Statistical Computing CW A

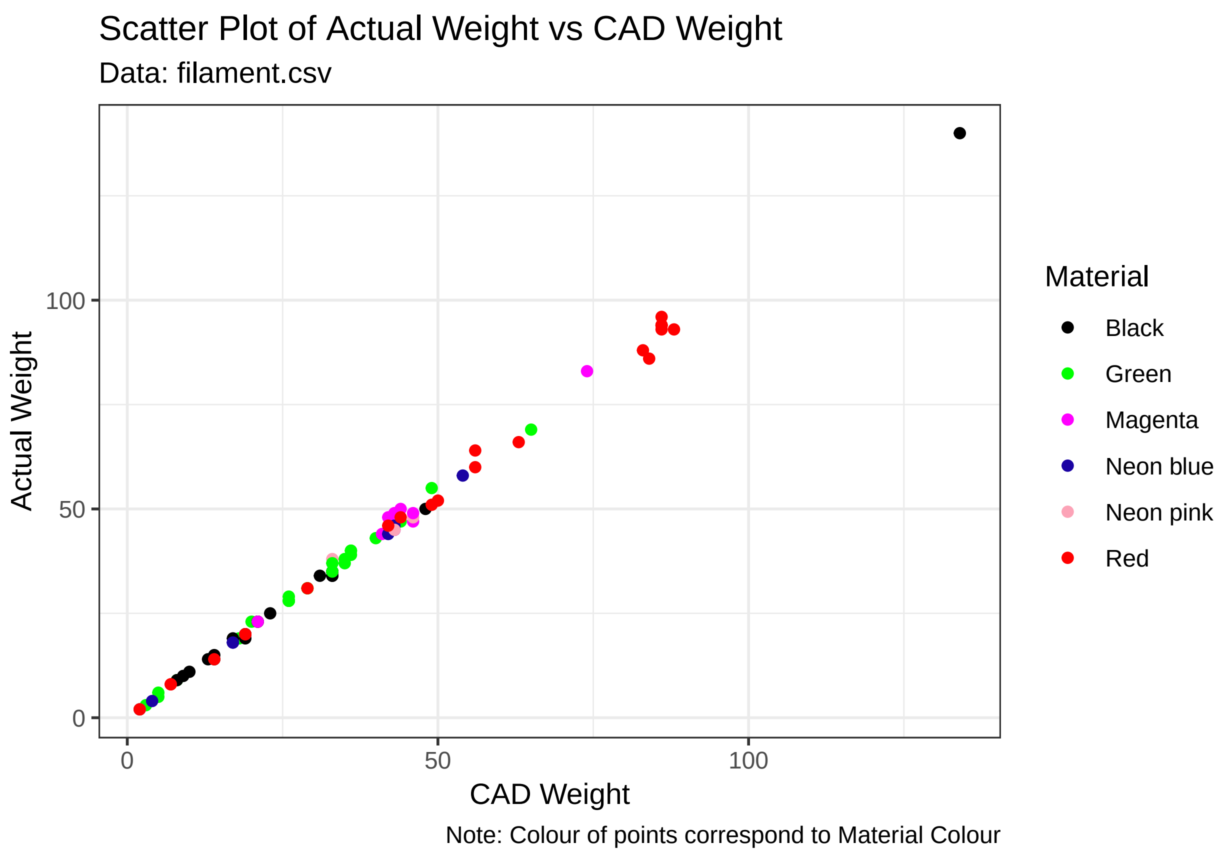
Matin Mahmood (s1841215)

25 February 2020

```r
#Given in Coursework Document
source("CWA2020code.R")
suppressPackageStartupMessages(library(tidyverse))
theme_set(theme_bw())
filament <- read.csv("filament.csv", stringsAsFactors = FALSE)
```

Task 1: Actual Weight and CAD Weight Plot

```r
ggplot(filament) +
  geom_point(aes(CAD_Weight, Actual_Weight, col = Material)) +
  scale_color_manual(values = c("Black" = "black",
                                "Red" = "red",
                                "Green" = "green",
                                "Magenta" = "magenta",
                                "Neon pink" = "#fca3b7", #differentiate from magenta
                                "Neon blue"= "#1b03a3")) +

  labs(subtitle="Data: filament.csv",
       y="Actual Weight",
       x="CAD Weight",
       title="Scatter Plot of Actual Weight vs CAD Weight",
       caption = "Note: Colour of points correspond to Material Colour")
```



Task 2: Model Estimate Function

```
model_estimate <- function(formulas, data, response){

  z_data=model_Z(formulas,data)

  opt <- optim(rep(0, ncol(z_data[["ZE"]])),
               fn = neg_log_lik,
               Z = z_data,
               y = data[[response]],
               method = "BFGS",
               control = list(maxit = 5000), # Anouncement on Learn, Ensures Convergence
               hessian = TRUE)

  theta <- opt$par
  Sigma_theta <- solve(opt$hessian)


  return (list(theta = theta, formulas = formulas, Sigma_theta=Sigma_theta))

  }
```

