Task 3

Task 3: Build a logistic-LASSO model to select features, and implement a path-wise coordinate-wise optimization algorithm to obtain a path of solutions with a sequence of descending λ's.

Reference: Friedman J, Hastie T, Tibshirani R. Regularization Paths for Generalized Linear Models via Coordinate Descent. J Stat Softw. 2010;33(1):1-22. PMID: 20808728; PMCID: PMC2929880.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2929880/#FD14

Algorithm

Log-likelihood f in task 1:

$$f(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{n} \left[Y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \log\left(1 + e^{\mathbf{x}_i^\top \boldsymbol{\beta}}\right)\right]. \tag{1}$$

LASSO estimates the logistic model parameters $\boldsymbol{\beta}$ by optimizing a penalized loss function:

$$\min_{\boldsymbol{\beta}} \; -\frac{1}{n} f(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{p} |\beta_k|. \tag{2}$$

where $\lambda \geq 0$ is the tuning parameter. Note that the intercept is not penalized and all predictors are standardized.

Algorithm Structure

OUTER LOOP: Decrement λ.
MIDDLE LOOP: Update $\tilde{w}_i$, $\tilde{p}_i$, and thus the quadratic approximation ℓ using the current parameters $\tilde{\boldsymbol{\beta}}$.
INNER LOOP: Run the coordinate descent algorithm on the penalized weighted-least-squares problem.

OUTER LOOP In the outer loop, we compute the solutions of the optimization problem (2) for a decreasing sequence of values for λ: $\{\lambda_1, \ldots, \lambda_m\}$, starting at the smallest value $\lambda_1 = \lambda_{max}$ for which the estimates of all coefficients $\hat{\beta}_j = 0, \; j = 1, 2, \ldots, p$, which is

$$\lambda_{max} = \frac{1}{n} \max_j |\langle \mathbf{x}_{\cdot j}, \mathbf{y} \rangle|, \tag{3}$$

where $\mathbf{x}_{\cdot j}$ is the j-th column of the design matrix $\mathbf{X}$, for $j = 1, \ldots, p$.
For tuning parameter value λ_{k+1}, we initialize coordinate descent algorithm at the computed solution for λ_k (warm start). Apart from giving us a path of solutions, this scheme exploits warm starts, and leads to a more stable algorithm.

MIDDLE LOOP In the middle loop, we find the estimates of $\boldsymbol{\beta}$ by solving the optimization problem (2) for a fixed λ. For each iteration of the middle loop, based on the current parameter estimates $\tilde{\boldsymbol{\beta}}$, we form a quadratic approximation to the log-likelihood f using a Taylor expansion:

$$\begin{aligned} f(\boldsymbol{\beta}) \approx \ell(\boldsymbol{\beta}) &= f(\tilde{\boldsymbol{\beta}}) + (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})^\top \nabla f(\tilde{\boldsymbol{\beta}}) + \frac{1}{2}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})^\top \nabla^2 f(\tilde{\boldsymbol{\beta}})(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) \\ &= f(\tilde{\boldsymbol{\beta}}) + [\mathbf{X}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})]^\top (\mathbf{y} - \tilde{\mathbf{p}}) - \frac{1}{2}[\mathbf{X}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})]^\top \tilde{\mathbf{W}} \mathbf{X}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) \\ &= f(\tilde{\boldsymbol{\beta}}) + \sum_{i=1}^n (Y_i - \tilde{p}_i)\mathbf{x}_i^\top (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) - \frac{1}{2}\sum_{i=1}^n \tilde{w}_i \left[\mathbf{x}_i^\top (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})\right]^2 \\ &= -\frac{1}{2}\sum_{i=1}^n \tilde{w}_i \left\{ \left[\mathbf{x}_i^\top (\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta})\right]^2 + 2\frac{Y_i - \tilde{p}_i}{\tilde{w}_i}\left[\mathbf{x}_i^\top (\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta})\right] \right\} + f(\tilde{\boldsymbol{\beta}}) \\ &= -\frac{1}{2}\sum_{i=1}^n \tilde{w}_i \left[\mathbf{x}_i^\top (\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}) + \frac{Y_i - \tilde{p}_i}{\tilde{w}_i}\right] + \frac{1}{2}\sum_{i=1}^n \tilde{w}_i \left(\frac{Y_i - \tilde{p}_i}{\tilde{w}_i}\right)^2 + f(\tilde{\boldsymbol{\beta}}), \end{aligned}$$

