An introduction to geostatistical analysis of spatio-temporal data with R

D. Allard, L. Bel, E. Gabriel, T. Opitz, E. Parent

METMA IX
9th Workshop on Spatio-temporal modeling
Montpellier, 12 June 2018

1. Handling and importing large spatio-temporal data using structured objects; projection coordinate systems for geolocated data - *Eric Parent*.

2. **Visualizing data according to their temporal, spatial or spatio-temporal structures** - *Edith Gabriel*.

3. Statistical inference for spatio-temporal models: method of moments; maximum likelihood, pairwise composite likelihoods - *Thomas Opitz*.

4. Prediction and validation - *Liliane Bel*.

Plotting spatio-temporal data

We have a multivariate spatio-temporal dataset.

Its visual exploration allows us

- to gain insight into the data,
- to guide the statistical modeling of the dependencies in time, in space and over several variables.

Many R packages are available to plot spatial or temporal data, but only few are of interest for exploring the type of data set we are interested in.

Visualizing spatio-temporal data often consists in displaying change in a given dimension (e.g. space) over the other dimension (e.g. time) to highlight particular features.

Main R packages

⇒ different combinations

Spatial location	Time	Nb of var.	Visualization		Dependencies	
			Package	Function	Package	Function
1	n_t	1	tseries ggplot2	irts autoplot	stats	acf
1	n_t	n	tseries ggplot2	irts autoplot	stats	acf
n_s	1	1	sp plotKML	spplot plotKML	gstat gstat	variogram plot.gstatVariogram
n_s	1	n	sp	spplot	gstat gstat	variogram plot.gstatVariogram
n_s	n_t	1	tseries ggplot2 plotKML spacetime	irts autoplot plotKML stplot	stats gstat gstat	acf variogram plot.StVariogram
n_s	n_t	n	sp	spplot		

Spatio-temporal plot types

- **Animation** (or a movie): evolution of the spatial data through time.
- **Space (1-D)/time plots**: data in a space-time cross section.

  e.g. the Hovmöller diagram represents space on the x-axis (longitude, station...) and time on the y-axis.
- **Multi-panel spatial maps**: spatial maps for given times or aggregates over time.
- **Time series plots**: time series associated with different stations.

Spatio-temporal objects

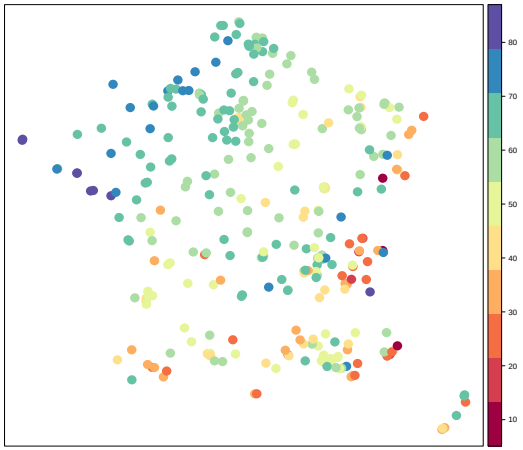
- **STFDF**: spatio-temporal data with *full* space-time grid; for n_s spatial locations and n_t times, $n_s \times n_t$ observations are available
- **STSDF**: spatio-temporal data with *partial* space-time grids; for n_s spatial locations and n_t times, an index table is kept for which nodes observations are available
- **STIDF**: unstructured spatio-temporal data; for m spatial locations and times, m observations are available

Static spatial plots for given time(s)

Examples of `spplot` (plotting spatial data with attributes)

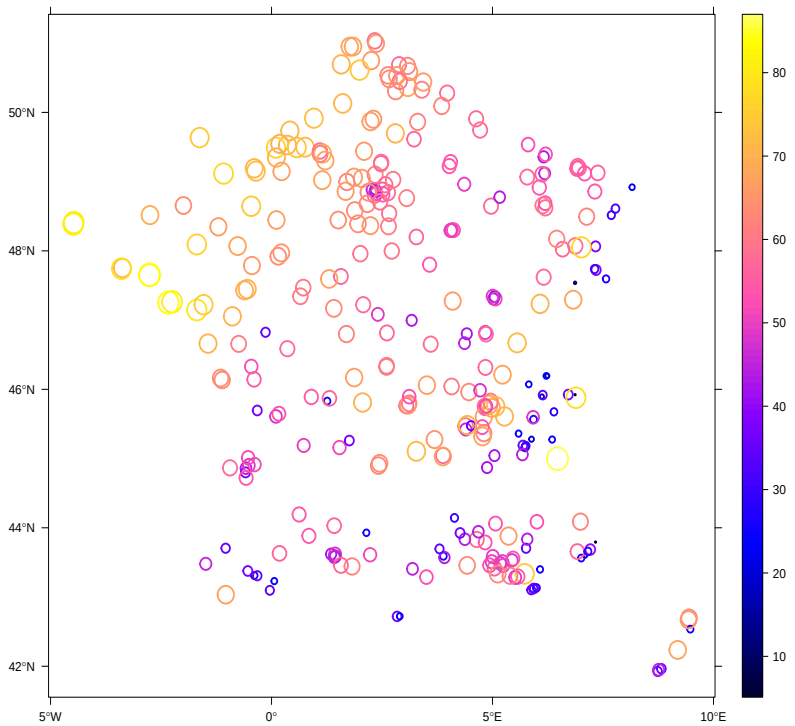
```
load('data/STFDF_daily.Rdata')
library(RColorBrewer)
library(sp)
```

```
#### Selecting all locations, 1 day (by date), 1 variable (O3)
sel_O3 <- STFDF_day[,"2014-01-01",'O3']

#### 1) Plot the selection
spplot(sel_O3,cuts=10,colorkey=TRUE,col.regions=brewer.pal(10,"Spectral"))
```



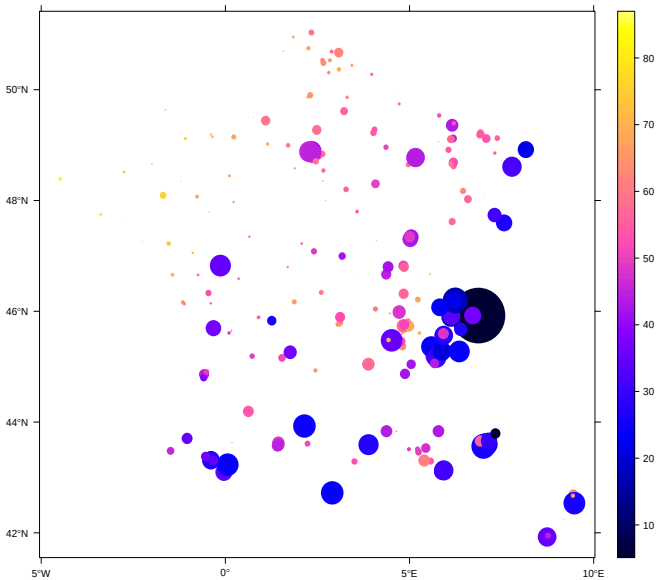
One variable

```
#### 2) The symbol size is now proportional to O3 value at the station

cxO3 = attr(sel_O3,'data')/max(attr(sel_O3,'data'),na.rm=TRUE)

spplot(sel_O3, scales=list(draw=TRUE), colorkey=TRUE, pch=1,
       cex=ifelse(is.na(STFDF_day[['O3']]),0,cxO3$O3))
```



Two variables

```
#### 3) Plot all locations, 1 day (by date), 2 variables (O3 and NO2)
#### Symbol size is proportional to NO2 value, color to O3 value

cxNO2=attr(STFDF_day[,"2014-01-01",'NO2'],'data')
      /max(attr(STFDF_day[,"2014-01-01",'NO2'],'data'),na.rm=TRUE) * 3
cxNO2[is.na(cxNO2)]<-0
spplot(sel_O3,cex=ifelse(is.na(STFDF_day[['O3']]),0,cxNO2$NO2),
       scales=list(draw=TRUE),colorkey=TRUE)
```

