Reproduction Code for 2025 Research Prelim Exam Report

Wenhao Pan

2025-06-12

Variable name interpretation

Here is the interpretation of the variable names in the Asadi et al.'s (2015) code

- `StsEucCoords`: Plain Euclidean locations
- `CatCntrWt`: Hydrological locations $H(t)$.
- `EucDis`: Euclidean distances between gauging stations.
- `RiverDis`: River distances between gauging stations, $d(s, t)$.
- `Weight`: River weights, π_k's.
- `FlowCon`: Flow-connection status of all station pairs.
- `StsInfo`: River name, latitude, longtitude, and average volume of each gauging station.
- `StsTSs`: Average daily discharge of each station from 1900 to 2014 with missing data.
- `ComTSs`: Common measurements (daily measurement in June, July, and August during 1960 - 2010) at all stations.
- `CommonMaxima`: Annual maxima of common measurements at all stations.
- `ThetaOfBlockMaxima`: Madogram-based estimated extremal coefficients for all pairs of gauging stations

Data verification and preprocessing

```
library(ggplot2)
library(tidyr)
library(dplyr)
root.dir <- rprojroot::find_rstudio_root_file()
source(paste(root.dir, "/Codes/Functions.R", sep=""))

# Loading the data provided by the original authors
FileList <- list.files(path = "C:/Code/Prelim/Data", pattern = "\\.RData$")
# FileList <- list.files(path = "D:/Code/Prelim/Data")
for (i in 1:length(FileList)){
  load(paste("C:/Code/Prelim/Data", FileList[i], sep="/"))
  # load(paste("D:/Code/Prelim/Data", FileList[i], sep="/"))
}
StsChos <- c(1:31) # Station indices
NoSt <- length(StsChos) # Total number of stations
```

```
# check the range of daily average discharges at each station
sapply(StsTSs, function(x) mean(x$Val))
```

```
# check CommonMaxima
ComTSs.copy <- data.frame(ComTSs)
colnames(ComTSs.copy)[2:32] <- as.character(1:31)
CommonMaxima.test <- ComTSs.copy %>%
```

```r
  pivot_longer(
    -Date,
    names_to = "Station",
    values_to = "Discharge"
  ) %>%
  mutate(Station = as.integer(Station), Year = format(Date, "%Y")) %>%
  filter(Year <= 2009) %>%
  group_by(Station, Year) %>%
  summarize(AnnualMax = max(Discharge, na.rm = TRUE), .groups = "drop") %>%
  arrange(Station, Year)

# should return true
all(CommonMaxima.test$AnnualMax == as.vector(CommonMaxima))
```

Plotting

```r
library(ggplot2)
library(tidyr)
library(dplyr)
root.dir <- rprojroot::find_rstudio_root_file()
source(paste(root.dir, "/Codes/Functions.R", sep=""))

# Loading the data provided by the original authors
FileList <- list.files(path = "C:/Code/Prelim/Data", pattern = "\\.RData$")
# FileList <- list.files(path = "D:/Code/Prelim/Data")
for (i in 1:length(FileList)){
  load(paste("C:/Code/Prelim/Data", FileList[i], sep="/"))
  # load(paste("D:/Code/Prelim/Data", FileList[i], sep="/"))
}
StsChos <- c(1:31) # Station indices
NoSt <- length(StsChos) # Total number of stations

# obtain declustered events
Years <- unique(as.numeric(substr(ComTSs[, 1], 1, 4)))
U <- 0.1
Lag <- 4
Plotting <- 0
StNames <- as.character(StsChos)
YearsWithEvent <- numeric()
AllEvMat <- matrix(0, length(StNames), 1)
AllRes <- list()
for (i in 1:length(Years)){
  cat(paste("Declustering year", Years[i], "\n"))
  Res <- ObtainMultVarEvents(
    TSs = ComTSs, U = U, Lag = Lag, Year = Years[i],
    StNames = StNames, Plotting = Plotting, mfrow = c(NoSt,1))  ## For Censored Likelihood
  Events <- Res$AllEvMat
  Locations <- Res$Location
  if (length(Events) > 0) {
    YearsWithEvent <- c(YearsWithEvent, rep(Years[i], dim(Events)[2]))
    AllEvMat <- cbind(AllEvMat, Events)
  }
```

```r
  AllRes[[i]] <- Res
}
DataEvents <- t(AllEvMat[, -1])
rownames(DataEvents) <- YearsWithEvent

# normalize data to standard Pareto margins
TSNew <- matrix(0, dim(DataEvents)[1], dim(DataEvents)[2])
for (i in 1:dim(DataEvents)[2]) {
  TSNew[, i] <- Margin2Pareto(DataEvents[, i], empirical = TRUE)
}

# transform estimates to the scale used in the paper
par2paper <- function(par)
  return(c(2 * par[4], par[5] / par[3], par[1] * 500, par[2], par[6], par[7]))
```

