5.

We start by simulating data as described in the assignment. We simulate 10^5 data points from the multivariate normal distribution with mean $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ and variance $\begin{pmatrix} 1 & 0.25 & 0.5 \\ 0.25 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{pmatrix}$.

```
set.seed(7878)
n <- 10^5
sigma <- matrix(c(1, 0.25, 0.5, 0.25, 1, 0.5, 0.5, 0.5, 1), nrow = 3)
X <- mvrnorm(n, c(0,0, 0), sigma)
```

To illustrate that the conditional distribution of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} | X_3 = x_3$ is as expected, we condition on $X_3 = 0$. We approximate this (X_3 has a continuous distribution), by subsetting the rows of X for which $|X_3| < 0.05$:

```
cond <- which(abs(X[,3]) < 0.05)
```

We calculate the correlation of X_1 and X_2 for which $|X_3| < 0.05$.

```
cor(X[cond,1], X[cond,2]) |> knitr::kable(col.names = " ")
```

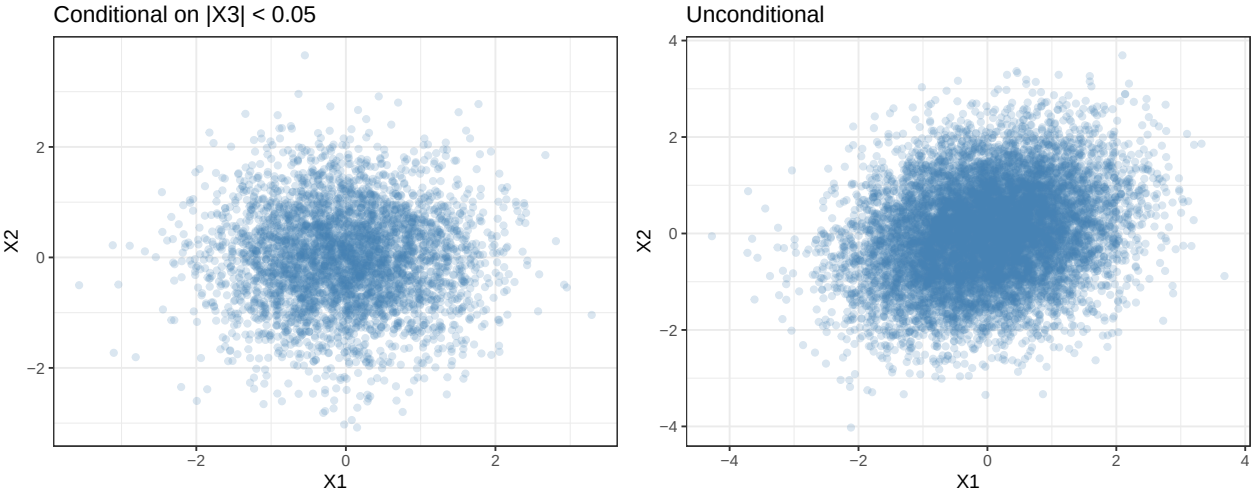
0.0013816

Which is close to 0 as expected. This is different from the correlation of X_1 and X_2 in general, which is ≈ 0.25 by construction:

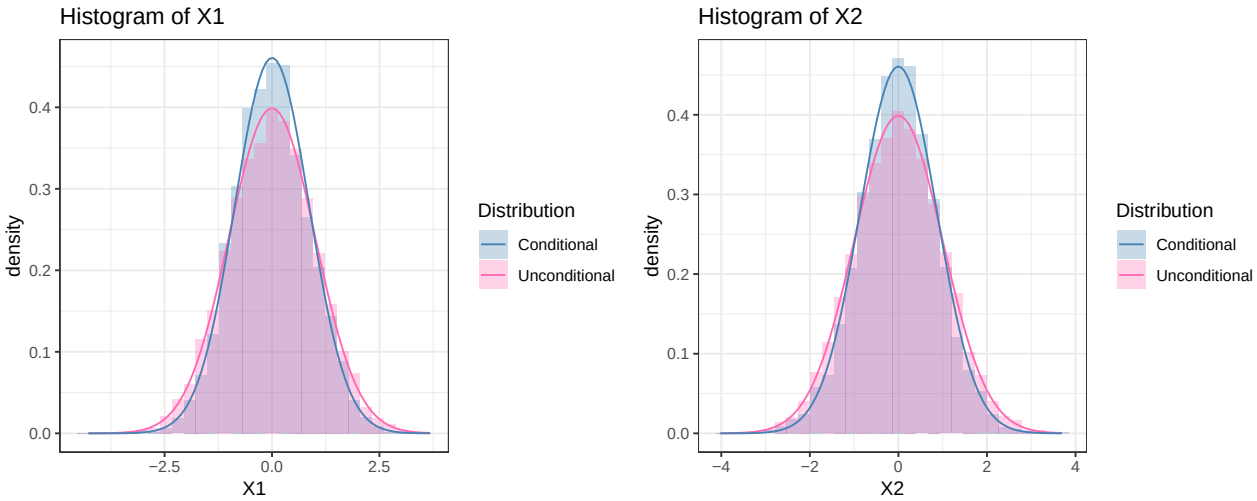
```
cor(X[,1], X[,2]) |> knitr::kable(col.names = " ")
```

0.2495407

To visualize the conditional distribution of X_1 and X_2 we create the two scatterplot below. In the unconditional distribution the two variables are positively correlated, while they in the conditional distribution are uncorrelated, and thus (since they are normal) independent.



As a last illustration, we have made histograms of the marginal distributions of the two random variables, conditional on $X_3 = 0$ and unconditionally. The marginal distribution of both variables unconditionally is $\mathcal{N}(0, 1)$ and conditionally on $X_3 = 0$, it is $\mathcal{N}(0, 0.75)$ per calculations from the previous exercises.



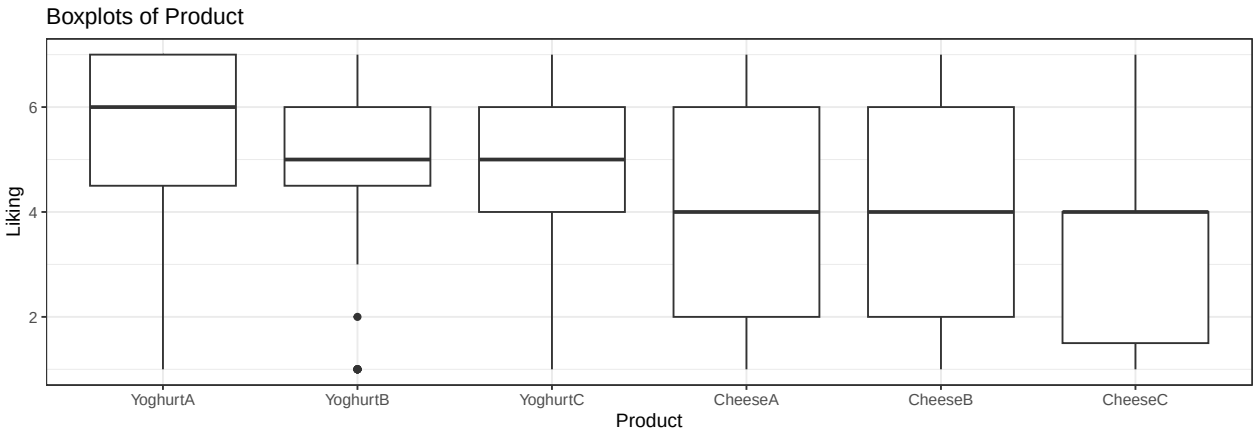
As expected, the variance reduces when we condition, since we have more information.