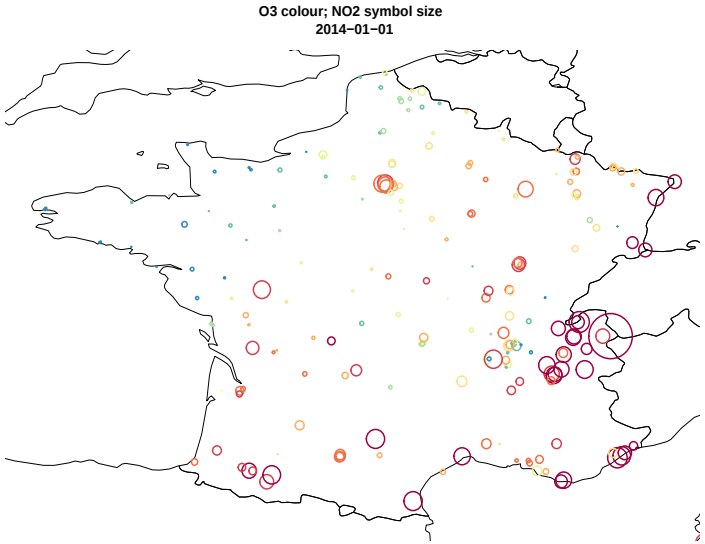


Adding maps in the background

```
### 4) Download and add map of countries

library(rworldmap)

world <- getMap(resolution="low")
proj4string(world)

europe.lim <- which(world$NAME%in%c("France"))
europe.lim <- lapply(europe.lim, function(i){
  df <- data.frame(world@polygons[[i]]@Polygons[[1]]@coords)
  colnames(df) <- list("lon", "lat")
  return(df) })

europe.limits <- do.call("rbind", europe.lim)
```

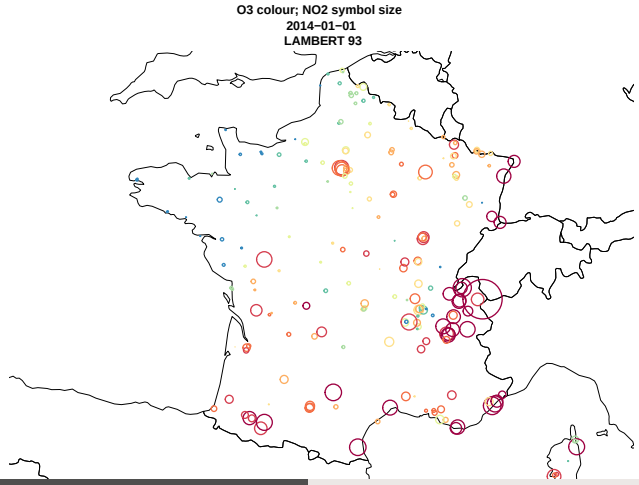
Plotting two variables on countries map

```
#### Symbol size is proportional to NO2 value, color to O3 value

plot(world,xlim=range(europe.limits$lon),ylim=range(europe.limits$lat),asp=1,
     main='O3 colour; NO2 symbol size \n 2014-01-01')

colours <- brewer.pal(10,"Spectral")
quantO3 <- quantile(attr(sel_O3,'data')$O3,seq(0,1,length.out=10),na.rm=TRUE)

plot(sel_O3,col=colours[findInterval(O3,quantO3,all.inside=TRUE)],
     cex=cxNO2$NO2,scales=list(draw=TRUE),add=TRUE,pch=1)
```



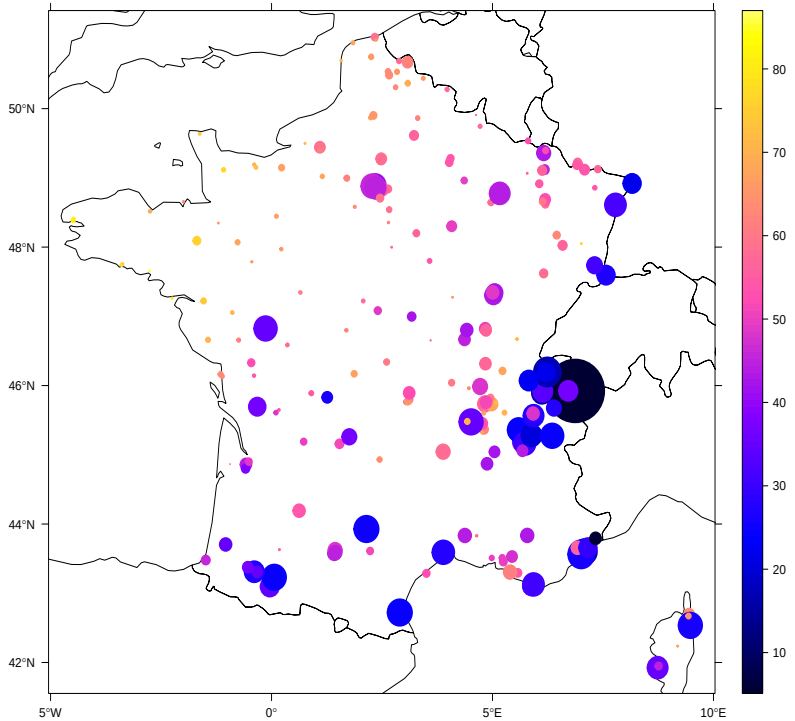
Changing the projection system

```
### 5) Same as before but using Lambert 93 (L93) projection

splimit <- SpatialPoints(europe.limits)
proj4string(splimit) <- proj4string(world)
splimit_l93 <- spTransform(splimit,CRSobj=CRS("+init=epsg:2154"))
worldL93 <- spTransform(world,CRSobj=CRS("+init=epsg:2154"))
STFDF_dayL93 <- spTransform(STFDF_day,CRSobj=CRS("+init=epsg:2154"))

plot(worldL93,xlim=range(splimit_l93$lon),ylim=range(splimit_l93$lat),asp=1,
     main='O3 colour; NO2 symbol size \n 2014-01-01 \n LAMBERT 93')
plot(STFDF_dayL93[,"2014-01-01",'O3'],add=TRUE,pch=1,scales=list(draw=TRUE),
    col=colours[findInterval(O3,quantO3,all.inside=TRUE)],cex=cxNO2$NO2)
```



Adding a layout in spplot

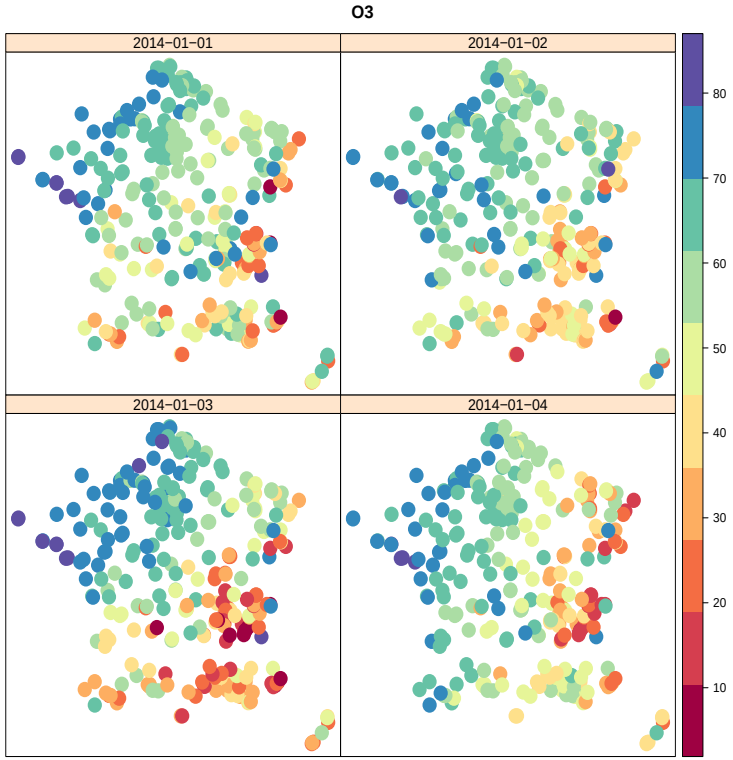
```
### 6) The same, adding a layout in spplot
###    (easier and centered around our data)

spplot(sel_O3,cex=cxNO2$NO2, scales=list(draw=TRUE), pch=19,
       sp.layout=list(world,first=TRUE), colorkey=TRUE)
```



