STAT 575 Project #2

Cecilia Albert-Black 10/29/2025

The objective of this project is to analyze a time series regression dataset and to perform a subset selection of predictor variables using advanced covariance estimation methods.

Here I will estimate models using heteroskedasticity- (HC) and autocorrelation-consistent (HAC) covariance estimators and compare model performance using three information-theoretic criteria: AIC, SBC (BIC), and CICOMP.

Many time series data sets typically contain autocorrelation and/or heteroskedasticity of unknown form and, for statistical inference to model such data, it is essential to use covariance matrix estimators to estimate model parameters; in which case we use HC or HAC estimators.

Dataset

The dataset used for this project is not publicly available since it is propertly of Oak Ridge National Laboratory. Therefore, I will only preview the data but not provide it in the submission of this project to the instructor.

Dataset details: This dataset includes several stream measurements (e.g. temperature, pH, etc.) collected every couple minutes from 2023-10-20 to 2024-09-10.

```r
setwd("C:/Users/q4q/OneDrive - Oak Ridge National Laboratory/Research/Bioavailability Project/2. Time Se

library(readxl)
mca1_dat <- read_excel("mca1_bioavail_interpolation.xlsx")
View(mca1_dat)
str(mca1_dat)
```

```
## tibble [46,991 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Site       : chr [1:46991] "MCA1" "MCA1" "MCA1" "MCA1" ...
##  $ DateTime   : POSIXct[1:46991], format: "2023-10-20 07:20:00" "2023-10-20 07:30:00" ...
##  $ Temp       : num [1:46991] 21.8 21.9 21.9 22.3 22.2 ...
##  $ Intensity  : num [1:46991] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Observation: num [1:46991] 0 1 2 3 4 5 6 7 8 9 ...
##  $ pH         : num [1:46991] 7.57 7.57 7.57 7.57 7.57 ...
##  $ Temp_C     : num [1:46991] 15.6 15.6 15.6 15.6 15.6 ...
##  $ DOC_M      : num [1:46991] 8.07e-05 8.07e-05 8.07e-05 8.07e-05 8.06e-05 ...
##  $ Ca_uM      : num [1:46991] 1555 1555 1555 1555 1555 ...
##  $ Mg_uM      : num [1:46991] 494 494 494 494 494 ...
##  $ Na_uM      : num [1:46991] 128 128 128 128 128 ...
##  $ K_uM       : num [1:46991] 27.3 27.3 27.3 27.3 27.3 ...
##  $ SO4_uM     : num [1:46991] 56 56 56 56 56 ...
##  $ Cl_uM      : num [1:46991] 150 150 150 150 150 ...
##  $ DIC_mM     : num [1:46991] 4.26 4.26 4.26 4.26 4.26 ...
##  $ Meas_Zn_nM : num [1:46991] 90 90 90 90 90.1 ...
```

Because I experienced difficulty in the run time of my data, I will decrease the temporal resolution to daily averages

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
daily_avg <- mca1_dat %>%
  mutate(Date = as.Date(DateTime)) %>%         # extract date
  group_by(Site, Date) %>%                     # group by site and day
  summarise(across(where(is.numeric), mean, na.rm = TRUE)) %>%  # daily means
  ungroup()
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(where(is.numeric), mean, na.rm = TRUE)`.
## i In group 1: `Site = "MCA1"` `Date = 2023-10-20`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))

## `summarise()` has grouped output by 'Site'. You can override using the
## `.groups` argument.
```

```r
# Check result
head(daily_avg)
```

```
## # A tibble: 6 x 16
##   Site  Date        Temp Intensity Observation    pH Temp_C    DOC_M Ca_uM Mg_uM
##   <chr> <date>     <dbl>     <dbl>       <dbl> <dbl>  <dbl>    <dbl> <dbl> <dbl>
## 1 MCA1  2023-10-20  20.9        0         49.5  7.57   15.6 8.01e-5 1556.  494.
## 2 MCA1  2023-10-21  20.9        0        172.   7.57   15.5 7.87e-5 1561.  495.
## 3 MCA1  2023-10-22  20.9        0        316.   7.56   15.5 7.70e-5 1566.  497.
## 4 MCA1  2023-10-23  21.1        0        460.   7.56   15.4 7.55e-5 1571.  498.
## 5 MCA1  2023-10-24  21.1        0        604.   7.55   15.3 7.40e-5 1576.  498.
## 6 MCA1  2023-10-25  18.3   126259.       748.   7.55   15.3 7.26e-5 1580.  499.
## # i 6 more variables: Na_uM <dbl>, K_uM <dbl>, SO4_uM <dbl>, Cl_uM <dbl>,
## #   DIC_mM <dbl>, Meas_Zn_nM <dbl>
```

I will select the response variable (measured Zinc) and its 10 predictors (temperature, pH, DOC , DIC, Ca, Mg, Na, K, SO4, Cl)

