Variance calculation of LDL and AIP

Set working directory and load required libraries

```
setwd(".")
require(LDLcalc)
require(ggplot2)
#> Loading required package: ggplot2
#> Warning: package 'ggplot2' was built under R version 4.3.2
require(gridExtra)
#> Loading required package: gridExtra
require(data.table)
#> Loading required package: data.table
require(tidyr)
#> Loading required package: tidyr
```

Determination of the uncertainty in measurement of Low Density Lipoprotein (LDL) and Atherogenic Index of Plasma (AIP).

1. Introduction[1]

A measurement is the process of determining the concentration of a biologically significant molecule in a biological fluid. This definiton applies both generally and specifically in clinical chemistry. The aim of all measurements is to obtain the true value. However, the result of the measurement will be only an estimate of the "true" value which remains unknown. We cannot know how near our measured value is to the "true" one.

This difference between the true and the measured value is the error. It can be random or systematic and its true value is also unknown to us. When the systematic error (bias) has been accounted for, the remaining random error component is characterized by the measurement uncertainty or simply uncertainty, according to the *Guide to the expression of uncertainty in measurement* (hereafter referred to as GUM) [1]. According to the *International vocabulary of metrology – Basic and general concepts and associated terms* (referred to as VIM) [2], uncertainty is defined as "a non-negative parameter characterising the dispersion of the quantity values being attributed to a measurand, based on the information used" (paragraph 2.26). It defines an interval around the measured value, into which the true value lies with some probability.

A measure of the uncertainty is the variance. The variance is the expectation (or mean) of the squared deviations (residuals) of a random variable from its mean. In more statistical terms it is the second central moment (the expectation or mean being the first). However, when talking about aggregate measurement uncertainty, other sources of uncertainty are often included. Many times, the standard deviation (the square root of the variance) is reported instead, since it has the same units as the measured quantity.

Although the uncertainty characterizes the error, it is not a difference between two values, does not have a sign (hence no direction) -unlike the error which does have a sign- and cannot be used to correct the result

[1] This vignette draws heavily from the paper "Determination of the Variance of Complex Calculated Clinical Chemistry Tests; Application in Calculated Low Density Lipoprotein and Atherogenic Index of Plasma", *Journal of Clinical and Diagnostic Research*, 2020 Jul, Vol-14(7): BC11-BC16. It shows in more detail how its results can be obtained using the `LDLcalc` package.

of the measurement. Although both the true value and the error are abstract concepts whose exact values cannot be determined, they are nevertheless useful. Their estimates can be determined and are useful.

1.2 Uncertainty in clinical chemistry measurements

All measurements, including those in clinical chemistry, have an uncertainty. Most of the clinical chemistry parameters are determined directly through measurement of some physical property, such as the change in the optical absorbance or fluoresence intensity of a reaction mixture.

However, there are some parameters that are determined as a result of some calculation using other experimentally determined parameters as [3] describe. A common one is the calculation of Low Density Lipoprotein Cholesterol (LDL) using the Friedewald formula [4], from total cholesterol (CHOL), high density lipoprotein cholesterol (HDL) and triglycerides (TG):

$$LDL = CHOL - HDL - \frac{TG}{5} \tag{1}$$

Another parameter calculated from lipid values is the atherogenic index of plasma (AIP), which has been reported to be a good biomarker associated with cardiovascular risk and is calculated as:

$$AIP = log_{10} \frac{TG\ (mmol/lt)}{HDL\ (mmol/lt)} \tag{2}$$

Both LDL and AIP are calculated using other variables whose values have been experimentally determined and where each one has its own measurement uncertainty. In these cases, we want to calculate the variance of these random variables, which are themselves a function of other random variables. This is called error propagation or propagation of uncertainty [5].

The package `LDLcalc` uses various methods to calculate the mean and variance of LDL and AIP. In this vignette each of the methods will be described, presented and the results will be compared, using functions contained in this package.

2. Sample data contained in the LDLcalc package.

Included in the LDLcalc package are two sample data files with repeated measurements of cholesterol, LDL and triglycerides. These measurements are from two different samples with different values of these analytes, measured in replicate for 34 consecutive days. For the rest of the analysis, we will use sample A, which we will load into a data frame (dfSmplA.

```
dfSmplA <- sampleA
```

The descriptions of columns of the two samples are as follows:

- **day**: Day number of each measurement.
- **CHOLrep1, CHOLrep2, CHOL**: Replicate measurements of cholesterol for each day and their mean.
- **HDLrep1, HDLrep2, HDL**: Replicate measurements of HDL for each day and their mean.
- **TGrep1, TGrep2, TG**: Replicate measurements of triglycerides for each day and their mean.
- **LDLrep1, LDLrep2, LDL**: LDL calculated according to the Friedewald formula mentioned above and its mean. All measurement units are in mg/dl and not mmol/L.

3. Calculation of AIP

Before proceeding, we will use the function `LDLcalc::AIPcalc` to calculate the atherogenic index of plasma and append it to `dfSmplA`. Since the AIP is defined in mmol/L units and the TG and HDL units are mg/dl, the SI parameter must be set to FALSE.

```
dfSmplA$AIPrep1 <- AIPcalc(TG=dfSmplA$TGrep1,
                                    HDL=dfSmplA$HDLrep1, SI=F)
dfSmplA$AIPrep2 <- AIPcalc(TG=dfSmplA$TGrep2,
                                    HDL=dfSmplA$HDLrep2, SI=F)
dfSmplA$AIP <- AIPcalc(TG=dfSmplA$TG,
                                  HDL=dfSmplA$HDL, SI=F)
```

4. Examination of the data

Before proceeding with variance determination, we will see that the data does not follow some commonly held assumptions.

4.1 Normality of the data

A common assumption made is that repeated measurements will give an (approximately) normal distribution. Moreover, an assumption made in error propagation (which we will use later) is that the distribution of the output quantity is normal or t-distributed. However, this is not always the case. We will create the discrete histograms of our experimentally measured quantities (CHOL, HDL and TG), to check if they are (at least nearly) normal.

Note:The values given in the data are discrete. We can treat each clinical chemistry measurement as a random variable that follows a distribution. Although most clinical chemistry measurements have continuous values, since in practice they are determined and reported up to a small number of significant digits we can treat their distributions as discrete, which is what we will do in this vignette. It is of no practical interest if a glucose value is 90.1 or 90.2 or if a calcium value is 9.44 or 9.45.

We will use the function `LDLcalc::PlotDiscrHist`. This function plots a discrete histogram (essentially a barplot) of the data frame column named in the `param` argument. It also plots the mean as a vertical continuous line, the mean plus/minus 2 standard deviations as vertical dotted lines and overlays a density plot of the normal distribution with mean and standard deviation corresponding to those of the data.

```
# Create a list of the parameters to be plotted.
lstParam <- list("CHOL", "HDL", "TG")
# Use lapply to apply the PlotDiscrHist function to all parameters.
lstPlt <- lapply(lstParam, LDLcalc::PlotDiscrHist, DF=dfSmplA)
do.call("grid.arrange", c(lstPlt, ncol = 3))
```

For all input quantities CHOL, HDL and TG it is evident from Figure 1 that they are not normally distributed, even without a formal test for normality (e.g. Kolmogorov-Smirnov).

4.2 Jensen-Shannon divergence between the empirical and corresponding normal distribution

Since we often use the normal distribution for repeated measurement data although this is not always correct, it would be useful to have a measure to quantify how much the empirical data diverges from the normal distribution. One such measure is the Jensen-Shannon distance metric D_{JS}^{m}, which is the square root of the Jensen-Shannon divergence [6]. It is based on the Kullback-Leibler divergence [7] between two distributions

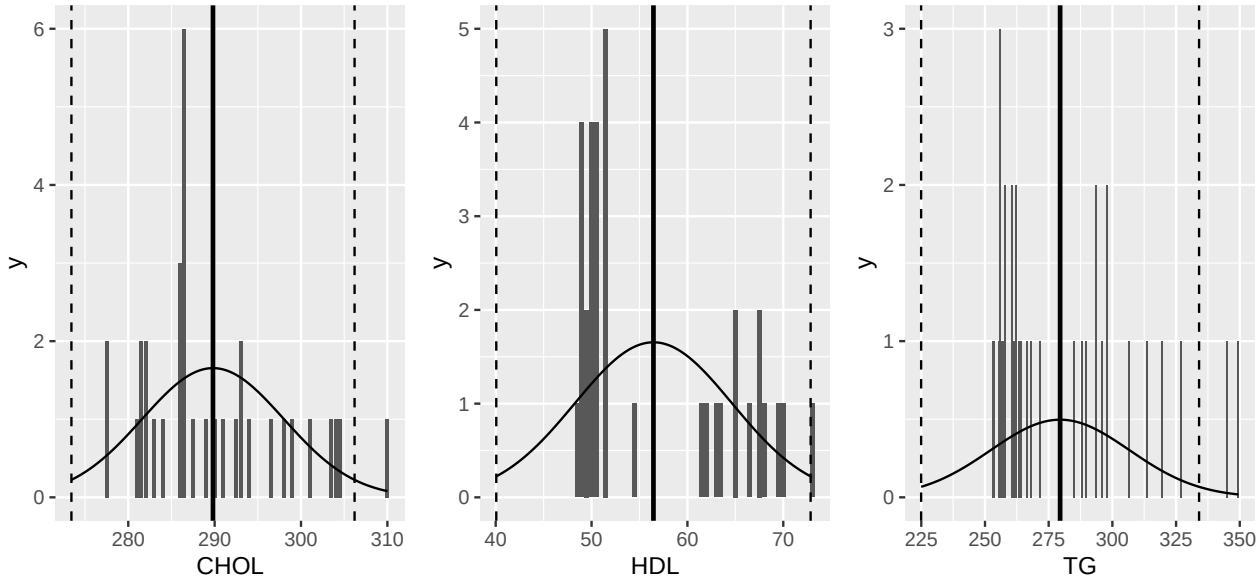


Figure 1: Discrete histogram and the corresponding normal distribution.

P and Q (which in this case P is the empirical distribution P_E and Q is the predicted normal distribution P_{NORM} with mean and standard deviation equal to those of the empirical distribution.

$$D_{KL}(P_E||P_{NORM}) = -\sum_{i}^{N} P_E(i) log\left(\frac{P_{NORM}(i)}{P_E(i)}\right) \tag{3}$$

D_{KL} (also referred to as relative entropy) is a measure of how much information we lose when we summarize a distribution using another one. In this case we aim to summarise the experimental distribution using only the mean and variance of the normal distribution. D_{KL} is not a distance in the sense that it is not symmetric (that is $D_{KL}(P||Q)$ is not necessarily equal to $D_{KL}(Q||P)$).

A modification of D_{KL} which is symmetric is the Jensen-Shannon divergence (or information radius):

$$D_{JS}(P_E||P_{NORM}) = \frac{1}{2}D_{KL}(P_E||M) + \frac{1}{2}D_{KL}(P_{NORM}||M) \tag{4}$$

where $M = \frac{1}{2}(P_E + P_{NORM})$.

However the D_{JS} is not a metric because it does not satisfy the triangle inequality. This can be rectified by replacing D_{JS} with its square root (the Jensen-Shannon distance metric, D^m_{JS}) [8,9].

Another useful property of the Jensen-Shannon distance is that, unlike the Kullback-Leibler divergence, it has both upper and lower bounds. Both D_{KL} and D_{JS} have a lower bound of 0, when the two distributions are identical. However, for the general case of D_{KL}, no upper bound exists. On the other hand, D_{JS} has an upper bound of 1, when (as is the case here), the base 2 logarithm is used: $0 \leq D_{JS} \leq 1$.

