GeostatisticsPart2

Kostas Goulias

3/4/2018

```
library(rgdal)     # Used to read data

## Warning: package 'rgdal' was built under R version 3.4.2

## Loading required package: sp

## rgdal: version: 1.2-13, (SVN revision 686)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.1.3, released 2017/20/01
##  Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.4/R
esources/library/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.4
/Resources/library/rgdal/proj
##  Linking to sp version: 1.2-5

library(tmap)      # Used in creating maps

## Warning: package 'tmap' was built under R version 3.4.3

library(maptools)  # Used to create a grid of points

## Checking rgeos availability: TRUE

library(gstat) # Use gstat's idw routine
library(sp)    # Used for the spsample function
library(raster)  #  to use raters and grids
```

Geostatistics Part 2

In this session we first do some preparatory tasks for the data to have in a form that can be used for models.

Then, we interpolate using an Inverse Distance Weighting (IDW) method.

Preparatory Tasks

Data of locations in california

```
my2k = readRDS('~/Documents/COURSES UCSB/Course Winter 2018/California/h2kb.r
ds')
data <- my2k
```

Create matrix of coordinates in the data with points and attributes

```
sp_point <- matrix(NA, nrow=nrow(data),ncol=2)
sp_point[,1] <- jitter(data$XCORD,.001)
sp_point[,2] <- jitter(data$YCORD, .001)
colnames(sp_point) <- c("XCORD","YCORD")
```

Create spatial object with the coordinates we extracted from original data

Note I give the projection here when creating the data.sp

```
data.sp <- SpatialPointsDataFrame(coords=sp_point,data,proj4string=CRS("+proj
=longlat +datum=WGS84"))   ## Projection: UTM zone 10
proj4string(data.sp)

## [1] "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"

class(data.sp)

## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

Preparatory Task for Ordinary Kriging and idw

I need a spatial database with NEWDATA points that I will use for my predictions

Create empty grid

Note: Make sure we use the same projection as the observed spatial points and

then read a shp file of the State of California

I downloaded the States polygons from the US Census

```
projection.x <- CRS("+proj=longlat +datum=WGS84")
USstate <- readShapePoly ("~/Downloads/cb_2016_us_state_5m/cb_2016_us_state_5
m.shp", verbose=TRUE, proj4string=projection.x)

## Warning: use rgdal::readOGR or sf::st_read
```

```
## Shapefile type: Polygon, (5), # of Shapes: 56

class(USstate@data)

## [1] "data.frame"

CA.poly <- USstate[USstate@data$NAME =='California',]
plot(CA.poly)
```

```
proj4string(CA.poly)

## [1] "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"

class(CA.poly)

## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

Prefer to use "generic" names to repeat this

Save my observed data in P

```
P <- data.sp
plot(P)
```

```
P@bbox
```

```
##               min        max
## XCORD -124.21882 -114.36030
## YCORD   32.55885   41.95175
```

Load California boundary map

```
W <- CA.poly
plot(W)
```

```
W@bbox
```

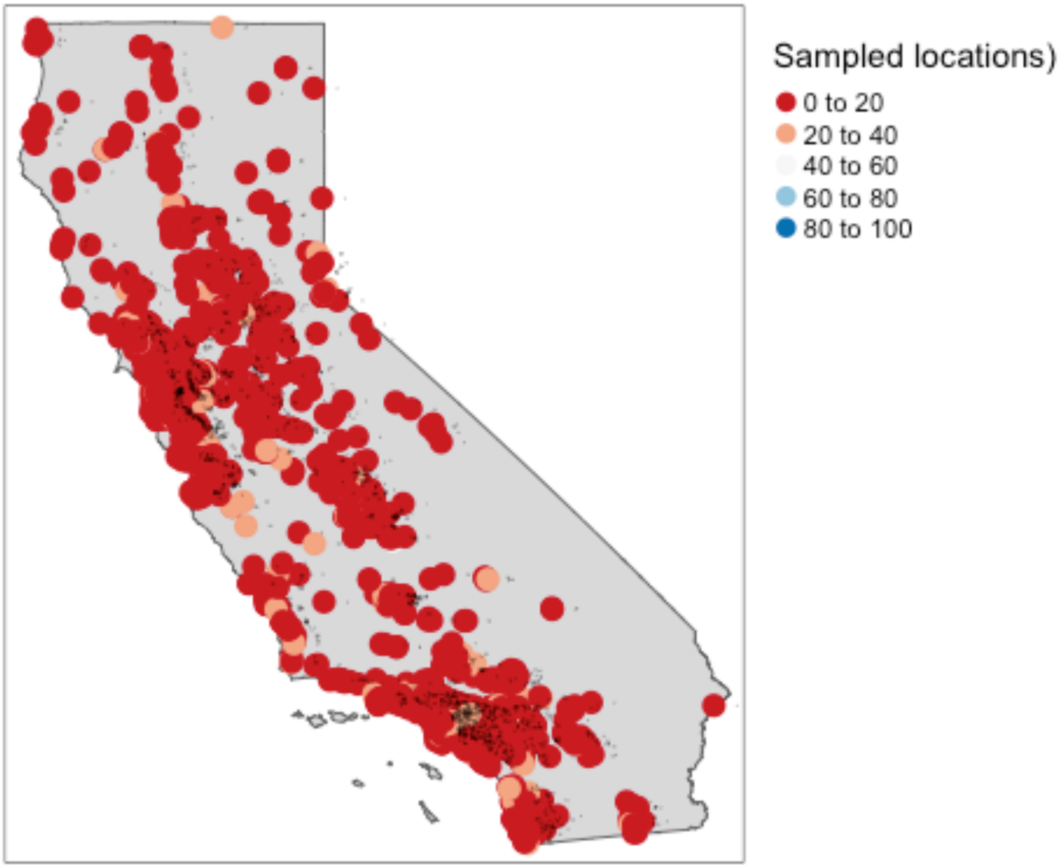
```
##           min        max
## x -124.40959 -114.13443
## y   32.53416   42.00952
```

Replace point boundary extent with that of The California Polygon

```
P@bbox <- W@bbox
```

Use tmap package to display boundary, points, and the variable Z2

```
tm_shape(W) + tm_polygons() +
  tm_shape(P) +
  tm_dots(col="Z2", palette = "RdBu", auto.palette.mapping = FALSE,
          title="Sampled locations)", size=0.5) +
  tm_text("Z2", just="left", xmod=.5, size = 0.1) +
  tm_legend(legend.outside=TRUE)
```


Sampled locations)
0 to 20
20 to 40
40 to 60
60 to 80
80 to 100

Inverse Distance Weighting (idw)

From the Bivand, Pebesma, Gomez-Rubio 2013 book and gstat manual

Inverse distance-based weighted interpolation (IDW) computes a weighted average,

$$\hat{Z}(s_0) = \frac{\sum_{i=1}^{n} w(s_i) Z(s_i)}{\sum_{i=1}^{n} w(s_i)},$$

where weights for observations are computed according to their distance to the interpolation location,

$$w(s_i) = ||s_i - s_0||^{-p},$$

As P goes to infinity w(s) -> 0 nearest neighbors (with smaller distances) play the role of contiguity

As P -> 0 w(s) -> 1 all observations get equal weight

When P = 1 the weights are just inverse functions of distance

When P < 1 we look at the impact below

Create an empty grid where n is the total number of cells

This was the trick to make sure the grid has the same names of coordinates as the points database

