Statistical Computing Coursework A

Zhao Yifei

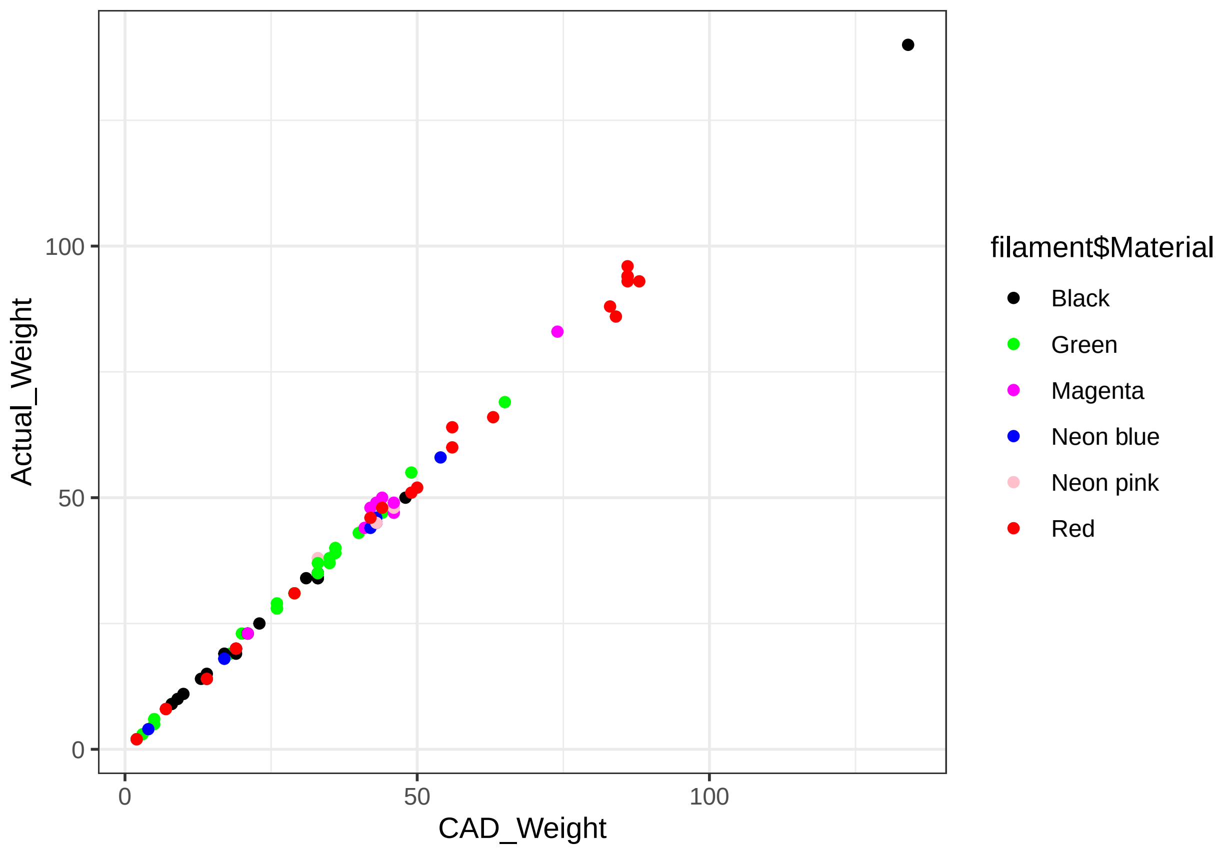
24/02/2020

```
library(xtable)
Sys.setenv(LANGUAGE = "en")
source("CWA2020code.R")
suppressPackageStartupMessages(library(tidyverse))
theme_set(theme_bw())
filament <- read.csv("filament.csv", stringsAsFactors = FALSE)
```

Task 1

We use ggplot() to plot the Actual_Weight with respect to CAD_Weight, and mark diffrent material with different color label. The scale_color_manual() is similiar to that in lab3code, and we use color'pink','blue' to substitute 'neon pink','neon blue' since they do not exist in R.

```
#use ggplot() for the filament
#use geom_point() and scale_color_manual() to add points with manual color
ggplot(filament) +
  geom_point(aes(x = CAD_Weight, y = Actual_Weight, col = filament$Material)) +
  scale_color_manual(values = c("Neon pink" = "pink",
                                "Green" = "green",
                                "Black" = "black",
                                "Magenta" = "magenta",
                                "Red" = "red",
                                "Neon blue" = "blue"))
```



Task2

For this task, we firstly consider spliting the original set, filament, into data_obs for estimation and data_test for test using subset().
Comparing model_Z() with that in lab3, we add parameter formula to make this function more flexible so that it will estimates different model with respect to different input formula.
Since we want to estimate theta and Sigma_theta here, we use optim() to get the optimisation with input neg_log_lik function and method is 'BFGS'.
In lab3, we just want to estimate a four-parameter model, and in this coursework, the model_Z is flexible so we use rep(0,ncol(Z$ZE)) to determine the parameter in optim() since matrices ZE and ZV mentioned in CWA2020code have the same size and expectation of the model is ZE*theta.
The input of model_estimate() are three parameters: formulas, data and string variable, response. In latter tasks, we use data_obs to estimate the model.
The output of model_estimate() is a list of three elements: theta, Sigma_theta and formulas, the first two results we want are the parameters estimated by the funtion, and the formulas is the same as input.
Above all, our flexible model is $Y \sim N(Z_E\theta, exp(Z_V\theta))$.
Here Y is the actual value, the expectation has a linear model and the variance has a log-linear model, where the same parameter could influence both the expectation and the variance.

```
#split filament into estimation and test sets
data_obs <- subset(filament,filament$Class=="obs")
data_test <- subset(filament,filament$Class=="test")

#define model_estimate() with three input parameters
model_estimate <- function(formulas,data,response){
```

```
  Z <- model_Z(formulas,data)
  opt <- optim(rep(0,ncol(Z$ZE)),fn = neg_log_lik,Z = Z, y = data[[response]],
               method = "BFGS",control = list(maxit = 5000),hessian = TRUE)
  list(theta = opt$par,Sigma_theta = solve(opt$hessian),formulas = formulas)
}
```