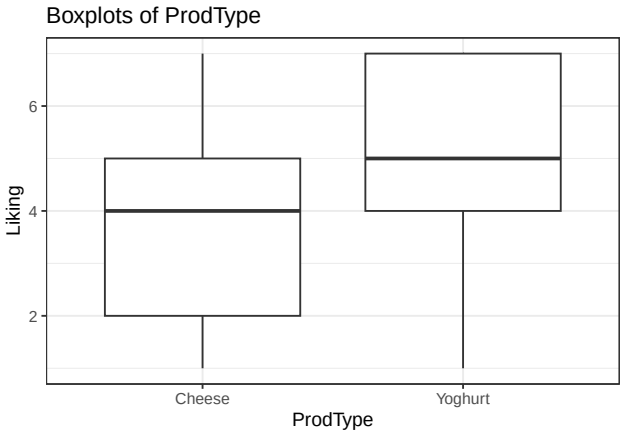
From example 2.5 we know that the variance matrix in the conditional distribution of Gaussian random variables does not depend on the value of the conditioning variable. Having illustrated that the two random variables $X_1|X_3 = 0$ and $X_2|X_3 = 0$ are independent and with the variance we expect, this will also be true for all other values of X_3.

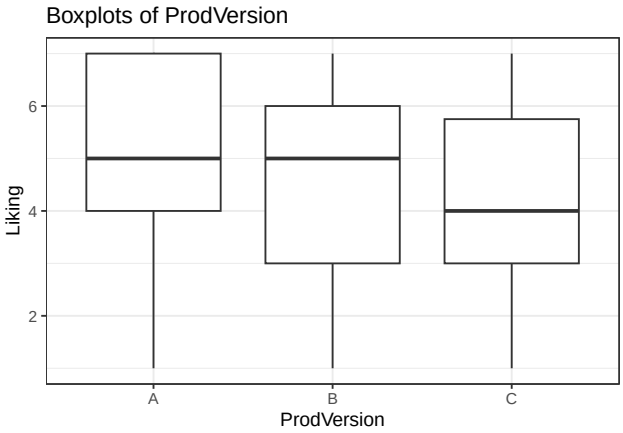
All in all the conditional distribution of $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} | X_3$ behaves as expected.

Part 2

We start out by visualizing the data. We first simply plot boxplots for each of the different products.

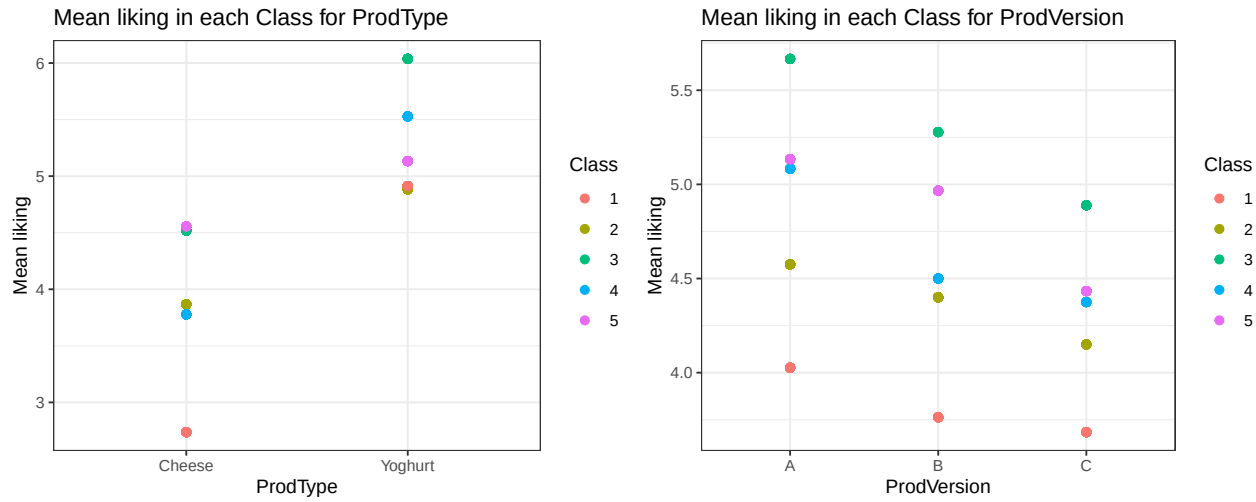






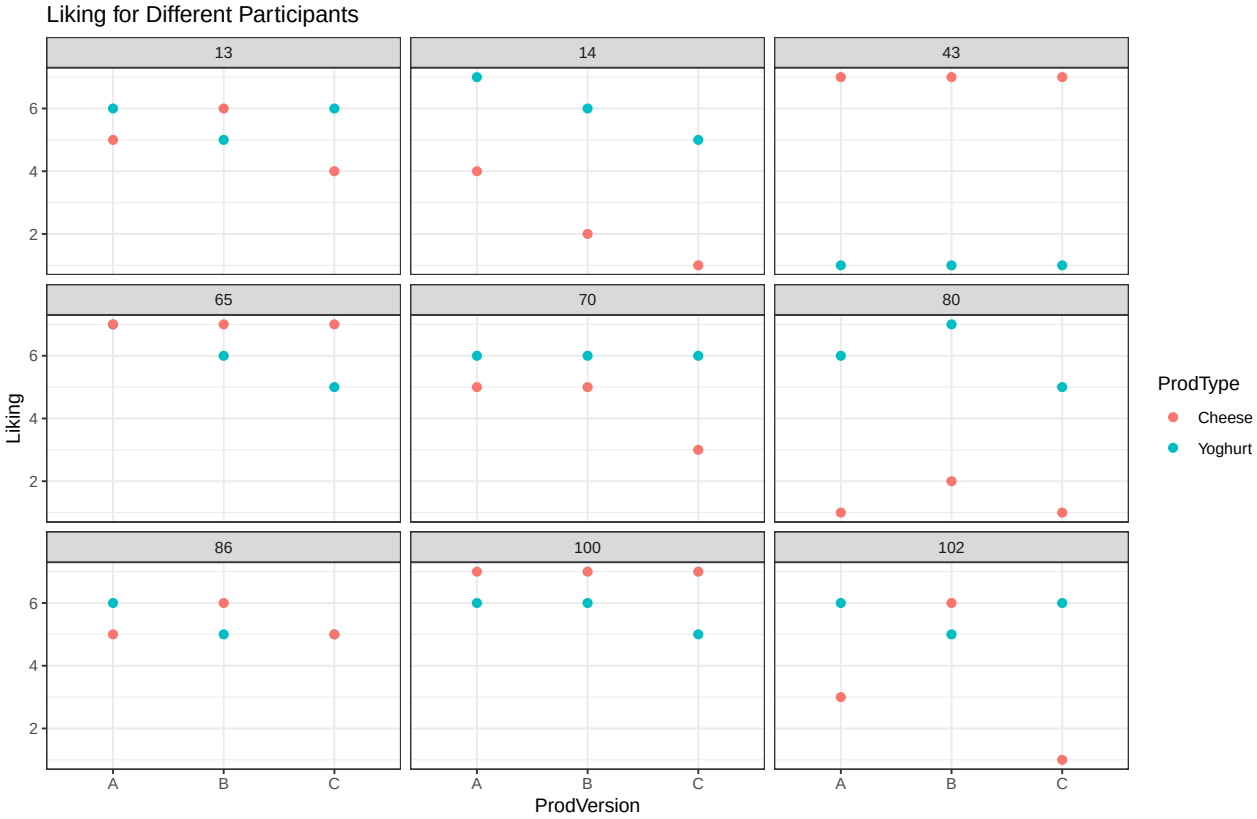
There is definetely a difference in the overall liking when looking at the product type. And it does seem that there might be a tendency for the liking to decrease as we move from products A to C.