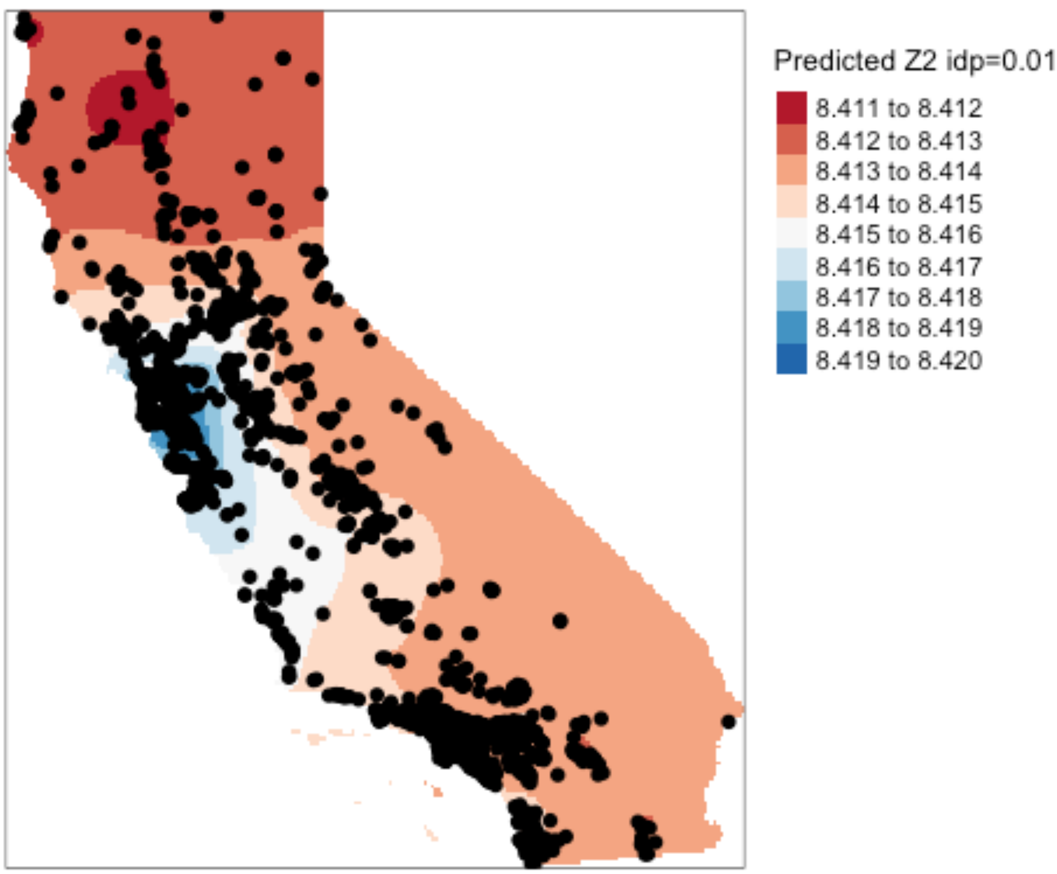
```
grd              <- as.data.frame(spsample(P, "regular", n=50000))
names(grd)       <- c("XCORD", "YCORD")
coordinates(grd) <- c("XCORD", "YCORD")
gridded(grd)     <- TRUE  # Create SpatialPixel object
fullgrid(grd)    <- TRUE  # Create SpatialGrid object
plot(grd)
```

Sorry this looks like a black box from Rmarkdown. In rstudio shows it is a grid.

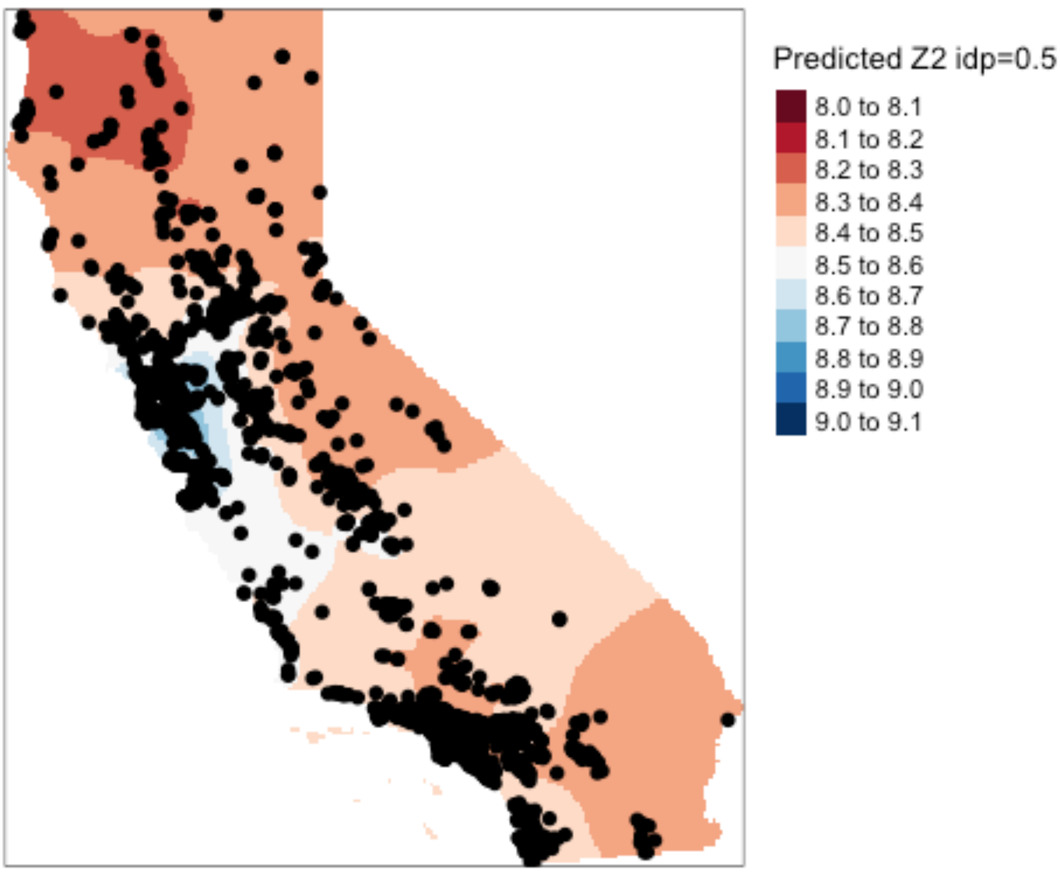
Add P's projection information to the empty grid

```
proj4string(grd) <- proj4string(P)

# My note: This gave me a hard time in the code before March 3.  this is the
reason I check it again.
proj4string(grd)

## [1] "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"

proj4string(P)

## [1] "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"
```

```
#### Interpolate the grid cells using a power value of 0.01 (idp=0.01)
P001.idw <- gstat::idw(Z2 ~ 1, P, newdata=grd, idp=0.01)

## [inverse distance weighted interpolation]

#### Convert to raster object then clip to California
####  This is one way to use only the area of California
####  The function mask does this

r       <- raster(P001.idw)
r.m     <- mask(r, W)

#### Plot Prediction
tm_shape(r.m) +
  tm_raster(n=10,palette = "RdBu", auto.palette.mapping = FALSE,
            title="Predicted Z2 idp=0.01") +
  tm_shape(P) + tm_dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



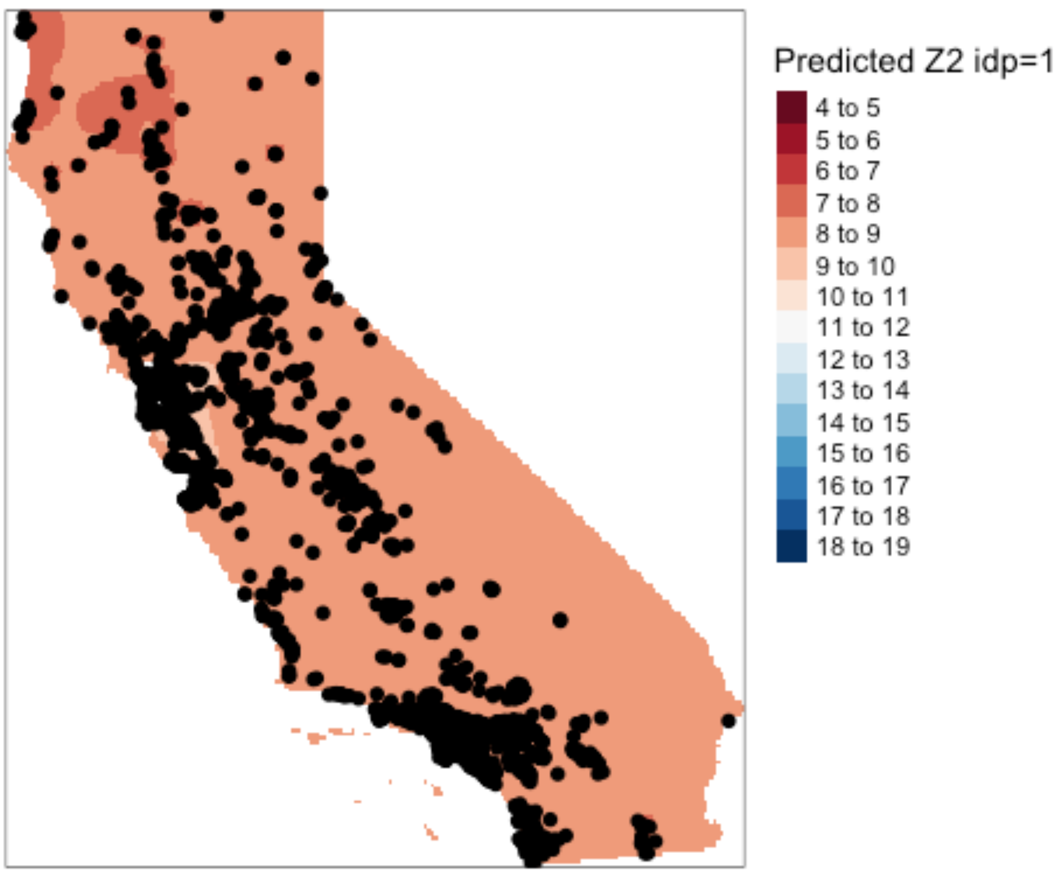
```r
#### Interpolate the grid cells using a power value of 0.5 (idp=0.5)