where $\tilde{\mathbf{p}} = (\tilde{p}_1, \ldots, \tilde{p}_n)^\top$ and $\tilde{\mathbf{W}} = \mathrm{diag}(\tilde{w}_1, \ldots, \tilde{w}_n)$ are the estimates of $\mathbf{p}$ and $\mathbf{W}$ based on $\tilde{\boldsymbol{\beta}}$. We rewrite the function $\ell(\boldsymbol{\beta})$ as follows:

$$\ell(\boldsymbol{\beta}) = -\frac{1}{2}\sum_{i=1}^n \tilde{w}_i(\tilde{z}_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + C(\tilde{\boldsymbol{\beta}}), \tag{4}$$

where

$$\tilde{z}_i = \mathbf{x}_i^\top \tilde{\boldsymbol{\beta}} + \frac{Y_i - \tilde{p}_i}{\tilde{w}_i}$$

is the working response, $\tilde{w}_i$ is the working weight, and C is a function that does not depend on $\boldsymbol{\beta}$.

INNER LOOP. In the inner loop, we find the estimates of $\boldsymbol{\beta}$ by solving a modified optimization problem of (2). With fixed $\tilde{w}_i$'s, $\tilde{z}_i$'s, and a fixed form of ℓ based on the estimates of $\boldsymbol{\beta}$ in the previous iteration of the middle loop, we use coordinate descent to solve the penalized weighted least-squares problem

$$\min_{\boldsymbol{\beta}} \; -\frac{1}{n}\ell(\boldsymbol{\beta}) + \lambda \sum_{k=1}^p |\beta_k|, \tag{5}$$

and update the estimates of $\boldsymbol{\beta}$. For each iteration of the inner loop, suppose we have the current estimates $\tilde{\beta}_k$ for $k \neq j$ and we wish to partially optimize with respect to β_j:

$$\min_{\beta_j} \; \frac{1}{2n}\sum_{i=1}^n \tilde{w}_i \left(\tilde{z}_i - x_{ij}\beta_j - \sum_{k\neq j} x_{ik}\tilde{\beta}_k\right)^2 + \lambda|\beta_j| + \lambda \sum_{k\neq j}|\tilde{\beta}_k|.$$

Updates:

$$\begin{aligned} \tilde{\beta}_0 &\leftarrow \frac{\sum_{i=1}^n \tilde{w}_i(\tilde{z}_i - \sum_{k=1}^p x_{ik}\tilde{\beta}_k)}{\sum_{i=1}^n \tilde{w}_i}, \\ \tilde{\beta}_j &\leftarrow \frac{S\left(\frac{1}{n}\sum_{i=1}^n \tilde{w}_i x_{ij}(\tilde{z}_i - \sum_{k\neq j} x_{ik}\tilde{\beta}_k), \lambda\right)}{\frac{1}{n}\sum_{i=1}^n \tilde{w}_i x_{ij}^2}, \; j = 1, \ldots, p \end{aligned}$$

where $S(z, \gamma)$ is the soft-thresholding operator with value

$$S(z,\gamma) = \mathrm{sign}(z)(|z| - \gamma)_+ = \begin{cases} z - \gamma, & \text{if } z > 0 \text{ and } \gamma < |z| \\ z + \gamma, & \text{if } z < 0 \text{ and } \gamma < |z| \\ 0, & \text{if } \gamma \geq |z| \end{cases}$$

We can then update estimates of β_j's repeatedly for $j = 0, 1, 2, ..., p, 0, 1, 2, ...$ until convergence.

Note: Care is taken to avoid coefficients diverging in order to achieve fitted probabilities of 0 or 1. When a probability is within $\epsilon = 10^{-5}$ of 1, we set it to 1, and set the weights to ϵ. 0 is treated similarly.