Plots in the prelim exam report

Figure 3: River distance vs Euclidean distance

```r
# Remove diagonal entries (same as before)
EucDis_no_diag <- EucDis[lower.tri(EucDis)]
RiverDis_no_diag <- RiverDis[lower.tri(RiverDis)]

# Create a data frame for plotting
df <- data.frame(Euclidean = EucDis_no_diag, River = RiverDis_no_diag)

# Create the ggplot
ggplot(df, aes(x = Euclidean, y = River)) +
  geom_point(color = "#2E86AB", alpha = 0.6, size = 1.5) +
  geom_abline(intercept = 0, slope = 1,
              color = "#E63946", linetype = "dashed", linewidth = 1) +
  labs(
    title = "Comparison of Euclidean and River Distances",
    subtitle = "Each point represents a pair of locations",
    x = "Euclidean Distance (km)",
    y = "River Distance (km)"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 11, hjust = 0.5, color = "gray50"),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10),
    panel.grid.minor = element_blank(),
    panel.border = element_rect(color = "gray80", fill = NA, linewidth = 0.5)
  ) +
  # coord_fixed(ratio = 1) +  # Equal aspect ratio
  annotate("text", x = 0.05, y = 0.95,
           label = "y = x", color = "#E63946", size = 3.5, fontface = "italic")
ggsave(paste(root.dir, "/Plots/river_vs_euc.png", sep=""))
```

Figure 6

```
# check that most extreme discharges happen in summer

# station 1 in 2005
station.1 <- as.data.frame(StsTSs[1])
station.1$Date <- as.Date(station.1$Date)
ggplot(station.1, aes(x = Date, y = Val)) +
  annotate("rect",
           xmin = as.Date("2005-06-01"),
           xmax = as.Date("2005-08-31"),
           ymin = -Inf, ymax = Inf,
           alpha = 0.2, fill = "skyblue") +
  geom_line() +
  scale_x_date(
    date_breaks = "1 month", date_labels = "%b",
    limits = as.Date(c("2005-01-01", "2005-12-31"))) +
  theme_minimal() +
  theme(text = element_text(size = 16)) +
  labs(
    title = "Average daily dicharge of gauging station 1 in 2005",
    x = "Time", y = "Average daily dicharge")
ggsave(paste(root.dir, "/Plots/station_1_2005.png", sep=""))

# station 19 in 2002
station.19 <- as.data.frame(StsTSs[19])
station.19$Date <- as.Date(station.19$Date)
ggplot(station.19, aes(x = Date, y = Val)) +
  annotate("rect",
           xmin = as.Date("2002-06-01"),
           xmax = as.Date("2002-08-31"),
           ymin = -Inf, ymax = Inf,
           alpha = 0.2, fill = "skyblue") +
  geom_line() +
  scale_x_date(
    date_breaks = "1 month", date_labels = "%b",
    limits = as.Date(c("2002-01-01", "2002-12-31"))) +
  theme_minimal() +
  theme(text = element_text(size = 16)) +
  labs(
    title = "Average daily dicharge of gauging station 19 in 2002",
    x = "Time", y = "Average daily dicharge")
ggsave(paste(root.dir, "/Plots/station_19_2002.png", sep=""))
```

Figure 7: declustering

```
# check declustering
PlottingDeclusterdEvents(Year = 1989,
                         Stations = c(1, 2, 11, 13, 15),
                         AllRes = AllRes,
                         ComTSs = ComTSs,
                         StsInfo = StsInfo,
                         filename = paste(root.dir, "/Plots/Declustering.pdf", sep=""))
```

Plots in the original paper (Asadi et al. 2015)

Figure 2

```
par  <- fitM4$par
# Parameters of the transformation matrix
beta <- par[6]
c <- par[7]

rot.mat <- cbind(c(cos(beta), c*sin(beta)),c(-sin(beta), c*cos(beta))) # Transformation matrix
Catch.dist <- as.matrix(dist(CatCntrWt %*% t(rot.mat)))  # Hydrological distance with the transformatio


plotEucVsHyd(ECemp = ThetaOfBlockMaxima,
             ECtheo = ThetaOfBlockMaxima,
             DistEuc = EucDis,
             DistCatch = Catch.dist,
             is.con = FlowCon,
             StsIdx = StsChos,
             which.plots = c(TRUE,TRUE,FALSE),
             which.labels = c(FALSE,FALSE,TRUE),
             PDF = TRUE,
             filename = paste(root.dir, "/Plots/EucVSCatch.pdf", sep=""))
```

Figure 5

```
PlottingDeclusterdEvents(Year = 1989,
                         Stations = c(1, 2, 11, 13),
                         AllRes = AllRes,
                         ComTSs = ComTSs,
                         StsInfo = StsInfo,
                         filename = paste(root.dir, "/Plots/MultiEv1.pdf", sep=""))
```

