

- 
- Tutorial: Trend surfaces in R**
- Ordinary Least Squares
  - Generalized Additive Models
  - Thin-plate splines
  - Generalized Additive Models
  - Thin-plate splines
  - Generalized Least Squares
  - GLS-Regression Kriging
  - Universal Kriging
- 

*D G Rossiter*  
*Cornell University, Section of Soil & Crop Sciences*  
*ISRIC-World Soil Information*

February 12, 2022

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Preparing for the exercise</b>	<b>1</b>
2.1	Computing environment . . . . .	1
2.2	Loading R packages . . . . .	1
2.3	Adjusting processing options . . . . .	3
<b>3</b>	<b>Dataset</b>	<b>3</b>
3.1	Loading and adjusting the dataset . . . . .	4
3.2	Making a spatial object . . . . .	7
3.3	Specifying a Coördinate Reference System . . . . .	9
<b>4</b>	<b>Exploratory analysis</b>	<b>11</b>

---

Version 3.11 Copyright © 2017, 2020-22 D G Rossiter All rights reserved.  
Reproduction and dissemination of the work as a whole (not parts) freely  
permitted if this original copyright notice is included. Sale or placement  
on a web site where payment must be made to access this document  
is strictly prohibited. To adapt or translate please contact the author  
([d.g.rossiter@cornell.edu](mailto:d.g.rossiter@cornell.edu)).

<b>5</b>	<b>Trend surface analysis by Ordinary Least Squares</b>	<b>14</b>
5.1	First-order trend surface . . . . .	15
5.2	Second-order trend surface . . . . .	20
<b>6</b>	<b>Trend surface prediction</b>	<b>25</b>
6.1	Creating a prediction grid . . . . .	25
6.2	Mapping the trend surface . . . . .	28
<b>7</b>	<b>Generalized Additive Models</b>	<b>32</b>
7.1	Fitting a Generalized Additive Model . . . . .	34
7.2	GAM prediction over the study area . . . . .	39
<b>8</b>	<b>Thin-plate spline interpolation</b>	<b>42</b>
<b>9</b>	<b>Spatial correlation of trend surface residuals</b>	<b>46</b>
9.1	The empirical variogram . . . . .	47
<b>10</b>	<b>Trend surface analysis by Generalized Least Squares</b>	<b>49</b>
10.1	Computing the GLS trend surface . . . . .	50
10.2	Predicting from the GLS trend surface . . . . .	52
<b>11</b>	<b>Local interpolation of the residuals</b>	<b>54</b>
11.1	Visualizing the residuals . . . . .	54
11.2	Variogram modelling . . . . .	56
11.3	The Ordinary Kriging system . . . . .	58
11.4	OK predictions . . . . .	59
<b>12</b>	<b>GLS-Regression Kriging</b>	<b>61</b>
<b>13</b>	<b>Universal Kriging (UK)</b>	<b>64</b>
13.1	Residual variogram . . . . .	65
13.2	The Universal Kriging system . . . . .	67
13.3	Prediction . . . . .	68
<b>14</b>	<b>Discussion</b>	<b>72</b>
<b>15</b>	<b>Answers</b>	<b>74</b>
<b>A</b>	<b>Derivation of the OLS solution to the linear model</b>	<b>79</b>
<b>B</b>	<b>Standardized residuals</b>	<b>79</b>
<b>C</b>	<b>Theory of thin-plate splines</b>	<b>80</b>
<b>D</b>	<b>Theory of GLS and REML</b>	<b>81</b>
D.1	GLS . . . . .	82
D.2	GLS with spatially-correlated residuals . . . . .	83
D.3	REML estimation of the covariance parameters . . . . .	84
	<b>References</b>	<b>85</b>
	<b>Index of R concepts</b>	<b>87</b>

## 1 Introduction

This exercise shows how to compute trend surfaces using the **R environment for statistical computing** [8, 18].

A *trend surface* is a map of some continuous variable, computed as a function of the coördinates. This corresponds to the concept of a *geographic trend*, where the variable changes its value along a geographic gradient.

The trend can be modelled as a *linear* trend, i.e., the variable increases or decreases a fixed amount for each unit change in the coördinates in some direction. This is called a *first-order* trend surface (§5.1). It can also be modelled as a *polynomial* trend, i.e., a linear model of some polynomials of the coördinates, for example, a quadratic, which is called a *second-order* trend surface (§5.2). It can also be modelled as an *empirical* smooth function of the coördinates, for example a generalized additive model (§7) or a minimum-curvature surface (thin-plate spline) (§8).

The residuals from any of the above approaches may have spatial structure (§9). This has two implications:

1. The OLS fit may not be optimal, and a Generalized Least Squares (GLS) trend should be fit (§10).
2. The OLS or GLS trend surfaces can be modified by (1) interpolating the residuals from the trend-surface fit (§11) and (2) adding these to the trend.
3. The trend and local deviations can be modelled together with Universal Kriging (UK) (§13).

In this exercise we compare these approaches.

## 2 Preparing for the exercise

### 2.1 Computing environment

The code to complete this tutorial can be executed in any R environment. A good choice is the RStudio<sup>1</sup> integrated development environment (IDE) for R.

### 2.2 Loading R packages

R has thousands of user-contributed packages beyond the core packages automatically loaded with R<sup>2</sup>.

In this exercise we use several of these packages. Their use will be explained as they are encountered in the tutorial.

---

<sup>1</sup> <https://www.rstudio.org>

<sup>2</sup> stats, graphics, grDevices, utils, datasets, methods, base

---

**TASK 1 :** Load the `sf` “simple (spatial) features”, the `gstat` “geostatistics”, the `ggplot2` “grammar of graphics”, the `gridExtra` “arrange multiple ggplot graphics on one figure”, the `units` “units of measure”, the `terra` “gridded (raster) data structures”, the `mgcv` “Generalized Additive Models”, the `fields` “curve and function fitting”, and the `nlme` “Linear and Nonlinear Mixed Effects Models” packages into your search path. They will be explained at their first use in the tutorial. •

**Note:** You can also load these via checkboxes in the RStudio “Packages” pane.

The `require` or `library` functions are used to load R packages.

```
require(sf)          # 'simple features' representations of spatial objects
## Loading required package:  sf
## Linking to GEOS 3.8.1, GDAL 3.2.1, PROJ 7.2.1; sf_use_s2() is TRUE
require(gstat)       # geostatistics
## Loading required package:  gstat
require(ggplot2)     # Grammer of Graphics plots
## Loading required package:  ggplot2
require(gridExtra)   # arrange multiple ggplot graphics on one figure
## Loading required package:  gridExtra
require(units)       # units of measure
## Loading required package:  units
## udunits database from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/units
require(terra)       # gridded data structures ("rasters")
## Loading required package:  terra
## terra 1.5.17
##
## Attaching package:  'terra'
## The following object is masked from 'package:ggplot2':
##
##   arrow
## The following object is masked from 'package:knitr':
##
##   spin
require(mgcv)        # for Generalized Additive Models
## Loading required package:  mgcv
## Loading required package:  nlme
## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.
require(fields)      # NCAR etc. approach to surfaces
## Loading required package:  fields
## Loading required package:  spam
## Spam version 2.8-0 (2022-01-05) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
```

```
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##      backsolve, forwardsolve

## Loading required package: viridis
## Loading required package: viridisLite

##
## Try help(fields) to get started.

##
## Attaching package: 'fields'

## The following object is masked from 'package:terra':
##
##      describe

require(nlme)      # Linear and Nonlinear Mixed Effects Models
```

## 2.3 Adjusting processing options

R has many options, which can be listed with the `options` function. Here we use this function to change the default option of showing the so-called “significance stars” in model summaries. These stars (\*, \*\*, \*\*\*) for various levels of the “significance level”<sup>3</sup>  $\alpha$  have been widely criticized because they are a lazy way to assess the success of models and the importance of predictors.<sup>4</sup>

```
options(show.signif.stars=FALSE)
```

## 3 Dataset

We use an example dataset that is well-suited to illustrate the concepts of trend surface: a set of observations on the elevation above mean sea level of the top of an aquifer in western Kansas, USA measured in 161 wells.

**Note:** This aquifer is in Miocene-Pliocene sedimentary rocks, the Ogalalla formation, and is an important source of irrigation water, especially for centre-pivot irrigation systems.

This dataset is used as an example in the well-known geology statistics text of Davis [3, pp. 435-438]<sup>5</sup>. The practical task is to map the elevation of the top of the aquifer over the study area.

---

**Q1 :** *What is the purpose of producing a map of the the elevation of the*

<sup>3</sup> interpreted as the probability of incorrectly rejecting a true null hypothesis of no effect

<sup>4</sup> This is part of a major debate about how statistics should be used to draw conclusions about the “real world”, see for example [6].

<sup>5</sup> The datasets for this book are available at <http://www.kgs.ku.edu/Mathgeo/Books/Stat/index.html>

top of the aquifer over the study area? In other words, who would use the map and for what purpose? Jump to A1 •

**Note:** More information on the aquifer monitoring network from which this dataset is taken is available at the Kansas Geological Survey<sup>6</sup>, for example Olea and Davis [14, 15]. The water-level logs are also available on-line<sup>7</sup>.

Figure 1 is taken from the original report [14]. It shows the location of wells, the boundary of the aquifer, and the well IDs. The example dataset uses a small portion of this, in the SE corner of the study area<sup>8</sup>. Figure 2 is a Google Earth view of part of the study area, with the location of several of the wells as placemarks.

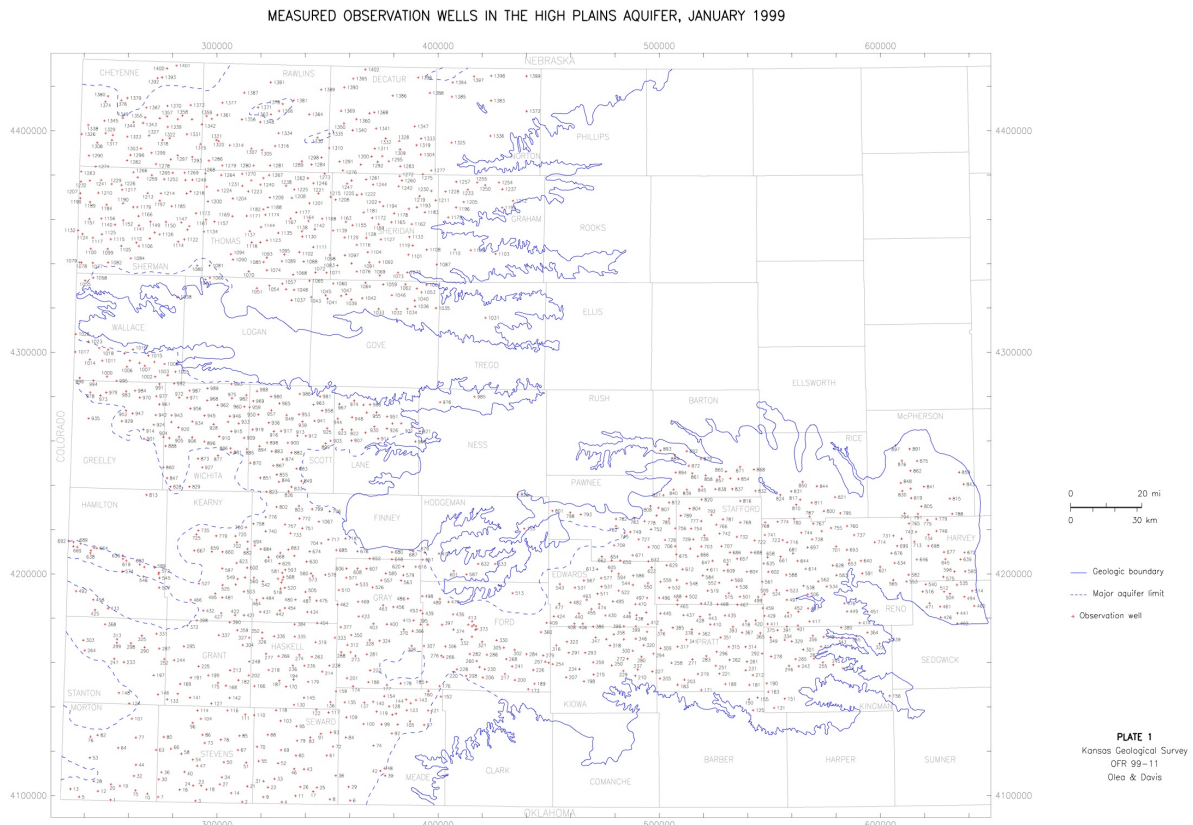


Figure 1: Location of aquifer monitoring wells, SE Kansas (USA). Source: [14], plate 1

### 3.1 Loading and adjusting the dataset

The dataset is a plain-text file, AQUIFER.TXT.

