Supplemental Analysis

#-Load required packages

```
#rm(list=ls(all=T))
require(tidyverse)
require(gridExtra)
require(data.table)
require(knitr)
require(kableExtra)
require(magrittr)
require(moments)
require(plot3D)
require(DataCombine)
```

#-Functions ##flst2DF - Convert a list (e.g. from lapply) to a data frame.

```
flst2DF <- function(lst) {
  DF = as.data.frame(do.call(rbind, lst))
}
```

##%tin%-Function to assign multiple variables at once. E.g. c(E0, F0, Rp, Km, kcat) %tin% dfParamsAndCI[1,]

```
`%tin%` <- function(x, y) {
  mapply(assign, as.character(substitute(x)[-1]), y,
         MoreArgs = list(envir = parent.frame()))
  invisible()
}
```

##fCV-Calculate the CV of a set of values.

```
fCV <- function(x, roundDigits=2) {
  return(round((sd(x) /mean(x))*100, roundDigits))
}
```

##fScaleNormDistr Scale normal distribution frequencies by a factor of n, so as to add the normal curve to a barplot (discrete histogram). It is to be used in fPlotDiscrHist

```
fScaleNormDistr <- function(x, mean, sd, n) {
            dnorm(x = x, mean = mean, sd = sd) * n
}
```

##fPlotDiscrHist Plot discrete histogram (barplot) of the parameter “param” for the 4 samples

```
fPlotDiscrHist <- function(param) {
  grid.arrange(
      ggplot(data=smplA, aes_string(x=param)) + geom_bar() +
        stat_function(fun=fScaleNormDistr, args=c(mean=mean(smplA[,param]), sd=sd(smplA[,param]), n=nrow(smpl
A))) +
        geom_vline(xintercept = mean(smplA[,param]), size=1) +
        geom_vline(xintercept = mean(smplA[,param]) - 2*sd(smplA[,param]), linetype=2) +
        geom_vline(xintercept = mean(smplA[,param]) + 2*sd(smplA[,param]), linetype=2) +
        ggtitle("Sample A") + theme(plot.title = element_text(hjust = 0.5)),

      ggplot(data=smplB, aes_string(x=param)) + geom_bar() +
        stat_function(fun=fScaleNormDistr, args=c(mean=mean(smplB[,param]), sd=sd(smplB[,param]), n=nrow(smpl
B))) +
        geom_vline(xintercept = mean(smplB[,param]), size=1) +
        geom_vline(xintercept = mean(smplB[,param]) - 2*sd(smplA[,param]), linetype=2) +
        geom_vline(xintercept = mean(smplB[,param]) + 2*sd(smplA[,param]), linetype=2) +
        ggtitle("Sample B") + theme(plot.title = element_text(hjust = 0.5)),

      ggplot(data=smplC, aes_string(x=param)) + geom_bar() +
        stat_function(fun=fScaleNormDistr, args=c(mean=mean(smplC[,param]), sd=sd(smplC[,param]), n=nrow(smpl
C))) +
        geom_vline(xintercept = mean(smplC[,param]), size=1) +
        geom_vline(xintercept = mean(smplC[,param]) - 2*sd(smplA[,param]), linetype=2) +
        geom_vline(xintercept = mean(smplC[,param]) + 2*sd(smplA[,param]), linetype=2) +
        ggtitle("Sample C") + theme(plot.title = element_text(hjust = 0.5)),

      ggplot(data=smplD, aes_string(x=param)) + geom_bar() +
        stat_function(fun=fScaleNormDistr, args=c(mean=mean(smplD[,param]), sd=sd(smplD[,param]), n=nrow(smpl
D))) +
        geom_vline(xintercept = mean(smplD[,param]), size=1) +
        geom_vline(xintercept = mean(smplD[,param]) - 2*sd(smplA[,param]), linetype=2) +
        geom_vline(xintercept = mean(smplD[,param]) + 2*sd(smplA[,param]), linetype=2) +
        ggtitle("Sample D") + theme(plot.title = element_text(hjust = 0.5)),

      ncol=2
            )
}
```

##fPlotCorrWithRegrLine Scatterplot of pairs of parameters for the 4 samples with the corresponding regression line

```
fPlotCorrWithRegrLine <- function(xParam, yParam) {
  grid.arrange(
    ggplot(data=smplA, aes_string(x=xParam, y=yParam)) +
      geom_point() +
      geom_smooth(method="lm", se=F) +
      ggtitle(paste("Sample A - ", xParam, " versus ", yParam, sep="")) + theme(plot.title = element_text(hjus
t = 0.5)),
    ggplot(data=smplB, aes_string(x=xParam, y=yParam)) +
          geom_point() +
          geom_smooth(method="lm", se=F) +
          ggtitle(paste("Sample B - ", xParam, " versus ", yParam, sep="")) + theme(plot.title = element_text
(hjust = 0.5)),
    ggplot(data=smplC, aes_string(x=xParam, y=yParam)) +
          geom_point() +
          geom_smooth(method="lm", se=F) +
          ggtitle(paste("Sample C - ", xParam, " versus ", yParam, sep="")) + theme(plot.title = element_text
(hjust = 0.5)),
    ggplot(data=smplD, aes_string(x=xParam, y=yParam)) +
          geom_point() +
          geom_smooth(method="lm", se=F) +
          ggtitle(paste("Sample D - ", xParam, " versus ", yParam, sep="")) + theme(plot.title = element_text
(hjust = 0.5)),
    ncol=2

            )
}
```

##fErrorCorCov Function to calculate the correlation and covariance between the errors of two groups and plot them.

```
fErrorCorCov <- function(DF, param1, param2, plot=F) {
  if(DF==dfA) {sample="Sample A"} else if(DF==dfB) {sample="Sample B"} else if(DF==dfC) {sample="Sample C"} el
se {sample="Sample D"}
  grp1 = DF[,c(param1)]
  grp2= DF[,c(param2)]
  grp1Error = (grp1 - mean(grp1))^2
  grp2Error = (grp2 - mean(grp2))^2
  correlation = round(cor(grp1Error, grp2Error),2)
  covariance = round(cov(grp1Error, grp2Error),2)
  if(plot) {
    title = paste("Correlation between the errors of", param1, "and", param2, "in", sample, "\n Correlation Co
efficient=",correlation, "Covariance=", covariance)
    plot(grp1Error, grp2Error, main=title, xlab=paste(param1, "Error"), ylab=paste(param2, "Error"))
  }
    return(list(distr1Error=grp1Error, distr2Error=grp2Error, Correlation=correlation, Covariance=covariance))
}
```

##fLDLErrPrp Function to calculate LDL Variance according to error propagation theory, as derived in the supplementary material

```
fLDLErrPrp <- function(CHOL, HDL, TG, divFactor=5) {
  LDLVar = var(CHOL) + var(HDL) + (var(TG)/(divFactor^2)) -2*cov(CHOL, HDL) - 2*(cov(CHOL,TG)/divFactor) + 2 *
(cov(HDL,TG)/divFactor)
  return(LDLVar)
}
```

fAIPErrPrp

```
fAIPErrPrp <- function(TG, HDL) {
  (  mean(TG*0.0113)^2 * var(HDL*0.0259)   +
     mean(HDL*0.0259)^2 * var(TG*0.0113)   -
     2 *  mean(TG*0.0113) * mean(HDL*0.0259) * cov(TG*0.0113,HDL*0.0259)
  )   /
     (log(10)^2 * mean(TG*0.0113)^2 * mean(HDL*0.0259)^2)
}
```

fAIPErrPrp2Ord

```
# x1=TG, x2=HDL
fAIPErrPrp2Ord <- function(x1,x2) {
  u1 <- mean(x1)
  u2 <- mean(x2)
  m2_1 <- moment(x1,order=2,central=T)#4th central moment of x1
  m2_2 <- moment(x2,order=2,central=T)#4th central moment of x2
  m4_1 <- moment(x1,order=4,central=T)#4th central moment of x1
  m4_2 <- moment(x2,order=4,central=T)#4th central moment of x2
  corx1x2 <- cor(x1, x2)
  nominator <-
    u2^4 * m4_1 -
    4 * u1^3 * u2^3 * corx1x2 * sd(x1) * sd(x2) +
    2 * u1^2 * u2^2 * m2_1 * (u2 - corx1x2 * sd(x2)) * (u2 + corx1x2 * sd(x2)) +
    u1^4 * (2 * u2^2 * m2_2 + m4_2)

  denom <- 2 * u1^4 * u2^4 * log(10)^2

  AIPVrnc2Ord <- nominator / denom
  return(AIPVrnc2Ord)
}
```

##fLDLBoot Function to calulate the variance of LDL using Bootstrapping. sampleSize: Number of samples drawn uniformly and with replacement from the vec Dataset noOfReps: Number of new distributions to draw

```
fLDLBoot <- function(vecCHOL, vecHDL, vecTG, sampleSize=length(vecCHOL), noOfReps=1000, pb=F) {
    if((length(vecCHOL) != length(vecHDL)) | (length(vecCHOL) != length(vecTG)) | (length(vecHDL) != length(ve
cTG)) ) {
      print("The three parameters must have the same number of measurements"); return()
    }
  n = length(vecCHOL)
    if(sampleSize>n) {
      print("Size of bootstraped datasets cannot be larger than the original. Setting k=n")
      sampleSize=n;
      print(paste("=", n))
    }
    dfTmp <- data.frame(cbind(vecCHOL, vecHDL, vecTG))
    colnames(dfTmp) <- c("CHOL", "HDL", "TG")
    dtLDLBoot = data.table("Mean"=numeric(noOfReps), "Var"=numeric(noOfReps), "CV"=numeric(noOfReps))
    if(pb) pb <- txtProgressBar(min = 0, max = noOfReps, style = 3)
      for(bootIdx in seq(1,noOfReps)) {
          dfSmpl <- dfTmp[sample(nrow(dfTmp), size=sampleSize, replace = T),]
          dfLDLSmpl = dfSmpl$CHOL - dfSmpl$HDL - (dfSmpl$TG/5)
          LDLSmplMean = mean(dfLDLSmpl)
          LDLSmplVar = var(dfLDLSmpl)
          LDLSmplCV = fCV(dfLDLSmpl)
          set(dtLDLBoot, i=bootIdx, j=1L, value=LDLSmplMean)
          set(dtLDLBoot, i=bootIdx, j=2L, value=LDLSmplVar)
          set(dtLDLBoot, i=bootIdx, j=3L, value=LDLSmplCV)
          if(pb) setTxtProgressBar(pb, bootIdx)
      }
    return(list("Mean"=median(dtLDLBoot$Mean), "Var"=median(dtLDLBoot$Var), "CV"=median(dtLDLBoot$CV),
                "dataTable"=dtLDLBoot))
  }
```

##fAIPBoot Function to calulate the variance of LDL using Bootstrapping. sampleSize: Number of samples drawn uniformly and with replacement from the vec Dataset noOfReps: Number of new distributions to draw

```
fAIPBoot <- function(vecTG, vecHDL, sampleSize=length(vecTG), noOfReps=1000, pb=F) {
    if( (length(vecHDL) != length(vecTG)) ) {
      print("The two parameters must have the same number of measurements"); return()
    }
  n = length(vecTG)
    if(sampleSize>n) {
      print("Size of bootstraped datasets cannot be larger than the original. Setting k=n")
      sampleSize=n;
      print(paste("=", n))
    }
    dfTmp <- data.frame(cbind(vecTG, vecHDL))
    colnames(dfTmp) <- c("TG", "HDL")
    dtAIPBoot = data.table("Mean"=numeric(noOfReps), "Var"=numeric(noOfReps), "CV"=numeric(noOfReps))
    if(pb) pb <- txtProgressBar(min = 0, max = noOfReps, style = 3)
      for(bootIdx in seq(1,noOfReps)) {
          dfSmpl <- dfTmp[sample(nrow(dfTmp), size=sampleSize, replace = T),]
          dfAIPSmpl = log10( (dfSmpl$TG * 0.0113) / (dfSmpl$HDL * 0.0259) )
          AIPSmplMean = mean(dfAIPSmpl)
          AIPSmplVar = var(dfAIPSmpl)
          AIPSmplCV = fCV(dfAIPSmpl)
          set(dtAIPBoot, i=bootIdx, j=1L, value=AIPSmplMean)
          set(dtAIPBoot, i=bootIdx, j=2L, value=AIPSmplVar)
          set(dtAIPBoot, i=bootIdx, j=3L, value=AIPSmplCV)
          if(pb) setTxtProgressBar(pb, bootIdx)
      }
    return(list("Mean"=median(dtAIPBoot$Mean), "Var"=median(dtAIPBoot$Var), "CV"=median(dtAIPBoot$CV),
                "dataTable"=dtAIPBoot))
  }
```

fPltDensMedPerc

Plot density of the Variance along with the median, 2.5 and 97.5 percentile as vertical lines

```
fPltDensMedPerc <- function(DF, title) {
  plot(density(DF$dataTable$Var), main=title)
  abline(v=median(DF$dataTable$Var))
  abline(v=quantile(DF$dataTable$Var, probs=c(0.025)), lty=2)
  abline(v=quantile(DF$dataTable$Var, probs=c(0.975)), lty=2)
}
```

Plot density of the bootstrap variance for LDL along with the median, 2.5 and 97.5 percentile as vertical lines. Also plot the empirical variance as a segment from the bottom to the middle of the graph and the error propagation variance as a line segment from the top tho the miidle of the graph. DF must the the dataframe of the list returned from the respective (LDL or AIP) Bootstrap variance function, e.g. DF=AIPSmplABootVrnc$dataTable.

```
fPltLDLVrncDens <- function(DF, title, empirVrnc, errPropVrnc) {
  xMin <- min(density(DF$Var)$x)
  xMax <- max(density(DF$Var)$x)
  yMin <- min(density(DF$Var)$y)
  yMax <- max(density(DF$Var)$y)
  plot(density(DF$Var), main="", xlab="LDL Variance", xlim=c(0,xMax), lwd=2)
  text(x=xMin*1.05, y=yMax*0.90, title, cex=1)
  abline(v=median(DF$Var), lwd=2)
  abline(v=quantile(DF$Var, probs=c(0.025)), lty=2, lwd=2)
  abline(v=quantile(DF$Var, probs=c(0.975)), lty=2, lwd=2)
  segments(x0=empirVrnc,x1=empirVrnc, y0=yMin, y1=yMax/2, lty=3, lwd=2)# Plot segment
                                              #corresponding to empirical variance
  segments(x0=errPropVrnc, x1=errPropVrnc, y0=yMax, y1=yMax/2, lty=4, lwd=2)# Plot
                                      #segment corresponding to error propagation variance
}
```

Plot density of the bootstrap variance for AIP along with the median, 2.5 and 97.5 percentile as vertical lines. Also plot the empirical variance as a segment from the bottom to the middle of the graph and the error propagation variance as a line segment from the top tho the miidle of the graph. DF must the the dataframe of the list returned from the respective (LDL or AIP) Bootstrap variance function, e.g. DF=AIPSmplABootVrnc$dataTable.

```
fPltAIPVrncDens <- function(DF, title, empirVrnc, errPropVrnc, errPropVrnc2Ord) {
  xMin <- min(density(DF$Var)$x)
  xMax <- max(density(DF$Var)$x)
  yMin <- min(density(DF$Var)$y)
  yMax <- max(density(DF$Var)$y)
  plot(density(DF$Var), main="", xlim=c(0,xMax), xlab="AIP Variance", lwd=2)
  text(x=xMin, y=yMax*0.90, title, cex=1)
  abline(v=median(DF$Var), lwd=2)
  abline(v=quantile(DF$Var, probs=c(0.025)), lty=2, lwd=2)
  abline(v=quantile(DF$Var, probs=c(0.975)), lty=2, lwd=2)
  segments(x0=empirVrnc,x1=empirVrnc, y0=yMin, y1=yMax/2, lty=3, lwd=2)# Plot segment
                                              #corresponding to empirical variance
  segments(x0=errPropVrnc, x1=errPropVrnc, y0=yMax *0.25, y1=yMax*0.75, lty=5, lwd=2)#Plot
                                      #segment corresponding to error propagation variance
  segments(x0=errPropVrnc2Ord, x1=errPropVrnc2Ord, y0=yMax, y1=yMax/2, lty=4, lwd=2)# Plot
                              #segment corresponding to 2nd Order error propagation variance
}
```

##exclude_runif Function to give a random integer number using the random uniform distribution after excluding a specified value.

```
exclude_runif <- function(n, min, max, valueToExclude) {
  res = valueToExclude
  while (res == valueToExclude) {
    res = round(runif(1, min, max))
  }
  return(res)
}
```

##fChngDistVrncMeanConst Function to change the variance of a set of (integer) numbers by randomly selecting two numbers from the set, increasing one of them by 1 and decreasing the other by 1.

```
fChngDistVrncMeanConst <- function(grp) {
  grpLen = length(grp)
  indxOfValueToDecrease = round(runif(1, 1, grpLen))
  indxOfValueToIncrease = exclude_runif(1, 0, grpLen, indxOfValueToDecrease )
  grpNew = grp
  grpNew[indxOfValueToDecrease] = grpNew[indxOfValueToDecrease] - 1
  grpNew[indxOfValueToIncrease] = grpNew[indxOfValueToIncrease] + 1
  return(grpNew)
}
```

##fIncrCVMeanConst Function to take a set of values and increase its CV (using function fChngDistVrncMeanConst) until the CV reaches the maximum specified value.

```
fIncrCVMeanConst <- function(grp, maxCV, maxIter=10000) {
  grpInit = vector(mode="numeric", length = length(grp))
  grpCombIncrVariance = data.frame(cbind(grpInit, grp))
  currCV = fCV(grp)
  iterCounter = 0
  while( (currCV<=maxCV)   ) {
    grpOld = grpCombIncrVariance[,ncol(grpCombIncrVariance)]
    grpNew = fChngDistVrncMeanConst(grpOld)
    if( (var(grpNew)>var(grpOld)) & (mean(grpNew)==mean(grpOld)) ) {
      grpCombIncrVariance = cbind( grpCombIncrVariance, grpNew)
    }
    currCV = fCV(grpCombIncrVariance[,ncol(grpCombIncrVariance)])
    #iterCounter = iterCounter + 1; print(iterCounter); print(currCV)
  }
  grpCombIncrVariance$grpInit = NULL
  return(grpCombIncrVariance)
}
```

##fDecrCVMeanConst Function to take a set of values and decrease its CV (using function fChngDistVrncMeanConst) until the CV reaches the minimun specified value.

```
fDecrCVMeanConst <- function(grp, minCV, maxIter=10000) {
  iterCounter = 1
  grpInit = vector(mode="numeric", length = length(grp))
  grpCombDecVariance = data.frame(cbind(grpInit, grp))
  currCV = fCV(grp)
  while( (currCV>minCV) & (iterCounter<maxIter) ) {
    grpOld = grpCombDecVariance[,ncol(grpCombDecVariance)]
    grpNew = fChngDistVrncMeanConst(grpOld)
    if( (var(grpNew)<var(grpOld)) & (mean(grpNew)==mean(grpOld)) ) {
      grpCombDecVariance = cbind( grpCombDecVariance, grpNew)
    }
    currCV = fCV(grpCombDecVariance[,ncol(grpCombDecVariance)])
    iterCounter = iterCounter + 1
  }
  grpCombDecVariance$grpInit = NULL
  return(grpCombDecVariance)
}
```

