# Analysing spatio-temporal data with R

Edzer Pebesma

Westfälische Wilhelms-Universität Münster

ifgi
Institute for Geoinformatics
University of Münster

Development · Center R

edzer.pebesma@uni-muenster.de

agile, May 14, 2013

# What are spatio-temporal data?

```
> head(cars)

  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

`?cars` reveals these data were recorded in the 1920s. The metric units (mph, ft) suggest: UK or US.
Are these data spatio-temporal?

# What are spatio-temporal data?

```
> head(cars)

  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

`?cars` reveals these data were recorded in the 1920s. The metric units (mph, ft) suggest: UK or US.
Are these data spatio-temporal?
No – we (geographers, geoinformaticians)

- insist on *known* coordinates $x, y, t$
- prefer known reference systems – but don't insist?

# Are these spatio-temporal data?

```
> data(Produc, package="plm")
> head(Produc)

    state year     pcap     hwy   water    util       pc   gsp    emp unemp
1 ALABAMA 1970 15032.67 7325.80 1655.68 6051.20 35793.80 28418 1010.5   4.7
2 ALABAMA 1971 15501.94 7525.94 1721.02 6254.98 37299.91 29375 1021.9   5.2
3 ALABAMA 1972 15972.41 7765.42 1764.75 6442.23 38670.30 31303 1072.3   4.7
4 ALABAMA 1973 16406.26 7907.66 1742.41 6756.19 40084.01 33430 1135.5   3.9
5 ALABAMA 1974 16762.67 8025.52 1734.85 7002.29 42057.31 33749 1169.8   5.5
6 ALABAMA 1975 17316.26 8158.23 1752.27 7405.76 43971.71 33604 1155.4   7.7
```

Answer:

# Are these spatio-temporal data?

```
> data(Produc, package="plm")
> head(Produc)

    state year     pcap     hwy   water    util       pc   gsp    emp unemp
1 ALABAMA 1970 15032.67 7325.80 1655.68 6051.20 35793.80 28418 1010.5   4.7
2 ALABAMA 1971 15501.94 7525.94 1721.02 6254.98 37299.91 29375 1021.9   5.2
3 ALABAMA 1972 15972.41 7765.42 1764.75 6442.23 38670.30 31303 1072.3   4.7
4 ALABAMA 1973 16406.26 7907.66 1742.41 6756.19 40084.01 33430 1135.5   3.9
5 ALABAMA 1974 16762.67 8025.52 1734.85 7002.29 42057.31 33749 1169.8   5.5
6 ALABAMA 1975 17316.26 8158.23 1752.27 7405.76 43971.71 33604 1155.4   7.7
```

Answer:

- Yes, if you're willing to do a lot of understanding,
- No, if you don't know where ALABAMA is, or from which country it is a state, or if you don't know what the *year* 1970 refers to.
- state and year do refer, but in a soft (unstandardized) way.

# Organisation

| | |
|---|---|
| **9:00–10:30** | R, spatial data in R |
| **11:00-12:30** | time, time series data in R |
| **13:30-15:00** | spatio-temporal data types, operations, statistics |
| **15:30-17:00** | ... ctd., flexible. |

# R

- what is R?
- what is an R package? what is a vignette?
- what is CRAN?
- where is the spatial task view? spatio-temporal task view?
- how do I find in package *x* how to do task *y*?
- how do I find out how to do ... with R?