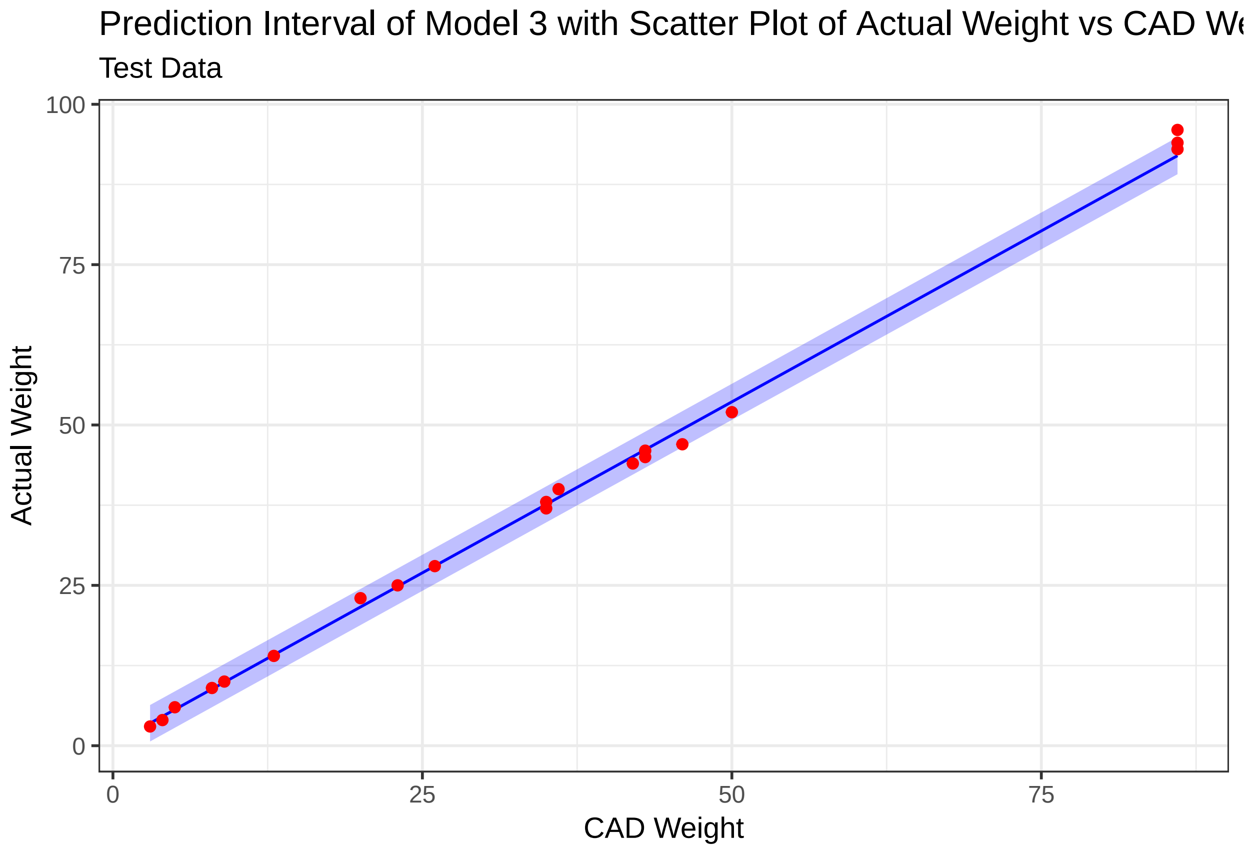
Task 3: Estimating Model 3

```
data_obs <- filament[filament$Class=="obs",] #Extract Observed Data

formulas_3 <- list(E = ~ 1 + CAD_Weight, V = ~ 1)

estimates_3 <- model_estimate(formulas_3, data_obs, "Actual_Weight")

estimates_3
```

```
## $theta
## [1] 0.3058488 1.0660628 0.6901968
##
## $formulas
## $formulas$E
## ~1 + CAD_Weight
##
## $formulas$V
## ~1
##
##
## $Sigma_theta
##                [,1]          [,2]          [,3]
## [1,]  9.026507e-02 -1.708832e-03  6.877290e-08
## [2,] -1.708832e-03  4.791775e-05 -1.475095e-09
## [3,]  6.877290e-08 -1.475095e-09  2.941178e-02
```

Task 4: Model 3 Prediction Intervals Plot

```
data_test <- filament[filament$Class=="test",] #Extract Test Data

model3 <- model_predict(estimates_3[["theta"]], estimates_3[["formulas"]], estimates_3[["Sigma_theta"]]


pred_plot_4 <- cbind(data_test, model3)

ggplot() +
  geom_ribbon(data = pred_plot_4,
              aes(CAD_Weight, ymin = lwr, ymax = upr),
              alpha = 0.25, fill = "blue") +
  geom_line(data = pred_plot_4, aes(CAD_Weight, mu), col = "blue") +
  geom_point(data = data_test, aes(CAD_Weight, Actual_Weight), col = "red") +
  labs(subtitle="Test Data",
       y="Actual Weight",
       x="CAD Weight",
       title="Prediction Interval of Model 3 with Scatter Plot of Actual Weight vs CAD Weight")
```



Task 5: Estimating Model 5

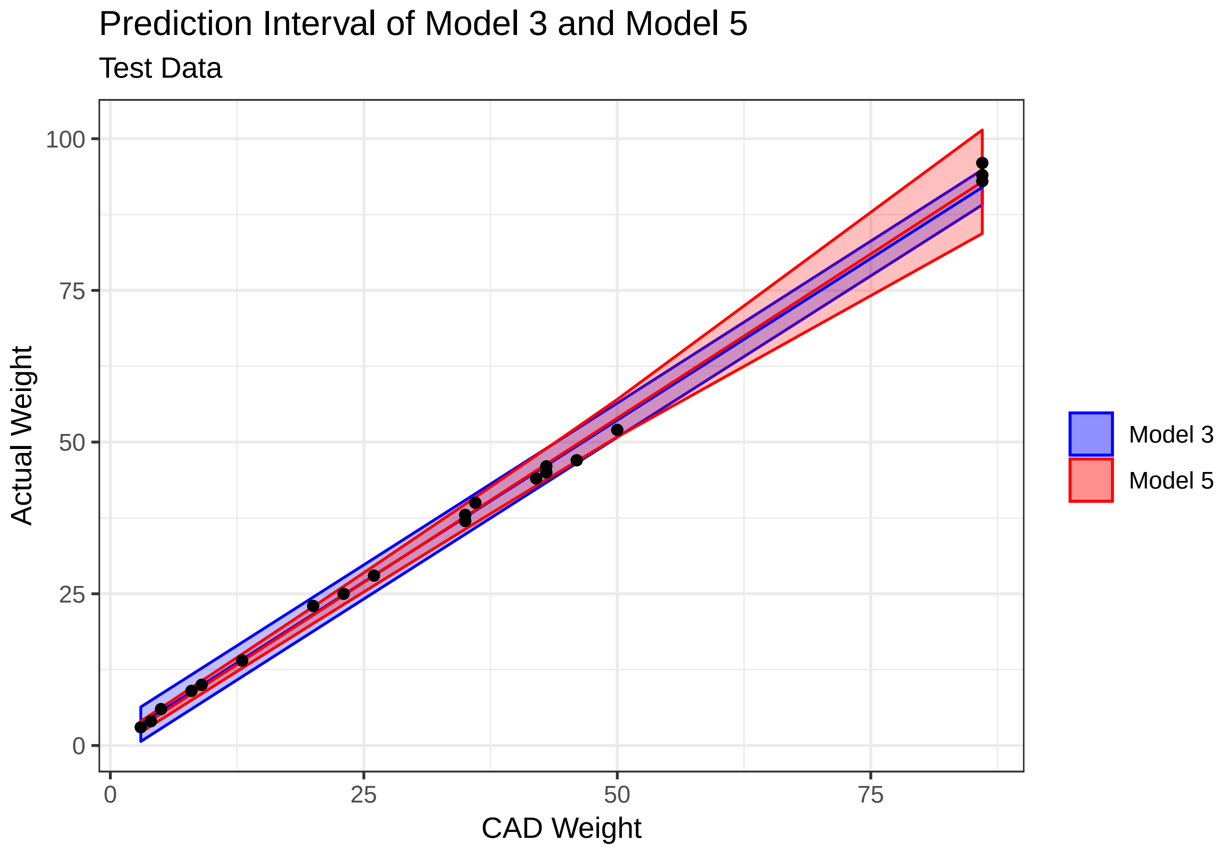
```
formulas_5 <- list(E = ~ 1 + CAD_Weight, V = ~ 1+ CAD_Weight)

estimates_5 <- model_estimate(formulas_5, data_obs, "Actual_Weight")

estimates_5
```

```
## $theta
## [1] -0.16389778  1.08222367 -1.80995547  0.05356663
##
## $formulas
## $formulas$E
## ~1 + CAD_Weight
##
## $formulas$V
## ~1 + CAD_Weight
##
##
## $Sigma_theta
##               [,1]          [,2]          [,3]          [,4]
## [1,]  0.0237301631 -8.276727e-04 -1.111045e-03  3.112610e-05
## [2,] -0.0008276727  4.699560e-05  6.293605e-05 -1.763996e-06
```

```
## [3,] -0.0011110445  6.293605e-05  1.487803e-01 -3.347233e-03
## [4,]  0.0000311261 -1.763996e-06 -3.347233e-03  9.385984e-05
```