Task3

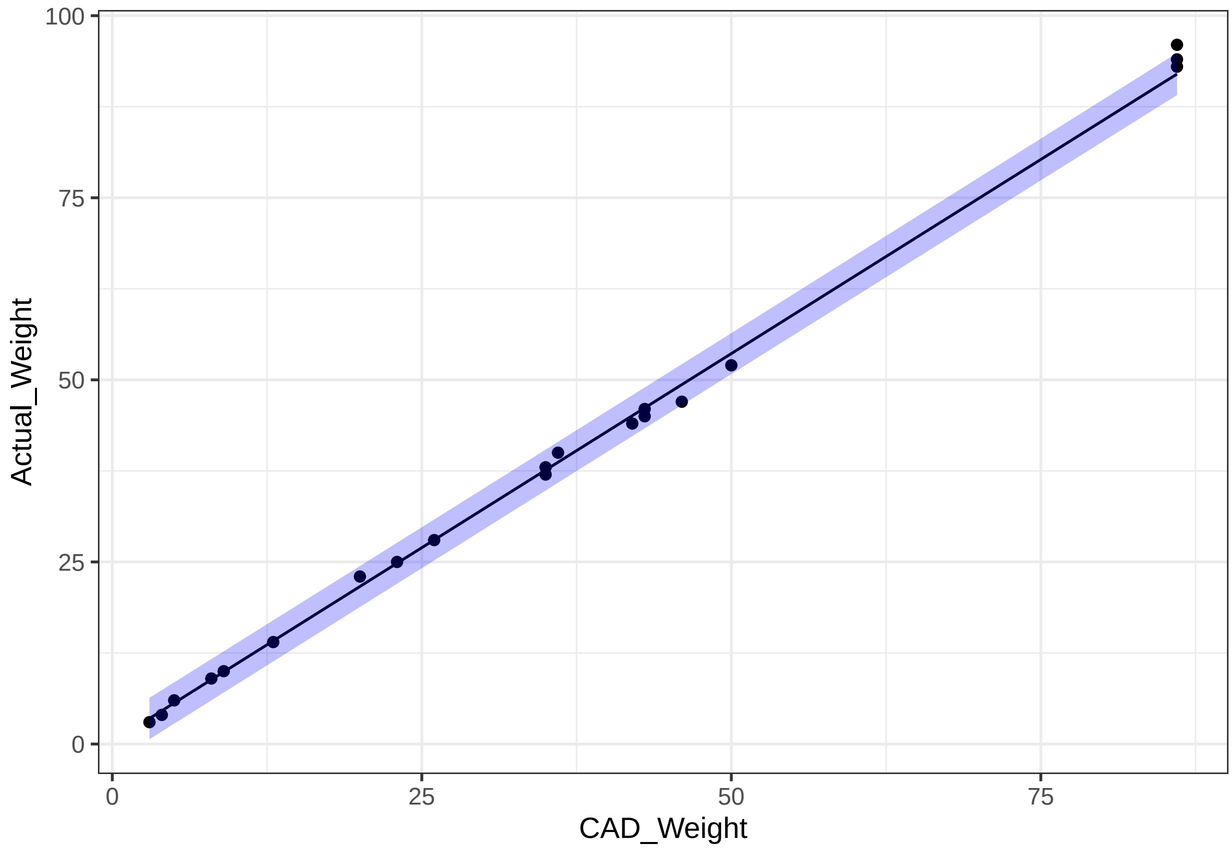
The target is to estimate the model for Actual_Weight with an intercept and covariate CAD_Weight for the model expectations, but only an intercept for the model log-variances.
As mentioned in CWA2020code, E and V are formulas for expectation part and log-variance part, for the linear model with respect to expectation and log-variance, we define E=~1+CAD_Weight to achieve the estimation with an intercept and covariate CAD_Weight for model expectation, and define V=~1 to just consider the intercept for model log-variance.

```
#The formulas f_t3 get the needed E and V
f_t3 <- list(E = ~1+CAD_Weight,V = ~1)

#Estimate the model_t3
model_t3 <- model_estimate(f_t3,data = data_obs,"Actual_Weight")
```

Task4

We compute predictions using model_predict() and then use ggplot() for prediction to plot prediction intervals as functions of CAD_Weight.
Consider the model_predict(), it uses results of model_t3 as input parameters, class='observation' to achieve the pred_test_t3.

```
#define the x-axis as CAD_Weight of data_test
x_plot <-data_test$CAD_Weight

#compute prediction pred_test_t3 using model_predict()
pred_test_t3 <- model_predict(model_t3$theta,model_t3$formulas,model_t3$Sigma_theta,
                              data_test,type ="observation",
                              alpha = 0.05, df = Inf,nonlinear.correction = TRUE)

#use ggplot() to plot the points and prediction intervals and specify the colors
ggplot() +
  geom_point(data = data_test,aes(x = CAD_Weight, y = Actual_Weight)) +
  geom_line(data = pred_test_t3,aes(x = x_plot, y = mu))+
  geom_ribbon(data=pred_test_t3,aes(x=x_plot,ymin=pred_test_t3$lwr,ymax=pred_test_t3$upr),
              alpha = 0.25,fill = 'blue')
```



Task5

The target is to estimate a model that uses an intercept and covariate CAD Weight formula for both the model expectations and log-variances.
Therefore, since the model_Z is flexible, so in this question we can see the advantage of it is that we just need to change the parameter formulas to get new estimated model.
We can find the target model here is very similiar to the model in Task3 and what we only need to change is to define V=~1+CAD_Weight using an intercept and covariate CAD Weight for new model log-variance.

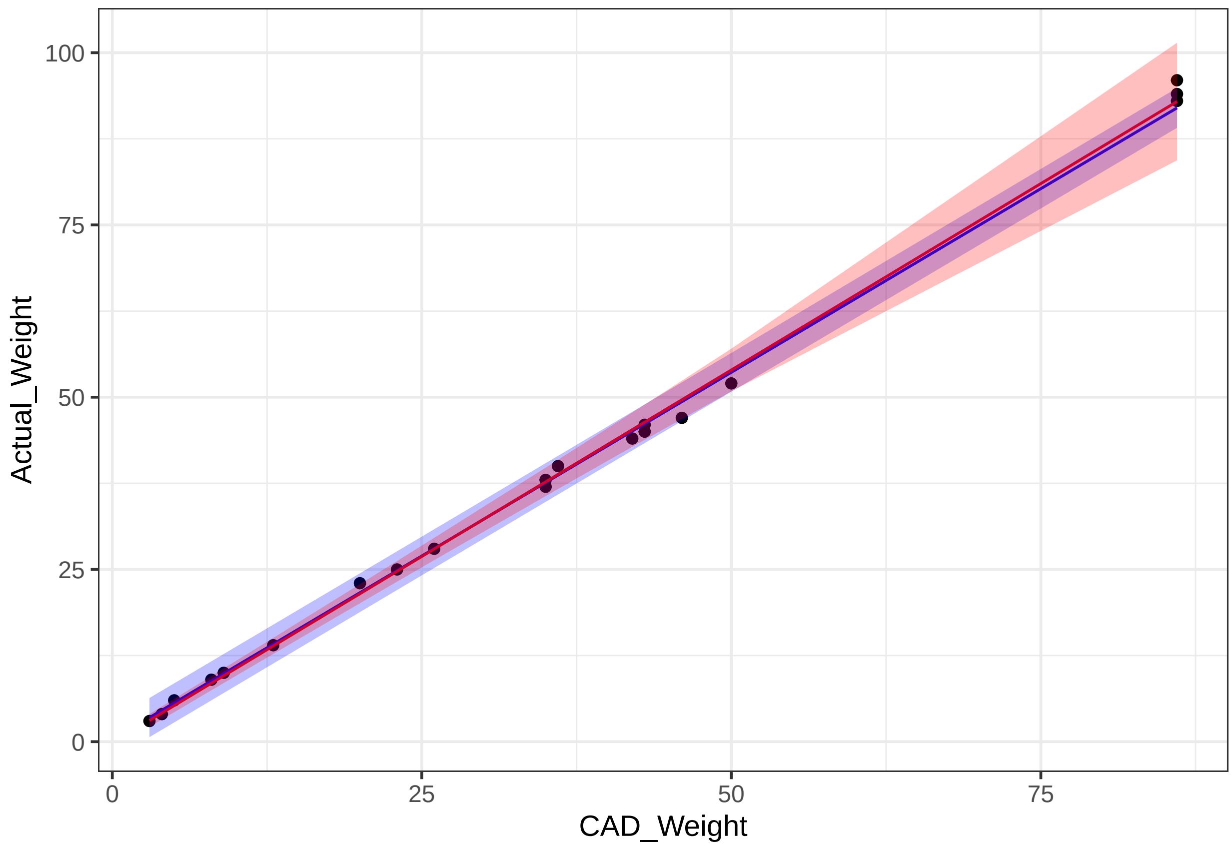
```
#The formulas f_t5 get the needed E and V
f_t5 <- list(E = ~1+CAD_Weight,V = ~1+CAD_Weight)

#Estimate the model_t5
model_t5 <- model_estimate(f_t5,data = data_obs,"Actual_Weight")
```

