```
## Error in library(SEI): there is no package called 'SEI'
```

Calculating Standardised Indices Using SEI

Sam Allen
University of Bern
Oeschger Centre for
Climate Change Research

Noelia Otero
University of Bern
Oeschger Centre for
Climate Change Research

Abstract

Standardised indices are measurements of variables on a standardised scale. The standardised scale facilitates comparisons between different variables, and its probabilistic interpretation means the indices are effective for risk management and decision making. Standardised indices have become popular in weather and climate settings, for example within operational drought monitoring systems, and have also been applied in other contexts, such as to energy variables. To facilitate their implementation in practice, the **SEI** package in R allows the flexible calculation of standardised indices. This vignette discusses the theory underlying well-known standardised indices, outlines the general framework to construct them, and provides implementation details for the **SEI** package.

Keywords: standardised indices, drought monitoring, energy droughts, R.

1. Introduction

It is common to monitor the value of a variable over time: climate scientists monitor several weather and climate variables in the hope of identifying trends in the atmosphere; energy suppliers monitor energy demand and production, with particular focus recently on production from renewable sources; financial traders monitor financial indices to assess the behaviour of stock markets, and so on. Doing so allows practitioners to not only identify long term patterns in the variables of interest, but also to analyse abnormal situations with potentially adverse socio-economic consequences.

However, it can be difficult to compare time series of variables defined on different scales. For example, different spatial regions have different climates, meaning a rainfall event that is impactful at one location may not be at another. Similarly, energy demand will depend on the local climate, as well as on installed energy capacities. To account for this, it is convenient to transform variables onto the same, standardised scale. The resulting standardised variables are often referred to as *standardised indices*.

Well-known examples of standardised indices include the Standardised Precipitation Index (SPI; McKee *et al.* 1993) and Standardised Precipitation Evapotranspiration Index (SPEI; Vicente-Serrano *et al.* 2010), which are ubiquitously used to monitor hydrometeorological droughts (Beguería *et al.* 2014). The framework used to construct the SPI and SPEI is simple and interpretable, framing the observed values in terms of their similarity to past values. As such, while other methods of standardisation exist, this framework has also been used to construct many other weather and climate indices.

This includes, for example, a Standardised Runoff Index (Shukla and Wood 2008), a Standardised Streamflow Index (Vicente-Serrano *et al.* 2012), a Standardised Groundwater Index (Bloomfield and Marchant 2013), and a Standardised Temperature Index (Zscheischler *et al.* 2014). Multivariate extensions of these standardised indices have also been proposed (Erhardt and Czado 2018). Hao *et al.* (2019), for example, define a Standardised Compound Event

Software Foundation 2017). The **spei** package (Vonk 2017) in Python additionally allows computation of the SPI and SPEI, as well as the Standardised Runoff and Groundwater Indices, while the **climate-indices** package in Python also provides functionality to compute the SPI, SPEI, and other drought indices. The developmental package **standaRdized** (Maetens 2019) can also be used for this purpose, while the **SPIGA** package (Ayala-Bizarro and Zúñiga-Mendoza 2016) in R allows the SPI to be calculated using genetic algorithms.

However, existing software packages are limited in their flexibility, with functionality typically restricted to particular standardised indices (generally the SPI and SPEI), and particular assumptions about the reference data or parametric distributions used to calculate the indices (see next section for details). In this vignette, we introduce the **SEI** R package, which seeks to assimilate the advantages of these various packages to provide a flexible and comprehensive software package with which to calculate standardised indices. In particular, the package is not designed solely for climate indices, and is therefore applicable in much broader generality. The package can handle variables defined on any time scale and with any underlying distribution, permits user-specified reference data when calculating the indices, and additionally provides plot capabilities to visualise the resulting standardised indices.

The remainder of this vignette is organised as follows. Section 2 provides some theoretical background on standardised indices and describes the general formula to construct them. Section 3 discusses diagnostic checks that confirm the standardisation was appropriate. Section 4 demonstrates how the **SEI** package can be used to calculate standardised indices in practice. An application is then presented in Section 5, whereby the **SEI** package is used to calculate standardised energy indices based on time series of renewable energy production. Section 6 concludes the vignette and discusses possible extensions to the package.

2. How to construct standardised indices

There is no unique way to define standardised indices. However, several widely-adopted indices are constructed using the same, simple procedure. Using this procedure, standardised indices are straightforward to calculate, can be defined on any timescale, and behave in a relative sense, meaning they can readily be compared for different variables, spatial regions, or points in time.

The general approach is to choose a univariate variable X that measures the quantity of interest, and then to estimate X's distribution: in the SPI, X represents the precipitation accumulation aggregated over the time scale of interest, which is usually assumed to follow a gamma distribution; in the SPEI, X represents the difference between precipitation and evapotranspiration, which is usually assumed to follow a log-logistic distribution; in the standardised temperature index, X represents the temperature, which is usually assumed to follow a normal distribution. More complicated variables have also been considered. Hao *et al.* (2019), for example, consider X to be the probability of a simultaneously hot and dry event, as derived from a copula analysis.

The distribution of X is typically estimated from an archive of realisations, $x_1, \ldots, x_n$. These could represent precipitation accumulations in different months, for example, or energy demand on different days. It is common to make parametric assumptions about the distribution, as in the examples above, though non- and semi-parametric density estimation methods could also be employed. The choice of distribution is discussed in detail in the following section. In

any case, the distributional assumptions must be verified when the index is calculated.

