Chapter 1: Inflow time series and stochastic techniques

Manual for R package `reservoir`

In this chapter:

- *Understand the role of inflows in reservoir modelling*
- *Create time series (`ts`) objects from vectors*
- *Inspect and analyse `ts` objectes*
- *Generate synthetic inflow replicates using stochastic models*

Modelling a reservoir

Reservoir simulation and optimization models are built around the very basic assumption of mass balance:

$S_{t+1} = S_t + Q_t - R_t - L_t$

subject to:

$0 \leq R_t \leq \min(S_t + Q_t - L_t, R_{max})$

$0 \leq S_t \leq S_{max}$

where S_t is the volume of water in storage, Q_t **is the inflow to the reservoir**, R_t is the release from the reservoir and L_t represents other losses (evaporation, seepage) at time period t. $Smax$ is the maximum storage (or the capacity) of the reservoir and $Rmax$ the maximum controlled release (which may be water released downstream, water withdrawn for supply, or a combination of both). The equation states that the storage in the next time period (e.g., next day, month, year) will be equal storage in the current time period plus the balance of inflows (Q) minus outflows (R and L). Since the reservoir is finite, it is not possible to release more than is available in storage and incoming flow (first constraint). If outflows exceed inflows, storage levels will decrease—unless the reservoir is empty. If inflows exceed outflows, storage levels will increase—unless the reservoir is full, in which case excess water will be spilled.

Typically a reservoir modelling exercise will involve some adjustment of the release or storage capacity and analysis of resulting storage behavior through time. The time series of inflows (Q) is a fixed input to this process. Inflows determine how much water can be continually released from a reservoir (its *yield*) and how often and quickly the reservoir will fill and empty. Ultimately the purpose of a reservoir is to provide storage so that inflows are **regulated** for some useful purpose, like flood protection (reduce peak inflows by charging up storage during a storm) or water supply (prevent water shortage by releasing stored water during drought). A modeller will usually use the streamflow just upstream of the reservoir to represent the inflow, although more rigorous studies might also incorporate natural flows entering from embankments as well rainfall directly into storage. Inflows can also be back-calculated from storage measurements if the downstream release and losses are known.

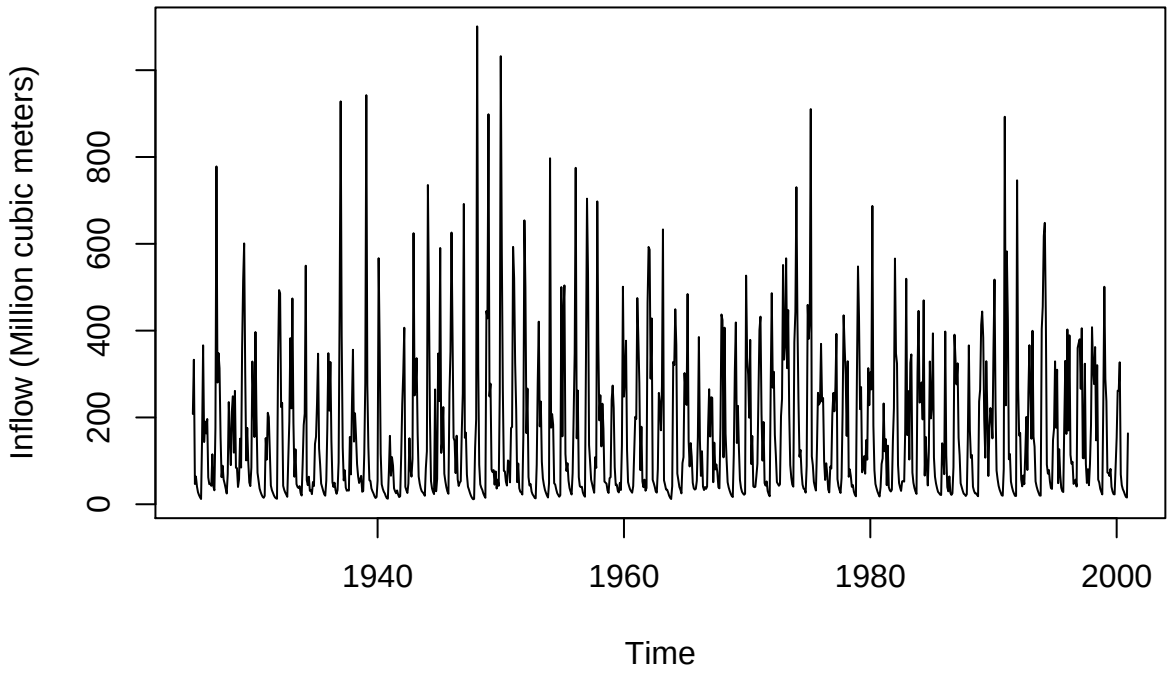
`reservoir` comes with an example inflow time series, stored inside an object named `resX`:

```
library(reservoir)
str(resX)  # View the contents of the resX object
```

```
## List of 5
##  $ Q_Mm3      : Time-Series [1:912] from 1925 to 2001: 208 332.9 46.6 63.8 40.9 ...
##  $ cap_Mm3    : num 61.9
##  $ A_km2      : num 4.1
##  $ y_m        : num 28
```

```
##  $ Inst_cap_MW: num 33.7
```

```
inflow <- resX$Q_Mm3
plot(inflow, ylab = "Inflow (Million cubic meters)")
```