Task6

The task here uses the same method in Task4, we change the model_t3 to model_t5 we just defined.

```
#define the x-axis as CAD_Weight of data_test
x_plot <- data_test$CAD_Weight

#compute prediction pred_test_t5 using model_predict()
pred_test_t5 <- model_predict(model_t5$theta,model_t5$formulas,model_t5$Sigma_theta,
```

```
                              data_test,type ="observation",
                              alpha = 0.05, df = Inf,nonlinear.correction = TRUE)

#plot the points and prediction intervals with respect to both predictions
ggplot() +
  geom_point(data = data_test,aes(x = CAD_Weight, y = Actual_Weight)) +
  geom_line(data = pred_test_t3,aes(x = x_plot, y = mu),col='blue')+
  geom_line(data = pred_test_t5,aes(x = x_plot, y = mu),col='red')+
  geom_ribbon(data=pred_test_t3,aes(x=x_plot,ymin=pred_test_t3$lwr,ymax=pred_test_t3$upr),
              alpha = 0.25,fill = 'blue')+
  geom_ribbon(data=pred_test_t5,aes(x=x_plot,ymin=pred_test_t5$lwr,ymax=pred_test_t5$upr),
              alpha = 0.25,fill = 'red')
```



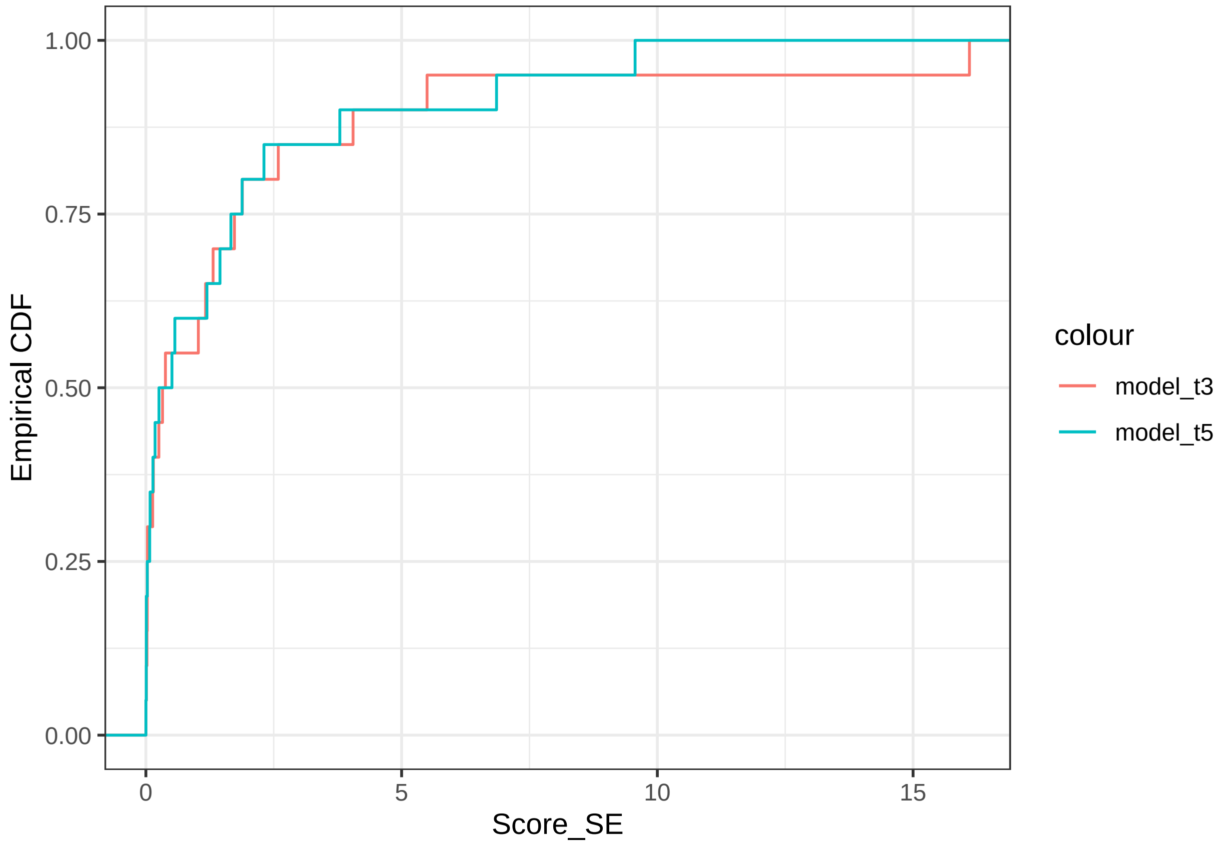
Task7

We use score_se(),score_ds() and score_interval() defined in CWA2020code to compute the scores with respect to model_t3 and model_t5 that we defined above.
By comparing the scores from the table, since the scores we get here are all negatively oriented, which means that smaller scores indicate better predictions, we find the model_t5 is better. Additionally, we also want to get an overview of the contributions to the average scores, so we plot ECDF for the scores and correspondingly find model_t5 is better since the probability reaches one earlier and graph starts with non-zero probability earlier.