Figure 7

```
# left
ScSts <- c(2,5)
filename <- paste(
  paste(as.character(ScSts[1]), as.character(ScSts[2]), sep="-"),
  ".pdf",
  sep="")

ScatterPlot(ScSts = ScSts,
            StsInfo,
            X = TSNew,
            filename = paste(root.dir, "Plots", filename, sep="/"))

# right
ScSts <- c(13,16)

filename <- paste(
  paste(as.character(ScSts[1]), as.character(ScSts[2]), sep="-"),
  ".pdf",
```

```
  sep="")

ScatterPlot(ScSts = ScSts,
            StsInfo,
            X=TSNew,
            filename = paste(root.dir, "Plots", filename, sep="/"))
```

Figure 8

```
# compute EC for HR model with censored estimation
Theta.cen <- matrix(1, ncol = NoSt, nrow = NoSt)

for(i in 1:(NoSt-1))
  for(j in (i+1):NoSt) {
    Data.biv <- DataEvents[,c(i,j)]
    Theta.cen[i,j] <- CensoredEstimationHR(Data.biv, thresholds = .90)
    Theta.cen[j,i] <- Theta.cen[i,j]
  }

# extract kernel parameters
par  <- fitM4$par
beta <- par[6]
c <- par[7]

rot.mat <- cbind(c(cos(beta), c*sin(beta)),c(-sin(beta), c*cos(beta))) # Transformation matrix
Catch.dist <- as.matrix(dist(CatCntrWt %*% t(rot.mat)))  # Hydrological distance with the transformatio
euc.dist <- as.matrix(dist(CatCntrWt %*% t(rot.mat)))

# left
plotECF(ECemp = ThetaOfBlockMaxima,
        ECtheo = Theta.cen,
        Dist = CatchEucDisWt,
        is.con = FlowCon,
        StsIdx  = StsChos,
        which.plots = c(FALSE,FALSE,TRUE),
        which.labels = c(FALSE, FALSE,FALSE),
        PDF = TRUE,
        filename = paste(root.dir, "/Plots/ECF_Emp_HR.pdf", sep=""))

# center
plotECF(ECemp = ThetaOfBlockMaxima,
        ECtheo = Vario2EC(
            vario(par,
                  riv.dist = RiverDis,
                  euc.coords = CatCntrWt,
                  riv.weights = Weight
                  )
            ),
        Dist = euc.dist,
        is.con = FlowCon,
        StsIdx  = StsChos,
        which.plots = c(FALSE,TRUE,FALSE),
        which.labels = c(FALSE, FALSE,FALSE),
```

```
        PDF = TRUE,
        filename = paste(root.dir, "/Plots/EmpVSModel2.pdf", sep=""))


# right
plotECF(ECemp = ThetaOfBlockMaxima,
        ECtheo = Vario2EC(
            vario(par,
                  riv.dist = RiverDis,
                  euc.coords = CatCntrWt,
                  riv.weights = Weight
                  )
            ),
        Dist = euc.dist,
        is.con = FlowCon,
        StsIdx  = StsChos,
        which.plots = c(FALSE,FALSE,TRUE),
        which.labels = c(FALSE, FALSE,FALSE),
        PDF = TRUE,
        filename = paste(root.dir, "/Plots/EmpVSModel3.pdf", sep=""))
```

Figure 9

```
require("SpatialExtremes")
require("mvtnorm")
library("ismev")
Maxima <- CommonMaxima

# N=NoSt=31 reproduce the bottom-right plot of Fig. 9. For other random groups, change N
N <- 31
StsForPlot <- sort(sample(StsChos,N))

# transformation to Unit Frechet

TransMax <- matrix(0,nrow(Maxima),N)
for (i in 1:N){
    Pars <- gev.fit(Maxima[,StsForPlot[i]])$mle
    TransMax[,i] <- gev2frech(Maxima[,StsForPlot[i]], Pars[1], Pars[2], Pars[3], emp = FALSE)
}

# here the data quantiles are computed ###
block.size <- 1
nb.blocks <- nrow(Maxima)
TSmax <- double(nb.blocks)
TSmax <- apply(TransMax,1,max)
quants.TSmax <- sort(TSmax)

# here the model quantiles are computed ###
d <- ncol(TransMax)
V <- function(x, par){
    d <- length(x)
    varioHalf <- 1/2 * vario(par,
                              riv.dist = RiverDis[StsForPlot,StsForPlot],
```

```
                              euc.coords = CatCntrWt[StsForPlot,],
                              riv.weights = Weight[StsForPlot,StsForPlot]
                              )

    f1 <- function(i,x){
        vario0 <- matrix(rep(varioHalf[-i,i],d-1),ncol = d-1)
        S <- vario0 + t(vario0) - varioHalf[-i,-i]
        return(1/x[i]*pmvnorm(upper=(log(x/x[i])+varioHalf[,i])[-i],mean=rep(0,d-1),sigma= S)[1])
    }
    return(sum(apply(cbind(1:d),1,f1,x=x)))
}


quant.level <- ((1:nb.blocks) - 1/2) / nb.blocks

V11 <- V(x = rep(1, times = d),par= fitM4$par)

quants.modelMax <-  - V11 * 1 / log(quant.level)

quants.indep <- - length(StsForPlot) * 1 / log(quant.level) ## quantiles if all points were independent
quants.dep <- - 1 / log(quant.level) ## quantiles if all points complete dependent

pdf(paste(root.dir, "/Plots/QQMaxima_", length(StsForPlot),".pdf", sep=""))
par(cex = 1.3, cex.lab = 2.3, cex.axis = 2.3, cex.main = 1.5, pch = 19,
    mar = c(5,5,4,2) +.1)
plot(log(quants.modelMax),log(quants.TSmax),xlab="Model Quantile",ylab="Data Quantile")
abline(a = 0, b =1,col="blue")
lines(log(quants.modelMax), log(quants.indep),col="blue",lty=2)
lines(log(quants.modelMax), log(quants.dep),col="blue",lty=3)
dev.off()
```

