

# Calibration

Carlo, Ata, Hailey

2023-05-11

```
library(sensitivity)

## Registered S3 method overwritten by 'sensitivity':
##   method      from
##   print.src    dplyr

library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.1
## v tibble  3.1.8      v dplyr  1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ----- tidyverse_conflicts() --
## x tidyr::extract() masks sensitivity::extract()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x dplyr::src()     masks sensitivity::src()

library(lubridate)

## Loading required package: timechange
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##   date, intersect, setdiff, union

library(reldist)

## reldist: Relative Distribution Methods
## Version 1.7-2 created on 2023-02-16.
## copyright (c) 2003, Mark S. Handcock, University of California-Los Angeles
## For citation information, type citation("reldist").
## Type help(package="reldist") to get started.

library(purrr)
library(ggpubr)
library(dplyr)
library(lubridate)
library(patchwork)
```

## Assignment

Final piece will be to produce a graph of maximum likelihood estimate given you acceptable parameters!

To hand in - an Rmarkdown and R function. Please knit and turn in either an html or pdf of the markdown.

Part 1 from above: R function that codes a metric for performance evaluation

- must be a combination of at least two performance measures
- include some comments that explain ‘why’ this metric

As a group we wanted to see if the model was getting things right on average but we thought that an annual metric would lose some of the seasonal variability, so we created two indicators that measure monthly average model performance and combined them.

```
# create custom performance metric
combined_performance_metric = function(m, o, month) {

  # create a data frame and calculate monthly mean for m and o
  df = data.frame(m, o, month) |>
    group_by(month) |>
    summarise(ave_m = mean(m), ave_o = mean(o))

  # create a monthly average NSE score
  err = df$ave_m - df$ave_o
  meanobs = mean(df$ave_o)
  mse = sum(err*err)
  ovar = sum((df$ave_o-meanobs)*(df$ave_o-meanobs))
  monthly_nse = 1.0-mse/ovar

  # create a monthly error score and normalize
  month_error = sum(df$ave_m - df$ave_o)
  month<- abs(month_error)
  month_error <- exp(-month)

  # simple multiplication of performance metrics
  combined = monthly_nse * month_error

  # return combined performance metric
  return(combined)
}
```

Part 2:

Apply your performance function to a subset of the Sagehen data set (with multiple simulations) that you want to use for calibration

Summarize the performance over the calibration period in 1-2 graphs; you can decide what is useful

Record your ‘best’ and ‘worst’ parameter set in this spreadsheet and in your Rmd

**Worst Simulation: S95**

**Best Simulation: S48**

```
# read in data
sager = read.table("sager.txt", header=T)

# add date
```

```

sager = sager %>% mutate(date = paste(day,month,year, sep="/"))
sager$date = as.Date(sager$date,"%d/%m/%Y")

# read in model data
msage = read.table("sagerm.txt", header=T)

# rename columns
nsim = ncol(msage)
snames = sprintf("S%d",seq(from=1, to=nsim))
colnames(msage)=snames

# lets say we know the start date from our earlier output
msage$date = sager$date
msage$month = sager$month
msage$year = sager$year
msage$day = sager$day
msage$wy = sager$wy

# add sager observation data
msage = left_join(msage, sager[,c("obs","date")], by=c("date"))

# map the the performance of the different models
res = msage %>% select(!c("date","month",
                        "year","day","wy",
                        "obs")) %>%
  map_dbl(combined_performance_metric, month = msage$month, msage$obs)

# make the output of the map function a df
res <- as.data.frame(res)

# put the names of the models on the data frame
res$sim = snames

# get the best and worst performing models
highest_row <- which.max(res$res)
lowest_row <- which.min(res$res)
highest_sim <- res$sim[highest_row]
lowest_sim <- res$sim[lowest_row]

# put worst and best model outputs into a df
best_worst_output <- msage |>
  select(S95, S48)

# add the time and observations for the outputs
best_worst_data <- msage %>%
  select(c("date","month",
          "year","day","wy",
          "obs")) |>
  cbind(best_worst_output)

# create the graphing df by pivoting longer and filtering by date
ggplot_df = best_worst_data %>%
  pivot_longer(cols=!c(date, month, year, day, wy),
              names_to="sim",