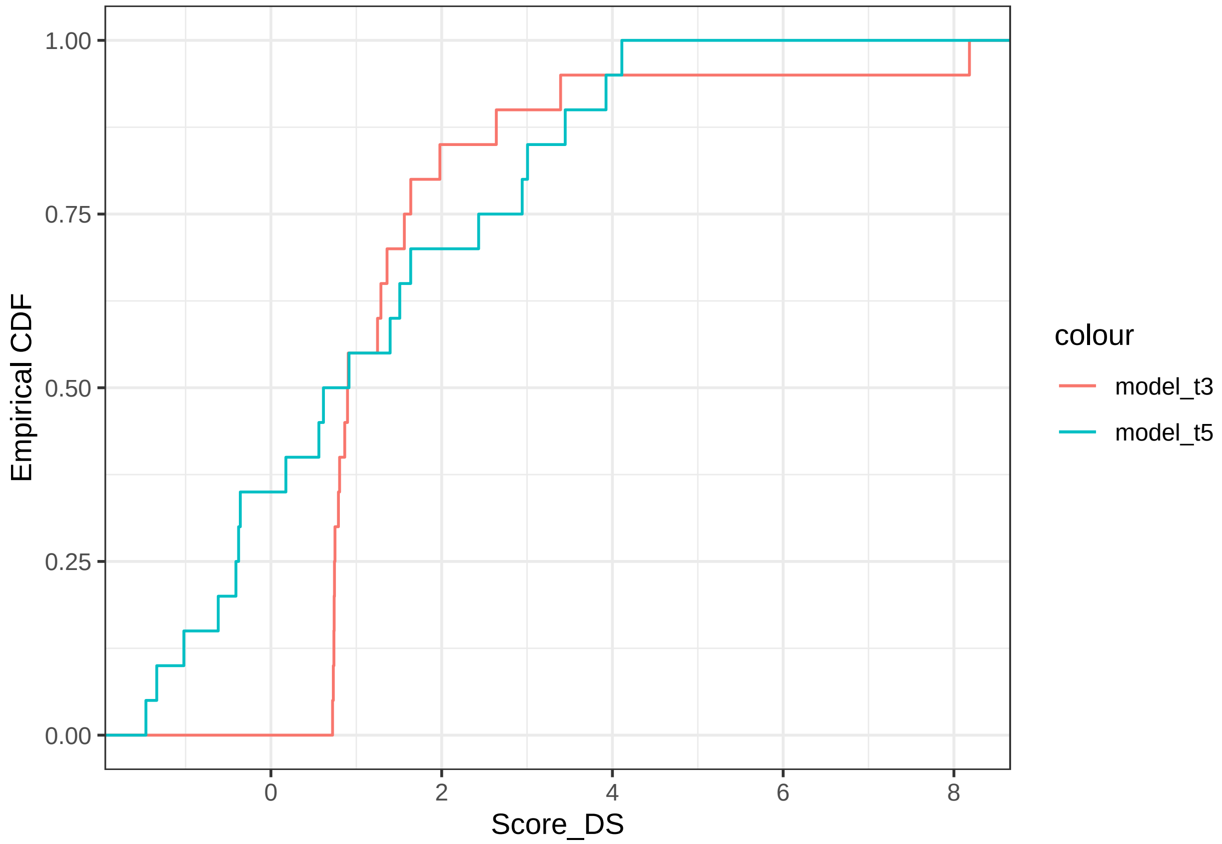
```r
#define model_scores() to compute three types of scores
model_scores <- function(pred,data){
  rbind(score_se = mean(score_se(pred,data$Actual_Weight)),
        score_ds = mean(score_ds(pred,data$Actual_Weight)),
        score_int = mean(score_interval(pred,data$Actual_Weight,alpha = 0.1)))
}

#define x7 as the output of scores for model_t3 and model_t5
x7<-cbind(model_scores(pred_test_t3,data_test),model_scores(pred_test_t5,data_test))
#change the column name of the table
colnames(x7)<-c("model_t3","model_t5")
#use knitr::kable(output) to better show and compare the scores
knitr::kable(x7)
```

	model_t3	model_t5
score_se	1.836020	1.528382
score_ds	1.598931	1.054257
score_int	6.783174	5.726996

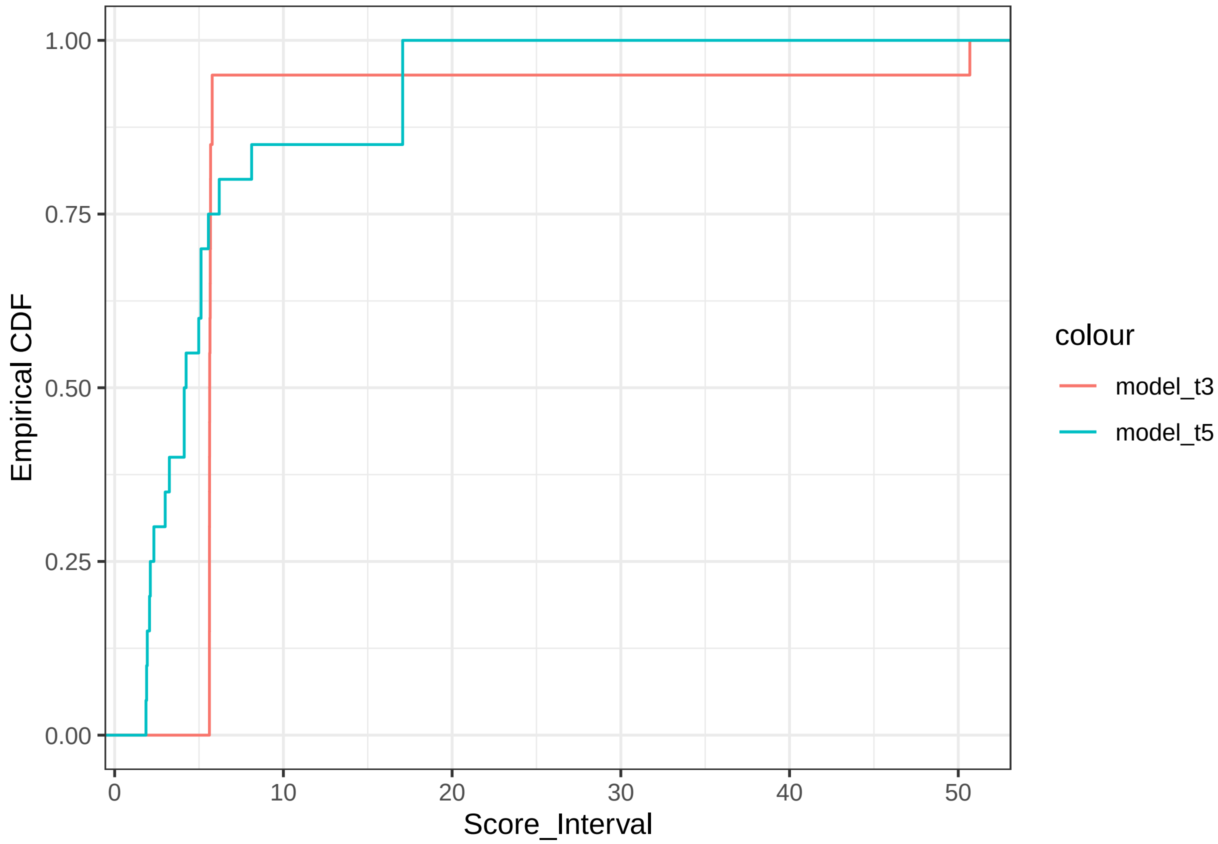
```r
#plot ECDF of the scores to see detailed contributions
ggplot() +
    stat_ecdf(aes(x = score_se(pred_test_t3,data_test$Actual_Weight),col = "model_t3")) +
    stat_ecdf(aes(x = score_se(pred_test_t5,data_test$Actual_Weight),col = "model_t5")) +
    xlab("Score_SE") + ylab("Empirical CDF")
```




```r
ggplot() +
    stat_ecdf(aes(x = score_ds(pred_test_t3,data_test$Actual_Weight),col = "model_t3")) +
    stat_ecdf(aes(x = score_ds(pred_test_t5,data_test$Actual_Weight),col = "model_t5")) +
    xlab("Score_DS") + ylab("Empirical CDF")
```




```r
ggplot() +
    stat_ecdf(aes(x=score_interval(pred_test_t3,data_test$Actual_Weight),col="model_t3"))+
    stat_ecdf(aes(x=score_interval(pred_test_t5,data_test$Actual_Weight),col="model_t5"))+
    xlab("Score_Interval") + ylab("Empirical CDF")
```



