Scenario Weights for Importance Measurement (**SWIM**) – an **R** package for sensitivity analysis

Silvana M. Pesenti*[2], Alberto Bettini[3], Pietro Millossovich[4,5], Andreas Tsanakas[5]

[2]University of Toronto, [3]Assicurazioni Generali S.p.A, [4]DEAMS, University of Trieste, [5]The Business School (formerly Cass), City, University of London

05. March 2021

Abstract

The **SWIM** package implements a flexible sensitivity analysis framework, based primarily on results and tools developed by Pesenti et al. (2019). **SWIM** provides a stressed version of a stochastic model, subject to model components (random variables) fulfilling given probabilistic constraints (stresses). Possible stresses can be applied on moments, probabilities of given events, and risk measures such as Value-at-Risk and Expected Shortfall. **SWIM** operates upon a single set of simulated scenarios from a stochastic model, returning scenario weights, which encode the required stress and allow monitoring the impact of the stress on all model components. The scenario weights are calculated to minimise the relative entropy with respect to the baseline model, subject to the stress applied. As well as calculating scenario weights, the package provides tools for the analysis of stressed models, including plotting facilities and evaluation of sensitivity measures. **SWIM** does not require additional evaluations of the simulation model or explicit knowledge of its underlying statistical and functional relations; hence it is suitable for the analysis of black box models. The capabilities of **SWIM** are demonstrated through a case study of a credit portfolio model.

Keywords: Sensitivity analysis; risk measures; stress testing; sensitivity measures, Kullback-Leibler divergence

1 Introduction

1.1 Background and contribution

Complex quantitative models are used extensively in actuarial and financial risk management applications, as well as in wider fields such as environmental risk modelling (Tsanakas and Millossovich, 2016; Borgonovo and Plischke, 2016; Pesenti et al., 2019). The complexity of such models (high dimensionality of inputs; non-linear relationships) motivates the performance of sensitivity analyses, with the aim of providing insight into the ways that model inputs interact and impact upon the model output.

When model inputs are subject to uncertainty, *global* sensitivity methods are often used, considering the full space of (randomly generated) multivariate scenarios, which represent possible configurations

*Correspondence to Silvana Pesenti, Department of Statistical Sciences, University of Toronto, Canada. silvana.pesenti@utoronto.ca

of the model input vector. The particular task of ranking the importance of different model inputs leads to the use of sensitivity measures, which assign a score to each model input. A rich literature on global sensitivity analysis exists, with variance decomposition methods being particularly prominent; see Saltelli et al. (2008) and Borgonovo and Plischke (2016) for wide-ranging reviews. The **R** package **sensitivity** (Iooss et al., 2019) implements a wide range of sensitivity analysis approaches and measures.

We introduce an alternative approach to sensitivity analysis called *Scenario Weights for Importance Measurement* (**SWIM**) and present the **R** package implementing it (Pesenti et al., 2020). This approach was developed with actuarial risk models in mind, particularly those used for risk management and economic capital calculations. The aim of this paper is to provide an accessible introduction to the concepts underlying **SWIM** and a vignette demonstrating how the package is used. **SWIM** quantifies how distorting a particular model component (which could be a model input, output, or an intermediate quantity) impacts all other model components. Such analyses allow a risk modeller, for example, to rank the importance of model inputs either by the extent that their being stressed impacts the output or, conversely, the way that they respond to a stress in model output – the latter has been termed reverse sensitivity testing by Pesenti et al. (2019). The **SWIM** approach can be summarised as follows:

1. The starting point is a table of simulated scenarios, each column containing realisations of a different model component. This table forms the *baseline model* as well as the dataset on which the **SWIM** bases its calculations.

2. A *stress* is defined as a particular modification of a model component (or group of components). This could relate to a change in moments, probabilities of events of interest, or risk measures, such as Value-at-Risk or Expected Shortfall (e.g. McNeil et al. (2015)). Furthermore, there is the facility for users to design their own stresses, involving potentially more than one model component.

3. **SWIM** calculates a set of *scenario weights*, acting upon the simulated scenarios and thus modifying the relative probabilities of scenarios occurring. Scenario weights are derived such that the defined stress on model components is fulfilled, while keeping the distortion to the baseline model to a minimum, as quantified by the Kullback-Leibler divergence (relative entropy). Alternatively, users are able to import their own set of weights, generated by a method of their choice.

4. Given the calculated scenario weights, the impact of the stress on the distributions of all model components is worked out and sensitivity measures, useful for ranking model components, are evaluated.

A key benefit of **SWIM** are that it provides a sensitivity analysis framework that is economical both computationally and in terms of the information needed to perform the analysis. Specifically, sensitivity analysis is performed using only one set of simulated scenarios. No further simulations are needed, thus eliminating the need for repeated evaluation of the model, which could be numerically expensive. Furthermore, the user of **SWIM** needs to know neither the explicit form of the joint distribution of model components nor the exact form of functional relations between them. Hence, **SWIM** is appropriate for the analysis of *black box* models, thus having a wide scope of applications. Specifically, SWIM is well suited to simulation models used in insurance risk management, which are characterised by high dimensions, complex interactions between risk factors, and high computational cost of re-simulating under different assumptions.

While there is an extensive literature on sensitivity analysis and there exist multitudes of sensitivity measures, our proposed sensitivity analysis framework differs in that it is model independent and can

be applied in a numerically efficient way since it does not require potentially expensive re-evaluation of the model's output. Moreover, our sensitivity analysis framework focuses on risk measures which are widely used in risk management (McNeil et al., 2015), whereas much of the literature on sensitivity analysis focuses on variance-based and moment-independent sensitivity measures (Borgonovo et al., 2016); thus the current manuscript adds a perspective that is missing by standard variance-based and moment-independent approaches.

The proposed sensitivity analysis framework implemented in **SWIM** is based on theoretical results derived in Pesenti et al. (2019). While these results hold in generality, the **SWIM** package fundamentally hinges on the fact that it works on a set of Monte Carlo simulations. Thus, the quality of the sensitivity analysis conducted using **SWIM** is intimately connected with the quality of the dataset. Specifically, as we work on an empirical space, the user is constrained to change the probability of already simulated scenarios, without the ability to introduce new ones. Hence, the user must specify stresses judiciously so that they can be supported by the given dataset.

The **SWIM** approach is largely based on Pesenti et al. (2019) and uses theoretical results on risk measures and sensitivity measures developed in that paper. An early sensitivity analysis approach based on scenario weighting was proposed by Beckman and McKay (1987). The Kullback-Leibler divergence has been used extensively in the financial risk management literature – papers that are conceptually close to **SWIM** include Weber (2007); Breuer and Csiszár (2013); and Cambou and Filipović (2017). Some foundational results related to the minimisation of the Kullback-Leibler divergence are provided in Csiszár (1975).

1.2 Installation

The **SWIM** package can be installed from CRAN or through GitHub:

```
# directly from CRAN
install.packages("SWIM")
# and the development version from GitHub
devtools::install_github("spesenti/SWIM")
```

1.3 Structure of the paper

Section 2 provides an introduction to **SWIM**, illustrating key concepts and basic functionalities of the package on a simple example. Section 3 contains technical background on the optimisations that underlay the **SWIM** package implementation. Furthermore, Section 3 includes a brief reference guide, providing an overview of implemented **R** functions, objects, and graphical/analysis tools. Finally, a detailed case study of a credit risk portfolio is presented in Section 4. Through this case study, advanced capabilities of **SWIM** for sensitivity analysis are demonstrated, including more complex user-designed stresses.

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 $(X_1, \ldots, X_d)$, along with *scenario weights* W assigned to individual realisations, as shown in Table 1. Hence each 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.

There is a conceptual distinction between the model from which the scenarios are simulated and the model as understood here. Specifically, as we are considering the case when an analyst has access to simulated scenarios only, and not the data-generating mechanism, we are consistently working with the empirical probability measure and any expectations or probabilities stated below are given with respect to that measure. Furthermore, as we work on an empirical space, we systematically conflate a random variable with its vector of realisations – in particular, the scenario weights W can be identified with a Radon-Nikodym on that space. For relevant notation on the space we are working on, see Section 3.

Table 1: Illustration of the **SWIM** framework, that is the baseline model, the stressed model and the scenario weights.