Fitting

```
library(ismev)
library(evd)
library(ggplot2)
library(tidyr)
library(dplyr)
root.dir <- rprojroot::find_rstudio_root_file()
source(paste(root.dir, "/Codes/Functions.R", sep=""))

# Loading the data provided by the original authors
FileList <- list.files(path = "C:/Code/Prelim/Data", pattern = "\\.RData$")
# FileList <- list.files(path = "D:/Code/Prelim/Data")
for (i in 1:length(FileList)){
  load(paste("C:/Code/Prelim/Data", FileList[i], sep="/"))
  # load(paste("D:/Code/Prelim/Data", FileList[i], sep="/"))
}
StsChos <- c(1:31) # Station indices
NoSt <- length(StsChos) # Total number of stations

# obtain declustered events
```

```
Years <- unique(as.numeric(substr(ComTSs[, 1], 1, 4)))
U <- 0.1
Lag <- 4
Plotting <- 0
StNames <- as.character(StsChos)
YearsWithEvent <- numeric()
AllEvMat <- matrix(0, length(StNames), 1)
AllRes <- list()
for (i in 1:length(Years)){
  cat(paste("Declustering year", Years[i], "\n"))
  Res <- ObtainMultVarEvents(
    TSs = ComTSs, U = U, Lag = Lag, Year = Years[i],
    StNames = StNames, Plotting = Plotting, mfrow = c(NoSt,1))  ## For Censored Likelihood
  Events <- Res$AllEvMat
  Locations <- Res$Location
  if (length(Events) > 0) {
    YearsWithEvent <- c(YearsWithEvent, rep(Years[i], dim(Events)[2]))
    AllEvMat <- cbind(AllEvMat, Events)
  }
  AllRes[[i]] <- Res
}
DataEvents <- t(AllEvMat[, -1])
rownames(DataEvents) <- YearsWithEvent

# normalize data to standard Pareto margins
TSNew <- matrix(0, dim(DataEvents)[1], dim(DataEvents)[2])
for (i in 1:dim(DataEvents)[2]) {
  TSNew[, i] <- Margin2Pareto(DataEvents[, i], empirical = TRUE)
}

# transform estimates to the scale used in the paper
par2paper <- function(par)
  return(c(2 * par[4], par[5] / par[3], par[1] * 500, par[2], par[6], par[7]))
```