Task8

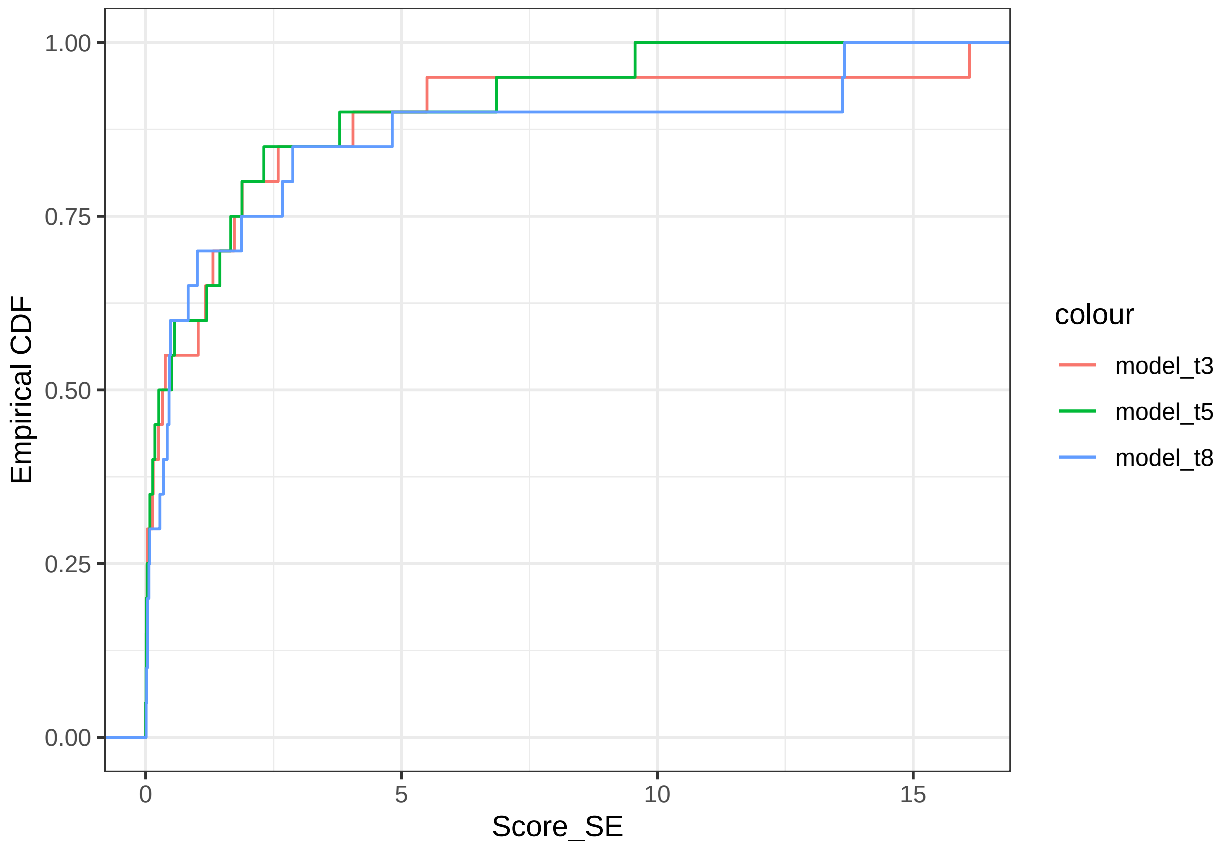
Since we want to consider role of interaction syntax A:B in this new model (syntax A:B here is CAD_Weight:Material), we estimate model_t8 with new formula f_t8. For f_t8, the expectation part is corresponding to the "~ 1 + A:B" syntax, and for the variance part, we decide to use the best model(achieved in Task7), "~1+CAD_Weight", to set the formula.
According to the table, by comparing scores of three models, we find the model_t8 just estimated in Task8 is not the best, we find the model_t5 estimated in Task5 is better than other two models since all the scores of model_t5 are smaller.
For overview of the contributions to the average scores, we use the same method to plot ECDF and compare them. We also find model_t5 is the best.

```
#define f_t8 and model_t8 we desired
f_t8<-list(E=~1+CAD_Weight:Material,V=~1+CAD_Weight)
model_t8<-model_estimate(f_t8,data=data_obs,response="Actual_Weight")

#compute prediction pred_test_t8 using model_predict()
pred_test_t8 <- model_predict(model_t8$theta,model_t8$formulas,model_t8$Sigma_theta,
                              data_test,type ="observation",
                              alpha = 0.05, df = Inf,nonlinear.correction = TRUE)

#define x8 as the output of scores for model_t3, model_t5 and model_t8
x8<-cbind(model_scores(pred_test_t3,data_test),
          model_scores(pred_test_t5,data_test),
          model_scores(pred_test_t8,data_test))
```

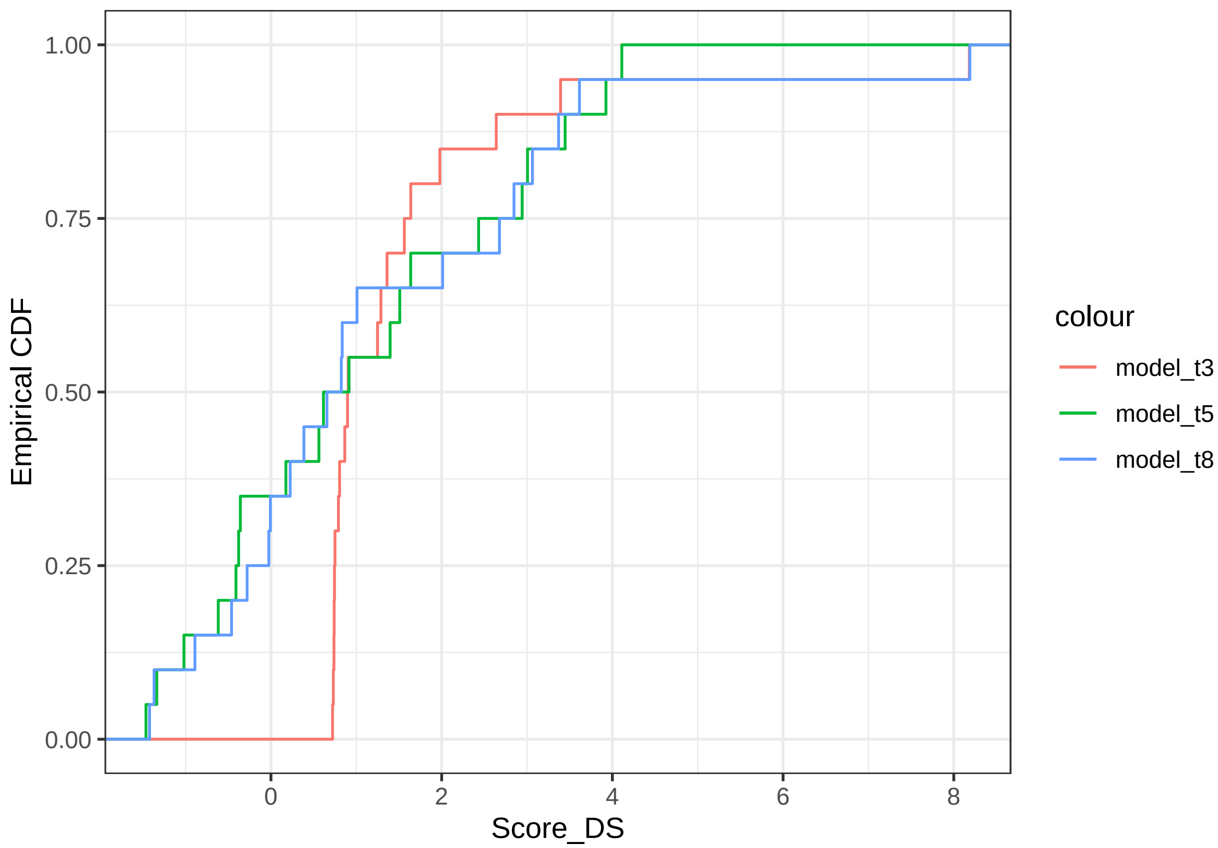
```r
#change the column name of the table
colnames(x8)<-c("model_t3","model_t5","model_t8")
#use knitr::kable(output) to better show and compare the scores
knitr::kable(x8)
```

	model_t3	model_t5	model_t8
score_se	1.836020	1.528382	2.202068
score_ds	1.598931	1.054257	1.262439
score_int	6.783174	5.726996	6.275502