<sup>6</sup> <http://www.kgs.ku.edu>

<sup>7</sup> <http://www.kgs.ku.edu/Magellan/WaterLevels/>

<sup>8</sup> portions of Pratt, Kingman, Stafford and Reno counties

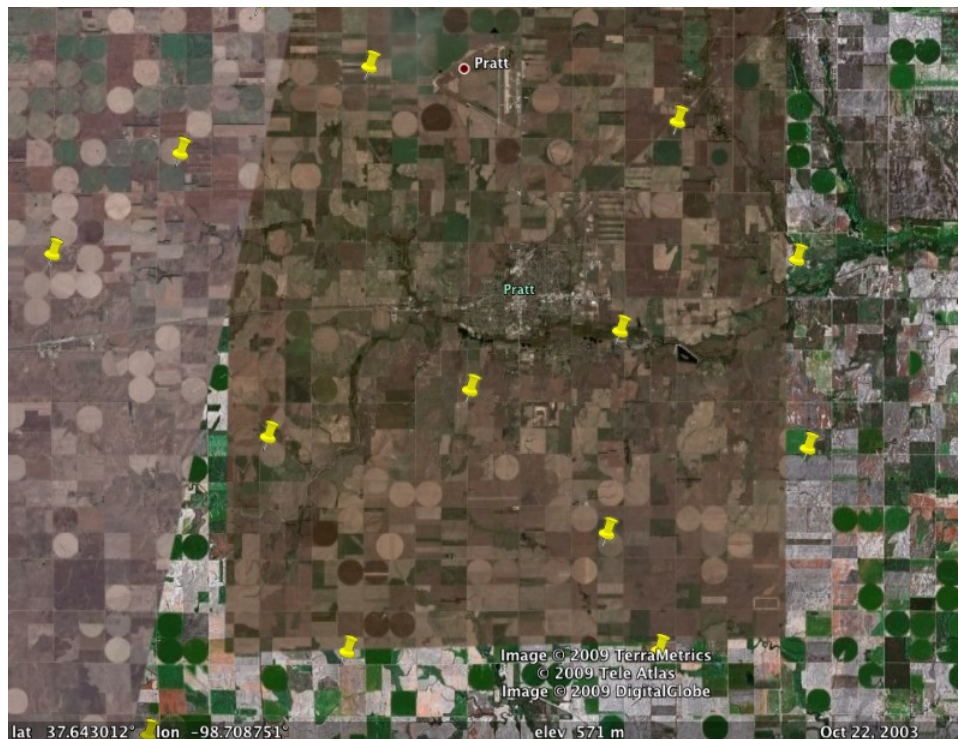


Figure 2: Google Earth view of part of the study area, with the location of several of the wells as placemarks

---

**TASK 2 :** Obtain this dataset, either from the instructor or by download from the KGS<sup>9</sup>. •

---

**TASK 3 :** Change R's *working directory* to where you have downloaded the text file `AQUIFER.TXT`. •

You can do this with the RStudio menu command Tools | Change directory..., or with the `setwd` function.

---

**TASK 4 :** Examine the contents of file `AQUIFER.TXT`. •

You can view this file from within RStudio, by opening it from the Files pane.

The first few lines look like this:

```
UTM easting UTM northing Water Table, ft.
569464.5 4172114.75 1627.66
573151.25 4167192.75 1588.83
```

---

<sup>9</sup> <http://www.kgs.ku.edu/Mathgeo/Books/Stat/ASCII/AQUIFER.TXT>



```
559973.94 4169585 1675.72
553514.44 4174584.5 1689.52
```

The field names are self-explanatory.

The Coördinate Reference System (CRS) is not specified, although we can guess from the field headers that the projection is Universal Transmector (UTM). The UTM zone is 14N (see Davis [3, Fig. 5-100 caption]) and the coördinates are defined in the UTM system as meters North from the equator and East from a false origin of 500 000 at the central meridian of the UTM zone. For zone 14 this is 99° W.

However to fully know the CRS, we must know the datum on which the projection is developed. This is not explained in the several Kansas Geological Survey reports; however a more comprehensive report covering the entire High Plains [11], states that the datum is the North American Datum of 1983 (NAD 83).

The aquifer elevation is in US feet<sup>10</sup> above mean sea level according to an unspecified vertical datum (probably NAVD 88).

---

**TASK 5 :** Read text file `AQUIFER.TXT` into an R data frame, rename the columns to shorter names, and examine its structure. •

The `read.table` function can read many kinds of tabular data. It has many arguments, to adjust to different text formats. See the R Data Import/Export Manual [17] for details.

By default the *data fields* in the text file are assumed to be separated by *white space* (tabs, spaces), as is the case here. Another optional argument is `skip`; we use it here because the header line of `AQUIFER.TXT` has more spaces than the other lines, so if we try to use the header for the variable names, R thinks the other lines are incomplete. One solution would be to place quotes around the variable names, or rename the variables, in the text file. What we do here is skip the first line and assign variable names ourselves in R.

Name the R data frame `aq`, standing for “aquifer”:

```
aq <- read.table("AQUIFER.TXT", skip=1)
```

Give the fields meaningful names:

```
str(aq)

## 'data.frame': 161 obs. of 3 variables:
## $ V1: num 569464 573151 559974 553514 550350 ...
## $ V2: num 4172115 4167193 4169585 4174584 4171337 ...
## $ V3: num 1628 1589 1676 1690 1691 ...

names(aq) <- c("E", "N", "Z")
str(aq)

## 'data.frame': 161 obs. of 3 variables:
## $ E: num 569464 573151 559974 553514 550350 ...
```

---

<sup>10</sup> 1 foot = 0.3048 m exactly



```
## $ N: num 4172115 4167193 4169585 4174584 4171337 ...
## $ z: num 1628 1589 1676 1690 1691 ...
```

The elevation should be converted to meters, to conform to international standards.

---

**TASK 6 :** Convert the elevation in feet above sea level to elevation in meters above sea level (m.a.s.l.), and add it as a new field in the dataframe.

This is conveniently done with the `units` package. This provides a class for maintaining unit metadata associated with numerical values. It automatically converts units, and simplifies units of results when possible. Here we just want to convert from feet to meters; this conversion is built into the known units of the package.

We first specify the units of elevation, which we know from the dataset description, using the `set_units` function of the `units` package. We then create a new field using the converted units.

```
aq$z <- set_units(aq$z, feet)
aq$zm <- set_units(aq$z, m)
summary(aq)
```

	E	N	z	zm
## Min.	:500361	Min. :4150248	Min. :1560	Min. :475.5
## 1st Qu.	:518465	1st Qu.:4176120	1st Qu.:1721	1st Qu.:524.6
## Median	:533366	Median :4197238	Median :1814	Median :552.8
## Mean	:535668	Mean :4198439	Mean :1808	Mean :551.0
## 3rd Qu.	:553569	3rd Qu.:4220405	3rd Qu.:1901	3rd Qu.:579.4
## Max.	:574430	Max. :4248312	Max. :2045	Max. :623.2

## 3.2 Making a spatial object

To do spatial analysis we need to make the dataset into an explicitly *spatial* data structure. We create **spatial objects** using the `sf` “simple (spatial) features” package<sup>11</sup>. Simple Features is an OGC and ISO set of standards that specify how geographic features can be stored and accessed by computer programs. The R implementation in `sf` places the spatial information in a special field of a `data.frame` named `geometry`.

**Note:** This implementation uses open-source code GDAL for reading and writing data, GEOS for geometrical operations, and PROJ for projection conversions and datum transformations. In this simple application we do not use any of these.

A spatial object is a set of entities, each with explicit coördinates. The `aq` dataframe does have coördinates, but “hidden” as attributes. These in fact have a special status. To continue the analysis, we identify these explicitly as being spatial.

---

**TASK 7 :** Make an explicitly-spatial version of the point dataset, using

<sup>11</sup> <https://r-spatial.github.io/sf/articles/sf1.html>

the UTM coördinates for the geometry.

The `st_as_sf` method of the `sf` package converts a dataframe into a spatial object, by specifying which fields (columns) in the dataframe contain the coördinates. Here we use the columns `E` and `N`, in that order. Note that the local coördinates `e` and `n` created just above are still in the data frame as fields.

**Note:** All methods of this package begin with `st_`, an abbreviation of “space-time”. The `*_as_*` methods convert between data types.

```
aq.sf <- st_as_sf(aq, coords=c("E", "N"))
str(aq.sf)

## Classes 'sf' and 'data.frame': 161 obs. of 3 variables:
## $ z : Units: [feet] num 1628 1589 1676 1690 1691 ...
## $ zm : Units: [m] num 496 484 511 515 516 ...
## $ geometry:sfc_POINT of length 161; first list element: 'XY' num 569464 4172115
## - attr(*, "sf_column")= chr "geometry"
## - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",...: NA NA
## ..- attr(*, "names")= chr [1:2] "z" "zm"
```

This structure display is different from the previous (non-spatial) one: The object now is of class `sf`, which is an extension of the `data.frame` class. There is now a `geometry` field, which contains the spatial information. The attribute `sf_column` gives this name. The real-world coördinates have been removed from the dataframe into this field and now have special status. The names we had given to the coördinates were not preserved in the geometry; they were replaced

The information in the original dataframe is now clearly split into two kinds:

Geographic space : Coordinates; location of the observation in some coördinate reference system, here UTM14N localized to the local centre;

Feature-space : Also called *attribute space*: properties of the observation. Here there is only one, the aquifer elevation, in both feet and metres. However, the local coördinates are still shown as feature-space attributes, even though they are not. We will use these for models that are not explicitly spatial.

We can see the coördinates with the `st_coordinates` function:

```
str(st_coordinates(aq.sf))

## num [1:161, 1:2] 569464 573151 559974 553514 550350 ...
## - attr(*, "dimnames")=List of 2
## ..$ : chr [1:161] "1" "2" "3" "4" ...
## ..$ : chr [1:2] "X" "Y"

#
summary(st_coordinates(aq.sf))

##           X           Y
## Min.      :500361   Min.      :4150248
## 1st Qu.:518465   1st Qu.:4176120
## Median :533366   Median :4197238
## Mean      :535668   Mean      :4198439
## 3rd Qu.:553569   3rd Qu.:4220405
## Max.      :574430   Max.      :4248312
```

This is a 2D matrix, with the X (East) coördinate in the first column and the Y (North) coördinate in the second column.

### 3.3 Specifying a Coördinate Reference System

Since the coördinates refer to the real world, we should specify their Coördinate Reference System (CRS); this allows integration with any other geographic data set.

The CRS is displayed and set with the `st_crs` function:

```
st_crs(aq.sf)

## Coordinate Reference System: NA
```

At this point it is NA, “not available”. This is because the CSV file from which we obtained the coördinates only has these as numbers. This file format has no place for metadata such as a CRS.

In the analysis of this tutorial we do not need to specify a CRS, since we just work with the numbers as independent (predictor) variables. However, if we want to later use with other spatial data we should specify the CRS. Above (§3.1) we determined the CRS is UTM14N on the NAD83 datum.

To specify this we find the EPSG code for this CRS in the EPSG database<sup>12</sup> and discover this is code 26914. See Figure 3. Note that UTM14N has been defined on several datums, we select the correct one for this dataset based on the metadata.

**Note:** The EPSG database follows the ISO 19111:2019 international standard for referencing by coordinates, including its provision for dynamic datums, geoid-based vertical datums, datum ensembles and derived projected coordinate reference systems. The “EPSG” name is a historical artefact, from the original name “European Petroleum Survey Group”, which is now expanded into the International Association of Oil & Gas Producers (IOGP)<sup>13</sup>. The fossil fuel industry needed a harmonized method to resolve competing claims for resources, specified using many CRS.

Specify the CRS of the spatial object by its EPSG code and show its long form as Well-Known Text (WKT)<sup>14</sup>.

```
st_crs(aq.sf) <- 26914 # EPSG code
print(st_crs(aq.sf))

## Coordinate Reference System:
##   User input: EPSG:26914
##   wkt:
##   PROJCRS["NAD83 / UTM zone 14N",
##     BASEGEOGCRS["NAD83",
##       DATUM["North American Datum 1983",
##         ELLIPSOID["GRS 1980",6378137,298.257222101,
##           LENGTHUNIT["metre",1]]],
##       PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##       ID["EPSG",4269]],
```

<sup>12</sup> <https://epsg.org/>

<sup>13</sup> <https://www.iogp.org/>

<sup>14</sup> <http://docs.opengeospatial.org/is/12-063r5/12-063r5.html>

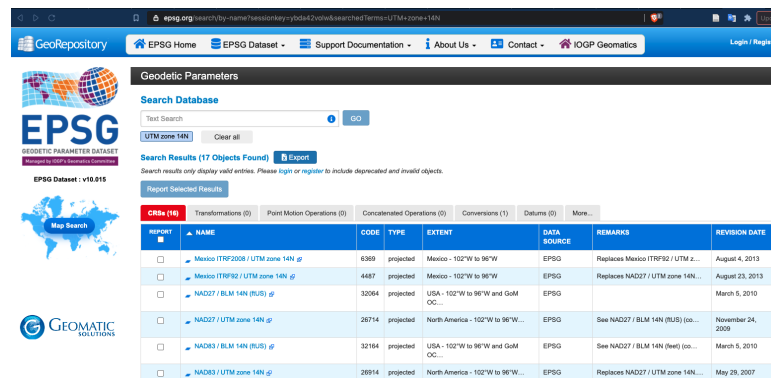


Figure 3: Finding a code in the EPSG database

```
## CONVERSION["UTM zone 14N",
## METHOD["Transverse Mercator",
## ID["EPSG",9807]],
## PARAMETER["Latitude of natural origin",0,
## ANGLEUNIT["degree",0.0174532925199433],
## ID["EPSG",8801]],
## PARAMETER["Longitude of natural origin",-99,
## ANGLEUNIT["degree",0.0174532925199433],
## ID["EPSG",8802]],
## PARAMETER["Scale factor at natural origin",0.9996,
## SCALEUNIT["unity",1],
## ID["EPSG",8805]],
## PARAMETER["False easting",500000,
## LENGTHUNIT["metre",1],
## ID["EPSG",8806]],
## PARAMETER["False northing",0,
## LENGTHUNIT["metre",1],
## ID["EPSG",8807]],
## CS[Cartesian,2],
## AXIS["(E)",east,
## ORDER[1],
## LENGTHUNIT["metre",1]],
## AXIS["(N)",north,
## ORDER[2],
## LENGTHUNIT["metre",1]],
## USAGE[
## SCOPE["Engineering survey, topographic mapping."],
## AREA["North America - between 102°W and 96°W - onshore and offshore."],
## BBOX[25.83,-102,84,-96]],
## ID["EPSG",26914]]
```

**Note:** It is possible to define a custom CRS by a long list of parameters, if there is no EPSG code.

We've done some work to get this data set into proper form for spatial analysis; so we save it in this format, as well as the data frame (non-spatial) format, using the save function.

