R SCRIPT

1 STUDY 1.1 – MONOTONIC LOAD TEST

1.1 NS-ISO 6891 (1991)

```
# Study 1.1 Monotonic load test - Follows NS-ISO 6891 (1991) calculations.
# Graphs, calculations and results.

#PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("RColorBrewer")

# DATA SET INFO
specimen_name <- "HNT"
study <- "Study 1.1 MONO"
file_type <- ".txt"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA
          ANALYZING\\", study ,"\\", specimen_name, sep=""),
          pattern =(paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))

all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(file_type,"",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  dat <- read.table(myFiles[i],sep="",dec=",",header=TRUE)
  result["max load(N)",i] <- max(dat$`Kraft`)
  row1 <- which.max(dat$`Kraft`)
  result["max displ(mm)",i] <- dat[row1,1]
  all_data[[i]] <- dat

}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(file_type,"",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=7,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#-----------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,"s",sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[name_specimen])
  specimen$lopende_kraft<-NA
```

```
  for (p in 5:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<-
          mean(specimen[intervall[1]:intervall[2],2])
  }

  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- specimen[1]
  my_data[2] <- specimen[3]
  colnames(my_data)<-c("Position","Force (smoothed)")
  plot(-my_data$Position,my_data$`Force (smoothed)`,type = "l",
      main = main_title, xlab = "Displacement (mm)",ylab = "Force (N)")

  # --------------------------------------------------------------------
# FAILURE IN GRAPH
  # By calculating the diff of each datapoint, verifying a large outlier
      and calculating the percentage of change in the drop - big drop
    yields failure point.
  differ <- diff(my_data$`Force (smoothed)`)/diff(my_data$Position)
  differ2 <- differ[-1] # exluding the first value, because it is infinite.
  rowdiffer <- which.max(differ2)
  # subset
  x <- my_data[(rowdiffer-20):(rowdiffer+20),]
  xmax <- which.max(x$`Force (smoothed)`)
  xmin <- which.min(x$`Force (smoothed)`)
  V1 <- x[xmax,2]
  V2 <- x[xmin,2]
  diff_perce <- abs(V1-V2)/((V1+V2)/2)*100
  # if the difference percentage is bigger than 8% and below 15 mm
      displacement, we classify the drop as a failure and the “maximum”
      load.
  if (diff_perce > 8 & my_data[rowdiffer,1] > -15){
    F_failure <- V1
    x_failure <- my_data[rowdiffer,1]
    abline (h=F_failure,col="blue", lty = 2, lwd = 1, pch = 3,
          lend = 0, ljoin = 2)
    abline (v=-x_failure,col="blue", lty = 2, lwd = 1, pch = 3,
          lend = 0, ljoin = 2)
    result["F_failure(N)",i] <- F_failure
    result["Displ_failure(mm)",i] <- x_failure
    F_maxAdj <- F_failure
    F_maxDispl <- x_failure
  } else { # identifying 15 mm slip
      sub_fmax <- subset(my_data,
                my_data$Position >= -15) # remember that my_data$position
                values are negative
      row_fmax <- which.max(sub_fmax$`Force (smoothed)`)
      F_max15 <- sub_fmax$`Force (smoothed)`[row_fmax]
      v_max15 <- sub_fmax$Position[row_fmax]
      result["F_max15",i] <- F_max15
      result["v_max15",i] <- v_max15
      result["diff_F_Fmax15",i] <- result[1,i]-F_max15
      result["percentage diff(%)",i] <- (1- (F_max15/result[1,i]))*100
      abline (h=F_max15,col="blue", lty = 2, lwd = 1, pch = 3,
          lend = 0, ljoin = 2)
      abline (v=-v_max15,col="blue", lty = 2, lwd = 1, pch = 3,
          lend = 0, ljoin = 2)
      F_maxAdj <- F_max15
      F_maxDispl <- v_max15
```

```
}

#--------------------------------------------------------------------------
--
# CALCULATING
# Since the preload test is missing, we retrive k_ser as the ratio
  between yield load and yield slip.
# Need firstly to find the yield values. Follows the 10-40-90 procedure
  AKA Yasumura and Kawai procedure.

Force_perc <- c(0.1*F_maxAdj,0.4*F_maxAdj,0.9*F_maxAdj)
coord <- as.data.frame(matrix(nrow=0,ncol=3))
colnames(coord)<- c("F10","F40","F90")

# subset at 15 mm slip, which gives 90% F_max after the slip point to be
  excluded
dat2 <- subset(my_data,my_data$Position >= -15,
        select = c("Position","Force (smoothed)"))

# function drawing line
segmentInf <- function(xs, ys){
  fit <- lm(ys~xs)
  abline(fit, col="gray",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)
}

line1 <- as.data.frame(matrix(nrow=0,ncol=6))
colnames(line1)<- c("Intercept","Slope","x1","x2","y1","y2")
for (n in 1:length(Force_perc)){
  rowY <- which.min(abs(dat2$`Force (smoothed)`-Force_perc[n]))
  x_coord <- dat2[rowY,1]
  y_coord <- dat2[rowY,2]
  coord["x_coord",n]<- x_coord
  coord["y_coord",n]<- y_coord
  if (n==2|n==3) { # Starts drawing lines when n=2
    x_line <- c(-coord[1,n-1],-coord[1,n])
    y_line <- c(coord[2,n-1],coord[2,n])
    segmentInf(x_line,y_line)
    if (n==2) {
      line1[1,"Intercept"]<-coef(lm(y_line~x_line))[[1]]
      line1[1,"Slope"]<-coef(lm(y_line~x_line))[[2]]
      line1[1,"x1"]<-x_line[1]
      line1[1,"x2"]<-x_line[2]
      line1[1,"y1"]<-y_line[1]
      line1[1,"y2"]<-y_line[2]
    }
  }
}

line2_slope <- coef(lm(y_line~x_line))[[2]] # the slope of line going
               through 40% and 90% F_max
# finding the slope in graph that is similar to line2_slope
row3 <- which(dat2$`Force (smoothed)`==y_coord)
excl2 <- dat2[row3,2]
dat3 <- subset(dat2,dat2$`Force (smoothed)` < excl2)

slope_data <- as.data.frame(matrix(nrow=0,ncol=6))
colnames(slope_data) <- c("Intersept","Slope", "x1","x2","y1","y2" )
for (m in 100:nrow(dat3)){
  x_graph <- c(-dat3[m-99,1],-dat3[m,1])
  y_graph <- c(dat3[m-99,2],dat3[m,2])
  coef_inter <- coef(lm(y_graph~x_graph))[1]
```

```
    coef_slope <- coef(lm(y_graph~x_graph))[2]
    slope_data[m,1] <- coef_inter
    slope_data[m,2] <- coef_slope
    slope_data[m,3] <- x_graph[1]
    slope_data[m,4] <- x_graph[2]
    slope_data[m,5] <- y_graph[1]
    slope_data[m,6] <- y_graph[2]
  }
  row_y <- which.min(abs(slope_data[,2]-line2_slope))
  line3 <- slope_data[row_y,]
  x2_coord <- c(slope_data[row_y,3],slope_data[row_y,4])
  y2_coord <- c(slope_data[row_y,5],slope_data[row_y,6])
  segmentInf(x2_coord,y2_coord)

  #----------------------------------------------------------------------
--
  # FINDING F_y, v_y and k_ser

  # Need to find the intersection between line1 and line3
  # Algebraically calculated using intersection and slope from line1 and
3
  # the yield point is horizontally projected to the curve from this
    intersection point.
  xx <- (line3[1,1]-line1[1,1])/(line1[1,2]-line3[1,2])
  yy <- line1[1,1]+line1[1,2]*xx

  # Finding the projected F_y value
  rowyy <- which.min(abs(sub_fmax$`Force (smoothed)`- yy))
  F_y <- sub_fmax$`Force (smoothed)`[rowyy]
  # yield slip is the cooresponding displacement value from F_y
  v_y <- sub_fmax$Position[rowyy]

  abline (h=yy,col="darkgray", lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  abline (v=xx,col="darkgray", lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  points(x = -v_y,y = F_y, col = "red")

  result["v_y",i] <- v_y
  result["F_y",i] <- F_y

  k_ser <- F_y/v_y
  result["k_ser",i]<- k_ser

  # ---------------------------------------------------------------------
--
  # FINAL RESULT FOR TERMOWOOD REPORT
  # Constructing a final table with results
  Final_results[i,"Maximum Force (kN)"] <- F_maxAdj/1000
  Final_results[i,"Displacement at max force (mm)"] <- -F_maxDispl
  Final_results[i,"Yield slip (mm)"] <- v_y
  Final_results[i,"Yield load (kN)"] <- F_y/1000
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- k_ser/1000

}

# -----------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")
```

```
for (o in 1:length(Final_results)) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
               sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# SAVING TABLES AS .TXT FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results.xlsx",
          row.names = T, col.names = T)
```

1.2 NS-EN 12512 (2002)

```
# Study 1.1 Monotonic load test - Follows NS-EN 12512 (2002) calculations.
# Graphs, calculations and results.

# PACKAGES
# install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "HNT"
study <- "Study 1.1 MONO"
file_type <- ".txt"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA
          ANALYZING\\", study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))

all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(file_type,"",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  dat <- read.table(myFiles[i],sep="",dec=",",header=TRUE)
  result["max load(N)",i] <- max(dat$`Kraft`)
  row1 <- which.max(dat$`Kraft`)
  result["max displ(mm)",i] <- dat[row1,1]
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(file_type,"",myFiles)
names(all_data)<-myFiles
```

```
Final_results <- as.data.frame(matrix(nrow=length(myFiles)+2,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#-----------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,"s",sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[name_specimen])
  specimen$lopende_kraft<-NA
  for (p in 5:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<
          mean(specimen[intervall[1]:intervall[2],2])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- -specimen[1] # changes negative values to positive
  my_data[2] <- specimen[3] # force
  colnames(my_data)<-c("Position","Force (smoothed)")
  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
    main = main_title, xlab = "Displacement (mm)",ylab = "Force (N)")

  #---------------------------------------------------------------------
--
  # MAX LOAD
  row_Fmax <- which.max(my_data$`Force (smoothed)`)
  F_max <- my_data$`Force (smoothed)`[row_Fmax]
  V_Fmax <- my_data$Position[row_Fmax]

  # ULTIMATE LOAD CASES:
# Vu = min{failure,80%Fmax,30mm}
  # Identifying failure with locator(). ESC key if there are none.
  failure_coord <- as.data.frame(matrix(nrow = length(myFiles),ncol = 2))
  rownames(failure_coord) <- myFiles
  colnames(failure_coord) <- c("x","y")
  failure_locator <- locator(1)
  if (!is.null(failure_locator)) {
    failure_coord[name_specimen,"x"] <- failure_locator$x
    failure_coord[name_specimen,"y"] <- failure_locator$y
  } else {
    failure_locator$x <- NA
    failure_locator$y <- NA
  }
  result["Failure Load (N)",i] <- failure_locator$y
  result["Failure displacement (mm)",i] <- failure_locator$x

  # 80%Fmax
  F_80 <- 0.8*F_max
  subs <- subset(my_data,my_data$Position > V_Fmax)
  row <- which.min(abs(subs$Force - F_80))
  V_F80 <- subs$Position[row]
  result["80% Max Load (N)",i] <- F_80
  result["Displ at 80% Max Load (mm)",i] <- V_F80

  # Slip at 30 mm
  row_30 <- which.min(abs(my_data$Position - 30))
  F_30mm <- my_data$`Force (smoothed)`[row_30]
  V_30mm <- my_data$Position[row_30]
```

```
  result["Force Load at 30 mm (N)",i] <- F_30mm
  result["displ 30mm (mm)",i] <- V_30mm

  # FINAL ULTIMATE LOAD:
  # the displacement that occurs first.
  Displ_Values <- c(failure_locator$x, V_F80, V_30mm)
  Load_Values <- c(failure_locator$y, F_80,F_30mm)
  minimum <- which.min(Displ_Values)
  Ultimate_displ <- Displ_Values[minimum]
  Ultimate_force <- Load_Values[minimum]

  result["Ultimate Force (N)",i] <- Ultimate_force
  result["Ultimate Displ (mm)",i] <- Ultimate_displ

  # plotting ultimate displ
  abline (v=Ultimate_displ,col="grey", lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  text(Ultimate_displ+0.1,Ultimate_force,labels = "Vu",adj = c(0,0),
    cex = 0.8, col = "grey")

  #----------------------------------------------------------------------
--
  # CALCULATING THE YIELD LOAD/SLIP
  # Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
  xsub <- subset(my_data,my_data$Position < V_Fmax)
  row3 <- which.min(abs(xsub$Force - (0.1*F_max)))
  V10 <- xsub[row3,1]
  F10 <- xsub[row3,2]
  row4 <- which.min(abs(xsub$Force - (0.4*F_max)))
  V40 <- xsub[row4,1]
  F40 <- xsub[row4,2]
  xcoord <- c(V10,V40)
  ycoord <- c(F10,F40)
  # plotting points and slope in the graph
  points(x = V10, y = F10,col="Red")
  points(x = V40, y=F40, col="Red")
  fit <- lm(ycoord~xcoord)
  abline(fit, col="black",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)

  # The intercept and slope for line1 is then found and so are the elastic
    stiffness which is equal to the slope.
  b <- coef(lm(ycoord~xcoord))[1] #intercept
  a <- coef(lm(ycoord~xcoord))[2] #slope
  Vu <- Ultimate_displ

  # YIELD LOAD
  # The yield load is the interception between line1 and a second line2.
# Line 2 angle is 1/6 of line1s' slope and is touching the graph.
  # Firstly, we find the slope of line2 and secondly, we found the point
in
    the graph that has the same slope.
  line2_slope <- (1/6)*a
  # finding the slope in graph that is similar to line2_slope
  slope_data <- as.data.frame(matrix(nrow=0,ncol=6))
  colnames(slope_data) <- c("Intersept","Slope", "x1","x2","y1","y2" )
  sub_slope <- subset(my_data,
                my_data$Position > V40 & my_data$Position < V_Fmax)
  for (m in 50:nrow(sub_slope)){# calculating slope with a step of 50
    vertices, because the rawdata alows it.
    x_graph <- c(sub_slope[m-49,1],sub_slope[m,1])
    y_graph <- c(sub_slope[m-49,2],sub_slope[m,2])
```

```
    coef_inter <- coef(lm(y_graph~x_graph))[1]
    coef_slope <- coef(lm(y_graph~x_graph))[2]
    slope_data[m,1] <- coef_inter
    slope_data[m,2] <- coef_slope
    slope_data[m,3] <- x_graph[1]
    slope_data[m,4] <- x_graph[2]
    slope_data[m,5] <- y_graph[1]
    slope_data[m,6] <- y_graph[2]
  }
  row_y <- which.min(abs(slope_data[,2]-line2_slope))
  line2_data <- slope_data[row_y,]
  x2_coord <- c(slope_data[row_y,3],slope_data[row_y,4])
  y2_coord <- c(slope_data[row_y,5],slope_data[row_y,6])
  line2_fit <- lm(y2_coord~x2_coord)
  abline(line2_fit, col="gray",lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)

  # FINDING F_y, v_y and k_ser
  # Need to find the intersection between line1 and line2
  # Algebraically calculated using intersection and slope from line1 and
2
  V_y <- (line2_data[1,1]-b)/(a-line2_data[1,2])
  F_y <- b+a*V_y
  result["V_y",i] <- V_y
  result["F_y",i] <- F_y

  abline (h=F_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  abline (v=V_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)

  k_ser <- F_y/V_y
  result["k_ser",i]<- k_ser


  # calculating ductility
  D <- Ultimate_displ/V_y

  # ---------------------------------------------------------------------
--
  # FINAL RESULT
  # Constructing a final table with results
  Final_results[i,"maximum load (kN)"] <- F_max/1000
  Final_results[i,"Ultimate displacement (mm)"] <- Ultimate_displ
  Final_results[i,"Ultimate Load (kN)"] <- Ultimate_force/1000
  Final_results[i,"Yield Load (kN)"] <- F_y/1000
  Final_results[i,"Yield slip (mm)"] <- V_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- k_ser/1000
  Final_results[i,"Ductility"] <- D

}


# -----------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations

for (o in 1:length(Final_results)) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
```

```
          sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)


# SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - EN12512(2002).xlsx",
row.names = T, col.names = T)
```