Let F denote the distribution function of X, and let $\hat{F}$ denote an estimate of F obtained from $x_1, \ldots, x_n$. If F is continuous, then the *probability integral transform* (PIT) $F(X)$ will be standard uniformly distributed. Hence, this transformation provides a convenient approach with which to standardise an observation X^*.

This *PIT variable* $\hat{F}(X^*)$ is bounded between 0 and 1, with values intrinsically linked to probabilities. This therefore itself provides a standardised index of sorts. This PIT variable can also be rescaled using $2\hat{F}(X^*) - 1$ to get an index that is centred at zero and bounded between -1 and 1, though it is more common to use the Gaussian quantile function Φ^{-1} to transform $\hat{F}(X^*)$. In this case, if X^* is identically distributed to X, and $\hat{F}$ is a good approximation of F, then the standardised index will follow a standard normal distribution.

That is, for any variable of interest, the probability integral transform can be used to define three different types of standardised indices:

- *Normal indices*: $\Phi^{-1}(\hat{F}(X^*))$
- *Probability indices*: $\hat{F}(X^*)$
- *Bounded indices*: $2\hat{F}(X^*) - 1$

All three types of indices can be calculated from observations of the variable X and an estimate of its distribution F. In practice, it is most common to use normal indices, though one could argue that probability and bounded indices have a more direct probabilistic interpretation. For example, if we observe the 90^{th} percentile of X, then (assuming F is correctly estimated) the corresponding normal index will be 1.28, the 90^{th} percentile of the standard normal distribution, whereas the probability index will be 0.9, and the bounded index will be 0.8. Similarly, for the 10^{th} percentile of X, the normal index will be -1.28, the probability index will be 0.1, and the bounded index will be -0.8. For the median of X, the normal and bounded indices will be zero, whereas the probability index will be 0.5.

The general framework can therefore be summarised as follows.

1. Estimate F from the observations $x_1, \ldots, x_n$.
2. Check that the estimate $\hat{F}$ correctly fits the data.
3. Apply $\hat{F}$ to an observation X^* (or several observations).
4. Transform $\hat{F}(X^*)$ so that the index is on the desired scale.

The probabilistic interpretation of these standardised indices has made them a useful tool when analysing shortages and excesses of the variable X. This was first proposed by McKee *et al.* (1993) in the context of hydrometeorological droughts, and Allen and Otero (2023) demonstrate that the same framework can be employed to define droughts in energy systems. In this case, droughts are defined when the standardised index exceeds or falls below a relevant threshold. Different classes of droughts can be defined using different thresholds, with a more severe drought corresponding to a more extreme threshold.

Following McKee *et al.* (1993), it is most common to employ three categories, typically labelled "Moderate", "Severe", and "Extreme" droughts. These often correspond to the thresholds

	Exceed			Fall below		
Index Type	Moderate	Severe	Extreme	Moderate	Severe	Extreme
Normal	1.28	1.64	1.96	-1.28	-1.64	-1.96
Probability	0.9	0.95	0.975	0.1	0.05	0.025
Bounded	0.8	0.9	0.95	-0.8	-0.9	-0.95

Table 1: Possible thresholds used to define droughts when using each of the three types of standardised indices.

1, 1.5, and 2 of the normal standardised indices, respectively, though the thresholds 1.28, 1.64, and 1.96 could also be used, which correspond to quantiles of the standard normal distribution. This assumes that a drought is defined as an exceedance of a threshold, though the negative of these values could analogously be used when interest is on values that fall below a threshold. Table 1 gives corresponding threshold values for the other two types of indices.

The **SEI** package allows the computation of all three types of indices, and additionally allows drought occurrences and characteristics to be computed from a time series of standardised indices.

3. The choice of distribution

The construction of standardised indices relies on an estimate of the distribution function F. This has led to much debate about what distributional assumptions are most appropriate. Consider the SPI, for example. McKee *et al.* (1993) initially introduced the SPI using the gamma distribution to model F. However, Guttman (1999) argued that the Pearson III distribution is more appropriate, due to the additional flexibility afforded by an extra parameter, while results in Stagge *et al.* (2015) suggest the Weibull distribution fits the monthly precipitation accumulations in their study better than alternative parametric choices. Russo *et al.* (2013) also consider using non-stationary distributions that include parameters that depend on time, while Li *et al.* (2015) incorporate additional covariates based on other climate variables. In reality, there is generally no true parametric distribution underlying the data, and the optimal assumptions will change depending on the data under consideration.