**TASK 8 :** Save the spatial object as an R Data file.

```
save(aq, aq.sf, file="aquifer.rda")
```

This can be read into a later R session with the load method.

## 4 Exploratory analysis

As with any unfamiliar dataset, the first step is to examine its contents. In the case of spatially-explicit datasets, that includes visualizing its geography.

---

**TASK 9 :** Summarize the dataset. •

The `dim` function gives the dimensions of the matrix (here, a `data.frame`); the `summary` function summarizes it appropriately.

```
dim(aq.sf)

## [1] 161  3

summary(aq.sf)

##           z           zm           geometry
## Min.      :1560   Min.    :475.5   POINT      :161
## 1st Qu.:1721   1st Qu.:524.6   epsg:26914  : 0
## Median :1814   Median :552.8   +proj=utm ...: 0
## Mean    :1808   Mean    :551.0
## 3rd Qu.:1901   3rd Qu.:579.4
## Max.    :2045   Max.    :623.2
```

---

**Q2 :** *How many observations are there? What was recorded at each point?* [Jump to A2](#) •

---

**Q3 :** *What are the geographic limits of the study area? What is its area, in  $km^2$ ?* [Jump to A3](#) •

We can see the bounding box with the `st_bbox` function:

```
st_bbox(aq.sf)

##      xmin      ymin      xmax      ymax
## 500361.3 4150248.2 574429.6 4248312.5
```

The `range` function computes the range of numeric variable; the `diff` function computes the difference between two numeric values. Here it's simpler to refer the coordinate fields in the non-spatial object. We display the ranges as  $km$  and the area as  $km^2$ .

```
range(aq$E)/1000

## [1] 500.3613 574.4296

range(aq$N)/1000

## [1] 4150.248 4248.312

print(diff(range(aq$E)) * diff(range(aq$N))/1000^2)

## [1] 7263.444
```

---

**TASK 10 :** Find the location of this sample area in the large study area,

shown in Fig. 1.

**Q4:** What is the range of elevations (in metres) in the sample set? *Jump to A4*

```
range(aq.sf$zm); diff(range(aq.sf$zm))

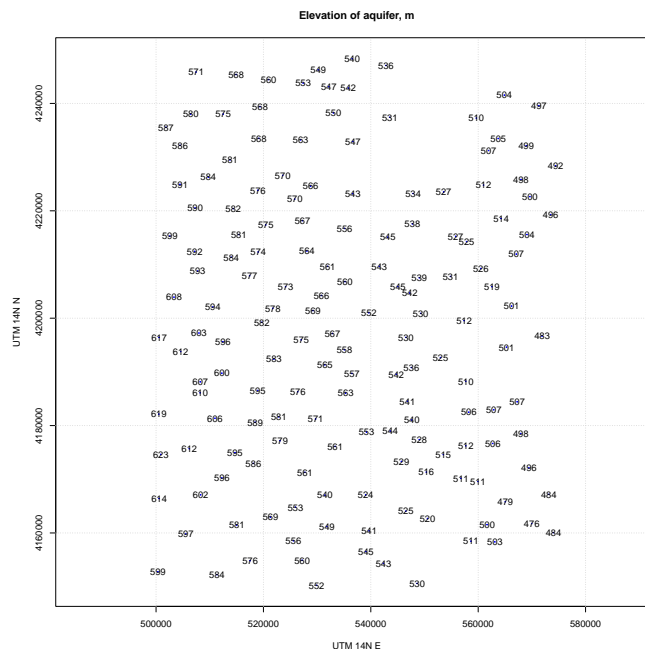
## Units: [m]
## [1] 475.5032 623.2428
## 147.7396 [m]
```

We now try three different **visualizations** of the distribution of the data values (i.e. aquifer elevations); these are known as **postplots**. To keep the geographic reference, we use the original UTM 14N coördinates.

**TASK 11:** Display a **text postplot** of the data values, showing the elevations, rounded to the nearest metre, as text labels centred at the observation point.

We use the two coördinates as plot axes, so this looks like a map. Here we use the plot methods of the R *base graphics*.

```
plot(st_coordinates(aq.sf)[,2] ~ st_coordinates(aq.sf)[,1],
     pch=20, cex=0.2, col="blue", asp=1,
     xlab="UTM 14N E", ylab="UTM 14N N")
grid()
text(st_coordinates(aq.sf)[,1], st_coordinates(aq.sf)[,2],
     round(aq$zm), adj=c(0.5,0.5))
# text(aq$E, aq$N, round(aq$zm), adj=c(0.5,0.5))
title("Elevation of aquifer, m")
```



The aquifer elevation is clearly higher in the west (towards the Rocky Mountains about 650 km to the west, where it outcrops).

**Note:** Parameter `cex` is an expansion factor; here we plot a very small blue dot and then add the data value at each point with the `text` method. The `adj` argument centres the text at the point. The `asp=1` argument makes the two axes have the same scale. This is necessary to get a true map when the study area is not square.

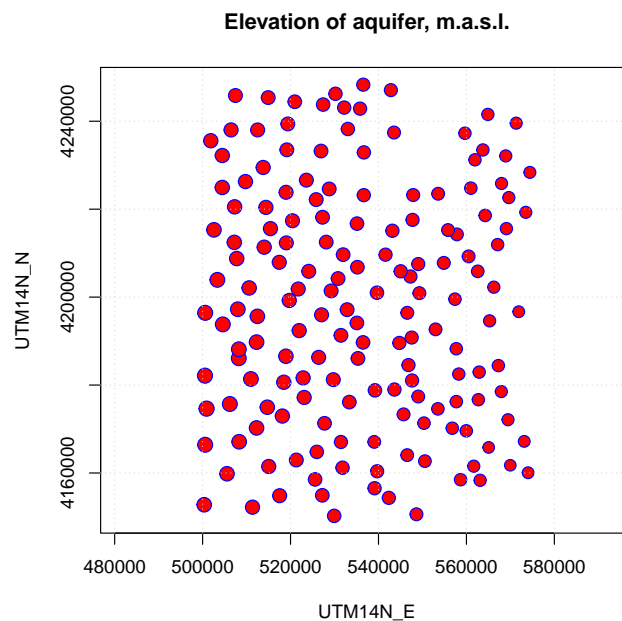
Another visualization is with the symbol size proportional to the the data value.

---

**TASK 12 :** Display a graphical postplot of the data values, with size proportional to the data value. •

Here we use the non-spatial dataset for convenience.

```
plot(aq$N ~ aq$E,  
     cex=1.8*aq$zm/max(aq$zm),  
     col="blue", bg="red", pch=21, asp=1,  
     xlab="UTM14N_E", ylab="UTM14N_N")  
grid()  
title("Elevation of aquifer, m.a.s.l.")
```



**Note:** Print character (`pch`) 21 has both a symbol (`col`) and fill (`bg`) colour.

A final visualization combines both size and colour:

---

**TASK 13 :** Display a graphical postplot of the data values, with size and colour proportional to the data value •

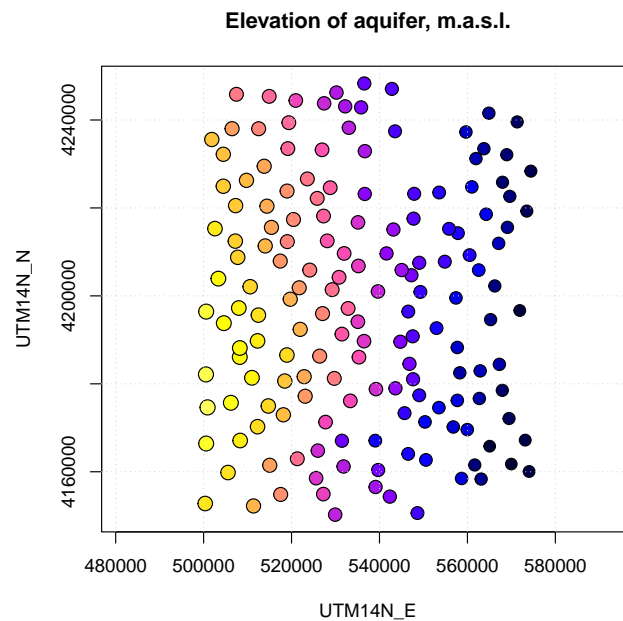
Notice the use of the `rank` function to give the rank order of the elevations; these are then used as indices into a vector of colours, created with the `bpy.colors` function, of the same length as the vector of elevation values.



The `~` formula operator show the functional relation between two variables; here it is the North coordinate for the y-axis, depending on the East coordinate for the x-axis.

**Note:** The `bpy.colors` function of the `sp` package provides a colour palette that also show the colour ramp if printed in grey scale.

```
plot(aq$N ~ aq$E, pch=21,
     xlab="UTM14N_E", ylab="UTM14N_N",
     bg=sp::bpy.colors(length(aq$zm))[rank(aq$zm)],
     cex=1.8*aq$zm/max(aq$zm), asp=1)
grid()
title("Elevation of aquifer, m.a.s.l.")
```




---

**Q5 :** Describe the spatial pattern of the elevations. Do nearby points have similar values? Is there a trend across the whole area? Are there local exceptions to the trend? [Jump to A5](#) •

---

**Q6 :** Discuss the relative advantages of the three types of postplot. [Jump to A6](#) •

## 5 Trend surface analysis by Ordinary Least Squares

The visualizations suggest a **trend surface**, i.e., the aquifer elevations is some smooth function of the coordinates. This is a polynomial function of the coordinates to any degree (1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup> etc.), which is called the **order** of the surface.

The higher the degree, the more the surface can match the points, but the degree should also be chosen to match a plausible process, in this

case, the structure of the aquifer. Also, the higher the degree, the more extreme are extrapolations, i.e., predictions outside the convex hull of the calibration points.

## 5.1 First-order trend surface

We begin with a **first-order** trend: a plane defined by the two coördinates and an intercept that sets the overall level, here the aquifer elevation.

---

**Q7 :** *What is the geological interpretation of a first-order trend surface of the aquifer?* Jump to A7 •

A trend surface has the same form as a standard linear model, using the coördinates as regression predictors. The first-order trend surface model has the form:

$$z = \beta_0 + \beta_1 E + \beta_2 N + \varepsilon \quad (1)$$

where  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ , i.e., independently and normally distributed. This assumption allows us to fit the trend surface with Ordinary Least Squares (OLS).

In the linear model, with *any* number of predictors, there is a  $n \times p$  **design matrix** of predictor values usually written as **X**, with one row per observation (data point), i.e.,  $n$  rows, and one column per predictor, i.e.,  $p$  columns. In the first-order trend surface case, it is a  $n \times 3$  matrix with three columns: (1) a column of 1 representing the intercept, to center the response, (2) a column of predictor values  $e_i$  from the Easting, and (3) a column of predictor values  $n_i$  from the Northing.

The `model.matrix` function shows the model matrix for the right-hand side of an R model formula; the `head` function limits the number of rows to display. Because of limitations in the `lm` function (see below) we work with the non-spatial version of the dataset, where the coördinates are simply fields in the data frame.

```
head(model.matrix(~E + N, data=aq))
```

```
##      (Intercept)      E      N
## 1      1 569464.5 4172115
## 2      1 573151.2 4167193
## 3      1 559973.9 4169585
## 4      1 553514.4 4174584
## 5      1 550349.6 4171337
## 6      1 556788.6 4170184
```

The predictand (response variable), here the aquifer elevation is a  $n \times 1$  column vector **y**, one row per observation. The coefficient vector  $\beta$  is a  $p \times 1$  column vector, i.e., one row per predictor (here, 3). This multiplies the design matrix to produce the response:<sup>15</sup>

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon \quad (2)$$

---

<sup>15</sup> The dimensions of the matrix multiplication are  $n \times 1 = (n \times p)(p \times 1)$

where  $\varepsilon$  is a  $n \times 1$  column vector of **residuals**, also called **errors**, i.e., the lack of fit. We know the values in the predictor matrix  $X$  and the response vector  $y$  from our observations, so the task is to find the optimum values of the coefficients vector  $\beta$ . This can be found directly; see the Appendix [A](#) for the derivation. The OLS solution is:

$$\hat{\beta}_{\text{ols}} = (X^T X)^{-1} X^T \cdot y \quad (3)$$

where  $X$  is the design matrix.

The term “first-order” refers to the power to which each coördinate is raised; here it is the first power, so it’s a first-order trend surface.

**Note:** This assumption of uncorrelated residuals is in fact *not* true in this case; we prove this in §9 below. So the trend surface should in fact be fit not by OLS but by Generalized Least Squares (GLS), taking into account the spatial auto-correlation of the residuals. We pursue this further in §10.

For this dataset with many observations well-spread in space, the result will be similar to the OLS estimate.

---

**TASK 14 :** Fit a first-order trend surface (i.e. linear in the E and N coördinates) to the elevations. Summarize the model and evaluate its goodness-of-fit. •

The `lm` “linear model” function fits linear models.

We can do this two ways (1) from the coördinates stored in the `data.frame`; (2) from the coördinates stored in the `geometry` field of the `sf` object; these are extracted with the `st_coordinates` method.