1.3 EN 12512 (2018)

```
# Study 1.1 Monotonic load test
# EN 12512 (2018) Draft Proposal version 20180410

# PACKAGES
# install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "HNT"
study <- "Study 1.1 MONO"
file_type <- ".txt"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA ANALYZING\\",
          study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))

all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(file_type,"",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  dat <- read.table(myFiles[i],sep="",dec=",",header=TRUE)
  result["max load(N)",i] <- max(dat$`Kraft`)
  row1 <- which.max(dat$`Kraft`)
  result["max displ(mm)",i] <- dat[row1,1]
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(file_type,"",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=length(myFiles)+2,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")
```

```
#---------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,"s",sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[name_specimen])
  specimen$lopende_kraft<-NA
  for (p in 5:(nrow(specimen)-1)){

    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<
          mean(specimen[intervall[1]:intervall[2],2])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- -specimen[1] # changes negative values to positive
  my_data[2] <- specimen[3] # force
  colnames(my_data)<-c("Position","Force (smoothed)")
  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
    main = main_title, xlab = "Displacement (mm)",ylab = "Force (N)")

  #-------------------------------------------------------------------------
--
  # CALCULATIONS
  # Peak load, Pl - max load reached during monotonic test
  row_Pl <- which.max(my_data$`Force (smoothed)`)
  Pl <- my_data$`Force (smoothed)`[row_Pl]
  V_Pl <- my_data$Position[row_Pl]

  # ULTIMATE LOAD CASES:
  # Vu - min{failure,80%Pl,30mm}
  # Identifying failure with locator(). ESC key if there are none.
  failure_coord <- as.data.frame(matrix(nrow = length(myFiles),ncol = 2))
  rownames(failure_coord) <- myFiles
  colnames(failure_coord) <- c("x","y")
  failure_locator <- locator(1)
  if (!is.null(failure_locator)) {
    failure_coord[name_specimen,"x"] <- failure_locator$x
    failure_coord[name_specimen,"y"] <- failure_locator$y
  } else {
    failure_locator$x <- NA
    failure_locator$y <- NA
  }
  result["Failure Load (N)",i] <- failure_locator$y
  result["Failure displacement (mm)",i] <- failure_locator$x

  # 80%Plmax - Load/displacement after peak load.
  Pl_80 <- 0.8*Pl
  subs <- subset(my_data,my_data$Position > V_Pl)
  row <- which.min(abs(subs$Force - Pl_80))
  V_Pl80 <- subs$Position[row]
  result["80% Peak Load (N)",i] <- Pl_80
  result["Displ at 80% Peak Load (mm)",i] <- V_Pl80

  # Slip at 30 mm
  row_30 <- which.min(abs(my_data$Position - 30))
  F_30mm <- my_data$`Force (smoothed)`[row_30]
  V_30mm <- my_data$Position[row_30]
  result["Force Load at 30 mm (N)",i] <- F_30mm
```

```
result["displ 30mm (mm)",i] <- V_30mm

# FINAL ULTIMATE LOAD
# the displacement that occurs first.
Displ_Values <- c(failure_locator$x, V_Pl80, V_30mm)
Load_Values <- c(failure_locator$y, Pl_80,F_30mm)
minimum <- which.min(Displ_Values)
Ultimate_displ <- Displ_Values[minimum]
Ultimate_force <- Load_Values[minimum]

result["Ultimate Force (N)",i] <- Ultimate_force
result["Ultimate Displ (mm)",i] <- Ultimate_displ

# MAX LOAD
# max load lower than or equal to ultimate displacement
sub_Fmax <- subset(my_data,my_data$Position <= Ultimate_displ)
rowmax <- which.max(sub_Fmax$Force)
F_max <- sub_Fmax[rowmax,2]

# plotting ultimate displ
abline (v=Ultimate_displ,col="grey", lty = 2, lwd = 1, pch = 3,
  lend = 0, ljoin = 2)
text(Ultimate_displ+0.1,Ultimate_force,labels = "Vu",adj = c(0,0),
     cex = 0.8, col = "grey")

#----------------------------------------------------------------------
--
# CALCULATING THE EEEP CURVE
# First, we retrieve the area under the curve.
# AUC is the area under the curve with boundaries from origin to ultimate
  displacement
xsub <- subset(my_data,my_data$Position <= Ultimate_displ)
x <- xsub$Position
y <- xsub$Force
id <- order(x)
AUC <- sum(diff(x[id])*rollmean(y[id],2))

# Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
row3 <- which.min(abs(xsub$Force - (0.1*F_max)))
V10 <- xsub[row3,1]
F10 <- xsub[row3,2]
row4 <- which.min(abs(xsub$Force - (0.4*F_max)))
V40 <- xsub[row4,1]
F40 <- xsub[row4,2]
xcoord <- c(V10,V40)
ycoord <- c(F10,F40)
# plotting points and slope in LEC graph
points(x = V10, y = F10,col="Red")
points(x = V40, y=F40, col="Red")
fit <- lm(ycoord~xcoord)
abline(fit, col="black",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)

# The intercept and slope for line1 is then found and so are the elastic
  stiffness which is equal to the slope.
b <- coef(lm(ycoord~xcoord))[1] #intercept
a <- coef(lm(ycoord~xcoord))[2] #slope
K <- a[[1]] # Elastic stiffness [N/mm]
Vu <- xsub[nrow(xsub),1] # ultimate displacement retrieved from subset
of
   the graph
```

```r
  # the equation to find the horizontal line (line2) that gives an area
    equal to the LEC1 area is a quadratic equation.
  # The y-solution to the line is then given as plus and minus, referring
    to the quadratic formula:

  mMinus = a[[1]]*((Vu+b[[1]]/a[[1]]) -
           sqrt((Vu+b[[1]]/a[[1]])^2 - 4*((b[[1]]/(4*a[[1]]))^2 +
           AUC/(2*a[[1]])))) #with x=0 as initial boundary

  # calculating the intersection between line1 and line2.
  # the intersection is the yield load and displacement.
  F_ymMinus <- mMinus
  v_ymMinus <- (mMinus - b[[1]])/a[[1]]
  abline(h = F_ymMinus, col="grey",lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  abline(v = v_ymMinus, col="grey",lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  result["v_y",i] <- v_ymMinus
  result["F_y",i] <- F_ymMinus
  v_y <- v_ymMinus
  F_y <- F_ymMinus

  #---------------------------------------------------------------------
--
  # calculating ductility
  D <- Ultimate_displ/v_y


  # --------------------------------------------------------------------
--
  # FINAL RESULT
  # Constructing a final table with results
  Final_results[i,"Peak Load (kN)"] <- Pl/1000
  Final_results[i,"maximum load (kN)"] <- F_max/1000
  Final_results[i,"Ultimate displacement (mm)"] <- Ultimate_displ
  Final_results[i,"Ultimate Load (kN)"] <- Ultimate_force/1000
  Final_results[i,"Yield Load (kN)"] <- F_y/1000
  Final_results[i,"Yield slip (mm)"] <- v_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- K/1000
  Final_results[i,"Ductility"] <- D
}


# ----------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations

for (o in 1:length(Final_results)) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
          sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - EN12512(2018).xlsx",
row.names = T, col.names = T)
```