Task 6: Model 3 & Model 5 Prediction Intervals Plot

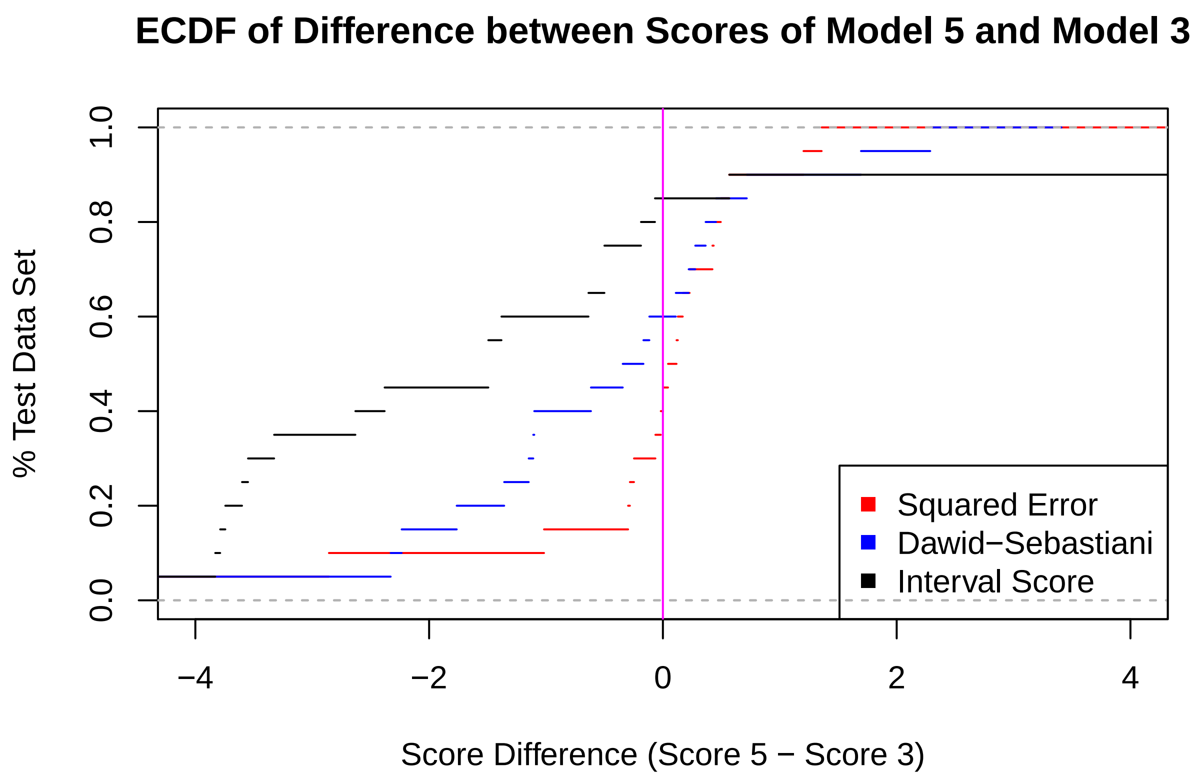
```
model5 <- model_predict(estimates_5[["theta"]], estimates_5[["formulas"]], estimates_5[["Sigma_theta"]]


pred_plot_6 <- cbind(data_test, model5)

ggplot() +
  geom_ribbon(data = pred_plot_4,
              aes(CAD_Weight, ymin = lwr, ymax = upr,col="Model 3",fill = "Model 3"),
              alpha = 0.25) +
  geom_ribbon(data = pred_plot_6,
              aes(CAD_Weight, ymin = lwr, ymax = upr, col="Model 5", fill = "Model 5"),
              alpha = 0.25) +
  geom_line(data = pred_plot_4, aes(CAD_Weight, mu), col = "blue") +
  geom_line(data = pred_plot_6, aes(CAD_Weight, mu), col = "red") +
  geom_point(data = data_test, aes(CAD_Weight, Actual_Weight), col = "black")+
  scale_colour_manual("",values = c("Model 3" = "blue","Model 5" = "red"))+
  scale_fill_manual("",values = c("Model 3" = "blue", "Model 5" = "red"))+
  labs(subtitle="Test Data",
       y="Actual Weight",
       x="CAD Weight",
       title="Prediction Interval of Model 3 and Model 5")
```



Task 7: Model 3 and Model 5 Score Comparison

```
model_scores <- function (modelQ, data, response, alpha){

return(data.frame(
  SES=c((score_se(modelQ, data[[response]]))),  #Squared Error
  DSS=c((score_ds(modelQ, data[[response]]))),  #Dawid-Sebastiani
  IS=c((score_interval(modelQ, data[[response]], alpha = alpha))) #Interval Score
))
}

Score_3 = model_scores(model3, data_test, 'Actual_Weight', 0.1)
Score_5 = model_scores(model5, data_test, 'Actual_Weight', 0.1)

S5_S3 = Score_5 - Score_3

plot(ecdf(S5_S3[["SES"]]),
      xlim=c(-4,4),
      xlab="Score Difference (Score 5 - Score 3)",
      ylab="% Test Data Set",
      main="ECDF of Difference between Scores of Model 5 and Model 3",
      col="red",cex=0)
lines(ecdf(S5_S3[["DSS"]]),
```

```r
      col="blue",cex=0)
lines(ecdf(S5_S3[["IS"]]),
      col="black",cex=0)
abline(v=0, col="magenta")

legend('bottomright',
       legend=c("Squared Error","Dawid-Sebastiani","Interval Score"),
       col=c("red","blue","black"),
       pch=15)
```



Comparing Model 3 and Model 5

The interval score for Model 5 is better than Model 3 on appoximately 80% of the test data set. The Dawid-Sebastiani score for Model 5 is better than Model 3 on approximately 60% of the test data set. The Squared-Error score for Model 5 is better than Model 3 on approximately 35% of the test data set.