**Note:** Unfortunately, `lm` does not properly propagate the units defined with the `units` package through its calculations. Therefore we have to first remove them with the `drop_units` function.

```
model.ts1 <- lm(zm ~ E + N, data=drop_units(aq))
summary(model.ts1)

##
## Call:
## lm(formula = zm ~ E + N, data = drop_units(aq))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.3550  -5.8267   0.2674   7.1062  16.7349
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.557e+03  1.080e+02  14.41  <2e-16
## E          -1.617e-03  3.201e-05 -50.51  <2e-16
## N           -3.336e-05  2.528e-05  -1.32   0.189
##
## Residual standard error: 8.629 on 158 degrees of freedom
## Multiple R-squared:  0.9417, Adjusted R-squared:  0.941
## F-statistic: 1276 on 2 and 158 DF, p-value: < 2.2e-16

model.ts1.sf <- lm(zm ~ st_coordinates(aq.sf), data=drop_units(aq.sf))
summary(model.ts1.sf)
```

```
##
## Call:
## lm(formula = zm ~ st_coordinates(aq.sf), data = drop_units(aq.sf))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.3550  -5.8267   0.2674   7.1062  16.7349
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.557e+03  1.080e+02   14.41  <2e-16
## st_coordinates(aq.sf)X -1.617e-03  3.201e-05 -50.51  <2e-16
## st_coordinates(aq.sf)Y -3.336e-05  2.528e-05  -1.32   0.189
##
## Residual standard error: 8.629 on 158 degrees of freedom
## Multiple R-squared:  0.9417, Adjusted R-squared:  0.941
## F-statistic: 1276 on 2 and 158 DF, p-value: < 2.2e-16
```

In this summary the proportion of variability explained is given by the adjusted  $R^2$ . This is  $(1 - \text{RSS}/\text{TSS})$ , i.e., a perfect fit less the ratio of the residual sum-of-squares  $\text{RSS} = \sum_i (z_i - \hat{z}_i)^2$  to the total sum of squares  $\text{TSS} = \sum_i (z_i - \bar{z})^2$ , adjusted for the degrees of freedom and number of observations, where  $z_i$  is the observed value and  $\hat{z}_i$  is the value predicted by the fitted linear model.

---

**Q8:** *What is the equation of the trend surface? How does elevation vary with the E and N coordinates? Is the relation statistically-significant? How much of the total variability does it explain? Are all the coefficients statistically-significant?* Jump to A8 •

**Challenge:** Linear models can be unstable if the predictors are correlated. Do you expect a problem with this for this model? Why or why not? Can you prove your conjecture statistically?

---

**TASK 15:** Summarize the **residuals** (lack of fit) from the trend surface both numerically and graphically, in feature space. Express this in terms of the median elevation. •

The `residuals` function extracts the residuals from a linear model object. The `hist` function displays a histogram of a numeric vector.

```
res.ts1 <- set_units(residuals(model.ts1), m); summary(res.ts1)

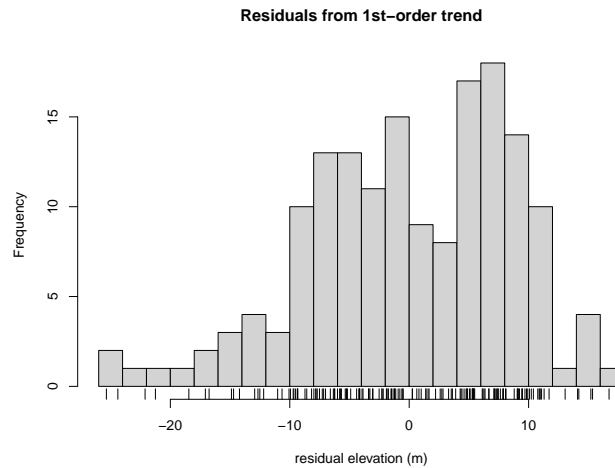
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -25.3550 -5.8267   0.2674   0.0000   7.1062  16.7349

hist(res.ts1, breaks=16,
      main="Residuals from 1st-order trend",
      xlab="residual elevation (m)")
rug(res.ts1)
range(res.ts1)

## Units: [m]
## [1] -25.35498 16.73489

max(abs(res.ts1))/median(aq$zm)*100

## 4.586769 [1]
```




---

**Q9 :** *What is the range of residuals? How does this compare with the target variable? How are the residuals distributed in feature space?*  
[Jump to A9](#) •

---

**TASK 16 :** Show the diagnostic plots of the linear model. •

The `plot` method applied to a linear model object produces some *diagnostic plots*. We will display the most important: (1) residuals vs. fitted values; (2) quantile-quantile (“QQ”) plot of the standardized residuals<sup>16</sup>.

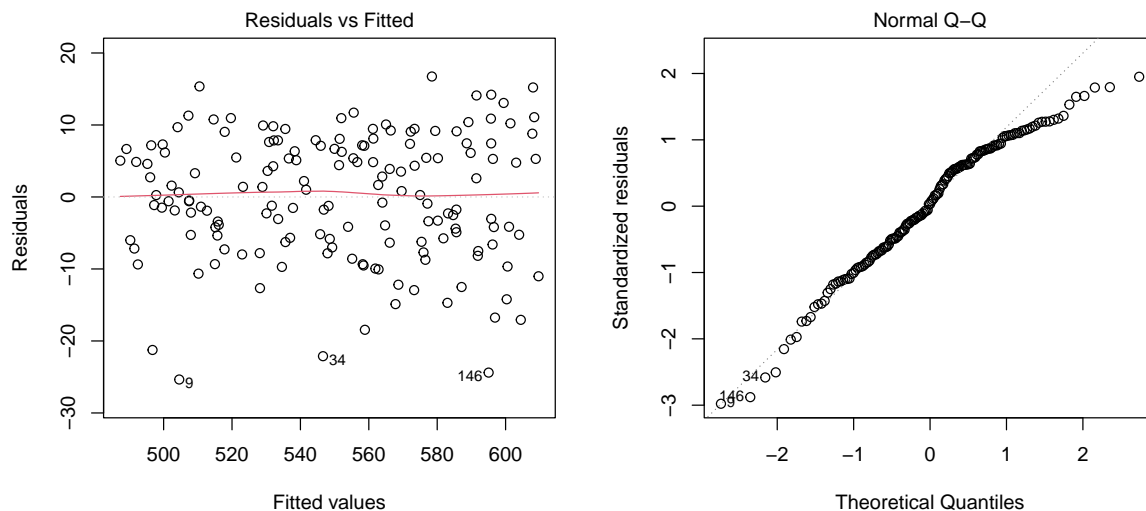
The Q-Q plot shows (1) on the y-axis, the standardized residuals, (2) on the x-axis, the standardized residuals that *would be expected* if the residuals were from a normal distribution with the mean and standard deviation computed from the actual standardized residuals. These two should match exactly on 1:1 line.

**Note:** The `par` “parameters” function here sets up a 1 row by 2 column matrix of plots, because the `plot.lm` function here will display two plots as requested by the `which` optional argument.

---

<sup>16</sup> These are explained in §B

```
par(mfrow=c(1,2))
plot(model.ts1, which=1:2)
par(mfrow=c(1,1))
```



**Q10 :** Does this model meet the feature-space requirements for a valid linear model?

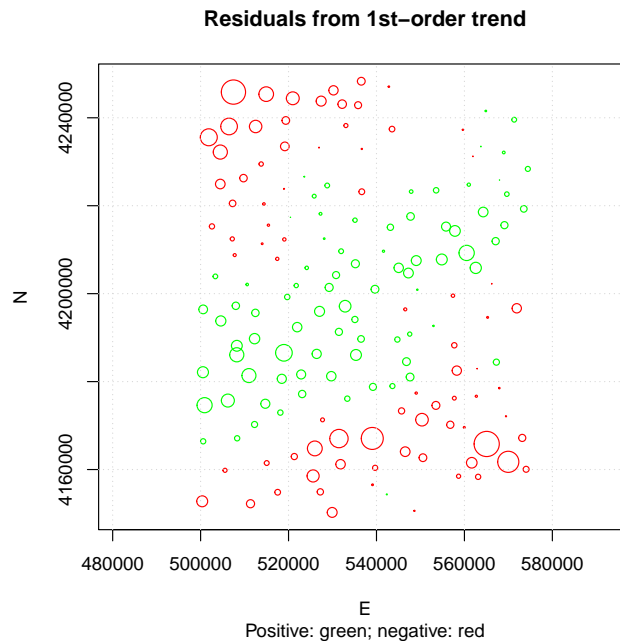
1. No relation between the fitted values and the residuals;
2. Equal spread of residuals across the range of fitted values;
3. Normally-distributed standardized residuals.

*Jump to A10 •*

**TASK 17 :** Display the residuals as a postplot. •

**Note:** In the following code, the expression for the `cex` “character expansion” optional argument sets the size of the circle as each residual's proportion of the maximum residual, so that the larger absolute values of the residuals show larger circles. This way we can visualize where are the largest over- and under-predictions. The `ifelse` statement applied to the `col` “color” optional argument sets the color of the circle according to whether the residual is positive (under-prediction) or negative (over-prediction).

```
plot(aq$N ~ aq$E, cex=3*abs(res.ts1)/max(abs(res.ts1)),
     col=ifelse(res.ts1 > set_units(0, m), "green", "red"),
     xlab="E", ylab="N",
     main="Residuals from 1st-order trend",
     sub="Positive: green; negative: red", asp=1)
grid()
```




---

**Q11 :** *Is there a spatial pattern to the residuals? Is there local spatial correlation without an overall pattern? What does this imply about the suitability of a first-order trend surface?* Jump to A11 •

## 5.2 Second-order trend surface

We see from the pattern of residuals from the first-order surface that there is still structure, in particular clear bands of positive and negative residuals. These suggest that a higher-order trend surface might fit better. A higher-order trend might also fix the problems with the regression diagnostics.

A **second-order** trend includes linear and quadratic (squared) functions of the coördinates. It allows the surface to be convex or concave, not just a plane.

---

**Q12 :** *What is the geological interpretation of a second-order trend surface of the aquifer?* Jump to A12 •

A full second-order surface uses the coördinates, their squares, and their cross-products.

$$z = \beta_0 + \beta_1 E + \beta_2 N + \beta_3 E^2 + \beta_4 N^2 + \beta_5 (E * N) + \varepsilon \quad (4)$$

This can also be expressed in matrix notation of Equation 2, where the design matrix has six columns, one per predictor.

```
head(model.matrix(~ E + N + I(E^2) + I(N^2) + I(E*N),
  data=aq))
```



```
## (Intercept)      E      N      I(E^2)      I(N^2)      I(E * N)
## 1      1 569464.5 4172115 324289816760 1.740654e+13 2.375871e+12
## 2      1 573151.2 4167193 328502355377 1.736550e+13 2.388432e+12
## 3      1 559973.9 4169585 313570813479 1.738544e+13 2.334859e+12
## 4      1 553514.4 4174584 306378235289 1.742716e+13 2.310693e+12
## 5      1 550349.6 4171337 302884638192 1.740005e+13 2.295693e+12
## 6      1 556788.6 4170184 310013500547 1.739043e+13 2.321911e+12
```

---

**TASK 18:** Fit a second-order trend surface to the aquifer elevations. •

We fit with a full second-order polynomial, for convenience using the non-spatial object:

```
model.ts2 <- lm(zm ~ E + N + I(N^2) + I(E^2) + I(E*N),
               data=drop_units(aq))
summary(model.ts2)

##
## Call:
## lm(formula = zm ~ E + N + I(N^2) + I(E^2) + I(E * N), data = drop_units(aq))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.847  -3.366   0.822   3.538  14.807
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.161e+05  1.156e+04 -10.043  < 2e-16
## E           -2.798e-02  3.387e-03  -8.263  5.92e-14
## N            5.937e-02  5.439e-03  10.914  < 2e-16
## I(N^2)       -7.500e-09  6.435e-10 -11.655  < 2e-16
## I(E^2)       -1.648e-09  1.074e-09  -1.534   0.127
## I(E * N)      6.700e-09  7.781e-10   8.610  7.74e-15
##
## Residual standard error: 5.598 on 155 degrees of freedom
## Multiple R-squared:  0.9759, Adjusted R-squared:  0.9751
## F-statistic: 1256 on 5 and 155 DF, p-value: < 2.2e-16
```

**Note:** The I “identity” function must be used for the squares and cross-product terms, because the ^ and \* symbols represent the usual mathematical operators.

If this function is not used, `lm` interprets the ^ and \* symbols as formula operators, rather than as their normal mathematical meanings.

---

**Q13:** *How much of the variance does the second-order surface explain?*  
[Jump to A13](#) •

---

**TASK 19:** Compare the second-order model statistically with the first-order model. •

The anova “analysis of variance” method compares the residual sums-of-squares of two or more models and computes the probability that the more complicated model<sup>17</sup> is not better than the less complicated model.

We list the simpler model first, i.e., the one with fewer predictors and

---

<sup>17</sup> with more predictors, and thus fewer degrees of freedom

more degrees of freedom, and the more complex model second. We expect the second model will have a lower values of the residual sum-of-squares, since it will explain more of the variance. But, because we have used more degrees of freedom, perhaps the F-ratio of the two variances adjusted for their degrees of freedom will be low, showing that the more complex model is not in fact an improvement.