To visualize if there could possibly be a difference in overall liking based on `Class` we calculate the average liking in each Class for both `ProdType` and `ProdVersion` and plot them.



Based on the `ProdVersion`-plot it definitely seems that there might be a difference in the overall liking of the products based on `Class`. There also seems to be some variations for `ProdVersion`. We are also interested in visualizing any potential effect of the different students on the liking of the products. So we draw 9 random participants and plot their `Liking` of the different product versions and colored according to the type of

product.



Here we can see that there indeed does seem to be variations between the overall preferences of the participants, both in terms of overall rating, but also how the liking is linked to the different product types.

1.

We proceed with a brief description of the stastical model fitted in `fit1`. Let Y_{ijk} define the response of participant j in class k testing product i. The first model is then defined by

$$Y_{ijk} = \beta_0 + B_j^P + B_k^C + \beta_i + \epsilon_{ijk}$$

where β_0 is the global intercept corresponding to `ProductYoghurtA` and β_i for $i \in \{$ `ProductYoghurtA`, `ProductYoghurtB`, `ProductYoghurtC`, `ProductCheeseA`, `ProductCheeseB`, `ProductCheeseC` $\}$ describes the difference between the global intercept and each other product. We have included `ProductYoghurtA` in the list for proper indexation, but in fact there is no $\beta_{ProductYoghurtA}$ in the model as this is β_0. The $B_j^P \overset{i.i.d}{\sim} \mathcal{N}(0, \tau_P^2 I)$ is the random parameter of participant $j = 1, ..., 75$ and $B_k^C \overset{i.i.d}{\sim} \mathcal{N}(0, \tau_C^2 I)$ is the random parameter of class $k = 1, ..., 5$. Finally, $\epsilon_{ijk} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2)$ is the residual for participant j in class k, testing product i. We assume independence between all B_j^P, B_k^C and ϵ_{ijk}.

We model participant and class as random effects since they represent random samples from a larger population, so we are only concerned with the variance they add to the model estimates. In contrast, we model `Product` as a fixed effect to estimate how the different products influence the overall liking.

2.

The formula for the correlation for two random variables X and Y is $\frac{Cov(X,Y)}{\sqrt{(VX \cdot VY)}}$. By the independence of the random variables in the statistical model for `fit1` we get that $VY_{ijk} = \tau_P^2 + \tau_C^2 + \sigma^2$ for all i, j and k. We now consider the two different cases:

If we look at the correlation between two observations for the same participants Y_{ijk} and Y_{ljk}, we have independence between all random variables in the statistical model, except for B_j^P and B_k^C, which are not independent of themselves. We can use this alongside the bilinear properties of the covariance to see:

$$Cov(Y_{ijk}, Y_{ljk}) = Cov(B_j^P, B_j^P) + Cov(B_k^C, B_k^C) = V(B_j^P) + V(B_k^C) = \tau_P^2 + \tau_C^2$$

Thus we get that

$$cor(Y_{ijk}, Y_{ljk}) = \frac{\tau_P^2 + \tau_C^2}{\tau_P^2 + \tau_C^2 + \sigma^2}$$

for observations from the same participant. Plugging in the estimates from `fit1` and `fit2` we get:

	$\text{Cor}(Y_{ijk}, Y_{ljk})$
fit1	0.2781642
fit2	0.2791743

If we now look at the correlation between observations from the same class, but not the same participant, we have independence between all random variables in the model except for B_k^C. So by a similar argument to that above we get:

$$cor(Y_{ijk}, Y_{lpk}) = \frac{\tau_C^2}{\tau_P^2 + \tau_C^2 + \sigma^2}$$

Plugging in the estimates of the standard deviations from `fit1` and `fit2` we get:

	$\text{Cor}(Y_{ijk}, Y_{lpk})$
fit1	0.0594315
fit2	0.0595567

3.

The factor `Product` is the interaction of the two factors `ProdVersion` and `ProdType`, therefore the subspace spanned by `ProdVersion` and `ProdType` is included in the subspace spanned by `Product`, and `fit2` is thus a submodel of `fit1`. In other words if we know the value of `Product` we also know the value of `ProdVersion` and of `ProdType`.

In `fit1` we estimate 6 different fixed effects parameters, 1 intercept parameter, and then 5 additional parameters for each additional interaction level between `ProdVersion` and `ProdType`. In `fit2` we estimate 4 fixed effect parameters, 1 intercept parameter, 2 for each additional `ProdVersion` level and 1 for the last level of `ProdType`. We assume no interaction effects between the two factors `ProdVersion` and `ProdType` in `fit2`.

Letting L_0 denote the subspace of $\mathbb{R}^n$ spanned by the model matrix from `fit2` and L_X denote the subspace of $\mathbb{R}^n$ spanned by the model matrix from `fit1`, we can test the hypothesis of $EY \in L_0 \subseteq L_X$ with the likelihood ratio statistic. The test relies on asymptotic results, which we use without further arguments. We perform the test by use of the anova command:

```
anova(fit1,fit2)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: likingdata
## Models:
## fit2: Liking ~ ProdVersion + ProdType + (1 | Participant) + (1 | Class)
## fit1: Liking ~ Product + (1 | Participant) + (1 | Class)
##      npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## fit2    7 1777.8 1806.6 -881.91   1763.8
## fit1    9 1781.1 1818.1 -881.56   1763.1 0.7031  2     0.7036
```

The test returns a p-value of 0.7, so with a significance level of 0.05 we cannot reject the null hypothesis, and we can conclude that the interaction factor `product` does not improve the model fit significantly.

4.

