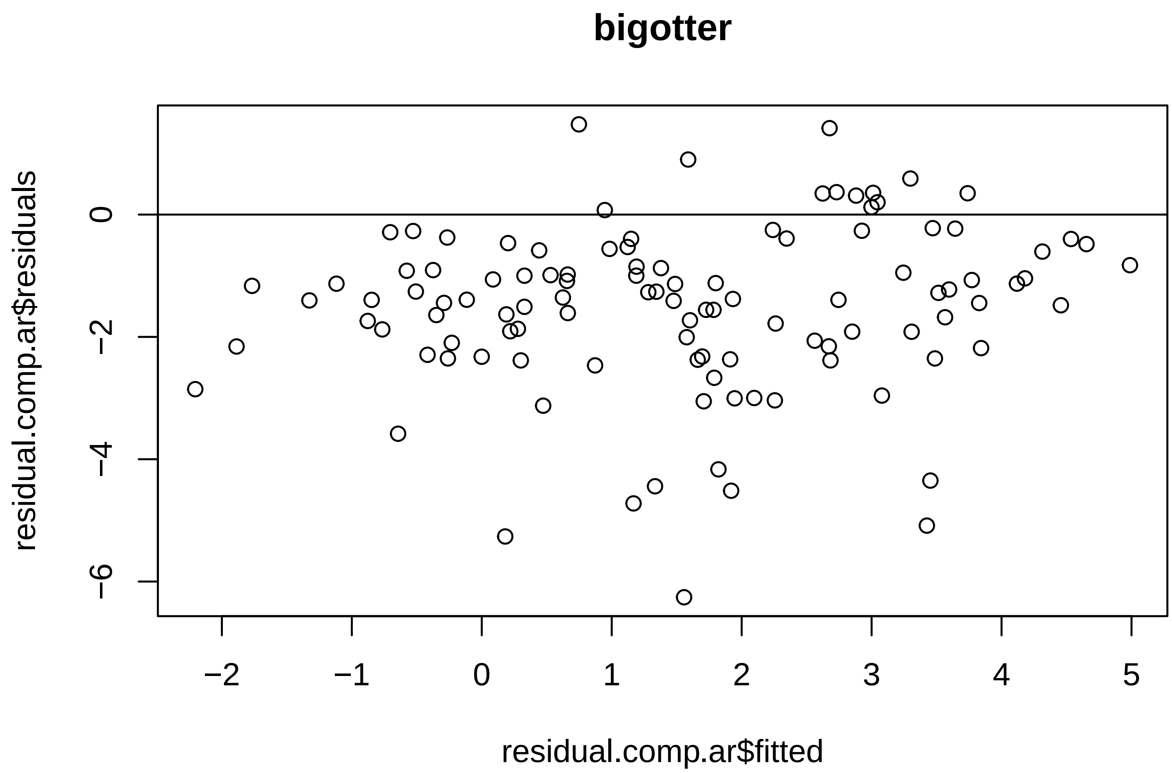
```
res[,"location"] <- na.omit(dftrain$location)
res[,"res.t"] <- as.vector(residuals(annual.comp))
res <- res %>% group_by(location) %>%
  mutate(res.t1 = lag(res.t, 1),
         res.t2 = lag(res.t, 2))
res[,"res.w"] <- residuals(nls(water ~ a+b*sin(2*pi/365*(yday(date)+t0)),
                               start = list(a=0.05, b=5, t0=-26),
                               data = dftrain))

# get the water temperature residual component
residual.comp.ar <- gls(res.w ~ res.t + res.t1 + res.t2,
                        correlation = corAR1(),
                        data = res, na.action = na.omit,
                        control = ctrl)

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

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

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

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

compare <- merge(dftest, p, by=c("location","date"))
compare$week <- ceiling(as.numeric(format(compare$date, "%j"))/7)

# 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)

complist[[loc]][[2]] <- compare.w
```

```
  # store in results dataframes
  rmse.r[loc,2]=round(sqrt(mean((compare.w$preds.ar-compare.w$water)^2)),2)
  bias.r[loc,2]=round(mean(compare.w$preds.ar-compare.w$water),2)
  nsc.r[loc,2]=round(1-sum((compare.w$water-compare.w$preds.ar)^2) / sum((compare.w$water-mean(compare.w


  ## assumptions
  plot(residual.comp.ar$residuals~residual.comp.ar$fitted, main=loc)
  abline(h=0)


  ## prediction plots
  pl <- ggplot(data=compare.w, aes(x=week))+
          geom_line(aes(x=week, y=water), color = "black")+
          geom_line(aes(x=week, y=preds.ar), color = "blue")+
          ggtitle(paste(
            loc, ": (RMSE:", rmse.r[loc,2], "&", "bias:",
            bias.r[loc,2], ")"))
  print(pl)


  ## get coefs
  dfcoef <- rbind(dftrain[,1:17], dftest[,1:17])
  annual.air <- nls(air ~ a+b*sin(2*pi/365*(yday(date)+t0)),
                    start = list(a=0.05, b=5, t0=-26),
                    data=dfcoef)
  annual.water <- nls(water ~ a+b*sin(2*pi/365*(yday(date)+t0)),
                      start = list(a=0.05, b=5, t0=-26),
                      data=dfcoef)
  # get the air temperature residuals
  res <- as.data.frame(matrix(NA, ncol = 2,
                              nrow = length(na.omit(dfcoef$air))))
  # dataframe to store the residuals
  colnames(res) <- c("res.t", "location")
  res[,"location"] <- na.omit(dfcoef$location)
  res[,"res.t"] <- as.vector(residuals(annual.air))
  res <- res %>% group_by(location) %>%
    mutate(res.t1 = lag(res.t, 1),
           res.t2 = lag(res.t, 2))
  res[,"res.w"] <- residuals(annual.water)
  # get the water temperature residual component
  residual.comp.ar <- gls(res.w ~ res.t + res.t1 + res.t2,
                          correlation = corAR1(),
                          data = res, na.action = na.omit,
                          control = ctrl)

  for (c in 1:3){
    coef.season[loc,c]=signif(coef(annual.air)[c],2)
    coef.season[loc,c+3]=signif(coef(annual.water)[c],2)
    coef.season[loc,c+6]=signif(coef(residual.comp.ar)[c+1],2)
  }
}
```

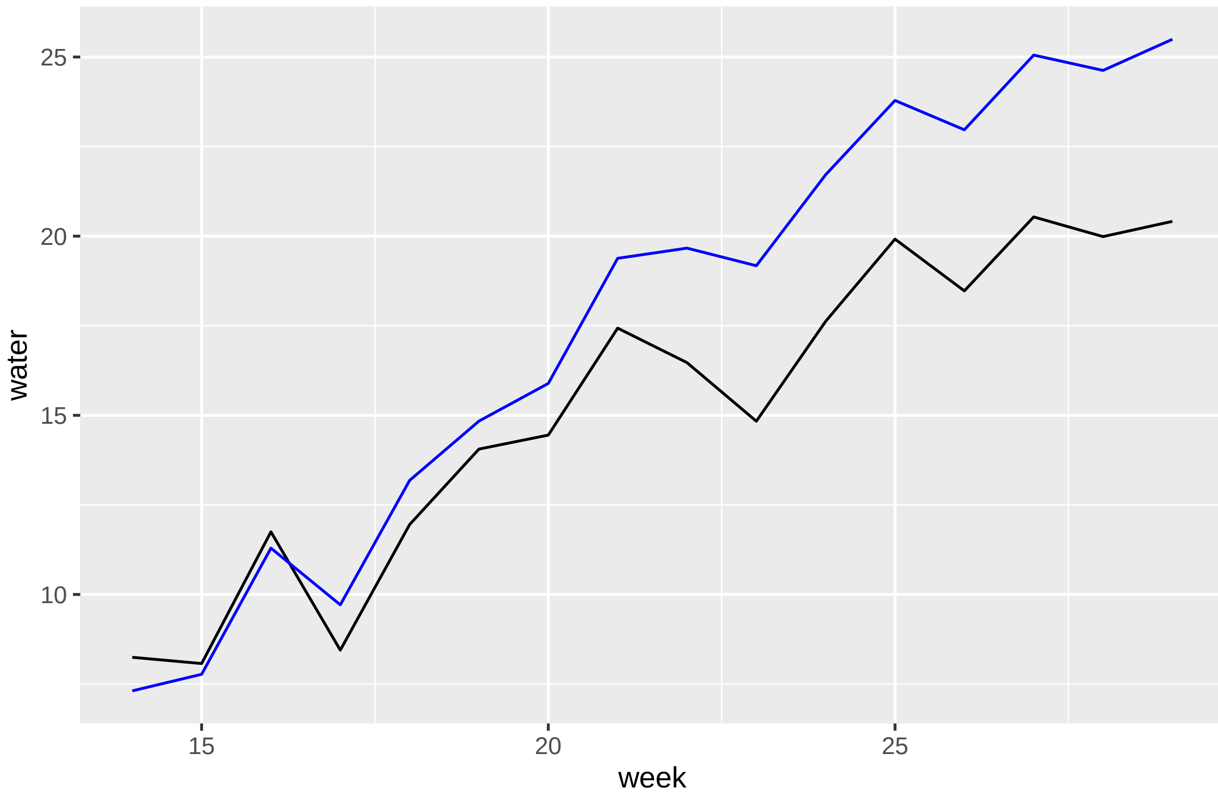


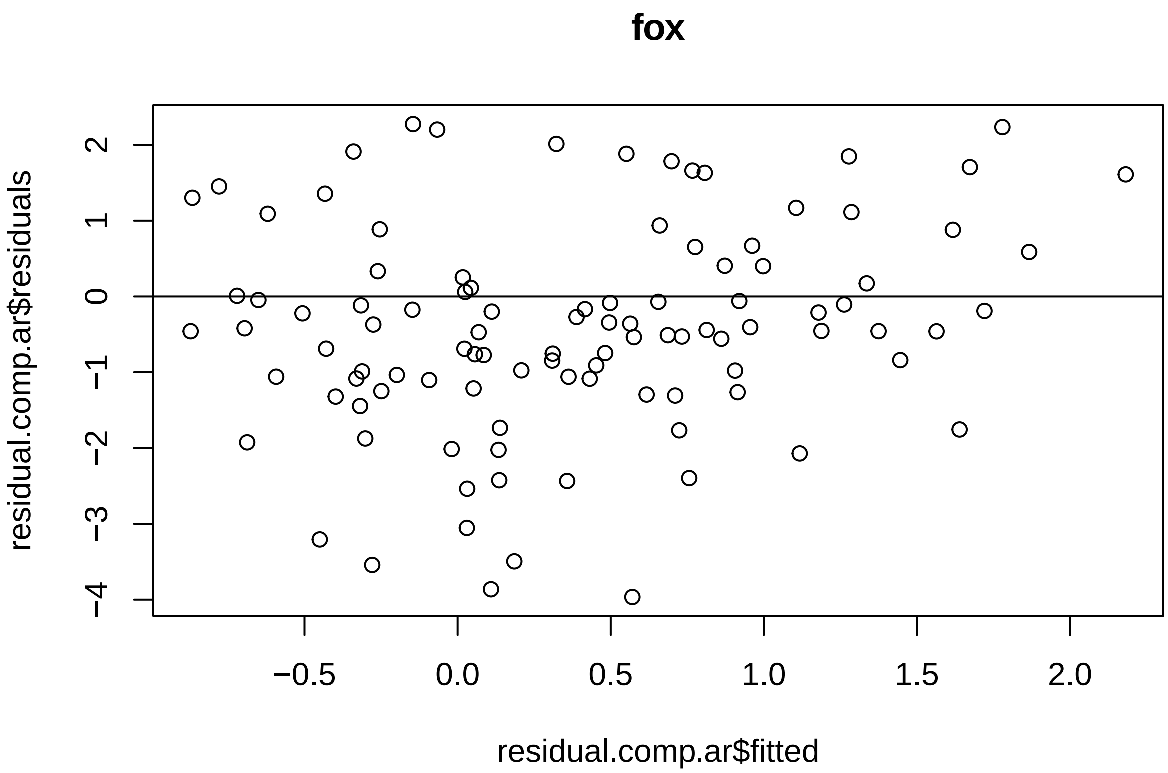
bigcreek
residual.comp.ar$residuals
1.5
1.0
0.5
0.0
−1.0
−3
−2
−1
0
1
2
3
residual.comp.ar$fitted

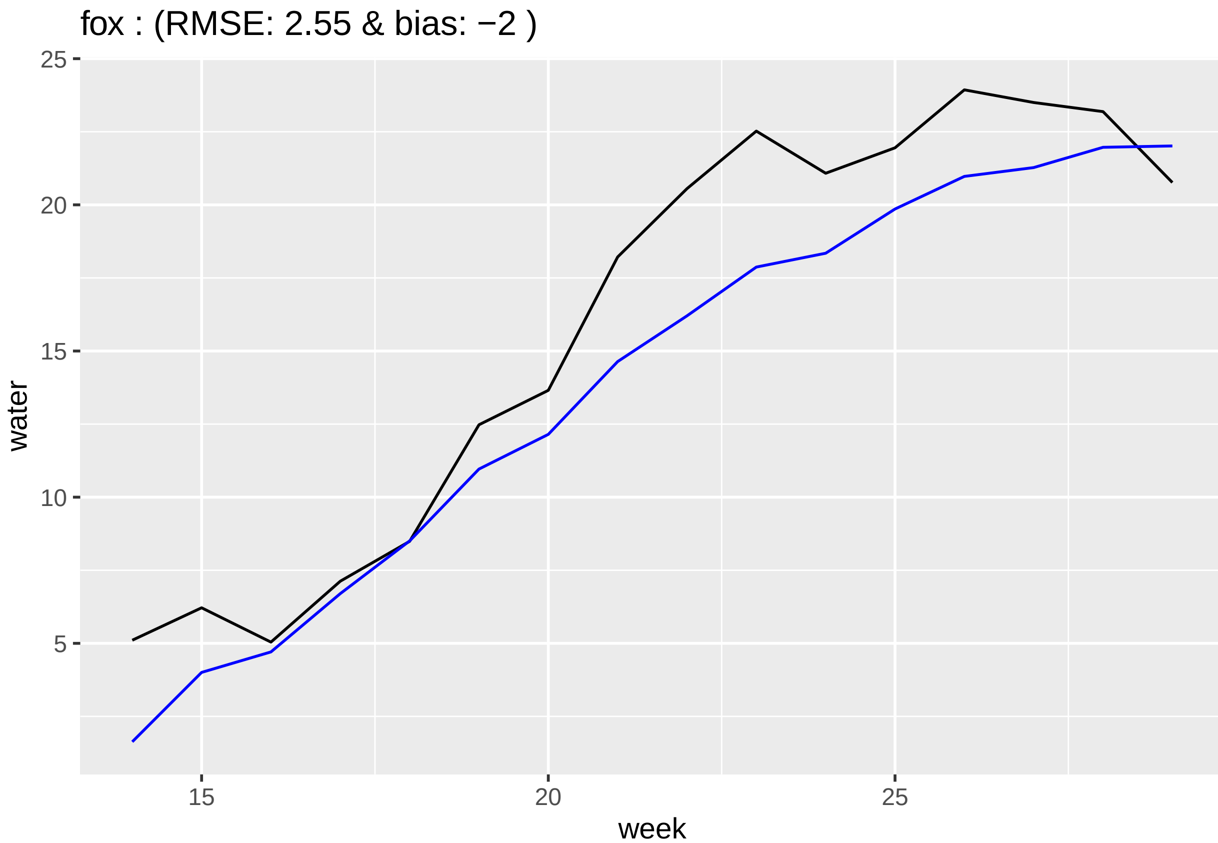

bigcreek : (RMSE: 1.99 & bias: 1.3)
water
20
15
10
5
15
20
25
week

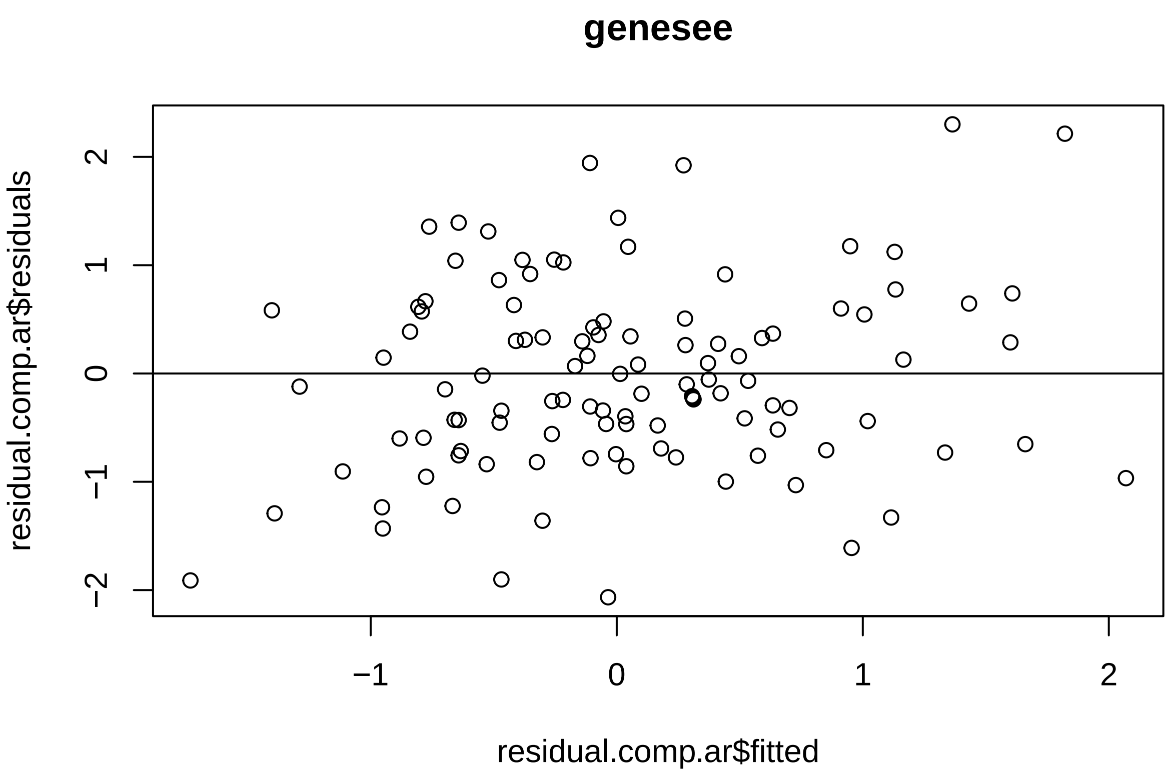

bigotter
residual.comp.ar$residuals
residual.comp.ar$fitted
0
−2
−4
−6
−2
−1
0
1
2
3
4
5