Marginal fitting

```
### GEV analysis  ###
method <- "BFGS"
control <- list(maxit = 1000, reltol=10^(-30), abstol=0.0001, trace=2)

Maxima <- SummerMaxima

ResGev <- list()
GevPars <- matrix(0, NoSt, 3)
FreeGevNLLTemp <- numeric()
for (i in 1:NoSt){
    ResGev[[i]] <- gev.fit(Maxima[[i]])
    GevPars[i,] <- ResGev[[i]]$mle
    FreeGevNLLTemp[i] <- ResGev[[i]]$nll
}

### Extraction of univariate events  ###
U <- 0.9
```

```
Lag <- 5
StNames <- as.character(1:31)
mfrow <- c(1,1)
YearsWithEvents <- list()
Threshold <- numeric()
AllEvents <- list()
NoOfYears <- numeric()
for (i in 1:NoSt) {
    TSs <- TSMonths[[i]]
    X <- TSMonths[[i]]$Val
    Threshold[i] <- quantile(X,U)
    Years <- unique(as.numeric(substr(TSs[,1],1,4)))
    Events <- numeric()
    YearEvents <- numeric()
    for (j in 1:length(Years)) {
        X1 <- ObtainMultVarEvents(TSs = TSMonths[[i]],U=U,Lag=Lag,Year=Years[j],StNames=StNames[i],Plot
        X2 <- as.numeric(X1$AllEvMat)
        Events  <- c(Events,X2)
        YearEvents <- c(YearEvents,rep(Years[j],length(X2)))
    }
    AllEvents[[i]] <- Events
    YearsWithEvents[[i]] <- YearEvents
    NoOfYears[i] <- length(Years)
}

### Fitting a GEVD to each station (by maximizing the joint Poisson process likelihood; cf. formula (21
PPRes <- list()
ParsPP <- matrix(0,NoSt,3)
PPLLFree <- numeric()
for (i in 1:NoSt) {
    Init <- GevPars[i,]
    PPRes[[i]] <- PPFit(Data = AllEvents[i], u = Threshold[i], NoYears = NoOfYears[i],
                        Init = Init, CovarMu = matrix(1,1,1), CovarSc = matrix(1,1,1),
                        CovarXi = matrix(1,1,1), LogModel = FALSE, method = method, control = control)
    ParsPP[i,] <- PPRes[[i]]$par
    PPLLFree[i] <- PPRes[[i]]$value
}

M1LL <- sum(PPLLFree)

### Regionalized model with covariates; cf. formula (32) ###
Grp1 <- c(11,12,16,17,18,19,21,22)
Grp2 <- c(13,28:31)
Grp3 <- c(1:10,14,15,20)
Grp4 <- c(23,24,25,26,27)
Grp <- list(Grp1,Grp2,Grp3,Grp4)

Lat <- StsCenter[,2]
CovarStsAll <- cbind(rep(1,length(StsArea)),StsArea,StsAlt,StsSlope,StsDensity,Lat)
PPCovariateModels <- list()
CovarLL <- numeric()
ParsCovModel <- matrix(0,31,3)
Normalized <- FALSE
```

```
CovarLog <- cbind(CovarStsAll[,1],log(CovarStsAll[,2:6]))

### The covariates which are siginficant, obtained by log-likelihood ratio test###
MuCovars <- list(c(2,3,4,6), c(2,4,6), c(2,3,4,6), c(2,4,6))
ScCovars <- list(c(2,3,4), c(2,4), c(2,3,4), c(2,4))
InitAll <- list(c(1.1, 0.9, 0.7, 1.2, 0, -0.1),
                c(0.6, 0.5, 0.4, 1.7, 0.1, -0.3),
                c(1.2, 0.7, 0.9, 0.7, -0.1, -0.5),
                c(0.1, 1, -0.1, 1.3, 0.5, 0.1))

for (i in 1:length(Grp)) {
    GrSts <- Grp[[i]]
    CovarMu <- CovarLog[,MuCovars[[i]]]
    InitMu <- InitAll[[i]][MuCovars[[i]]]
    CovarSc <- CovarLog[,ScCovars[[i]]]
    InitSc <- InitAll[[i]][ScCovars[[i]]]
    InitXi <- mean(ParsPP[GrSts,3])
    CovarMuReg <- CovarMu[GrSts,]
    CovarScReg <-  CovarSc[GrSts,]
    CovarXi <- as.matrix(CovarLog[GrSts ,1])
    Init <- c(InitMu,InitSc,InitXi)
    PPCovariateModels[[i]] <- PPFit(Data = AllEvents[GrSts], u = Threshold[GrSts],
                                    NoYears = NoOfYears[GrSts], Init = Init, CovarMu = CovarMuReg, Cova
                                    CovarXi = CovarXi, LogModel = TRUE, method = method, control = cont
    CovarLL[i] <- PPCovariateModels[[i]]$value
    ParsModelTemp <- PPCovariateModels[[i]]$par
    MuCov <- exp((CovarMuReg %*% ParsModelTemp [(1):(ncol(CovarMu))]))
    ScCov <- exp((CovarScReg %*%ParsModelTemp[(ncol(CovarMu)+1):(ncol(CovarMu)+ncol(CovarSc))]))
    XiCov <- as.matrix(CovarXi) %*%  ParsModelTemp[(ncol(CovarMu)+ncol(CovarSc)+1):length(ParsModelTemp
    ParsCovModel[Grp[[i]], ] <- cbind(MuCov, ScCov, XiCov)
}

M2LL <- sum(CovarLL)


M1AIC <- 2*(NoSt*3+M1LL)
M2AIC <- 2*(length(Grp)+length(unlist(ScCovars))+length(unlist(MuCovars))+M2LL)
```

