Exposure Spaces

Sam Culley, Bree Bennett, Seth Westra

5 September 2017

Contents

---

1 Introduction

A variable and changing climate presents significant challenges to the performance of a range of engineered systems across the municipal, agricultural, energy, mining, industrial and transport sectors. In many cases, the systems are expected operate satisfactorily under a range of climate regimes, however, it is critical to understand the conditions under which system performance might degrade so that alternative contingency measures can be put into place. A range of approaches have emerged in recent years to provide a vulnerability analysis for a system. These include bottom-up approaches (e.g. Prudhomme et al. 2010), as well as decision-centric approaches for managing climate uncertainty (e.g. Brown 2011, Culley et al. 2016, McPhail et al. 2017). In general, a vulnerability analysis should aim to:

- Treat the system as the central concern of the analysis;
- Recognise systems are inherently complex and links with climate are often 'non-trivial';
- Emphasise the importance of system understanding by 'stress-testing' systems against a range of hypothetical and projected climate states;

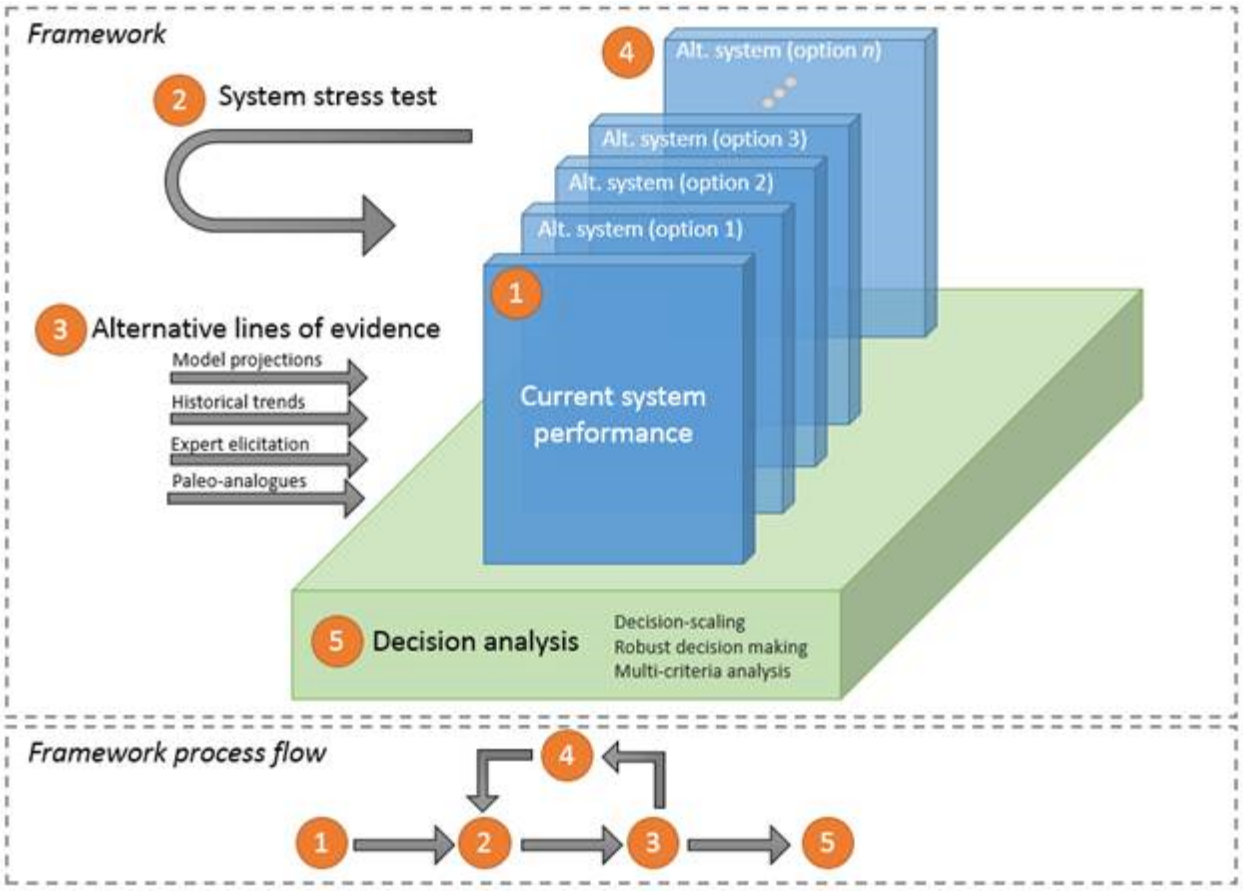
- Provide a basis for iterative dialogue between decisions makers, system modelers and climate experts about model uncertainty and the implications of different design and operation options;
- Allow exploration of the implications of deep uncertainty by combining climate projections (via top-down approaches) with hypothetical climate scenarios (via bottom-up approaches) that cover a wider range of possible climatic changes;
- Enable rapid update of impact assessments under new lines of evidence (e.g. if new climate model results become available);
- Provide a basis for adaptive planning (including adaptive pathways).

1.1 General analysis framework

We have developed a general framework, which draws from those presented in climate change impact assessment literature that will ensure the above aims are achieved. The framework contains the following elements:

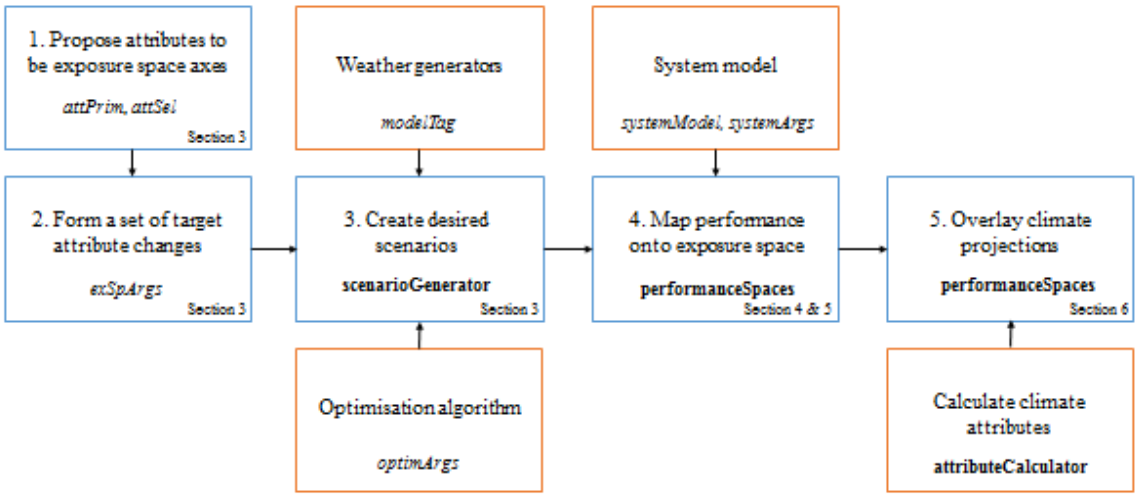
- A climate 'stress test' is applied to both the current system design and a set of alternative system designs, to assess the rate of system performance degradation and/or identify situations under which systems can fail;
- 'System performance' can be defined in a number of ways, including binary success/failure criteria or quantitative performance measures, and across multiple economic, social and/or environmental measures;
- Multiple lines of evidence can be applied to understand possible future climate changes, including climate model projections (by combining global climate models with dynamical and/or statistical downscaling, or bias corrections), historical climatic changes, expert judgement and/or analogues from paleo records; and
- Decision analysis can proceed in multiple ways, depending on user preference and interpretation of climate uncertainty.

In doing so, the system combines bottom-up style system 'stress testing' with projections from multiple lines of evidence, and combines traditional climate impact assessments with an analysis of multiple system configurations (or 'options') to identify adaptation options or pathways. Furthermore, decision-centric approaches can be tailored depending on whether a user interprets climate projections probabilistically (in which case approaches such as cost-benefit analyses and/or quantitative risk assessments may be appropriate) or as scenarios (in which case robustness approaches may be required).



1.2 Overview of this package

This package is a tool to be used in many stages of the larger climate impact anlaysis described above. Its core functionality is to create climate scenarios, and then simulate and visualise system performance. This functionality is useful in the exploratory stage of an assessment, where the goal is to identify system vulnerabilities. It is also a necessary tool to create a final description of system vulnerabilities. The process behind the functionality is show below, with key function arguments in italics and functions themselves in bold.



The main function of the package is to aid the development of scenario neutral spaces. Section 3 describes this process, introducing the necessary arguments. First, you must specify the climate attributes you wish to change, and the attributes you wish to monitor but hold constant. Next, you detail how the attributes are changed using *exSpargs*. This will construct a list of target scenarios - time series of climate variables - that will be produced using the **scenarioGenerator** function.

Creating scenarios requires two modules, shown above in orange: a weather generator and an optimisation algorithm. This package includes a selection of weather generators but external models can be implemented instead. The same goes for the optimisation algorithm - this package incudes a genetic algorithm but others can be used in its place.