Algorithm 1 Path-wise coordinate-wise optimization algorithm

Require: $g(\boldsymbol{\beta}, \lambda) = -\frac{1}{n} f(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{p} |\beta_k|$ - target function, where $f(\boldsymbol{\beta})$ is given in (1); $\boldsymbol{\beta}_0$ - starting value; $\{\lambda_1, \ldots, \lambda_m\}$ - a sequence of descending λ's, where $\lambda_1 = \lambda_{max}$ is given in (3); ϵ - tolerance; N_s, N_t - maximum number of iterations of the middle and inner loops

Ensure: $\widehat{\boldsymbol{\beta}}(\lambda_r)$ such that $\widehat{\boldsymbol{\beta}}(\lambda_r) \approx \arg\min_{\boldsymbol{\beta}} \; g(\boldsymbol{\beta}, \lambda_r), \; r = 1, \ldots, m$

$\tilde{\boldsymbol{\beta}}_0(\lambda_1) \leftarrow \boldsymbol{\beta}_0$
OUTER LOOP
for $r \in \{1, \ldots, m\}$, where r is the current number of iterations of the outer loop, **do**
    $s \leftarrow 0$, where s is the current number of iterations of the middle loop
    $g(\tilde{\boldsymbol{\beta}}_{-1}(\lambda_r), \lambda_r) \leftarrow \infty$
    MIDDLE LOOP
    while $t \geq 2$ and $s < N_s$ **do**
        $s \leftarrow s + 1$
        Update $\tilde{w}_i^{(s)}$, $\tilde{z}_i^{(s)}$ $(i = 1, \ldots, n)$, and thus $\ell_s(\boldsymbol{\beta})$ as given in (4) based on $\tilde{\boldsymbol{\beta}}_{s-1}(\lambda_r)$
        $t \leftarrow 0$, where t is the current number of iterations of the inner loop
        $\tilde{\boldsymbol{\beta}}_s^{(0)}(\lambda_r) \leftarrow \tilde{\boldsymbol{\beta}}_{s-1}(\lambda_r)$
        $h_s(\tilde{\boldsymbol{\beta}}_s^{(-1)}(\lambda_r), \lambda_r) \leftarrow \infty$, where $h_s(\boldsymbol{\beta}, \lambda) = -\frac{1}{n} \ell_s(\boldsymbol{\beta}) + \lambda \sum_{k=1}^{p} |\beta_k|$
        INNER LOOP
        while $\left| h_s(\tilde{\boldsymbol{\beta}}_s^{(t)}(\lambda_r), \lambda_r) - h_s(\tilde{\boldsymbol{\beta}}_s^{(t-1)}(\lambda_r), \lambda_r) \right| > \epsilon$ and $t < N_t$ **do**
            $t \leftarrow t + 1$
            $\tilde{\beta}_0^{(t)}(\lambda_r) \leftarrow \sum_{i=1}^{n} \tilde{w}_i^{(s)} \left(\tilde{z}_i^{(s)} - \sum_{k=1}^{p} x_{ik} \tilde{\beta}_k^{(t-1)}(\lambda_r) \right) \Big/ \sum_{i=1}^{n} \tilde{w}_i^{(s)}$
            for $j \in \{1, \ldots, p\}$ **do**
                $\tilde{\beta}_j^{(t)}(\lambda_r) \leftarrow S\left(\frac{1}{n} \sum_{i=1}^{n} \tilde{w}_i^{(s)} x_{ij} \left(\tilde{z}_i^{(s)} - \sum_{k<j} x_{ik} \tilde{\beta}_k^{(t)}(\lambda_r) - \sum_{k>j} x_{ik} \tilde{\beta}_k^{(t-1)}(\lambda_r) \right), \lambda_r \right) \Big/ \frac{1}{n} \sum_{i=1}^{n} \tilde{w}_i^{(s)} x_{ij}^2$
            end for
        end while
        $\tilde{\boldsymbol{\beta}}_s(\lambda_r) \leftarrow \tilde{\boldsymbol{\beta}}_s^{(t)}(\lambda_r)$
    end while
    $\widehat{\boldsymbol{\beta}}(\lambda_r) \leftarrow \tilde{\boldsymbol{\beta}}_s(\lambda_r)$
    $\tilde{\boldsymbol{\beta}}_0(\lambda_{r+1}) \leftarrow \widehat{\boldsymbol{\beta}}(\lambda_r)$
end for