P.idw <- gstat::idw(Z2 ~ 1, P, newdata=grd, idp=0.5)

## [inverse distance weighted interpolation]

#### Convert to raster object then clip to California
r       <- raster(P.idw)
r.m     <- mask(r, W)

# Plot
tm_shape(r.m) +
  tm_raster(n=10,palette = "RdBu", auto.palette.mapping = FALSE,
            title="Predicted Z2 idp=0.5") +
  tm_shape(P) + tm_dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



```r
# Interpolate the grid cells using a power value of 1 (idp=1)
P1.idw <- gstat::idw(Z2 ~ 1, P, newdata=grd, idp=1)
```

```
## [inverse distance weighted interpolation]
```

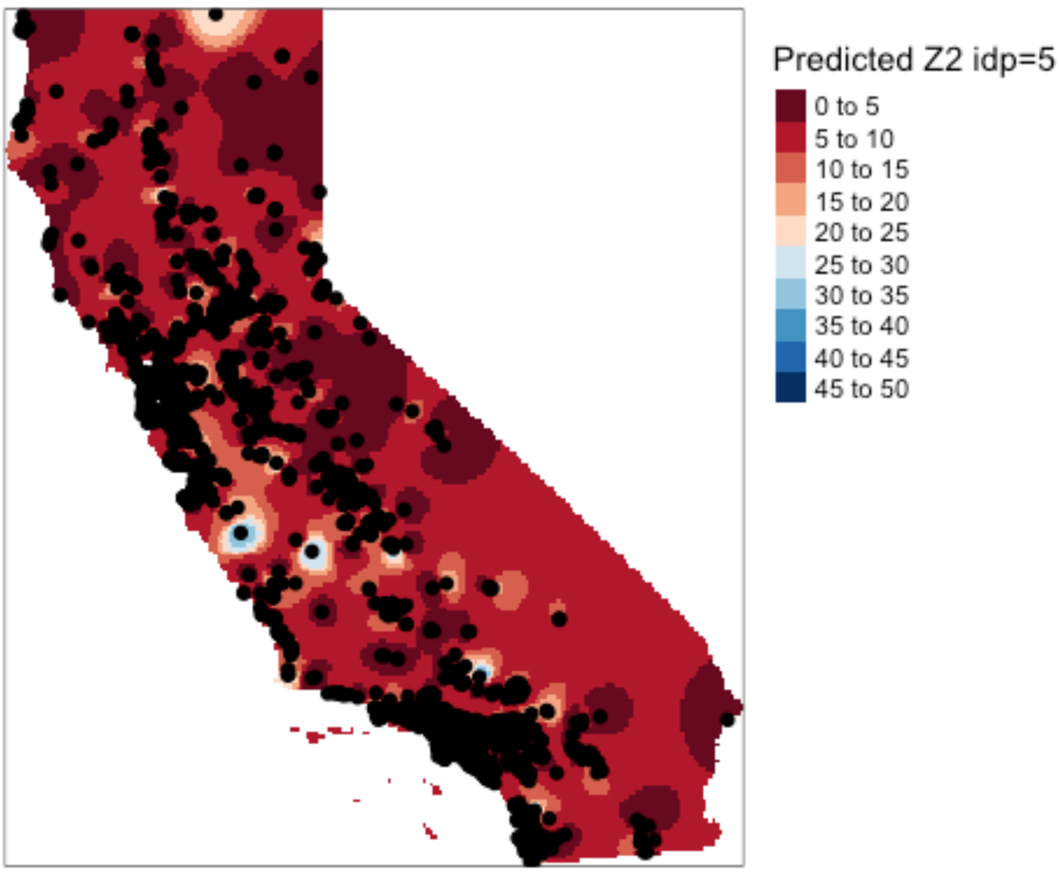
```r
# Convert to raster object then clip to California
r       <- raster(P1.idw)
r.m     <- mask(r, W)
# Plot
tm_shape(r.m) +
  tm_raster(n=10,palette = "RdBu", auto.palette.mapping = FALSE,
            title="Predicted Z2 idp=1") +
  tm_shape(P) + tm_dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



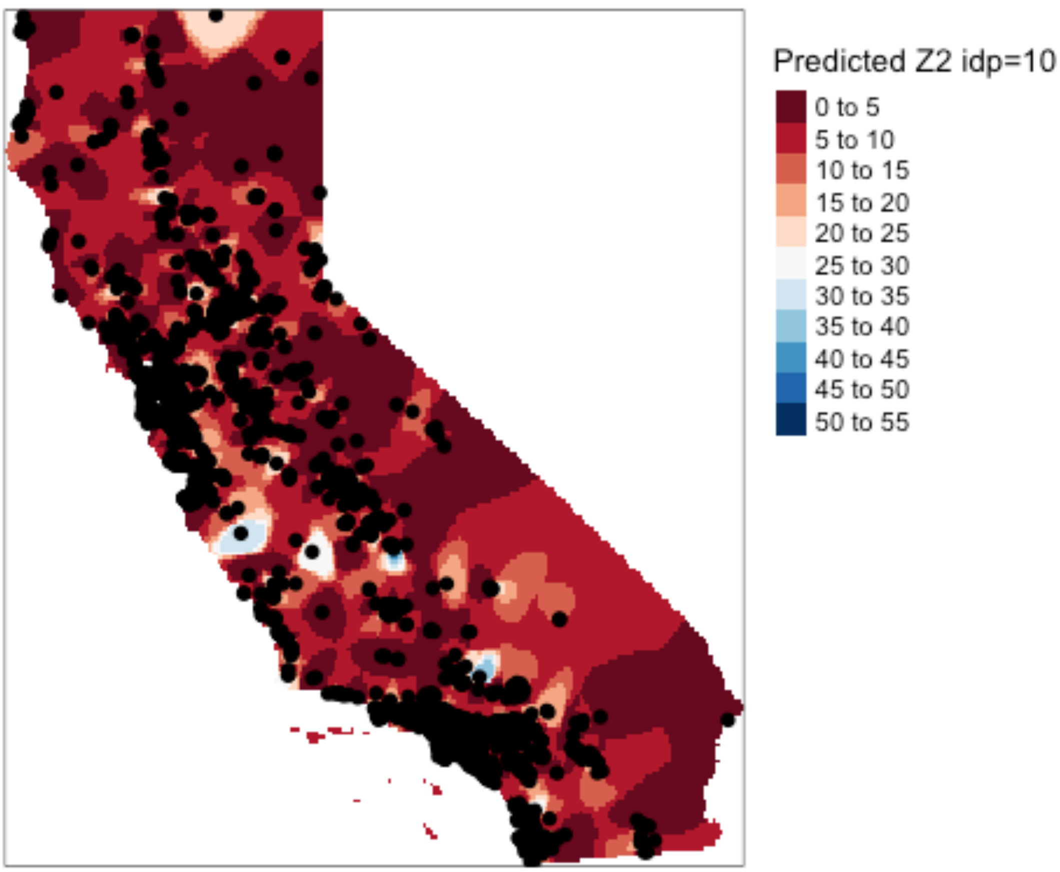
```r
# Interpolate the grid cells using a power value of 5 (idp=5)
P5.idw <- gstat::idw(Z2 ~ 1, P, newdata=grd, idp=5)
```