Trellis plot

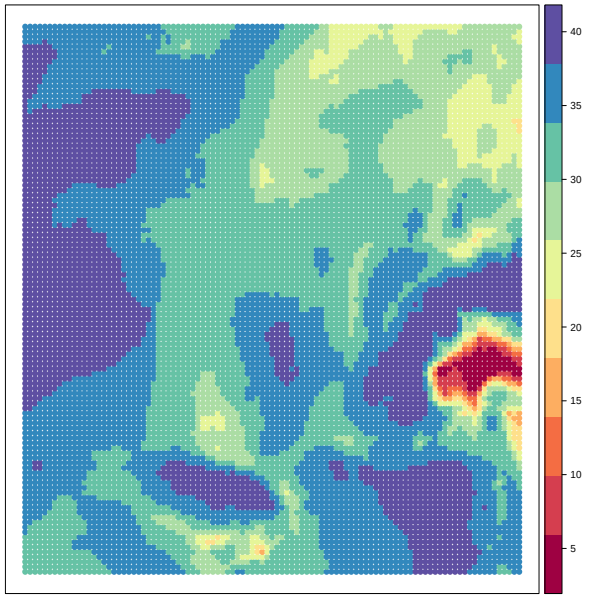
Examples of trellis plot using spplot and stplot

```
#### 7) Plot all locations, 4 days (by date), 1 variable (O3)

stplot(STFDF_day[,"2014-01-01::2014-01-04",'O3'],cuts=10,colorkey=TRUE,
       col.regions=brewer.pal(10,"Spectral"))
```



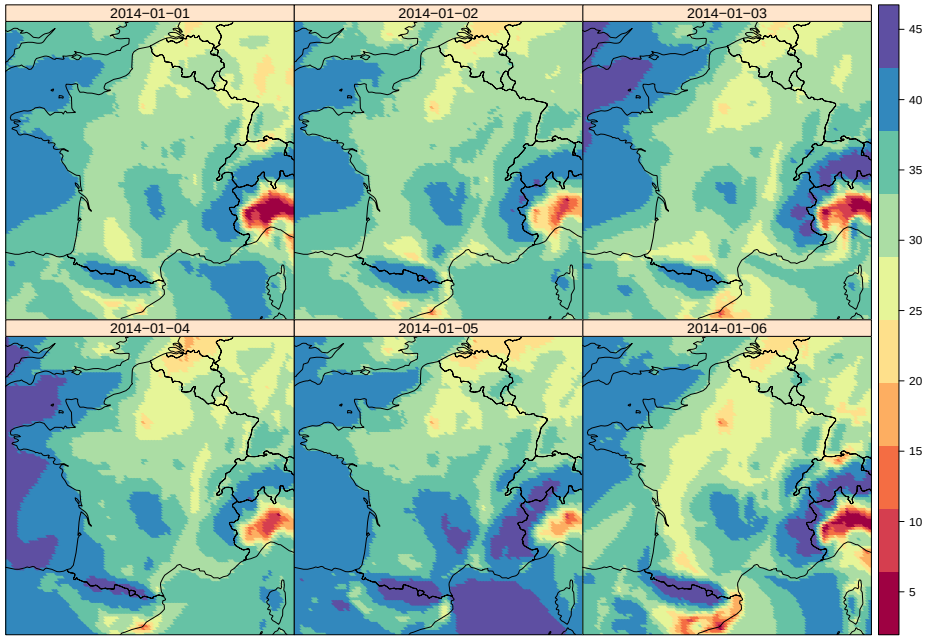
Grid plot for STFDF objects

```
#### 8) Plot full GRID data, 1 day (by date), 1 variable (O3)

load('data/CHM_daily.Rdata')
chim_O3_1d <- CHM_day[,"2014-01-01",'O3']
spplot(chim_O3_1d,col.regions=brewer.pal(10,"Spectral"),cuts=10,colorkey=TRUE)
```



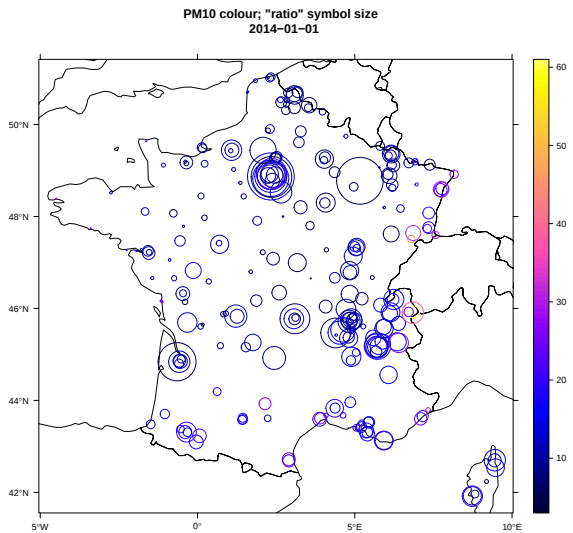
Grid plot for STSDF objects

```
#### 9) Plot partial GRID data, 7 days selected, 6 plotted

chim_O3_7d <- CHM_day[,"2014-01-01::2014-01-07",'O3']

stplot(chim_O3_7d,number=6,col.regions=brewer.pal(10,"Spectral"),
       cuts=10,sp.layout=list(world,first=FALSE),colorkey=TRUE)
```



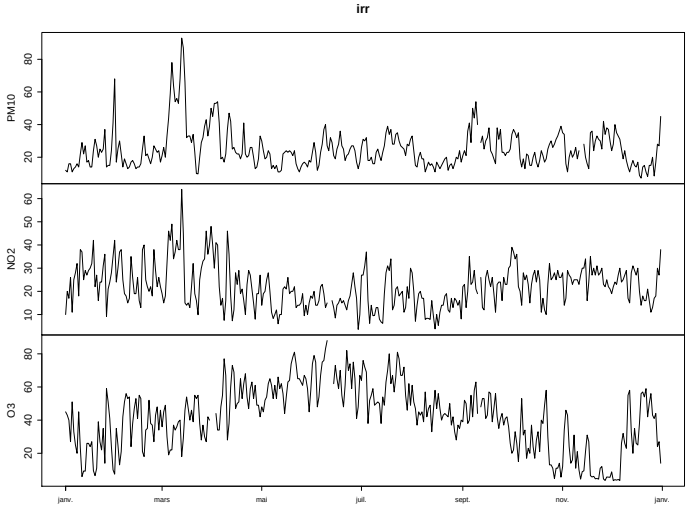
Adding a new variable

```
#### 10) Add and plot a new variable (ratio between NO2 and PM10)

NO2   <- attr(STFDF_day[,"2014-01-01",'NO2'],'data')
PM10  <- attr(STFDF_day[,"2014-01-01",'PM10'],'data')
ratio <- NO2$NO2/PM10$PM10
ratio[is.na(ratio)]<-0

spplot(STFDF_day[,"2014-01-01",'PM10'],sp.layout=list(world),
       cex=ifelse(is.na(CHM_day[['PM10']]),0,ratio),pch=1,
       scales=list(draw=TRUE),colorkey=TRUE,
       main='PM10 colour; "ratio" symbol size \n 2014-01-01')
```