Implementation in R

target functions needed to be optimized and soft-threshold operator

```r
# function -ell/n (without C) with penalties (minimize!) used in inner loop's convergence criterion
coordinate_func <- function(X, z, w, betavec, lambda) {
  0.5 * sum(w * (z - X %*% betavec)^2) / nrow(X) + lambda * sum(abs(betavec[-1]))
}

# soft-threshold operator used in inner loop
soft.threshold <- function(z, gamma) {
  sign(z) * max(abs(z) - gamma, 0)
}
```

We implement the algorithm in **R**.

```r
# outer loop
LogisticLASSO <- function(dat, start, lambda) {
  r <- length(lambda)
  X <- as.matrix(cbind(rep(1, nrow(dat)), dat[, -1])) # design matrix
  y <- dat[, 1] # response vector
  res <- matrix(NA, nrow = r, ncol = ncol(dat) + 1)
  for (i in 1:r) {
    betavec <- MiddleLoop(X = X, y = y, start = start, lambda = lambda[i])
    res[i, ] <- c(lambda[i], betavec)
    start <- betavec
  }
  colnames(res) <- c("lambda", "(Intercept)", names(dat)[-1])
  return(res)
}

# middle loop

MiddleLoop <- function(X, y, start, lambda, maxiter = 100) {
  betavec <- start
  u <- X %*% betavec
  p_vec <- sigmoid(u) # function `sigmoid` to compute exp(x)/(1 + exp(x))
  w <- p_vec * (1 - p_vec)
  eps <- 1e-5
  # see note
  p_vec[p_vec < eps] <- 0
  p_vec[p_vec > 1 - eps] <- 1
  w[p_vec == 1 | p_vec == 0] <- eps
  z <- u + (y - p_vec) / w
  s <- 0
  t <- 2
  while (t > 1 && s < maxiter) { # if number of iterations of inner loop = 1, converge.
    s <- s + 1
    betavec <- InnerLoop(X = X, z = z, w = w, betavec = betavec, lambda = lambda)
    t <- betavec[1]
    betavec <- betavec[-1]
    u <- X %*% betavec
  }
  return(betavec)
```

```r
}

# inner loop
InnerLoop <- function(X, z, w, betavec, lambda, tol = 1e-10, maxiter = 1000) {
  prevfunc <- Inf
  curfunc <- coordinate_func(X = X, z = z, w = w, betavec = betavec, lambda = lambda)
  t <- 0
  while (abs(curfunc - prevfunc) > tol && t < maxiter) {
    t <- t + 1
    prevfunc <- curfunc
    betavec[1] <- sum(w * (z - X[, -1] %*% betavec[-1])) / sum(w)
    for (j in 2:length(betavec)) {
      betavec[j] <- soft.threshold(z = sum(w * X[, j] * (z - X[, -j] %*% betavec[-j])) / nrow(X), gamma
    }
    curfunc <- coordinate_func(X = X, z = z, w = w, betavec = betavec, lambda = lambda)
  }
  return(c(t, betavec))
}
```