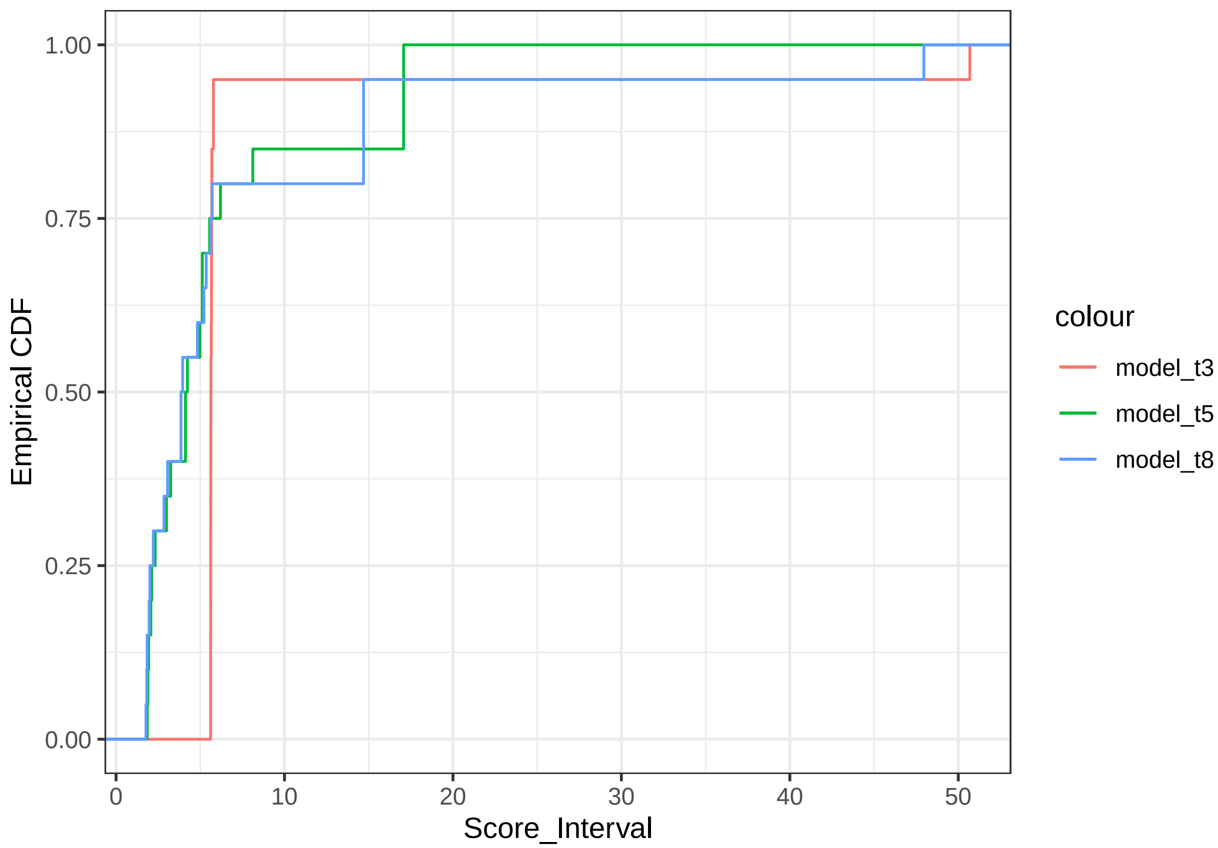
```r
#plot ECDF of the scores to see detailed contributions
ggplot() +
    stat_ecdf(aes(x = score_se(pred_test_t3,data_test$Actual_Weight),col = "model_t3")) +
    stat_ecdf(aes(x = score_se(pred_test_t5,data_test$Actual_Weight),col = "model_t5")) +
    stat_ecdf(aes(x = score_se(pred_test_t8,data_test$Actual_Weight),col = "model_t8")) +
    xlab("Score_SE") + ylab("Empirical CDF")
```




```r
ggplot() +
    stat_ecdf(aes(x = score_ds(pred_test_t3,data_test$Actual_Weight),col = "model_t3")) +
    stat_ecdf(aes(x = score_ds(pred_test_t5,data_test$Actual_Weight),col = "model_t5")) +
    stat_ecdf(aes(x = score_ds(pred_test_t8,data_test$Actual_Weight),col = "model_t8")) +
    xlab("Score_DS") + ylab("Empirical CDF")
```




```r
ggplot() +
    stat_ecdf(aes(x=score_interval(pred_test_t3,data_test$Actual_Weight),col="model_t3"))+
    stat_ecdf(aes(x=score_interval(pred_test_t5,data_test$Actual_Weight),col="model_t5"))+
    stat_ecdf(aes(x=score_interval(pred_test_t8,data_test$Actual_Weight),col="model_t8"))+
    xlab("Score_Interval") + ylab("Empirical CDF")
```

