Trend Analysis

Peter Claussen

9/5/2017

Libraries

rsm : Response-Surface Analysis

```
library(rsm)
```

```
library(ape)
```

```
## Warning: package 'ape' was built under R version 3.3.2
```

```
library(ncf)
```

```
##
## Attaching package: 'ncf'
```

```
## The following object is masked from 'package:ape':
##
##     mantel.test
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#0072B2", "#D55E00", "#F0E442","#CC79A7","#
```

Data

```
load(file="autocorrelation.Rda")
attach(autocorrelation.dat)

load(file="sample.dat.Rda")
```

Stationarity

Many of the models for autocorrelation assume a stationary mean. The expected value, excluding random effects, is assumed to be constant everywhere.

That assumption will frequently fail, particularly in the case where we are using yield monitor data to establish a fertility map, assuming that some regions of a field will be generally more or less fertile than other regions.

If the mean is not constant, and instead it changes with position, then we no longer can assume stationarity. However, we can account for non-stationarity and removing the moving mean. This may involving fitting data to a trend surface, so we can refer to this as detrending.

Artificial Example in One Dimension

If we remember the trend plus error example from our discussion of autocorrelation,

We can't fit this using the `arima` function

```
> arima(trend.error,c(1,0,1))
Error in arima(trend.error, c(1, 0, 1)) : non-stationary AR part from CSS
```

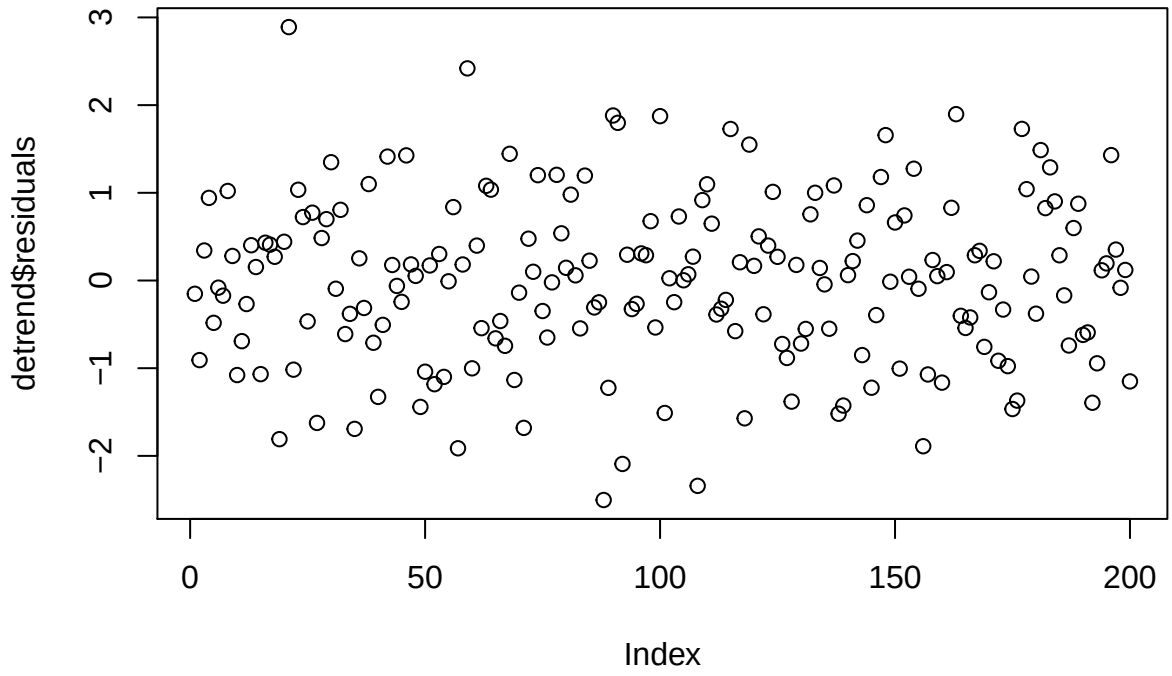
We generated a non-stationary mean by approximating a random walk with a polynomial. Adding white noise to this fit was not enough to convert this series back to a random walk, and this series fails the assumption of non-stationarity.

However, we can detrend this series.

```
x <- 1:length(trend.error)
detrend <- lm(trend.error ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5))
```

and see if there is correlation among the residuals. Since we added white noise to a polynomial, the residuals from a polynomial fit should themselves be white noise.

```
plot(detrend$residuals)
```




```
arima(detrend$residuals,c(1,0,0))
```

```
##
## Call:
## arima(x = detrend$residuals, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
##       -0.0416     0.0003
## s.e.   0.0707     0.0638
##
## sigma^2 estimated as 0.8838:  log likelihood = -271.44,  aic = 548.87
```

```
arima(detrend$residuals,c(0,0,1))
```

```
##
## Call:
## arima(x = detrend$residuals, order = c(0, 0, 1))
##
## Coefficients:
##           ma1  intercept
##       -0.0576     0.0004
## s.e.   0.0846     0.0627
##
## sigma^2 estimated as 0.8832:  log likelihood = -271.37,  aic = 548.75
```

```
arima(detrend$residuals,c(1,0,1))
```

```
##
## Call:
## arima(x = detrend$residuals, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.7989  -1.0000     0.0000
## s.e.  0.0434   0.0135     0.0051
##
## sigma^2 estimated as 0.7942:  log likelihood = -262.32,  aic = 532.65
```