2 STUDY 1.2 – CYCLIC LOAD TEST

2.1 NS-EN 12512 (2002)

```
# Study 1.2 Cylic load test - Follows NS-EN 12512 (2002) calculations.
# Graphs, calculations and results.

# PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
# install.packages("zoo")
library("zoo")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")
library("xlsx", lib.loc="~/R/win-library/3.5")

# DATA SET INFO
specimen_name <- "HNSc"
study <- "STUDY 1.2 CYCLIC"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA
          ANALYZING\\", study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))

all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(".csv","",myFiles2)

result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2
Final_resultsCOMP                                                       <-
as.data.frame(matrix(nrow=length(myFiles2)+2,ncol=0))
Final_resultsTENS                                                       <-
as.data.frame(matrix(nrow=length(myFiles2)+2,ncol=0))
Final_resultsSPEC                                                       <-
as.data.frame(matrix(nrow=length(myFiles2)+2,ncol=0))
rownames(Final_resultsCOMP)<-c(myFiles2,"Mean Values",
          "Standard Deviations")
rownames(Final_resultsTENS)<-c(myFiles2,"Mean Values",
          "Standard Deviations")
rownames(Final_resultsSPEC)<-c(myFiles2,"Mean Values",
          "Standard Deviations")
beta <- intToUtf8(0x03B2L)
Delta <- intToUtf8(916)
v_eq_values <- c()

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], nlines = 1, what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=1, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=1,sep=";",dec=",",header=TRUE)
```

```r
  colnames(dat) <- paste(header,header2,sep="")
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles

#------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,
          " According to EN12512(2002)",sep = "")
  specimen <- as.data.frame(all_data[[i]])
  specimen$lopende_kraft<-NA
  for (p in 10:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-10,linje+10)
    specimen$lopende_kraft[linje]<
          mean(specimen[intervall[1]:intervall[2],3])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=3))
  my_data <- specimen[1]
  my_data[2] <- specimen[2]
  my_data[3] <- specimen[4]
  colnames(my_data)<-c("Time", "Position","Force (smoothed)")

  # --------------------------------------------------------------------
--
  # ENVELOPE CURVE (LEC)
  # COMPRESSION (negative values)
  # Find maximum for each subset and construct the envelope curve for
    compressive load
  Csubplot_data <- c()
  Cset1_3 <- subset(my_data,my_data$Time < 120 & my_data$Position < 0)
    # includes 1st,2nd and 3rd cyclesets.
  Csubplot_data[["Cset1_3"]]<- Cset1_3

  Cset1V <- subset(my_data,my_data$Time > 120 & my_data$Time < 240
          & my_data$Position < 0 ) # 4th cycleset
  Csubplot_data[["Cset1V"]]<- Cset1V

  Cset2V <- subset(my_data,my_data$Time > 240 & my_data$Time < 480
          & my_data$Position < 0) # 5th cycleset
  Csubplot_data[["Cset2V"]]<- Cset2V

  Cset4V <- subset(my_data,my_data$Time > 480 & my_data$Time < 960
          & my_data$Position < 0) # 6th cycleset
  Csubplot_data[["Cset4V"]]<- Cset4V

  Cset6V <- subset(my_data,my_data$Time > 960 & my_data$Time < 1680
          & my_data$Position < 0) # 7th cycleset
  Csubplot_data[["Cset6V"]]<- Cset6V

  Cset8V <- subset(my_data,my_data$Time > 1680 & my_data$Time < 2640
          & my_data$Position < 0) # 8th cycleset
  Csubplot_data[["Cset8V"]]<- Cset8V
```

```
  Cset10V <- subset(my_data,my_data$Time > 2640 & my_data$Time < 3840
          & my_data$Position < 0) # 9th cycleset
  Csubplot_data[["Cset10V"]]<- Cset10V

  Cset12V <- subset(my_data,my_data$Time > 3840 & my_data$Time < 5280
          & my_data$Position < 0) # 10th cycleset
  if (any(Cset12V$Position < -23.90)) { # if-statement to check if the
last
    cycle is complete
    Csubplot_data[["Cset12V"]] <- Cset12V
  }

  Cset14V <- subset(my_data,my_data$Time > 5280 & my_data$Time < 6960
          & my_data$Position < 0) # 11th cycleset
  if (any(Cset14V$Position < -27.90)) { # if-statement to check if the
last
    cycle is complete
    Csubplot_data[["Cset14V"]] <- Cset14V
  }

  # Constructing load envelope curve (LEC) for the first load curve in
each
    cycle
  LEC1_COMP <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
    fill with max values
  colnames(LEC1_COMP) <- c("Position","Force")

  # Points in 1st, 2nd and 3rd cycleset
  LEC1_COMP["zero point",1:2] <- c(0,0)

  points1 <- subset(Cset1_3,Cset1_3$Position > -0.1)
  row <- which.min(points1$`Force (smoothed)`)
  point1 <- points1[row,]
  LEC1_COMP["initial point",1:2] <- point1[1,2:3]

  points2 <- subset(Cset1_3,Cset1_3$Position > -0.55
          & Cset1_3$Position < -0.45)
  row <- which.min(points2$`Force (smoothed)`)
  point2 <- points2[row,]
  LEC1_COMP["-0.5 mm",1:2] <- point2[1,2:3]

  points3 <- subset(Cset1_3,Cset1_3$Position > -1.05
          & Cset1_3$Position < -0.95)
  row <- which.min(points3$`Force (smoothed)`)
  point3 <- points3[row,]
  LEC1_COMP["-1.0 mm",1:2] <- point3[1,2:3]
  LEC1_COMP["-1.5 mm",1:2] <-
                Cset1_3[which.min(Cset1_3$`Force (smoothed)`),2:3]

  # Points in 4th to 11th cycleset 1st LEC
  for (n in 2:length(Csubplot_data)) {
    displ_COMP <- -c(0,1,seq(2,14,2))*2
    subpoints <- subset(Csubplot_data[[n]],
                        Csubplot_data[[n]][2] > displ_COMP[n]-0.05
                      & Csubplot_data[[n]][2] < displ_COMP[n]+0.05,
                    select = c("Position","Force (smoothed)"))
 #subsetting for position
    rowmin <- which.min(subpoints$`Force (smoothed)`)
    name_row <- paste(displ_COMP[n],"mm")
    LEC1_COMP[name_row,1:2] <- subpoints[rowmin,1:2]
  }
```

```
  # Constructing load envelope curve (LEC) for the third load curve in
each
    cycle
  LEC3_COMP <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
    fill with max values
  colnames(LEC3_COMP) <- c("Position","Force")

  LEC3_COMP <- LEC1_COMP[1:4,]
  # starts from 1.5 mm
  points1_5 <- subset(Cset1_3,Cset1_3$Position > -1.505
          & Cset1_3$Position < -1.495)
  row <- which.max(points1_5$`Force (smoothed)`)
  point1_5 <- points1_5[row,]
  LEC3_COMP["-1.5 mm",1:2] <- point1_5[1,2:3]

  # Points in 4th to 9th cycleset 3rd LEC
  for (n in 2:length(Csubplot_data)) {
    displ_COMP <- -c(0,1,seq(2,14,2))*2
    subpoints <- subset(Csubplot_data[[n]],
                Csubplot_data[[n]][2] > displ_COMP[n]-0.005
                  & Csubplot_data[[n]][2] < displ_COMP[n]+0.005,
                select = c("Position","Force (smoothed)"))
    rowmin <- which.max(subpoints$`Force (smoothed)`)
    name_row <- paste(displ_COMP[n],"mm")
    LEC3_COMP[name_row,1:2] <- subpoints[rowmin,1:2]
  }

  # TENSION (positive values)
  Tsubplot_data <- c()
  Tset1_3 <- subset(my_data,my_data$Time < 120 & my_data$Position > 0)
    # includes 1st,2nd and 3rd cyclesets.
  Tsubplot_data[["Tset1_3"]]<- Tset1_3

  Tset1V <- subset(my_data,my_data$Time > 120 & my_data$Time < 240
          & my_data$Position > 0 ) # 4th cycleset
  Tsubplot_data[["Tset1V"]]<- Tset1V

  Tset2V <- subset(my_data,my_data$Time > 240 & my_data$Time < 480
          & my_data$Position > 0) # 5th cycleset
  Tsubplot_data[["Tset2V"]]<- Tset2V

  Tset4V <- subset(my_data,my_data$Time > 480 & my_data$Time < 960
          & my_data$Position > 0) # 6th cycleset
  Tsubplot_data[["Tset4V"]]<- Tset4V

  Tset6V <- subset(my_data,my_data$Time > 960 & my_data$Time < 1680
          & my_data$Position > 0) # 7th cycleset
  Tsubplot_data[["Tset6V"]]<- Tset6V

  Tset8V <- subset(my_data,my_data$Time > 1680 & my_data$Time < 2640
          & my_data$Position > 0) # 8th cycleset
  Tsubplot_data[["Tset8V"]]<- Tset8V

  Tset10V <- subset(my_data,my_data$Time > 2640 & my_data$Time < 3840
          & my_data$Position > 0) # 9th cycleset
  Tsubplot_data[["Tset10V"]]<- Tset10V

  Tset12V <- subset(my_data,my_data$Time > 3840 & my_data$Time < 5280
          & my_data$Position > 0) # 10th cycleset
  if (any(Tset12V$Position > 23.90)) { # if-statement to check if the last
```

```
  cycle is complete
  Tsubplot_data[["Tset12V"]] <- Tset12V
}

Tset14V <- subset(my_data,my_data$Time > 5280 & my_data$Time < 6960
        & my_data$Position > 0) # 11th cycleset
if (any(Tset14V$Position > 27.90)) { # if-statement to check if the last
  cycle is complete
  Tsubplot_data[["Tset14V"]] <- Tset14V
}

# Constructing load envelope curve (LEC1) for the first load curve in
  each cycle
LEC1_TENS <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
  fill with max values
colnames(LEC1_TENS) <- c("Position","Force")

# Points in 1st, 2nd and 3rd cycleset
LEC1_TENS["zero point",1:2] <- c(0,0)

Tpoints1 <- subset(Tset1_3,Tset1_3$Position < 0.1)
row <- which.max(Tpoints1$`Force (smoothed)`)
Tpoint1 <- Tpoints1[row,]
LEC1_TENS["initial point",1:2] <- Tpoint1[1,2:3]

Tpoints2 <- subset(Tset1_3,Tset1_3$Position < 0.55
        & Tset1_3$Position > 0.45)
row <- which.max(Tpoints2$`Force (smoothed)`)
Tpoint2 <- Tpoints2[row,]
LEC1_TENS["0.5 mm",1:2] <- Tpoint2[1,2:3]

Tpoints3 <- subset(Tset1_3,Tset1_3$Position < 1.05
        & Tset1_3$Position > 0.95)
row <- which.max(Tpoints3$`Force (smoothed)`)
Tpoint3 <- Tpoints3[row,]
LEC1_TENS["1.0 mm",1:2] <- Tpoint3[1,2:3]
LEC1_TENS["1.5 mm",1:2] <-
        Tset1_3[which.max(Tset1_3$`Force (smoothed)`),2:3]

# Points in 4th to 11th cycleset 1st LEC
for (n in 2:length(Tsubplot_data)) {
  displ_TENS <- c(0,1,seq(2,14,2))*2
  Tsubpoints <- subset(Tsubplot_data[[n]],
              Tsubplot_data[[n]][2] > displ_TENS[n]-0.05
                & Tsubplot_data[[n]][2] < displ_TENS[n]+0.05,
              select = c("Position","Force (smoothed)"))
  rowmin <- which.max(Tsubpoints$`Force (smoothed)`)
  name_row <- paste(displ_TENS[n],"mm")
  LEC1_TENS[name_row,1:2] <- Tsubpoints[rowmin,1:2]
}

# Constructing load envelope curve (LEC3) for the third load curve in
  each cycle
LEC3_TENS <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
  fill with max values
colnames(LEC3_TENS) <- c("Position","Force")

LEC3_TENS <- LEC1_TENS[1:4,]
# starts from 1.5 mm
Tpoints1_5 <- subset(Tset1_3,Tset1_3$Position < 1.505
              & Tset1_3$Position > 1.495)
```

```r
  row <- which.min(Tpoints1_5$`Force (smoothed)`)
  Tpoint1_5 <- Tpoints1_5[row,]
  LEC3_TENS["1.5 mm",1:2] <- Tpoint1_5[1,2:3]

  # Points in 4th to 11th cycleset 3rd LEC
  for (n in 2:length(Tsubplot_data)) {
    displ_TENS <- c(0,1,seq(2,14,2))*2
    Tsubpoints <- subset(Tsubplot_data[[n]],
                Tsubplot_data[[n]][2] > displ_TENS[n]-0.005
                  & Tsubplot_data[[n]][2] < displ_TENS[n]+0.005,
                select = c("Position","Force (smoothed)"))
    rowmin <- which.min(Tsubpoints$`Force (smoothed)`)
    name_row <- paste(displ_TENS[n],"mm")
    LEC3_TENS[name_row,1:2] <- Tsubpoints[rowmin,1:2]
  }

  # interpolating LEC1 and LEC3
  LEC1_COMP <- as.data.frame(approx(LEC1_COMP$Position,
                LEC1_COMP$Force, method = "linear",
                n = 1000)) #interpolation
  LEC1_TENS <- as.data.frame(approx(LEC1_TENS$Position,
                LEC1_TENS$Force,method = "linear",
                n = 1000))
  colnames(LEC1_COMP) <- c("Position","Force")
  colnames(LEC1_TENS) <- c("Position","Force")
  LEC3_COMP <- as.data.frame(approx(LEC3_COMP$Position,
                LEC3_COMP$Force, method = "linear",
                n = 1000)) #interpolation
  LEC3_TENS <- as.data.frame(approx(LEC3_TENS$Position,
                LEC3_TENS$Force,method = "linear",
                n = 1000))
  names(LEC3_COMP) <- c("Position","Force")
  names(LEC3_TENS) <- c("Position","Force")

#-------------------------------------------------------------------------
--
  # CALCULATION | Ultimate load (Failure; 80% F_max, Delta_F):
  # Failure is not calculated due to no distinct failure-drop in these
    experiments LECs.

  # MAX LOAD IN LEC1, COMPRESSION
  row1 <- which.min(LEC1_COMP$Force)
  F_MAX_C <- LEC1_COMP$Force[row1]
  V_FMAX_COMP <- LEC1_COMP$Position[row1]

  # MAX LOAD IN LEC1, TENSION
  row2 <- which.max(LEC1_TENS$Force)
  F_MAX_T <- LEC1_TENS$Force[row2]
  V_FMAX_TENS <- LEC1_TENS$Position[row2]

  result["Maximum Load Compression (N)", i] <- F_MAX_C
  result["Displ at max Load compression (mm)",i] <- V_FMAX_COMP
  result["Maximum Load Tension (N)", i] <- F_MAX_T
  result["Displ at max Load Tension (mm)",i] <- V_FMAX_TENS

  # DISPLACEMENT 80% F_max after max load less than 30mm, COMPRESSION
  F_80_COMP <- 0.8*F_MAX_C
  subs <- subset(LEC1_COMP,LEC1_COMP$Position < V_FMAX_COMP)
  row <- which.min(abs(subs$Force - F_80_COMP))
  V_F80_COMP <- subs$Position[row]
  result["80% Max Load Compression (N)",i] <- F_80_COMP
```

```
  result["Displ at 80% Max Load Compression (mm)",i] <- V_F80_COMP

  # DISPLACEMENT 80% F_max after peak load less than 30 mm, TENSION
  F_80_TENS <- 0.8*F_MAX_T
  subs <- subset(LEC1_TENS,LEC1_TENS$Position > V_FMAX_TENS)
  row <- which.min(abs(subs$Force - F_80_TENS))
  V_F80_TENS <- subs$Position[row]
  result["80% Max Load  Tension (N)",i] <- F_80_TENS
  result["Displ at 80% Max Load Tension (mm)",i] <- V_F80_TENS

  # DISPLACEMENT AND LOAD AT 30 mm
  # COMPRESSION
  row <- which.min(abs(LEC1_COMP$Position + 30))
  V_30mm_C <- LEC1_COMP$Position[row]
  F_30mm_C <- LEC1_COMP$Force[row]
  # TENSION
  row <- which.min(abs(LEC1_TENS$Position - 30))
  V_30mm_T <- LEC1_TENS$Position[row]
  F_30mm_T <- LEC1_TENS$Force[row]

  # FINAL ULTIMATE LOAD
  # the displacement that occurs first.
  Displ_Values_COMP <- c(V_F80_COMP,V_30mm_C)
  Load_Values_COMP <- c(F_80_COMP,F_30mm_C)
  mini_COMP <- which.max(Displ_Values_COMP)
  Ultimate_displ_COMP <- Displ_Values_COMP[mini_COMP]
  Ultimate_force_COMP <- Load_Values_COMP[mini_COMP]

  Displ_Values_TENS <- c(V_F80_TENS,V_30mm_T)
  Load_Values_TENS <- c(F_80_TENS,F_30mm_T)
  mini_TENS <- which.min(Displ_Values_TENS)
  Ultimate_displ_TENS <- Displ_Values_TENS[mini_TENS]
  Ultimate_force_TENS <- Load_Values_TENS[mini_TENS]

  result["Ultimate Force Compression (N)",i] <- Ultimate_force_COMP
  result["Ultimate Displ Compression (mm)",i] <- Ultimate_displ_COMP
  result["Ultimate Force Tension (N)",i] <- Ultimate_force_TENS
  result["Ultimate Displ Tension (mm)",i] <- Ultimate_displ_TENS

#-------------------------------------------------------------------------
--
  # PLOTTING GRAPH
  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
        main = main_title, col = "Orange",cex = 0.2,
        xlab = "Displacement (mm)",ylab = "Force (N)")

  # plotting 1st LEC
  points(LEC1_COMP, cex=0.2,col="Black",type="l")
  points(LEC1_TENS, cex=0.2,col="Black",type="l")
  # plotting 3rd LEC
  points(LEC3_COMP, cex=0.2,col="Blue",type="l")
  points(LEC3_TENS, cex=0.2,col="Blue",type="l")

  # plotting ultimate displ
  abline (v=Ultimate_displ_COMP,col="grey", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
  abline (v=Ultimate_displ_TENS,col="grey", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
  text(Ultimate_displ_TENS+0.1,F_MAX_C,labels = "Vu",adj = c(0,0),
        cex = 0.8, col = "grey")
  text(Ultimate_displ_COMP+0.1,F_MAX_C,labels = "Vu",adj = c(1,0),
```

```
        cex = 0.8, col = "grey")

  # plotting only the positive LECs
  main_title2 <- paste("Envelope curve LEC1 and LEC3 - ",
        specimen_name,i,sep="")
  plot(LEC1_TENS, type = "l", col = "Black", main = main_title2 )
  points(LEC3_TENS, type = "l", col = "Lightblue", cex = 0.2)
  points(LEC1_TENS, type = "l", col = "Black")
  # plotting ultimate displ and force
  abline (v=Ultimate_displ_TENS,col="Black", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
  text(Ultimate_displ_TENS+0.1,0,labels = "Vu",adj = c(0,0),
        cex = 0.8, col = "Black")

#-----------------------------------------------------------------------------
--
  # CALCULATIONS
  # Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
  xsub <- subset(LEC1_TENS,LEC1_TENS$Position < V_FMAX_TENS)
  row3 <- which.min(abs(xsub$Force - (0.1*F_MAX_T)))
  V10 <- xsub[row3,1]
  F10 <- xsub[row3,2]
  row4 <- which.min(abs(xsub$Force - (0.4*F_MAX_T)))
  V40 <- xsub[row4,1]
  F40 <- xsub[row4,2]
  xcoord <- c(V10,V40)
  ycoord <- c(F10,F40)
  # plotting points and slope in LEC graph
  points(x = V10, y = F10,col="Red")
  points(x = V40, y=F40, col="Red")
  fit <- lm(ycoord~xcoord)
  abline(fit, col="black",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)

  # The intercept and slope for line1 is then found, and further the
    Elastic stiffness, which is equal to the slope.
  b <- coef(lm(ycoord~xcoord))[[1]] #intercept
  a <- coef(lm(ycoord~xcoord))[[2]] #slope
  Vu <- Ultimate_displ_TENS

  # YIELD LOAD
  # The yield load is the interception between line1 and a second line2
  # Line 2 angle is 1/6 of line1s' angle and is touching the graph.
  # Firstly, we find the slope of line2 and secondly, we found the point
in
    LEC1 that has the same slope.
  line2_slope <- (1/6)*a
  # finding the slope in graph that is similar to line2_slope
  slope_data <- as.data.frame(matrix(nrow=0,ncol=6))
  colnames(slope_data) <- c("Intersept","Slope", "x1","x2","y1","y2" )
  sub_line2 <- subset(LEC1_TENS,LEC1_TENS$Position <= V_FMAX_TENS+0.5)
  for (m in 2:nrow(sub_line2)){
    x_graph <- c(sub_line2[m-1,1],sub_line2[m,1])
    y_graph <- c(sub_line2[m-1,2],sub_line2[m,2])
    coef_inter <- coef(lm(y_graph~x_graph))[1]
    coef_slope <- coef(lm(y_graph~x_graph))[2]
    slope_data[m,1] <- coef_inter
    slope_data[m,2] <- coef_slope
    slope_data[m,3] <- x_graph[1]
    slope_data[m,4] <- x_graph[2]
    slope_data[m,5] <- y_graph[1]
    slope_data[m,6] <- y_graph[2]
```

```
}
row_y <- which.min(abs(slope_data[,2]-line2_slope))
line2_data <- slope_data[row_y,]
x2_coord <- c(slope_data[row_y,3],slope_data[row_y,4])
y2_coord <- c(slope_data[row_y,5],slope_data[row_y,6])
line2_fit <- lm(y2_coord~x2_coord)
abline(line2_fit, col="gray",lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)

# FINDING F_y, v_y and k_ser
# Need to find the intersection between line1 and line2
# Algebraically calculated using intersection and slope from line1 and
2
V_y <- (line2_data[1,1]-b)/(a-line2_data[1,2])
F_y <- b+a*V_y
result["V_y",i] <- V_y
result["F_y",i] <- F_y

abline (h=F_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)
abline (v=V_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)

k_ser <- F_y/V_y
result["k_ser",i]<- k_ser


# DUCTILITY, D_c
D_c <- Ultimate_displ_TENS/V_y
result["D_c",i] <- D_c

# DISSIPATION OF ENERGY - Equivalent viscous damping ratio, v_eq

# https://math.blogoverflow.com
  /2014/06/04/greens-theorem-and-area-of-polygons/
# Creates an area function that is derived from Greens Theorem. If the
  area is negative it is due to a clockwise direction.
area1<-function(X){
  X<-rbind(X,X[1,])
  x<-X[,1]; y<-X[,2]; lx<-length(x)
  sum(((x[2:lx]+x[1:lx-1])*(y[2:lx]-y[1:lx-1])))/2
}

cycles <- c()
# need to subset each cycle to calculate the energy dissipation
cycle075_1st <- subset(my_data,my_data$Time >= 30 & my_data$Time <= 60,
              select = c("Position","Force (smoothed)"))
cycle075_2nd <- subset(my_data,my_data$Time >= 60 & my_data$Time <= 90,
              select = c("Position","Force (smoothed)"))
cycle075_3rd <- subset(my_data,my_data$Time >= 90 & my_data$Time <= 120,
              select = c("Position","Force (smoothed)"))
cycles[[1]] <- cycle075_1st
cycles[[2]] <- cycle075_2nd
cycles[[3]] <- cycle075_3rd

cycle1_1st <- subset(my_data,my_data$Time >= 120
  & my_data$Time <= 160,select = c("Position","Force (smoothed)"))
cycle1_2nd <- subset(my_data,my_data$Time >= 160
  & my_data$Time <= 200,select = c("Position","Force (smoothed)"))
cycle1_3rd <- subset(my_data,my_data$Time >= 200
  & my_data$Time <= 240,select = c("Position","Force (smoothed)"))
```

```
cycles[[4]] <- cycle1_1st
cycles[[5]] <- cycle1_2nd
cycles[[6]] <- cycle1_3rd

cycle2_1st <- subset(my_data,my_data$Time >= 240
  & my_data$Time <= 320,select = c("Position","Force (smoothed)"))
cycle2_2nd <- subset(my_data,my_data$Time >= 320
  & my_data$Time <= 400,select = c("Position","Force (smoothed)"))
cycle2_3rd <- subset(my_data,my_data$Time >= 400
  & my_data$Time <= 480,select = c("Position","Force (smoothed)"))
cycles[[7]] <- cycle2_1st
cycles[[8]] <- cycle2_2nd
cycles[[9]] <- cycle2_3rd

cycle4_1st <- subset(my_data,my_data$Time >= 480
  & my_data$Time <= 640,select = c("Position","Force (smoothed)"))
cycle4_2nd <- subset(my_data,my_data$Time >= 640
  & my_data$Time <= 800,select = c("Position","Force (smoothed)"))
cycle4_3rd <- subset(my_data,my_data$Time >= 800
  & my_data$Time <= 960,select = c("Position","Force (smoothed)"))
cycles[[10]] <- cycle4_1st
cycles[[11]] <- cycle4_2nd
cycles[[12]] <- cycle4_3rd

cycle6_1st <- subset(my_data,my_data$Time >= 960
  & my_data$Time <= 1200,select = c("Position","Force (smoothed)"))
cycle6_2nd <- subset(my_data,my_data$Time >= 1200
  & my_data$Time <= 1440,select = c("Position","Force (smoothed)"))
cycle6_3rd <- subset(my_data,my_data$Time >= 1440
  & my_data$Time <= 1680,select = c("Position","Force (smoothed)"))
cycles[[13]] <- cycle6_1st
cycles[[14]] <- cycle6_2nd
cycles[[15]] <- cycle6_3rd

cycle8_1st <- subset(my_data,my_data$Time >= 1680
  & my_data$Time <= 2000,select = c("Position","Force (smoothed)"))
cycle8_2nd <- subset(my_data,my_data$Time >= 2000
  & my_data$Time <= 2320,select = c("Position","Force (smoothed)"))
cycle8_3rd <- subset(my_data,my_data$Time >= 2320
  & my_data$Time <= 2640,select = c("Position","Force (smoothed)"))
cycles[[16]] <- cycle8_1st
cycles[[17]] <- cycle8_2nd
cycles[[18]] <- cycle8_3rd

cycle10_1st <- subset(my_data,my_data$Time >= 2640
  & my_data$Time <= 3040,select = c("Position","Force (smoothed)"))
cycle10_2nd <- subset(my_data,my_data$Time >= 3040
  & my_data$Time <= 3440,select = c("Position","Force (smoothed)"))
cycle10_3rd <- subset(my_data,my_data$Time >= 3440
  & my_data$Time <= 3840,select = c("Position","Force (smoothed)"))
cycles[[19]] <- cycle10_1st
cycles[[20]] <- cycle10_2nd
cycles[[21]] <- cycle10_3rd

cycle12_1st <- subset(my_data,my_data$Time >= 3840
  & my_data$Time <= 4320,select = c("Position","Force (smoothed)"))
cycle12_2nd <- subset(my_data,my_data$Time >= 4320
  & my_data$Time <= 4800,select = c("Position","Force (smoothed)"))
cycle12_3rd <- subset(my_data,my_data$Time >= 4800
  & my_data$Time <= 5280,select = c("Position","Force (smoothed)"))
cycles[[22]] <- cycle12_1st
```

```
cycles[[23]] <- cycle12_2nd
cycles[[24]] <- cycle12_3rd

cycle14_1st <- subset(my_data,my_data$Time >= 5280
  & my_data$Time <= 5840,select = c("Position","Force (smoothed)"))
cycle14_2nd <- subset(my_data,my_data$Time >= 5840
  & my_data$Time <= 6400,select = c("Position","Force (smoothed)"))
cycle14_3rd <- subset(my_data,my_data$Time >= 6400
  & my_data$Time <= 6960,select = c("Position","Force (smoothed)"))
cycles[[25]] <- cycle14_1st
cycles[[26]] <- cycle14_2nd
cycles[[27]] <- cycle14_3rd

names(cycles) <- c("cycle075_1st","cycle075_2nd","cycle075_3rd",
    "cycle1_1st","cycle1_2nd","cycle1_3rd","cycle2_1st","cycle2_2nd",
    "cycle2_3rd","cycle4_1st","cycle4_2nd","cycle4_3rd","cycle6_1st",
    "cycle6_2nd","cycle6_3rd","cycle8_1st","cycle8_2nd","cycle8_3rd",
  "cycle10_1st","cycle10_2nd","cycle10_3rd","cycle12_1st","cycle12_2nd"
  ,"cycle12_3rd","cycle14_1st","cycle14_2nd",
    "cycle14_3rd")

# checking cycle12 and cycle14
#If the last list in cycles is empty, then delete that list. Stop when
  the last element is not zero.
while (nrow(cycles[[length(cycles)]]) == 0) {
  cycles[[length(cycles)]] <- NULL
}

cycle_values <- as.data.frame(matrix(nrow = 3,ncol = length(cycles)))
colnames(cycle_values) <- c(names(cycles))
rownames(cycle_values) <- c("Ed (J)","Ep (J)","v_eq")

# Dissipation Energy and potential Energy
for (k in 1:length(cycles)) {
  cycle_half <- subset(cycles[[k]],cycles[[k]]$Position > 0)
  Ed <- area1(cycle_half)
  if (Ed < 0) { # If the area is negative, the value is correct, but the
        direction was clockwise
    Ed <- -Ed
  }
  cycle_values["Ed (J)",k] <- Ed/1000

  rowEp <- which.max(cycles[[k]]$Position)
  xx <- c(0,cycles[[k]]$Position[rowEp],cycles[[k]]$Position[rowEp])
  yy <- c(0,0,cycles[[k]]$`Force (smoothed)`[rowEp])
  xy <- cbind(xx,yy)
  Ep <- area1(xy)
  cycle_values["Ep (J)",k] <- Ep/1000

  v_eqCycle <- Ed/(Ep*2*pi)
  cycle_values["v_eq",k] <- v_eqCycle
}

# Stores all the viscous damping ratio for each specimen in a list.
v_eq_values[[name_specimen]] <- cycle_values

# IMPAIRMENT OF STRENGTH, Delta_F
Delta_F <- as.data.frame(matrix(nrow = 0,ncol = 2))
colnames(Delta_F) <- c(paste(Delta,"F_Tension", sep = ""),
              paste(Delta,"F_Compression", sep = ""))
for (s in c(seq(3,length(cycles),3))) {
```

```
    # Compression
    sub_Delta1_COMP <- subset(cycles[[s-2]],cycles[[s-2]]$Position <= 0)
    sub_Delta3_COMP <- subset(cycles[[s]],cycles[[s]]$Position <= 0)
    V1 <- which.min(sub_Delta1_COMP$Position)
    F1 <- sub_Delta1_COMP[V1,2]
    V1 <- sub_Delta1_COMP[V1,1]

    V3 <- which.min(sub_Delta3_COMP$Position)
    F3 <- sub_Delta3_COMP[V3,2]
    V3 <- sub_Delta3_COMP[V3,1]

    DeltaF_COMP <- abs(F1-F3)
    Delta_F[s,2] <- DeltaF_COMP

    # Tension
    sub_Delta1_TENS <- subset(cycles[[s-2]],cycles[[s-2]]$Position >= 0)
    sub_Delta3_TENS <- subset(cycles[[s]],cycles[[s]]$Position >= 0)
    V1 <- which.max(sub_Delta1_TENS$Position)
    F1 <- sub_Delta1_TENS[V1,2]
    V1 <- sub_Delta1_TENS[V1,1]

    V3 <- which.max(sub_Delta3_TENS$Position)
    F3 <- sub_Delta3_TENS[V3,2]
    V3 <- sub_Delta3_TENS[V3,1]

    DeltaF_TENS <- abs(F1-F3)
    Delta_F[s,1] <- DeltaF_TENS
  }
  Delta_F <- Delta_F[complete.cases(Delta_F),]

  #-----------------------------------------------------------------------
--
  # FINAL RESULTS TABLE

  # COMPRESSION
  Final_resultsCOMP[i,"Maximum Load (kN)"] <- F_MAX_C/1000
  Final_resultsCOMP[i,"Displacement at Max Load (mm)"] <- V_FMAX_COMP
  Final_resultsCOMP[i,"Ultimate Load (kN)"] <- Ultimate_force_COMP/1000
  Final_resultsCOMP[i,"Ultimate displacement (mm)"] <- Ultimate_displ_COMP

  # TENSION
  Final_resultsTENS[i,"Maximum Load (kN)"] <- F_MAX_T/1000
  Final_resultsTENS[i,"Displacement at Max Load (mm)"] <- V_FMAX_TENS
  Final_resultsTENS[i,"Ultimate Load (kN)"] <- Ultimate_force_TENS/1000
  Final_resultsTENS[i,"Ultimate displacement (mm)"] <- Ultimate_displ_TENS

  Final_resultsSPEC[i,"Yield slip (mm) - V_y"] <- V_y
  Final_resultsSPEC[i,"Yield Load (kN) - Fy"] <- F_y/1000
  Final_resultsSPEC[i,"Slip modulus - k_ser (kN/mm)"] <- k_ser/1000
  Final_resultsSPEC[i,"Static ductility - D"] <- D_c

}

# MEAN VALUES, PARAMETERS AND STANDARD DEVIATIONS
for (o in 1:ncol(Final_resultsCOMP)) {
  Final_resultsCOMP["Mean Values",o] <- mean(Final_resultsCOMP
          [1:length(myFiles2),o], na.rm = TRUE)
  Final_resultsCOMP["Standard Deviations",o] <- sd(Final_resultsCOMP
          [(1:length(myFiles2)),o],na.rm=TRUE)
}
```

```
for (o in 1:ncol(Final_resultsTENS)) {
  Final_resultsTENS["Mean Values",o] <- mean(Final_resultsTENS
          [1:length(myFiles2),o], na.rm = TRUE)
  Final_resultsTENS["Standard Deviations",o] <- sd(Final_resultsTENS
          [(1:length(myFiles2)),o],na.rm=TRUE)
}

for (o in 1:ncol(Final_resultsSPEC)) {
  Final_resultsSPEC["Mean Values",o] <- mean(Final_resultsSPEC
          [1:length(myFiles2),o], na.rm = TRUE)
  Final_resultsSPEC["Standard Deviations",o] <- sd(Final_resultsSPEC
          [(1:length(myFiles2)),o],na.rm=TRUE)
}

# Writing viscous damping for each specimen to excel
for (f in 1:length(v_eq_values)) {
  filename <- paste("EN12512(2002) Viscous damping ",
          specimen_name,"-",f,".xlsx",sep="")
  v_eq_specimen <- v_eq_values[[f]]
  is.num <- sapply(v_eq_specimen, is.numeric)
  v_eq_specimen[is.num] <- lapply(v_eq_specimen[is.num], round, 3)
  write.xlsx(v_eq_specimen,file = filename,row.names = T,col.names = T)
}

# Restricts decimals to three numbers in the tables of interest.
is.num <- sapply(Final_resultsCOMP, is.numeric)
Final_resultsCOMP[is.num] <- lapply(Final_resultsCOMP[is.num], round, 3)
is.num <- sapply(Final_resultsTENS, is.numeric)
Final_resultsTENS[is.num] <- lapply(Final_resultsTENS[is.num], round, 3)
is.num <- sapply(Final_resultsSPEC, is.numeric)
Final_resultsSPEC[is.num] <- lapply(Final_resultsSPEC[is.num], round, 3)
is.num <- sapply(result, is.numeric)
result[is.num] <- lapply(result[is.num], round, 3)

# SAVING TABLES AS .TXT FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_resultsCOMP,
          file = "EN12512(2002) Final_resultsCOMP.xlsx",
          row.names = T, col.names = T)
write.xlsx(Final_resultsTENS,
          file = "EN12512(2002) Final_resultsTENS.xlsx",
          row.names = T, col.names = T)
write.xlsx(Final_resultsSPEC,
          file = "EN12512(2002) Final_resultsParameters.xlsx",
          row.names = T, col.names = T)
```