Joint fitting

Kernel 1

```
# Define grids for each parameter
k = 5 # grid size
theta_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
alpha_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
s0 <- 289.73
riv_lmb <- 0
euc_lmb_grid <- seq(0.01, 0.1, by = (0.1 - 0.01) / (k - 1))
beta_grid <- seq(pi/4, 3*pi/4, by = (3*pi/4 - pi/4) / (k - 1))
c_grid <- seq(0.1, 1.0, by = (1.0 - 0.1) / (k - 1))

# Create all combinations of parameters
```

```r
param_grid <- expand.grid(
  theta = theta_grid,
  alpha = alpha_grid,
  euc_lmb = euc_lmb_grid,
  beta = beta_grid,
  c = c_grid
)
cat(paste("The number of parameter combinations for kernel model 1 is", nrow(param_grid), "\n"))

# Function to fit model with given initial values
fit_with_init <- function(init_params) {
  tryCatch({
    fit <- pareto_BR_River(
      data = TSNew,
      riv.dist = RiverDis,
      euc.coord = StsEucCoords,
      is.connected = FlowCon,
      riv.weights = Weight,
      u = quantile(rowSums(TSNew), 0.87),
      init = c(init_params$theta, init_params$alpha, s0,
               riv_lmb, init_params$euc_lmb,
               init_params$beta, init_params$c),
      model = 1,
      maxit = 1000,
      method = "BFGS"
    )
    return(list(params = init_params, fit = fit, nllik = fit$nllik,
                converged = TRUE))
  }, error = function(e) {
    return(list(params = init_params, fit = NULL, nllik = Inf,
                converged = FALSE))
  })
}

# Perform grid search
results <- list()
best_nllik <- Inf
best_fit <- NULL
best_init <- NULL

for(i in 1:nrow(param_grid)) {
  if (i %% 500 == 0) {
    cat("Fitting model", i, "of", nrow(param_grid), "\n")
  }

  result <- fit_with_init(param_grid[i, ])
  results[[i]] <- result

  # Track best result
  if(result$converged && result$nllik < best_nllik) {
    best_nllik <- result$nllik
    best_fit <- result$fit
    best_init <- result$params
```

```r
  }
}

# Extract results and find best fit
converged_results <- results[sapply(results, function(x) x$converged)]
nllik_values <- sapply(converged_results, function(x) x$nllik)

# Find best result
best_idx <- which.min(nllik_values)
best_result <- converged_results[[best_idx]]

# save or load the best result
saveRDS(best_result, file = paste(root.dir, "/Data/best_result_M1.rds", sep=""))
best_result <- readRDS(paste(root.dir, "/Data/best_result_M1.rds", sep=""))

cat("Best negative log-likelihood:", best_result$nllik, "\n")
cat("Best initial values of spectral estimation for kernel model 1:\n")
print(best_result$params)
cat("The fitted parameters of spectral estimation for kernel model 1:\n")
print(par2paper(best_result$fit$par))
```

```r
# censored estimation
fit.M1 <- pareto_BR_River_cen(
    data = TSNew,
    riv.dist = RiverDis,
    euc.coord = StsEucCoords,
    riv.weights = Weight,
    u = quantile(TSNew, .90),
    init = best_result$fit$par,
    model = 1,
    maxit = 200,
    method = "BFGS",
    )

# save or load fitted model
saveRDS(fit.M1, file = paste(root.dir, "/Data/fit_m1.rds", sep=""))
fit.M1 <- readRDS(paste(root.dir, "/Data/fit_m1.rds", sep=""))

# fitted negative log-likelihood
cat(paste("The fitted negative log-likelihood of kernel model 1 is", fit.M1$nllik, "\n"))

# transform estimates to the scale used in the paper
par <- fit.M1$par
Par.paper <- par2paper(par)
cat("Final fitted parameter (lam_riv, lam_euc, tau, alpha, beta, c) of kernel model 1:\n")
cat(Par.paper)
```