Model fit on training data

We fit a logistic-LASSO model on the training data using our function `LogisticLASSO` with a sequence of descending λ's.

```r
lambda_max <- max(abs(t(x) %*% y)) / length(y)

lambdas <- exp(seq(log(lambda_max), log(lambda_max) - 10, length = 30))
res <- LogisticLASSO(dat = Training, start = rep(0, ncol(Training)),
                     lambda = lambdas)
res
```

```
##              lambda  (Intercept) radius_mean texture_mean perimeter_mean
##  [1,] 3.979882e-01 -0.517543860    0.000000   0.00000000              0
##  [2,] 2.819120e-01 -0.533897877    0.000000   0.00000000              0
##  [3,] 1.996902e-01 -0.593698075    0.000000   0.00000000              0
##  [4,] 1.414491e-01 -0.646046689    0.000000   0.00000000              0
##  [5,] 1.001944e-01 -0.691239940    0.000000   0.00000000              0
##  [6,] 7.097193e-02 -0.725502588    0.000000   0.00000000              0
##  [7,] 5.027243e-02 -0.748529926    0.000000   0.00000000              0
##  [8,] 3.561010e-02 -0.754581250    0.000000   0.00000000              0
##  [9,] 2.522415e-02 -0.742351771    0.000000   0.10158081              0
## [10,] 1.786733e-02 -0.715204050    0.000000   0.22230995              0
## [11,] 1.265619e-02 -0.672777500    0.000000   0.32694054              0
## [12,] 8.964918e-03 -0.606505846    0.000000   0.42426767              0
## [13,] 6.350232e-03 -0.526588917    0.000000   0.50012864              0
## [14,] 4.498139e-03 -0.455222398    0.000000   0.55486439              0
## [15,] 3.186223e-03 -0.374146491    0.000000   0.54428639              0
## [16,] 2.256937e-03 -0.289568251    0.000000   0.42355906              0
## [17,] 1.598684e-03 -0.171951582    0.000000   0.25014413              0
## [18,] 1.132416e-03 -0.004904588    0.000000   0.10928909              0
## [19,] 8.021384e-04  0.157451909    0.000000   0.00000000              0
## [20,] 5.681887e-04  0.300056546    0.000000   0.00000000              0
```

```
## [21,] 4.024722e-04  0.469856386    0.000000   0.00000000            0
## [22,] 2.850881e-04  0.935719826   -2.150523   0.00000000            0
## [23,] 2.019400e-04  1.554049686   -4.283734   0.00000000            0
## [24,] 1.430427e-04  1.922931999   -5.854787   0.00000000            0
## [25,] 1.013232e-04  2.348929169   -7.157514   0.00000000            0
## [26,] 7.177154e-05  3.038562322   -8.632536   0.09220341            0
## [27,] 5.083883e-05  4.060686796  -10.356141   0.41300570            0
## [28,] 3.601130e-05  5.225265936  -12.303108   0.79125547            0
## [29,] 2.550834e-05  6.301615722  -16.303325   1.11805612            0
## [30,] 1.806864e-05  7.547567728  -28.296403   1.48968203            0
##       area_mean smoothness_mean compactness_mean concavity_mean
##  [1,]  0.000000       0.0000000        0.0000000      0.0000000
##  [2,]  0.000000       0.0000000        0.0000000      0.0000000
##  [3,]  0.000000       0.0000000        0.0000000      0.0000000
##  [4,]  0.000000       0.0000000        0.0000000      0.0000000
##  [5,]  0.000000       0.0000000        0.0000000      0.0000000
##  [6,]  0.000000       0.0000000        0.0000000      0.0000000
##  [7,]  0.000000       0.0000000        0.0000000      0.0000000
##  [8,]  0.000000       0.0000000        0.0000000      0.0000000
##  [9,]  0.000000       0.0000000        0.0000000      0.0000000
## [10,]  0.000000       0.0000000        0.0000000      0.0000000
## [11,]  0.000000       0.0000000        0.0000000      0.0000000
## [12,]  0.000000       0.0000000        0.0000000      0.0000000
## [13,]  0.000000       0.0000000        0.0000000      0.0000000
## [14,]  0.000000       0.0000000        0.0000000      0.0000000
## [15,]  0.000000       0.0000000        0.0000000      0.0000000
## [16,]  0.000000       0.0000000        0.0000000      0.0000000
## [17,]  0.000000       0.0000000       -0.1912138      0.0000000
## [18,]  0.000000       0.0000000       -0.7493055      0.0000000
## [19,]  0.000000       0.0000000       -1.2787734      0.2578362
## [20,]  0.000000       0.0000000       -1.9108781      0.9315239
## [21,]  0.000000       0.0906043       -2.6706512      1.8040434
## [22,]  0.000000       0.5899481       -3.7324323      2.9012870
## [23,]  0.000000       0.8589757       -4.4304619      3.7437976
## [24,]  0.000000       1.1851443       -5.6885636      5.2986213
## [25,]  0.000000       1.5179486       -7.0969174      6.8906384
## [26,]  0.000000       2.0012623       -8.2464293      8.2784926
## [27,]  0.000000       2.3909543       -9.4175049      9.5992603
## [28,]  0.000000       2.7658359      -10.9047218     11.1381550
## [29,]  2.162019       3.3065333      -12.7665350     13.6089973
## [30,] 12.843759       3.9708951      -14.8284661     14.7708252
##       concave.points_mean symmetry_mean fractional_dimension_mean  radius_se
##  [1,]           0.0000000   0.000000000               0.00000000 0.00000000
##  [2,]           0.0000000   0.000000000               0.00000000 0.00000000
##  [3,]           0.0000000   0.000000000               0.00000000 0.00000000
##  [4,]           0.0000000   0.000000000               0.00000000 0.00000000
##  [5,]           0.0283080   0.000000000               0.00000000 0.00000000
##  [6,]           0.1416278   0.000000000               0.00000000 0.00000000
##  [7,]           0.2940330   0.000000000               0.00000000 0.00000000
##  [8,]           0.4595283   0.000000000               0.00000000 0.00000000
##  [9,]           0.5353671   0.000000000               0.00000000 0.09017743
## [10,]           0.5453989   0.000000000               0.00000000 0.26158589
## [11,]           0.5168195   0.000000000               0.00000000 0.48613943
## [12,]           0.5107436   0.000000000               0.00000000 0.82071908
```

```
## [13,]          0.5437438   0.000000000           0.00000000 1.21699801
## [14,]          0.6206920   0.000000000          -0.02933289 1.55440286
## [15,]          0.7237084   0.000000000          -0.10311576 1.89535704
## [16,]          0.9296081   0.000000000          -0.16232842 2.31133791
## [17,]          1.2403706   0.000000000          -0.11945319 2.77646348
## [18,]          1.7514283   0.000000000           0.00000000 3.23534128
## [19,]          2.1092213   0.000000000           0.00000000 3.71853237
## [20,]          2.1886883   0.000000000           0.00000000 4.13181348
## [21,]          2.0526477   0.000000000           0.00000000 3.94776437
## [22,]          1.8823162  -0.006518855           0.00000000 1.96367634
## [23,]          1.9244691  -0.091595515           0.00000000 0.00000000
## [24,]          1.7323825  -0.187263631           0.12386310 0.00000000
## [25,]          1.6246398  -0.281627730           0.34090779 0.00000000
## [26,]          1.4804896  -0.401719772           0.32351604 0.00000000
## [27,]          1.5323789  -0.543502421           0.16581886 0.00000000
## [28,]          1.6911256  -0.686932812           0.00000000 0.89403183
## [29,]          1.6767869  -0.919951257           0.00000000 1.87205504
## [30,]          2.4971485  -1.149376362          -0.10604076 8.53670499
##       texture_se perimeter_se    area_se smoothness_se compactness_se
##  [1,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [2,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [3,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [4,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [5,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [6,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [7,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [8,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
##  [9,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
## [10,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
## [11,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
## [12,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
## [13,]  0.0000000    0.0000000  0.0000000    0.00000000     0.00000000
## [14,]  0.0000000    0.0000000  0.0000000    0.01435153    -0.12149777
## [15,]  0.0000000    0.0000000  0.0000000    0.12926680    -0.42029436
## [16,] -0.1114649    0.0000000  0.0000000    0.26825686    -0.68392244
## [17,] -0.2913033    0.0000000  0.0000000    0.40459573    -0.83969038
## [18,] -0.4453232    0.0000000  0.0000000    0.53015149    -0.86588042
## [19,] -0.5787146    0.0000000  0.0000000    0.65226431    -0.87559422
## [20,] -0.6531722    0.0000000  0.0000000    0.72442901    -0.77393548
## [21,] -0.7181726    0.0000000  0.9970647    0.74527756    -0.54855323
## [22,] -0.7678958    0.0000000  4.6867520    0.87764915     0.00000000
## [23,] -0.7872603   -0.2030798  9.1133939    0.93985024     0.09667533