As an alternative, it has been argued (e.g. Erhardt and Czado 2018; Hao *et al.* 2019; Allen and Otero 2023) that the choice of a parametric distribution can be circumvented by using the empirical distribution function of the observations,

$$\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{x_i \leq x\}, \tag{1}$$

where $\mathbb{1}\{\cdot\}$ is equal to one if the statement inside the brackets is true, and zero otherwise. Since $\Phi^{-1}(0) = -\infty$ and $\Phi^{-1}(1) = \infty$, it is common to rescale $\hat{F}_n(x)$ so that it is never equal to zero or one, thereby ensuring that the standardised indices are real-valued; for example, using $(n\hat{F}_n(x) + 1)/(n + 2)$. The empirical distribution function returns a standardised index that can only take on $n + 1$ possible values. This is not an issue in practice if n is large, but

Distribution	Argument code	Support
Empirical	`"empirical"`	$\text{range}(x_1, \ldots, x_n)$
Kernel density estimation	`"kde"`	$\mathbb{R}$
Normal	`"norm"`	$\mathbb{R}$
Log-normal	`"lnorm"`	$[0, \infty)$
Logistic	`"logis"`	$\mathbb{R}$
Log-logistic	`"llogis"`	$[0, \infty)$
Exponential	`"exp"`	$[0, \infty)$
Gamma	`"gamma"`	$[0, \infty)$
Weibull	`"weibull"`	$[0, \infty)$

Table 2: Distributions available in **SEI**, and their supports.

can become problematic with smaller sample sizes, particularly when interest is on extreme events.

Allen and Otero (2023) propose the use of semi-parametric density estimation methods, which provide a compromise between parsimonious parametric distributions and the empirical distribution function. For example, kernel density estimators estimate the density function f corresponding to F using

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

where K is a kernel function, and $h > 0$ is a smoothing parameter called the bandwidth. The **SEI** package provides functionality to perform kernel density estimation using a Gaussian kernel. Non- and semi-parametric density estimation methods are considerably more flexible than parametric methods, though they require more data to accurately model F. Further details about density estimation methods can be found, for example, in Silverman (2018).

Table 2 lists the distributions available in the **SEI** package for estimating F. The choice of distribution should align with the variable of interest. For example, if interest is on precipitation, then we should not choose a distribution that assigns positive probability density to negative values, such as the normal distribution. Conversely, if interest is on temperature (in Celsius), then we should not choose a distribution that has support $[0, \infty)$, such as the exponential distribution. Table 2 additionally contains the support of the available distributions; the support of the empirical distribution is determined by the archive of observations $x_1, \ldots, x_n$. Wherever relevant, $\hat{F}$ is estimated using maximum likelihood estimation.

Regardless of the method used to estimate F, it is necessary to check that the estimated distribution fits the data. Several goodness-of-fit tests exist for this purpose. The **SEI** package implements the Kolmogorov-Smirnov test (Massey Jr 1951), which is based on the test statistic

$$\sup_{x \in \mathbb{R}} |\hat{F}_n(x) - \hat{F}(x)|, \tag{2}$$

that is, the maximum difference between the estimated distribution function $\hat{F}$ and the em-

pirical distribution function $\hat{F}_n$. If this statistic is close to zero, then it suggests that the hypothesised distribution fits the data well. Clearly, if the empirical distribution is used to estimate F, then this test statistic is zero.

The fit of the distribution could also be visualised graphically by plotting the PIT values $\hat{F}(x_1), \ldots, \hat{F}(x_n)$ or the corresponding standardised indices. If the model fits the data well, then these PIT values should resemble a sample from a standard uniform distribution; a histogram of these values should therefore be flat, up to sampling variation. Similarly, the normal standardised indices should resemble a sample from a standard normal distribution. An example of how this can be checked using the **SEI** package is presented in Section 5.

4. Package functionality

4.1. Standardised indices

The **SEI** package is designed to facilitate the computation of standardised indices. The principal function in the package is `std_index()`, which takes a time series or vector `x_new` as an input, and returns a time series or vector of corresponding standardised index values.

```
std_index(
  x_new,
  x_ref = x_new,
  timescale = NULL,
  dist = "empirical",
  return_fit = FALSE,
  moving_window = NULL,
  window_scale = NULL,
  agg_period = NULL,
  agg_scale = NULL,
  agg_fun = "sum",
  rescale = NULL,
  rescale_fun = "sum",
  index_type = "normal",
  ignore_na = FALSE
)
```

The input `x_new` must be either a vector or an **xts** time series object. The argument `x_ref` is similarly a vector or time series, containing the archive $x_1, \ldots, x_n$ of past observations that are used to estimate the distribution F. This distribution is then applied to each value in `x_new`, which therefore correspond to values of X^* in the previous section. The default is that `x_ref = x_new`, in which case the standardised indices are calculated *in-sample*.

As a very simple example, consider a vector of observations drawn from a gamma distribution. These can be converted to (normal) standardised indices as follows.

```
plot_raw <- plot_sei(x, type = "hist", title = "Raw")

## Error in plot_sei(x, type = "hist", title = "Raw"):  could not find
function "plot_sei"

plot_std <- plot_sei(x_std, type = "hist", title = "Standardised")

## Error in plot_sei(x_std, type = "hist", title = "Standardised"):  could
not find function "plot_sei"

grid.arrange(plot_raw, plot_std, nrow = 1)

## Error in eval(expr, envir, enclos):  object 'plot_raw' not found
```

Figure 1: Histogram of values sampled from a gamma distribution, and corresponding standardised values.

```
x <- rgamma(1000, shape = 2, scale = 1)
x_std <- std_index(x)

## Error in std_index(x):  could not find function "std_index"
```

Histograms of the raw and standardised values are displayed in Figure 1. While the raw values are heavily skewed, the standardised values closely resemble a standard normal distribution.