# R

- what is R?
- what is an R package? what is a vignette?
- what is CRAN?
- where is the spatial task view? spatio-temporal task view?
- how do I find in package *x* how to do task *y*?
- how do I find out how to do ... with R?

```
> library(fortunes)
> fortune("only how")

Evelyn Hall: I would like to know how (if) I can extract some
of the information from the summary of my nlme.

Simon Blomberg: This is R. There is no if. Only how.
   -- Evelyn Hall and Simon 'Yoda' Blomberg
      R-help (April 2005)
```

# I will assume you understand this:

```
> a = data.frame(varA = c(1,1.5,2),
+  varB = c("a", "a", "b"))
> a[1,]

  varA varB
1    1    a

> a[1, drop=FALSE]

  varA
1  1.0
2  1.5
3  2.0

> a[,1]

[1] 1.0 1.5 2.0

> a[1]

  varA
1  1.0
2  1.5
3  2.0

> a[[1]]

[1] 1.0 1.5 2.0
```

```
> a["varA"]

  varA
1  1.0
2  1.5
3  2.0

> a[c("varA", "varB")]

  varA varB
1  1.0    a
2  1.5    a
3  2.0    b

> a$varA

[1] 1.0 1.5 2.0

> a$varA <- 3:1
> a

  varA varB
1    3    a
2    2    a
3    1    b
```

## Spatial data

Spatial data refresher:

- points, lines, polygons, grids
- storage: shapefiles, grid files, in- or out-of-memory
- data bases (e.g. PostGIS): geometry + attributes
- topology representation of polygons
- spatial indexes
- projected data, or long/lat?

# What makes a GIS a GIS?

## What makes a GIS a GIS?

- store, retrieve spatial data
- visualize spatial data
- manipulate spatial data
- analyze, model spatial data
  - analyze attributes, as in a data base
  - analyze geometries, or attributes depending on geometry

*"A geographic information system is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data"* (wikipedia, from esri.com)
*"In the simplest terms, GIS is the merging of cartography, statistical analysis, and database technology."* (wikipedia)

# How to get spatial data into R?

Simple answer: using `rgdal` (`readGDAL` or `readOGR`).

## How to get spatial data into R?

Simple answer: using `rgdal` (`readGDAL` or `readOGR`). More complete:

- `readGDAL` or `readOGR` read the whole data set from disk into R, that is, into the computers main or working memory (“RAM”).
- for grids, there are low-level routines: `GDAL.open` opens a file, and `getRasterData` (or `getRasterTable`) to read *portions* of data; but also (little known!):

```
> library(rgdal)
> x = GDAL.open("NDV_19980401_Gambia__the_Extract.tif")
> class(x)

[1] "GDALReadOnlyDataset"
attr(,"package")
[1] "rgdal"

> image(x[1:100, 1:100])
> class(x[1:100, 1:100])

[1] "SpatialGridDataFrame"
attr(,"package")
[1] "sp"
```

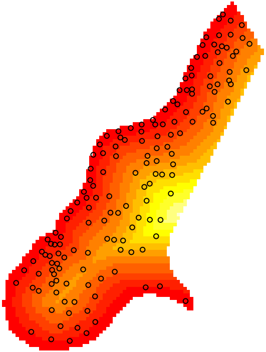
reads only the portion requested into memory

## overlay: visual

base plot: plotting sequentially, e.g.
lattice (spplot):
note: transparency is a colour attribute

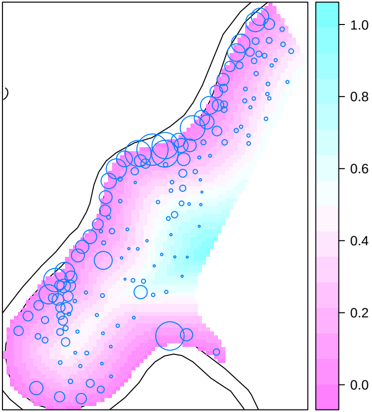
# Overlay: visual - 1. by incrementally plotting

```
> library(sp)
> data(meuse.grid)
> coordinates(meuse.grid) = ~x+y
> gridded(meuse.grid) = TRUE
> image(meuse.grid["dist"])
> data(meuse)
> coordinates(meuse) = ~x+y
> points(meuse)
> # add lines, legend, text, ...
```



# Overlay: visual - 2. using compound plot functions

```
> size = meuse$zinc / mean(meuse$zinc)
> pts = list("sp.points", meuse, pch = 1, cex = size)
> demo(meuse, echo=FALSE, ask=FALSE)
> riv = list("sp.polygons", meuse.riv)
> plt = spplot(meuse.grid["dist"], sp.layout = list(pts, riv))
> class(plt)

[1] "trellis"

> print(plt)
```



- a plot object is created, which contains everything
- this object can be manipulated, but the most used option is to print (i.e., show) it.