We will plot the discrete histogram of LDL of our sample and then calculate its Jensen-Shannon distance between the distributions of our sample and the corresponding normal distribution.

```
PlotDiscrHist(dfSmplA, param="LDL", title="LDL")
```

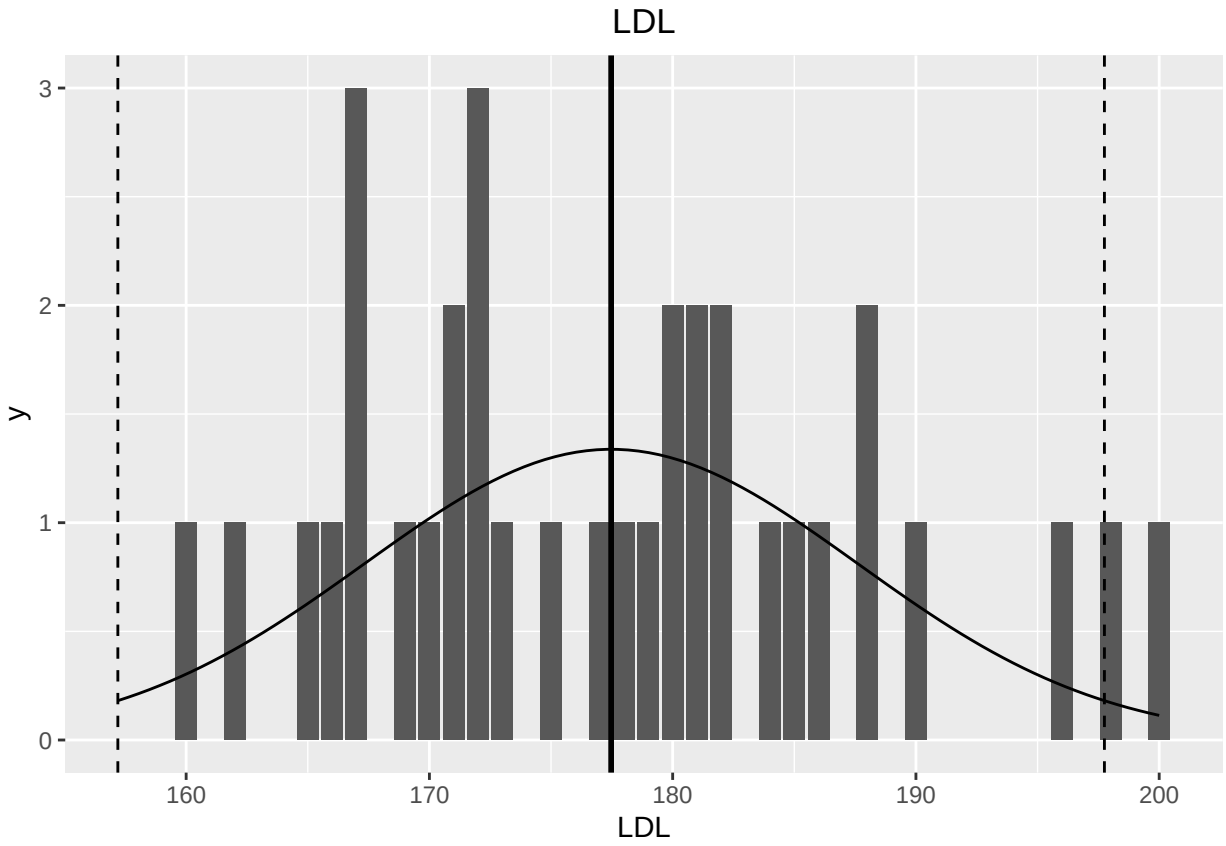


Figure 2: Discrete histogram and corresponding normal distribution of LDL.

```
JSDNormal(dfSmplA, "LDL")$JSD
#> Metric: 'jensen-shannon' using unit: 'log2'; comparing: 2 vectors.
#> jensen-shannon
#>      0.2638425
```

We can see in Figure 2 that there is a significant difference between the empirical LDL distribution and the corresponding normal distribution. Also, the value of D_{JS} as calculated is 0.26. To better see how the difference between two distributions affects the Jensen-Shannon distance, we can create two test distributions that are completely different and see the value the Jensen-Shannon distance takes.

```
distr1 <- seq(from=100, to=130, by=1)
distr2 <- seq(from=140, to=170, by=1)
JSD(distr1, distr2)$JSD
#> Metric: 'jensen-shannon' using unit: 'log2'; comparing: 2 vectors.
#> jensen-shannon
#>              1
```

We can see that when two distributions are completely different, $D_{JS} = 1$.

4.3 Chebyshev's inequality

In the normal distribution, 68.2%, 95.4% and 99.7% of values lie within 1, 2 and 3 standard deviations (SD) of the mean respectively. Hence the variance is a measure of the uncertainty. However, we saw above that it may not always be appropriate to use the normal distribution, but the empirical one. In this case, we can use Chebyshev's inequality [10,11] to find the proportion of data points that lie within a certain number of SD from the mean. If X is a random variable with finite mean μ and finite variance σ^2 and k is a positive real number, Chebyshev's inequality is:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{5}$$

It states that no more than $\frac{1}{k^2}$ of the values can be more than k SD away from the mean (or that at least $1 - \frac{1}{k^2}$ of the distribution's values are within k SD of the mean). For $k = 2$, at most $\frac{1}{2^2} = \frac{1}{4} = 25\%$ of the values are outside the $\mu \pm 2SD$ and at least 75% are within this range.

Although there are formal tests for normality (e.g. Kolmogorov-Smirnov and Shapiro-Wilk), in this case a simple visual inspection of the plot in Figure 2 shows clearly that the LDL distribution is far from normal. Therefore, we should not use the normal distribution intervals for the SD. Instead, we will use Chebyshev's inequality with the function `LDLcalc::chebyshev`.

```
LDLchebyshev <- chebyshev(vec=dfSmplA$LDL)
LDLchebyshev <- unlist(LDLchebyshev)
print(LDLchebyshev)
#> LowerBound UpperBound LowerRange UpperRange
#>        157        198        160        200
```

We can see from the results that the range for which at least 75% of the values should fall in is 157-198 mg/dl. In fact, 97% of the values are included in this range:

```
sum(dfSmplA$LDL>=157 & dfSmplA$LDL<=198) / nrow(dfSmplA) * 100
#> [1] 97.05882
```

4.4 Correlation of parameters

An assumption often made when dealing with multiple random variables is that they are independent and uncorrelated. In the case of CHOL, HDL and TG, even if they are independent is is certainly possible that they are correlated since, even if the determination reactions are independent, there are underlying common biological processes that affect the levels of many lipids.

To check for correlation, we will determine Pearson's correlation coefficient and the covariance for the pairs (CHOL-HDL), (CHOL- TG) and (HDL-TG) for the sample.

```
lstParam1 <- list("CHOL", "CHOL", "HDL")# List of the first parameter to calculate correlation on.
lstParam2 <- list("HDL", "TG", "TG")# List of the second parameter to calculate correlation on.
#Function to calculate correlation and covariance of the pairs of parameters
fCor <- function(param1, param2) {
  correlation <- round(cor(dfSmplA[, param1], dfSmplA[, param2]), 2)
  covariance <- round(cov(dfSmplA[, param1], dfSmplA[, param2]), 2)
  return(list(correlation=correlation, covariance=covariance))
}

mtrxCorCov <- mapply(fCor, lstParam1, lstParam2)
colnames(mtrxCorCov) <- c("CHOL-HDL", "CHOL-TG", "HDL-TG")

print(mtrxCorCov)
#>             CHOL-HDL CHOL-TG HDL-TG
#> correlation -0.35    0.64    -0.59
#> covariance  -23.68   143.93  -132.24
```

From the correlation coefficients we can conclude that in general there is a significant correlation between parameters, stronger (e.g. 0.64 for CHOL-TG) or weaker (e.g. -0.35 for CHOL-HDL).

4.4.1 Correlation plots We will also create scaterplots of the pairs of the parameters (CHOL-HDL, CHOL-TG and HDL-TG) along with the linear regression lines, using the `LDLcalc::PlotCorrWithRegrLine` function.

```
grid.arrange(PlotCorrWithRegrLine(dfSmplA, "CHOL", "HDL"),
             PlotCorrWithRegrLine(dfSmplA, "CHOL", "TG"),
             PlotCorrWithRegrLine(dfSmplA, "HDL", "TG"),
             ncol=3)
#> `geom_smooth()` using formula = 'y ~ x'
#> `geom_smooth()` using formula = 'y ~ x'
#> `geom_smooth()` using formula = 'y ~ x'
```

We can see in Figure 3 that in most cases the parameters cluster in two groups: one in which there does not seem to be any correlation and a smaller cluster of potential outliers which appears to affect the whole correlation.

In the GUM (and measurement theory in general) a distinction is made when the quantities are independent (section 5.1-Uncorrelated input quantities) and when they are correlated (section 5.2 - Correlated input quantities). As stated, the former case is *"valid only if the input quantities X_i are independent or uncorrelated... If some of the X_i are significantly correlated, the correlations must be taken into account"*. Since in this case the quantities are correlated (even if they may be independent),[2] it would be prudent not to make assumptions of no correlation and use instead the joint probability distribution function for error propagation, as we will see later.

[2]Note: When two random variables are independent, they are always uncorrelated. However, if they are uncorrelated (that is their correlation coefficient is zero $\rho(X, Y) = 0$), they can still be dependent.

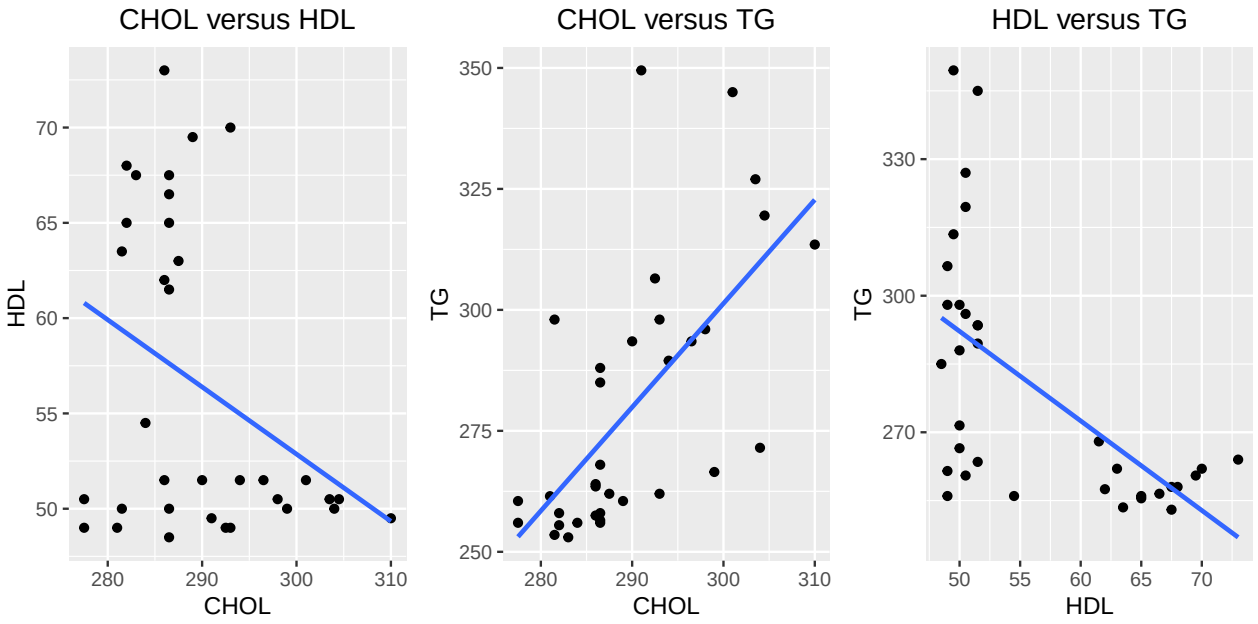


Figure 3: Scatterplot of pairs of the parameters.