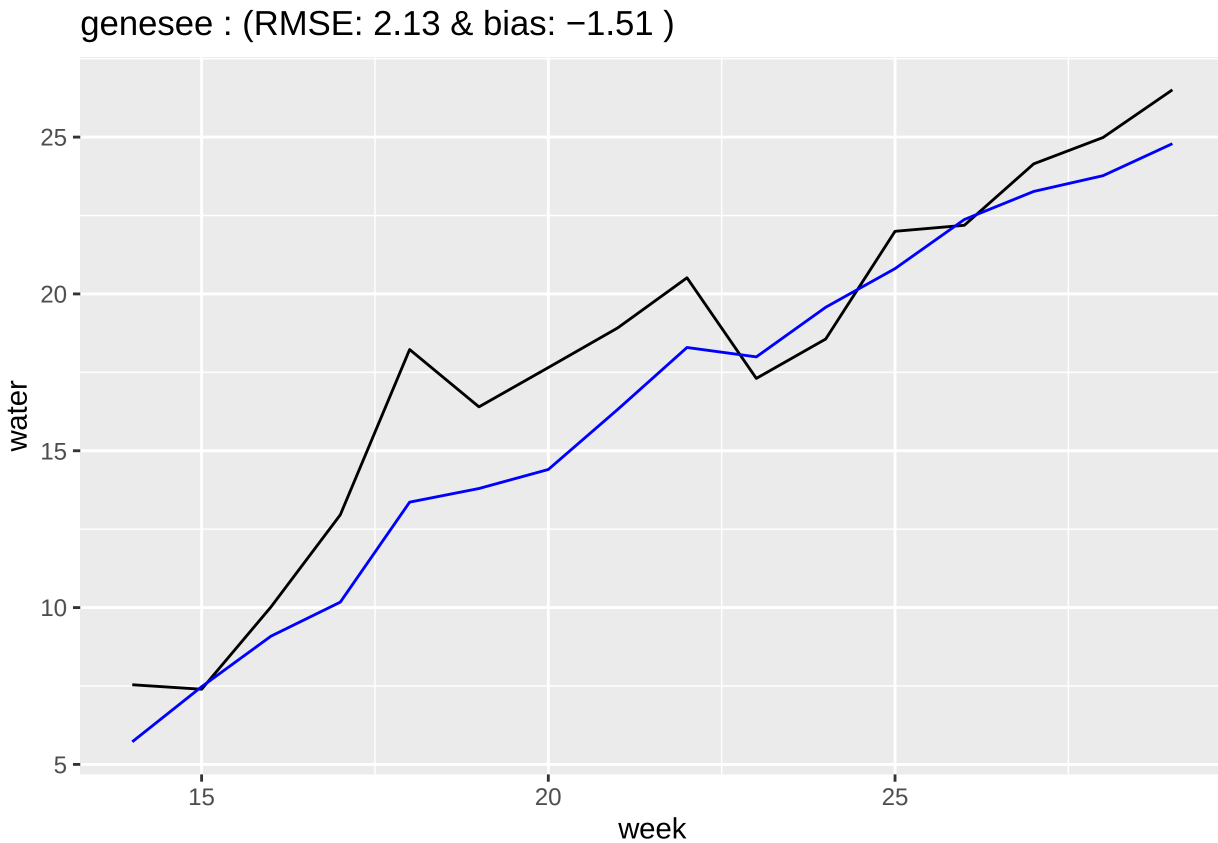
bigotter : (RMSE: 3.16 & bias: 2.45)

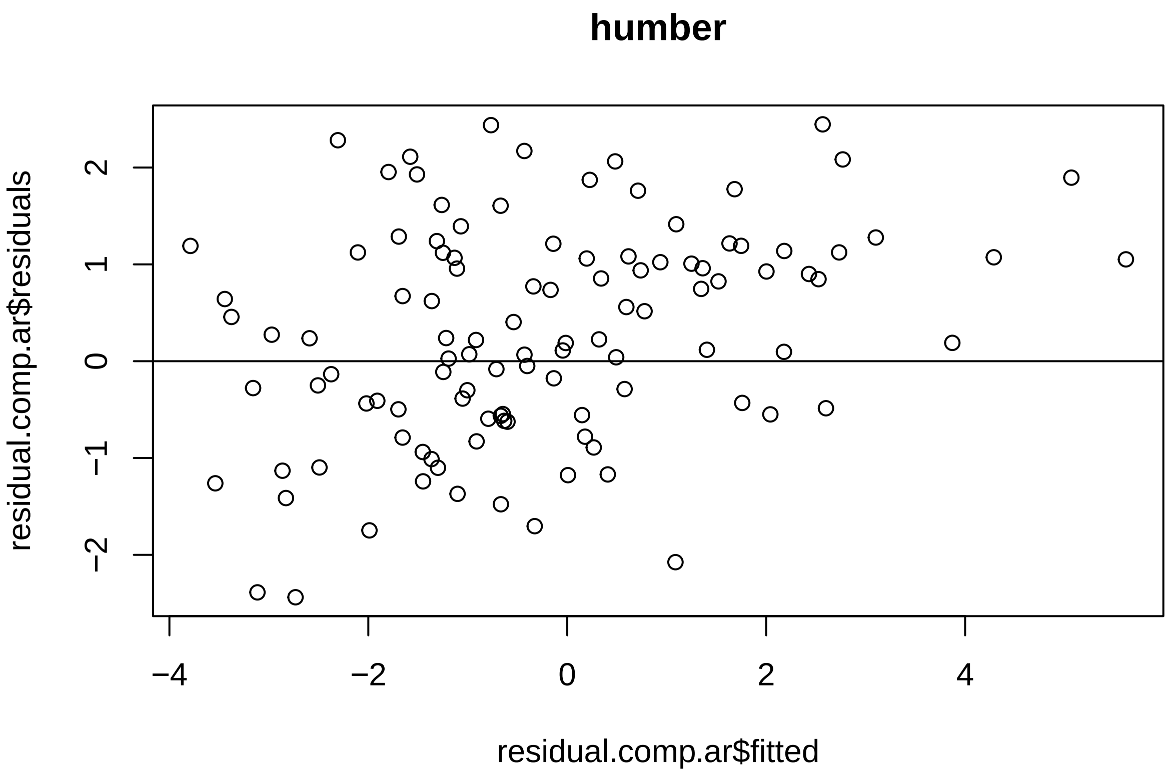


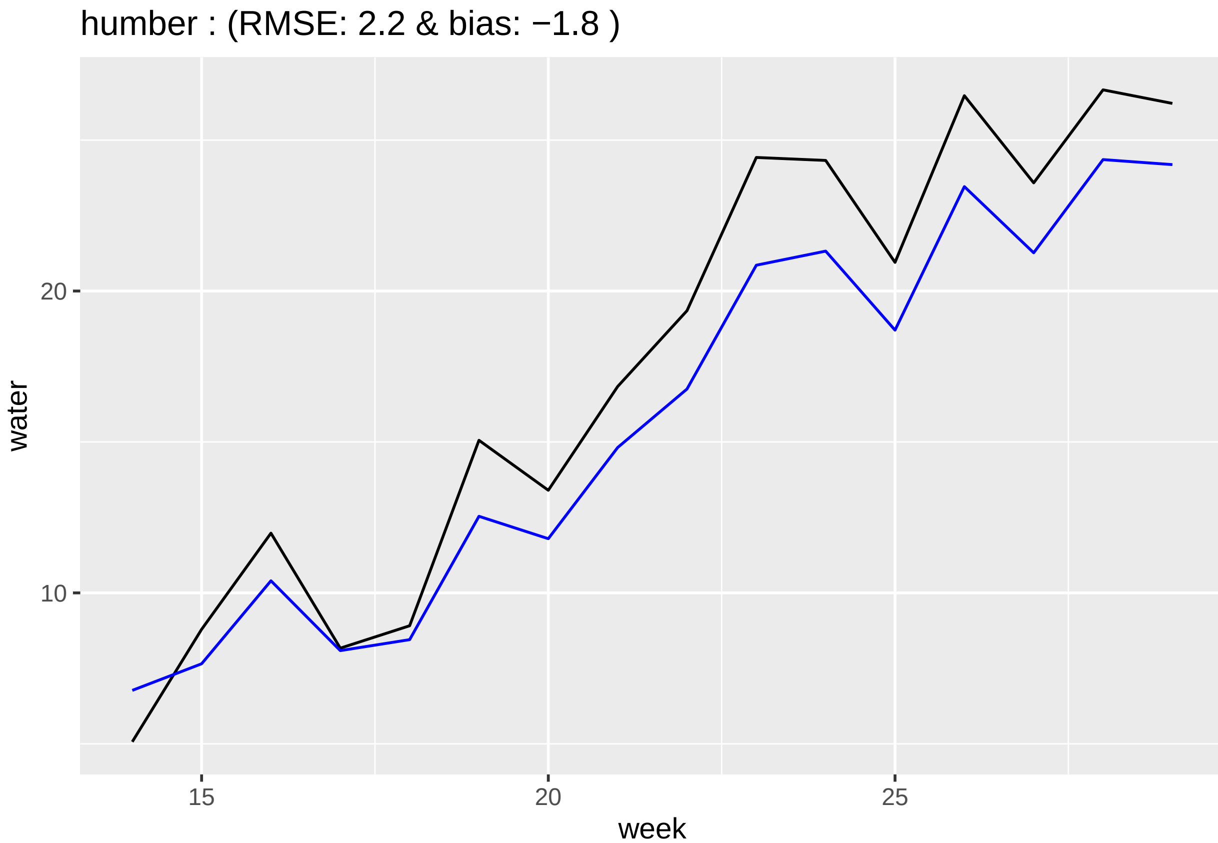

fox
residual.comp.ar$residuals
residual.comp.ar$fitted
2
1
0
−1
−2
−3
−4
−0.5
0.0
0.5
1.0
1.5
2.0

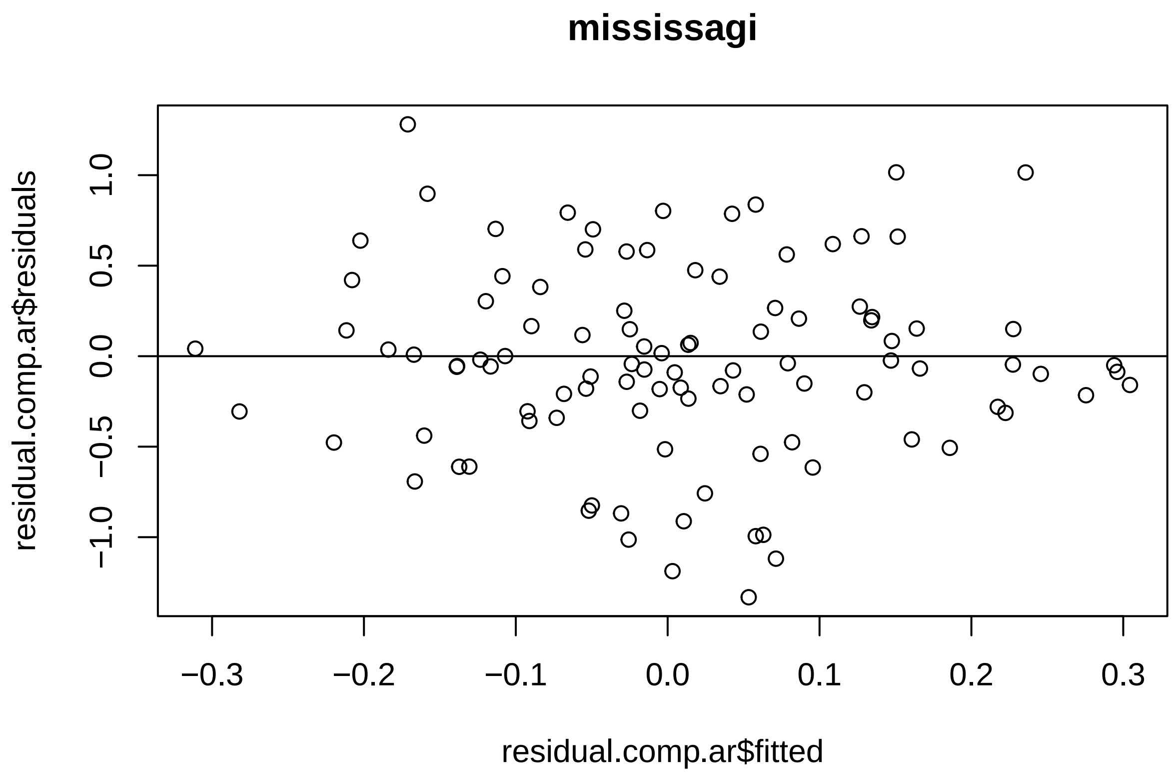

fox : (RMSE: 2.55 & bias: −2)
25
20
15
10
5
water
15
20
25
week

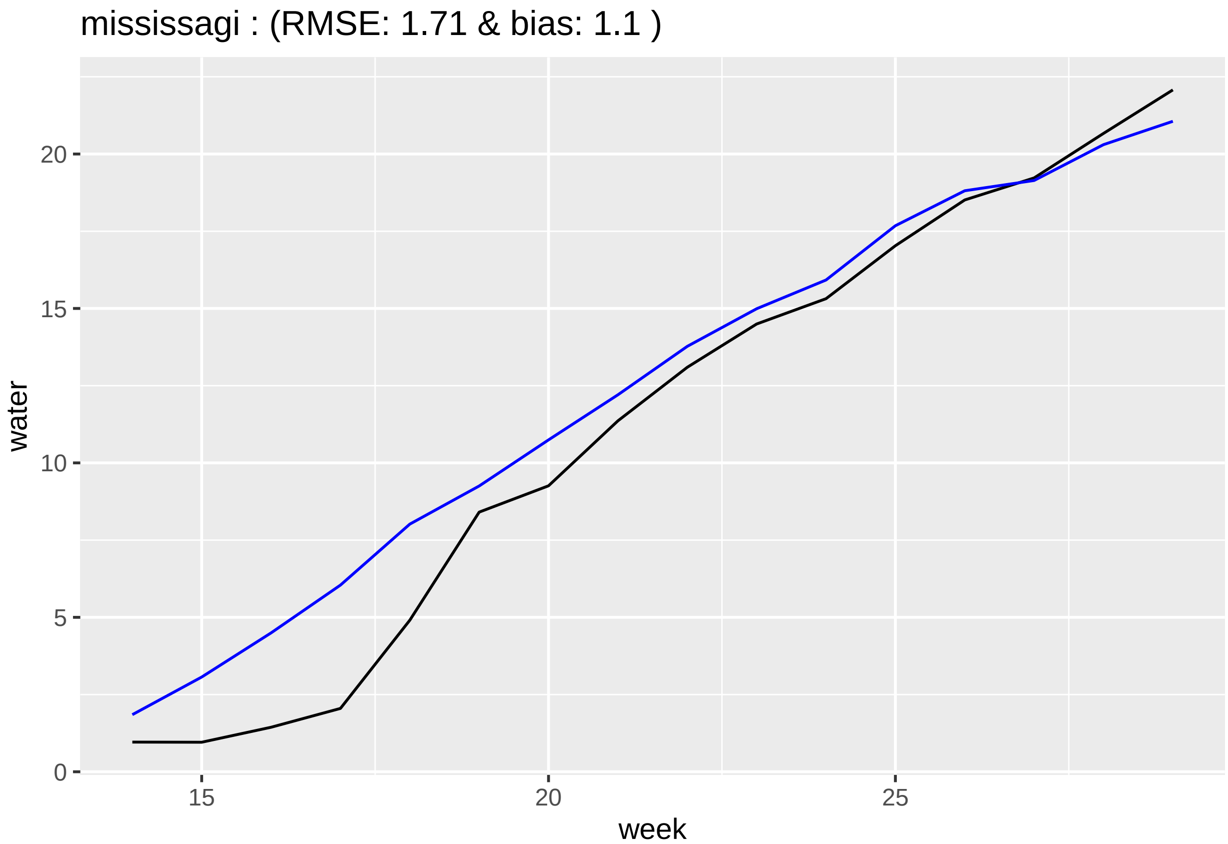

genesee
residual.comp.ar$residuals
residual.comp.ar$fitted
2
1
0
−1
−2
−1
0
1
2

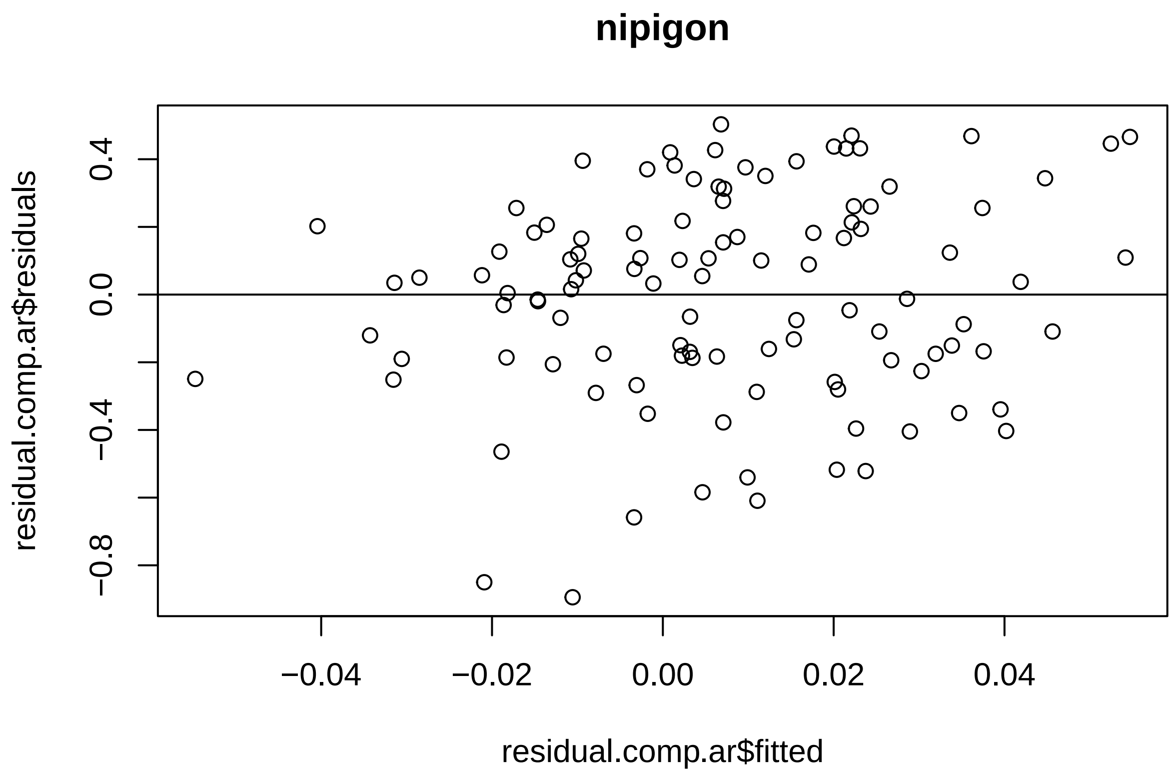

genesee : (RMSE: 2.13 & bias: −1.51)
water
25
20
15
10
5
15
20
25
week

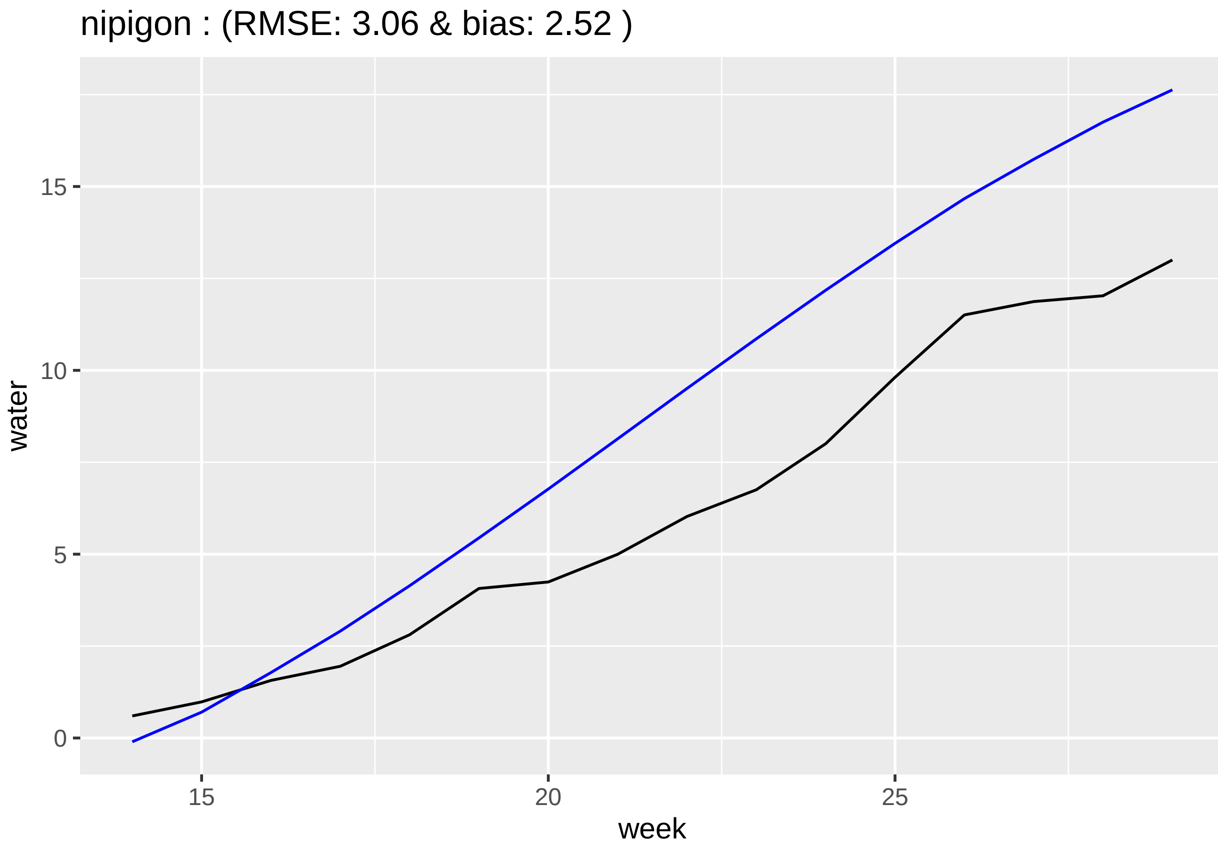

humber
residual.comp.ar$residuals
residual.comp.ar$fitted
−4
−2
0
2
4
−2
−1
0
1
2