```r
library(dplyr)

# Select variables
subset_daily_avg <- daily_avg %>%
  select(Date, Meas_Zn_nM, Temp_C, pH, DOC_M, DIC_mM, Ca_uM, Mg_uM, Na_uM, K_uM,
         SO4_uM, Cl_uM)

# Preview
head(subset_daily_avg)
```

```
## # A tibble: 6 x 12
##   Date       Meas_Zn_nM Temp_C    pH     DOC_M DIC_mM Ca_uM Mg_uM Na_uM  K_uM
##   <date>          <dbl>  <dbl> <dbl>     <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2023-10-20       90.3   15.6  7.57 0.0000801   4.26 1556.  494.  128.  27.3
## 2 2023-10-21       91.1   15.5  7.57 0.0000787   4.27 1561.  495.  129.  27.3
## 3 2023-10-22       92.0   15.5  7.56 0.0000770   4.28 1566.  497.  129.  27.3
## 4 2023-10-23       93.0   15.4  7.56 0.0000755   4.30 1571.  498.  129.  27.3
## 5 2023-10-24       93.9   15.3  7.55 0.0000740   4.31 1576.  498.  129.  27.3
## 6 2023-10-25       95.0   15.3  7.55 0.0000726   4.31 1580.  499.  130.  27.4
## # i 2 more variables: SO4_uM <dbl>, Cl_uM <dbl>
```

Exploratory plots of each variable

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v stringr 1.5.1
## v ggplot2 3.5.2     v tibble  3.3.0
## v purrr   1.0.4     v tidyr   1.3.1
## v readr   2.1.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
# Pivot only numeric columns
subset_long <- subset_daily_avg %>%
  pivot_longer(
    cols = where(is.numeric),      # only numeric variables
    names_to = "Variable",
    values_to = "Value"
  ) %>%
  mutate(Source = "Original")       # placeholder, can add "Interpolated" if you have that

# Plot
variable_plots <- ggplot(subset_long, aes(x = Date, y = Value)) +
  geom_point(data = filter(subset_long, Source == "Original"), size = 1.5, alpha = 0.8) +
  # geom_line(data = filter(subset_long, Source == "Interpolated"), linewidth = 0.8) +
  facet_wrap(~ Variable, scales = "free_y", ncol = 2) +
  scale_x_date(
    date_breaks = "1 month",
    date_labels = "%b\n%Y"
  ) +
  labs(
    x = "Date",
    y = "Value",
    color = "Data Source",
    title = "Daily Averages of Bioavailable Variables"
  ) +
  theme_bw(base_size = 11) +
  theme(
    legend.position = "top",
    strip.background = element_rect(fill = "gray90"),
    strip.text = element_text(face = "bold"),
    panel.grid.minor = element_blank(),
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.spacing = unit(1.0, "lines")
  )

# Export
ggsave("Variable_plots.png", plot = variable_plots)
```

```
## Saving 6.5 x 4.5 in image
```