5. Calculation of variance

5.1 Variance using the empirical distribution.

The most simple and obvious method to determine the variance of a calculated test would be to use the empirical distribution. In this case we would measure the experimentally measured quantities (CHOL, HDL and TG in this case) and use them to determine the calculated quantities (LDL and AIP). Then we would use these values to calculate the variance. This empirical distribution variance for both LDL and AIP has been calculated below:

```
vecEmpirVar <- c("LDL Empirical Variance" = round(var(dfSmplA$LDL), 2),
                 "AIP Empirical Variance" = round(var(dfSmplA$AIP), 4))
print(vecEmpirVar)
#> LDL Empirical Variance AIP Empirical Variance
#>                 102.8000                 0.0084
```

The reason this is not the correct approach is explained in [12]. Briefly, we have one or more random input variables (e.g. CHOL, HDL, TG) each one with its own distribution, and a function $f()$ of these variables (e.g. a formula for the calculation of LDL or AIP). The output is a random variable with its own distribution (e.g. LDL or AIP). As shown in Figure 4 and [12], when the input distribution is mapped through a non-linear function, the output distribution may be skewed.

5.2 Variance using error propagation

There are cases when a measurement is the result of a calculation of other experimentally measured variables. In that case, error propagation (or propagation of uncertainty) is the effect of the uncertainty (random error, quantified by the variance) of each variable on the uncertainty of a function based on them. This approach is also known as the Delta method ([13], Section 5.5.4).

To derive the formulas for the variance of LDL and AIP using error propagation, the general formula of the uncertainty for correlated input variables in the GUM section 5.2, equation (13) was used.

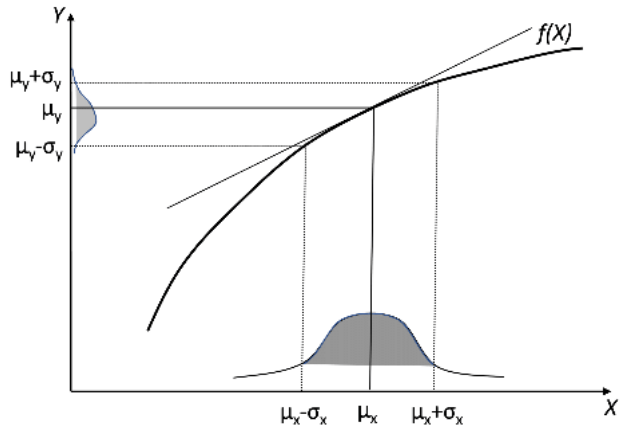


Figure 4: Mapping of the distribution of an independent random variable X to the (distorted) distribution of the dependent random variable Y, via the function $f(X)$.

$$\sigma_y^2 = \sum_{i=1}^{N} \left(\frac{\partial f}{\partial x_i} \right)^2 \sigma_{x_i}^2 + 2 \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} cov(x_i, x_j) \tag{6}$$

where x_i and x_j are the expected values of the random variables X_i and X_j, σ^2 is the variance of the subscripted variable and $cov(x_i, x_j)$ is the estimated covariance associated with x_i and x_j.

For LDL, if we set $x_1 \equiv CHOL, x_2 \equiv HDL$ and $x_3 \equiv TG$, the partial derivatives are $\frac{\partial LDL}{\partial CHOL} = 1$, $\frac{\partial LDL}{\partial HDL} = -1$ and $\frac{\partial LDL}{\partial TG} = -\frac{1}{5}$. Substituting in equation (6) we get:

$$\sigma_{LDL}^2 = \sigma_{CHOL}^2 + \sigma_{HDL}^2 + \frac{1}{5^2}\sigma_{TG}^2 - 2cov(CHOL, HDL) - \frac{2}{5}cov(CHOL, TG) + \frac{2}{5}cov(HDL, TG) \tag{7}$$

For AIP, if we set $x_1 \equiv TG$ and $x_2 \equiv HDL$, the partial derivatives are $\frac{\partial AIP}{\partial TG} = \frac{1}{TG}$ and $\frac{\partial LDL}{\partial HDL} = -\frac{1}{HDL}$ and the equation for the AIP variance is:

$$\sigma_{AIP}^2 = \frac{-2cov(x_1, x_2)\overline{x_2}\ \overline{x_1} + \sigma_{x_1}^2 \overline{x_2}^2 + \sigma_{x_2}^2 \overline{x_1}^2}{\overline{x_1}^2 \overline{x_2}^2 \log^2(10)} \tag{8}$$

where $\overline{x_i}$ is the expected value of the subscripted variable. Note that for AIP, TG and HDL must be expressed in $mmol/lt$. If expressed in mg/dl, TG must be multiplied by 0.0113 and HDL by 0.0259.

The above equations (7) and (8) that stem from the general form (6) are derived according to [13] (section 5.5.4) and the GUM. The proof involves an approximation using the first-order Taylor expansion of the mapping function $f()$. This approximation is sufficient when $f()$ is linear or approximately linear in the interval $[\bar{x} - \sigma_x, \bar{x} + \sigma_x]$, as is the case for LDL. However, in the case of AIP that involves a log ratio (and also in cases of other highly non-linear functions) this approximation will not hold, as shown later.

The variance was determined using the method of [13], but taking into account the second-order Taylor series term, too. The derived formula for the variance of AIP (given $x_1 \equiv TG, x_2 \equiv HDL$) is:

$$\sigma^2_{AIP} = \frac{\overline{x_2}^4 \mu_4^{x_1} - 4\overline{x_1}^3 \overline{x_1}^3 cor(x_1, x_2)\sigma_{x_1}\sigma_{x_2} +}{2\overline{x_1}^4 \overline{x_2}^4 log^2(10)} + \frac{2\overline{x_1}^2 \overline{x_2}^2 \sigma^2_{x_1} \left(\overline{x_2} - cor(x_1, x_2)\sigma_{x_2}\right)\left(\overline{x_2} + cor(x_1, x_2)\sigma_{x_2}\right)}{2\overline{x_1}^4 \overline{x_2}^4 log^2(10)} + \frac{\overline{x_1}^4 \left(2\overline{x_2}^2 \sigma^2_{x_2} + \mu_4^{x_2}\right)}{2\overline{x_1}^4 \overline{x_2}^4 log^2(10)} \tag{9}$$

where $\mu_4^{x_i}$ is the fourth central moment of the subscripted variable and $cor(x_1, x_2)$ is the correlation coefficient between x_1 and x_2 (where $x_1 \equiv TG$ and $x_2 \equiv HDL$).

After the theory, we will calculate the error propagation variance for LDL. We will also calculate the error propagation variance for AIP using both the first- and second-order Taylor approximation, as described above.

The LDL error propagation variance is calculated using the function `LDLcalc::LDLErrPrp`.

```
LDLErrPrpVar <- LDLErrPrp(CHOL = dfSmplA$CHOL,
                                    HDL = dfSmplA$HDL,
                                    TG = dfSmplA$TG)
print(round(LDLErrPrpVar), 2)
#> [1] 101
```

The AIP first- and second-order error propagation variance is:

```
AIPErrPrpVar1Ord <-
  AIPErrPrp(TG = dfSmplA$TG,
                        HDL = dfSmplA$HDL,
                        SI = F)

AIPErrPrpVar2Ord <-
  AIPErrPrp2Ord(TG = dfSmplA$TG,
                        HDL = dfSmplA$HDL,
                        SI = F)

AIPErrProp <- c("AIP First Order Error Propagation" = AIPErrPrpVar1Ord,
                "AIP Second Order Error Propagation" = AIPErrPrpVar2Ord)
print(AIPErrProp)
#>  AIP First Order Error Propagation AIP Second Order Error Propagation
#>                            0.00893                             0.00885
```

If we put together the empirical and error propagation variance for LDL in Table 1:

we can see that these two variances (empirical and error propagation) are approximately equal. This is not surprising, as they are essentially derived using the same formula and underlying principles. However, the error propagation framework makes the assumptions that errors are linear and that the distribution of the output quantity can be approximated by a normal or t- distribution. We saw above that this is not always the case. To test the effect of these assumptions (when assumed erroneously) we will use bootstrapping.

5.3. Variance using bootstrapping.

Bootstrapping is a method that uses random sampling with replacement. Bootstrapping for variance determination works as follows:

Table 1: Empirical and Error Propagation Variance for LDL and AIP

Variance type	Variance Value
LDL	
Empirical Variance	102.8021
Error Propagation Variance	101.0134
AIP	
Empirical Variance	0.00838
First-Order Error Propagation	0.00893
Second-Order Error Propagation	0.00885

1. Take a data set of n measurements (those measurements have an empirical -that is experimental- distribution).
2. Use random sampling with replacement from these n measurements to create a new data set consisting of at most n measurements.
3. During the sampling process, if a measurement is selected, put it back so that it can be selected again (this is random sampling with replacement).
4. Determine the variance of the data set that was created by sampling and store the value.
5. Repeat the above process many times (e.g. >1000) and store the resulting variance of each iteration.
6. The set of variances calculated follow some distribution. Report either the whole distribution or some measure of it (e.g. mean, median, range etc).

For LDL and AIP variance determination using bootstrapping, samples are created from CHOL, HDL and TG. It should be noted that they are created on a per row basis for all instances of CHOL, HDL and TG. This means that if e.g. row 30 is sampled, CHOL, HDL and TG of row 30 are used. Since all three parameters are measured at the same time for each row, it would be wrong to sample them separately.

The bootstrap variance of LDL can be calculated with the function `LDLcalc::LDLbootVrnc` and that of AIP with `LDLcalc::AIPbootVrnc`. (*Note: These functions use a data table, from the data.table package, because it is much faster than a standard data frame when there is a large number of iterations. (Due to time constrains noofreps=10 instead of the suggested 10000)*)

```
LDLbootVar <- LDLbootVrnc(CHOL = dfSmplA$CHOL, HDL = dfSmplA$HDL,
                          TG = dfSmplA$TG, noOfReps = 10, pb=F)

AIPbootVar <- AIPbootVrnc(TG = dfSmplA$TG, HDL = dfSmplA$HDL,
                              SI= F, noOfReps = 10, pb=F)
```

Table summarizing all variances

The bootstrap variance is not a single value but a distribution of values of variances. In the previous table with the empirical and error propagation variances, we will also insert the bootstrap variances of LDL and AIP. Specifically, we will use the median of the variance distribution as the estimator. In Table 2, all variances are shown.

