SWIM: Scenario Weights for Importance Measurement

Silvana M. Pesenti[*,2], *Alberto Bettini*, *Pietro Millossovich*[3,4], *Andreas Tsanakas*[4]

[2]*University of Toronto*, [3]*DEAMS, University of Trieste Cass Business School*, [4]*Cass Business School, City, University of London*

Contents

1 Introduction

1.1 Abstract

The SWIM package is an efficient sensitivity analysis tool for stochastic models developed in Pesenti et al. (2019). It provides a stressed version of a stochastic

*silvana.pesenti@utoronto.ca

model, subject to model components (random variables) fulfilling given probabilistic constraints (stresses). Possible constraints include stressing moments, propability intervals and risk measures such as the Value-at-Risk and the Expected Shortfall. Provided with simulated scenarios from a stochastic model, the SWIM package returns scenario weights under which the stochastic model satisfies the stress and minimises the relative entropy with respect to the baseline model.

1.2 Background

a short literature review

1.3 Concepts of the `SWIM` package

1.3.1 Installation

The SWIM package can be install from CRAN :

https://CRAN.R-project.org/package=SWIM;

alternatively from GitHub in R studio:

```
install.packages("spesenti/SWIM")
```

1.4 Structure of the vignette

Section 3 contains the mathematical background and the description of the optimisation that underlies the implementation of the SWIM package. For readers interested in the application and usage of the SWIM package, Section 3 can serve as a reference, as all implemented `R` functions, including stresses and graphical and analysis tools are described in detail.

2 What is SWIM?

2.1 Sensitivity testing and scenario weights

The purpose of SWIM is to enable sensitivity analysis of models implemented in a Monte Carlo simulation framework, by distorting ('stressing') some of the models' components and monitoring the resulting impact on quantities of interest.

To clarify this idea and explain how SWIM works, we first define the terms used. By a *model*, we mean a set of n (typically simulated) realisations from a vector of random variables $(X_1, \dots, X_d)$, along with *scenario weights* W assigned to individual realisations, as shown in the table below. Hence each of of the columns 1 to d corresponds to a random variable, called a *model component*, while each row corresponds to a *scenario*, that is, a state of the world.

X_1	X_2	$\dots$	X_d	W
x_{11}	x_{21}	$\dots$	x_{d1}	w_1
x_{12}	x_{22}	$\dots$	x_{d2}	w_2
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
x_{1n}	x_{2n}	$\dots$	x_{dn}	w_n

Each scenario has a *scenario weight*, shown in the last column, such that, scenario i has probability $\frac{w_i}{n}$ of occurring. Scenario weights are always greater and equal than zero and have an average of 1. When all scenario weights are equal to 1, such that the probability of each scenario is $\frac{1}{n}$ (the standard Monte Carlo framework), we call the model a *baseline model* – and consequently never explicitly talk about the scenario weights of baseline models. When scenario weights are not identically equal to 1, we say that we have a *stressed model*.

The scenario weights make the joint distribution of model components under the stressed model different, compared to the baseline model. For example, under the baseline model, the expected value of X_1 and the cumulative distribution function of X_1 at threshold t, are respectively given by:

$$E(X_1) = \frac{1}{n}\sum_{i=1}^{n} x_{1i}, \quad F_{X_1}(t) = P(X_1 \leq t) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{x_{1i} \leq t},$$

where $\mathbf{1}_{x_{1i} \leq t} = 1$ if $x_{1i} \leq t$ and 0 otherwise. For a stressed model with scenario weights W, the expected value and distribution function become:

$$E^W(X_1) = \frac{1}{n}\sum_{i=1}^{n} w_i x_{1i}, \quad F^W_{X_1}(t) = P^W(X_1 \leq t) = \frac{1}{n}\sum_{i=1}^{n} w_i \mathbf{1}_{x_{1i} \leq t}.$$

Similar expressions can be derived for more involved quantities, such as higher (joint) moments and quantiles.

The logic of stressing a model with SWIM then proceeds as follows. An analyst or modeller is supplied with a baseline model, in the form of a matrix of equiprobable simulated scenarios of model components. The modeller wants to investigate the impact of a change in the distribution of, say, X_1. To this effect, she chooses a set of scenario weights, such that the stressed distribution of X_1 satisfies a particular constraint, e.g. $E^W(X_1) = m$, which we call a *stress*; we then say that she is *stressing* X_1 and, by extension, the model. The scenario weights are chosen such that the distortion to the baseline model induced by the stress is as small as possible; specifically in SWIM the Kullback-Leibler divergence (or relative entropy) between the baseline and stressed models is minimised, subject to the constraint of the stress (see Section 3.1 for more detail on the different types of possible stresses and the corresponding optimisation problems).

Once scenario weights are obtained, they can be used to obtain the stressed distribution of any model component or any function of model components.

For example, for scenario weights W obtained through a stress on X_1, we may calculate

$$E^W(X_2) = \frac{1}{n}\sum_{i=1}^{n} w_i x_{2i}, \quad E^W(X_1^2 + X_2^2) = \frac{1}{n}\sum_{i=1}^{n} w_i \left(x_{1i}^2 + x_{2i}^2\right).$$

Through this process the modeller can monitor the impact of the stress on X_1 on any other random variable of interest. It is notable that this approach does not necessitate generating new simulations from a stochastic model. However, as the SWIM approach requires a single set of simulated scenarios (the baseline model) it offers a clear computational benefit.

2.2 An introductory example

Here, through an example, we illustrate the basic concepts and usage of SWIM for sensitivity analysis. More advanced usage of SWIM and options for constructing stresses are demonstrated in Sections xxx.