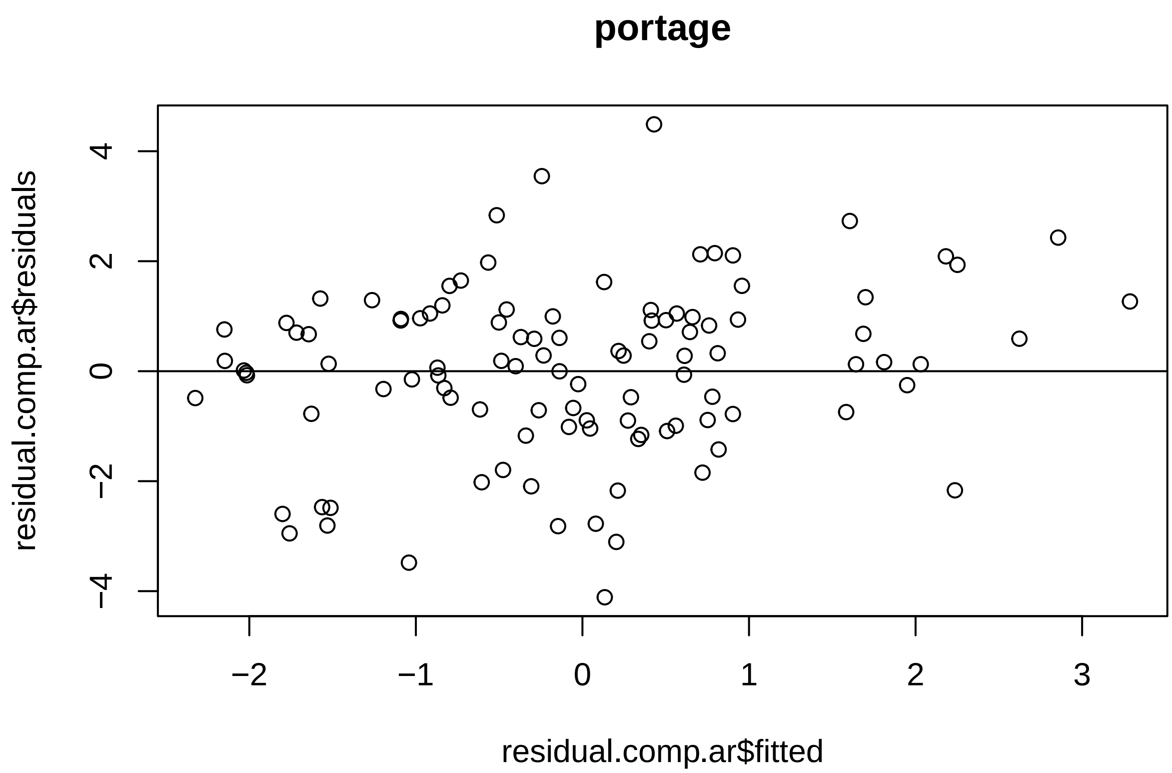

humber : (RMSE: 2.2 & bias: −1.8)
water
20
10
15
20
25
week

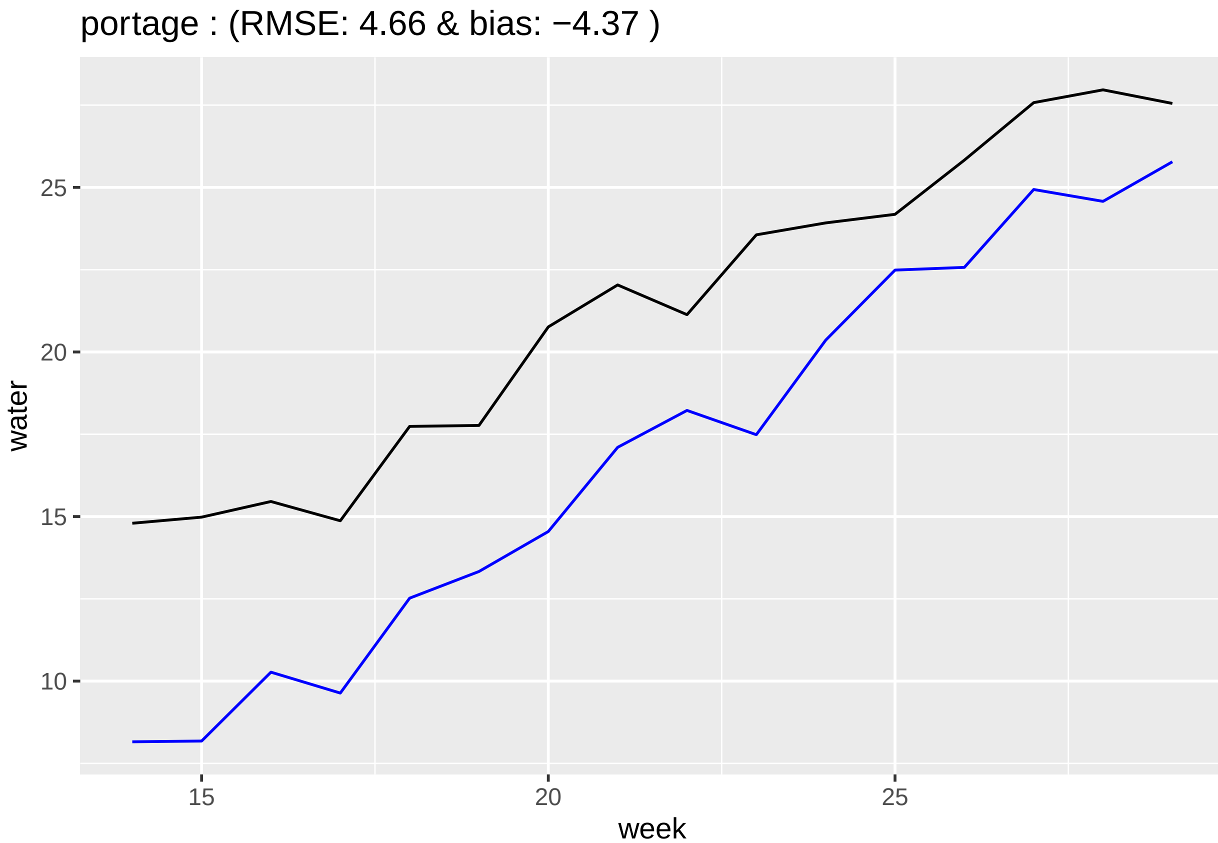

mississagi
residual.comp.ar$residuals
1.0
0.5
0.0
−0.5
−1.0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
residual.comp.ar$fitted

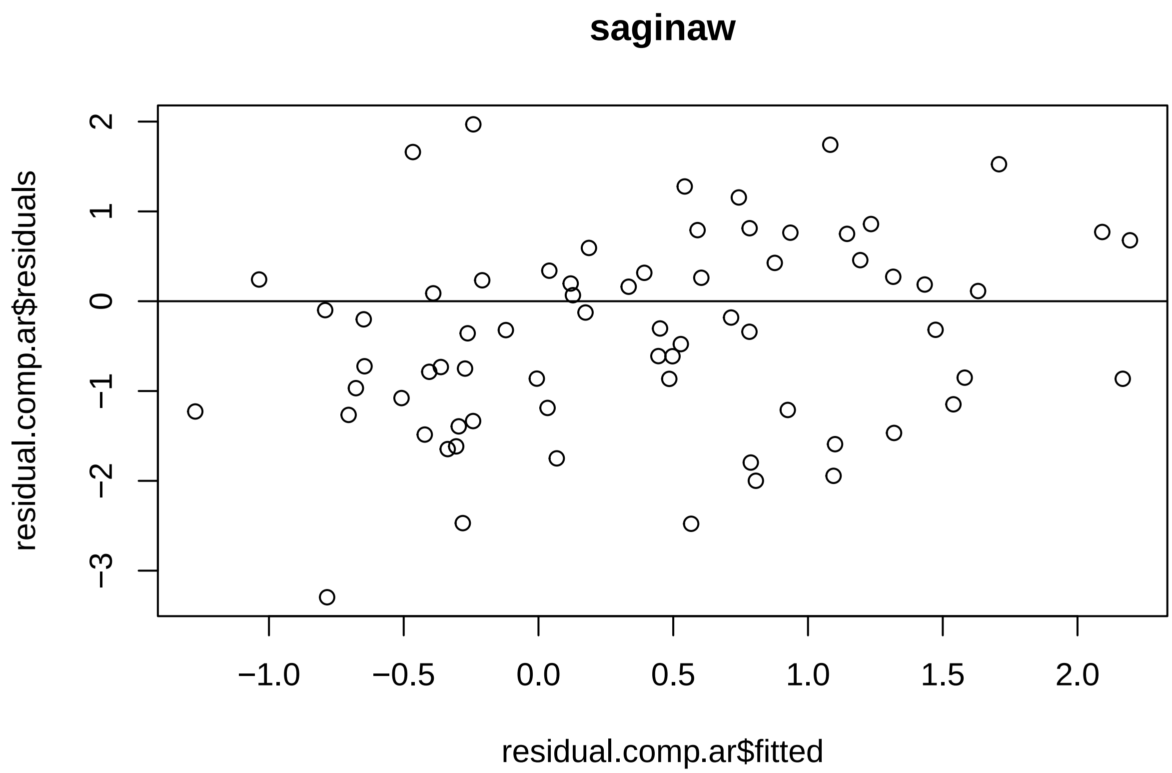

mississagi : (RMSE: 1.71 & bias: 1.1)
water
20
15
10
5
0
15
20
25
week

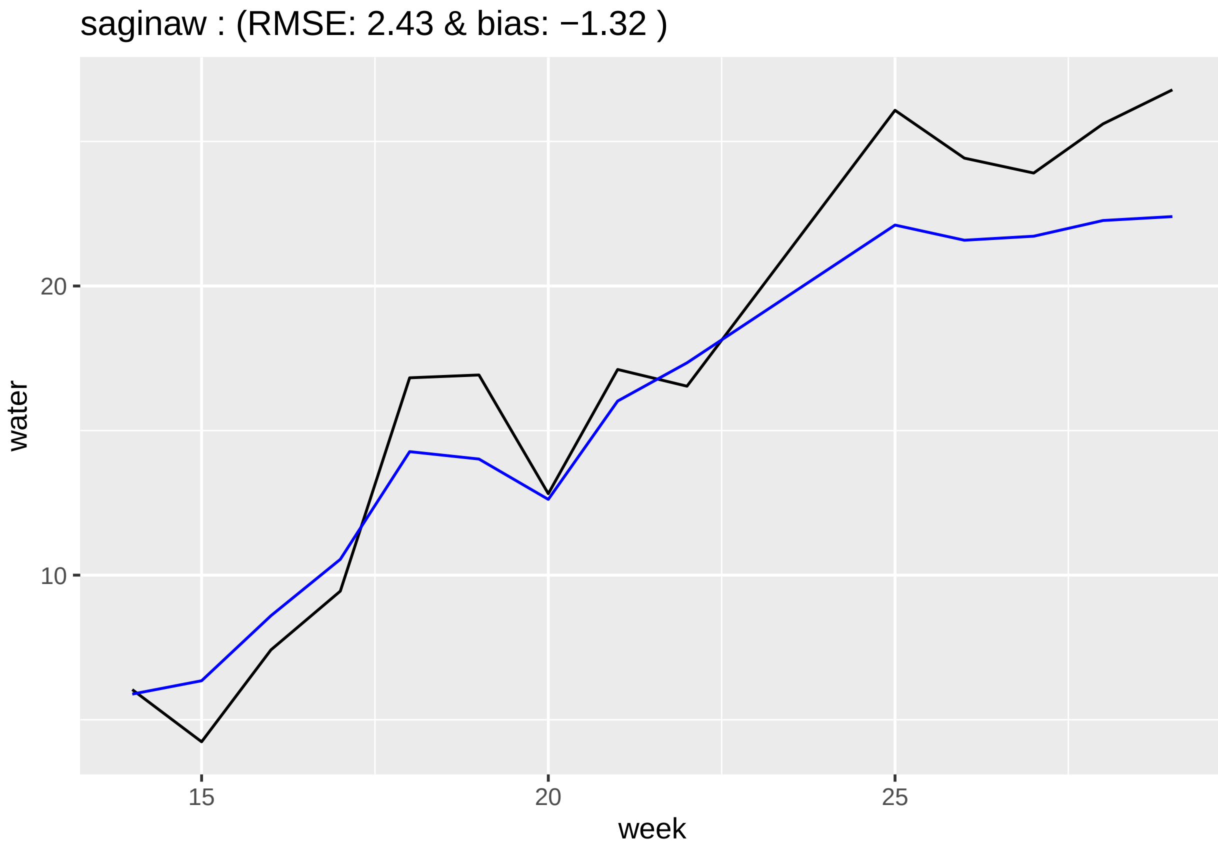

nipigon
residual.comp.ar$residuals
residual.comp.ar$fitted
0.4
0.0
−0.4
−0.8
−0.04
−0.02
0.00
0.02
0.04

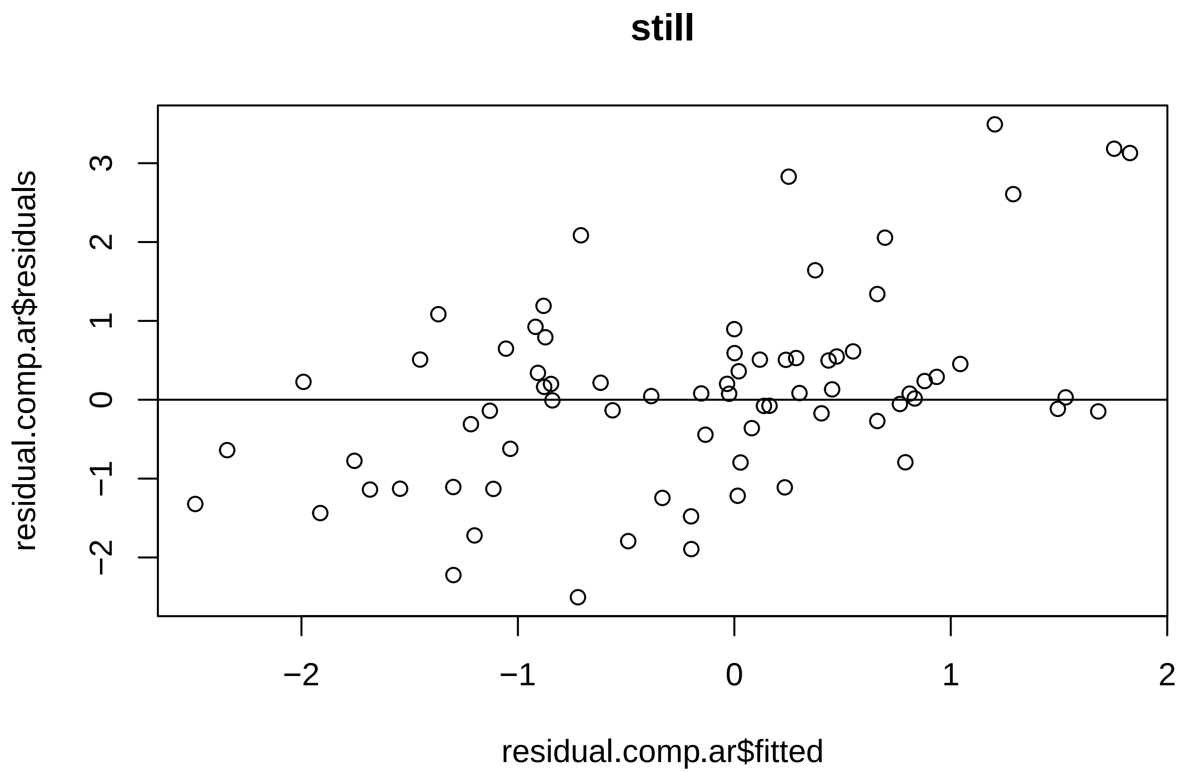

nipigon : (RMSE: 3.06 & bias: 2.52)
water
0
5
10
15
15
20
25
week

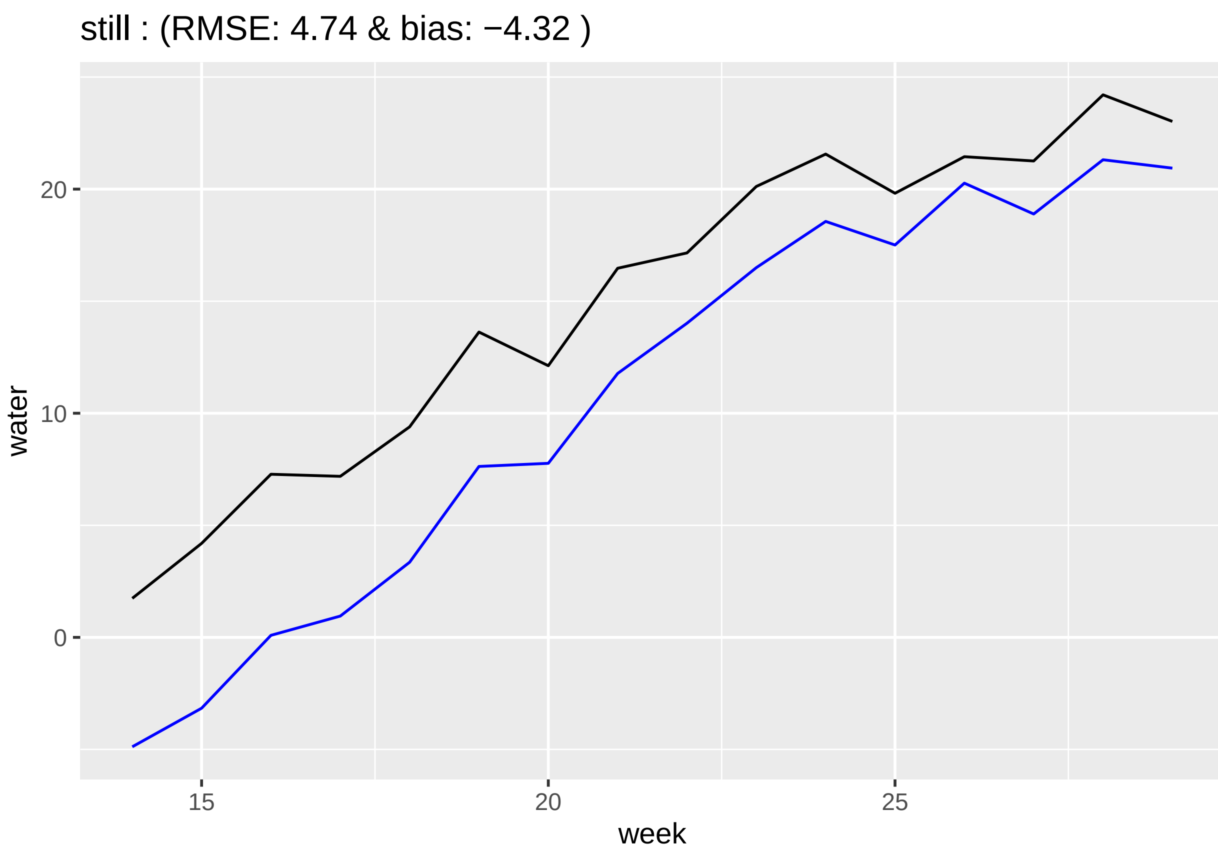

portage
residual.comp.ar$residuals
4
2
0
−2
−4
−2
−1
0
1
2
3
residual.comp.ar$fitted

