Assessing the Risk of Breast Cancer using Physical Attributes from Cell Nuclei

Maxwell Hoare

October 24, 2024

Abstract

An examination of interdependence in cell nuclei traits from tissue samples taken is conducted using a Bayesian inference approach with a semi-parametric Gaussian Copula model. MCMC samples are generated using an inverse-Wishart prior distribution and extended rank likelihood function. It was found that samples with a positive cancer diagnosis exhibit patterns of dependency with other variables which could be used to identify malignant cells.

1 Introduction

“How are the physical qualities of cell nuclei related to one another? How can understanding dependencies between them provide insights into the distinctions between benign and malignant cells?”

Due to improvements in detection technology over the past 3 decades, 5 year survival rates for individuals with breast cancer have increased to 92% from from 77% (Cancer Council 2022). Despite developments, breast cancer is the second most common cancer diagnosis in Australia (Cancer Council 2022) and efforts to understand this disease remain pertinent. There are known qualities of cell nuclei including size, shape and shade which indicate possible malignancy (Al and Catoi 2007). In this paper, a Gaussian copula model is used to identify interdependence between these metrics Through modelling the joint distribution of these metrics, we can identify patterns in the relationships between the physical features of malignant and benign cells.

The dataset used is “Breast Cancer Wisconsin (Diagnostic)” from “Nuclear feature extraction for breast tumor diagnosis” (Street, Wolberg, and Mangasarian 1993). Street and others obtained characteristic features from images of cell tissue samples taken using fine needle aspiration (FNA). They used image pre-processing techniques including deformable splines, or “snakes” made up of edges, to construct outlines of cell nuclei. The edges along these perimeters were used to obtain 10 features relating to the size, shape and texture of a set of cell nuclei in 569 images. For each feature and each set of nuclei the mean, largest value and standard error for that feature were recorded. The 10 features derived through image pre-processing include 3 size features, 6 shape features, 1 colour feature and 1 pathological variable indicating the diagnosis.

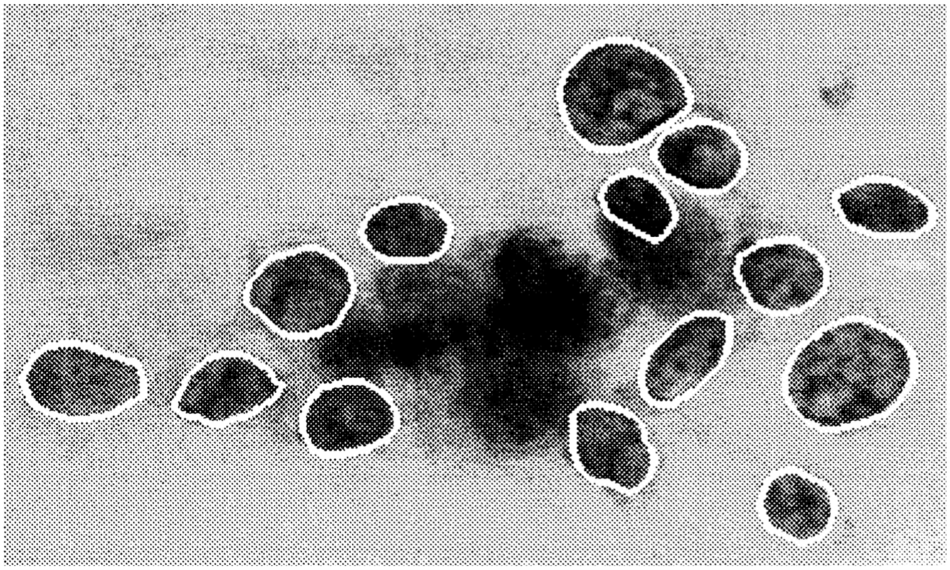

Figure 1: Visualisation of 'snakes' after convergence to nuclei

1.1 Variable Selection for the Model

Taking the required computational resources into consideration, all variables will be included in the model to ensure all relevant dependencies are captured. There are 30 variables in the data, excluding the classification variable 'diagnosis', which denotes whether an observation is benign or malignant. The other variables and their definitions are outlined below.

Size:

Radius: the radius of individual nuclei is measured by averaging the length of the radial line segments from the centroid of the nucleus to each vertices on the perimeter

Perimeter: a measurement of the sum of the distance between all vertices in the snake.

Area: sum of the number of pixels inside the snake boundary.

Irregularity of Shape:

Compactness: uses perimeter and area to provide a composite measure indicating compactness of the cell nuclei using the formula $compactness = \frac{perimeter^2}{area}$. This measurement is minimised by a circular disk and will increase in the case of a more irregular boundary. Less compact nuclei can be an indicator of malignancy.

Smoothness: a measurement of the difference between radial lines and the mean length of the lines surrounding it. There is a negative relationship between smoothness of cell nucleui and malignance.

Concavity: a measure of the magnitude of boundary area that lie on the inside of lines between non-adjacent vertices of the snake.

Concave points: The number of concave points as defined in the concavity measure

Symmetry: Compares the length of the major axis to the length of lines perpendicular to the major axis

Fractal Dimension: An estimate of the fractal dimension of the nuclei using the 'coastline approximation' method from Mandelbrot's seminal paper (Mandelbrot 1967).

Colouring:

Texture: Variance in the greyscale intensity of pixels in the nuclei's. Staining of the nuclei, or hyperchromasia, can be a determinant in diagnosing a malignant cell.

2 The Model

2.1 Background

To distinguish between malignant and benign tissue samples we need to understand the relationships between the features of nuclei discussed above. A Gaussian Copula model can highlight the interdependence between these features making it effective in helping us achieve our objective.

Sklar's theorem provides the scaffolding for such models to do this by showing that joint probability distribution functions can handle information about dependencies between variables ('copulas') and the marginal probability distribution functions separately (Chen 2024). This unique aspect of Gaussian copulas means they can easily handle features with different individual distributions. We can instead guide our focus to multivariate relationships in the data.

2.2 Gaussian Copula Model

This paper utilises a semi-parametric Gaussian copula model to examine the dependencies between nuclei features. The sampling modal can be expressed as follows:

$$\begin{aligned} z_1, z_2, \ldots, z_n \mid C &\sim \text{i.i.d. multivariate normal}(0, C) \\ y_{i,j} &= F_j^{-1}[\Phi(z_{i,j})] \end{aligned}$$

In this equation, F_j^{-1} represents the pseudo inverse of an unknown univariate cumulative distribution. The model uses a technique called extended rank-likelihood (Hoff 2007) which is a function of of the association parameters only, meaning it is suited to handle discrete and continuous variables. The $y_i j$ values denote the $z_i j$ values after the Gaussian copula transformation has been applied. As such, the dependence structure is able to capture the dependence structure of the Gaussian copula without depending on the marginal distributions of the underlying variables (Dey and Zipunnikov 2022). The Gaussian copula then produces a correlation matrix of the transformed variables. The correlation matrix for the Gaussian Copula can be defined as follows:

$$C(u) := \Phi_C(\Phi^-1(u_1), \Phi^-1(u_2), \ldots, \Phi^-1(u_d))$$

2.3 Prior Assumptions

The prior distribution for the correlation matrix C is characterized by the latent variable V, so that: $V \sim inverse - Wishart(\nu_0, \nu_0 V_0)$. Additionally, V is parameterised so that $E[V^{-1}] = V_0^{-1}$. Thus, the distribution of the matrix C is defined to be equivalent to the matrix with entries given by: $C[i,j] = V[i,j] \div \sqrt{V[i,i]V[j,j]}$

2.4 Sampling Method

To conduct Bayesian inference of the correlation matrix C, we will use the Gibbs sampling algorithm, a Markov Chain Monte Carlo method. By implementing a markov chain with a stationary distribution that is equal to $p(C) \mid Z \in D) \propto p(C) \times p(Z \in D \mid C)$, samples of C can be taken which will be used in posterior inference. The following steps outline the process used to generate the Gibbs algorithm.