Kernel 2

```r
# Define grids for each parameter
k = 5 # grid size
theta_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
alpha_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
```

```r
s0 <- 289.73
riv_lmb <- 0
euc_lmb_grid <- seq(0.01, 0.1, by = (0.1 - 0.01) / (k - 1))
beta_grid <- seq(pi/4, 3*pi/4, by = (3*pi/4 - pi/4) / (k - 1))
c_grid <- seq(0.1, 1.0, by = (1.0 - 0.1) / (k - 1))

# Create all combinations of parameters
param_grid <- expand.grid(
  theta = theta_grid,
  alpha = alpha_grid,
  euc_lmb = euc_lmb_grid,
  beta = beta_grid,
  c = c_grid
)
cat(paste("The number of parameter combinations for kernel model 2 is", nrow(param_grid), "\n"))

# Function to fit model with given initial values
fit_with_init <- function(init_params) {
  tryCatch({
    fit <- pareto_BR_River(
      data = TSNew,
      riv.dist = RiverDis,
      euc.coord = CatCntrWt,
      is.connected = FlowCon,
      riv.weights = Weight,
      u = quantile(rowSums(TSNew), 0.87),
      init = c(init_params$theta, init_params$alpha, s0,
               riv_lmb, init_params$euc_lmb,
               init_params$beta, init_params$c),
      model = 2,
      maxit = 1000,
      method = "BFGS"
    )
    return(list(params = init_params, fit = fit, nllik = fit$nllik,
                converged = TRUE))
  }, error = function(e) {
    return(list(params = init_params, fit = NULL, nllik = Inf,
                converged = FALSE))
  })
}

# Perform grid search
results <- list()
best_nllik <- Inf
best_fit <- NULL
best_init <- NULL

for(i in 1:nrow(param_grid)) {
  if (i %% 500 == 0) {
    cat("Fitting model", i, "of", nrow(param_grid), "\n")
  }

  result <- fit_with_init(param_grid[i, ])
```

```r
  results[[i]] <- result

  # Track best result
  if(result$converged && result$nllik < best_nllik) {
    best_nllik <- result$nllik
    best_fit <- result$fit
    best_init <- result$params
  }
}

# Extract results and find best fit
converged_results <- results[sapply(results, function(x) x$converged)]
nllik_values <- sapply(converged_results, function(x) x$nllik)

# Find best result
best_idx <- which.min(nllik_values)
best_result <- converged_results[[best_idx]]

# save or load the best result
saveRDS(best_result, file = paste(root.dir, "/Data/best_result_M2.rds", sep=""))
best_result <- readRDS(paste(root.dir, "/Data/best_result_M2.rds", sep=""))

cat("Best negative log-likelihood:", best_result$nllik, "\n")
cat("Best initial values of spectral estimation for kernel model 2:\n")
print(best_result$params)
cat("The fitted parameters of spectral estimation for kernel model 2:\n")
print(par2paper(best_result$fit$par))
```

```r
# censored estimation
fit.M2 <- pareto_BR_River_cen(
    data = TSNew,
    riv.dist = RiverDis,
    euc.coord = CatCntrWt,
    riv.weights = Weight,
    u = quantile(TSNew, .90),
    init = best_result$fit$par,
    model = 2,
    maxit = 200,
    method = "BFGS",
    )

# save or load fitted model
saveRDS(fit.M2, file = paste(root.dir, "/Data/fit_m2.rds", sep=""))
fit.M2 <- readRDS(paste(root.dir, "/Data/fit_m2.rds", sep=""))

# fitted negative log-likelihood
cat(paste("The fitted negative log-likelihood of kernel model 2 is", fit.M2$nllik, "\n"))

# transform estimates to the scale used in the paper ###
par  <- fit.M2$par
Par.paper <- par2paper(par)
cat("Final fitted parameter (lam_riv, lam_euc, tau, alpha, beta, c) of kernel model 2:\n")
cat(Par.paper)
```

Kernel 3

```r
# Define grids for each parameter
k = 4 # grid size
theta_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
alpha_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
s0 <- 289.73
riv_lmb_grid <- seq(0.1, 1.0, by = (1.0 - 0.1) / (k - 1))
euc_lmb_grid <- seq(0.01, 0.1, by = (0.1 - 0.01) / (k - 1))
beta_grid <- seq(pi/4, 3*pi/4, by = (3*pi/4 - pi/4) / (k - 1))
c_grid <- seq(0.1, 1.0, by = (1.0 - 0.1) / (k - 1))

# Create all combinations of parameters
param_grid <- expand.grid(
  theta = theta_grid,
  alpha = alpha_grid,
  riv_lmb = riv_lmb_grid,
  euc_lmb = euc_lmb_grid,
  beta = beta_grid,
  c = c_grid
)
cat(paste("The number of parameter combinations for kernel model 3 is", nrow(param_grid)))

# Function to fit model with given initial values
fit_with_init <- function(init_params) {
  tryCatch({
    fit <- pareto_BR_River(
      data = TSNew,
      riv.dist = RiverDis,
      euc.coord = CatCntrWt,
      is.connected = FlowCon,
      riv.weights = Weight,
      u = quantile(rowSums(TSNew), 0.87),
      init = c(init_params$theta, init_params$alpha, s0,
               init_params$riv_lmb, init_params$euc_lmb,
               init_params$beta, init_params$c),
      model = 3,
      maxit = 1000,
      method = "BFGS"
    )
    return(list(params = init_params, fit = fit, nllik = fit$nllik,
                converged = TRUE))
  }, error = function(e) {
    return(list(params = init_params, fit = NULL, nllik = Inf,
                converged = FALSE))
  })
}

# Perform grid search
results <- list()
best_nllik <- Inf
best_fit <- NULL
best_init <- NULL
```

```r
for(i in 1:nrow(param_grid)) {
  if (i %% 500 == 0) {
    cat("Fitting model", i, "of", nrow(param_grid), "\n")
  }

  result <- fit_with_init(param_grid[i, ])
  results[[i]] <- result

  # Track best result
  if(result$converged && result$nllik < best_nllik) {
    best_nllik <- result$nllik
    best_fit <- result$fit
    best_init <- result$params
  }
}

# Extract results and find best fit
converged_results <- results[sapply(results, function(x) x$converged)]
nllik_values <- sapply(converged_results, function(x) x$nllik)

# Find best result
best_idx <- which.min(nllik_values)
best_result <- converged_results[[best_idx]]

# save or load the best result
saveRDS(best_result, file = paste(root.dir, "/Data/best_result_M3.rds", sep=""))
best_result <- readRDS(paste(root.dir, "/Data/best_result_M3.rds", sep=""))

cat("Best negative log-likelihood:", best_result$nllik, "\n")
cat("Best initial values of spectral estimation for kernel model 3:\n")
print(best_result$params)
cat("The fitted parameters of spectral estimation for kernel model 3:\n")
print(par2paper(best_result$fit$par))
```

```r
# censored estimation
fit.M3 <- pareto_BR_River_cen(
    data = TSNew,
    riv.dist = RiverDis,
    euc.coord = CatCntrWt,
    riv.weights = Weight,
    u = quantile(TSNew, .90),
    init = best_result$fit$par,
    model = 3,
    maxit = 200,
    method = "BFGS",
    )

# save or load fitted model
saveRDS(fit.M3, file = paste(root.dir, "/Data/fit_m3.rds", sep=""))
fit.M3 <- readRDS(paste(root.dir, "/Data/fit_m3.rds", sep=""))

# fitted negative log-likelihood
cat(paste("The fitted negative log-likelihood of kernel model 3 is", fit.M3$nllik, "\n"))
```

```r
# transform estimates to the scale used in the paper ###
par  <- fit.M3$par
Par.paper <- par2paper(par)
cat("The fitted parameter (lam_riv, lam_euc, tau, alpha, beta, c) of kernel model 3 are:\n")
cat(Par.paper)
```