Note that the plot already includes time information along the x-axis. This is because the sequence of inflows is stored as a time-stamped vector, known as a time series, or `ts` object. In `reservoir`, most functions require that the inflow data are input as `ts` objects. `ts` objects provide a range of advantages over ordinary vectors. The purpose of this chapter is to introduce you `ts` objects so you can make the most out of the capabilities in `reservoir`.

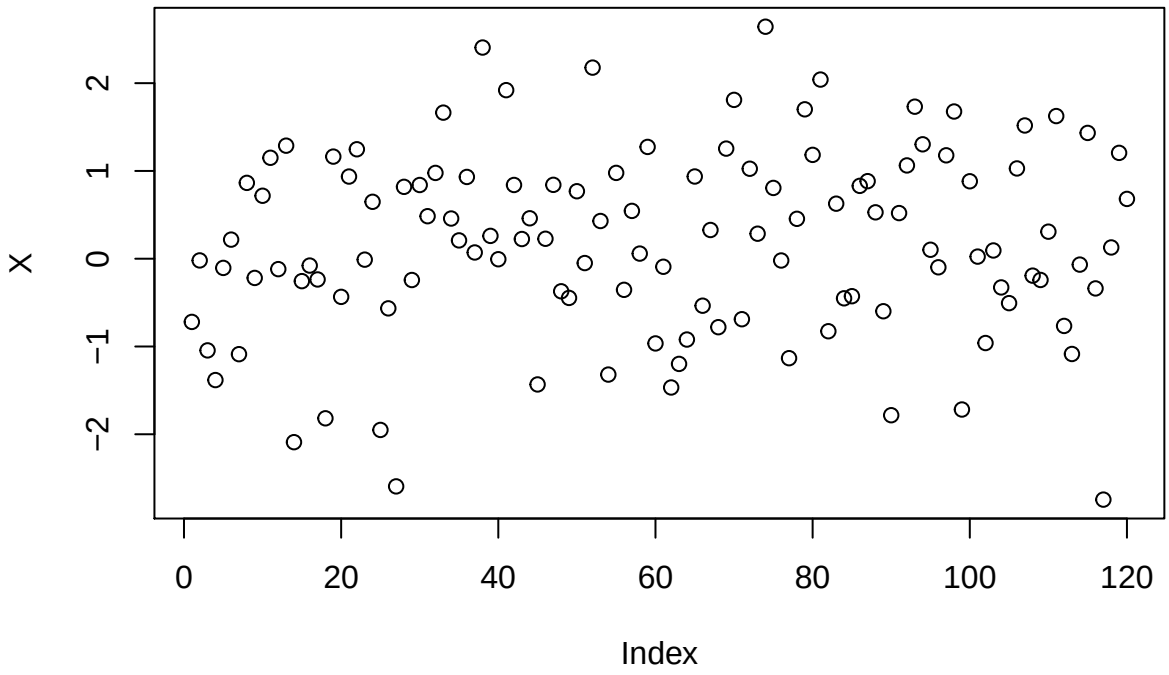
Working with `ts` (time series) objects

In R, a time series object is simply a vector of evenly-spaced, time-ordered observations. To illustrate, consider a simple vector, `X`, comprising 120 observations of some variable.

