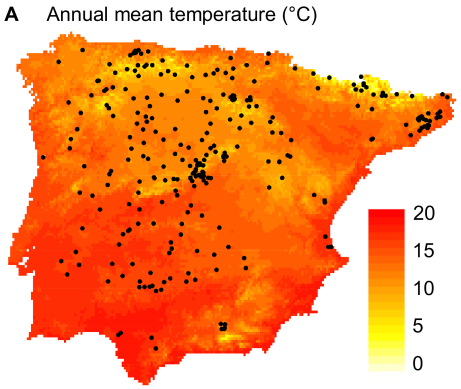
Lecture 25. GLS for spatial data.

Applying GLS to spatial data is conceptually the same as the time series case: we want to use spatially extensive data to model the effect of some predictor(s) on the response, but we also need to account for any spatial autocorrelation in the response, in order to have valid inferences. The main differences are: 1) now we're working in two dimensions, and 2) now we have data that is not spaced at regular intervals, unlike the annual time series we looked at previously. In reality temporal data can also have irregular spacing (e.g. haphazard sampling over time), and so the methods we discuss for space are also useful for time series.

Let's look at some example data. This is from a paper by Manzano-Piedras et al. (2014, PloS One, "Deciphering the Adjustment between Environment and Life History in Annuals: Lessons from a Geographically-Explicit Approach in *Arabidopsis thaliana*"). The authors took seeds from 279 individuals of the model plant *Arabidopsis*, which were spread around the Iberian peninsula (essentially Spain and Portugal).



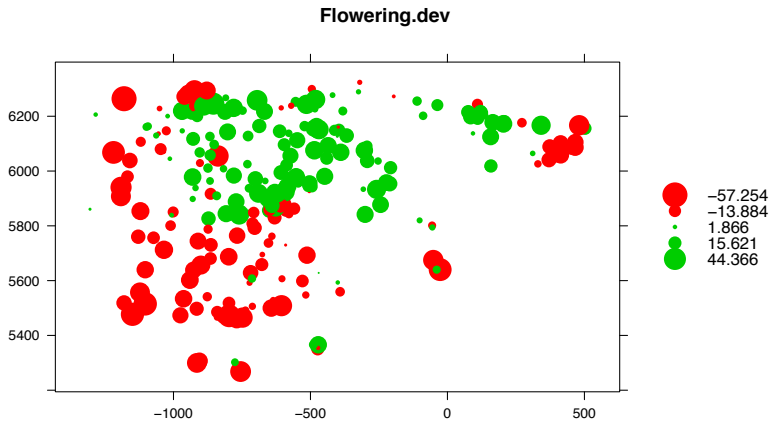
The plot above shows the sample locations as well as the mean annual temperature at those locations. The authors took these seeds, grew a new generation in the greenhouse (they are self-compatible), and then planted the second generation in a botanical garden to measure a bunch of life history traits on each accession (the descendants of each individual). Here we'll focus on flowering time, which in this experiment is the number of days between when the seeds were sown and when the germinated individuals flowered. Flowering time is a key life history trait for annual plants, because it sets the pace for subsequent events (fertilization, seed set, etc.) and therefore has a strong effect on individual fitness. The authors wanted to compare flowering times (and other traits) of the accession to the environmental conditions where they were collected, to sort out the main environmental drivers of life history variation. To do this kind of analysis, one can use a LM or GLM or GAM, but the spatial structure of the data needs to be considered as well. It is likely that individuals from nearby locations have more similar traits than individual from distance locations, which will lead to non-independence in the data.

To start off, let's visualize the variation in flowering time over space. First I'm going to convert the Lat-Lon coordinates to UTM (Universal Transverse Mercator) coordinates. UTM coordinates essentially convert Lat-Lon to distances in meters. This is important for our purposes, because degrees latitude and degrees longitude do not have the same length, and because the degrees longitude represent less distance at higher latitudes.

```
library(rgdal)
xy = cbind(arab$Latitude, arab$Longitude)
utms = project(xy, "+proj=utm +zone=30 ellps=WGS84")
arab$northing = utms[,1]/1000
arab$easting = utms[,2]/1000
```

Here I used the geostats package 'rgdal'. The project() function takes a matrix of latitudes and longitudes, and converts them to UTM (the other options say to use zone 30, which is where Iberia is, and to use the WGS84 representation of earth). Then I divide the northing and easting coordinates by 1000, so that the numbers are in units of kilometers instead of meters (easier to look at on a plot).

```
library(sp)
arab.sp = arab
coordinates(arab.sp) = c('easting', 'northing')
arab.sp$Flowering.dev = arab.sp$Flowering.time - mean(arab.sp$Flowering.time)
bubble(arab.sp, zcol = 'Flowering.dev', scales = list(draw = T))
```



Here I've used the bubble() function in the 'sp' package. I plotted the the deviation from the mean flowering time, to help visualize values greater than the mean (green) and less than the mean (red). Mean flowering time is 143 days, and it ranges from 86 to 188 days in the experiment. From the bubble plot we can see there are strong spatial patterns, such that flowering time is much later in the north-

centralish part of the peninsula, which is also where altitude tends to be higher and temperature tends to be lower.