Let's see:

```
anova(model.ts1, model.ts2)

## Analysis of Variance Table
##
## Model 1: zm ~ E + N
## Model 2: zm ~ E + N + I(N^2) + I(E^2) + I(E * N)
##   Res.Df    RSS Df Sum of Sq    F    Pr(>F)
## 1      158 11763.5
## 2      155  4858.2   3    6905.3 73.438 < 2.2e-16
```

---

**Q14:** *Is the second-order surface statistically superior to the first-order surface?* [Jump to A14](#) •

---

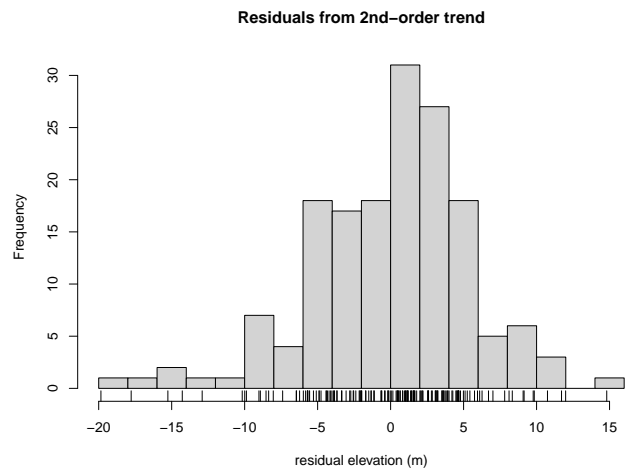
**TASK 20:** Summarize the residuals from the second-order trend surface both numerically and graphically, in feature space. Express this in terms of the median elevation. •

```
res.ts2 <- set_units(residuals(model.ts2), m)
summary(res.ts2)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -19.847  -3.366   0.822   0.000   3.538   14.807

hist(res.ts2, breaks=16,
      main="Residuals from 2nd-order trend",
      xlab="residual elevation (m)")
rug(res.ts2)
max(abs(res.ts2))/median(aq$zm)

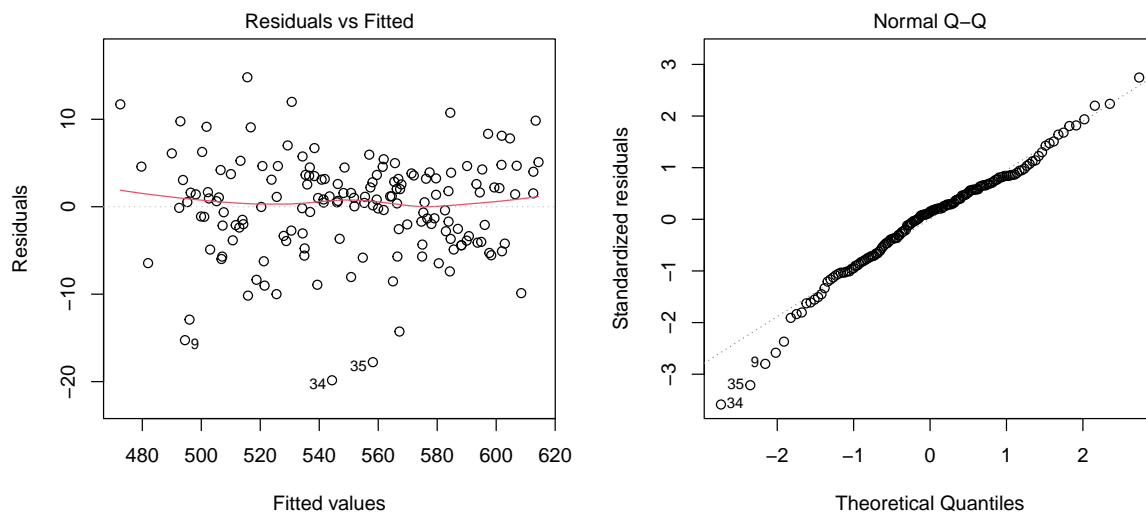
## 0.03590352 [1]
```



**Q15 :** What is the range of residuals? How does this compare with the target variable? How are they distributed in feature space? How do these compare with the residuals from the first-order surface? [Jump to A15](#) •

**TASK 21 :** Show the diagnostic plots of the residuals, as for the first-order trend surface residuals. •

```
par(mfrow=c(1,2))
plot(model.ts2, which=1:2)
par(mfrow=c(1,1))
```



**Q16 :** Does this model meet the feature-space requirements for a valid linear model? How do these diagnostics compare to those from the first-

order surface?

1. No relation between the fitted values and the residuals;
2. Equal spread of residuals across the range of fitted values;
3. Normally-distributed standardized residuals.

[Jump to A16](#) •

---

**TASK 22 :** Identify the largest over-predictions that do not fit the normal Q-Q plot. •

We look for the rows in the data frame with large negative residuals and then show their entries. First we need to define what we mean by “large”; here we’ll take it as two standard deviations of the normalized residuals. These are the values shown in the normal Q-Q plot (right-hand panel of the previous figure). The standardized residuals are extracted with the `rstandard` function.

```
summary(sres.ts2 <- rstandard(model.ts2))

##      Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
## -3.586915 -0.630210  0.151225  0.000582  0.644235  2.745775

(ix <- which(sres.ts2 < -2))

##  9 34 35 41 72
##  9 34 35 41 72

(cbind(actual=aq[ix, "zm"], fitted=fitted(model.ts2)[ix],
        residual=res.ts2[ix],
        std.res <- sres.ts2[ix]))

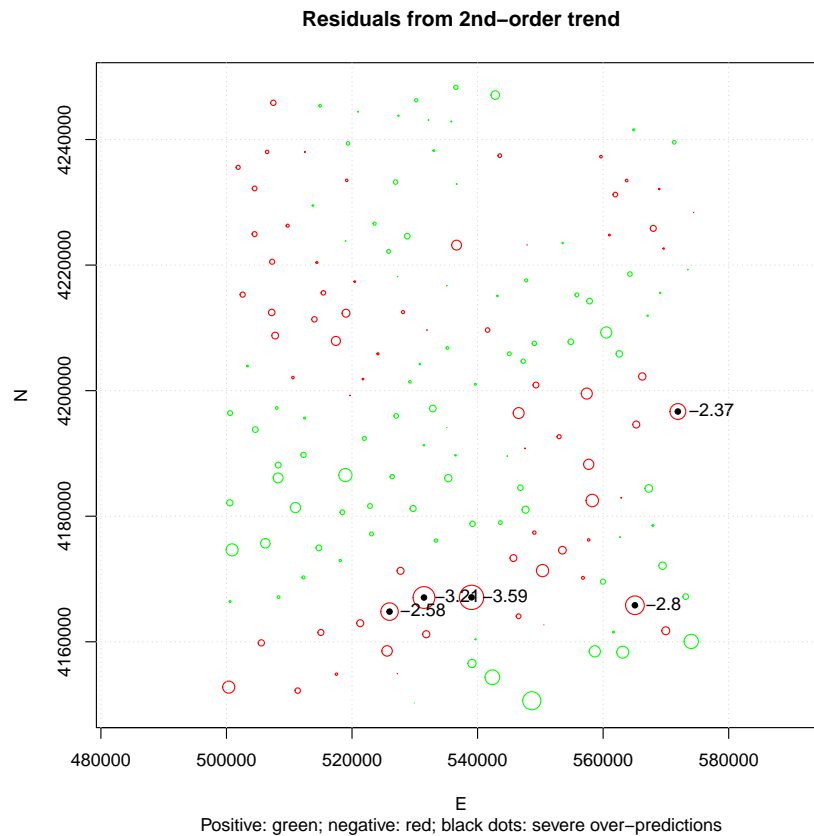
##      actual   fitted  residual
##  9  479.2096  494.4700 -15.26043 -2.799307
## 34  524.4876  544.3346 -19.84694 -3.586915
## 35  540.4074  558.1822 -17.77486 -3.212499
## 41  552.9255  567.1991 -14.27359 -2.584241
## 72  483.0989  496.0088 -12.90997 -2.369209
```

---

**TASK 23 :** Display the residuals as a postplot; compare to the postplot from the first-order trend surface. •

We also highlight the largest over-predictions and display their standardized residuals.

```
plot(aq$N ~ aq$E, cex=3*abs(res.ts2)/max(abs(res.ts2)),
     col=ifelse(res.ts2 > set_units(0, m), "green", "red"),
     xlab="E", ylab="N", asp=1,
     main="Residuals from 2nd-order trend",
     sub="Positive: green; negative: red; black dots: severe over-predictions")
points(aq[ix, "N"] ~ aq[ix, "E"], pch=20)
text(aq[ix, "E"], aq[ix, "N"], round(sres.ts2[ix], 2), pos=4)
grid()
```




---

**Q17 :** *Is there an overall pattern to the residuals? Is there local spatial correlation without an overall pattern? Are the largest model residuals clustered? Does there seem to be any anisotropy (stronger spatial dependence in one direction than the orthogonal direction)?* [Jump to A17](#)

Since this second-order trend surface is much better than the first-order trend surface, we will use it for subsequent modelling.

## 6 Trend surface prediction

We now use the trend surface model of the previous section to **predict** over the study area, discretized as an **interpolation grid** at some resolution that we choose.

### 6.1 Creating a prediction grid

We first make a grid over which to predict.

---

**TASK 24 :** Create a grid of equally-spaced (1 x 1 km) points across the study area, beginning with UTM (500 000E, 4150 000N) in the lower-left corner, as in Davis [3, Fig. 5-100, 5-101, 5-102].

**Note:** The choice of grid resolution depends on (1) the support of the observations used to model the trend surface, (2) the resolution needed by the map user; (3) for larger areas, computer memory and processing time.

Here the observations are essentially point support, so there is no minimum grid size. The map user will use this to decide on whether to drill a well into the aquifer, based on cost which depends on the depth from the surface. In this area the surface elevation is quite uniform and does not vary much. Also, the aquifer in this area does not have sharp changes in structure, it is gently dipping to the E, with a slight dome. We know from the 1<sup>st</sup>-order OLS model that for each km E (direction of maximum dip) the aquifer elevation decreases by  $1.5573268 \times 10^6$  m, which we suppose is hardly significant to the well driller. Indeed, a coarser grid would probably be sufficient.

The `seq` function creates a regular sequence of numbers; the `expand.grid` function makes a grid from two sequences.

First, find the bounding box:

```
range(aq$E); range(aq$N)

## [1] 500361.3 574429.6
## [1] 4150248 4248312

range(st_coordinates(aq.sf)[, "X"]); range(st_coordinates(aq.sf)[, "Y"])

## [1] 500361.3 574429.6
## [1] 4150248 4248312
```

Then use these, rounded below and above to the nearest kilometer, as the limits of the two axes. For this we use `min` and `max` to get the extremes, and then `floor` and `ceiling`, respectively, to round them to the nearest integer below and above. We could just examine the above output and enter the required integers directly, but it is more elegant and reliable to do this programmatically. We align the grid on even km.

```
(n.col <- length(seq.e <- seq(min.x <- floor(min(aq$E)/1000)*1000,
                             max.x <- ceiling(max(aq$E)/1000)*1000, by=1000)))

## [1] 76

(n.row <- length(seq.n <- seq(min.y <- floor(min(aq$N)/1000)*1000,
                             max.y <- ceiling(max(aq$N)/1000)*1000, by=1000)))

## [1] 100
```

The `rast` method of the `terra` package sets up a raster grid, of class `SpatRaster`. Here we have no values, just a structure. We will add values from the model results. For now, define the layer as real-valued NA “not available”, i.e., undefined. The CRS is set to match the point dataset.

```
grid1km <- rast(nrows = n.row, ncols = n.col,
               xmin=min.x, xmax=max.x,
               ymin=min.y, ymax=max.y, crs = st_crs(aq.sf)$proj4string,
               resolution = 1000, names="z")
values(grid1km) <- NA_real_
class(grid1km)

## [1] "SpatRaster"
## attr(,"package")
```

```
## [1] "terra"

dim(grid1km)

## [1] 99 75 1

summary(grid1km)

##           z
## Min.      : NA
## 1st Qu.: NA
## Median : NA
## Mean     :NaN
## 3rd Qu.: NA
## Max.      : NA
## NA's     :7425

st_crs(grid1km)$proj4string

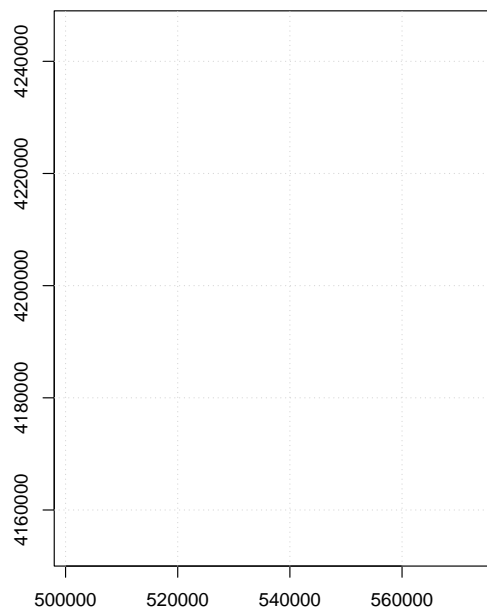
## [1] "+proj=utm +zone=14 +datum=NAD83 +units=m +no_defs"

st_bbox(grid1km) # replaces `raster` package `bbox()`

##      xmin      ymin      xmax      ymax
## 500000 4150000 575000 4249000
```

There are three dimensions: E, N, and z.

```
plot(grid1km); grid()
```




---

**TASK 25 :** Create a `data.frame` version of the grid, with the coördinates as fields; use the optional `xy=TRUE` argument for this). Name these fields to match coördinate names of the point dataset. Also use the optional `na.rm = FALSE` argument to keep the cells with NA values – we will replace these with the results of model predictions. •



```
grid1km.df <- as.data.frame(grid1km, xy = TRUE, na.rm = FALSE)
names(grid1km.df)[1:2] <- c("E", "N") # match the names of the point dataset
summary(grid1km.df)
```

##	E	N	z
## Min.	:500500	Min. :4150500	Min. : NA
## 1st Qu.:	:518500	1st Qu.:4174500	1st Qu.: NA
## Median :	:537500	Median :4199500	Median : NA
## Mean :	:537500	Mean :4199500	Mean :NaN
## 3rd Qu.:	:556500	3rd Qu.:4224500	3rd Qu.: NA
## Max. :	:574500	Max. :4248500	Max. : NA
##			NA's :7425

## 6.2 Mapping the trend surface

**TASK 26 :** Map the values of the second-order trend surface onto this grid. Compute both the best fit and a 95% prediction interval for each point on the grid. •

The `predict.lm` function, applied to a linear model object, computes the predicted values at new locations, in this case the regular grid. The optional `interval` argument specifies that a prediction interval, as well as the best fits, should also be computed. The optional `level` argument specifies the  $(1 - \alpha)$  probability, where  $\alpha$  is the probability that, on repeated calculation from a similar sample, the true value at the point would not be included in the computed prediction interval.

```
pred.ts2 <- predict.lm(model.ts2,
                      newdata = grid1km.df,
                      interval = "prediction", level = 0.95)
summary(pred.ts2)
```

##	fit	lwr	upr
## Min.	:463.1	Min. :451.1	Min. :475.2
## 1st Qu.:	:517.5	1st Qu.:506.3	1st Qu.:528.8
## Median :	:547.9	Median :536.7	Median :559.2
## Mean :	:547.4	Mean :536.1	Mean :558.7
## 3rd Qu.:	:577.3	3rd Qu.:566.0	3rd Qu.:588.6
## Max. :	:614.4	Max. :603.0	Max. :625.8

```
class(pred.ts2)
## [1] "matrix" "array"
```

The `predict.lm` produces three fields in the resulting object: `fit` (the best fit value), `lwr` (the value at the lowest 2.5% limit) and `upr` (the value at the upper 2.5% limit). The prediction interval is a range in which future observations are expected to fall, with a given probability specified by the analyst. It is based on the known observations and the regression model.