X_1	X_2	$\ldots$	X_d	W
x_{11}	x_{21}	$\ldots$	x_{d1}	w_1
x_{12}	x_{22}	$\ldots$	x_{d2}	w_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{1n}	x_{2n}	$\ldots$	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 or 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* – consequently weights of a baseline model will never be explicitly mentioned. When scenario weights are not identically equal to 1, such that some scenarios are more weighted than others, 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 E^W and cumulative distribution function F^W 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 *stress* on the distribution of X_i, for example requiring that $E^W(X_1) = m$;

we then say that she is *stressing* X_1 and, by extension, the model. Subsequently, **SWIM** calculates the scenario weights such that the stress is fulfilled and the distortion to the baseline model induced by the stress is as small as possible; specifically the Kullback-Leibler divergence (or relative entropy) between the baseline and stressed models is minimised. (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 determine the stressed distribution of any model component or 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. 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 3 and 4.

In sensitivity analysis, one often considers a model with a vector of inputs $\mathbf{Z}$ and an output $Y = g(Z)$, for some aggregation function g that maps inputs to the real line. Then the importance of model inputs can be investigated by stressing the distribution of inputs and observing the impact on the output distribution and vice versa (Pesenti et al., 2019). (We note that in **SWIM** there is no ex ante assumption about which of the model components $(X_1, \ldots, X_d)$ should be interpreted as inputs or outputs – these variables could represent any randomly varying quantity in the model. For that reason, we use X_i for variable labels when documenting the **SWIM**'s capabilities, but use alternative notation in illustrations where variables have specific interpretable meanings, as in the current section and in the case study of Section 4).

Here, we consider a simple portfolio model, with the portfolio loss defined by $Y = Z_1 + Z_2 + Z_3$. The random variables Z_1, Z_2, Z_3 represent normally distributed losses, with $Z_1 \sim N(100, 40^2)$, $Z_2 \sim Z_3 \sim N(100, 20^2)$. Z_1 and Z_2 are correlated, while Z_3 is independent of (Z_1, Z_2). 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.

```
set.seed(0)
# number of simulated scenarios
n.sim <- 10 ^ 5
# correlation between Z1 and Z2
r <- 0.5
# simulation of Z1  and Z2
# constructed as a 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
# data of baseline model
dat <- data.frame(Z1, Z2, Z3, Y)
```

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 a **SWIM** object, which we call `str.mean`. This object stores the stressed model, i.e. the realisations of the model components and the scenario weights. In the function call, the argument `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.

```
library(SWIM)
str.mean <- stress(type = "mean", x = dat, k = 1, new_means = 110)
```

```
##   cols required_moment achieved_moment abs_error rel_error
## 1    1              110             110  -8.8e-10    -8e-12
```

```
summary(str.mean, base = TRUE)
```

```
## $base
##                    Z1       Z2       Z3        Y
## mean          1.0e+02  99.9404  99.9843 299.9811
## sd            4.0e+01  19.9970  19.9819  56.6389
## skewness     -6.1e-04   0.0012  -0.0025  -0.0023
## ex kurtosis -1.1e-02  -0.0090  -0.0126  -0.0094
## 1st Qu.       7.3e+01  86.4745  86.4816 261.6121
## Median        1.0e+02  99.9866 100.0091 300.0548
## 3rd Qu.       1.3e+02 113.3957 113.4934 338.2670
##
## $`stress 1`
##                    Z1       Z2       Z3        Y
## mean         110.0000 102.4437  99.9828 312.4265
## sd            40.0333  19.9954  19.9762  56.6173
## skewness      -0.0024  -0.0015  -0.0049  -0.0037
## ex kurtosis   -0.0050  -0.0032  -0.0155  -0.0012
## 1st Qu.       82.9984  88.9771  86.4815 274.2200
## Median       110.0759 102.4810  99.9954 312.5039
## 3rd Qu.      136.9310 115.8744 113.5019 350.6120
```

The `summary` function, applied to the **SWIM** object `str.mean`, shows how the distributional characteristics of all random variables 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 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.

```
# 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)
```

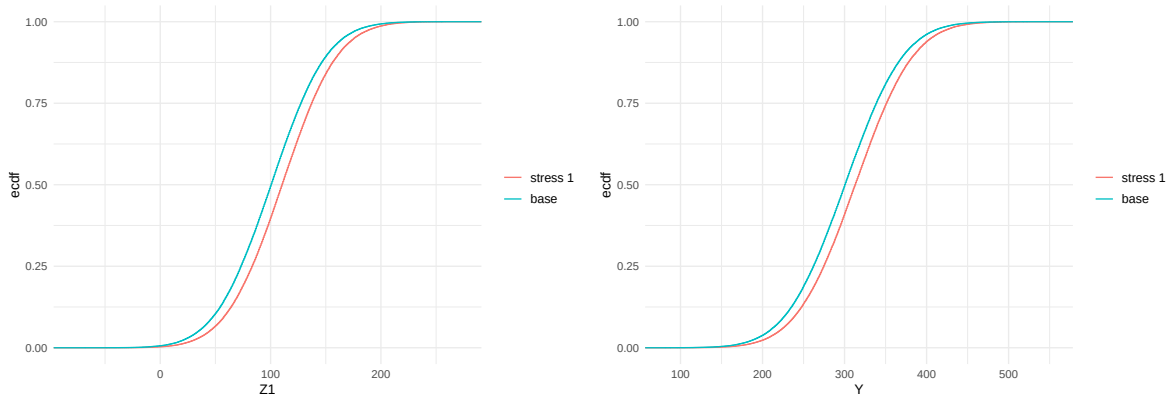

Figure 1: Baseline and stressed empirical distribution functions of model components Z_1 (left) and Y (right), subject to a stress on the mean of Z_1.

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 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). Figure 3 displays the scenario weights as a function of the input variable Z_1 and Z_2. The different colours of the scenario weights indicate their relative sizes. We observe that the scenario weights are increasing jointly in Z_1 and Z_2.

```
# parameter n specifies the number of scenario weights plotted
plot_weights(str.mean, xCol = "Z1", n = 1000)
# specifying the limits of the x-axis
plot_weights(str.mean, xCol = "Y", x_limits = c(90, 550), n = 1000)
```

The stress to 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.

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

```
## $`stress 1`
```

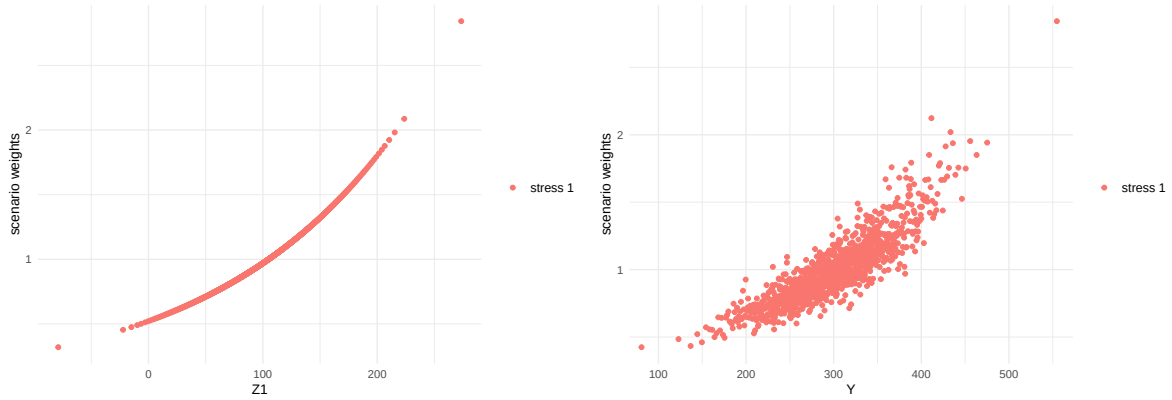

Figure 2: Scenario weights against observations of model components Z_1 (left) and Y (right), subject to a stress on the mean of Z_1.

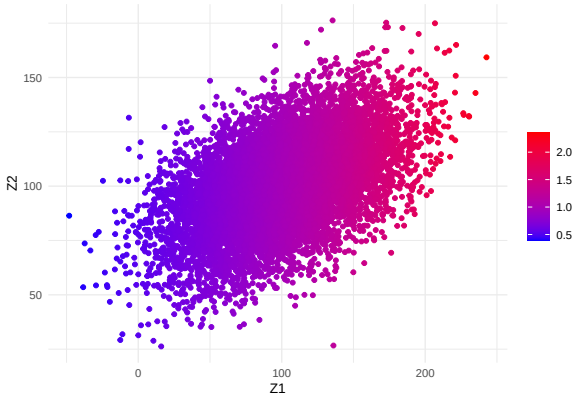

Figure 3: Scenario weights for observations of model components Z_1 Z_2 subject to a stress on the mean of Z_1.

```
##                    Z1       Z2       Z3        Y
## mean         100.0000   99.941  99.9782 299.9187
## sd            50.0005   21.349  19.9800  67.9233
## skewness      -0.0027    0.007  -0.0034   0.0049
## ex kurtosis   -0.0556   -0.033  -0.0061  -0.0427
## 1st Qu.       66.0964   85.495  86.4822 253.7496
## Median       100.1290   99.974 100.0455 299.9766
## 3rd Qu.      133.7733 114.301 113.4701 345.9159
```

Furthermore, in Figure 4, we compare the baseline and stressed cdfs of Z_1 and Y, under the new stress on Z_1. The crossing of probability distributions reflects the increase in volatility.

