TMA4315: Solution for exercise 1

1

a)

Let the random variable Y be Bernoulli distributed with success probability $p = \Phi(\eta)$, where Φ is the Gaussian CDF, and $\eta = \mathbf{x}^T\boldsymbol{\beta}$ is a linear predictor with explanatory variables $\mathbf{x}$ and regression coefficients $\boldsymbol{\beta}$.

The probability mass function of Y is

$$f(y) = p^y(1-p)^{1-y}.$$

Thus, given n independent realisations $\boldsymbol{y} = (y_1, \dots, y_n)^T$ of Y and the design matrix $\mathbf{X} = (\mathbf{x}_1 \cdots \mathbf{x}_n))^T$, the likelihood equals

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i},$$

where $p_i = \Phi(\mathbf{x}_i^T\boldsymbol{\beta})$. This gives the log-likelihood

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i) = \sum_{i=1}^{n} y_i \log \Phi(\mathbf{x}_i^T\boldsymbol{\beta}) + (1-y_i)\log(1-\Phi(\mathbf{x}_i^T\boldsymbol{\beta})).$$

The score function is the gradient of the log-likelihood. We get

$$s(\boldsymbol{\beta}) = \frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \frac{\partial l_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}},$$

with $l_i(\boldsymbol{\beta}) = y_i \log p_i + (1-y_i)\log(1-p_i)$. We get

$$\frac{\partial l_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{\partial l_i(\boldsymbol{\beta})}{\partial p_i}\frac{\partial p_i}{\partial \boldsymbol{\beta}} = \left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\right)\phi(\mathbf{x}_i^T\boldsymbol{\beta})\mathbf{x}_i.$$

Thus, the score function is equal to

$$s(\boldsymbol{\beta}) = \sum_{i=1}^{n}\left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\right)\phi(\mathbf{x}_i^T\boldsymbol{\beta})\mathbf{x}_i = \mathbf{X}^T\left(\phi(\boldsymbol{\eta})\left(\frac{\boldsymbol{y}}{\boldsymbol{p}} - \frac{1-\boldsymbol{y}}{1-\boldsymbol{p}}\right)\right),$$

where $\phi(\boldsymbol{\eta}) = (\phi(\eta_1), \dots, \phi(\eta_n))^T$ and $\boldsymbol{p} = (p_1, \dots, p_n)^T$.

The observed Fisher information matrix is the negative Hessian matrix of the log-likelihood,

$$H(\boldsymbol{\beta}) = -\frac{\partial s(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T} = -\sum_{i=1}^{n}\frac{\partial s_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T},$$

with $s_i(\boldsymbol{\beta}) = \frac{\partial l_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$. We get

$$\begin{aligned}
\frac{\partial s_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T} &= \frac{\partial s_i(\boldsymbol{\beta})}{\partial \eta_i}\frac{\partial \eta_i}{\partial \boldsymbol{\beta}^T} \\
&= \mathbf{x}_i\left(\phi'(\eta_i)\left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\right) - \phi(\eta_i)^2\left(\frac{y_i}{p_i^2} + \frac{1-y_i}{(1-p_i)^2}\right)\right)\frac{\partial \eta_i}{\partial \boldsymbol{\beta}^T} \\
&= \mathbf{x}_i\left(\phi'(\eta_i)\left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}\right) - \phi(\eta_i)^2\left(\frac{y_i}{p_i^2} + \frac{1-y_i}{(1-p_i)^2}\right)\right)\mathbf{x}_i^T.
\end{aligned}$$

This means that the observed Fisher information matrix is equal to

$$H(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \cdot \left(\phi'(\eta_i) \left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i} \right) - \phi(\eta_i)^2 \left(\frac{y_i}{p_i^2} + \frac{1-y_i}{(1-p_i)^2} \right) \right) = -\mathbf{X}^T \mathbf{D} \mathbf{X},$$

where $\mathbf{D}$ is a diagonal matrix with element nr. i equal to $\phi'(\eta_i) \left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i} \right) - \phi(\eta_i)^2 \left(\frac{y_i}{p_i^2} + \frac{1-y_i}{(1-p_i)^2} \right)$.