The `dist` argument in `std_index()` specifies how the distribution F will be estimated from `x_ref`. The default is to use the empirical distribution of `x_ref` (Equation 1). To accurately estimate F, the empirical distribution requires a sufficiently large archive, and a warning is therefore returned if the length of `x_ref` is smaller than 100. An error message is returned if the distribution support does not align with the data; for example, if a distribution with support $[0, \infty)$ is chosen while the data contains negative values.

Information about the distribution fit, including estimated parameters and goodness-of-fit statistics, can be returned by setting `return_fit = TRUE`; the default is `return_fit = FALSE`. If `return_fit = TRUE`, the output of `std_index()` is a list containing the time series of standardised indices (`si`), the estimated distribution parameters (`params`), and a named vector containing properties of the model fit (`fit_props`); this includes the number of observations used to estimate the distribution, the number and percentage of `NA` values in the data, the Akaike Information Criterion (AIC) value, and the p-value of the Kolmogorov-Smirnov test that the distribution correctly fits the data (Equation 2).

The distribution fitting is performed using `fit_dist()`, which is essentially a wrapper for `fitdist()` in the **fitdistrplus** package, along with extensions that are relevant for the construction of standardised indices. The `fit_dist()` function is also exported by **SEI**, and further details can be found on the associated help page.

The argument `index_type` in `std_index()` specifies which type of standardised index should be used. This must be one of `"normal"` (the default), `"probability"`, or `"bounded"`, corresponding to the three definitions in the previous section.

The remaining arguments are only applicable when `x_new` and `x_ref` are time series, rather

than vectors, since they all involve manipulations based on the date of the data. For example, the argument `moving_window` can be used to calculate standardised indices using a moving window. In this case, `x_ref` is updated for each observation in `x_new` so that it contains the previous `moving_window` (an integer) observations in the time series; the indices are therefore determined relative to recently observed values. The argument `window_scale` specifies the timescale that `moving_window` refers to. For example, setting `moving_window = 10` and `window_scale = "days"` would calculate standardised indices using a moving window of length 10 days.

If `moving_window` is specified but `window_scale = NULL`, then it is assumed that the units of `moving_window` correspond to the timescale of `x_new`. If the time difference between consecutive observations is consistent throughout the time series, then this can be determined automatically within `std_index()`. Alternatively, and more robustly, it can be specified by the user using the `timescale` argument. This (and `window_scale`) must be one of `"hours"`, `"days"`, `"weeks"`, `"months"`, `"quarters"`, or `"years"`.

If the original time series contains, say, hourly data, but the time series of standardised indices is desired on a different timescale, then the `rescale` argument can be used to rescale the data. This must also be one of the timescales listed above. By default, `std_index()` assumes the sum of the values should be returned when rescaling. For example, if interest is on precipitation accumulations, then converting hourly data to daily data will return the daily aggregations. However, alternative rescaling could also be performed using the `rescale_fun` argument, which is a function specifying what operation to perform over the aggregated data. If `rescale_fun = "mean"` or `rescale_fun = "max"`, for example, then the daily (or weekly, monthly, etc) mean or maximum would be provided, respectively, in place of the hourly data.

Similarly, the `agg_period` argument can be used to aggregate data across multiple time steps. Like `moving_window` and `window_scale`, `agg_period` is an integer representing the number of previous time steps over which the data should be aggregated, while `agg_scale` specifies the timescale of the aggregation period, which is by default assumed to be the timescale of the data. The argument `agg_fun` represents the function used to aggregate the data over the aggregation period.

This differs from `rescale` in the following way. If we want to convert a time series of daily observations to a time series of weekly indices, then we can use `rescale = "weeks"` (and `timescale = "days"`). On the other hand, if we set `agg_period = 1` and `agg_scale = "weeks"`, then we get a daily time series of weekly aggregated data.

To perform the rescaling, `std_index()` calls one of the **xts** functions `apply.daily`, `apply.weekly`, `apply.monthly`, `apply.quarterly`, and `apply.yearly`, alongside the argument `rescale_fun`. To perform the aggregation, `std_index()` uses the function `aggregate_xts()`, which is additionally exported by **SEI**.

```
aggregate_xts(
  x,
  len,
  scale = c("days", "hours", "weeks", "quarters", "years"),
  fun = "sum",
  timescale = c("days", "hours", "weeks", "quarters", "years"),
  na_thres = 10
)
```

The interpretation of the arguments is equivalent to in the discussion above: `x` represents the **xts** time series to be aggregated, `len` and `scale` are the length and time scale of the aggregation period, `fun` is the function used in the aggregation, and `timescale` is the time scale of `x`. The final argument `na_thres` represents the maximum proportion of values that can be missing (i.e. `NA`) in the aggregation period, before the aggregation itself returns `NA`. For example, if on one day 23 hours of precipitation accumulations are missing and only one hourly accumulation is available, this sole observation is not representative of the daily accumulation, so `NA` is returned instead. The default is a threshold of 10% missing values in the aggregation period. Further information can be found on the corresponding help pages.