## [24,] -0.8858881   -1.1569885 11.5469472    0.98836780     0.73406892
## [25,] -1.0490713   -2.2918535 14.6478971    1.00842251     1.43355067
## [26,] -1.2132160   -5.2169007 21.1087294    0.99730436     2.74216505
## [27,] -1.3780718   -9.3417843 30.3011700    0.79603545     4.28652341
## [28,] -1.6230060  -14.6650639 40.4524417    0.47025968     5.88974325
## [29,] -1.9354366  -18.2339995 47.6397017    0.32244805     7.42417490
## [30,] -2.2839320  -26.9577286 51.9859779   -0.31773622     9.80069263
##       concavity_se concave.points_se symmetry_se fractal_dimension_se
##  [1,]   0.00000000         0.0000000  0.00000000           0.00000000
##  [2,]   0.00000000         0.0000000  0.00000000           0.00000000
##  [3,]   0.00000000         0.0000000  0.00000000           0.00000000
##  [4,]   0.00000000         0.0000000  0.00000000           0.00000000
```

```
##  [5,]  0.00000000         0.0000000  0.00000000          0.00000000
##  [6,]  0.00000000         0.0000000  0.00000000          0.00000000
##  [7,]  0.00000000         0.0000000  0.00000000          0.00000000
##  [8,]  0.00000000         0.0000000  0.00000000          0.00000000
##  [9,]  0.00000000         0.0000000  0.00000000          0.00000000
## [10,]  0.00000000         0.0000000  0.00000000          0.00000000
## [11,]  0.00000000         0.0000000  0.00000000          0.00000000
## [12,]  0.00000000         0.0000000  0.00000000         -0.07285232
## [13,]  0.00000000         0.0000000  0.00000000         -0.20133120
## [14,]  0.00000000         0.0000000  0.00000000         -0.23837485
## [15,]  0.00000000         0.0000000 -0.02050779         -0.19071371
## [16,]  0.00000000         0.0000000 -0.08765136         -0.16691519
## [17,] -0.06284107         0.0000000 -0.14656895         -0.16494574
## [18,] -0.17493311         0.0000000 -0.17761924         -0.23432305
## [19,] -0.26579341         0.0000000 -0.20569765         -0.40822274
## [20,] -0.45591558         0.1639715 -0.26004477         -0.78167583
## [21,] -0.80356063         0.7487320 -0.34043908         -1.56749503
## [22,] -1.30306751         1.7448456 -0.44472417         -3.15989112
## [23,] -1.71199764         2.4981744 -0.47868045         -4.02445935
## [24,] -2.32778164         3.3028864 -0.68112546         -5.28923256
## [25,] -2.92849523         4.1226063 -0.94620598         -6.67381782
## [26,] -3.76402706         5.1683994 -1.40163801         -8.72250285
## [27,] -4.83012986         6.4659780 -2.02136668        -10.81678784
## [28,] -6.12986018         8.0510042 -2.72681035        -12.87974995
## [29,] -7.55401701         9.6685389 -3.44387256        -15.25496551
## [30,] -9.94513787        12.1650110 -4.45175691        -17.20744331
##       radius_worst texture_worst perimeter_worst area_worst smoothness_worst
##  [1,]    0.0000000     0.0000000        0.000000  0.0000000       0.00000000
##  [2,]    0.0000000     0.0000000        0.145245  0.0000000       0.00000000
##  [3,]    0.3020222     0.0000000        0.000000  0.0000000       0.00000000
##  [4,]    0.4913272     0.0000000        0.000000  0.0000000       0.00000000
##  [5,]    0.6944021     0.0000000        0.000000  0.0000000       0.00000000
##  [6,]    0.8737915     0.1023149        0.000000  0.0000000       0.00000000
##  [7,]    1.0795904     0.2321073        0.000000  0.0000000       0.00000000
##  [8,]    1.3679269     0.3562374        0.000000  0.0000000       0.06477867
##  [9,]    1.6608009     0.3831170        0.000000  0.0000000       0.16155509
## [10,]    1.9612391     0.3876711        0.000000  0.0000000       0.26993058
## [11,]    2.3016910     0.4031883        0.000000  0.0000000       0.39095494
## [12,]    2.5742102     0.4248715        0.000000  0.0000000       0.50339268
## [13,]    2.8159202     0.4706183        0.000000  0.0000000       0.60745080
## [14,]    3.1037568     0.5440297        0.000000  0.0000000       0.70435895
## [15,]    3.3571853     0.6635502        0.000000  0.0000000       0.72758306
## [16,]    3.5243369     0.9496412        0.000000  0.0000000       0.71114305
## [17,]    3.6864906     1.3233921        0.000000  0.0000000       0.68674928
## [18,]    3.8860821     1.6417291        0.000000  0.0000000       0.67136375
## [19,]    4.0474499     1.9172042        0.000000  0.0000000       0.68133990
## [20,]    4.2889430     2.0535154        0.000000  0.0000000       0.73748821
## [21,]    4.5008413     2.1786772        0.000000  0.0000000       0.76745448
## [22,]    6.9494880     2.3163389        0.000000  0.0000000       0.39093374
## [23,]    9.1331499     2.4816606        0.000000  0.0000000       0.17421616
## [24,]   11.1092859     2.7401278        0.000000  0.0000000       0.00000000
## [25,]   12.8019588     3.0875439        0.000000  0.0000000      -0.08605698
## [26,]   10.5231618     3.3675595        3.783599  0.0000000      -0.32996816
## [27,]    5.8749172     3.4899617        9.860282  0.0000000      -0.42288500
```