Step 1 Re-sample Z For each $y \in \text{unique}\{y_{1,j}, \ldots, y_{n,j}\}$:

a. Compute z_l and z_u: $z_l = \max(z_{i,j} : y_{i,j} < y) \quad \text{and} \quad z_u = \min(z_{i,j} : y_{i,j} > y)$

b. For each i such that $y_{i,j} = y$:

- (i) Compute σ_j^2: $\sigma_j^2 = V_{j,j} - V_{j,-j} V_{-j,-j}^{-1} V_{-j,j}$
- (ii) Compute $\mu_{i,j}$: $\mu_{i,j} = Z_{i,-j}(V_{i,-j} V_{-j,-j}^{-1})^T$
- (iii) Sample $u_{i,j}$ uniformly: $u_{i,j} \sim \text{Uniform}\left(\Phi\left(\frac{z_l - \mu_{i,j}}{\sigma_j}\right), \Phi\left(\frac{z_u - \mu_{i,j}}{\sigma_j}\right)\right)$
- (iv) Set $z_{i,j}$: $z_{i,j} = \mu_{i,j} + \sigma_j \times \Phi^{-1}(u_{i,j})$

Step 2: Re-sample V Re-sample V from an inverse-Wishart distribution such that $V \sim \text{inverse-Wishart}(\nu_0 + n, \nu_0 V_0 + Z^T Z)$

Step 3: Compute C Compute C from the aforementioned formula $C_{i,j} = V_{i,j} \div \sqrt{V[i,i]V[j,j]}$.

Iterating this algorithm generates a Markov chain in C whose stationary distribution is $p(C \mid Z \in D)$. A straightforward method is also given for imputation of data that is missing-at-random. For missing value $y_{i,j}$, the full conditional distribution of $z_{i,j}$ is the unconstrained normal distribution with mean $\mu_{i,j}$ and variance σ_j^2 given above.

2.5 Implementation in R: sgbcop

The sgbcop package in R provides methods specially designed to produce estimation and inference for parameters in a semiparametric Gaussian copula model. The sgbcop.mcmc function implements the Gibbs sampling method discussed above as prescribed in the paper "Extending the Rank Likelihood for Semiparametric Copula Estimation" (Hoff 2007). The function generates MCMC samples from the posterior distribution of the correlation matrix C, using a scaled inverse- Wishart prior distribution as well as an extended rank likelihood. sgbcop.mcmc also includes functionality for a semiparametric bayesian imputation procedure which Hoff also describes in the paper.

The procedure was compiled and executed using R version 4.4.1. The model was fitted using the 'sbgcop.mcmc' function, running a total of 10,000 times with parameter values saved every 5 iterations, resulting in 2000 samples of C for posterior analysis. The most important output of the algorithm is an array of dimension p x p x n containing a collection of the correlation matrix posterior samples where p = 31 and n = 2000. Each p x p cross-section of the array represents a correlation matrix sampled from the posterior distribution.

3 Simulation

From the Gibbs output a simulation was designed to produce posterior predictive samples of y which would capture underlying structures in the observed data. The procedure used to simulate the data is as follows. Samples from the posterior distribution of the correlation matrix C were taken through the aforementioned Gibbs sampling procedure. The correlation matrices were then used to draw a sample z from a multivariate normal distribution with mean 0 and covariance matrix C. Finally, the empirical marginal distributions F^j of variables in the observed data are used to transform the samples.

4 Results

4.1 Posterior Inference

Mixing of the Markov chain was successful. Figure 2 shows MCMC samples of 15 variables' correlation with "perimeter_se" (top) and "area_worst" (bottom) in C. It is evident that convergence to stationarity occurs before the first 500 iterations. Thus, a burn-in period of 500 iterations will be employed to maximise posterior density of data for inference. The mean effective sample size was 982.3 with a standard deviation of 204 which is quite high. Lower effective sample sizes came primarily from correlations involving 'diagnosis'. Convergence of the posterior samples is confirmed in trace plots of individual correlation coefficients in figure 3.

4.2 Simulated Data

Now we will compare the distribution of the simulated data with the observed data set. Based on the summaries in table 1 and table 2, we can see that the Gaussian copula model has replicated the distributions of variables in the original data well. We can also see that the range of the variables across the simulated and original data is about even, since the maximum and minimum values across both are comparable.