There are two sources of prediction error:

1. The uncertainty of fitting the best regression parameters from the available data;
2. The uncertainty in the prediction, even with perfect regression parameters, because of uncertainty in the process which is revealed

by the regression, i.e., the inherent noise in the process.

The prediction interval is computed from the *prediction variance*, which is then assumed to represent the variance of a *t*-distribution.

The prediction variance  $s_{Y_0}^2$  for predictand  $x_0$  depends on the variance of the regression  $s_{Y.x}^2$  but also on the distance of the predictor  $x_0$  from the value of the predictor at the centroid of the regression,  $\bar{x}$ . The further from the centroid, the more any error in estimating the slope of the line will affect the prediction:

$$s_{Y_0}^2 = s_{Y.x}^2 \left[ 1 + \frac{1}{n} + \frac{(\mathbf{x}_0 - \bar{\mathbf{x}})^2}{\sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})^2} \right] \quad (5)$$

where  $\mathbf{x}$  refers to both coördinates.

The variance of the regression  $s_{Y.x}^2$  is computed from the squared deviations of actual ( $y_i$ ) and estimated ( $\hat{y}_i$ ) values:

$$s_{Y.x}^2 = \frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (6)$$

---

**TASK 27:** Add the model predictions (fits) to the grid, using the `values` method of the `terra` package. •

```
summary(values(grid1km))

##           z
## Min.      : NA
## 1st Qu.:  NA
## Median:    NA
## Mean      :NaN
## 3rd Qu.:  NA
## Max.      : NA
## NA's      :7425

summary(pred.ts2[, "fit"])

##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
## 463.1   517.5   547.9   547.4   577.3   614.4

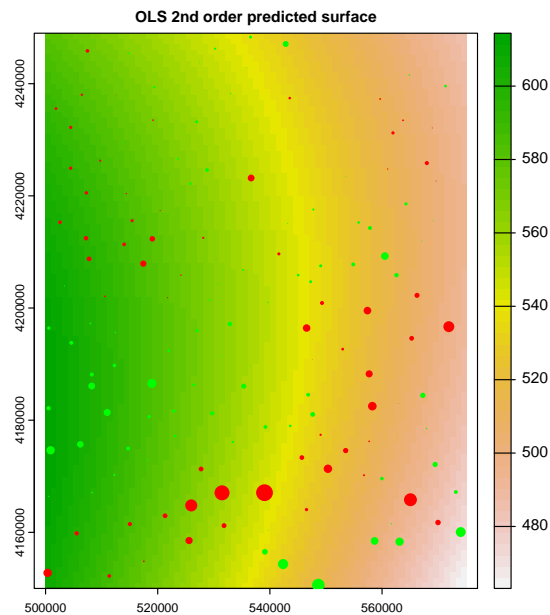
values(grid1km) <- pred.ts2[, "fit"]
summary(values(grid1km))

##           z
## Min.      :463.1
## 1st Qu.: 517.5
## Median: 547.9
## Mean      :547.4
## 3rd Qu.: 577.3
## Max.      :614.4
```

---

**TASK 28:** Display the interpolated surface, with the residuals at each observation point superimposed. •

```
plot(grid1km, main = "OLS 2nd order predicted surface")
points(st_coordinates(aq.sf)[,2] ~ st_coordinates(aq.sf)[,1], pch=16,
       col = ifelse(res.ts2 < set_units(0, m), "red", "green"),
       cex=2*abs(res.ts2)/max(abs(res.ts2))
)
```



The generic `plot` method applied to an `SpatRaster` object gives a reasonable plot.

We also show the *residual* from the model at each observation point is shown (1) in colour: red = negative (actual < predicted), green = positive (actual > predicted), (2) with the size proportional to the error. So points with no prediction error will not be visible. Note the use of the `ifelse` function to select between two alternatives, here the colours, based on a logical expression.

---

**Q18 :** *How well does the trend surface fit the points? Are there obvious problems?* Jump to A18 •

---

**TASK 29 :** Add the three prediction fields (fit, lower, upper) to the data frame version of the grid. •

```
summary(pred.ts2)
```

##	fit	lwr	upr
##	Min. :463.1	Min. :451.1	Min. :475.2
##	1st Qu.:517.5	1st Qu.:506.3	1st Qu.:528.8
##	Median :547.9	Median :536.7	Median :559.2
##	Mean :547.4	Mean :536.1	Mean :558.7
##	3rd Qu.:577.3	3rd Qu.:566.0	3rd Qu.:588.6
##	Max. :614.4	Max. :603.0	Max. :625.8

```
summary(grid1km.df)
```

##	E	N	z
##	Min. :500500	Min. :4150500	Min. : NA
##	1st Qu.:518500	1st Qu.:4174500	1st Qu.: NA
##	Median :537500	Median :4199500	Median : NA
##	Mean :537500	Mean :4199500	Mean :NaN
##	3rd Qu.:556500	3rd Qu.:4224500	3rd Qu.: NA

```
## Max. :574500 Max. :4248500 Max. : NA
## NA's :7425

grid1km.df[, 3:5] <- pred.ts2
names(grid1km.df)[3:5] <- c("ts2.fit", "ts2.lwr", "ts2.upr")
summary(grid1km.df)

##           E           N           ts2.fit           ts2.lwr
## Min. :500500 Min. :4150500 Min. :463.1 Min. :451.1
## 1st Qu.:518500 1st Qu.:4174500 1st Qu.:517.5 1st Qu.:506.3
## Median :537500 Median :4199500 Median :547.9 Median :536.7
## Mean :537500 Mean :4199500 Mean :547.4 Mean :536.1
## 3rd Qu.:556500 3rd Qu.:4224500 3rd Qu.:577.3 3rd Qu.:566.0
## Max. :574500 Max. :4248500 Max. :614.4 Max. :603.0
## ts2.upr
## Min. :475.2
## 1st Qu.:528.8
## Median :559.2
## Mean :558.7
## 3rd Qu.:588.6
## Max. :625.8
```

---

**TASK 30 :** Summarize the uncertainty from the trend surface, as absolute differences between the upper and lower prediction limits, and then this as a percentage of the best fit value. •

```
summary(grid1km.df$ts2.diff.range <- grid1km.df$ts2.upr - grid1km.df$ts2.lwr)

## Min. 1st Qu. Median Mean 3rd Qu. Max.
## 22.30 22.33 22.45 22.56 22.71 24.21

summary(100*grid1km.df$ts2.diff.range/grid1km.df$ts2.fit)

## Min. 1st Qu. Median Mean 3rd Qu. Max.
## 3.706 3.897 4.093 4.141 4.345 5.204
```

The `lwr` “lower” and `upr` “upper” fields of the prediction object contain the lower and upper limits of the 95% *prediction interval* for each point on the grid. Their difference is the range of uncertainty; this divided by the fit is an approximation to 2 standard deviations.

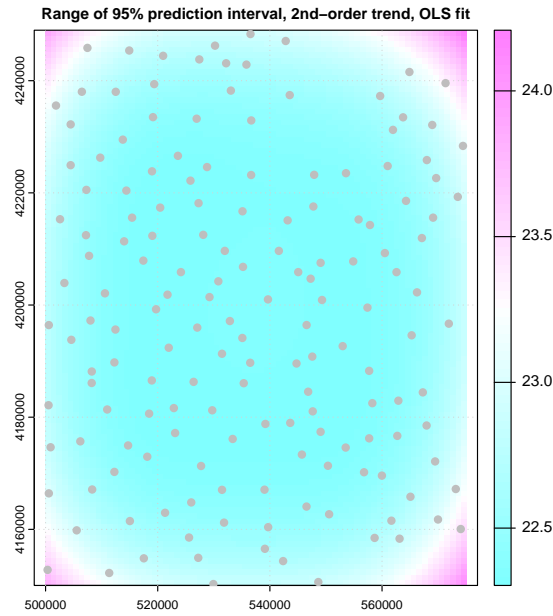
---

**TASK 31 :** Display the prediction interval of the trend surface as a map, showing also the location of the observation points. •

**Note:** We should use a different colour palette for different types of information. Here we want to show the standard error of prediction, which is a different concept than the prediction itself. So we choose a different palette, with the `col` argument.

We make a new raster with this information.

```
grid1km.diff <- grid1km
values(grid1km.diff) <- grid1km.df$ts2.diff.range
plot(grid1km.diff, col=cm.colors(64),
     main="Range of 95% prediction interval, 2nd-order trend, OLS fit")
grid()
points(st_coordinates(aq.sf)[,2] ~ st_coordinates(aq.sf)[,1], pch=16,
      col = "gray")
```



**Note:** Here we only show the locations of the observations, because their values do not affect the prediction variance, once the model is fit.

---

**Q19 :** *What are the units of prediction interval? How large are they? How does this compare to the variable we are trying to predict?* [Jump to A19](#) •

---

**Q20 :** *Describe the spatial pattern of the prediction interval.* [Jump to A20](#) •

## 7 Generalized Additive Models

Generalized Additive Models (GAM) are similar to multiple linear regression, except that each term in the linear sum of predictors need not be the predictor variable itself, but can be an empirical smooth function of it. So instead of the linear model of  $k$  predictors:

$$y_i = \beta_0 + \sum_k \beta_k x_{k,i} + \varepsilon_i \quad (7)$$

we allow *functions*  $f_k$  of these:

$$y_i = \beta_0 + \sum_k f_k(x_{k,i}) + \varepsilon_i \quad (8)$$

The **advantage** is that non-linear relations in nature can be fit, without any need to try transformations or to fit piecewise regressions. If this is a better model fit, it should result in better predictions. The model is

additive, so the marginal contribution of each predictor to the model fit can be determined.

The **disadvantage** is that it is just an empirical fit and can not be extrapolated beyond the range of calibration. A further disadvantage is that the choice of function is arbitrary; it is generally some smooth function of the predictor, with the degree of smoothness determined by cross-validation.

**Note:** The GAM should never be extrapolated (there is no data to support it), whereas a polynomial can, with caution, be extrapolated, on the theory that the data used to fit the model extends outside the range. This is of course very dangerous for higher-order polynomials, which are a main competitor to GAM.

Hastie et al. [5, §9.1] give a thorough explanation of GAM; a simplified explanation of the same material is given in James et al. [9, §7.7]. In a geostatistical setting, we can choose the coördinates as the predictors (as in a trend surface) but fit these with smooth functions, rather than polynomials. We can also fit any other predictor this way, e.g., in this example, the elevation.

The smooth functions can be chosen in many ways; the most common are cubic splines with knots at each value of the predictor. We first examine whether a smooth curve, rather than one line (as in linear regression) better matches the dependence of the annual ground water level by the two coördinates.

---

**TASK 32 :** Display a scatterplot of the three predictors against the annual GDD50, with an empirical smoother. •

We use the `ggplot2` graphics package to produce the scatterplot and show a smoother with standard error. A simple way to visualize the trend is with a local polynomial regression, provided with the `loess` function, and incorporated into the scatterplot with the `geom_smooth` function.

**Note:** The `loess` function has an `span` argument, which controls the degree of smoothing by setting the neighbourhood for the local fit as a proportion of the number of points. The default `span=0.75` thus uses the 3/4 of the total points closest each point. These are then weighted so that closer points have more weight; see `?loess` for details. The default works well in most situations, and here we only want a visual impression, not a “best fit” in a statistical sense.

We use the `gridExtra` package, which includes a function `grid.arrange` to arrange saved plots in a grid.