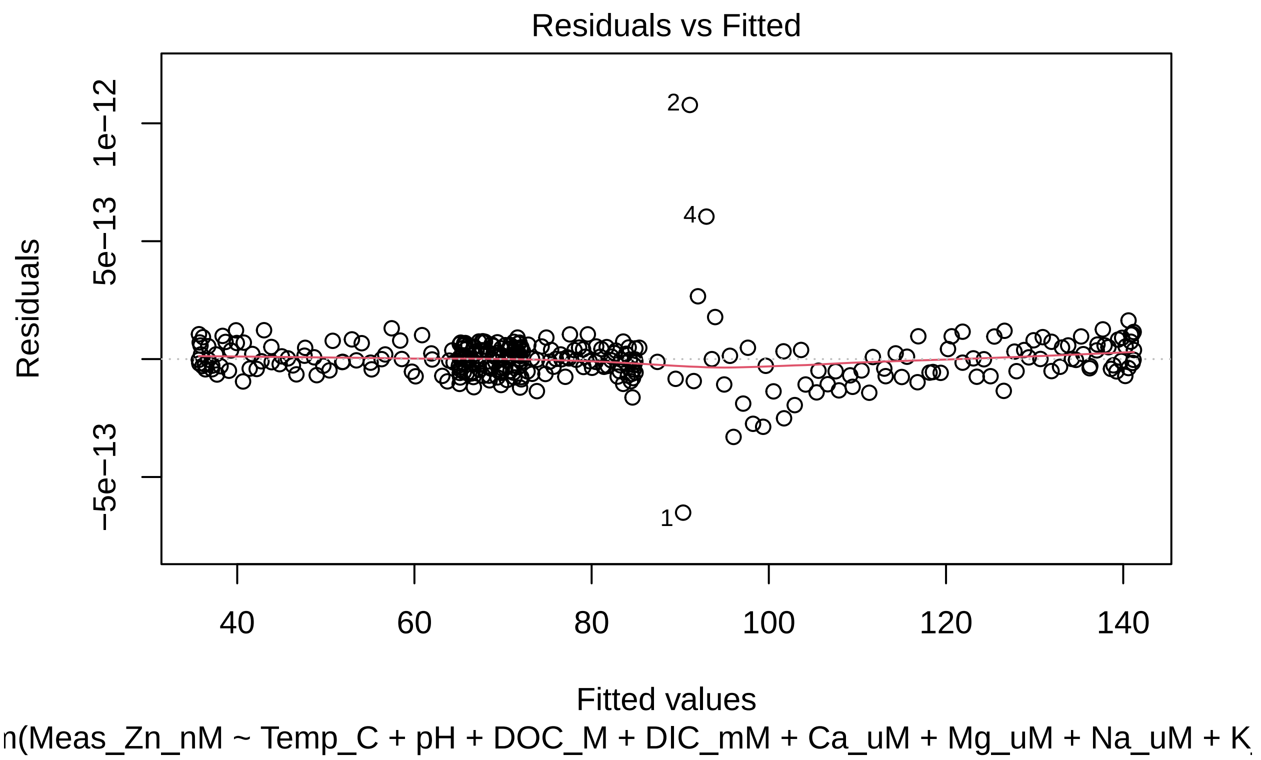
The number of observations (n) is large, as shown below

```r
n <- length(subset_daily_avg$Date)
n
```

```
## [1] 327
```

Check for heteroskedasticity

```r
model <- lm(Meas_Zn_nM ~ Temp_C + pH + DOC_M + DIC_mM + Ca_uM + Mg_uM + Na_uM + K_uM + SO4_uM + Cl_uM, d

# Base R diagnostic plot
diagnostic_plot <- plot(model, which = 1)
```



```r
# View
diagnostic_plot
```

```
## NULL
```

```r
# Notes: generally homoscedastic with a couple outliers
```

Summary table of variables

```r
# Remove date column
subset_daily_avg$Date <- NULL

library(tibble)

summary_table <- tibble(
  Variable = c("y_t", "x_1t", "x_2t", "x_3t", "x_4t", "x_5t", "x_6t",
               "x_7t", "x_8t", "x_9t", "x_10t"),
  ID = colnames(subset_daily_avg),
  Description = c("Measured Zinc",
                  "Temp",
                  "pH",
                  "Dissolved organic carbon",
                  "Dissolved inrganic carbon",
                  "Calcium",
                  "Magnesium",
                   "Sodium",
                   "Potassium",
                   "Sulfate",
                   "Chloride"),
  Unit = c("uM", "Celsius", "", "M", "mM", "uM", "uM", "uM", "uM", "uM", "uM")
)

# View
summary_table
```

```
## # A tibble: 11 x 4
##    Variable ID         Description               Unit
##    <chr>    <chr>      <chr>                     <chr>
##  1 y_t      Meas_Zn_nM Measured Zinc             "uM"
##  2 x_1t     Temp_C     Temp                      "Celsius"
##  3 x_2t     pH         pH                        ""
##  4 x_3t     DOC_M      Dissolved organic carbon  "M"
##  5 x_4t     DIC_mM     Dissolved inrganic carbon "mM"
##  6 x_5t     Ca_uM      Calcium                   "uM"
##  7 x_6t     Mg_uM      Magnesium                 "uM"
##  8 x_7t     Na_uM      Sodium                    "uM"
##  9 x_8t     K_uM       Potassium                 "uM"
## 10 x_9t     SO4_uM     Sulfate                   "uM"
## 11 x_10t    Cl_uM      Chloride                  "uM"
```

```
# Export
library(writexl)
write_xlsx(summary_table, path = "Summary_table_variables.xlsx")
```