The other key functionality of the package is to visualise the exposure spaces. This requires system response to be attached to each scenario. This can be completed using an internal model using a wrapper function in R, or alternatively, the scenarios can be modelled with an external model and supplied as a response vector. Both of these processes are described in section 4.

The performance should then be mapped on to the exposure space to visualise the system response. This package supports many graphical options, including heatmaps, contour plots, and pass/fail plots, all used by the **performanceSpaces** function. Section 5 details these options, as well as controls over the final visualisations such as plot titles, axis labels and legends.

The final stage of this process is to add other information to the plots. This typically involves climate change projections, which are mapped onto the response surface to examine how plausible the system vulnerabilities are. This information is added using the **performanceSpaces** function, and the necessary arguments are detailed in section 6.

---

2 An example problem

To demonstrate the package's functionality, an example system model and accompanying dataset called 'tank' is used. 'Tank' is a domestic rainwater tank model with enough complexity to highlight some of the key

features of this package. The dataset and model can be loaded from the package at any time using the data command: {r data(tank)}

The tank model simulates a domestic rainwater tank system, which can be used for both indoor (grey water) applications and garden irrigation. Rain falling on the roof of the house is captured and directed towards the rainwater tank. Before the rainwater is able to enter the tank the first flush is removed from the start of each storm. This removed amount equates to the first millimetre of water falling on the roof and is required for water quality reasons. The water remaining after the first flush extraction flows into the rainwater tank. Water demand is assumed to be constant throughout the year for indoor use, and vary seasonally for outdoor use. The amount of water supplied by the tank depends on the current water level in the tank calculated at a daily time step.

The outdoor seasonal demand pattern responds to the daily temperature (i.e. on hot days, above 28 degrees, the gardener waters more than the seasonal average, but on cool days , below 10 degrees, the gardener waters less than the seasonal average). The tank model simulates each stage of the rainwater capture and use process based on the supplied daily rainfall and temperature time series. The size of both the tank and roof can be varied in the model. Performance is measured according to four metrics:

- Reliability - the fraction of days on which the full demand could be supplied
- Volumetric reliability - the total water supplied as fraction of the total demand
- System efficiency - the amount of water used as a percentage of the water captured by the roof
- Storage efficiency - the amount of water spilled as a percentage of the water captured by the rainwater tank

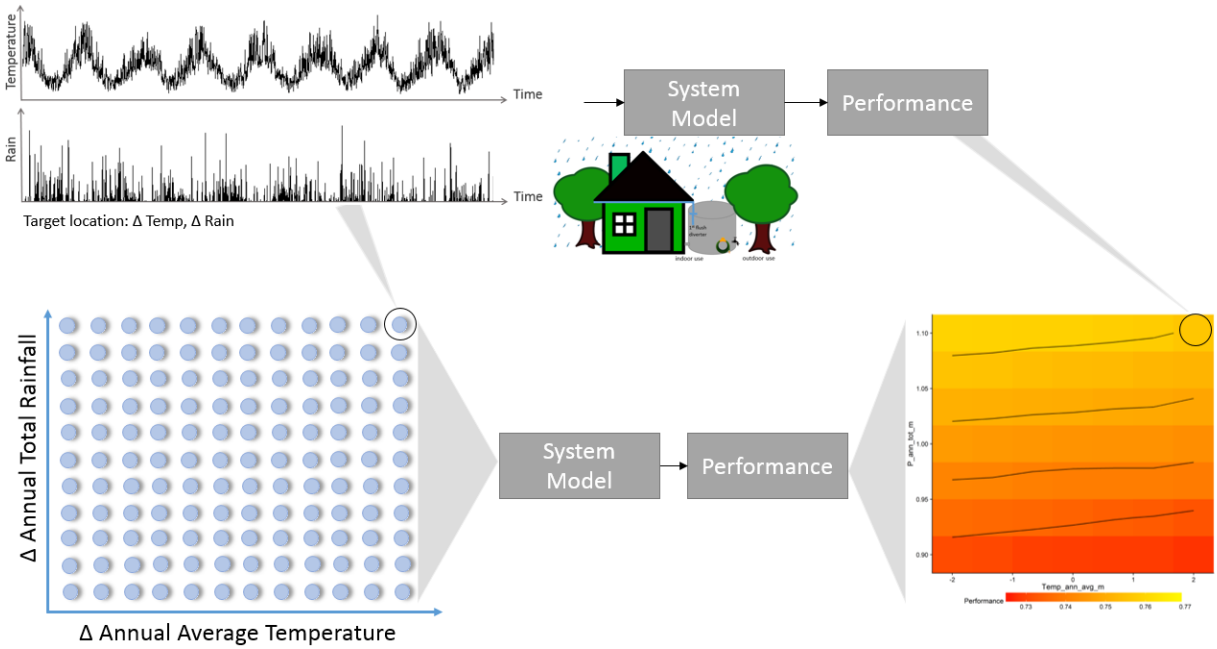
This example model provides sufficient scope for investigation. This is because the tank responds to multiple climate drivers (i.e. rainfall and temperature), and the removal of the first flush volume at the start of the storm mean that the wet-dry pattern of the rainfall and the seasonality of the demand pattern may become important.

A function call to the tank model is shown below. The call returns the system performance which is stored in the object 'performance'. How to use your system model in conjunction with the package will be discussed later.

```
performance<-tankPerformance(data=data,        # Climate time series data
                             roofArea=100,     # Roof area in m
                             tankVol=3000)     # Tank volume in L
```

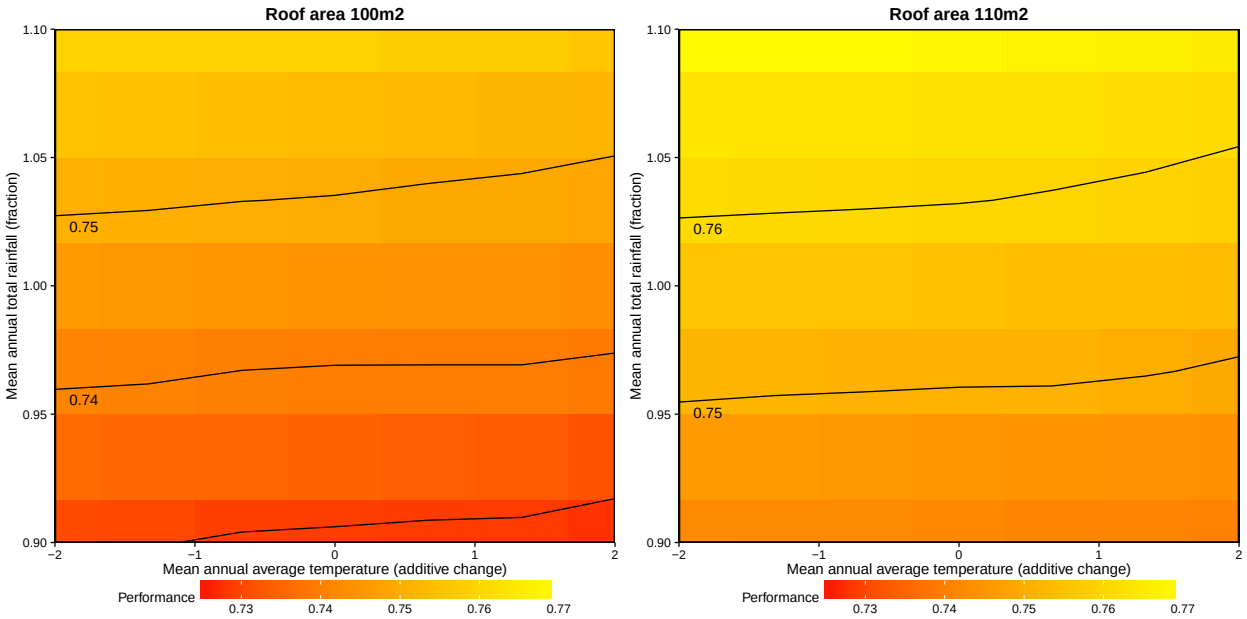
Now that we have a system model we can start 'stress-testing' the system's performance under a range of plausible climate scenarios using the scenario-neutral approach.

The scenario-neutral approach proceeds by first generating an exposure space. This exposure space is composed of target locations (attribute combinations) that are of interest. Each target location is a set of climate variable time series (e.g. temperature and rainfall). Each of these sets of climate variable time series are then run through the system model and the resulting performance stored in a performance space. A schematic diagram of the process of generating a performance space is shown below.



Over the course of this vignette tutorial you will learn to produce a whole exposure space and in turn a performance space.

Once you are comfortable producing performance spaces you can then go on to evaluate and compare different system configurations. For example, you can compare different tank sizes for the same property, or perhaps compare different contributing roof areas feeding into the same size tank (see below). A comparison of two performance spaces (contributing roof area of 100 and 110 m2) shows the impact roof area has on the reliability performance metric. The 10% increase in the contributing roof area increases reliability across the space, but the system still responds to changes in climate in a similar way (e.g. the system is still more sensitive to changes in annual total rainfall than annual average temperature).