We consider a simple model, with the random variables Z_1, Z_2, Z_3 represent normally distributed losses in a portfolio. Z_1 and Z_2 are correlated, while Z_3 is independent of (Z_1, Z_2). The portfolio loss is defined by $Y = Z_1 + Z_2 + Z_3$. Our purpose in this example is to investigate how a stress on the loss Z_1, impacts on the overall portfolio loss Y.

First we derive simulated data from the random vector (Z_1, Z_2, Z_3, Y), forming our baseline model.

```r
set.seed(0)
# number of simulated scenarios
n.sim <- 10^5
# correlation between Z1 and Z2
r <- 0.5
# simulation of Z1  and Z2
# simple construction as combination of independent standard normals U1, U2
U1 <- rnorm(n.sim)
U2 <- rnorm(n.sim)
Z1 <- 100 + 40 * U1
Z2 <- 100 + 20 * (r * U1 + sqrt(1 - r^2) * U2)
# simulation of Z3
Z3 <- rnorm(n.sim, 100, 20)
# portfolio loss Y
Y <- Z1 + Z2 + Z3
```

Now we introduce a stress to our baseline model. For our first stress, we require that the mean of Z_1 is increased from 100 to 110. This is done using the `stress` function, which generates as output the SWIM object `str.mean`. This object stores the stressed model, i.e. the realisations of the model components and the scenario weights. In the function call, `k=1` indicates that the stress is applied on the first column of `dat`, that is, on the realisations of the random variable Z_1.

```r
library(SWIM)
dat <- data.frame(Z1, Z2, Z3, Y)
str.mean <- stress(type="mean", x = dat, k=1, new_means = 110)
summary(str.mean, base = TRUE)
```

```
## $base
##                   Z1        Z2        Z3         Y
## mean         1.00e+02  99.94040  99.98433 299.98111
## sd           4.00e+01  19.99695  19.98195  56.63887
## skewness    -6.08e-04   0.00117  -0.00247  -0.00234
## ex kurtosis -1.06e-02  -0.00897  -0.01257  -0.00938
## 1st Qu.      7.28e+01  86.47450  86.48161 261.61212
## Median       1.00e+02  99.98657 100.00907 300.05480
## 3rd Qu.      1.27e+02 113.39567 113.49342 338.26698
##
## $`stress 1`
##                   Z1        Z2        Z3         Y
## mean        110.00000 102.44371  99.98278 312.42649
## sd           40.03328  19.99542  19.97616  56.61726
## skewness     -0.00240  -0.00151  -0.00493  -0.00368
## ex kurtosis  -0.00502  -0.00316  -0.01547  -0.00119
## 1st Qu.      82.99837  88.97706  86.48152 274.22003
## Median      110.07585 102.48100  99.99544 312.50394
## 3rd Qu.     136.93102 115.87438 113.50186 350.61205
```

The summary function, applied on the SWIM object `str.mean`, shows how the distributional characteristics of all random variables of interest change from the baseline to the stressed model. In particular, we see that the mean of Z_1 changes to its required value, while the mean of Y also increases. Furthermore there is a small impact on Z_2, due to its positive correlation to Z_1.

Beyond considering the standard statistics evaluated via the summary function, stressed probability distributions can be plotted. In Figure 1 we show the impact of the stress on on the cumulative distribution functions (cdf) of Z_1 and Y. It is seen how the stressed cdfs are lower than the original (baseline) ones. Loosely speaking, this demonstrates that the stress has increased (in a stochastic sense) both random variables Z_1 and Y. While the stress was on Z_1, the impact on the distribution of the portfolio Y is clearly visible.

```r
# can refer to variable of interest by name...
plot_cdf(str.mean, xCol = "Z1", base = TRUE)
# ... or column number
plot_cdf(str.mean, xCol = 4, base = TRUE)
```

The scenario weights, given their central role, can be extracted from a SWIM object. In Figure 2, the scenario weights from `str.mean` are plotted against realisations from Z_1 and Y respectively. It is seen how the weights are increasing

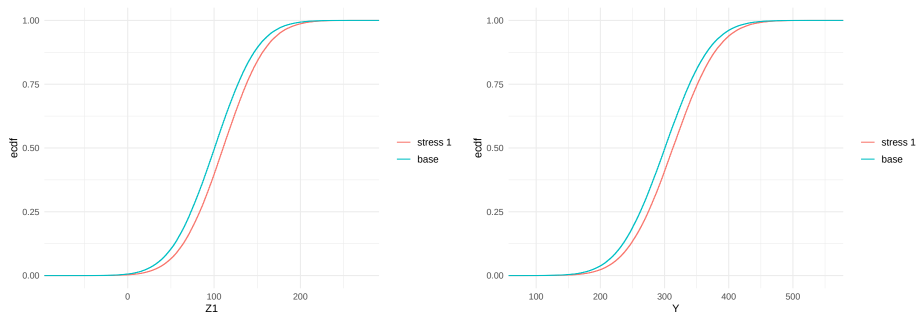


Figure 1: Baseline and stressed empirical distribution functions of model components Z_1 (left) and Y (right), subject to a stress on the mean of Z_1.