Kernel 4

```r
# Define grids for each parameter
k = 8 # grid size
theta_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
alpha_grid <- seq(0.1, 2.0, by = (2.0 - 0.1) / (k - 1))
s0 <- 289.73
riv_lmb_grid <- seq(0.1, 1.0, by = (1.0 - 0.1) / (k - 1))
euc_lmb_grid <- seq(0.01, 0.1, by = (0.1 - 0.01) / (k - 1))
beta <- 0
c <- 1

# Create all combinations of parameters
param_grid <- expand.grid(
  theta = theta_grid,
  alpha = alpha_grid,
  riv_lmb = riv_lmb_grid,
  euc_lmb = euc_lmb_grid
)
cat(paste("The number of parameter combinations for kernel model 4 is", nrow(param_grid)))

# Function to fit model with given initial values
fit_with_init <- function(init_params) {
  tryCatch({
    fit <- pareto_BR_River(
      data = TSNew,
      riv.dist = RiverDis,
      euc.coord = CatCntrWt,
      is.connected = FlowCon,
      riv.weights = Weight,
      u = quantile(rowSums(TSNew), 0.87),
      init = c(init_params$theta, init_params$alpha, s0,
               init_params$riv_lmb, init_params$euc_lmb,
               beta, c),
      model = 4,
      maxit = 1000,
      method = "BFGS"
    )
    return(list(params = init_params, fit = fit, nllik = fit$nllik,
                converged = TRUE))
  }, error = function(e) {
    return(list(params = init_params, fit = NULL, nllik = Inf,
                converged = FALSE))
  })
}

# Perform grid search
```

```r
results <- list()
best_nllik <- Inf
best_fit <- NULL
best_init <- NULL

for(i in 1:nrow(param_grid)) {
  if (i %% 500 == 0) {
    cat("Fitting model", i, "of", nrow(param_grid), "\n")
  }

  result <- fit_with_init(param_grid[i, ])
  results[[i]] <- result

  # Track best result
  if(result$converged && result$nllik < best_nllik) {
    best_nllik <- result$nllik
    best_fit <- result$fit
    best_init <- result$params
  }
}

# Extract results and find best fit
converged_results <- results[sapply(results, function(x) x$converged)]
nllik_values <- sapply(converged_results, function(x) x$nllik)

# Find best result
best_idx <- which.min(nllik_values)
best_result <- converged_results[[best_idx]]

# save or load the best result
saveRDS(best_result, file = paste(root.dir, "/Data/best_result_M4.rds", sep=""))
best_result <- readRDS(paste(root.dir, "/Data/best_result_M4.rds", sep=""))

cat("Best negative log-likelihood:", best_result$nllik, "\n")
cat("Best initial values of spectral estimation for kernel model 34:\n")
print(best_result$params)
cat("The fitted parameters of spectral estimation for kernel model 4:\n")
print(par2paper(best_result$fit$par))
```

```r
# censored estimation
fit.M4 <- pareto_BR_River_cen(
    data = TSNew,
    riv.dist = RiverDis,
    euc.coord = CatCntrWt,
    riv.weights = Weight,
    u = quantile(TSNew, .90),
    init = best_result$fit$par,
    model = 4,
    maxit = 200,
    method = "BFGS",
    )

# save or load fitted model
saveRDS(fit.M4, file = paste(root.dir, "/Data/fit_m4.rds", sep=""))
```

```r
fit.M4 <- readRDS(paste(root.dir, "/Data/fit_m4.rds", sep=""))

# fitted negative log-likelihood
cat(paste("The fitted negative log-likelihood of kernel model 4 is", fit.M4$nllik, "\n"))

# transform estimates to the scale used in the paper ###
par  <- fit.M4$par
Par.paper <- par2paper(par)
cat("The fitted parameter (lam_riv, lam_euc, tau, alpha, beta, c) of kernel model 4 are:\n")
cat(Par.paper)
```