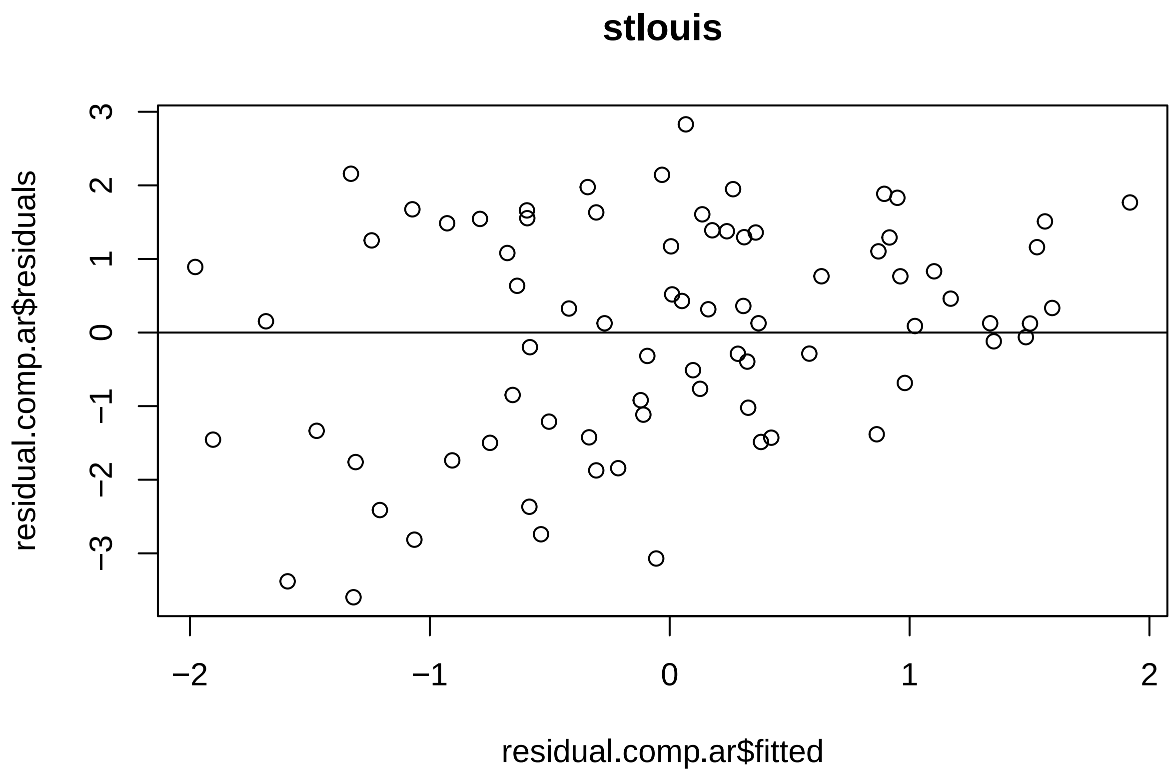

portage : (RMSE: 4.66 & bias: −4.37)
water
25
20
15
10
15
20
25
week

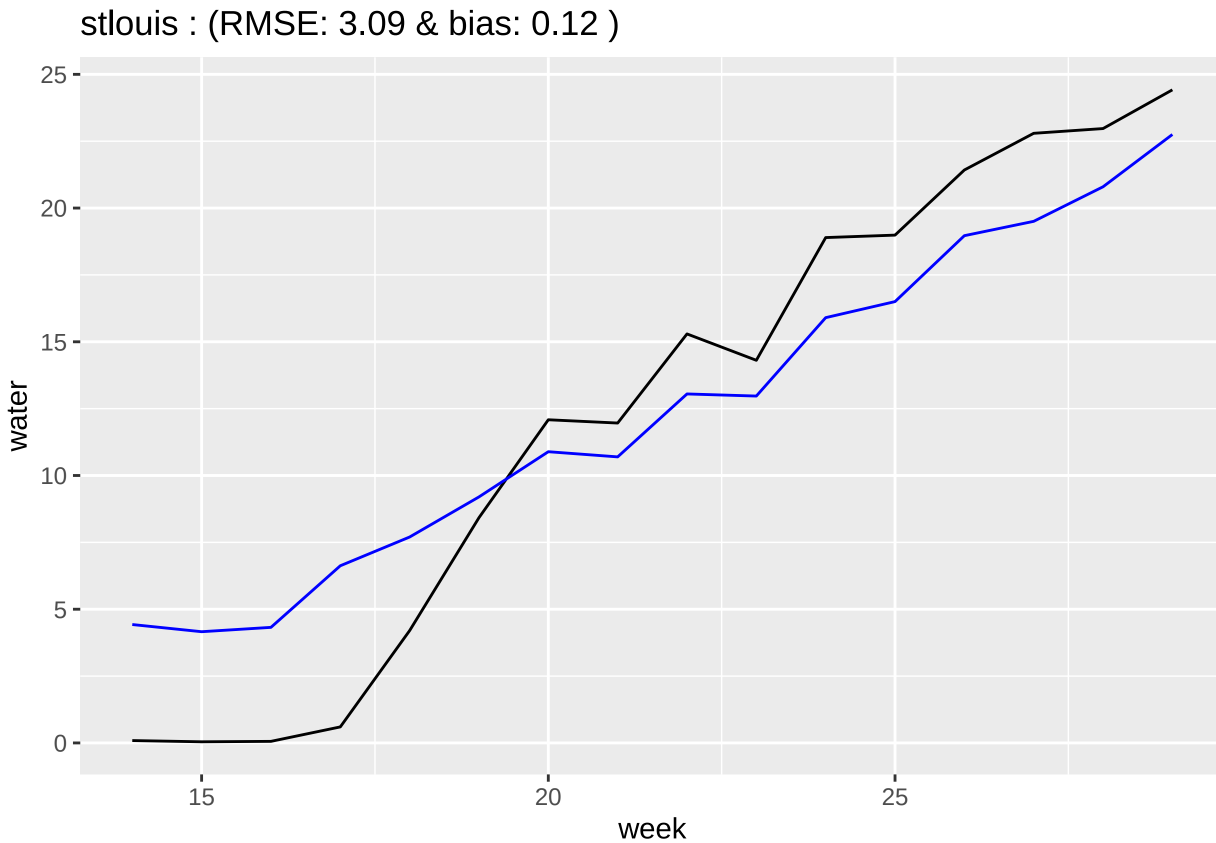

saginaw
residual.comp.ar$residuals
residual.comp.ar$fitted
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
−3
−2
−1
0
1
2


saginaw : (RMSE: 2.43 & bias: −1.32)
water
20
10
15
20
25
week

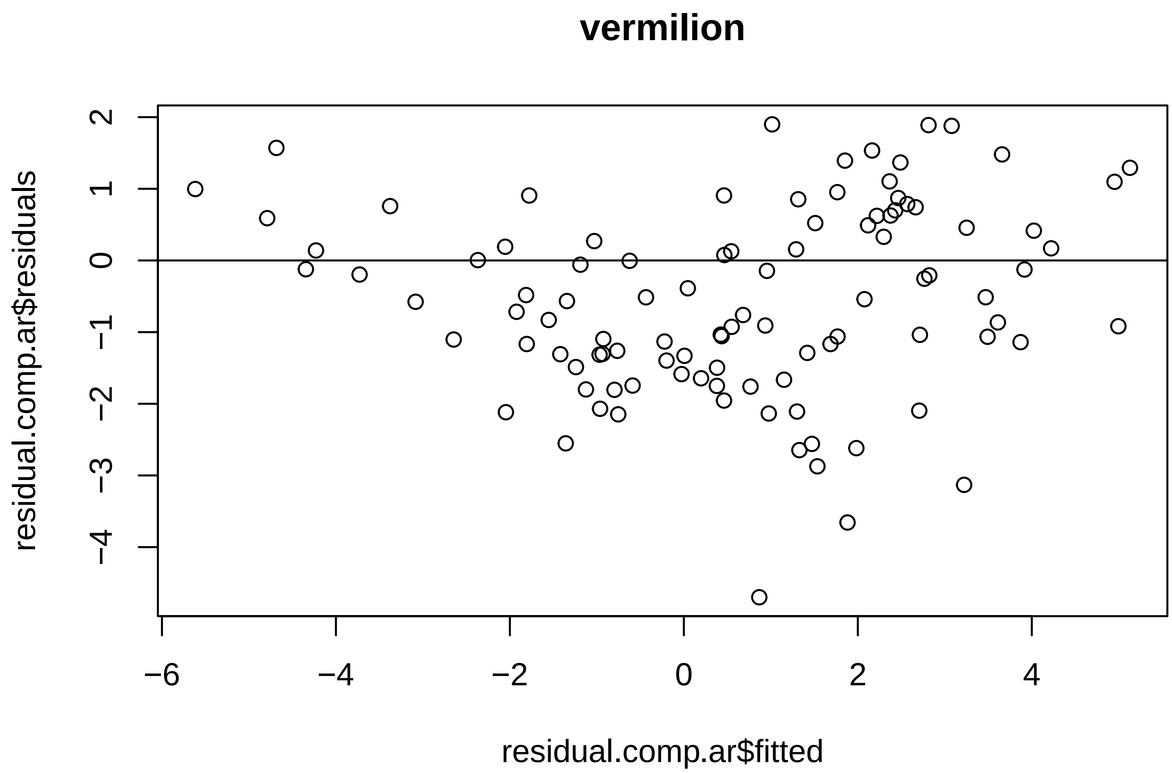

still
residual.comp.ar$residuals
residual.comp.ar$fitted
3
2
1
0
−1
−2
−2
−1
0
1
2

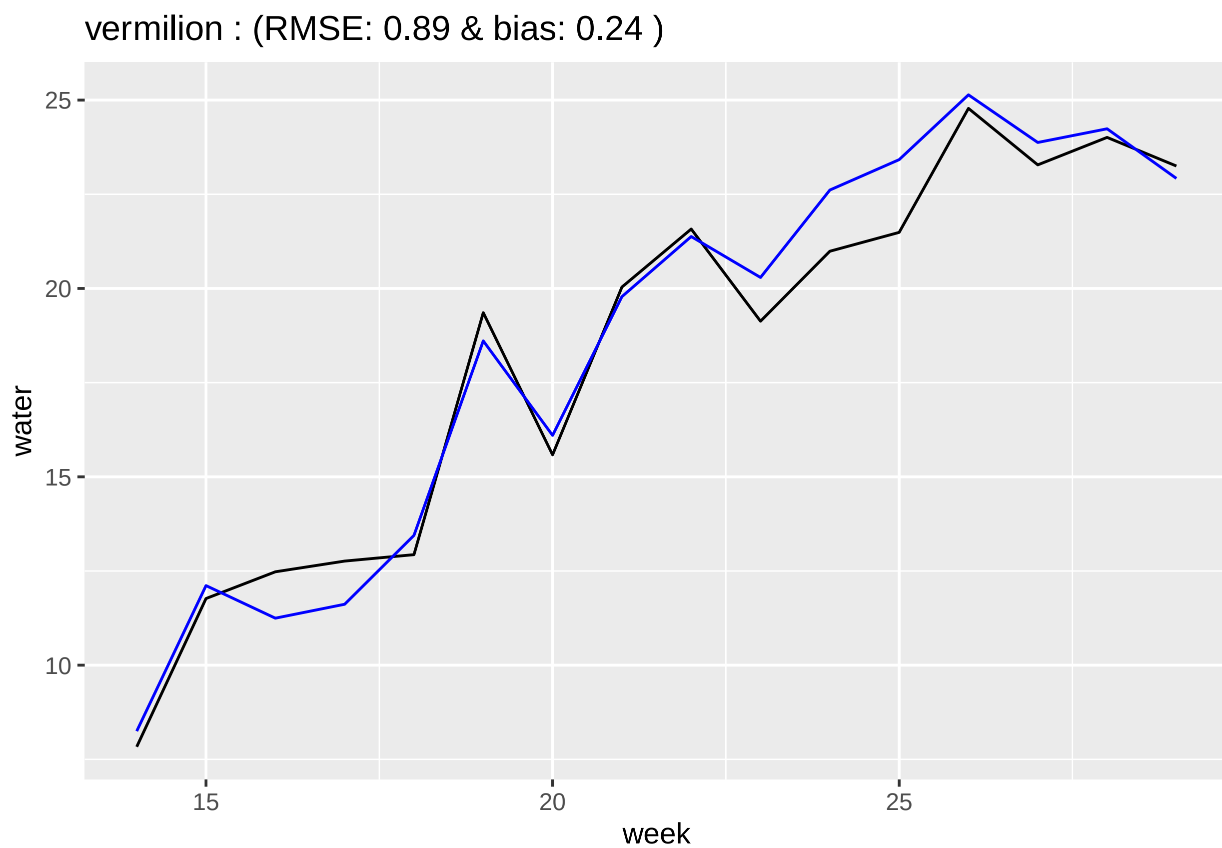

still : (RMSE: 4.74 & bias: −4.32)
water
20
10
0
15
20
25
week


stlouis
residual.comp.ar$residuals
residual.comp.ar$fitted
−2
−1
0
1
2
3



stlouis : (RMSE: 3.09 & bias: 0.12)
water
week
25
20
15
10
5
0
15
20
25

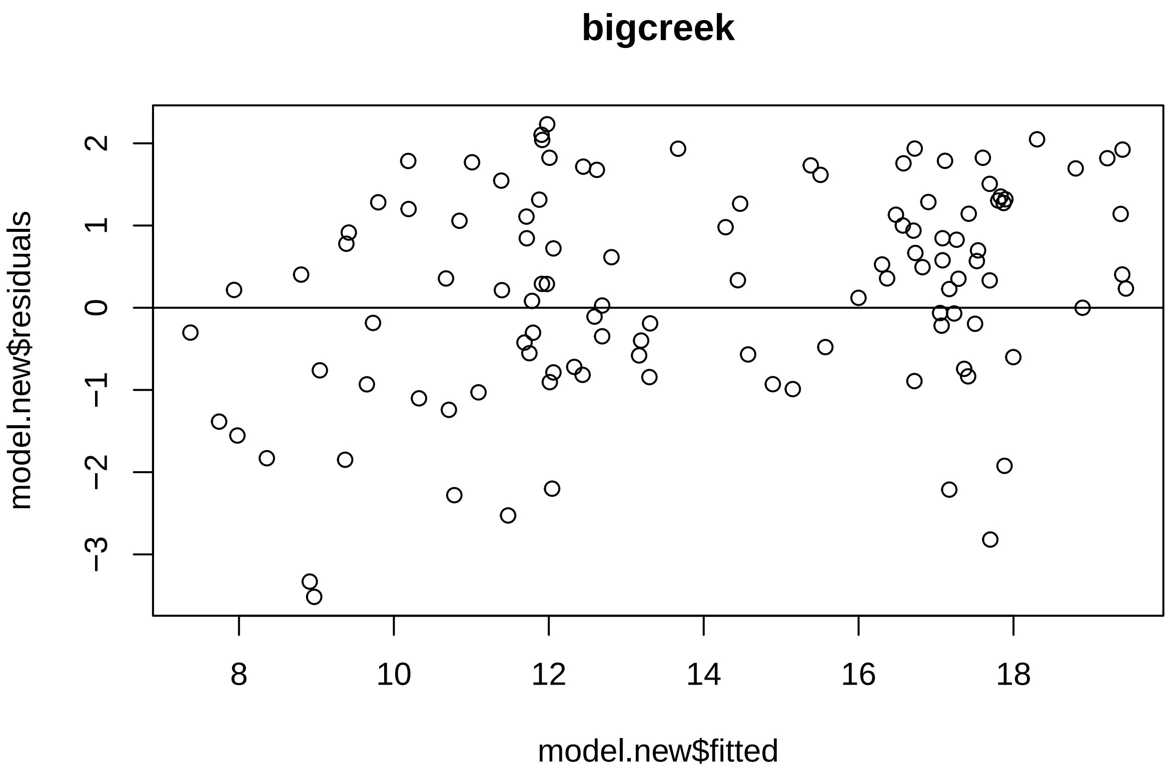

vermilion
residual.comp.ar$residuals
residual.comp.ar$fitted
2
1
0
−1
−2
−3
−4
−6
−4
−2
0
2
4