Plot LDL variance density

Next, in Figure 5, we will plot the density of the bootstrap variance distribution for LDL, along with the median and 2.5 and 97.5 percentiles. In the same density graph we will also plot the empirical and error propagation variances. We will use the function `LDLcalc::LDL_DensityPlotOfbootst`.

```
LDL_DensityPlotOfbootst(LDLbootVar$dataTable,
                                    empirVrnc = vecEmpirVar[[1]],
                                    errPropVrnc = LDLErrPrpVar)
```

Table 2: Empirical, Error Propagation and Bootstrap Variance for LDL and AIP

Variance type	Variance Value
LDL	
Empirical Variance	102.8021
Error Propagation Variance	101.0134
Bootstrap Variance	100.3886
AIP	
Empirical Variance	0.00838
First-Order Error Propagation	0.00893
Second-Order Error Propagation	0.00885
Bootstrap Variance	0.00852

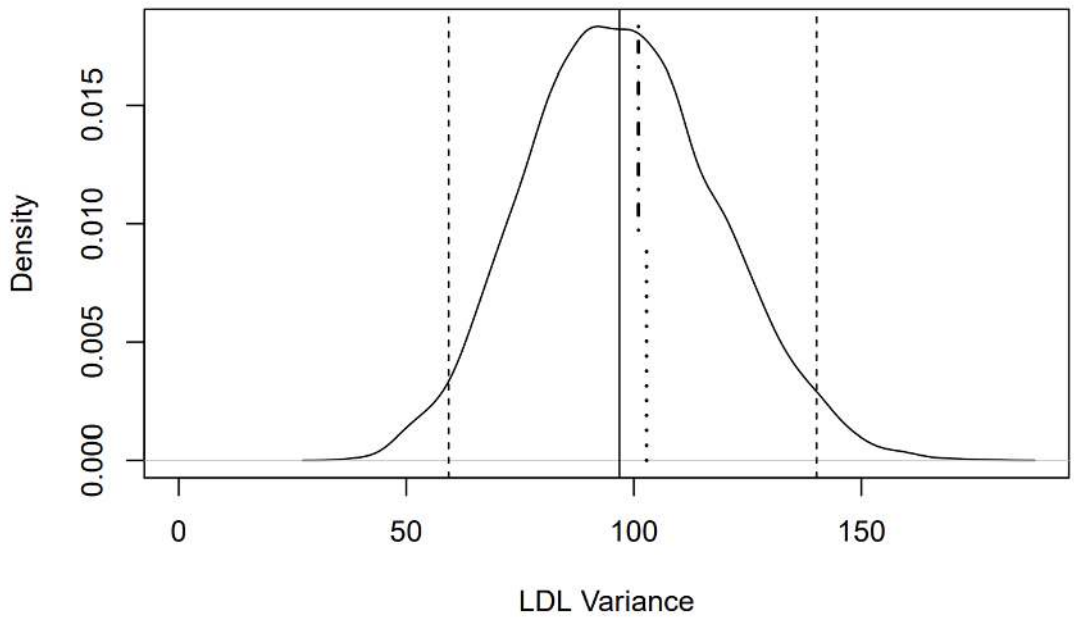


Figure 5: Distribution density of the bootstrap LDL variance. The median is shown as a continuous vertica line and the 2.5 and 97.5 percentiles as dashed vertical lines. The empirical variance is depicted as a dotted vertical segment from the bottom to the middle of the graph. The error propagation variance is depicted as a dash-dot vertical segment from the top to the middle.

Plot AIP variance density

We will also plot in Figure 6 the density of the bootstrap variance distribution, along with the median and 2.5 and 97.5 percentiles for AIP, along with tthe empirical and error propagation variances.

```
AIP_DensityPlotOfbootst(AIPbootVar$dataTable,
                                  empirVrnc = vecEmpirVar[[2]],
                                  errPropVrnc = AIPErrPrpVar1Ord,
                                  errPropVrnc2Ord = AIPErrPrpVar2Ord)
```

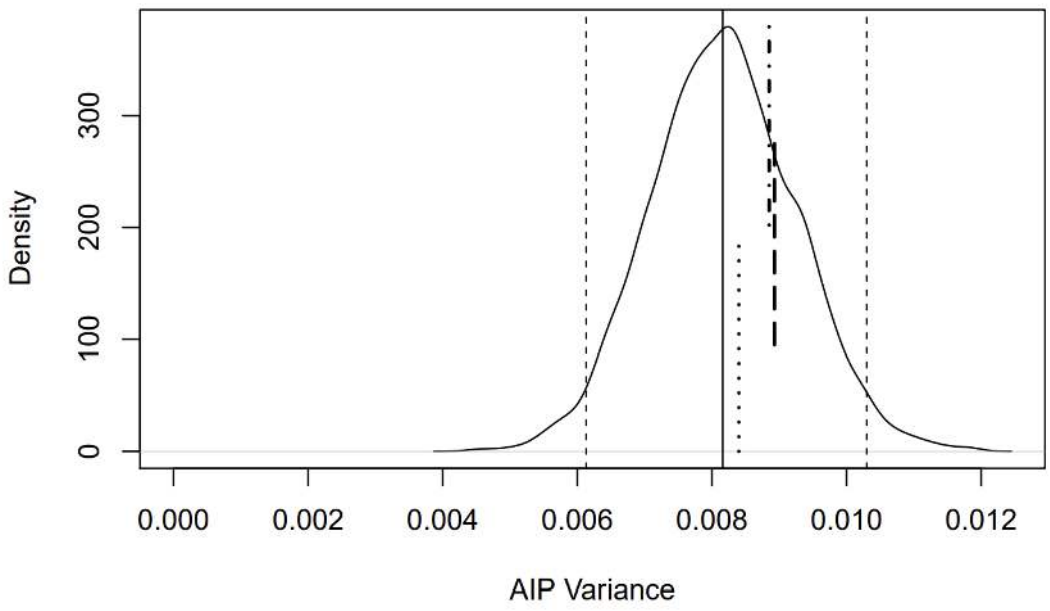


Figure 6: Distribution density of the bootstrap AIP variance. The median is shown as a continuous vertical line and the 2.5 and 97.5 percentiles as dashed vertical lines. The empirical variance is depicted as a dotted vertical segment from the bottom to the middle of the graph. The error propagation variance is depicted as a dash-dot vertical segment from the top to the middle and the second order error propagation variance as a dashed segment in the middle.

It is evident from both Table 2 and Figures 5 and 6 that the bootstrap median variance is lower than the empirical and error propagation variances.

6 Variance using bootstrapping with changing variances of CHOL, HDL or TG

To see if this difference between variances occurs not only for that specific sample and to determine its behavior in general, we determined the variance of LDL using both the error propagation formula and bootstrapping using a series of distributions with increasing variance of CHOL, HDL and TG (one at a time) while keeping the mean constant.

A series of distributions of increasing variance and constant mean were created. The mean was kept equal to the mean of each sample. For each distribution with increasing variance of CHOL, HDL and TG (separately), the LDL and AIP variances were calculated using the empirical, error propagation and bootstrap approaches.

6.1 LDL Variance when variances of CHOL, HDL and TG change.

6.1.1 LDL variance with changing CHOL variance

First we will create a data frame whose columns will be distributions of CHOL with increasing CV. We will use the function `LDLcalc::CV_Range`. The CV of the initial CHOL distribution is 2.83.

```
CV(dfSmplA$CHOL)
#> [1] 2.83
```

We will create a data frame whose columns will be distributions of CHOL, with increasing CV from 0 to 20%. The function uses a stochastic algorithm to change the CV values from the lower to the upper value desired. If these bounds have not been achieved after maxRandIter iterations (default=10000), the function will exit, in order to avoid a possible infinite loop. The function to be used is `LDLcalc::CV_Range`. This function can also plot the mean and the CV of the resulting data frame columns.

```
dfSmplACHOLChngCV <- CV_Range(dfSmplA$CHOL,
                                        lower_CV_Bound = 0,
                                        upper_CV_Bound = 20,
                                        maxRandIter = 1000,
                                        plot=T)
```

We can see in Figure 7 that the mean stays constant from the first to the last columns, whereas the CV increases from 0 to 20.

After creating the data frame with increasing CHOL variances (dfSmplACHOLChngCV), we will use it to calculate the error propagation and the bootstrap variance of all these distributions. Essentialy, it will use bootstrap to calculate the variance of each column of the data frame. ***Note: This may take a long time, depending on the computer speed.***

```
LDL_Vrnc_Chng_CHOL <- LDL_CHOLVrnc(dfSmplACHOLChngCV,
                                                dfSmplA$HDL,
                                                dfSmplA$TG,
                                                bootStrpReps = 500)
```

Next we will create a plot with the changing variance of CHOL in the abscissa and the variance of LDL (error propagation and bootstrap in the ordinate).

```
# First we will convert the list output of LDLcalc::LDL_CHOLVrnc to a
# data frame
LDL_Vrnc_Chng_CHOL <- do.call(cbind.data.frame, LDL_Vrnc_Chng_CHOL)

# Create a data frame with the changing CHOL varince and the error propagation
# and bootstrap variances as columns, in order to use it for ggplot and
# remove the original data frame.
dfCHOLChngVrnc <- cbind.data.frame(CHOLVrnc= apply(dfSmplACHOLChngCV, 2, var),
                 LDLErrPrp = LDL_Vrnc_Chng_CHOL$ErrPropVrnc,
                 LDLBootstrp = LDL_Vrnc_Chng_CHOL$BootVrnc)
rm(LDL_Vrnc_Chng_CHOL)

dfCHOLChngVrnc <- pivot_longer(dfCHOLChngVrnc, cols=2:3,
                               names_to = "Variance_Type",
                               values_to = "Variance")
```

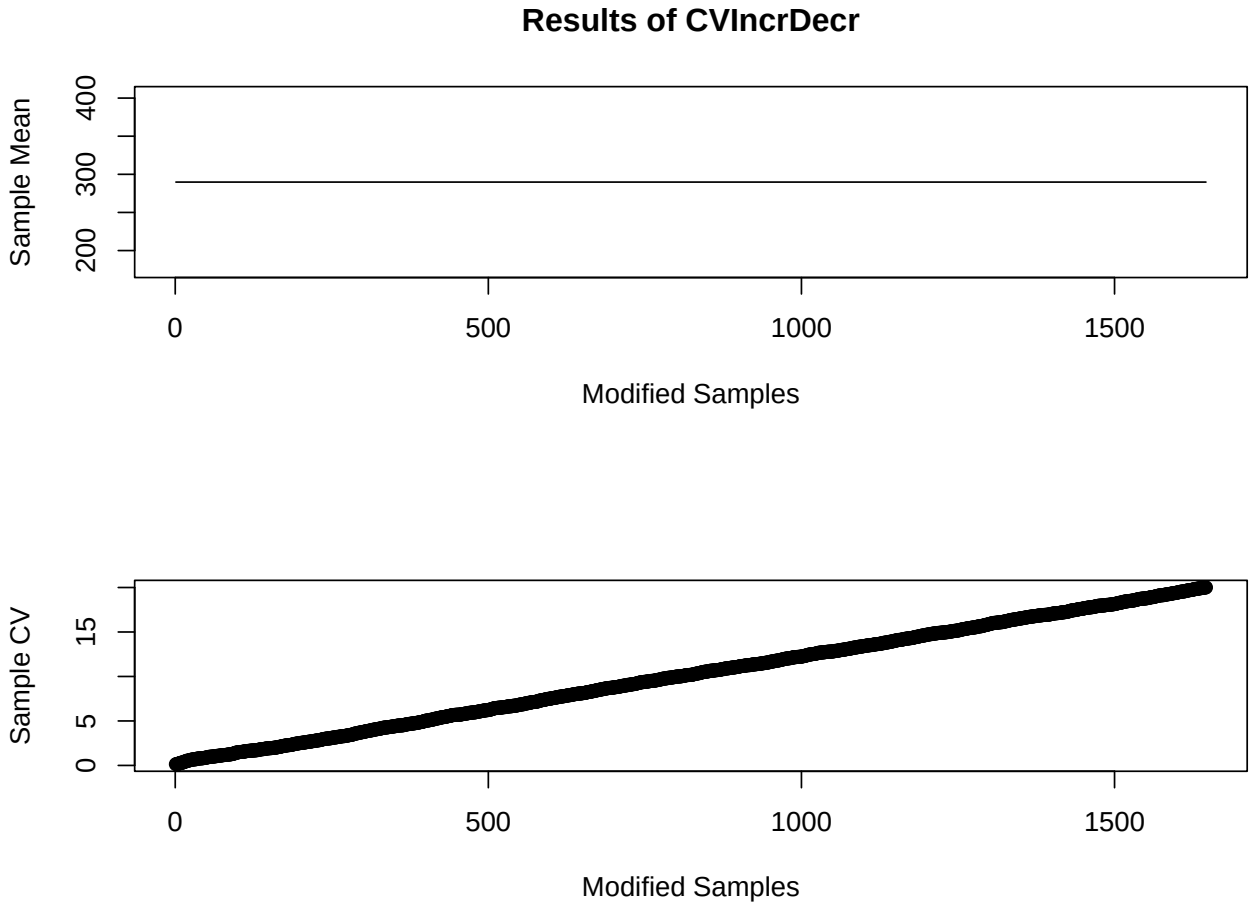


Figure 7: Means and variances for the CHOL distributions, in order to check that the mean stays constant and the variance increases.

```
ggplot(data=dfCHOLChngVrnc,
       aes(x=CHOLVrnc, y=Variance, linetype=Variance_Type)) +
  geom_line() +
  xlab("Cholesterol Variance") + ylab("LDL Variance") +
  scale_linetype_discrete(name="Variance Type",
                          labels = c("Bootstrap Variance",
                                     "Error Propagation Variance"))
```

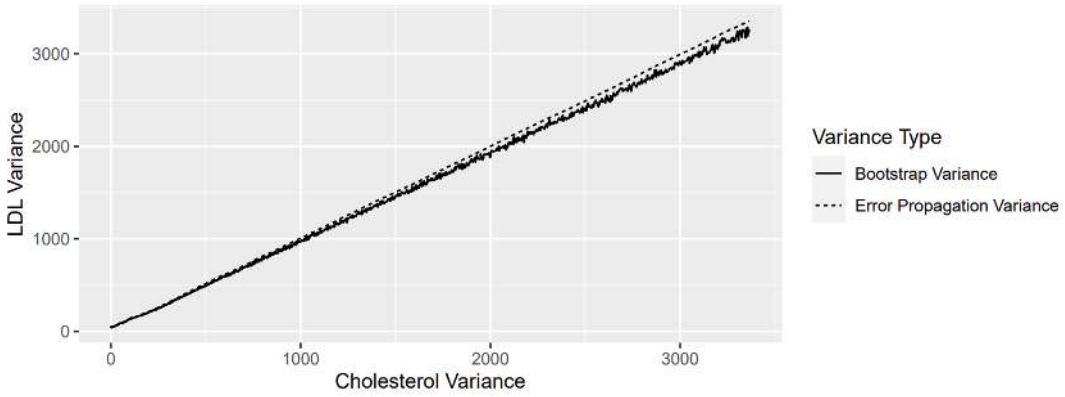


Figure 8: Variance of LDL (error propagation and boostrap for 500 iterations), when the CHOL variance changes.