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

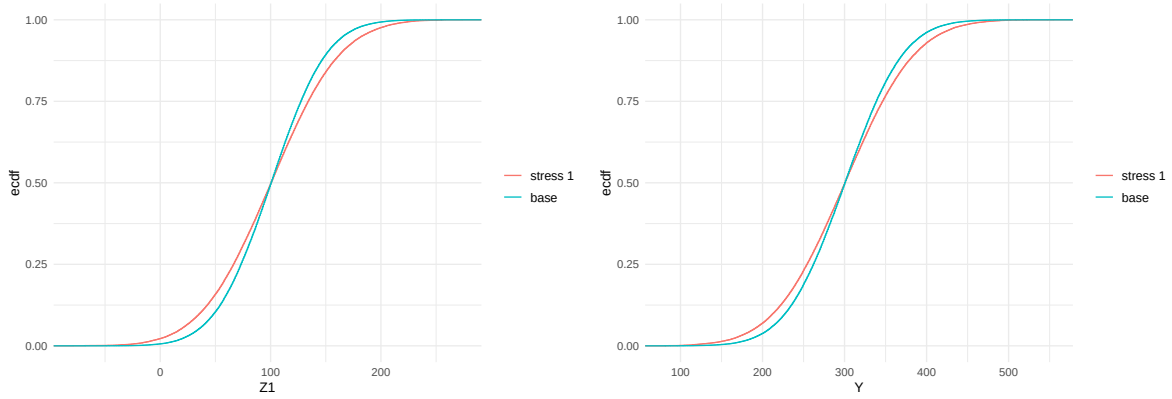

Figure 4: Baseline and stressed empirical distribution functions of model components Z_1 (left) and Y (right), subject to a stress on the standard deviation of Z_1.

The different way in which a stress on the standard deviation of Z_1 impacts on the model, compared to a stress on the mean, is reflected by the scenario weights. Figure 5 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.

```
plot_weights(str.sd, xCol = "Z1", n = 2000)
plot_weights(str.sd, xCol = "Y", n = 2000)
```

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 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.

```
stress(type = "mean", x = dat, k = 1, new_means = 300)
```

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

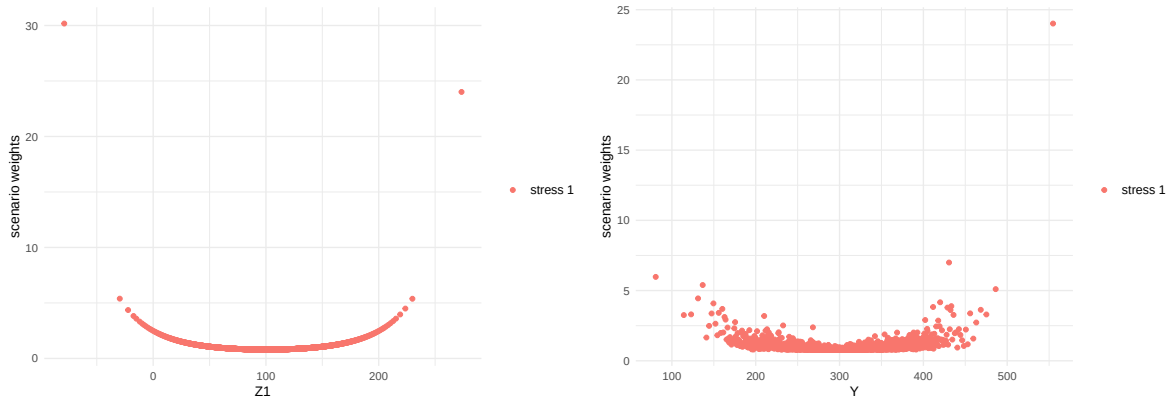

Figure 5: Scenario weights against observations of model components Z_1 (left) and Y (right), subject to a stress on the standard deviation of Z_1.

```
max(Z1)
```

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

3 Scope of the SWIM package

3.1 Stressing a model

We briefly introduce key concepts, using slightly more technical language compared to Section 2. A *model* consists of a random vector of *model components* $\mathbf{X} = (X_1, \ldots, X_d)$ and a probability measure; we denote the probability measure of a *baseline model* by P and that of a *stressed model* by P^W, where $W = \frac{dP^W}{dP}$, satisfying $E(W) = 1$ and $W \geq 0$, is a Radon-Nikodym derivative. In a Monte Carlo simulation context, the probability space is discrete with n states $\Omega = \{\omega_1, \ldots, \omega_n\}$, each of which corresponds to a simulated scenario. To reconcile this formulation with the notation of Section 2, we denote, for $i = 1, \ldots, n,\ j = 1, \ldots, d$, the realisations $X_j(\omega_i) := x_{ji}$ and $W(\omega_i) := w_i$; the latter are the *scenario weights*. Under the baseline model, each scenario has the same probability $P(\omega_i) = 1/n$, while under a stressed model it is $P^W(\omega_i) = W(\omega_i)/n = w_i/n$.

The stressed model thus arises from a change of measure from P to P^W, which entails the application of scenario weights $w_1, \ldots, w_n$ on individual simulations. **SWIM** calculates scenario weights such that model components fulfil specific stresses, while the distortion to the baseline model is as small as possible when measured by the Kullback-Leibler divergence (or relative entropy). The Kullback-Leibler divergence of the probability measure of the baseline model P with respect to that of a stressed model P^W is defined as

$$E(W \log(W)) = \int \frac{dP^W}{dP} \log\left(\frac{dP^W}{dP}\right) dP. \tag{1}$$

The Kullback-Leibler divergence is non-negative, vanishes if and only if the probabilities coincide, i.e., if $P = P^W$, and is often used as a measure of discrepancy between probability measures (Pesenti et al., 2019).

A stressed model is defined as the solution to

$$\min_W \ E(W \log(W)), \quad \text{subject to constraints on } \mathbf{X} \text{ under } P^W. \tag{2}$$

In what follows, we denote by a superscript W operators under the stressed model, such as F^W, E^W for the probability distribution and expectation under the stressed model, respectively. We refer to Pesenti et al. (2019) and references therein for further mathematical details and derivations of solutions to (2).

Table 2 provides a collection of all implemented types of stresses in the **SWIM** package. The precise constraints of (2) are explained below.

Table 2: Implemented types of stresses in **SWIM**.

R function	Stress	`type`	Reference
`stress`	wrapper for the `stress_type` functions		Sec. 3.1.1
`stress_user`	user defined scenario weights	`user`	Sec. 3.1.5
`stress_prob`	probabilities of disjoint intervals	`prob`	Eq. (3)
`stress_mean`	means	`mean`	Eq. (4)
`stress_mean_sd`	means and standard deviations	`mean sd`	Eq. (5)
`stress_moment`	moments (of functions)	`moment`	Eq. (6)
`stress_VaR`	VaR risk measure (quantile)	`VaR`	Eq. (7)
`stress_VaR_ES`	VaR and ES risk measures	`VaR ES`	Eq. (8)

The solutions to the optimisations (3) and (7) are worked out fully analytically (Pesenti et al., 2019), whereas problems (4), (5), (6) and (8) require some root-finding. Specifically, problems (4), (5) and (6) rely on the package **nleqslv**, whereas (8) uses the `uniroot` function.

3.1.1 The `stress` function and the SWIM object

The `stress` function is a wrapper for the `stress_type` functions, where `stress(type = "type", )` and `stress_type` are equivalent. The `stress` function solves optimisation (2) for constraints specified through `type` and returns a `SWIM` object, that is, a list including the elements shown in Table 3:

Table 3: The **SWIM** object, returned by any `stress` function.

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

The data frame containing the realisations of the baseline model, `x` in the above table, 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 scenario weights. The details of the applied stress can be obtained using `get_specs`.

3.1.2 Stressing disjoint probability intervals

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

`stress_prob` solves (2) with the constraints

$$P^W(X_j \in B_k) = \alpha_k,\ k = 1, \ldots, K, \tag{3}$$

for disjoint intervals $B_1, \ldots, B_K$ with $P(X_j \in B_k) > 0$ for all $k = 1, \ldots, K$, and $\alpha_1, \ldots, \alpha_K > 0$ such that $\alpha_1 + \ldots + \alpha_K \leq 1$ and a model component X_j.