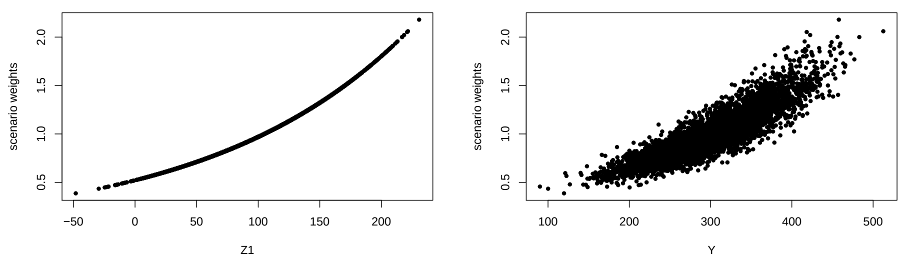


Figure 2: Scenario weights against observations of model components Z_1 (left) and Y (right), subject to a stress on the mean of Z_1.

in the realisations from Z_1. This is a consequence of the weights' derivation via a stress on the model component Z_1. The increasingness shows that those scenarios for which Z_1 is largest are assigned a higher weight. The relation between scenario weights and Y is still increasing (reflecting that high outcomes of Y tend to receive higher weights), but no longer deterministic (showing that Y is not completely driven by changes in Z_1).

```
# extract weights from SWIM object
w.mean <- get_weights(str.mean)
plot(Z1[1 : 5000], w.mean[1 : 5000], pch = 20, xlab = "Z1", ylab = "scenario weights")
plot(Y[1 : 5000], w.mean[1 : 5000], pch = 20, xlab = "Y", ylab = "scenario weights")
```

Stress the mean of Z_1 did not impact the volatility of either Z_1 or Y, as can be seen by the practically unchanged standard deviations in the output of `summary(str.mean)`. Thus, we introduce an alternative stress that keeps the mean of Z_1 fixed at 100, but increases its standard deviation from 40 to 50. This new stress is seen to impact the standard deviation of the portfolio loss Y.

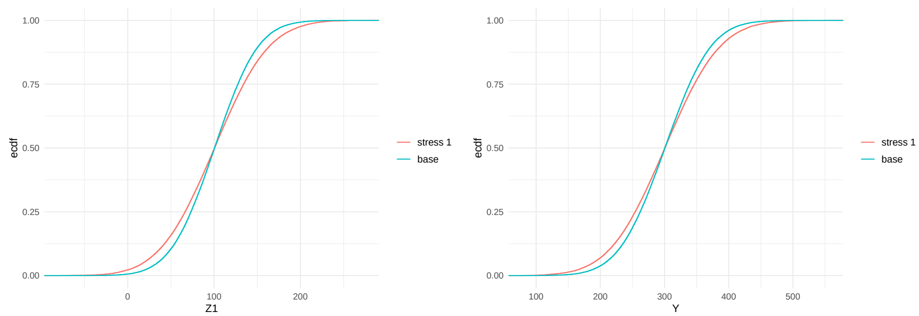


Figure 3: Baseline and stressed empirical distribution functions of model components Z_1 (left) and Y (right), subject to a stress on the standard deviation of Z_1.

```
str.sd <- stress(type="mean sd", x = dat, k=1, new_means = 100, new_sd=50)
summary(str.sd, base = FALSE)
```

```
## $`stress 1`
##                    Z1        Z2        Z3         Y
## mean        100.00000  99.94055  99.97817 299.91872
## sd           50.00050  21.34937  19.97997  67.92330
## skewness     -0.00272   0.00703  -0.00342   0.00491
## ex kurtosis  -0.05561  -0.03317  -0.00612  -0.04270
## 1st Qu.      66.09643  85.49520  86.48219 253.74962
## Median      100.12904  99.97427 100.04553 299.97662
## 3rd Qu.     133.77335 114.30108 113.47008 345.91590
```

Furthermore, in Figure 3, we compare the baseline and stressed cdfs of Z_1 and Y, under the new stress on Z_1. The crossing of probability distribution reflects the increase in volatility.

```
plot_cdf(str.sd, xCol = "Z1", base = TRUE)
plot_cdf(str.sd, xCol = 4, base = TRUE)
```

The different ways how a stress on the standard deviation of Z_1, compared to a stress on its mean, impact on the model, is reflected by the scenario weights. Figure 4 shows the pattern of the scenario weights and how, when stressing standard deviations, higher weight is placed on scenarios where Z_1 is extreme, either much lower or much higher than its mean of 100.

```
w.sd <- get_weights(str.sd)
plot(Z1[1:5000],w.sd[1:5000],pch=20,xlab="Z1",ylab="scenario weights")
plot(Y[1:5000],w.sd[1:5000],pch=20,xlab="Y",ylab="scenario weights")
```

Finally we ought to note that not all stresses that one may wish to apply are feasible. Assume for example that we want to increase the mean of Z_1 from

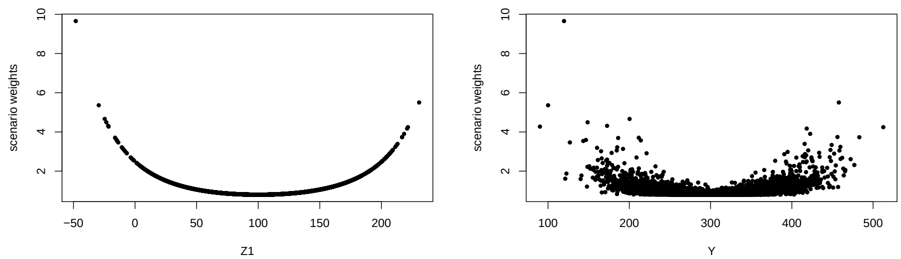


Figure 4: Scenario weights against observations of model components Z_1 (left) and Y (right), subject to a stress on the standard deviation of Z_1.