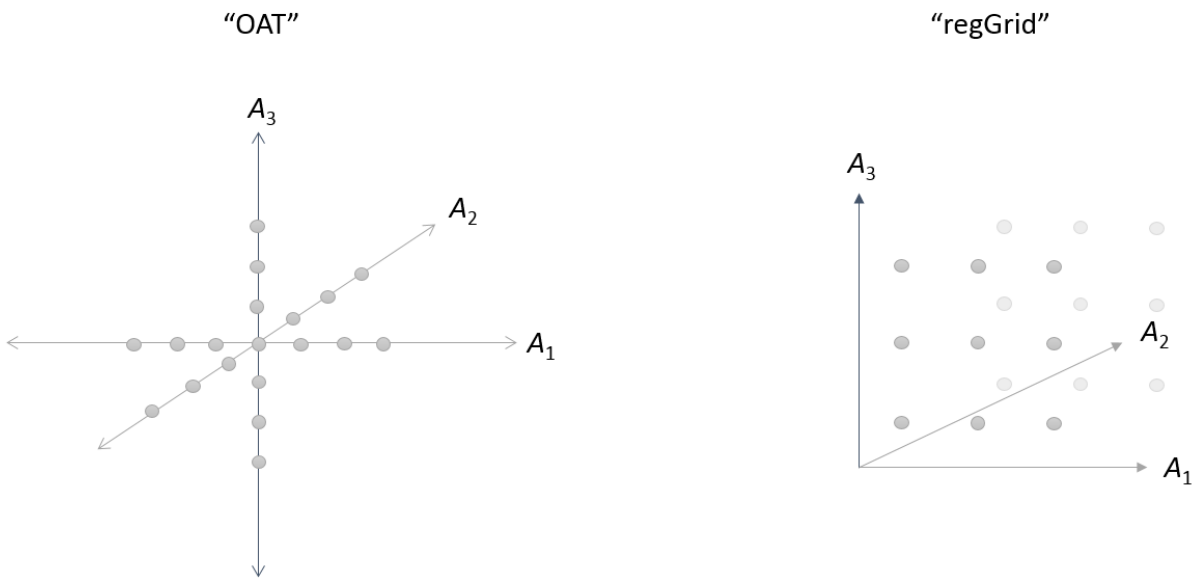
3 Creating exposure spaces

This section introduces the **scenarioGenerator** function, which is used to take a historical climate record and perturb it to have particular characteristics for 'stress-testing'. These characteristics of the climate variable time series are termed attributes. The **scenarioGenerator** function requires a number of arguments to specify which attributes should be perturbed and which should be held constant, the combinations of which create a list of target scenarios that together, build an exposure space. Section 3.1 explains these arguments, and section 3.2 details the options for creating the target scenarios.

3.1 Specifying an exposure space

Each call of the **scenarioGenerator** function begins by calculating attributes on provided historical data. These values are used as a baseline for further scenario creation. There are two arguments that are used to specify which attributes should be calculated, and which one you should use depends on plans for the exposure space.

exSpArgs is then used to specify the size of the exposure space. *exSpArgs* contains three aguments itself; *type*, *samp* and *bounds*. *bounds* controls the range each attribute should be perturbed over, and *samp* specifies the number of discrete points in that range. *type* sets the type of exposure space constructed from the perturbations, with two currently supported options: one at a time (oat) and regular grid (regGrid).

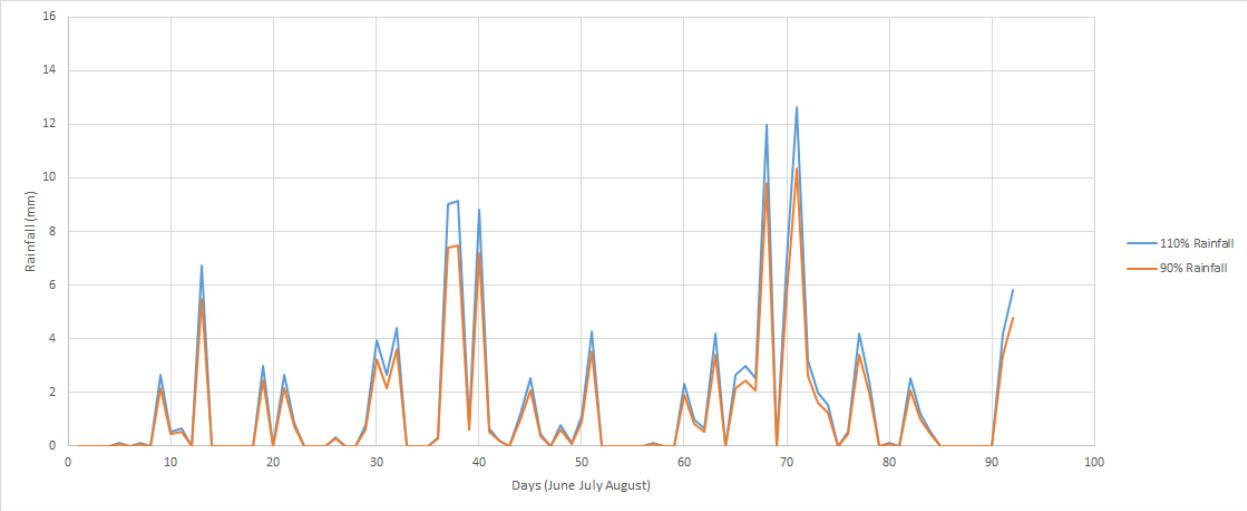


3.2 Creating an exposure space

This package provides two techniques for creating exposure spaces: simple scaling and stochastic simulation (inverse approach). Simple scaling applies an additive or multiplicative change to the historical time series to create perturbed time series, whereas stochastic simulation (inverse approach) uses a stochastic weather generator to produce perturbed time series with the required characteristics for 'stress-testing'. The production of perturbed time series via simple scaling and stochastic simulation (inverse approach) are introduced separately.

3.2.1 Simple scaling

Simple scaling applies an additive or multiplicative change to the historical time series to create perturbed time series. As a result simple scaling is limited in the variety of attributes it can perturb. This is due to the scenario creation method which relies on the historical time series pattern. Therefore simple scaling cannot be used to implement certain changes to the time series. For example, by applying a multiplicative change to a rainfall time series the overall wet-dry pattern (e.g. wet-spell duration, number of wet days) will not change. The maintenance of the overall rainfall time-series pattern is illustrated below by comparing two different scenarios created with this technique, one with 10% less rainfall in the year and the other with 10% more. Additionally, as the same change is applied to all days of the time series it is not possible to evaluate what would happen if only certain parts of the rainfall distribution where to change (e.g. if the rainfall intensity on extreme days became more extreme).



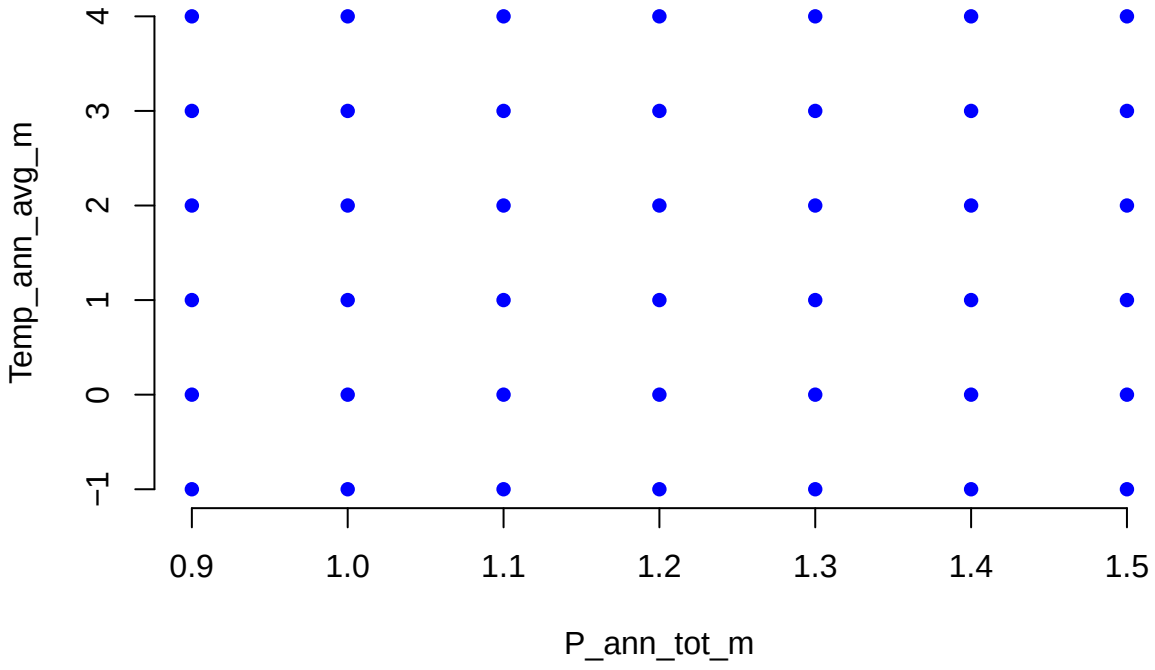
3.2.1.1 Supported attributes

Due to the inability of simple scaling to change the underlying pattern in the historical time series, only a few attributes can be selected for perturbation in the package. The attributes that can be created by simple scaling are:

- P_ann_tot_m - an annual total of rainfall (mm)
- Temp_ann_avg_m - the average annual temperature (C)
- PET_ann_avg_m - the average annual potential evapotranspiration (mm)

3.2.1.2 Perturbed spaces

So let's generate and exposure space using simple scaling. For example, let's generate an exposure space for both temperature and precipitation. For this test you wish to vary temperature from -1 degrees to 4 degrees in 1 degree increments, and precipitation from 90% of historical average through to 150% of historical average in 10% increments, as illustrated below.