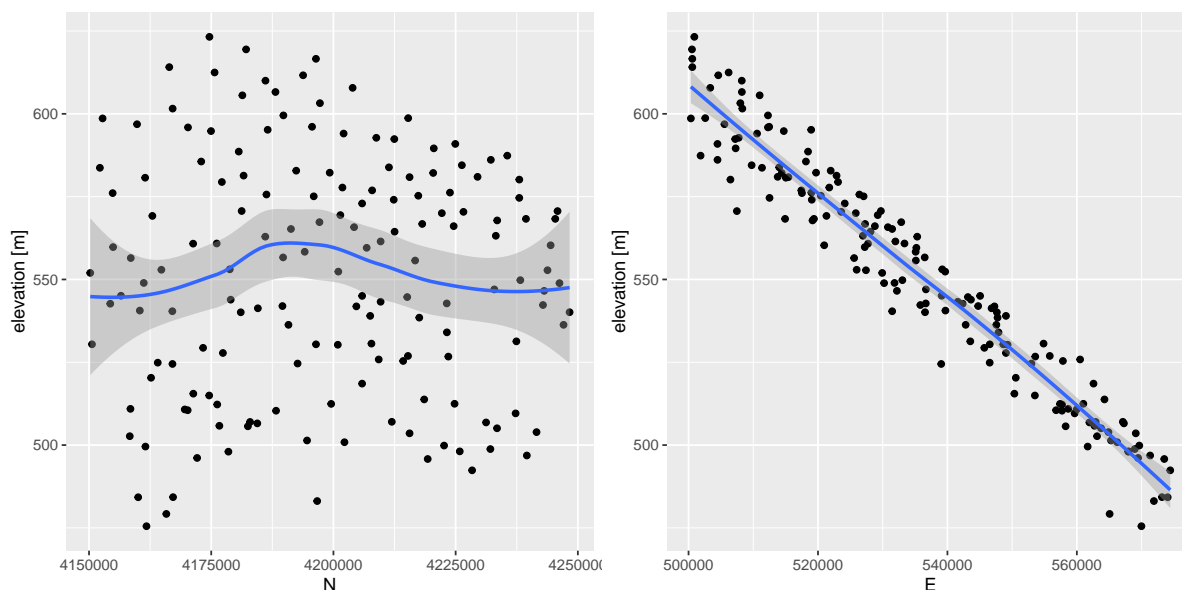
**Note:** The `ggplot2` package does not understand units, so these have to be removed from the data frame `aq`.

```

g1 <- ggplot(drop_units(aq), aes(x=N, y=zm)) +
  geom_point() +
  geom_smooth(method="loess") +
  labs(y = "elevation [m]")
g2 <- ggplot(drop_units(aq), aes(x=E, y=zm)) +
  geom_point() +
  geom_smooth(method="loess") +
  labs(y = "elevation [m]")
grid.arrange(g1, g2, ncol = 2)

## 'geom_smooth()' using formula 'y ~ x'
## 'geom_smooth()' using formula 'y ~ x'

```



**Q21 :** *Do these marginal relations appear to be linear in the predictors?*  
[Jump to A21](#) •

## 7.1 Fitting a Generalized Additive Model

GAM can be fit in R with the `gam` function of the `mgcv` “Mixed GAM Computation Vehicle” package. This specifies the model with a formula, as with `lm`, but terms can now be arbitrary functions of predictor variables, not just the variables themselves or simple transformations that apply to the whole range of the variable, e.g. `sqrt` or `log`. Smooth functions of one or more variables are specified with the `s` function of the `mgcv` package.

Common practice in GAM for models using coordinates is to smooth them together, i.e., in 2D, with a bivariate smoother. By default this smoother is a thin plate regression spline (see §8, below). These control the degree of smoothness by penalizing increasingly complex models, i.e., those with more curvature; see the help text ?s “defining smooths in GAM formulae” for details. In practice the default parameters work well.



---

**TASK 33 :** Fit a GAM to the aquifer elevation at the observation stations, with the predictors being a two-dimensional thin-plate spline of the coördinates. •

```
model.gam <- gam(zm ~ s(E, N), data=drop_units(aq))
summary(model.gam)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## zm ~ s(E, N)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 551.0134      0.2814   1958  <2e-16
##
## Approximate significance of smooth terms:
##             edf Ref.df    F p-value
## s(E,N) 27.35  28.83 543.2  <2e-16
##
## R-sq.(adj) =  0.99   Deviance explained = 99.2%
## GCV = 15.478   Scale est. = 12.752    n = 161
```

---

**Q22 :** *How well does this model fit the calibration observations?* [Jump to A22](#) •

---

**TASK 34 :** Compare the residuals from the GAM with those from the best linear trend surface, i.e., the quadratic. •

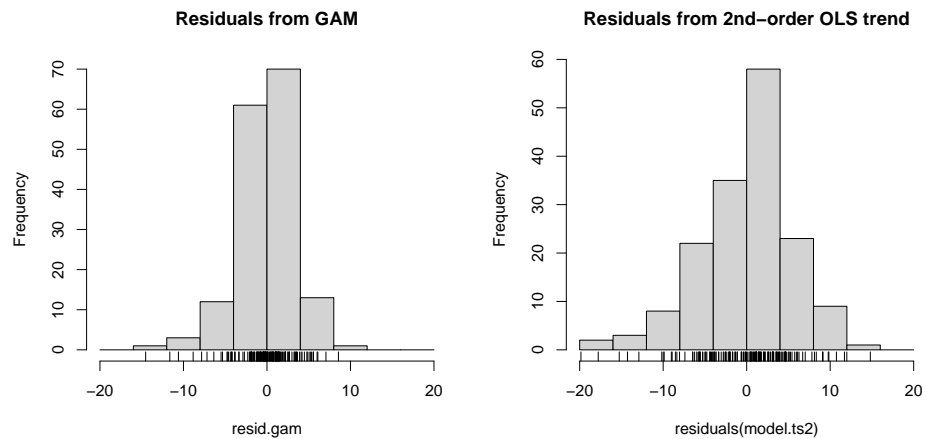
```
resid.gam <- residuals(model.gam)
summary(resid.gam)

##      Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
## -14.52803 -1.16470  0.08707  0.00000  1.51482  8.56012

summary(residuals(model.ts2))

##      Min. 1st Qu.  Median     Mean 3rd Qu.     Max.
## -19.847 -3.366  0.822  0.000  3.538  14.807

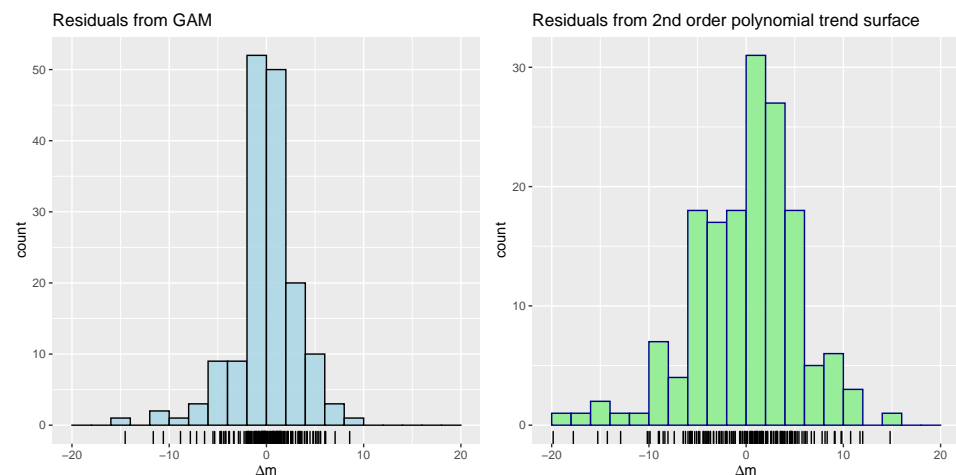
par(mfrow=c(1,2))
hist(resid.gam, xlim=c(-20, 20),
     breaks=seq(-20, 20, by=4), main="Residuals from GAM")
rug(residuals(model.gam))
hist(residuals(model.ts2), xlim=c(-20, 20),
     breaks=seq(-20, 20, by=4), main="Residuals from 2nd-order OLS trend")
rug(residuals(model.ts2))
par(mfrow=c(1,1))
```



We can also do histograms with `ggplot2`.

**Note:** Base R has 657 named colours, which you can see with the `colors` or `colours` functions. Their hexadecimal codes can then be found with the `col2rgb` function.

```
g1 <- ggplot(data=as.data.frame(resid.gam), aes(resid.gam)) +
  geom_histogram(breaks=seq(-20,20,by=2),
    fill="lightblue", color="black", alpha=0.9) +
  geom_rug() +
  labs(title = "Residuals from GAM",
    x = expression(paste(Delta, m)))
g2 <- ggplot(data=as.data.frame(residuals(model.ts2)), aes(residuals(model.ts2))) +
  geom_histogram(breaks=seq(-20,20,by=2),
    fill="lightgreen", color="darkblue", alpha=0.9) +
  geom_rug() +
  labs(title = "Residuals from 2nd order polynomial trend surface",
    x = expression(paste(Delta, m)))
grid.arrange(g1, g2, nrow=1)
```




---

**Q23 :** Which histogram shows the narrowest spread? [Jump to A23](#) •

An important consideration with GAM, as well as with linear models (see previous sections) is whether the residuals have any spatial structure.

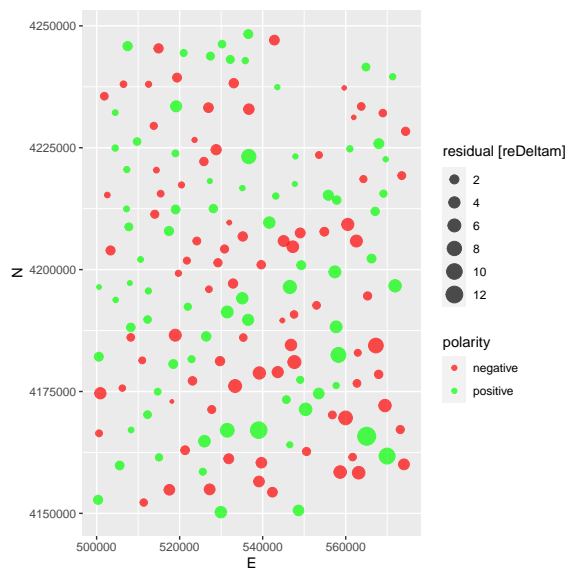
---

**TASK 35 :** Plot the residuals as a bubble plot. This shows the size of the residuals by the size of a point, and the polarity (positive vs. negative) by colour.

```

aq$resid.gam <- resid.gam
ggplot(data = drop_units(aq)) +
  aes(x=E, y=N, size = abs(resid.gam),
      col = ifelse((resid.gam < 0), "red", "green")) +
  geom_point(alpha=0.7) +
  scale_size_continuous(name = expression(paste(plain("residual ["),
                                                    reDelta, m, plain("]"))),
                        breaks=seq(0,12, by=2)) +
  scale_color_manual(name = "polarity",
                    labels = c("negative", "positive"),
                    values = c("red", "green", "blue"))

```




---

**Q24 :** Does there appear to be any local spatial correlation of the residuals?

[Jump to A24](#)

---

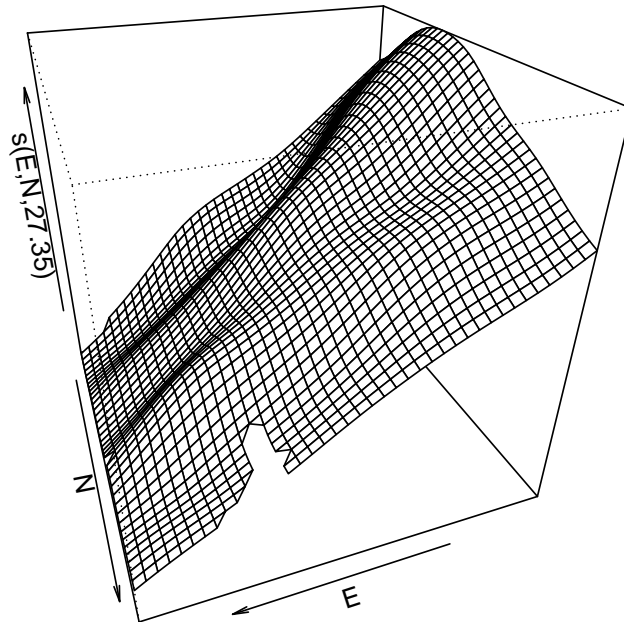
**TASK 36 :** Display the fit as a 3D surface.

The `plot.gam` function of the `mgcv` package displays the marginal smooth fit. For the 2D surface (model term  $s(E, N)$ ), this is shown as a wireframe plot if the optional `scheme` argument is set to 1. The `select` argument selects which model term to display. We orient it to see lowest elevation towards viewer, using the `theta` argument:

```

plot.gam(model.gam, rug = TRUE, se = TRUE, select=1,
         scheme=1, theta=30+130, phi=30)

```

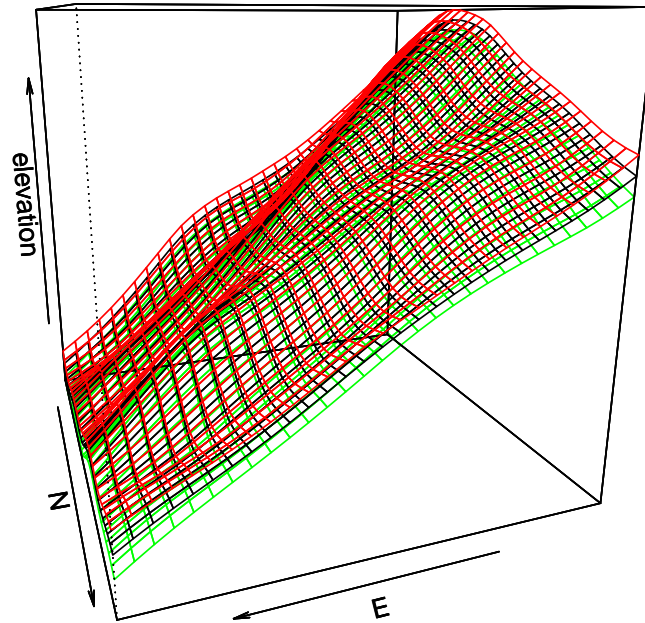



---

**Q25 :** *How does the GAM trend differ from a polynomial trend surface?*  
[Jump to A25 •](#)

This surface can also be shown with the `vis.gam` function of the `mgcv` package, also showing  $\pm 1$  standard error of fit:

```
vis.gam(model.gam, plot.type="persp", color="terrain",
        theta=160, zlab="elevation", se=1.96)
```



red/green are  $\pm 1.96$  s.e.

The fit is very good, the standard error is quite small. This is because the GAM adjusts to local deviations from the overall trend.