Only the interval score and Dawid-Sebastiani score agree on Model 5 being better than Model 3. The Squared-Error score suggest that Model 3 is marginally better than Model 5.

Task 8

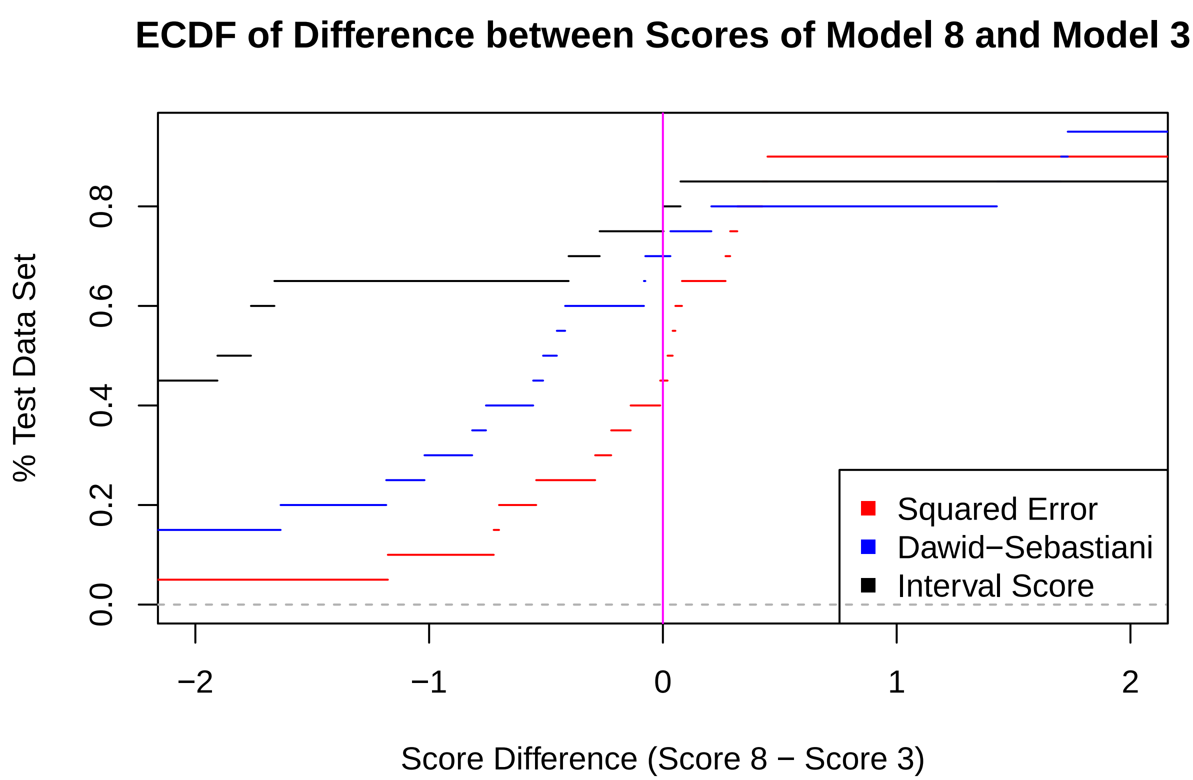
```r
formulas_8 <- list(E = ~ 1 + CAD_Weight:Material, V = ~ 1+ CAD_Weight)

estimates_8 <- model_estimate(formulas_8, data_obs, "Actual_Weight")
```

```r
estimates_8
```

```
## $theta
## [1] -0.07467418  1.06083919  1.08529823  1.10359160  1.06565852  1.09929646
## [7]  1.07417646 -1.89254757  0.05068071
##
## $formulas
## $formulas$E
## ~1 + CAD_Weight:Material
##
## $formulas$V
## ~1 + CAD_Weight
##
##
## $Sigma_theta
##                [,1]          [,2]          [,3]          [,4]
##  [1,]  0.0220436724 -8.933195e-04 -7.814548e-04 -5.665713e-04
##  [2,] -0.0008933195  9.998818e-05  3.139510e-05  2.235492e-05
##  [3,] -0.0007814548  3.139510e-05  8.027081e-05  2.023356e-05
##  [4,] -0.0005665713  2.235492e-05  2.023356e-05  1.129322e-04
##  [5,] -0.0009598999  3.770394e-05  3.432199e-05  2.532102e-05
##  [6,] -0.0006035296  2.811462e-05  2.049873e-05  1.352638e-05
##  [7,] -0.0006636840  2.687504e-05  2.353288e-05  1.706942e-05
##  [8,]  0.0030452444 -4.922692e-04 -1.783771e-05  1.219876e-04
##  [9,] -0.0000853920  1.380483e-05  4.908143e-07 -3.422572e-06
##                [,5]          [,6]          [,7]          [,8]
##  [1,] -9.598999e-04 -6.035296e-04 -6.636840e-04  3.045244e-03
##  [2,]  3.770394e-05  2.811462e-05  2.687504e-05 -4.922692e-04
##  [3,]  3.432199e-05  2.049873e-05  2.353288e-05 -1.783771e-05
##  [4,]  2.532102e-05  1.352638e-05  1.706942e-05  1.219876e-04
##  [5,]  3.070196e-04  2.235800e-05  2.892262e-05  2.631421e-04
##  [6,]  2.235800e-05  3.959546e-04  1.810284e-05 -1.293340e-03
##  [7,]  2.892262e-05  1.810284e-05  8.642906e-05 -8.482079e-05
##  [8,]  2.631421e-04 -1.293340e-03 -8.482079e-05  1.519116e-01
##  [9,] -7.379281e-06  3.626801e-05  2.378478e-06 -3.435042e-03
##                [,9]
##  [1,] -8.539200e-05
##  [2,]  1.380483e-05
##  [3,]  4.908143e-07
##  [4,] -3.422572e-06
##  [5,] -7.379281e-06
##  [6,]  3.626801e-05
##  [7,]  2.378478e-06
##  [8,] -3.435042e-03
##  [9,]  9.631711e-05
```

```r
model8 <- model_predict(estimates_8[["theta"]], estimates_8[["formulas"]], estimates_8[["Sigma_theta"]])


Score_8 = model_scores(model8, data_test, 'Actual_Weight', 0.1)

S8_S3 = Score_8 - Score_3
S8_S5 = Score_8 - Score_5
```

```r
plot(ecdf(S8_S3[["SES"]]),
      xlim=c(-2,2),
      xlab="Score Difference (Score 8 - Score 3)",
      ylab="% Test Data Set",
      main="ECDF of Difference between Scores of Model 8 and Model 3",
      col="red",cex=0)
lines(ecdf(S8_S3[["DSS"]]),
     col="blue",cex=0)
lines(ecdf(S8_S3[["IS"]]),
     col="black",cex=0)
abline(v=0, col="magenta")
legend('bottomright',
       legend=c("Squared Error","Dawid-Sebastiani","Interval Score"),
       col=c("red","blue","black"),
       pch=15)
```