Time series plot for given stations

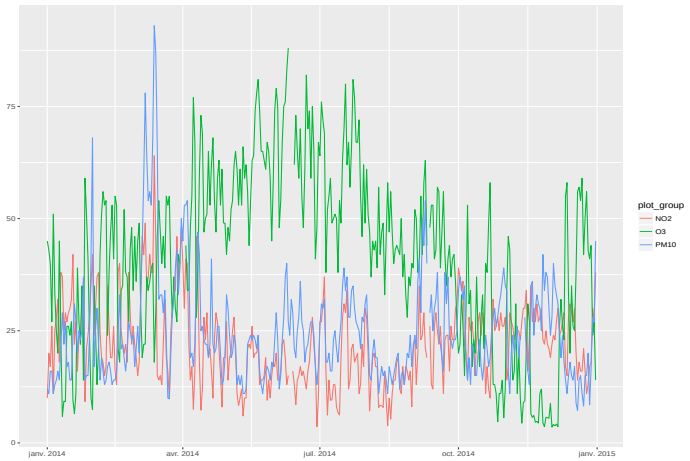
```
### One time series plot can represent the data corresponding
### to one or multiple locations and one or multiple variables.

#### 1) single location, multiple variables

IDs <- levels(OBS_jour$ID)
ts1 <- STFDF_day[which(IDs %in% c('FR01011')),,c("PM10","NO2","O3")]
irr <- irts(index(ts1),as.matrix(ts1[,1:3]))
plot(irr)
```

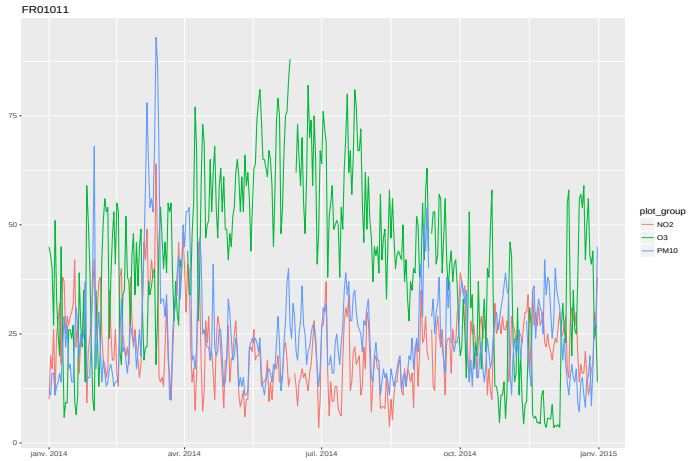


Create more elegant plot using ggplot2

autoplot draws a particular plot for an object of a particular class

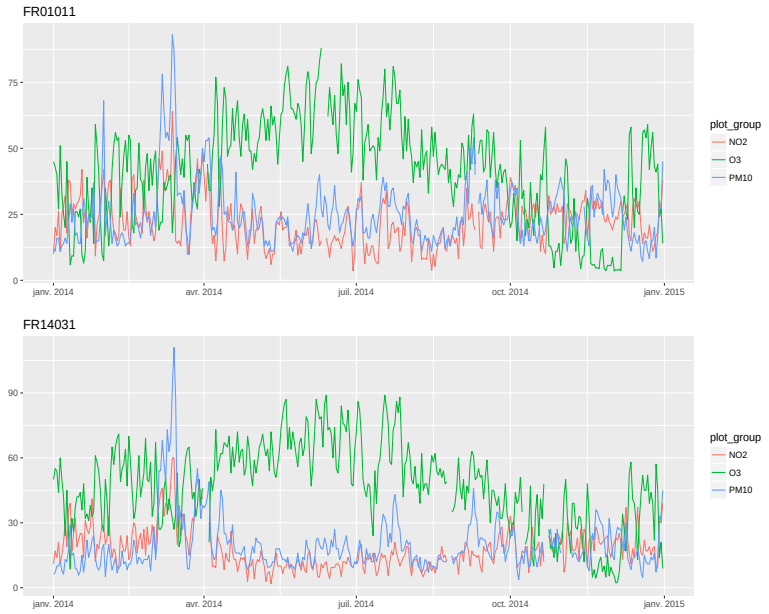
```
#### The same, refined using ggplot2

p1 <- autoplot(irr, facets = FALSE)
p1

p1 <- p1+ggtitle('FR01011')
p1
```





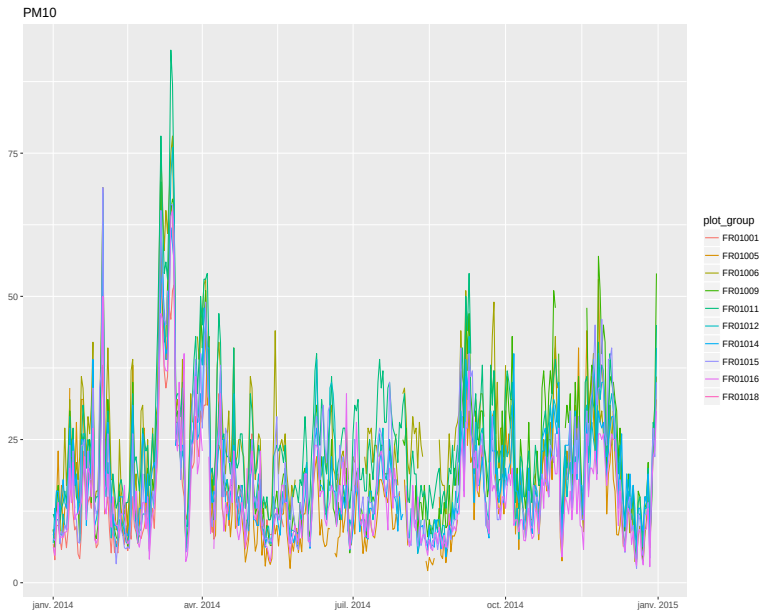
Multiplots

```
#### 2) Two locations, multiple variables

ts2  <- STFDF_day[which(IDs %in% c('FR14031')),,c("PM10","NO2","O3")]
irr2 <- irts(index(ts2),as.matrix(ts2[,1:3]))
p2   <- autoplot(irr2, facets = FALSE)+ggtitle('FR14031')

new("ggmultiplot",plots=list(p1,p2),ncol=1)
```



Multilocations

```
#### 3)  Multiple locations (>3), single variable (PM10)

init <- STFDF_day[which(IDs==IDs[1]),,c("PM10")][,1]
selPM10 <- matrix(0,ncol=10,nrow=length(init))
for(i in 1:10) selPM10[,i] <- STFDF_day[which(IDs==IDs[i]),,c("PM10")][,1]
colnames(mysel) <- IDs[1:10]
irr <- irts(index(init),selPM10)
p3  <- autoplot(irr, facets=FALSE) + ggtitle('PM10')
p3
```