The expected Fisher information matrix is the expected value of the observed Fisher information matrix. Since $Ey_i = p_i$ we get that

$$E\left[\phi'(\eta_i) \left(\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i} \right) - \phi(\eta_i)^2 \left(\frac{y_i}{p_i^2} + \frac{1-y_i}{(1-p_i)^2} \right) \right] = \phi'(\eta_i)(1-1) - \phi(\eta_i)^2 \left(\frac{1}{p_i} + \frac{1}{1-p_i} \right),$$

which gives

$$F(\boldsymbol{\beta}) = \mathbf{X}^T \mathbf{V} \mathbf{X},$$

where $\mathbf{V}$ is a diagonal matrix with element nr. i equal to $\frac{\phi(\eta_i)^2}{p_i(1-p_i)}$.

b)

We start by implementing all the functions from problem 1a:

```
loglik_p = function(y, p) sum(dbinom(y, 1, p, log = TRUE))
loglik = function(y, X, beta) {
  p = as.numeric(pnorm(X %*% beta))
  loglik_p(y, p)
}

score = function(y, X, beta) {
  eta = as.numeric(X %*% beta)
  p = pnorm(eta)
  # Use ifelse to ensure that we don't divide 0 by 0 and get NaNs
  rhs = dnorm(eta) * (ifelse(y == 0, 0, y / p) - ifelse(y == 1, 0, (1 - y) / (1 - p)))
  as.numeric(t(X) %*% rhs)
}

observed_fisher_information= function(y, X, beta) {
  eta = as.numeric(X %*% beta)
  p = pnorm(eta)
  phi_der = function(x) -x / sqrt(2 * pi) * exp(-x^2 / 2)
  middle = phi_der(eta) * (ifelse(y == 0, 0, y / p) - ifelse(y == 1, 0, (1 - y) / (1 - p))) -
    dnorm(eta)^2 * (ifelse(y == 0, 0, y / p^2) + ifelse(y == 1, 0, (1 - y) / (1 - p)^2))

  - t(X) %*% diag(middle) %*% X
}

expected_fisher_information = function(X, beta) {
  eta = as.numeric(X %*% beta)
  p = pnorm(eta)
  middle = dnorm(eta)^2 / ifelse(p == 1 | p == 0, 1, (p * (1 - p)))
  t(X) %*% diag(middle) %*% X
}
```

Then we use numerical derivation to test if these are correct.

```r
library(numDeriv)

n = 20
k = 3

set.seed(1)

score_diff = NULL
observed_diff = NULL
expected_diff = NULL
for (i in 1:10) {
  X = cbind(1, matrix(rnorm(n * k), nrow = n, ncol = k))
  b = rnorm(k + 1)

  # Add bad data to make it more tricky
  X = rbind(X, c(1, 9999, rep(0, k - 1)), c(1, -9999, rep(0, k - 1)))

  p = as.numeric(pnorm(X %*% b))
  y = rbinom(length(p), 1, p)

  score_diff[i] = mean(
    (numDeriv::grad(function(b) loglik(y, X, b), b) - score(y, X, b)) /
      score(y, X, b))
  observed_diff[i] = mean(
    (numDeriv::hessian(function(b) loglik(y, X, b), b) + observed_fisher_information(y, X, b)) /
      observed_fisher_information(y, X, b))

  h = 0
  for (j in 1:1000) {
      y = rbinom(length(p), 1, p)
      h = h + observed_fisher_information(y, X, b)
  }
  h = h / 1000
  expected_diff[i] = mean(
    (h - expected_fisher_information(X, b)) / expected_fisher_information(X, b))

}

summary(score_diff)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -4.578e-09 -4.580e-11 -1.920e-11 -2.984e-10  1.789e-11  1.666e-09
```

```r
summary(observed_diff)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -6.634e-11  3.796e-12  1.599e-11  7.703e-11  2.012e-11  5.688e-10
```

```r
summary(expected_diff)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.006329 -0.002875  0.002767  0.010706  0.006732  0.097933
```

Everything seems to work! Now we write the actual myglm function. This is based on the Fisher scoring algorithm where we estimate $\boldsymbol{\beta}$ iteratively by $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + F(\boldsymbol{\beta}^{(t)})^{-1} s(\boldsymbol{\beta}^{(t)})$. Inside the myglm function we

also compute the deviance, which is equal to

$$D = -2(l(\text{candidate model}) - l(\text{saturated model})),$$

where the candidate model is our implemented model, and the saturated model is a model where we set $p_i = y_i$.

```
myglm = function(formula, data, start = rep(0, ncol(model.matrix(formula, data)))) {
  response_name = as.character(formula)[2]
  y = data[[response_name]]
  X = model.matrix(formula, data)
  beta = start
  s = 1
  while (s > 1e-10) {
    score_val = score(y, X, beta)
    fisher_val = expected_fisher_information(X, beta)
    beta = beta + solve(fisher_val, score_val)
    s = sum(score_val^2)
  }
  vcov_mat = solve(fisher_val) # covariance matrix
  p_hat = as.numeric(pnorm(X %*% beta))
  loglik_saturated = loglik_p(y, y)
  loglik = loglik_p(y, p_hat)
  D = 2 * sum(loglik_saturated - loglik) # deviance
  coefficients = cbind(beta, sqrt(diag(vcov_mat)))
  colnames(coefficients) = c("estimate", "se")
  rownames(coefficients) = paste0("beta_", seq_along(beta))
  list(coefficients = coefficients, deviance = D, vcov = vcov_mat)
}
```

c)