```
X <- rnorm(120)  # 120 random values from a standard, normal distribution
head(X)
```

```
## [1] -0.72007289 -0.01927725 -1.04443730 -1.38243712 -0.10485662  0.21754079
```

```r
plot(X)
```

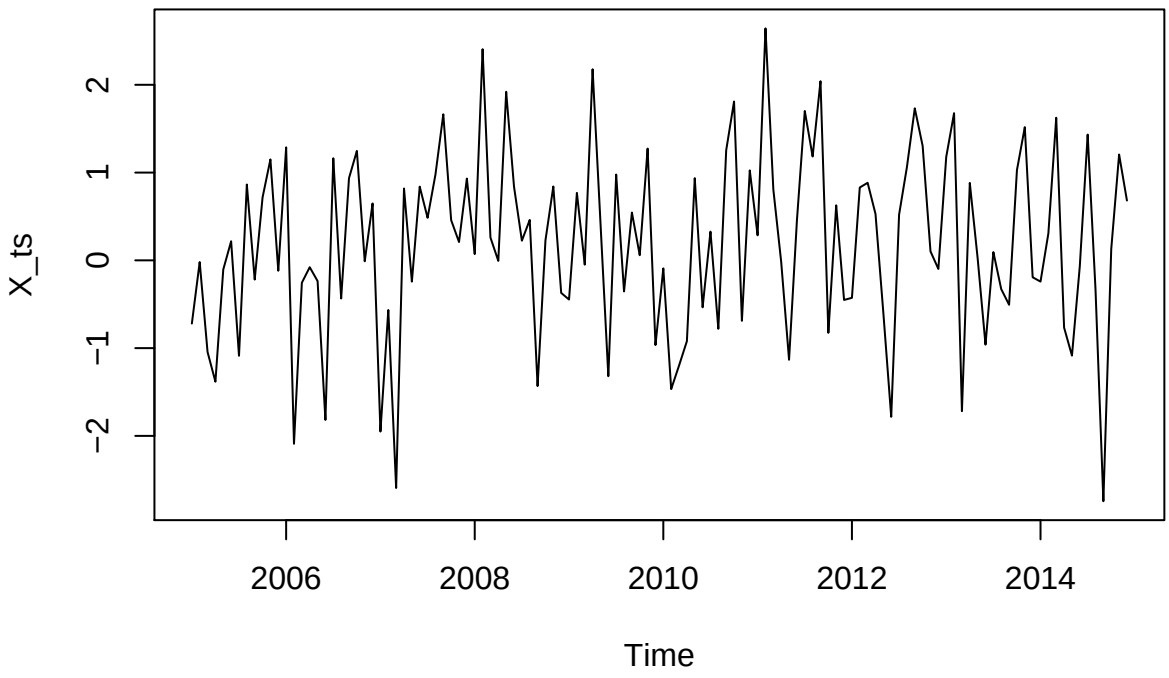


If `X` is a monthly time series (i.e., observations taken each month) with a known start date (say, January 2005), then it can be transformed to a time series object as follows:

```r
X_ts <- ts(X, start = c(2005, 1), frequency = 12)
head(X_ts)
```

```
## [1] -0.72007289 -0.01927725 -1.04443730 -1.38243712 -0.10485662  0.21754079
```

```r
plot(X_ts)
```