Now, consider the case if we don't know the correct order of polynomial to fit to a series. We might try

```
arima(lm(trend.error ~ poly(x,1))$residuals,c(1,0,0))
```

```
##
## Call:
## arima(x = lm(trend.error ~ poly(x, 1))$residuals, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.8636     0.3591
## s.e.  0.0387     0.6761
##
## sigma^2 estimated as 1.759:  log likelihood = -340.97,  aic = 687.95
```

```
arima(lm(trend.error ~ poly(x,2))$residuals,c(1,0,0))
```

```
##
## Call:
## arima(x = lm(trend.error ~ poly(x, 2))$residuals, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.7971     0.1242
## s.e.  0.0434     0.4441
##
## sigma^2 estimated as 1.679:  log likelihood = -336.12,  aic = 678.24
```

```
arima(lm(trend.error ~ poly(x,3))$residuals,c(1,0,0))
```

```
##
```

```
## Call:
## arima(x = lm(trend.error ~ poly(x, 3))$residuals, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.7205     0.0828
## s.e.  0.0504     0.3181
##
## sigma^2 estimated as 1.614:  log likelihood = -332.05,  aic = 670.1
```

```
arima(lm(trend.error ~ poly(x,4))$residuals,c(1,0,0))
```

```
##
## Call:
## arima(x = lm(trend.error ~ poly(x, 4))$residuals, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.3055    -0.0029
## s.e.  0.0679     0.1119
##
## sigma^2 estimated as 1.213:  log likelihood = -303.15,  aic = 612.3
```

```
arima(lm(trend.error ~ poly(x,5))$residuals,c(1,0,0))
```

```
##
## Call:
## arima(x = lm(trend.error ~ poly(x, 5))$residuals, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
##       -0.0416     0.0003
## s.e.   0.0707     0.0638
##
## sigma^2 estimated as 0.8838:  log likelihood = -271.44,  aic = 548.87
```

```
arima(lm(trend.error ~ poly(x,6))$residuals,c(1,0,0))
```

```
##
## Call:
## arima(x = lm(trend.error ~ poly(x, 6))$residuals, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
##       -0.0456     0.0002
## s.e.   0.0706     0.0634
##
## sigma^2 estimated as 0.8785:  log likelihood = -270.84,  aic = 547.68
```

```
arima(lm(trend.error ~ poly(x,7))$residuals,c(1,0,0))
```

```
##
## Call:
## arima(x = lm(trend.error ~ poly(x, 7))$residuals, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
```

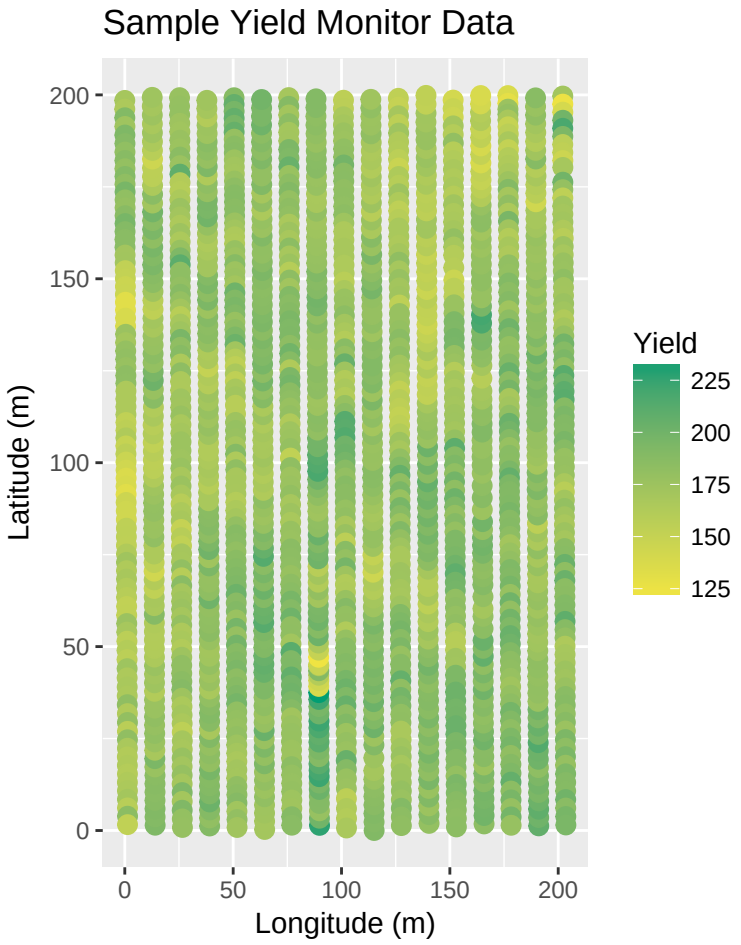
```
##       -0.0455     0.0002
## s.e.   0.0706     0.0634
##
## sigma^2 estimated as 0.8785:  log likelihood = -270.84,  aic = 547.67
```

We have minimal improvement in AR fit to the residuals after a 5th degree polynomial; this criteria allows us to recover the process the generated our data.

Trend Surface - Yield Data in Two Dimensions.