## Cartography?

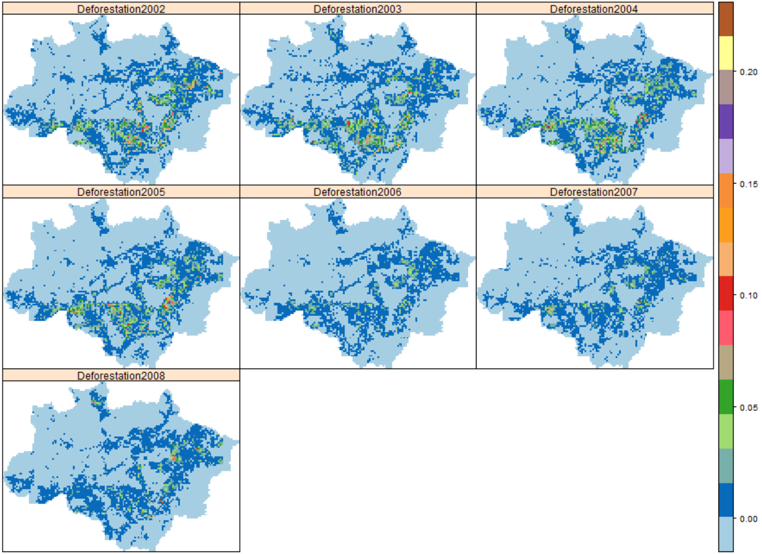
- A map is a plot with longitude and latitude, and a controlled aspect ratio; any plotting software can "do" maps, however
- reference comes from coast lines, rivers, lakes, topography, political boundaries, cities, land use etc.
- reference grid lines (parallels, meridians) may be required, and be non-straigh
- axes tics usually show little, but some information
- custom elements are often present (arrow, scale bar, multi-type legend)
- label placement is challenging (but see: rgeos::polyLabel)

## What is R good at?

- simple, repetitive graphs:
  - many, similar graphs, over different pages
  - many graphs combined in a lattice (grid: lattice, ggplot)
- non-interactive, reproducible use
- control of all details
- richness of graphics devices,
- portability, cross-platform, options for deployment

### What is R bad at?

- interactive use: zoom, pan, edit graph element etc.
- control is not trivial
- (?) incompatible plotting systems: base, lattice, ggplot, ...



G. Espindola, 2012: Spatiotemporal trends of land use change in the Brazilian Amazon; INPE

# Two work horses: rgdal, rgeos

```
> library(rgdal)
Geospatial Data Abstraction Library extensions to R successfully loaded
Loaded GDAL runtime: GDAL 1.9.1, released 2012/05/15
Path to GDAL shared files: /usr/share/gdal/1.9
Loaded PROJ.4 runtime: Rel. 4.7.1, 23 September 2009, [PJ_VERSION: 470]
Path to PROJ.4 shared files: (autodetected)
>
> library(rgeos)
Loading required package: stringr
Loading required package: plyr
rgeos: (SVN revision (unknown))
GEOS runtime version: 3.3.3-CAPI-1.7.4
Polygon checking: TRUE
```

- `rgdal` links to the GDAL (raster) and OGR (vector) data I/O library, as well as `PROJ.4` for CRS (coordinate reference systems) (re)projections
- `rgeos` links to the GEOS (Geometry Open Source) library, which powers PostGIS: does the “usual” geometry operations for features