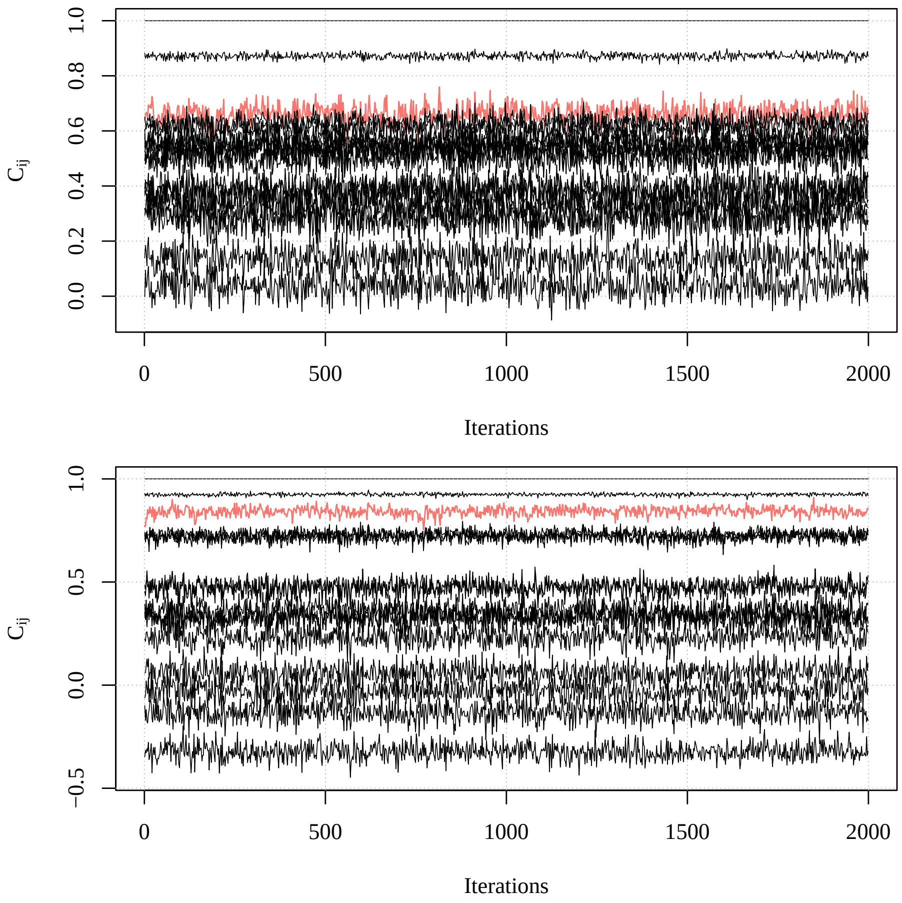

Figure 2: Variable trace plots

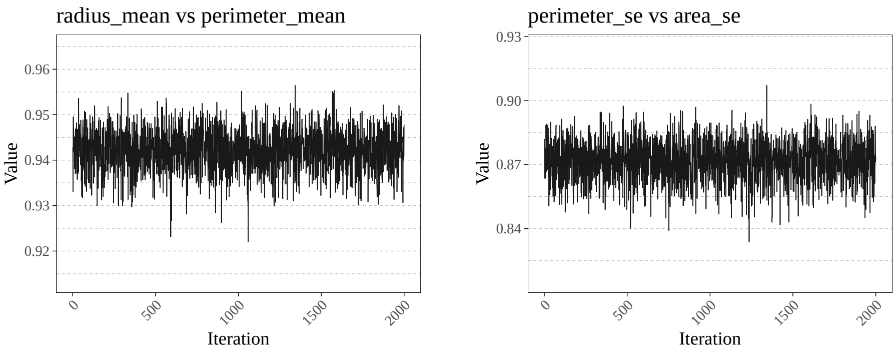

Figure 3: Correlation coefficient trace plots

Table 1: Simulated data summary

	area_mean	compactness_se	concave.points_worst	concavity_worst	diagnosis	smoothness_se
Min	143.5000	0.0022520	0.0000000	0.0000000	1.000000	0.0017130
1st Qu.	408.9598	0.0136108	0.0651380	0.1175730	1.000000	0.0052337
Median	552.8276	0.0206407	0.1042258	0.2342402	1.000000	0.0065028
Mean	648.7891	0.0257817	0.1169895	0.2776072	1.373414	0.0071066
3rd Qu.	800.9777	0.0330239	0.1671592	0.3890442	2.000000	0.0081136
Max	2501.0000	0.1354000	0.2910000	1.2520000	2.000000	0.0299837

Table 2: Original data summary

	area_mean	compactness_se	concave.points_worst	concavity_worst	diagnosis	smoothness_se
Min	143.5000	0.0022520	0.0000000	0.0000000	1.000000	0.001713
1st Qu.	420.3000	0.0130800	0.0649300	0.1145000	1.000000	0.005169
Median	551.1000	0.0204500	0.0999300	0.2267000	1.000000	0.006380
Mean	654.8891	0.0254781	0.1146062	0.2721885	1.372583	0.007041
3rd Qu.	782.7000	0.0324500	0.1614000	0.3829000	2.000000	0.008146
Max	2501.0000	0.1354000	0.2910000	1.2520000	2.000000	0.031130

A common criticism of Gaussian copula models is the inability to capture tail dependencies (AghaKouchak, Sellars, and Sorooshian 2013). Tail dependencies can be calculated using the following equations:

$$\lambda_U = \lim_{q \to 1} P(U_2 \geq q \mid U_1 \geq q) \ \lambda_L = \lim_{q \to 0} P(U_2 \leq q \mid U_1 \leq q)$$

Where U1 and U2 are two variables that we are looking at the dependencies between. The Gaussian copula model assumes that the tail dependencies are symmetrical.

$$\lambda_L = \lambda_U = 2\Phi\left(-\sqrt{\frac{\rho}{1-\rho}}\right) - 1$$

Where ρ is a Pearson coefficient in the matrix C. As such, $\lambda_L = \lambda_U = 0$ when $\rho \neq 1$ or the correlation is imperfect. This assumption is not always reflective of empirical data and can cause the model to systemically underestimate the likelihood of extreme events. Figure 4 provides a more detailed comparison of the upper tail distribution, showing data above the 90th percentile across the original and simulated data for 3 different variable couples.

There is an approximately equal amount of data in the simulated graph than the original graphs which is a good sign. While the relative dispersion and direction of the data is not precisely captured in the simulated tail plots, there are some indications of the original shape. Thus, the assumption of symmetric tail dependencies is appropriate here. Figure 5 contains pairs plots of a selection of variables. The dependence structures of the variables are roughly linear or normal, with singular modes, approximating a multivariate normal distribution. This is optimal under the symmetric variance assumption.