We can see in Figure 8 that the line of the bootstrap variance is not very smooth. This is because of the relatvely few bootstrap iterations we used (500). If we use more (e.g. 2000) the resulting line will be smoother, as below:

```
LDL_Vrnc_Chng_CHOL <- LDL_CHOLVrnc(dfSmplACHOLChngCV,
                                   dfSmplA$HDL,
                                   dfSmplA$TG,
                                   bootStrpReps = 2000)
```

```
LDL_Vrnc_Chng_CHOL <- do.call(cbind.data.frame, LDL_Vrnc_Chng_CHOL)

dfCHOLChngVrnc <- cbind.data.frame(CHOLVrnc= apply(dfSmplACHOLChngCV, 2, var),
                LDLErrPrp = LDL_Vrnc_Chng_CHOL$ErrPropVrnc,
                LDLBootstrp = LDL_Vrnc_Chng_CHOL$BootVrnc)
rm(LDL_Vrnc_Chng_CHOL)

dfCHOLChngVrnc <- pivot_longer(dfCHOLChngVrnc, cols=2:3,
                               names_to = "Variance_Type",
                               values_to = "Variance")
pltLDL_CHOLVrnc <-
  ggplot(data=dfCHOLChngVrnc,
         aes(x=CHOLVrnc, y=Variance, linetype=Variance_Type)) +
    geom_line() +
    xlab("Cholesterol Variance") + ylab("LDL Variance") +
    scale_linetype_discrete(name="Variance Type",
                            labels = c("Bootstrap Variance",
```

```
                                        "Error Propagation Variance"))
pltLDL_CHOLVrnc
```

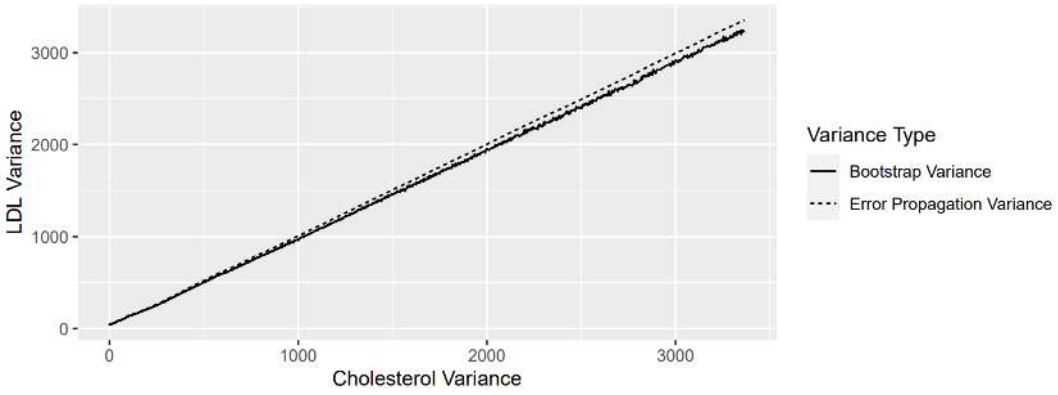


Figure 9: Variance of LDL (error propagation and boostrap for 2000 iterations), when the CHOL variance changes

We can see in Figure 9, that with 2000 bootstrap iterations the line becomes smoother.

6.1.2 LDL variance with changing HDL variance

Like in the case of CHOL, we will create a data frame whose columns will be distributions of HDL with increasing CV with the function `LDLcalc::CV_Range`. The CV of the initial HDL distribution is 14.52.

```
CV(dfSmplA$HDL)
#> [1] 14.52
```

We will again create a data frame whose columns will be distributions of HDL, with increasing CV from 0 to 20%.

```
dfSmplAHDLChngCV <- CV_Range(dfSmplA$HDL,
                                        lower_CV_Bound = 0,
                                        upper_CV_Bound = 20,
                                        maxRandIter = 10000,
                                        plot=T)
```

We can see in Figure 10 that the mean stays constant and the CV increases from 0 to 20.

After creating the data frame with increasing HDL variances (dfSmplAHDLChngCV), we will use it to calculate the error propagation and the bootstrap variance of these distributions. This time we will use 2000 bootstrap iterations, since this gives us a smoother line, even though it will take longer ***Note: This may take a long time, depending on the computer speed.***

```
LDL_Vrnc_Chng_HDL <- LDL_HDLVrnc(dfSmplAHDLChngCV,
                                                dfSmplA$CHOL,
                                                dfSmplA$TG,
                                                bootStrpReps = 2000)
```

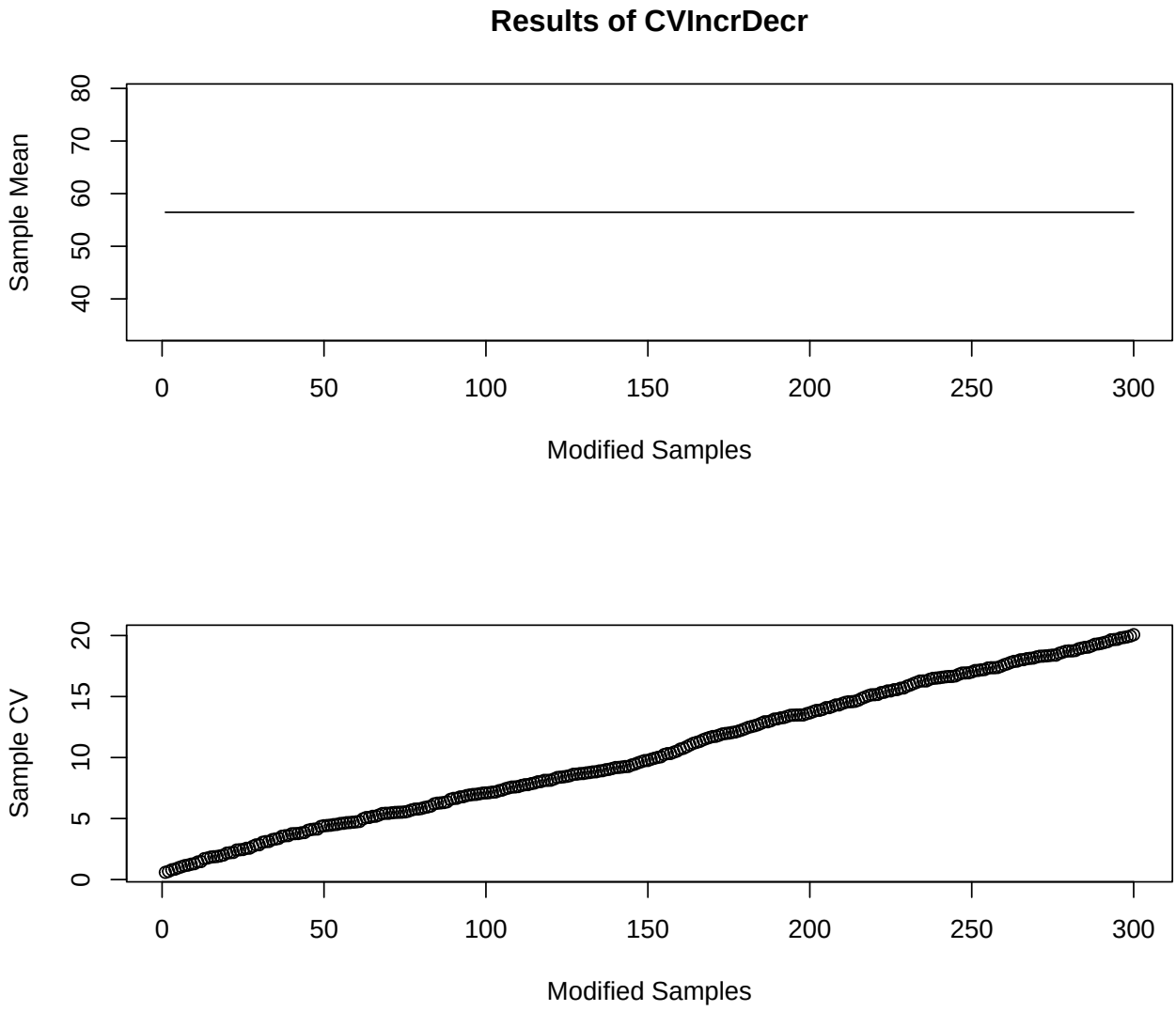


Figure 10: Means and variances for the HDL distributions, in order to check that the mean stays constant and the variance increases.

Next, as before we will create a plot with the changing variance of HDL in the abscissa and the variance of LDL (error propagation and bootstrap in the ordinate).

We can see in Figure 11 the effect of the increasing HDL variance.

```
LDL_Vrnc_Chng_HDL <- do.call(cbind.data.frame, LDL_Vrnc_Chng_HDL)

# Create a data frame with the changing HDL variance and the error propagation
# and bootstrap variances as columns, in order to use it for ggplot and
# remove the original data frame.
dfHDLChngVrnc <- cbind.data.frame(HDLVrnc= apply(dfSmplAHDLChngCV, 2, var),
                 LDLErrPrp = LDL_Vrnc_Chng_HDL$ErrPropVrnc,
                 LDLBootstrp = LDL_Vrnc_Chng_HDL$BootVrnc)
rm(LDL_Vrnc_Chng_HDL)

dfHDLChngVrnc <- pivot_longer(dfHDLChngVrnc, cols=2:3,
                              names_to = "Variance_Type",
                              values_to = "Variance")
pltLDL_HDLVrnc <-
  ggplot(data=dfHDLChngVrnc,
         aes(x=HDLVrnc, y=Variance, linetype=Variance_Type)) +
    geom_line() +
    xlab("HDL Variance") + ylab("LDL Variance") +
    scale_linetype_discrete(name="Variance Type",
                            labels = c("Bootstrap Variance",
                                       "Error Propagation Variance"))
pltLDL_HDLVrnc
```

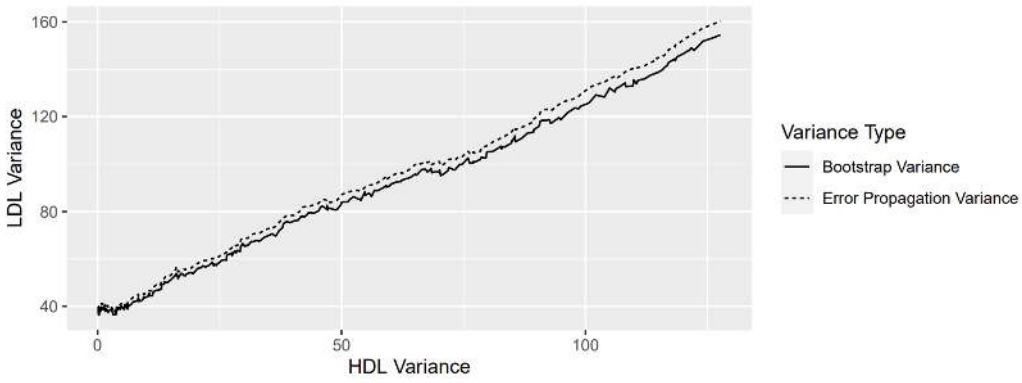


Figure 11: Variance of LDL (error propagation and boostrap for 2000 iterations), when the HDL variance changes.