2.2 EN 12512 (2018) Draft proposal version 20180410

```
# Study 1.2 Cyclic load test
# EN 12512 (2018) Draft Proposal version 20180410

# PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
# install.packages("zoo")
library("zoo")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")
library("xlsx", lib.loc="~/R/win-library/3.5")
```

```
# DATA SET INFO
specimen_name <- "HNSc"
study <- "STUDY 1.2 CYCLIC"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA
          ANALYZING\\", study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master  2018\\3  -  DATA  ANALYZING\\",  study  ,"\\",
          specimen_name,sep=""))

all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(".csv","",myFiles2)

result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2
Final_resultsCOMP                                                        <-
as.data.frame(matrix(nrow=length(myFiles2)+2,ncol=0))
Final_resultsTENS                                                        <-
as.data.frame(matrix(nrow=length(myFiles2)+2,ncol=0))
rownames(Final_resultsCOMP)<-c(myFiles2,"Mean Values",
          "Standard Deviations")
rownames(Final_resultsTENS)<-c(myFiles2,"Mean Values",
          "Standard Deviations")
beta <- intToUtf8(946)
v_eq_values <- c()

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], nlines = 1, what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=1, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=1,sep=";",dec=",",header=TRUE)
  colnames(dat) <- paste(header,header2,sep="")
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles

#-------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[[i]])
  specimen$lopende_kraft<-NA
  for (p in 8:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-8,linje+8)
    specimen$lopende_kraft[linje]<
          mean(specimen[intervall[1]:intervall[2],3])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=3))
```

```
  my_data <- specimen[1]
  my_data[2] <- specimen[2]
  my_data[3] <- specimen[4]
  colnames(my_data)<-c("Time", "Position","Force (smoothed)")

  # -----------------------------------------------------------------------
--
  # ENVELOPE CURVE (LEC)
  # COMPRESSION (negativ values)
  # Find maximum for each subsets and construct the envelope curve for
    compressive load
  Csubplot_data <- c()
  Cset1_3 <- subset(my_data,my_data$Time < 120 & my_data$Position < 0)
    # includes 1st,2nd and 3rd cyclesets.
  Csubplot_data[["Cset1_3"]]<- Cset1_3

  Cset1V <- subset(my_data,my_data$Time > 120 & my_data$Time < 240
          & my_data$Position < 0 ) # 4th cycleset
  Csubplot_data[["Cset1V"]]<- Cset1V

  Cset2V <- subset(my_data,my_data$Time > 240 & my_data$Time < 480
          & my_data$Position < 0) # 5th cycleset
  Csubplot_data[["Cset2V"]]<- Cset2V

  Cset4V <- subset(my_data,my_data$Time > 480 & my_data$Time < 960
          & my_data$Position < 0) # 6th cycleset
  Csubplot_data[["Cset4V"]]<- Cset4V

  Cset6V <- subset(my_data,my_data$Time > 960 & my_data$Time < 1680
          & my_data$Position < 0) # 7th cycleset
  Csubplot_data[["Cset6V"]]<- Cset6V

  Cset8V <- subset(my_data,my_data$Time > 1680 & my_data$Time < 2640
          & my_data$Position < 0) # 8th cycleset
  Csubplot_data[["Cset8V"]]<- Cset8V

  Cset10V <- subset(my_data,my_data$Time > 2640 & my_data$Time < 3840
          & my_data$Position < 0) # 9th cycleset
  Csubplot_data[["Cset10V"]]<- Cset10V

  Cset12V <- subset(my_data,my_data$Time > 3840 & my_data$Time < 5280
          & my_data$Position < 0) # 10th cycleset
  if (any(Cset12V$Position < -23.90)) { # if-statement to check if the
last
    cycle is complete
    Csubplot_data[["Cset12V"]] <- Cset12V
  }

  Cset14V <- subset(my_data,my_data$Time > 5280 & my_data$Time < 6960
          & my_data$Position < 0) # 11th cycleset
  if (any(Cset14V$Position < -27.90)) { # if-statement to check if the
last
    cycle is complete
    Csubplot_data[["Cset14V"]] <- Cset14V
  }

  # Constructing load envelope curve (LEC) for the first load curve in
each
    cycle
  LEC1_COMP <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
    fill with max values
```

```
colnames(LEC1_COMP) <- c("Position","Force")

# Points in 1st, 2nd and 3rd cycleset
LEC1_COMP["zero point",1:2] <- c(0,0)

points1 <- subset(Cset1_3,Cset1_3$Position > -0.1)
row <- which.min(points1$`Force (smoothed)`)
point1 <- points1[row,]
LEC1_COMP["initial point",1:2] <- point1[1,2:3]

points2 <- subset(Cset1_3,Cset1_3$Position > -0.55
        & Cset1_3$Position < -0.45)
row <- which.min(points2$`Force (smoothed)`)
point2 <- points2[row,]
LEC1_COMP["-0.5 mm",1:2] <- point2[1,2:3]

points3 <- subset(Cset1_3,Cset1_3$Position > -1.05
        & Cset1_3$Position < -0.95)
row <- which.min(points3$`Force (smoothed)`)
point3 <- points3[row,]
LEC1_COMP["-1.0 mm",1:2] <- point3[1,2:3]
LEC1_COMP["-1.5 mm",1:2] <-
        Cset1_3[which.min(Cset1_3$`Force (smoothed)`),2:3]

# Points in 4th to 11th cycleset 1st LEC
for (n in 2:length(Csubplot_data)) {
  displ_COMP <- -c(0,1,seq(2,14,2))*2
  subpoints <- subset(Csubplot_data[[n]],
                      Csubplot_data[[n]][2] > displ_COMP[n]-0.05
                    & Csubplot_data[[n]][2] < displ_COMP[n]+0.05,
                      select = c("Position","Force (smoothed)"))
 #subsetting for position
  rowmin <- which.min(subpoints$`Force (smoothed)`)
  name_row <- paste(displ_COMP[n],"mm")
  LEC1_COMP[name_row,1:2] <- subpoints[rowmin,1:2]
}

# Constructing load envelope curve (LEC) for the third load curve in
each
  cycle
LEC3_COMP <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
  fill with max values
colnames(LEC3_COMP) <- c("Position","Force")
# starts from 1.5 mm
points1_5 <- subset(Cset1_3,Cset1_3$Position > -1.505
            & Cset1_3$Position < -1.495)
row <- which.max(points1_5$`Force (smoothed)`)
point1_5 <- points1_5[row,]
LEC3_COMP["-1.5 mm",1:2] <- point1_5[1,2:3]

# Points in 4th to 9th cycleset 3rd LEC
for (n in 2:length(Csubplot_data)) {
  displ_COMP <- -c(0,1,seq(2,14,2))*2
  subpoints <- subset(Csubplot_data[[n]],
              Csubplot_data[[n]][2] > displ_COMP[n]-0.005
                & Csubplot_data[[n]][2] < displ_COMP[n]+0.005,
              select = c("Position","Force (smoothed)"))
  rowmin <- which.max(subpoints$`Force (smoothed)`)
  name_row <- paste(displ_COMP[n],"mm")
  LEC3_COMP[name_row,1:2] <- subpoints[rowmin,1:2]
}
```

```
# TENSION (positive values)
Tsubplot_data <- c()
Tset1_3 <- subset(my_data,my_data$Time < 120 & my_data$Position > 0)
  # includes 1st,2nd and 3rd cyclesets.
Tsubplot_data[["Tset1_3"]]<- Tset1_3

Tset1V <- subset(my_data,my_data$Time > 120 & my_data$Time < 240
        & my_data$Position > 0 ) # 4th cycleset
Tsubplot_data[["Tset1V"]]<- Tset1V

Tset2V <- subset(my_data,my_data$Time > 240 & my_data$Time < 480
        & my_data$Position > 0) # 5th cycleset
Tsubplot_data[["Tset2V"]]<- Tset2V

Tset4V <- subset(my_data,my_data$Time > 480 & my_data$Time < 960
        & my_data$Position > 0) # 6th cycleset
Tsubplot_data[["Tset4V"]]<- Tset4V

Tset6V <- subset(my_data,my_data$Time > 960 & my_data$Time < 1680
        & my_data$Position > 0) # 7th cycleset
Tsubplot_data[["Tset6V"]]<- Tset6V

Tset8V <- subset(my_data,my_data$Time > 1680 & my_data$Time < 2640
        & my_data$Position > 0) # 8th cycleset
Tsubplot_data[["Tset8V"]]<- Tset8V

Tset10V <- subset(my_data,my_data$Time > 2640 & my_data$Time < 3840
        & my_data$Position > 0) # 9th cycleset
Tsubplot_data[["Tset10V"]]<- Tset10V

Tset12V <- subset(my_data,my_data$Time > 3840 & my_data$Time < 5280
        & my_data$Position > 0) # 10th cycleset
if (any(Tset12V$Position > 23.90)) { # if-statement to check if the last
  cycle is complete
  Tsubplot_data[["Tset12V"]] <- Tset12V
}

Tset14V <- subset(my_data,my_data$Time > 5280 & my_data$Time < 6960
        & my_data$Position > 0) # 11th cycleset
if (any(Tset14V$Position > 27.90)) { # if-statement to check if the last
  cycle is complete
  Tsubplot_data[["Tset14V"]] <- Tset14V
}

LEC1_TENS <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
  fill with max values
colnames(LEC1_TENS) <- c("Position","Force")

# Points in 1st, 2nd and 3rd cycleset
LEC1_TENS["zero point",1:2] <- c(0,0)

Tpoints1 <- subset(Tset1_3,Tset1_3$Position < 0.1)
row <- which.max(Tpoints1$`Force (smoothed)`)
Tpoint1 <- Tpoints1[row,]
LEC1_TENS["initial point",1:2] <- Tpoint1[1,2:3]

Tpoints2 <- subset(Tset1_3,Tset1_3$Position < 0.55
        & Tset1_3$Position > 0.45)
row <- which.max(Tpoints2$`Force (smoothed)`)
Tpoint2 <- Tpoints2[row,]
```

```
LEC1_TENS["0.5 mm",1:2] <- Tpoint2[1,2:3]

Tpoints3 <- subset(Tset1_3,Tset1_3$Position < 1.05
        & Tset1_3$Position > 0.95)
row <- which.max(Tpoints3$`Force (smoothed)`)
Tpoint3 <- Tpoints3[row,]
LEC1_TENS["1.0 mm",1:2] <- Tpoint3[1,2:3]
LEC1_TENS["1.5 mm",1:2] <-
        Tset1_3[which.max(Tset1_3$`Force (smoothed)`),2:3]

# Points in 4th to 11th cycleset 1st LEC
for (n in 2:length(Tsubplot_data)) {
  displ_TENS <- c(0,1,seq(2,14,2))*2
  Tsubpoints <- subset(Tsubplot_data[[n]],
              Tsubplot_data[[n]][2] > displ_TENS[n]-0.05
                & Tsubplot_data[[n]][2] < displ_TENS[n]+0.05,
              select = c("Position","Force (smoothed)"))
  rowmin <- which.max(Tsubpoints$`Force (smoothed)`)
  name_row <- paste(displ_TENS[n],"mm")
  LEC1_TENS[name_row,1:2] <- Tsubpoints[rowmin,1:2]
}

# Constructing load envelope curve (LEC) for the third load curve in
each
  cycle
LEC3_TENS <- as.data.frame(matrix(nrow = 0,ncol = 2)) # empty matrix to
  fill with max values
colnames(LEC3_TENS) <- c("Position","Force")

# starts from 1.5 mm
Tpoints1_5 <- subset(Tset1_3,Tset1_3$Position < 1.505
        & Tset1_3$Position > 1.495)
row <- which.min(Tpoints1_5$`Force (smoothed)`)
Tpoint1_5 <- Tpoints1_5[row,]
LEC3_TENS["1.5 mm",1:2] <- Tpoint1_5[1,2:3]

# Points in 4th to 11th cycleset 3rd LEC
for (n in 2:length(Tsubplot_data)) {
  displ_TENS <- c(0,1,seq(2,14,2))*2
  Tsubpoints <- subset(Tsubplot_data[[n]],
              Tsubplot_data[[n]][2] > displ_TENS[n]-0.005
                & Tsubplot_data[[n]][2] < displ_TENS[n]+0.005,
              select = c("Position","Force (smoothed)"))
  rowmin <- which.min(Tsubpoints$`Force (smoothed)`)
  name_row <- paste(displ_TENS[n],"mm")
  LEC3_TENS[name_row,1:2] <- Tsubpoints[rowmin,1:2]
}

# interpolating LEC1 and LEC3
LEC1_COMP <- as.data.frame(approx(LEC1_COMP$Position,LEC1_COMP$Force,
        method = "linear", n = 1000)) #interpolation
LEC1_TENS <- as.data.frame(approx(LEC1_TENS$Position,
        LEC1_TENS$Force,method = "linear", n = 1000))
colnames(LEC1_COMP) <- c("Position","Force")
colnames(LEC1_TENS) <- c("Position","Force")
LEC3_COMP <- as.data.frame(approx(LEC3_COMP$Position,
        LEC3_COMP$Force, method = "linear", n = 1000)) #interpolation
LEC3_TENS <- as.data.frame(approx(LEC3_TENS$Position,LEC3_TENS$Force,
        method = "linear", n = 1000))
names(LEC3_COMP) <- c("Position","Force")
names(LEC3_TENS) <- c("Position","Force")
```

```r
#--------------------------------------------------------------------------
# CALCULATION | Ultimate load (Failure; 80% F_max, beta):
# Failure is not calculated due to no distinct failure-drop in these experiments LECs.

# PEAK LOAD (Pl) IN LEC1, COMPRESSION
row1 <- which.min(LEC1_COMP$Force)
Pl_COMP <- LEC1_COMP$Force[row1]
V_Pl_COMP <- LEC1_COMP$Position[row1]

# PEAK LOAD (Pl) IN LEC1, TENSION
row2 <- which.max(LEC1_TENS$Force)
Pl_TENS <- LEC1_TENS$Force[row2]
V_Pl_TENS <- LEC1_TENS$Position[row2]

result["Peak Load Compression (N)", i] <- Pl_COMP
result["Displ at Peak Load compression (mm)",i] <- V_Pl_COMP
result["Peak Load Tension (N)", i] <- Pl_TENS
result["Displ at Peak Load Tension (mm)",i] <- V_Pl_TENS

# DISPLACEMENT 80% Pl_max after peak load, COMPRESSION
Pl_80_COMP <- 0.8*Pl_COMP
subs <- subset(LEC1_COMP,LEC1_COMP$Position < V_Pl_COMP)
row <- which.min(abs(subs$Force - Pl_80_COMP))
V_Pl80_COMP <- subs$Position[row]
result["80% Peak Load Compression (N)",i] <- Pl_80_COMP
result["Displ at 80% Peak Load Compression (mm)",i] <- V_Pl80_COMP

# DISPLACEMENT 80% Pl_max after peak load, TENSION
Pl_80_TENS <- 0.8*Pl_TENS
subs <- subset(LEC1_TENS,LEC1_TENS$Position > V_Pl_TENS)
row <- which.min(abs(subs$Force - Pl_80_TENS))
V_Pl80_TENS <- subs$Position[row]
result["80% Peak Load  Tension (N)",i] <- Pl_80_TENS
result["Displ at 80% Peak Load Tension (mm)",i] <- V_Pl80_TENS

#--------------------------------------------------------------------------
# STRENGTH DEGRADATION FACTOR BETWEEN LEC1 AND LEC3
beta_min <- 0.75 # A given beta_min

# limiting the decimals to two digits, so that it is possible to compare and divide on each other.
is.num <- sapply(LEC1_TENS, is.numeric)
LEC1_TENS[is.num] <- lapply(LEC1_TENS[is.num], round, 2)
is.num <- sapply(LEC3_TENS, is.numeric)
LEC3_TENS[is.num] <- lapply(LEC3_TENS[is.num], round, 2)
is.num <- sapply(LEC1_COMP, is.numeric)
LEC1_COMP[is.num] <- lapply(LEC1_COMP[is.num], round, 2)
is.num <- sapply(LEC3_COMP, is.numeric)
LEC3_COMP[is.num] <- lapply(LEC3_COMP[is.num], round, 2)

# Compression
# Firstly, need to make a new data-frame, d12, where LEC1 and LEC3 are joined after the same positions
# Secondly, dividing the force that matches the same position and binding them to a new data-frame Beta_COMP
df12 <- left_join(LEC1_COMP, LEC3_COMP, by = 'Position')
beta_COMP <- cbind(df12[1], df12[3] / df12[2])
```

```
beta_COMP <- beta_COMP[beta_COMP$Force.y < 1,]
beta_COMP <- beta_COMP[complete.cases(beta_COMP), ]
beta_COMP <- as.data.frame(approx(beta_COMP$Position,beta_COMP$Force.y,
        method = "linear", n = 1000)) #interpolation
colnames(beta_COMP) <- c("Position","Force.y")

rowC <- which.min(beta_COMP$Force.y)
betaMinValue_C <- beta_COMP$Force.y[rowC]
if (betaMinValue_C <= beta_min) {
  #if BetaMinValue_T is lower than beta_min it is valid as a ultimate
  displ.
  sub_beta_C <- subset(beta_COMP,
              beta_COMP$Position > beta_COMP$Position[rowC])
  rowCC <- which.min(abs(sub_beta_C$Force.y - beta_min))
  V_beta_COMP <- sub_beta_C$Position[rowCC]
  row_Fbeta <- which.min(abs(LEC1_COMP$Position - V_beta_COMP))
  F_beta_COMP <- LEC1_COMP$Force[row_Fbeta]

  sub_beta_C2 <- subset(beta_COMP,beta_COMP$Position > V_beta_COMP)
  rowCCC <- which.min(sub_beta_C2$Force.y)
  betaMinValue_C2 <- sub_beta_C2$Force.y[rowCCC]
  if (betaMinValue_C2 < sub_beta_C$Force.y[rowCC]) { # if there are any
  values lower than beta_COMP
    sub_beta_C3 <- subset(sub_beta_C2,
              sub_beta_C2$Position >= sub_beta_C2$Position[rowCCC])
    rowCC2 <- which.min(abs(sub_beta_C3$Force.y - beta_min))
    V_beta_COMP <- sub_beta_C3$Position[rowCC2]
    row_Fbeta <- which.min(abs(LEC1_COMP$Position - V_beta_COMP))
    F_beta_COMP <- LEC1_COMP$Force[row_Fbeta]
  }
} else { # if not, V_beta is ignored with NA when deciding ultimate
  displacement.
  V_beta_COMP <- NA
  F_beta_COMP <- NA
}

# PLOTTING BETA.
plot(beta_COMP$Position,beta_COMP$Force.y,type = "l",main = beta,
        xlab = "Displacement (mm)",
        ylab = paste(beta,"_compression"),col = "red")
abline(h=0.75)
text(-1,0.75,labels = "0.75", adj = c(1,1))
abline(v=V_P180_COMP)
text(V_P180_COMP,0.65,labels = "80% \nPeak Load",adj = c(0,0))
abline(v=V_beta_COMP)
text(V_beta_COMP,0.65,labels = "V_beta",adj=c(1,0))

# Tension
# Firstly, need to make a new data-frame, d12, where LEC1 and LEC3 are
  joined after the same positions
# Secondly, dividing the force that matches the same position and binding
  them to a new data-frame Beta_TENS
df12 <- left_join(LEC1_TENS, LEC3_TENS, by = 'Position')
beta_TENS <- cbind(df12[1], df12[3] / df12[2])
beta_TENS <- beta_TENS[beta_TENS$Force.y < 1,]
beta_TENS <- beta_TENS[complete.cases(beta_TENS), ]
beta_TENS <- as.data.frame(approx(beta_TENS$Position,
              beta_TENS$Force,method = "linear", n = 1000))
colnames(beta_TENS) <- c("Position","Force.y")

rowT <- which.min(beta_TENS$Force.y)
```

```
betaMinValue_T <- beta_TENS$Force.y[rowT]
if (betaMinValue_T <= beta_min) {
  #if BetaMinValue_T is lower than beta_min it is valid as an ultimate
 displacement.
  sub_beta_T <- subset(beta_TENS,
                beta_TENS$Position <= beta_TENS$Position[rowT])
  rowTT <- which.min(abs(sub_beta_T$Force.y - beta_min))
  V_beta_TENS <- sub_beta_T$Position[rowTT]
  row_Fbeta <- which.min(abs(LEC1_TENS$Position - V_beta_TENS))
  F_beta_TENS <- LEC1_TENS$Force[row_Fbeta]

  sub_beta_T2 <- subset(beta_TENS,beta_TENS$Position < V_beta_TENS)
  rowTTT <- which.min(sub_beta_T2$Force.y)
  betaMinValue_T2 <- sub_beta_T2$Force.y[rowTTT]
  if (betaMinValue_T2 < sub_beta_T$Force.y[rowTT]) { # if there are any
 values lower than F_beta_TENS
    sub_beta_T3 <- subset(sub_beta_T2,
                sub_beta_T2$Position <= sub_beta_T2$Position[rowTTT])
    rowTT2 <- which.min(abs(sub_beta_T3$Force.y - beta_min))
    V_beta_TENS <- sub_beta_T3$Position[rowTT2]
    row_Fbeta <- which.min(abs(LEC1_TENS$Position - V_beta_TENS))
    F_beta_TENS <- LEC1_TENS$Force[row_Fbeta]
  }
} else { # if not, V_beta is ignored with NA when deciding ultimate
 displacement.
  V_beta_TENS <- NA
  F_beta_TENS <- NA
}

# PLOTTING BETA.
plot(beta_TENS$Position,beta_TENS$Force.y,type = "l",main = beta,
        xlab = "Displacement (mm)",
        ylab = paste(beta,"_tension"),col = "red")
abline(h=0.75)
text(1,0.75,labels = "0.75", adj = c(1,1))
abline(v=V_P180_TENS)
text(V_P180_TENS,0.65,labels = "80% \nPeak Load",adj = c(1,0))
abline(v=V_beta_TENS)
text(V_beta_TENS,0.65,labels = "V_beta",adj=c(1,0))

result["Degradation Load Compression (N)",i] <- F_beta_COMP
result["Degradation Displ Compression (mm)",i] <- V_beta_COMP
result["Degradation Load Tension (N)",i] <- F_beta_TENS
result["Degradation Displ Tension (mm)",i] <- V_beta_TENS

#-----------------------------------------------------------------------
--
# FINAL ULTIMATE LOAD
# the displacement that occurs first.
Displ_Values_COMP <- c(V_P180_COMP,V_beta_COMP)
Load_Values_COMP <- c(Pl_80_COMP,F_beta_COMP)
mini_COMP <- which.max(Displ_Values_COMP)
Ultimate_displ_COMP <- Displ_Values_COMP[mini_COMP]
Ultimate_force_COMP <- Load_Values_COMP[mini_COMP]

Displ_Values_TENS <- c(V_P180_TENS,V_beta_TENS)
Load_Values_TENS <- c(Pl_80_TENS,F_beta_TENS)