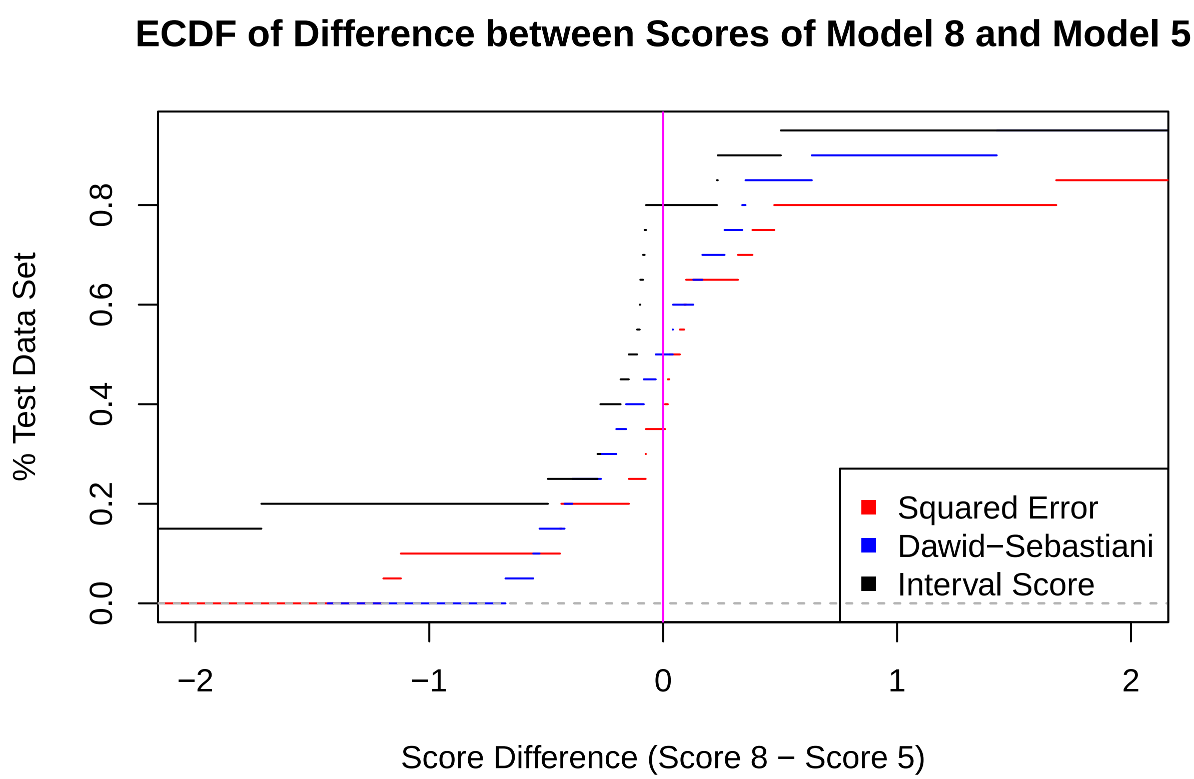
Comparing Model 8 and Model 3

The interval score for Model 8 is better than Model 3 on appoximately **75%** of the test data set. The Dawid-Sebastiani score for Model 8 is better than Model 3 on approximately **70%** of the test data set. The Squared-Error score for Model 8 is better than Model 3 on approximately **40%** of the test data set.

Only the interval score and Dawid-Sebastiani score agree on Model 8 being better than Model 3. The Squared-Error score suggest that Model 3 is marginally better than Model 8.

```r
plot(ecdf(S8_S5[["SES"]]),
      xlim=c(-2,2),
```

```r
       xlab="Score Difference (Score 8 - Score 5)",
       ylab="% Test Data Set",
       main="ECDF of Difference between Scores of Model 8 and Model 5",
       col="red",cex=0)
lines(ecdf(S8_S5[["DSS"]]),
      col="blue",cex=0)
lines(ecdf(S8_S5[["IS"]]),
      col="black",cex=0)
abline(v=0, col="magenta")
legend('bottomright',
        legend=c("Squared Error","Dawid-Sebastiani","Interval Score"),
        col=c("red","blue","black"),
        pch=15)
```