`ts` objects are easy to inspect:

```
start(inflow)
```

```
## [1] 1925    1
```

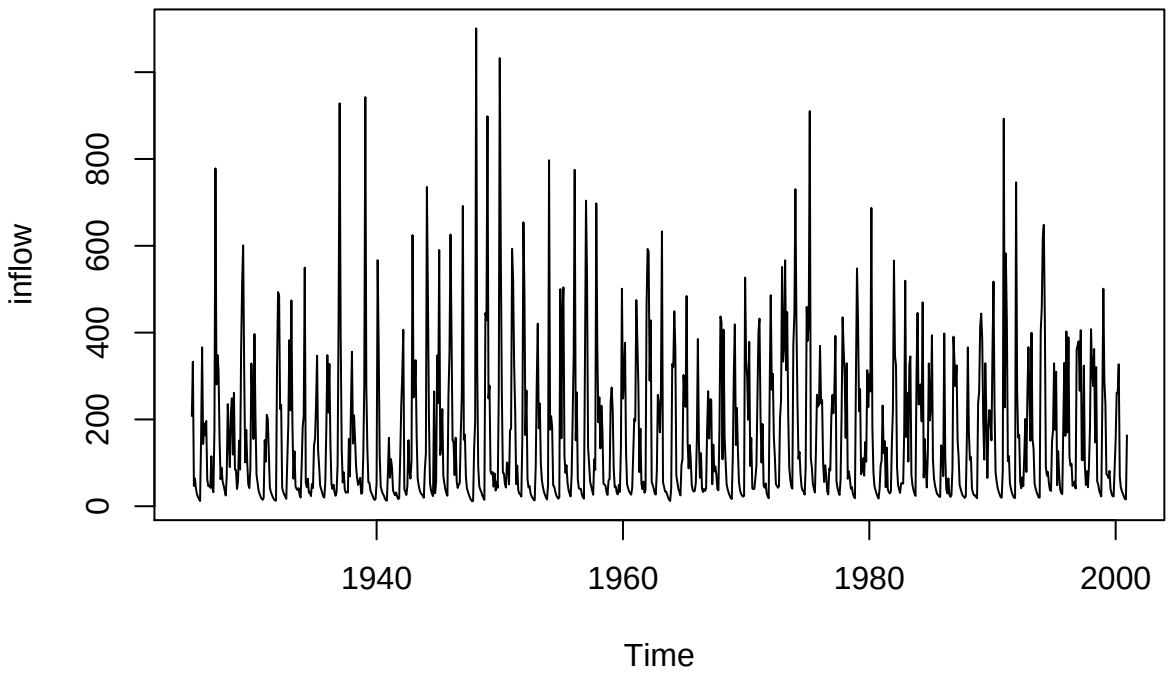
```
end(inflow)
```

```
## [1] 2000   12
```

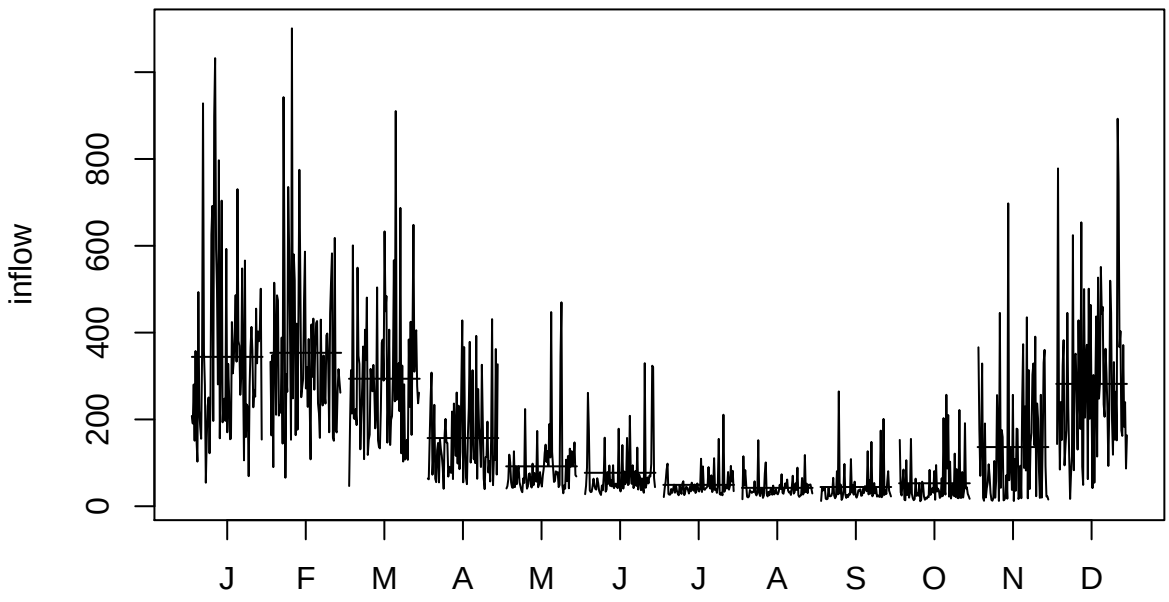
```
frequency(inflow)
```

```
## [1] 12
```

```
plot(inflow)
```