We can extend the polynomial model to two dimensions. Again, we use sample data.

```
ggplot(sample.dat, aes(LonM,LatM)) +
geom_point(aes(colour = Yield),size=3) +
scale_colour_gradient(low=cbPalette[7], high=cbPalette[4]) +
labs(colour = "Yield", x="Longitude (m)", y="Latitude (m)", title = "Sample Yield Monitor Data")
```

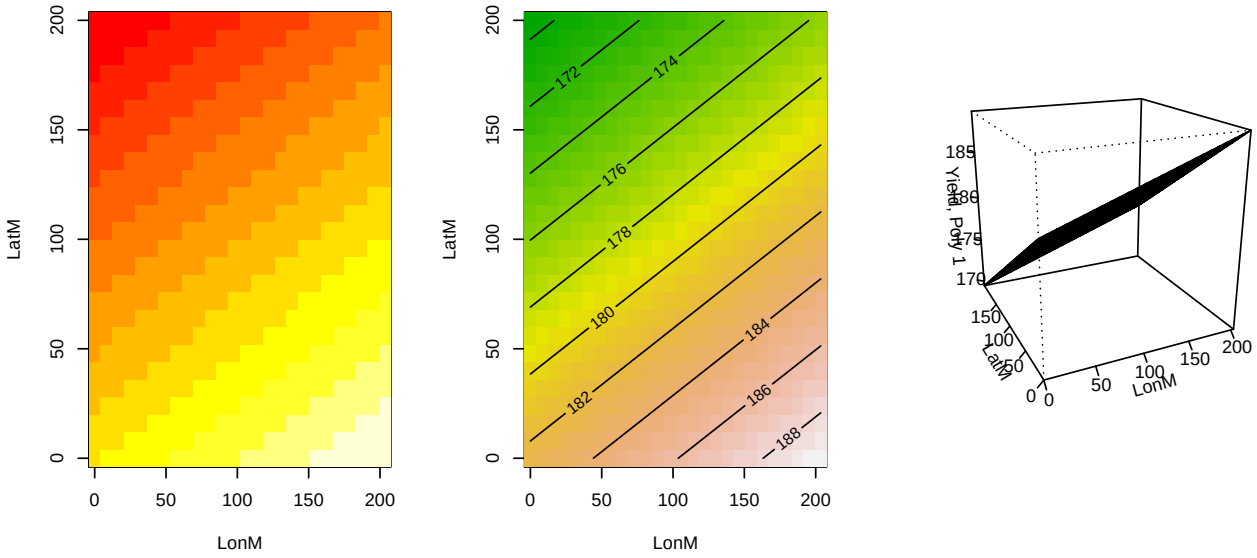


We can fit two dimensional polynomials using `lm`. We start with a simple, one-degree model and increase order.

```
Yield1.lm <- lm(Yield ~ poly(LonM, LatM, degree=1), data=sample.dat)
```

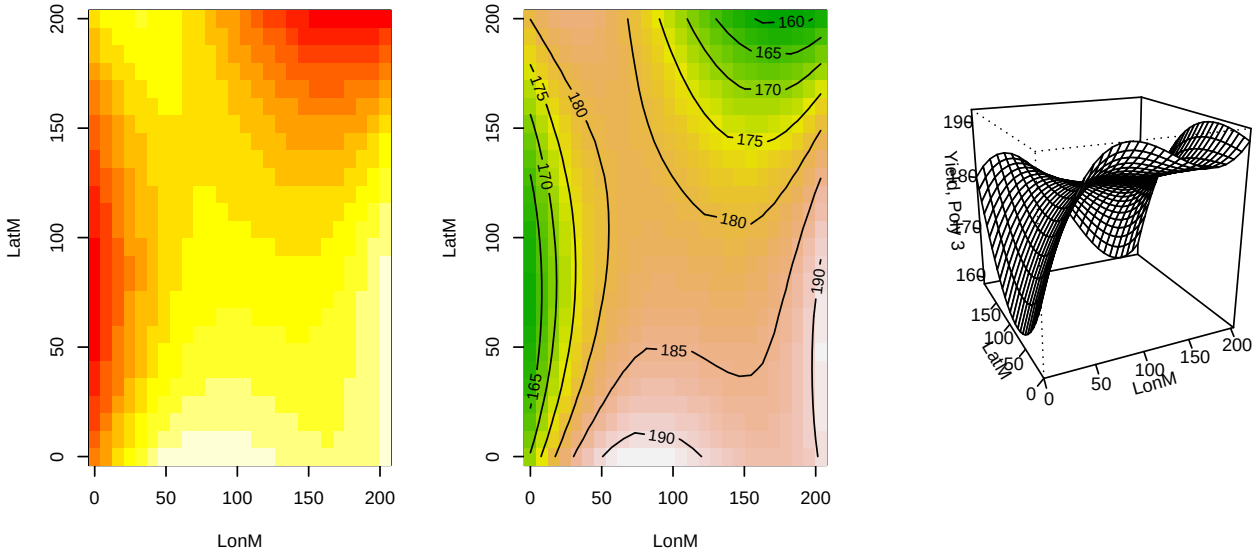
The response surface library `rsm` has some useful plots to help us visualize the trend surface

```
par(mfrow=c(1,3))
image(Yield1.lm, LatM ~ LonM)
contour(Yield1.lm,LatM ~ LonM, image = TRUE)
persp(Yield1.lm, LatM ~ LonM, zlab = "Yield, Poly 1")
```