To generate this space you would set up a call like the following.

```
#Scenario generation arguments

modelTag="Simple-ann"

attSel<-c("P_ann_tot_m","Temp_ann_avg_m")

exSpArgs<-list(type = "regGrid",
               samp = c(7,6),
               bounds = list("P_ann_tot_m"=c(0.9,1.5),
                             "Temp_ann_avg_m"=c(-1,4)))

#Function call

scenarioGenerator(obs=obs,
                  modelTag = modelTag,
                  attSel=attSel,
                  exSpArgs = exSpArgs)
```

As shown above a simple scaled exposure space can be created using four arguments in the **scenarioGenerator** function. Each of the scenarioGenerator arguments will now be explained in turn.

The first argument is the historical data, *obs*, which needs to be prepared in a specific data frame format (see below). The dates need to be separated into three columns: year, month, day. After the date columns the climate data can be entered in any order, but with the headings P, Temp or PET as appropriate. An example of an appropriately set up data frame is shown below.

```
##   year month day   P  Temp
## 1 2007     1   1 0.0 25.50
## 2 2007     1   2 0.0 24.50
## 3 2007     1   3 0.0 29.75
## 4 2007     1   4 0.0 32.25
## 5 2007     1   5 0.0 32.50
## 6 2007     1   6 4.5 26.50
```

The second argument is *modelTag*. This argument dictates which weather generator blocks will be used. For simple scaling cases, *modelTag* must be set to "Simple-ann".

The third argument, *attSel*, is a vector of the attributes selected for perturbation. These will also form the each axis of the exposure space. Remember, only annual averages (or annual totals when they apply, as the actions are the same) are supported for simple scaling.

Finally, the *exSpArgs* argument is used to control the target sampling. *exSpArgs* is a list of all the arguments relating to the geometric layout of the exposure space. It contains *type*, *samp* and *bounds* arguments. The *type* argument is used to set the type of sampling, which is in this case a uniform grid. This has the tag "regGrid". The *bounds* argument is used to set an axis range for each attribute. Its format is therefore required to be a list, with every attribute in attSel given *bounds* explicitly. For temperature you need to specify the change in degrees; however, for rainfall or potential evapotranspiration changes should be specified as fractions of the historical climate attribute value. The *samp* argument specifies the number of samples to take inside the bound range, inclusive of the limits themselves (see above example).

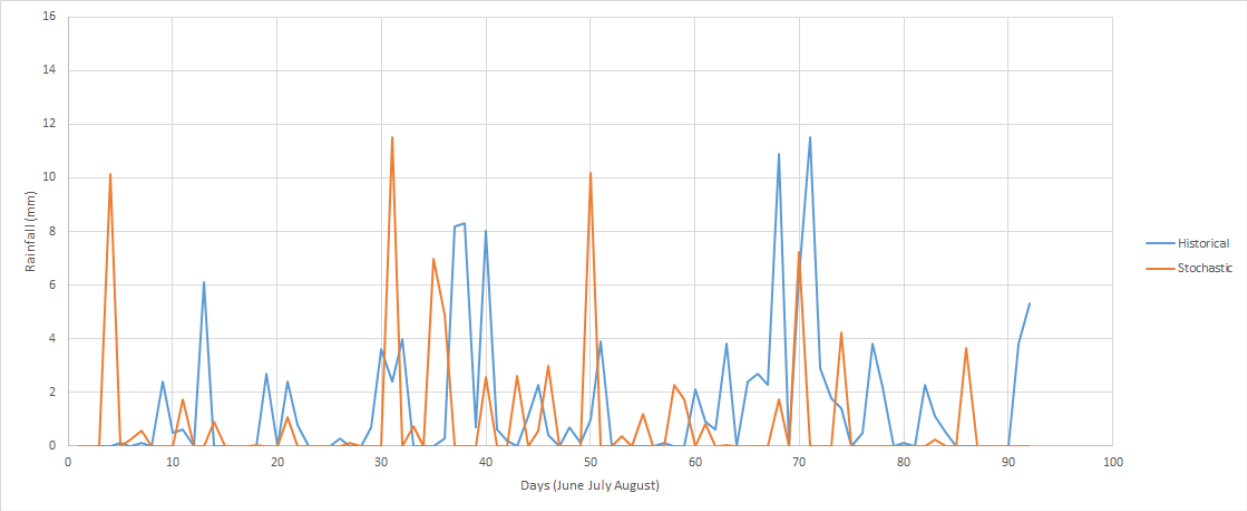
The perturbed time series produced using the **scenarioGenerator** function are outputted as both .CSV and RData files.

3.2.2 Stochastic series (inverse approach)

Stochastically generated perturbed time series are produced using an inverse approach (Guo et al., 2016a) whereby the target location (the combination of attributes) are specified first and an optimisation approach is used to determine the parameters required to simulate that specified target using a stochastic model.

Creating scenarios using stochastic weather generators is useful in that one aspect (attribute) of a time series can be changed, while still maintaining other attributes at historical levels. This property provides a powerful tool for creating exposure spaces that cover a wide range of plausible climate scenarios. This is a fundamental feature of the scenario-neutral approach. The approach can also be used to target specific changes to the rainfall pattern.

Additionally, by using a stochastic weather generator, the time series can be created with a length longer than historical data. Using the stochastic approach the system's performance can be evaluated over longer time periods - even under historical conditions. For example, below is a historical rainfall record compared to a stochastically generated rainfall series with the same attributes as the historical.



3.2.2.1 Supported attributes

The attributes that can be selected for perturbation using the stochastic weather generation option in this package are listed below.

- P_ann_tot_m - an annual total of rainfall (mm)
- P_ann_R10_m - an annual count of days over 10mm of rainfall (days)
- P_ann_maxDSD_m - an annual count of the maximum duration of consecutive dry days (days)
- P_ann_maxWSD_m - an annual count of the maximum duration of consecutive wet days (days)
- P_ann_P99_m - the 99th percentile day of rainfall (including zero days)(mm)
- P_ann_dyWet99p_m - the 99th rainfall day of annual wet days (mm)
- P_ann_ratioWS_m - an annual ratio of winter to summer rainfall
- P_ann_dyWet_m - the annual wet day average (mm)
- P_seas_tot_cv - a numeric vector
- P_mon_tot_cv - a numeric vector
- P_ann_avgWSD_m - an average of the annual dry spell durations (days)
- P_ann_avgDSD_m - an average of the annual wet spell durations (days)
- P_JJA_avgWSD_m - an average of the June-July_August wet spell durations (days) (Same for periods of MAM,DJF,SON)
- P_JJA_avgDSD_m - an average of the June-July_August dry spell durations (days) (Same for periods of MAM,DJF,SON)
- P_JJA_dyWet_m - an average of the June-July_August wet day amounts (mm) (Same for periods of MAM,DJF,SON)
- P_JJA_tot_m - a total of the June-July_August wet day amounts (mm) (Same for periods of MAM,DJF,SON)
- Temp_ann_avg_m - the average annual temperature (C)
- Temp_ann_P5_m - the annual 5th percentile temperature (C)
- Temp_ann_P95_m - the annual 95th percentile temperature (C)
- Temp_ann_F0_m - an annual count of the frost days (temperature less than zero) (days)
- Temp_ann_GSL_m - an annual count of the growing season length (NH)(days)
- Temp_ann_CSL_m - an annual count of the cold season length (SH)(days)
- Temp_ann_rng_m - a measure of difference between Temp_ann_P95_m and Temp_ann_P5_m (C)

3.2.2.2 Creating stochastic exposure spaces

Now let's generate and exposure space using a stochastically simulated rainfall. For example, you would like to vary the annual total rainfall between 90% of the historical average through to 110% of the historical

average in 10% increments. You would also like to vary the 99th percentile rainfall between 90% of the historical average through to 110% of the historical average in 10% increments. However, you want to keep the maximum dry spell duration at historical levels. To produce this exposure space you have decided to use a weather generator that has a constant set of parameters across the whole year.

To generate this space you would set up a call like the following.