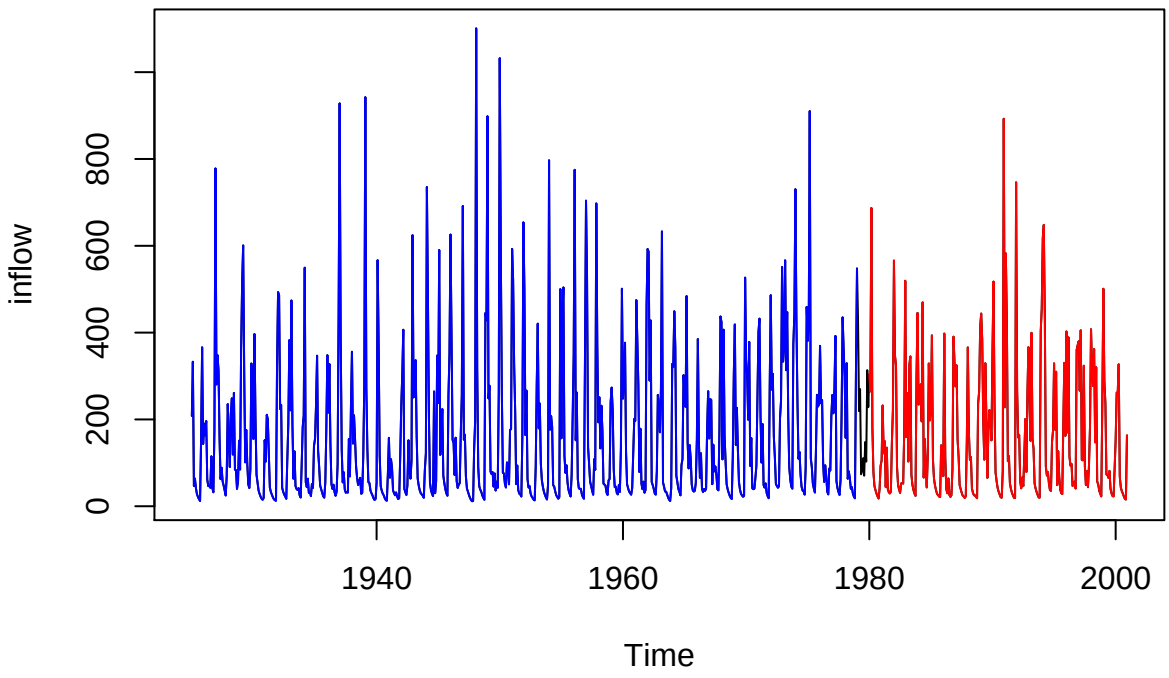



```
monthplot(inflow)
```

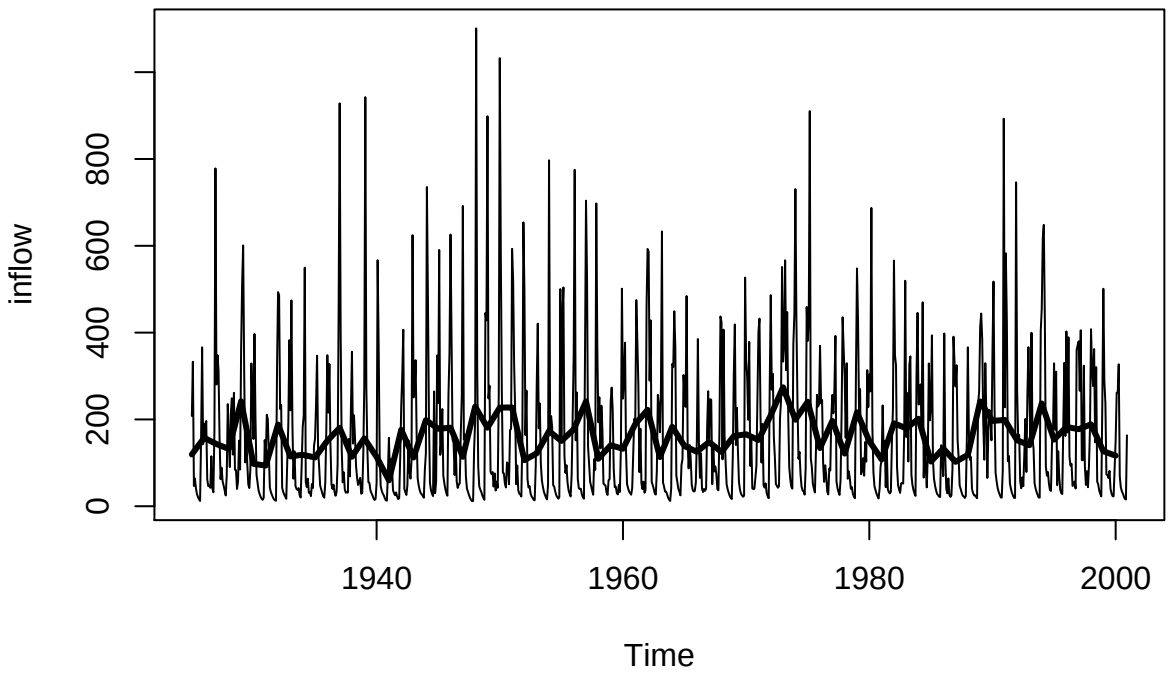


`ts` objects are easy to work with and manipulate, too:

```
inflow_post_1980 <- window(inflow, start = c(1980))
inflow_pre_1980 <- window(inflow, end = c(1979))
plot(inflow); lines(inflow_pre_1980, col = "blue"); lines(inflow_post_1980, col = "red")
```




```
inflow_annual <- aggregate(inflow, FUN = mean) # get mean annual inflows
plot(inflow); lines(inflow_annual, lwd = 3)
```



If you want to look at seasonal statistics, the `cycle` function is useful:

```
tapply(inflow, cycle(inflow), mean)  # get monthly means
```

