Missing or unavailable (NA) objects in spatstat

Adrian Baddeley

September 28, 2025
For spatstat.geom version 3.6-0.001

Abstract

This document describes experimental new code in spatstat which supports missing or unavailable (NA) objects.

Contents

1 Introduction

This document describes a new, experimental feature of `spatstat` which supports missing or unavailable ("`NA`") spatial objects.

The base R system allows for missing or unavailable entries in a numeric vector, logical vector, character vector and so on. The value `NA` is assigned to these missing entries.

Similarly the new code in `spatstat` allows for missing or unavailable entries in a list of **spatial objects**. For example, in a list of spatial point patterns, one of the entries in the list could be designated as missing or unavailable — that is, the entire point pattern is not available. This could happen because a microscope slide was broken, a patient refused to participate, a simulation algorithm failed to generate a realisation, etc.

Additionally the new code in `spatstat` allows for missing or unavailable **entries in a hyperframe**. For example, in a hyperframe representing the results of a designed experiment, in which the response from each experimental unit is a spatial point pattern, the column of point patterns could include entries which are missing or unavailable. There could also be unavailable entries in a column of pixel images, and so on.

There are two ways to indicate that an entry in a list or hyperframe is missing/unavailable:

1. a "missing object" can be created using the function `NAobject`.
2. in an existing list or hyperframe, the relevant entry can be assigned the value `NA`, and this will be coerced to a missing object.

2 Creating a missing object

2.1 A missing object belongs to a particular class

In base R, there are `NA` values of different types. The missing entries in a numeric vector are numeric `NA` values (equal to `NA_real_`) so that the vector is nevertheless treated as a vector of numeric values. The missing entries in a character vector are character `NA` values (`NA_character_`), and so on. We use a similar approach in `spatstat`: each missing/unavailable object *belongs to a particular class*.

2.2 Creating a missing object

In `spatstat`, use the function `NAobject` to create a missing or unavailable **object**.

To create a missing object that belongs to class `"foo"`, use `NAobject("foo")`. For example, a missing spatial point pattern (class `"ppp"`) is created by:

```
> X <- NAobject("ppp")
> X

<NA ppp>
```

The printout indicates that `spatstat` recognises `X` as a missing object of class `"ppp"`.

A missing or unavailable object of any particular class `"foo"` is represented in `spatstat` by an object of class `c("NAobject", "foo")`. For example, a missing spatial point pattern (class `"ppp"`) is represented as an object of class `c("NAobject", "ppp")`. This ensures that the object is treated as both a point pattern and a missing object in appropriate circumstances.

A missing object can be included as an entry in a list:

```
> pats <- solist(cells, NAobject("ppp"), redwood)
> pats
```

```
List of point patterns

Component 1:
Planar point pattern: 42 points
window: rectangle = [0, 1] x [0, 1] units

Component 2:
<NA ppp>

Component 3:
Planar point pattern: 62 points
window: rectangle = [0, 1] x [-1, 0] units
```

The printout indicates that the second entry in the list is a missing point pattern, and the entire list is nevertheless a list of point patterns.

A missing object can be included in a column of a hyperframe:

```
> m <- hyperframe(X=runif(3), Y=pats)
> m

Hyperframe:
          X     Y
1 0.9148060 (ppp)
2 0.9370754  <NA>
3 0.2861395 (ppp)
```

3 Recognising a missing object

3.1 Testing whether an object is a missing object

The function `is.NAobject` can be used to test whether an object is a missing/unavailable object.

```
> Z <- NAobject("ppp")
> is.NAobject(Z)

[1] TRUE

> is.NAobject(cells)

[1] FALSE
```

Of course one could also use `inherits`:

```
> inherits(Z, what="NAobject")

[1] TRUE
```

3.2 Missing entries in a list of objects

The generic function `is.na` is used in base R to determine which **entries** of a vector or matrix are missing values.

Similarly in `spatstat`, for lists which belong to class `"solist"`, `"ppplist"`, `"imlist"` or `"anylist"`, missing entries can be detected using methods for `is.na`:

```
> is.na(pats)

[1] FALSE  TRUE FALSE
```

For a list that does not belong to one of these special types, there may not be a method for `is.na`. Instead one could use `lapply` and friends:

```
> U <- list(cells, Z, cells)
> sapply(U, is.NAobject)

[1] FALSE  TRUE FALSE

> sapply(U, inherits, what="NAobject")

[1] FALSE  TRUE FALSE
```

3.3 Missing entries in a hyperframe

A missing object can also be included as an entry in a hyperframe:

```
> h <- hyperframe(z=1:3, p=pats)
> h

Hyperframe:
  z     p
1 1 (ppp)
2 2  <NA>
3 3 (ppp)
```

The generic `is.na` has a method for hyperframes, and returns a logical matrix indicating whether each entry of the hyperframe is missing:

```
> is.na(h)

      z     p
1 FALSE FALSE
2 FALSE  TRUE
3 FALSE FALSE
```

4 Coercion of NA to NAobject

4.1 Coercion of NA in base R

In base R, an assignment of the form `x[i] <- NA` works for an atomic vector `x` of any type. The `NA` will be converted ('coerced') to an `NA` value of the type appropriate to `x`. For example:

```
> blah <- letters[1:4]
> blah[2] <- NA
> blah

[1] "a" NA  "c" "d"
```