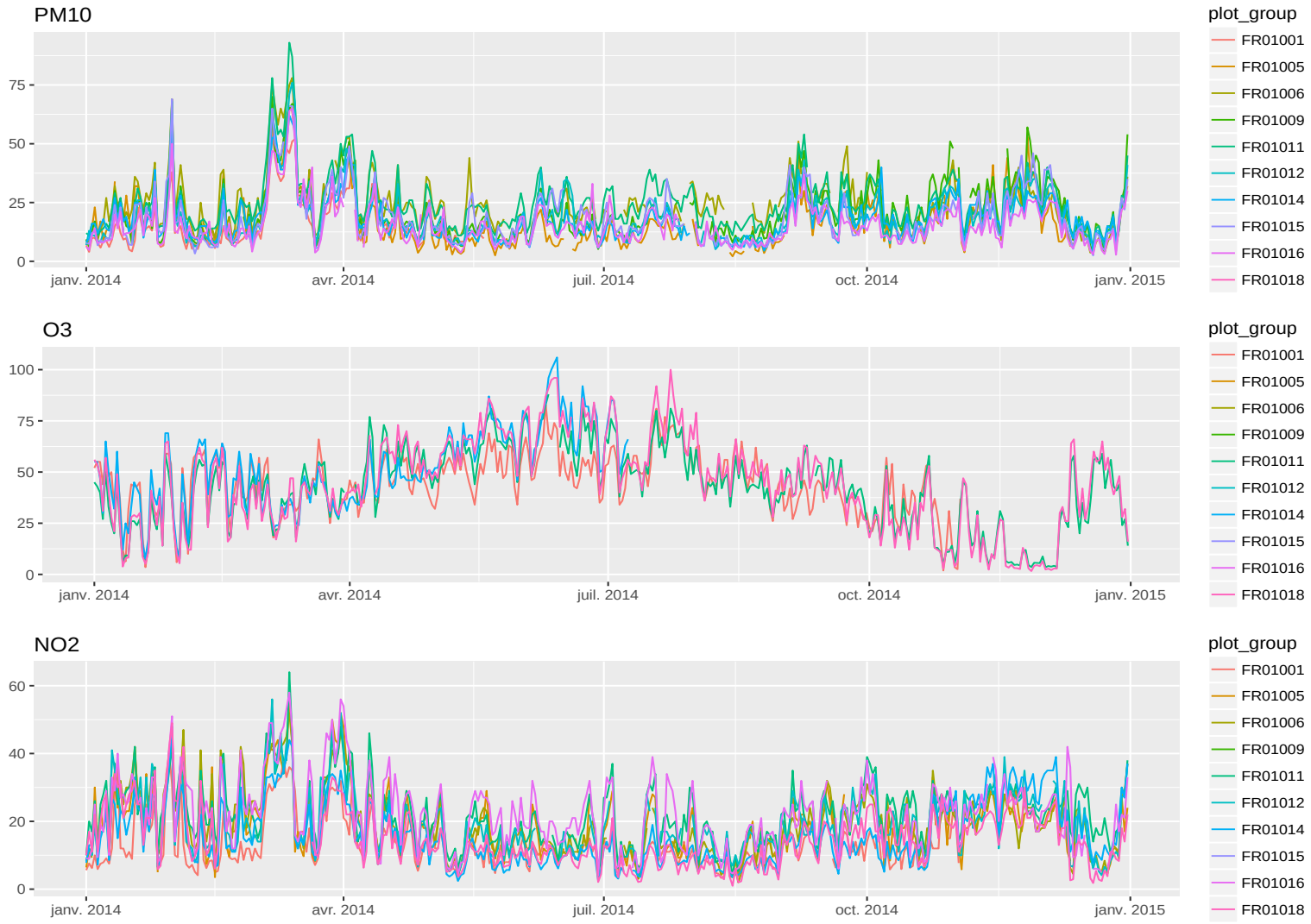
Multilocations and multivariables

Grouped by variable

```
####  4) Multiple locations (>3), multiples variables (grouped by variable)

#####  O3
init <- STFDF_day[which(IDs==IDs[1]),,c("O3")][,1]
selO3 <- matrix(0,ncol=10,nrow=length(init))
for(i in 1:10) selO3[,i] <- STFDF_day[which(IDs==IDs[i]),,c("O3")][,1]
colnames(selO3) <- IDs[1:10]
irr <- irts(index(init),selO3)
p4 <- autoplot(irr, facets=FALSE) + ggtitle('O3')

##### NO2
init <- STFDF_day[which(IDs==IDs[1]),,c("NO2")][,1]
selNO2 <- matrix(0,ncol=10,nrow=length(init))
for(i in 1:10) selNO2[,i] <- STFDF_day[which(IDs==IDs[i]),,c("NO2")][,1]
colnames(selNO2) <- IDs[1:10]
irr <- irts(index(init),selNO2)
p5 <- autoplot(irr, facets=FALSE) + ggtitle('NO2')

new("ggmultiplot",plots=list(p3,p4,p5),ncol=1)
```