To test whether the children like all three versions of the product equally well, we check that none of the fixed effects are significantly different from each other. That is, we test the two hypotheses:

$$H_{01} : \beta_{ProdVersion} = 0, \qquad H_{02} : \beta_{ProdType} = 0$$

. Both hypotheses are tested using the `drop1()` function with `test = "Chisq"`:

```
drop1(fit2, test = "Chisq")
```

```
## Single term deletions
##
## Model:
## Liking ~ ProdVersion + ProdType + (1 | Participant) + (1 | Class)
##             npar    AIC    LRT Pr(Chi)
## <none>           1777.8
## ProdVersion    2 1782.9  9.051 0.01083 *
## ProdType       1 1856.5 80.707 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We get a p-value of `0.011` for `ProdVersion` and a p-value of `< 2e-16` for `ProdType`, implying that on a 5% significance level, we reject both null hypotheses. This means, there is evidence that the children have a preference for both the version of the product and for the type of the product. We should keep in mind, that the chisquare test is an asymptotic test, and we do not know if the asymptotics have set in, so we should be careful drawing conclusions, especially for `ProdVersion` as this is only borderline significant. However, based on the tests we continue to examine the preferences.

It is evident from the model coefficients that the children like yoghurt better than cheese. To check which product version the children like better, we determine the 95%-profile likelihood confidence interval for the model parameters:

```
confint(fit2, oldNames = FALSE)
```

```
## Computing profile confidence intervals ...
```

```
##                                 2.5 %      97.5 %
## sd_(Intercept)|Participant  0.6595997  1.11693070
## sd_(Intercept)|Class        0.0000000  1.00530558
## sigma                       1.4591641  1.68384627
## (Intercept)                 3.5917611  4.70819767
## ProdVersionB               -0.6417187  0.06838533
## ProdVersionC               -0.9017187 -0.19161467
## ProdTypeYoghurt             1.1101013  1.68989874
```

The children seem to like `ProdVersionA` better than `ProdVersionC` (CI for `ProdVersionC` does not include 0), but there is not strong evidence that they like `ProdVersionA` over `ProdVersionB` (CI for `ProdVersionB`

includes 0). To investigate if the children like `ProdVersionB` better than `ProdVersionC`, we refit the model with `ProdVersionB` as the reference level and determine the 95%-profile likelihood confidence intervals in this parametrization:

```
## Computing profile confidence intervals ...
```

```
##                                       2.5 %    97.5 %
## sd_(Intercept)|Participant      0.65959969 1.1169307
## sd_(Intercept)|Class            0.00000000 1.0053055
## sigma                           1.45916409 1.6838463
## (Intercept)                     3.30509444 4.4215310
## relevel(ProdVersion, ref = 2)A -0.06838533 0.6417187
## relevel(ProdVersion, ref = 2)C -0.61505200 0.0950520
## ProdTypeYoghurt                 1.11010126 1.6898987
```

From the confidence intervals there is not enough evidence that the children like `ProductVersionB` better than `ProductVersionC`. To sum up: On a 95% significance level the children like `ProductVersionA` better than `ProductVersionC`, but there is not enough evidence that the children like `ProductVersionA` better than `ProductVersionB` or `ProductVersionB` better than `ProductVersionC`, according to this model.

5.

We use the code from the hint to run 2000 simulations from `fit2`. We first extract the model matrices X and Z and the parameter estimates for $\beta, \tau_P^2, \tau_C^2$ and σ^2. We then create an empty 2000×3 matrix to store the simulations of $\hat{\delta}$ and the corresponding Wald confidence bands. Then we fill out the entries of the matrix with a for loop. So in each $i = 1, ..., 2000$ we simulate 75, 5 and 450 values from B^P, B^C and ϵ respectively, and use those to create a new set of y-values $y = X\beta + Z(B^P + B^C) + \epsilon$. We then fit a new model specified as `fit2` and calculate corresponding confidence intervals using the Wald-method.

```
library(MASS)
M <- 2000

#Extracting parameters and modelmatrices
X <- fit2 %>% model.matrix()
Z <- fit2 %>% getME(name="Z")
beta <- fixef(fit2)
tauP <- data.frame(VarCorr(fit2))$sdcor[1]
tauC <- data.frame(VarCorr(fit2))$sdcor[2]
sigma <- data.frame(VarCorr(fit2))$sdcor[3]

#Creating matrix
deltasim <- matrix(NA, M, 3)
#Iteratively simulating delta and confidence intervals
for (i in 1:M)
{
B1 <- mvrnorm(75, 0, tauP)
B2 <- mvrnorm(5, 0, tauC)
eps <- mvrnorm(450, 0, sigma)
B <- c(B1,B2)
y <- X %*% beta + Z %*% B + eps
y <- y %>% as.numeric() # NB. This seems to be necessary
lmm2 <- lmer(y ~ ProdVersion + ProdType + (1|Participant) + (1|Class), data=likingdata)
deltasim[i,1] <- fixef(lmm2)[4]
deltasim[i,2:3] <- (lmm2 %>% confint(method="Wald"))[7,]
}
```

```r
#Changing format to a dataframe and changing names
deltasim <- deltasim %>% data.frame()
names(deltasim) <- c("est","lower","upper")
```

Running the simulations gives a few warnings of type `boundary (singular) fit: see help('isSingular')`. This happens when the variance of a random effect is estimated to be 0. Although this warning should be taken into account, it is not an immediate problem, as it is simply a consequence of the variation in the data. Keeping track of the number of warnings, we find that it happens between 50 and 70 times during the 2000 simulations. We argue that this is a small enough proportion, to not affect the validity of the simulation.

To see if $\hat{\delta}$ is approximately an unbiased estimator of δ, we check whether the mean of $\hat{\delta}$ is approximately equal to δ.

```r
delta <- fixef(fit2)[4]
unname(mean(deltasim$est) - delta) %>% knitr::kable(col.names = " ")
```

-0.0015407

This is close to 0, so $\hat{\delta}$ seems unbiased.

To check if we obtain the desired coverage, we simply calculate what percentage of the simulated intervals cover the true δ-vaule.

```r
mean(deltasim$lower <= fixef(fit2)[4] & fixef(fit2)[4] <= deltasim$upper) |>
  knitr::kable(col.names = " ")
```

0.948

As we can see, we obtain approximately the desired coverage.

6.

Simulating from the t-distribution

The mean of the t-distribution is already 0. The variance of the t-distribution with ν degrees of freedom is $\frac{\nu}{\nu-2} = \frac{3}{3-2} = 3$. In order to achieve a variance of σ^2 we would therefore need to scale $X \sim t(3)$ with

$$\sigma^2 = V(c \cdot X) = c^2 3 \Leftrightarrow c = \frac{\sigma}{\sqrt{3}}$$

We define the scaling factors

```r
c_par <- tauP / sqrt(3)
c_class <- tauC / sqrt(3)
c_eps <- sigma / sqrt(3)
```

We modify the simulation from question 5 to draw from the t-distribution. The variables drawn are scaled by the scaling factors defined above.