mini_TENS <- which.min(Displ_Values_TENS)
Ultimate_displ_TENS <- Displ_Values_TENS[mini_TENS]
Ultimate_force_TENS <- Load_Values_TENS[mini_TENS]
```

```
  result["Ultimate Force Compression (N)",i] <- Ultimate_force_COMP
  result["Ultimate Displ Compression (mm)",i] <- Ultimate_displ_COMP
  result["Ultimate Force Tension (N)",i] <- Ultimate_force_TENS
  result["Ultimate Displ Tension (mm)",i] <- Ultimate_displ_TENS

  # MAXIMUM LOAD - equal to or lower than the ultimate displacement
  # COMPRESSION
  sub_Fmax_C        <-        subset(LEC1_COMP,LEC1_COMP$Position        >=
Ultimate_displ_COMP)
  rowmaxC <- which.min(sub_Fmax_C$Force)
  F_maxC <- sub_Fmax_C[rowmaxC,2]

  # TENSION
  sub_Fmax_T        <-        subset(LEC1_TENS,LEC1_TENS$Position        <=
Ultimate_displ_TENS)
  rowmaxT <- which.max(sub_Fmax_T$Force)
  F_maxT <- sub_Fmax_T[rowmaxT,2]

#-------------------------------------------------------------------------
--
  # PLOTTING GRAPH
  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
          main = main_title, col = "Orange",cex = 0.2,
          xlab = "Displacement (mm)",ylab = "Force (N)")
  # plotting 1st LEC
  points(LEC1_COMP, cex=0.2,col="Black",type="l")
  points(LEC1_TENS, cex=0.2,col="Black",type="l")
  # plotting 3rd LEC
  points(LEC3_COMP, cex=0.2,col="Blue",type="l")
  points(LEC3_TENS, cex=0.2,col="Blue",type="l")
  # plotting ultimate displ
  abline (v=Ultimate_displ_COMP,col="grey", lty = 2, lwd = 1,
          pch = 3, lend = 0, ljoin = 2)
  abline (v=Ultimate_displ_TENS,col="grey", lty = 2, lwd = 1,
          pch = 3, lend = 0, ljoin = 2)
  text(Ultimate_displ_TENS+0.1,Pl_COMP,labels = "Vu",adj = c(0,0),
          cex = 0.8, col = "grey")
  text(Ultimate_displ_COMP+0.1,Pl_COMP,labels = "Vu",adj = c(0,0),
          cex = 0.8, col = "grey")


#-------------------------------------------------------------------------
--
  # CALCULATING THE EEEP CURVE

  # COMPRESSION
  # First we retrieve the area under the LEC1 curve.
  # AUC is the area under the curve with boundaries from origin to ultimate
    displacement
  xsub <- subset(LEC1_COMP,LEC1_COMP$Position >= Ultimate_displ_COMP)
  x <- xsub$Position
  y <- xsub$Force
  id <- order(x)
  AUC <- sum(diff(x[id])*rollmean(y[id],2))

  # Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
  row3 <- which.min(abs(xsub$Force - (0.1*F_maxC)))
  V10_COMP <- xsub[row3,1]
  F10_COMP <- xsub[row3,2]
  row4 <- which.min(abs(xsub$Force - (0.4*F_maxC)))
  V40_COMP <- xsub[row4,1]
```

```
F40_COMP <- xsub[row4,2]
xcoord_COMP <- c(V10_COMP,V40_COMP)
ycoord_COMP <- c(F10_COMP,F40_COMP)
# plotting LECs and points
main_title2 <- paste("Envelope curve LEC1 and LEC3 Compression - ",
        specimen_name,i,sep="")
plot(LEC1_COMP, type = "l", col = "Black", main = main_title2 )
points(LEC3_COMP, type = "l", col = "Lightblue", cex = 0.2)
points(LEC1_COMP, type = "l", col = "Black")
# plotting ultimate displ and force
abline (v=Ultimate_displ_COMP,col="Black", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
text(Ultimate_displ_COMP-0.1,0,labels = "Vu",adj = c(0,0),
        cex = 0.8, col = "Black")
points(x = V10_COMP, y = F10_COMP,col="Red")
points(x = V40_COMP, y=F40_COMP, col="Red")
fit_COMP <- lm(ycoord_COMP~xcoord_COMP)
abline(fit_COMP, col="black",lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)

# The intercept and slope for line1 is then found and so are the elastic
  stiffness which is equal to the slope.
b <- coef(lm(ycoord_COMP~xcoord_COMP))[1] #intercept
a <- coef(lm(ycoord_COMP~xcoord_COMP))[2] #slope
K_COMP <- a[[1]] # Elastic stiffness [N/mm]
Vu <- Ultimate_displ_COMP

# the equation to find the horisontal line (line2) that gives an area
  equal to the LEC1 area is an quadratic equation.
# The y-solution to the line is then given as plus and minus, referring
  to the quadratic formula:
# for values in 3rd quadrant the correct quadratic formula is the
  equation with plus-sign.
cPlus = a[[1]]*(((b[[1]]/a[[1]]) + Vu) +
        sqrt((-b[[1]]/a[[1]]-Vu)^2                                    -
        4*((1/(2*a[[1]]))*((3*b[[1]]^2/(2*a[[1]])) - AUC)))) #with x=0
        as initial boundary

# if-statement to finds out which m is in our interval and calculating
  the intersection between line1 and line2.
# the intersection is the yield load and displacement.
F_ycPlus <- cPlus
v_ycPlus <- (F_ycPlus - b[[1]])/a[[1]]
if (v_ycPlus >= Ultimate_displ_COMP & F_ycPlus < 0) {
  abline(h = F_ycPlus, col="grey",lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)
  abline(v = v_ycPlus, col="grey",lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)
  result["v_y_COMP",i] <- v_ycPlus
  result["F_y_COMP",i] <- F_ycPlus
  v_y_COMP <- v_ycPlus
  F_y_COMP <- F_ycPlus
}

# TENSION
# First we retrieve the area under the LEC1 curve.

# AUC is the area under the curve with boundaries from origin to ultimate
  displacement
xsub <- subset(LEC1_TENS,LEC1_TENS$Position <= Ultimate_displ_TENS)
x <- xsub$Position
```

```
y <- xsub$Force
id <- order(x)
AUC <- sum(diff(x[id])*rollmean(y[id],2))

# Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
row3 <- which.min(abs(xsub$Force - (0.1*F_maxT)))
V10_TENS <- xsub[row3,1]
F10_TENS <- xsub[row3,2]
row4 <- which.min(abs(xsub$Force - (0.4*F_maxT)))
V40_TENS <- xsub[row4,1]
F40_TENS <- xsub[row4,2]
xcoord_TENS <- c(V10_TENS,V40_TENS)
ycoord_TENS <- c(F10_TENS,F40_TENS)
# plotting points and slope in LEC graph
main_title2 <- paste("Envelope curve LEC1 and LEC3 Tension - ",
        specimen_name,i,sep="")
plot(LEC1_TENS, type = "l", col = "Black", main = main_title2 )
points(LEC3_TENS, type = "l", col = "Lightblue", cex = 0.2)
points(LEC1_TENS, type = "l", col = "Black")
points(x = V10_TENS, y = F10_TENS,col="Red")
points(x = V40_TENS, y=F40_TENS, col="Red")
# plotting ultimate displ and force
fit_TENS <- lm(ycoord_TENS~xcoord_TENS)
abline(fit_TENS, col="black",lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)
abline (v=Ultimate_displ_TENS,col="Black", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
text(Ultimate_displ_TENS+0.1,0,labels = "Vu",adj = c(0,0),
        cex = 0.8, col = "Black")

# The intercept and slope for line1 is then found and so are the elastic
  stiffness which is equal to the slope.
b <- coef(lm(ycoord_TENS~xcoord_TENS))[1] #intercept
a <- coef(lm(ycoord_TENS~xcoord_TENS))[2] #slope
K_TENS <- a[[1]] # Elastic stiffness [N/mm]
Vu <- Ultimate_displ_TENS

# the equation to find the horisontal line (line2) that gives an area
  equal to the LEC1 area is an quadratic equation.
# The y-solution to the line is then given as plus and minus, referring
  to the quadratic formula:
# For values in 1st quadrant the correct quadratic formula is the
  equation with minus-sign.
cMinus = a[[1]]*((Vu+b[[1]]/a[[1]]) -
        sqrt((Vu+b[[1]]/a[[1]])^2    -    4*((b[[1]]/(4*a[[1]]))^2    +
        AUC/(2*a[[1]])))) #with x=0 as initial boundary

# if-statement to finds out which m is in our interval and calculating
  the intersection between line1 and line2.
# the intersection is the yield load and displacement.
F_ycMinus <- cMinus
v_ycMinus <- (cMinus - b[[1]])/a[[1]]
if (v_ycMinus <= Ultimate_displ_TENS & F_ycMinus > 0) {
  abline(h = F_ycMinus, col="grey",lty = 2, lwd = 1, pch = 3, lend = 0,
  ljoin = 2)
  abline(v = v_ycMinus, col="grey",lty = 2, lwd = 1, pch = 3, lend = 0,
  ljoin = 2)
  result["v_y_TENS",i] <- v_ycMinus
  result["F_y_TENS",i] <- F_ycMinus
  v_y_TENS <- v_ycMinus
  F_y_TENS <- F_ycMinus
```

```
}

# DUCTILITY, D_c
# COMPRESSION
D_c_COMP <- Ultimate_displ_COMP/v_y_COMP
result["D_c_COMP",i] <- D_c_COMP
# TENSION
D_c_TENS <- Ultimate_displ_TENS/v_y_TENS
result["D_c_TENS",i] <- D_c_TENS

#---------------------------------------------------------------------
--
# DISSIPATION OF ENERGY - Equivalent viscous damping ratio, v_eq
# https://math.blogoverflow.com
  /2014/06/04/greens-theorem-and-area-of-polygons/
# Creates an area function that is derived from Greens Theorem. If the
  area is negative it is due to a clockwise direction.
area1<-function(X){
  X<-rbind(X,X[1,])
  x<-X[,1]; y<-X[,2]; lx<-length(x)
  sum(((x[2:lx]+x[1:lx-1])*(y[2:lx]-y[1:lx-1])))/2
}

cycles <- c()
# need to subset each cycle to calculate the energy dissipation
cycle075_1st <- subset(my_data,my_data$Time >= 30 & my_data$Time <= 60,
              select = c("Position","Force (smoothed)"))
cycle075_2nd <- subset(my_data,my_data$Time >= 60 & my_data$Time <= 90,
              select = c("Position","Force (smoothed)"))
cycle075_3rd <- subset(my_data,my_data$Time >= 90 & my_data$Time <= 120,
              select = c("Position","Force (smoothed)"))
cycles[[1]] <- cycle075_1st
cycles[[2]] <- cycle075_2nd
cycles[[3]] <- cycle075_3rd

cycle1_1st <- subset(my_data,my_data$Time >= 120
  & my_data$Time <= 160,select = c("Position","Force (smoothed)"))
cycle1_2nd <- subset(my_data,my_data$Time >= 160
  & my_data$Time <= 200,select = c("Position","Force (smoothed)"))
cycle1_3rd <- subset(my_data,my_data$Time >= 200
  & my_data$Time <= 240,select = c("Position","Force (smoothed)"))
cycles[[4]] <- cycle1_1st
cycles[[5]] <- cycle1_2nd
cycles[[6]] <- cycle1_3rd

cycle2_1st <- subset(my_data,my_data$Time >= 240
  & my_data$Time <= 320,select = c("Position","Force (smoothed)"))
cycle2_2nd <- subset(my_data,my_data$Time >= 320
  & my_data$Time <= 400,select = c("Position","Force (smoothed)"))
cycle2_3rd <- subset(my_data,my_data$Time >= 400
  & my_data$Time <= 480,select = c("Position","Force (smoothed)"))
cycles[[7]] <- cycle2_1st
cycles[[8]] <- cycle2_2nd
cycles[[9]] <- cycle2_3rd

cycle4_1st <- subset(my_data,my_data$Time >= 480
  & my_data$Time <= 640,select = c("Position","Force (smoothed)"))
cycle4_2nd <- subset(my_data,my_data$Time >= 640
  & my_data$Time <= 800,select = c("Position","Force (smoothed)"))
cycle4_3rd <- subset(my_data,my_data$Time >= 800
```

```
  & my_data$Time <= 960,select = c("Position","Force (smoothed)"))
cycles[[10]] <- cycle4_1st
cycles[[11]] <- cycle4_2nd
cycles[[12]] <- cycle4_3rd

cycle6_1st <- subset(my_data,my_data$Time >= 960
  & my_data$Time <= 1200,select = c("Position","Force (smoothed)"))
cycle6_2nd <- subset(my_data,my_data$Time >= 1200
  & my_data$Time <= 1440,select = c("Position","Force (smoothed)"))
cycle6_3rd <- subset(my_data,my_data$Time >= 1440
  & my_data$Time <= 1680,select = c("Position","Force (smoothed)"))
cycles[[13]] <- cycle6_1st
cycles[[14]] <- cycle6_2nd
cycles[[15]] <- cycle6_3rd

cycle8_1st <- subset(my_data,my_data$Time >= 1680
  & my_data$Time <= 2000,select = c("Position","Force (smoothed)"))
cycle8_2nd <- subset(my_data,my_data$Time >= 2000
  & my_data$Time <= 2320,select = c("Position","Force (smoothed)"))
cycle8_3rd <- subset(my_data,my_data$Time >= 2320
  & my_data$Time <= 2640,select = c("Position","Force (smoothed)"))
cycles[[16]] <- cycle8_1st
cycles[[17]] <- cycle8_2nd
cycles[[18]] <- cycle8_3rd

cycle10_1st <- subset(my_data,my_data$Time >= 2640
  & my_data$Time <= 3040,select = c("Position","Force (smoothed)"))
cycle10_2nd <- subset(my_data,my_data$Time >= 3040
  & my_data$Time <= 3440,select = c("Position","Force (smoothed)"))
cycle10_3rd <- subset(my_data,my_data$Time >= 3440
  & my_data$Time <= 3840,select = c("Position","Force (smoothed)"))
cycles[[19]] <- cycle10_1st
cycles[[20]] <- cycle10_2nd
cycles[[21]] <- cycle10_3rd

cycle12_1st <- subset(my_data,my_data$Time >= 3840
  & my_data$Time <= 4320,select = c("Position","Force (smoothed)"))
cycle12_2nd <- subset(my_data,my_data$Time >= 4320
  & my_data$Time <= 4800,select = c("Position","Force (smoothed)"))
cycle12_3rd <- subset(my_data,my_data$Time >= 4800
  & my_data$Time <= 5280,select = c("Position","Force (smoothed)"))
cycles[[22]] <- cycle12_1st
cycles[[23]] <- cycle12_2nd
cycles[[24]] <- cycle12_3rd

cycle14_1st <- subset(my_data,my_data$Time >= 5280
  & my_data$Time <= 5840,select = c("Position","Force (smoothed)"))
cycle14_2nd <- subset(my_data,my_data$Time >= 5840
  & my_data$Time <= 6400,select = c("Position","Force (smoothed)"))
cycle14_3rd <- subset(my_data,my_data$Time >= 6400
  & my_data$Time <= 6960,select = c("Position","Force (smoothed)"))
cycles[[25]] <- cycle14_1st
cycles[[26]] <- cycle14_2nd
cycles[[27]] <- cycle14_3rd

names(cycles) <- c("cycle075_1st","cycle075_2nd","cycle075_3rd",
              "cycle1_1st","cycle1_2nd","cycle1_3rd","cycle2_1st",
              "cycle2_2nd","cycle2_3rd","cycle4_1st","cycle4_2nd",
              "cycle4_3rd","cycle6_1st","cycle6_2nd","cycle6_3rd",
              "cycle8_1st","cycle8_2nd","cycle8_3rd","cycle10_1st",
              "cycle10_2nd","cycle10_3rd","cycle12_1st","cycle12_2nd",
```

```
                "cycle12_3rd","cycle14_1st","cycle14_2nd","cycle14_3rd")
  # checking cycle12 and cycle14
  #If the last list in cycles is empty, then delete that list. Stop when
    the last element is not zero.
  while (nrow(cycles[[length(cycles)]]) == 0) {
    cycles[[length(cycles)]] <- NULL
  }

  cycle_values <- as.data.frame(matrix(nrow = 3,ncol = length(cycles)))
  colnames(cycle_values) <- c(names(cycles))
  rownames(cycle_values) <- c("Ed (J)","Ep (J)","v_eq")
  for (k in 1:length(cycles)) {
    Ed <- area1(cycles[[k]])
    if (Ed < 0) { # If the area is negative, the value is correct, but the
    direction was clockwise
      Ed <- -Ed
    }
    cycle_values["Ed (J)",k] <- Ed/1000

    rowEp <- which.max(cycles[[k]]$Position)
    xx <- c(0,cycles[[k]]$Position[rowEp],cycles[[k]]$Position[rowEp])
    yy <- c(0,0,cycles[[k]]$`Force (smoothed)`[rowEp])
    xy <- cbind(xx,yy)
    Ep <- area1(xy)
    cycle_values["Ep (J)",k] <- Ep/1000

    v_eqCycle <- Ed/(Ep*4*pi)
    cycle_values["v_eq",k] <- v_eqCycle
  }

  # Stores all the viscous damping ratio for each cycle in a list.
  v_eq_values[[name_specimen]] <- cycle_values


#------------------------------------------------------------------------
--
  # DESIGN STRENGTH DEGRADATION FACTOR, Beta_sd
  # The strength degradation factor shall ble calculated for values of
    displacement lower than the ultimate displacement.
  # Need to subset the values that are lower than Vu.

  # COMPRESSION
  beta_sub <- subset(beta_COMP,beta_COMP$Position > Ultimate_displ_COMP)
  row5 <- which.min(beta_sub$Force.y)
  beta_sd_COMP <- beta_sub$Force.y[row5] # minimum value of beta in this
    interval
  # If beta_sd is lower than beta_min, then beta_sd should be set equal to
    beta_min. EN 12512 - 3.18 (V.20180410)
  if (beta_sd_COMP < beta_min) {
    beta_sd_COMP <- beta_min
  }

  # TENSION
  beta_sub_T <- subset(beta_TENS,beta_TENS$Position < Ultimate_displ_TENS)
  row6 <- which.min(beta_sub_T$Force.y)
  beta_sd_TENS <- beta_sub_T$Force.y[row6] # minimum value of beta in this
    interval
  if (beta_sd_TENS < beta_min) {
    beta_sd_TENS <- beta_min
  }
```

```
  result["beta_sd_COMP",i] <- beta_sd_COMP
  result["beta_sd_TENS",i] <- beta_sd_TENS

  #-----------------------------------------------------------------------
--
  # FINAL RESULTS TABLE

  # Compression
  Final_resultsCOMP[i,"Peak Load (kN)"] <- Pl_COMP/1000
  Final_resultsCOMP[i,"Displacement at Peak Load (mm)"] <- V_Pl_COMP
  Final_resultsCOMP[i,"Maximum Load (kN)"] <- F_maxC/1000
  Final_resultsCOMP[i,"Ultimate load (kN)"] <- Ultimate_force_COMP/1000
  Final_resultsCOMP[i,"Ultimate displacement (mm)"] <- Ultimate_displ_COMP
  Final_resultsCOMP[i,paste0("Design Strength Degradation factor - ",
              beta,"_sd")] <- beta_sd_COMP
  Final_resultsCOMP[i,"Yield slip (mm) - v_y"] <- v_y_COMP
  Final_resultsCOMP[i,"Yield Load (kN) - F_y"] <- F_y_COMP/1000
  Final_resultsCOMP[i,"Slip modulus - k_ser (kN/mm)"] <- K_COMP/1000
  Final_resultsCOMP[i,"Static ductility - D"] <- D_c_COMP

  # Tension
  Final_resultsTENS[i,"Peak Load (kN)"] <- Pl_TENS/1000
  Final_resultsTENS[i,"Displacement at Peak Load (mm)"] <- V_Pl_TENS
  Final_resultsTENS[i,"Maximum Load (kN)"] <- F_maxT/1000
  Final_resultsTENS[i,"Ultimate Load (kN)"] <- Ultimate_force_TENS/1000
  Final_resultsTENS[i,"Ultimate displacement (mm)"] <- Ultimate_displ_TENS
  Final_resultsTENS[i,paste0("Design Strength Degradation Factor - ",
              beta,"_sd")] <- beta_sd_TENS
  Final_resultsTENS[i,"Yield slip (mm) - v_y"] <- v_y_TENS
  Final_resultsTENS[i,"Yield Load (kN) - F_y"] <- F_y_TENS/1000
  Final_resultsTENS[i,"Slip modulus - k_ser (kN/mm)"] <- K_TENS/1000
  Final_resultsTENS[i,"Static ductility - D"] <- D_c_TENS
}
#-----------------------------------------------------------------------
--
# MEAN VALUES AND STANDARD DEVIATIONS
for (o in 1:ncol(Final_resultsCOMP)) {
  Final_resultsCOMP["Mean Values",o] <-
        mean(Final_resultsCOMP[1:length(myFiles2),o], na.rm = TRUE)
  Final_resultsCOMP["Standard Deviations",o] <-
        sd(Final_resultsCOMP[(1:length(myFiles2)),o],na.rm=TRUE)
}

for (o in 1:ncol(Final_resultsTENS)) {
  Final_resultsTENS["Mean Values",o] <-
        mean(Final_resultsTENS[1:length(myFiles2),o], na.rm = TRUE)
  Final_resultsTENS["Standard Deviations",o] <-
        sd(Final_resultsTENS[(1:length(myFiles2)),o],na.rm=TRUE)
}
#-----------------------------------------------------------------------
--
# Saving viscous damping for each specimen
for (f in 1:length(myFiles)) {
  filename <- paste("Viscous damping ",specimen_name,"-",f,".xlsx",sep="")
  v_eq_specimen <- v_eq_values[[f]]
  is.num <- sapply(v_eq_specimen, is.numeric)
  v_eq_specimen[is.num] <- lapply(v_eq_specimen[is.num], round, 3)
  write.xlsx(v_eq_specimen,file = filename,row.names = T,col.names = T)
}

# Restricts decimals to three numbers in the tables of interest.
```

```
is.num <- sapply(Final_resultsCOMP, is.numeric)
Final_resultsCOMP[is.num] <- lapply(Final_resultsCOMP[is.num], round, 2)
is.num <- sapply(Final_resultsTENS, is.numeric)
Final_resultsTENS[is.num] <- lapply(Final_resultsTENS[is.num], round, 2)
is.num <- sapply(result, is.numeric)
result[is.num] <- lapply(result[is.num], round, 3)

# # SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_resultsCOMP, file = "Final_resultsCOMP.xlsx",
          row.names = T, col.names = T)
write.xlsx(Final_resultsTENS, file = "Final_resultsTENS.xlsx",
          row.names = T, col.names = T)
```