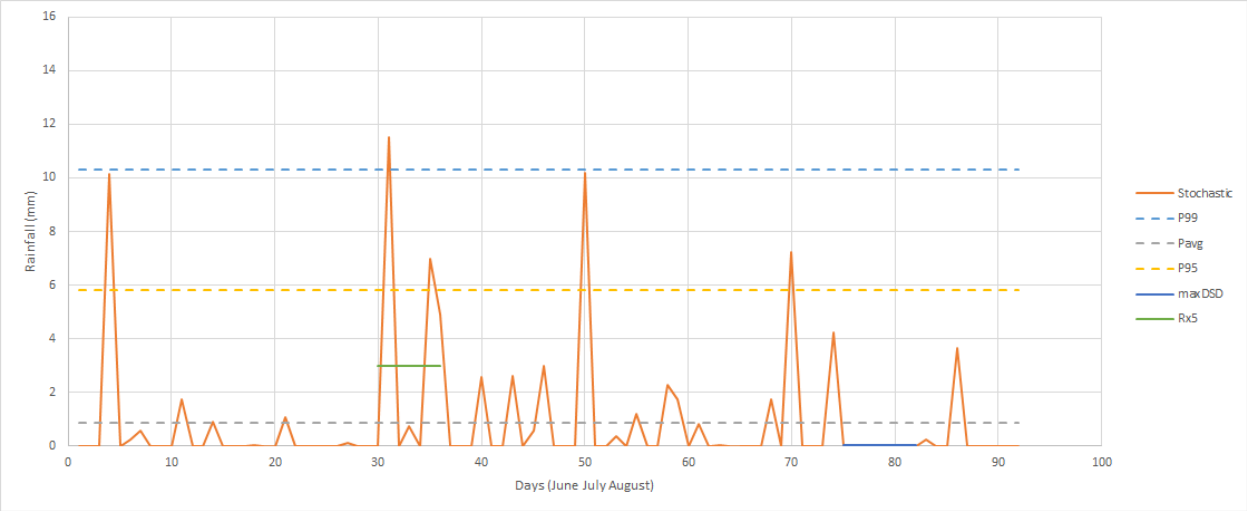
```
#Scenario generation arguments

modelTag=c("P-ann-wgen")

attSel<-c("P_ann_tot_m","P_ann_P99_m","P_ann_maxDSD_m")

attPrim<-c("P_ann_tot_m","P_ann_P99_m")

exSpArgs<-list(type = "regGrid",
               samp = c(3,3,1),
               bounds = list("P_ann_tot_m"=c(0.9,1.1),
                             "P_ann_P99_m"=c(0.9,1.1),
                             "P_ann_maxDSD_m"=1))

#Function call

scenarioGenerator(obs=obs,
                  modelTag = modelTag,
                  attSel=attSel,
                  attPrim=attPrim,
                  exSpArgs = exSpArgs)
```

As shown above the stochastic exposure space is generated using the same function, **scenarioGenerator**. But this call to the **scenarioGenerator** function requires a little more instruction and so has five arguments.

exSpArgs is constructed the same way as in the simple scaling demonstration. However, *modelTag* is now used to specify the annual weather generator ("P-ann-wgen"). Another change to note is to *attSel*. *attSel* is now longer than the simple scaling case, for a few reasons. Firstly there are more changes to climate that can be targeted at once, and there is now a need to specify more attributes that should be kept at historical levels. A new argument, *attPrim*, is used to specify the attributes that form the axes of the space (e.g. attributes that are perturbed). The primary attribute list should be a subset of *attSel*, so ensure they are listed in both.

To hold an attribute constant, it can be included in *attSel*, and then included in *exSpArgs* but with a sample number of 1. This will tell the package that this attribute isn't to be sampled or included in the space, but held at historical levels as scenarios are created. Additionally, instead of including bounds, the attribute has no range as is kept at historical conditions.

As the pattern is no longer fixed, there are many attributes of the time series that can change freely, unless specified as a part of the target. For example, a rainfall time series has many inherent attributes - 99th percentile daily rainfall (P99), 95th percentile daily rainfall (P95), average daily rainfall (Pavg), maximum dry-spell duration (maxDSD), maximum 5-day rainfall totals (Rx5) and many more (see figure below). So if you are stochastically generating rainfall and only specify a change in annual average rainfall many other attributes may also change as a result. Your system model might be sensitive to changes in some of these attributes so the choice of attributes (to both perturb or hold constant at historical levels) need to be considered carefully. Later on we will look at ways to evaluate how sensitive your system is to different attributes.



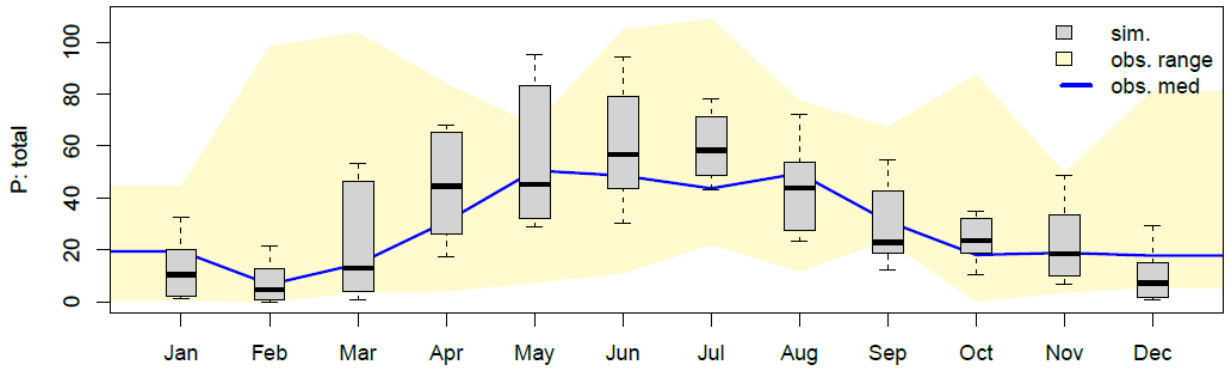
To investigate how the perturbed climate time series are looking you can check the metrics diagnostic PDF file created for each target. Each file is labelled using the target location. For example, 1Panntotm_0.9PJJAdyAllm_1PannP99m.pdf is the summary file for the target set at historical levels for annual total rainfall and 99th percentile daily rainfall amounts and 90% of the historical average daily rainfall in June-July-August.

The generated diagnostic plots include:

- a traffic light report card that summarises how well the stochastic generator has simulated the requested target,
- plots of simulated time series for each hydroclimate variable, and
- plots of the time series characteristics for different temporal aggregations (i.e. monthly, seasonal and annual) summarising daily mean values, standard deviation of daily values and totals for each simulated hydroclimate variable. If the simulated variable is rainfall the number of wet days, wet day amount means and standard deviations are also plotted. The historical climate levels are included on these plots for comparison.

The first page of the diagnostics file summarises the model run so that a clear record of the simulation arguments is attached to the diagnostics (e.g. attributes, primary attributes, model selected). The second page presents a traffic light report card that provides a quick visual summary of how close the stochastic simulation has got to the requested target location. An example of the traffic light report card is shown below. Green indicates 'good' performance (e.g. between +/- 5% difference from target), yellow indicates 'fair' performance (e.g. greater than +/- 5% difference from target but less than +/- 10% difference from target), and red indicates 'poor' performance (e.g. greater than +/- 10% difference from target).

The remaining pages of the diagnostics file show a comparison of the simulated time series at different aggregations compared to the historical climate. This enables you to visually determine whether an unexpected change in the time series characteristics has occurred. For example, a comparison the simulated monthly total rainfall for a selected target against observed levels is shown below.



3.2.2.3 Stochastic spaces continued

The R package uses 'Richardson-type' weather generator model configurations (Richardson, 1981) to stochastically generate rainfall, temperature and potential evapotranspiration at a daily time step.

In the above example, the exposure space created only evaluated changes in rainfall, and as a result only one *modelTag* entry was required. For a multiple climate variable exposure space, the *modelTag* argument will become a vector of model tags corresponding to each simulated climate variable.

This package supports multiple weather generator options for each climate variable. The full list is detailed below.

- "Simple-ann" - for simple annual scaling
- "P-ann-wgen" - for a four parameter annual rainfall model
- "P-seas-wgen" - for a 16 parameter seasonal rainfall model
- "P-har26-wgen" - for a harmonic rainfall model
- "P-2har26-wgen" - for a double harmonic rainfall model
- "Temp-har26-wgen-wd" - for a harmonic temperature model dependent on wet or dry day

- "Temp-har26-wgen" - for a harmonic temperature model not conditional on rainfall

The selection of a particular *modelTag* will change which attributes can be selected for perturbation. For example, an annual rainfall model (e.g. P-ann-wgen) does not allow the winter total rainfall attribute (P_JJA_tot_m) to be perturbed as there are no mechanisms for change. As a result the package will not allow you to select this attribute when using the annual model. The list of supported attributes for each model can be found in the documentation for *attSel*.

It should be noted that as the models become more complex they can perturb the climate variables in more ways but will also require more attributes to be held constant.