We draw some random data using the `rbinom` function, and then we compare the results from `myglm` with those from `glm`. We also compare the results with the actual truth. This is performed 10 times, with new random values each time

```
n = 5000
k = 3

set.seed(1)

coeff_diff = NULL
vcov_diff = NULL
truth_diff = NULL
for (i in 1:10) {

  X = matrix(rnorm(n * k), nrow = n, ncol = k)
  b = rnorm(k + 1)

  p = as.numeric(pnorm(X %*% b[-1] + b[1]))
  y = rbinom(n, 1, p)

  df = as.data.frame(X)
  df$y = y
  formula = y~.
```

```
  res = myglm(formula, df)
  res2 = glm(formula, family = binomial(link = "probit"), df)

  coeff_diff[i] = mean( (res$coefficients[, 1] - res2$coefficients) ^ 2)
  vcov_diff[i] = mean( (res$vcov - vcov(res2)) ^ 2)
  truth_diff[i] = mean( (res$coefficients[, 1] - b) ^ 2)
}

summary(coeff_diff)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 5.977e-16 2.223e-15 1.828e-14 1.342e-13 8.440e-14 9.632e-13
```

```
summary(vcov_diff)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 1.985e-18 2.369e-17 4.547e-17 1.375e-15 1.569e-15 8.307e-15
```

```
summary(truth_diff)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 6.355e-05 2.152e-04 4.718e-04 5.376e-04 6.872e-04 1.249e-03
```

Once again, everything is working nicely. The difference between estimates from `myglm` and `glm` is really small, and the estimates are close to the true values.

2

a)

We fit the probit model using the built-in `glm` function:

```
library(ISwR) # Install the package if needed
data(juul)
juul$menarche <- juul$menarche - 1
juul.girl <- subset(juul,age>8 & age<20 & complete.cases(menarche))

res = glm(
  formula = menarche~age,
  family = binomial(link = "probit"),
  data = juul.girl)

summary(res)
```

```
##
## Call:
## glm(formula = menarche ~ age, family = binomial(link = "probit"),
##     data = juul.girl)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.32986  -0.15223   0.00028   0.07228   2.48281
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.37033    1.06346  -10.69   <2e-16 ***
```

```
## age           0.86233    0.08106   10.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 719.39  on 518  degrees of freedom
## Residual deviance: 197.39  on 517  degrees of freedom
## AIC: 201.39
##
## Number of Fisher Scoring iterations: 8
```

Now, we use a Wald-test to test if there is an effect of age on the probability that a girl has had her first period. The Wald-statistic for the hypothesis

$$H_0 : \boldsymbol{C\beta} = \boldsymbol{d} \text{ vs. } H_1 : \boldsymbol{C\beta} \neq \boldsymbol{d}$$

is equal to

$$\boldsymbol{w} = (\boldsymbol{C\hat{\beta}} - \boldsymbol{d})^T [\boldsymbol{C} F^{-1}(\hat{\boldsymbol{\beta}}) \boldsymbol{C}^T]^{-1} (\boldsymbol{C\hat{\beta}} - \boldsymbol{d}),$$

and is χ^2 distributed with r degrees of freedom under H_0, where r is the rank of $\boldsymbol{C}$. We perform the Wald-test and compute a p-value:

```
C = matrix(c(0, 1), nrow = 1)
d = 0
beta_hat = res$coefficients
w = t(C %*% beta_hat - d) %*% solve(C %*% vcov(res) %*% t(C), C %*% beta_hat - d)
print(1 - pchisq(as.numeric(w), 1))
```

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

The p-value is so small that we reject H_0 at all meaningful levels, so we can assume that age has an effect in our model.

b)

Assume that the time of the first period of a girl has a Gaussian distribution $T \sim \mathcal{N}(\mu, \sigma^2)$, with mean μ and variance σ^2. The probability that girl nr. i has had her first period before the time t_i is then

$$p_i = P(T_i \leq t_i) = \Phi(\frac{t_i - \mu}{\sigma}).$$

Now, the probit model states that

$$p_i = \Phi(\beta_0 + \beta_1 t_i),$$

so if we set $\mu = -\beta_0/\beta_1$ and $\sigma = 1/\beta_1$, then these two models correspond perfectly.