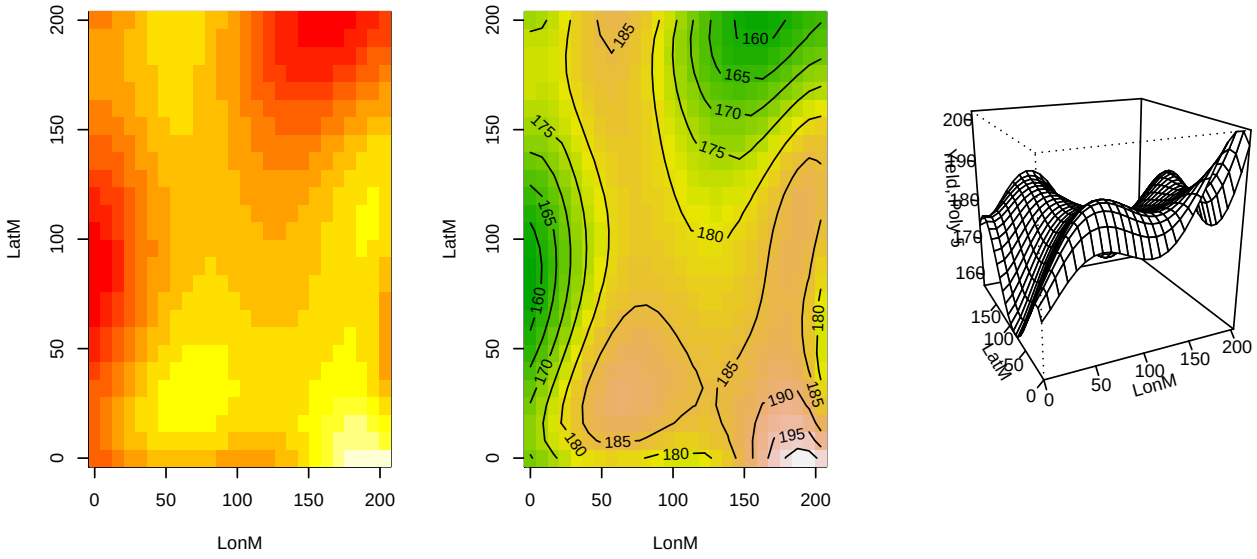

```
Yield3.lm <- lm(Yield ~ poly(LonM, LatM, degree=3), data=sample.dat)
```

```
par(mfrow=c(1,3))
image(Yield3.lm, LatM ~ LonM)
contour(Yield3.lm, LatM ~ LonM,image = TRUE)
persp(Yield3.lm, LatM ~ LonM, zlab = "Yield, Poly 3")
```



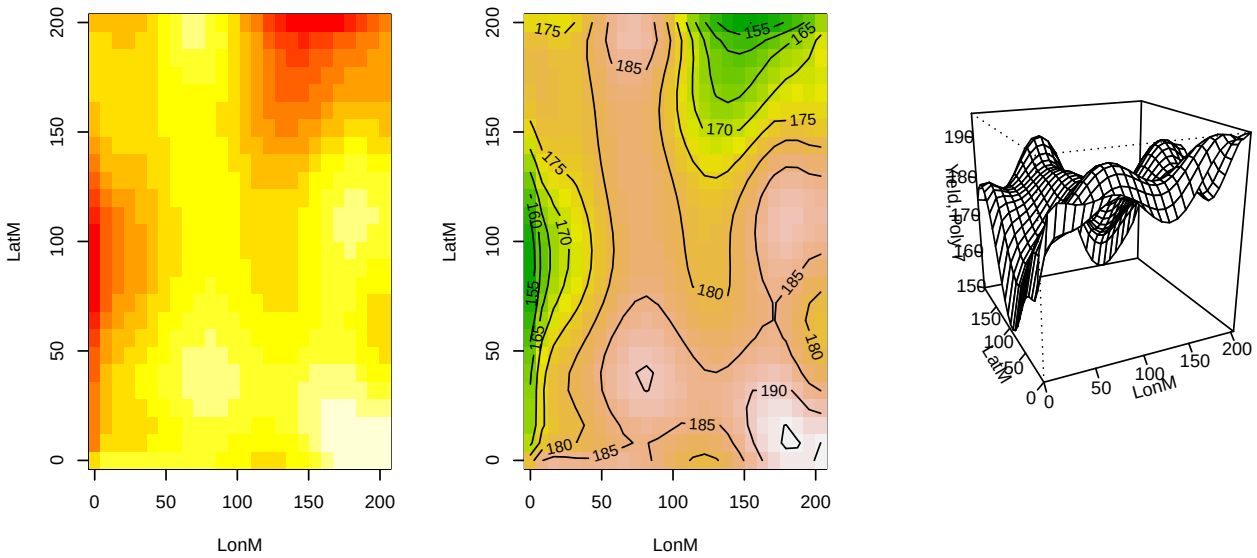

```
Yield5.lm <- lm(Yield ~ poly(LonM, LatM, degree=5), data=sample.dat)
```

```
par(mfrow=c(1,3))
image(Yield5.lm, LatM ~ LonM)
contour(Yield5.lm, LatM ~ LonM,image = TRUE)
persp(Yield5.lm, LatM ~ LonM, zlab = "Yield, Poly 5")
```



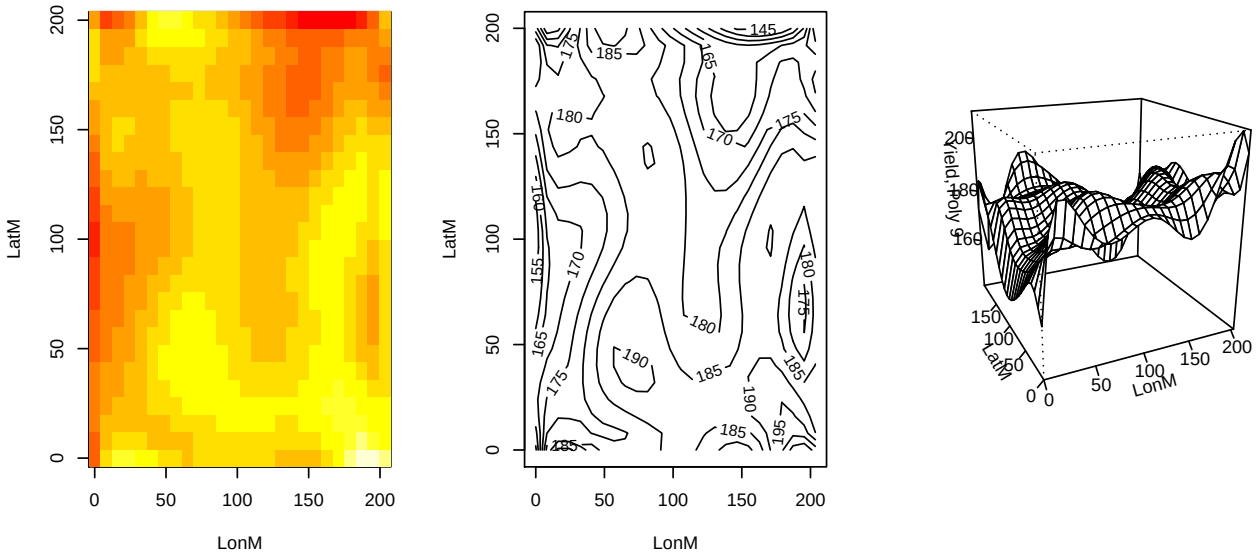

```
Yield7.lm <- lm(Yield ~ poly(LonM, LatM, degree=7), data=sample.dat)
```

```
par(mfrow=c(1,3))
image(Yield7.lm, LatM ~ LonM)
contour(Yield7.lm, LatM ~ LonM,image = TRUE)
persp(Yield7.lm, LatM ~ LonM, zlab = "Yield, Poly 7")
```