# What is numerical overlay?

Method `over(x,y)` provides: consistent spatial overlay for points, grids, lines and polygons: at the spatial locations of object x retrieve the indexes or attributes from spatial object y and `NA` in case of no match (index `vector` if y has only geometry, attribute `data.frame` if it has attributes too).

```
> over(meuse, geometry(meuse.grid))[1:10]

 [1]   9  24  28  41  93 128  75  71
 [9] 138 161

> over(meuse, meuse.grid)[1:3,]

  part.a part.b       dist soil
1      1      0 0.00135803    1
2      1      0 0.01222430    1
3      1      0 0.10302900    1
  ffreq
1     1
2     1
3     1
```

In SQL, this resembles a left outer join of two tables

# What if there are no, or multiple matches?

## No match:

```
> m2 = meuse[-1,] # remove first record
> over(meuse, geometry(m2))[1:5]

[1] NA  1  2  3  4

> over(meuse, m2)[1:3, 1:4]

  cadmium copper lead zinc
1      NA     NA   NA   NA
2     8.6     81  277 1141
3     6.5     68  199  640
```

## Multiple matches:

```
> m2 = meuse[c(1,1:155),] # duplicate first record
> over(meuse, geometry(m2))[1:5]

[1] 2 3 4 5 6

> over(meuse, m2)[1:3, 1:4]

  cadmium copper lead zinc
1    11.7     85  299 1022
2     8.6     81  277 1141
3     6.5     68  199  640
```

So, by default, all multiple maches are ignored.

# What if we want multiple matches?

```
> m2 = meuse[c(1,1:155),] # duplicate first record
> over(meuse, geometry(m2), returnList = TRUE)[1:3]

[[1]]
[1] 1 2

[[2]]
[1] 3

[[3]]
[1] 4

> over(meuse, m2[1:4], returnList = TRUE)[1:2]

[[1]]
    cadmium copper lead zinc
1      11.7     85  299 1022
1.1    11.7     85  299 1022

[[2]]
  cadmium copper lead zinc
2     8.6     81  277 1141
```

# What if we want to compute over multiple matches?

```
> m2 = meuse[c(1,1:155),] # duplicate first record
> over(meuse, m2[1:4], returnList = FALSE, fn = max)[1:2,1:4]

  cadmium copper lead zinc
1    11.7     85  299 1022
2     8.6     81  277 1141
```

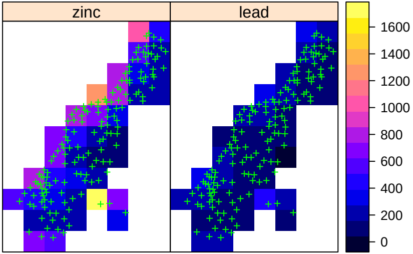
Although this is the same as

```
> over(meuse, meuse)[1:2,1:4]

  cadmium copper lead zinc
1    11.7     85  299 1022
2     8.6     81  277 1141
```

in the first case, actually the maximum is computed (`fn = max`) over the multiple matched records, and returned, as record values. In fact, this process is called *aggregation*.

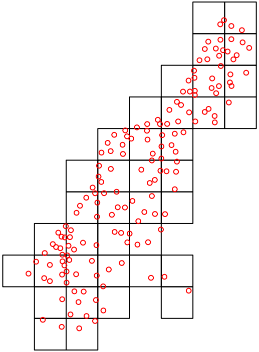
# Aggregation, the R way

```
> # for a data.frame, based on a table column:
> m = as(meuse, "data.frame")[c("zinc", "lead")]
> aggregate(m, by = list(ffreq = meuse$ffreq), mean)

  ffreq     zinc     lead
1     1 625.7500 197.9762
2     2 273.2083  99.3750
3     3 309.9565 103.0870

> # create a coarse grid:
> off = gridparameters(meuse.grid)$cellcentre.offset + 20
> gt = GridTopology(off, c(400,400), c(8,11))
> SG = SpatialGrid(gt)
> proj4string(SG) = proj4string(meuse.grid)
> # for a Spatial object, based on another Spatial object:
> agg = aggregate(meuse[c("zinc","lead")], SG, FUN = mean)
> spplot(agg, sp.layout = pts)
```