The spatial variogram and correlogram

For our purposes, we're interested in whether this pattern can be predicted by environmental variables like temperature, and also what the spatial autocorrelation structure looks like. Let's visualize the spatial correlation of the raw data first, to get a sense for what it looks like. The most common tool is the variogram (often called the semivariogram, which is confusing), which we already saw back when we were doing GAMs. The formula for the empirical variogram is:

$$\hat{\gamma}(h) = \frac{1}{2N(h)} \sum_{(i,j) \in N(h)} (z_i - z_j)^2$$

Here $\hat{\gamma}(h)$ is the estimated variance of observations that are separated by a distance h; $N(h)$ is the number of observations that are separated by a distance h; and z_i and z_j are values of two observations i and j, that are separated by distance h. The assumption for spatial data is that observations close to each other are more likely to be similar, and this function allows us to visualize to what extent that is true. In order to actually calculate the variogram, we have to take all pairs of observations and bin them into groups based on how far apart they are, and then use those bins to calculate the variance of the data at that distance. An idealized variogram looks like this:

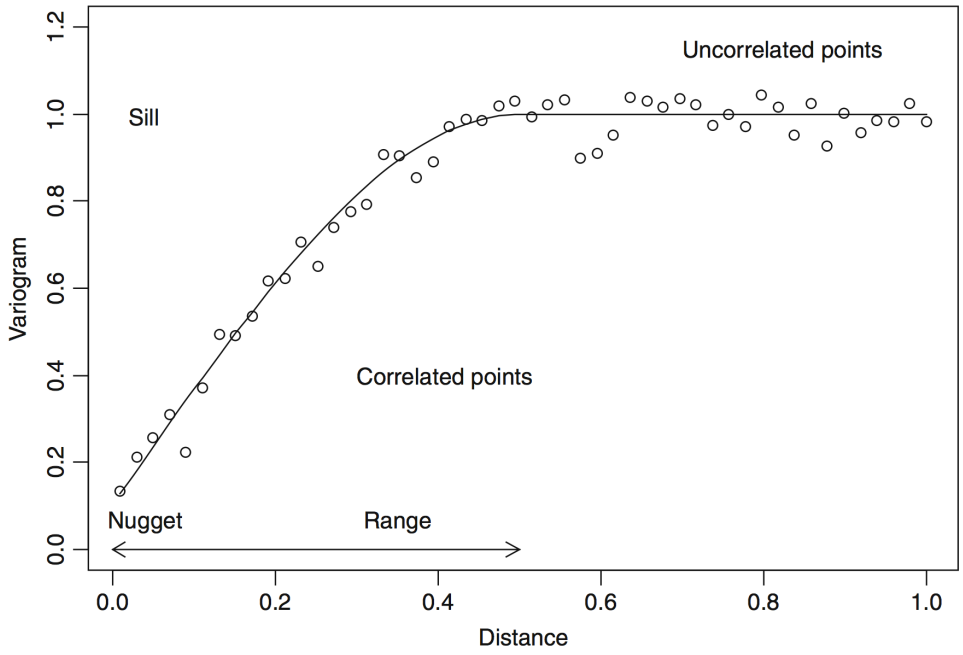
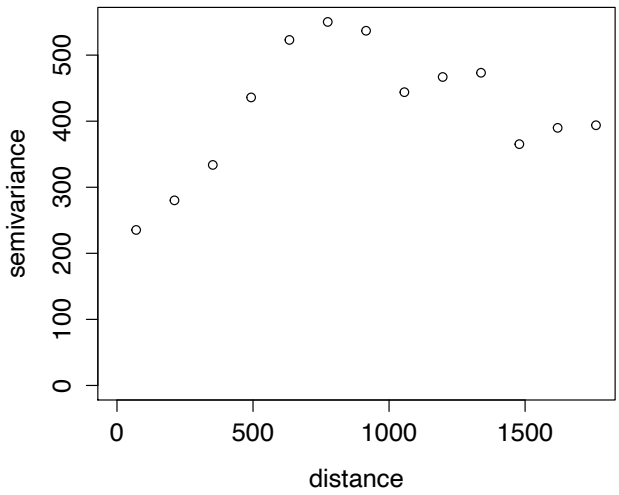