##fCombineVariances Function to combine the two data frames that resulted from fIncrCVMeanConst() and fDecrCVMeanConst() into one data frame with the samples arranged in increasing variance order.

```
fCombineVariances <- function(dfVariancesDec, dfVariancesIncr, plot=F) {
  dfVariancesDecRev = rev(dfVariancesDec)
  dfVariances = cbind(dfVariancesDecRev, dfVariancesIncr[,2:ncol(dfVariancesIncr)])
    if(plot) {
      par(mfrow=c(2,1))
        plot(1:ncol(dfVariances), apply(dfVariances, 2, mean), type="l")
        plot(1:ncol(dfVariances), apply(dfVariances, 2, fCV))
      par(mfrow=c(1,1))
    }
  return(dfVariances)
}
```

##fChngDistrCVMeanConst Function to change the CV of a distribution while keeping the mean constant. An upper and a lower value is specified for the CV. It uses the functions fIncrCVMeanConst(), fDecrCVMeanConst() and fCombineVariances().

```
fChngDistrCVMeanConst <- function(vecDistr, lowerCV, upperCV, maxIter=10000, plot=F) {
  dfDecrCV = fDecrCVMeanConst(vecDistr, minCV=lowerCV, maxIter = maxIter)
  dfIncrCV = fIncrCVMeanConst(vecDistr, maxCV=upperCV, maxIter = maxIter)
  dfCombVrncs = fCombineVariances(dfDecrCV, dfIncrCV, plot=plot)
  return(dfCombVrncs)
}
```

fLDLCHOLVrnc

Function to calculate the Error Propagation and the Bootstrap variance of LDL using a data frame containing multiple realizations of CHOL with a set of values of increasing variance as the column index increases. Each column contains a set of values of the CHOL variable.

```
fLDLCHOLVrnc <- function(dfCHOL, vecHDL, vecTG, bootStrpNoOfReps=5000 ) {
  LDLVrncChangingErrPropVrnc = vector(mode="numeric", length=ncol(dfCHOL))
  LDLVrncChangingBootVrnc = vector(mode="numeric", length=ncol(dfCHOL))
    for(i in 1:ncol(dfCHOL)) {
      LDLVrncErrProp = fLDLErrPrp(dfCHOL[,i], vecHDL, vecTG)
      LDLVrncChangingErrPropVrnc[i] = LDLVrncErrProp
      LDLVrncBoot = fLDLBoot(dfCHOL[,i], vecHDL, vecTG, noOfReps = bootStrpNoOfReps)
      LDLVrncChangingBootVrnc[i] = LDLVrncBoot$Var
    print(i)
    }
  return(list(ErrPropVrnc = LDLVrncChangingErrPropVrnc, BootVrnc = LDLVrncChangingBootVrnc))
}
```

fLDLHDLVrnc

Function to calculate the Error Propagation and the Bootstrap variance of LDL using a data frame containing multiple realizations of HDL with a set of values of increasing variance as the column index increases. Each column contains a set of values of the HDL variable.

```
fLDLHDLVrnc <- function(vecCHOL, dfHDL, vecTG, bootStrpNoOfReps=5000 ) {
  LDLVrncChangingErrPropVrnc = vector(mode="numeric", length=ncol(dfHDL))
  LDLVrncChangingBootVrnc = vector(mode="numeric", length=ncol(dfHDL))
  for(i in 1:ncol(dfHDL)) {
    LDLVrncErrProp = fLDLErrPrp(vecCHOL, dfHDL[,i], vecTG)
    LDLVrncChangingErrPropVrnc[i] = LDLVrncErrProp
    LDLVrncBoot = fLDLBoot(vecCHOL, dfHDL[,i], vecTG, noOfReps = bootStrpNoOfReps)
    LDLVrncChangingBootVrnc[i] = LDLVrncBoot$Var
    print(i)
  }
  return(list(ErrPropVrnc = LDLVrncChangingErrPropVrnc, BootVrnc = LDLVrncChangingBootVrnc))
}
```

##fLDLTGVrnc Function to calculate the Error Propagation and the Bootstrap variance of LDL using a data frame containing multiple realizations of TG with a set of values of increasing variance as the column index increases. Each column contains a set of values of the TG variable.

```
fLDLTGVrnc <- function(vecCHOL, vecHDL, dfTG, bootStrpNoOfReps=5000 ) {
  LDLVrncChangingErrPropVrnc = vector(mode="numeric", length=ncol(dfTG))
  LDLVrncChangingBootVrnc = vector(mode="numeric", length=ncol(dfTG))
  for(i in 1:ncol(dfTG)) {
    LDLVrncErrProp = fLDLErrPrp(vecCHOL, vecHDL, dfTG[,i])
    LDLVrncChangingErrPropVrnc[i] = LDLVrncErrProp
    LDLVrncBoot = fLDLBoot(vecCHOL, vecHDL, dfTG[,i], noOfReps = bootStrpNoOfReps)
    LDLVrncChangingBootVrnc[i] = LDLVrncBoot$Var
    print(i)
  }
  return(list(ErrPropVrnc = LDLVrncChangingErrPropVrnc, BootVrnc = LDLVrncChangingBootVrnc))
}
```

fAIPHDLVrnc

Function to calculate the Error Propagation and the Bootstrap variance of AIP using a data frame containing multiple realizations of HDL with a set of values of increasing variance as the column index increases. Each column contains a set of values of the HDL variable.

```
fAIPHDLVrnc <- function(vecTG, dfHDL, bootStrpNoOfReps=5000 ) {
  #browser()
  AIPVrncChangingErrPropVrnc = vector(mode="numeric", length=ncol(dfHDL))
  AIPVrnc2OrdChangingErrPropVrnc = vector(mode="numeric", length=ncol(dfHDL))
  AIPVrncChangingBootVrnc = vector(mode="numeric", length=ncol(dfHDL))
  for(i in 1:ncol(dfHDL)) {
    AIPVrncErrProp = fAIPErrPrp(vecTG,  dfHDL[,i])
    AIPVrncErrProp2Ord = fAIPErrPrp2Ord(vecTG,  dfHDL[,i])
    AIPVrncChangingErrPropVrnc[i] = AIPVrncErrProp
    AIPVrnc2OrdChangingErrPropVrnc[i] = AIPVrncErrProp2Ord
    AIPVrncBoot = fAIPBoot(vecTG, dfHDL[,i], noOfReps = bootStrpNoOfReps)
    AIPVrncChangingBootVrnc[i] = AIPVrncBoot$Var
    print(i)
  }
  return(list(ErrPropVrnc = AIPVrncChangingErrPropVrnc,
              ErrPropVrnc2Ord = AIPVrnc2OrdChangingErrPropVrnc,
              BootVrnc = AIPVrncChangingBootVrnc))
}
```

fAIPTGVrnc

Function to calculate the Error Propagation and the Bootstrap variance of LDL using a data frame containing multiple realizations of TG with a set of values of increasing variance as the column index increases. Each column contains a set of values of the TG variable.

```
fAIPTGVrnc <- function(dfTG, vecHDL, bootStrpNoOfReps=5000 ) {
  AIPVrncChangingErrPropVrnc = vector(mode="numeric", length=ncol(dfTG))
  AIPVrnc2OrdChangingErrPropVrnc = vector(mode="numeric", length=ncol(dfTG))
  AIPVrncChangingBootVrnc = vector(mode="numeric", length=ncol(dfTG))
  for(i in 1:ncol(dfTG)) {
    AIPVrncErrProp = fAIPErrPrp(dfTG[,i], vecHDL)
    AIPVrncErrProp2Ord = fAIPErrPrp2Ord(dfTG[,i], vecHDL)
    AIPVrncChangingErrPropVrnc[i] = AIPVrncErrProp
    AIPVrnc2OrdChangingErrPropVrnc[i] = AIPVrncErrProp2Ord
    AIPVrncBoot = fAIPBoot(dfTG[,i], vecHDL, noOfReps = bootStrpNoOfReps)
    AIPVrncChangingBootVrnc[i] = AIPVrncBoot$Var
    print(i)
  }
  return(list(ErrPropVrnc = AIPVrncChangingErrPropVrnc,
              ErrPropVrnc2Ord = AIPVrnc2OrdChangingErrPropVrnc,
              BootVrnc = AIPVrncChangingBootVrnc))
}
```

fChngDistrMeanVrncConst

Function to take an empirical distribution and change its mean while keeping its variance constant. This can be achieved by adding (increase mean) or subtracting (decrease mean) the same value from all the values contained in the distribution.

```
fChngDistrMeanVrncConst <- function(vecDistr, lowerMean, upperMean, increment=1, plot=F) {
  nrows = length(vecDistr)
  vecDistrMean = mean(vecDistr)
  currMean = mean(vecDistr)
  ncols = round((vecDistrMean - lowerMean) / increment) + 3 # Calculate and create a matrixc with the appropri
ate number of
  mtrxDecrMeans = matrix(nrow=nrows, ncol=ncols) # columns.  Each column will hold a distribution of decreasin
g mean.
  mtrxDecrMeans[,1] = vecDistr
  colCounter = 2
  while(currMean >= lowerMean) {
    currDistr = mtrxDecrMeans[,colCounter - 1]
    currMean = mean(currDistr)
    newDistr = currDistr - increment
    mtrxDecrMeans[,colCounter] = newDistr
    colCounter = colCounter + 1
  }
  #Same as above, but now  each column of the matrix will hold a distribution of increasing mean.
  vecDistrMean = mean(vecDistr)
  currMean = mean(vecDistr)
  ncols = round((upperMean - vecDistrMean) / increment) + 3
  mtrxIncrMeans = matrix(nrow=nrows, ncol=ncols)
  mtrxIncrMeans[,1] = vecDistr
  colCounter = 2
  while(currMean <= upperMean) {
    currDistr = mtrxIncrMeans[,colCounter - 1]
    currMean = mean(currDistr)
    newDistr = currDistr + increment
    mtrxIncrMeans[,colCounter] = newDistr
    colCounter = colCounter + 1
  }
  mtrxChangeMeans = cbind(mtrxDecrMeans, mtrxIncrMeans[,2:ncol(mtrxIncrMeans)]) # Combine the two matrices in
one.
  mtrxChangeMeans = mtrxChangeMeans[,colSums(is.na(mtrxChangeMeans)) != nrow(mtrxChangeMeans)]# Remove COLUMNS
that are all NAs
  mtrxChangeMeansColsAvgs = colMeans(mtrxChangeMeans, na.rm=T) # Calculate column means of the sorted matrix
  mtrxChangeMeansColOrder = order(mtrxChangeMeansColsAvgs) # Determine the order of the cols by increasing col
umn average
  mtrxChangeMeans = mtrxChangeMeans[,mtrxChangeMeansColOrder]
  if(plot) {
    par(mfrow=c(2,1))
    plot(colMeans(mtrxChangeMeans), ylab="Column Mean")
    plot(apply(mtrxChangeMeans, 2, var), ylab="Column Variance")
    par(mfrow=c(1,1))
  }
  return(mtrxChangeMeans)
}
```

##fAverageData Average data using various number of values in order to demonstrate the CLT.

```
fAverageData <- function(dat, sampleSize, noOfReps) {
  vecMeans = vector(mode="numeric", length=noOfReps)
  for(i in seq(1,noOfReps)) {
    smpl = sample(dat, size=sampleSize, replace = T)
    vecMeans[i] = mean(smpl)
  }
  return(vecMeans)
}
```

#-Load data and define quantities, lists and other stuff ## Read and Load data

```
smplA <- read.csv("./dataFiles/smplA.csv", header=T, sep=",")
smplB <- read.csv("./dataFiles/smplB.csv", header=T, sep=",")
smplC <- read.csv("./dataFiles/smplC.csv", header=T, sep=",")
smplD <- read.csv("./dataFiles/smplD.csv", header=T, sep=",")
```

Calculate AIP

```
smplA$AIP <- round(log10((smplA$TG*0.0113)/(smplA$HDL*0.0259)),4)
smplB$AIP <- round(log10((smplB$TG*0.0113)/(smplB$HDL*0.0259)),4)
smplC$AIP <- round(log10((smplC$TG*0.0113)/(smplC$HDL*0.0259)),4)
smplD$AIP <- round(log10((smplD$TG*0.0113)/(smplD$HDL*0.0259)),4)
```

Suppl. Tbls 1-4

Print samples

```
kable(smplA, booktabs=T, caption="Supplemental Table 1. Measurements of sample A") %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down"))
```

Supplemental Table 1. Measurements of sample A

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
1	281	285	283.0	66	69	67.5	251	255	253.0	164.8	165.0	165	0.2136
2	290	288	289.0	70	69	69.5	263	258	260.5	167.4	167.4	167	0.2136
3	292	294	293.0	70	70	70.0	262	262	262.0	169.6	171.6	171	0.2130
4	278	285	281.5	63	64	63.5	254	253	253.5	164.2	170.4	167	0.2410
5	287	286	286.5	65	65	65.0	257	255	256.0	170.6	170.0	170	0.2351
6	282	282	282.0	65	65	65.0	254	257	255.5	166.2	165.6	166	0.2343
7	289	286	287.5	63	63	63.0	267	257	262.0	172.6	171.6	172	0.2587
8	286	287	286.5	61	62	61.5	278	258	268.0	169.4	173.4	171	0.2790
9	287	285	286.0	62	62	62.0	258	257	257.5	173.4	171.6	173	0.2582
10	283	285	284.0	54	55	54.5	258	254	256.0	177.4	179.2	178	0.3116
11	286	287	286.5	66	67	66.5	254	259	256.5	169.2	168.2	169	0.2260
12	287	286	286.5	67	68	67.5	259	257	258.0	168.2	166.6	167	0.2221
13	282	282	282.0	68	68	68.0	259	257	258.0	162.2	162.6	162	0.2189
14	285	287	286.0	76	70	73.0	268	260	264.0	155.4	165.0	160	0.1981
15	279	276	277.5	49	49	49.0	258	254	256.0	178.4	176.2	177	0.3578
16	283	279	281.0	49	49	49.0	263	260	261.5	181.4	178.0	180	0.3671
17	299	286	292.5	49	49	49.0	316	297	306.5	186.8	177.6	182	0.4360
18	289	284	286.5	50	50	50.0	294	282	288.0	180.2	177.6	179	0.4002
19	278	277	277.5	50	51	50.5	263	258	260.5	175.4	174.4	175	0.3523
20	294	292	293.0	49	49	49.0	308	288	298.0	183.4	185.4	184	0.4238
21	280	283	281.5	50	50	50.0	308	288	298.0	168.4	175.4	172	0.4150
22	307	291	299.0	51	49	50.0	267	266	266.5	202.6	188.8	196	0.3665
23	319	301	310.0	49	50	49.5	334	293	313.5	203.2	192.4	198	0.4414
24	291	291	291.0	50	49	49.5	383	316	349.5	164.4	178.8	172	0.4886
25	288	285	286.5	48	49	48.5	282	288	285.0	183.6	178.4	181	0.4089
26	304	304	304.0	49	51	50.0	278	265	271.5	199.4	200.0	200	0.3746

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
27	303	299	301.0	51	52	51.5	345	345	345.0	183.0	178.0	181	0.4658
28	304	303	303.5	52	49	50.5	331	323	327.0	185.8	189.4	188	0.4510
29	299	294	296.5	51	52	51.5	309	278	293.5	186.2	186.4	186	0.3956
30	290	290	290.0	51	52	51.5	309	278	293.5	177.2	182.4	180	0.3956
31	299	289	294.0	52	51	51.5	312	267	289.5	184.6	184.6	185	0.3896
32	289	283	286.0	51	52	51.5	267	260	263.5	184.6	179.0	182	0.3488
33	299	297	298.0	51	50	50.5	301	291	296.0	187.8	188.8	188	0.4078
34	303	306	304.5	50	51	50.5	321	318	319.5	188.8	191.4	190	0.4410

```
kable(smplB, booktabs=T, caption="Supplemental Table 2. Measurements of sample B") %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down"))
```

Supplemental Table 2. Measurements of sample B

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
1	196	204	200.0	72	75	73.5	104	107	105.5	103	108	105	-0.2033
2	197	209	203.0	71	75	73.0	114	112	113.0	103	112	107	-0.1705
3	210	207	208.5	75	74	74.5	112	113	112.5	113	110	112	-0.1812
4	202	202	202.0	72	72	72.0	111	109	110.0	108	108	108	-0.1762
5	203	203	203.0	73	72	72.5	109	110	109.5	108	109	109	-0.1811
6	208	203	205.5	73	73	73.0	117	105	111.0	112	109	110	-0.1782
7	223	215	219.0	70	70	70.0	159	145	152.0	121	116	119	-0.0235
8	223	204	213.5	67	68	67.5	164	108	136.0	123	114	119	-0.0560
9	200	206	203.0	68	69	68.5	97	100	98.5	113	117	115	-0.2025
10	204	204	204.0	61	62	61.5	100	97	98.5	123	123	123	-0.1557
11	204	205	204.5	73	75	74.0	113	96	104.5	108	111	110	-0.2103
12	205	203	204.0	75	76	75.5	96	99	97.5	111	107	109	-0.2492
13	195	202	198.5	76	77	76.5	97	97	97.0	100	106	103	-0.2571
14	204	204	204.0	76	77	76.5	137	97	117.0	101	108	104	-0.1757
15	197	195	196.0	54	54	54.0	105	98	101.5	122	121	122	-0.0861
16	196	193	194.5	54	54	54.0	103	100	101.5	121	119	120	-0.0861
17	189	191	190.0	52	53	52.5	120	97	108.5	113	119	116	-0.0450
18	195	192	193.5	54	55	54.5	106	96	101.0	120	118	119	-0.0923
19	194	185	189.5	54	54	54.0	96	95	95.5	121	112	116	-0.1126
20	194	192	193.0	54	54	54.0	104	99	101.5	119	118	119	-0.0861
21	196	192	194.0	54	54	54.0	99	120	109.5	122	114	118	-0.0532
22	196	197	196.5	54	54	54.0	96	124	110.0	123	118	121	-0.0512
23	196	199	197.5	54	54	54.0	109	99	104.0	120	125	123	-0.0756

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
24	199	197	198.0	54	51	52.5	101	100	100.5	125	126	125	-0.0782
25	196	196	196.0	53	54	53.5	100	101	100.5	123	122	122	-0.0864
26	193	197	195.0	55	55	55.0	100	101	100.5	118	122	120	-0.0984
27	196	191	193.5	54	54	54.0	107	126	116.5	121	112	116	-0.0263
28	201	200	200.5	53	49	51.0	105	116	110.5	127	128	127	-0.0244
29	198	201	199.5	55	54	54.5	104	103	103.5	122	126	124	-0.0817
30	200	222	211.0	54	54	54.0	109	106	107.5	124	147	136	-0.0612
31	201	195	198.0	53	51	52.0	107	108	107.5	127	122	125	-0.0448
32	204	200	202.0	54	53	53.5	108	106	107.0	128	126	127	-0.0592
33	203	202	202.5	54	53	53.5	113	110	111.5	126	127	127	-0.0413
34	201	207	204.0	53	54	53.5	111	111	111.0	126	131	128	-0.0433

```
kable(smplC, booktabs=T, caption="Supplemental Table 3. Measurements of sample C") %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down"))
```