```
## [inverse distance weighted interpolation]
```

```r
# Convert to raster object then clip to California
r       <- raster(P5.idw)
r.m     <- mask(r, W)
# Plot
tm_shape(r.m) +
  tm_raster(n=10,palette = "RdBu", auto.palette.mapping = FALSE,
            title="Predicted Z2 idp=5") +
  tm_shape(P) + tm_dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



```r
# Interpolate the grid cells using a power value of 10 (idp=10)
P10.idw <- gstat::idw(Z2 ~ 1, P, newdata=grd, idp=10)

## [inverse distance weighted interpolation]

# Convert to raster object then clip to California
r       <- raster(P10.idw)
r.m     <- mask(r, W)

# Plot
tm_shape(r.m) +
  tm_raster(n=10,palette = "RdBu", auto.palette.mapping = FALSE,
            title="Predicted Z2 idp=10") +
  tm_shape(P) + tm_dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



Before moving to kriging let's compare the maps.

Ordinary Kriging

gstat needs to have the variogram instead of the covariance function we used in the Isaacs and Srivastava example

Unlike our small example of 7 points, instead of imposing to the data our equation we fit a nonlinear equation to the data we have

From Bivand, Pebesma, and Gomez-Rubio (2013) and gstat manual

In standard statistical problems, correlation can be estimated from a scatterplot, when several data pairs $\{x, y\}$ are available. The spatial correlation between two observations of a variable $z(s)$ at locations s_1 and s_2 cannot be estimated, as only a single pair is available. To estimate spatial correlation from observational data, we therefore need to make stationarity assumptions before we can make any progress. One commonly used form of stationarity is *intrinsic stationarity*, which assumes that the process that generated the samples is a random function $Z(s)$ composed of a mean and residual

$$Z(s) = m + e(s), \tag{8.1}$$

with a constant mean

$$\mathrm{E}(Z(s)) = m \tag{8.2}$$

and a variogram defined as

$$\gamma(h) = \frac{1}{2}\mathrm{E}(Z(s) - Z(s+h))^2. \tag{8.3}$$

Under this assumption, we basically state that the variance of Z is constant, and that spatial correlation of Z does not depend on location s, but only on separation distance h. Then, we can form *multiple* pairs $\{z(s_i), z(s_j)\}$ that have (nearly) identical separation vectors $h = s_i - s_j$ and estimate correlation from them. If we further assume *isotropy*, which is direction independence of semivariance, we can replace the vector h with its length, $||h||$.

Under this assumption, the variogram can be estimated from N_h sample data pairs $z(s_i)$, $z(s_i + h)$ for a number of distances (or distance intervals) $\tilde{h}_j$ by

$$\hat{\gamma}(\tilde{h}_j) = \frac{1}{2N_h}\sum_{i=1}^{N_h}(Z(s_i) - Z(s_i + h))^2, \quad \forall h \in \tilde{h}_j \tag{8.4}$$

and this estimate is called the *sample* variogram.

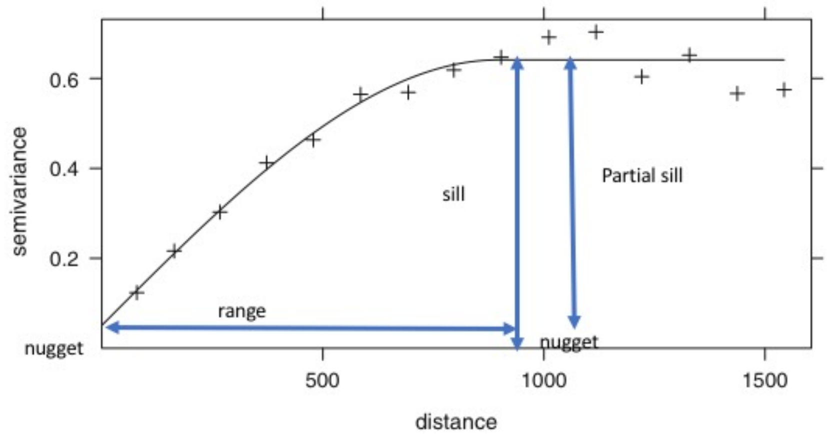
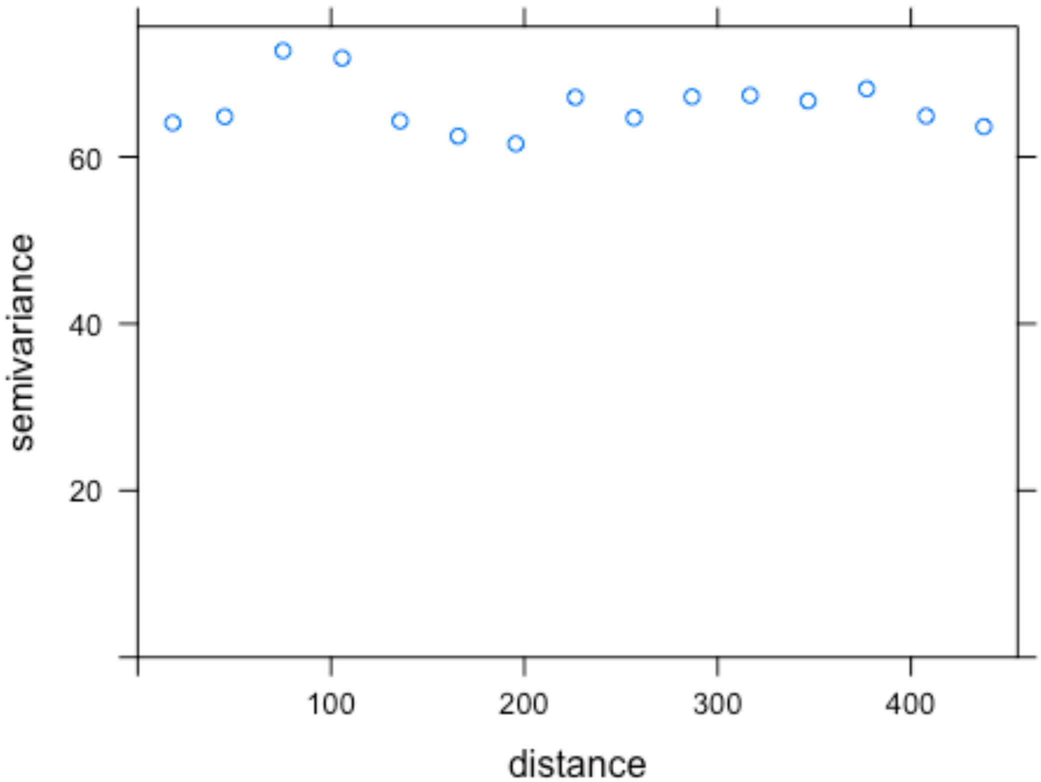


Fig. 8.6. Sample variogram (*plus*) and fitted model (*dashed line*)

This part computes the gammas from the data we have