3 Study 2

3.1 NS-ISO 6891 (1991)

```
# Study 2 Monotonic load test - Follows NS-ISO 6891 (1991) calculations.
# Graphs, calculations and results.

#PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("RColorBrewer", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "HNS"
study <- "Study 2"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA ANALYZING\\",
          study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master  2018\\3  -  DATA  ANALYZING\\",  study  ,"\\",
          specimen_name,sep=""))
all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(".csv","",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=8))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], skip=11, nlines = 1,
          what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=12, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=12,sep=";",dec=",",header=TRUE)
  colnames(dat) <- paste(header,header2,sep="")
  result["max load",i] <- max(dat$`Load(N)`)
  row_nr <- which.max(dat$`Load(N)`)
  result["max displ",i] <- dat[row_nr,2]
  all_data[[i]] <- dat
```

```
}
myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=10,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#-------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
par(mfrow = c(2,2))
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[[name_specimen]])
  specimen$lopende_kraft<-NA
  specimen[specimen < 0] <- NA
  for (p in 5:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<-
      mean(specimen[intervall[1]:intervall[2],3])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- specimen[2]
  my_data[2] <- specimen[4]
  colnames(my_data)<-c("Position","Force (smoothed)")
  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
   main = main_title, xlab = "Displacement (mm)",ylab = "Force (N)")

  # -------------------------------------------------------------------
# FAILURE IN GRAPH
  # By calculating the diff of each datapoint, verifying a large outlier
     and calculating the percentage of change in the drop - big drop
   yields failure point.
  my_data_subset <- subset(my_data,my_data$Position > 2.5
   & my_data$Position <= 15,select = c("Position","Force (smoothed)"))
  differ <- diff(my_data_subset$`Force (smoothed)`)/
          diff(my_data_subset$Position)
  differ2 <- differ[-1] # exluding the first value, because it is infinite.
  rowdiffer <- which.min(differ)
  # subset
  x <- my_data_subset[(rowdiffer-10):(rowdiffer+40),]
  xmax <- which.max(x$`Force (smoothed)`)
  xmin <- which.min(x$`Force (smoothed)`)
  V1 <- x[xmax,2]
  V2 <- x[xmin,2]
  diff_perce <- abs(V1-V2)/((V1+V2)/2)*100
  # if the difference percentage is bigger than 8% and below 15 mm
   displacemen,
  # we classify the drop as a failure and the maximum load
  F_maxDispl <- 100 # imaginary number
  row_max <- which.max(my_data$`Force (smoothed)`)
  V_max <- my_data$Position[row_max]
  if (V_max < 15 & V_max < F_maxDispl) { #If the max value is lower than
   x_failure it is the max value
    F_max <- my_data$`Force (smoothed)`[row_max]
    result["F_max",i] <- F_max
    result["V_max",i] <- V_max
    abline (h=F_max,col="blue", lty = 2, lwd = 1, pch = 3,
```

```
        lend = 0, ljoin = 2)
  abline (v=V_max,col="blue", lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)
  F_maxAdj <- F_max
  F_maxDispl<-V_max
}
if (diff_perce > 8 & my_data_subset[rowdiffer,1] < F_maxDispl){
  F_failure <- V1
  x_failure <- my_data_subset[rowdiffer,1]
  abline (h=F_failure,col="blue", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
  abline (v=-x_failure,col="blue", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
  result["F_failure(N)",i] <- F_failure
  result["Displ_failure(mm)",i] <- x_failure
  F_maxAdj <- F_failure
  F_maxDispl <- x_failure
  }
row_v15 <- which.min(abs(my_data$Position-15.000))
v_max15 <- my_data[row_v15,1]
if (v_max15 < F_maxDispl){ # If 15 mm slip is the lowest value, it is
the
   max value.
  F_max15 <- my_data[row_v15,2]
  result["F_max15",i] <- F_max15
  result["v_max15",i] <- v_max15
  abline (h=F_max15,col="blue", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
  abline (v=15,col="blue", lty = 2, lwd = 1, pch = 3,
        lend = 0, ljoin = 2)
  F_maxAdj <- F_max15
  F_maxDispl <- v_max15
  }

#------------------------------------------------------------------------
--
# CALCULATING
subs04 <- subset(specimen,specimen$`Time(sec)`< 1.1*60)
row04 <- which.min(abs(subs04$lopende_kraft - F_maxAdj*0.4))
row01 <- which.min(abs(subs04$lopende_kraft - F_maxAdj*0.1))
v_04 <- subs04$`Extension(mm)`[row04]
v_01 <- subs04$`Extension(mm)`[row01]

v_imod <- (4/3)*(v_04 - v_01)

K_ser <- 0.4*(F_maxAdj/v_imod)


#------------------------------------------------------------------------
--
# FINDING F_y, v_y
subs14 <- subset(specimen,specimen$`Time(sec)`> 1.1*60
        & specimen$`Time(sec)`< 2.5*60
        & specimen$`Extension(mm)` <= F_maxDispl)
row14 <- which.min(abs(subs14$`lopende_kraft` - 0.4*F_maxAdj))
v14 <- subs14$`Extension(mm)`[row14]
row11 <- which.min(abs(subs14$lopende_kraft - 0.1*F_maxAdj))
v11 <- subs14$`Extension(mm)`[row11]
subs24 <- subset(specimen,specimen$`Time(sec)`> 2.5*60
        & specimen$`Extension(mm)` <= F_maxDispl)
row24 <- which.min(abs(subs24$lopende_kraft - 0.4*F_maxAdj))
```

```
  v24 <- subs24$`Extension(mm)`[row24]
  row21 <- which.min(abs(subs24$lopende_kraft - 0.1*F_maxAdj))
  v21 <- subs24$`Extension(mm)`[row21]

  v_y <- (2/3)*(v14 +v24 - v11 - v21)

  # ---------------------------------------------------------------------
--
  # FINAL RESULT FOR TERMOWOOD REPORT
  # Constructing a final table with results
  Final_results[i,"Maximum Force (kN)"] <- F_maxAdj/1000
  Final_results[i,"Displacement at max force (mm)"] <- F_maxDispl
  Final_results[i,"Yield slip (mm)"] <- v_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- K_ser/1000
}

# -----------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations
for (o in 1:4) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
          sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - NS-ISO 6891(1991).xlsx",
row.names = T, col.names = T)
```