3.1.3 Stressing moments

The functions `stress_mean`, `stress_mean_sd` and `stress_moment` implement the solution in Csiszár (1975) and provide stressed models with moment constraints. The function `stress_mean` returns a stressed model that fulfils constraints on the first moment of model components. Specifically,

`stress_mean` solves (2) with the constraints

$$E^W(X_j) = m_j,\ j \in J, \tag{4}$$

for $m_j,\ j \in J$, where J is a subset of $\{1, \ldots, d\}$.

The arguments m_j are specified in the `stress_mean` function through the argument `new_means`. The `stress_mean_sd` function allows to stress simultaneously the mean and the standard deviation of model components. Specifically,

`stress_mean_sd` solves (2) with the constraints

$$E^W(X_j) = m_j \text{ and } \mathrm{Var}^W(X_j) = s_j^2,\ j \in J, \tag{5}$$

for $m_j, s_j,\ j \in J$, where J is a subset of $\{1, \ldots, d\}$.

The arguments m_j, s_j are defined in the `stress_mean_sd` function by the arguments `new_means` and `new_sd` respectively. The functions `stress_mean` and `stress_mean_sd` are special cases of the general `stress_moment` function, which allows for stressed models with constraints on functions of the (joint) moments of model components. Specifically

For $k = 1, \ldots, K$, J_k subsets of $\{1, \ldots, d\}$ and functions $f_k \colon \mathbb{R}^{|J_k|} \to \mathbb{R}$, `stress_moment` solves (2) with the constraints

$$E^W(f_k(\mathbf{X}_{J_k})) = m_k,\ k = 1, \ldots, K, \tag{6}$$

for $m_k,\ k = 1, \ldots, K$ and $\mathbf{X}_{J_k}$ the subvector of model components with indices in J_k.

Note that `stress_moment` not only allows to define constraints on higher moments of model components, but also to construct constraints that apply to multiple model components simultaneously. For example, the stress $E^W(X_h X_l) = m_k$ is achieved by setting $f_k(x_h, x_l) = x_h x_l$ in (6) above. 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 scenario weights calculated via numerical optimisation, using the **nleqslv** package. Thus, depending on the choice of constraints, existence or uniqueness of a stressed model is not guaranteed. The `stress_moment` function will print a message stating the specified values for the required moments, alongside the moments achieved under the stressed model resulting from the function call. If the two match, the stress specification has been successfully fulfilled.

3.1.4 Stressing risk measures

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

$$\mathrm{VaR}_\alpha(Z) = F^{-1}(\alpha) = \inf\{y \in \mathbb{R} \mid F(y) \geq \alpha\}.$$

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

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

The details of the constraints that `stress_VaR` and `stress_VaR_ES` solve, are as follows:

For $0 < \alpha < 1$ and q, s such that $q < s$, `stress_VaR` solves (2) with the constraint

$$\mathrm{VaR}_\alpha^W(X_j) = q; \tag{7}$$

and `stress_VaR_ES` solves (2) with the constraints

$$\mathrm{VaR}_\alpha^W(X_j) = q \text{ and } \mathrm{ES}_\alpha^W(X_j) = s. \tag{8}$$

Note that, since **SWIM** works with discrete distributions, the exact required constraints may not be achievable, see Pesenti et al. (2019) for more details. In that case, the `stress` function will print a message with the achieved and required constraints. For example `stress_VaR` will return scenario weights inducing the largest quantile in the dataset smaller or equal to the required VaR (i.e. q); this guarantees that $P^W(X_j \leq q) = \alpha$.

3.1.5 User defined scenario weights

The option `type = "user"` allows to generate a **SWIM** object with scenario weights defined by a user. The scenario weights can be provided directly via the `new_weights` argument or through a list of functions, `new_weightsfun`, that applied to the data `x` generates the scenario weights.

3.2 Analysis of stressed models

Table 4 provides a complete list of all implemented **R** functions in **SWIM** for analysing stressed models, which are described below in detail.

Table 4: Implemented **R** function in **SWIM** for analysing stressed models.

R function	Analysis of Stressed Models
`summary`	summary statistics
`cdf`	cumulative distribution function
`quantile_stressed`	quantile function
`VaR_stressed`	VaR
`ES_stressed`	ES
`sensitivity`	sensitivity measures
`importance_rank`	importance ranks

R function	Analysis of Stressed Models
`plot_cdf`	plots cumulative distributions functions
`plot_quantile`	plots quantile functions
`plot_weights`	plots scenario weights
`plot_hist`	plots histograms
`plot_sensitivity`	plots sensitivity measures

3.2.1 Distributional comparison

The **SWIM** package contains functions to compare the distribution of model components under different (stressed) models. The function `summary` is a method 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 `summary` function returns for each set of scenario weights a list, containing the elements shown in Table 5.

Table 5: The output of the `summary` function applied to a **SWIM** object.

`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 function of model components under a stressed model[1] can be calculated using the `cdf` function of the **SWIM** package, applied to a **SWIM** object. To calculate sample quantiles of stressed model components, the function `quantile_stressed` can be used. The function `VaR_stressed` and `ES_stressed` provide the stressed VaR and ES of model components, which is of particular interest for stressed models resulting from constraints on risk measures, see Section 3.1.4. (While `quantile_stressed` works very similarly to the base **R** function `quantile`, `VaR_stressed` provides better capabilities for comparing different models and model components.)

Implemented visualisation of distribution functions are `plot_cdf`, for plotting empirical distribution functions, `plot_quantile`, for plotting empirical quantile functions, and `plot_hist`, for plotting histograms of model components under different (stressed) models. The scenario weights can be plotted against a model component using the function `plot_weights`.

3.2.2 Sensitivity measures

Determining sensitivities of different model components is a fundamental component of model building, interpretation, and validation and we refer to Saltelli et al. (2008) and Borgonovo and Plischke (2016) for a comprehensive review. A key tool in sensitivity analysis are sensitivity measures that associate to every model component a sensitivity score. Here we introduce the sensitivity measures implemented in the **SWIM** package for comparing baseline and stressed models and how model components change

[1]Note that **R** functions implementing the empirical cdf or the quantile, `ecdf` and `quantile`, will not return the empirical distribution function or the quantile function under a stressed model.

under different models. In particular, the **SWIM** packages contains the `sensitivity` function, which calculates sensitivity measures of stressed models and model components. The implemented sensitivity measures, summarised in the table below, are the *Wasserstein of order 1*, *Kolmogorov* and the *Gamma* sensitivity measures, see also Pesenti et al. (2016), Pesenti et al. (2019), and Emmer et al. (2015). While the Wasserstein and the Kolmogorov sensitivity measures are defined as distances on the space of distributions, they nevertheless provide useful insight in how the baseline model changes under a stress.

Table 6: Definition of the sensitivity measures implemented in **SWIM**.

Metric	**Definition**
Wasserstein	$\int \lvert F_X^W(x) - F_X(x)\rvert dx$
Kolmogorov	$\sup_x \lvert F_X^W(x) - F_X(x)\rvert$
Gamma	$\frac{E^W(X)-E(X)}{c}$, for a normalisation c

The Wasserstein distance is typically defined via the mass-transportation problem and we refer to Villani (2008) for a detailed treatment of its properties. For distributions on the real line, the Wasserstein distance admits the above representation which is convenient for numerical evaluations (Vallender, 1974). The difference between the Wasserstein and the Kolmogorov sensitivity is that, while the latter determines the largest pointwise distance between a stressed and the baseline model, the former sensitivity sensitivity reflects the entire distribution.

The Gamma sensitivity is introduced in Pesenti et al. (2019) and we refer to that paper for its properties and a comparison to variance-based sensitivity measures and moment independent sensitivity measures. Loosely speaking, the Gamma sensitivity measure represents the difference between the first moments of the stressed and the baseline distribution of a model component. The Gamma measure is normalised such that it takes values between -1 and 1, with higher positive (negative) values corresponding to a larger positive (negative) impact of the stress on the particular model component. The sensitivity measures can be plotted using `plot_sensitivity`. The function `importance_rank` returns the effective rank of model components according to the chosen sensitivity measure. A small rank of a model component's sensitivity measure corresponds to high sensitivity to that model component. The functions `sensitivity` together with `plot_sensitivity` and `importance_rank` allow for a numerically efficient sensitivity analysis of a model and provide visual assessments of model components' sensitivities to a stressed model.