```
v <-variogram(Z2~1, P)
plot(v)
```



####

This part fits an exponential model to the computed variogram values from the data we have

```
dat.fit <- fit.variogram(v, vgm(psill=40, model="Exp", range=5))
```

This part computes the weights and creates Kriging predictions for the points in grd (the grid points we created earlier)

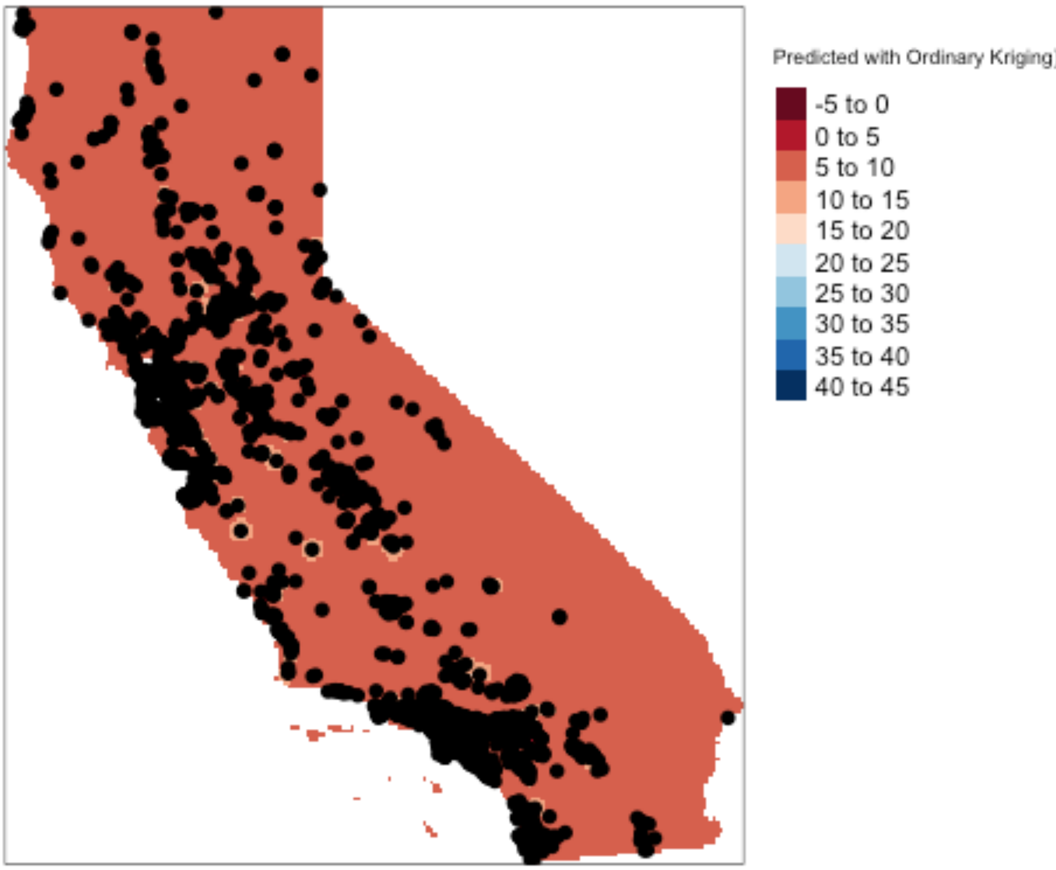
```
f.1 <- as.formula(Z2 ~ 1)


# Perform the krige interpolation (note the use of the variogram fitted model
created in the earlier step)
dat.krg <- krige( f.1, P, grd, dat.fit)

## [using ordinary kriging]

OK <- dat.krg

summary(OK)

## Object of class SpatialGridDataFrame
## Coordinates:
##              min        max
## XCORD -124.41536 -114.13370
## YCORD   32.52501   42.01237
## Is projected: FALSE
## proj4string :
## [+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0]
## Grid attributes:
##       cellcentre.offset   cellsize cells.dim
## XCORD         -124.39330 0.04412729       233
## YCORD           32.54707 0.04412729       215
## Data attributes:
##    var1.pred          var1.var
##  Min.   :-0.6578   Min.   : 0.5518
##  1st Qu.: 7.8605   1st Qu.:66.9975
##  Median : 7.8651   Median :67.0535
##  Mean   : 7.8629   Mean   :64.8462
##  3rd Qu.: 7.8651   3rd Qu.:67.0535
##  Max.   :41.4606   Max.   :67.0535
```

Convert kriged surface to a raster object for clipping to the California boundary

```
r <- raster(dat.krg)
r.m <- mask(r, W)
```

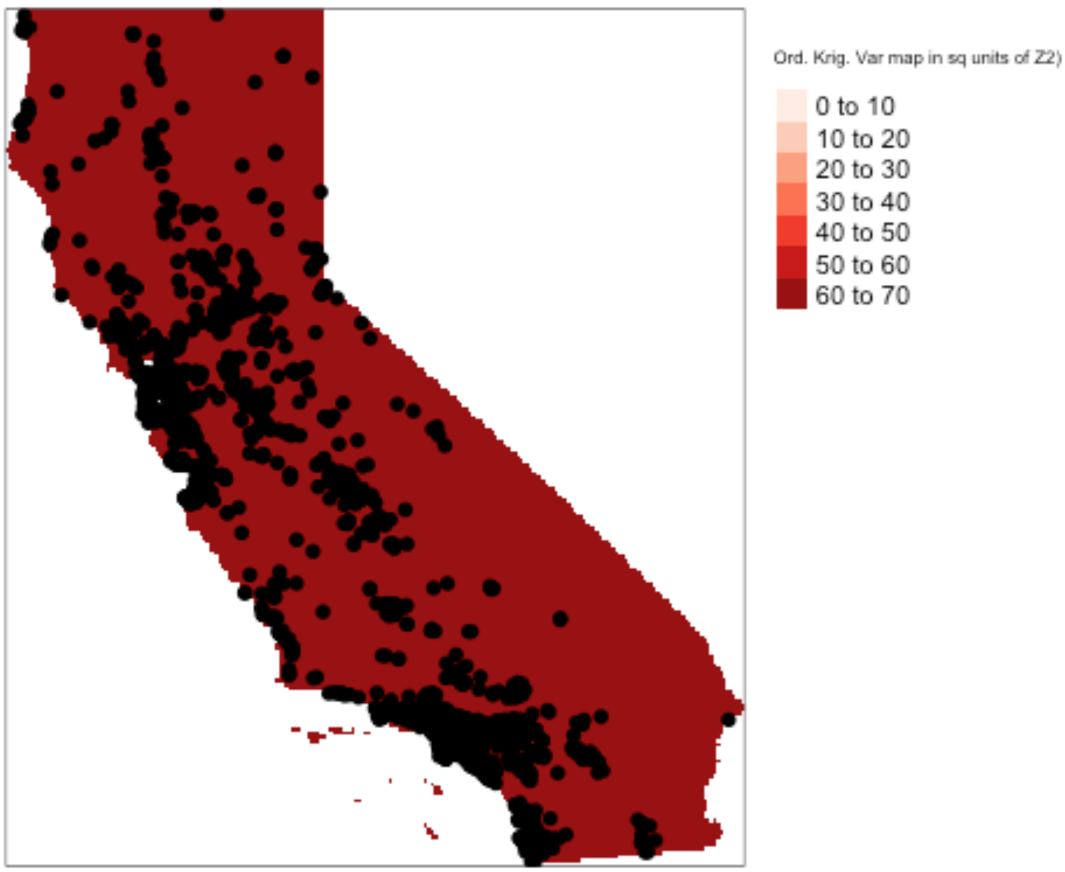
Plot the Ordinary Kriging map