4.2. Plotting indices

To visualise the indices, **SEI** includes the `plot_sei()` function.

```
plot_sei(
  x,
  type = c("ts", "hist"),
  title = NULL,
  lab = "Std. Index",
  xlims = NULL,
  ylims = NULL,
  n_bins = 30
)
```

The argument `x` is either a vector or an **xts** time series object that contains the index values to be displayed. This function can either be used to plot a time series of the values (`type = "ts"`), or a histogram (`type = "hist"`). A time series can only be plotted if the input `x` is a time series and not a vector. If `type = "hist"`, the function is essentially a wrapper for `geom_histogram()` in **ggplot2**. Here, `n_bins` can be used to specify the number of bins in the histogram.

Additional aspects of the plot can be specified using `title`, `lab`, `xlims` and `ylims`. For the time series plot, `lab` refers to the label of the y-axis, whereas it corresponds to the x-axis label of the histogram.

4.3. Drought definitions

Having obtained a time series of standardised indices, one might wish to perform an analysis of shortages, or droughts, in the variable of interest. As discussed, this can be achieved using standardised indices with the definitions of droughts in Table 1. To facilitate such analyses, the `get_drought()` function takes a time series or vector `x` as input, and outputs a dataframe containing information regarding drought occurrences and characteristsics.

```
get_drought(x, thresholds = c(1.28, 1.64, 1.96), exceed = TRUE, lag = FALSE)
```

The argument `thresholds` is a vector containing the thresholds used to define the droughts; by default, these correspond to the 90^{th}, 95^{th}, and 97.5^{th} percentiles of a standard normal distribution. It is also assumed by default that a drought is defined when the standardised

indices exceed these thresholds. This can be specified using the `exceed` argument. If a drought should instead correspond to an instance where the indices fall below the given thresholds, `exceed` should be `FALSE`.

When defining meteorological droughts, it is often common to introduce a lag such that the drought does not end when the standardised index no longer exceeds the drought threshold, but rather when the index changes sign (e.g. McKee *et al.* 1993). This accounts for cases where there are small fluctuations in the index around the drought threshold, classing this as one persistent drought rather than several shorter droughts. This can also be implemented here by setting the `lag` argument to `TRUE`.

The output is a dataframe containing the original vector or time series, as well as four additional columns: the intensity (`ins`), which corresponds to the category of drought, a higher value referring to a more severe drought category; the occurrence (`occ`), which corresponds to the intensity being higher than zero; the duration (`dur`), which provides the number of consecutive time steps that are in a drought state; and the drought magnitude (`mag`), which is the sum of all indices within the drought event. If only one threshold is used to define a drought, rather than the three used in Table 1, then the intensity is equivalent to the occurrence, and is therefore not returned.

5. Application

5.1. Renewable energy production

Consider an application of standardised indices to renewable energy production in Europe. Hourly time series of wind and solar power generation is publicly available for 27 European countries at `https://researchdata.reading.ac.uk/275/` (see Bloomfield *et al.* 2020, for details). A subset of this data set corresponding to production in 2019 can be accessed from **SEI** using

```
data("data_supply", package = "SEI")

## Error in find.package(package, lib.loc, verbose = verbose):  there is no package
called 'SEI'

head(data_supply)

## Error in eval(expr, envir, enclos):  object 'data_supply' not found
```

The dataframe `data_supply` contains a `POSIXct` time series of dates, along with the corresponding country and wind and solar power production (`PWS`). The units of the production are Gigawatt hours (GWh).

The resources available for a country to generate wind and solar power can be quantified using its corresponding installed wind and solar capacities. The installed wind (`IC17_wind`) and solar (`IC17_solar`) capacities for the 27 countries (in 2017) are also available from **SEI**.

```
data("data_ic2017", package = "SEI")

## Error in find.package(package, lib.loc, verbose = verbose):  there is no package
called 'SEI'

head(data_ic2017)

## Error in eval(expr, envir, enclos):  object 'data_ic2017' not found
```

For concision, we restrict attention here to renewable energy production in Germany.

```
de_supply_h <- subset(data_supply, country == "Germany")

## Error in eval(expr, envir, enclos):  object 'data_supply' not found

de_supply_h <- xts::xts(de_supply_h$PWS, de_supply_h$date) # convert to xts

## Error in eval(expr, envir, enclos):  object 'de_supply_h' not found
```

This can be rescaled from hourly to daily or weekly time series using **xts** functionality.

```
de_supply_d <- xts::apply.daily(de_supply_h, "sum")    # daily data

## Error in eval(expr, envir, enclos):  object 'de_supply_h' not found

de_supply_w <- xts::apply.weekly(de_supply_h, "sum")   # weekly data

## Error in eval(expr, envir, enclos):  object 'de_supply_h' not found
```

These time series of renewable energy production in Germany can be visualised using the `plot_sei()` function. Figure 2 demonstrates that all three time series display the same patterns, though the weekly time series removes the hourly and daily fluctuations.