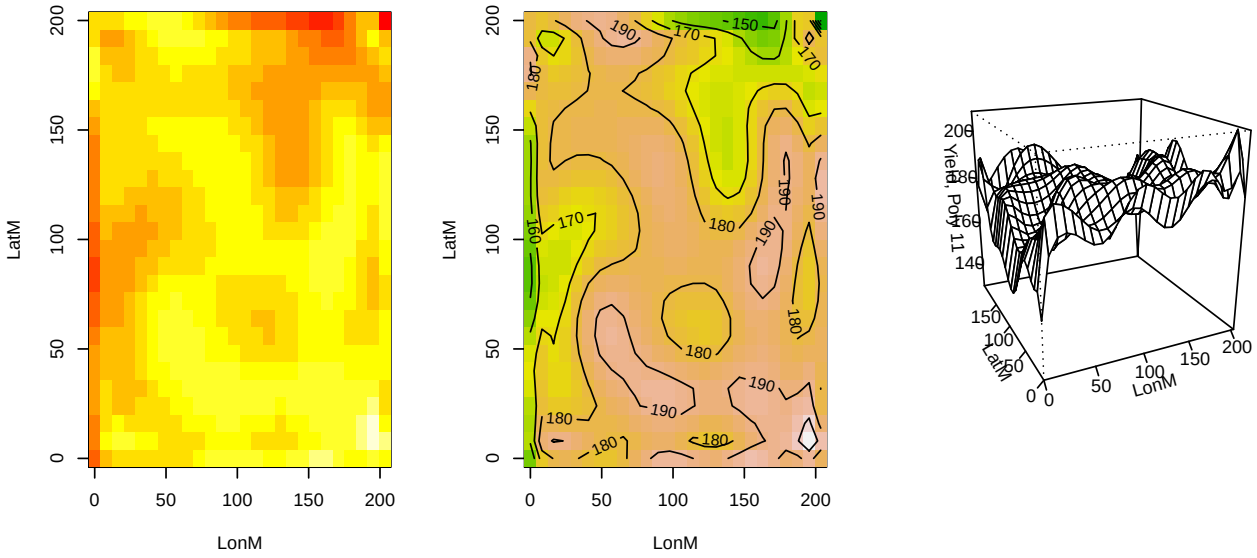

```
Yield9.lm <- lm(Yield ~ poly(LonM, LatM, degree=9), data=sample.dat)
```

```
par(mfrow=c(1,3))
image(Yield9.lm, LatM ~ LonM)
contour(Yield9.lm, LatM ~ LonM)
persp(Yield9.lm, LatM ~ LonM, zlab = "Yield, Poly 9")
```



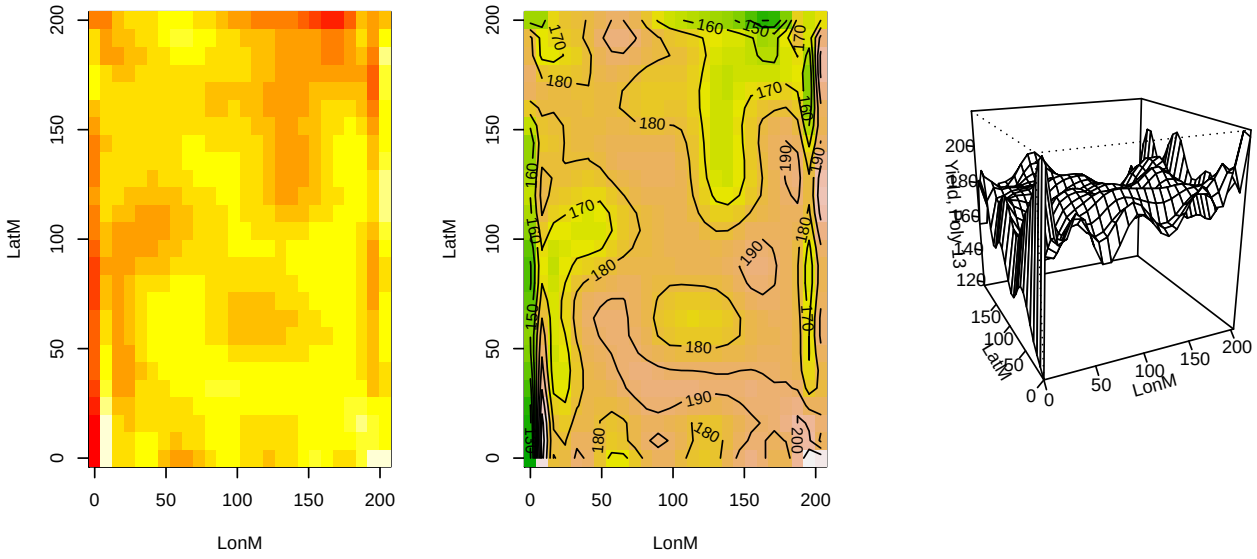

```
Yield11.lm <- lm(Yield ~ poly(LonM, LatM, degree=11), data=sample.dat)
```

```
par(mfrow=c(1,3))
image(Yield11.lm, LatM ~ LonM)
contour(Yield11.lm, LatM ~ LonM,image = TRUE)
persp(Yield11.lm, LatM ~ LonM, zlab = "Yield, Poly 11")
```