4 Case study

4.1 A credit risk portfolio

In this section we provide a detailed case study of the use of **SWIM** in analysing a credit risk model. Through this analysis, we also illustrate more advanced capabilities of the package. The credit model in this section is a conditionally binomial loan portfolio model, including systematic and specific portfolio risk. We refer to the Appendix A for details about the model and the generation of the simulated data. A key variable of interest is the total aggregate portfolio loss $L = L_1 + L_2 + L_3$, where L_1, L_2, L_3 are homogeneous subportfolios on a comparable scale (say, thousands of \$). The dataset contains 100,000 simulations of the portfolio L, the subportfolios L_1, L_2, L_3 as well as the random default probabilities within each subportfolio, H_1, H_2, H_3. These default probabilities represent the systematic risk *within* each subportfolio, while their dependence structure represents a systematic risk

effect *between* the subportfolios. We may thus think of L as the model output, H_1, H_2, H_3 as model inputs, and L_1, L_2, L_3 as intermediate model outputs.

The simulated data of the credit risk portfolio are included in the **SWIM** package and can be accessed via `data("credit_data")`. A snippet of the dataset looks as follows:

```
data("credit_data")
head(credit_data)
```

```
##         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 portfolio loss

In this section, we follow a reverse sensitivity approach, similar to Pesenti et al. (2019). Specifically, we study the effects that stresses on (the tail of) the aggregate portfolio loss L have on the three subportfolios. This enables us to assess their comparative importance. If a subportfolio's loss distribution substantially changes following a stress on the portfolio loss, we interpret this as a high sensitivity to that subportfolio.

First, we impose a 20% increase on the VaR at level 90% of the portfolio loss.

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

```
## Stressed VaR specified was 2174.25 , stressed VaR achieved is 2173.75
```

The 20% increase was specified by setting the `q_ratio` argument to 1.2 – alternatively the argument `q` can be set to the actual value of the stressed VaR.

Using the function `VaR_stressed`, we can quantify how tail quantiles of the aggregate portfolio loss change, when moving from the baseline to the stressed model. We observe that the increase in the VaR of the portfolio loss changes more broadly its tail quantiles; thus the stress on VaR also induces an increase in ES. The implemented functions `VaR_stressed` and `ES_stressed` calculate respectively VaR and ES; the argument `alpha` specifies the levels of VaR and ES, respectively, while the stressed model under which the risk measures are calculated can be chosen using `wCol` (by default equal to 1).

```
VaR_stressed(object = stress.credit, alpha = c(0.75, 0.9, 0.95, 0.99),
    xCol = "L", wCol = 1, base = TRUE)
```

```
##        L base L
## 75% 1506   1399
## 90% 2174   1812
## 95% 2426   2085
## 99% 2997   2671
```

```
ES_stressed(object = stress.credit, alpha = 0.9, xCol = "L", wCol = 1,
    base = TRUE)
```

```
##        L base L
## 90% 2535   2191
```

As a second stress, we consider, additionally to the 20% increase in the $\text{VaR}_{0.9}$, an increase in $\text{ES}_{0.9}$ of the portfolio loss L. When stressing VaR and ES together via `stress_VaR_ES`, both VaR and ES need to be stressed at the same level, here `alpha = 0.9`. We observe that when stressing the VaR alone, ES increases to 2535. For the second stress we want to induce a greater impact on the tail of the portfolio loss distribution, thus we require that the stressed ES be equal to 3500. This can be achieved by specifying the argument `s`, which is the stressed value of ES (rather than `s_ratio`, the proportional increase).

```
stress.credit <- stress(type = "VaR ES", x = stress.credit, k = "L", alpha = 0.9,
    q_ratio = 1.2, s = 3500)
```

```
## Stressed VaR specified was 2174.25 , stressed VaR achieved is 2173.75
```

When applying the `stress` function or one of its alternative versions to a **SWIM** object rather than to a data frame (via `x = stress.credit` in the example above), the result will be a new **SWIM** object with the new stress "appended" to existing stresses. This is convenient when large datasets are involved, as the `stress` function returns an object containing the original simulated data and the scenario weights. Note however, that this only works if the underlying data are exactly the same.

4.3 Analysing stressed models

The `summary` function provides a statistical summary of the stressed models. Choosing `base = TRUE` compares the stressed models with the 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        1289.90 21.70 558.27 709.93 0.000437 0.01180 0.0566
## sd           875.90 30.57 507.78 447.30 0.000448 0.01045 0.0351
## skewness       1.90  2.17   2.10   1.57 2.090425 2.10128 1.5384
## ex kurtosis    3.67  6.74   4.79   2.80 6.203429 4.97000 2.6142
## 1st Qu.      739.38  0.00 234.38 405.00 0.000123 0.00512 0.0328
## Median      1065.62 20.00 412.50 605.00 0.000297 0.00879 0.0484
## 3rd Qu.     1505.62 40.00 675.00 875.00 0.000594 0.01439 0.0697
```

In the `summary` output, `stress 1` corresponds to the 20% increase in the VaR, while `stress 2` corresponds to the stress in both VaR and ES. The information on individual stresses can be recovered through the `get_specs` function, and the actual scenario weights using `get_weights`. Since the **SWIM** object `stress.credit` contains two stresses, the scenario weights that are returned by `get_weights` form a data frame consisting of two columns, corresponding to `stress 1` and to `stress 2`, respectively. We can observe from the summary that the two stresses modify the distributions of model components in somewhat different ways. For example, the more tail-oriented `stress 2` leads to an increase in both the skewness and excess kurtosis of the portfolio loss.

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

```
##            type k alpha       q    s
## stress 1    VaR L   0.9 2173.75 <NA>
## stress 2 VaR ES L   0.9 2173.75 3500
```

Next, we illustrate the difference between the two stresses applied, by a scatter plot of the scenario weights against the portfolio loss L. As the number of scenario weights is large, we only 5000 data points. This can be achieved via the parameter `n` in the function `plot_weights`, that has a default of $n = 5000$.

```
plot_weights(stress.credit, xCol = "L", wCol = 1, n = 2000)
# parameter `wCol` specifies the stresses, whose scenario weights are plotted.
plot_weights(stress.credit, xCol = "L", wCol = 2, n = 7000)
```

It is seen in Figure 6 that the weights generated to stress VaR, and VaR and ES together, follow different patterns to the weights used to stress means and standard deviations, as shown in Section 2. Recall that **SWIM** calculates the scenario weights such that under the stressed model the given constraints are fulfilled. Thus, an increase in the VaR and/or ES of the portfolio loss L results in large positive realisations of L being assigned higher weight. On the other hand, when the standard deviation is stressed, scenario weights are calculated that inflate the probabilities of both large positive and negative values. When we compare `stress 1` and `stress 2` in this example, we see that stressing VaR induces a high but constant weight on scenarios that correspond to large outcomes of L, while when stressing VaR and ES, the weights are exponentially increasing in (tail observations of) L. This difference in pattern is associated with the different impacts on the shape of the tail of L.

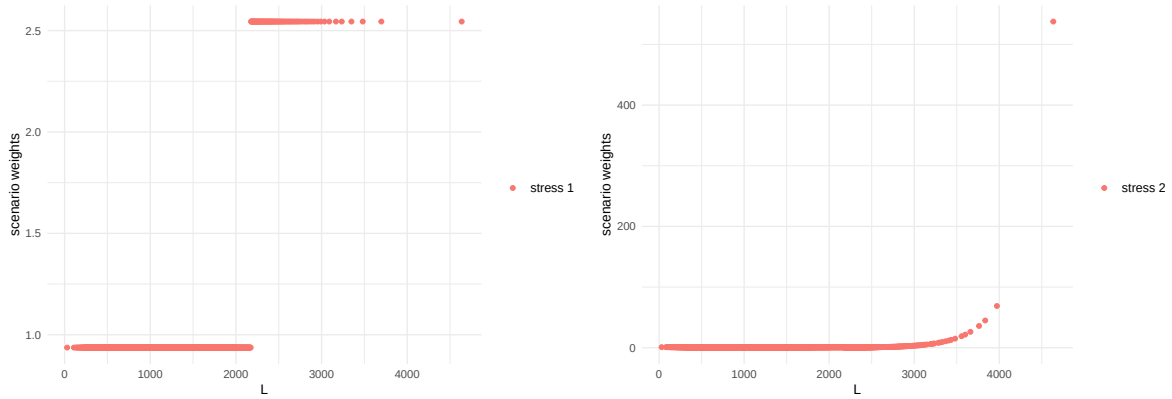

Figure 6: Scenario weights against the portfolio loss L for stressing VaR (left) and stressing both VaR and ES (right).