```
tm_shape(r.m) +
  tm_raster(n=10, palette="RdBu", auto.palette.mapping=FALSE,
            title="Predicted with Ordinary Kriging)") +
  tm_shape(P) + tm_dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```

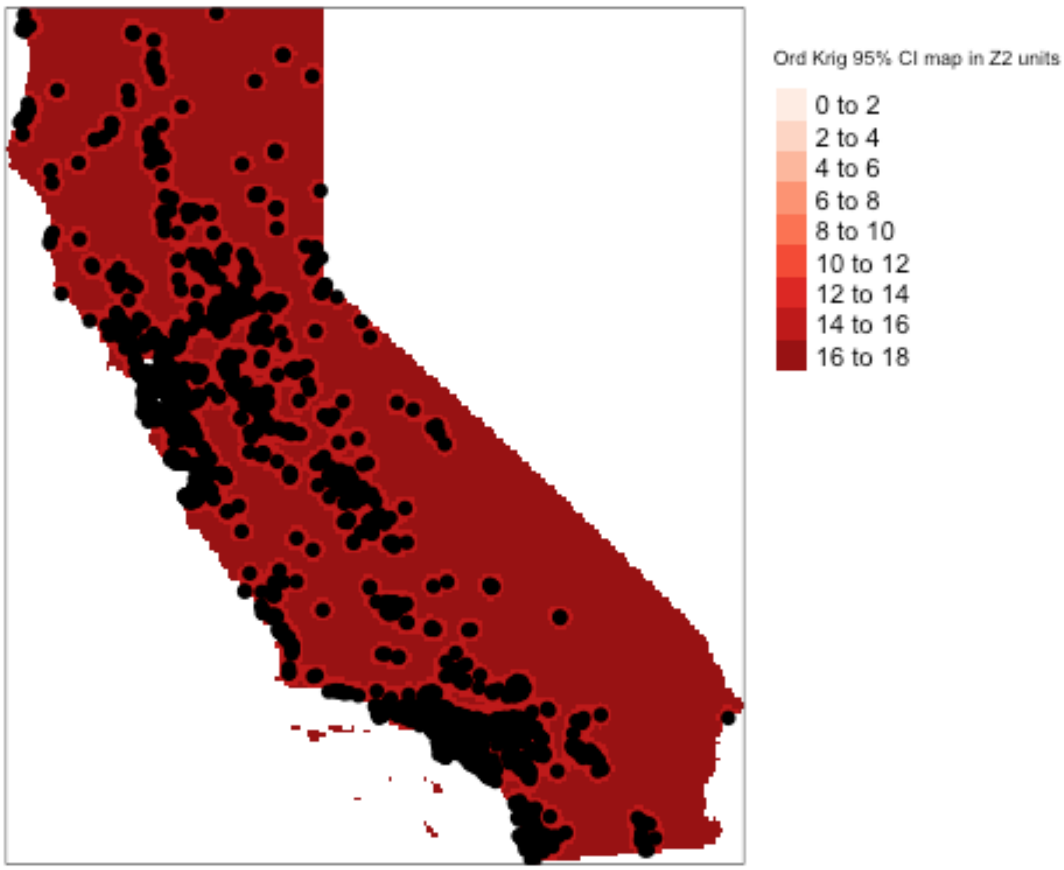


The second layer of data in a kriging object contains the variance of predictions

```
r   <- raster(dat.krg, layer="var1.var")
r.m <- mask(r, W)

tm_shape(r.m) +
  tm_raster(n=7, palette ="Reds",
            title="Ord. Krig. Var map in sq units of Z2)") +tm_shape(P) + tm_
dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



```r
#  In confidence interval around z2
r   <- sqrt(raster(dat.krg, layer="var1.var")) * 1.96
r.m <- mask(r, W)

tm_shape(r.m) + 
  tm_raster(n=7, palette ="Reds",
            title="Ord Krig 95% CI map in Z2 units") +tm_shape(P) + tm_dots(s
ize=0.2) +
  tm_legend(legend.outside=TRUE)
```



Universal Kriging

Universal kriging when we allow the prediction to be influenced by the location (x,y)

Define the function

```
f.1 <- as.formula(Z2 ~ XCORD + YCORD)
```

Compute the sample variogram; note that the f.1 trend model is one of the

parameters passed to variogram(). In this way the variogram computational algorithm knows we are doing universal kriging