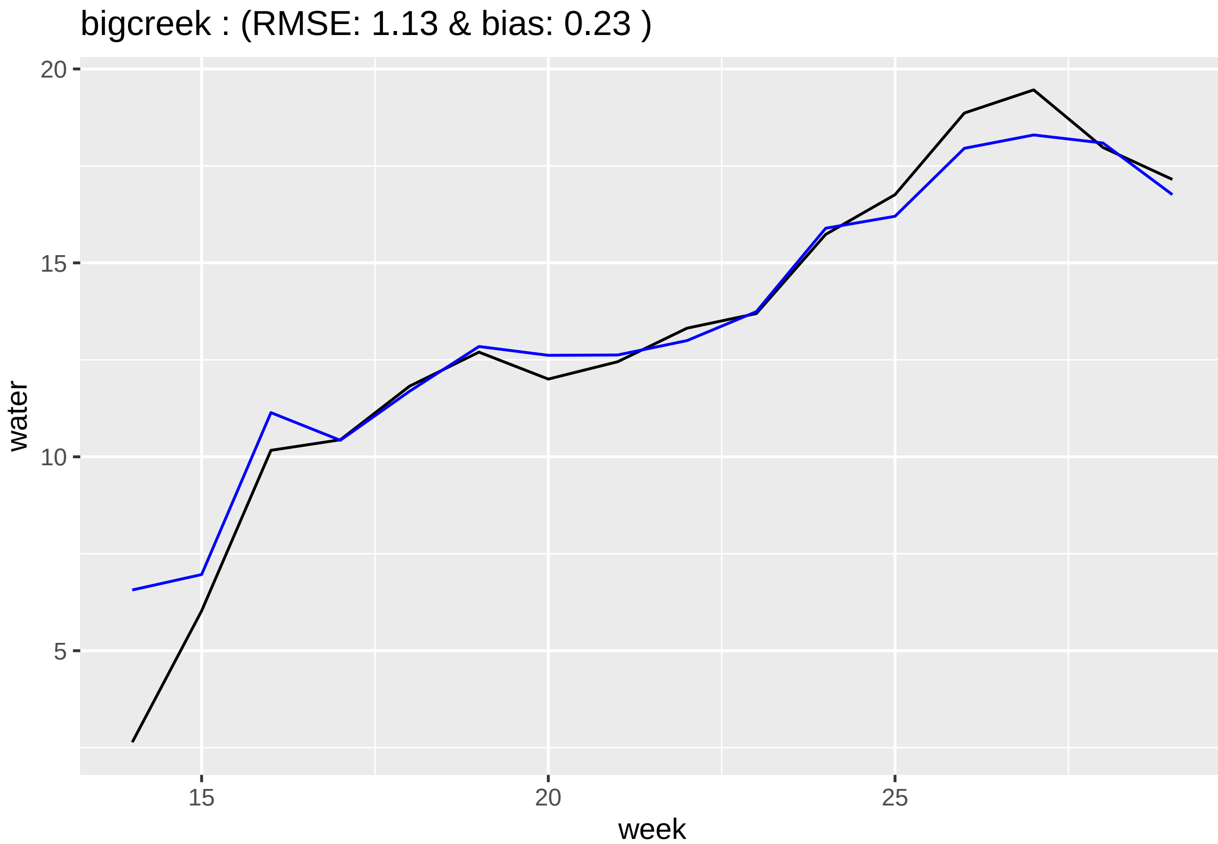


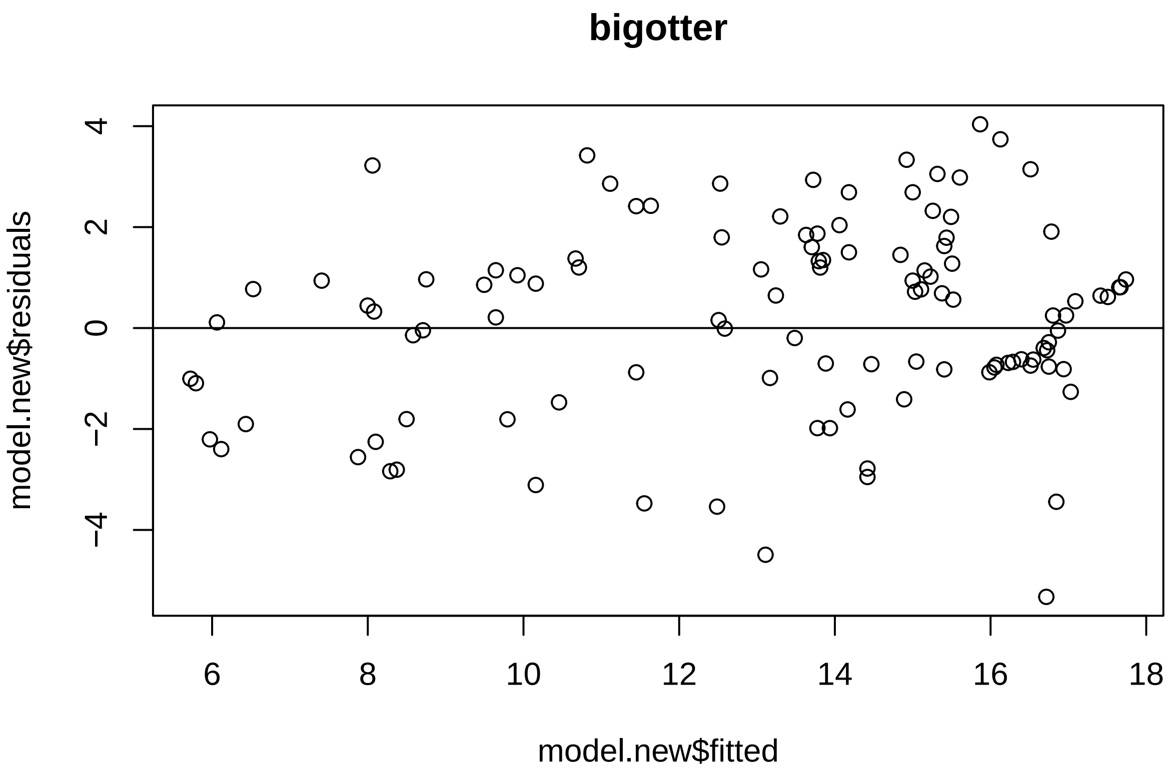
Non-linear

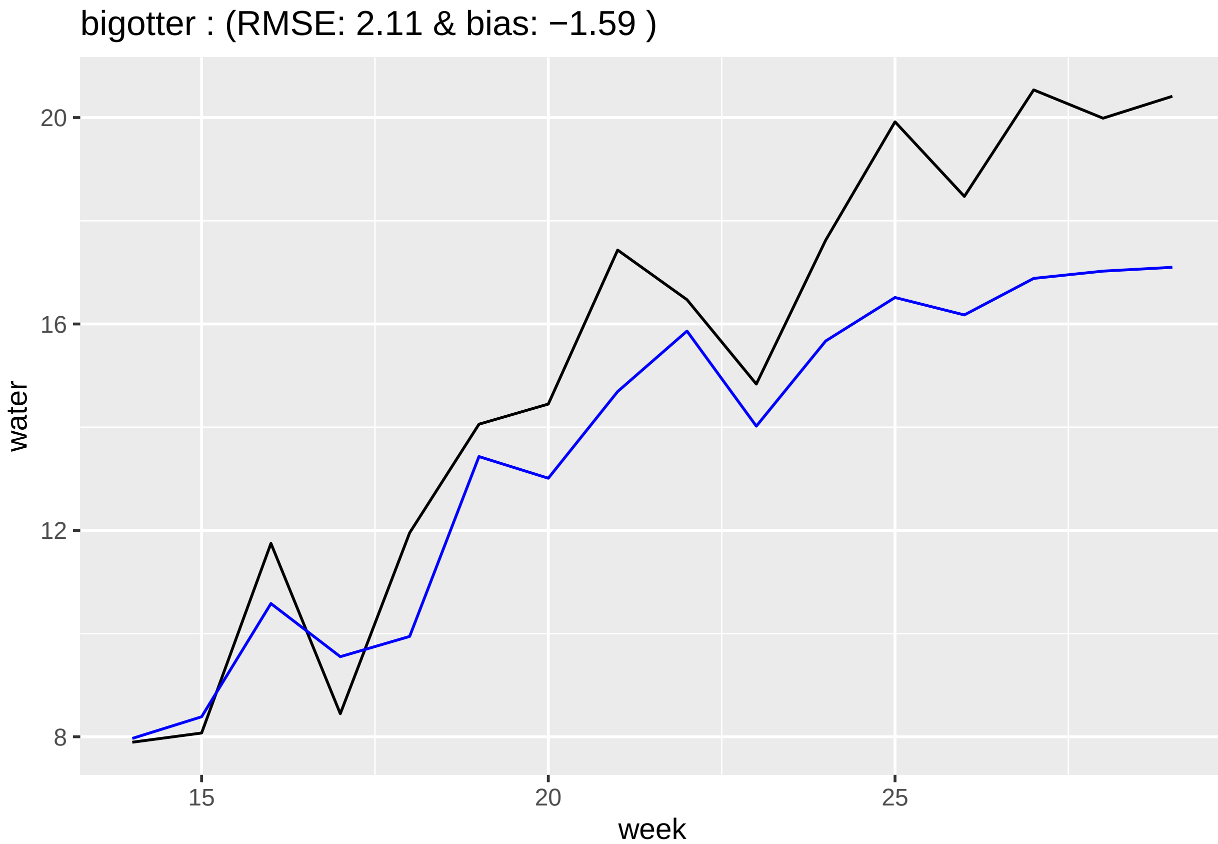
```
## Coefs
coef.nonl <- matrix(NA, ncol=3, nrow=12)
colnames(coef.nonl) <- c("alpha","beta","gamma")
rownames(coef.nonl) <- loc_seq


# For each location
for (loc in loc_seq){

    dftrain = current.training[current.training$location == loc,]
    dftest = current.testing[current.testing$location == loc,]
    compare <- dftest

    # coefs
    modelco <- drm(water ~ air, fct = L.3(), data = dftrain)
    co = c(alpha=as.numeric(coef(modelco)[2]),
           beta=as.numeric(coef(modelco)[3]),
           gamma=as.numeric(-coef(modelco)[1]))


    # model
    model.new <- gnls(water ~ alpha / (1 + exp(gamma * (beta - cair))),
                      data = dftrain, na.action = na.omit,
```

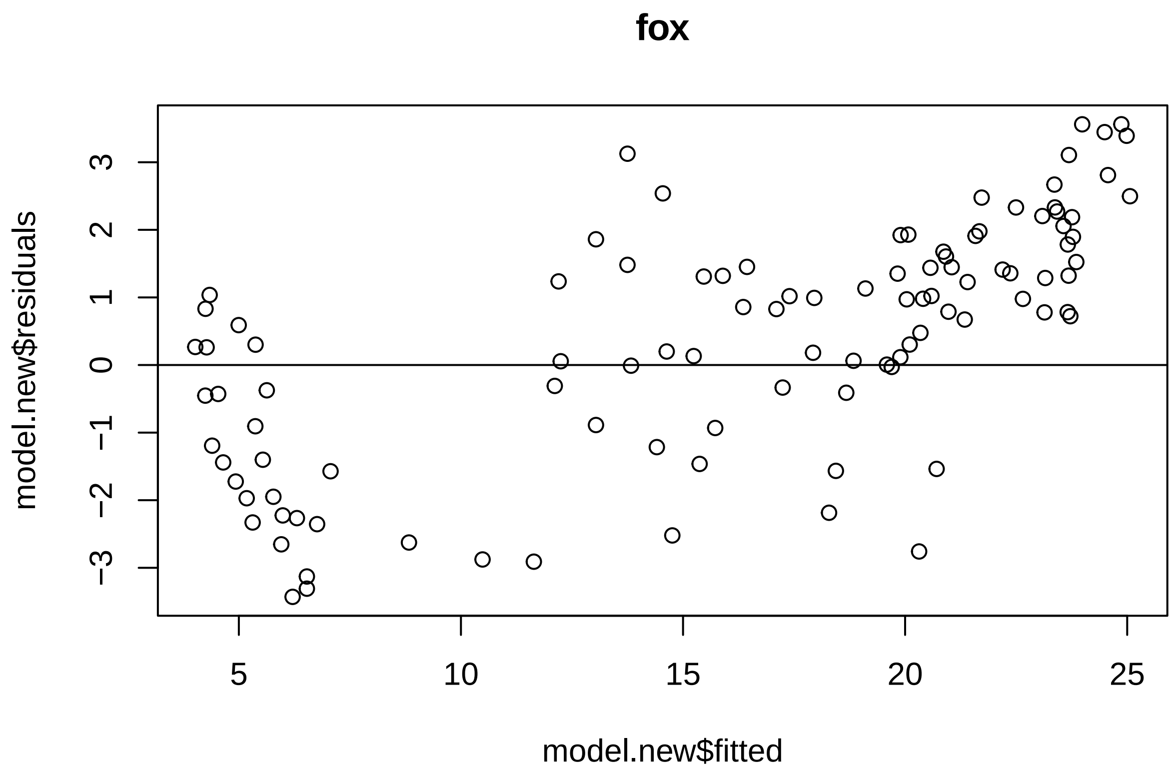
```
                      start = co,
                      correlation = corAR1(),
                      control=gnlsControl(nlsTol=1000, maxIter=1000))


    compare$preds.new <- as.vector(predict(model.new, newdata = dftest))
    compare$week <- ceiling(as.numeric(format(compare$date, "%j"))/7)

    # weekly comparison dataframe
    compare.w <- merge(
      aggregate(preds.new~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)

    complist[[loc]][[3]] <- compare.w

  # store in results dataframes
  rmse.r[loc,3]=round(sqrt(mean((compare.w$preds.new-compare.w$water)^2)),2)
  bias.r[loc,3]=round(mean(compare.w$preds.new-compare.w$water),2)
  nsc.r[loc,3]=round(1-sum((compare.w$water-compare.w$preds.new)^2) / sum((compare.w$water-mean(compare


  ## assumptions
  plot(model.new$residuals~model.new$fitted, main=loc)
  abline(h=0)


  ## prediction plots
  pl <- ggplot(data=compare.w, aes(x=week))+
          geom_line(aes(x=week, y=water), color = "black")+
          geom_line(aes(x=week, y=preds.new), color = "blue")+
          ggtitle(paste(
            loc, ": (RMSE:", rmse.r[loc,3], "&", "bias:",
            bias.r[loc,3], ")"))
  print(pl)


  ## get coefs
  dfcoef <- rbind(dftrain[,1:17], dftest[,1:17])
  model.new <- gnls(water ~ alpha / (1 + exp(gamma * (beta - cair))),
                    data = dfcoef, na.action = na.omit,
                    start = co, correlation = corAR1(),
                    control=gnlsControl(nlsTol=1000, maxIter=1000))
  for (c in 1:3){coef.nonl[loc,c]=signif(coef(model.new)[c],2)}
}
```