# Which pixels are covered by points? Selection with `over`

```
> SP = as(SG, "SpatialPolygons")
> over(SP, geometry(meuse))

 [1]  NA  NA  NA  NA  NA  NA   1   3
 [9]  NA  NA  NA  NA  NA  NA   8   5
[17]  NA  NA  NA  NA  NA  19  12  24
[25]  NA  NA  NA  NA  52  37  32  31
[33]  NA  NA  NA  55  58  42  NA  NA
[41]  NA  NA  64  62  49 106  NA  NA
[49]  NA  NA  65  86 104 107  NA  NA
[57]  NA  72  70  68 103  NA  NA  NA
[65]  92  77  76  94  82 118  NA  NA
[73]  NA 150  95 143  NA 155  NA  NA
[81]  NA 147 144  NA  NA  NA  NA  NA

> length(SP[!is.na(over(SP, geometry(meuse)))])

[1] 38

> length(SP[meuse]) # equivalent!

[1] 38

> plot(as(SP[meuse], "SpatialPolygons"))
> points(meuse, col = 'red')
```



## Possible over methods

Spatial data (see vignette in sp):

| | y: points | y: lines | y: polygons | y: pixels/grids |
|---|---|---|---|---|
| x: points | sp | rgeos | sp | sp |
| x: lines | rgeos | rgeos | rgeos | rgeos |
| x: polygons | sp | rgeos | rgeos | sp |
| x: pixels/grids | sp | rgeos | sp | sp |

**Table:** `over` methods implemented for different `x` and `y` arguments.

Spatio-temporal data: see `spacetime` vignette

# Nine-intersection model

The nine-intersection model is a comprehensive model, from which most relations (touches, overlaps, intersects) can be derived – see http://en.wikipedia.org/wiki/DE-9IM

```
> library(rgeos)
> gRelate(meuse.riv, meuse.riv)

[1] "2FFF1FFF2"

> gRelate(meuse.riv, SP)

[1] "212101212"

> gRelate(meuse.riv, SP, byid = TRUE)[1:5]

[1] "FF2FF1212" "FF2FF1212"
[3] "FF2FF1212" "FF2FF1212"
[5] "FF2FF1212"
```

## Exercises: spatial

- run the examples of vignette `intro_sp` in package `sp`
- run the examples of vignette `gstat` in package `gstat`

## Representing time series data in R

zoo:

- zoo: (S3) classes for ordered observations, including time series
- powerful temporal aggregation, using as.yearmon, as.yearqtr etc, and user-supplied grouping functions
- na.fill, na.approx, na.spline

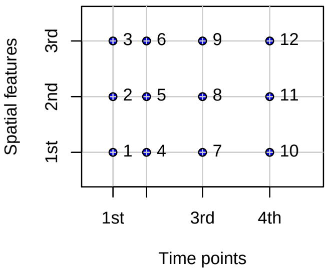
xts:

- builds on top of zoo
- (S3) explicit time reference required
- “supports” several time based systems: ”Date”, ”POSIXt”, ”chron”, ”dates”, ”times”, ”timeDate”, ”yearmon”, ”yearqtr”, ”xtime”
- ... but stores time as POSIXct, remembers original class.
- ISO 8601 time (interval) selection
- partly written in C, increasing performance; high-speed trading

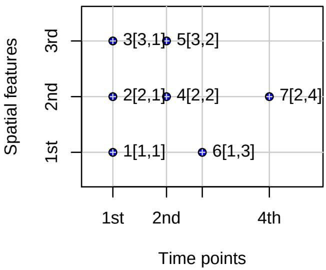
Interestingly, both have no notion of time *intervals*!