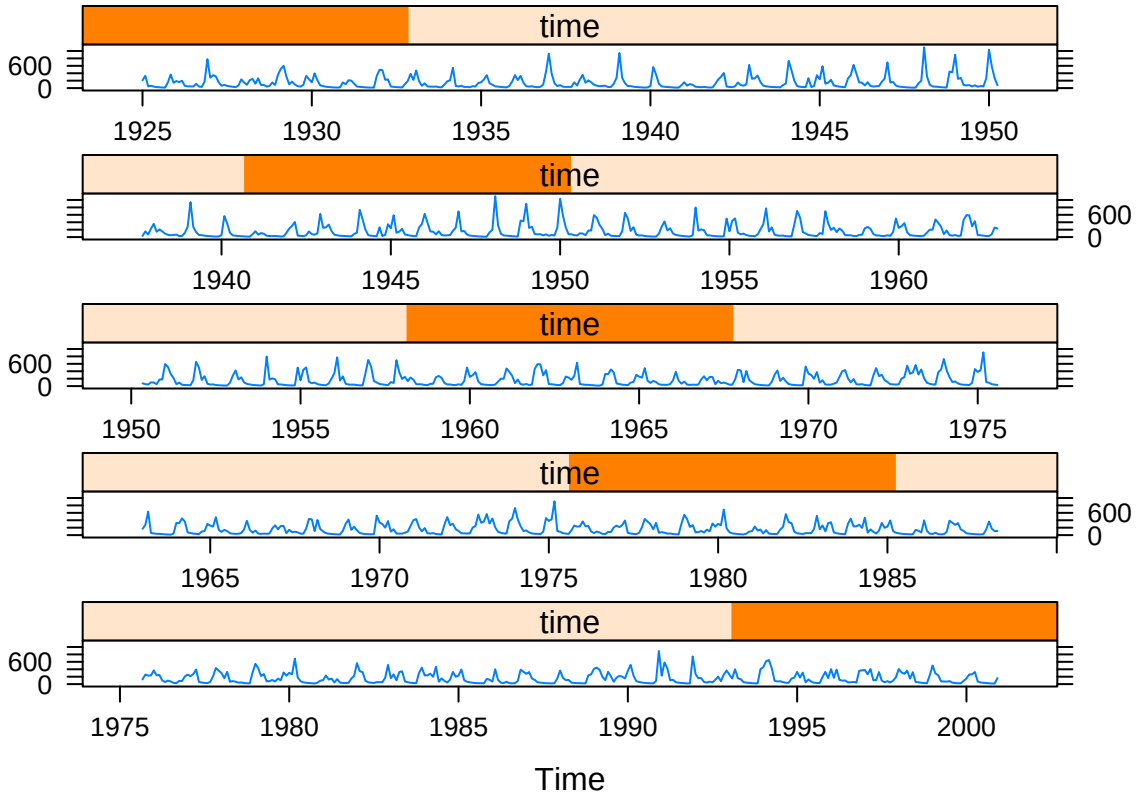
```
##         1         2         3         4         5         6         7
## 344.11425 353.45613 293.73682 157.07741  91.94790  77.03077  49.19599
##         8         9        10        11        12
##  42.33467  44.28776  52.92679 136.31578 281.84563
```

```
tapply(inflow, cycle(inflow), sd)  # get monthly standard deviations
```

```
##         1         2         3         4         5         6         7
## 203.93969 188.06221 159.03802 101.26219  77.86508  66.60370  30.21038
##         8         9        10        11        12
##  24.39514  42.87115  54.00690 137.32965 183.62329
```

Other supporting packages offer additional functionality. For example, you can use the `lattice` package to inspect a long time series by cutting into chunks:

```
lattice::xyplot.ts(inflow, cut = 5)
```



`reservoir` recognises when user input data is in the form of a time series, and performs its operations accordingly.

