Tinkering

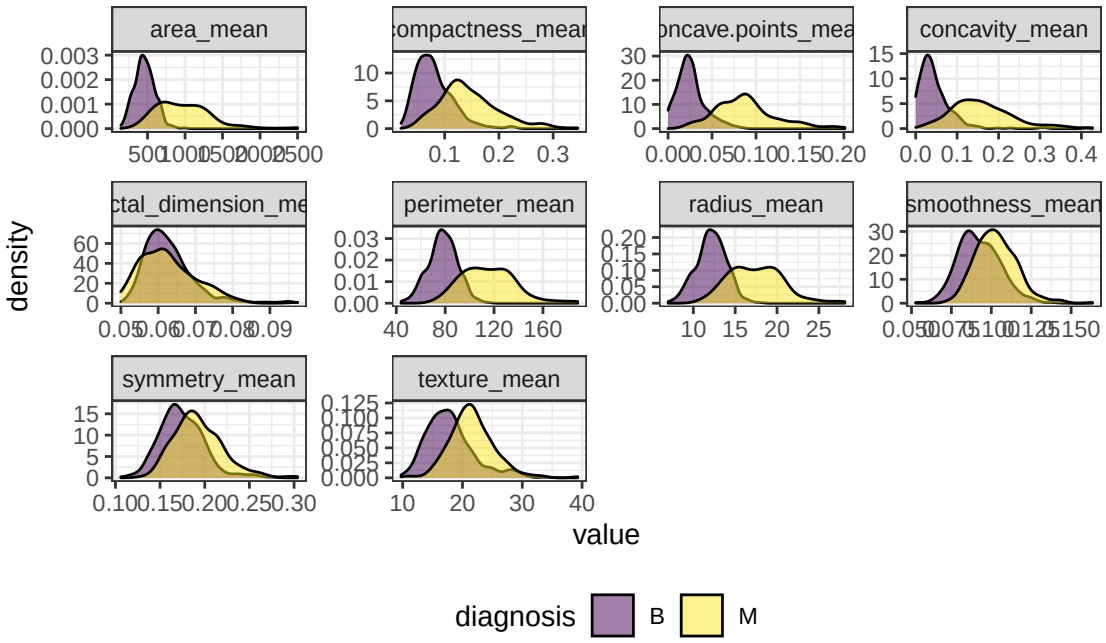
Waveley Qiu (wq2162)

2022-03-17

EDA

Let's import and take a look at the data.

Let's take a look at the distributions of other variables.




```
tbl_summary(bc, by = diagnosis)
```

```
## Table printed with `knitr::kable()`, not {gt}. Learn why at
## https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
## To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	**B**, N = 357	**M**, N = 212
id	908,916 (874,662, 8,812,816)	895,366 (861,345, 8,911,290)
radius_mean	12.2 (11.1, 13.4)	17.3 (15.1, 19.6)
texture_mean	17.4 (15.2, 19.8)	21.5 (19.3, 23.8)
perimeter_mean	78 (71, 86)	114 (99, 130)
area_mean	458 (378, 551)	932 (705, 1,204)
smoothness_mean	0.091 (0.083, 0.101)	0.102 (0.094, 0.111)
compactness_mean	0.08 (0.06, 0.10)	0.13 (0.11, 0.17)
concavity_mean	0.04 (0.02, 0.06)	0.15 (0.11, 0.20)
concave.points_mean	0.02 (0.02, 0.03)	0.09 (0.06, 0.10)

Characteristic	**B**, N = 357	**M**, N = 212
symmetry_mean	0.171 (0.158, 0.189)	0.190 (0.174, 0.210)
fractal_dimension_mean	0.062 (0.059, 0.066)	0.062 (0.057, 0.067)
radius_se	0.26 (0.21, 0.34)	0.55 (0.39, 0.76)
texture_se	1.11 (0.80, 1.49)	1.10 (0.89, 1.43)
perimeter_se	1.85 (1.45, 2.39)	3.68 (2.72, 5.21)
area_se	20 (15, 25)	58 (36, 94)
smoothness_se	0.0065 (0.0052, 0.0085)	0.0062 (0.0051, 0.0080)
compactness_se	0.016 (0.011, 0.026)	0.029 (0.020, 0.039)
concavity_se	0.018 (0.011, 0.031)	0.037 (0.027, 0.050)
concave.points_se	0.009 (0.006, 0.012)	0.014 (0.011, 0.017)
symmetry_se	0.019 (0.016, 0.024)	0.018 (0.015, 0.022)
fractal_dimension_se	0.0028 (0.0021, 0.0042)	0.0037 (0.0027, 0.0049)
radius_worst	13.3 (12.1, 14.8)	20.6 (17.7, 23.8)
texture_worst	22.8 (19.6, 26.5)	28.9 (25.8, 32.7)
perimeter_worst	87 (78, 97)	138 (119, 160)
area_worst	547 (447, 670)	1,303 (970, 1,713)
smoothness_worst	0.125 (0.110, 0.138)	0.143 (0.130, 0.156)
compactness_worst	0.17 (0.11, 0.23)	0.36 (0.24, 0.45)
concavity_worst	0.14 (0.08, 0.22)	0.40 (0.33, 0.56)
concave.points_worst	0.07 (0.05, 0.10)	0.18 (0.15, 0.21)
symmetry_worst	0.27 (0.24, 0.30)	0.31 (0.28, 0.36)
fractal_dimension_worst	0.077 (0.070, 0.085)	0.088 (0.076, 0.103)