```r
set.seed(174)
M <- 2000
n_eps <- nrow(likingdata)
n_par <- unique(likingdata$Participant) |> length()
n_class <- unique(likingdata$Class) |> length()

X <- fit2 |> model.matrix()
```

```r
Z <- getME(fit2, "Z")
beta <- (summary(fit2) |> coef())[,1]
tauP <- data.frame(VarCorr(fit2))[1,5]
tauC <- data.frame(VarCorr(fit2))[2,5]
sigma <- data.frame(VarCorr(fit2))[3,5]
deltasim2 <- matrix(NA,M,3)

for (i in 1:M){
  B1 <- rt(n = n_par, df = 3) * c_par
  B2 <- rt(n = n_class, df = 3) * c_class
  eps <- rt(n = n_eps, df = 3) * c_eps
  B <- c(B1,B2)
  y <- X %*% beta + Z %*% B + eps
  y <- y |> as.numeric() # NB. This seems to be necessary
  lmm2 <- lmer(y ~ ProdVersion + ProdType + (1|Participant) + (1|Class), data=likingdata)
  deltasim2[i,1] <- fixef(lmm2)[4]
  deltasim2[i,2:3] <- (lmm2 |> confint(method="Wald"))[7,]
}

deltasim2 <- deltasim2 |> data.frame()
names(deltasim2) <- c("est","lower","upper")
```

We calculate the bias:

```r
mean(deltasim2$est - fixef(fit2)[4]) |> knitr::kable(col.names = " ")
```

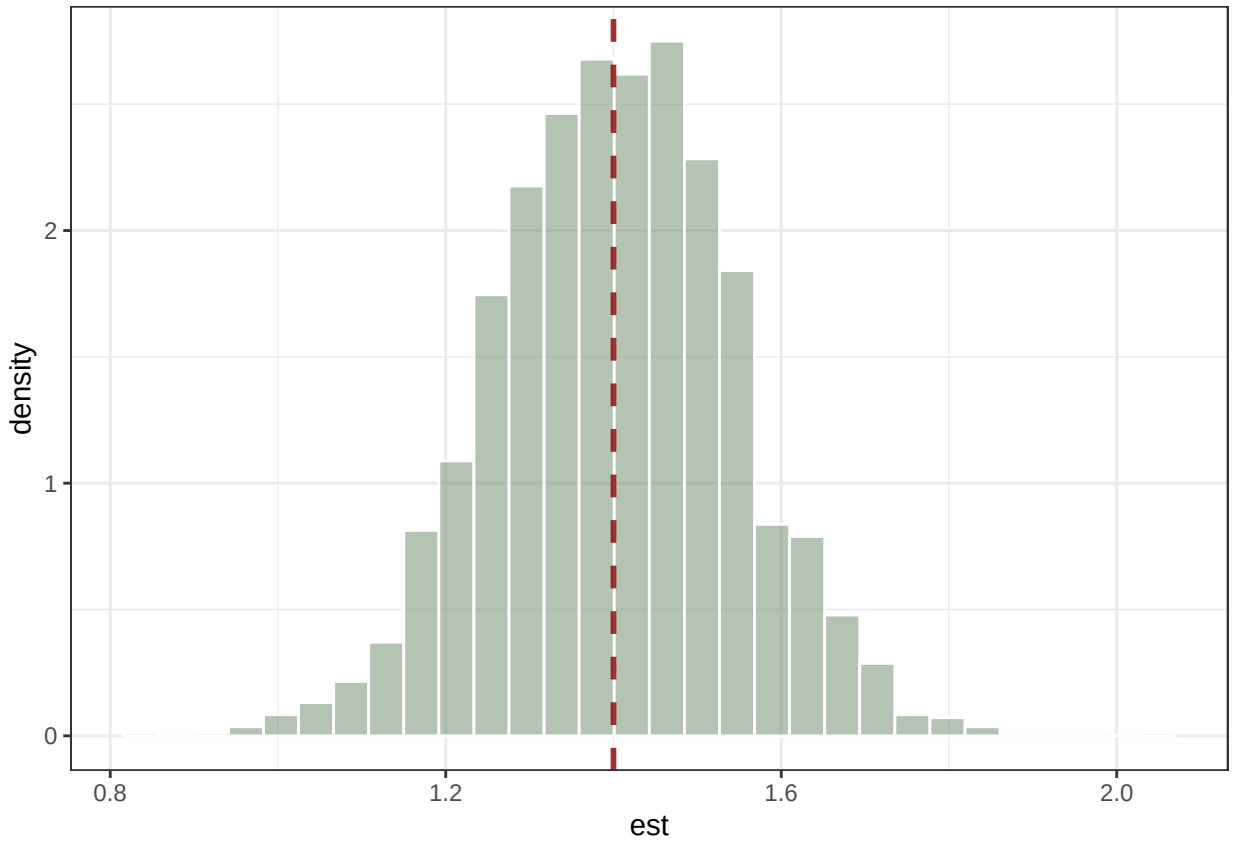
0.0016939

And the coverage:

```r
mean(deltasim2$lower <= fixef(fit2)[4] & fixef(fit2)[4] <= deltasim2$upper) %>% knitr::kable(col.names =
```

0.958

And make a histogram of the estimates:



The estimates are still practically unbiased, the confidence intervals achieve quite accurate coverage, and the distribution looks relatively normal.

Simulating from the exponential distribution

The mean of an exponentially distributed random variable X with rate $\lambda = 1$, is $E(X) = 1$. And the variance is $\frac{1}{\lambda^2} = 1$. In order to achieve a mean of 0 and a variance of σ^2 we would therefore need to shift and scale $X \sim \exp(1)$ with

$$\sigma^2 = V(c \cdot X - k) = c^2 \Leftrightarrow c = \sigma$$

and

$$0 = E(\sigma X - k) = \sigma - k \Leftrightarrow k = \sigma$$

We define the shift and scaling constants

```
c_par <- tauP
c_class <- tauC
c_eps <- sigma
```

```
set.seed(765)
deltasim3 <- matrix(NA,M,3)

for (i in 1:M){
  B1 <- rexp(n = n_par, rate = 1) * c_par - c_par
  B2 <- rexp(n = n_class, rate = 1) * c_class -  c_class
  eps <- rexp(n = n_eps, rate = 1) * c_eps - c_eps
```

```
  B <- c(B1,B2)
  y <- X %*% beta + Z %*% B + eps
  y <- y |> as.numeric() # NB. This seems to be necessary
  lmm2 <- lmer(y ~ ProdVersion + ProdType + (1|Participant) + (1|Class), data=likingdata)
  deltasim3[i,1] <- fixef(lmm2)[4]
  deltasim3[i,2:3] <- (lmm2 |> confint(method="Wald"))[7,]
}

deltasim3 <- deltasim3 |> data.frame()
names(deltasim3) <- c("est","lower","upper")
```

We calculate the bias:

```
mean(deltasim3$est - fixef(fit2)[4]) |> knitr::kable(col.names = " ")
```

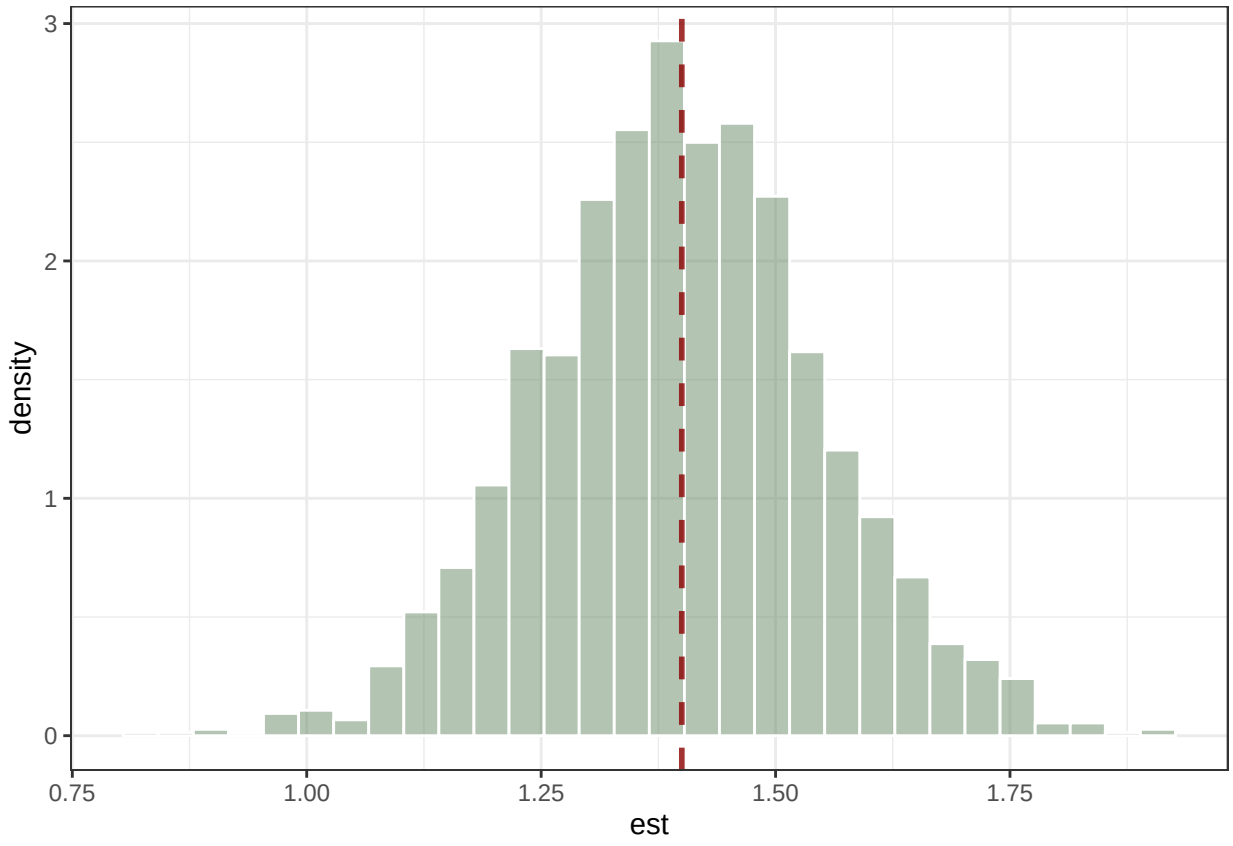
-0.0019755

And the coverage:

```
mean(deltasim3$lower <= fixef(fit2)[4] & fixef(fit2)[4] <= deltasim3$upper) |>
  knitr::kable(col.names = " ")
```

0.946

And make a histogram of the estimates:

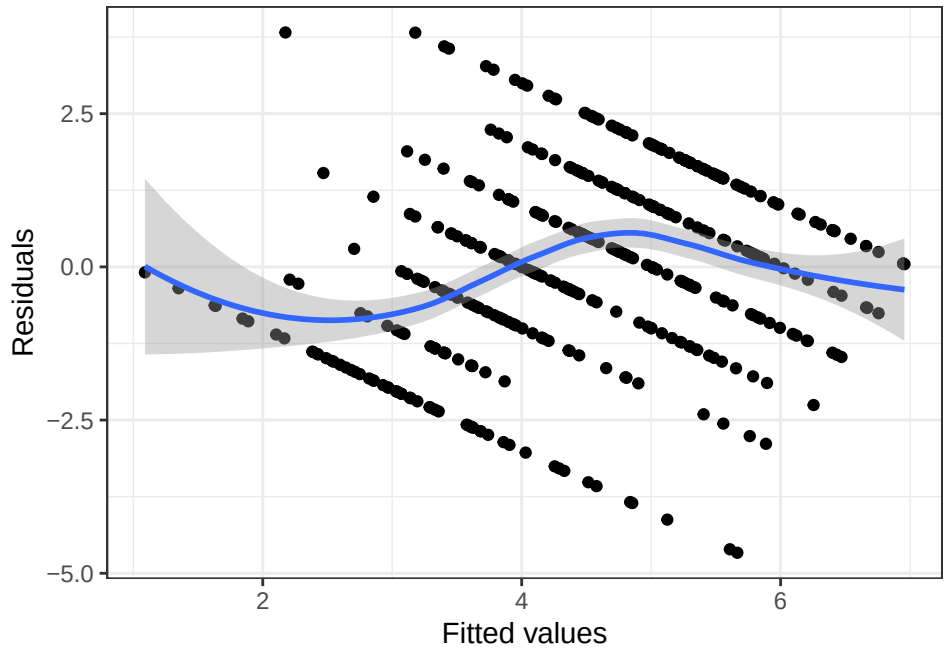


The estimates still appear unbiased, the confidence intervals achieve accurate coverage and the distribution again looks normal.

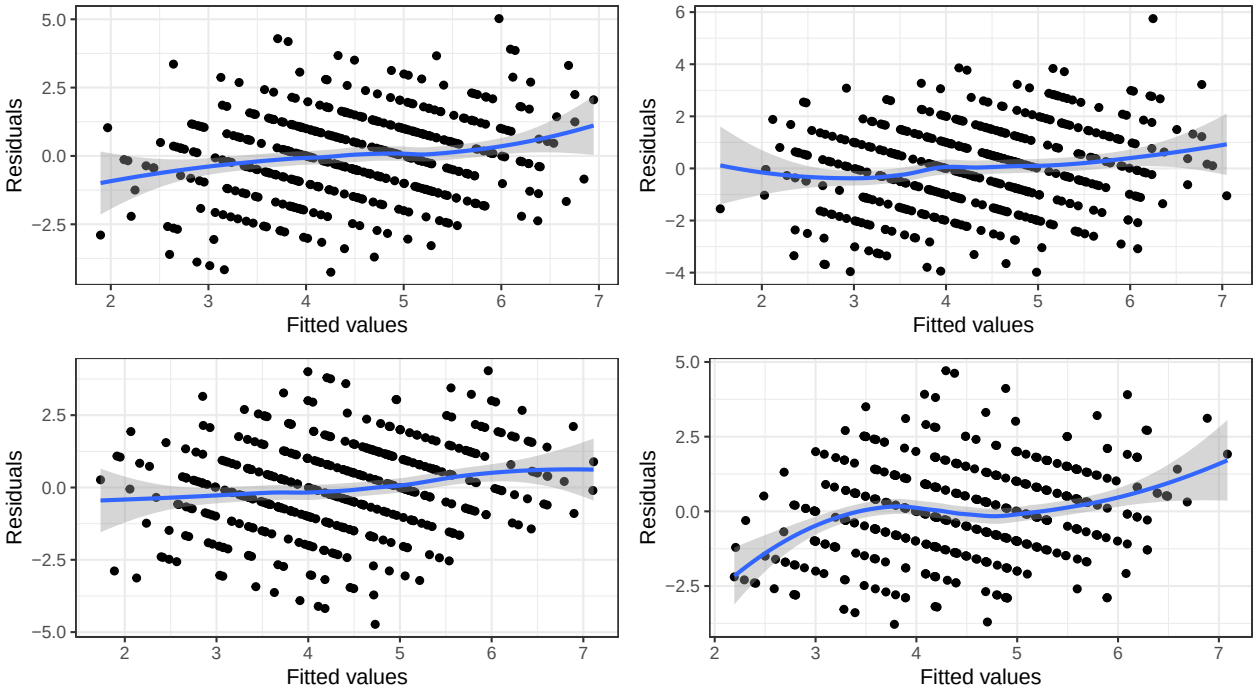
We can conclude that the model estimates and confidence intervals are robust to various types of distribution of the random effects and noise, as long as the mean and variance structure is correctly specified.

7.

We proceed to carry out model validation for `fit2`. We follow the procedure given in chapter 5 of the lecture notes on mixed models and start with a residual plot:

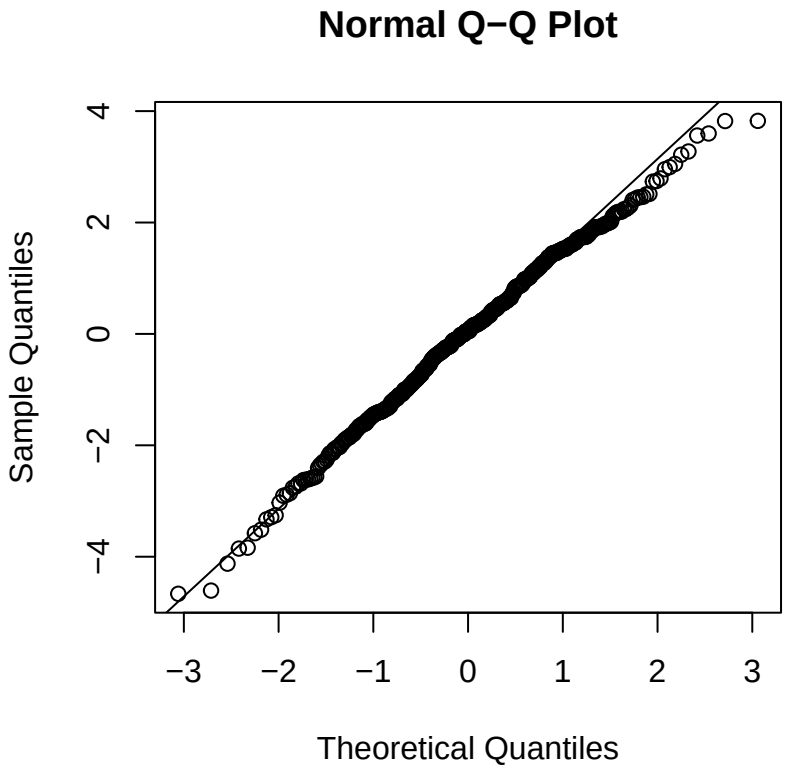


We may be tempted to conclude that the residuals display an alarming pattern. We should however keep in mind, that the response is discrete while the fitted values are not. Hence, the lines in the plot, are in fact to be expected. In particular, for each response value, $a = 1, ..., 7$, we have a linear relationship between the fitted values and the residuals $Resdiual = a - Fitted$. The blue line (fitted with `geom_smooth`) suggests, that the residuals have mean around zero and no concerning trend. We run a few simulations to check if we can replicate the residual plot:



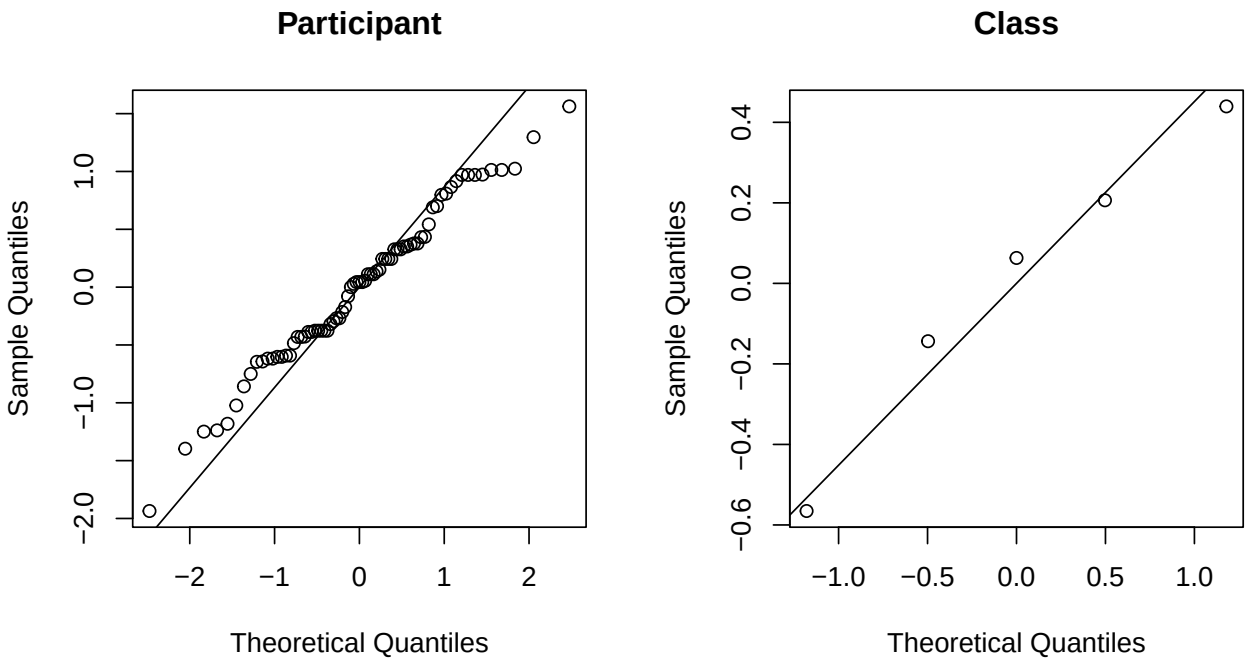
Note, we have discretized the response in the simulations to get a comparable pattern. We have no other restrictions on the predictions, so the model simulates responses outside the range 1 to 7. This gives more lines than in the original residual plot, but other than this, the residual plots for the simulations resemble the residual plot for the original model. Hence, it seems reasonable to conclude that the model captures the mean structure of the data.

We proceed to check normality of the residuals with a qq-plot:



The qq-plot shows that the residuals to a large extent are normal. There are some issues in the upper tail suggesting that the distribution of the residuals may be slightly left-skewed, so they are not perfectly normal. This is not a major concern though, as the assumption of normality is secondary for the validity of the model, as we also saw in the simulations in the previous question.

We finally inspect the predicted random effects. As we have 75 random effects for participants, it makes sense to make a qq-plot for this random effect. With only 5 classes we may argue that a qq-plot for this random effect is less informative, but for completeness we plot this as well.



The qq-plot to the left suggests that the random effects for participants are not perfectly normally distributed. Again this is not a major concern, as the assumption of normality is secondary for the validity of the model, and as we saw in the simulations in the previous question, the model is robust to deviations from normality. It is difficult to draw conclusions from the qq-plot to the right due to the few classes, but there are no immediate concerns. To sum up, we find the model to be a decent fit to the data.

8.

In order to examine how discretization of the response affects the LMM-based estimator for δ, we simulate from the `fit2` model, discretize the simulations, fit a model on the simulated discretized data and extract the estimates and the CI boundaries. The discretization can be done in various ways. One approach is to use the `round` function, which is done below. We call this Method 1.