Supplemental Table 3. Measurements of sample C

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
1	269	269	269.0	90	91	90.5	106	107	106.5	157.8	156.6	157	-0.2895
2	273	276	274.5	92	93	92.5	110	108	109.0	159.0	161.4	160	-0.2889
3	276	277	276.5	93	93	93.0	108	112	110.0	161.4	161.6	162	-0.2873
4	257	255	256.0	85	84	84.5	103	101	102.0	151.4	150.8	151	-0.2785
5	258	257	257.5	85	85	85.0	103	101	102.0	152.4	151.8	152	-0.2810
6	258	257	257.5	85	86	85.5	101	105	103.0	152.8	150.0	151	-0.2794
7	258	256	257.0	83	83	83.0	103	101	102.0	154.4	152.8	154	-0.2707
8	262	255	258.5	82	82	82.0	102	101	101.5	159.6	152.8	156	-0.2676
9	261	261	261.0	81	81	81.0	104	104	104.0	159.2	159.2	159	-0.2517
10	260	258	259.0	74	74	74.0	103	105	104.0	165.4	163.0	164	-0.2124
11	257	259	258.0	91	93	92.0	103	105	104.0	145.4	145.0	145	-0.3070
12	259	263	261.0	93	99	96.0	105	104	104.5	145.0	143.2	144	-0.3234
13	262	262	262.0	93	94	93.5	108	105	106.5	147.4	147.0	147	-0.3037
14	260	262	261.0	94	93	93.5	104	105	104.5	145.2	148.0	147	-0.3119
15	252	252	252.0	66	66	66.0	106	105	105.5	164.8	165.0	165	-0.1565
16	254	252	253.0	66	66	66.0	105	107	106.0	167.0	164.6	166	-0.1545
17	255	254	254.5	66	66	66.0	112	109	110.5	166.6	166.2	166	-0.1364
18	251	252	251.5	67	67	67.0	104	107	105.5	163.2	163.6	163	-0.1630
19	254	253	253.5	66	67	66.5	106	109	107.5	166.8	164.2	166	-0.1516
20	254	257	255.5	67	67	67.0	111	106	108.5	164.8	168.8	167	-0.1509

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
21	257	257	257.0	67	67	67.0	111	106	108.5	167.8	168.8	168	-0.1509
22	255	258	256.5	67	67	67.0	110	119	114.5	166.0	167.2	167	-0.1275
23	260	257	258.5	67	68	67.5	113	113	113.0	170.4	166.4	168	-0.1364
24	265	258	261.5	66	66	66.0	124	112	118.0	174.2	169.6	172	-0.1079
25	260	258	259.0	67	67	67.0	110	109	109.5	171.0	169.2	170	-0.1469
26	256	258	257.0	68	68	68.0	110	109	109.5	166.0	168.2	167	-0.1533
27	263	263	263.0	69	68	68.5	119	119	119.0	170.2	171.2	171	-0.1204
28	264	261	262.5	67	67	67.0	113	118	115.5	174.4	170.4	172	-0.1237
29	261	262	261.5	68	68	68.0	116	110	113.0	169.8	172.0	171	-0.1397
30	260	262	261.0	68	68	68.0	114	119	116.5	169.2	170.2	170	-0.1264
31	264	264	264.0	67	67	67.0	117	114	115.5	173.6	174.2	174	-0.1237
32	266	261	263.5	68	68	68.0	119	114	116.5	174.2	170.2	172	-0.1264
33	264	222	243.0	67	67	67.0	119	120	119.5	173.2	131.0	152	-0.1089
34	262	268	265.0	67	68	67.5	118	118	118.0	171.4	176.4	174	-0.1176

```
kable(smplD, booktabs=T, caption="Supplemental Table 4. Measurements of sample D") %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down"))
```

Supplemental Table 4. Measurements of sample D

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
1	139	144	141.5	44	45	44.5	185	190	187.5	58.0	61.0	60	0.2644
2	147	147	147.0	44	44	44.0	185	183	184.0	66.0	66.4	66	0.2611
3	145	148	146.5	44	44	44.0	185	185	185.0	64.0	67.0	66	0.2635
4	148	148	148.0	43	43	43.0	186	186	186.0	67.8	67.8	68	0.2758
5	149	146	147.5	42	43	42.5	190	183	186.5	69.0	66.4	68	0.2821
6	147	147	147.0	43	43	43.0	187	191	189.0	66.6	65.8	66	0.2828
7	148	149	148.5	41	41	41.0	186	187	186.5	69.8	70.6	70	0.2977
8	151	148	149.5	39	39	39.0	189	187	188.0	74.2	71.6	73	0.3229
9	150	147	148.5	39	39	39.0	187	191	189.0	73.6	69.8	72	0.3252
10	149	152	150.5	32	32	32.0	193	189	191.0	78.4	82.2	80	0.4157
11	148	152	150.0	38	39	38.5	187	188	187.5	72.6	75.4	74	0.3273
12	152	151	151.5	39	39	39.0	188	188	188.0	75.4	74.4	75	0.3229
13	151	152	151.5	39	28	33.5	189	187	188.0	74.2	86.6	80	0.3889
14	150	154	152.0	39	39	39.0	188	188	188.0	73.4	77.4	75	0.3229
15	140	139	139.5	28	28	28.0	189	190	189.5	74.2	73.0	74	0.4702
16	138	140	139.0	27	28	27.5	191	198	194.5	72.8	72.4	73	0.4894
17	138	140	139.0	27	28	27.5	190	190	190.0	73.0	74.0	74	0.4792

day	CHOLrep1	CHOLrep2	CHOL	HDLrep1	HDLrep2	HDL	TGrep1	TGrep2	TG	LDLrep1	LDLrep2	LDL	AIP
18	139	141	140.0	28	28	28.0	193	195	194.0	72.4	74.0	73	0.4804
19	141	139	140.0	28	28	28.0	191	192	191.5	74.8	72.6	74	0.4748
20	140	142	141.0	28	28	28.0	193	190	191.5	73.4	76.0	75	0.4748
21	142	141	141.5	28	28	28.0	193	190	191.5	75.4	75.0	75	0.4748
22	141	142	141.5	28	28	28.0	197	192	194.5	73.6	75.6	75	0.4815
23	143	144	143.5	27	27	27.0	198	197	197.5	76.4	77.6	77	0.5040
24	148	144	146.0	28	27	27.5	199	193	196.0	80.2	78.4	79	0.4927
25	147	146	146.5	28	29	28.5	194	194	194.0	80.2	78.2	79	0.4727
26	146	146	146.0	29	29	29.0	201	198	199.5	76.8	77.4	77	0.4773
27	147	147	147.0	28	28	28.0	198	199	198.5	79.4	79.2	79	0.4904
28	147	146	146.5	28	28	28.0	197	200	198.5	79.6	78.0	79	0.4904
29	148	148	148.0	28	28	28.0	200	198	199.0	80.0	80.4	80	0.4915
30	149	148	148.5	28	28	28.0	201	199	200.0	80.8	80.2	81	0.4937
31	149	150	149.5	27	27	27.0	201	201	201.0	81.8	82.8	82	0.5116
32	151	151	151.0	28	28	28.0	205	204	204.5	82.0	82.2	82	0.5033
33	154	154	154.0	27	28	27.5	208	210	209.0	85.4	84.0	85	0.5206
34	154	157	155.5	29	29	29.0	208	210	209.0	83.4	86.0	85	0.4975

Define lists for CHOL, HDL and TG samples

```
lstDF <- list(smplA, smplB, smplC, smplD)
vecSmplNames = c("Sample A", "Sample B", "Sample C", "Sample D")
```

Descriptive statistics

For Cholesterol

```
CHOL_Means = unlist( lapply( lstDF, function(DF) { round(mean(DF$CHOL))} ) )
CHOL_Vrncs = unlist(lapply(lstDF, function(DF) {round(var(DF$CHOL),1)}))
CHOL_CVs = unlist(lapply(lstDF, function(DF) {round(sd(DF$CHOL)/mean(DF$CHOL)*100,1)})) # CV
dfCHOLDescStats = as.data.frame(rbind(CHOL_Means, CHOL_Vrncs, CHOL_CVs))
colnames(dfCHOLDescStats) = vecSmplNames
CHOLDescStats <-
  unlist(
    split(
      t(dfCHOLDescStats), seq(nrow(t(dfCHOLDescStats)))
      ))
```

For HDL

```
HDL_Means = unlist( lapply( lstDF, function(DF) { round(mean(DF$HDL))} ) )
HDL_Vrncs = unlist(lapply(lstDF, function(DF) {round(var(DF$HDL),1)}))
HDL_CVs = unlist(lapply(lstDF, function(DF) {round(sd(DF$HDL)/mean(DF$HDL)*100,1)})) # CV
dfHDLDescStats = as.data.frame(rbind(HDL_Means, HDL_Vrncs, HDL_CVs))
colnames(dfHDLDescStats) = vecSmplNames
HDLDescStats <-
  unlist(
    split(
      t(dfHDLDescStats), seq(nrow(t(dfHDLDescStats)))
      ))
```

For TG

```
TG_Means = unlist( lapply( lstDF, function(DF) { round(mean(DF$TG))} ) )
TG_Vrncs = unlist(lapply(lstDF, function(DF) {round(var(DF$TG),1)}))
TG_CVs = unlist(lapply(lstDF, function(DF) {round(sd(DF$TG)/mean(DF$TG)*100,1)})) # CV
dfTGDescStats = as.data.frame(rbind(TG_Means, TG_Vrncs, TG_CVs))
colnames(dfTGDescStats) = vecSmplNames
TGDescStats <-
  unlist(
    split(
      t(dfTGDescStats), seq(nrow(t(dfTGDescStats)))
      ))
```

Suppl.Tbl 5 Descriptive statistics

```
supplTable5 <-
  rbind(CHOLDescStats, HDLDescStats, TGDescStats)
supplTable5 <-  cbind(c("CHOL", "HDL", "TG"), supplTable5)
colnames(supplTable5) <- c("Parameter", rep(c("Mean", "Variance", "CV"),4))
rownames(supplTable5) <- NULL
kable(supplTable5, booktabs=T, caption="Supplemental table 5. Mean, Variance and CV of TG for the four sample
s") %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down")) %>%
  add_header_above(c(" " = 1, "Sample A" = 3, "Sample B" = 3, "Sample C" = 3,  "Sample D" = 3))
```

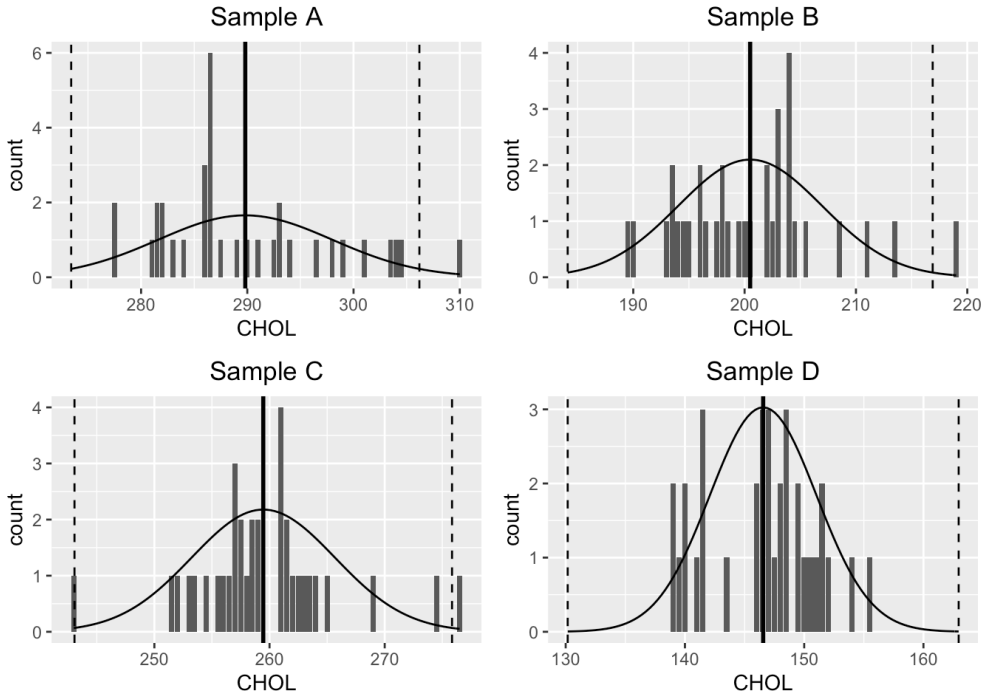
Supplemental table 5. Mean, Variance and CV of TG for the four samples

	Sample A			Sample B			Sample C			Sample D		
Parameter	**Mean**	**Variance**	**CV**	**Mean**	**Variance**	**CV**	**Mean**	**Variance**	**CV**	**Mean**	**Variance**	**CV**
CHOL	290	67.2	2.8	200	41.8	3.2	259	38.8	2.4	147	20.1	3.1
HDL	56	67.2	14.5	61	91.8	15.7	76	120.8	14.5	33	43.2	19.9
TG	279	744	9.8	108	119.3	10.1	109	31.4	5.1	193	43.5	3.4

Discrete distribution histograms of CHOL, HDL, TG

Suppl.Fig. 1 CHOL distribution

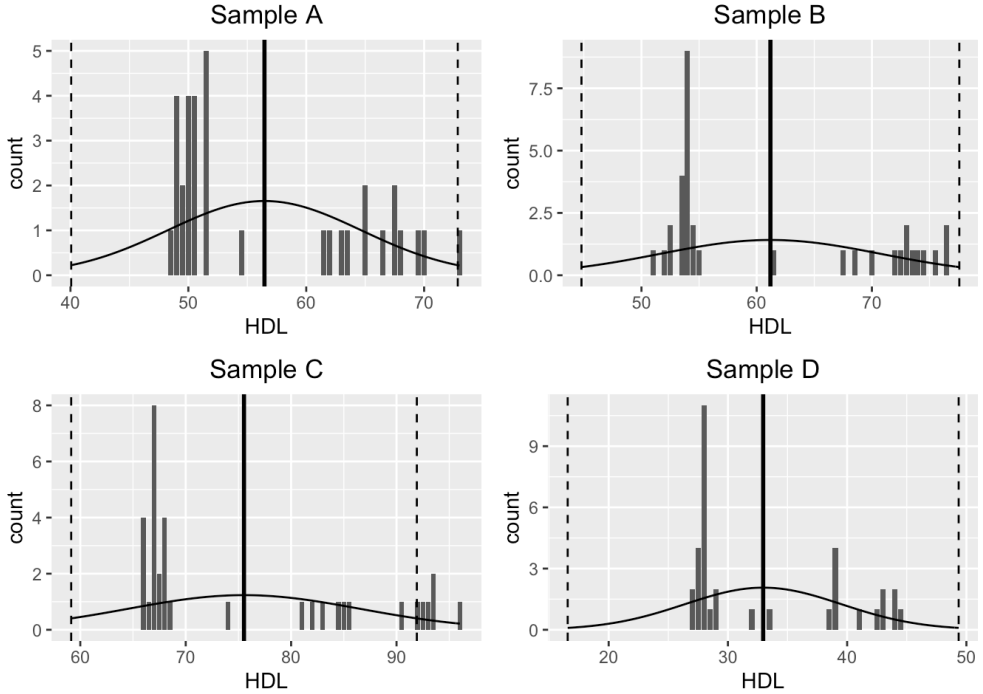
```
fPlotDiscrHist("CHOL")
```



Distribution of Cholesterol values of the 4 samples. The mean is indicated with a thick vertical lines and ±2SD with thin dashed vertical lines. The normal distribution with mean and standard deviation equal to these of the sample are superimposed.

Suppl.Fig. 2 HDL distribution

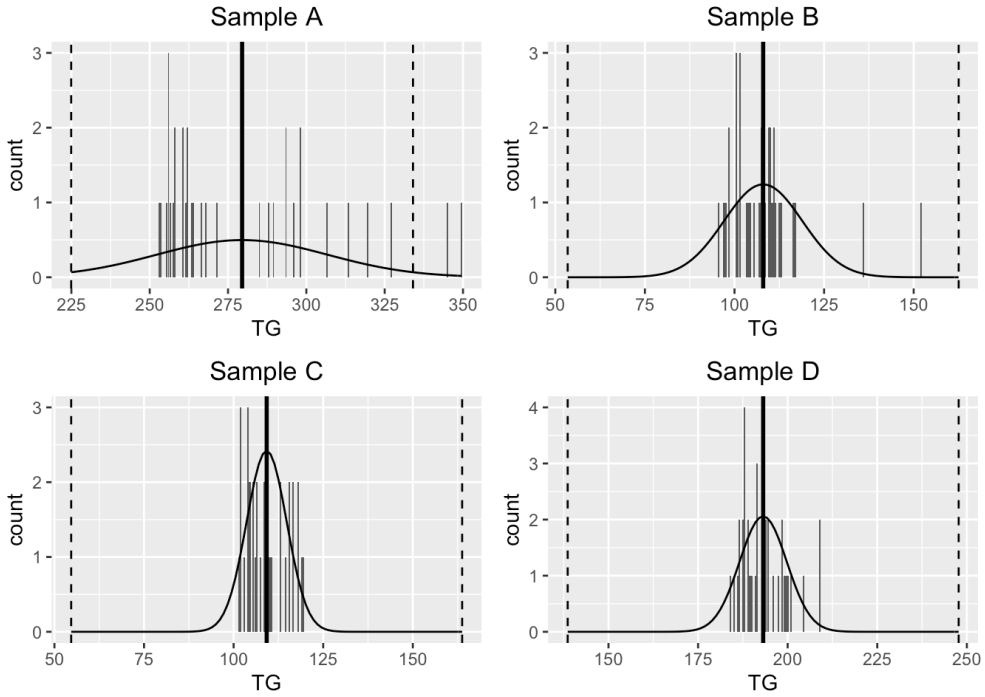
```
fPlotDiscrHist("HDL")
```



Distribution of HDL values of the 4 samples. The mean is indicated with a thick vertical lines and ±2SD with thin dashed vertical lines. The normal distribution with mean and standard deviation equal to these of the sample are superimposed.

Suppl.Fig. 3 TG distribution

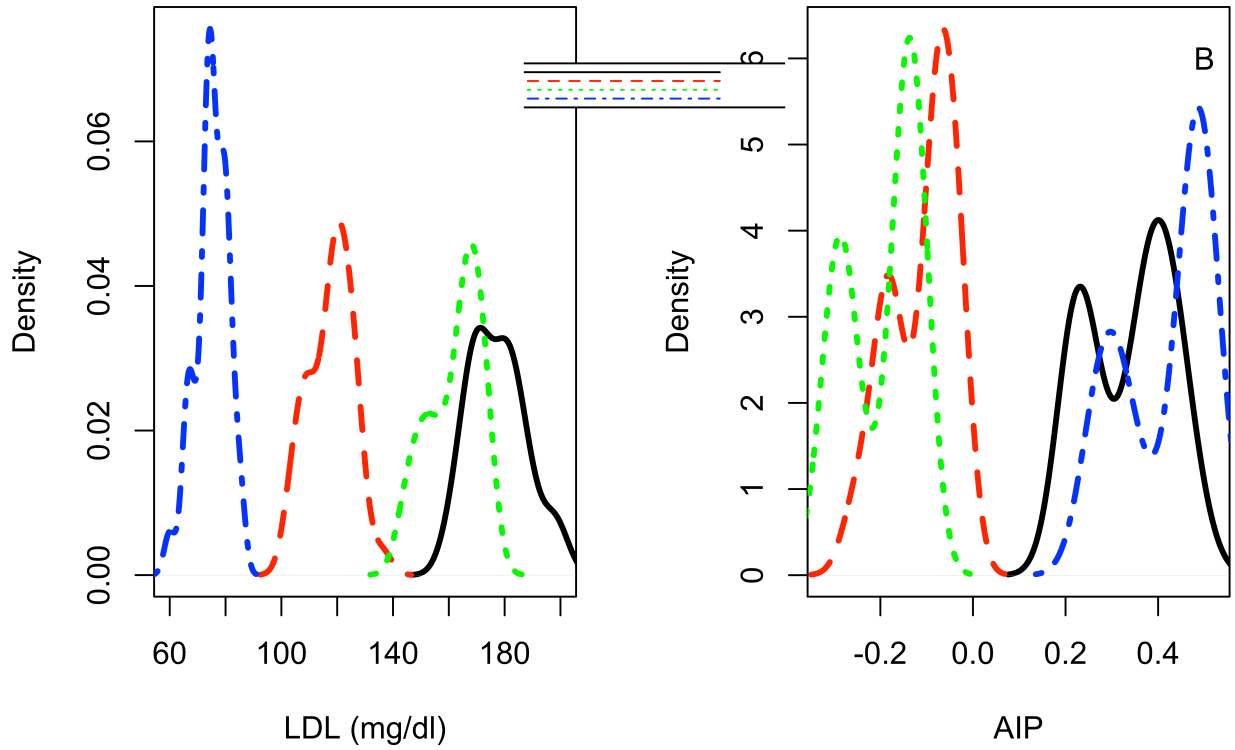
```
fPlotDiscrHist("TG")
```



Distribution of TG values of the 4 samples. The mean is indicated with a thick vertical lines and ±2SD with thin dashed vertical lines. The normal distribution with mean and standard deviation equal to these of the sample are superimposed.