4.4 Visualising stressed distributions

The change in the distributions of the portfolio and subportfolio losses, when moving from the baseline to the stressed models, can be visualised through the functions `plot_hist` and `plot_cdf`. The following figure displays the histogram of the aggregate portfolio loss under the baseline and the two stressed models. It is seen how stressing VaR and ES has a higher impact on the right tail of L, compared to stressing VaR only. This is consistent with the tail-sensitive nature of the Expected Shortfall risk measure (McNeil et al., 2015). Moreover, the discontinuity in the way that, for `stress 1`, values of L map to the weights W, as seen in Figure 6, makes the stressed density of L no longer monotonic in the tail. These observations indicate that stressing both VaR and ES together may be a preferable option for risk management applications.

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

The arguments `xCol` and `wCol` (with default to plot all stresses) define the columns of the data and the columns of the scenario weights, respectively, that are used for plotting. Next, we analyse the impact that stressing the aggregate loss L has on the subportfolios L_1, L_2 L_3. Again, we use the function `plot_hist` and `plot_cdf` for visual comparison, but this time placing the distribution plots and histograms of subportfolio losses along each other via the function `ggarrange` (from the package **ggpubr**). The plots obtained from `plot_hist` and `plot_cdf` can be further customised when specifying the argument `displ = FALSE`, as then the graphical functions `plot_hist` and `plot_cdf` return data frames compatible with the package **ggplot2**.

```
pL1.cdf <- plot_cdf(object = stress.credit, xCol = 2, wCol = "all", base = TRUE)
pL2.cdf <- plot_cdf(object = stress.credit, xCol = 3, wCol = "all", base = TRUE)
pL3.cdf <- plot_cdf(object = stress.credit, xCol = 4, wCol = "all", base = TRUE)

pL1.hist <- plot_hist(object = stress.credit, xCol = 2, wCol = "all", base = TRUE)
pL2.hist <- plot_hist(object = stress.credit, xCol = 3, wCol = "all", base = TRUE)
pL3.hist <- plot_hist(object = stress.credit, xCol = 4, wCol = "all", base = TRUE)

ggarrange(pL1.cdf, pL1.hist, pL2.cdf, pL2.hist, pL3.cdf, pL3.hist, ncol = 2,
    nrow = 3, common.legend = TRUE)
```

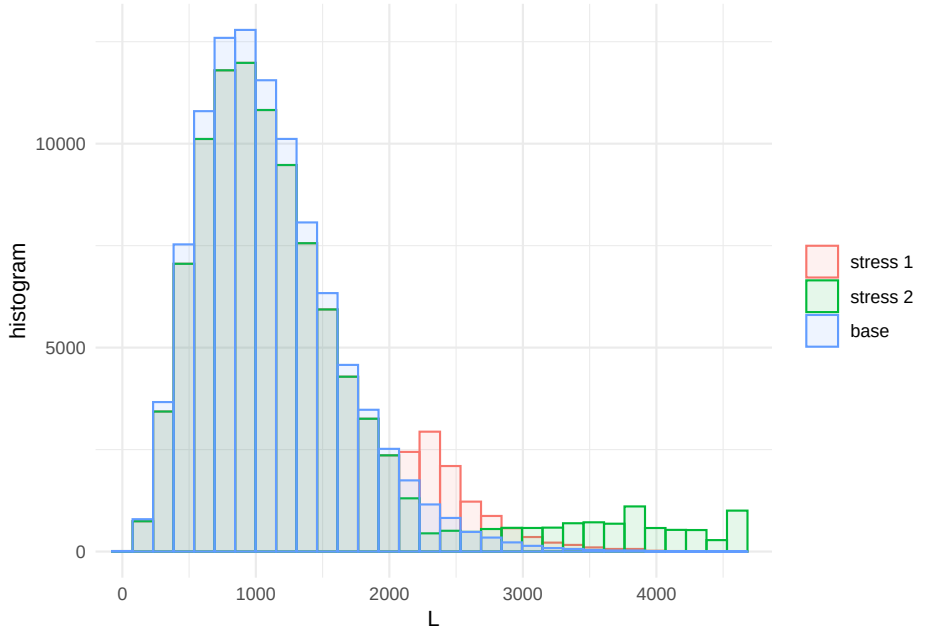

Figure 7: Histogram of the portfolio loss L under the baseline and the two stressed models.

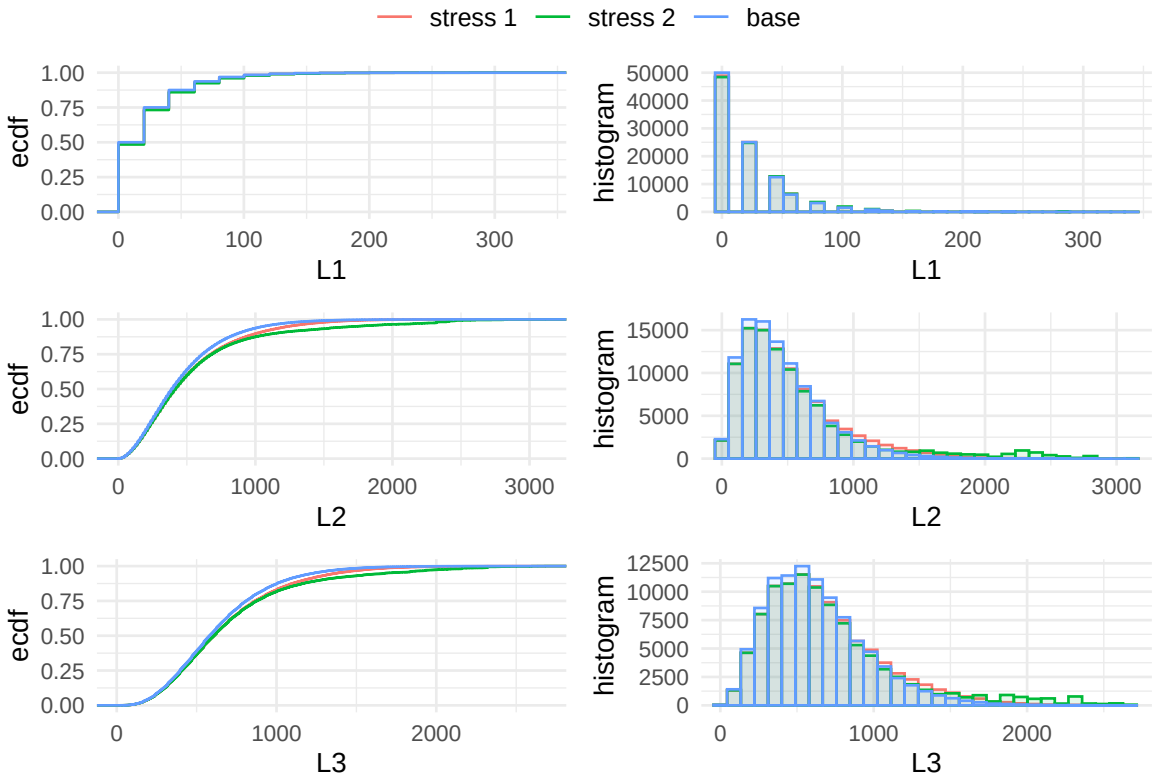

Figure 8: Distribution functions and histograms of the subportfolios L_1, L_2, L_3 for the stresses on the VaR (stress 1) and on both the VaR and ES (stress 2) of the portfolio loss L.

It is seen from both the distribution plots and the histograms in Figure 8 that the stresses have no substantial impact on L_1, while L_2 and L_3 are more affected, indicating a higher sensitivity. Specifically, the distributions of L_1 under the baseline model and the two stresses are visually indistinguishable. This indicates the lack of importance of L_1 with respect to portfolio tail risk. The higher impact on the tails of `stress 2` (on both VaR and ES) is also visible. Sensitivity measures quantifying these effects are introduced in the following subsection.

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* distance, the *Wasserstein* distance, and the sensitivity measure *Gamma*; the choice of measure is by the argument `type`. We refer to Section 3.2 for the definitions of those sensitivity measures. The Kolmogorov distance is useful for comparing different stressed models. Calculating the Kolmogorov distance, we observe that `stress 2` produces a larger Kolmogorov distance compared to `stress 1`, which reflects the additional stress on the ES for the stressed model `stress 2`.

```
sensitivity(object = stress.credit, xCol = 1, wCol = "all", type = "Kolmogorov")
```

```
##     stress       type      L
## 1 stress 1 Kolmogorov 0.0607
## 2 stress 2 Kolmogorov 0.0748
```

We now rank the sensitivities of model components by the measure Gamma, for each stressed model. Consistently with what the distribution plots showed, L_2 is the most sensitive subportfolio, followed by L_3 and L_1. The respective default probabilities H_1, H_2, H_3 are similarly ranked.