The raw renewable energy production values can be transformed to a standardised index using `std_index()`. Following, Allen and Otero (2023), we refer to this as the standardised renewable energy production index (SREPI). Time series of hourly, daily, and weekly standardised indices are presented in Figure 3.

```
srepi_h <- std_index(de_supply_h)

## Error in std_index(de_supply_h):  could not find function "std_index"

srepi_d <- std_index(de_supply_d)

## Error in std_index(de_supply_d):  could not find function "std_index"

srepi_w <- std_index(de_supply_w)

## Error in std_index(de_supply_w):  could not find function "std_index"
```

```
lab <- "Renewable Energy Production (GWh)"
plot_h <- plot_sei(de_supply_h, lab = lab, title = "Hourly")

## Error in plot_sei(de_supply_h, lab = lab, title = "Hourly"):  could not
find function "plot_sei"

plot_d <- plot_sei(de_supply_d, lab = lab, title = "Daily")

## Error in plot_sei(de_supply_d, lab = lab, title = "Daily"):  could not
find function "plot_sei"

plot_w <- plot_sei(de_supply_w, lab = lab, title = "Weekly")

## Error in plot_sei(de_supply_w, lab = lab, title = "Weekly"):  could not
find function "plot_sei"

grid.arrange(plot_h, plot_d, plot_w, nrow = 1)

## Error in eval(expr, envir, enclos):  object 'plot_h' not found
```

Figure 2: Time series of 2019 renewable energy production in Germany at hourly, daily, and weekly time scales.

In this case, the reference data `x_ref` is assumed to be the input time series itself, meaning the indices are calculated relative to this input data. By default, `std_index()` returns normal indices (rather than probability or bounded indices), and estimates the distribution of the renewable energy production using the empirical distribution. This returns a warning for the weekly data, since there are fewer than 100 weekly production values in the time series, whereas the empirical distribution function is only recommended if at least 100 reference values are available. Alternative distributions could be implemented by changing the `dist` argument in `std_index()`.

While these indices are created by standardising the respective hourly, daily, and weekly time series of raw production values, they could all be obtained from the original hourly time series by specifying the `rescale` argument in `std_index()`.

```
z <- std_index(de_supply_h, rescale = "days")

## Error in std_index(de_supply_h, rescale = "days"):  could not find function
"std_index"

all.equal(srepi_d, z)

## Error in eval(expr, envir, enclos):  object 'srepi_d' not found
```

By default, the `plot_sei()` function displays the time series of input values. However, by specifying `type = "hist"` (rather than the default `type = "ts"`), this function can additionally be used to plot a histogram of the input values. An example of this for the daily data production and corresponding SREPI values is presented in Figure 4. As discussed in Section

```
lab <- "SREPI"
plot_h <- plot_sei(srepi_h, lab = lab, title = "Hourly")

## Error in plot_sei(srepi_h, lab = lab, title = "Hourly"):  could not find
function "plot_sei"

plot_d <- plot_sei(srepi_d, lab = lab, title = "Daily")

## Error in plot_sei(srepi_d, lab = lab, title = "Daily"):  could not find
function "plot_sei"

plot_w <- plot_sei(srepi_w, lab = lab, title = "Weekly")

## Error in plot_sei(srepi_w, lab = lab, title = "Weekly"):  could not find
function "plot_sei"

grid.arrange(plot_h, plot_d, plot_w, nrow = 1)

## Error in eval(expr, envir, enclos):  object 'plot_h' not found
```

Figure 3: Time series of 2019 SREPI values in Germany at hourly, daily, and weekly time scales.

3, such a plot can be used to validate the distributional assumptions made when calculating the indices. While the raw renewable energy production follows a heavily skewed distribution, the SREPI values resemble a sample from a standard normal distribution, suggesting the empirical distribution function is appropriate in this example. If probability or bounded indices were used, then the histogram of SREPI values should be flat, rather than normal. This is illustrated in Figure 5.

Finally, the **SEI** package additionally allows shortages or drought characteristics of the time series to be assessed. In this case, an energy supply drought could occur if the renewable energy production is low compared to previously observed values (Allen and Otero 2023). This corresponds to a low SREPI. As in Table 1, we define three categories of droughts, each defined using increasing thresholds of the SREPI. The function `get_drought()` can then be used to obtain a time series of drought occurrences, intensities, durations, and magnitudes.

```
thresholds <- -qnorm(c(0.9, 0.95, 0.975)) # -1.28, -1.64, -1.96
drought_df <- get_drought(srepi_d, thresholds, exceed = F)

## Error in get_drought(srepi_d, thresholds, exceed = F): could not find function
"get_drought"
```

From the dataframe `drought_df`, it is straightforward to analyse the properties of these drought events. For example, we can list the frequency that a moderate, severe, or extreme drought event occurs,

```
plot_raw <- plot_sei(de_supply_d, type = "hist",
                     lab = "Renewable Energy Production (GWh)")

## Error in plot_sei(de_supply_d, type = "hist", lab = "Renewable Energy
Production (GWh)"):  could not find function "plot_sei"

plot_ind <- plot_sei(srepi_d, type = "hist", lab = "SREPI")

## Error in plot_sei(srepi_d, type = "hist", lab = "SREPI"):  could not find
function "plot_sei"

grid.arrange(plot_raw, plot_ind, nrow = 1)

## Error in eval(expr, envir, enclos):  object 'plot_raw' not found
```