6.1.3 LDL variance with changing TG variance

Like in the case of CHOL and HDL, we will create a data frame whose columns will be distributions of TG with increasing CV with the function `LDLcalc::CV_Range`. The CV of the initial TG distribution is 9.76.

```
CV(dfSmplA$TG)
#> [1] 9.76
```

We will again create a data frame whose columns will be distributions of TG, with increasing CV from 0 to 20%.

```
dfSmplATGChngCV <- CV_Range(dfSmplA$TG,
                                        lower_CV_Bound = 0,
                                        upper_CV_Bound = 20,
                                        maxRandIter = 10000,
                                        plot=T)
```

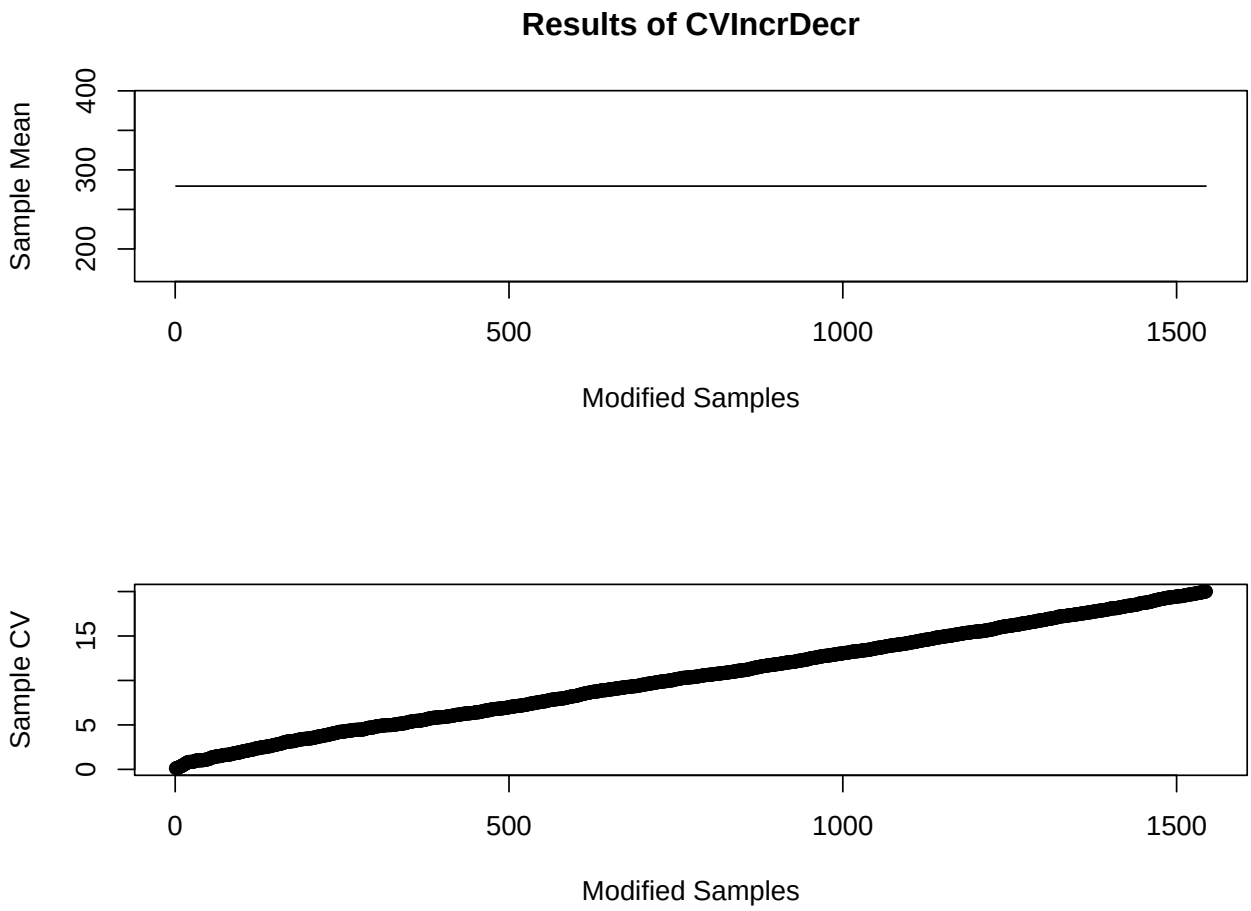


Figure 12: Means and variances for the TG distributions, in order to check that the mean stays constant and the variance increases.

We can see again in Figure 12 that the mean stays constant, whereas the CV increases from 0 to 20.

After creating the data frame with increasing TG variances (dfSmplATGChngCV), we will use it to calculate the error propagation and the bootstrap variance of these distributions, with 2000 bootstrap iterations

```
LDL_Vrnc_Chng_TG <- LDL_TGVrnc(dfSmplATGChngCV,
                                            dfSmplA$CHOL,
                                            dfSmplA$HDL,
                                            bootStrpReps = 2000)
```

Next, in Figure 13, we will again create a plot with the changing variance of TG in the abscissa and the variance of LDL (error propagation and bootstrap in the ordinate).

```r
LDL_Vrnc_Chng_TG <- do.call(cbind.data.frame, LDL_Vrnc_Chng_TG)

# Create a data frame with the changing TG varince and the error propagation
# and bootstrap variances as columns, in order to use it for ggplot and
# remove the original data frame.
dfTGChngVrnc <- cbind.data.frame(TGVrnc= apply(dfSmplATGChngCV, 2, var),
                 LDLErrPrp = LDL_Vrnc_Chng_TG$ErrPropVrnc,
                 LDLBootstrp = LDL_Vrnc_Chng_TG$BootVrnc)
rm(LDL_Vrnc_Chng_TG)

dfTGChngVrnc <- pivot_longer(dfTGChngVrnc, cols=2:3,
                             names_to = "Variance_Type",
                             values_to = "Variance")
pltLDL_TGVrnc <-
  ggplot(data=dfTGChngVrnc,
         aes(x=TGVrnc, y=Variance, linetype=Variance_Type)) +
    geom_line() +
    xlab("TG Variance") + ylab("LDL Variance") +
    scale_linetype_discrete(name="Variance Type",
                            labels = c("Bootstrap Variance",
                                       "Error Propagation Variance"))
pltLDL_TGVrnc
```

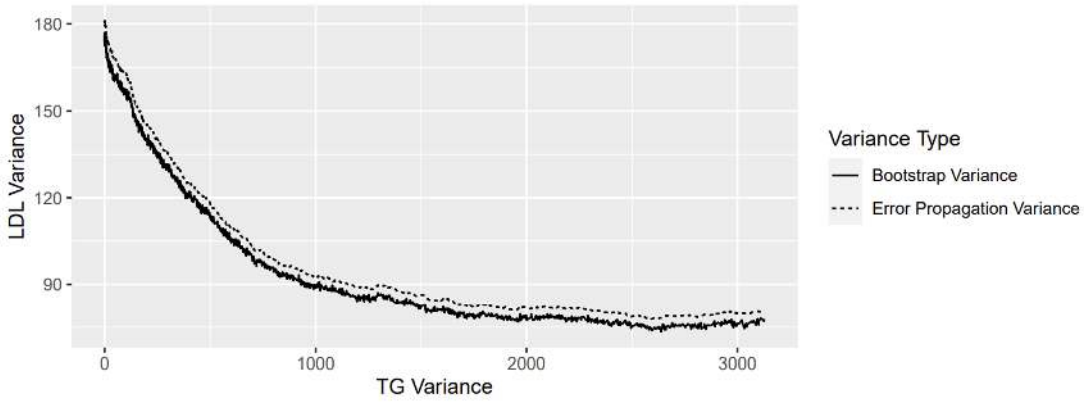


Figure 13: Variance of LDL (error propagation and boostrap for 2000 iterations), when the TG variance changes.

6.2 AIP variance when variances of HDL and TG change

6.2.1 AIP variance with changing HDL variance

Next we will see what happens to the AIP variance when the HDL variance changes. Like in the case of LDL, we will create a data frame whose columns will be distributions of HDL with increasing CV with the function `LDLcalc::CV_Range`. The CV of the initial HDL distribution is 2.83.

```r
CV(dfSmplA$HDL)
#> [1] 14.52
```

We will again create a data frame whose columns will be distributions of HDL, with increasing CV from 0 to 20%.

```
dfSmplAHDLChngCV <- CV_Range(dfSmplA$HDL,
                                        lower_CV_Bound = 0,
                                        upper_CV_Bound = 20,
                                        maxRandIter = 10000,
                                        plot=T)
```

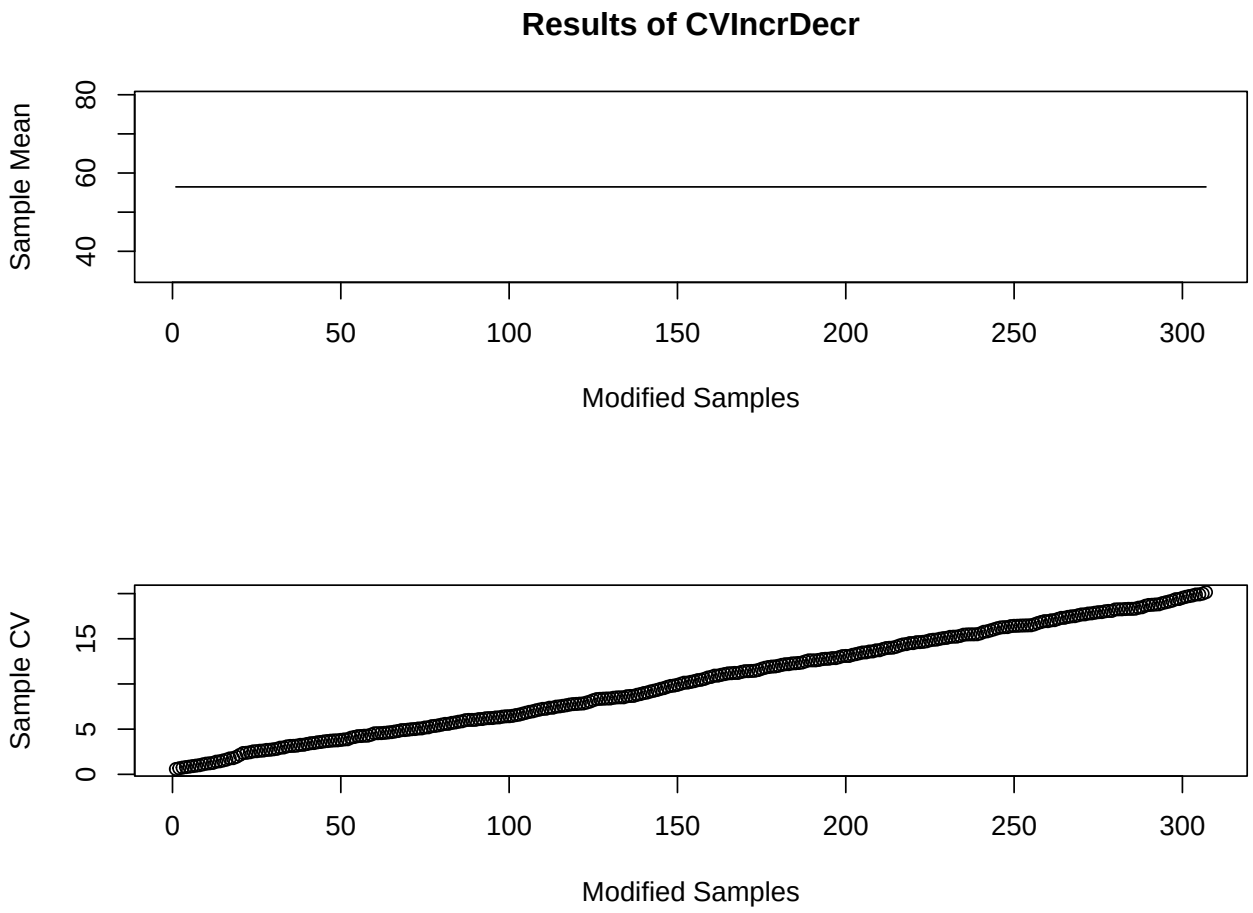


Figure 14: Means and variances for the HDL distributions, in order to check that the mean stays constant and the variance increases.