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

```
##     stress  type    L1    L2    L3    H1    H2    H3
## 1 stress 1 Gamma 0.150 0.819 0.772 0.196 0.811 0.767
## 2 stress 2 Gamma 0.113 0.734 0.639 0.171 0.708 0.636
```

Using the `sensitivity` function we can analyse whether the sensitivity of the joint subportfolio $L_1 + L_3$ exceeds the sensitivity of the (most sensitive) subportfolio L_2. This can be accomplished by specifying, through the argument `f`, a list of functions applicable to the columns `k` of the dataset. By setting `xCol = NULL` only the transformed data is considered. The sensitivity measure of functions of columns of the data is particularly useful when high dimensional models are considered, providing a way to compare the sensitivity of blocks of model components.

```
sensitivity(object = stress.credit, type = "Gamma", f = sum, k = c(2, 4),
    wCol = 1, xCol = NULL)
```

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

We observe that the sensitivity of $L_1 + L_3$ is larger than the sensitivity to either L_1 and L_3, reflecting the positive dependence structure of the credit risk portfolio. Nonetheless, subportfolio L_2 has not

only the largest sensitivity compared to L_1 and L_3 but also a higher sensitivity than the combined subportfolios $L_1 + L_3$. This has a clear risk management implication, as it shows conclusively that L_2 should be an area of priority for the owner of this portfolio of credit liabilities. Loosely speaking, L_2 is the portfolio from which 'problems may arise'.

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

4.6 Constructing more advanced stresses

4.6.1 Sensitivity of default probabilities

From the preceding analysis, it transpires that the subportfolios L_2 and L_3 are, in that order, most responsible for the stress in the portfolio loss, under both stresses considered. Furthermore, most of the sensitivity seems to be attributable to the systematic risk components H_2 and H_3, reflected by their high values of the Gamma measure. To investigate this, we perform another stress, resulting once again in a 20% increase in VaR(L), but this time fixing some elements of the distribution of H_2. Specifically, in addition to the 20% increase in VaR(L), we fix the mean and the 75% quantile of H_2 to the same values as in the baseline model. Hence, we once again perform a reverse stress test, but this time intentionally restricting the movement in the distribution of H_2, to enable us to focus on other variables. This set of constraints is implemented via the function `stress_moment`.

```
# 90% VaR of L under the baseline model
VaR.L <- quantile(x = credit_data[, "L"], prob = 0.9, type = 1)
# 75th quantile of H2 under the baseline model
q.H2 <- quantile(x = credit_data[, "H2"], prob = 0.75, type = 1)
# columns to be stressed (L, H2, H2)
k.stressH2 = list(1, 6, 6)
# functions to be applied to columns
f.stressH2 <- list(
                  # indicator function for L, for stress on VaR
                  function(x)1 * (x <= VaR.L * 1.2),
                  # mean of H2
                  function(x)x,
                  # indicator function for 75th quaantile of H2
                  function(x)1 * (x <= q.H2))
# new values for the 90% VaR of L, mean of H2, 75th quantile of H2
m.stressH2 = c(0.9, mean(credit_data[, "H2"]), 0.75)
stress.credit <- stress_moment(x = stress.credit, f = f.stressH2, k = k.stressH2,
                               m = m.stressH2)
```

Using the `summary` function, we verify that the distribution of H_2 under the new stress has unchanged mean and 75^{th} quantile. Then we compare the sensitivities of the subportfolio losses under all three stresses applied.

```r
summary(stress.credit, wCol = 3, xCol = 6, base = TRUE)
```

```
## $base
##                   H2
## mean         0.00968
## sd           0.00649
## skewness     1.30834
## ex kurtosis 2.49792
## 1st Qu.      0.00490
## Median       0.00829
## 3rd Qu.      0.01296
##
## $`stress 3`
##                   H2
## mean         0.00968
## sd           0.00706
## skewness     1.39135
## ex kurtosis 2.26506
## 1st Qu.      0.00453
## Median       0.00786
## 3rd Qu.      0.01296
```

```r
sensitivity(object = stress.credit, xCol = c(2:4), type = "Gamma")
```

```
##     stress  type     L1     L2    L3
## 1 stress 1 Gamma 0.1501 0.8195 0.772
## 2 stress 2 Gamma 0.1131 0.7336 0.639
## 3 stress 3 Gamma 0.0102 0.0203 0.366
```

It is seen that, by fixing part of the distribution of H_2, the importance ranking of the subportfolios changes, with L_2 now being significantly less sensitive than L_3. This confirms, in the credit risk model, the dominance of the systematic risk, reflected in the randomness of default probabilities.

4.6.2 Stressing tails of subportfolios

Up to now, we have considered the impact of stressing the aggregate portfolio loss on subportfolios. Now, following a forward sensitivity approach, we consider the opposite situation: stressing the subportfolio losses and monitoring the impact on the aggregate portfolio loss L. First, we impose a stress requiring a simultaneous 20% increase in the 90^{th} quantile of the losses in subportfolios L_2 and L_3. Note that the function `stress_VaR` (and `stress_VaR_ES`) allow to stress the VaR and/or the ES of only one model component. Thus, to induce a stress on the 90^{th} quantiles of L_2 and L_3, we use the function `stress_moments` and interpret the quantile constraints as moment constraints, via $E(1_{L_2 \leq \text{VaR}^W(L_2)})$ and $E(1_{L_3 \leq \text{VaR}^W(L_3)})$, respectively, where $\text{VaR}^W = \text{VaR} \cdot 1.2$ denotes the VaRs in the stressed model.

```r
# VaR of L2 and L3, respectively
VaR.L2 <- quantile(x = credit_data[, "L2"], prob = 0.9, type = 1)
VaR.L3 <- quantile(x = credit_data[, "L3"], prob = 0.9, type = 1)
#stressing VaR of L2 and L3
```

```
f.stress <- list(function(x)1 * (x <= VaR.L2 * 1.2),
                 function(x)1 * (x <= VaR.L3 * 1.2))
stress.credit.L2L3 <- stress_moment(x = credit_data, f = f.stress, k = list(3, 4),
                                    m = c(0.9, 0.9))
#impact on portfolio tail
VaR_stressed(stress.credit.L2L3, alpha = c(0.75, 0.9, 0.95, 0.99), xCol = "L",
             base = TRUE)
```

```
##        L base L
## 75% 1556   1399
## 90% 2086   1812
## 95% 2423   2085
## 99% 3072   2671
```

It is seen how the stressing of subportfolios L_2 and L_3 has a substantial impact on the portfolio loss. Given the importance of dependence for the distribution of the aggregate loss of the portfolio, we strengthen this stress further, by additionally requiring that the frequency of joint high losses from L_2 and L_3 is increased. Specifically, we require the joint exceedance probability to be $P^W(L_2 > VaR^W(L_2),\ L_3 > VaR^W(L_3)) = 0.06$, which is almost doubling the corresponding probability in the last stressed model, which was equal to 0.0308.

```
# probability of joint exceendance under the baseline model
mean(1 * (credit_data[, "L2"] > VaR.L2 * 1.2) * (credit_data[, "L3"] >
    VaR.L3 * 1.2))
```

```
## [1] 0.00865
```

```
# probability of joint exceendance under the stressed model
mean(get_weights(stress.credit.L2L3) * (credit_data[, "L2"] > VaR.L2 *
    1.2) * (credit_data[, "L3"] > VaR.L3 * 1.2))
```

```
## [1] 0.0308
```

```
# additionally stress joint exceedance probability of L2 and L3
f.stress.joint <- c(f.stress, function(x) 1 * (x[1] > VaR.L2 * 1.2) * (x[2] >
    VaR.L3 * 1.2))
stress.credit.L2L3 <- stress_moment(x = stress.credit.L2L3, f = f.stress.joint,
    k = list(3, 4, c(3, 4)), m = c(0.9, 0.9, 0.06))
```

We analyse the impact the stresses of the tail of the subportfolios L_2 and L_3 have on the aggregate portfolio L. For this, we plot in Figure 9 the quantile of the aggregate portfolio under the baseline model (blue), under the stress on the tail of L_2 and L_3 (red), and under the additional stress on the joint tail of L_2 and L_3 (green).

```
plot_quantile(stress.credit.L2L3, xCol = "L", wCol = "all", base = TRUE,
    x_limits = c(0.75, 1))
```

The results and the plots indicate that the additional stress on joint exceedances of subportfolios, increases the tail quantiles of L even further, demonstrating the importance of (tail-)dependence in portfolio risk management.