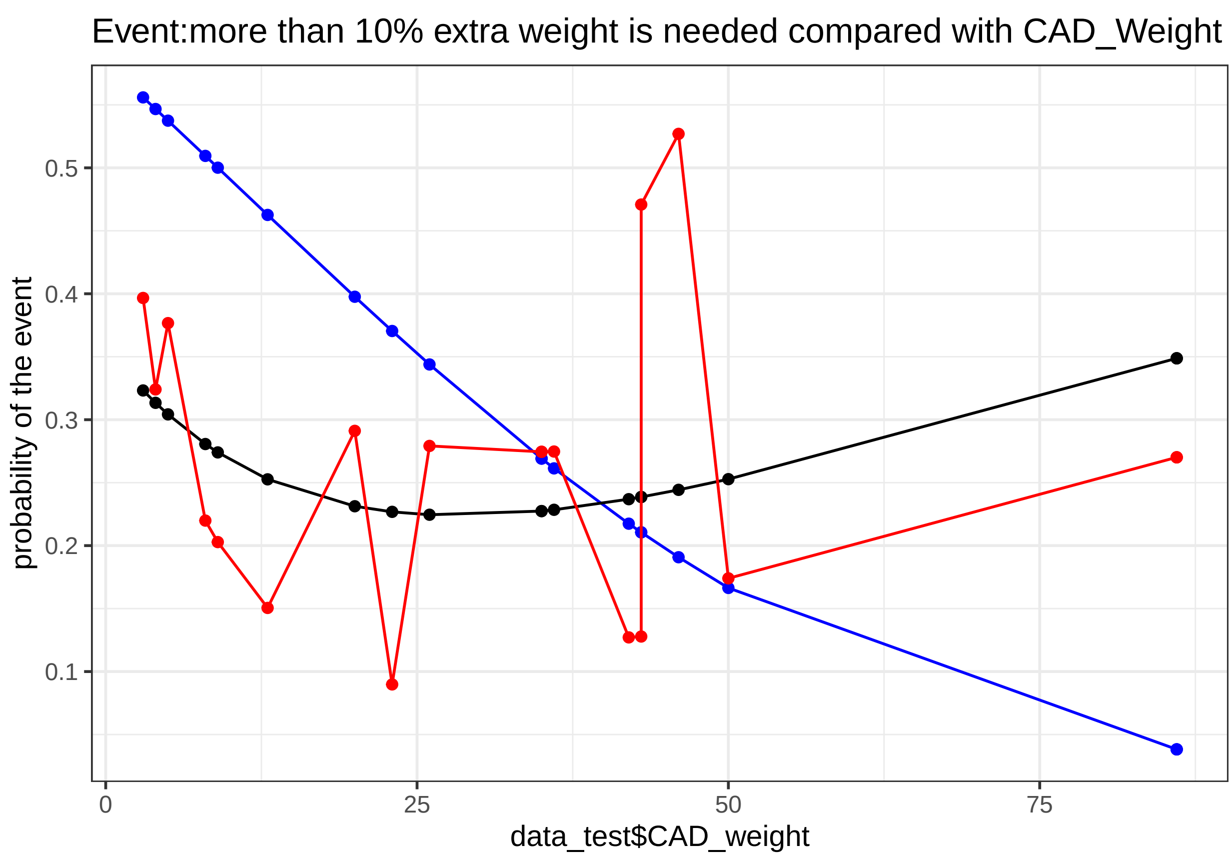


Task9

Since the predictive distributions are well approximated by Gaussian distributions, we use pnorm() to create three distributions with respect to pred_test_t3,pred_test_t5 and pred_test_t8. We know the event here is "more than 10% extra weight is needed compared with CAD_Weight", so we set the input vector q of pnorm to 1.1*data_test_CAD_Weight, and lower.tail = FALSE corresponding to indicator variable z in BS_score().
For BS_score, we create z as indicator variable, the condition for true is
$y \geq 1.1 * data_{test}CAD_{Weight}$, and then we can compute Brier score with (z-prob)^2) for single observation and at last compare the mean of them from the table.
Finally, we plot the probabilities for the event for three models and result a table to compare the Brier scores. Additionally, we plot the ECDF for three scores. By analyzing the table and plot, we find model_t3 is the best since the Brier score is the smallest and the overview of the contribution is correspondingly the best.

```
#compute the probabilities for the event for previous three models
prob_t3 <- pnorm(1.1*data_test$CAD_Weight,mean = pred_test_t3$mu,sd = pred_test_t3$sigma,
                lower.tail = FALSE,log.p = FALSE)
prob_t5 <- pnorm(1.1*data_test$CAD_Weight,mean = pred_test_t5$mu,sd = pred_test_t5$sigma,
                 lower.tail = FALSE,log.p = FALSE)
prob_t8 <- pnorm(1.1*data_test$CAD_Weight,mean = pred_test_t8$mu,sd = pred_test_t8$sigma,
                 lower.tail = FALSE,log.p = FALSE)

#define BS_score function to compute Brier score for the event
BS_score<-function(prob,y){
```

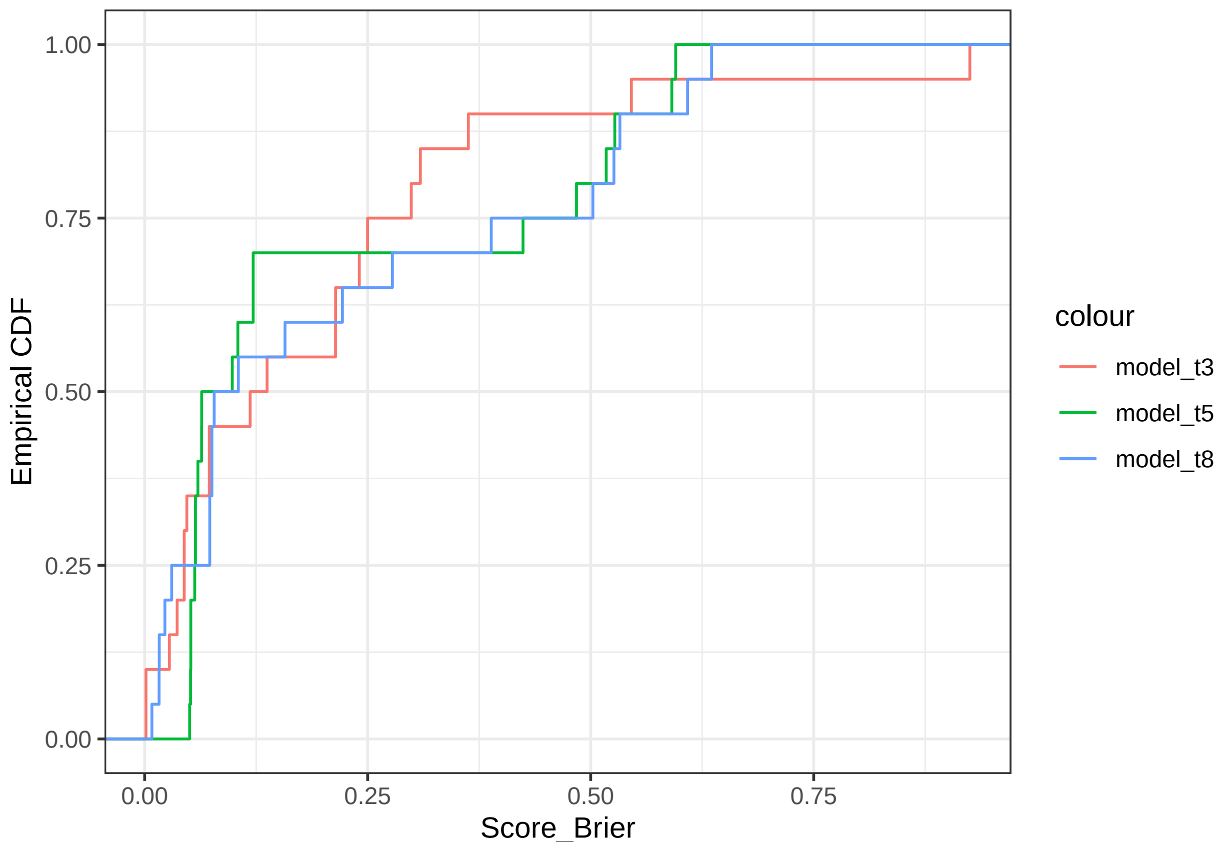
```
#create indicator vector z
  z <- ifelse(y>=1.1*data_test$CAD_Weight,1,0)
  (z-prob)^2
}

#plot the probabilities fot the event for previous three models
ggplot()+
      geom_point(aes(x = data_test$CAD_Weight, y = prob_t3), col="blue") +
      geom_line(aes(x = data_test$CAD_Weight, y = prob_t3), col="blue")+
      geom_point(aes(x = data_test$CAD_Weight, y = prob_t5), col="black") +
      geom_line(aes(x = data_test$CAD_Weight, y = prob_t5), col="black")+
      geom_point(aes(x = data_test$CAD_Weight, y = prob_t8), col="red")+
      geom_line(aes(x = data_test$CAD_Weight, y = prob_t8), col="red")+
      xlab("data_test$CAD_weight") + ylab("probability of the event")+
      ggtitle("Event:more than 10% extra weight is needed compared with CAD_Weight")
```




```
#get the table and compare Brier scores for the test data for the event
x9<-cbind(mean(BS_score(prob_t3,data_test$Actual_Weight)),
          mean(BS_score(prob_t5,data_test$Actual_Weight)),
          mean(BS_score(prob_t8,data_test$Actual_Weight)))
colnames(x9)<-c("model_t3","model_t5","model_t8")
knitr::kable(x9)
```

model_t3	model_t5	model_t8
0.198158	0.207376	0.2211858

```
#plot ECDF of the scores to see detailed contributions
ggplot() +
    stat_ecdf(aes(x = BS_score(prob_t3,data_test$Actual_Weight),col = "model_t3")) +
    stat_ecdf(aes(x = BS_score(prob_t5,data_test$Actual_Weight),col = "model_t5")) +
    stat_ecdf(aes(x = BS_score(prob_t8,data_test$Actual_Weight),col = "model_t8")) +
    xlab("Score_Brier") + ylab("Empirical CDF")
```