3.2 NS-EN 12512 (2002)

```
# Study 2 Monotonic load test - Follows NS-EN 12512 (2002) calculations.
# Graphs, calculations and results.

# PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "HNS"
study <- "Study 2"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA ANALYZING\\",
          study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))
all_data <- c()
```

```
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(".csv","",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], skip=11, nlines = 1,
          what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=12, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=12,sep=";",dec=",",header=TRUE)
  colnames(dat) <- paste(header,header2,sep="")
  result["max load",i] <- max(dat$`Load(N)`)
  row_nr <- which.max(dat$`Load(N)`)
  result["max displ",i] <- dat[row_nr,2]
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=length(myFiles)+2,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#-------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[name_specimen])
  specimen$lopende_kraft<-NA
  for (p in 5:(nrow(specimen)-1)){

    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<
   mean(specimen[intervall[1]:intervall[2],3])

  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- specimen[2] # changes negative values to positive
  my_data[2] <- specimen[4] # force
  colnames(my_data)<-c("Position","Force (smoothed)")
  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
          main = main_title,xlab = "Displacement (mm)",
          ylab = "Force (N)")

  #-----------------------------------------------------------------------
--
  # MAX LOAD
  row_Fmax <- which.max(my_data$`Force (smoothed)`)
  F_max <- my_data$`Force (smoothed)`[row_Fmax]
  V_Fmax <- my_data$Position[row_Fmax]

  # ULTIMATE LOAD CASES:
  # Vu = min{failure,80%Fmax,30mm}
  # Identifying failure with locator(). ESC key if there are none.
  failure_coord <- as.data.frame(matrix(nrow = length(myFiles),ncol = 2))
  rownames(failure_coord) <- myFiles
```

```r
colnames(failure_coord) <- c("x","y")
failure_locator <- locator(1)
if (!is.null(failure_locator)) {
  failure_coord[name_specimen,"x"] <- failure_locator$x
  failure_coord[name_specimen,"y"] <- failure_locator$y
} else {
  failure_locator$x <- NA
  failure_locator$y <- NA
}
result["Failure Load (N)",i] <- failure_locator$y
result["Failure displacement (mm)",i] <- failure_locator$x

# 80%Fmax
F_80 <- 0.8*F_max
subs <- subset(my_data,my_data$Position > V_Fmax)
row <- which.min(abs(subs$Force - F_80))
V_F80 <- subs$Position[row]
result["80% Max Load (N)",i] <- F_80
result["Displ at 80% Max Load (mm)",i] <- V_F80

# Slip at 30 mm
row_30 <- which.min(abs(my_data$Position - 30))
F_30mm <- my_data$`Force (smoothed)`[row_30]
V_30mm <- my_data$Position[row_30]
result["Force Load at 30 mm (N)",i] <- F_30mm
result["displ 30mm (mm)",i] <- V_30mm

# FINAL ULTIMATE LOAD
# the displacement that occurs first.
Displ_Values <- c(failure_locator$x, V_F80, V_30mm)
Load_Values <- c(failure_locator$y, F_80,F_30mm)
minimum <- which.min(Displ_Values)
Ultimate_displ <- Displ_Values[minimum]
Ultimate_force <- Load_Values[minimum]

result["Ultimate Force (N)",i] <- Ultimate_force
result["Ultimate Displ (mm)",i] <- Ultimate_displ


# plotting ultimate displ
abline (v=Ultimate_displ,col="grey", lty = 2, lwd = 1,
        pch = 3, lend = 0, ljoin = 2)
text(Ultimate_displ+0.1,Ultimate_force,labels = "Vu",adj = c(0,0),
        cex = 0.8, col = "grey")

#----------------------------------------------------------------------
--
# CALCULATING THE YIELD LOAD/SLIP
# Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
xsub <- subset(my_data,my_data$Position < V_Fmax)
row3 <- which.min(abs(xsub$Force - (0.1*F_max)))
V10 <- xsub[row3,1]
F10 <- xsub[row3,2]
row4 <- which.min(abs(xsub$Force - (0.4*F_max)))
V40 <- xsub[row4,1]
F40 <- xsub[row4,2]
xcoord <- c(V10,V40)
ycoord <- c(F10,F40)
# plotting points and slope in the graph
points(x = V10, y = F10,col="Red")
points(x = V40, y=F40, col="Red")
```

```
  fit <- lm(ycoord~xcoord)
  abline(fit, col="black",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)

  # The intercept and slope for line1 is then found and so are the elastic
    stiffness which is equal to the slope.
  b <- coef(lm(ycoord~xcoord))[1] #intercept
  a <- coef(lm(ycoord~xcoord))[2] #slope
  Vu <- Ultimate_displ

  # YIELD LOAD
  # The yield load is the interception between line1 and a second line2.
  # Line 2 angle is 1/6 of line1s' slope and is thouching the graph.
  # Firstly, we find the slope of line2 and secondly, we found the point
in
    the graph that has the same slope.
  line2_slope <- (1/6)*a
  # finding the slope in graph that is similar to line2_slope
  slope_data <- as.data.frame(matrix(nrow=0,ncol=6))
  colnames(slope_data) <- c("Intersept","Slope", "x1","x2","y1","y2" )
  sub_slope <- subset(my_data,my_data$Position > V40
          & my_data$Position < Ultimate_displ)
  for (m in 100:nrow(sub_slope)){# calculating slope with a step of 100
    vertices, because the rawdata alows it.
    x_graph <- c(sub_slope[m-99,1],sub_slope[m,1])
    y_graph <- c(sub_slope[m-99,2],sub_slope[m,2])
    coef_inter <- coef(lm(y_graph~x_graph))[1]
    coef_slope <- coef(lm(y_graph~x_graph))[2]
    slope_data[m,1] <- coef_inter
    slope_data[m,2] <- coef_slope
    slope_data[m,3] <- x_graph[1]
    slope_data[m,4] <- x_graph[2]
    slope_data[m,5] <- y_graph[1]
    slope_data[m,6] <- y_graph[2]
  }
  row_y <- which.min(abs(slope_data[,2]-line2_slope))
  line2_data <- slope_data[row_y,]
  x2_coord <- c(slope_data[row_y,3],slope_data[row_y,4])
  y2_coord <- c(slope_data[row_y,5],slope_data[row_y,6])
  line2_fit <- lm(y2_coord~x2_coord)
  abline(line2_fit, col="gray",lty = 2, lwd = 1, pch = 3,
          lend = 0, ljoin = 2)

  # FINDING F_y, v_y and k_ser
  # Need to find the intersection between line1 and line2
  # Algebraically calculated using intersection and slope from line1 and
2
  V_y <- (line2_data[1,1]-b)/(a-line2_data[1,2])
  F_y <- b+a*V_y
  result["V_y",i] <- V_y
  result["F_y",i] <- F_y
  abline (h=F_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
          lend = 0, ljoin = 2)
  abline (v=V_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
          lend = 0, ljoin = 2)

  k_ser <- F_y/V_y
  result["k_ser",i]<- k_ser

  # Ductility
  D <- Ultimate_displ/V_y
```

```
  # ---------------------------------------------------------------------
--
  # FINAL RESULT
  # Constructing a final table with results
  Final_results[i,"maximum load (kN)"] <- F_max/1000
  Final_results[i,"Ultimate Load (kN)"] <- Ultimate_force/1000
  Final_results[i,"Ultimate displacement (mm)"] <- Ultimate_displ
  Final_results[i,"Yield Load (kN)"] <- F_y/1000
  Final_results[i,"Yield slip (mm)"] <- V_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- k_ser/1000
  Final_results[i,"Ductility"] <- D

}


# ----------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations
for (o in 1:length(Final_results)) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
          sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - EN12512(2002).xlsx",
row.names = T, col.names = T)
```

3.3 EN 12512 (2018) Draft Version 20180410

```
# Study 2 Monotonic load test
# EN 12512 (2018) Draft Proposal version 20180410

# PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("zoo")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "HNS"
study <- "Study 2"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA ANALYZING\\",
          study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))
```

```
all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(".csv","",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], skip=11, nlines = 1,
          what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=12, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=12,sep=";",dec=",",header=TRUE)
  colnames(dat) <- paste(header,header2,sep="")
  result["max load",i] <- max(dat$`Load(N)`)
  row_nr <- which.max(dat$`Load(N)`)
  result["max displ",i] <- dat[row_nr,2]
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=length(myFiles)+2,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#-------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[name_specimen])
  specimen$lopende_kraft<-NA
  for (p in 5:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<-
      mean(specimen[intervall[1]:intervall[2],3])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- specimen[2] # changes negative values to positive
  my_data[2] <- specimen[4] # force
  colnames(my_data)<-c("Position","Force (smoothed)")

  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
      main = main_title,xlab = "Displacement (mm)",ylab = "Force (N)")

#-------------------------------------------------------------------------
  # CALCULATIONS
  # Peak load, Pl - max load reached during monotonic test
  row_Pl <- which.max(my_data$`Force (smoothed)`)
  Pl <- my_data$`Force (smoothed)`[row_Pl]
  V_Pl <- my_data$Position[row_Pl]

  # ULTIMATE LOAD CASES:
  # Vu - min{failure,80%Pl,30mm}
  # Identifying failure with locator(). ESC key if there are none.
  failure_locator <- locator(1)
  if (length(failure_locator) == 0) { # if there are located no failures
    failure_locator$x <- NA
```

```
    failure_locator$y <- NA
  }
  result["Failure Load (N)",i] <- failure_locator$y
  result["Failure displacement (mm)",i] <- failure_locator$x

  # 80%Plmax - Load/displacement after peak load.
  Pl_80 <- 0.8*Pl
  subs <- subset(my_data,my_data$Position > V_Pl)
  row <- which.min(abs(subs$Force - Pl_80))
  V_Pl80 <- subs$Position[row]
  result["80% Peak Load (N)",i] <- Pl_80
  result["Displ at 80% Peak Load (mm)",i] <- V_Pl80

  # Slip at 30 mm
  row_30 <- which.min(abs(my_data$Position - 30))
  F_30mm <- my_data$`Force (smoothed)`[row_30]
  V_30mm <- my_data$Position[row_30]
  result["Force Load at 30 mm (N)",i] <- F_30mm
  result["displ 30mm (mm)",i] <- V_30mm

  # FINAL ULTIMATE LOAD
  # the displacement that occurs first.
  Displ_Values <- c(failure_locator$x, V_Pl80, V_30mm)
  Load_Values <- c(failure_locator$y, Pl_80,F_30mm)
  minimum <- which.min(Displ_Values)
  Ultimate_displ <- Displ_Values[minimum]
  Ultimate_force <- Load_Values[minimum]

  result["Ultimate Force (N)",i] <- Ultimate_force
  result["Ultimate Displ (mm)",i] <- Ultimate_displ

  # MAX LOAD
  # max load lower than or equal to ultimate displacement
  sub_Fmax <- subset(my_data,my_data$Position <= Ultimate_displ)
  rowmax <- which.max(sub_Fmax$Force)
  F_max <- sub_Fmax[rowmax,2]

  # plotting ultimate displ
abline (v=Ultimate_displ,col="grey", lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  text(Ultimate_displ+0.1,Ultimate_force,labels = "Vu",adj = c(0,0),
      cex = 0.8, col = "grey")

  #----------------------------------------------------------------------
--
  # CALCULATING THE EEEP CURVE
  # First, we retrieve the area under the curve.
  # AUC is the area under the curve with boundaries from origin to ultimate
    displacement
  xsub <- subset(my_data,my_data$Position <= Ultimate_displ)
  x <- xsub$Position
  y <- xsub$Force
  id <- order(x)
  AUC <- sum(diff(x[id])*rollmean(y[id],2))

  # Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
  row3 <- which.min(abs(xsub$Force - (0.1*F_max)))
  V10 <- xsub[row3,1]
  F10 <- xsub[row3,2]
  row4 <- which.min(abs(xsub$Force - (0.4*F_max)))
  V40 <- xsub[row4,1]
```

```
  F40 <- xsub[row4,2]
  xcoord <- c(V10,V40)
  ycoord <- c(F10,F40)
  # plotting points and slope in LEC graph
  points(x = V10, y = F10,col="Red")
  points(x = V40, y=F40, col="Red")
  fit <- lm(ycoord~xcoord)
  abline(fit, col="black",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)

  # The intercept and slope for line1 is then found and so are the elastic
    stiffness which is equal to the slope.
  b <- coef(lm(ycoord~xcoord))[1] #intercept
  a <- coef(lm(ycoord~xcoord))[2] #slope
  K <- a[[1]] # Elastic stiffness [N/mm]
  Vu <- xsub[nrow(xsub),1] # ultimate displacement retrieved from subset
of
    the graph

  # the equation to find the horisontal line (line2) that gives an area
    equal to the LEC1 area is an quadratic equation.
  # The y-solution to the line is then given as plus and minus, referring
    to the quadratic formula:

  mMinus = a[[1]]*((Vu+b[[1]]/a[[1]]) -
          sqrt((Vu+b[[1]]/a[[1]])^2    -    4*((b[[1]]/(4*a[[1]]))^2    +
          AUC/(2*a[[1]])))) #with x=0 as initial boundary

  # calculating the intersection between line1 and line2.
  # the intersection is the yield load and displacement.
  F_ymMinus <- mMinus
  v_ymMinus <- (mMinus - b[[1]])/a[[1]]
  abline(h = F_ymMinus, col="grey",lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  abline(v = v_ymMinus, col="grey",lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  result["v_y",i] <- v_ymMinus
  result["F_y",i] <- F_ymMinus
  v_y <- v_ymMinus
  F_y <- F_ymMinus

  # Ductility
  D <- Ultimate_displ/v_y

  # ---------------------------------------------------------------------
--
  # FINAL RESULT
  # Constructing a final table with results
  Final_results[i,"Peak Load (kN)"] <- Pl/1000
  Final_results[i,"maximum load (kN)"] <- F_max/1000
  Final_results[i,"Ultimate Load (kN)"] <- Ultimate_force/1000
  Final_results[i,"Ultimate displacement (mm)"] <- Ultimate_displ
  Final_results[i,"Yield Load (kN)"] <- F_y/1000
  Final_results[i,"Yield slip (mm)"] <- v_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- K/1000
  Final_results[i,"Ductility"] <- D
}

# -----------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations
```

```
for (o in 1:length(Final_results)) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
          sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - EN12512(2018).xlsx",
row.names = T, col.names = T)
```

4 STUDY 3

4.1 NS-ISO 6891 (1991)

```
# Study 3 Monotonic load test - Follows NS-ISO 6891 (1991) calculations.
# Graphs, calculations and results.

#PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("RColorBrewer", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "WF"
study <- "Study 3"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA ANALYZING\\",
          study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))

all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(".csv","",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], skip=10, nlines = 1,
          what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=11, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=11,sep=";",dec=",",header=TRUE)
  colnames(dat) <- paste(header,header2,sep="")
  result["max load",i] <- max(dat$`Load(N)`)
  row_nr <- which.max(dat$`Load(N)`)
```

```
  result["max displ",i] <- dat[row_nr,2]
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=length(myFiles)+2,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#-------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[[name_specimen]])
  specimen$lopende_kraft<-NA
  for (p in 5:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<-
          mean(specimen[intervall[1]:intervall[2],3])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- -specimen[2]
  my_data[2] <- -specimen[4]
  colnames(my_data)<-c("Position","Force (smoothed)")
  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
   main = main_title, xlab = "Displacement (mm)",ylab = "Force (N)")

  # -------------------------------------------------------------------
# FAILURE IN GRAPH
  # By calculating the diff of each datapoint, verifying a large outlier
      and calculating the percentage of change in the drop - big drop
   yields failure point.
  my_data_subset <- subset(my_data,my_data$Position > 2.5
                 & my_data$Position <= 15,
                 select = c("Position","Force (smoothed)"))
  differ <- diff(my_data_subset$`Force (smoothed)`)/
          diff(my_data_subset$Position)
  differ2 <- differ[-1] # exluding the first value, because it is infinite.
  rowdiffer <- which.min(differ)
  # subset
  x <- my_data_subset[(rowdiffer-10):(rowdiffer+40),]
  xmax <- which.max(x$`Force (smoothed)`)
  xmin <- which.min(x$`Force (smoothed)`)
  V1 <- x[xmax,2]
  V2 <- x[xmin,2]
  diff_perce <- abs(V1-V2)/((V1+V2)/2)*100
  # if the difference percentage is bigger than 8% and below 15 mm
   displacemen,
  # we classify the drop as a failure and the maximum load
  F_maxDispl <- 100 # imaginary number
  row_max <- which.max(my_data$`Force (smoothed)`)
  V_max <- my_data$Position[row_max]
  if (V_max < 15 & V_max < F_maxDispl) { #If the max value is lower than
   x_failure it is the max value
    F_max <- my_data$`Force (smoothed)`[row_max]
    result["F_max",i] <- F_max
```

```
    result["V_max",i] <- V_max
    abline (h=F_max,col="blue", lty = 2, lwd = 1, pch = 3,
   lend = 0, ljoin = 2)
    abline (v=V_max,col="blue", lty = 2, lwd = 1, pch = 3,
   lend = 0, ljoin = 2)
    F_maxAdj <- F_max
    F_maxDispl<-V_max
  }
  if (diff_perce > 8 & my_data_subset[rowdiffer,1] < F_maxDispl){
    F_failure <- V1
    x_failure <- my_data_subset[rowdiffer,1]
    abline (h=F_failure,col="blue", lty = 2, lwd = 1, pch = 3,
   lend = 0, ljoin = 2)
    abline (v=-x_failure,col="blue", lty = 2, lwd = 1, pch = 3,
   lend = 0, ljoin = 2)
    result["F_failure(N)",i] <- F_failure
    result["Displ_failure(mm)",i] <- x_failure
    F_maxAdj <- F_failure
    F_maxDispl <- x_failure
    }
  row_v15 <- which.min(abs(my_data$Position-15.000))
  v_max15 <- my_data[row_v15,1]
  if (v_max15 < F_maxDispl){ # If 15 mm slip is the lowest value, it is
the
   max value.
    F_max15 <- my_data[row_v15,2]
    result["F_max15",i] <- F_max15
    result["v_max15",i] <- v_max15
    abline (h=F_max15,col="blue", lty = 2, lwd = 1, pch = 3,
   lend = 0, ljoin = 2)
    abline (v=15,col="blue", lty = 2, lwd = 1, pch = 3, lend = 0,
   ljoin = 2)
    F_maxAdj <- F_max15
    F_maxDispl <- v_max15
    }

  #-----------------------------------------------------------------------
--
  # CALCULATING
  subs04 <- subset(specimen,specimen$`Time(sec)`< 1.1*60)
  row04 <- which.min(abs(-subs04$lopende_kraft - F_maxAdj*0.4))
  row01 <- which.min(abs(-subs04$lopende_kraft - F_maxAdj*0.1))
  v_04 <- -subs04$`Extension(mm)`[row04]
  v_01 <- -subs04$`Extension(mm)`[row01]

  v_imod <- (4/3)*(v_04 - v_01)

  K_ser <- 0.4*(F_maxAdj/v_imod)


  #-----------------------------------------------------------------------
--
  # FINDING F_y, v_y and k_ser
  subs14 <- subset(specimen,specimen$`Time(sec)`> 1.1*60
   & specimen$`Time(sec)`< 2.5*60)
  row14 <- which.min(abs(-subs14$`lopende_kraft` - 0.4*F_maxAdj))
  v14 <- -subs14$`Extension(mm)`[row14]
  row11 <- which.min(abs(-subs14$lopende_kraft - 0.1*F_maxAdj))
  v11 <- -subs14$`Extension(mm)`[row11]
  subs24 <- subset(specimen,specimen$`Time(sec)`> 2.5*60)
  row24 <- which.min(abs(-subs24$lopende_kraft - 0.4*F_maxAdj))
```

```
  v24 <- -subs24$`Extension(mm)`[row24]
  row21 <- which.min(abs(-subs24$lopende_kraft - 0.1*F_maxAdj))
  v21 <- -subs24$`Extension(mm)`[row21]

  v_y <- (2/3)*(v14 +v24 - v11 - v21)

  # -----------------------------------------------------------------------
--
  # FINAL RESULT FOR TERMOWOOD REPORT
  # Constructing a final table with results
  Final_results[i,"Maximum Force (kN)"] <- F_maxAdj/1000
  Final_results[i,"Displacement at max force (mm)"] <- F_maxDispl
  Final_results[i,"Yield slip (mm)"] <- v_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- K_ser/1000
}

# -------------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations
for (o in 1:4) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
        sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - NS-ISO 6891(1991).xlsx",
row.names = T, col.names = T)
```