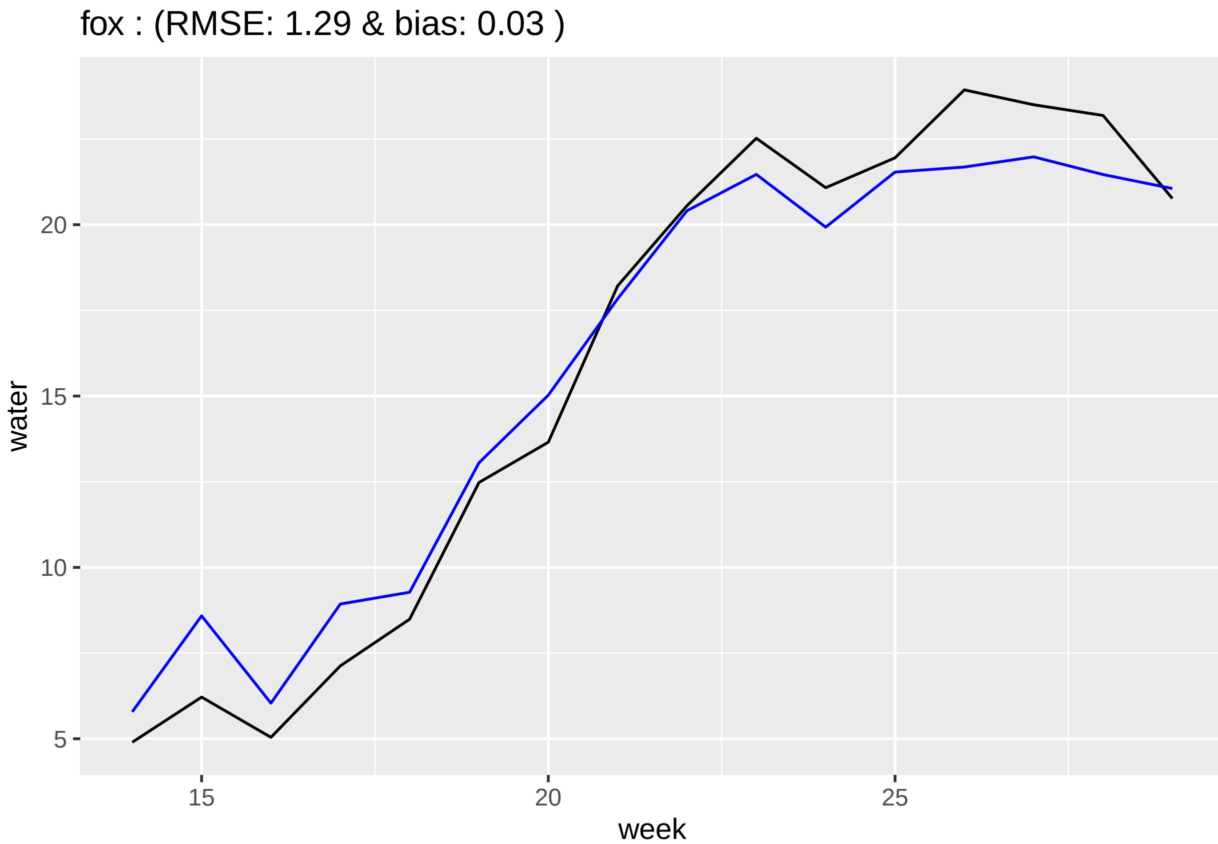

bigcreek
model.new$residuals
2
1
0
−1
−2
−3
8
10
12
14
16
18
model.new$fitted

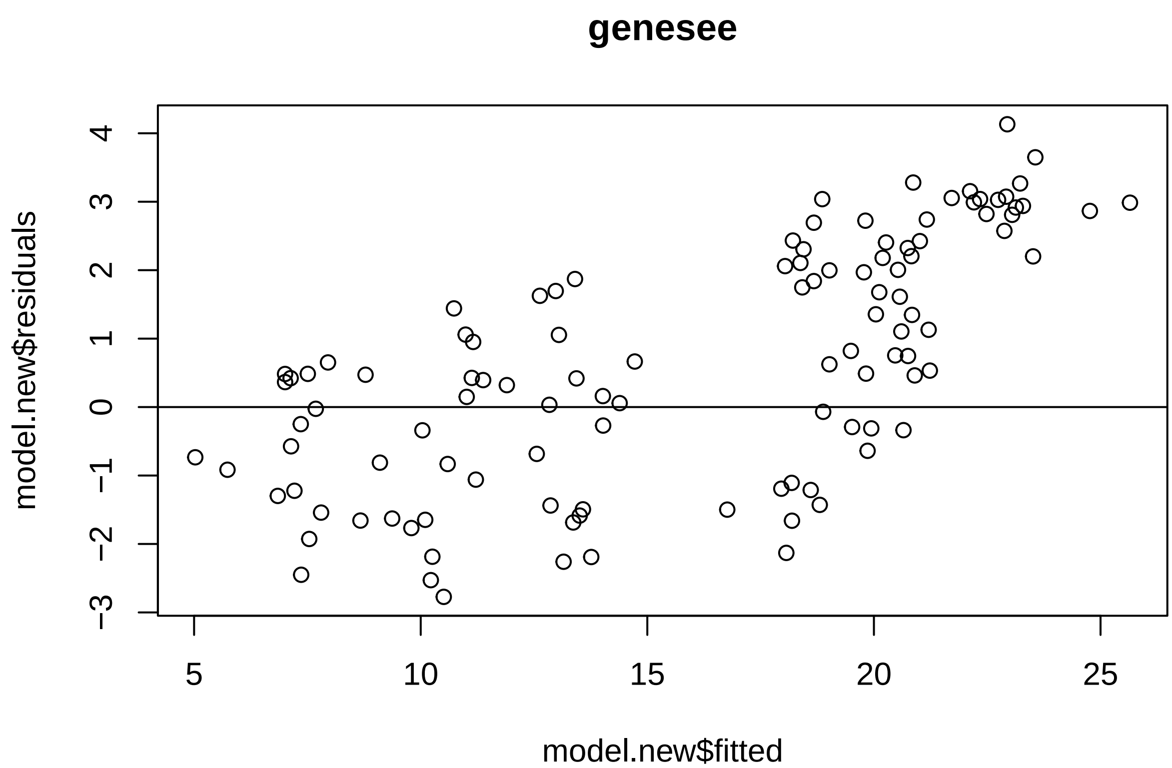

bigcreek : (RMSE: 1.13 & bias: 0.23)
20
15
10
5
water
15
20
25
week

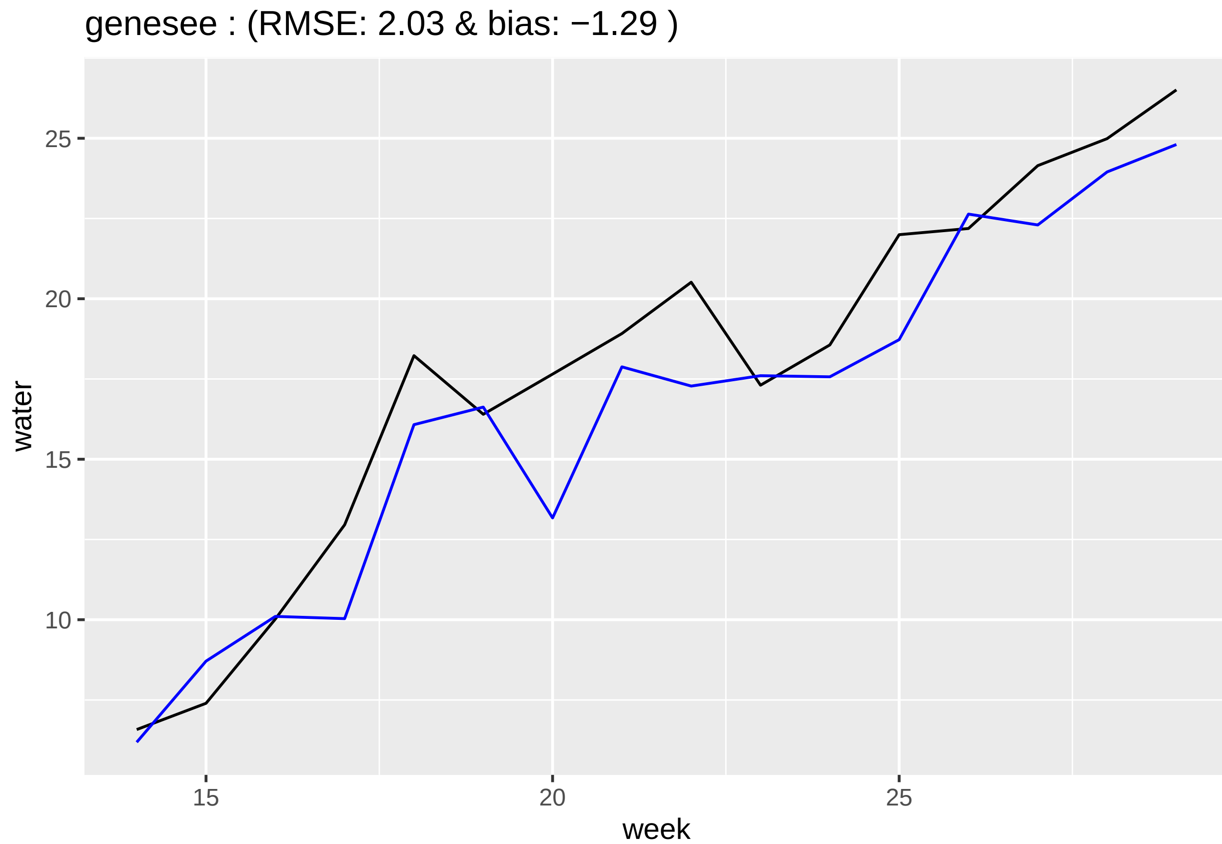

bigotter
model.new$residuals
model.new$fitted
4
2
0
−2
−4
6
8
10
12
14
16
18

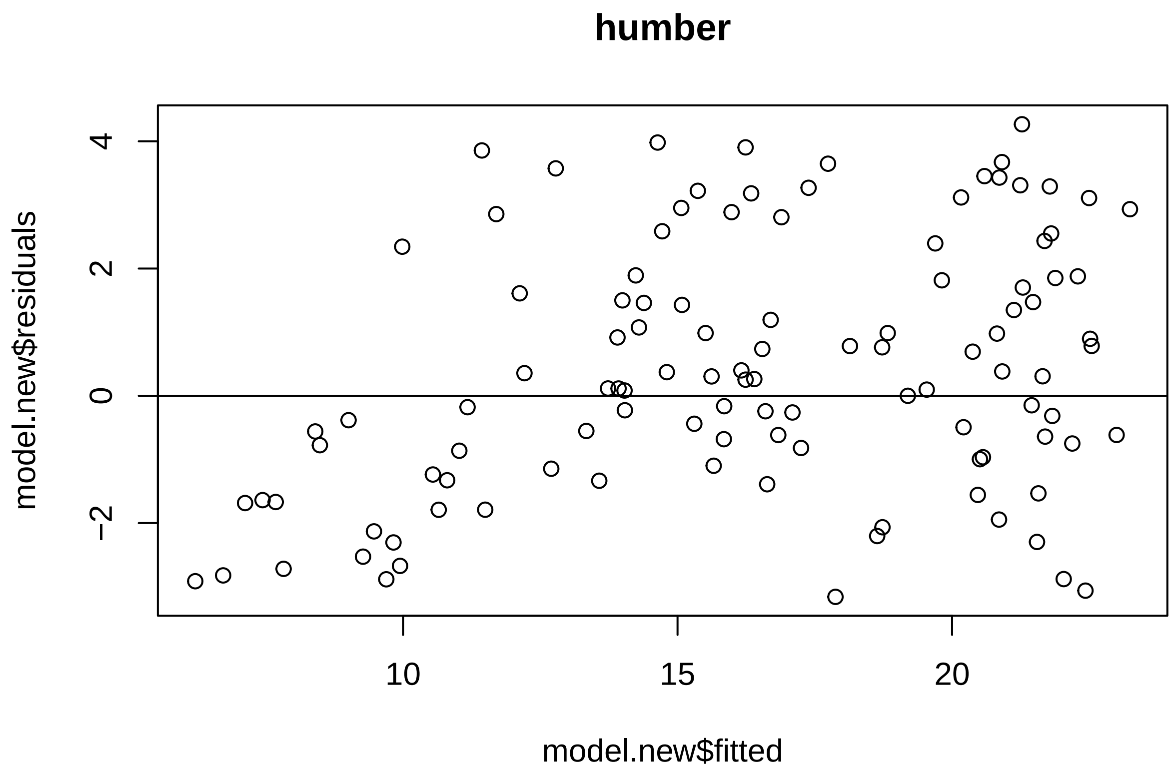

bigotter : (RMSE: 2.11 & bias: −1.59)
20
16
12
8
water
15
20
25
week

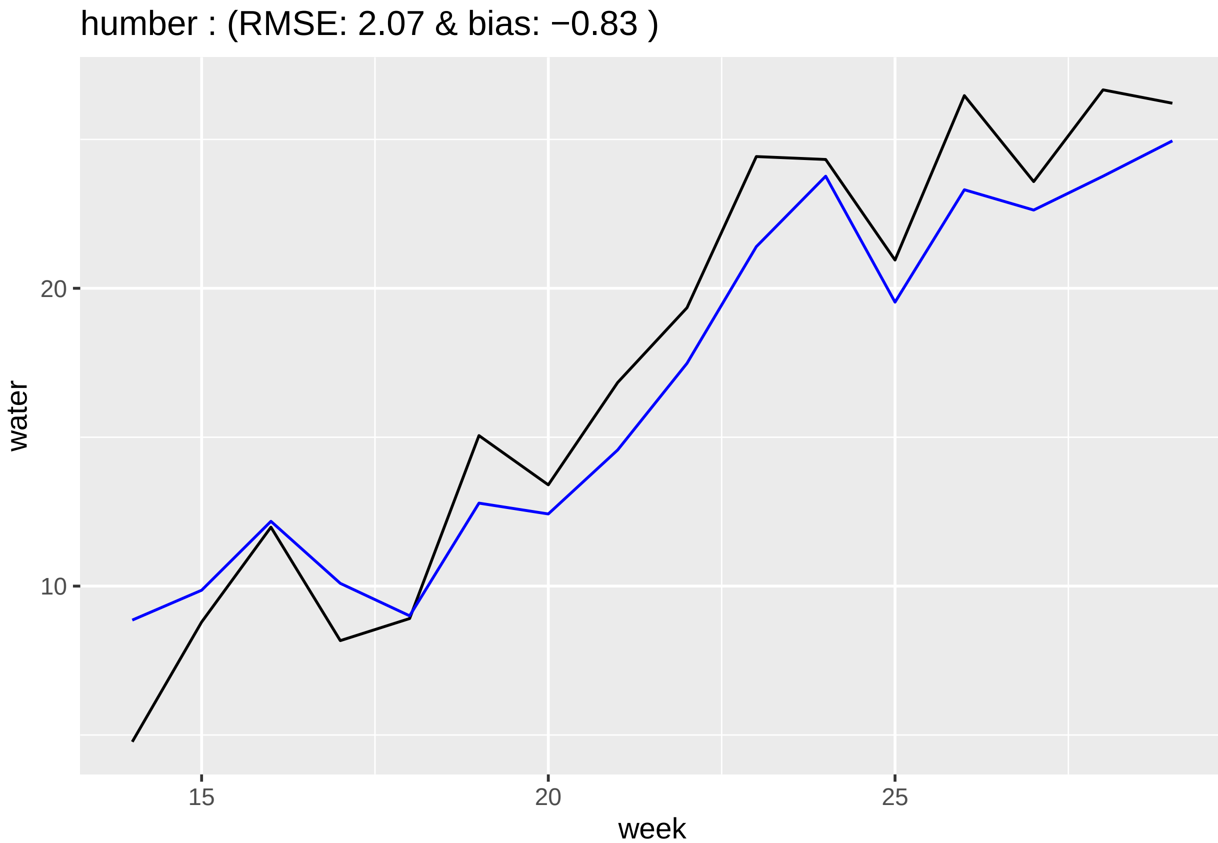

fox
model.new$residuals
model.new$fitted
3
2
1
0
−1
−2
−3
5
10
15
20
25


fox : (RMSE: 1.29 & bias: 0.03)
water
20
15
10
5
15
20
25
week

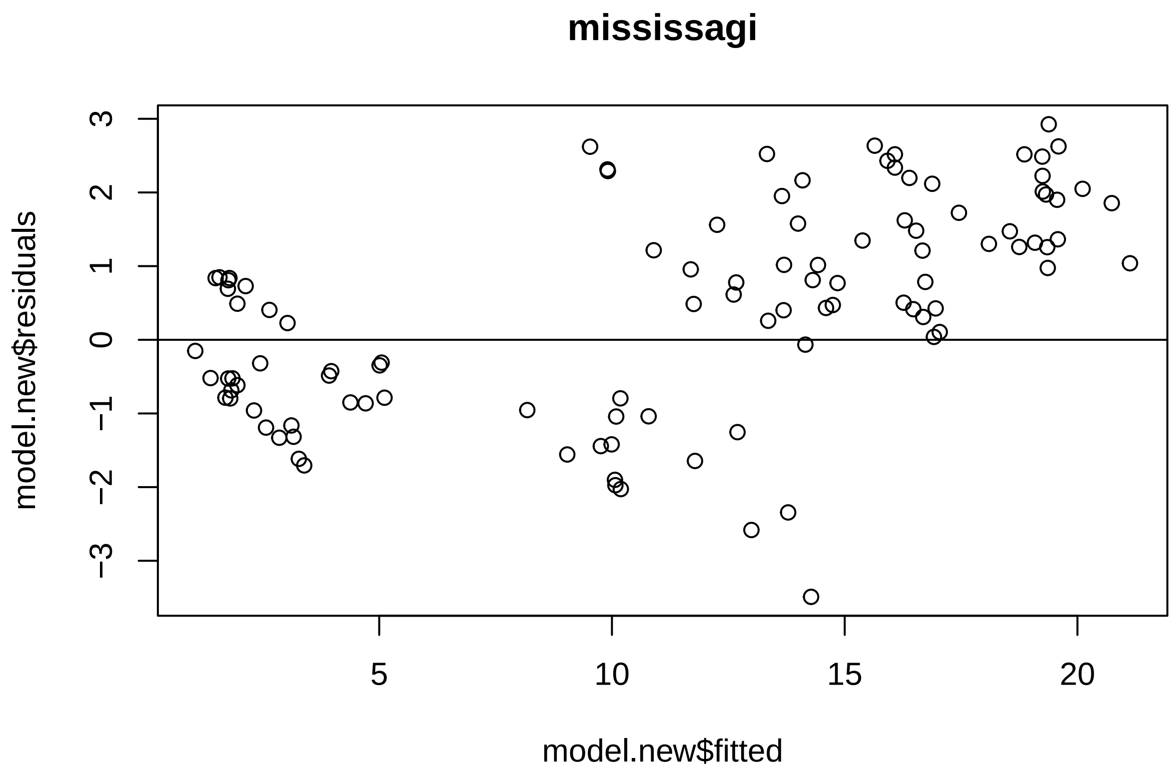

genesee
model.new$residuals
model.new$fitted
4
3
2
1
0
−1
−2
−3
5
10
15
20
25

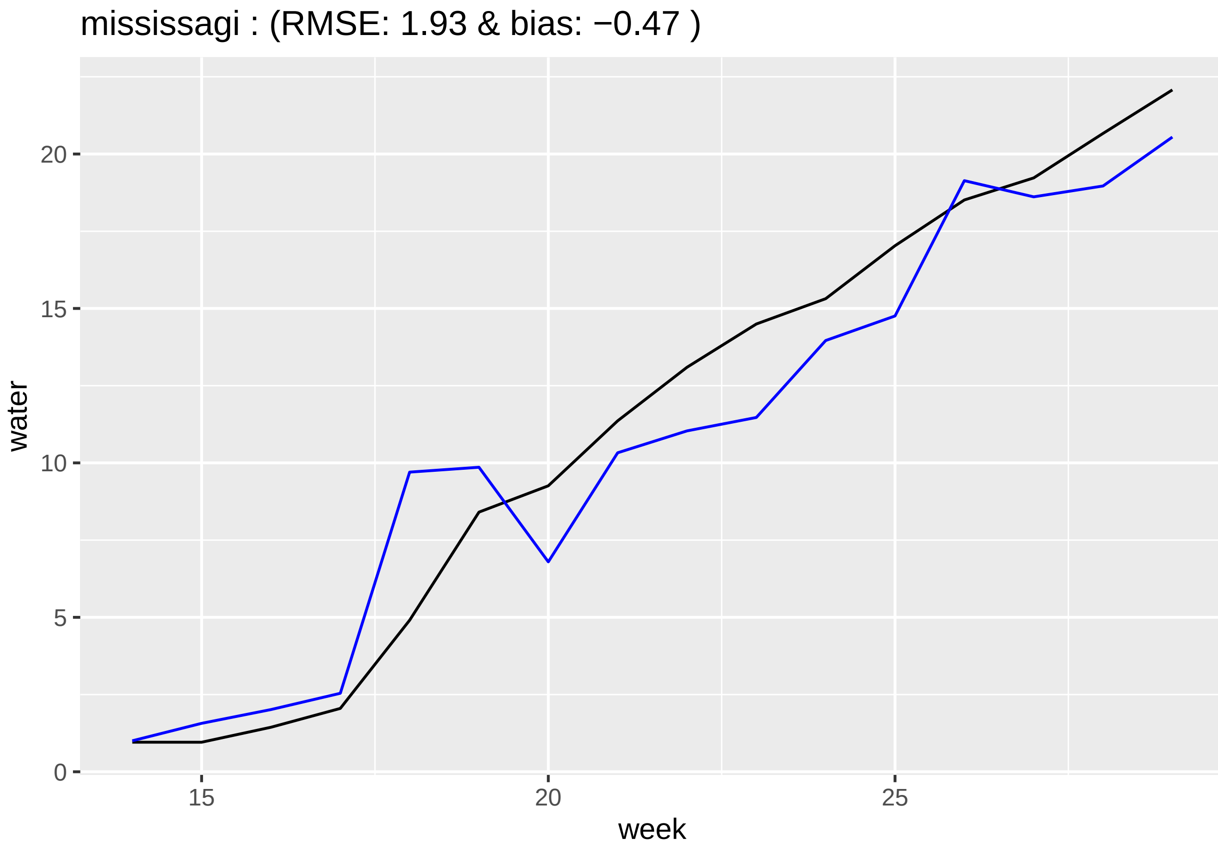

genesee : (RMSE: 2.03 & bias: −1.29)
water
25
20
15
10
15
20
25
week

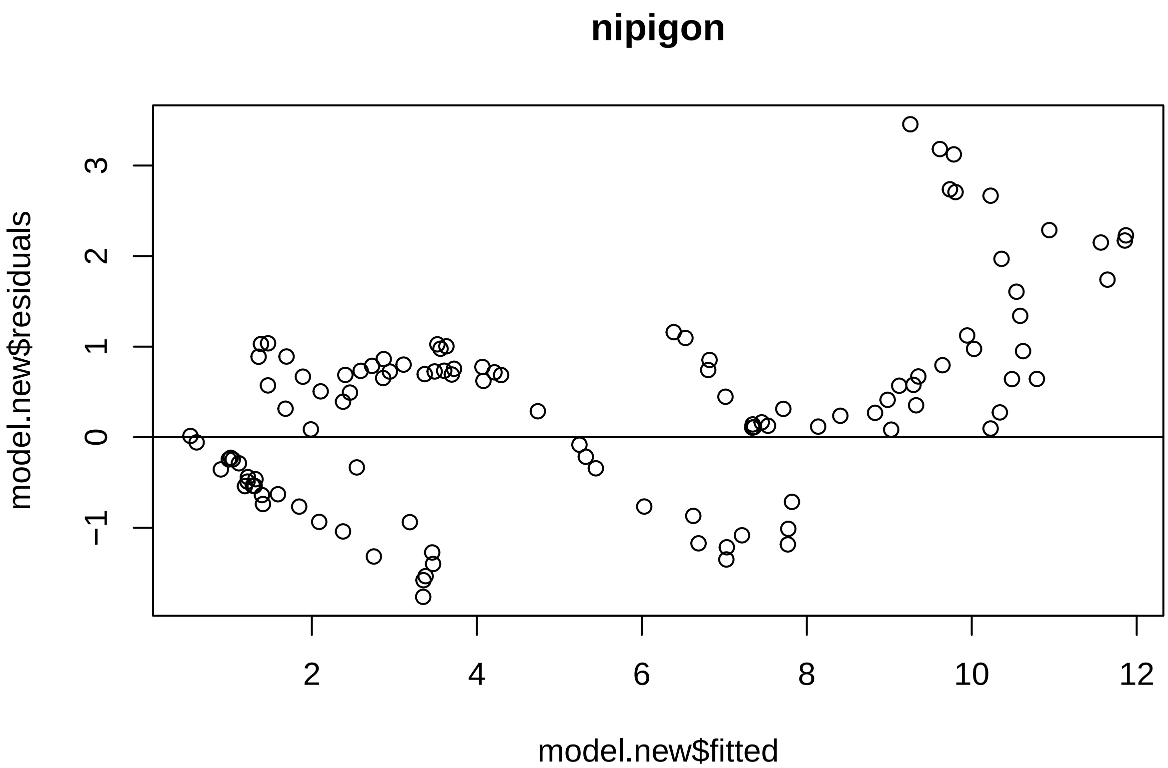

humber
model.new$residuals
4
2
0
−2
10
15
20
model.new$fitted



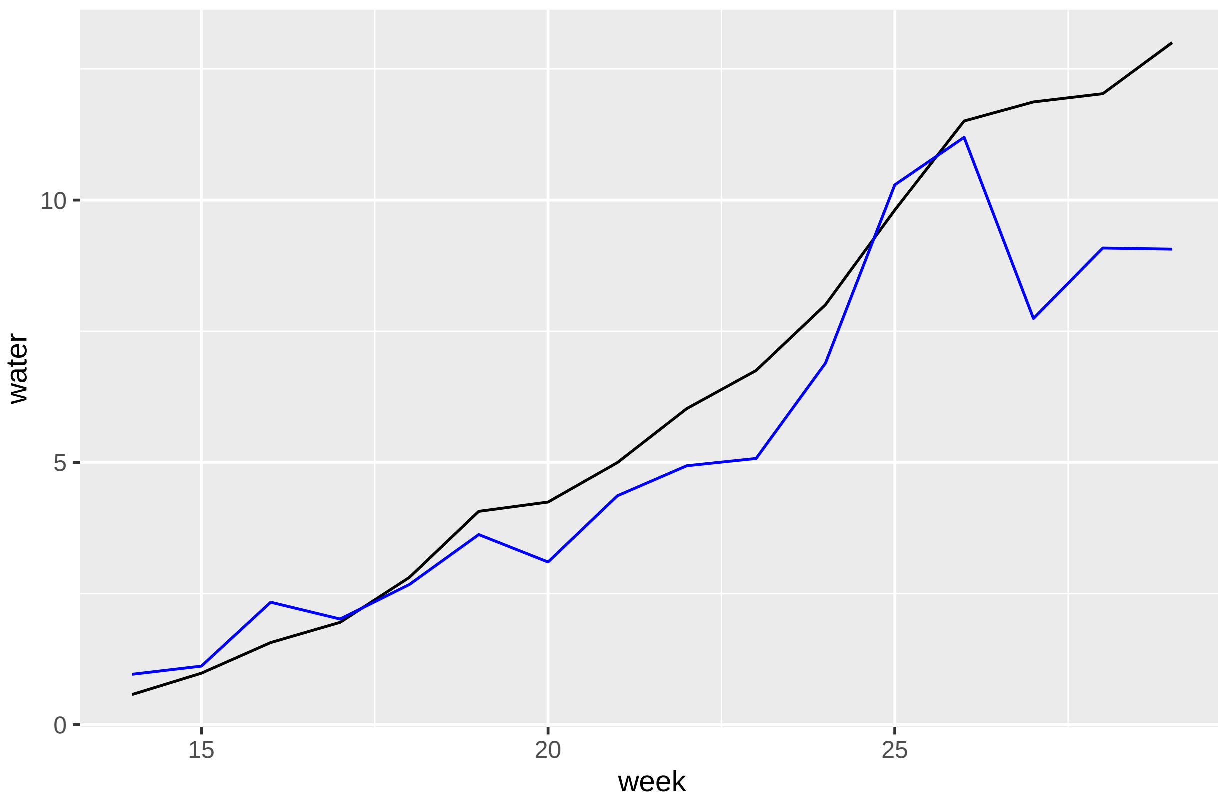
humber : (RMSE: 2.07 & bias: −0.83)
water
20
10
15
20
25
week