```
bc <- bc %>% mutate(bin_out = ifelse(diagnosis == "M", 1, 0)) %>% relocate(bin_out)
```

Logistic Function

The likelihood function is defined as follows:

$$f(\beta_0, \beta_1, ..., \beta_{30}) = \sum_{i=1}^{n} \left(Y_i \left(\beta_0 + \sum_{j=1}^{30} \beta_j x_{ij} \right) - \log(1 + e^{\left(\beta_0 + \sum_{j=1}^{30} \beta_j x_{ij}\right)}) \right)$$

Let $\pi_i = \frac{e^{\beta_0 + \sum_{j=1}^{30} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{30} \beta_j x_{ij}}}$. Then, the gradient of this function is defined as follows:

$$\nabla f(\beta_0, \beta_1, ..., \beta_{30}) = \begin{pmatrix} \sum_{i=1}^{n} Y_i - \pi_i \\ \sum_{i=1}^{n} x_{i1}(Y_i - \pi_i) \\ \sum_{i=1}^{n} x_{i2}(Y_i - \pi_i) \\ \vdots \\ \sum_{i=1}^{n} x_{i30}(Y_i - \pi_i) \end{pmatrix}$$

Finally, we define the Hessian of this function as follows:

$$\nabla^2 f(\beta_0, \beta_1, ..., \beta_{30}) = -\sum_{i=1}^{n} \begin{pmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{i30} \end{pmatrix} (1 \; x_{i1} \; x_{i2} \; \dots \; x_{i30}) \, \pi_i(1-\pi_i)$$

$$= -\begin{pmatrix} \sum_{i=1}^n \pi_i(1-\pi_i) & \sum_{i=1}^n x_{i1}\pi_i(1-\pi_i) & \dots & \sum_{i=1}^n x_{i30}\pi_i(1-\pi_i) \\ \sum_{i=1}^n x_{i1}\pi_i(1-\pi_i) & \sum_{i=1}^n x_{i1}^2\pi_i(1-\pi_i) & \dots & \sum_{i=1}^n x_{i30}x_{i1}\pi_i(1-\pi_i) \\ \sum_{i=1}^n x_{i2}\pi_i(1-\pi_i) & \sum_{i=1}^n x_{i1}x_{i2}\pi_i(1-\pi_i) & \dots & \sum_{i=1}^n x_{i30}x_{i2}\pi_i(1-\pi_i) \\ \vdots & \ddots & \ddots & \vdots \\ \sum_{i=1}^n x_{i30}\pi_i(1-\pi_i) & \sum_{i=1}^n x_{i1}x_{i30}\pi_i(1-\pi_i) & \dots & \sum_{i=1}^n x_{i30}^2\pi_i(1-\pi_i) \end{pmatrix}$$

$$= (1 \; x_{i1} \; x_{i2} \; \dots \; x_{i30}) \, I(\pi_i(1-\pi_i)) \begin{pmatrix} 1 \\ x_{i1} \\ x_{i2} \\ \vdots \\ x_{i30} \end{pmatrix}$$