Fig. 7.2 Variogram with fitted line. The sill is the asymptotic value and the range is the distance where this value occurs. Pairs of points that have a distance larger than the range are uncorrelated. The nugget effect occurs if $\hat{\gamma}(\mathbf{h})$ is far from 0 for small $\boldsymbol{h}$

Because nearby observations are similar, there is lower variance among them. That causes the variance to increase with distance, until it reaches an asymptote, at which point there is no more signal of spatial autocorrelation. The 'range' is the distance at which the variogram levels off. The 'sill' is the value of the variogram when is levels off (the asymptote). The 'nugget' is the value of the variogram when the distance is zero. This represents the amount of variation among samples taken from the same location, which could be due to numerous causes.

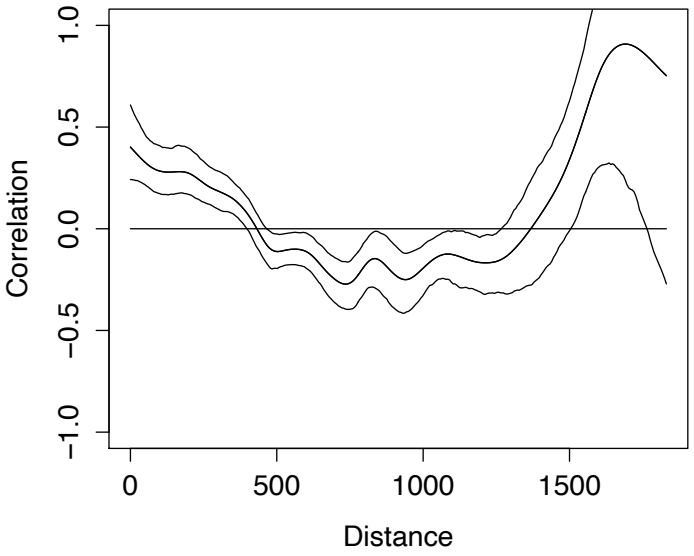
Let's plot one for the *Arabidopsis* data:

```
library(geoR)

v1 <- variog(coords = arab[,c('easting', 'northing')], data = arab$Flow
ering.time)

plot(v1)
```



This isn't quite the idealized variogram. The variance in flowering time increases to a distance of about 750 km, but then it declines a bit after that. So we definitely have a signal of spatial autocorrelation, but it's a little more complex than the simple "closer points are more similar" model. It's not surprising we don't see the ideal pattern, because we don't think the data came from some random process that has spatial autocorrelation but no other pattern. The reason why the variogram might have a peak is because flowering time might be *negative* correlated at certain distances. To take a look at this, we can make a plot similar to what acf() does, but for spatially continuous data:

```
ncf.scor <- spline.correlog(arab$easting, arab$northing, arab$Flowering.time,
resamp=500, quiet = TRUE)
plot(ncf.scor)
```

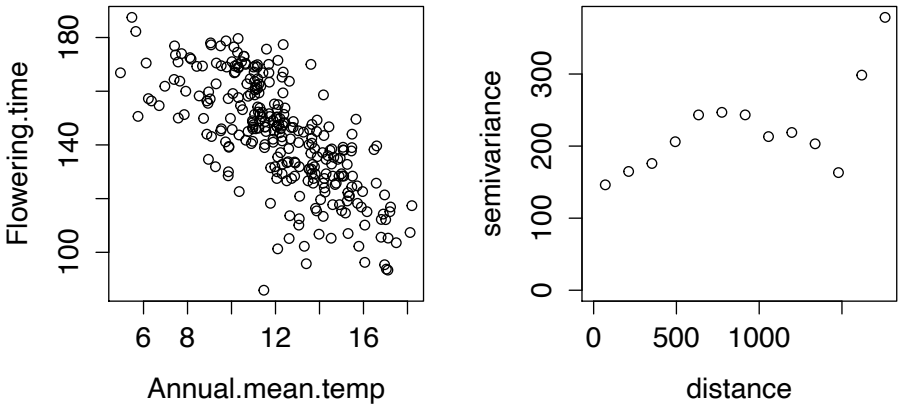


This is a spline correlogram. It uses a non-parametric estimator to get a smooth function for how the correlation between observations changes as the distance between them increases. For flowering time, it tends to be positively correlated up to about 500 km, then it becomes weakly negatively correlated from about 500 – 1250 km, and finally positively correlated at the greatest distances (though with more uncertainty). This pattern is actually pretty intuitive if we compare to the bubble plot of the raw data. There is a lot of smaller-scale similarity, but sites about 700 km apart are very different. And the most distant sites are on the opposite east/west sides of the peninsula, and both have early flowering times. It's really those most extreme distances that are making the plot look different from the classic expectation.