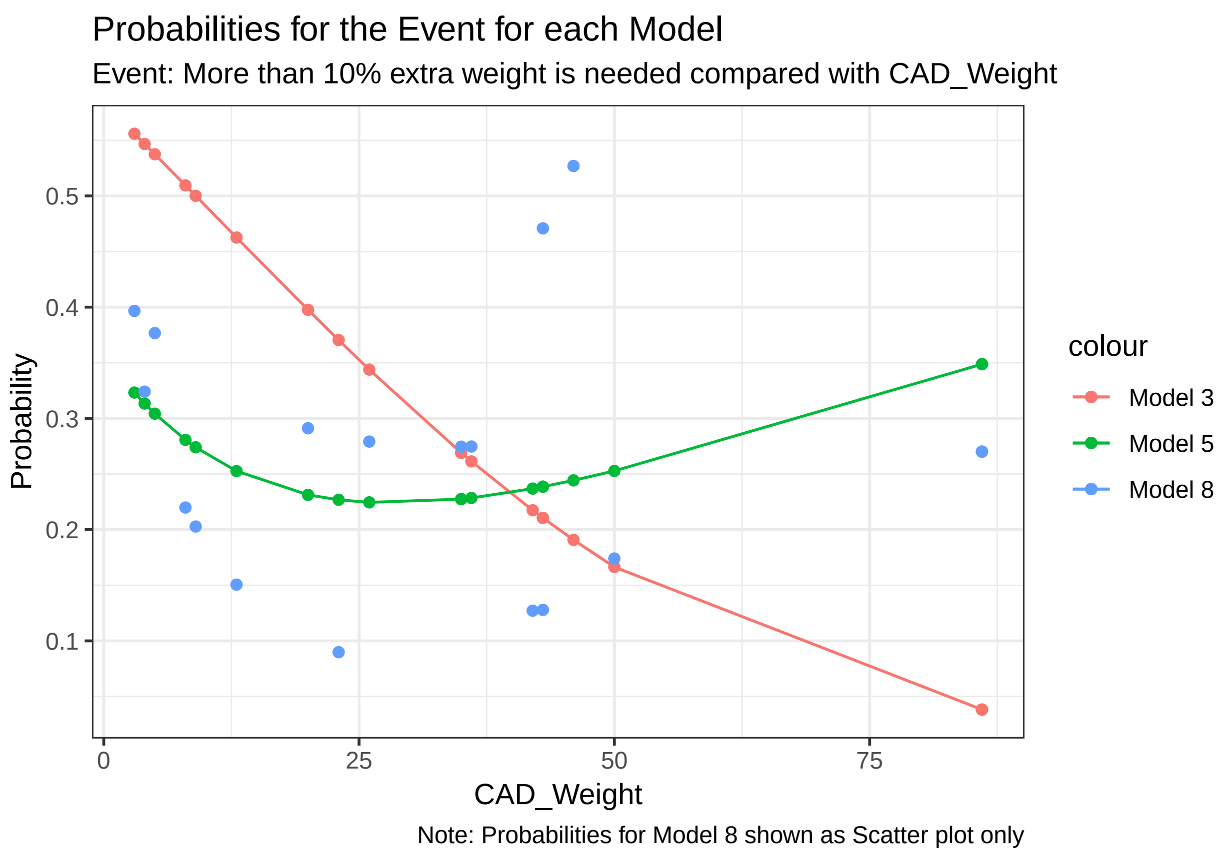
Comparing Model 8 and Model 5

The interval score for Model 8 is better than Model 5 on appoximately **80%** of the test data set. The Dawid-Sebastiani score for Model 8 is better than Model 5 on approximately **50%** of the test data set. The Squared-Error score for Model 8 is better than Model 5 on approximately **35%** of the test data set.

Only the interval score agrees on Model 8 being better than Model 5. The Dawid-Sebastiani score dow not give a conclusive indication on which model is better. The Squared-Error score suggest that Model 5 is better than Model 8.

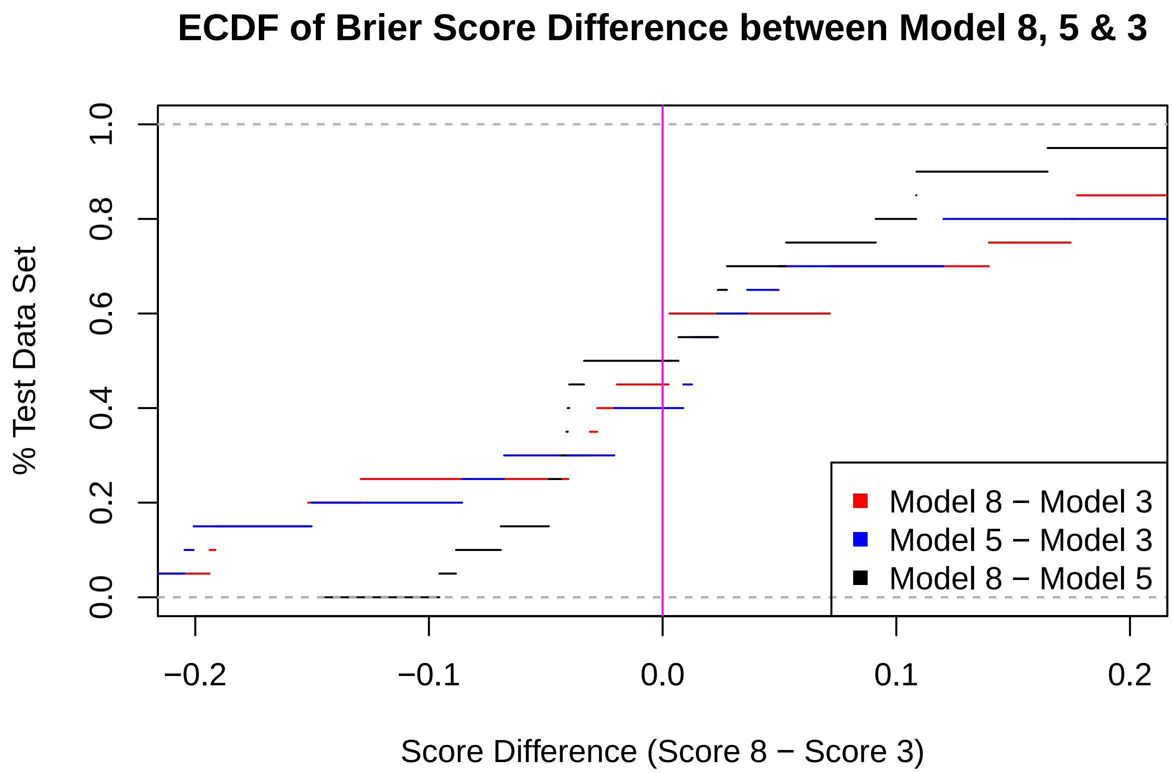
Task 9

```r
#Estimating Probability Distributions
Prob_3 = pnorm(q=1.1*data_test$CAD_Weight,mean = model3$mu,sd = model3$sigma,lower.tail = FALSE)
Prob_5 = pnorm(q=1.1*data_test$CAD_Weight,mean = model5$mu,sd = model5$sigma,lower.tail = FALSE)
Prob_8 = pnorm(q=1.1*data_test$CAD_Weight,mean = model8$mu,sd = model8$sigma,lower.tail = FALSE)

Prob_CAD=cbind(Prob_3,Prob_5,Prob_8,data_test)

ggplot() +
  geom_point(data=Prob_CAD,aes(CAD_Weight,Prob_3, col = "Model 3" )) +
  geom_point(data=Prob_CAD,aes(CAD_Weight,Prob_5, col = "Model 5")) +
  geom_line(data=Prob_CAD,aes(CAD_Weight,Prob_3, col = "Model 3" )) +
  geom_line(data=Prob_CAD,aes(CAD_Weight,Prob_5, col = "Model 5")) +
  geom_point(data=Prob_CAD,aes(CAD_Weight,Prob_8, col = "Model 8")) +
  labs(subtitle="Event: More than 10% extra weight is needed compared with CAD_Weight",
       y="Probability",
       x="CAD_Weight",
       title="Probabilities for the Event for each Model",
       caption = "Note: Probabilities for Model 8 shown as Scatter plot only")
```




```r
#Brier Score Function
score_brier <- function (z, probF){
  (z - probF)^2
}
```

```r
# if Actual_Weight < 1.1*CAD_Weight then z=1 otherwise z=0
indicator<- ifelse(data_test$Actual_Weight>data_test$CAD_Weight*1.1,1,0)

BS_3=c((score_brier(indicator,Prob_3)))
BS_5=c((score_brier(indicator,Prob_5)))
BS_8=c((score_brier(indicator,Prob_8)))

df.brier=data.frame("Brier Score for Model 3"=BS_3,"Brier Score for Model 5"=BS_5,"Brier Score for Model

B5_B3 = BS_5-BS_3
B8_B3 = BS_8-BS_3
B8_B5 = BS_8-BS_5

plot(ecdf(B8_B3),
     xlim=c(-0.2,0.2),
      xlab="Score Difference (Score 8 - Score 3)",
      ylab="% Test Data Set",
      main="ECDF of Brier Score Difference between Model 8, 5 & 3",
      col="red",cex=0)
lines(ecdf(B5_B3),
     col="blue",cex=0)
lines(ecdf(B8_B5),
     col="black",cex=0)
abline(v=0, col="magenta")
legend('bottomright',
       legend=c("Model 8 - Model 3","Model 5 - Model 3","Model 8 - Model 5"),
       col=c("red","blue","black"),
       pch=15)
```