Translation of Matlab Code to R using ChatGPT

In this section I had chatGPT translate the Matlab module given by the instructor into R code.

This section will

- Enumerate all possible subsets of predictors (2^p models, p <= 10);
- Fit each subset model via OLS and compute HAC covariance matrices;
- Calculate AIC, SBC, and CICOMP for each subset;
- Export a .csv, .tex, and .rtf summary of the best model(s);
- Generate comparative plots illustrating model complexity vs. information criterion.
- Which Kernel Covariance is the best choice for your dataset? Hint: You can compare the score of the criteria on all the variables, saturated model.

```
library(tidyverse)
library(combinat)  # for generating all subsets
```

```
##
## Attaching package: 'combinat'
## The following object is masked from 'package:utils':
##
##     combn
```

```
library(sandwich)  # HAC covariance
library(lmtest)    # coeftest
```

```
## Loading required package: zoo
```

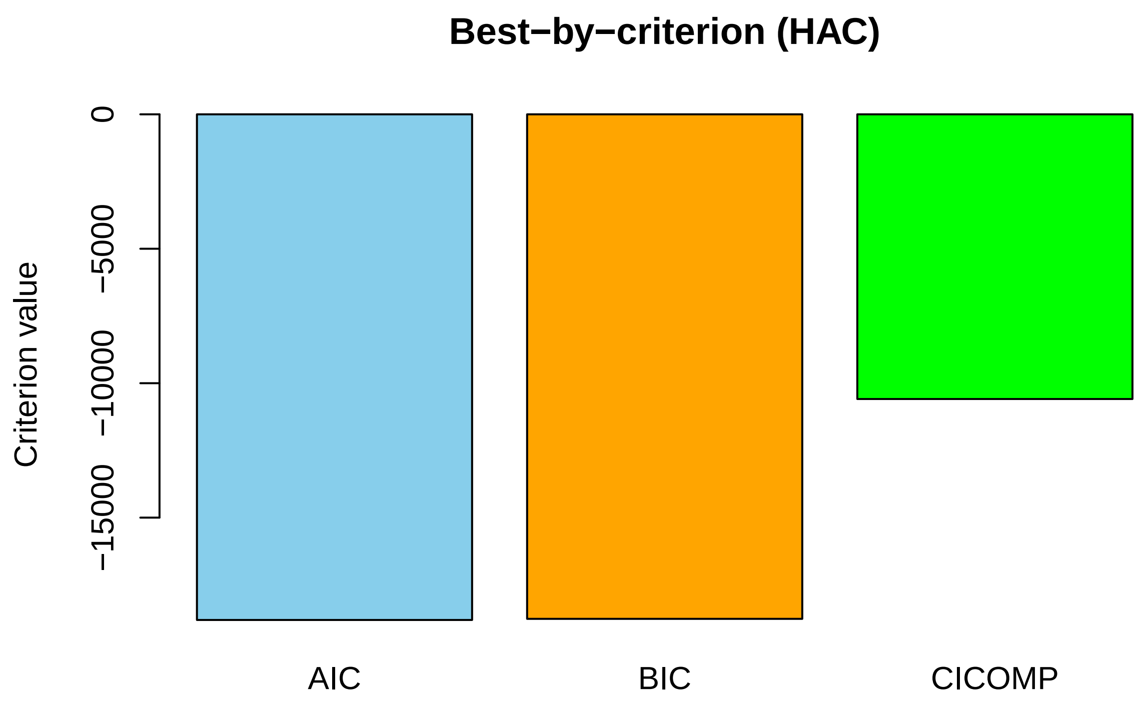
```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# ---------------- User controls ----------------
data <- subset_daily_avg
y_var <- "Meas_Zn_nM"
predictors <- setdiff(colnames(data), y_var)  # exclude response
addIntercept <- TRUE
topN <- 1
hacLag <- NULL     # auto lag if NULL

# ---------------- Enumerate all subsets ----------------
p <- length(predictors)
all_subsets <- unlist(lapply(0:p, function(k) combn(p, k, simplify=FALSE)), recursive=FALSE)

results <- vector("list", length(all_subsets))
T_obs <- nrow(data)
hacLag_auto <- ifelse(is.null(hacLag), floor(4*(T_obs/100)^(2/9)), hacLag)

for(i in seq_along(all_subsets)){
  idx <- all_subsets[[i]]
  X_vars <- predictors[idx]

  formula_str <- if(length(X_vars)==0) paste0(y_var," ~ 1") else paste0(y_var," ~ ",paste(X_vars,collap
  fit <- lm(as.formula(formula_str), data=data)

  # ---------------- HAC covariance ----------------
  cov_HAC <- tryCatch({
    sandwich::vcovHAC(fit, prewhite=FALSE, adjust=TRUE)
  }, error=function(e) vcov(fit))  # fallback to standard covariance

  # ---------------- Residual variance ----------------
  e <- resid(fit)
  sig2 <- sum(e^2)/T_obs
  k <- length(coef(fit))

  # ---------------- AIC / BIC ----------------
  AICv <- AIC(fit)
  BICv <- BIC(fit)
  # ---------------- CIC metric ----------------
  ev <- eigen(cov_HAC, symmetric=TRUE, only.values=TRUE)$values
  evbar <- mean(ev)
  if(!is.finite(evbar) || evbar <= 0) evbar <- mean(ev[ev>0])
  C1F <- (1/(4*(evbar^2))) * sum((ev - evbar)^2)