Figure 4: Histogram of 2019 daily renewable energy production and SREPI values in Germany.

```
srepi_d_prob <- std_index(de_supply_d, index_type = "probability")

## Error in std_index(de_supply_d, index_type = "probability"):  could not
find function "std_index"

plot_prob <- plot_sei(srepi_d_prob, type = "hist", lab = "SREPI",
                      ylims = c(0, 2), title = "Probability")

## Error in plot_sei(srepi_d_prob, type = "hist", lab = "SREPI", ylims = c(0,
:  could not find function "plot_sei"

srepi_d_bnd <- std_index(de_supply_d, index_type = "bounded")

## Error in std_index(de_supply_d, index_type = "bounded"):  could not find
function "std_index"

plot_bnd <- plot_sei(srepi_d_bnd, type = "hist", lab = "SREPI",
                     ylims = c(0, 1), title = "Bounded")

## Error in plot_sei(srepi_d_bnd, type = "hist", lab = "SREPI", ylims = c(0,
:  could not find function "plot_sei"

grid.arrange(plot_prob, plot_bnd, nrow = 1)

## Error in eval(expr, envir, enclos):  object 'plot_prob' not found
```

Figure 5: Histogram of 2019 daily renewable energy production and SREPI values in Germany, where the indices are probability and bounded indices.

```
num_ev <- sapply(0:3, function(i) sum(drought_df$ins == i))

## Error in FUN(X[[i]], ...):  object 'drought_df' not found

names(num_ev) <- c("None", "Moderate", "Severe", "Extreme")

## Error:  object 'num_ev' not found

print(num_ev)

## Error in eval(expr, envir, enclos):  object 'num_ev' not found
```

we could display the relative frequency of drought durations,

```
table(drought_df$dur[drought_df$dur > 0])

## Error in eval(expr, envir, enclos):  object 'drought_df' not found
```

or we could calculate the average drought magnitude

```
mean(drought_df$mag[drought_df$mag != 0])

## Error in eval(expr, envir, enclos):  object 'drought_df' not found
```

These are just simple examples of analyses that could be performed using the output of `get_drought()`. For a more thorough analysis, readers are referred to Allen and Otero (2023).

6. Summary

This vignette documents the **SEI** package in R, which provides comprehensive functionality to calculate time series of standardised indices. Standardised indices are projections of variables onto an interpretable and probabilistically meaningful scale, and they have become popular tools when monitoring variables of interest over time, particularly in the context of drought analysis. While there is no unique way to define standardised indices, several widely-adopted indices are constructed using the same, simple procedure. We outline this general framework for constructing standardised indices, and reiterate that this approach can be applied to any variable of interest, not just those previously considered in the literature.

The **SEI** package converts a time series of observations to a time series of standardised indices. The package allows the time series to be aggregated and rescaled to different time scales, provides plot capabilities to visualise the resulting indices, and offers a range of distributions with which to calculate the indices, including flexible non- and semi-parametric methods. The package additionally allows users to define and analyse shortages, or droughts, of the variable under consideration. An example is presented in Section 5 whereby the package is employed to calculate standardised indices to monitor shortages in renewable energy production.

The package has been designed to facilitate applications of standardised indices in practice. While standardised indices have received considerable attention within the field of climate science, such indices would also be relevant in several other domains. The **SEI** package is applicable in broad generality, though the package could also be extended so that it is suitable for more elaborate analyses. For example, several extensions of standardised indices have been proposed in the literature, and these could be integrated into the package: the package currently only considers indices corresponding to univariate variables, though multivariate extensions have also been studied in the literature (e.g. Erhardt and Czado 2018; Hao *et al.* 2019); non-stationary standardised indices have also been proposed that allow the indices to depend on covariates (Russo *et al.* 2013; Li *et al.* 2015); functionality could also be added to automatically implement methods that remove seasonality and temporal dependence in the data (Erhardt and Czado 2018).

Moreover, while the package extends existing packages by providing flexible density estimation methods, alternative such methods could also be made available, particularly methods that are appropriate when estimating the distribution of bounded variables. Functionality could also be added to handle spatial data, and, finally, the functionality of the package could be transferred to software packages in other programming languages, facilitating the implementation of standardised indices in monitoring systems that have not been designed in R.

Acknowledgements

This package was created under funding by the Oeschger Centre for Climate Change Research and the Swiss Federal Office of Meteorology and Climatology (MeteoSwiss).

References

Allen S, Otero N (2023). "Standardised Indices to Monitor Energy Droughts." *Renewable Energy*, **217**, 119206. doi:10.1016/j.renene.2023.119206.

Ayala-Bizarro I, Zúñiga-Mendoza J (2016). ***SPIGA**: Compute SPI Index Using the Methods Genetic Algorithm and Maximum Likelihood*. R package version 1.0.0, URL https://CRAN.R-project.org/package=SPIGA.

Beguería S, Vicente-Serrano SM, Reig F, Latorre B (2014). "Standardized Precipitation Evapotranspiration Index (SPEI) Revisited: Parameter Fitting, Evapotranspiration Models, Tools, Datasets and Drought Monitoring." *International Journal of Climatology*, **34**, 3001–3023. doi:10.1002/joc.3887.

Beguería S, Vicente-Serrano SM (2023). ***SPEI**: Calculation of the Standardized Precipitation-Evapotranspiration Index*. R package version 1.8.1, URL https://CRAN.R-project.org/package=SPEI.

Bloomfield H, Brayshaw D, Charlton-Perez A (2020). "ERA5 Derived Time Series of European Country-Aggregate Electricity Demand, Wind Power Generation and Solar Power Generation: Hourly Data From 1979-2019." doi:10.17864/1947.272.

Bloomfield J, Marchant B (2013). "Analysis of groundwater drought building on the standardised precipitation index approach." *Hydrology and Earth System Sciences*, **17**, 4769–4787. doi:10.5194/hess-17-4769-2013.