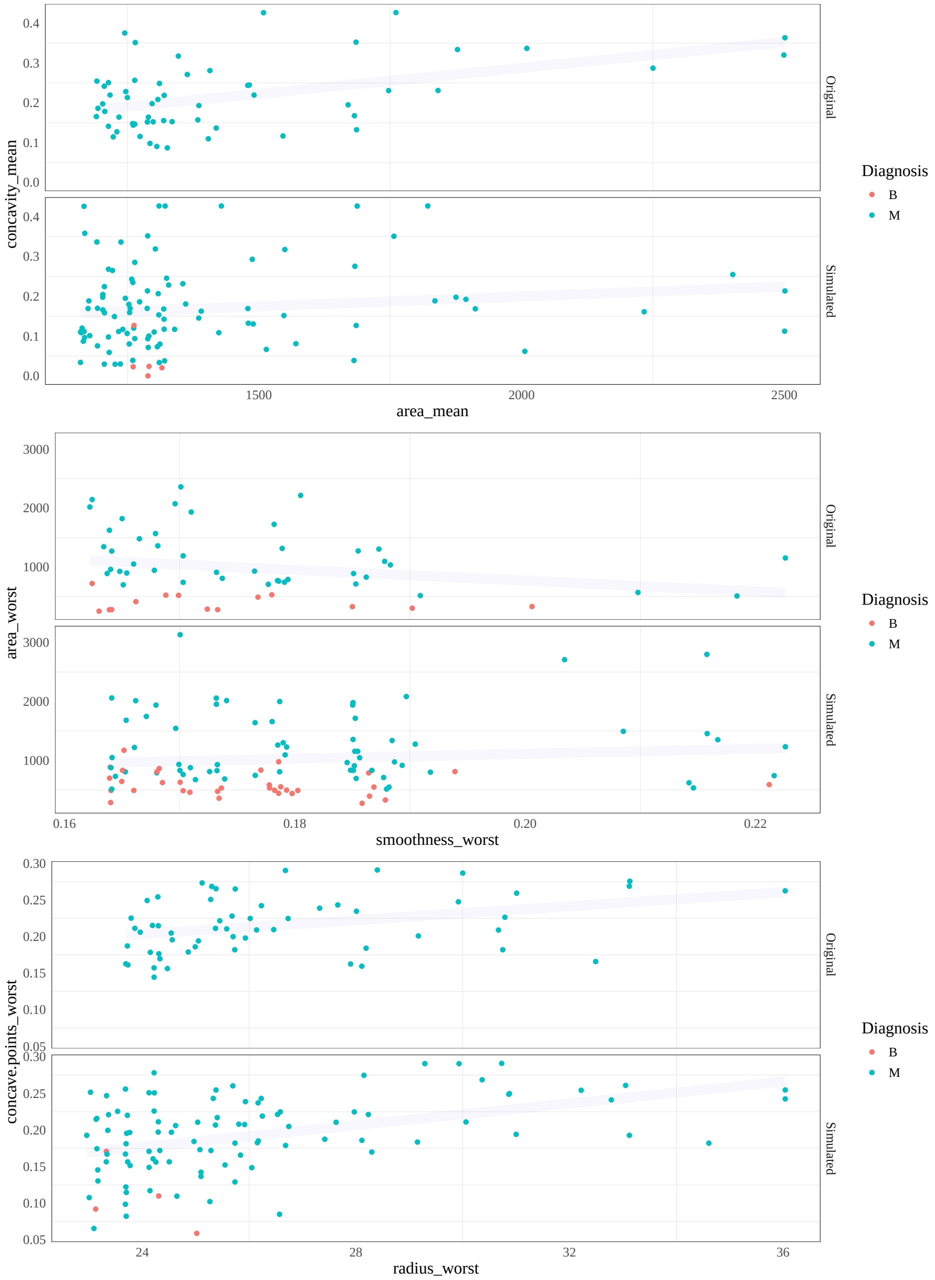

Figure 4: Tail dependencies for observed and simulated data

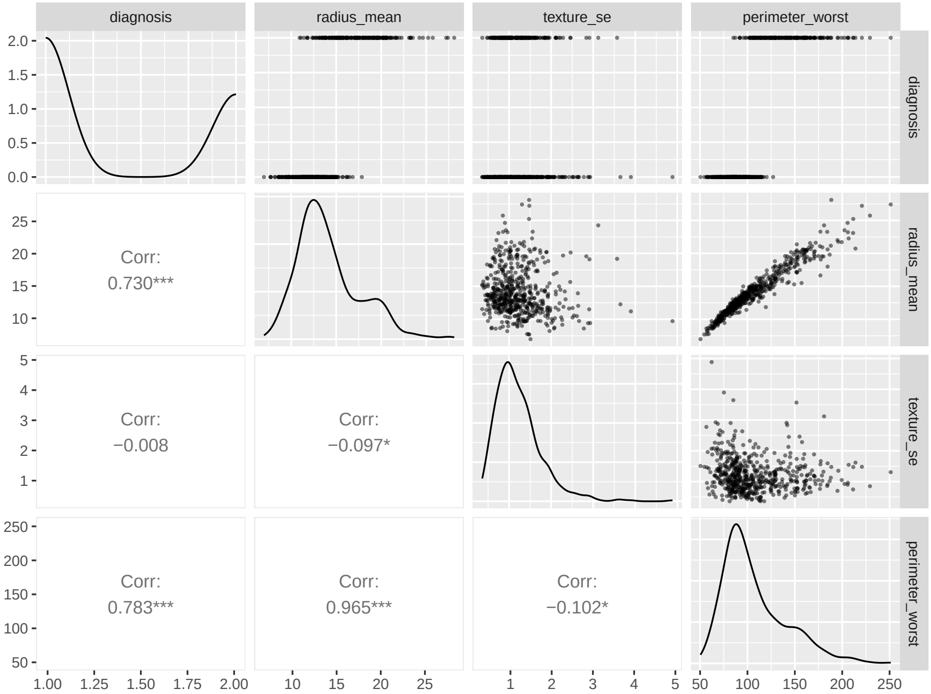

Figure 5: Pairs plots

To help answer the question "How can understanding these dependencies provide insights into the distinctions between benign and malignant cells?" a random forest classification model was built on the simulated data with 500 trees. This model was then used to predict the diagnosis variable in the observed data set. The random forest had an true positive rate of 95%, which indicates that the dependencies uncovered by the Gaussian model accurately represent the set of physical attributes in malignant cells to high degree.

5 Conclusions

To conclude, this report has explored the interdependencies between the physical features of cell nuclei. These interdependencies were examined in-depth using a semiparametric Gaussian copula model. The model proved to be well suited to the observed data, and was able to effectively represent dependencies variables as expected based on the existing medical literature (Al and Catoi 2007). Namely, positive relationships were observed between size, shape irregularity, texture and positive diagnosis of tissue samples.

A central objective of this paper was to uncover the characteristic variable interdependence patterns of benign and malignant cells. The strongest correlation with diagnosis belonged to "concave.points_worst", the largest number of concave vertices in the outline edges of cell nuclei in a sample. This was one of a number of proxy measures for shape irregularity which is common to cancerous cells including "smoothness", "concavity" and "Compactness". However, "concavity.points" was most associated with diagnosis by a non-trivial margin. There was a strong relationship between "perimeter_worst" and “diagnosis”, and between “area_worst” and “diagnosis”, meaning instances of large nuclei convey the possibility of a positive diagnosis. The use of a random forest model trained on the simulated data to classify diagnosis in the observed data provided further confidence in dependencies which showed connections to malignancy. The model achieved an accuracy rate of 95%, though testing against more unseen data is required for a robust result.

The results of the Gaussian copula model give us a more precise comprehension of the relationships between the variables than other methods. The central advantage of this model is that it allows us to examine interdependence between nuclei characteristics without placing any stringent constraints regarding the empirical marginal distributions in the observed data. This has proved beneficial in the analysis by allowing us to direct our focus toward the characteristic relationships between variables in the data.

5.1 Limitations and Reflection

The most salient shortcoming of the approach taken in this report are the limitations of the Gaussian copula model itself. As discussed, Gaussian copulas use the assumption of normality in the joint distribution’s margins when transformed by the cumulative distribution function (CDF). While the data set proved to be well suited to this assumption, there were instances of a-normal joint marginal distributions which may not have been successfully captured by this model. Approaches such as the t-copula, and the Archimedean copula supplement this weakness by using more flexible distributions to model the dependencies and are worth considering for future applications. Given more time, the report could also benefit from using the Modified Gaussian copula model put forward by Fang and Madsen which observed improved results for modelling data with tail and elliptically distributed dependence (Fang and Madsen 2013).

It is important to note the potential for significant clinical and diagnostic applications of this approach. A substantial body of literature uses statistical analysis of physical attributes to determine malignancy. For instance, fractal analysis has been leveraged to distinguish between benign and malignant prostate tumors with a high degree of accuracy (Michallek et al. 2022). The model used in this paper proposes utilizing a combination of measures to identify characteristic dependencies in the physical attributes, which maximises the possibility of positively identifying malignant cells. The unique insight provided by this approach in the report highlights the possibility of integrating these tools into existing histological methods.