```
deltasim4 <- matrix(NA,M,3)

for (i in 1:M){
  y <- simulate(fit2)$sim_1 |> round()
  lmm2 <- lmer(y ~ ProdVersion + ProdType + (1|Participant) + (1|Class), data=likingdata)
  deltasim4[i,1] <- fixef(lmm2)[4]
  deltasim4[i,2:3] <- (lmm2 |> confint(method="Wald"))[7,]
}

deltasim4 <- deltasim4 |> data.frame()
names(deltasim4) <- c("est","lower","upper")
```

Since the model simulates data outside the range of $(1, 7)$, the discretized data also has integer values below 1 and above 7. A way of discretizing data and ensuring that the discretized values fall within the range of the predictor variable, is to scale and shift the data before rounding, we do this in the following. We call this Method 2.

```
deltasim5 <- matrix(NA,M,3)

for (i in 1:M){
  y <- simulate(fit2)$sim_1
  y <- (y - min(y))/(max(y) - min(y)) * (7 - 1) + 1 # shift and scaling
  y <- y |> round()
  lmm2 <- lmer(y ~ ProdVersion + ProdType + (1|Participant) + (1|Class), data=likingdata)
  deltasim5[i,1] <- fixef(lmm2)[4]
  deltasim5[i,2:3] <- (lmm2 |> confint(method="Wald"))[7,]
}

deltasim5 <- deltasim5 |> data.frame()
names(deltasim5) <- c("est","lower","upper")
```

We finally try a method, where we set all values below 1 to 1 and all values above 7 to 7. We call this Method 3.

```
deltasim6 <- matrix(NA,M,3)

for (i in 1:M){
  y <- simulate(fit2)$sim_1
  y <- ifelse(y < 1, 1, y)
  y <- ifelse(y > 7, 7, y)
  y <- y |> round()
  lmm2 <- lmer(y ~ ProdVersion + ProdType + (1|Participant) + (1|Class), data=likingdata)
  deltasim6[i,1] <- fixef(lmm2)[4]
  deltasim6[i,2:3] <- (lmm2 |> confint(method="Wald"))[7,]
}

deltasim6 <- deltasim6 |> data.frame()
names(deltasim6) <- c("est","lower","upper")
```

We calculate the bias for all three approaches:

	Bias
Method 1	-0.0025222
Method 2	-0.6840733
Method 3	-0.1776933

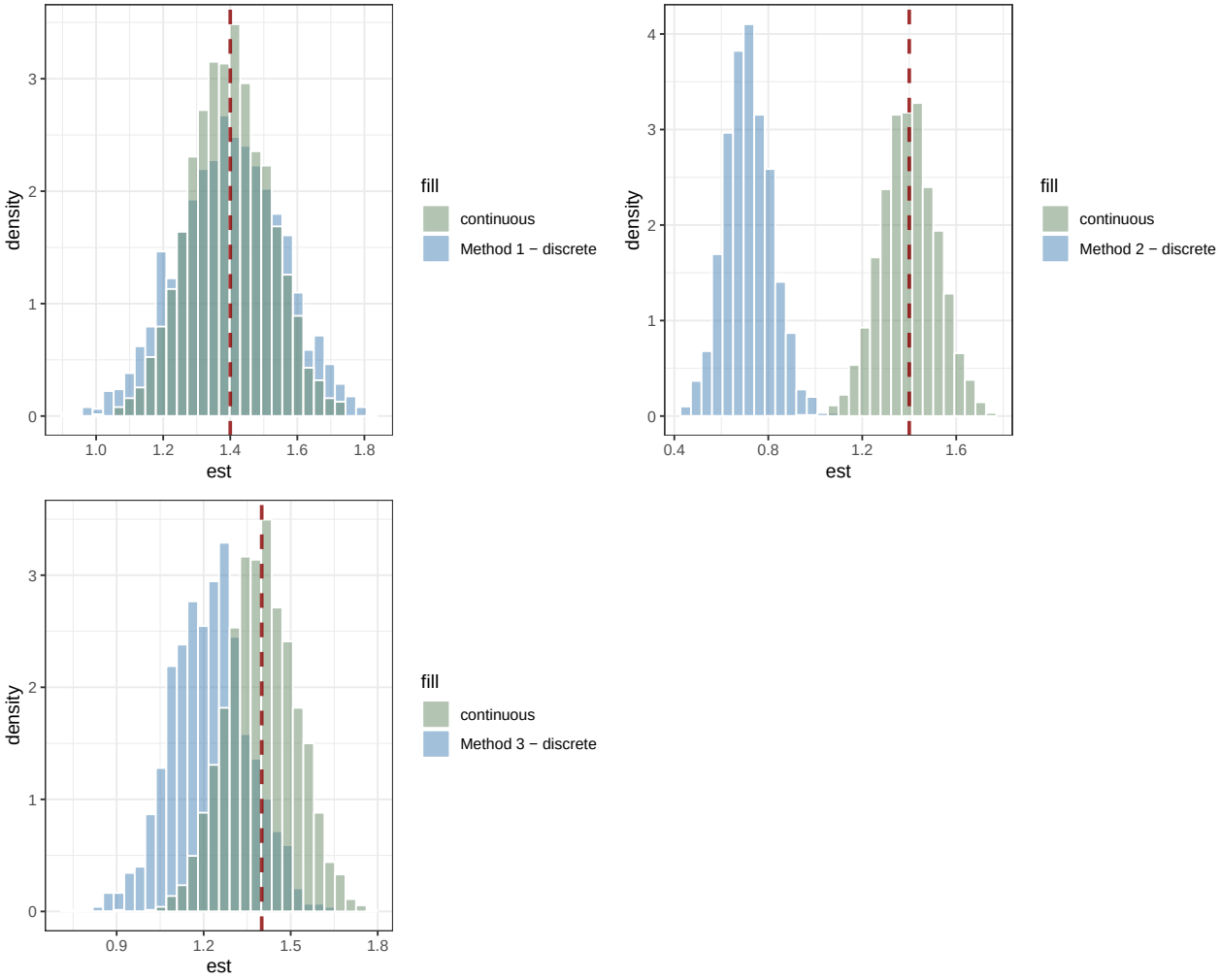
And coverage:

```
tibble(" " = c("Method 1", "Method 2", "Method 3"),
       "Coverage" = c(mean(deltasim4$lower <= fixef(fit2)[4] & fixef(fit2)[4] <= deltasim4$upper),
                      mean(deltasim5$lower <= fixef(fit2)[4] & fixef(fit2)[4] <= deltasim5$upper),
                      mean(deltasim6$lower <= fixef(fit2)[4] & fixef(fit2)[4] <= deltasim6$upper))) |>
  knitr::kable()
```

	Coverage
Method 1	0.947
Method 2	0.000
Method 3	0.733

We note, that with method 1, we obtain the desired coverage and it seems that $\hat{\delta}$ is still an unbiased estimator. This is not the case with method 2 and 3. Method 2 seems to be an especially poor choice, as the coverage is 0 and the bias is large.

To further investigate how the estimators behave with the different types of discretization, we plot histograms of the estimate of δ for the three methods. We also include the estimate of δ from the continuous data.



The histograms confirm the picture we got from the bias and coverage calculations. The first method resembles the continuous case fairly well with a slight increase in variance. The other methods appear to be poor choices.