Using the invariance property of MLEs, we find the MLEs of μ and σ as $\hat{\mu} = -\hat{\beta}_0/\hat{\beta}_1$ and $\hat{\sigma} = 1/\hat{\beta}_1$. We use the delta method for estimating the standard deviations of $\hat{\mu}$ and $\hat{\sigma}$. A linearisation of the function $f(\boldsymbol{\beta}) = (\mu, \sigma)^T$ around the point $\boldsymbol{\beta}$ gives

$$f(\hat{\boldsymbol{\beta}}) \approx f(\boldsymbol{\beta}) + J(\boldsymbol{\beta})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}),$$

where $J(\boldsymbol{\beta})$ is the Jacobian of $f(\boldsymbol{\beta})$. This gives us an estimate for the variance

$$\text{Cov}(f(\hat{\boldsymbol{\beta}})) \approx J(\boldsymbol{\beta}) \text{Cov}(\hat{\boldsymbol{\beta}}) J(\boldsymbol{\beta})^T.$$

Of course, we don't know the true value of $\boldsymbol{\beta}$, but we can further approximate this as

$$\text{Cov}(f(\hat{\boldsymbol{\beta}})) \approx J(\hat{\boldsymbol{\beta}}) \text{Cov}(\hat{\boldsymbol{\beta}}) J(\hat{\boldsymbol{\beta}})^T.$$

We compute the covariance matrix:

```r
jacobian = function(beta) matrix(c(-1 / beta[2], 0, beta[1] / beta[2]^2, -1 / beta[2]), 2, 2)
J = jacobian(beta_hat)
cov_mat = J %*% vcov(res) %*% t(J)
cov_mat
```

```
##              [,1]        [,2]
## [1,] 0.014049085 0.001119651
## [2,] 0.001119651 0.008836764
```

This gives an estimated standard deviation of 0.1185 for $\hat{\mu}$ and 0.0940 for $\hat{\sigma}$.

c)

We fit a new binary regression model where we use the logit link function

$$logit(p_i) = \log \frac{p_i}{1-p_i} = \eta_i \iff p_i = \frac{1}{1+\exp(-\eta_i)}$$

```r
res_logit = glm(
  formula = menarche~age,
  family = binomial(link = "logit"),
  data = juul.girl)
summary(res_logit)
```

```
##
## Call:
## glm(formula = menarche ~ age, family = binomial(link = "logit"),
##     data = juul.girl)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.32759  -0.18998   0.01253   0.12132   2.45922
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.0132     2.0284  -9.867   <2e-16 ***
## age           1.5173     0.1544   9.829   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 719.39  on 518  degrees of freedom
## Residual deviance: 200.66  on 517  degrees of freedom
## AIC: 204.66
##
## Number of Fisher Scoring iterations: 7
```

This model is corresponds to a different latent model for the age T of a girls first period. We get

$$P(T_i \leq t_i) = p_i = \frac{1}{1+\exp(-\beta_0 - \beta_1 t_i)},$$

which we recognise as the CDF of the logistic distribution

$$F_T(t) = \frac{1}{1+\exp(-(t-\mu)/s)},$$

with parameters $\mu = -\beta_0/\beta_1$ and $s = 1/\beta_1$. We know that the mean of the logistic distribution is μ and the standard deviation is $s\pi/\sqrt{3}$. Using the invariance property of the MLE we estimate these variables and get

$$\widehat{E[T]} = 13.1901, \quad \widehat{\mathrm{SD}(T)} = 1.1954.$$

```
beta_hat_logit = res_logit$coefficients
mu_T = -beta_hat_logit[1] / beta_hat_logit[2]
sigma_T = pi / sqrt(3) / beta_hat_logit[2]
print(unname(c(mu_T, sigma_T)))
```

```
## [1] 13.190115  1.195421
```

d)

We know assume that $T \sim \log \mathcal{N}(\mu, \sigma^2)$, which means that $\log T$ has a Gaussian distribution with mean μ and variance σ^2. This gives the expression for p_i:

$$p_i = P(T_i \leq t_i) = P(\log T_i \leq \log t_i) = \Phi(\frac{\log t_i - \mu}{\sigma})$$