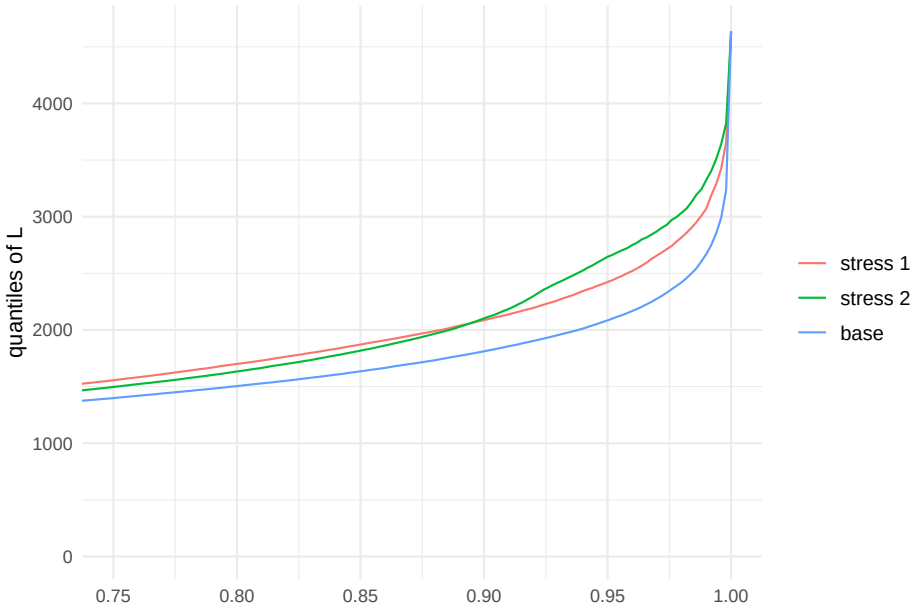

Figure 9: Quantiles of the aggregate loss L under the baseline (blue), the stress on the tails of L_2 and L_3 (red), and the additional stress on the joint tail of L_2 and L_3 (green).

5 Conclusion and future work

The **SWIM** package enables users to perform flexible and efficient sensitivity analyses of simulation models, using the method of stressing model components by re-weighting simulated scenarios. Multiple possibilities were demonstrated, from prioritising risk factors via reverse stress testing, to evaluating the impact on a portfolio distribution of increasing the probability of subportfolios' joint exceedances. The implemented analysis and visualisation tools help users derive insights into their models and perform formal comparisons of the importance of model components. Since **SWIM** does not require re-simulation from the model, these sensitivity analyses have a low computational cost; moreover, they can be performed on black-box models.

While working with a single set of simulated scenarios is computationally efficient, it also poses some limitations, as **SWIM** cannot currently consider states of the world outside those already generated by the simulation model. This places a constraint on how different the baseline and stressed model can be; technically speaking the latter must be absolutely continuous with respect to the former. We aim to address this limitation in the future, for example by allowing users to augment a given set of simulated scenarios with additional ones and/or to consider more than one set of scenarios as part of a single sensitivity analysis.

A different issue emanating from working with a limited set of scenarios relates to the impact of sampling error on estimates of stressed distributions and sensitivity measures. Substantial sampling errors may occur in cases where users specify stresses that are technically feasible, but concentrate a lot of weight on only a few scenarios. In future updates of the package we intend to implement relevant warnings, as well as bootstrap estimates of confidence intervals for key **SWIM** outputs.

Furthermore, future work includes enhancing analysis tools, for example functions that will make it easier to extract distributional characteristics of stressed models – e.g. a `stressed_cor` function that enables the monitoring of correlation changes when models are stressed.

Acknowledgements

This vignette was written in **R** (R Core Team, 2019) using the packages **SWIM** (Pesenti et al., 2020) and **bookdown** (Xie, 2019).

SP would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) with funding reference numbers DGECR-2020-00333 and RGPIN-2020-04289.

A Appendix Credit Model

A.1 Credit Model assumptions

The credit risk portfolio of Section 4 is based on the conditionally binomial credit model described in Section 11.2 of McNeil et al. (2015) which belongs to the family of mixture models. Specifically, we consider a portfolio that consists of three homogeneous subportfolios and denote the aggregate portfolio loss by $L = L_1 + L_2 + L_3$, with L_1, L_2, L_3 the losses of each subportfolio, given by

$$L_i = e_i \cdot \text{LGD}_i \cdot M_i, \quad i = 1, 2, 3, \tag{9}$$

where e_i and M_i are the exposure and number of defaults of the i^{th} subportfolio, respectively, and LGD_i is the loss given default of subportfolio i. M_i is Binomially distributed, conditional on H_i, a random common default probability. Specifically $M_i|H_i \sim Binomial(m_i, H_i)$, where m_i is the portfolio size. The H_is follow a Beta distributions with parameters chosen so as to match given overall unconditional default probabilities p_i and default correlations ρ_i, that is, the correlation between (the indicators of) two default events within a subportfolio, see McNeil et al. (2015). The dependence structure of (H_1, H_2, H_3) is modelled via a Gaussian copula with correlation matrix

$$\Sigma = \begin{pmatrix} 1 & 0.3 & 0.1 \\ 0.3 & 1 & 0.4 \\ 0.1 & 0.4 & 1 \end{pmatrix}. \tag{10}$$

Table 7 summarises the parameter values used in the simulation.

Table 7: Parameter values used in the simulation for the credit risk portfolio in Section 4.

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

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

# counterparties subportfolio 1, 2 and 3
```

```r
  m1 <- 2500
  m2 <- 5000
  m3 <- 2500

  # prob of default for subportfolios 1, 2 and 3
  p1 <- 0.0004
  p2 <- 0.0097
  p3 <- 0.0503

  # correlation between default probabilities
  rho1 <- 0.0004
  rho2 <- 0.0044
  rho3 <- 0.01328

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

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

# beta parameters: matching subportfolios 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 subportfolios
  cor12 <- 0.3
  cor13 <- 0.1
  cor23 <- 0.4

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

# 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)
```

```r
# simulate number of default per subportfolios (binomial distributions)
  M1 <- rbinom(n = nsim, size = m1, prob = H[, 1])
  M2 <- rbinom(n = nsim, size = m2, prob = H[, 2])
  M3 <- rbinom(n = nsim, size = m3, prob = H[, 3])

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

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

# the credit data included in SWIM
  credit_data <- cbind(L, L1, L2, L3, H)
  colnames(credit_data) <- c("L", "L1", "L2", "L3", "H1", "H2", "H3")
```

References

Beckman, R. J. and McKay, M. D. (1987). Monte Carlo estimation under different distributions using the same simulation. *Technometrics*, 29(2):153–160.

Borgonovo, E., Hazen, G. B., and Plischke, E. (2016). A common rationale for global sensitivity measures and their estimation. *Risk Analysis*, 36(10):1871–1895.

Borgonovo, E. and Plischke, E. (2016). Sensitivity analysis: a review of recent advances. *European Journal of Operational Research*, 248(3):869–887.

Breuer, T. and Csiszár, I. (2013). Systematic stress tests with entropic plausibility constraints. *Journal of Banking & Finance*, 37(5):1552–1559.

Cambou, M. and Filipović, D. (2017). Model uncertainty and scenario aggregation. *Mathematical Finance*, 27(2):534–567.

Csiszár, I. (1975). I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, pages 146–158.

Emmer, S., Kratz, M., and Tasche, D. (2015). What is the best risk measure in practice? A comparison of standard measures. *Journal of Risk*, 18(2):31–60.

Iooss, B., Janon, A., and Pujol, G. (2019). *Sensitivity: global sensitivity analysis of model outputs*. R package version 1.16.2.

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

Pesenti, S. M., Bettini, A., Millossovich, P., and Tsanakas, A. (2020). *SWIM: Scenario Weights for Importance Measurement*. R package version 0.2.0.

Pesenti, S. M., Millossovich, P., and Tsanakas, A. (2016). Robustness regions for measures of risk aggregation. *Dependence Modeling*, 4(1):348–367.

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.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S. (2008). *Global Sensitivity Analysis: The Primer*. John Wiley & Sons.

Tsanakas, A. and Millossovich, P. (2016). Sensitivity analysis using risk measures. *Risk Analysis*, 36(1):30–48.

Vallender, S. (1974). Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications*, 18(4):784–786.

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Weber, S. (2007). Distribution-invariant risk measures, entropy, and large deviations. *Journal of Applied Probability*, 44(1):16–40.

Xie, Y. (2019). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.12.