Multilocations and multivariables

Grouped by variable



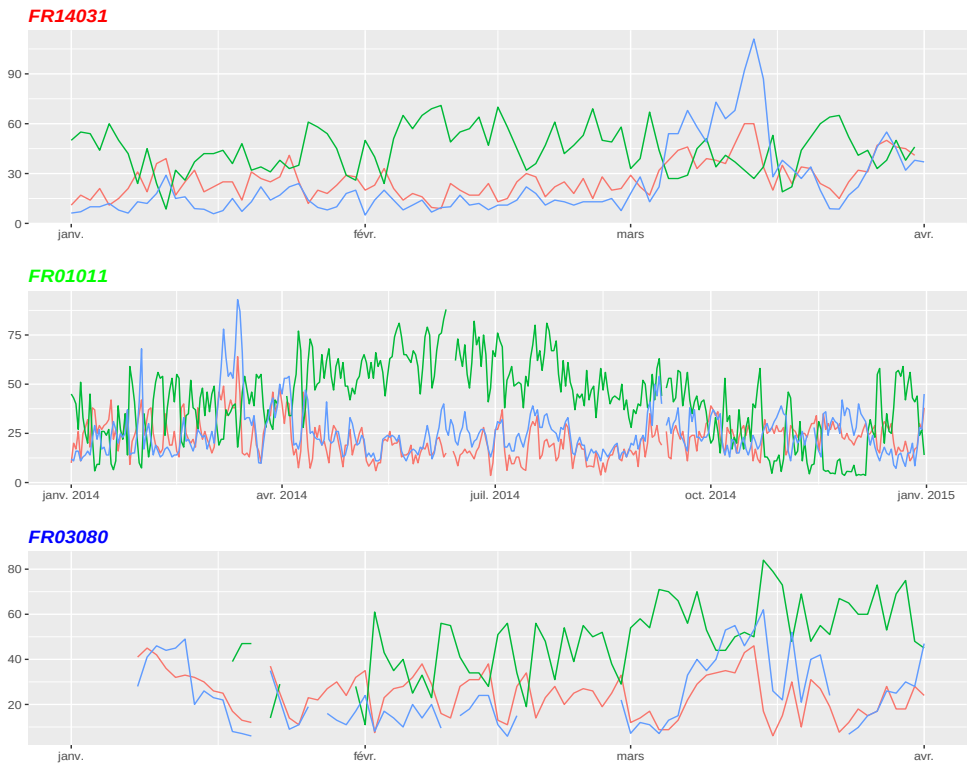
Multilocations and multivariables

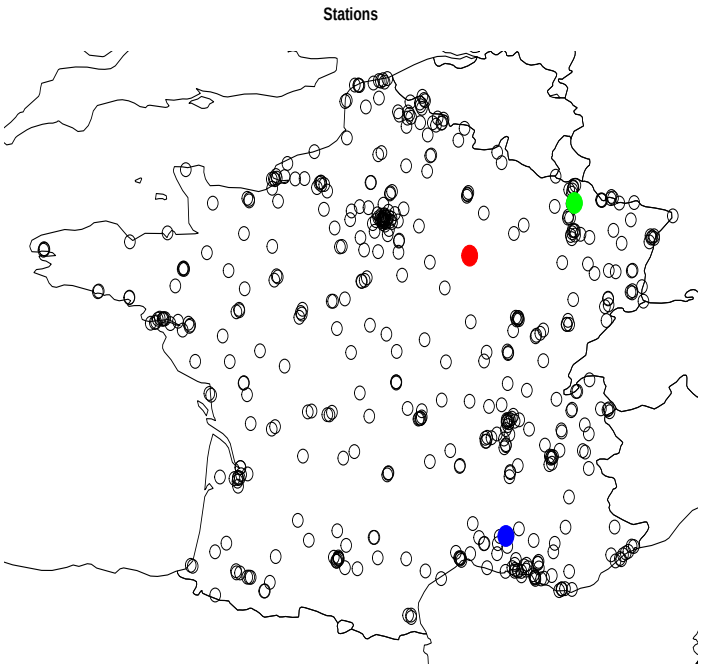
Grouped by location

```
####  5) As before, but grouped by location

ts6 <- STFDF_day[which(IDs %in% c('FR14031')),"2014-01-01::2014-04-01",
       c("PM10","NO2","O3")]
irr <- irts(index(ts6),ts6[,1:3])
p6 <- autoplot(irr, facets=FALSE) + ggtitle('FR14031')
      + theme(plot.title = element_text(color="red", face="bold.italic"))

ts7  <- STFDF_day[which(IDs %in% c('FR01011')),"2014-01-01::2015-04-01",
        c("PM10","NO2","O3")]
irr <- irts(index(ts7),ts7[,1:3])
p7 <- autoplot(irr, facets=FALSE) + ggtitle('FR01011')
      + theme(plot.title = element_text(color="green", face="bold.italic"))

ts8  <- STFDF_day[which(IDs %in% c('FR03080')),"2014-01-01::2014-04-01",
        c("PM10","NO2","O3")]
irr <- irts(index(ts8),ts8[,1:3])
p8   <- autoplot(irr, facets=FALSE) + ggtitle('FR03080')
        + theme(plot.title = element_text(color="blue", face="bold.italic"))

new("ggmultiplot",plots=list(p6,p7,p8),ncol=1)
```

Multilocations and multivariables

Grouped by Location





Animations

```
### 10 days, two variables

selNO2_10d <- STFDF_day[,"2014-01-01::2014-01-10",'NO2']
cxNO2_10d <- attr(selNO2_10d,'data')/max(attr(selNO2_10d,'data'),na.rm=TRUE)
stplot(STFDF_day[,"2014-01-01::2014-01-10",'O3'],
       cex=ifelse(is.na(cxNO2_10d$NO2),0,cxNO2_10d$NO2*5),
       col.regions=brewer.pal(10,"Spectral"),cuts=10,colorkey=TRUE,
       animate=0.2,do.repeat=FALSE,sp.layout =list(world,first=TRUE))
```

Animations

```
### The same, without sp.layout: better refresh

stplot(STFDF_day[,"2014-01-01::2014-01-10",'O3'],
       cex=ifelse(is.na(cxNO2_10d$NO2),0,cxNO2_10d$NO2*5),
       col.regions=brewer.pal(10,"Spectral"),cuts=10,colorkey=TRUE,
       animate=0.2,do.repeat=FALSE)
```

Movies

Saving plots sequentially

```
### The animation is saved in mpeg format with high quality
### (requires installation of command line tool imagemagick)
stplot(STFDF_day[,"2014-01-01::2014-01-10",'O3'],
       cex=ifelse(is.na(cxNO2_10d$NO2),0,cxNO2_10d$NO2),
       col.regions=brewer.pal(10,"Spectral"),cuts=10, colorkey=TRUE)
# Need to create a directory 'images' in your working directory
for(i in c('01','02','03','04','05','06','07','08','09','10')){
 par(mar=c(3,3,2,0.5),mgp=c(2,0.5,0),tcl=-0.3)
 cxNO2_1d <- attr(STFDF_day[,paste("2014-01-",i,sep=''),'NO2'],'data')
             /max(attr(STFDF_day[,paste("2014-01-",i,sep=''),'NO2'],'data'),
             na.rm=TRUE)
 jpeg(paste('images/se',i,'.jpg',sep=''),width=5, height=5, units="in",res=600)
 cat(paste('se',i,'.jpg',sep=''),'\n')
 print(spplot(STFDF_day[,paste("2014-01-",i,sep=''),'O3'],
              cex=ifelse(is.na(cxNO2_1d$NO2),0,cxNO2_1d$NO2),
              col.regions=brewer.pal(10,"Spectral"),cuts=10,colorkey=TRUE))
 dev.off() }
# For Linux
system('convert  -delay 25 -quality 100  images/*.jpg  animation10days.mpeg')
system('open animation10days.mpeg')
```

Movies

Saving a video

```
### Example in Windows
### Need to install ffmpeg

library(animation)
ani.options(ffmpeg = "/yourpath/ffmpeg.exe")
saveVideo({
  stplot(STFDF_day[,"2014-01-01::2014-01-10",'O3'],
         cex=ifelse(is.na(cxNO2_10d$NO2),0,cxNO2_10d$NO2*10),
         col.regions=brewer.pal(10,"Spectral"),cuts=10,colorkey=TRUE,
         animate=0.2,do.repeat=FALSE)},
  interval=0.8, video.name = "anim.mp4", other.opts = "-b 200k")
```

Temporal dependence structures

Autocorrelation

It refers to the correlation of the time series at a given time t with its own past/future values $t-1$, $t-2$, ... / $t+1$, $t+2$, ...

For a time series $X(t)$ with mean μ and variance σ^2, the autocorrelation of lag k is:

$$\rho(k) = \frac{1}{\sigma^2}\mathbb{E}[(X(t)-\mu)(X(t+k)-\mu)]$$

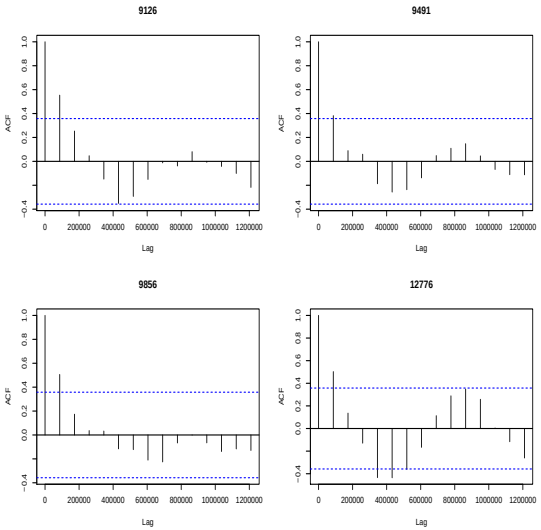
Cross-correlation

For two time series $X(t)$ and $Y(t)$ with mean μ_X, μ_Y and variance σ_X^2, σ_Y^2, the cross-correlation of lag k is:

$$\rho_{XY}(k) = \frac{1}{\sigma_X \sigma_Y}\mathbb{E}[(X(t)-\mu_X)(Y(t+k)-\mu_Y)]$$