100 to 300, which exceeds the maximum realisation of Z_1 in the baseline model. Then, clearly, no set of scenario weights can be found that produce a stress that yields the required mean for Z_1; consequently an error message is produced.

```
str.sd <- stress(type="mean",x = dat, k=1, new_means = 300)
```

```
## Error in stress_moment(x = x, f = means, k = k, m = new_means, ...): Values in m must be
```

```
max(Z1)
```

```
## [1] 273
```

3 Scope of the SWIM package

3.1 Stressing a model

While the SWIM package is designed to work on (Monte Carlo) realisations of model components, the scenario weights are derived in a general probabilistic framework. A baseline probability (representing the equiprobable Monte Carlo simulations) can be described by a probability measure P, and a stressed model by a different probabitlity measures Q. The stressed model is uniquely described by the change from the baseline to the stressed model, which can be seen as the scenario weights $W = \frac{dQ}{dP}$. A stressed model, under which the distribution of the model components fulfil specific stresses, is chosen such that the distortion to the baseline model is as small as possible in the Kullback-Leibler divergence (or relative entropy). Mathematically, a stressed model is the solutions to.

$$\min_{W} \; E(W \log(W)), \quad \text{subject to constraints under } P^W. \tag{1}$$

Subsequently, we denote by a superscript W the quantity of interest under the stressed model, such as P^W, E^W for the probabtility distribution and expectation under the stressed model, respectivley. We refer to Pesenti et al.

(2019) and references therein for further mathematical details and the derivations of solutions to (1).

The subsequent table is a collection of all implemented types of stresses. The pricise constraints of (1) are explained below.

R function	Stress	type	Reference
`stress()`	wrapper for the `stress_type` functions		Section 3.1.1
`stress_user()`	user defined scenario weights	`user`	
`stress_prob()`	disjoint intervals	`prob`	(2)
`stress_mean()`	means	`mean`	(3)
`stress_mean_sd()`	means and standard deviations	`mean` `sd`	(3)
`stress_moment()`	moments, functions of moments	`moment`	(3)
`stress_VaR()`	VaR risk measure, a quantile	`VaR`	(4)
`stress_VaR_ES()`	VaR and ES risk measures	`VaR` `ES`	(5)

3.1.1 The `stress` function and the `SWIM` object

The `stress()` function is a wapper for the `stress_` functions, with `stress(type = "type", ... )` and `stress_type(...)` being equivalent. The `stress()` function solves optimisation (1) for constraints specified through `type`and returns a `SWIM` object containing a list of:

`x`	realistaions of the model
`new_weights`	scenario weights
`type`	"type" of stress
`specs`	details about the stress

The data frame, `x` in the above table, containing the realisations of the baseline model, can be extracted from a `SWIM` object using `get_data()`. Similarly, `get_weights()` and `get_weightsfun()` provide the scenario weights, respectively the functions, that when applied to `x` generate the scenarion weights. The specification of the applied stress can be obtained using `get_specs()`.

3.1.2 Stressing disjoint probability intervals

Stressing disjoint probability intervals, allows to define stresses by altering regions or events of a model component. The scenario weights are calculated via `stress_prob()`, or equivalently `stress(type = "prob", ...)`, and the stressed probability intervals are specified through the `lower` and `upper` endpoints of the intervals.

For disjoint intervals $B_1, \dots, B_I$ with $P(X \in B_i) > 0$, for all $i = 1, \dots, I$, and $\alpha_1, \dots, \alpha_I > 0$ such that $\alpha_1 + \ldots + \alpha_I < 1$, `stress_prob()` solves for the constraints

$$P^W(X \in B_i) = \alpha_i, \ i = 1, \dots, I. \tag{2}$$

3.1.3 Stressing moments

The functions `stress_mean()`, `stress_mean_sd()` and `stress_moment()` can be applied to multiple model components and are the only `stress` functions that have scenrio weights calculated via numerical optimisation using the nleqslv package. Thus, dependending on the choice of moment stresses, existence of a stressed model is not guaranteed.

For $i = 1, \dots, I$ with $J_i \subset \{1, \dots, n\}$ and functions $f_i \colon \mathbb{R}^{J_i} \to \mathbb{R}$, `stress_moment()` solves for the constraints

$$E^W(f_i(X_{J_i})) = m_i, \ i = 1, \dots, I. \tag{3}$$

3.1.4 Stressing risk measures

The functions `stress_VaR` and `stress_VaR_ES` provides stressed models, under which a model components fulfils a stress on the Value-at-Risk (VaR) and/or Expected Shortfall (ES) risk measures. The VaR at level $0 < \alpha < 1$ of a random variable Z with distrbution F, is defined as the $\alpha-$quantile of F, that is

$$\mathrm{VaR}_\alpha(Z) = F^{-1}(\alpha).$$

The ES at level $0 < \alpha < 1$ of a random variable Z is given by

$$\mathrm{ES}_\alpha(Z) = \int_0^1 \mathrm{VaR}_u(Z)\mathrm{d}u.$$

For $0 < \alpha < 1$ and $q, s \in \mathbb{R}$ such that $\mathrm{VaR}_\alpha(Y) < q < s$, `stress_VaR()` solves for the constraints

$$\mathrm{VaR}_\alpha^W(Y) = q; \tag{4}$$

and `stress_VaR_ES()` solves for the constraints

$$\mathrm{VaR}_\alpha^W(Y) = q, \ \mathrm{ES}_\alpha^W(Y) = s. \tag{5}$$

3.2 Analysis of stressed models

The function `summary()` is a methods for an object of class SWIM and provides summary statistics of the baseline and stressed models. If the SWIM object contains more than one set of scenario weights, each corresponding to one stressed model, the `stress()` function returns for each set of scenarion weights a list containting:

`mean`	sample mean
`sd`	sample standard deviation
`skewness`	sample skewness
`ex kurtosis`	sample excess kurtosis
`1st Qu.`	25% quantile
`Median`	median, 50% quantile
`3rd Qu.`	75% quantile

The empirical distribution functions of model components under a stressed model can be calculated by evaluation of `cdf()` on a SWIM object. It is important to note, that the standard empirical distribution function, `ecdf()` applied to a SWIM object will **not** return empirical distribution functions under a stressed model. Similarly, to calculate sample quantiles of stressed models components, the function `quantile_stressed()` should be used. Implemented visualisation of distribution functions are `plot_cdf()`, for plotting empirical distribution functions, and `plot_hist()`, for plotting histograms of model components under stressed models.