  CICv <- T_obs*log(2*pi) + T_obs*log(max(sig2, .Machine$double.eps)) + T_obs + k + 2*log(T_obs)*C1F

  results[[i]] <- tibble(
    Size = length(X_vars),
    Indices = if(length(X_vars)==0) "" else paste(X_vars, collapse=","),
    AIC = AICv,
    BIC = BICv,
    CICOMP = CICv
  )

  if(i %% 100 == 0) message("Processed subset ", i, " / ", length(all_subsets))
}
```

```
## Processed subset 100 / 1023

## Processed subset 200 / 1023

## Processed subset 300 / 1023

## Processed subset 400 / 1023

## Processed subset 500 / 1023

## Processed subset 600 / 1023

## Processed subset 700 / 1023

## Processed subset 800 / 1023

## Processed subset 900 / 1023

## Processed subset 1000 / 1023

## Warning in summary.lm(x): essentially perfect fit: summary may be unreliable
```

```
res_df <- bind_rows(results)

# ---------------- Top-N subsets ----------------
TopAIC  <- res_df %>% arrange(AIC) %>% slice(1:topN)
TopBIC  <- res_df %>% arrange(BIC) %>% slice(1:topN)
TopCIC  <- res_df %>% arrange(CICOMP) %>% slice(1:topN)

# ---------------- Export CSV ----------------
write_csv(res_df, paste0("ALL_SUBSETS_HAC_", Sys.Date(), ".csv"))
write_csv(bind_rows(
  TopAIC %>% mutate(Criterion="AIC"),
  TopBIC %>% mutate(Criterion="BIC"),
  TopCIC %>% mutate(Criterion="CICOMP")
), paste0("TOP_SUBSETS_HAC_", Sys.Date(), ".csv"))

# ---------------- Quick bar plot ----------------
best_vals <- c(AIC=TopAIC$AIC[1], BIC=TopBIC$BIC[1], CICOMP=TopCIC$CICOMP[1])
barplot(best_vals, main="Best-by-criterion (HAC)",
        ylab="Criterion value", col=c("skyblue","orange","green"))
```



```
# ---------------- Trajectories vs model size ----------------
sz <- 0:p
bestBySize_AIC <- sapply(sz, function(m) min(res_df$AIC[res_df$Size==m], na.rm=TRUE))
```

```
## Warning in min(res_df$AIC[res_df$Size == m], na.rm = TRUE): no non-missing
## arguments to min; returning Inf
```

```
bestBySize_BIC <- sapply(sz, function(m) min(res_df$BIC[res_df$Size==m], na.rm=TRUE))
```

```
## Warning in min(res_df$BIC[res_df$Size == m], na.rm = TRUE): no non-missing
## arguments to min; returning Inf
```

```r
bestBySize_CIC <- sapply(sz, function(m) min(res_df$CICOMP[res_df$Size==m], na.rm=TRUE))
```

```
## Warning in min(res_df$CICOMP[res_df$Size == m], na.rm = TRUE): no non-missing
## arguments to min; returning Inf
```

```r
matplot(sz, cbind(bestBySize_AIC, bestBySize_BIC, bestBySize_CIC), type="b", pch=1:3, lty=1,
        xlab="Subset size", ylab="Best value at size", col=c("blue","red","green"))
legend("topright", legend=c("AIC","BIC","CICOMP"), col=c("blue","red","green"), pch=1:3)
```