Suppl.Fig. 4 Density of LDL and AIP

```
xMinLDL=min(smplA$LDL, smplB$LDL, smplC$LDL, smplD$LDL)
xMaxLDL=max(smplA$LDL, smplB$LDL, smplC$LDL, smplD$LDL)
yMinLDL=min(density(smplA$LDL)$y, density(smplB$LDL)$y ,density(smplC$LDL)$y, density(smplD$LDL)$y)
yMaxLDL=max(density(smplA$LDL)$y, density(smplB$LDL)$y ,density(smplC$LDL)$y, density(smplD$LDL)$y)
xMinAIP=min(smplA$AIP, smplB$AIP, smplC$AIP, smplD$AIP)
xMaxAIP=max(smplA$AIP, smplB$AIP, smplC$AIP, smplD$AIP)
yMinAIP=min(density(smplA$AIP)$y, density(smplB$AIP)$y ,density(smplC$AIP)$y, density(smplD$AIP)$y)
yMaxAIP=max(density(smplA$AIP)$y, density(smplB$AIP)$y ,density(smplC$AIP)$y, density(smplD$AIP)$y)

par(mfrow=c(1,2))
plot(density(smplA$LDL), main="", xlab="LDL (mg/dl)", lty=1, lwd=3, col="black",
     xlim=c(xMinLDL, xMaxLDL), ylim=c(yMinLDL, yMaxLDL))
lines(density(smplB$LDL), main="", xlab="LDL (mg/dl)", lty=2, lwd=3, col="red")
lines(density(smplC$LDL), main="", xlab="LDL (mg/dl)", lty=3, lwd=3, col="green")
lines(density(smplD$LDL), main="", xlab="LDL (mg/dl)", lty=4, lwd=3, col = "blue")
text(x=250, y=0.07, "A")

plot(density(smplA$AIP), main="",  xlab="AIP", lty=1, lwd=3, col="black",
     xlim=c(xMinAIP, xMaxAIP), ylim=c(yMinAIP, yMaxAIP))
lines(density(smplB$AIP), main="", xlab="AIP (mg/dl)", lty=2, lwd=3, col="red")
lines(density(smplC$AIP), main="", xlab="AIP (mg/dl)", lty=3, lwd=3, col="green")
lines(density(smplD$AIP), main="", xlab="AIP (mg/dl)", lty=4, lwd=3, col="blue")
text(x=0.5, y=6, "B")

par(xpd=T)
par(fig=c(0.40, 0.60, 0.70, 0.80), new=T)
legend("center",  lty=c(4,3,2,1), col=c("blue", "green", "red", "black"),
       legend=c("Sample D", "Sample C" ,"Sample B", "Sample A"))
```



Density plot of LDL and AIP for the four samples

```
par(xpd=F)
tiff("TblFig2.pdf", units="cm", width=29.7, height=21.0, res=300)
dev.off()
```

```
## quartz_off_screen
##                 2
```

Correlations and covariances

Suppl.Tbl 6. Corrrelation and covariance between (CHOL-HDL), (CHOL-TG) and (HDL-TG) for all 4 samples

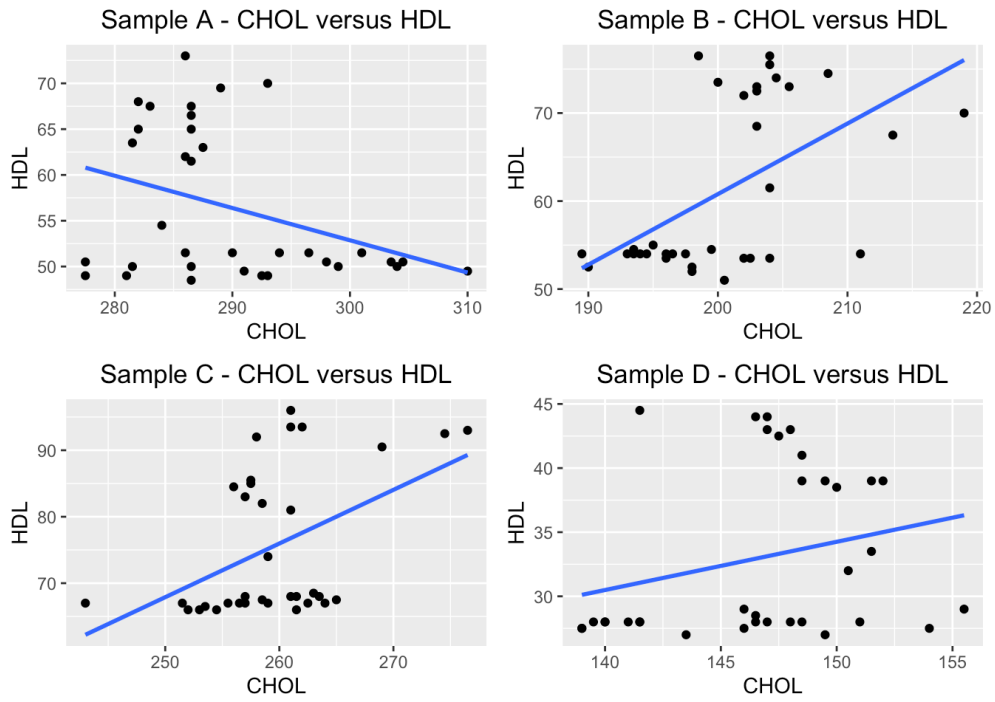
```
  CHOL_HDL_cor = unlist(lapply(lstDF, function(DF) { round(cor(DF$CHOL, DF$HDL),2) }))
  CHOL_HDL_cov = unlist(lapply(lstDF, function(DF) { round(cov(DF$CHOL, DF$HDL),2) }))
  CHOL_TG_cor  = unlist(lapply(lstDF, function(DF) {round(cor(DF$CHOL, DF$TG),2)}))
  CHOL_TG_cov  = unlist(lapply(lstDF, function(DF) { round(cov(DF$CHOL, DF$TG),2) }))
  HDL_TG_cor   = unlist(lapply(lstDF, function(DF) {round(cor(DF$HDL, DF$TG),2)}))
  HDL_TG_cov   = unlist(lapply(lstDF, function(DF) {round(cov(DF$HDL, DF$TG),2)}))

  CHOL_HDL_stats = c(matrix(c(CHOL_HDL_cor, CHOL_HDL_cov), 2, byrow=T))
  CHOL_TG_stats = c(matrix(c(CHOL_TG_cor, CHOL_TG_cov), 2, byrow=T))
  HDL_TG_stats = c(matrix(c(HDL_TG_cor, HDL_TG_cov), 2, byrow=T))
  dfCorCov <-  rbind(CHOL_HDL_stats, CHOL_TG_stats, HDL_TG_stats)
  dfCorCov <- cbind(c("Cor(CHOL,HDL)", "Cor(CHOL,TG)", "Cor(HDL,TG)"), dfCorCov)
  rownames(dfCorCov) <- NULL
  colnames(dfCorCov) <- c("Correlation \n between", rep(c("Correlation", "Covariance"), 4) )
  knitr::kable(dfCorCov, booktabs=T, caption="Supplemental Table 2. Corrrelation and covariance between (CHOL-
HDL), (CHOL-TG) and (HDL-TG) for the four samples") %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down")) %>%
  add_header_above(c(" " = 1, "Sample A" = 2, "Sample B" = 2, "Sample C" = 2,  "Sample D" = 2))
```

Supplemental Table 2. Corrrelation and covariance between (CHOL-HDL), (CHOL-TG) and (HDL-TG) for the four samples

	Sample A		Sample B		Sample C		Sample D	
Correlation between	**Correlation**	**Covariance**	**Correlation**	**Covariance**	**Correlation**	**Covariance**	**Correlation**	**Covariance**
Cor(CHOL,HDL)	-0.35	-23.68	0.54	33.54	0.46	31.31	0.26	7.56
Cor(CHOL,TG)	0.64	143.93	0.66	46.62	0.12	4.11	0.26	7.8
Cor(HDL,TG)	-0.59	-132.24	0.23	23.73	-0.58	-36.03	-0.72	-31.34

Suppl.Fig. 5 CHOL vs HDL scatterplot

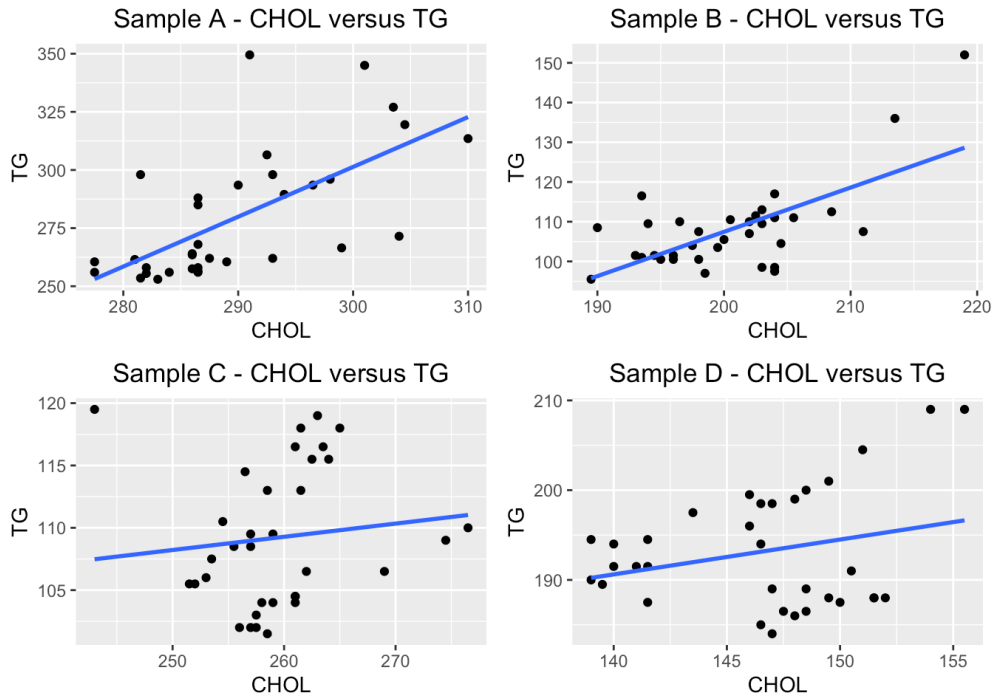
```
fPlotCorrWithRegrLine("CHOL", "HDL")
```



Scatterplot of CHOL versus HDL for the four samples along with the corresponding linear regression line.

Suppl.Fig. 6 CHOL vs TG scatterplot

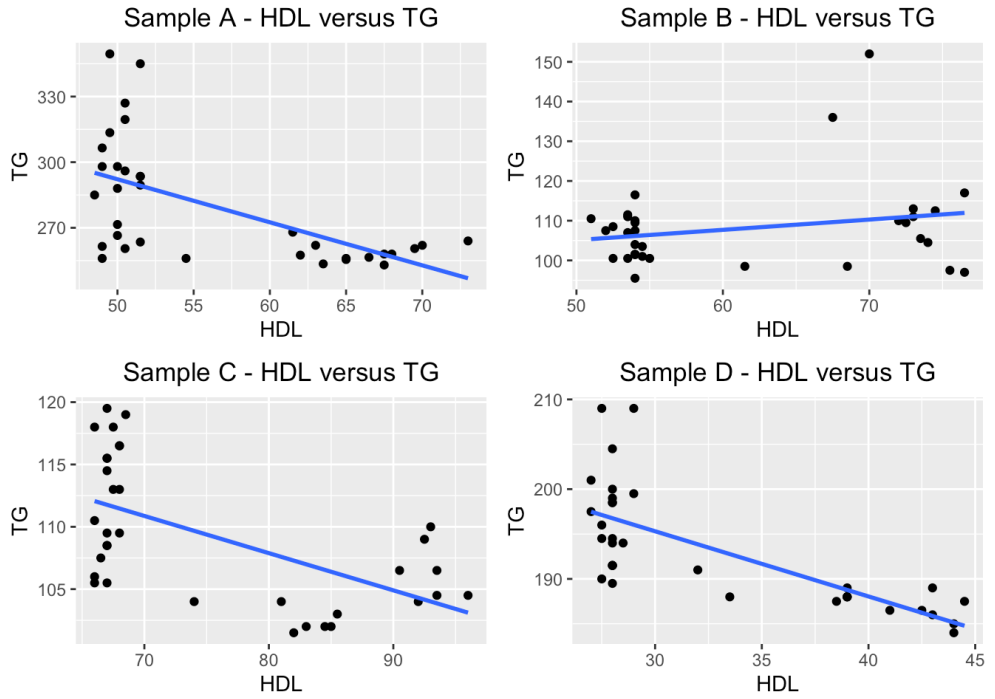
```
fPlotCorrWithRegrLine("CHOL", "TG")
```



Scatterplot of CHOL versus TG for the four samples along with the corresponding linear regression line.

Suppl.Fig. 7 HDL vs TG scatterplot

```
fPlotCorrWithRegrLine("HDL", "TG")
```



Scatterplot of HDL versus TG for the four samples along with the corresponding liner regression line.

Suppl.Fig.8. Mapping of distribution

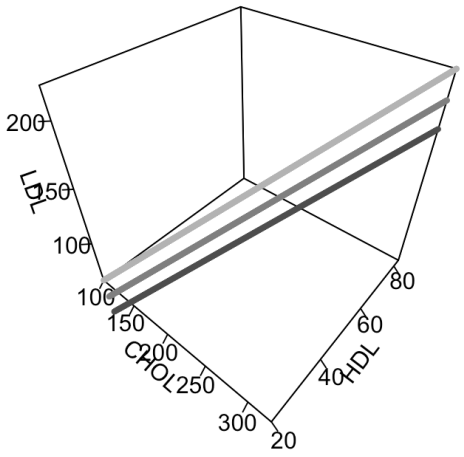
```
knitr::include_graphics("SupplemetalFigure8.tiff")
```

Mapping of the distribution of an independent random variable X to the (distorted) distribution of the dependent random variable Y, via the function $f(X)$.
Mapping of the distribution of an independent random variable X to the (distorted) distribution of the dependent random variable Y, via the function $f(X)$.

Suppl.Fig.9. 3D plot of LDL versus CHOL and HDL for

TG=100, 200 and 300 mg/dl

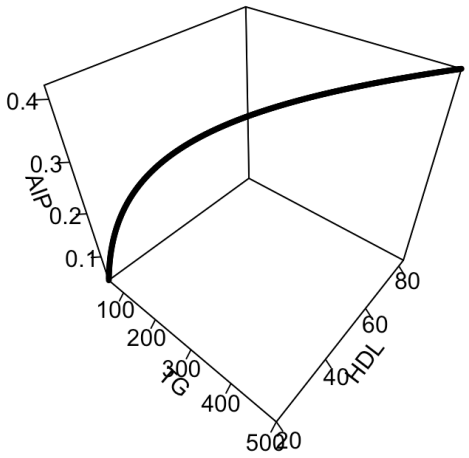
```
CHOL <- seq(100,325,0.5)
HDL <- seq(20,83,0.14)
TG100 <- rep(100,length(CHOL))
TG200 <- rep(200,length(CHOL))
TG300 <- rep(300,length(CHOL))
LDLforTG100 <- CHOL - HDL - TG100/5
LDLforTG200 <- CHOL - HDL - TG200/5
LDLforTG300 <- CHOL - HDL - TG300/5
scatter3D(CHOL, HDL, LDLforTG100, type="l", colvar = NULL, xlab="CHOL", ylab="HDL", zlab="LDL",
          lwd=4, ticktype="detailed", col="gray70")
scatter3D(CHOL, HDL, LDLforTG200, type="l", colvar = NULL, xlab="CHOL", ylab="HDL", zlab="LDL",
          lwd=4, ticktype="detailed", col="gray50", add=T)
scatter3D(CHOL, HDL, LDLforTG300, type="l", colvar = NULL, xlab="CHOL", ylab="HDL", zlab="LDL",
          lwd=4, ticktype="detailed", col="gray30", add=T)
```



Plot of LDL versus CHOL and HDL for TG=100, 200 and 300 mg/dl (from top to bottom)

Suppl.Fig.10. 3D Plot of AIP versus HDL and TG

```
TG <- seq(50,500,1)
HDL <- seq(20,83,0.14)
AIP <- log10( (TG*0.0113) / (HDL*0.0259) )
scatter3D(TG,HDL,AIP, type="l", colvar = NULL, xlab="TG", ylab="HDL", zlab="AIP", ticktype="detailed", lwd=4)
```



Plot of AIP versus HDL and TG

Variance calculation

LDL

Empirical variance

```
dfLDLExperVrnc = # Experimental variance
  flst2DF(
    lapply(lstDF,
         function(DF) {
           round(var(DF[,"LDL"]),2)
                     } ))
```

Error Propagation variance

```
dfLDLErrPrpVrnc=
  flst2DF(
    lapply(lstDF,
           function(DF) {
             round(fLDLErrPrp(DF[,"CHOL"],  DF[,"HDL"], DF[,"TG"]),2)
                       }))
```

Bootstap variance

Calculate bootstrap variance of LDL for the 4 samples. **Results may not be exactly the same with each execution of the code because of the random sampling**.

```
list(LDLSmplABootVrnc, LDLSmplBBootVrnc, LDLSmplCBootVrnc, LDLSmplDBootVrnc) %tin%
  lapply(lstDF, function(DF) {fLDLBoot(DF$CHOL, DF$HDL, DF$TG, noOfReps=1e4)})
# Get the variance mean
LDLBootVrnc = round(unlist(lapply(list(LDLSmplABootVrnc, LDLSmplBBootVrnc, LDLSmplCBootVrnc, LDLSmplDBootVrn
c), function(DF) {DF$Var})),1)
# Get the variance 95% quantile
LDLBootVrncLow95Quant = unlist(lapply(list(LDLSmplABootVrnc, LDLSmplBBootVrnc, LDLSmplCBootVrnc, LDLSmplDBootV
rnc), function(DF) {
  quantile(DF$dataTable$Var, probs=c(0.025, 0.975))[[1]]}))
LDLBootVrncUpper95Quant = unlist(lapply(list(LDLSmplABootVrnc, LDLSmplBBootVrnc, LDLSmplCBootVrnc, LDLSmplDBoo
tVrnc), function(DF) {
  quantile(DF$dataTable$Var, probs=c(0.025, 0.975))[[2]]}))
LDLBootVrnc95Quant = paste(round(LDLBootVrncLow95Quant,1), round(LDLBootVrncUpper95Quant,1), sep="-")
dfLDLBoot = rbind(LDLBootVrnc, LDLBootVrnc95Quant)
colnames(dfLDLBoot) = vecSmplNames
rownames(dfLDLBoot) = c("Median", "Central 95 percentile range")
```