```r
rep_col <- function(x, n){
  matrix(rep(x, each = n), ncol = n, byrow = TRUE)
}

logistic_stuff <- function(dat, beta){

  x <- dat[[1]] %>% unname() %>% as.matrix()
  y <- dat[[2]] %>% unname() %>% as.matrix()

  x_with_1 <- cbind(1, x)

  u <- x_with_1 %*% beta
 # return(u)

  expu <- exp(u)

  loglik <- sum(y*u - log(1 + expu))

  p <- expu/(1 + expu)
  #  return(p)
  # return(p)
  grad <- t(x_with_1) %*% (y - p)

  i_mat <- diag(nrow(p))
  diag(i_mat) <- p*(1 - p)

  hess <- -(t(x_with_1) %*% i_mat %*% x_with_1)
  return(list(
    loglik = loglik,
    grad = grad,
    hess = hess
  ))
}
```

```r
NewtonRaphson <- function(dat, func, start, tol = 1e-8, maxiter = 200) {
  i <- 0
  cur <- start
  stuff <- func(dat, cur)
  res <- c(0, stuff$loglik, cur)
  prevloglik <- -Inf

  while (i < maxiter && abs(stuff$loglik - prevloglik) > tol && !is.na(stuff$loglik)) {
    i <- i + 1
    prevloglik <- stuff$loglik
    prev <- cur
    newhess <- ((stuff$hess + t(stuff$hess))/2)

    if (!is.negative.definite(newhess)) { # redirection
     while (!is.negative.definite(newhess)) {
       # subtracts identity matrix until a negative definite matrix is achieved
        newhess1 <- newhess - 0.0001*diag(31)
       # sanity check print("changing ascent direction")
        newhess <- ((newhess1 + t(newhess1))/2)
      }
    }

    cur <- prev - solve(newhess) %*% stuff$grad
    stuff <- func(dat, cur)

    if (stuff$loglik < prevloglik) {  # back tracking (half-step)
      j = 1
      while (stuff$loglik < prevloglik & (!is.na(stuff$loglik))) {
         halfstep = 1/(2^j)
         cur <- prev - halfstep*solve(newhess) %*% stuff$grad
         stuff <- func(dat, cur)
        # sanity check print("backtracking")
         j = j + 1
      }
    }
    res <- rbind(res, c(i, stuff$loglik, cur))
  }
  return(res)
}
```

```r
beta_init <- rep(0.001, 31) %>% as.matrix()

test1 <- logistic_stuff(
  list(x = bc[,-c(1,2, 3)] %>% as.matrix(),
       y = bc$bin_out %>% as.matrix()),
  beta = beta_init)

ans <- NewtonRaphson(
      list(x = bc[,-c(1,2, 3)] %>% as.matrix(),
       y = bc$bin_out %>% as.matrix()),
       logistic_stuff,
       beta_init)
ans
```