6 Appendix

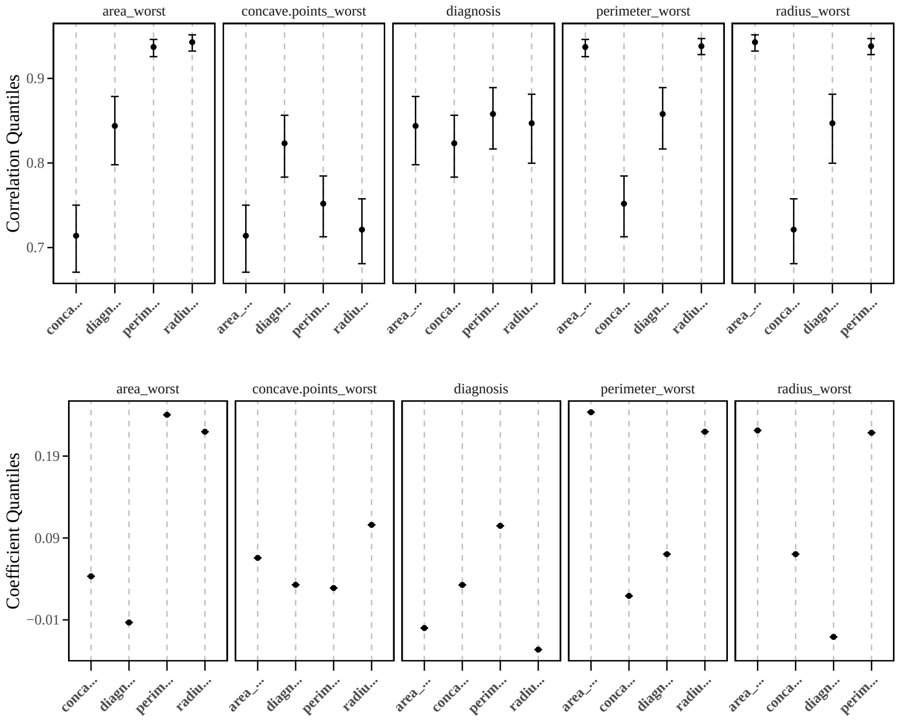

Figure 6: 2.5%, 50% and 97.5% quantiles for correlation coefficients (row 1) and regression coefficients (row 2)

Table 3: Average effective sample sizes for variable correlation coefficients in C

	Average_ESS
diagnosis	428.35
radius_mean	1017.19
texture_mean	1001.88
perimeter_mean	1010.84
area_mean	1012.17
smoothness_mean	1047.91
compactness_mean	951.91
concavity_mean	997.68
concave.points_mean	1122.52
symmetry_mean	988.25
fractal_dimension_mean	970.81
radius_se	988.91
texture_se	1010.89
perimeter_se	983.07
area_se	1007.42
smoothness_se	1017.26
compactness_se	925.81
concavity_se	967.52
concave.points_se	967.22
symmetry_se	995.22

fractal_dimension_se	925.77
radius_worst	1003.64
texture_worst	985.02
perimeter_worst	1028.26
area_worst	1040.19
smoothness_worst	1020.57
compactness_worst	994.86
concavity_worst	1011.31
concave.points_worst	1023.57
symmetry_worst	997.26
fractal_dimension_worst	1010.41