Now let's make an exposure space that investigates change in two different climate variables. This time we need to include two modelTags (one each for the two climate variables). *attSel* contains all our attributes (both rainfall and temperature). The primary attributes, *attPrim*, include one from each variable (these become the axes). Note the 5th and 95th percentile temperature attributes are also included - this is to ensure the temperature time series keeps the same range as the mean temperature changes.

```
#Scenario generation arguments

modelTag=c("P-ann-wgen","Temp-har26-wgen")

attSel<-c("P_ann_tot_m","Temp_ann_avg_m","Temp_ann_P5_m","Temp_ann_P95_m")

attPrim<-c("P_ann_tot_m","Temp_ann_avg_m")

exSpArgs<-list(type = "regGrid",
               samp = c(7,6,1,1),
               bounds = list("P_ann_tot_m"=c(0.9,1.5),
                             "Temp_ann_avg_m"=c(-1,4),
                             "Temp_ann_P5_m"=1,
                             "Temp_ann_P95_m"=1))

#Function call

scenarioGenerator(obs=obs,
                  modelTag = modelTag,
                  attSel=attSel,
                  attPrim=attPrim,
                  exSpArgs = exSpArgs)
```

3.2.2.4 One at a time testing

Now that you have the skills to generate exposure space you need to make some decisions on what attributes to include in your exposure space. Let's have a look at how sensitive our tank system is to a range of attributes.

This package provides a simple method for doing so - a one at a time scenario creation mode. The goal here is to just change one attribute of a scenario while holding all others at historical levels. By just focusing on one attribute at a time we can get a general measure of how sensitive the system is to changes in this selected attribute. All attributes to be investigated should be included in *attSel*, and *exSpArgs* can be used to specify the range and interval.

```
modelTag=c("P-ann-wgen")

attSel<-c("P_ann_tot_m","P_ann_P99_m","P_ann_nWet_m")
```

```
exSpArgs<-list(type = "OAT",
               samp = c(5,5,5),
               bounds = list("P_ann_tot_m"=c(0.7,1.3),
                             "P_ann_P99_m"=c(0.7,1.3),
                             "P_ann_nWet_m"=c(0.7,1.3)))

scenarioGenerator(obs=obs,
                  modelTag = modelTag,
                  attSel=attSel,
                  attPrim=attPrim,
                  exSpArgs = exSpArgs)
```

---

4 Simulating performance (stress-testing)

Now you have a fully simulated exposure space you can stress-test your system to see how it responds to changes in the climate. The results of your stress-test are termed a performance space as it contains the system performance at each target location in your exposure space.

This package has two ways of considering system impact: the first is to provide an R function that calls the models to obtain system performance, and the second is to simulate the performance space externally (perhaps your system model runs through a gui), and then provide the system performance as an additional data set and then use the package to produce performance space plots and diagnostics.

This section will take you through using the embedded system model 'tank' to determine system performance and then demonstrate how you can incorporate your own system model function.

4.1 Embedded system model

Providing the system model as a function in R is the simplest way to get the complete bottom up analysis, as the whole process can be automated using the initial function call. Even if the models have been developed in other languages, they could still be executed with an R wrapper function.

To work inside the package functions your system model function must be adhere to a particular format. These format conditions are:

- The model must be a functional, in that it provides one performance value at a time. Given the variations in models across case studies, it is likely that a wrapper function will be needed even if the models are in R, to meet the required format for the package.
- The model must accept the same 'obs' input format as the historical data, as scenarios will be produced in the same way. The package will produce scenarios to match this format also, allowing for repeat assessment of performance under different climates.
- The model function must use the following arguments:
  - *data* - climate time series data with columns year, month, day, P and Temp (e.g. same format as obs)
  - *systemArgs* - a list containing the all the relevant system properties (e.g. roof area, tank volume, metric to be reported).

While many parameters can be set in the model, the use of external arguments will allow for easy changes to system operation, design parameters or performance metrics. An example function wrapper is shown below:

```
systemArgs<-list(roofArea=50,                    #System arguments for tank model example
                 nPeople=1,
```

```
                  tankVol=3000)

tankWrapper<-function(data=NULL,                #Two arguments, the data file and a list of system argume
                      systemArgs=NULL
) {
  performance<-tankPerformance(data=data,
                               roofArea=systemArgs$roofArea,
                               nPeople=systemArgs$nPeople,
                               tankVol=systemArgs$tankVol)

  return(performance)                           #Return one value each simulation
}
```

4.2 External system model

The system response can also be provided manually, if keeping the system models separate is more convenient. In this case, the **scenarioGenerator** function is first used to create the scenarios. One of the outputs of the **scenarioGenerator** function is *target*, a list of the scenario targets in order of their creation. It can be seen in the following example:

```
output<-scenarioGenerator(obs=obs,
                          modelTag = modelTag,
                          attSel=attSel,
                          attPrim=attPrim,
                          exSpArgs = exSpArgs)

head(output$target)
```

```
##   P_ann_tot_m Temp_ann_avg_m
## 1         0.7             -2
## 2         0.8             -2
## 3         0.9             -2
## 4         1.0             -2
## 5         1.1             -2
## 6         1.2             -2
```

The way to input performance from an external model is to create a vector of performance measures in the same order as the target list. It can become an argument in the functions to create performance spaces, which are detailed in the next section.

5 Mapping Performance

Now you have the methods to fully simulate a performance space you can look at the mapped performance of your system across change in different attributes.

This section will look at visualising the performance space and interpreting performance across the space, using the package functions **scenarioGenerator** and **performanceSpaces**.

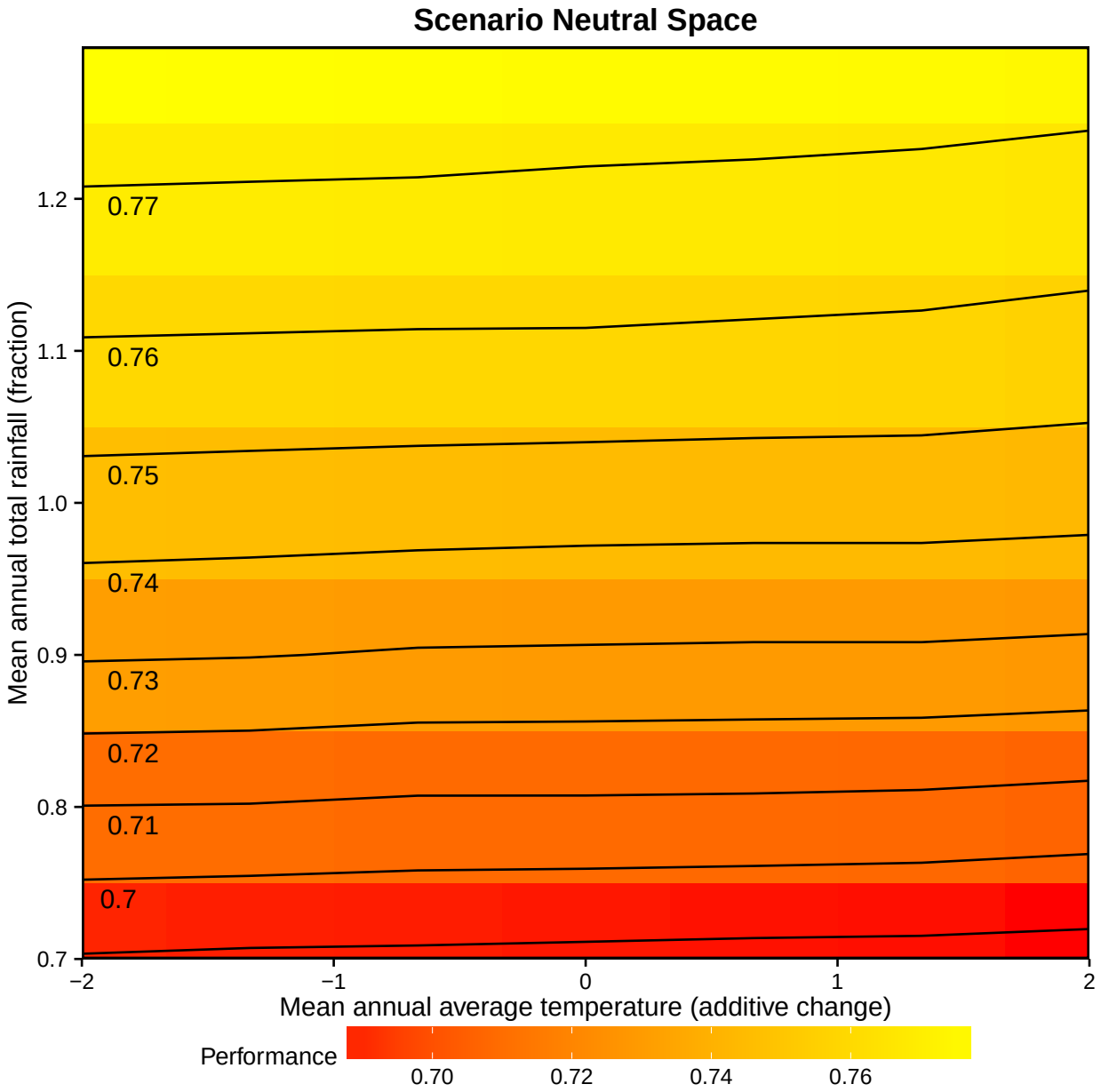
5.1 Using the main scenarioGenerator function

If you are using an internal system model, it is possible to fully automate the process of making an exposure space and mapping performance on top. This can be done by adding the internal system model functions to

the main code, like below.

```
scenarioGenerator(obs=obs,                          #Arguments for creating the exposure space
                  modelTag = modelTag,
                  attSel=attSel,
                  attPrim=attPrim,
                  exSpArgs = exSpArgs,
                  systemModel = tankwrapper,        #Arguments for simulating performance, as described i
                  systemArgs = systemArgs)
```

This call to the function will simulate the scenarios for an exposure space and then plot the default performance map, which is displayed as a heatmap.