In some ways it doesn't make much sense to spend a lot of time thinking about whether the raw data are well-represented by a variogram or correlogram. Classic geostatistical approaches try to model data as some random process that is *stationary* across space, which means that all points are drawn from the same distribution, and any patterns are created only by spatial correlation. For the flowering time data (and most ecological data) there is clearly important geographic structure, i.e. it's not a stationary random process. However, it's useful for exploratory purposes to make these plots, especially because now we are going to think about putting predictors in the model and whether the *residuals* have any spatial structure.

Modeling spatial autocorrelation with GLS

In this dataset, flowering time is strongly correlated with mean annual temperature (plot on the left):



This relationship is strong enough that it probably doesn't matter whether we account for residual correlation or not. But this will not always be the case, so I will perservere in showing how to model the residuals. To start, we can fit a linear model and see what the residuals look like:

```
mod.nocorr = lm(Flowering.time ~ Annual.mean.temp, data = arab)
v1 <- variog(coords = arab[,c('easting', 'northing')], data =
resid(mod.nocorr))
```

This produces on the plot on the right above. This is not the same pattern from the raw data, but it is similar, and still indicates spatial correlation. It is important to note that temperature could have explained the spatial correlation of the data, because temperature itself is spatially correlated, in which case the residuals could indeed be independently distributed. But here that is not the case.

Now I want to add in a spatial correlation structure for the residuals. To start I'll make new model that has no predictors, just a spatial correlation structure. I'm going to do this out of curiosity, to see how well a simple spatial correlation model can model this data. For this we'll use the gls() function, in the same way as we did for the time series example. But now we need a different correlation model. In nlme there are 5 different correlation models that would be appropriate for this situation; they differ in the details of the shape of the resulting variogram:

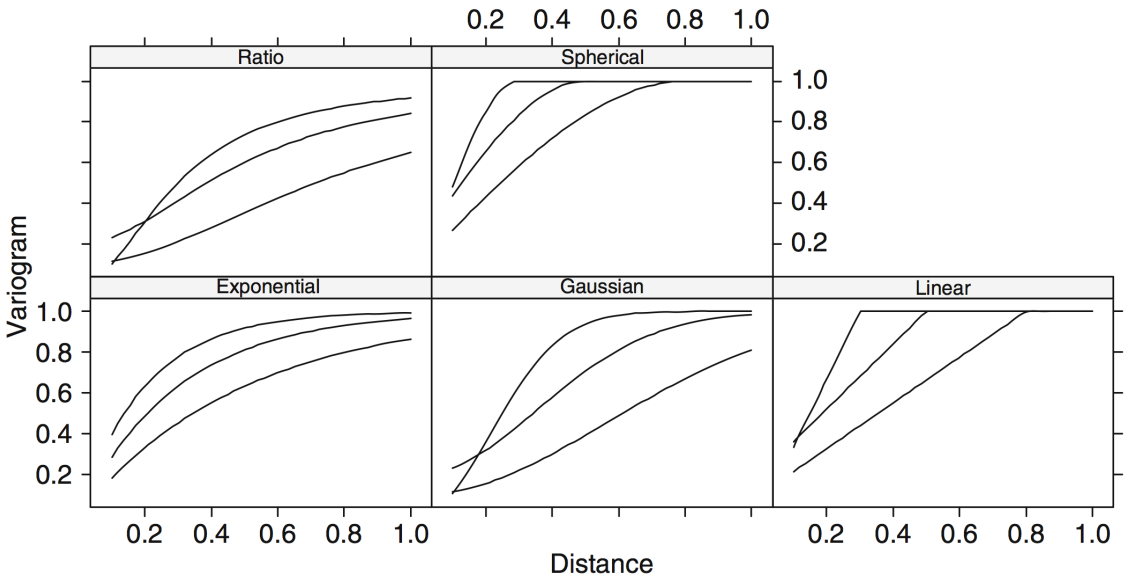


Fig. 7.4 Different variogram patterns. The three lines in the same panel were obtained using different values for the range and nugget

So all these correlation models (Exponential, Gaussian, Linear, Ratio, Spherical) assume that correlation declines with distance. The Exponential model looks like this:

$$\gamma(s,p) = 1 - e^{-\frac{s}{p}}$$

Where p is the parameter for the range (the distance at which correlation is negligible) and s is the distance. This model is similar to the AR(1) model for time series, because similarity is declining exponentially with distance. If there is a 'nugget effect', i.e. some variability among observations from the same site (this seems pretty typical to me), then the equation is

$$\gamma(s,p) = c_0 + (1 - c_0)(1 - e^{-\frac{s}{p}})$$

Now the curve starts at the nugget c_0 when distance (s) is zero, and increases to one as distance increases.