Comparison of baseline and stressed models and how stressed model impact model components, can be done via the `sensitivity()` function. The implemented sensitivity measures are summarised in the table below. The Wasserstein and Kolmogorov, sensitivities are to compare stressed (and baseline) models, as these sensitivities only depend on the scenario weights, whereas the Gamma sensitivity is useful to compare the impact of a stress model on the model components.

Wasserstein	$\int \lvert F_X^W(x) - F_X(x)\rvert dx$	comparing models
Kolmogorov	$\sup_x \lvert F_X^W(x) - F_X(x)\rvert$	comparing models
Gamma	$\frac{E^W(X)-E(X)}{c}$, for a normalisation c	comparing model components

The normalisation for the Gamma sensitivity is such that Gamma takes values between -1 and 1, where positive values correspond to a larger impact on a larger impact. The sensitivities of model components can be plotted using `plot_sensitivity()`. The function `importance_rank()`, returns the effective rank of model component according ot the chosen sensitivity measures.

4 Simulation study

4.1 The credit risk portfolio

The credit model in this section is a conditionally binomial credit model and we refer to the Appendix A for details and the generation of the simulated data. Of interest ist the total aggregate portfolio loss $L = L_1 + L_2 + L_3$, where

L_1, L_2, L_3 are homogeneous subportfolios on comparable scale. The data set contains 100,000 simulations of the portfolio L, the sub-portfolios L_1, L_2, L_3 as well as the (conditional) default probability of each subportfolio H_1, H_2, H_3. A snipped of data set looks as follows:

```
##          L L1    L2  L3       H1      H2     H3
## [1,]   692  0 346.9 345 1.24e-04 0.00780 0.0294
## [2,] 1006 60 515.6 430 1.16e-03 0.01085 0.0316
## [3,] 1661  0 806.2 855 5.24e-04 0.01490 0.0662
## [4,] 1708  0 937.5 770 2.58e-04 0.02063 0.0646
## [5,]  807  0  46.9 760 8.06e-05 0.00128 0.0632
## [6,] 1159 20 393.8 745 2.73e-04 0.00934 0.0721
```

4.2 Stressing the aggregate portfolio loss

In this section, we study the effect of stresses on (the tail of) the aggregate portfolio on the three sub-portfolios. First, we stress the $VaR_{0.9}$ of the total loss of the aggregate portfolio by 20%. For this we use the `stress` function with the argument `type = "VaR"`. The input parameter `x` is the simulated data, `k` corresponds to the name of the row of `x` on which the stress is applied to, `alpha` determines the level of the stresses VaR and `q_ratio` the percentage increase.

```
stress.credit <- stress(type = "VaR", x = credit_data, k = "L",
                        alpha = 0.9, q_ratio = 1.2)
```

Second, we consider, additionally to the 20% increase in $\mathrm{VaR}_{0.9}$, a 30% increase in $\mathrm{ES}_{0.9}$ of the aggregate portfolio L. Generating a stressed model, resulting from a simultaneous stress on the VaR and the ES can be acchieve using `type = "VaR ES"`. Note that both VaR and ES need be stressed at the same level `alpha = 0.9`. The additonal input parameter `s_ratio` determines the percentage increase in the ES. Instead of providing the percentage inceases in the VaR and ES, the `stress` function allows for the actual stressed values of the VaR and ES using the parameters `s` and `q` instead of `s_ratio` and `q_ratio`, respectively.

```
stress.credit <- stress(type = "VaR ES", x = stress.credit, k = "L",
                        alpha = 0.9, q_ratio = 1.2, s_ratio = 1.3)
```

Note, that as input `x` we used the above calucalted stressed model, resulting from a stress on the $VaR_{0.9}$. Using a stressed model as an input for the `stress` function is convenient for large data sets, as the `stress` function returns an object (`stress.credit`) that countains the original simualted data and the scenario weights.

++++MAYBE CHANGE THE SECOND STRESS ACCORDING TO ANDREAS' SUGGESTION? IE LEAVE VAR UNCHANGED (CURRENTLY NOT POSSIBLE) AND STRESS ES ONLY?

4.3 Analysing the stressed model

The `summary` function provides a statistical summary of the stressed models. Choosing `base = TRUE`, compares the stressed models with the the simulated data - the baseline model.

```
summary(stress.credit, base = TRUE)
```

```
## $base
##                     L    L1     L2      L3       H1      H2     H3
## mean         1102.914 19.96 454.04 628.912 0.000401 0.00968 0.0503
## sd            526.538 28.19 310.99 319.715 0.000400 0.00649 0.0252
## skewness        0.942  2.10   1.31   0.945 1.969539 1.30834 0.9501
## ex kurtosis     1.326  6.21   2.52   1.256 5.615908 2.49792 1.2708
## 1st Qu.       718.750  0.00 225.00 395.000 0.000115 0.00490 0.0318
## Median       1020.625  0.00 384.38 580.000 0.000279 0.00829 0.0464
## 3rd Qu.      1398.750 20.00 609.38 810.000 0.000555 0.01296 0.0643
##
## $`stress 1`
##                   L    L1     L2     L3       H1      H2     H3
## mean        1193.39 20.83 501.10 671.46 0.000417 0.01066 0.0536
## sd           623.48 29.09 363.57 361.21 0.000415 0.00756 0.0285
## skewness       1.01  2.09   1.36   1.02 1.973337 1.35075 1.0283
## ex kurtosis    0.94  6.14   2.23   1.22 5.630153 2.23353 1.2382
## 1st Qu.      739.38  0.00 234.38 405.00 0.000120 0.00512 0.0328
## Median      1065.62 20.00 412.50 605.00 0.000290 0.00878 0.0483
## 3rd Qu.     1505.62 40.00 675.00 865.00 0.000578 0.01422 0.0688
##
## $`stress 2`
##                   L    L1     L2     L3       H1      H2     H3
## mean        1224.76 21.13 519.17 684.46 0.000423 0.01102 0.0547
## sd           707.59 29.61 410.43 390.67 0.000427 0.00851 0.0308
## skewness       1.48  2.13   1.77   1.28 2.034985 1.76908 1.2802
## ex kurtosis    2.69  6.49   4.18   2.15 6.009169 4.26790 2.1077
## 1st Qu.      739.38  0.00 234.38 405.00 0.000121 0.00512 0.0328
## Median      1065.62 20.00 412.50 605.00 0.000293 0.00878 0.0484
## 3rd Qu.     1505.62 40.00 675.00 870.00 0.000584 0.01430 0.0692
```