7 R Code

```
bc_raw <- read.csv("input/breast_cancer_data.csv")

remove_vars <- which(colnames(bc_raw) %in% c("X"))
bc_data <- as.data.frame(bc_raw[,-remove_vars])

bc_data$diagnosis <- as.numeric(factor(bc_raw$diagnosis, levels = c("B","M"), labels =

bc_mat <- as.matrix(bc_data[,-1])
```

```
summary(bc_mat)
bc_mat <- na.omit(bc_mat)
any(is.infinite(bc_mat))

set.seed(123)

modelfit <- sbgcop.mcmc(bc_mat, S0 = diag(ncol(bc_mat)), nsamp = 10000, odens = 5)
summary(modelfit)
plot(modelfit)

no_saved_iterations <- dim(modelfit$C.psamp)[3]
simulations <- array(NA, dim = c(n_sims, ncol(bc_mat)))

for (i in 1:no_saved_iterations) {
  ccorr <- modelfit$C.psamp[, , i]
  value <- rmvnorm(1, sigma = ccorr)
  for (j in 1:ncol(bc_mat)) {
    simulations[i, j] <- quantile(bc_mat[, j],
                                  probs = pnorm(value[j]),
                                  na.rm = TRUE,
                                  type = 8)
  }
}

sim_data <- as.data.frame(simulations)
colnames(sim_data) <- colnames(bc_mat)

dat_summary <- summary(bc_mat)
sim_summary <- summary(sim_data)

correlation_samples <- modelfit$C.psamp
```

```
options(scipen = 999)
dat_summary <- as.data.frame(bc_mat[, c(1, 5, 16, 17, 28, 29)]) %>%
  pivot_longer(cols = colnames(bc_mat)[c(1, 5, 16, 17, 28, 29)]) %>%
  group_by(name) %>%
  summarise(
    Min = min(value),
    `1st Qu.` = quantile(value, 0.25),
    Median = median(value),
    Mean = mean(value),
    `3rd Qu.` = quantile(value, 0.75),
    Max = max(value)
  )
dat_summary <- t(dat_summary)
dat_summary = cbind(rownames(dat_summary),dat_summary)
dat_summary[1,1] = ""
colnames(dat_summary) = dat_summary[1,]
dat_summary = dat_summary[-1,]
dat_summary = as.data.frame(dat_summary)
write_csv(dat_summary, "tables/dat_summary.csv")

sim_summary <- sim_data[, c(1, 5, 16, 17, 28, 29)] %>%
  pivot_longer(cols = colnames(sim_data)[c(1, 5, 16, 17, 28, 29)]) %>%
  group_by(name) %>%
  summarise(
    Min = min(value),
    `1st Qu.` = quantile(value, 0.25),
    Median = median(value),
    Mean = mean(value),
    `3rd Qu.` = quantile(value, 0.75),
    Max = max(value)
  )
sim_summary <- t(sim_summary)
sim_summary = cbind(rownames(sim_summary),sim_summary)
sim_summary[1,1] = ""
colnames(sim_summary) = sim_summary[1,]
sim_summary = sim_summary[-1,]
sim_summary = as.data.frame(sim_summary)
write_csv(sim_summary, "tables/sim_summary.csv")
```

```
choice_cols <- c("radius_mean",
                 "perimeter_mean",
                 "perimeter_se",
                 "area_se",
                 "radius_worst",
                 "concave.points_worst")
index <- which(colnames(bc_mat) %in% choice_cols)
sim_data$Iteration = seq_along(sim_data$radius_mean)
# Combine the data frames
combined_df <- pivot_longer(sim_data, -c(Iteration), names_to = "Parameter", values_to
col_names = colnames(bc_data[, -1])

combined_df2 <- combined_df[combined_df$Parameter %in% choice_cols,]

value_ranges <- combined_df2 %>%
  group_by(Parameter) %>%
  summarise(
    y_min = min(Value, na.rm = TRUE) - 0.25 * (max(Value, na.rm = TRUE) - min(Value, n
    y_max = max(Value, na.rm = TRUE) + 0.25 * (max(Value, na.rm = TRUE) - min(Value, n
    .groups = 'drop'
  )

combined_df2 <- combined_df2 %>%
  left_join(value_ranges, by = "Parameter")

grDevices::pdf(file = "plots/converge_sim.pdf", width = 10, height = 7)
for(i in 1:6) {
  varname = choice_cols[i]
  combined_df3 = combined_df2[combined_df2$Parameter == varname, ]
  ymin = combined_df3$y_min[1]
  ymax = combined_df3$y_max[1]

  assign(paste0("p.",varname),
         ggplot(combined_df3, aes(x = Iteration, y = Value)) +
           geom_line(alpha = 0.9, size = 0.3) +
           coord_cartesian(ylim = c(ymin, ymax)) +
           labs(title = varname, x = "Iteration", y = "Value") +
           theme_minimal() +
           theme(
             legend.position = "none",
             text = element_text(family = "serif", size = 14),
             axis.text.x = element_text(
               angle = 45,
               hjust = 1),
             panel.border = element_rect(
               colour = "black",
               fill = NA,
               linewidth = 0.25
             ),
             panel.grid.major.y = element_line(
               color = "grey",
               linewidth = 0.25,
               linetype = "dashed"
             ),
             panel.grid.minor.y = element_line(
               color = "grey",
               linewidth = 0.25,
               linetype = "dashed"
             ),
             panel.grid.major.x = element_blank(),
             panel.grid.minor.x = element_blank(),
             axis.ticks.x = element_line(color = "black", linewidth = 0.25),
             axis.ticks.length.x = unit(0.1, "cm"),
             axis.ticks.y = element_line(color = "black", linewidth = 0.25),
             axis.ticks.length.y = unit(0.1, "cm"),
             plot.margin = unit(margins, "cm"))
  )
}

grid.arrange(p.radius_mean,
             p.perimeter_mean,
             p.perimeter_se,
             p.area_se,
             p.radius_worst,
             p.concave.points_worst,
             nrow = 2,
             ncol = 3)

dev.off()
```

```
mcmc_data <- as.mcmc(sim_data)
autocorr.plot(mcmc_data[,c(1:9)], lag.max=50)

set.seed(1)
r1 = round(runif(4, min = 1, max = 31),0)
r2 = round(runif(4, min = 1, max = 31),0)
r3 = round(runif(4, min = 1, max = 31),0)
mcmc_data <- as.mcmc(t(correlation_samples[r1[1],r1[c(2:4)],]))
mcmc_data2 <- as.mcmc(t(correlation_samples[r2[1],r2[c(2:4)],]))
```

```
mcmc_data3 <- as.mcmc(t(correlation_samples[r3[1],r3[c(2:4)],]))
par(mfrow=c(3,3))
autocorr.plot(mcmc_data, lag.max=15, auto.layout = FALSE)
autocorr.plot(mcmc_data2, lag.max=15, auto.layout = FALSE)
autocorr.plot(mcmc_data3, lag.max=15, auto.layout = FALSE)
par(mfrow=c(1,1))
```

```
n = dim(correlation_samples)[2]
effective_samples <- matrix(ncol = n, nrow = n)
for(i in 1:31){
   effective_samples[,i] <- as.numeric(effectiveSize(t(correlation_samples[i,,c(1:1000
}
diag(effective_samples) <- NA
mean_effective_size <- mean(effective_samples, na.rm = TRUE)
sd_effective_size <- sd(matrix(effective_samples), na.rm = TRUE)
```

```
colnames(effective_samples) <- colnames(bc_mat)

Average_ESS <- round(colMeans(effective_samples, na.rm = TRUE),2)

Average_ESS = data.frame(Average_ESS)

Average_ESS = cbind(rownames(Average_ESS),Average_ESS)
# Average_ESS = rbind(colnames(Average_ESS),Average_ESS)

write_csv(Average_ESS, "tables/average_ess.csv")
```

```
gg_color_hue <- function(n) {
  hues = seq(15, 375, length = n + 1)
  hcl(h = hues, l = 65, c = 100)[1:n]
}
cols = gg_color_hue(10)

plot(1:10, pch = 16, cex = 2, col = cols)
```

```
n_sims <- dim(correlation_samples)[3]
n_vars = dim(correlation_samples)[2]
grDevices::pdf(file = "plots/trace.pdf", width = 7, height = 7)
par(mfrow = c(2,1),
    family = "serif",
    mar = c(4, 4.25, 1, 1),
    oma = c(0.5, 0.25, 0.5, 0.5))
interval = 2
cols = gg_color_hue(1)
for(i in c(14,25)) {
  convergence_matrix <- as.data.frame(t(correlation_samples[i, , ]))
  convergence_matrix$Iterations = seq_along(convergence_matrix[, 1])
  minimum = min(convergence_matrix)
  colour <- ifelse(colnames(convergence_matrix)[i] == "diagnosis",
                   cols[1],
                   "black")

  lwidth <- ifelse(colnames(convergence_matrix)[i] == "diagnosis",
                       1,
                       0.6)


  plot(
    x = convergence_matrix$Iterations[seq(1,nrow(convergence_matrix),interval)],
    y = convergence_matrix[seq(1,nrow(convergence_matrix),interval), i],
    type = "l",
    ylim = c(minimum, 1),
    main = NULL,
    xlab = "Iterations",
    ylab = expression(C[ij]),
    col = colour,
    lwd = lwidth)
  grid(nx = NULL, ny = NULL, col = "darkgray", lwd = 0.7)

  for (j in seq(1,n_vars,2)) {
    if (j != i) {
      colour = ifelse(colnames(convergence_matrix)[j] == "diagnosis",
                       cols[1],
                       "black")

      lwidth <- ifelse(colnames(convergence_matrix)[j] == "diagnosis",
                       1,
                       0.6)

      lines(x = convergence_matrix$Iterations[seq(1,nrow(convergence_matrix),interval)
            y = convergence_matrix[seq(1,nrow(convergence_matrix),interval), j],
            col = colour,
            lwd = lwidth)
    }
  }
}
par(mfrow = c(1,1))
dev.off()
```

```
choice_cols <- c("radius_mean",
                 "perimeter_mean",
                 "perimeter_se",
                 "area_se",
                 "radius_worst",
                 "concave.points_worst")
index <- which(colnames(bc_mat) %in% choice_cols)
# 11 fractal_dimension_mean
corr_data <- data.frame(correlation_samples[index[1],index[2],])
corr_data2 <- data.frame(correlation_samples[index[3],index[4],])
corr_data3 <- data.frame(correlation_samples[index[5],index[6],])

corr_data$Iteration = seq_along(corr_data[,1])
corr_data2$Iteration = seq_along(corr_data[,1])
corr_data3$Iteration = seq_along(corr_data[,1])

colnames(corr_data)[1] = paste(choice_cols[1],"vs",choice_cols[2])
colnames(corr_data2)[1] = paste(choice_cols[3],"vs",choice_cols[4])
colnames(corr_data3)[1] = paste(choice_cols[5],"vs",choice_cols[6])

# Combine the data frames
combined_df <- left_join(corr_data,
                         left_join(corr_data2,
                                   corr_data3, by = "Iteration"),
                         by = "Iteration") %>%
  pivot_longer(cols = c(colnames(corr_data)[1],
                        colnames(corr_data2)[1],
                        colnames(corr_data3)[1]),
               names_to = "Parameter",
               values_to = "Value")

value_ranges <- combined_df %>%
  group_by(Parameter) %>%
  summarise(
    y_min = min(Value, na.rm = TRUE) - 0.25 * (max(Value, na.rm = TRUE) - min(Value, n
    y_max = max(Value, na.rm = TRUE) + 0.25 * (max(Value, na.rm = TRUE) - min(Value, n
    .groups = 'drop'
  )



combined_df2 <- combined_df %>%
  left_join(value_ranges, by = "Parameter")

grDevices::pdf(file = "plots/converge_corr.pdf", width = 10, height = 4)
for(i in 1:2) {
  varname = colnames(get(paste0("corr_data",ifelse(i==1,"",i))))[1]
  combined_df3 = combined_df2[combined_df2$Parameter == varname, ]
  ymin = combined_df3$y_min[1]
  ymax = combined_df3$y_max[1]

  assign(paste0("p.",i,i,i),
         ggplot(combined_df3, aes(x = Iteration, y = Value)) +
           geom_line(alpha = 0.9, size = 0.3) +
           coord_cartesian(ylim = c(ymin, ymax)) +
           labs(title = varname, x = "Iteration", y = "Value") +
           theme_minimal() +
           theme(
             legend.position = "none",
             text = element_text(family = "serif", size = 14),