```
var.smpl <- variogram(f.1, P)
```

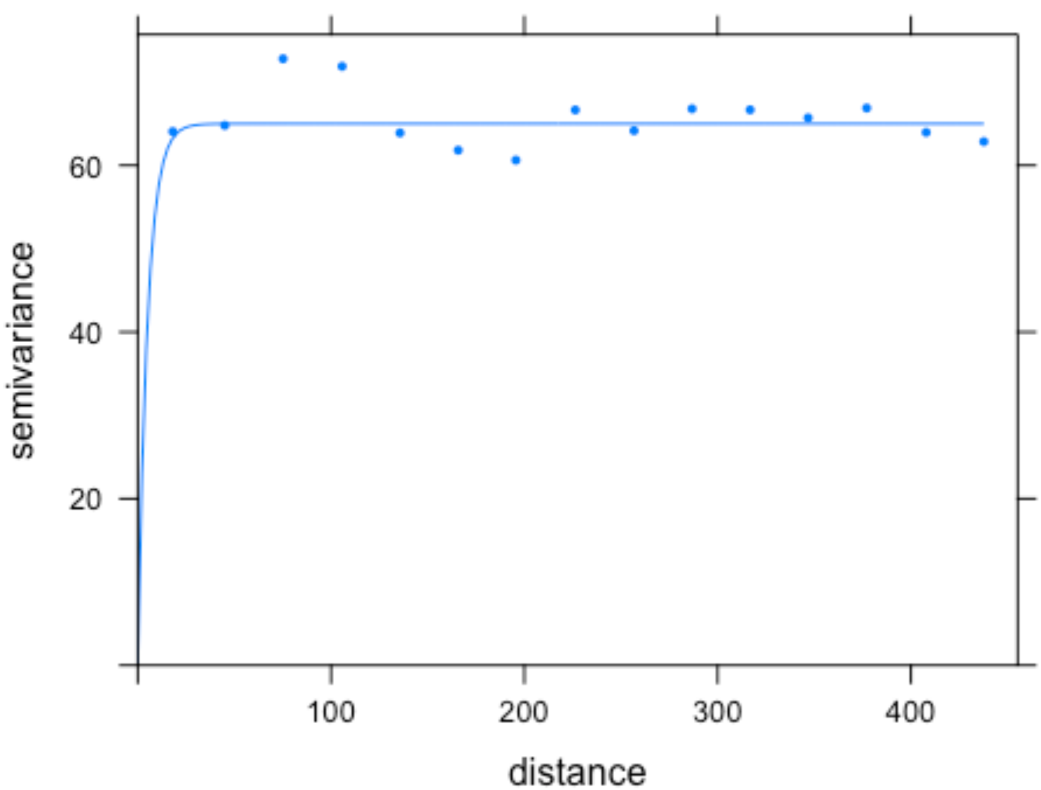
Compute the variogram model by passing the nugget, sill and range values

to fit.variogram() via the vgm() function.

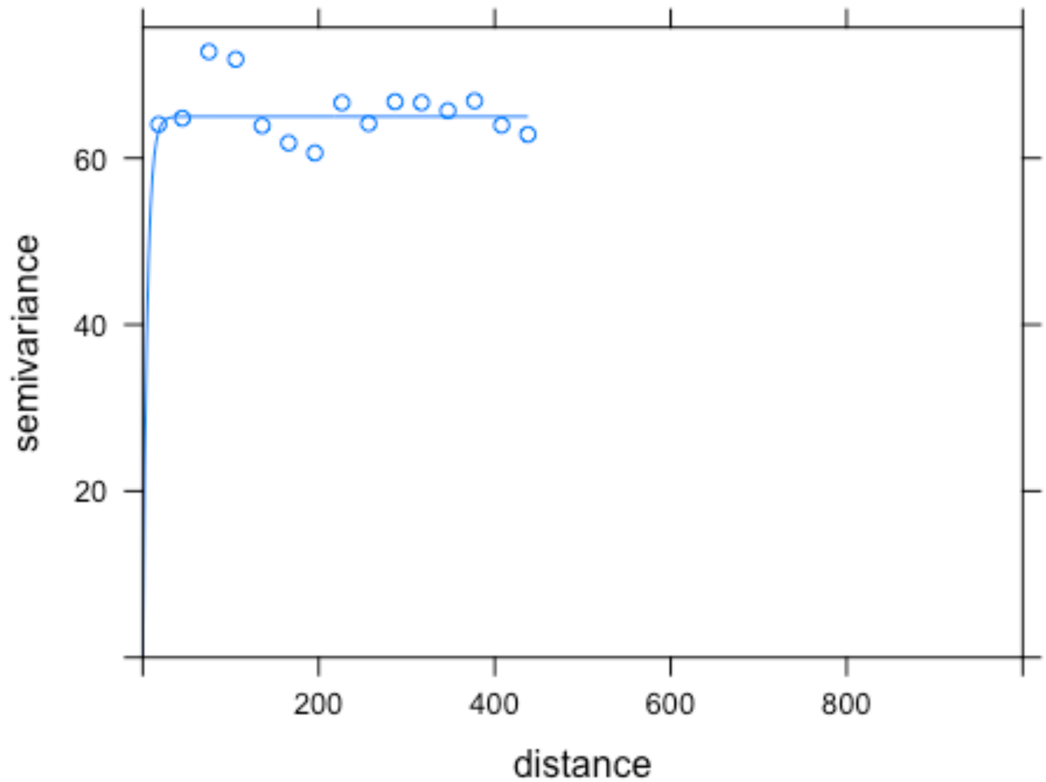
```
dat.fit  <- fit.variogram(var.smpl, fit.ranges = FALSE, fit.sills = FALSE,
                          vgm(psill=65, model="Exp", range=5, nugget=0))
```

The following plot allows us to assess the fit

```
plot(var.smpl, dat.fit, pch = 16,cex=.5)
```



```r
plot(var.smpl, dat.fit, xlim=c(0,1000))
```

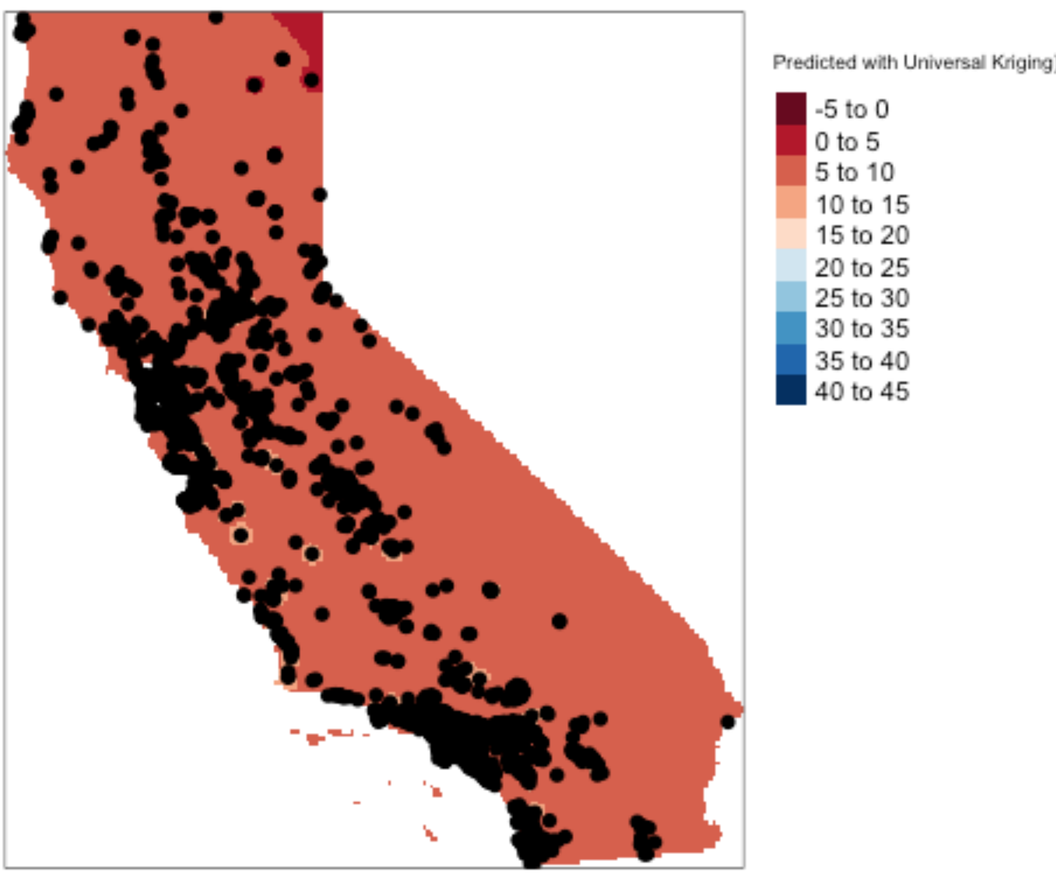



```r
dat.fit

##   model psill range
## 1   Nug     0     0
## 2   Exp    65     5

class(dat.fit)