```

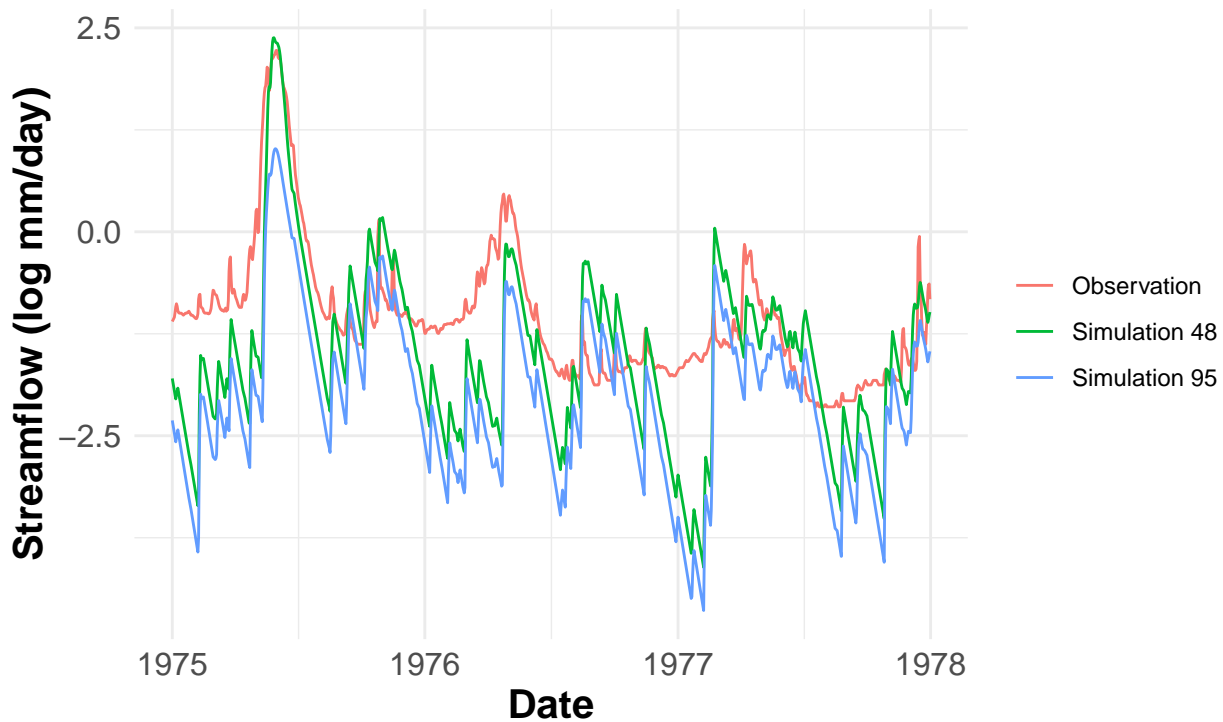
```
values_to="flow") |>
filter(year(date) >= 1975, year(date) <= 1977)
```

```
# make log flow line graph
ggplot(ggplot_df, aes(x = date, y = log(flow), color = sim)) +
  geom_line(size = 0.5) + # set line size to be very skinny
  scale_color_discrete(labels = c("Observation", "Simulation 48", "Simulation 95")) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
    plot.subtitle = element_text(hjust = 0.5, size = 15),
    axis.title = element_text(face = "bold", size = 15),
    axis.text = element_text(size = 12),
    legend.title = element_blank() # removes the legend title
  ) +
  labs(
    title = "Log Streamflow over Time",
    subtitle = "Simulations Vs Observation",
    x = "Date",
    y = "Streamflow (log mm/day)"
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```

## Log Streamflow over Time

### Simulations Vs Observation

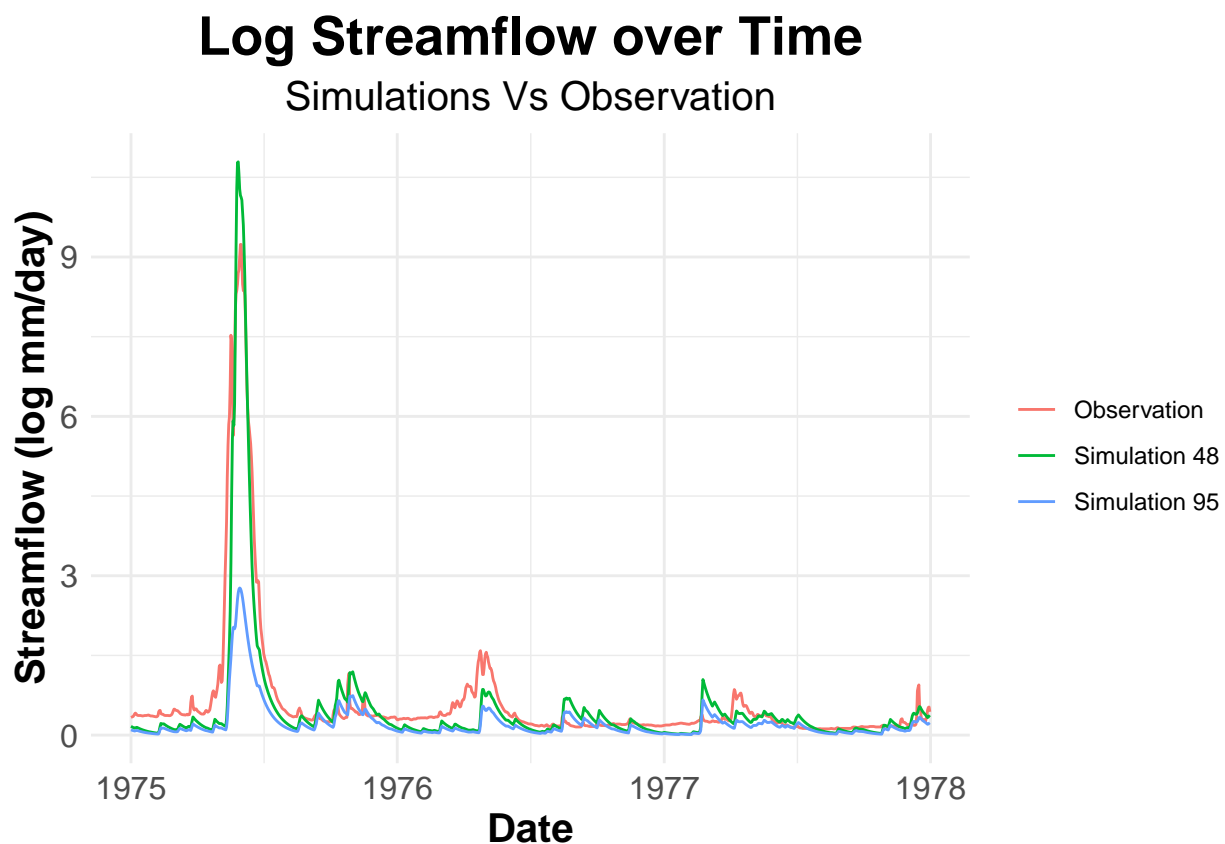