Combine empirical, error propagation and bootstrap variance into one data frame

```
dfLDLVrnc = cbind(vecSmplNames, dfLDLExperVrnc, dfLDLErrPrpVrnc, t(dfLDLBoot))
dfLDLVrnc$vecSmplNames <- NULL
colnames(dfLDLVrnc) = c("Empirical distribution Variance", "Error Propagation Variance", "Bootstrap variance m
edian", "Bootstrap variance 95 percentile")
```

AIP

Empirical variance

```
dfAIPExperVrnc = # Experimental variance
  flst2DF(
    lapply(lstDF,
         function(DF) {
           round(var(DF[,"AIP"]),5)
                     } ))
```

Error Propagation variance

```
dfAIPErrPrpVrnc=
  flst2DF(
    lapply(lstDF,
           function(DF) {
             round(fAIPErrPrp(DF[,"TG"], DF[,"HDL"]),5)
                       }))
```

2nd Order Taylor Error propagation variance

```
dfAIPErrPrpVrnc2Ord <-
  flst2DF(
    lapply(lstDF,
           function(DF) {
             round(fAIPErrPrp2Ord(DF[,"TG"], DF[,"HDL"]),5)
                       }))
```

Bootstap variance

Calculate bootstrap variance of LDL for the 4 samples. **Results may not be exactly the same with each execution of the code because of the random sampling**.

```
list(AIPSmplABootVrnc, AIPSmplBBootVrnc, AIPSmplCBootVrnc, AIPSmplDBootVrnc) %tin%
  lapply(lstDF, function(DF) {fAIPBoot(DF$TG, DF$HDL, noOfReps=1e4)})
# Get the variance mean
AIPBootVrnc = round(unlist(lapply(list(AIPSmplABootVrnc, AIPSmplBBootVrnc, AIPSmplCBootVrnc, AIPSmplDBootVrn
c), function(DF) {DF$Var})),4)
# Get the variance 95% quantile
AIPBootVrncLow95Quant <-
  unlist(
    lapply(
      list(AIPSmplABootVrnc, AIPSmplBBootVrnc, AIPSmplCBootVrnc, AIPSmplDBootVrnc),
      function(DF) {
        quantile(DF$dataTable$Var, probs=c(0.025, 0.975))[[1]]
        }))

AIPBootVrncUpper95Quant <-
  unlist(
    lapply(
      list(AIPSmplABootVrnc, AIPSmplBBootVrnc, AIPSmplCBootVrnc, AIPSmplDBootVrnc),
      function(DF) {
        quantile(DF$dataTable$Var, probs=c(0.025, 0.975))[[2]]
        }))

AIPBootVrnc95Quant <- paste(
  round(AIPBootVrncLow95Quant,4),
  round(AIPBootVrncUpper95Quant,4),
  sep="-")

dfAIPBoot = rbind(AIPBootVrnc, AIPBootVrnc95Quant)
colnames(dfAIPBoot) = vecSmplNames
rownames(dfAIPBoot) = c("Median", "Central 95 percentile range")
```

Combine empirical, error propagation and bootstrap variance into one data frame

```
dfAIPVrnc = cbind(vecSmplNames, dfAIPExperVrnc, dfAIPErrPrpVrnc, dfAIPErrPrpVrnc2Ord,  t(dfAIPBoot))
dfAIPVrnc$vecSmplNames <- NULL
colnames(dfAIPVrnc) = c("Empirical distribution Variance", "Error Propagation Variance",
                        "2nd Order Error Propagation Variance",
                        "Bootstrap variance median", "Bootstrap variance 95 percentile")
```

Combine LDL and AIP Variances

Suppl.Table 7.

```
dfLDL_AIPVrnc <- rbind(t(dfLDLVrnc), t(dfAIPVrnc))
dfLDL_AIPVrnc
```

```
##                                    Sample A       Sample B
## Empirical distribution Variance      "102.80"       " 62.43"
## Error Propagation Variance           "101.01"       " 62.18"
## Bootstrap variance median            "97.3"         "59.7"
## Bootstrap variance 95 percentile     "59.1-142.2"   "37.7-86.5"
## Empirical distribution Variance      "0.00838"      "0.00477"
## Error Propagation Variance           "0.00893"      "0.00520"
## 2nd Order Error Propagation Variance "0.00885"      "0.00517"
## Bootstrap variance median            "0.0081"       "0.0046"
## Bootstrap variance 95 percentile     "0.006-0.0103" "0.003-0.0063"
##                                    Sample C        Sample D
## Empirical distribution Variance      " 82.72"        " 33.24"
## Error Propagation Variance           " 82.11"        " 34.21"
## Bootstrap variance median            "79.5"          "32.5"
## Bootstrap variance 95 percentile     "51.5-107.1"    "18.7-50.3"
## Empirical distribution Variance      "0.00580"       "0.00883"
## Error Propagation Variance           "0.00614"       "0.00957"
## 2nd Order Error Propagation Variance "0.00607"       "0.00958"
## Bootstrap variance median            "0.0057"        "0.0087"
## Bootstrap variance 95 percentile     "0.0042-0.0068" "0.0061-0.0106"
```

```
knitr::kable(dfLDL_AIPVrnc, booktabs=T, caption="Supplemental Table 3. Variance of LDL and AIP.") %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down")) %>%
  pack_rows("LDL", 1,4) %>%
  pack_rows("AIP", 5, 9 )
```

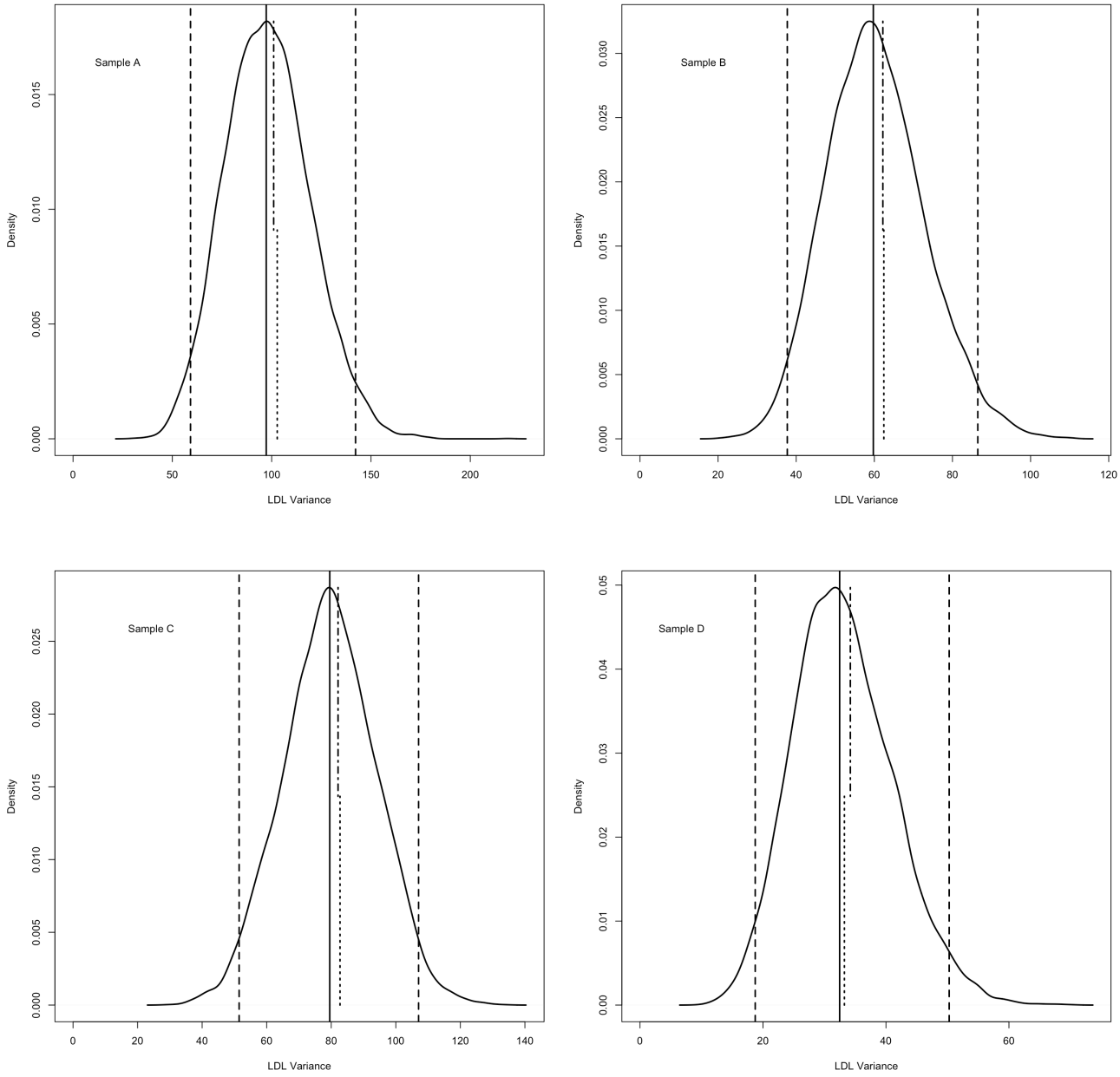
Supplemental Table 3. Variance of LDL and AIP.

	Sample A	Sample B	Sample C	Sample D
LDL				
Empirical distribution Variance	102.80	62.43	82.72	33.24
Error Propagation Variance	101.01	62.18	82.11	34.21
Bootstrap variance median	97.3	59.7	79.5	32.5
Bootstrap variance 95 percentile	59.1-142.2	37.7-86.5	51.5-107.1	18.7-50.3
AIP				
Empirical distribution Variance	0.00838	0.00477	0.00580	0.00883
Error Propagation Variance	0.00893	0.00520	0.00614	0.00957
2nd Order Error Propagation Variance	0.00885	0.00517	0.00607	0.00958
Bootstrap variance median	0.0081	0.0046	0.0057	0.0087
Bootstrap variance 95 percentile	0.006-0.0103	0.003-0.0063	0.0042-0.0068	0.0061-0.0106

Posterior densities

Suppl.Fig. 11 LDL Distribution density

```
# setEPS()
# postscript("Figure1.eps")
par(mfrow=c(2,2))
  fPltLDLVrncDens(LDLSmplABootVrnc$dataTable, "Sample A", dfLDLVrnc[1,1], dfLDLVrnc[1,2])
  fPltLDLVrncDens(LDLSmplBBootVrnc$dataTable, "Sample B", dfLDLVrnc[2,1], dfLDLVrnc[2,2])
  fPltLDLVrncDens(LDLSmplCBootVrnc$dataTable, "Sample C", dfLDLVrnc[3,1], dfLDLVrnc[3,2])
  fPltLDLVrncDens(LDLSmplDBootVrnc$dataTable, "Sample D", dfLDLVrnc[4,1], dfLDLVrnc[4,2])
```

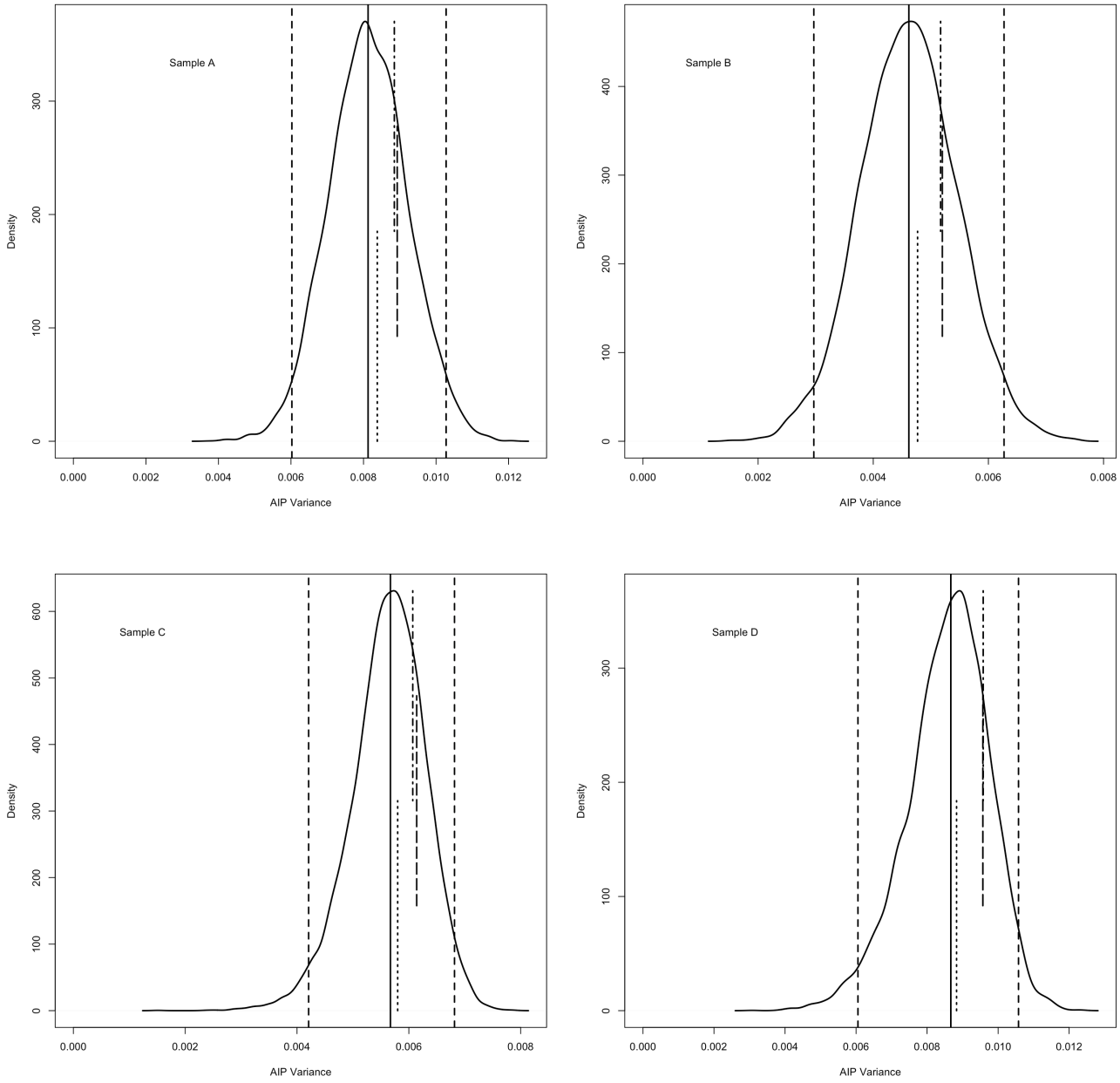


Distribution densitites of LDL variances calculated using 10 000 iterations of the bootstrapping. The median is shown by a solid vertical line and lower and upper 95 percentile range with dashed vertical lines. Segment from bottom to middle of graph: empirical variance. Segment from top to middle: Error propagation variance.

```
par(mfrow=c(1,1))
# dev.off()
```

Suppl.Fig. 12 AIP Distribution density

```
# setEPS()
# postscript("Figure2.eps")
par(mfrow=c(2,2))
  fPltAIPVrncDens(AIPSmplABootVrnc$dataTable, "Sample A", dfAIPVrnc[1,1], dfAIPVrnc[1,2], dfAIPVrnc[1,3])
  fPltAIPVrncDens(AIPSmplBBootVrnc$dataTable, "Sample B", dfAIPVrnc[2,1], dfAIPVrnc[2,2], dfAIPVrnc[2,3])
  fPltAIPVrncDens(AIPSmplCBootVrnc$dataTable, "Sample C", dfAIPVrnc[3,1], dfAIPVrnc[3,2], dfAIPVrnc[3,3])
  fPltAIPVrncDens(AIPSmplDBootVrnc$dataTable, "Sample D", dfAIPVrnc[4,1], dfAIPVrnc[4,2], dfAIPVrnc[4,3])
```



Distribution densitites of AIP variances calculated using 10 000 iterations of the bootstrapping. The median is shown by a solid vertical line and lower and upper 95 percentile range with dashed vertical lines. Segment from bottom to middle of graph: empirical variance. Segment in the middle:Error propagation variance. Segment from top to middle: 2nd Order Taylor Error propagation variance.

```
par(mfrow=c(1,1))
# dev.off()
```

Changing variances

CHOL Variance Changing

Calculate the error propagation and bootstrap variances using the 4 samples, using the corresponding distribution of CHOL with increasing CV, while keeping the respective distributions of HDL and TG constant. Note: If you rerun the analysis, the parts that involve the ramdom function may not give exactly the same results. The new results should however be close to the ones in this document.

Set no of repetitions for bootstrap

```
bootstrapNoOfReps = 2E3
```

CVs for CHOL for the 4 samples

```
unlist(lapply(lstDF, function(DF) {fCV(DF$CHOL) }))
```

```
## [1] 2.83 3.23 2.40 3.06
```

Create distributions of CHOL with increasing CV for the 4 samples.

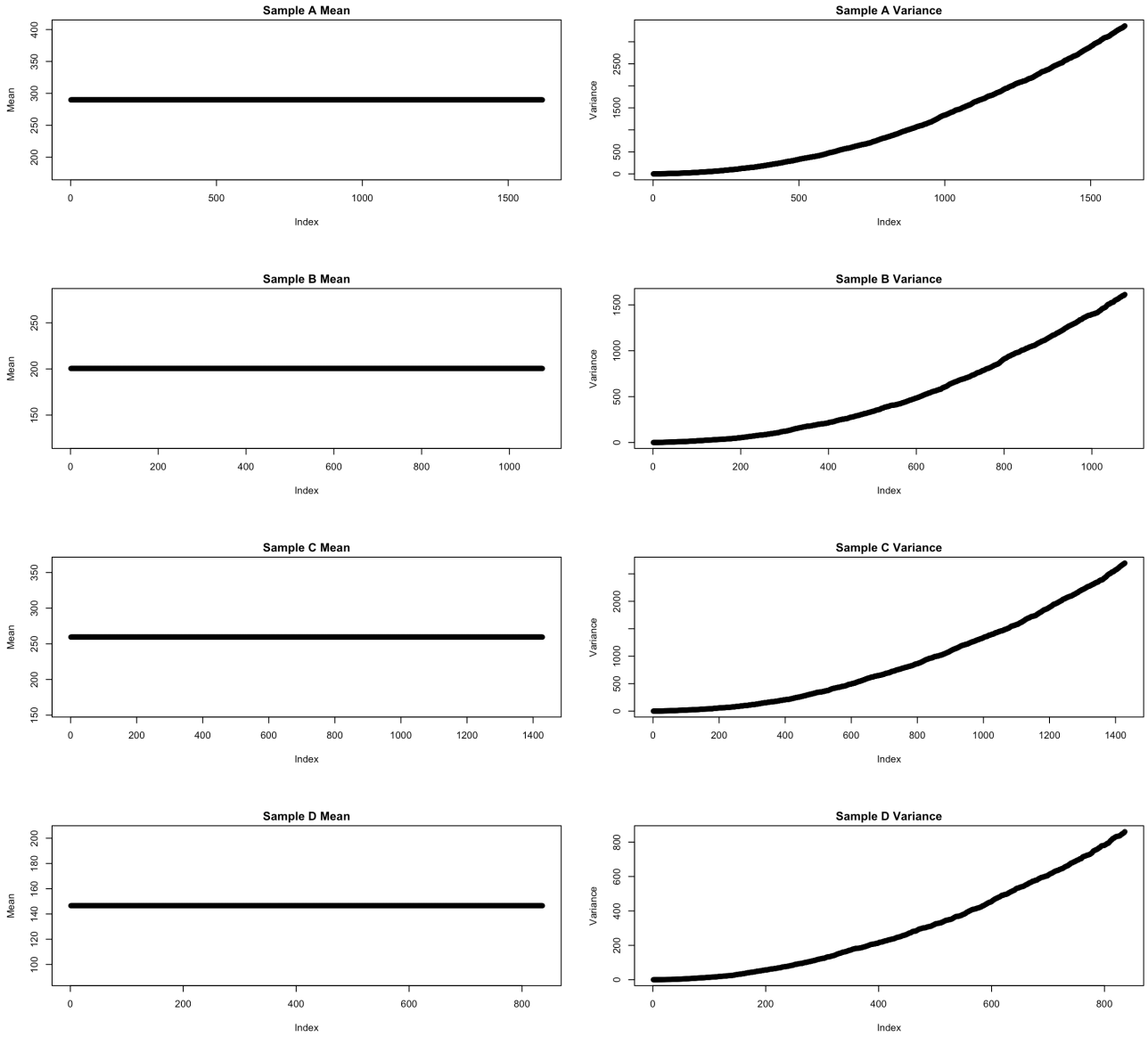
We will create a distribution of CHOL with CV from 0 to 20.

```
list(dfSmplACHOLChngCV, dfSmplBCHOLChngCV, dfSmplCCHOLChngCV, dfSmplDCHOLChngCV) %tin%
  lapply(lstDF, function(DF) {fChngDistrCVMeanConst(DF[,"CHOL"], lowerCV=0, upperCV=20, maxIter=10000, plot=
F)})
```

Suppl.Fig.13 Plot mean and variance

of the increasing CV CHOL distributions for the 4 samples, in order to check that the mean stays constant and the variance increases.

```
par(mfrow=c(4,2)); par(xpd=F)
lapply(1:4, function(i) {
  DF = list(dfSmplACHOLChngCV, dfSmplBCHOLChngCV, dfSmplCCHOLChngCV, dfSmplDCHOLChngCV)[[i]]
  plot(colMeans(DF), ylab="Mean"); title(paste("Sample", toupper(letters[i]), "Mean"), line=0.5)
  plot(apply(DF,2,var), ylab="Variance"); title(paste("Sample", toupper(letters[i]), "Variance"), line=0.5)
})
```



Means and variances for the CHOL distributions, in order to check that the mean stays constant and the variance increases.

```
par(mfrow=c(1,1))
```

HDL Variance changing

Calculate the error propagation and bootstrap variance using the 4 samples, using the corresponding distribution of HDL with increasing CV, while keeping the respective distributions of CHOL and TG constant. Note: If you rerun the analysis, the parts that involve the ramdom function may not give exactly the same results. The new results should however be close to the ones in this document.

