Gumbel hd42

Install and load required packages

```
R_packages <- c("bookdown", "copula", "evd", "ggplot2", "gridExtra", "ks",
                "plot3D", "viridis", "dplyr", "ggpubr", "grid", "cowplot", "parallel")
options(repos = c(CRAN="http://cran.rstudio.com"))
if (!requireNamespace("librarian", quietly = TRUE)) install.packages("librarian")
librarian::shelf(R_packages)
```

Set parameters

```
a <- 4 # copula parameter
gsz <- 100 # grid size
dim <- 4 # dimensions of copula
rname <- "Gumbel-hd42"
```

1 Gumbel Copula

Define the Gumbel copula

```
gumbel_cop <- gumbelCopula(param = a, dim = dim)
```

Create grid on the hypercube

```
# sequence of points in each dimension
seqs <- replicate(dim, seq(0.01, 0.99, length.out = gsz), simplify = FALSE)

# create names for the dimensions
names(seqs) <- paste0("u", 1:dim)

# construct the grid
grid <- expand.grid(seqs)
```

Compute copula density on $[0,1]^2$ copula grid

```
copula_dens <- dCopula(as.matrix(grid), gumbel_cop)
```

For graphs: join grid and copula density

```
gr_dens <- cbind(grid, round(copula_dens,4))
colnames(gr_dens)[ncol(gr_dens)] <- "copula_dens"
```

1.1 Plots over x-y subgrid

```
ux <- "u1"
uy <- "u4"

free_dims <- c(ux, uy)
all_dims  <- grep("^u[0-9]+$", names(gr_dens), value = TRUE)
fixed_dims <- setdiff(all_dims, free_dims)

# compute target values (averages of fixed dims)
target_vals <- colMeans(gr_dens[, fixed_dims, drop = FALSE])

# find closest grid values to those averages
closest <- vapply(fixed_dims, function(dim) {
  vals <- unique(gr_dens[[dim]])
  vals[which.min(abs(vals - target_vals[dim]))]
}, numeric(1))

# filter grid at fixed values
slice_df <- gr_dens
for(dm in fixed_dims){
  slice_df <- slice_df[slice_df[[dm]] == closest[[dm]], ]
}

# dataframe for plotting
plot_df <- slice_df[, c(ux, uy, "copula_dens")]
```

Plot copula density on $[0, 1]^2$ copula grid 3D

```
z_mat_3d <- matrix(as.numeric(plot_df[,"copula_dens"]), nrow = gsz, byrow = FALSE)
z_mat_3d <- pmin(z_mat_3d, 10) # Truncate for better visualization

persp3D(
  x = seqs[[ux]],
  y = seqs[[uy]],
  z = z_mat_3d,
  theta = 25,
  phi = 50,
  col = viridis::viridis(50),
  xlab = "u1", ylab = "u2", zlab = "density",
  main = "Bivariate Gumbel Copula Density (a = 2)",
  cex.main = 0.6, cex.lab = 0.6,
  colkey = list(length = 0.6, width = 0.6, cex.axis = 0.6)
)
```

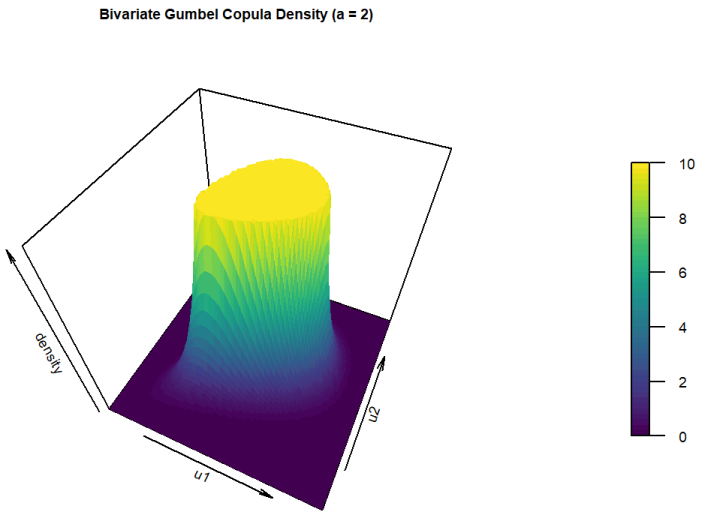

Figure 1.1:
Plot copula density on $[0, 1]^2$ copula grid contour

```
z_mat_2d <- matrix(as.numeric(plot_df[,"copula_dens"]), nrow = gsz, byrow = FALSE)
z_mat_2d <- pmin(z_mat_2d, 8) # Truncate for better visualization

filled.contour(
  x = seqs[[ux]], y = seqs[[uy]], z = z_mat_2d,
  color.palette = viridis::viridis,
  xlab = "", ylab = "",
  plot.axes = {axis(1, at = seq(0.2, 0.8, by = 0.2), line = -0.05); axis(2, line = -3.5)},
  plot.title = title(main = "Copula Density: Contour Plot", cex.main = 0.8),
  key.title = title(main = "Density", cex.main = 0.6), asp = 1, frame.plot = FALSE,
  colkey = list(addlines = FALSE, length = 0.6, width = 0.5, cex.clab = 0.8)
)
```