The information on individual stresses can be recovered through the `get_specs` function and the actual scenario weitghts using `get_weight`.

```
get_specs(stress.credit)
```

```
##              type k alpha       q            s
## stress 1      VaR L   0.9 2174.25         <NA>
## stress 2   VaR ES L   0.9 2174.25 2848.5562625
```

```
w <- get_weights(stress.credit)
colMeans(w)
```

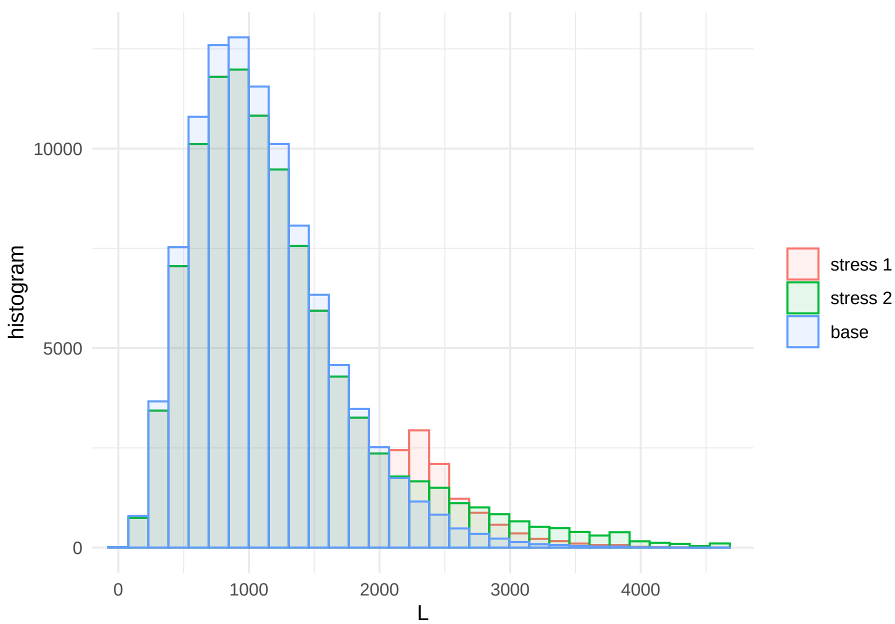
```
## stress 1 stress 2
##        1        1
```

+++HERE WE COULD USE THE "expected shortfall" FUNCTION (NOT AVAILABLE YET) TO CALCULATE THE EXPECTED SHORTFALL OF "stress.credit$stress1"

4.4 Visual comparison

The change of the distributions of the portfolio and subportfolios from the baseline to the stressed models can be visualised through `plot_hist` and `plot_cdf`. The following plot displays the empirical histogramm of the aggregate portfolio loss under the baseline and the two stressed models.

```
plot_hist(object = stress.credit, xCol = "L", base = TRUE)
```