I won't go through the formulas for the other corStruct options; you can see from the plots what their shapes look like, and all of them are parameterized in terms of a 'range' parameter and an optional nugget. Mostly people use the different correlation models in a model selection way to find one that best accounts for the structure in the data. To fit the exponential model to the flowering time data, we do this:

```
mod.gls.nopred = gls(Flowering.time ~1, data = arab, correlation = corExp(form
= ~easting + northing, nugget = TRUE, value = c(1000, 0.8)))
```

I've used correlation = corExp(), and form = ~easting + northing. 'easting' and 'northing' are the UTM coordinates corresponding to longitude and latitude. By specifying form = ~easting+northing, the corExp function will use those two variables to calculate the euclidean distance between each pair of points. And it will use those distance to fit an exponential correlation structure. I've also set nugget = TRUE, because the data definitely do not have zero variance at distance = 0. Finally, I've specified a 'value' argument. These are the starting values for the algorithm that will estimate the range and the nugget for the correlation model. It's important to give reasonable starting values, because the algorithm does not seem super robust in terms of finding the right answer. I gave a range value of 1000, because that is roughly when the variogram levels off, and a nugget of 0.8. The nugget parameter ranges from 0 to 1; 0 means no nugget, and 1 means there is not spatial autocorrelation.

Let's look at the model fit:

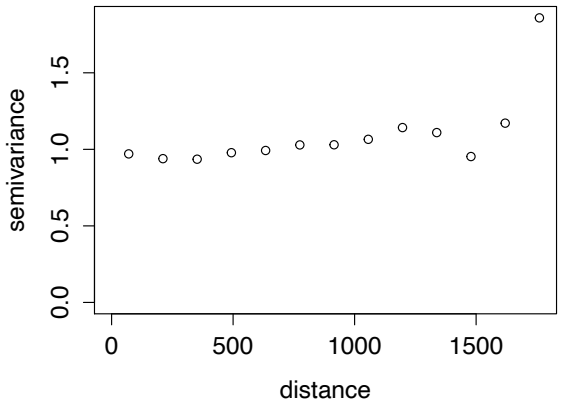
```
summary(mod.gls.nopred)
```

```
## Generalized least squares fit by REML
##   Model: Flowering.time ~ 1
##   Data: arab
##
## Correlation Structure: Exponential spatial correlation
##  Formula: ~easting + northing
##  Parameter estimate(s):
##    range   nugget
## 323.0969   0.2615
##
## Coefficients:
##             Value Std.Error t-value p-value
## (Intercept) 132.1     8.102    16.3       0
##
```

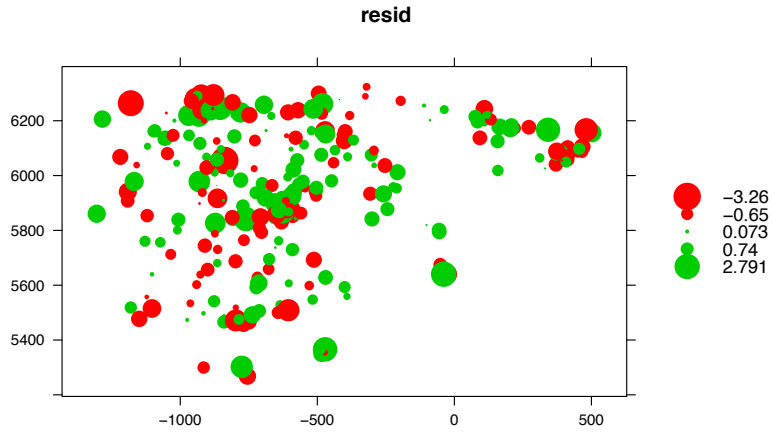
This model has three parameters: the Intercept (the mean of the data), and the range and the nugget for the exponential spatial correlation. The range is 323 and the nugget is 0.26. So this means that spatial correlation isn't important beyond 323 km, and that at small scales it's fairly strong (remember the nugget parameter varies between 0 and 1, and 0 is the strongest correlation). This is roughly consistent with what we saw from the empirical variogram.