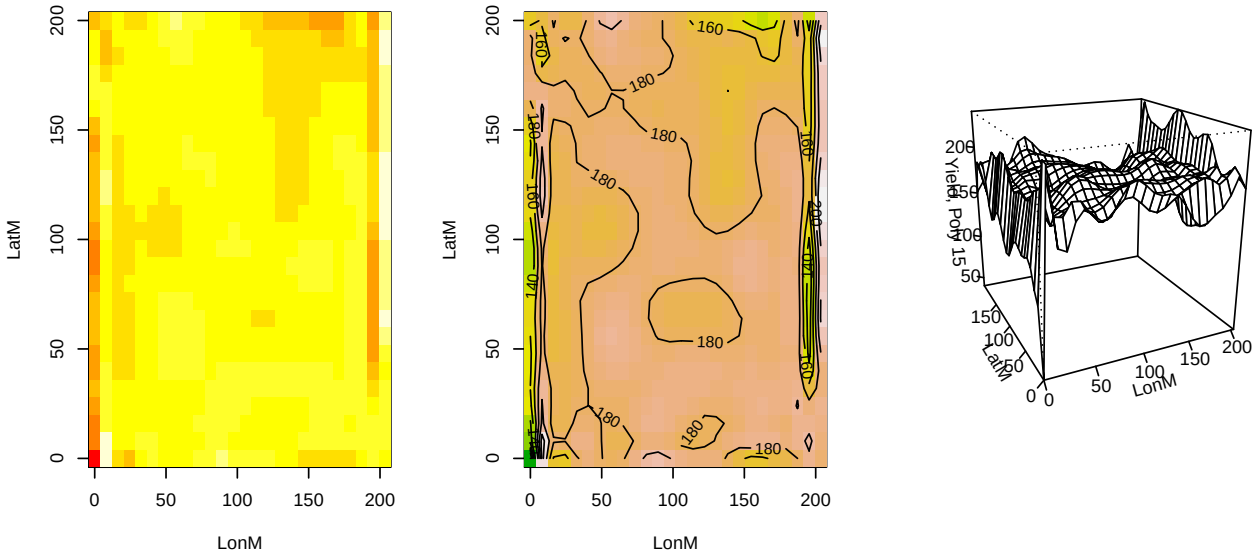

```
Yield13.lm <- lm(Yield ~ poly(LonM, LatM, degree=13), data=sample.dat)
```

```
par(mfrow=c(1,3))
image(Yield13.lm, LatM ~ LonM)
contour(Yield13.lm, LatM ~ LonM,image = TRUE)
persp(Yield13.lm, LatM ~ LonM, zlab = "Yield, Poly 13")
```




```
Yield15.lm <- lm(Yield ~ poly(LonM, LatM, degree=15), data=sample.dat)
```

```
par(mfrow=c(1,3))
image(Yield15.lm, LatM ~ LonM)
contour(Yield15.lm, LatM ~ LonM,image = TRUE)
persp(Yield15.lm, LatM ~ LonM, zlab = "Yield, Poly 15")
```



To compare models, we extract residuals.

```
sample.dat$Yield1.resid <- Yield1.lm$residuals
sample.dat$Yield3.resid <- Yield3.lm$residuals
sample.dat$Yield5.resid <- Yield5.lm$residuals
sample.dat$Yield7.resid <- Yield7.lm$residuals
sample.dat$Yield9.resid <- Yield9.lm$residuals
sample.dat$Yield11.resid <- Yield11.lm$residuals
sample.dat$Yield13.resid <- Yield13.lm$residuals
sample.dat$Yield15.resid <- Yield15.lm$residuals
```

Next, we compute global measures of spatial correlation among the residuals.

```r
sample.dists <- as.matrix(dist(cbind(sample.dat$LonM, sample.dat$LatM)))

sample.dists.inv <- 1/sample.dists
diag(sample.dists.inv) <- 0

Moran.I(sample.dat$Yield1.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.07446532
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001459662
##
## $p.value
## [1] 0
```

```r
Moran.I(sample.dat$Yield3.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.03934532
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001459585
##
## $p.value
## [1] 0
```

```r
Moran.I(sample.dat$Yield5.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.03134507
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001459363
##
## $p.value
## [1] 0
```

```r
Moran.I(sample.dat$Yield7.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.02985075
##
## $expected
## [1] -0.0006684492
##
## $sd
```

```
## [1] 0.001459295
##
## $p.value
## [1] 0
```

```
Moran.I(sample.dat$Yield9.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.02590196
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.00145921
##
## $p.value
## [1] 0
```

```
Moran.I(sample.dat$Yield11.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.02051911
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001459283
##
## $p.value
## [1] 0
```

```
Moran.I(sample.dat$Yield13.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.0181327
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.00145912
##
## $p.value
## [1] 0
```