```
# make flow line graph
ggplot(ggplot_df, aes(x = date, y = flow, color = sim)) +
  geom_line(size = 0.5) +
```

```

scale_color_discrete(labels = c("Observation", "Simulation 48", "Simulation 95")) +
theme_minimal() +
theme(
  plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
  plot.subtitle = element_text(hjust = 0.5, size = 15),
  axis.title = element_text(face = "bold", size = 15),
  axis.text = element_text(size = 12),
  legend.title = element_blank() # removes the legend title
) +
labs(
  title = "Log Streamflow over Time",
  subtitle = "Simulations Vs Observation",
  x = "Date",
  y = "Streamflow (log mm/day)"
)

```



```

# make dataframe for boxplot
boxplot_df = best_worst_data %>%
  pivot_longer(cols=!c(date, month, year, day, wy),
    names_to="sim",
    values_to="flow")

# make boxplot
ggplot(boxplot_df, aes(sim, flow, fill = sim)) +
  geom_boxplot(outlier.shape = NA) +
  scale_fill_discrete(labels = c("Observation", "Simulation 48", "Simulation 95")) +
  coord_cartesian(ylim = c(0, 2.5)) +

```

```

theme_minimal() +
theme(
  plot.title = element_text(hjust = 0.5, face = "bold", size = 20),
  plot.subtitle = element_text(hjust = 0.5, size = 15),
  axis.title = element_text(face = "bold", size = 15),
  axis.text = element_text(size = 12),
  legend.title = element_blank(),
  legend.text = element_text(size = 10)
) +
labs(
  title = "Total Flow Distrobution",
  subtitle = "Sumulations vs Observed",
  x = NULL,
  y = "Flow"
)

```