OK now let's see what the residual variogram looks like for this model:

```
v1 <- variog(coords = arab[,c('easting', 'northing')], data = resid(mod.gls.no
pred, type = "normalized"))

plot(v1)
```



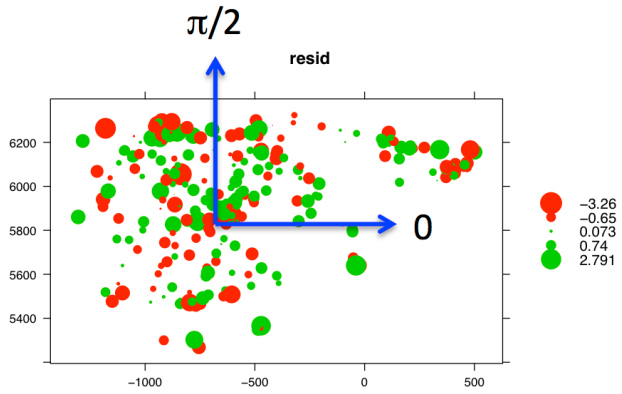
Now I'm plotting the normalized residuals, i.e. with the modeled spatial correlation effect removed. This is quite flat, except for the weird blip at the largest distance; that point is based on fewer samples, so I'm going to ignore it. Based on this, it looks like the exponential correlation model successfully accounts for the correlation structure of the data. We can also plot the residuals in space using a bubble plot:



The strong spatial pattern in the raw flowering time data is not as evident here. The implication is that the strong geographic pattern in the data has been accounted for as 'random' autocorrelation on a very large scale.

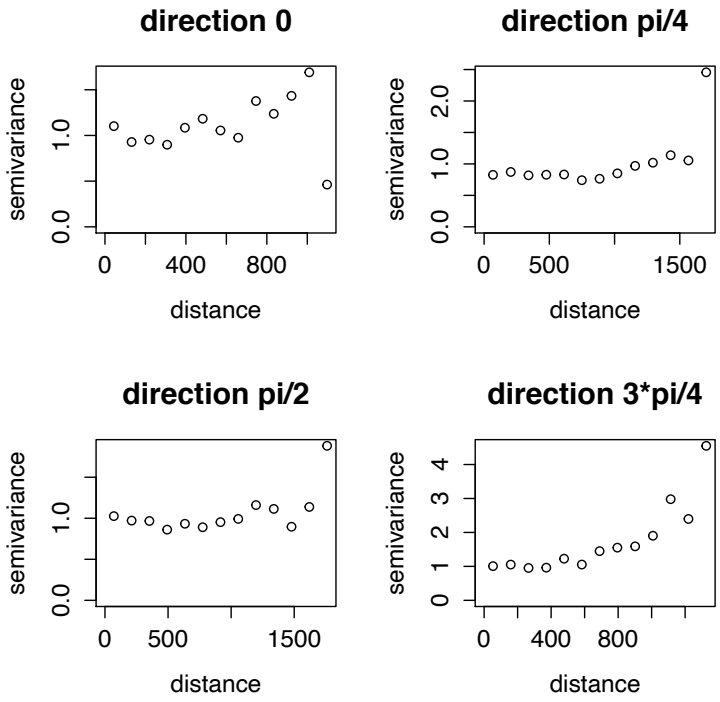
One thing I haven't mentioned yet is the issue of directionality in space. When we were modeling a time series, there is only one dimension of 'distance' to consider, i.e. forwards and backwards in time. For spatial data it is possible that patterns look different when we measure distance along different directions. So far I've treated all distances as equivalent, which means I'm assuming the patterns are *isotropic*. It's

possible that there's no autocorrelation on average, when considering all distances to be equivalent, but that there is still some autocorrelation in particular directions. We can visualize the directions on the bubble plot like this:



Here I'm measuring angles in radians (recall your basic trigonometry). I've drawn an angle of 0 and $\pi/2$, as examples. We can use the variogram function to look at correlation when we measure the distance between points only along the axis at a certain angle:

```
par(mfrow = c(2,2))
v1 <- variog(coords = arab[,c('easting', 'northing')], data = resid(mod.gls.no
pred, type = "normalized"), direction = 0)

plot(v1, main = 'direction 0')
v1 <- variog(coords = arab[,c('easting', 'northing')], data = resid(mod.gls.no
pred, type = "normalized"), direction = pi/4)

plot(v1, main = 'direction pi/4')
v1 <- variog(coords = arab[,c('easting', 'northing')], data = resid(mod.gls.no
pred, type = "normalized"), direction = pi/2)

plot(v1, main = 'direction pi/2')
v1 <- variog(coords = arab[,c('easting', 'northing')], data = resid(mod.gls.no
pred, type = "normalized"), direction = 3*pi/4)

plot(v1, main = 'direction 3*pi/4')
```