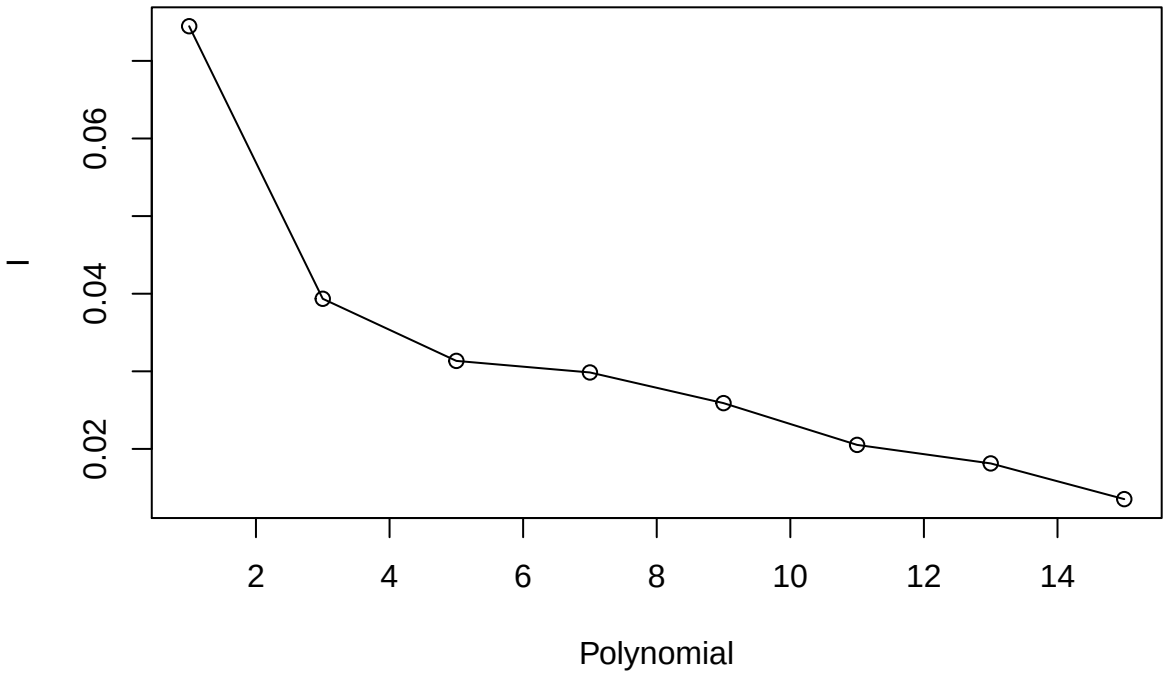
```
Moran.I(sample.dat$Yield15.resid, sample.dists.inv)
```

```
## $observed
## [1] 0.01353203
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001459062
```

```
##
## $p.value
## [1] 0
```

Can we improve the fit by adding higher order polynomials? We can plot I by x^n

```
fit.dat <- data.frame(
  Polynomial =  seq(1,15,2),
  I = c(Moran.I(sample.dat$Yield1.resid, sample.dists.inv)$observed,
       Moran.I(sample.dat$Yield3.resid, sample.dists.inv)$observed,
       Moran.I(sample.dat$Yield5.resid, sample.dists.inv)$observed,
       Moran.I(sample.dat$Yield7.resid, sample.dists.inv)$observed,
       Moran.I(sample.dat$Yield9.resid, sample.dists.inv)$observed,
       Moran.I(sample.dat$Yield11.resid, sample.dists.inv)$observed,
       Moran.I(sample.dat$Yield13.resid, sample.dists.inv)$observed,
       Moran.I(sample.dat$Yield15.resid, sample.dists.inv)$observed)
   )
plot(I ~Polynomial, data=fit.dat)
lines(I ~Polynomial, data=fit.dat)
```

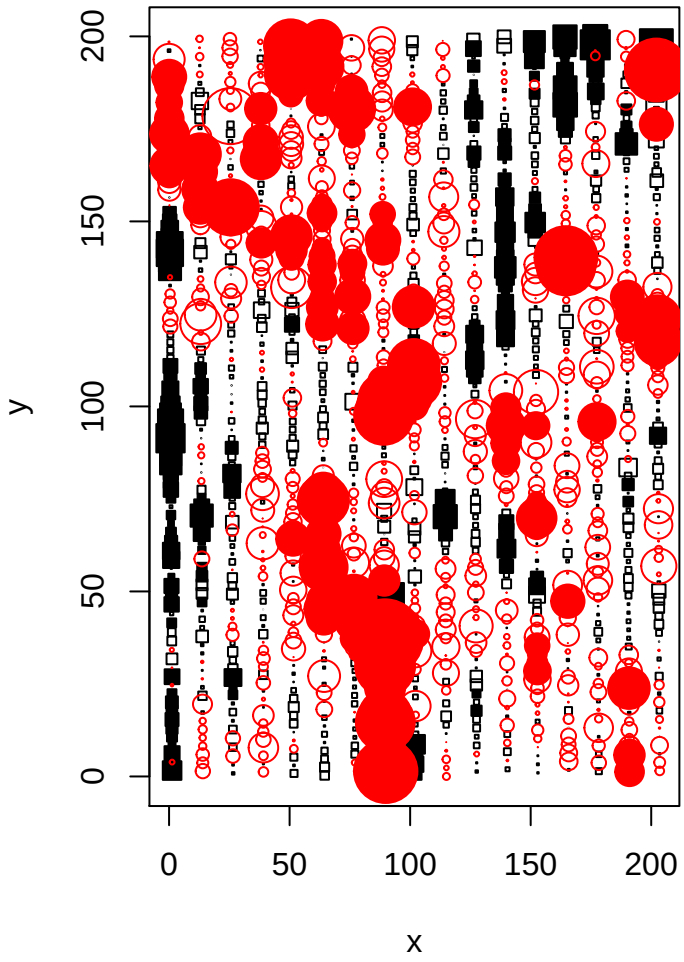


Fit diagnostics

To check the quality of the fit, we can examine LISA plots.