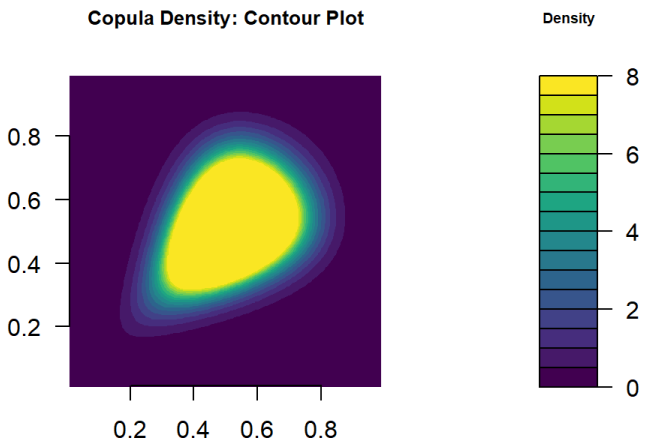

Figure 1.2:

2 Simulated Data

Simulate data

```
n <- 1000 # simulated sample size
set.seed(123456)
cs <- rCopula(n, gumbel_cop)
```

Plot function

```
plot_lower_triangular <- function(cs_df, title = NULL,
                                  point_alpha = 0.4, point_size = 0.2) {
  d <- ncol(cs_df)
  vars <- colnames(cs_df)
  if (d < 2) stop("Need at least 2 columns for pairwise plots.")

  my_scatter <- function(data, xvar, yvar) {
    ggplot(data, aes(x = {{xvar}}, y = {{yvar}})) +
      geom_point(alpha = point_alpha, size = point_size) +
      coord_fixed(ratio = 1) +
      labs(x = "", y = "") +
      theme_minimal() +
      theme(
        plot.margin  = unit(c(0,0,0,0), "in"),
        panel.grid   = element_blank(),
        axis.text    = element_blank(),
        axis.ticks   = element_blank(),
        panel.border = element_rect(colour = "black", fill = NA, linewidth = 0.5)
      )
  }

  empty_panel <- ggplot() + theme_void()

  # build full (d-1) x (d-1) matrix
  plot_list <- vector("list", (d-1)*(d-1))

  for (i in 2:d) {
    for (j in 1:(d-1)) {
      idx <- (i-2)*(d-1) + j
      if (j < i) {
        plot_list[[idx]] <- my_scatter(cs_df, .data[[vars[j]]], .data[[vars[i]]])
      } else {
        plot_list[[idx]] <- empty_panel
      }
    }
  }

  combined_plot <- ggarrange(
    plotlist = plot_list,
    ncol = d-1,
    nrow = d-1,
    align = "hv"
  )

  if (!is.null(title)) {
    combined_plot <- annotate_figure(
      combined_plot,
      top = text_grob(title, size = 8)
    )
  }

  return(combined_plot)
}
```

Scatter plot of simulated draws from copula, pairwise dimensions

```
cs_df <- as.data.frame(cs)
colnames(cs_df) <- paste0("U", 1:dim)
csplot <- plot_lower_triangular(cs_df, title="Gumbel 2")
csplot <- ggdraw(csplot) +
          theme(plot.background = element_rect(color = "gray", fill = NA, size = 0.5),
                plot.margin = unit(c(0., 0.1, -0.1, -0.1), "in"))
```

```
## Warning: The `size` argument of `element_rect()` is deprecated as of ggplot2 3.4.0.
## ℹ Please use the `linewidth` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
plot_file = paste0("../graphs_sim/",rname,"_true.png")
ggsave(plot_file, plot = csplot, width = 4, height = 4, units = "in", dpi = 300)

csplot
```

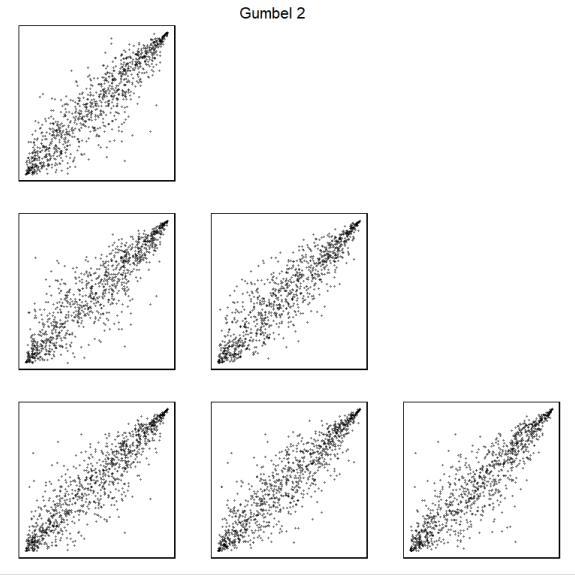

Figure 2.1:

3 Sparse Bernstein Copula

3.1 Set parameters

```
true_dens <- copula_dens
rm(copula_dens)

#sample_sizes <- c(100, 200, 300, 400, 500, 1000, 10000)
sample_sizes <- 10000
kmax <- 5 # number of batches for each sample size
```

Save true density

```
write.table(true_dens, file = paste0("../data_sim/",rname,"_dat_cdens_true.txt"),
            sep = ",", row.names = FALSE, col.names = FALSE)
```

```
#true_dens <- read.table(file = paste0("../data_sim/",rname,"_dat_cdens_true.txt"), header = FALSE)
#true_dens <- as.numeric(unlist(true_dens))
```

3.2 Generate Data of Varying Sample Sizes for Fortran Estimation

```
set.seed(123456)

for (n in sample_sizes) {

  for (k in 1:kmax) {

    cs <- rCopula(n, gumbel_cop)

    # Write out for Fortran input
    write.table(cs, file = paste0("../data_sim/",rname,"_dat_",n,"_",k,".csv"),
                sep = ",", row.names = FALSE, col.names = FALSE)
  }
}
```

Run Fortran estimation of SBP copula

3.3 Plot Simulated Draws from SBP copula

```
n  <- 1000 # for sample size
k <- 1 # for data from batch k
csim_SBP <- read.table(file = paste0("../output_sim/",rname,"_csim_",n,"_",k,".out"), header = FALSE)
csim_SBP_df <- as.data.frame(csim_SBP)
colnames(csim_SBP_df) <- paste0("U", seq_len(ncol(csim_SBP_df)))

csplot_sim <- plot_lower_triangular(csim_SBP_df, title="SBP copula")
csplot_sim <- ggdraw(csplot_sim) +
                theme(plot.background = element_rect(color = "gray", fill = NA, size = 0.5),
                plot.margin = unit(c(0., 0.1, -0.1, -0.1), "in"))

plot_file_sim = paste0("../graphs_sim/",rname,"_SBP.png")
ggsave(plot_file_sim, plot = csplot_sim, width = 4, height = 4, units = "in", dpi = 300)

csplot_sim
```

Figure 3.1:

3.4 Evaluate Kullback-Leibler Distance from True Density

```
KL <- numeric()
KLse <- numeric()
epsilon <- 1e-10 # Add small epsilon to avoid division by 0

# Make a list of tasks (one per n,k)
task_list <- expand.grid(n = sample_sizes, k = 1:kmax, KEEP.OUT.ATTRS = FALSE)

compute_KL <- function(task) {
  n <- task$n
  k <- task$k

  # Read density estimate from file
  fname <- paste0("../output_sim/", rname, "_cdens_", n, "_", k, ".out")
  cdens_SBP <- as.numeric(unlist(read.table(file = fname, header = FALSE)))

  # Compute KL contribution
  ratio <- (true_dens + epsilon) / (cdens_SBP + epsilon)
  KL_k_val <- log(ratio) * true_dens
  mean(KL_k_val)
}

# Split tasks across cores
ncores <- min(detectCores(), 8)
tasks_split <- split(seq_len(nrow(task_list)), cut(seq_len(nrow(task_list)), ncores, labels = FALSE))

# Parallel execution
if (.Platform$OS.type == "unix") {
  KL_vals <- unlist(mclapply(tasks_split, function(idx) {
    sapply(idx, function(i) compute_KL(task_list[i,]))
  }, mc.cores = ncores))
} else {
  cl <- makeCluster(ncores)
  clusterExport(cl, c("task_list", "true_dens", "rname", "epsilon", "compute_KL"),
                envir = environment())
  KL_vals <- unlist(parLapply(cl, tasks_split, function(idx) {
    sapply(idx, function(i) compute_KL(task_list[i,]))
  }))
  stopCluster(cl)
}

# Combine results back by sample size
for (n in sample_sizes) {
  KL_k <- KL_vals[task_list$n == n]
  KL    <- c(KL, mean(KL_k))
  KLse  <- c(KLse, sd(KL_k) / sqrt(length(KL_k)))
}

print(round(KL, 2))
```

```
## [1] 0.35
```

```
print(round(KLse, 2))
```

```
## [1] 0
```