In this case `blah` is a character vector, so the `NA` has been coerced to a character value:

```
> is.character(blah[2])

[1] TRUE

> identical(blah[2], NA_character_)

[1] TRUE
```

4.2 Coercion of NA in spatstat

Similarly in `spatstat` an assignment of the form `x[i] <- NA` or `x[[i]] <- NA` works for a **list of objects of the same class**. The value `NA` will be coerced to an `"NAobject"` of the appropriate class. For example:

```
> Y <- rpoispp(10, nsim=3)
> Y[[2]] <- NA
> Y

List of point patterns

Simulation 1:
Planar point pattern: 13 points
window: rectangle = [0, 1] x [0, 1] units

Simulation 2:
<NA ppp>

Simulation 3:
Planar point pattern: 6 points
window: rectangle = [0, 1] x [0, 1] units
```

Here `Y` is a list of point patterns (objects of class `"ppp"`) so `NA` is coerced to `NAobject("ppp")`. The coercion can occur when the list is created:

```
> solist(cells, NA, redwood)
```

```
List of point patterns

Component 1:
Planar point pattern: 42 points
window: rectangle = [0, 1] x [0, 1] units

Component 2:
<NA ppp>

Component 3:
Planar point pattern: 62 points
window: rectangle = [0, 1] x [-1, 0] units
```

(Note that, for lists of class `"solist"` or `"anylist"`, this only works if all of the non-missing entries belong to the same class, so that the intended class is unambiguous.)

Similarly in hyperframes,

```
> g <- hyperframe(A=letters[1:3], B=rpoispp(10, nsim=3), D=runif(3))
> g

Hyperframe:
  A     B         D
1 a (ppp) 0.6034741
2 b (ppp) 0.6315073
3 c (ppp) 0.9373858

> g[2,2] <- NA
> g

Hyperframe:
  A     B         D
1 a (ppp) 0.6034741
2 b  <NA> 0.6315073
3 c (ppp) 0.9373858
```

Each individual `NA` entry will be coerced to the appropriate kind of missing value:

```
> g[3, ] <- NA
> g

Hyperframe:
     A     B         D
1    a (ppp) 0.6034741
2    b  <NA> 0.6315073
3 <NA>  <NA>        NA
```

If an entire column of a hyperframe is replaced by `NA`, the result will be an atomic column of logical `NA` values (since otherwise the intended class of objects is ambiguous):

```
> g[,2] <- NA
> g
```

```
Hyperframe:
     A  B         D
1    a NA 0.6034741
2    b NA 0.6315073
3 <NA> NA        NA
```

5 Handling missing objects

5.1 Basic support

Missing objects are handled by the code for

1. creating lists of class `"solist"`, `"ppplist"`, `"imlist"` or `"anylist"`
2. creating hyperframes
3. extracting or replacing subsets of a list of class `"solist"`, `"ppplist"`, `"imlist"` or `"anylist"`
4. extracting or replacing subsets of a hyperframe
5. printing and plotting

There are methods for `print`, `plot` and `summary` for the class `"NAobject"`. The `print` and `summary` methods simply indicate that the object is missing. The `plot` method does not generate a plot, and just prints a message that the object was missing.

5.2 Most functions in `spatstat` do not recognise `NA` objects

Most functions in `spatstat` do not handle objects of class `"NAobject"`. For example the following would generate an **error**:

```
>   X <- NAobject("ppp")
>   K <- Kest(X)
```

The object `X` is recognised as a point pattern, but does not contain any of the data that are expected for such an object, so `Kest` will fail with some peculiar error message.

Such eventualities can be handled by checking the object first:

```
>   X <- NAobject("ppp")
>   K <- if(is.NAobject(X)) NAobject("fv") else Kest(X)
```

5.3 `solapply` and friends

`NA` objects are automatically handled by the `spatstat` functions `solapply`, `anylapply` and `with.hyperframe`.

5.3.1 `solapply` and `anylapply`

The functions `solapply` and `anylapply` are wrappers for `lapply`, called in the form

```
solapply(X, FUN, ...)
anylapply(X, FUN, ...)
```

where X is a list and FUN is a function that will be applied to each element of X. The difference is that solapply expects the results to be spatial objects (e.g. point patterns, windows), while anylapply allows them to be any kind of object (e.g. numbers, matrices, "fv" objects).

The functions solapply and anylapply now check whether any elements of X are missing or unavailable, and if so, they return an NAobject as the result for each such element. The function FUN is only applied to the entries which are not missing.

For example, using the list of point patterns pats which contains some missing entries:

```
> A <- solapply(pats, Window)
> B <- anylapply(pats, Kest)
> D <- solapply(pats, Kest, demote=TRUE)
> E <- anylapply(pats, npoints)
```

These tricks do **not** work with the base R functions lapply, sapply etc.

5.3.2 with.hyperframe

The spatstat function with.hyperframe is a method for the generic with. It evaluates a given expression in each row of the hyperframe, and returns a list containing the result for each row.

This function now checks for missing or unavailable entries in the hyperframe, and if they are needed to evaluate the expression, the result is returned as an NAobject for each row in which the data are missing.

For example, using the hyperframe m which contains some missing entries:

```
> K <- with(m, Kest(Y))
> m$G <- with(m, Gest(Y))
> m$u <- with(m, clarkevans.test(Y))
> with(m, u$p.value)

          1           2           3
9.792167e-14          NA 3.812994e-11
```