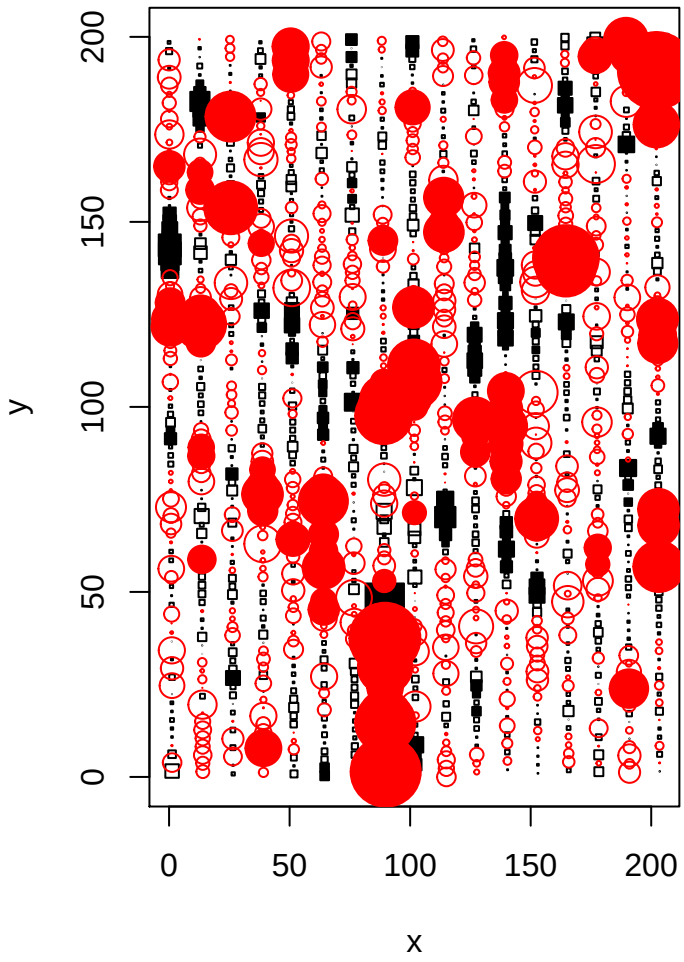
```
Yield1.resid.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$Yield1.resid,
                          neigh=10, resamp=500, quiet=TRUE)
```

```
plot.lisa(Yield1.resid.lisa, negh.mean=FALSE)
```



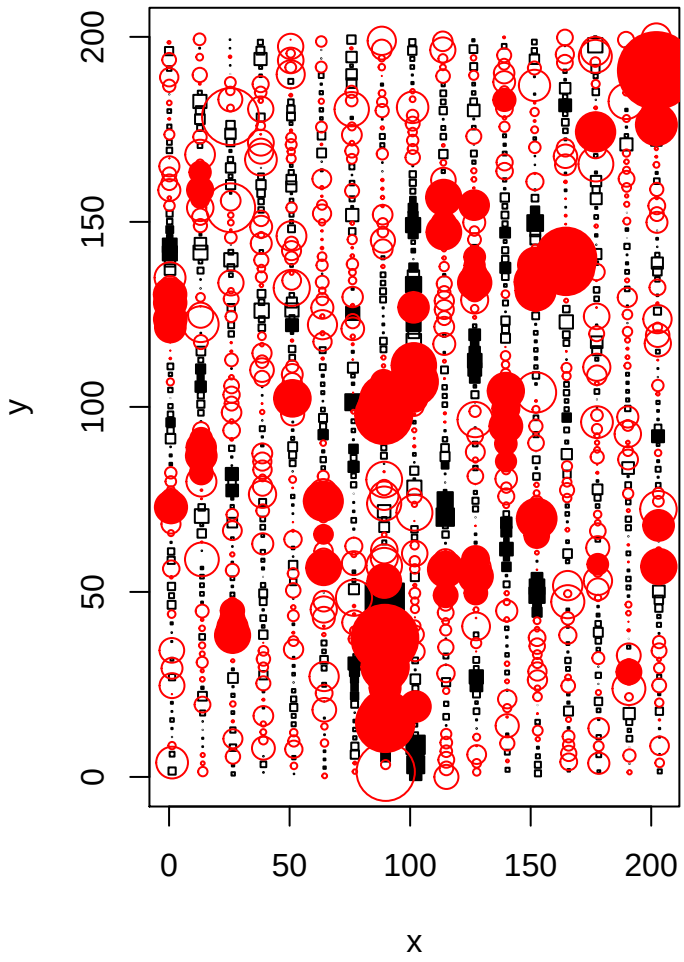

```
Yield7.resid.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$Yield7.resid,
                          neigh=10, resamp=500, quiet=TRUE)
```

```
plot.lisa(Yield7.resid.lisa, negh.mean=FALSE)
```




```
Yield13.resid.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$Yield13.resid,
                           neigh=10, resamp=500, quiet=TRUE)
```

```
plot.lisa(Yield13.resid.lisa, negh.mean=FALSE)
```



With the lisa plot, 'hot spots' - points with positive local correlation, are marked by filled red circles; local cold spots (negative correlation) with filled black squares. If we've completely detrended the data, we expect to see very little indication of local spatial correlation. The presence of hot or cold spots in the residuals would suggest there is more spatial information to be recovered.