```
## [28,]    0.0000000     3.6959652       17.499866  0.0000000      -0.41868191
## [29,]    0.0000000     4.0975441       19.864143  0.0000000      -0.54324081
## [30,]   -6.4840131     4.6562993       31.261960 -0.4097589      -0.51075045
##       compactness_worst concavity_worst concave.points_worst symmetry_worst
##  [1,]          0.000000     0.000000000           0.00000000      0.0000000
##  [2,]          0.000000     0.000000000           0.34482735      0.0000000
##  [3,]          0.000000     0.000000000           0.59316721      0.0000000
##  [4,]          0.000000     0.000000000           0.80972317      0.0000000
##  [5,]          0.000000     0.000000000           0.99819550      0.0000000
##  [6,]          0.000000     0.000000000           1.09544238      0.0000000
##  [7,]          0.000000     0.000000000           1.12454977      0.0317245
##  [8,]          0.000000     0.000000000           1.04052579      0.1044129
##  [9,]          0.000000     0.000000000           0.99991311      0.1847170
## [10,]          0.000000     0.009658227           1.00056933      0.2597712
## [11,]          0.000000     0.066727265           0.98791093      0.3206289
## [12,]          0.000000     0.179773751           0.99252417      0.3673717
## [13,]          0.000000     0.321415835           1.02476861      0.4068182
## [14,]          0.000000     0.489959623           1.07848592      0.4560095
## [15,]          0.000000     0.729120525           1.19257007      0.5494229
## [16,]          0.000000     0.952862425           1.24309997      0.6651787
## [17,]          0.000000     1.208567669           1.27349628      0.7665908
## [18,]          0.000000     1.444576540           1.29601442      0.8420443
## [19,]          0.000000     1.497408396           1.35039872      0.9132582
## [20,]          0.000000     1.372439662           1.35108241      1.0011820
## [21,]          0.000000     1.238240917           1.01701798      1.1033033
## [22,]          0.000000     1.079145371           0.37990992      1.2110735
## [23,]          0.000000     1.085622779           0.07914091      1.4345273
## [24,]          0.000000     0.804072472           0.00000000      1.7272707
## [25,]          0.000000     0.505703639           0.00000000      2.0727192
## [26,]         -1.300965     0.601467673           0.00000000      2.6342496
## [27,]         -3.299825     0.997199513           0.00000000      3.3949619
## [28,]         -5.466035     1.527984421           0.00000000      4.2720986
## [29,]         -6.740660     1.645308812           0.30744500      5.1960278
## [30,]         -9.309517     3.295853438           0.23905971      6.2216611
##       fractal_dimension_worst
##  [1,]               0.0000000
##  [2,]               0.0000000
##  [3,]               0.0000000
##  [4,]               0.0000000
##  [5,]               0.0000000
##  [6,]               0.0000000
##  [7,]               0.0000000
##  [8,]               0.0000000
##  [9,]               0.0000000
## [10,]               0.0000000
## [11,]               0.0000000
## [12,]               0.0000000
## [13,]               0.0000000
## [14,]               0.0000000
## [15,]               0.0000000
## [16,]               0.0000000
## [17,]               0.0000000
## [18,]               0.1642195
## [19,]               0.4758977
```

```
## [20,]                 0.9267988
## [21,]                 1.6726897
## [22,]                 2.7653288
## [23,]                 3.3972251
## [24,]                 4.1180966
## [25,]                 4.7951488
## [26,]                 6.2727073
## [27,]                 8.1060045
## [28,]                10.0764934
## [29,]                11.7478322
## [30,]                13.5016898
```