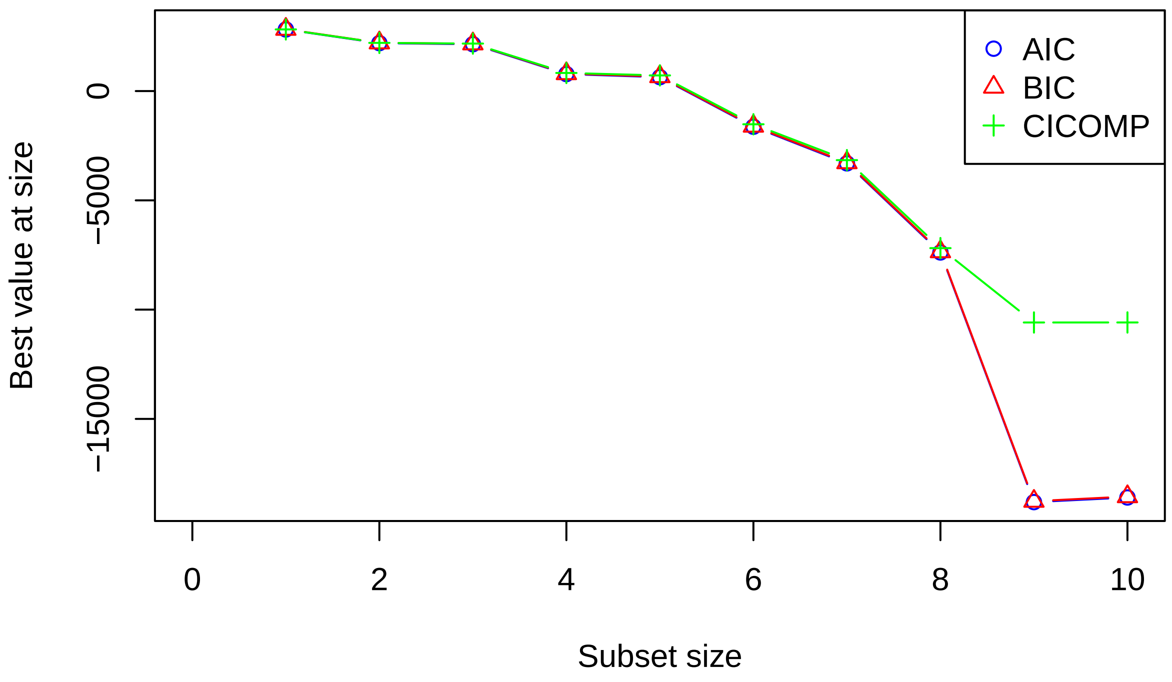


Table of information criteria

```r
library(dplyr)
library(stringr)

topN <- 3

# Combine top-N subsets for each criterion
TopAll <- bind_rows(
  TopAIC %>% slice_min(AIC, n = topN) %>% mutate(Criterion="AIC", Information_Criteria=AIC),
  TopBIC %>% slice_min(BIC, n = topN) %>% mutate(Criterion="BIC", Information_Criteria=BIC),
  TopCIC %>% slice_min(CICOMP, n = topN) %>% mutate(Criterion="CICOMP", Information_Criteria=CICOMP)
)

# Clean up variable names and handle intercept-only models
TopAll <- TopAll %>%
  rowwise() %>%
  mutate(Variables = if (Indices == "" | is.na(Indices)) "(Intercept only)" else Indices) %>%
  select(Criterion, Size, Variables, Information_Criteria)

# Print table
ic_table <- print(TopAll, n = nrow(TopAll))
```

```
## # A tibble: 3 x 4
## # Rowwise:
##   Criterion  Size Variables                                 Information_Criteria
##   <chr>     <int> <chr>                                                    <dbl>
## 1 AIC           9 pH,DOC_M,DIC_mM,Ca_uM,Mg_uM,Na_uM,K_uM,S~               -18806.
## 2 BIC           9 pH,DOC_M,DIC_mM,Ca_uM,Mg_uM,Na_uM,K_uM,S~               -18764.
## 3 CICOMP        9 Temp_C,pH,DOC_M,DIC_mM,Ca_uM,Mg_uM,Na_uM~               -10588.
```

```r
# Export
library(writexl)
write_xlsx(ic_table, path = "Information_criteria_table.xlsx")
```

Discussion and Conclusions

Across all three information criteria (AIC, SBC, and CICOMP), the optimal model size was consistently identified as containing nine out of ten predictor variables. This agreement suggests that including these nine variables captures the majority of the explainable variance in the dataset, while adding a tenth predictor does not yield meaningful improvement in model performance. In practical terms, the model with nine predictors achieves a balance between explanatory power and parsimony.

The AIC and SBC scores were nearly identical and substantially more negative than the CICOMP score. This difference arises because AIC and SBC both penalize model complexity primarily based on the number of parameters, whereas CICOMP (Covariance Complexity Information Criterion) imposes an additional penalty related to the structure of the covariance matrix of the model residuals. In other words, CICOMP explicitly accounts for the *interdependence* among model parameters rather than just their count.

The fact that the CICOMP score was considerably less negative (i.e., closer to zero) indicates that the model's residual covariance matrix is relatively complex. This suggests that the predictors may exhibit substantial collinearity or shared information, leading to a more intricate covariance structure. In time-series contexts, this can also reflect autocorrelation or cross-dependence among predictors and response variables over time.