             axis.text.x = element_text(
               angle = 45,
               hjust = 1),
             panel.border = element_rect(
               colour = "black",
               fill = NA,
               linewidth = 0.25
             ),
             panel.grid.major.y = element_line(
               color = "grey",
               linewidth = 0.25,
               linetype = "dashed"
             ),
             panel.grid.minor.y = element_line(
               color = "grey",
               linewidth = 0.25,
               linetype = "dashed"
             ),
             panel.grid.major.x = element_blank(),
             panel.grid.minor.x = element_blank(),
             axis.ticks.x = element_line(color = "black", linewidth = 0.25),
             axis.ticks.length.x = unit(0.1, "cm"),
             axis.ticks.y = element_line(color = "black", linewidth = 0.25),
             axis.ticks.length.y = unit(0.1, "cm"),
             plot.margin = unit(margins, "cm"))
  )
}


grid.arrange(p.111, p.222, nrow = 1, ncol = 2)



dev.off()
```

```
choice_cols <- c("diagnosis",
                 "area_mean",
                 "concavity_mean",
                 "smoothness_worst",
                 "area_worst",
                 "radius_worst",
                 "concave.points_worst")

index = which(colnames(bc_mat) %in% choice_cols)

dfbc_mat = as.data.frame(bc_mat)[,index] %>%
  mutate(rowno = row_number()) %>%
  pivot_longer(cols = choice_cols[-1]) %>%
  mutate(type = "Original")

sim_data2 = sim_data[,index] %>%
  mutate(rowno = row_number()) %>%
  pivot_longer(cols = choice_cols[-1]) %>%
  mutate(type = "Simulated")

combined_df = rbind(dfbc_mat, sim_data2)
qu = 0.9

plot_data <- combined_df %>%
  filter(name %in% c(choice_cols[2], choice_cols[3])) %>%
  mutate(diagnosis = round(diagnosis,0)) %>%
  pivot_wider() %>%
  group_by(type) %>%
  filter(area_mean > quantile(area_mean, qu)) %>%
  ungroup()

plot_data2 <- combined_df %>%
  filter(name %in% c(choice_cols[4], choice_cols[5])) %>%
  mutate(diagnosis = round(diagnosis,0)) %>%
  pivot_wider() %>%
  group_by(type) %>%
  filter(smoothness_worst > quantile(smoothness_worst, qu)) %>%
  ungroup()

plot_data3 <- combined_df %>%
  filter(name %in% c(choice_cols[6], choice_cols[7])) %>%
  mutate(diagnosis = round(diagnosis,0)) %>%
  pivot_wider() %>%
  group_by(type) %>%
  filter(radius_worst > quantile(radius_worst, qu)) %>%
  ungroup()
```

```r
pdf("plots/comp.pdf", width = 11, height = 15)
p1 <- ggplot(plot_data, aes(x = area_mean, y = concavity_mean,
                      colour =
                        factor(diagnosis,
                               levels = c(1,2),
                               labels = c("B","M")))) +
  geom_smooth(method = "glm", se = FALSE,
              color = rgb(0.4,0.4,0.9,0.05),
              linewidth = 4) +
  geom_point() +
  facet_grid(rows = "type", scales = "free_x") +
  labs(colour = "Diagnosis") +
  theme_minimal() +
  theme(
    text = element_text(family = "serif", size = 14),
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_blank(),
    panel.border = element_rect(
      colour = "black",
      fill = NA,
      linewidth = 0.2
    )
  )

p2 <- ggplot(plot_data2, aes(x = smoothness_worst, y = area_worst,
                      colour =
                        factor(diagnosis,
                               levels = c(1,2),
                               labels = c("B","M")))) +
  geom_smooth(method = "glm", se = FALSE,
              color = rgb(0.4,0.4,0.9,0.05),
              linewidth = 4) +
  geom_point() +
  facet_grid(rows = "type", scales = "free_x") +
  labs(colour = "Diagnosis") +
  theme_minimal() +
  theme(
    text = element_text(family = "serif", size = 14),
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_blank(),
    panel.border = element_rect(
      colour = "black",
      fill = NA,
      linewidth = 0.2
    )
  )
```

```
p3 <- ggplot(plot_data3, aes(x = radius_worst, y = concave.points_worst,
                      colour =
                        factor(diagnosis,
                               levels = c(1,2),
                               labels = c("B","M")))) +
  geom_smooth(method = "glm", se = FALSE,
              color = rgb(0.4,0.4,0.9,0.05),
              linewidth = 4) +
  geom_point() +
  facet_grid(rows = "type", scales = "free_x") +
  labs(colour = "Diagnosis") +
  theme_minimal() +
  theme(
    text = element_text(family = "serif", size = 14),
    panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_blank(),
    panel.border = element_rect(
      colour = "black",
      fill = NA,
      linewidth = 0.2
    )
  )

grid.arrange(p1, p2, p3, nrow = 3)

dev.off()
```

```
choice_cols <- c("radius_mean",
                 "texture_se",
                 "perimeter_worst",
                 "concavity.points_worst",
                 "diagnosis")
index <- which(colnames(bc_mat) %in% choice_cols)

pdf("plots/pairs.pdf", width = 8, height = 6)

ggpairs(bc_mat[,index], upper = list(continuous = wrap("points", alpha = 0.5, size =
dev.off()
```

```
n <- 31
num_simulations <- 1000
reg_coeff_array <- array(NA, dim = c(n, n, num_simulations))

# Compute regression coefficients for each matrix
for (k in 1:num_simulations) {
  for (j in 1:n) {
    minus_j <- setdiff(1:n, j)

    # Safely perform matrix operations
    tryCatch({
      inv_matrix <- solve(correlation_samples[minus_j, minus_j, k])
      reg_coeff_array[j, minus_j, k] <- inv_matrix %*% correlation_samples[minus_j, j,
    }, error = function(e) {
      cat("Error in matrix inversion or multiplication at simulation",
          k, "and variable", j, "\n")
    })
  }
}

# Optionally, convert list to an array for easier handling
reg_coeff_array <- array(unlist(regression_coefficients),
                         dim = c(31, 31, 1000))
```

```
choice_cols <- c("diagnosis",
                 "perimeter_worst",
                 "area_worst",
                 "radius_worst",
                 "concave.points_worst")
le <- length(choice_cols)
index <- which(colnames(bc_mat) %in% choice_cols)
results = list()
qts = c(0.025,0.5,0.975)

for(i in 1:le){
  for(j in index[-i]){
    tab <- t(correlation_samples[index[i], j, ])
    res <- quantile(tab, qts, names = FALSE)
    results[[paste0(colnames(bc_mat)[i],"+",colnames(bc_mat)[j])]] =
      data.frame(colfrom = colnames(bc_mat)[index[i]],
                                  colto = colnames(bc_mat)[j],
                 `2.5%` = res[1], `50%` = res[2], `97.5%` = res[3])
  }
}

corr_qt_results <- rbindlist(results)
names(corr_qt_results) <- c("variable_from","variable_to","2.5%","50%","97.5%")
names_to = character(nrow(corr_qt_results))
for(i in 1:length(names_to)){
  names_to[i] <- paste0(substr(corr_qt_results$variable_to[i], 1, 5),"...")
}
corr_qt_results$names_to = names_to
```

```
# Regression coefficient quantiles
results = list()
for(i in 1:le){
  for(j in index[-i]){
    tab <- t(reg_coeff_array[index[i], j, ])
    res <- quantile(tab, qts, names = FALSE)
    results[[paste0(colnames(bc_mat)[i],"+",colnames(bc_mat)[j])]] =
      data.frame(colfrom = colnames(bc_mat)[index[i]],
                                  colto = colnames(bc_mat)[j],
                 `2.5%` = res[1], `50%` = res[2], `97.5%` = res[3])
  }
}

reg_qt_results <- rbindlist(results)
names(reg_qt_results) <- c("variable_from","variable_to","2.5%","50%","97.5%")
names_to = character(nrow(reg_qt_results))
for(i in 1:length(names_to)){
  names_to[i] <- paste0(substr(reg_qt_results$variable_to[i], 1, 5),"...")
}
reg_qt_results$names_to = names_to

margins = c(0.5,0.5,0.5,1)

pdf(file = "plots/quantiles.pdf", width = 10, height = 8)

p9 <- ggplot(corr_qt_results, aes(x = names_to, y = `50%`, group = variable_from)) +
  geom_errorbar(aes(ymin = `2.5%`, ymax = `97.5%`), width = 0.2) +
  geom_point() +
  facet_grid(. ~ variable_from, scales = "free_x", space = "free_x") +
  theme_minimal() +
  scale_y_continuous(breaks = seq(0.2,1,0.1)) +
  labs(
    x = NULL,
    y = "Correlation Quantiles"
  ) +
  theme(
    text = element_text(family = "serif", size = 14),
    axis.text.x = element_text(angle = 45, hjust = 1, face = "bold"),
    panel.border = element_rect(
      colour = "black",
      fill = NA,
      linewidth = 1
    ),
    panel.grid.major.x = element_line(
      color = "grey",
      linewidth = 0.5,
      linetype = "dashed"
    ),
    panel.grid.minor.x = element_line(
      color = "lightgrey",
      linewidth = 0.25,
      linetype = "dashed"
    ),
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    axis.ticks.x = element_line(color = "black"),
    axis.ticks.length.x = unit(0.25, "cm"),
    axis.ticks.y = element_line(color = "black"),
    axis.ticks.length.y = unit(0.15, "cm"),
    plot.margin = unit(margins, "cm"))

p10 <- ggplot(reg_qt_results, aes(x = names_to, y = `50%`, group = variable_from)) +
  geom_errorbar(aes(ymin = `2.5%`, ymax = `97.5%`), width = 0.2) +
  geom_point() +
  facet_grid(. ~ variable_from, scales = "free_x", space = "free_x") +
  theme_minimal() +
  labs(
    x = NULL,
    y = "Coefficient Quantiles"
  ) +
  scale_y_continuous(breaks = seq(-0.01,1,0.1)) +
  theme(
    text = element_text(family = "serif", size = 14),
    axis.text.x = element_text(angle = 45, hjust = 1, face = "bold"),
    panel.border = element_rect(
      colour = "black",
      fill = NA,
      linewidth = 1
    ),
    panel.grid.major.x = element_line(
      color = "grey",
      linewidth = 0.5,
      linetype = "dashed"
    ),
    panel.grid.minor.x = element_line(
      color = "lightgrey",
      linewidth = 0.25,
      linetype = "dashed"
    ),
    panel.grid.major.y = element_blank(),
    panel.grid.minor.y = element_blank(),
    axis.ticks.x = element_line(color = "black"),
    axis.ticks.length.x = unit(0.25, "cm"),
    axis.ticks.y = element_line(color = "black"),
    axis.ticks.length.y = unit(0.15, "cm"),
    plot.margin = unit(margins, "cm"))


grid.arrange(p9, p10, nrow = 2)
dev.off()
```

```
library(randomForest)
burnin = c(1:499)
predictors <- sim_data[-burnin, -which(names(sim_data) %in% c("diagnosis", "Iteration"
response <- ifelse(sim_data$diagnosis[-burnin] >= 1.5, 2, 1)
response <- factor(response, levels = c(1,2), labels = c("M","B"))

predictors.t <- bc_mat[, -1]
response.t <- ifelse(bc_mat[,1] >= 1.5, 2, 1)
response.t <- factor(response.t, levels = c(1,2), labels = c("M","B"))


fit_rf <- randomForest(x = predictors, y = response, ntree = 500)

predictions <- predict(fit_rf, newdata = predictors.t)

library(caret)
confusionMatrix(data = predictions, reference = response.t)
```