Comparing Model 8, Model 5 & Model 3

The Brier score for Model 8 is better than Model 5 on appoximately **50%** of the test data set. The Brier score for Model 8 is better than Model 3 on approximately **45%** of the test data set. The Brier score for Model 5 is better than Model 3 on approximately **40%** of the test data set.

The Brier score does not give a conclusive indication on which model is better.

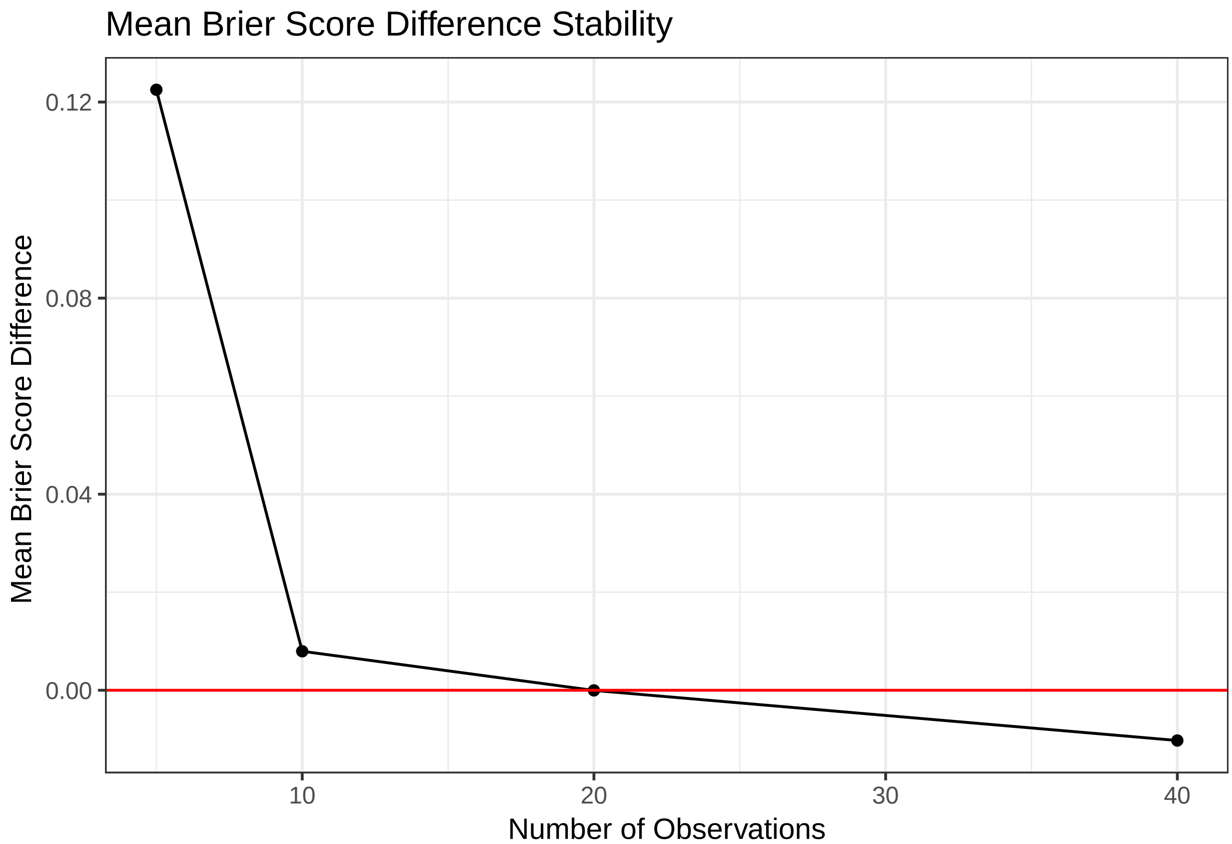
Task 10

```
#set.seed(pi) #for reproducability

# Generating observed data
c_5=rcauchy(5,location=2,scale=5)
c_10=rcauchy(10,location=2,scale=5)
c_20=rcauchy(20,location=2,scale=5)
c_40=rcauchy(40,location=2,scale=5)

# Generating test data
c_test=rcauchy(100000,location=2,scale=5)
```

```
neg_lik_cauchy <- function(theta, y) {
  -sum(dcauchy(
    y,
```

```r
    location = theta[1],
    scale = exp(theta[2])^0.5, #transforming scale
    log = TRUE))
}


opt_cauchy <- function (C_N){
  opt_c <- optim(c(0,0),
              fn = neg_lik_cauchy,
              y = C_N,
              method = "BFGS",
              control = list(maxit = 5000), # Anouncement on Learn, Ensures Convergence
              hessian = TRUE)

  location_e <- opt_c$par[1]
  scale_e <- opt_c$par[2]

  return (list(location_e=location_e,scale_e=scale_e, convergence=opt_c$convergence))
}

brier_cauchy <- function (C_N){
  opt_c <- opt_cauchy(C_N)

  #Estimated Probability Distribution
  prob_dist_e <- pcauchy(q=0,location = opt_c$location_e, scale=opt_c$scale_e,lower.tail = TRUE)
  #True Probability Distribution
  prob_dist_true = pcauchy(q=0,location = 2, scale=5,lower.tail = TRUE)


  indicator_e<- ifelse(c_test<0,1,0) #if y<0 then 1 else 0
  indicator_true<- ifelse(C_N<0,1,0)

  BS_10_e=c((score_brier(indicator_e,prob_dist_e))) #using  score_brier from Task 9
  BS_10_true=c((score_brier(indicator_true,prob_dist_true)))

  mean_BS_10_e=mean(BS_10_e) #Estimated Mean Brier Score
  mean_BS_10_true=mean(BS_10_true) #True Mean Brier Score

  mean_bs_d=mean_BS_10_e - mean_BS_10_true #Difference between scores (Estimated - True)

  return(list(mean_bs_e=mean_BS_10_e, mean_bs_true=mean_BS_10_true, mean_d=mean_bs_d,opt_c=opt_c))
}


b_5=brier_cauchy(c_5)
b_10=brier_cauchy(c_10)
b_20=brier_cauchy(c_20)
b_40=brier_cauchy(c_40)

mbs_n=data.frame(cbind(mean_d=c(b_5$mean_d,b_10$mean_d,b_20$mean_d,b_40$mean_d),N=c(5,10,20,40)))

ggplot()+
  geom_line(data=mbs_n,aes(N,mean_d))+
```