Erhardt TM, Czado C (2018). "Standardized drought indices: a novel univariate and multivariate approach." *Journal of the Royal Statistical Society Series C*, **67**, 643–664. doi:10.1111/rssc.12242.

Gudmundsson L, Stagge JH (2016). ***SCI**: Standardized Climate Indices such as SPI, SRI or SPEI*. R package version 1.0-2, URL https://CRAN.R-project.org/package=SCI.

Guttman NB (1999). "Accepting the standardized precipitation index: a calculation algorithm." *Journal of the American Water Resources Association*, **35**, 311–322. doi:10.1111/j.1752-1688.1999.tb03592.x.

Hao Z, Hao F, Singh VP, Zhang X (2019). "Statistical prediction of the severity of compound dry-hot events based on El Niño-Southern Oscillation." *Journal of Hydrology*, **572**, 243–250. doi:10.1016/j.jhydrol.2019.03.001.

Li J, Wang Y, Li S, Hu R (2015). "A nonstationary standardized precipitation index incorporating climate indices as covariates." *Journal of Geophysical Research: Atmospheres*, **120**, 12–082. doi:10.1002/2015JD023920.

Li J, Wang Z, Wu X, Zscheischler J, Guo S, Chen X (2021). "A Standardized Index for Assessing Sub-Monthly Compound Dry and Hot Conditions With Application in China." *Hydrology and Earth System Sciences*, **25**, 1587–1601. doi:10.5194/hess-25-1587-2021.

Maetens W (2019). ***standaRdized***. GitHub repository release v.1.0, URL https://github.com/WillemMaetens/standaRdized.

Massey Jr FJ (1951). "The Kolmogorov-Smirnov test for goodness of fit." *Journal of the American statistical Association*, **46**, 68–78. doi:doi.org/10.1080/01621459.1951.10500769.

McKee TB, Doesken NJ, Kleist J (1993). "The Relationship of Drought Frequency and Duration to Time Scales." In *Proceedings of the 8th Conference on Applied Climatology*, volume 17, pp. 179–183. Boston, MA, USA.

Nussbaumer E (2021). ***standard-precip**: Functions to calculate SPI and SPEI*. Python package version 1.0, URL https://pypi.org/project/standard-precip/.

Python Software Foundation (2017). *Python Software, Version 3.6.4*. Beaverton, OR. URL https://www.python.org/.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Russo S, Dosio A, Sterl A, Barbosa P, Vogt J (2013). "Projection of occurrence of extreme dry-wet years and seasons in Europe with stationary and nonstationary Standardized Precipitation Indices." *Journal of Geophysical Research: Atmospheres*, **118**, 7628–7639. doi:10.1002/jgrd.50571.

Shukla S, Wood AW (2008). "Use of a standardized runoff index for characterizing hydrologic drought." *Geophysical research letters*, **35**, L02405. doi:10.1029/2007GL032487.

Silverman BW (2018). *Density estimation for statistics and data analysis*. Routledge, New York. doi:10.1201/9781315140919.

Stagge JH, Tallaksen LM, Gudmundsson L, Van Loon AF, Stahl K (2015). "Candidate distributions for climatological drought indices (SPI and SPEI)." *International Journal of Climatology*, **35**, 4027–4040. doi:10.1002/joc.4267.

Vicente-Serrano SM, Beguería S, López-Moreno JI (2010). "A Multiscalar Drought Index Sensitive to Global Warming: The Standardized Precipitation Evapotranspiration Index." *Journal of Climate*, **23**, 1696–1718. doi:10.1175/2009JCLI2909.1.

Vicente-Serrano SM, López-Moreno JI, Beguería S, Lorenzo-Lacruz J, Azorin-Molina C, Morán-Tejeda E (2012). "Accurate computation of a streamflow drought index." *Journal of Hydrologic Engineering*, **17**, 318–332. doi:10.1061/(ASCE)HE.1943-5584.0000433.

Vonk M (2017). ***spei**: A simple Python package to calculate drought indices for time series such as the SPI, SPEI and SGI*. Python package version 2.0.0, URL https://pypi.org/project/spei/.

Zargar A, Sadiq R, Naser B, Khan FI (2011). "A Review of Drought Indices." *Environmental Reviews*, **19**, 333–349. doi:10.1139/a11-013.

Zscheischler J, Michalak AM, Schwalm C, Mahecha MD, Huntzinger DN, Reichstein M, Berthier G, Ciais P, Cook RB, El-Masri B, *et al.* (2014). "Impact of Large-Scale Climate Extremes on Biospheric Carbon Fluxes: An Intercomparison Based on MsTMIP Data." *Global Biogeochemical Cycles*, **28**, 585–600. doi:10.1002/2014GB004826.

Affiliation:

Sam Allen
University of Bern
Institute of Mathematical Statistics and Actuarial Science
Alpeneggstrasse 22
3012 Bern, Switzerland
E-Mail: sam.allen@unibe.ch
and
Oeschger Centre for Climate Change Research

Noelia Otero
University of Bern
Institute of Geography
Hallerstrasse 12
3012 Bern, Switzerland
E-Mail: noelia.otero@unibe.ch
and

Oeschger Centre for Climate Change Research