Task10

The computation method to get Brier score is partly similiar to Task9. However, here we need to define new negative log-likelihood function and optimal function to estimate new model.
We define total_score() to achieve scores. Firstly, since we need the size and parameters of Cauchy distribution, we create y_test to test and estimate y_est. For y_est, it is sampled from large size model y_est_huge which we think it gives us asymptotic properties for y_est. Then, we create indicator variable z1 with input vector q10=rep(0,N) since the size is N and condition is $y \leq 0$. We use optim() to get theta_est. Finally, we compute score_ds, score_se and score_bs respectively about true model and estimated model.
For the questions, we find the parameter estimates but score differences are complex.
By comparing the different scores from the table, we can clearly find, as sample size N increases, score_ds and score_se increase fast and the score differences between true model and estimated model become larger. However, for score_bs, we find the differences are extremely small with $N \geq 100$ and decrease quickly.

Therefore, we claim Brier score is the stabilised one among these three types of scores, and we do not need to expect a similiar comparison for Squared-Error or Dawid-Sebastiani scores to stabilise.

```r
#define neg_ll() to compute the neg_log_lik for the estimated Cauchy model in task10
neg_ll<-function(theta,y) {
  -sum(dcauchy(y, location = theta[1], scale = exp(theta[2]), log = TRUE))
}

#define total_score() to compute and compare three types of scores
total_score<-function(N){
#create true model and estimate y_test
     y_test <- rcauchy(N, location = 2, scale = 5)
     y_est_huge<-rcauchy(1000000, location = 2, scale = 5)
     y_est <- sample(y_est_huge,N)

#create indicator variable z1
     q10 <- rep(0,N)
     z1 <- ifelse(y_test<=q10,1,0)

#use optim() to get optimisation parameter theta_est.
     optimal <- optim(rep(0,2),fn = neg_ll, y = y_test,
                      method = "BFGS",control = list(maxit = 5000))

#achieve parameters theta_true and theta_est
     theta_true <- data.frame(mu=2,sigma=5)
     theta_est <-data.frame(mu=optimal$par[1],sigma=exp(optimal$par[2]))

#achieve two Cauchy distributions with respect to theta_true and theta_est
     x1 <- pcauchy(q10, location = 2, scale = 5,lower.tail = TRUE,log.p = FALSE)
     x2 <- pcauchy(q10, location = optimal$par[1], scale = exp(optimal$par[2]),
                   lower.tail = TRUE,log.p = FALSE)

#compute and compare scores to determine stabilised score
     rbind( DS_true = mean(score_ds(theta_true,y_test)),
            DS_est = mean(score_ds(theta_est,y_est)),
            SE_true = mean(score_se(theta_true,y_test)),
            SE_est = mean(score_se(theta_est,y_est)),
            BS_true = mean((z1-x1)^2),
            BS_est = mean((z1-x2)^2),
            DS_diff=abs(mean(score_ds(theta_true,y_test))-mean(score_ds(theta_est,y_est))),
            SE_diff=abs(mean(score_se(theta_true,y_test))-mean(score_se(theta_est,y_est))),
            BS_diff=abs(mean((z1-x1)^2)-mean((z1-x2)^2)))
}

#achieve scores for the model with respect to different size N
x1<-cbind(total_score(5), total_score(10), total_score(20),
          total_score(40),total_score(80))
x2<-cbind(total_score(160),total_score(320),total_score(640),
          total_score(1280),total_score(2560))
x3<-cbind(total_score(5120),total_score(10240),total_score(20480),
          total_score(40960),total_score(81920))

colnames(x1)<-c("N=5","N=10","N=20","N=40","N=80")
colnames(x2)<-c("N=160","N=320","N=640","N=1280","N=2560")
```

```r
colnames(x3)<-c("N=5120","N=10240","N=20480","N=40960","N=81920")
#get the tables and compare the score differences
knitr::kable(x1)
```

	N=5	N=10	N=20	N=40	N=80
DS_true	45.6681904	14.6459396	10.0969775	3.749333e+03	82.8310349
DS_est	14.9027029	23.6680668	5.9150845	6.083287e+01	67.1175588
SE_true	1061.2328646	285.6765944	171.9525413	9.365286e+04	1990.3039759
SE_est	3921.4999880	201.1885327	84.5452006	1.318596e+03	2044.1346601
BS_true	0.2404460	0.2404460	0.2162222	2.222782e-01	0.2162222
BS_est	0.2411272	0.2526006	0.2144644	2.202475e-01	0.2108059
DS_diff	30.7654876	9.0221272	4.1818930	3.688500e+03	15.7134761
SE_diff	2860.2671234	84.4880617	87.4073407	9.233426e+04	53.8306842
BS_diff	0.0006812	0.0121546	0.0017578	2.030600e-03	0.0054163

```r
knitr::kable(x2)
```

	N=160	N=320	N=640	N=1280	N=2560
DS_true	34.2557301	7.494991e+02	1.546894e+03	9.258114e+03	1.194826e+05
DS_est	282.0958599	1.244415e+02	1.568721e+03	7.511790e+02	3.359340e+03
SE_true	775.9213579	1.865701e+04	3.859187e+04	2.313724e+05	2.986985e+06
SE_est	5835.6415316	3.739641e+03	3.710717e+04	1.901961e+04	8.858221e+04
BS_true	0.2298481	2.449880e-01	2.362825e-01	2.374180e-01	2.363772e-01
BS_est	0.2293587	2.434003e-01	2.364293e-01	2.373594e-01	2.363585e-01
DS_diff	247.8401297	6.250576e+02	2.182732e+01	8.506935e+03	1.161233e+05
SE_diff	5059.7201737	1.491736e+04	1.484696e+03	2.123528e+05	2.898403e+06
BS_diff	0.0004894	1.587700e-03	1.468000e-04	5.860000e-05	1.870000e-05

```r
knitr::kable(x3)
```

	N=5120	N=10240	N=20480	N=40960	N=81920
DS_true	9.303929e+02	8.757243e+04	6.092096e+03	9.272507e+03	8.078570e+03
DS_est	9.399822e+02	1.323273e+04	1.738116e+04	7.529394e+03	1.954419e+04
SE_true	2.317935e+04	2.189230e+06	1.522219e+05	2.317322e+05	2.018838e+05
SE_est	2.421574e+04	3.464097e+05	4.459592e+05	1.873967e+05	4.791105e+05
BS_true	2.360460e-01	2.350524e-01	2.365664e-01	2.347094e-01	2.349637e-01
BS_est	2.360452e-01	2.350770e-01	2.365579e-01	2.347036e-01	2.349616e-01
DS_diff	9.589219e+00	7.433970e+04	1.128907e+04	1.743113e+03	1.146562e+04
SE_diff	1.036385e+03	1.842821e+06	2.937372e+05	4.433552e+04	2.772267e+05
BS_diff	8.000000e-07	2.460000e-05	8.500000e-06	5.800000e-06	2.200000e-06