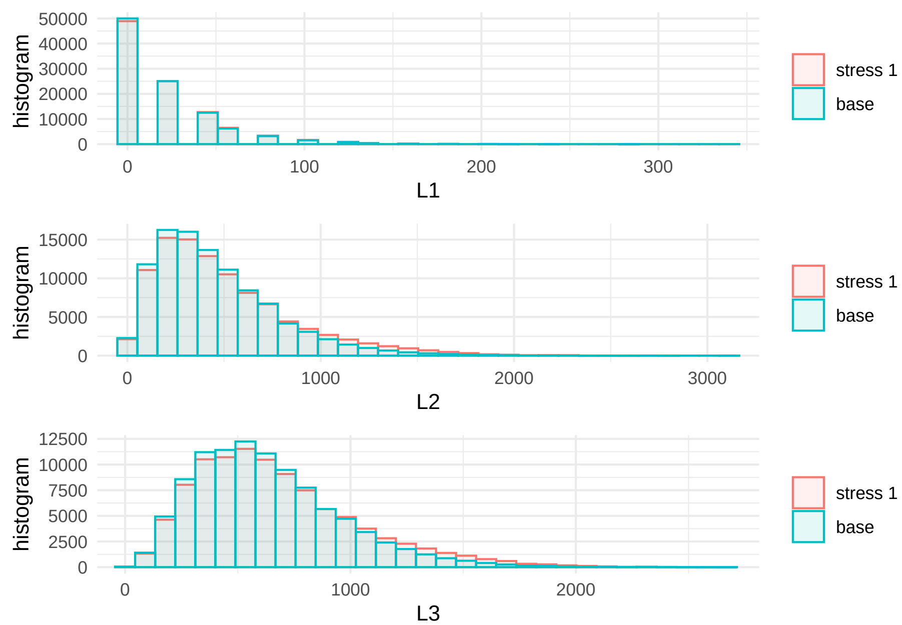


Both functions,`plot_hist` and `plot_cdf`, include the parameters `xCol` specifying the columns of the data and `wCol` determining the columns of the scenario weights. Thus, allowing to plot the impact of the stressed models on the subportfoios. The graphical functions `plot_hist` and `plot_cdf` functions return objects compatible with the package **ggplot2**. Thus, we can compare the histograms of the portfolio losses via the function `grid.arrange` (of the package **gridExtra**).

```r
library(gridExtra)
pL1 <- plot_hist(object = stress.credit, xCol = 2, wCol = 1, base = TRUE)
pL2 <- plot_hist(object = stress.credit, xCol = 3, wCol = 1, base = TRUE)
pL3 <- plot_hist(object = stress.credit, xCol = 4, wCol = 1, base = TRUE)
class(pL1)
```

```
## [1] "gg"     "ggplot"
```

```r
grid.arrange(pL1, pL2, pL3, ncol = 1, nrow = 3)
```



From the plots we observe, that the subportfolios L_2 and L_3 are significanlty affected by the stress, while the distribution of L_1 is almost unchanged.

4.5 Sensitivity measures

The impact of the stressed models on the model components can be quantified through sensitivity measures. The function `sensitivity` includes the *Kolmogorov*, the *Wasserstein* distance and the sensitivity measure *Gamma*, which can be specified through the optional parameter `type`. We refer to Section 3.2 for the definition. The Kolmogorov and the Wasserstein distance are useful to compare different stressed models, whereas the sensitivity measure Gamma ranks model components for one stressed model.

```r
sensitivity(object = stress.credit, xCol = c(2 : 7), wCol = 1, type = "Gamma")
```

```
##    stress  type   L1    L2    L3    H1    H2    H3
```

```
## 1 stress 1 Gamma 0.15 0.819 0.772 0.196 0.811 0.767
```

Using the `sensitivity` function we can analyse whether the first and third tranches are able to exceed the riskiness of the second. This can be accomplished specifying, through the option `f`, a list of functions applicable to the columns `k` of the dataset. Finally, setting `xCol = NULL` allows to consider only the transformed data:

```
sensitivity(object = stress.credit,type = "Gamma", f = list(function(x)x[1] + x[2]),
            k = list(c(2,4)), xCol = NULL, wCol = 1)
```

```
##     stress  type    f1
## 1 stress 1 Gamma 0.783
```

The `importance_rank` function, having the same structure as the `sensitivity` function, return the ranks of the sensitivity measures. This function is particularly useful when there are several risk factors involved.

```
importance_rank(object = stress.credit, xCol = c(2 : 7), wCol = 1, type = "Gamma")
```

```
##     stress  type L1 L2 L3 H1 H2 H3
## 1 stress 1 Gamma  6  1  3  5  2  4
```

It transpires that subportfolios 2 and 3 are, in this order, most responsible for the stress in the global loss. Also, most of the sensitivity seems to be due to the systematic risk components H_2 and H_3. To confirm this, another stress resulting in the same $\mathrm{VaR}_{90\%}(L)$, but controlling the distribution of H_2, can be imposed using the function `stress_moment`. More precisely, we impose that $E[H_2]$ and the 75% quantile of H_2 are fixed as in the base model.

```
VaR.L <- quantile(x = credit_data[, "L"], prob = 0.9, type = 1)
q.H2 <- quantile(x = credit_data[, "H2"], prob = 0.75, type = 1)
str.var.credit2 <- stress_moment(x = credit_data,
                              f = list(function(x)1 * (x <= VaR.L * 1.2),
                                       function(x)x,
                                       function(x)1 * (x <= q.H2)),
                              m = c(0.9, mean(credit_data[, "H2"]), 0.75),
                              k = c(1, 6, 6))
# stress.credit <- stress_moment(x = stress.credit,
#                                f = list(function(x)1 * (x <= VaR.L * 1.2),
#                                         function(x)x,
#                                         function(x)1 * (x <= q.H2)),
#                                m = c(0.9, mean(credit_data[, "H2"]), 0.75), k = c(1, 6, 6
summary(str.var.credit2)
```

```
## $`stress 1`
##                    L    L1    L2     L3       H1      H2     H3
## mean        1140.535 20.06 456.0 664.47 0.000400 0.00968 0.0530
## sd           616.930 28.48 340.9 371.14 0.000405 0.00706 0.0292
```

```
## skewness        1.059  2.13   1.4    1.09 2.013196 1.39135 1.0949
## ex kurtosis     0.895  6.40   2.3    1.31 5.899634 2.26506 1.3371
## 1st Qu.       695.000  0.00 206.2 395.00 0.000113 0.00453 0.0318
## Median       1001.875  0.00 365.6 590.00 0.000276 0.00786 0.0472
## 3rd Qu.      1430.625 20.00 609.4 855.00 0.000554 0.01296 0.0679
```

```
# summary(stress.credit)
sensitivity(object = str.var.credit2, xCol = c(2 : 7), type = "Gamma")
```

```
##     stress  type     L1     L2    L3        H1       H2    H3
## 1 stress 1 Gamma 0.0102 0.0203 0.366 -0.000521 1.17e-08 0.359
```

```
# sensitivity(object = stress.credit, xCol = c(2 : 7), type = "Gamma")
```

+++THIS SHOULD BE APPENDED TO "stress.credit" WHEN "stress_moment" IS FIXED It is then clear that systematic risk prevails on binomial (event) risk.