5.2 Using the performanceSpaces function

The above method of creating performance spaces is limited to producing heatmap plots, with system performance being the only overlay. It also requires an internal system model, to create scenarios and simulate performance all in one run. If more plotting options are needed, then the **performanceSpaces** function should be used in combination with the **scenarioGenerator** function, as in the below example.

```
output<-scenarioGenerator(obs=obs,                          #Arguments for creating the exposure space
                  modelTag = modelTag,
                  attSel=attSel,
                  attPrim=attPrim,
                  exSpArgs = exSpArgs)


performanceSpaces(data=output,                              #Results of the scenarioGenerator function
                  plotTag="Heat",
                  systemModel = tankWrapper,
                  systemArgs = systemArgs)
```

The first argument for the **performanceSpaces** function is *data*, which takes in the output of the **scenarioGenerator** function. This includes all the climate data for the scenarios, as well as many of the original arguments; *attSel*, *attPrim* etc.

The second argument is *plotTag*, which specifies the type of plot. In addition to "Heat" for heatmaps, support model types are:

- Contour plots ("Contours"),
- A single pass/fail space ("Binary"),
- One at a time sensitivity testing plots ("OAT")

Plotting options such as titles, axis labels and data limits can be input with the third argument *plotArgs*. This is a list that can accept many entries, some of which are shown below. The full supported list can be found in the documentation.

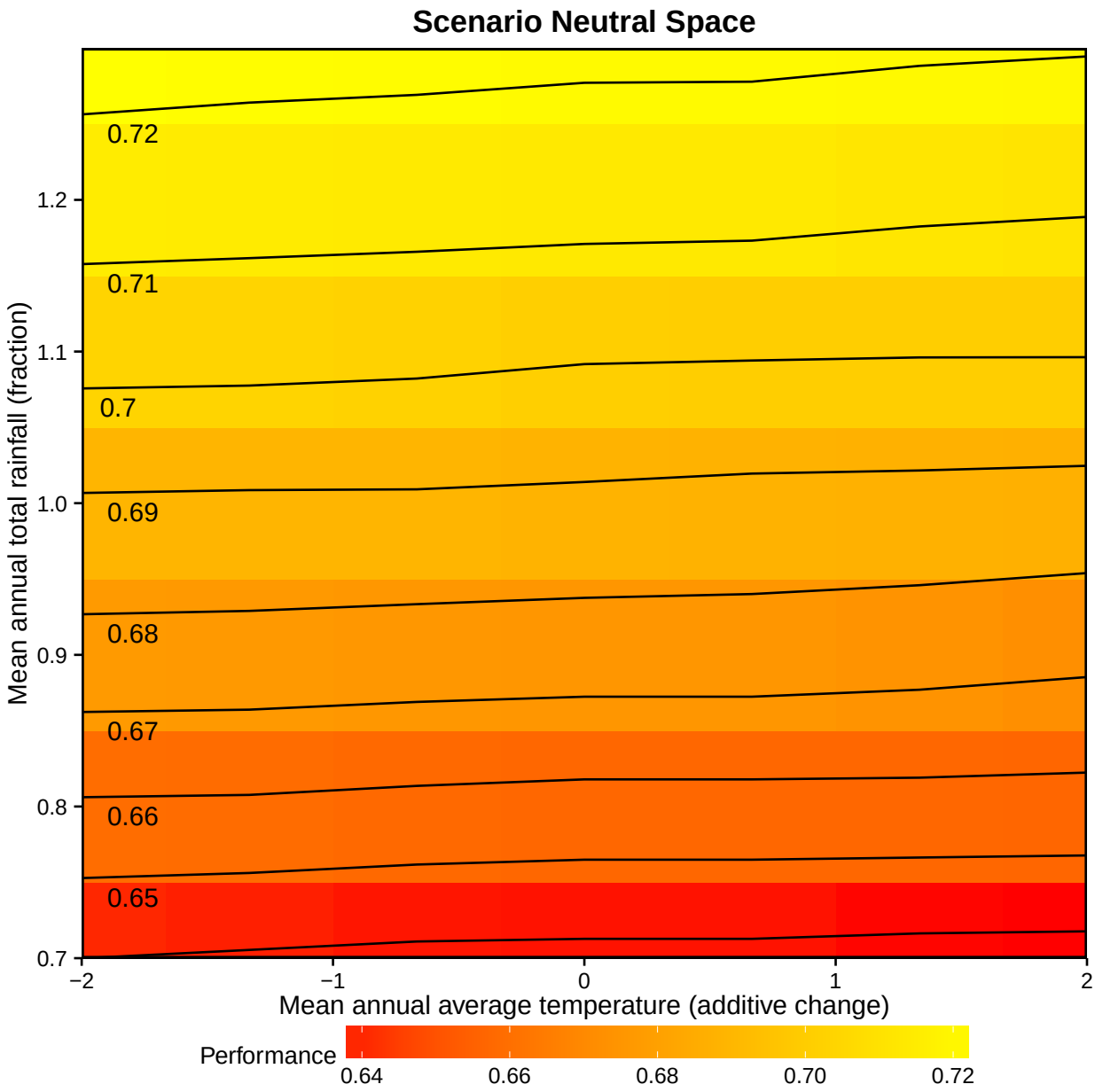
```
plotsArgs=list(title="Scenario Neutral Space",
               legendtitle="Reliability",
               xlim=c(-2,2),
               ylim=c(0.7,1.3),
               performancelimits=c(0.6,0.85))
```

After the above arguments, the function needs the system model response. This can be provided using either the embedded or external system model commands - either *systemModel* and *systemArgs* or the response vector *performance*.

5.2.1 Heatmaps

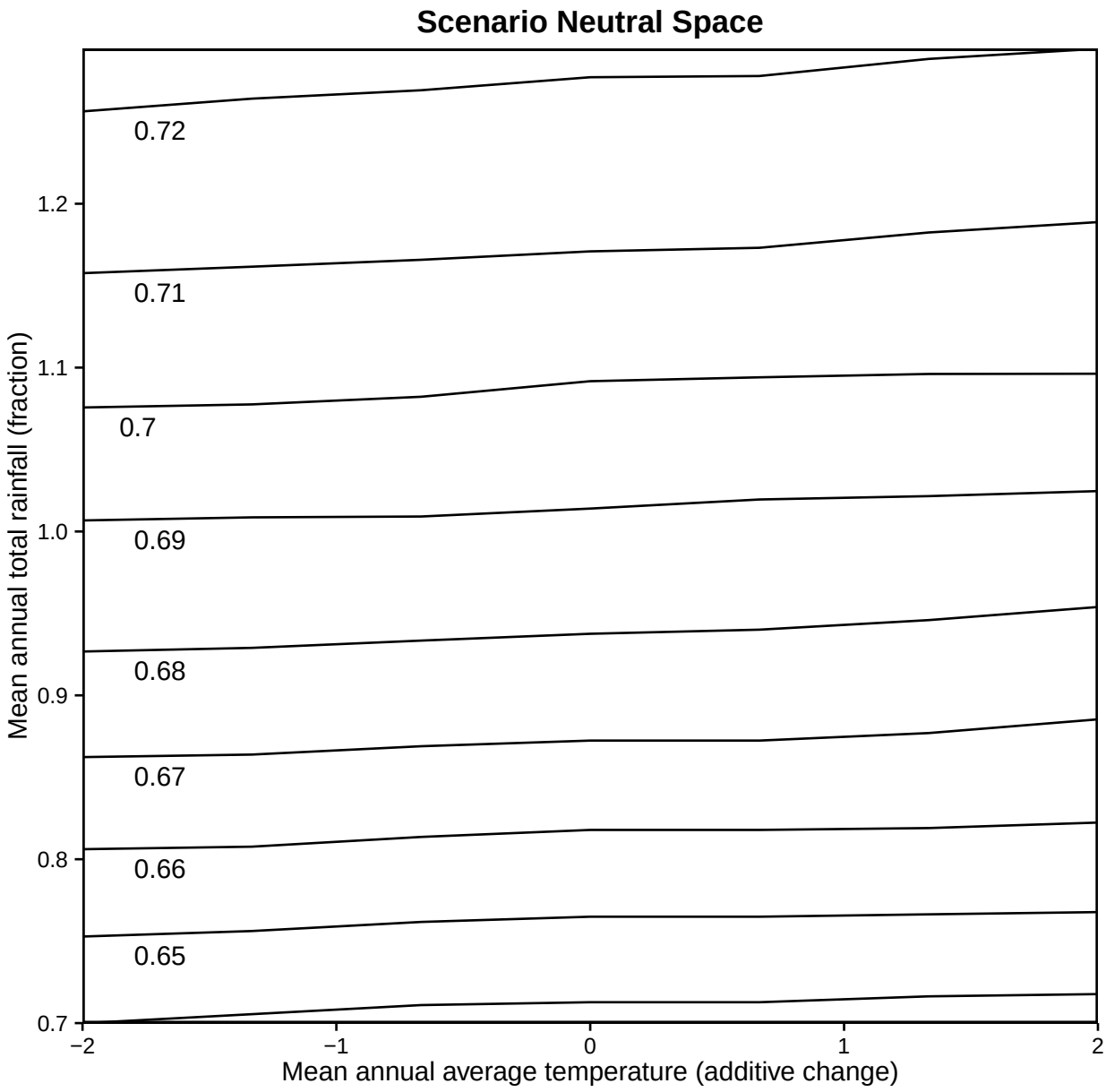
Using an already loaded **scenariogenerator** run, a Heatmap plot can be generated as follows:

```
performanceSpaces(data=out,
                  plotTag = "Heat",
                  plotArgs=plotArgs,
                  systemModel = tankWrapper,
                  systemArgs = systemArgs)
```