Generate stochastic inflow replicates using `dirtyreps`

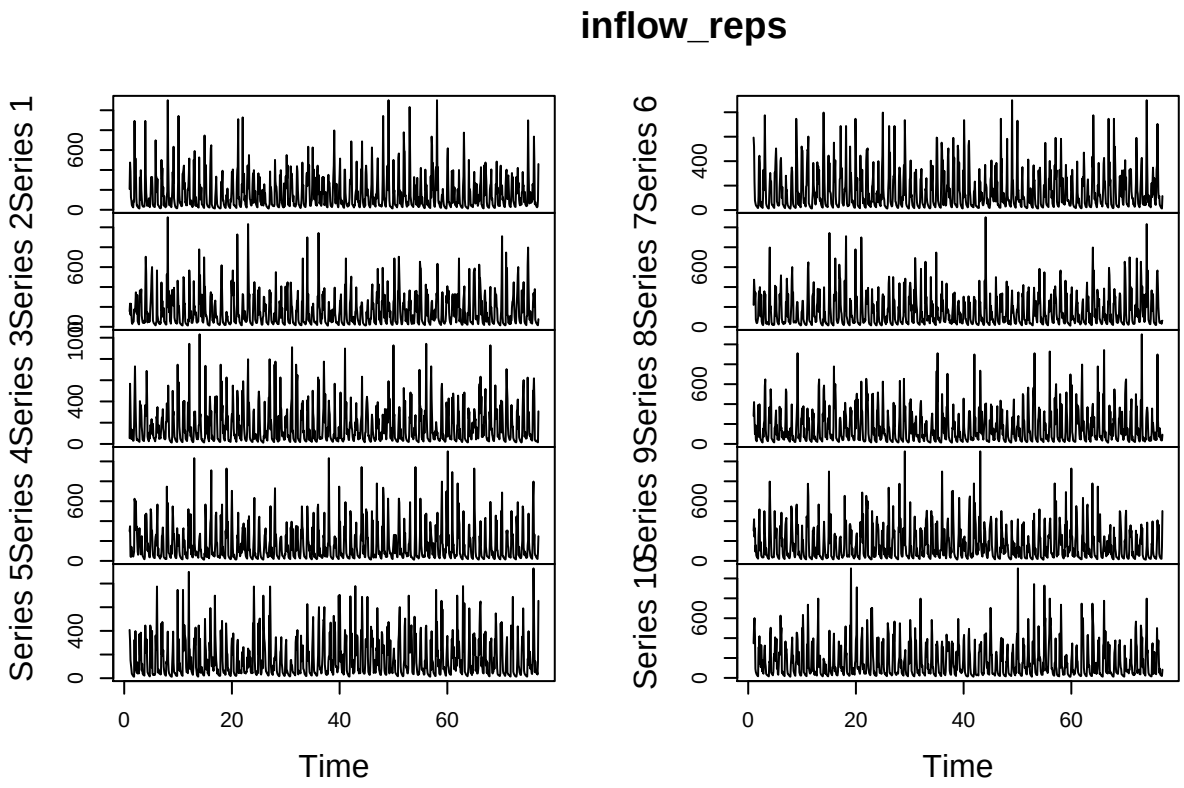
Inflow replicates are used in reservoir design and analysis to explore uncertainty. For example, with an inflow time series you can compute the storage capacity required to meet a given demand for water without reservoir failure. This is called the *no fail storage*. Given the importance of no fail storage to the design exercise, you might want to evaluate its uncertainty given the known inflow variability. You can do this by re-computing the no fail storage (or any other design or performance metric) multiple times under *stochastic replicates* of the inflow time series.

```
inflow_reps <- dirtyreps(inflow, reps = 10)
head(inflow_reps)
```

```
##       Series 1  Series 2  Series 3  Series 4  Series 5  Series 6 Series 7
## [1,] 207.95673 155.96014  54.17992 298.40263 407.87842 592.88496 220.9390
## [2,] 475.00097 233.42633 566.87622 348.24245 315.81699 520.21600 474.1232
## [3,] 384.18401 121.93194 325.69817  46.56996 153.05846 229.00645 383.8985
## [4,] 125.22938 231.68768  90.31267  39.39699 113.03011  77.79378 278.9539
## [5,]  91.92758 127.32743  45.84719 141.94811  76.69712  57.88344 345.2110
## [6,]  47.70514  46.39938 135.05825  80.10415  38.31011  41.74977  73.0425
##       Series 8  Series 9 Series 10
## [1,] 280.72108 303.73784 348.06221
## [2,] 420.67397 418.87283 514.90441
## [3,] 241.67014 241.67014 600.93007
```

```
## [4,]  99.29448 327.07558 431.01990
## [5,] 189.52133  92.71098  91.08406
## [6,]  46.39938 157.31366  59.61248
```

```
plot(inflow_reps)
```



In the next chapter, you'll put stochastic inflow replicates into practice to compute uncertainty of some reservoir summary statistics.