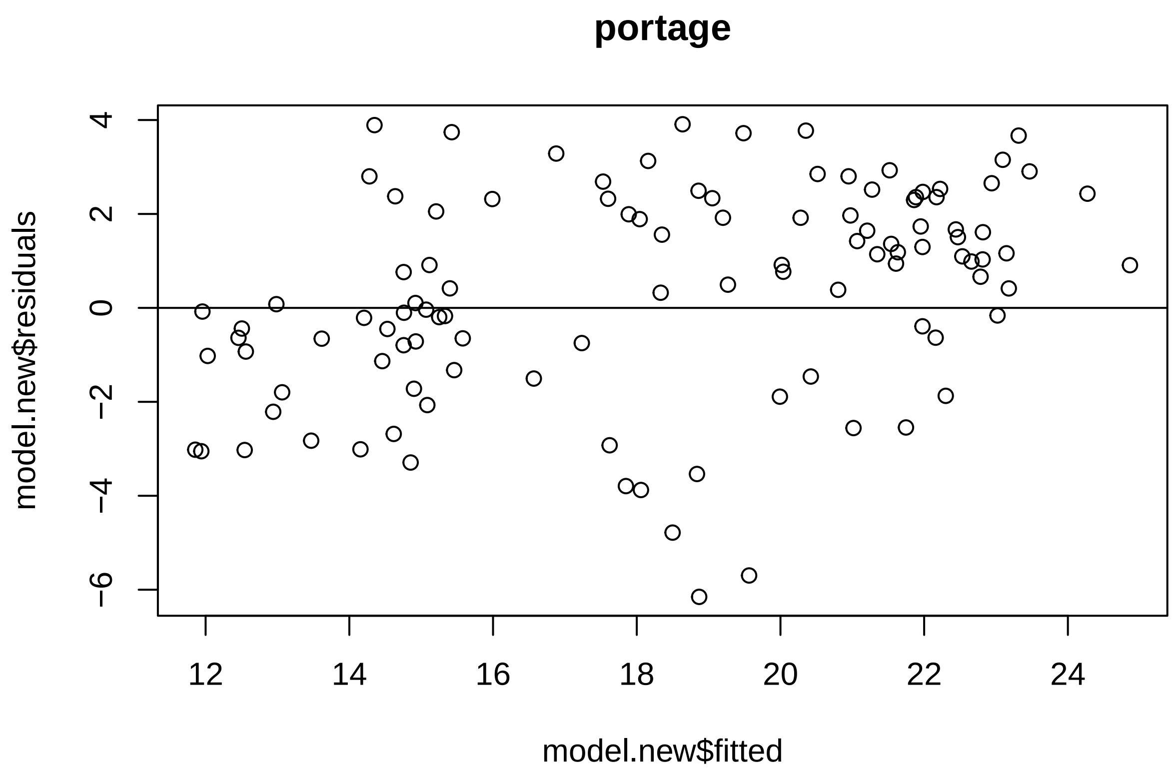

mississagi
model.new$residuals
model.new$fitted
3
2
1
0
−1
−2
−3
5
10
15
20

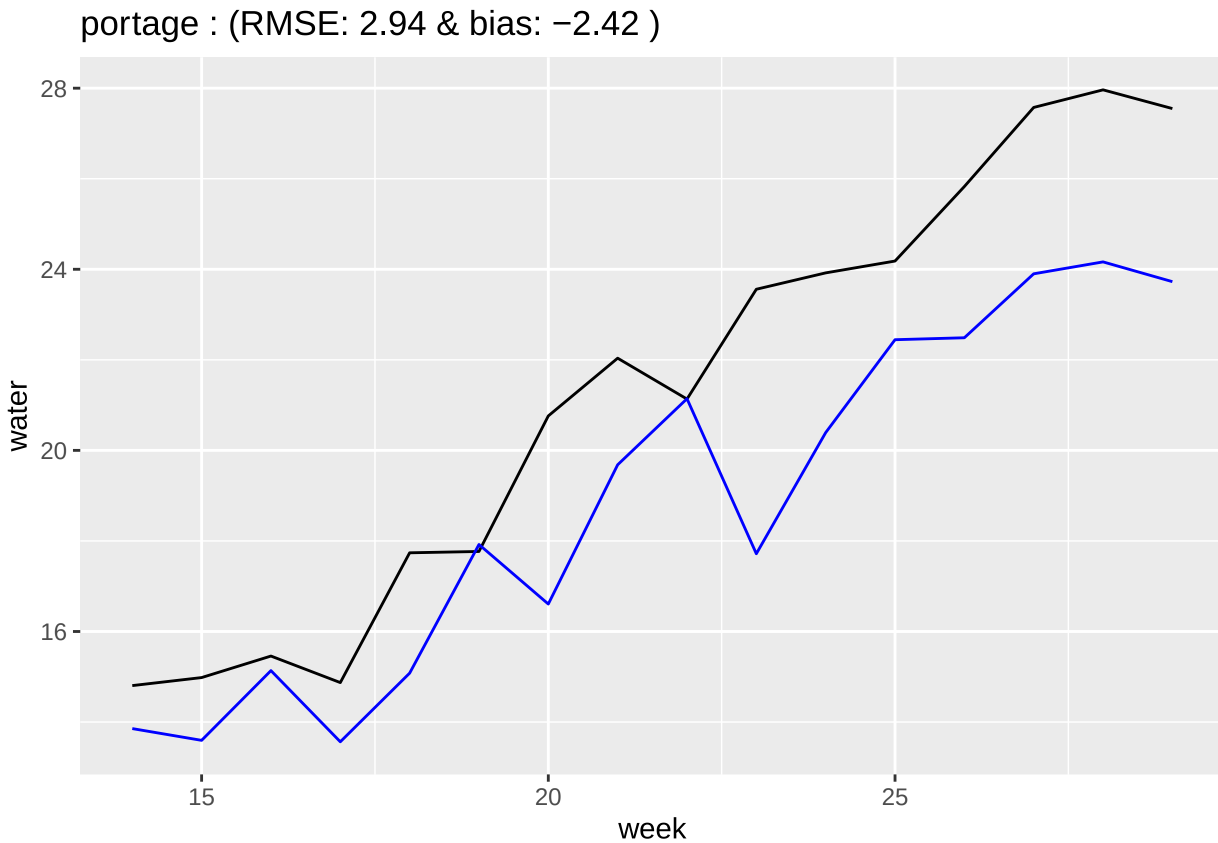

mississagi : (RMSE: 1.93 & bias: −0.47)
20
15
10
5
0
water
15
20
25
week

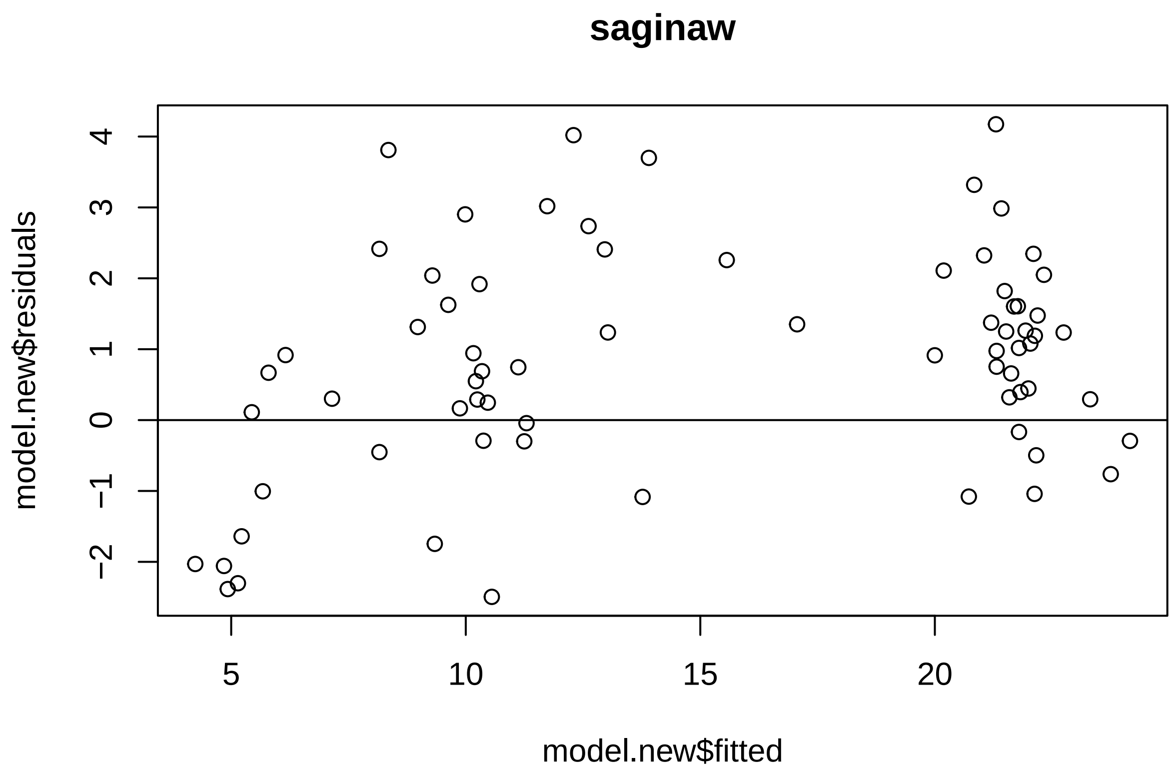

nipigon
model.new$residuals
model.new$fitted
3
2
1
0
−1
2
4
6
8
10
12

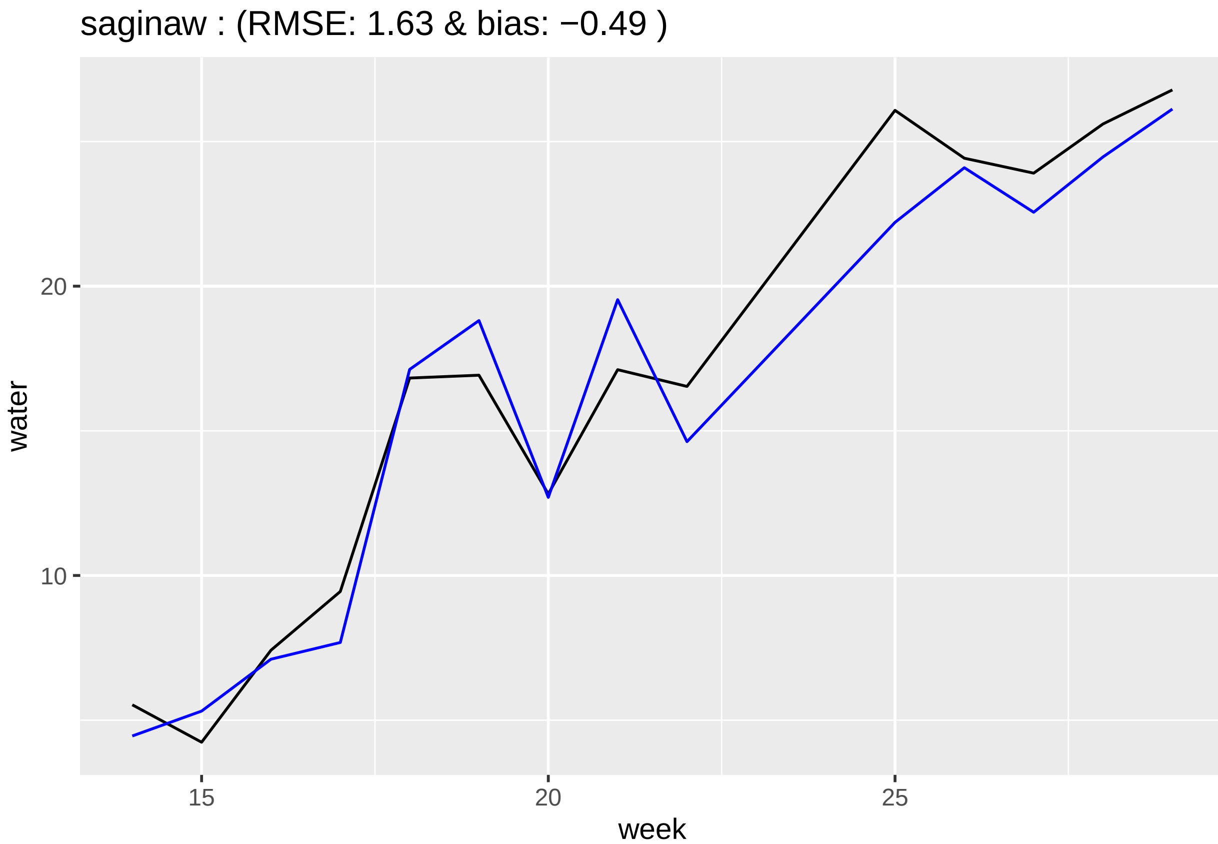
nipigon : (RMSE: 1.76 & bias: −0.98)