We can see again in Figure 14 that the mean stays constant, whereas the CV increases from 0 to 20.

Now, we will use the data frame with increasing HDL variances (dfSmplAHDLChngCV), to calculate the error propagation and the bootstrap variance of these distributions. ***Note: This may take a long time, depending on the computer speed.***

```
AIP_Vrnc_Chng_HDL <- AIP_HDLVrnc(dfSmplAHDLChngCV,
                                            dfSmplA$TG,
                                            SI=FALSE,
                                            bootStrpReps = 2000)
```

Next, in Figure 15, we will create a plot with the changing variance of HDL in the abscissa and the variance of AIP (first and second order error propagation and bootstrap in the ordinate).

```
AIP_Vrnc_Chng_HDL <- do.call(cbind.data.frame, AIP_Vrnc_Chng_HDL)

# Create a data frame with the changing HDL varince and the error propagation
# and bootstrap variances as columns, in order to use it for ggplot and
# remove the original data frame.
dfHDLChngVrnc <- cbind.data.frame(HDLVrnc= apply(dfSmplAHDLChngCV, 2, var),
                 AIPErrPrp = AIP_Vrnc_Chng_HDL$ErrPropVrnc,
                 AIPErrPrp2Ord = AIP_Vrnc_Chng_HDL$ErrPropVrnc2Ord,
                 AIPBootstrp = AIP_Vrnc_Chng_HDL$BootVrnc)
rm(AIP_Vrnc_Chng_HDL)

dfHDLChngVrnc <- pivot_longer(dfHDLChngVrnc, cols=2:4,
                              names_to = "Variance_Type",
                              values_to = "Variance")
pltAIP_HDLVrnc <-
  ggplot(data=dfHDLChngVrnc,
         aes(x=HDLVrnc, y=Variance, linetype=Variance_Type)) +
    geom_line() +
    xlab("HDL Variance") + ylab("AIP Variance") +
    scale_linetype_discrete(name="Variance Type",
                            labels = c("Bootstrap Variance",
                                       "Error Propagation Variance",
                                       "Error Propagation Variance 2nd Order"))
pltAIP_HDLVrnc
```

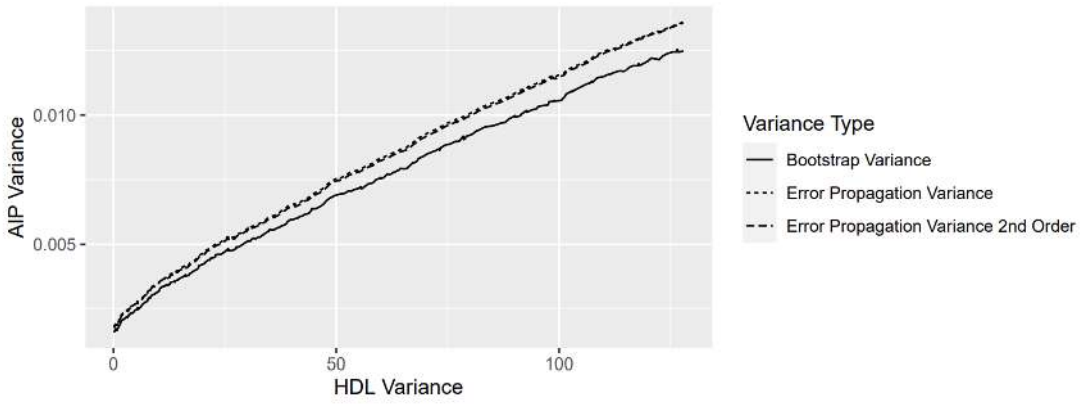


Figure 15: Variance of AIP (error propagation and boostrap for 2000 iterations), when the HDL variancechanges.

6.2.2 AIP variance with changing TG variance

Next we will see what happens to the AIP variance when the TG variance changes. The CV of the initial TG distribution is 9.76.

```
CV(dfSmplA$TG)
#> [1] 9.76
```

We will again create a data frame whose columns will be distributions of HDL, with increasing CV from 0 to 20%.

```
dfSmplATGChngCV <- CV_Range(dfSmplA$TG,
                                        lower_CV_Bound = 0,
                                        upper_CV_Bound = 20,
                                        maxRandIter = 10000,
                                        plot=T)
```

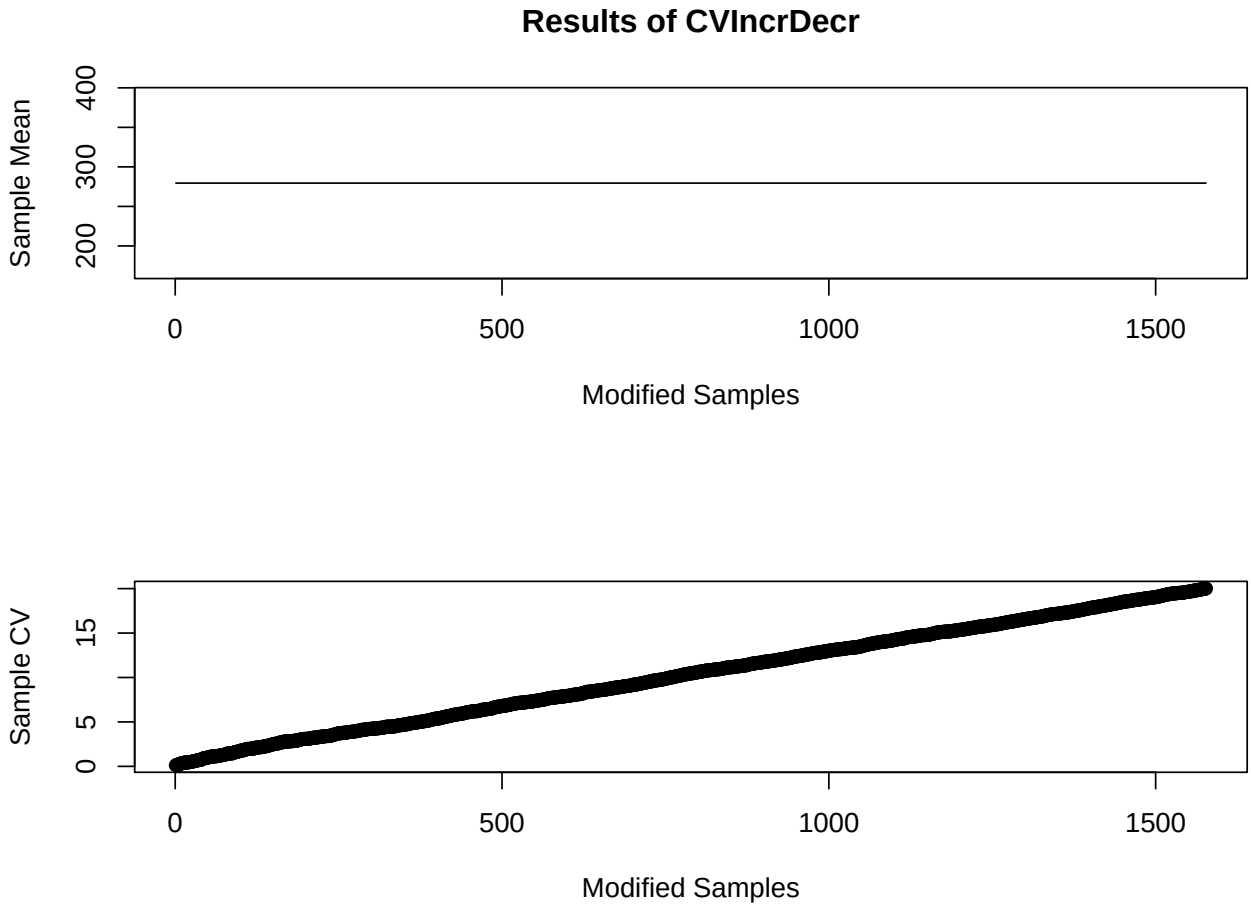


Figure 16: Means and variances for the TG distributions, in order to check that the mean stays constant and the variance increases.

We can see again in Figure 16 that the mean stays constant, whereas the CV increases from 0 to 20.

Now, we will use the data frame with increasing TG variances (dfSmplATGChngCV), to calculate the error propagation and the bootstrap variance of these distributions. ***Note: This may take a long time, depending on the computer speed.***

```
AIP_Vrnc_Chng_TG <- AIP_TGVrnc(dfSmplATGChngCV,
                                        dfSmplA$HDL,
                                        SI=FALSE,
                                        bootStrpReps = 2000)
```

Next, we will create a plot in Figure 17, with the changing variance of TG in the abscissa and the variance of AIP (error propagation and bootstrap in the ordinate).

```r
AIP_Vrnc_Chng_TG <- do.call(cbind.data.frame, AIP_Vrnc_Chng_TG)

# Create a data frame with the changing TG variance and the error propagation
# and bootstrap variances as columns, in order to use it for ggplot and
# remove the original data frame.
dfTGChngVrnc <- cbind.data.frame(TGVrnc= apply(dfSmplATGChngCV, 2, var),
                 AIPErrPrp = AIP_Vrnc_Chng_TG$ErrPropVrnc,
                 AIPErrPrp2Ord = AIP_Vrnc_Chng_TG$ErrPropVrnc2Ord,
                 AIPBootstrp = AIP_Vrnc_Chng_TG$BootVrnc)
rm(AIP_Vrnc_Chng_TG)

dfTGChngVrnc <- pivot_longer(dfTGChngVrnc, cols=2:4,
                             names_to = "Variance_Type",
                             values_to = "Variance")
pltAIP_TGVrnc <-
  ggplot(data=dfTGChngVrnc,
         aes(x=TGVrnc, y=Variance, linetype=Variance_Type)) +
    geom_line() +
    xlab("TG Variance") + ylab("AIP Variance") +
    scale_linetype_discrete(name="Variance Type",
                            labels = c("Bootstrap Variance",
                                       "Error Propagation Variance",
                                       "Error Propagation Variance 2nd Order"))
pltAIP_TGVrnc
```