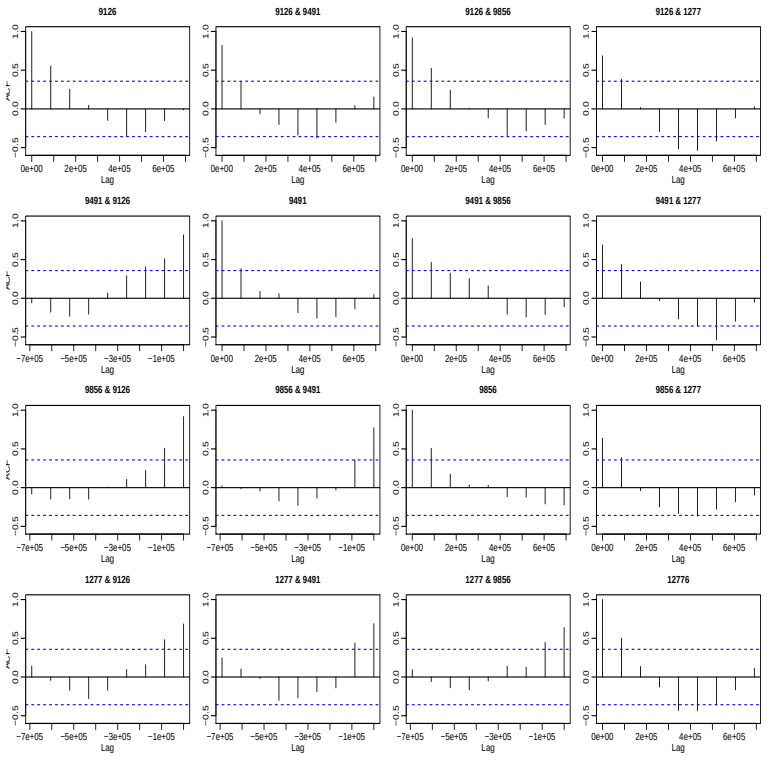
Temporal dependence structures

```
library(gstat)
load('data/STFDF_daily.Rdata')
### Temporal auto-correlation
### choose station with no NA for O3 for the period 2014-01-01::2014-01-30
rn <- IDs[c(26:28,36)]
## Multiple stations, one variable
## lags are expressed in secondes, not days
par(mfrow=c(2,2))
for(i in 1:4)
acf(STFDF_day[which(IDs %in% rn[i]),"2014-01-01::2014-01-30","O3"]$O3,
    main=IDs[i])
```



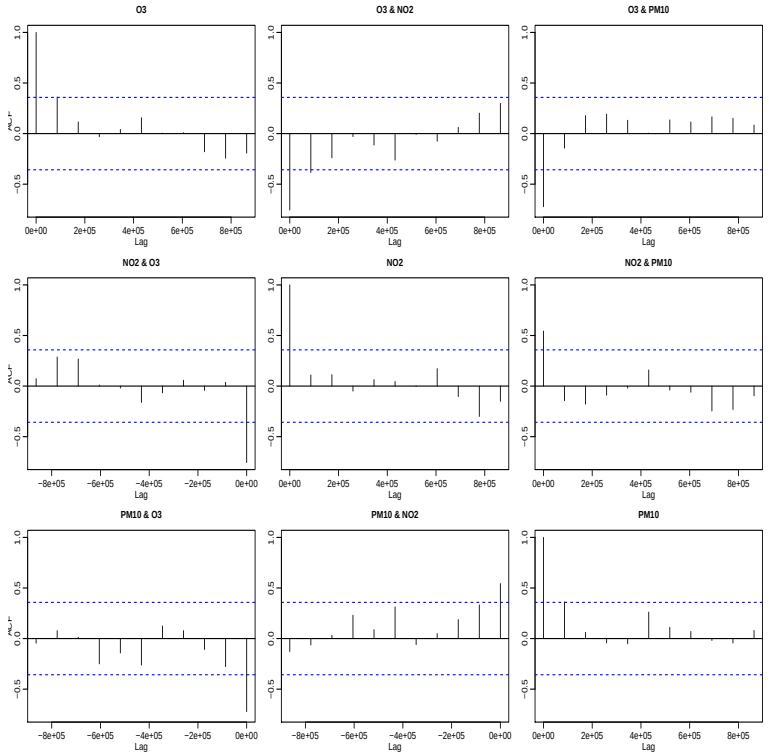
Cross correlation

```
## Cross correlation between multiple stations and one variable

par(mfrow=c(1,1))
acf(as(STFDF_day[c(26:28,36),"2014-01-01::2014-01-30","O3"],"xts"))
```



Cross correlation

```
## Cross correlation between one stations and three variables

acf(STFDF_day[5,"2014-01-01:::2014-01-30",c("O3","NO2","PM10")][,1:3])
```



Spatial dependence structures

Spatial correlation
It describes the relationship between the spatial proximity among observational units and the numerical similarity among their values.

The variogram is defined as the variance of the difference between the field values at two locations s_i and s_j:

$$\begin{aligned} 2\gamma(s_i, s_j) &= \mathbb{E}\left[((Z(s_i) - \mu_Z(s_1)) - (Z(s_j) - \mu_Z(s_j)))^2\right] \\ &= \mathbb{E}\left[(Z(s_i) - Z(s_j))^2\right], \text{ if Z has constant mean} \end{aligned}$$

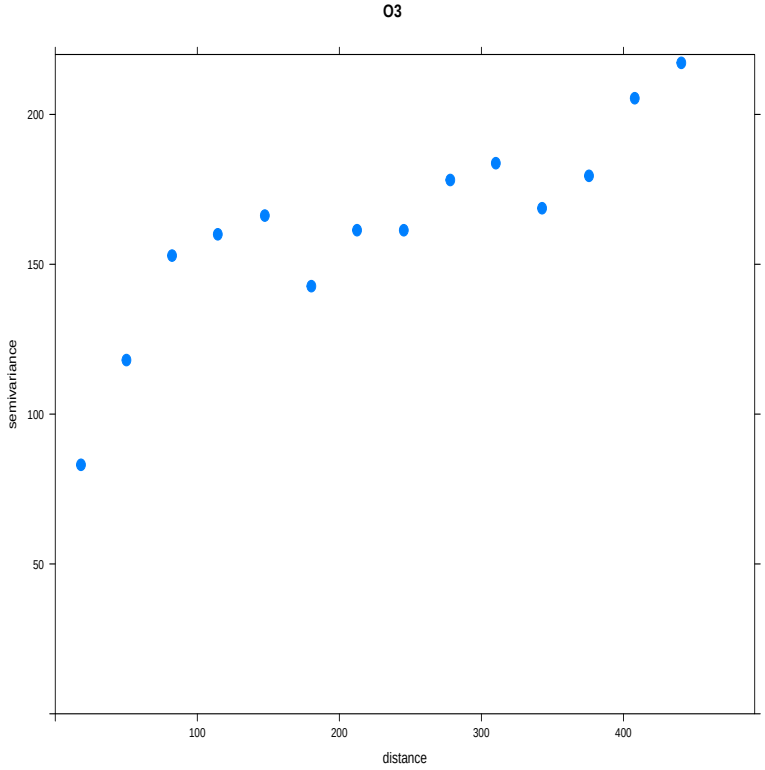
Spatial empirical (semi)variogram:

$$\widehat{\gamma}(h) = \frac{1}{2\sharp N(h)} \sum_{(i,j)\in N(h)} (Z(s_i) - Z(s_j))^2,$$

where $\sharp N(h)$ is the number of pairs in the set $N(h) = \left\{(s_i, s_j) : \mathbf{dist}(s_i, s_j) = h\right\}$.

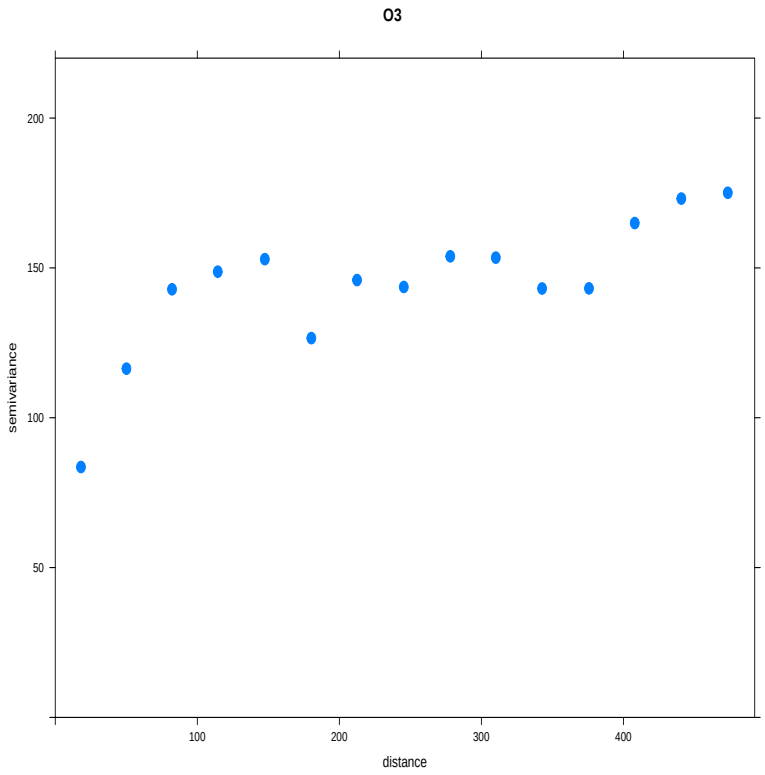
Spatial dependence structures

```
## Spatial auto-correlation for fixed date and one variable

library(gstat)
selO3 <- STFDF_day[!is.na(STFDF_day[,"2014-01-01","O3"]$O3),"2014-01-01","O3"]
v1 <- variogram(O3~1,selO3)
plot(v1,main="O3",pch=19,ylim=c(0,220))
```