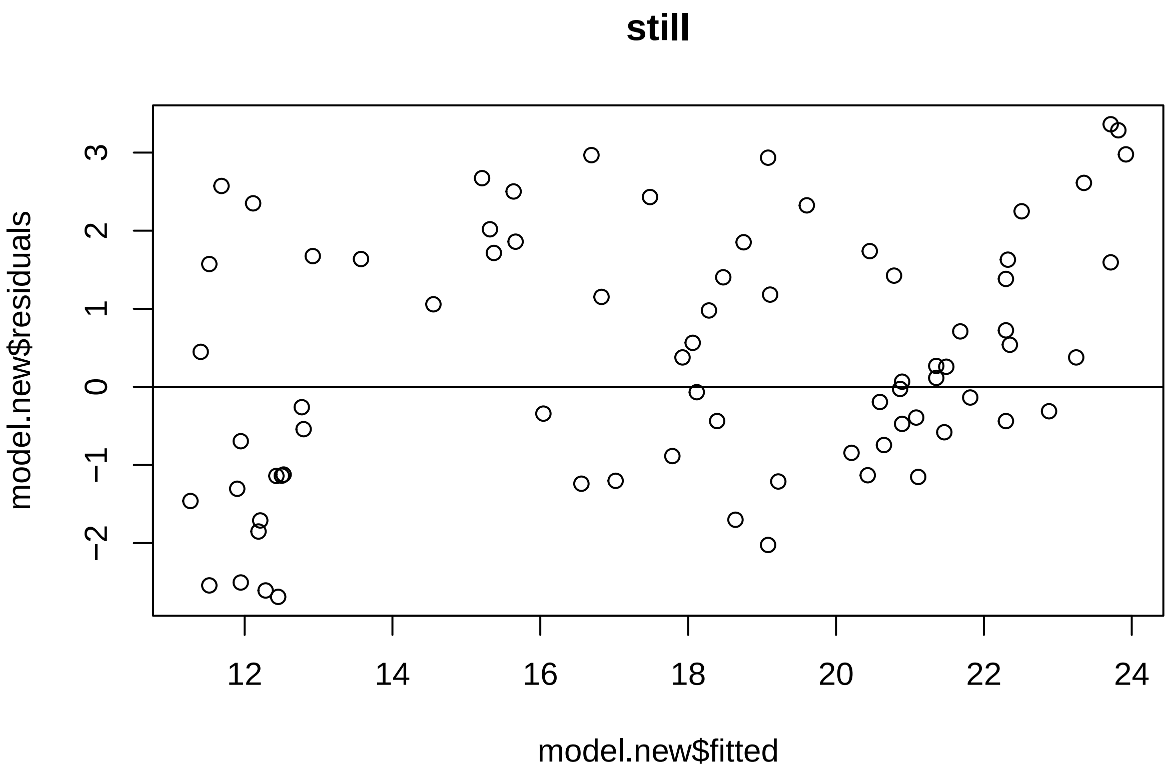


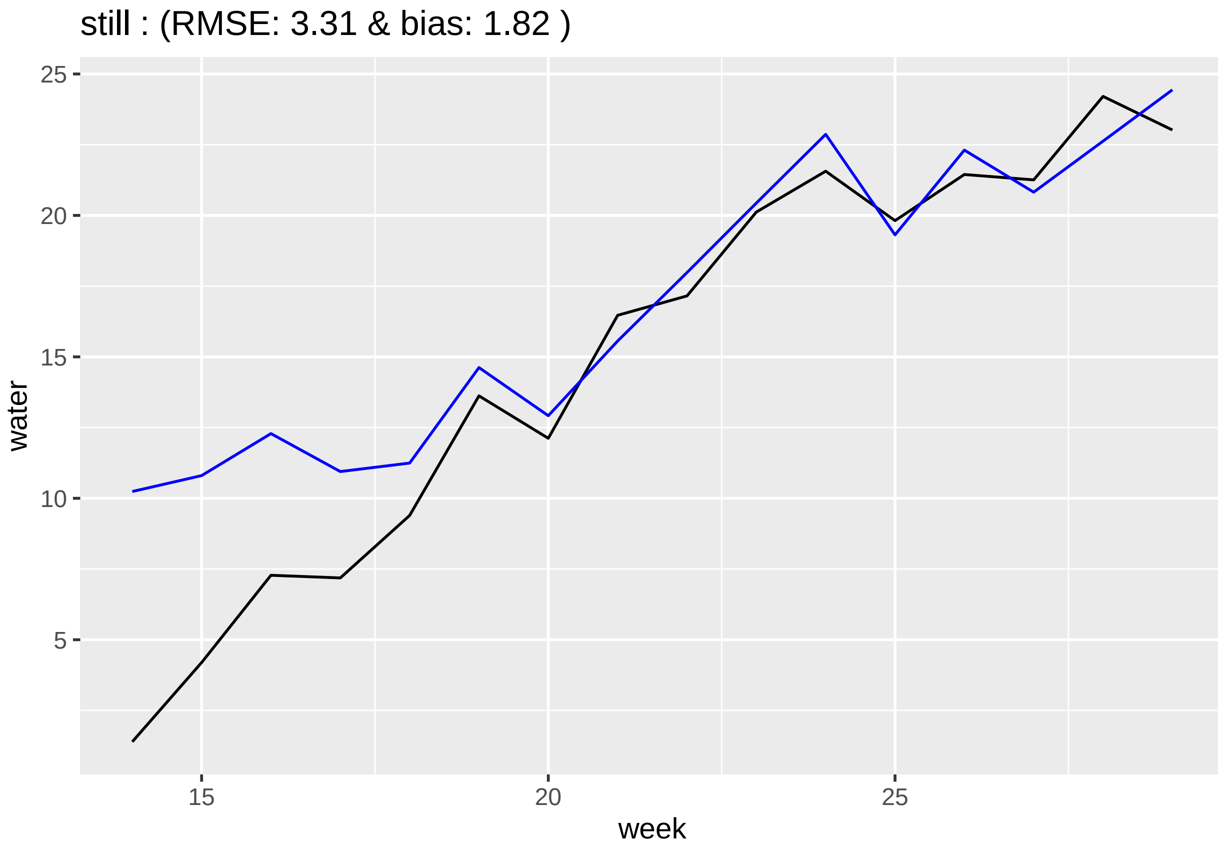

portage
model.new$residuals
model.new$fitted
4
2
0
−2
−4
−6
12
14
16
18
20
22
24

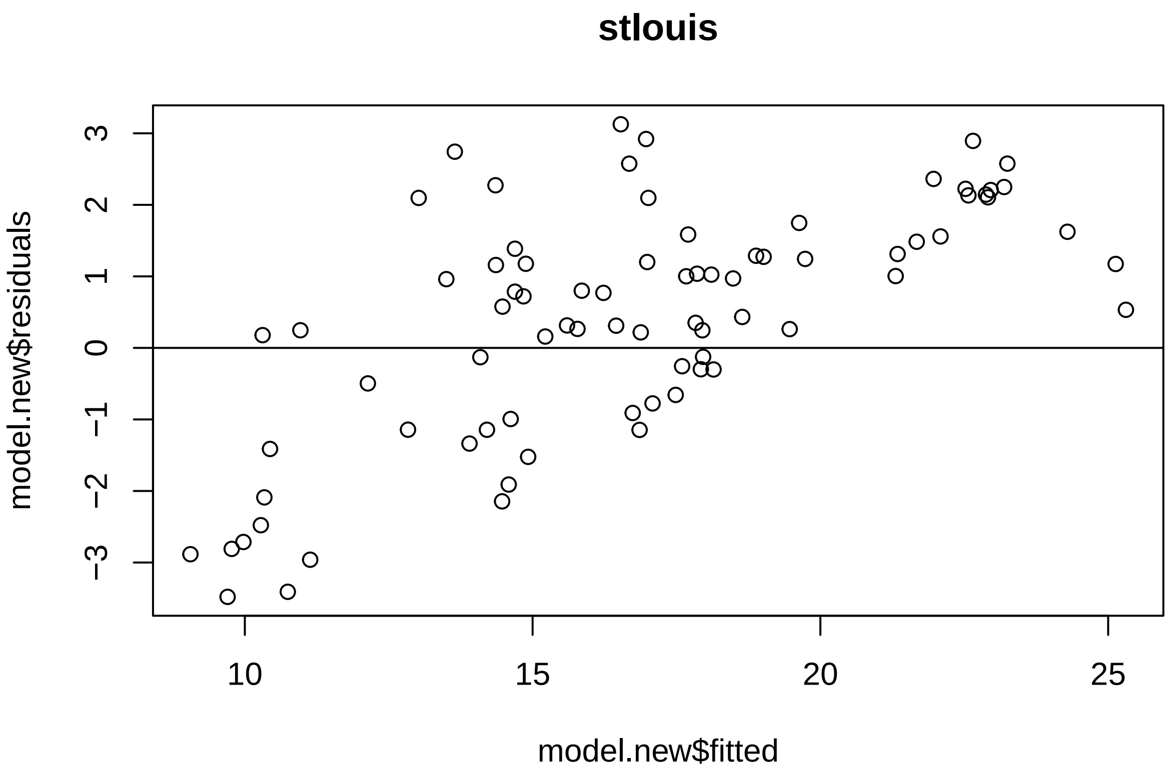

portage : (RMSE: 2.94 & bias: −2.42)
28
24
20
16
water
15
20
25
week

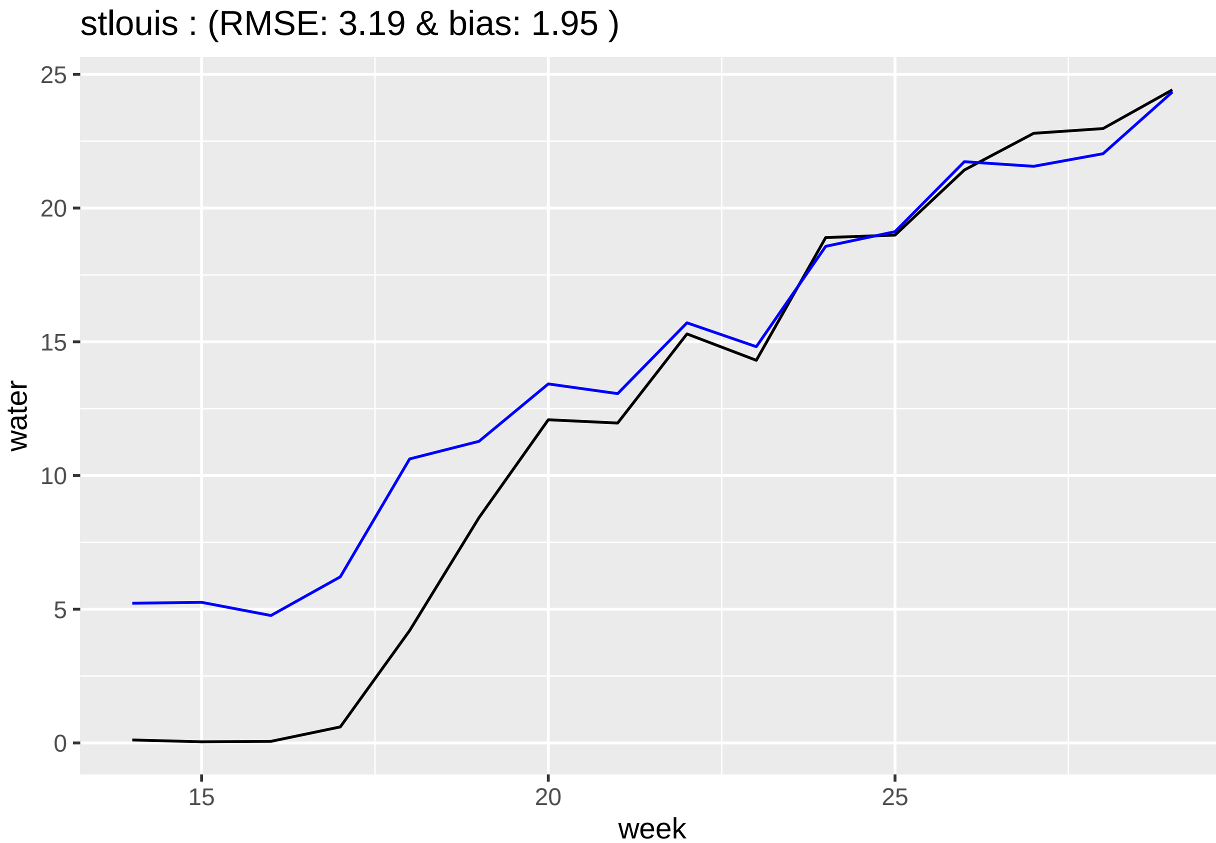

saginaw
model.new$residuals
model.new$fitted
4
3
2
1
0
−1
−2
5
10
15
20

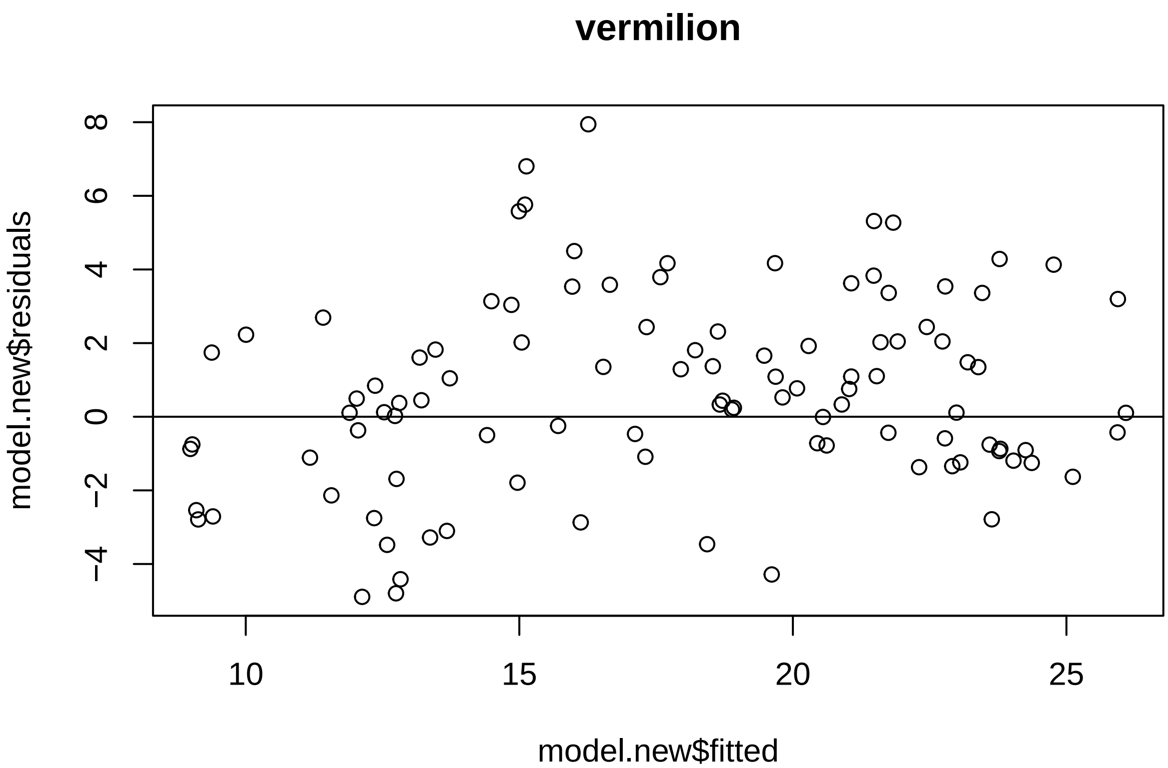

saginaw : (RMSE: 1.63 & bias: −0.49)
20
10
water
15
20
25
week

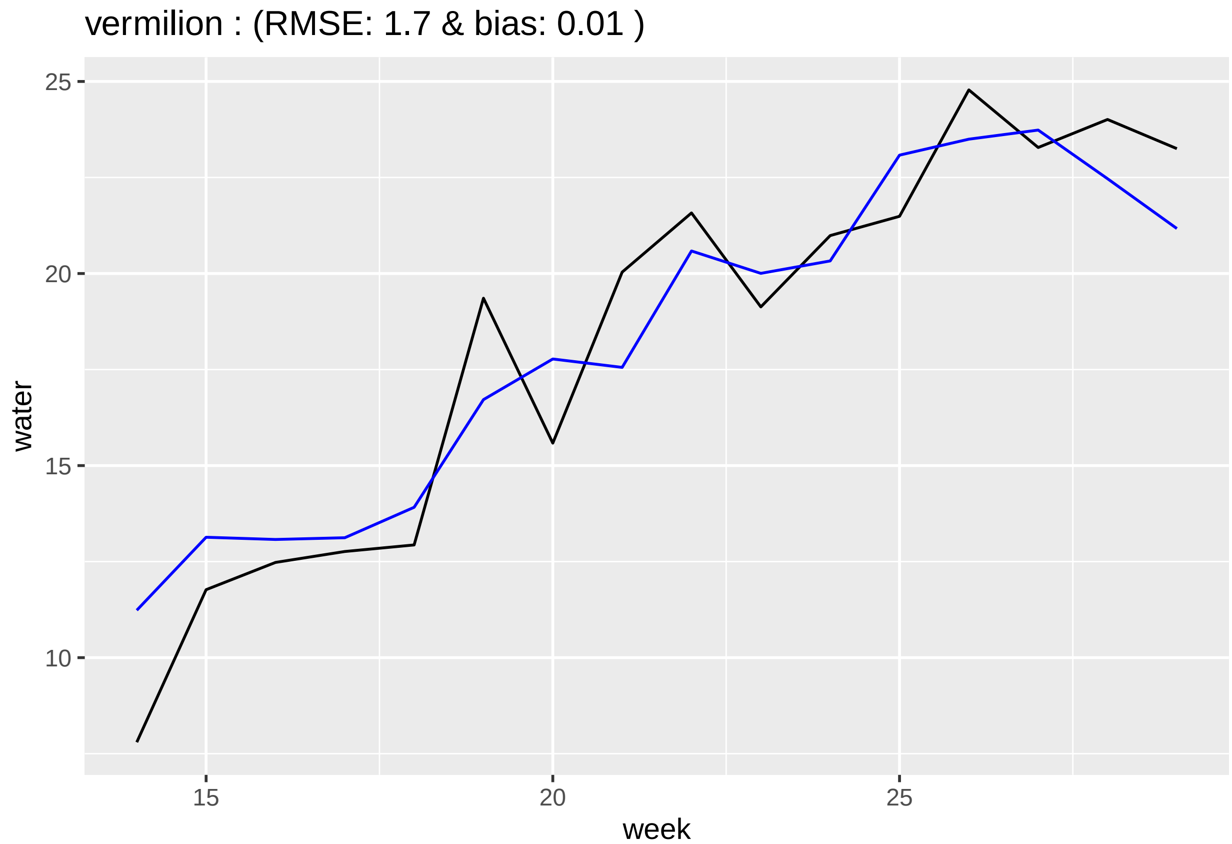

still
model.new$residuals
model.new$fitted
3
2
1
0
−1
−2
12
14
16
18
20
22
24


still : (RMSE: 3.31 & bias: 1.82)
25
20
15
10
5
water
15
20
25
week


stlouis
3
2
1
0
−1
−2
−3
model.new$residuals
10
15
20
25
model.new$fitted


stlouis : (RMSE: 3.19 & bias: 1.95)
25
20
15
10
5
0
water
15
20
25
week


vermilion
model.new$residuals
8
6
4
2
0
−2
−4
10
15
20
25
model.new$fitted



SECTION 4: Compare REGIONAL and GLOBAL

Global futureStream database

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

```
## Import the futureStreams modeled WT
futurestreams.temp <- read.csv("futureS modeled water temperature.csv") %>%
  rename(preds.futureS = modeled.temp) %>%
  arrange(location, year, week)
```

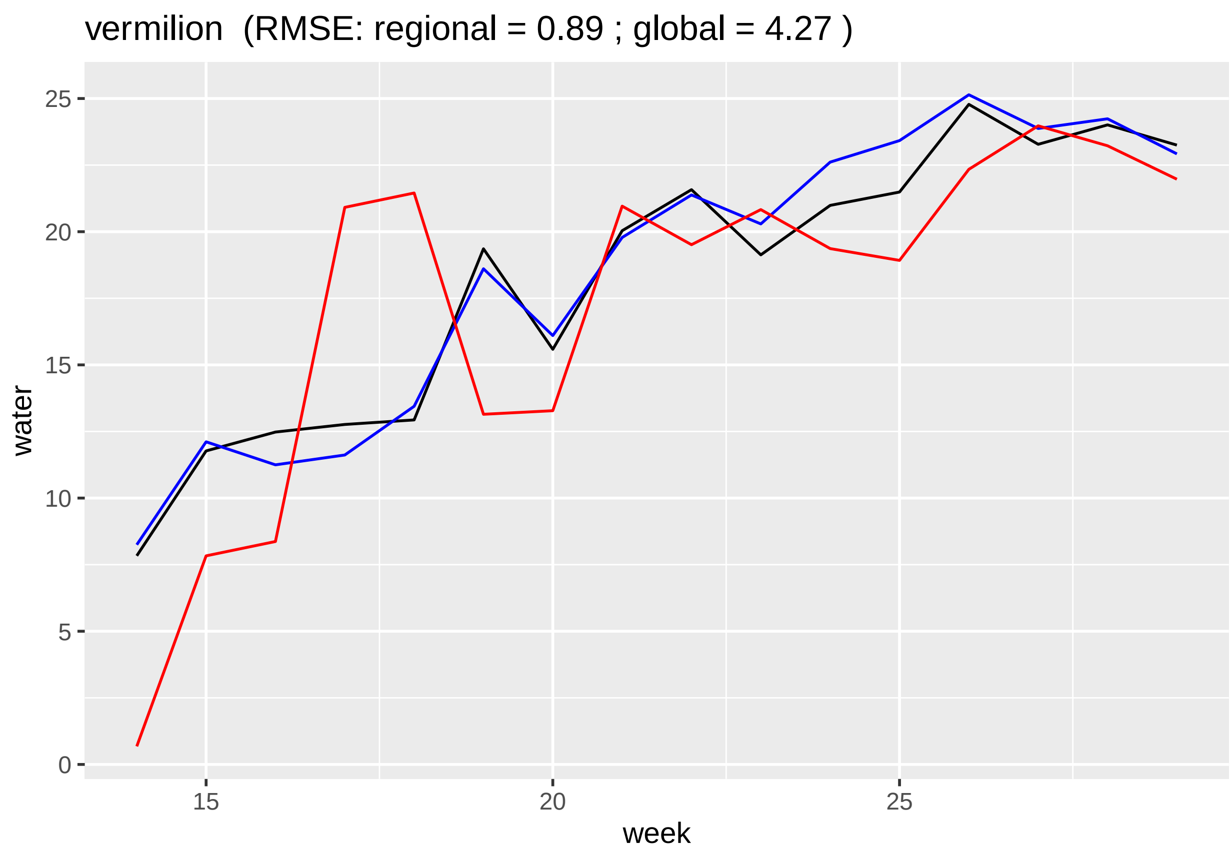
Combine REGIONAL and GLOBAL into one file

```
## Get future dataframe years and weeks
future.w <- merge(futurestreams.temp, y,
                  by.x = c("location","year"),
                  by.y = c("loc_seq","ytest")) %>%
  filter(week >= 14 & week <= 29) %>%
  arrange(location, week)
#' we are using data from Julian day 90-200, this is week 13-29


## Calculate performance metrics
for (loc in loc_seq) {

  compare.w <- merge(future.w[future.w$location == loc,],
```

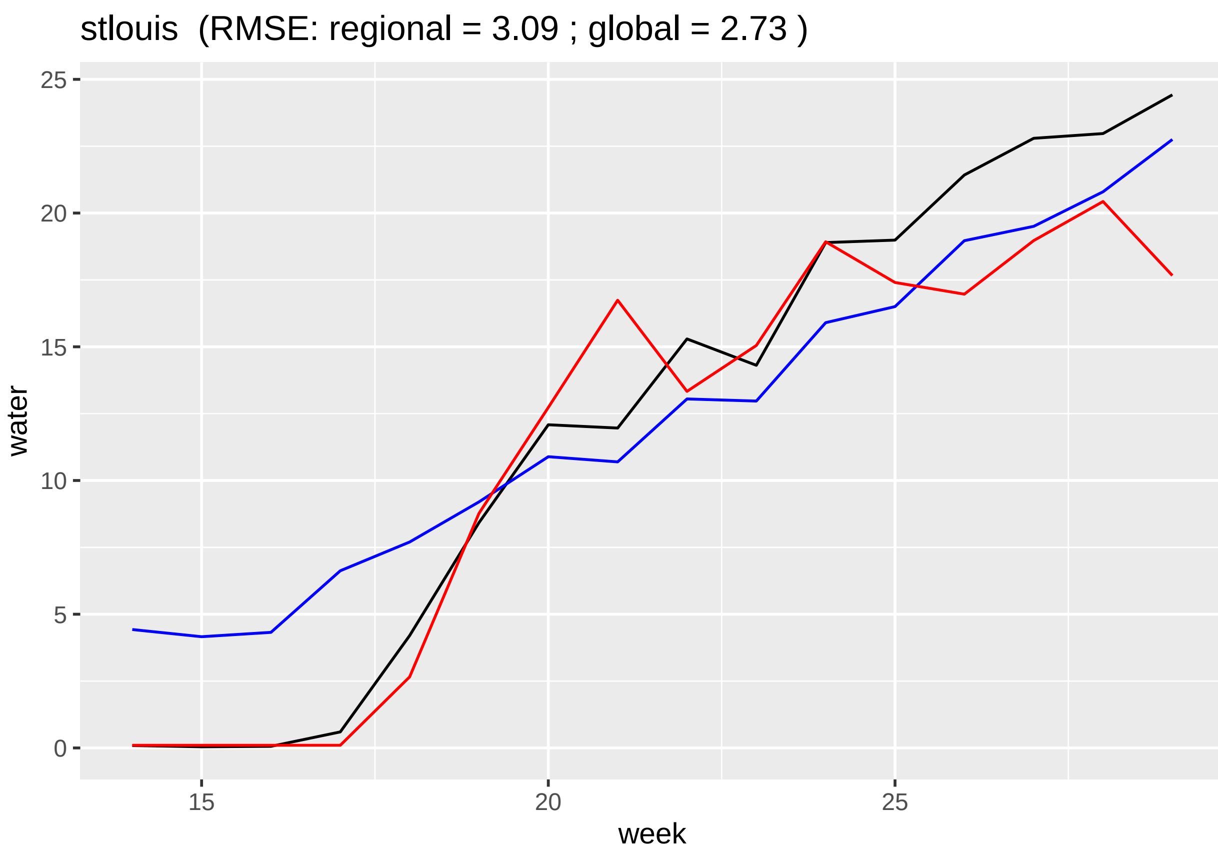
```r
                    complist[[loc]][[2]], # use seasonal output
                    by=c("location","week"))

  complist[[loc]][[4]] <- compare.w

  rmse.r[loc,4]=round(sqrt(mean((compare.w$preds.futureS-compare.w$water)^2)),2)
  bias.r[loc,4]=round(mean(compare.w$preds.futureS-compare.w$water),2)
  nsc.r[loc,4]=round(1-sum((compare.w$water-compare.w$preds.futureS)^2) / sum((compare.w$water-mean(com
}
```

Plot for comparison

```r
plot.df1 = complist[["vermilion"]][[4]]
plot.df2 = complist[["stlouis"]][[4]]


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)

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, ")"))
```




```r
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, ")"))
```



Write results into csv files

```
write.csv(rmse.r, file="RMSE for plot.csv")
write.csv(bias.r, file="bias for plot.csv")
write.csv(nsc.r, file="NSC for plot.csv")
```