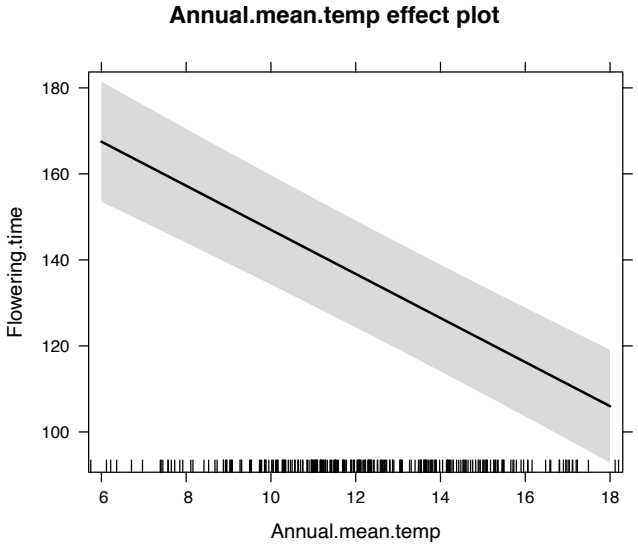
It looks like directions $\pi/4$ and $\pi/2$ look good, but direction 0 and direction $3\pi/4$ have some signal of autocorrelation. Trying to account for this extra complexity is possible though challenging; in this case, hopefully including environmental predictors will account for this signal. If not, it is generally much better to account for some of the spatial correlation structure, even if the fit is not perfect, than to not account for it at all.

Now I'll finally see what happens when we combine a predictor with the spatial correlation component:

```
mod.gls1 = gls(Flowering.time ~ Annual.mean.temp, data = arab, correlation = c
orExp(form = ~easting + northing, nugget = TRUE, value = c(1000, 0.8)))
summary(mod.gls1)
```

```
## Generalized least squares fit by REML
##   Model: Flowering.time ~ Annual.mean.temp
##   Data: arab
##
## Correlation Structure: Exponential spatial correlation
##  Formula: ~easting + northing
##  Parameter estimate(s):
##    range   nugget
## 510.6430   0.4853
##
## Coefficients:
##                    Value Std.Error t-value p-value
## (Intercept)      198.25     8.733   22.70       0
## Annual.mean.temp  -5.13     0.460  -11.15       0
##
```

The effect of temperature is negative and highly significant. The correlation model also appears to still be important; the parameter estimates are similar but no identical. We can plot a residual variogram again, but I'll skip it because it looks very similar to the last one. We can make an effects plot for temperature:\

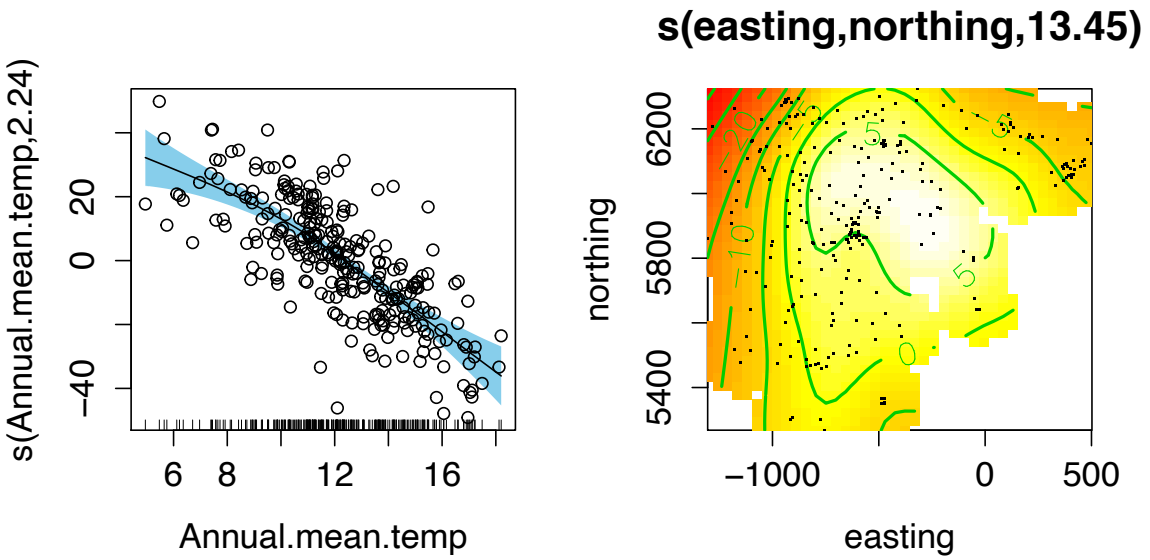


The effect of temperature on flowering time is quite large; across the range of annual temps, which vary by 12 degrees C, flowering time goes from 170 days to 110 days. Remember that this is not a direct effect of temperature; the plants were grown in a common garden, and this is the relationship between environmental temperature and genetic variation in flowering time. Why might this be, why would plants from warmer environments flower sooner? The authors suggest that they are speeding up their development to avoid unfavorable conditions (hot dry summers). The authors also looked at a number of other predictors, such as precipitation, but that will lead us too far afield.