# Space-time layouts

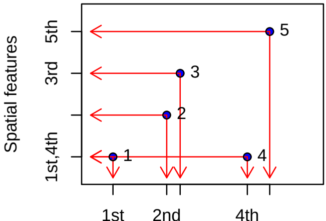
STF: full grid layout



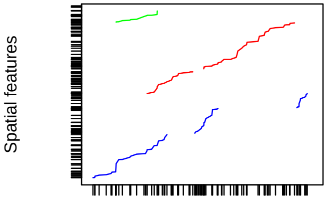
STS: sparse grid layout



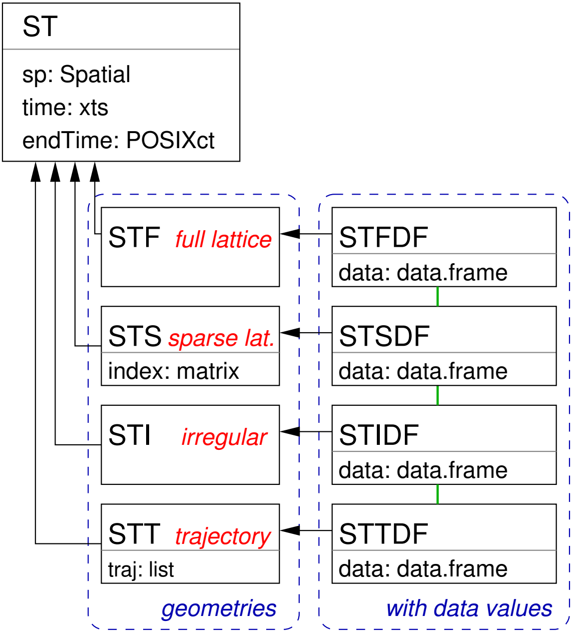
STI: irregular layout



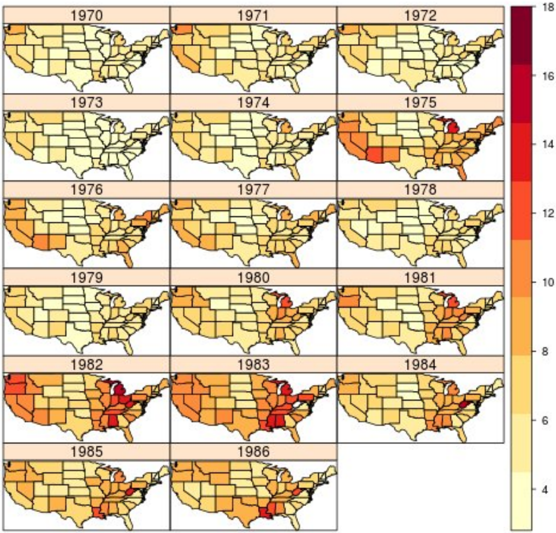
STT: trajectory



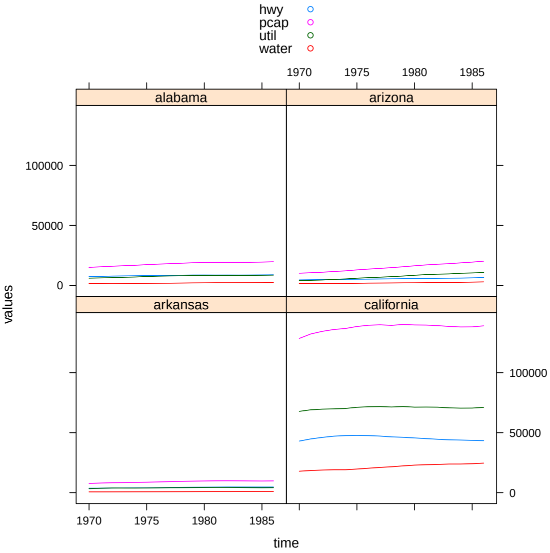
# Class layout in `spacetime`



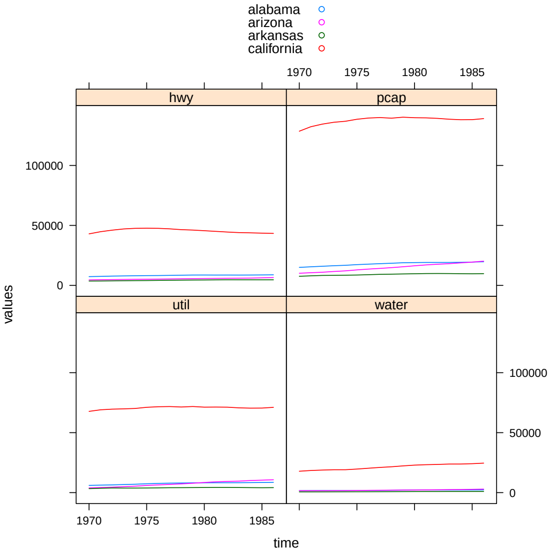
# STFDF example: `data(Produc)`



# STFDF example: `data(Produc)`



# STFDF example: `data(Produc)`

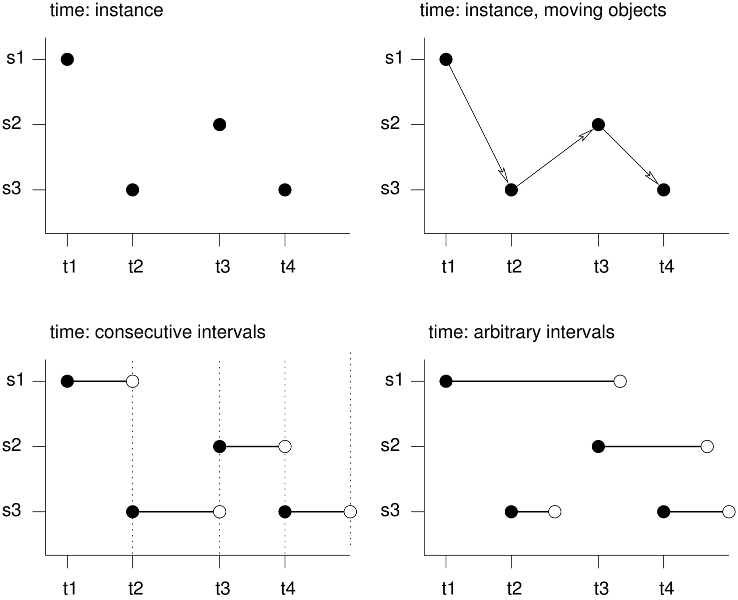


# Goals of spacetime

Provide classes and methods for a wide range of spatio-temporal data.
Methods include

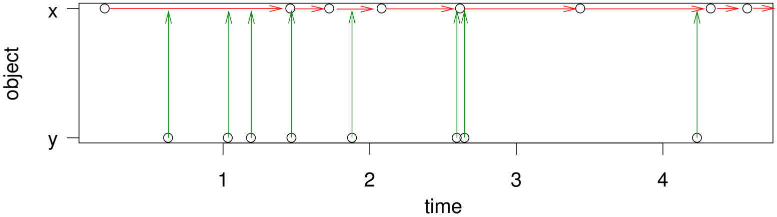
- stConstruct, construction from long, space-wide, or time-wide tables, and from xts or Spatial* objects
- selection using [, potentially resulting in xts or Spatial* objects
- coercion to Spatial* or xts
- "pass on" na.omit, na.interp, aggregate etc.
- spatio-temporal over and aggregate
- plotting
- interface with raster, grass, sos4R, ...

# time instance, intervals, movement



time: instance
time: instance, moving objects
time: consecutive intervals
time: arbitrary intervals
s1
s2
s3
t1
t2
t3
t4

# matching time & time intervals



## Spatio-temporal statistics

- See task view on CRAN for a comprehensive overview of packages
- distributed over geostatistical, point pattern, lattice, and trajectory data.

Exercise:

1. run the commands in vignette `st` in package `gstat`.
2. run the commands in the vignette `stpp` in package `stpp` (this is harder)