```
##     [,1]       [,2]        [,3]       [,4]         [,5]       [,6]        [,7]
## res    0 -565.43715   0.0010000   0.001000  0.001000000 0.00100000 0.001000000
##        1  -91.70414   0.5019526  -3.034802  0.066302302 0.21749314 0.011675675
##        2  -52.07094 -11.1756036  -1.604133  0.032895270 0.17713809 0.006297497
##        3  -35.67767 -15.5646769  -1.621475 -0.002842022 0.16258873 0.006054080
##        4  -26.50596 -22.3247328  -1.452758 -0.031546371 0.12371404 0.002134744
##        5  -20.80028 -35.3917233  -1.494781 -0.057434339 0.01908591 0.001013839
##        6  -17.34138 -49.9123673  -4.119278 -0.055631300 0.02218410 0.019576324
##        7  -15.08349 -52.7651387 -11.331115  0.028550517 0.20864300 0.073663152
##        8  -13.12769 -35.6788855 -25.845068  0.236402872 0.52339212 0.193097692
##        9        NaN         NaN        NaN          NaN        NaN         NaN
##          [,8]       [,9]     [,10]     [,11]      [,12]        [,13]      [,14]
## res   0.00100    0.00100   0.00100   0.00100   0.001000   0.00100000  0.0010000
##      24.04119  -38.42539  30.24004  12.46111  -2.675706  -0.01313438  1.7089451
##      27.09336  -30.47092  20.15599  10.55850  -5.656418  29.13492203  6.6387615
##      29.24348  -31.99905  24.80059  16.95479  -4.803026  10.77636237  7.8584617
##      45.41899  -42.02061  33.05738  31.26619  -4.523860  -1.84957666  6.8536444
##      92.40134  -66.77641  52.58713  45.36333  -9.722562   7.40875188  3.7111131
##     162.83707 -110.36850  76.37227  62.86011 -21.516940  59.37049255  1.7644960
##     252.80239 -175.15176  92.45599 104.33310 -41.209673 115.50419136  0.6632445
##     374.22318 -274.06617 112.37454 197.47360 -81.753588 159.02481490 -8.7382958
##           NaN        NaN       NaN       NaN        NaN          NaN        NaN
##          [,15]      [,16]        [,17]     [,18]      [,19]      [,20]
## res  0.00100000  0.0010000  0.001000000   0.00100   0.001000    0.00100
##     -0.06268972 -0.2280046  0.006576908  84.36835   2.853067  -33.22438
##     -0.52659403 -0.2264069 -0.003858034 157.18012  -1.549310  -32.61826
##     -0.92257398 -0.3563812  0.012412988 205.17913   9.001682  -43.58111
##     -1.32795053 -0.4293594  0.049191534 271.58020  33.506699  -59.97693
##     -1.88582818 -0.4681623  0.106601948 355.48248  94.533150  -93.49754
##     -2.53533725 -0.8234937  0.182962680 435.50312 192.817957 -151.22670
##     -3.06626175 -1.8819717  0.314232944 450.42757 321.120013 -240.23191
##     -3.68591070 -3.5505959  0.632005969 229.40481 491.982391 -394.74613
##            NaN        NaN          NaN       NaN        NaN        NaN
##          [,21]       [,22]       [,23]      [,24]      [,25]         [,26]
## res    0.00100    0.001000     0.00100  0.0010000 0.00100000  0.0010000000
##       68.98794    6.669865   -84.74908  0.3125715 0.02036321 -0.0071125658
##       68.10467  -34.156078  -190.94920 -0.1156485 0.13488936 -0.0124731726
##      120.09037  -36.651386  -417.88459 -0.1624562 0.22135186  0.0019629257
##      236.27769  -49.687877  -912.93126 -0.1426128 0.30872768  0.0006843888
##      451.60696  -76.568235 -1807.44049  0.5503240 0.42473946 -0.0043949247
##      734.06497 -119.656177 -2961.04485  2.1854084 0.56397131  0.0150189249
##     1115.69295 -182.301549 -4232.92112  4.5661338 0.68845193  0.0951485380
##     1835.82875 -331.377149 -5753.56782  9.2644874 0.82505208  0.1906329405
##            NaN         NaN         NaN        NaN        NaN           NaN
##           [,27]       [,28]       [,29]      [,30]      [,31]      [,32]
## res  0.001000000    0.001000   0.001000  0.001000   0.001000  0.001000
##      0.003447118   -5.800362   4.560373 -1.157068   0.613049  5.318947
##      0.004564451  -15.566247   1.388600  3.138799   3.766161 10.992641
##      0.006643111  -16.218220  -1.592291  5.099026   3.456398 12.614340
##      0.011726477  -23.957150  -4.560608  6.994139  -1.410753 15.560988
##      0.014360180  -43.346543 -10.146391  8.860869  -8.857547 21.276454
##      0.007658510  -68.324200 -18.834672 13.971995 -11.895674 31.279174
##     -0.008797066  -96.897606 -29.803609 26.065574 -10.882198 46.547232
##     -0.045430973 -131.405176 -43.753168 47.819582 -22.884306 80.073529
```

```
##              NaN         NaN        NaN        NaN         NaN        NaN
##          [,33]
## res   0.00100
##      20.78799
##      29.30287
##      55.37323
##     106.79220
##     200.59119
##     307.04914
##     416.27914
##     555.58436
##           NaN
```

The beta estimates are as follows:

```
if (sum(is.na(ans[nrow(ans),])) > 0) {
  beta_est <- ans[nrow(ans) - 1, -c(1,2)]
}

if (sum(is.na(ans[nrow(ans),])) == 0) {
  beta_est <- ans[nrow(ans), -c(1,2)]
}

tibble(beta_subscript = seq(0, 30), beta_estimates = beta_est) %>% knitr::kable()
```

beta_subscript	beta_estimates
0	-35.6788855
1	-25.8450679
2	0.2364029
3	0.5233921
4	0.1930977
5	374.2231831
6	-274.0661692
7	112.3745441
8	197.4735980
9	-81.7535878
10	159.0248149
11	-8.7382958
12	-3.6859107
13	-3.5505959
14	0.6320060
15	229.4048089
16	491.9823911
17	-394.7461298
18	1835.8287537
19	-331.3771487
20	-5753.5678242
21	9.2644874
22	0.8250521
23	0.1906329
24	-0.0454310
25	-131.4051756
26	-43.7531684
27	47.8195818

beta_subscript	beta_estimates
28	-22.8843059
29	80.0735290
30	555.5843554