GLS vs GAM

Back when we were introducing GAMs, we used a 2D spatial smoother to model spatial variation in mackerel egg density, and in coyote-wolf hybrid ancestry. This approach is particularly useful if you want to model spatial trends while also including other potential predictors in a model. It seems like a 2D smoother might be a good general alternative to using GLS to model spatial autocorrelation. Keep in mind that these two approaches are doing different things: a smoother is modeling variation in the mean of the response, while an autocorrelation model is modeling residual similarity as a function of distance. Nonetheless, it may often be the case that autocorrelation is driven mostly by spatial trends that can be modeled with a GAM. Let's try this for the flowering time data as a comparison.

```
mod.gam = gam(Flowering.time ~ s(easting, northing) + Annual.mean.temp, data =
 arab)
plot(mod.gam, scheme = 2, lwd = 2, labcex = 1)
```

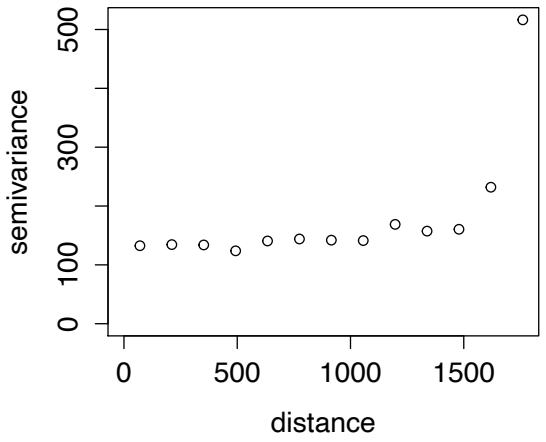



```
mod.gam = gam(Flowering.time ~ s(easting, northing) + s(Annual.mean.temp),
data = arab)
```

```
plot(mod.gam, select = 2, residuals = T, pch = 21, shade = T, shade.col = 'sky
blue')
plot(mod.gam, scheme = 2, lwd = 2, labcex = 1, select = 1)
```

Since I was fitting a GAM anyways, I also used a smoother for temperature. It is still nearly linear, but slightly concave down. The 2D spatial smoother still finds an additional signal of delayed flowering time in the middle of the peninsula, and decreasing from there, especially to the west. How do the residuals for this model look?

```
v1 <- variog(coords = arab[,c('easting', 'northing')], data = resid(mod.gam))

## variog: computing omnidirectional variogram

plot(v1)
```



There might be a hint of an upward trend (plus that final outlier I've been ignoring), but this definitely looks much better than the raw data. In this case it looks like a spatial smoother or GLS can both capture the spatial correlation structure pretty well.

Ways to deal with spatial or time series data

To sum up, we can compare the different ways you might build a regression model that is appropriate for spatial or time series data. For spatial data, we have considered four different possibilities:

1) Use the right predictors. The response variable may be non-independent across space, but the predictors may explain that spatial structure, resulting in residual variation that is spatially uncorrelated.
2) Model the residual correlation with GLS. If the residual pattern is primarily that closer site are more similar, this will work especially well.
3) Model the spatial trends with a smoother (GAM). This will not necessarily capture any spatial correlation pattern, but if there are big spatial trends then this will usually work pretty well.
4) Model space as a set of spatial groups using a mixed model. If you can chunk space into 'sites' or 'regions', and use a random effect for the spatial groups, this can do a good job of accounting for the fact that nearby observations are similar. This is coarser than the other options, but often simpler to use, especially if you are already using a mixed model. You can check the residuals and random effect estimates to see if there is still strong spatial correlation.

Using these different options will depend on the dataset and your question. In addition, all of these options can be used for time series as well. E.g., time can be chunked into years (with multiple samples per year), or a smoother can be fit to capture temporal trends. If necessary, multiple methods can be combined; a mixed

model with a smoother as well as a residual correlation model can be made, e.g. with the gamm() function in mgcv, though this level of complexity is definitely more difficult to implement correctly.