Interestingly, AIC—which typically applies the least severe penalty—produced the most negative criterion value, while CICOMP—which applies a stronger, structure-based penalty—produced a substantially greater (less negative) value. Although direct numerical comparison among information criteria is not meaningful because they are based on different scales and formulations, this contrast underscores how differently each metric balances goodness-of-fit versus structural simplicity.

While all three criteria favored a nine-variable model, the discrepancy in magnitude between AIC/SBC and CICOMP emphasizes that the selected model, though statistically optimal in fit, may still possess a complex covariance structure. This complexity may be driven by correlations among chemical parameters (e.g., major cations and dissolved carbon species), which often covary in natural systems and thus contribute to multicollinearity.

Model parsimony refers to achieving the simplest model that adequately explains the data. In this case, all three selection criteria converging at nine predictors supports the notion that additional predictors do not enhance model explanatory power and may even contribute unnecessary complexity. However, covariance complexity—as measured by CICOMP—reveals that even a parsimonious model can exhibit substantial structural interdependence if predictor variables are correlated.

In essence, covariance complexity quantifies the degree of collinearity and parameter interdependence within the model. A high covariance complexity value implies that parameter estimates are not statistically independent, meaning that changes in one predictor's effect are partly compensated by others. This reduces interpretability and may limit model generalization to new data. Therefore, while the nine-variable model is optimal in terms of information criteria, it may still benefit from dimensionality reduction (e.g., PCA) or diagnostic evaluation (e.g., VIF analysis) to ensure model stability.

From a scientific perspective, this structural complexity is likely reflecting the biogeochemical coupling among predictors (e.g., pH–cation–DOC–DIC relationships) rather than mere statistical redundancy. Thus, the observed covariance complexity is not necessarily problematic—it may represent the natural interconnectivity of processes in the system being modeled. Still, acknowledging and quantifying this complexity helps clarify that while the model performs well statistically, interpretability and mechanistic inference must be approached with caution.