References

AghaKouchak, Amir, Scott Sellars, and Soroosh Sorooshian. 2013. “Methods of Tail Dependence Estimation.” In *Extremes in a Changing Climate: Detection, Analysis and Uncertainty*, edited by Amir AghaKouchak, David Easterling, Kuolin Hsu, Siegfried Schubert, and Soroosh Sorooshian, 1:163–79. Water Science and Technology Library. Springer Dordrecht. https://doi.org/10.1007/978-94-007-4479-0.

Al, Baba, and Corneliu Catoi. 2007. “Tumor Cell Morphology.” In *Comparative Oncology*, 1st ed., Start page–End page. Bucharest: The Publishing House of the Romanian Academy.

Cancer Council. 2022. “Breast Cancer Statistics.” https://www.canceraustralia.gov.au/cancer-types/breast-cancer/statistics.

Chen, Yenchi. 2024. “Lecture 12: Copula Introduction and Sklar’s Theorem.” Online Lecture Notes. https://faculty.washington.edu/yenchic/Lec12...

Dey, Debangan, and Vadim Zipunnikov. 2022. “Semiparametric Gaussian Copula Regression Modeling for Mixed Data Types (SGCRM),” May.

Fang, Y., and L. Madsen. 2013. “Modified Gaussian Pseudo-Copula: Applications in Insurance and Finance.” *Insurance: Mathematics and Economics* 53 (1): 292–301.

https://doi.org/10.1016/j.insmatheco.2013.05.009.

Hoff, Peter D. 2007. "Extending the Rank Likelihood for Semiparametric Copula Estimation." *Annals of Applied Statistics* 1 (1): 265–83. https://doi.org/10.1214/07-AOAS107.

Mandelbrot, Benoit. 1967. "How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension." *Science* 156 (3775): 636–38. https://doi.org/10.1126/science.156.3775.636.

Michallek, F., H. Huisman, B. Hamm, S. Elezkurtaj, A. Maxeiner, and M. Dewey. 2022. "Prediction of Prostate Cancer Grade Using Fractal Analysis of Perfusion MRI: Retrospective Proof-of-Principle Study." *European Radiology* 32 (5): 3236–47. https://doi.org/10.1007/s00330-021-08394-8.

Street, W. Nick, W. H. Wolberg, and O. L. Mangasarian. 1993. "Nuclear Feature Extraction for Breast Tumor Diagnosis." In *Biomedical Image Processing and Biomedical Visualization*. Vol. 1905. SPIE. https://doi.org/10.1117/12.148698.