## [1] "variogramModel" "data.frame"

# Define the trend model
```

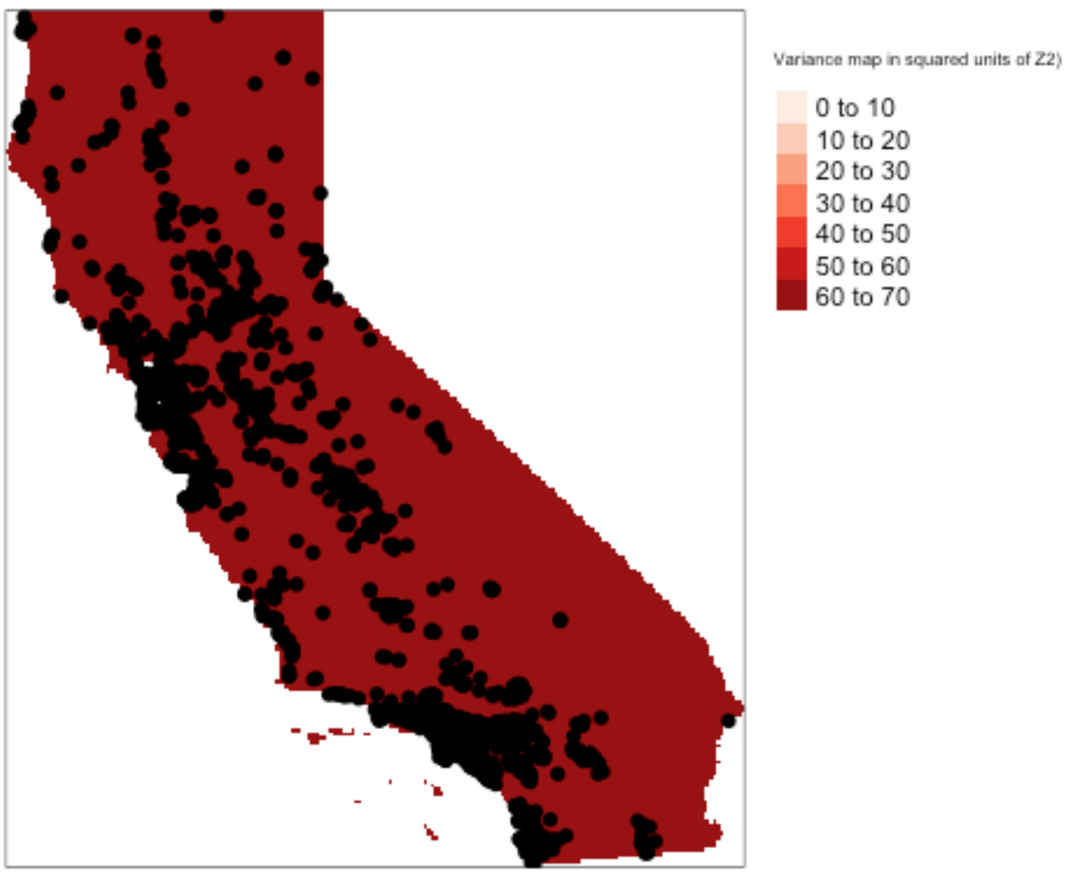
f.1 is repetitive but that's all right

```r
f.1 <- as.formula(Z2 ~ XCORD + YCORD)
```

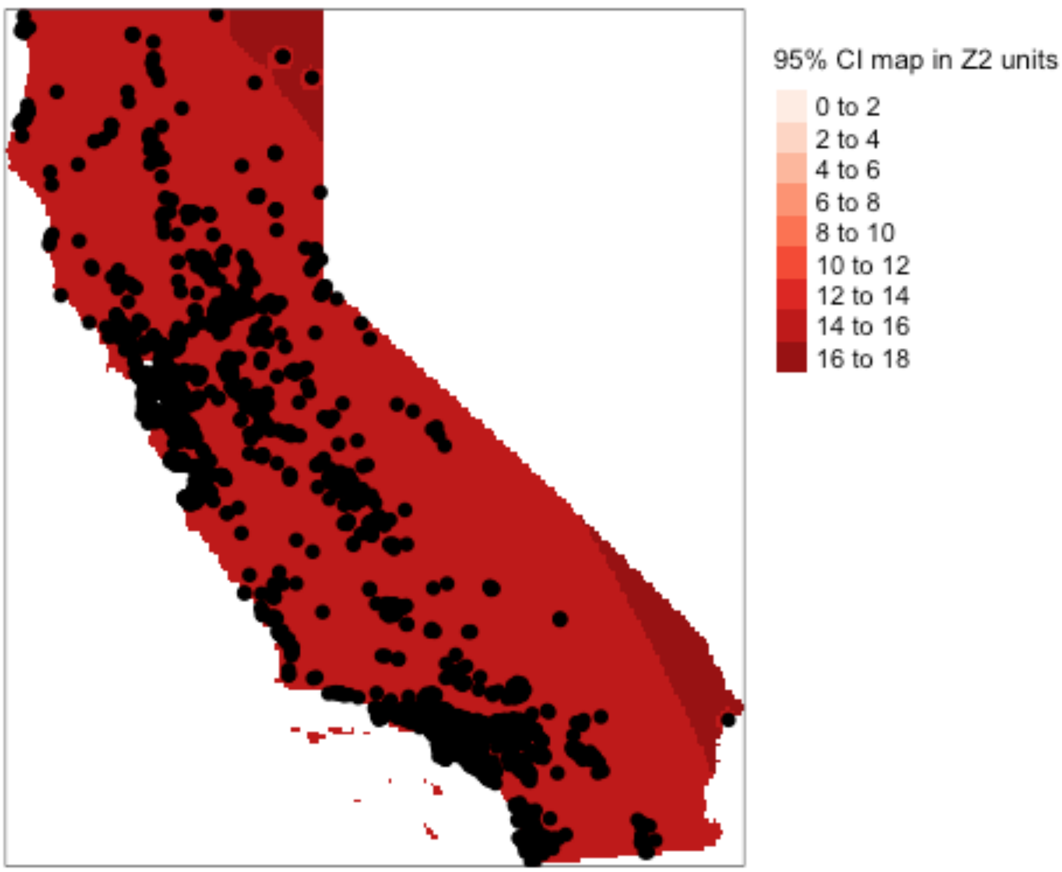
```r
# Perform the krige interpolation (note the use of the variogram model
# created in the earlier step)
dat.krg <- krige( f.1, P, grd, dat.fit)

## [using universal kriging]

# save the results in kxy to compare with ordinary kriging in object OK
kxy <- dat.krg

# Convert kriged surface to a raster object for clipping
r <- raster(dat.krg)
r.m <- mask(r, W)

# Plot the map
tm_shape(r.m) +
  tm_raster(n=10, palette="RdBu", auto.palette.mapping=FALSE,
            title="Predicted with Universal Kriging)") +
  tm_shape(P) + tm_dots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



```
#  The second layer of data in a kriging object contains the variance of pred
ictions

r   <- raster(dat.krg, layer="var1.var")
r.m <- mask(r, W)


tm_shape(r.m) +
  tm_raster(n=7, palette ="Reds",
            title="Variance map in squared units of Z2)") +tm_shape(P) + tm_d
ots(size=0.2) +
  tm_legend(legend.outside=TRUE)
```



```
#  A confidence interval around z2

r   <- sqrt(raster(dat.krg, layer="var1.var")) * 1.96
r.m <- mask(r, W)

tm_shape(r.m) +
  tm_raster(n=7, palette ="Reds",
            title="95% CI map in Z2 units") +tm_shape(P) + tm_dots(size=0.2)
+
  tm_legend(legend.outside=TRUE)
```



Are the two interpolations any different?

```
summary(OK)

## Object of class SpatialGridDataFrame
## Coordinates:
##              min        max
## XCORD -124.41536 -114.13370
## YCORD   32.52501   42.01237
## Is projected: FALSE
## proj4string :
## [+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0]
## Grid attributes:
##       cellcentre.offset   cellsize cells.dim
## XCORD        -124.39330 0.04412729       233
## YCORD          32.54707 0.04412729       215
## Data attributes:
##    var1.pred          var1.var
##  Min.   :-0.6578   Min.   : 0.5518
##  1st Qu.: 7.8605   1st Qu.:66.9975
##  Median : 7.8651   Median :67.0535
##  Mean   : 7.8629   Mean   :64.8462
##  3rd Qu.: 7.8651   3rd Qu.:67.0535
##  Max.   :41.4606   Max.   :67.0535

summary(kxy)

## Object of class SpatialGridDataFrame
## Coordinates:
##              min        max
## XCORD -124.41536 -114.13370
## YCORD   32.52501   42.01237
## Is projected: FALSE
## proj4string :
## [+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0]
## Grid attributes:
##       cellcentre.offset   cellsize cells.dim
## XCORD        -124.39330 0.04412729       233
## YCORD          32.54707 0.04412729       215
## Data attributes:
##    var1.pred          var1.var
##  Min.   :-0.1416   Min.   : 0.6138
##  1st Qu.: 5.3282   1st Qu.:65.4285
##  Median : 6.8938   Median :66.2749
##  Mean   : 7.1343   Mean   :65.3444
##  3rd Qu.: 9.0480   3rd Qu.:68.0121
##  Max.   :40.7528   Max.   :78.2874
```