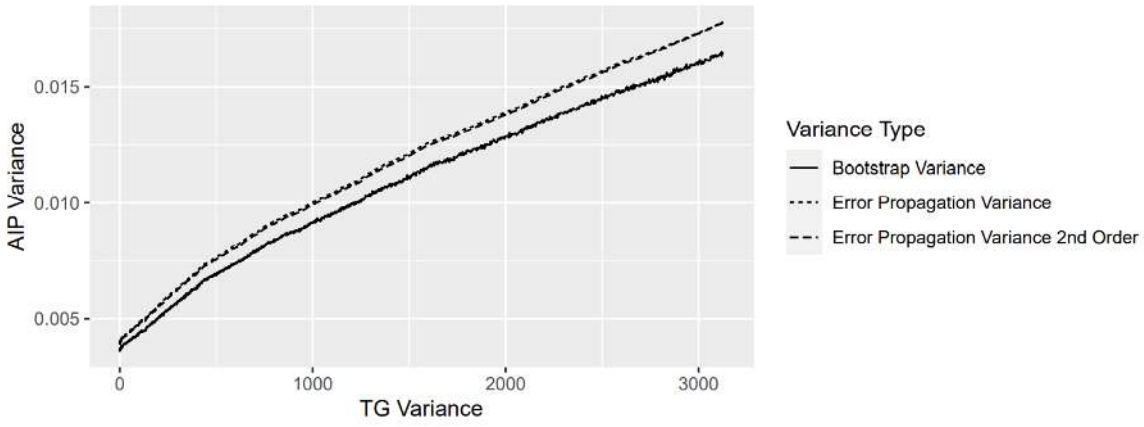
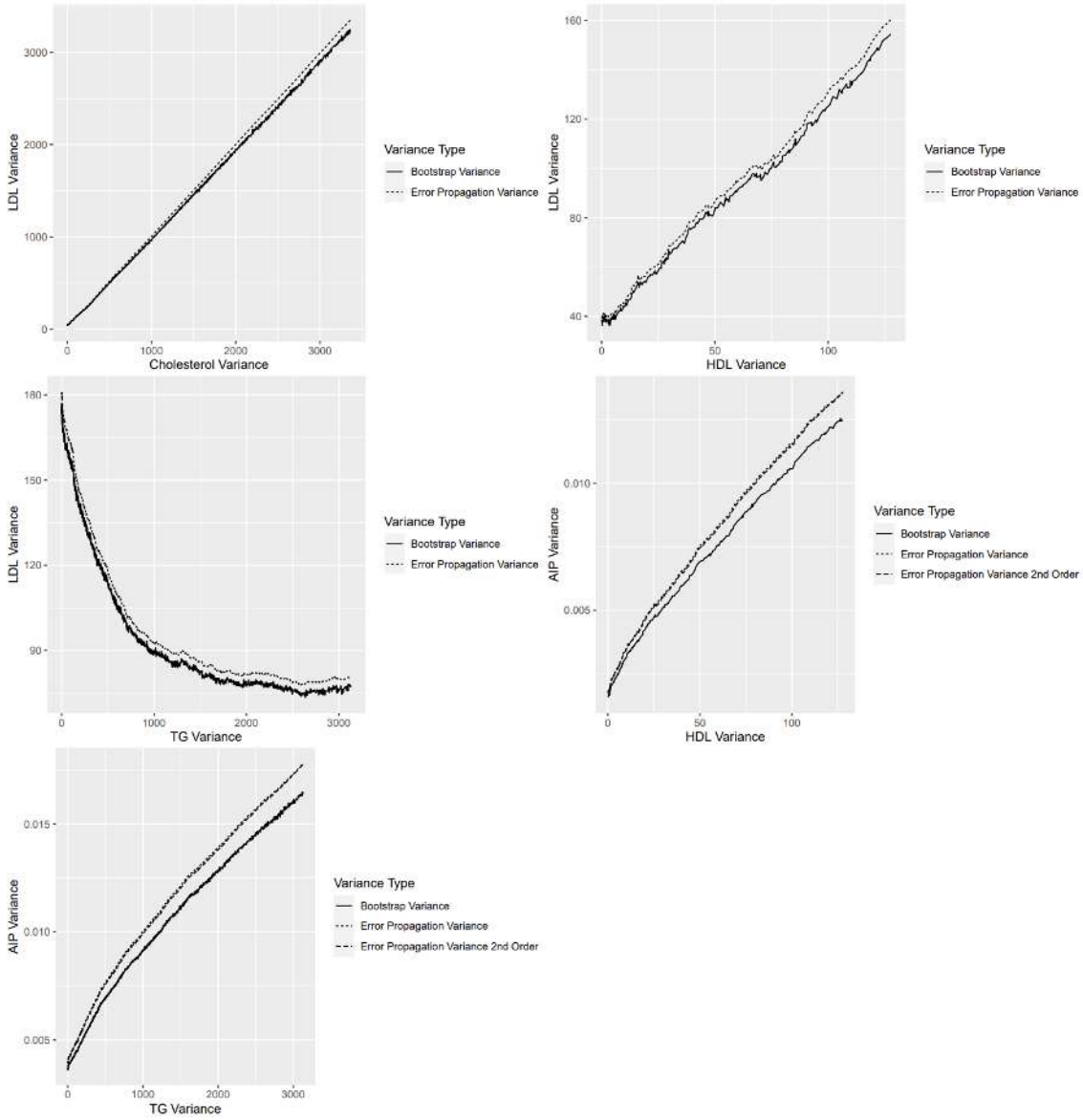


Figure 17: Variance of LDL (error propagation and boostrap for 2000 iterations), when the CHOL variance changes

6.3 Put all variance plots together, so as to compare them more easily.



It is evident that there is a large variation of the behavior of the LDL and AIP variances as the variance of either CHOL, HDL or TG changes. Some curves are linear and others are not. There are however, three common themes in all plots. The bootstrap variance is lower than the error propagation variance, for AIP the 2nd order error propagation variance is very near the first order one and all variances change in tandem.

7. Final remarks

All measurements have an inherent uncertainty, as quantified by the variance. There is a variety of approaches to determine the measurement uncertainty [16,17].

In calculated clinical chemistry tests (and generally in measurement calculated from other measurements), we must take into account the propagation of the uncertainty of multiple parameters into the final result.

There are three main methods to deal with this fact:

1. The first is the empirical/theoretical (using error propagation) approach. We bundle the empirical and error propagation methods together, since they both use the same function to map the distribution of one or more random input variables to the distribution of a random output variable.
2. The second is using Monte-Carlo sampling.
3. The third is by bootstrapping.

All three methods have their advantages and weaknesses. The empirical/theoretical approach has a rigorous mathematical background, however certain assumptions must be met, namely

- The errors in each variable should be uncorrelated. If correlated, a joint probability distribution function for all the variables must be derived.
- The probability distribution function for the output variable should be normal or nearly normal.
- Linearisation of the mapping function should result in an adequate approximation.

The Monte Carlo simulation is more robust, but

- it also suffers from the fact that a parametric form of a probability distribution function describing the data must be used, while in fact the data may not follow any such function.
- it can be computationally expensive.

Finally, the bootstrap approach makes no assumptions about the distribution of the data or the errors, but uses instead the empirical distribution and not a parametric one. Some of its drawbacks are that it

- can fail when the distribution does not have finite moments (not an issue in clinical chemistry),
- is not well suited to small sample sizes,
- can fail when large samples relative to the population size are used and
- is computationally expensive.

For lipid tests, we cannot assume that the parameters measured are either independent or uncorrelated, because the various lipid metabolic pathways (as well as many other biochemical pathways) are interconnected and correlated. This is the case with the data used in this vignette.

Therefore, when the error propagation approach was used, a joint probability distribution for all parameters was used to derive the variance. This is essentially the case in equation (6).

Another assumption in the error propagation approach is that the input variables should be normal. In Figure 1, the distribution of the input variables CHOL, HDL and TG are shown and it is evident that they are not normal. Also, neither the distribution of the output variable (LDL and AIP) is normal. In Figure 2, it is evident that the distribution of LDL is not normal. For the reasons stated before, the error propagation approach should not be used when the above assumptions are not met.

In Supplement 1 to the GUM an alternative is described, which does not use error propagation theory, but a Monte Carlo approach. This approach does not assume a normal or t- distribution [14]. It suffers, however, from the requirement that a probability distribution function must be assigned to the data, using either a Bayesian approach [15] or the principle of maximum entropy [16]. Although this is better than the error propagation method (when its assumptions are not met), it must also approximate the empirical distribution with a parametric one. This can be problematic when the empirical distribution cannot fit a parametric one.

The bootstrap approach used in this vignette, does not make any assumptions about the empirical distribution nor does it assign a distribution function to it.

For LDL and AIP the empirical/theoretical approach seems to overestimate the uncertainty. This is less pronounced for LDL, with a linear function and more pronounced for AIP with a non-linear function. The use of second order Taylor approximation for AIP does not improve things significantly, but only increases the complexity disproportionately to the improvement it confers. The use of higher order Taylor approximation would result in formulas too complex to be useful.

A caveat: Everything described up to here (use of the GUM, partial derivatives, bootstrapping and Taylor series), is trivial for statisticians. However, it may be too much for many clinical chemistry practitioners, no matter how well-intentioned they are. In the majority of clinical chemistry tests, their use is not necessary. Most of them can be performed without loss of accuracy by variance component analysis and simple addition of variances without partial derivatives and Taylor expansion. The more complicated approach presented in this paper is required only when functions based on a number of different parameters are performed including the LDL and especially the AIP. However, in these cases it can be very useful.

We can therefore conclude that for calculated tests it would be more appropriate to use the bootstrap approach, especially in non-linear functions like AIP, since the error propagation approach would overestimate the measurement uncertainty. In such highly non-linear functions even if the input variable is normal (which may very well not be) the output variable will not be normal and therefore the assumptions made for empirical calculations do not hold. Another reason to use bootstrapping is that we can get not only some numbers describing a distribution (e.g., mean and standard deviation), but the entire distribution of values, which is useful when the distribution is not a parametric one, as is the case here. The proper way to describe a distribution that does not have a closed form is to give the entire distribution.

References

[1] Joint Commitee for Guides in Metrology. Evaluation of measurement data - Guide to the expression of uncertainty in measurement. 2008.

[2] Joint Commitee for Guides in Metrology. International vocabulary of metrology (VIM) - basic and general concepts and associated terms. 2012.

[3] Hughes D, Doery JC, Weng KC, Flatman R. Calculated chemistry parameters – do they need to be harmonised? Clin Biochem Rev 2016;37:131–4.

[4] Friedewald W, Levy R, Fredrickson D. Estimation of the concentration of low-density lipoprotein cholesterol in plasma, without use of the preparative ultracentrifuge. Clin Chem 1972;18:499–502.

[5] Doob J. The limiting distributions of certain statistics. Annals Math Stat 1935;6:160–9.

[6] Fuglede B, Topsoe F. Jensen-shannon divergence and hilbert space embedding 2004.

[7] Kullback S. Information theory and statistics. John Wiley; Sons; 1959.

[8] Osterreicher F, Vajda I. A new class of metric divergences on probability spaces and its applicability in statistics. Annals of the Institute of Statistical Mathematics 2003;55:639–53.

[9] Endres DM, Schindelin JE. A new metric for probability distributions. IEEE Transactions of Information Theory 2003;49:1858–60.

[10] Bienayme I. Considerations al appui de la decouverte de laplace. Comptes Rendus de l Academie Des Sciences 1853;37:309–24.

[11] Chebyshev P. Des valeurs moyennes. Journal de Mathematiques Pures Et Appliquees 1867;2:177–84.

[12] Arras KO. An Introduction To Error Propagation: Derivation, Meaning and Examples of Equation $C_Y = F_X C_X F_X^T$. Autonomous Systems Lab, Institute of Robotic Systems, Swiss Federal Institute of Technology Lausanne (EPFL); 1998.

[13] Casella G, Berger RL. Statistical inference. 2nd ed. Duxbury Thomson Learning; 2002.

[14] Joint Commitee for Guides in Metrology. Evaluation of measurement data - Supplement 1 to the "Guide to the expression of uncertainty in measurement" - Propagation of distributions using a Monte Carlo method. 2008.

[15] Gelman A, Carlin JB, Stern HS, Rubin DB. Bayesian data analysis. Chapman; Hall, London; 2004.

[16] Woeger W. Probability assignment to systematic deviations by the principle of maximum entropy. IEEE Trans Instr Measurement 1987;36:655–8.