The `stress_moment` function is flexible and allows different type of stresses to be imposed on a model. The following example forces a 50% increase in correlation between the losses in the second and third portfolios, while keeping the means und standard deviations unchanged.

```
m.L2 <- mean(credit_data[, "L2"])
m.L3 <- mean(credit_data[, "L3"])
m2.L2 <- mean(credit_data[, "L2"] ^ 2)
m2.L3 <- mean(credit_data[, "L3"] ^ 2)
cov.L2.L3 <- cov(credit_data[, "L2"], credit_data[, "L3"])
# str.var.credit2 <- stress_moment(x = credit_data,
#                                  f = list(function(x)x,
#                                           function(x)x,
#                                           function(x)x ^ 2,
#                                           function(x)x ^ 2,
#                                           function(x)x[1] * x[2] - m.L2 * m.L3),
#                                  k = list(3, 4, 3, 4, c(3, 4)),
#                                  m = c(m.L2, m.L3, m2.L2, m2.L3, cov.L2.L3 * 1.5)
```

+++CURRENTLY DOES NOT RUN - NEEDS TO BE FIXED OR REPLACED

+++FINAL COMMENTS?

A Appendix Credit Model

A.1 Credit Model assumptions

The credit model is based on the conditionally binomial credit model described in McNeil et al. (2015) which belongs to the family of mixture models. Specifically, we consider a portfolio that consists of three homogeneous sub-portfolios and denote the total aggregate loss of the portfolio by $L = L_1 + L_2 + L_2$, where

L_1, L_2, L_3 are the aggregate losses of each sub-portfolio, given by

$$L_i = e_i \cdot \mathrm{LGD}_i \cdot M_i, \quad i = 1, 2, 3, \tag{6}$$

where e_i and M_i are the exposure and number of insolvencies of the i^{th} sub-portfolio, respectively, and LGD_i is the loss given default of sub-portfolio i. The number of insolvencies of the i^{th} sub-portfolio M_i is, conditionally on a $[0, 1]$ valued random variable H_i, independent and Binomially distributed with parameters m_i, the sub-portfolio size, and common random probability H_i. $H_i,\ i = 1, 2, 3$ follows a Beta distribution with parameters chosen such as to match the default probability p_i and the default correlation ρ_i, that is the correlation between two default events within a sub-portfolio, see McNeil et al. (2015). The dependence structure of (H_1, H_2, H_3) is modelled via a Gaussian copula with correlation matrix

$$\Sigma = \begin{pmatrix} 1 & 0.3 & 0.1 \\ 0.3 & 1 & 0.4 \\ 0.1 & 0.4 & 1 \end{pmatrix}. \tag{7}$$

The subsequent table summarises the parameter values used in the simulation.

i	m_i	e_i	p_i	ρ_i	LGD_i
1	2500	80	0.0004	0.00040	0.250
2	5000	25	0.0097	0.00440	0.375
3	2500	10	0.0503	0.01328	0.500

A.2 Code for generating the data

```r
  library(SWIM)
  set.seed(1)
  library(copula)
  nsim <- 100000

# data
  m1 <- 2500 # counterparties tranche A
  m2 <- 5000 # counterparties tranche B
  m3 <- 2500 # counterparties tranche C

  p1 <- 0.0004 # prob of default
  rho1 <- 0.0004 # correlation within the tranche

  p2 <- 0.0097
  rho2 <- 0.0044
```

```r
  p3 <- 0.0503
  rho3 <- 0.01328

# exposures
  e1 <- 80
  e2 <- 25
  e3 <- 10

# loss given default
  LGD1 <- 0.25
  LGD2 <- 0.375
  LGD3 <- 0.5

# beta-binomial model with copula

# beta parameters: matching tranches default probabilities and correlation
  alpha1 <- p1 * (1 / rho1 - 1)
  beta1 <- alpha1 * (1 / p1 - 1)

  alpha2 <- p2 * (1 / rho2 - 1)
  beta2 <- alpha2 * (1 / p2 - 1)

  alpha3 <- p3 * (1 / rho3 - 1)
  beta3 <- alpha3 * (1 / p3 - 1)

# correlations between sub-portfolios
  cor12 <- 0.3
  cor13 <- 0.1
  cor23 <- 0.4

# Gaussian copula structure
  myCop <- normalCopula(param = c(cor12, cor13, cor23), dim = 3, dispstr = "un")

# define multivariate beta with given copula
  myMvd <- mvdc(copula = myCop,
                margins = c("beta", "beta", "beta"),
                paramMargins = list(list(alpha1, beta1),
                                    list(alpha2, beta2),
                                    list(alpha3, beta3)))

# simulation from the chosen copula
  H <- rMvdc(nsim, myMvd)

# simulate number of default per tranches (binomial distributions)
  M1 <- rbinom(n = nsim, size = m1, prob = H[, 1])
```

```r
  M2 <- rbinom(n = nsim, size = m2, prob = H[, 2])
  M3 <- rbinom(n = nsim, size = m3, prob = H[, 3])

# total loss per sub-portfolio
  L1 <- M1 * e1 * LGD1
  L2 <- M2 * e2 * LGD2
  L3 <- M3 * e3 * LGD3

# aggregate portfolio loss
  L <- L1 + L2 + L3

# DB for SWIM
  credit_data <- cbind(L, L1, L2, L3, H)
  colnames(credit_data) <- c("L", "L1", "L2", "L3", "H1", "H2", "H3")
```

References

McNeil, A. J., Frey, R., and Embrechts, P. (2015). *Quantitative Risk Management: Concepts, Techniques and Tools-revised edition*. Princeton university press.

Pesenti, S. M., Millossovich, P., and Tsanakas, A. (2019). Reverse sensitivity testing: What does it take to break the model? *European Journal of Operational Research*, 274(2):654–670.