4.2 NS-EN 12512 (2002)

```
# Study 3 Monotonic load test - Follows NS-EN 12512 (2002) calculations.
# Graphs, calculations and results.

# PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "WF"
study <- "Study 3"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
         University of Life Sciences\\Master 2018\\3 - DATA ANALYZING\\",
         study ,"\\", specimen_name, sep=""),
         pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
         Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
         specimen_name,sep=""))

all_data <- c()
```

```
myFiles2 <- sub("-","",x = myFiles)
myFiles2 <- sub(".csv","",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], skip=10, nlines = 1,
          what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=11, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=11,sep=";",dec=",",header=TRUE)
  colnames(dat) <- paste(header,header2,sep="")
  result["max load",i] <- max(dat$`Load(N)`)
  row_nr <- which.max(dat$`Load(N)`)
  result["max displ",i] <- dat[row_nr,2]
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=length(myFiles)+2,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[name_specimen])
  specimen$lopende_kraft<-NA
  for (p in 5:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<-
      mean(specimen[intervall[1]:intervall[2],3])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- -specimen[2] # changes negative values to positive
  my_data[2] <- -specimen[4] # force
  colnames(my_data)<-c("Position","Force (smoothed)")

  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
      main = main_title,xlab = "Displacement (mm)",ylab = "Force (N)")

  #-----------------------------------------------------------------------
--
  # MAX LOAD
  row_Fmax <- which.max(my_data$`Force (smoothed)`)
  F_max <- my_data$`Force (smoothed)`[row_Fmax]
  V_Fmax <- my_data$Position[row_Fmax]

  # ULTIMATE LOAD CASES:
  # Vu = min{failure,80%Fmax,30mm}
  # Identifying failure with locator(). ESC key if there are none.
  failure_coord <- as.data.frame(matrix(nrow = length(myFiles),ncol = 2))
  rownames(failure_coord) <- myFiles
  colnames(failure_coord) <- c("x","y")
  failure_locator <- locator(1)
```

```
if (length(failure_locator) == 0) { # if there are located no failures
  failure_locator$x <- NA
  failure_locator$y <- NA
}
result["Failure Load (N)",i] <- failure_locator$y
result["Failure displacement (mm)",i] <- failure_locator$x

# 80%Fmax
F_80 <- 0.8*F_max
subs <- subset(my_data,my_data$Position > V_Fmax)
row <- which.min(abs(subs$Force - F_80))
if (subs$Force[row]-F_80 < 1 ) {
  V_F80 <- subs$Position[row]
} else {
  V_F80 <- NA
}

result["80% Max Load (N)",i] <- F_80
result["Displ at 80% Max Load (mm)",i] <- V_F80

# Slip at 30 mm
row_30 <- which.min(abs(my_data$Position - 30))
F_30mm <- my_data$`Force (smoothed)`[row_30]
V_30mm <- my_data$Position[row_30]
result["Force Load at 30 mm (N)",i] <- F_30mm
result["displ 30mm (mm)",i] <- V_30mm

# FINAL ULTIMATE LOAD
# the displacement that occurs first.
Displ_Values <- c(failure_locator$x, V_F80, V_30mm)
Load_Values <- c(failure_locator$y, F_80,F_30mm)
minimum <- which.min(Displ_Values)
Ultimate_displ <- Displ_Values[minimum]
Ultimate_force <- Load_Values[minimum]

result["Ultimate Force (N)",i] <- Ultimate_force
result["Ultimate Displ (mm)",i] <- Ultimate_displ

# plotting ultimate displ
abline (v=Ultimate_displ,col="grey", lty = 2, lwd = 1, pch = 3,
  lend = 0, ljoin = 2)
text(Ultimate_displ+0.1,Ultimate_force,labels = "Vu",adj = c(0,0),
  cex = 0.8, col = "grey")

#-----------------------------------------------------------------------
--
# CALCULATING THE YIELD LOAD/SLIP
# Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
xsub <- subset(my_data,my_data$Position < V_Fmax & dat$`Time(sec)` > 150)
# specifies time so that the 10% point will be at the mainload-stage and
  not in the preload-stage of the test.
row3 <- which.min(abs(xsub$Force - (0.1*F_max)))
V10 <- xsub[row3,1]
F10 <- xsub[row3,2]
row4 <- which.min(abs(xsub$Force - (0.4*F_max)))
V40 <- xsub[row4,1]
F40 <- xsub[row4,2]
xcoord <- c(V10,V40)
ycoord <- c(F10,F40)
# plotting points and slope in the graph
points(x = V10, y = F10,col="Red")
```

```
points(x = V40, y=F40, col="Red")
fit <- lm(ycoord~xcoord)
abline(fit, col="black",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)

# The intercept and slope for line1 is then found and so are the elastic
  stiffness which is equal to the slope.
b <- coef(lm(ycoord~xcoord))[1] #intercept
a <- coef(lm(ycoord~xcoord))[2] #slope
Vu <- Ultimate_displ

# YIELD LOAD
# The yield load is the interception between line1 and a second line2.
# Line 2 angle is 1/6 of line1s' slope and is thouching the graph.
# Firstly, we find the slope of line2 and secondly, we found the point
in
  the graph that has the same slope.
line2_slope <- (1/6)*a
# finding the slope in graph that is similar to line2_slope
slope_data <- as.data.frame(matrix(nrow=0,ncol=6))
colnames(slope_data) <- c("Intersept","Slope", "x1","x2","y1","y2" )
sub_slope <- subset(my_data,my_data$Position > V40
        & my_data$Position < V_Fmax)
for (m in 300:nrow(sub_slope)){# calculating slope with a step of 300
  vertices, because the rawdata alows it.
  x_graph <- c(sub_slope[m-299,1],sub_slope[m,1])
  y_graph <- c(sub_slope[m-299,2],sub_slope[m,2])
  coef_inter <- coef(lm(y_graph~x_graph))[1]
  coef_slope <- coef(lm(y_graph~x_graph))[2]
  slope_data[m,1] <- coef_inter
  slope_data[m,2] <- coef_slope
  slope_data[m,3] <- x_graph[1]
  slope_data[m,4] <- x_graph[2]
  slope_data[m,5] <- y_graph[1]
  slope_data[m,6] <- y_graph[2]
}
row_y <- which.min(abs(slope_data[,2]-line2_slope))
line2_data <- slope_data[row_y,]
x2_coord <- c(slope_data[row_y,3],slope_data[row_y,4])
y2_coord <- c(slope_data[row_y,5],slope_data[row_y,6])
line2_fit <- lm(y2_coord~x2_coord)
abline(line2_fit, col="gray",lty = 2, lwd = 1, pch = 3,
  lend = 0, ljoin = 2)

# FINDING F_y, v_y and k_ser
# Need to find the intersection between line1 and line2
# Algebraically calculated using intersection and slope from line1 and
2
V_y <- (line2_data[1,1]-b)/(a-line2_data[1,2])
F_y <- b+a*V_y
result["V_y",i] <- V_y
result["F_y",i] <- F_y
abline (h=F_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
  lend = 0, ljoin = 2)
abline (v=V_y,col="darkgray", lty = 2, lwd = 1, pch = 3,
  lend = 0, ljoin = 2)

k_ser <- F_y/V_y
result["k_ser",i]<- k_ser

# Ductility
D <- Ultimate_displ/V_y
```

```
  # ---------------------------------------------------------------------
--
  # FINAL RESULT
  # Constructing a final table with results
  Final_results[i,"maximum load (kN)"] <- F_max/1000
  Final_results[i,"Ultimate Load (kN)"] <- Ultimate_force/1000
  Final_results[i,"Ultimate displacement (mm)"] <- Ultimate_displ
  Final_results[i,"Yield Load (kN)"] <- F_y/1000
  Final_results[i,"Yield slip (mm)"] <- V_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- k_ser/1000
  Final_results[i,"Ductility"] <- D
}

# -----------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations
for (o in 1:length(Final_results)) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
  Final_results["Standard Deviations",o] <-
    sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - EN12512(2002).xlsx",
row.names = T, col.names = T)
```

4.3 EN 12512 (2018) Draft Version 20180410

```
# Study 3 Monotonic load test
# EN 12512 (2018) Draft Proposal version 20180410

# PACKAGES
#install.packages("gplots")
library("gplots", lib.loc="~/R/win-library/3.5")
#install.packages("ggplot2")
library("ggplot2")
library("zoo")
library("xlsx", lib.loc="~/R/win-library/3.5")
library("stats", lib.loc="C:/Program Files/R/R-3.5.0/library")

# DATA SET INFO
specimen_name <- "WF"
study <- "Study 3"
file_type <- ".csv"

# READING FILES
myFiles <- list.files(paste("C:\\Users\\Caroline\\OneDrive - Norwegian
          University of Life Sciences\\Master 2018\\3 - DATA ANALYZING\\",
          study ,"\\", specimen_name, sep=""),
          pattern = (paste("*",file_type,sep="")))
setwd(paste("C:\\Users\\Caroline\\OneDrive - Norwegian University of Life
          Sciences\\Master 2018\\3 - DATA ANALYZING\\", study ,"\\",
          specimen_name,sep=""))
all_data <- c()
myFiles2 <- sub("-","",x = myFiles)
```

```r
myFiles2 <- sub(".csv","",myFiles2)
result <- as.data.frame(matrix(nrow=0,ncol=length(myFiles)))
colnames(result)<-myFiles2

for (i in 1:length(myFiles)){
  header <- scan(myFiles[i], skip=10, nlines = 1,
          what = character(),sep=";")
  header2 <- scan(myFiles[i], skip=11, nlines = 1,
          what = character(),sep=";")
  dat <- read.csv(myFiles[i],skip=11,sep=";",dec=",",header=TRUE)
  colnames(dat) <- paste(header,header2,sep="")
  result["max load",i] <- max(dat$`Load(N)`)
  row_nr <- which.max(dat$`Load(N)`)
  result["max displ",i] <- dat[row_nr,2]
  all_data[[i]] <- dat
}

myFiles<-sub("-","",x = myFiles)
myFiles<-sub(".csv","",myFiles)
names(all_data)<-myFiles
Final_results <- as.data.frame(matrix(nrow=length(myFiles)+2,ncol=0))
rownames(Final_results)<-c(myFiles2,"Mean Values","Standard Deviations")

#------------------------------------------------------------------------
--
# FORCE SMOOTHING & PLOTTING GRAPHS
for (i in 1:length(all_data)){
  name_specimen <- paste(specimen_name,i,sep = "")
  main_title <- paste(study," - ", specimen_name,i,sep = "")
  specimen <- as.data.frame(all_data[name_specimen])
  specimen$lopende_kraft<-NA
  for (p in 5:(nrow(specimen)-1)){
    linje<-p+1
    intervall<-c(linje-5,linje+5)
    specimen$lopende_kraft[linje]<-
          mean(specimen[intervall[1]:intervall[2],3])
  }
  specimen <- specimen[complete.cases(specimen), ]
  my_data <- data.frame(matrix(nrow=0,ncol=2))
  my_data <- -specimen[2] # changes negative values to positive
  my_data[2] <- -specimen[4] # force
  colnames(my_data)<-c("Position","Force (smoothed)")

  plot(my_data$Position,my_data$`Force (smoothed)`,type = "l",
   main = main_title,xlab = "Displacement (mm)",ylab = "Force (N)")

#------------------------------------------------------------------------
  # CALCULATIONS
  # Peak load, Pl - max load reached during monotonic test
  row_Pl <- which.max(my_data$`Force (smoothed)`)
  Pl <- my_data$`Force (smoothed)`[row_Pl]
  V_Pl <- my_data$Position[row_Pl]

  # ULTIMATE LOAD CASES:
  # Vu - min{failure,80%Pl,30mm}
  # Identifying failure with locator(). ESC key if there are none.
  failure_locator <- locator(1)
  if (length(failure_locator) == 0) { # if there are located no failures
    failure_locator$x <- NA
    failure_locator$y <- NA
  }
```

```
result["Failure Load (N)",i] <- failure_locator$y
result["Failure displacement (mm)",i] <- failure_locator$x

# 80%Plmax - Load/displacement after peak load.
Pl_80 <- 0.8*Pl
subs <- subset(my_data,my_data$Position > V_Pl)
row <- which.min(abs(subs$Force - Pl_80))
V_Pl80 <- subs$Position[row]
result["80% Peak Load (N)",i] <- Pl_80
result["Displ at 80% Peak Load (mm)",i] <- V_Pl80

# Slip at 30 mm
row_30 <- which.min(abs(my_data$Position - 30))
F_30mm <- my_data$`Force (smoothed)`[row_30]
V_30mm <- my_data$Position[row_30]
result["Force Load at 30 mm (N)",i] <- F_30mm
result["displ 30mm (mm)",i] <- V_30mm

# FINAL ULTIMATE LOAD
# the displacement that occurs first.
Displ_Values <- c(failure_locator$x, V_Pl80, V_30mm)
Load_Values <- c(failure_locator$y, Pl_80,F_30mm)
minimum <- which.min(Displ_Values)
Ultimate_displ <- Displ_Values[minimum]
Ultimate_force <- Load_Values[minimum]

result["Ultimate Force (N)",i] <- Ultimate_force
result["Ultimate Displ (mm)",i] <- Ultimate_displ

# MAX LOAD
# max load lower than or equal to ultimate displacement
sub_Fmax <- subset(my_data,my_data$Position <= Ultimate_displ)
rowmax <- which.max(sub_Fmax$Force)
F_max <- sub_Fmax[rowmax,2]

# plotting ultimate displ
abline (v=Ultimate_displ,col="grey", lty = 2, lwd = 1, pch = 3,
   lend = 0, ljoin = 2)
text(Ultimate_displ+0.1,Ultimate_force,labels = "Vu",adj = c(0,0),
     cex = 0.8, col = "grey")

  #-----------------------------------------------------------------
--
# CALCULATING THE EEEP CURVE
# First, we retrieve the area under the curve.
# AUC is the area under the curve with boundaries from origin to ultimate
  displacement
xsub <- subset(my_data,my_data$Position <= Ultimate_displ)
x <- xsub$Position
y <- xsub$Force
id <- order(x)
AUC <- sum(diff(x[id])*rollmean(y[id],2))

# Finding the line that goes through 10%Fmax and 40%Fmax ~ line1
row3 <- which.min(abs(xsub$Force - (0.1*F_max)))
V10 <- xsub[row3,1]
F10 <- xsub[row3,2]
row4 <- which.min(abs(xsub$Force - (0.4*F_max)))
V40 <- xsub[row4,1]
F40 <- xsub[row4,2]
xcoord <- c(V10,V40)
```

```
  ycoord <- c(F10,F40)
  # plotting points and slope in LEC graph
  points(x = V10, y = F10,col="Red")
  points(x = V40, y=F40, col="Red")
  fit <- lm(ycoord~xcoord)
  abline(fit, col="black",lty = 2, lwd = 1, pch = 3, lend = 0, ljoin = 2)

  # The intercept and slope for line1 is then found and so are the elastic
    stiffness which is equal to the slope.
  b <- coef(lm(ycoord~xcoord))[1] #intercept
  a <- coef(lm(ycoord~xcoord))[2] #slope
  K <- a[[1]] # Elastic stiffness [N/mm]
  Vu <- xsub[nrow(xsub),1] # ultimate displacement retrieved from subset
of
    the graph

  # the equation to find the horisontal line (line2) that gives an area
    equal to the LEC1 area is an quadratic equation.
  # The y-solution to the line is then given as plus and minus, referring
    to the quadratic formula:

  mMinus = a[[1]]*((Vu+b[[1]]/a[[1]]) -
          sqrt((Vu+b[[1]]/a[[1]])^2    -    4*((b[[1]]/(4*a[[1]]))^2    +
          AUC/(2*a[[1]])))) #with x=0 as initial boundary

  # calculating the intersection between line1 and line2.
  # the intersection is the yield load and displacement.
  F_ymMinus <- mMinus
  v_ymMinus <- (mMinus - b[[1]])/a[[1]]
  abline(h = F_ymMinus, col="grey",lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  abline(v = v_ymMinus, col="grey",lty = 2, lwd = 1, pch = 3,
    lend = 0, ljoin = 2)
  result["v_y",i] <- v_ymMinus
  result["F_y",i] <- F_ymMinus
  v_y <- v_ymMinus
  F_y <- F_ymMinus

  # Ductility
  D <- Ultimate_displ/v_y

  # ----------------------------------------------------------------------
--
  # FINAL RESULT
  # Constructing a final table with results
  Final_results[i,"Peak Load (kN)"] <- Pl/1000
  Final_results[i,"maximum load (kN)"] <- F_max/1000
  Final_results[i,"Ultimate Load (kN)"] <- Ultimate_force/1000
  Final_results[i,"Ultimate displacement (mm)"] <- Ultimate_displ
  Final_results[i,"Yield Load (kN)"] <- F_y/1000
  Final_results[i,"Yield slip (mm)"] <- v_y
  Final_results[i,"Slip modulus - k_ser (kN/mm)"] <- K/1000
  Final_results[i,"Ductility"] <- D
}

# ------------------------------------------------------------------------
--
# FINAL CALCULATIONS
# Mean values & Standard Deviations
for (o in 1:length(Final_results)) {
  Final_results["Mean Values",o] <- mean(Final_results[,o], na.rm = TRUE)
```

```
  Final_results["Standard Deviations",o] <-
    sd(Final_results[(1:length(myFiles)),o])
}

is.num <- sapply(Final_results, is.numeric) # restricts decimals to two
Final_results[is.num] <- lapply(Final_results[is.num], round, 2)

# # SAVING TABLES AS .XLSX FILES IN THE SAME WORK DIRECTORY WRITTEN IN THE
TOP OF THIS SCRIPT
write.xlsx(Final_results, file = "Final_results - EN12512(2018).xlsx",
row.names = T, col.names = T)
```