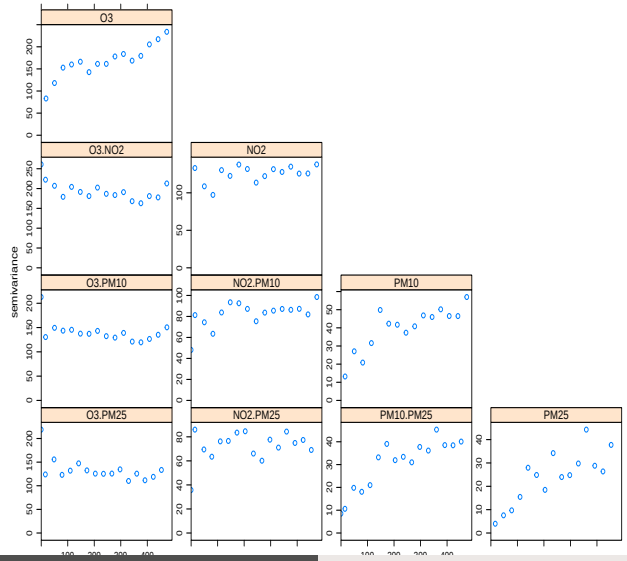
Spatial dependence structures

```
v <- variogram(O3~lat+long,selO3)
plot(v,main="O3",pch=19,ylim=c(0,220))
```



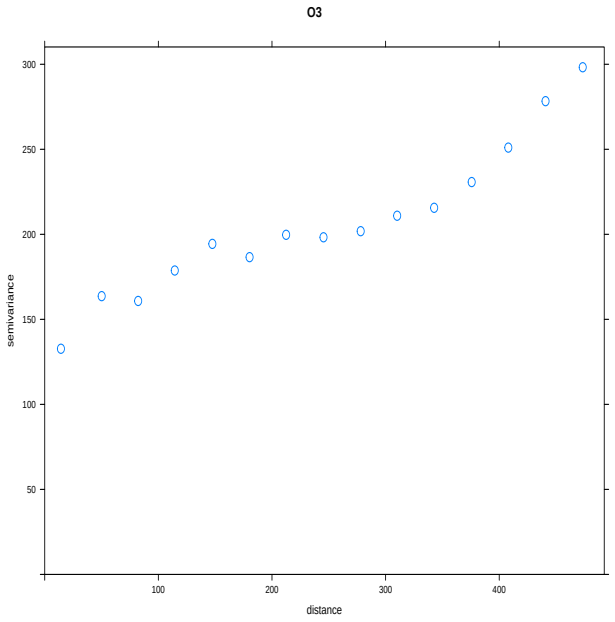
Spatial dependence structures

```
## Spatial auto-correlation for fixed date and multiple variables

sel <- STFDF_day[,"2014-01-01",c("O3","NO2","PM10","PM25")]
g <- gstat(NULL, "O3", O3 ~ 1, data = sel[!is.na(sel[,"O3"]$O3),"O3"])
g <- gstat(g, "NO2", NO2 ~ 1, data = sel[!is.na(sel[,"NO2"]$NO2),"NO2"])
g <- gstat(g, "PM10", PM10 ~ 1, data = sel[!is.na(sel[,"PM10"]$PM10),"PM10"])
g <- gstat(g, "PM25", PM25 ~ 1, data = sel[!is.na(sel[,"PM25"]$PM25),"PM25"])

vm <- variogram(g)
plot(vm)
```



Spatial dependence structures

```
## mean variogram on multiple dates

lst <- lapply(1:5,function(i){x=STFDF_day[,i,"O3"]; x$ti=i ;
                              rownames(x@coords)=NULL;x})
pts <- do.call(rbind,lst)

vv <- variogram(O3~ti,pts[!is.na(pts$O3),])
plot(vv, main="O3")
```



Spatio-temporal dependence structures

Spatio-temporal variogram:

$$2\gamma((s_1,t_1),(s_2,t_2)) = \mathbb{E}\left[(Z(s_1,t_1) - Z(s_2,t_2))^2\right].$$

Related to the covariance function under the assumption of stationarity:

$$\gamma(s_1 - s_2, t_1 - t_2) = \sigma^2 - C(s_1 - s_2, t_1 - t_2).$$

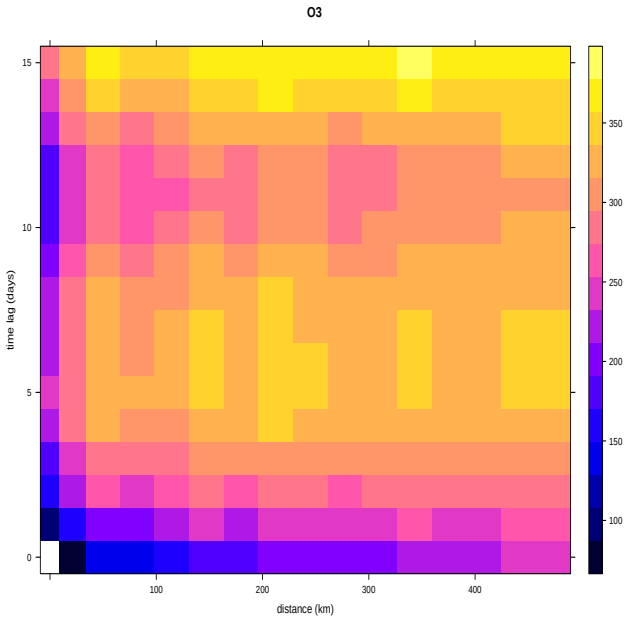
Spatio-temporal empirical variogram:

$$\widehat{\gamma}(h,u) = \frac{1}{2\sharp N(h,u)} \sum_{N(h,u)} (Z(s_i,t_i) - Z(s_j,t_j))^2,$$

where $\sharp N(h,u)$ is the number of pairs in the set $N(h,u) = \left\{\left((s_i,t_i),(s_j,t_j)\right) : s_i - s_j = h, \text{ and } t_i - t_j = u\right\}$.

Spatio-temporal dependence structures

```
## Spatio-temporal auto-correlation

selO3st <- STFDF_day[!is.na(STFDF_day[,"2014-01-01::2014-01-31","O3"]$O3),
                                        "2014-01-01::2014-01-31","O3"]
vst <- variogram(O3~ lat + long, selO3st)
plot(vst,main="O3")
```



Spatio-temporal dependence structures

```
vvst <- variogram(O3~1,STFDF_day[!is.na(STFDF_day[,"2014-01-01::2014-12-31",
                    "O3"]$O3),"2014-01-01::2014-12-31","O3"],
                    width=5,cutoff=200,tlags=0:5)
plot(vvst)
plot(vvst,map=FALSE)
plot(vvst,wireframe=T)
```