Consequently, this corresponds to a binary regression model with a probit link function with the logarithm of the age as an explanatory variable:

$$\eta_i = \beta_0 + \beta_1 \log t_i.$$

We then find that $\mu = -\beta_0/\beta_1$ and $\sigma = 1/\beta_1$. We fit this model to the data using the `glm` function:

```
juul.girl$log_age = log(juul.girl$age)
res_lognormal = glm(
  formula = menarche~log_age,
  family = binomial(link = "probit"),
  data = juul.girl)
summary(res_lognormal)
```

```
##
## Call:
## glm(formula = menarche ~ log_age, family = binomial(link = "probit"),
##     data = juul.girl)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.28617  -0.11683   0.00315   0.10715   2.54738
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -29.060      2.704  -10.75   <2e-16 ***
## log_age       11.287      1.049   10.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 719.39  on 518  degrees of freedom
## Residual deviance: 198.05  on 517  degrees of freedom
## AIC: 202.05
##
## Number of Fisher Scoring iterations: 8
```

It is known that the lognormal distribution has mean $\exp(\mu + \sigma^2/2)$ and variance $\exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)$. Using the invariance property of the MLE we get

$$\widehat{E[T]} = 13.1797, \quad \widehat{\mathrm{SD}(T)} = 1.1700.$$

```
sigma_lognormal = 1 / res_lognormal$coefficients[2]
mu_lognormal = - res_lognormal$coefficients[1] / res_lognormal$coefficients[2]
unname(c(exp(mu_lognormal + sigma_lognormal ^ 2 / 2),
         sqrt((exp(sigma_lognormal^2) - 1) * exp(2 * mu_lognormal + sigma_lognormal^2))))
```

```
## [1] 13.179704  1.170015
```

e)

We create a binary regression model with log age as explanatory variable and a cloglog link $p_i = 1 - \exp(-\exp(\beta_0 + \beta_1 \log t_i))$ This model gives a CDF for the latent age of:

$$P(T_i \leq t_i) = 1 - \exp(-\exp(\beta_0 + \beta_1 \log t_i))) = 1 - \exp(-t_i^{\beta_1} e^{\beta_0}),$$

which is equal to the Weibull distribution

$$F_T(t) = 1 - \exp(-\alpha t^{\beta}),$$

with $\alpha = e^{\beta_0}$ and $\beta = \beta_1$. It is known that the Weibull distribution has mean $\alpha^{-1/\beta}\Gamma(1 + 1/\beta)$ and variance $\alpha^{-2/\beta}(\Gamma(1 + 2/\beta) - \Gamma(1 + 1/\beta)^2)$, where $\Gamma(\cdot)$ is the gamma function. We fit the model to data and estimate the mean and standard deviance of T:

```
res_weibull = glm(
  formula = menarche~log_age,
  family = binomial(link = "cloglog"),
  data = juul.girl)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
alpha_weibull = exp(res_weibull$coefficients[1])
beta_weibull = res_weibull$coefficients[2]
unname(c(
  alpha_weibull^(-1/beta_weibull) * gamma(1 + 1 / beta_weibull),
  sqrt(alpha_weibull^(-2/beta_weibull) * (gamma(1 + 2 / beta_weibull) - gamma(1 + 1/beta_weibull)^2))))
```

```
## [1] 13.202556  1.188294
```

This gives

$$\widehat{E[T]} = 13.2026, \quad \widehat{\mathrm{SD}(T)} = 1.1883.$$

f)

The probit model results in a Gaussian latent model for T. The Gaussian distribution has nice theoretical properties and is quite popular within statistical modelling. This might therefore be preferable for many practitioners. However, using this model requires us to numerically compute the CDF of the Gaussian distribution. For really large amounts of data this means that inference can be much slower than for the other models. The logit link is the canonical link function for the Bernoulli distribution. This is often preferable. Using the logit link also allows for an easy interpretation of the regression coefficients, using odds. This can make the model more intuitive than some of the competing models. The log probit model enforces a positive age T, something the two previous models don't do. This is a nice property as ages must be positive in the

real world. However, when modelling the age of a girl that has her first period, the mean is estimated to be somewhere around 13, while the standard deviation is estimated to be a bit larger than 1. This means that the probit and logit models give astronomically small probabilities for a girl being less than zero years old when having her first period. Thus, in practice, enforcing $T > 0$ does not make much of a difference. The log probit model also requires computations of a Gaussian CDF, and it also has nice links to the Gaussian distribution, which is preferred by many practitioners. The Weibull distribution is popular within lifetime analysis for modelling the time until a specific event. Thus, practitioners coming from this field might prefer the theoretical properties of the Weibull when modelling the time to the first period of a girl.

In practice, all our models give very similar results. Thus, for such a small amount of data, and without more knowledge of the true distribution of T, there exists good arguments for choosing each of these models over the other, simply based on the preferences of the practitioner.