CVs for HDL for the 4 samples

```
unlist(lapply(lstDF, function(DF) {fCV(DF$HDL) }))
```

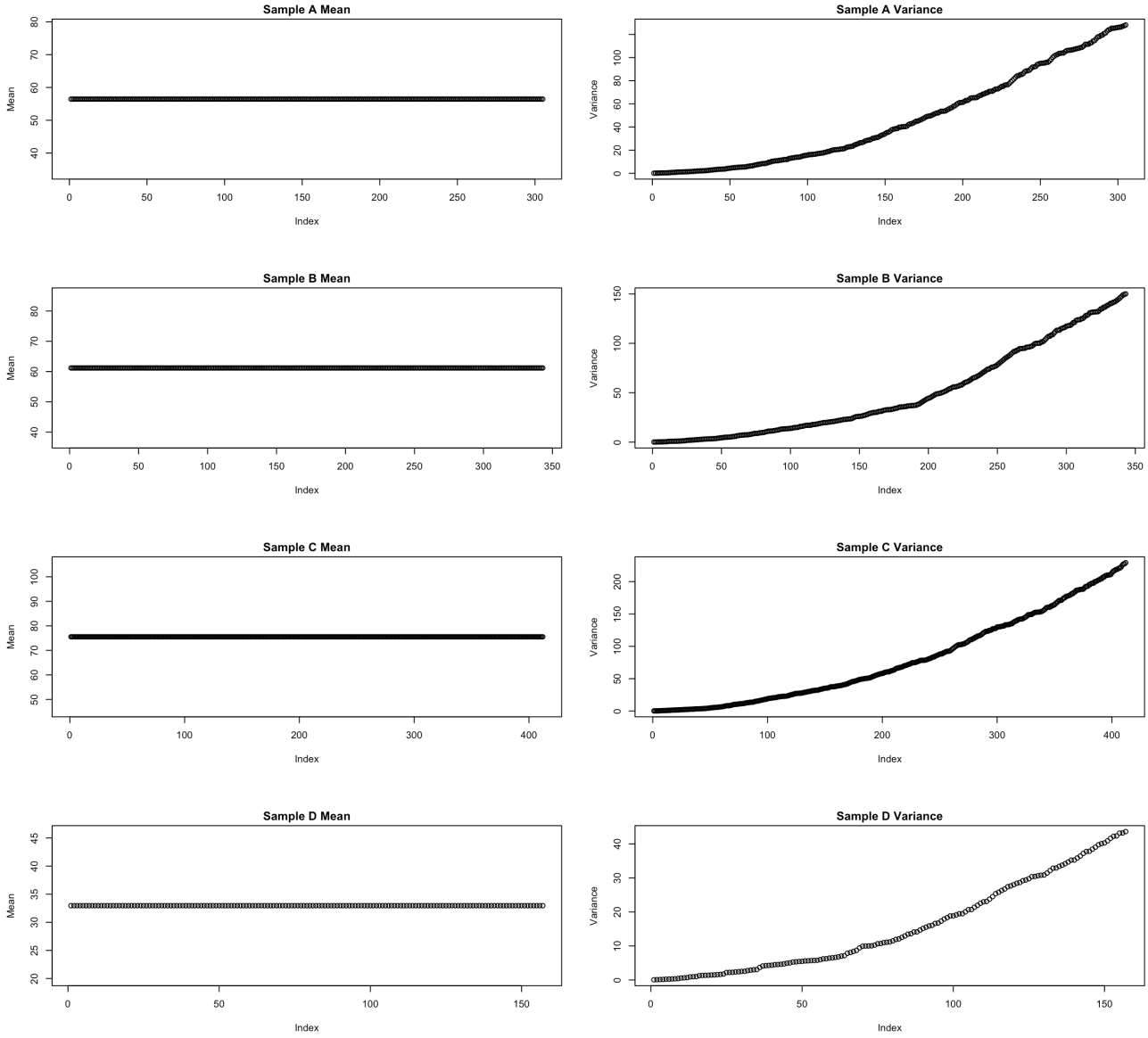
```
## [1] 14.52 15.66 14.55 19.93
```

Create distributions of HDL with increasing CV for the 4 samples.

```
list(dfSmplAHDLChngCV, dfSmplBHDLChngCV, dfSmplCHDLChngCV, dfSmplDHDLChngCV) %tin%
  lapply(lstDF, function(DF) {fChngDistrCVMeanConst(DF[,"HDL"], lowerCV=0, upperCV=20, maxIter=10000, plot=
F)})
```

Suppl.Fig.14 Plot mean and the variance of the increasing CV HDL distributions for the 4 samples, in order to check that the mean stays constant and the variance increases.

```
par(mfrow=c(4,2)); par(xpd=F)
lapply(1:4, function(i) {
  DF = list(dfSmplAHDLChngCV, dfSmplBHDLChngCV, dfSmplCHDLChngCV, dfSmplDHDLChngCV)[[i]]
  plot(colMeans(DF), ylab="Mean"); title(paste("Sample", toupper(letters[i]), "Mean"), line=0.5)
  plot(apply(DF,2,var), ylab="Variance"); title(paste("Sample", toupper(letters[i]), "Variance"), line=0.5)
})
```



Means and variances for the HDL distributions, in order to check that the mean stays constant and the variance increases.

```
par(mfrow=c(1,1))
```

TG Variance changing

Calculate the error propagation and bootstrap variance using the 4 samples, using the corresponding distribution of TG with increasing CV, while keeping the respective distributions of CHOL and HDL constant. Note: If you rerun the analysis, the parts that involve the ramdom function may not give exactly the same results. The new results should however be close to the ones in this document.

CVs for TG for the 4 samples

```
unlist(lapply(lstDF, function(DF) {fCV(DF$TG) }))
```

```
## [1]  9.76 10.11  5.13  3.41
```

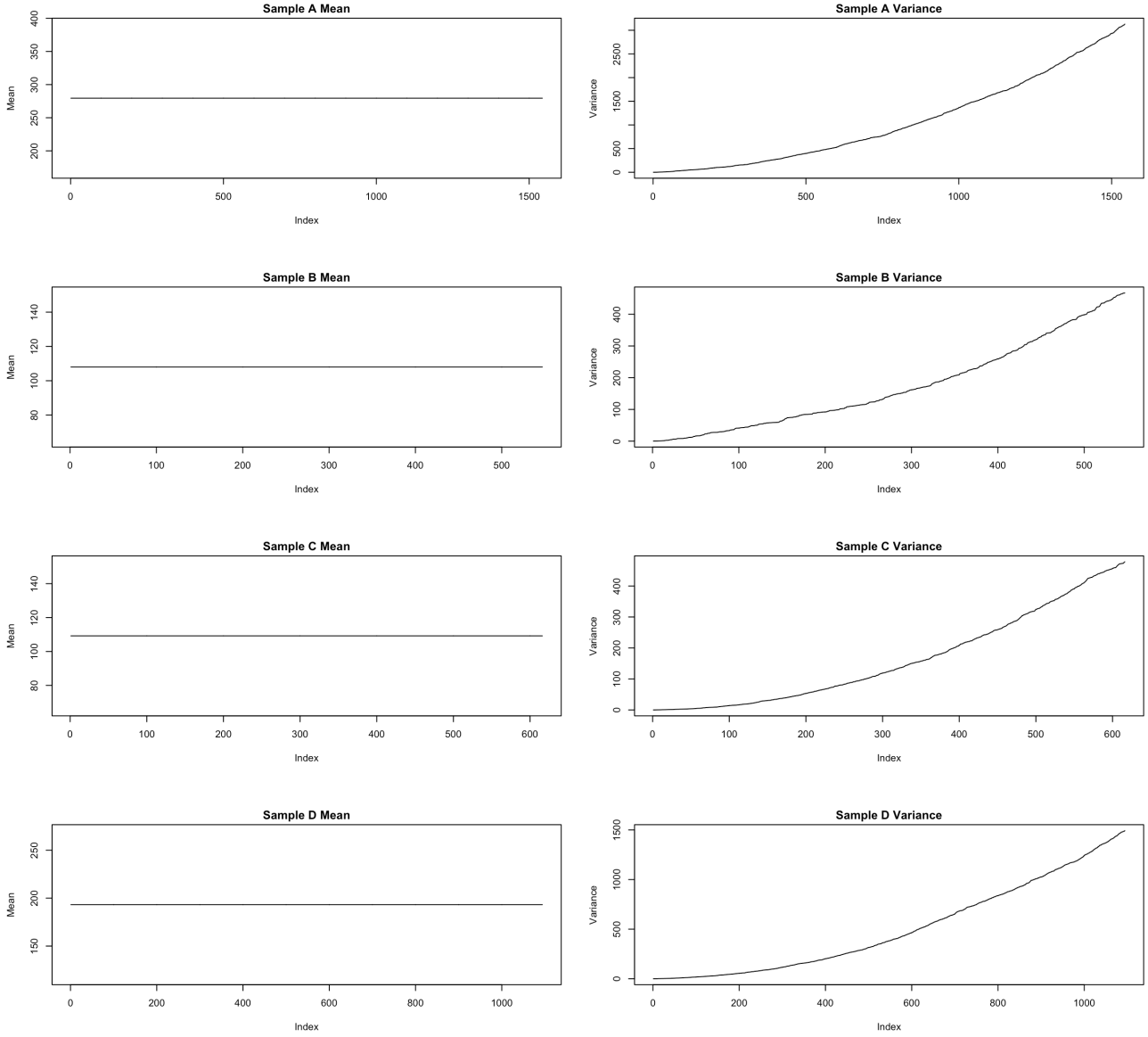
Create the distributions of TG with increasing CV for the 4 samples.

```
list(dfSmplATGChngCV, dfSmplBTGChngCV, dfSmplCTGChngCV, dfSmplDTGChngCV) %tin%
  lapply(lstDF, function(DF) {fChngDistrCVMeanConst(DF[,"TG"], lowerCV=0, upperCV=20, maxIter=100000, plot=
F)})
```

Suppl.Fig.15 Plot mean and variance

of the increasing CV HDL distributions for the 4 samples, in order to check that the mean stays constant and the variance increases.

```
par(mfrow=c(4,2)); par(xpd=F)
lapply(1:4, function(i) {
  DF = list(dfSmplATGChngCV, dfSmplBTGChngCV, dfSmplCTGChngCV, dfSmplDTGChngCV)[[i]]
  plot(colMeans(DF), ylab="Mean", type="l"); title(paste("Sample", toupper(letters[i]), "Mean"), line=0.5)
  plot(apply(DF,2,var), ylab="Variance", type="l"); title(paste("Sample", toupper(letters[i]), "Variance"), li
ne=0.5)
})
```



Means and variances for the TG distributions, in order to check that the mean stays constant and the variance increases.

```
par(mfrow=c(1,1))
```

LDL Variance

CHOL Variance changing

Calculate error propagation and bootstrap variance of LDL when the CHOL variance is increasing.

```
list(LDLSmplAVrncChngCHOLVrnc, LDLSmplBVrncChngCHOLVrnc, LDLSmplCVrncChngCHOLVrnc, LDLSmplDVrncChngCHOLVrnc) %
tin%
  lapply(1:4, function(i) {
    DF = list(dfSmplACHOLChngCV, dfSmplBCHOLChngCV, dfSmplCCHOLChngCV, dfSmplDCHOLChngCV)[[i]]
    dfHDL = lstDF[[i]][,"HDL"]
    dfTG = lstDF[[i]][,"TG"]
    fLDLCHOLVrnc(DF, dfHDL, dfTG, bootStrpNoOfReps = bootstrapNoOfReps)
  })
```

Assign error propagation variances to variables

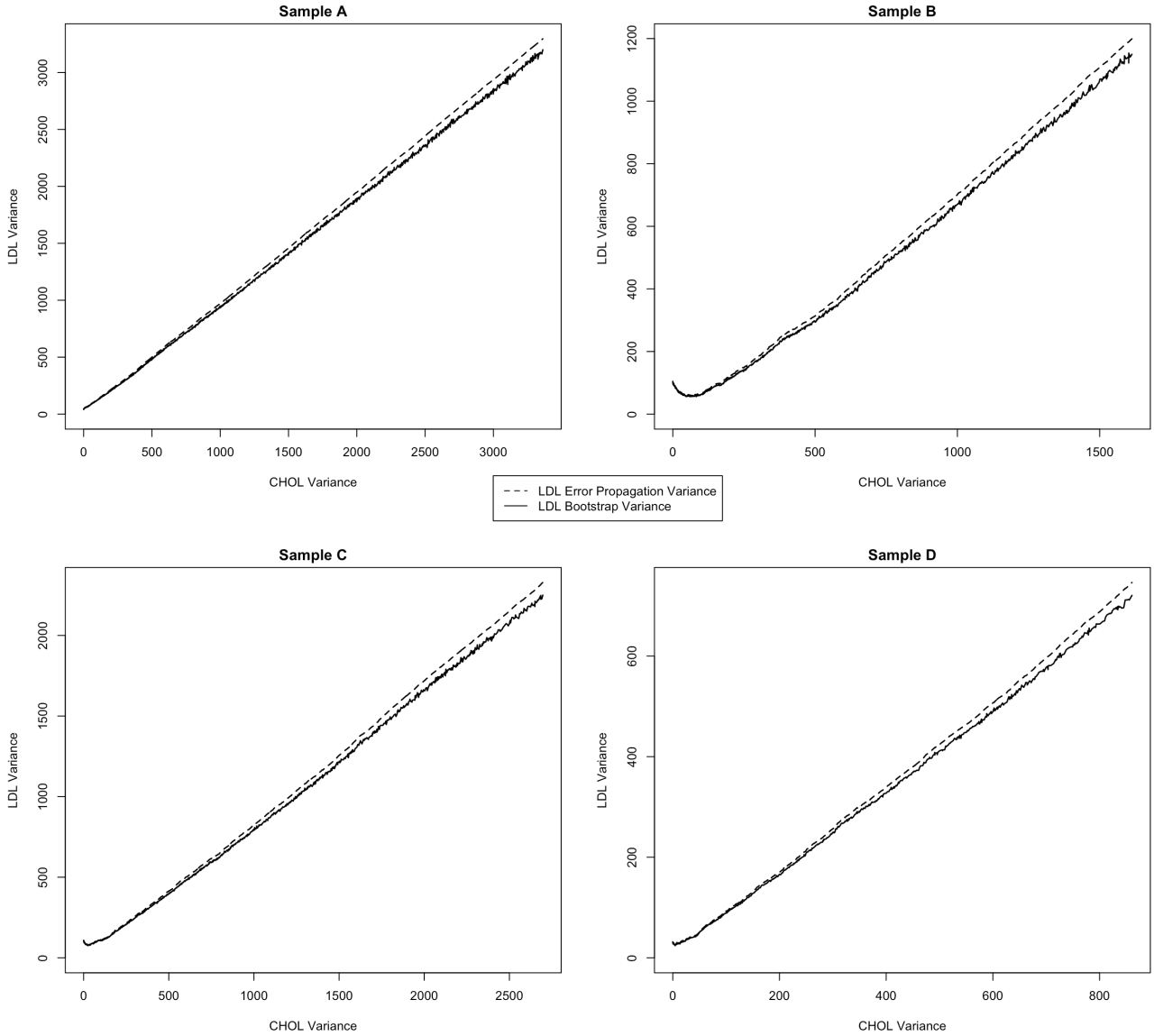
```
list(LDLSmplAVrncChngCHOLVrncErrProp, LDLSmplBVrncChngCHOLVrncErrProp, LDLSmplCVrncChngCHOLVrncErrProp, LDLSmp
lDVrncChngCHOLVrncErrProp) %tin%
  lapply(1:4, function(i){
    list(LDLSmplAVrncChngCHOLVrnc, LDLSmplBVrncChngCHOLVrnc, LDLSmplCVrncChngCHOLVrnc, LDLSmplDVrncChngCHOLVrn
c)[[i]]$ErrPropVrnc
  })
```

Assign bootstrap variances to variables

```
list(LDLSmplAVrncChngCHOLVrncBoot, LDLSmplBVrncChngCHOLVrncBoot, LDLSmplCVrncChngCHOLVrncBoot, LDLSmplDVrncChn
gCHOLVrncBoot) %tin%
  lapply(1:4, function(i){
    list(LDLSmplAVrncChngCHOLVrnc, LDLSmplBVrncChngCHOLVrnc, LDLSmplCVrncChngCHOLVrnc, LDLSmplDVrncChngCHOLVrn
c)[[i]]$BootVrnc
  })
```

Suppl.Fig.16 Plot CHOL variance for the 4 samples versus the variance of LDL

```
par(mfrow=c(2,2)); par(xpd=NA); # The last one sets plot clipping to the device.
lapply(1:4,
       function(i) {
         DFdistr = list(dfSmplACHOLChngCV, dfSmplBCHOLChngCV, dfSmplCCHOLChngCV, dfSmplDCHOLChngCV)[[i]]

         DFErrPropVrnc = list(LDLSmplAVrncChngCHOLVrncErrProp, LDLSmplBVrncChngCHOLVrncErrProp, LDLSmplCVrncCh
ngCHOLVrncErrProp, LDLSmplDVrncChngCHOLVrncErrProp)[[i]]

         DFBootVrnc = list(LDLSmplAVrncChngCHOLVrncBoot, LDLSmplBVrncChngCHOLVrncBoot, LDLSmplCVrncChngCHOLVrn
cBoot, LDLSmplDVrncChngCHOLVrncBoot)[[i]]

         plot(apply(DFdistr, 2, var), DFErrPropVrnc, type="l", lty=2, ylim=c(0, max(max(DFBootVrnc),max(DFErrP
ropVrnc)) ), lwd=1.5, xlab="CHOL Variance", ylab="LDL Variance")
         title(paste("Sample", toupper(letters[i])), line=0.5)
         lines(apply(DFdistr, 2, var), DFBootVrnc, type="l", lty=1, lwd=1.5)
       })
par(fig=c(0.25, 0.75, 0.25, 0.75), new=T)
legend("center",  lty=c(2,1), legend=c("LDL Error Propagation Variance", "LDL Bootstrap Variance"))
```