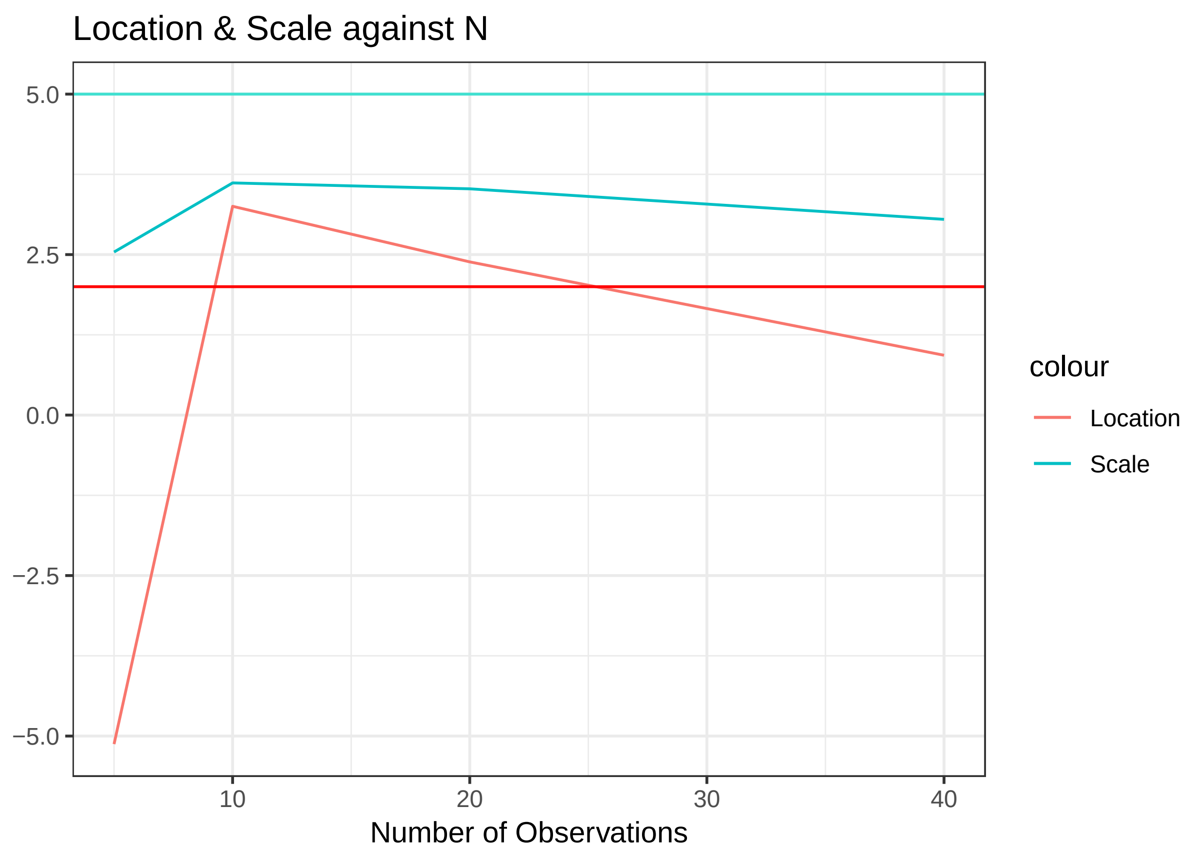
```r
  geom_point(data=mbs_n,aes(N,mean_d))+
  geom_hline(yintercept = 0, col="red")+
  labs(y="Mean Brier Score Difference ",
       x="Number of Observations",
       title="Mean Brier Score Difference Stability")
```



Stabilisation of Mean Brier Score Differences

The mean Brier Score difference stabilise as N increases. The difference approaches 0 as N increases.

```r
param_n=data.frame(cbind(opt_par_l=c(b_5$opt_c$location_e,b_10$opt_c$location_e,b_20$opt_c$location_e,b_

ggplot()+
  geom_line(data=param_n,aes(N,opt_par_l,col="Location"))+
  geom_line(data=param_n,aes(N,opt_par_s,col="Scale"))+
  geom_hline(yintercept = 2, col="red")+
  geom_hline(yintercept = 5, col="turquoise")+
  labs(y="",
       x="Number of Observations",
       title="Location & Scale against N")
```



Stabilisation of Optimization Paramaters

The optimization paramaters stabilise as N increases. They approach their respective true values as N increases.

Similar Comparison for Squared Error and Dawid Sebastiani Score

The squared error and dawid sebastiani scores will also stabalise as using a larger N results in better estimation of mu and sigma. Better estimates of mu and sigma result in better scores for squared error and dawid sebastiani.

Also since,

$$p_F = \mathrm{E}_F(z)$$

If the Brier Score stabalises then so will squared error score.

```
# Attempted Task 10 to run 25 times and then average


# run = function (rn){
#   Num <- c(5,10,20,40)
#
#   for (n in Num) {
#     sum_n=0
#     sum_l=0
#     sum_s=0
#     count=0
#     for (i in 1:rn) {
```

```
#       c_n=rcauchy(n,location=2,scale=5)
#       b_n=brier_cauchy(c_n)
#       sum_n=sum_n+b_n$mean_d
#       sum_l=sum_l+b_n$opt_c$location_e
#       sum_s=sum_s+b_n$opt_c$scale_e
#       count=count+1
#   }
#   print(c(sum_n/count,sum_l/count,sum_s/count))
#   }
# }
# run(25)
```