5.2.2 Contours

```
performanceSpaces(data=out,
                  plotTag = "Contours",
                  plotArgs=plotArgs,
                  systemModel = tankWrapper,
                  systemArgs = systemArgs)
```

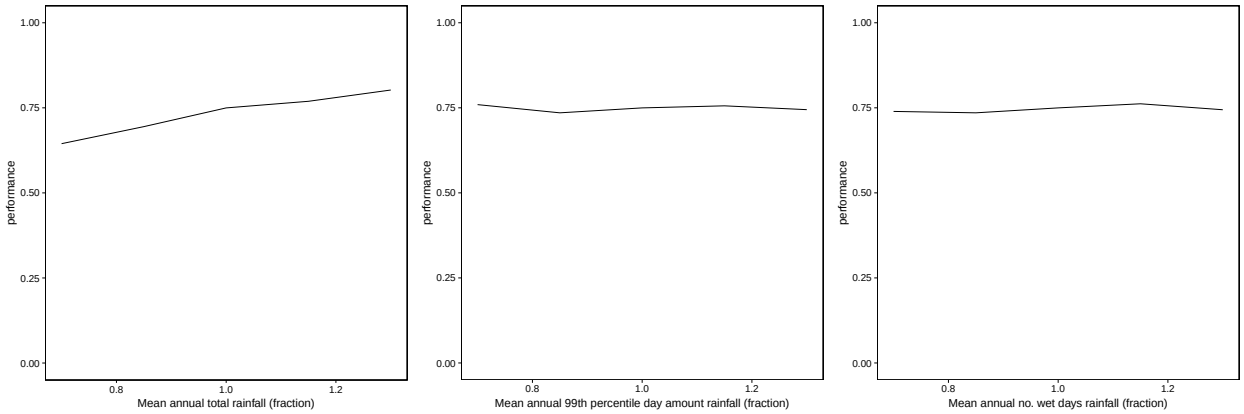


5.2.3 Binary

To be completed

5.2.4 One at a time plots

```
performanceSpaces(data=out,
                  plotTag = "OAT",
                  plotArgs=plotArgs,
                  systemModel = tankWrapper,
                  systemArgs = systemArgs)
```

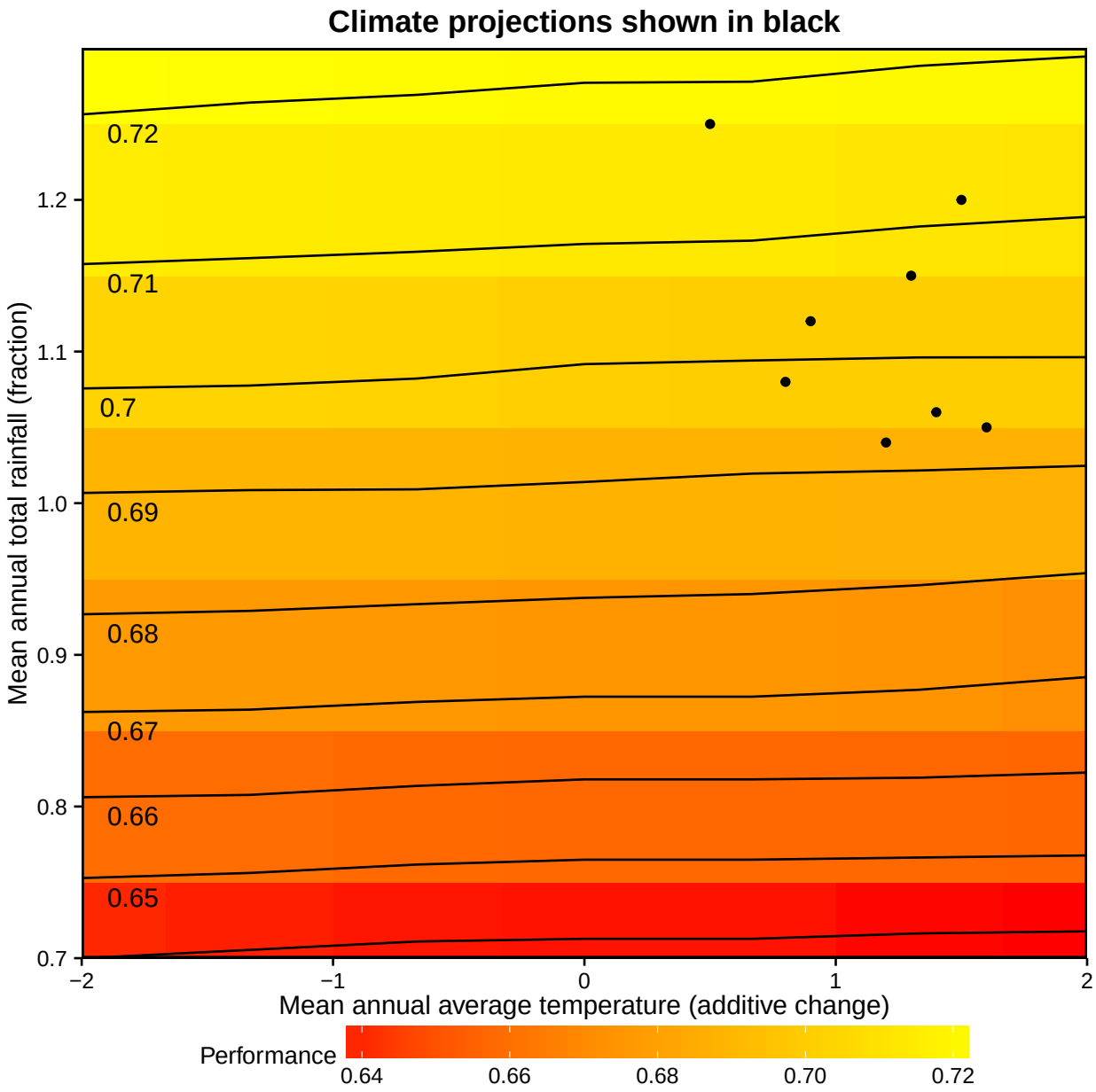


6 Adding context

To be completed

6.1 Mapping climate projections

```
data(tankclim)

plotArgs$title="Climate projections shown in black"

performanceSpaces(data=out,
                  plotTag = "Heat",
                  plotArgs=plotArgs,
                  systemModel = tankWrapper,
                  systemArgs = systemArgs,
                  climdata=climdata)
```



6.2 Exploring other objectives/decisions

---

7 Advanced functionality

To be completed

7.1 Optimisation parameters

$$OF = \sqrt{\sum_{i=1}^{n}(t_i - a_{si})^2}$$

$$OF = \sqrt{\sum_{i=1}^{n}(t_i - a_{si})^2} + \lambda * (t_p - a_p)$$

8 Glossary

A summary of the key technical terms used in this vignette are provided below.

- Scenario-neutral approach - Stress testing of a modelled system using perturbed hydrometeorological time series. This stress testing is performed independent of any climate change projections.
- Top-down approach - a climate impact assessment approach that focuses on the application of climate projections to a studied system. Stress-testing of this system is not performed as part of this approach.
- Attributes - Statistics/metrics (e.g. extremes, averages, variances, differences) of hydrometerological variables (e.g. rainfall, temperature, evapotranspiration).
- Target location - A desired combination of each of the 'n' attributes that are included in the exposure space
- Exposure space - An n-dimensional space made up of the plausible changes in the 'n' selected attributes. It comprises a number of sampled target locations. This space reflects the range of hydrometeorological conditions the system may confront in the future.
- System model - a quantitative model that is representative of the real-world system. This system model responds to changes in the climate variable drivers.
- System performance - the value of the performance metric used to evaluate the system
- Performance space - the system performance at each target location of the supplied exposure space
- Simple scaling - Perturbed time series are obtained by applying additive or multiplicative scaling factors directly to the historical hydrometeorological time series to yield an exposure space.
- Weather generator - a model used to synthetically generate weather time series
- Stochastic generation (Inverse approach) - Generate an exposure space by first selecting the desired values of the attributes of interest in the exposure space, followed by an optimization step to identify the stochastic generator parameters that produce stochastic sequences with these attributes
- Climate projection - INSERT AS NEEDED
- Objective function - INSERT AS NEEDED