LDL variance with increasing CHOL variance and constant mean equal to the mean of each of the four samples

```
par(mfrow=c(1,1)); par(xpd=F)
```

HDL Variance changing

Calculate error propagation and bootstrap variance of LDL when the HDL variance is increasing.

```
list(LDLSmplAVrncChngHDLVrnc, LDLSmplBVrncChngHDLVrnc, LDLSmplCVrncChngHDLVrnc, LDLSmplDVrncChngHDLVrnc) %tin%
  lapply(1:4, function(i) {
    DF = list(dfSmplAHDLChngCV, dfSmplBHDLChngCV, dfSmplCHDLChngCV, dfSmplDHDLChngCV)[[i]]
    dfCHOL = lstDF[[i]][,"CHOL"]
    dfTG = lstDF[[i]][,"TG"]
    fLDLHDLVrnc(dfCHOL, DF, dfTG, bootStrpNoOfReps = bootstrapNoOfReps)
  })
```

Assign the error propagation variances to variables

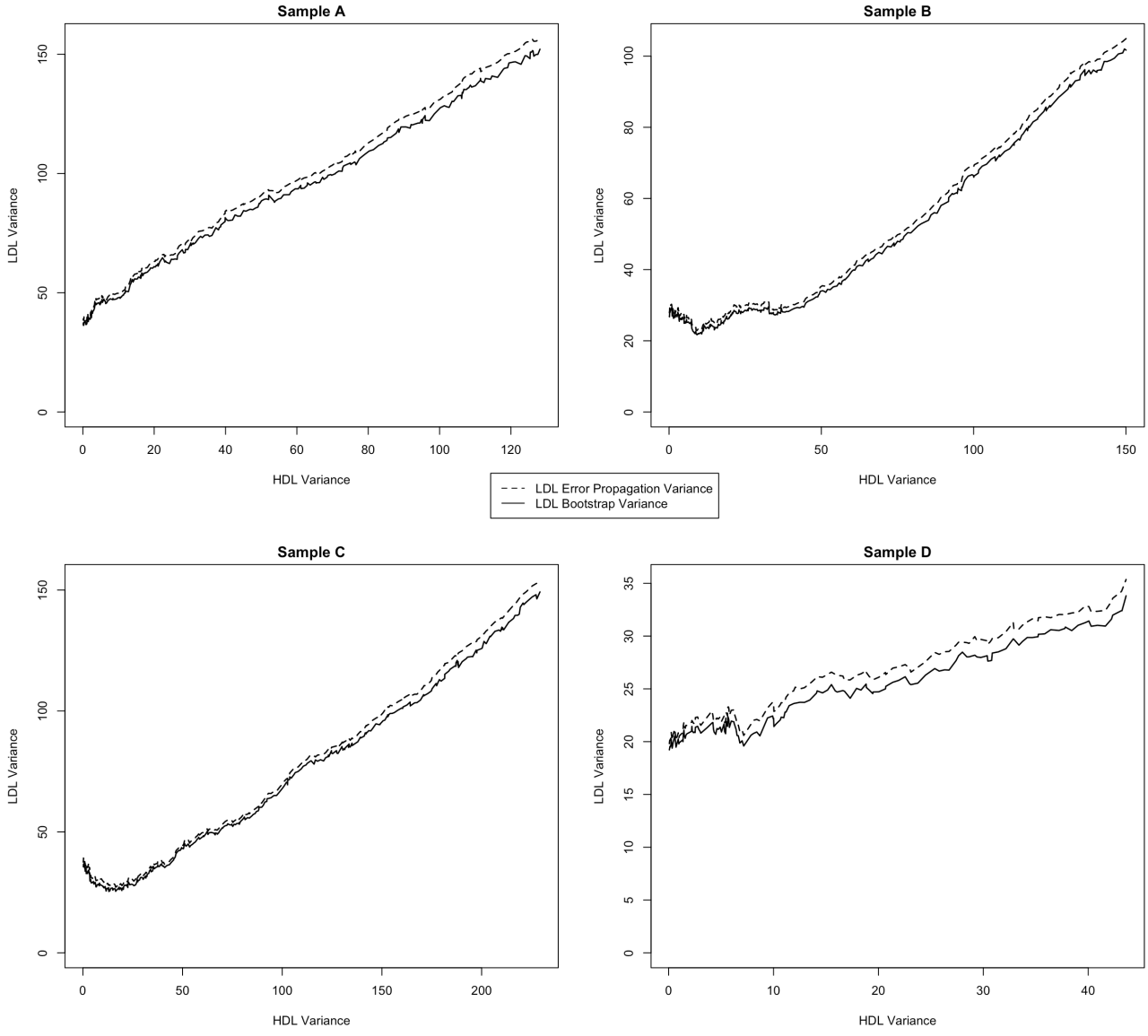
```
list(LDLSmplAVrncChngHDLVrncErrProp, LDLSmplBVrncChngHDLVrncErrProp, LDLSmplCVrncChngHDLVrncErrProp, LDLSmplDV
rncChngHDLVrncErrProp) %tin%
  lapply(1:4, function(i){
    list(LDLSmplAVrncChngHDLVrnc, LDLSmplBVrncChngHDLVrnc, LDLSmplCVrncChngHDLVrnc, LDLSmplDVrncChngHDLVrn
c)[[i]]$ErrPropVrnc
  })
```

Assign the bootstrap variances to variables

```
list(LDLSmplAVrncChngHDLVrncBoot, LDLSmplBVrncChngHDLVrncBoot, LDLSmplCVrncChngHDLVrncBoot, LDLSmplDVrncChngHD
LVrncBoot) %tin%
  lapply(1:4, function(i){
    list(LDLSmplAVrncChngHDLVrnc, LDLSmplBVrncChngHDLVrnc, LDLSmplCVrncChngHDLVrnc, LDLSmplDVrncChngHDLVrn
c)[[i]]$BootVrnc
  })
```

Suppl.Fig.17 Plot HDL variance HDL for the 4 samples versus the variance of LDL

```
par(mfrow=c(2,2)); par(xpd=NA); # The last one sets plot clipping to the device.
lapply(1:4,
       function(i) {
         DFdistr = list(dfSmplAHDLChngCV, dfSmplBHDLChngCV, dfSmplCHDLChngCV, dfSmplDHDLChngCV)[[i]]
         DFErrPropVrnc = list(LDLSmplAVrncChngHDLVrncErrProp, LDLSmplBVrncChngHDLVrncErrProp, LDLSmplCVrncChng
HDLVrncErrProp, LDLSmplDVrncChngHDLVrncErrProp)[[i]]
         DFBootVrnc = list(LDLSmplAVrncChngHDLVrncBoot, LDLSmplBVrncChngHDLVrncBoot, LDLSmplCVrncChngHDLVrncBo
ot, LDLSmplDVrncChngHDLVrncBoot)[[i]]
         plot(apply(DFdistr, 2, var), DFErrPropVrnc, type="l", lty=2, ylim=c(0, max(max(DFBootVrnc), max(DFErr
PropVrnc)) ), lwd=1.5, xlab="HDL Variance", ylab="LDL Variance")
         title(paste("Sample", toupper(letters[i])), line=0.5)
         lines(apply(DFdistr, 2, var), DFBootVrnc, type="l", lty=1, lwd=1.5)
       })
par(fig=c(0.25, 0.75, 0.25, 0.75), new=T)
legend("center",  lty=c(2,1), legend=c("LDL Error Propagation Variance", "LDL Bootstrap Variance"))
```



LDL variance with increasing HDL variance and constant mean equal to the mean of each of the four samples

```
par(mfrow=c(1,1)); par(xpd=F)
```

TG Variance changing

Calculate error propagation and bootstrap variance of LDL when the TG variance is increasing.

```
list(LDLSmplAVrncChngTGVrnc, LDLSmplBVrncChngTGVrnc, LDLSmplCVrncChngTGVrnc, LDLSmplDVrncChngTGVrnc) %tin%
  lapply(1:4, function(i) {
    DF = list(dfSmplATGChngCV, dfSmplBTGChngCV, dfSmplCTGChngCV, dfSmplDTGChngCV)[[i]]
    dfCHOL = lstDF[[i]][,"CHOL"]
    dfHDL = lstDF[[i]][,"HDL"]
    fLDLTGVrnc(dfCHOL, dfHDL, DF, bootStrpNoOfReps = bootstrapNoOfReps)
  })
```

Assign the error propagation variances to variables

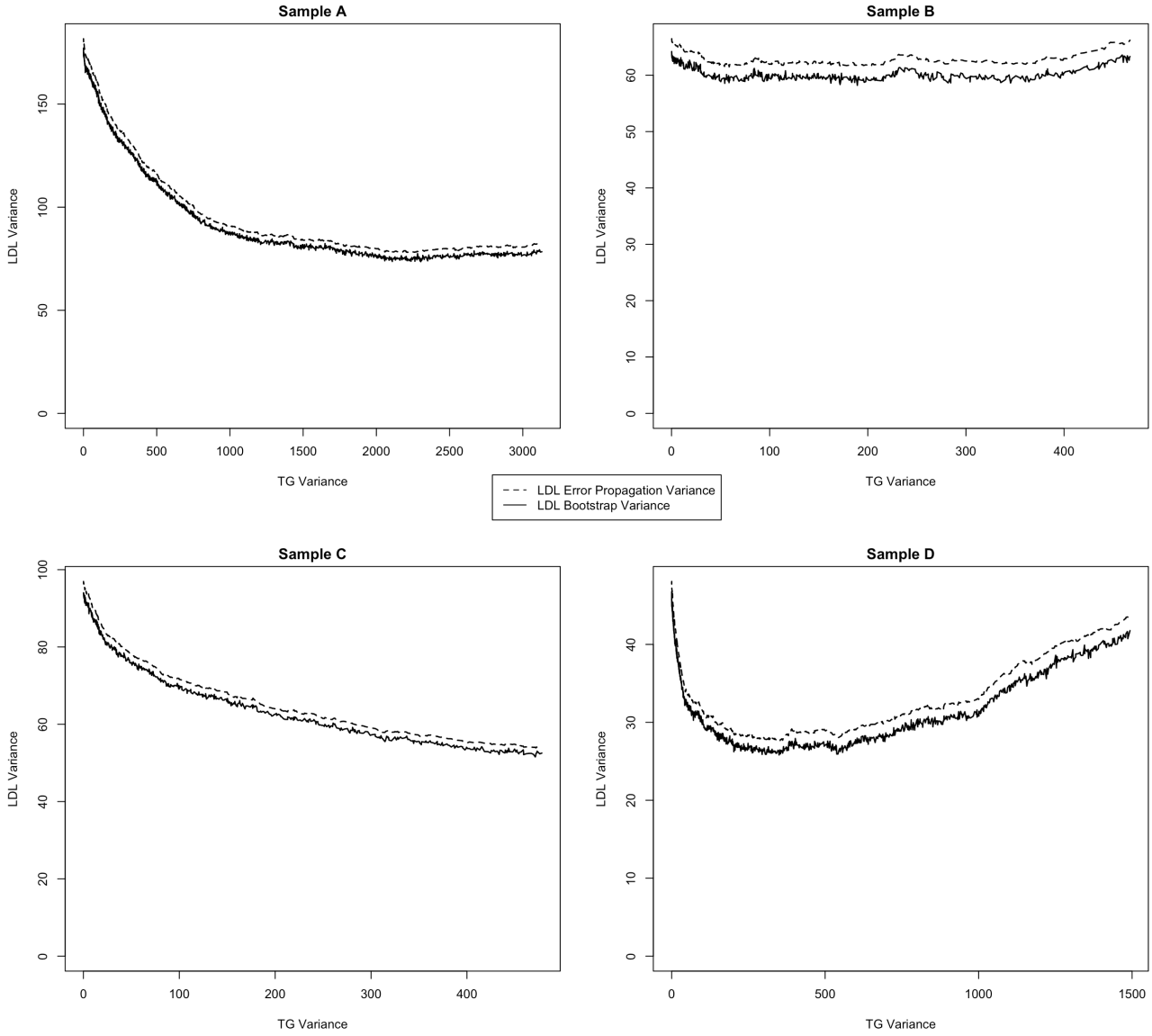
```
list(LDLSmplAVrncChngTGVrncErrProp, LDLSmplBVrncChngTGVrncErrProp, LDLSmplCVrncChngTGVrncErrProp, LDLSmplDVrnc
ChngTGVrncErrProp) %tin%
  lapply(1:4, function(i){
    list(LDLSmplAVrncChngTGVrnc, LDLSmplBVrncChngTGVrnc, LDLSmplCVrncChngTGVrnc, LDLSmplDVrncChngTGVrn
c)[[i]]$ErrPropVrnc
  })
```

Assign the bootstrap variances to variables

```
list(LDLSmplAVrncChngTGVrncBoot, LDLSmplBVrncChngTGVrncBoot, LDLSmplCVrncChngTGVrncBoot, LDLSmplDVrncChngTGVrn
cBoot) %tin%
  lapply(1:4, function(i){
    list(LDLSmplAVrncChngTGVrnc, LDLSmplBVrncChngTGVrnc, LDLSmplCVrncChngTGVrnc, LDLSmplDVrncChngTGVrn
c)[[i]]$BootVrnc
  })
```

Suppl.Fig.18 Plot TG variance for the 4 samples versus the variance of LDL

```
par(mfrow=c(2,2)); par(xpd=NA); # The last one sets plot clipping to the device.
lapply(1:4,
       function(i) {
         DFdistr = list(dfSmplATGChngCV, dfSmplBTGChngCV, dfSmplCTGChngCV, dfSmplDTGChngCV)[[i]]
         DFErrPropVrnc = list(LDLSmplAVrncChngTGVrncErrProp, LDLSmplBVrncChngTGVrncErrProp, LDLSmplCVrncChngTG
VrncErrProp, LDLSmplDVrncChngTGVrncErrProp)[[i]]
         DFBootVrnc = list(LDLSmplAVrncChngTGVrncBoot, LDLSmplBVrncChngTGVrncBoot, LDLSmplCVrncChngTGVrncBoot,
LDLSmplDVrncChngTGVrncBoot)[[i]]
         plot(apply(DFdistr, 2, var), DFErrPropVrnc, type="l", lty=2, ylim=c(0, max(max(DFBootVrnc), max(DFErr
PropVrnc)) ), lwd=1.5, xlab="TG Variance", ylab="LDL Variance")
         title(paste("Sample", toupper(letters[i])), line=0.5)
         lines(apply(DFdistr, 2, var), DFBootVrnc, type="l", lty=1, lwd=1.5)
       })
par(fig=c(0.25, 0.75, 0.25, 0.75), new=T)
legend("center",  lty=c(2,1), legend=c("LDL Error Propagation Variance", "LDL Bootstrap Variance"))
```



LDL variance with increasing TG variance and constant mean equal to the mean of each of the four samples

```
par(mfrow=c(1,1)); par(xpd=F)
```

AIP Variance

HDL Variance changing

Calculate error propagation and bootstrap variance of AIP when the HDL variance is increasing.

```
list(AIPSmplAVrncChngHDLVrnc, AIPSmplBVrncChngHDLVrnc, AIPSmplCVrncChngHDLVrnc, AIPSmplDVrncChngHDLVrnc) %tin%
  lapply(1:4, function(i) {
    DF = list(dfSmplAHDLChngCV, dfSmplBHDLChngCV, dfSmplCHDLChngCV, dfSmplDHDLChngCV)[[i]]
    dfTG = lstDF[[i]][,"TG"]
    fAIPHDLVrnc(dfTG, DF, bootStrpNoOfReps = bootstrapNoOfReps)
  })
```

Save the above results:

```
write.csv(AIPSmplAVrncChngHDLVrnc, "./dataFiles/AIPSmplAVrncChngHDLVrnc.csv", row.names = F)
write.csv(AIPSmplBVrncChngHDLVrnc, "./dataFiles/AIPSmplBVrncChngHDLVrnc.csv", row.names = F)
write.csv(AIPSmplCVrncChngHDLVrnc, "./dataFiles/AIPSmplCVrncChngHDLVrnc.csv", row.names = F)
write.csv(AIPSmplDVrncChngHDLVrnc, "./dataFiles/AIPSmplDVrncChngHDLVrnc.csv", row.names = F)
```

Read the results to variables

```
AIPSmplAVrncChngHDLVrnc <- read.table("./dataFiles/AIPSmplAVrncChngHDLVrnc.csv", header=T, sep=",")
AIPSmplBVrncChngHDLVrnc <- read.table("./dataFiles/AIPSmplBVrncChngHDLVrnc.csv", header=T, sep=",")
AIPSmplCVrncChngHDLVrnc <- read.table("./dataFiles/AIPSmplCVrncChngHDLVrnc.csv", header=T, sep=",")
AIPSmplDVrncChngHDLVrnc <- read.table("./dataFiles/AIPSmplDVrncChngHDLVrnc.csv", header=T, sep=",")
```

Assign the error propagation variances to variables

```
list(AIPSmplAVrncChngHDLVrncErrProp, AIPSmplBVrncChngHDLVrncErrProp,
     AIPSmplCVrncChngHDLVrncErrProp, AIPSmplDVrncChngHDLVrncErrProp) %tin%
  lapply(1:4, function(i){
    list(AIPSmplAVrncChngHDLVrnc, AIPSmplBVrncChngHDLVrnc, AIPSmplCVrncChngHDLVrnc, AIPSmplDVrncChngHDLVrn
c)[[i]]$ErrPropVrnc
  })
```

Assign the 2nd order Taylor error propagation variances to variables

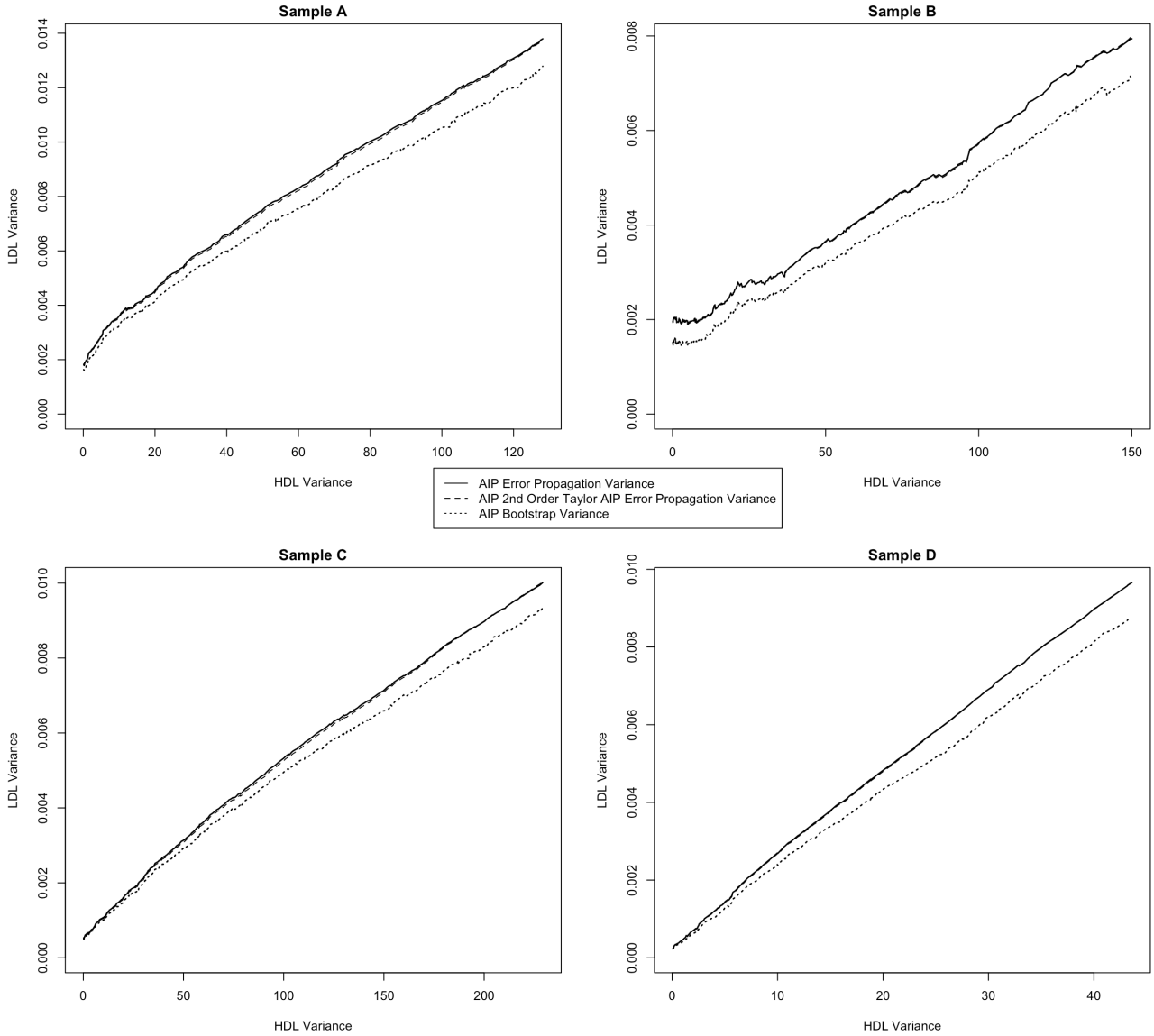
```
list(AIPSmplAVrncChngHDLVrncErrProp2Ord, AIPSmplBVrncChngHDLVrncErrProp2Ord,
     AIPSmplCVrncChngHDLVrncErrProp2Ord, AIPSmplDVrncChngHDLVrncErrProp2Ord) %tin%
  lapply(1:4, function(i){
    list(AIPSmplAVrncChngHDLVrnc, AIPSmplBVrncChngHDLVrnc,
         AIPSmplCVrncChngHDLVrnc, AIPSmplDVrncChngHDLVrnc)[[i]]$ErrPropVrnc2Ord
  })
```

Assign the bootstrap variances to variables

```
list(AIPSmplAVrncChngHDLVrncBoot, AIPSmplBVrncChngHDLVrncBoot,
     AIPSmplCVrncChngHDLVrncBoot, AIPSmplDVrncChngHDLVrncBoot) %tin%
  lapply(1:4, function(i){
    list(AIPSmplAVrncChngHDLVrnc, AIPSmplBVrncChngHDLVrnc,
         AIPSmplCVrncChngHDLVrnc, AIPSmplDVrncChngHDLVrnc)[[i]]$BootVrnc
  })
```

Suppl.Fig.19 Plot HDL variance HDL for the 4 samples versus the variance of AIP

```
par(mfrow=c(2,2)); par(xpd=NA); # The last one sets plot clipping to the device.
lapply(1:4,
   function(i) {
     DFdistr = list(dfSmplAHDLChngCV, dfSmplBHDLChngCV, dfSmplCHDLChngCV, dfSmplDHDLChngCV)[[i]]
     DFErrPropVrnc = list(AIPSmplAVrncChngHDLVrncErrProp, AIPSmplBVrncChngHDLVrncErrProp,
                          AIPSmplCVrncChngHDLVrncErrProp, AIPSmplDVrncChngHDLVrncErrProp)[[i]]
     DFErrPropVrnc2Ord = list(AIPSmplAVrncChngHDLVrncErrProp2Ord, AIPSmplBVrncChngHDLVrncErrProp2Ord,
                          AIPSmplCVrncChngHDLVrncErrProp2Ord, AIPSmplDVrncChngHDLVrncErrProp2Ord)[[i]]
     DFBootVrnc = list(AIPSmplAVrncChngHDLVrncBoot, AIPSmplBVrncChngHDLVrncBoot,
                       AIPSmplCVrncChngHDLVrncBoot, AIPSmplDVrncChngHDLVrncBoot)[[i]]
     plot(apply(DFdistr, 2, var), DFErrPropVrnc, type="l", lty=1,
          ylim=c(0, max(max(DFBootVrnc), max(DFErrPropVrnc)) ),
          lwd=1.5, xlab="HDL Variance", ylab="LDL Variance")
     title(paste("Sample", toupper(letters[i])), line=0.5)
     lines(apply(DFdistr, 2, var), DFErrPropVrnc2Ord, type="l", lty=2)
     lines(apply(DFdistr, 2, var), DFBootVrnc, type="l", lty=3, lwd=1.5)


   })
par(fig=c(0.25, 0.75, 0.25, 0.75), new=T)
legend("center",  lty=c(1,2,3), legend=c("AIP Error Propagation Variance",
                                         "AIP 2nd Order Taylor AIP Error Propagation Variance",
                                         "AIP Bootstrap Variance"))
```



AIP variance with increasing HDL variance and constant mean equal to the mean of each of the four samples

```
par(mfrow=c(1,1)); par(xpd=F)
```

TG Variance changing

Calculate error propagation and bootstrap variance of AIP when the TG variance is increasing.

```
list(AIPSmplAVrncChngTGVrnc, AIPSmplBVrncChngTGVrnc, AIPSmplCVrncChngTGVrnc, AIPSmplDVrncChngTGVrnc) %tin%
  lapply(1:4, function(i) {
    DF = list(dfSmplATGChngCV, dfSmplBTGChngCV, dfSmplCTGChngCV, dfSmplDTGChngCV)[[i]]
    dfHDL = lstDF[[i]][,"HDL"]
    fAIPTGVrnc(DF, dfHDL, bootStrpNoOfReps = bootstrapNoOfReps)
  })
```

Save the above results:

```
write.csv(AIPSmplAVrncChngTGVrnc, "./dataFiles/AIPSmplAVrncChngTGVrnc.csv", row.names = F)
write.csv(AIPSmplBVrncChngTGVrnc, "./dataFiles/AIPSmplBVrncChngTGVrnc.csv", row.names = F)
write.csv(AIPSmplCVrncChngTGVrnc, "./dataFiles/AIPSmplCVrncChngTGVrnc.csv", row.names = F)
write.csv(AIPSmplDVrncChngTGVrnc, "./dataFiles/AIPSmplDVrncChngTGVrnc.csv", row.names = F)
```

Read the results to variables

```
AIPSmplAVrncChngTGVrnc <- read.table("./dataFiles/AIPSmplAVrncChngTGVrnc.csv", header=T, sep=",")
AIPSmplBVrncChngTGVrnc <- read.table("./dataFiles/AIPSmplBVrncChngTGVrnc.csv", header=T, sep=",")
AIPSmplCVrncChngTGVrnc <- read.table("./dataFiles/AIPSmplCVrncChngTGVrnc.csv", header=T, sep=",")
AIPSmplDVrncChngTGVrnc <- read.table("./dataFiles/AIPSmplDVrncChngTGVrnc.csv", header=T, sep=",")
```

Assign the error propagation variances to variables

```
list(AIPSmplAVrncChngTGVrncErrProp, AIPSmplBVrncChngTGVrncErrProp,
     AIPSmplCVrncChngTGVrncErrProp, AIPSmplDVrncChngTGVrncErrProp) %tin%
  lapply(1:4, function(i){
    list(AIPSmplAVrncChngTGVrnc, AIPSmplBVrncChngTGVrnc,
         AIPSmplCVrncChngTGVrnc, AIPSmplDVrncChngTGVrnc)[[i]]$ErrPropVrnc
  })
```

Assign the 2nd order Taylor error propagation variances to variables

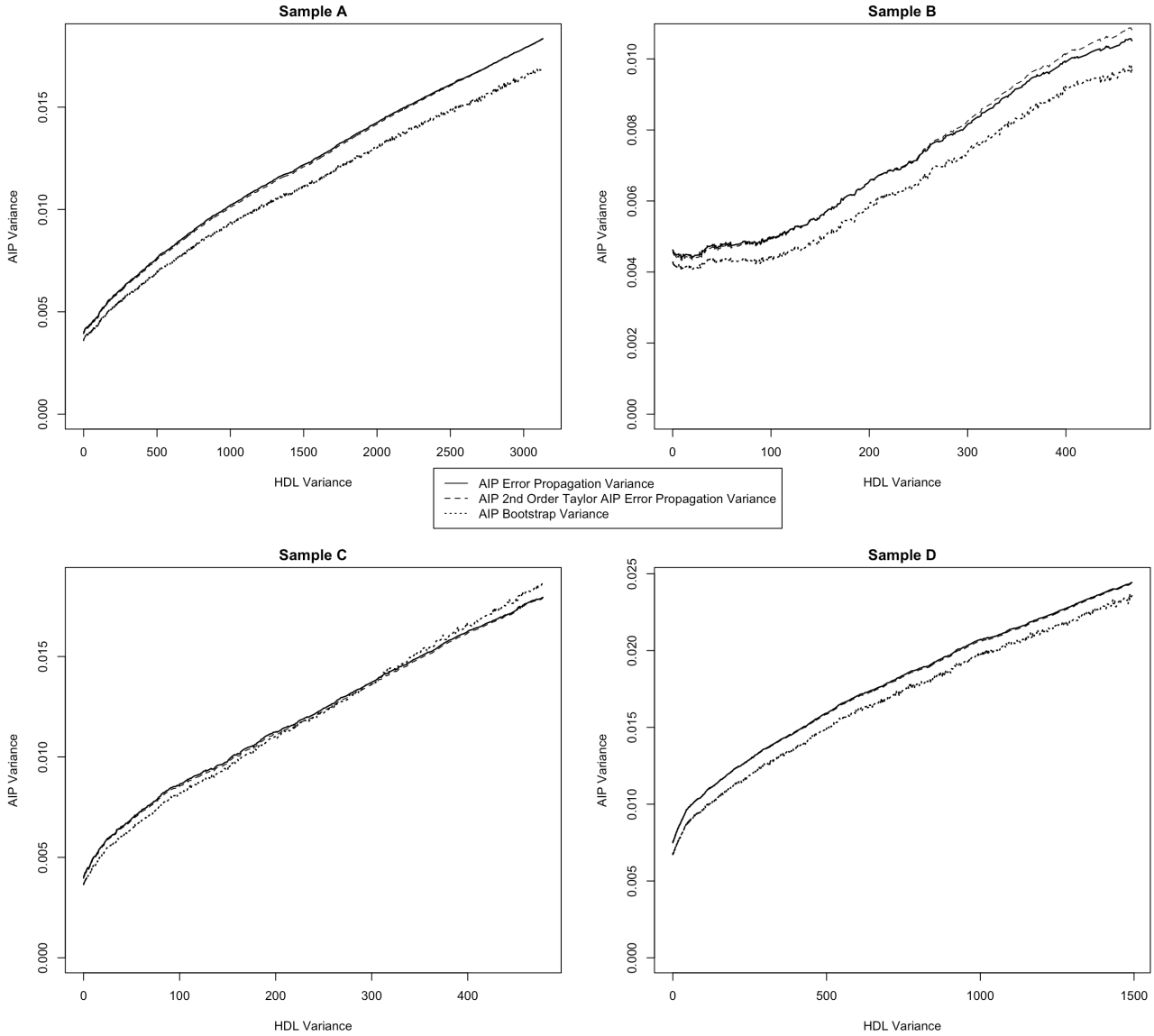
```
list(AIPSmplAVrncChngTGVrncErrProp2Ord, AIPSmplBVrncChngTGVrncErrProp2Ord,
     AIPSmplCVrncChngTGVrncErrProp2Ord, AIPSmplDVrncChngTGVrncErrProp2Ord) %tin%
  lapply(1:4, function(i){
    list(AIPSmplAVrncChngTGVrnc, AIPSmplBVrncChngTGVrnc,
         AIPSmplCVrncChngTGVrnc, AIPSmplDVrncChngTGVrnc)[[i]]$ErrPropVrnc2Ord
  })
```

Assign the bootstrap variances to variables

```
list(AIPSmplAVrncChngTGVrncBoot, AIPSmplBVrncChngTGVrncBoot,
     AIPSmplCVrncChngTGVrncBoot, AIPSmplDVrncChngTGVrncBoot) %tin%
  lapply(1:4, function(i){
    list(AIPSmplAVrncChngTGVrnc, AIPSmplBVrncChngTGVrnc,
         AIPSmplCVrncChngTGVrnc, AIPSmplDVrncChngTGVrnc)[[i]]$BootVrnc
  })
```

Suppl.Fig.20 Plot HDL variance for the 4 samples versus the variance of AIP

```
par(mfrow=c(2,2)); par(xpd=NA); # The last one sets plot clipping to the device.
lapply(1:4,
   function(i) {
     DFdistr = list(dfSmplATGChngCV, dfSmplBTGChngCV, dfSmplCTGChngCV, dfSmplDTGChngCV)[[i]]
     DFErrPropVrnc = list(AIPSmplAVrncChngTGVrncErrProp, AIPSmplBVrncChngTGVrncErrProp,
                          AIPSmplCVrncChngTGVrncErrProp, AIPSmplDVrncChngTGVrncErrProp)[[i]]
     DFErrPropVrnc2Ord = list(AIPSmplAVrncChngTGVrncErrProp2Ord, AIPSmplBVrncChngTGVrncErrProp2Ord,
                          AIPSmplCVrncChngTGVrncErrProp2Ord, AIPSmplDVrncChngTGVrncErrProp2Ord)[[i]]
     DFBootVrnc = list(AIPSmplAVrncChngTGVrncBoot, AIPSmplBVrncChngTGVrncBoot,
                       AIPSmplCVrncChngTGVrncBoot, AIPSmplDVrncChngTGVrncBoot)[[i]]
     plot(apply(DFdistr, 2, var), DFErrPropVrnc, type="l", lty=1,
          ylim=c(0, max(max(DFBootVrnc), max(DFErrPropVrnc)) ),
          lwd=1.5, xlab="HDL Variance", ylab="AIP Variance")
     title(paste("Sample", toupper(letters[i])), line=0.5)
     lines(apply(DFdistr, 2, var), DFErrPropVrnc2Ord, type="l", lty=2)
     lines(apply(DFdistr, 2, var), DFBootVrnc, type="l", lty=3, lwd=1.5)


   })
par(fig=c(0.25, 0.75, 0.25, 0.75), new=T)
legend("center",  lty=c(1,2,3), legend=c("AIP Error Propagation Variance",
                                         "AIP 2nd Order Taylor AIP Error Propagation Variance",
                                         "AIP Bootstrap Variance"))
```

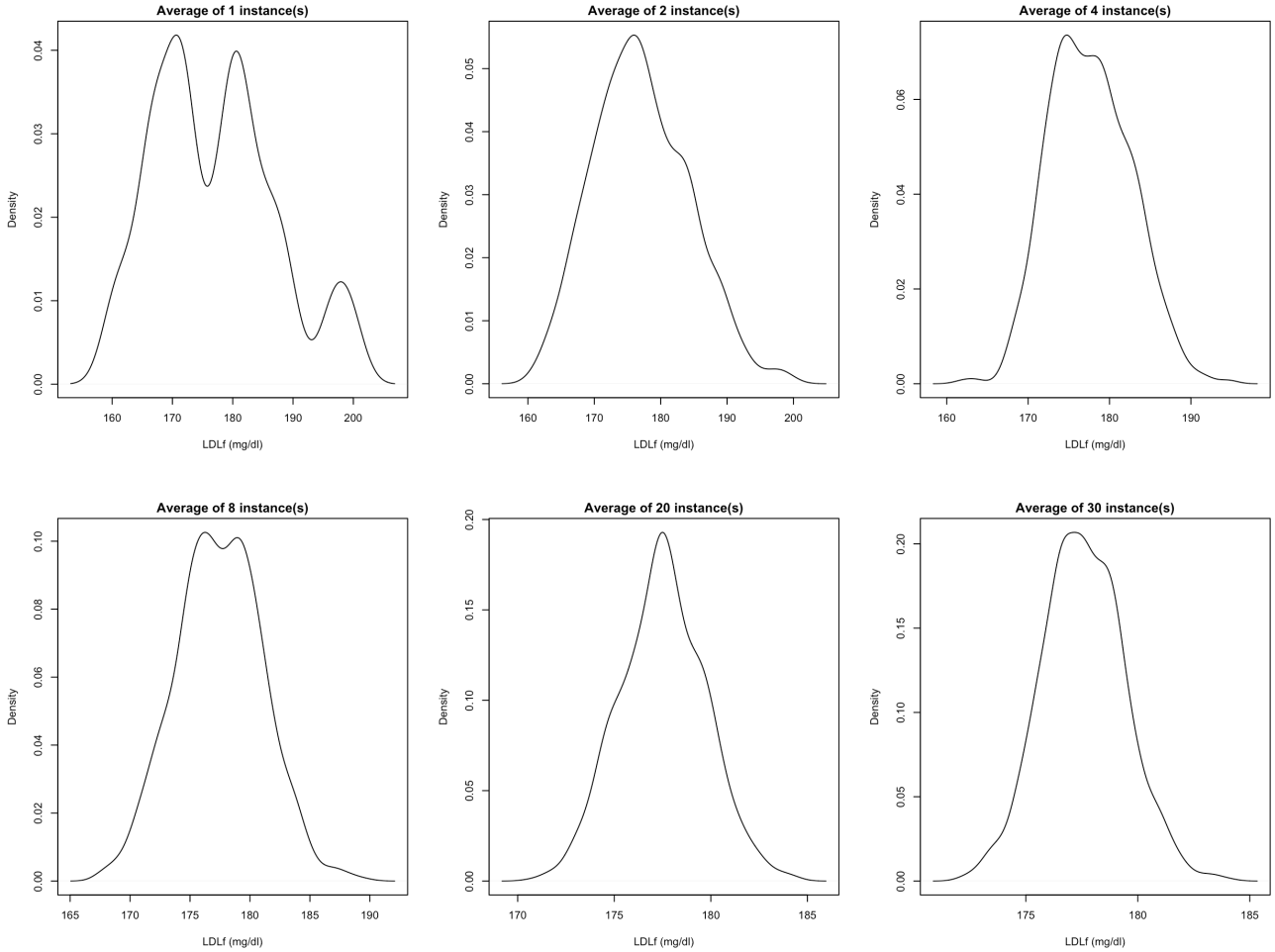


AIP variance with increasing TG variance and constant mean equal to the mean of each of the four samples

```
par(mfrow=c(1,1)); par(xpd=F)
```

Suppl.Fig.21 Demonstrate Central Limit Theorem by averaging increasing number of values from sample A LDL

```
par(mfrow=c(2,3))
lapply(list(1,2,4,8,20,30), function(x) {
  plot(density(fAverageData(smplA$LDL, sampleSize = x, noOfReps = 1000)), main="",xlab="LDLf (mg/dl)")
  title(paste("Average of", x, "instance(s)"), line=0.5)
})
```



Demostration of the Central Limit Theorem with increasing number of instances of a ramdon variable

```
par(mfrow=c(1,1))
```