---

**TASK 37 :** Compute the RMSE of the GAM model, i.e., compared to fits with the actual values. •

```
(rmse.gam <- sqrt(sum(residuals(model.gam)^2)/length(residuals(model.gam))))  
## [1] 3.241328
```

The RMSE is 3.24; this is a very small error.

## 7.2 GAM prediction over the study area

Since we now have a model which uses the coördinates, which are known across the prediction grid, we can use the model to predict over the grid.

---

**TASK 38 :** Predict the aquifer elevation, and the standard error of prediction, across the prediction grid, using the fitted GAM, and display the predictions. •

The `predict.gam` function predicts from a fitted GAM. The `se.fit` op-

tional argument specifies that the standard error of prediction should also be computed. Here we use the `data.frame` form of the spatial grid. We predict onto this grid into a temporary object, because it will have two fields (columns): the prediction and its standard error.

```
tmp <- predict.gam(object=model.gam,
                  newdata=grid1km.df,
                  se.fit=TRUE)

summary(tmp$fit)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    480.0   517.4   547.2   547.4   576.4   621.4

summary(tmp$se.fit)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.181   1.321   1.404   1.521   1.611   3.607

str(tmp)

## List of 2
## $ fit      : num [1:7425(1d)] 579 578 577 576 575 ...
## .. attr(*, "dimnames")=List of 1
## .. ..$ : chr [1:7425] "1" "2" "3" "4" ...
## $ se.fit: num [1:7425(1d)] 3.07 2.99 2.92 2.85 2.78 ...
## .. attr(*, "dimnames")=List of 1
## .. ..$ : chr [1:7425] "1" "2" "3" "4" ...
```

We then add these to the `data.frame` version of the spatial grid.

```
grid1km.df$pred.gam <- as.numeric(tmp$fit)
grid1km.df$pred.gam.se <- as.numeric(tmp$se.fit)
```

**Note:** In the above code, the `as.numeric` function is needed to convert the 1-d array returned by `predict.gam` to a vector, then stored as one field (column) in the data frame.

---

### TASK 39 : Display the map of the predicted surface. •

We make a new raster with this information and display it. The default palette for terra maps is the “terrain” palette, from white through yellow to green.

```
grid1km.gam <- grid1km
values(grid1km.gam) <- grid1km.df$pred.gam
plot(grid1km.gam, main="GAM prediction"); grid()
points(st_coordinates(aq.sf)[,2] ~ st_coordinates(aq.sf)[,1], pch=16,
       col=ifelse(aq$resid.gam < 0, "red", "green"),
       cex=2*abs(aq$resid.gam)/max(abs(aq$resid.gam)))
```

twidhtwidth

As with the OLS and GLS maps, in this plot the *residual* from the model at each observation point is shown (1) in colour: red = negative (actual < predicted), green = positive (actual > predicted). If a prediction is exactly on the trend surface it will not appear. This gives a nice visualization of the fit of the trend surface to the sample points.

---

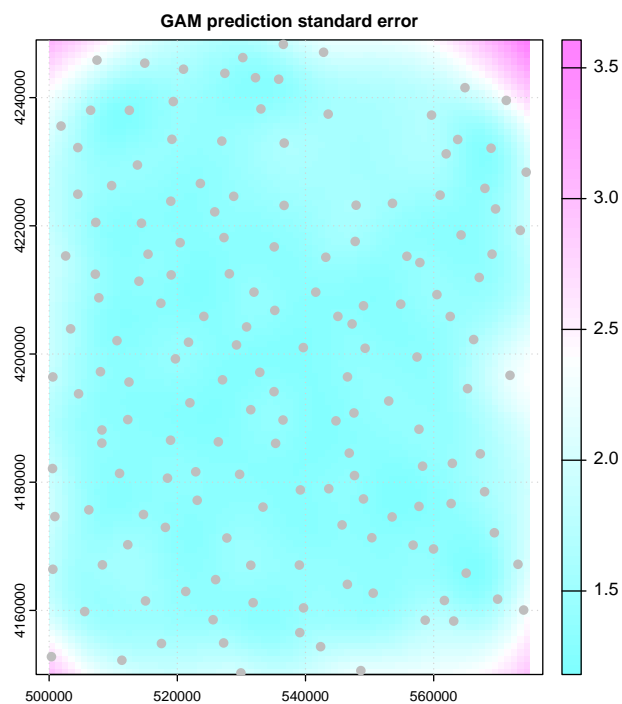
**Q26 :** *How well does the GAM trend surface fit the points? Are there obvious problems?* Jump to A26 •

---

**TASK 40 :** Display the map of the standard errors of prediction. •

As for the OLS trend surface we use the cyan-magenta palette to represent uncertainty.

```
grid1km.gam.se <- grid1km
values(grid1km.gam.se) <- grid1km.df$pred.gam.se
plot(grid1km.gam.se, main="GAM prediction standard error",
     col=cm.colors(64))
grid()
points(st_coordinates(aq.sf)[,2] ~ st_coordinates(aq.sf)[,1], pch=16,
       col="grey")
```



Consistent with the marginal plots, we see that the standard error is highest at the edges, but there is some local pattern due to the local adjustments of the GAM.

An obvious question is where this map differs from the parametric trend surfaces.

---

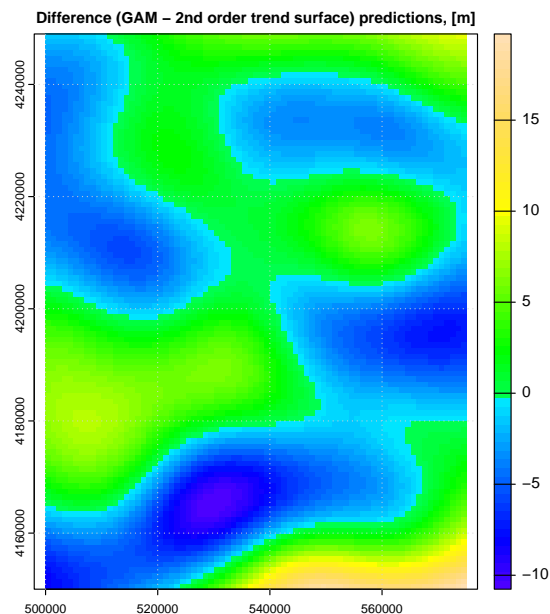
**TASK 41 :** Compute the differences between the trend surfaces produced by the GAM and the 2<sup>nd</sup>-order OLS trend surface predictions over the grid, summarize numerically, and display as a difference map. •

We choose yet another palette to represent differences, the “topography” palette from blue to yellow:

```
summary(grid1km.df$diff.gam.ols <-
        grid1km.df$pred.gam - grid1km.df$ts2.fit)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	-10.766013	-3.105651	-0.369673	0.004091	2.472404	19.735232

```
grid1km.diff.gam.ols <- grid1km
values(grid1km.diff.gam.ols) <- grid1km.df$diff.gam.ols
plot(grid1km.diff.gam.ols,
     main="Difference (GAM - 2nd order trend surface) predictions, [m]",
     col=topo.colors(64))
grid()
```



Here positive differences are where the GAM predicts higher values than the trend surface.

---

**Q27 :** *Where are the largest differences between the GAM and 2<sup>nd</sup> order OLS trend surface predictions? Explain why, considering how the two surfaces are computed.* Jump to A27 •

## 8 Thin-plate spline interpolation

A quick way to see the distribution of a variable in space as a surface is with an empirical method that adjusts locally to the data. A common empirical method is **thin-plate splines** (TPS), also referred to as “minimum curvature” surfaces, which are implemented in the `fields` package. The theory of thin-plate splines is explained in the Appendix, §C.

TPS is the mathematical equivalent of a thin (so, flexible) plate that is warped to fit the data. This can range from very “rigid”, i.e., just a single surface (the usual least-squares plane of a first-order trend surface) to very “flexible”, i.e., perfectly fitting every observation. In general we want something in between: if we think there is an overall surface we just fit it as one polynomial (first, second ... order polynomials on the coördinates), but if we want to fit more locally, we must expect local noise which should be somehow locally averaged-out.



This from ?Tps with some added emphasis; see that help text for much more detail.

A thin plate spline is the result of **minimizing** the residual sum of squares subject to a **constraint** that the function have a **certain level of smoothness** (or roughness penalty). Roughness is quantified by the integral of squared  $m$ -th order derivatives ... For two dimensions the roughness penalty is the integral of:

$$(D_{xx}(f))^2 + 2(D_{xy}(f))^2 + (D_{yy}(f))^2$$

---

**TASK 42 :** Set up for thin-plate splines and compute the minimum-curvature spline, subject to roughness constraint determined by generalized cross-validation. •

The Tps function of the fields package compute this; however the coordinates must be formatted as a matrix field in the dataframe, using the matrix function.

```
aq.tps <- aq[, c("E", "N", "zm")]
aq.tps$coords <- matrix(c(aq.tps$E, aq.tps$N), byrow=F, ncol=2)
str(aq.tps$coords)

## num [1:161, 1:2] 569464 573151 559974 553514 550350 ...
```

Now this matrix can be used to compute the spline surface:

```
surf.1 <- Tps(aq.tps$coords, aq.tps$zm)

## Warning:
## Grid searches over lambda (nugget and sill variances) with minima at the endpoints:
## (GCV) Generalized Cross-Validation
## minimum at right endpoint lambda = 8.53564e-06 (eff. df=
## 152.95 )

summary(surf.1)

## CALL:
## Tps(x = aq.tps$coords, Y = aq.tps$zm)
##
## Number of Observations:          161
## Number of unique points:         161
## Number of parameters in the null space 3
## Parameters for fixed spatial drift 3
## Effective degrees of freedom:      152.9
## Residual degrees of freedom:       8.1
## MLE tau                            0.6747
## GCV tau                            0.7098
## MLE sigma                          53330
## Scale passed for covariance (sigma) <NA>
## Scale passed for nugget (tau^2)    <NA>
## Smoothing parameter lambda         8.536e-06
##
## Residual Summary:
##      min      1st Q      median      3rd Q      max
## -0.6250000 -0.0701000 -0.0007448  0.0793500  0.5199000
##
## Covariance Model: Rad.cov
## Names of non-default covariance arguments:
##      p
```

```
##
## DETAILS ON SMOOTHING PARAMETER:
## Method used: GCV Cost: 1
## lambda trA GCV GCV.one GCV.model tauHat
## 8.536e-06 1.529e+02 1.008e+01 1.008e+01 NA 7.098e-01
##
## Summary of all estimates found for lambda
## lambda trA GCV tauHat -lnLike Prof converge
## GCV 8.536e-06 152.9 10.08 0.7098 462.5 NA
## GCV.model NA NA NA NA NA NA
## GCV.one 8.536e-06 152.9 10.08 0.7098 NA NA
## RMSE NA NA NA NA NA NA
## pure error NA NA NA NA NA NA
## REML 7.654e-05 117.7 11.24 1.7383 461.6 3
```

---

**TASK 43 :** Predict over the study area grid using the fitted thin-plate spline. •

The `predict.Krig` method of the `fields` package computes the prediction. Again, the coordinates must be a matrix. We have these in the grid, but we need to convert them to a matrix.

```
grid.coords.m <- as.matrix(grid1km.df[, c("E", "N")], ncol=2)
str(grid.coords.m)

## num [1:7425, 1:2] 500500 501500 502500 503500 504500 ...
## - attr(*, "dimnames")=List of 2
## ..$ : NULL
## ..$ : chr [1:2] "E" "N"
```

Predict, and add to the `data.frame` version of grid where we have the results from other methods.

```
surf.1.pred <- predict.Krig(surf.1, grid.coords.m)
summary(grid1km.df$pred.tps <- as.numeric(surf.1.pred))

## Min. 1st Qu. Median Mean 3rd Qu. Max.
## 475.3 517.1 546.8 547.4 576.3 623.6
```

**Note:** As in the results from the GAM we needed to convert the 1-d array returned by `predict.gam` to a vector before storing it as one field (column) in the data frame.

---

**TASK 44 :** Compute and summarize the residuals. •

The `predict.Krig` function does not compute residuals. We get the predictions from the prediction grid, with the `terra` function, at the locations of the known points. We then compute the difference of this predicted value from the actual value at that point, and add this as a field in the points spatial object.

```
grid1km.tps <- grid1km
values(grid1km.tps) <- surf.1.pred
tmp <- extract(grid1km.tps, st_coordinates(aq.sf))
names(tmp)

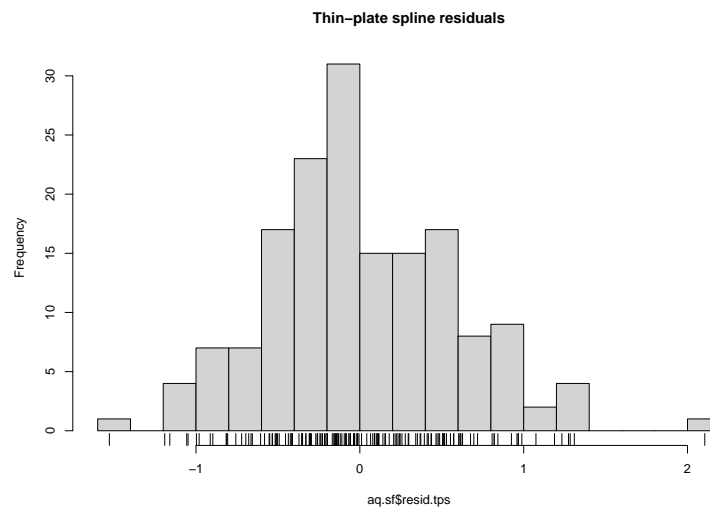
## [1] "z"

summary(aq.sf$resid.tps <- (drop_units(aq.sf$zm) - tmp$z))
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	-1.52882	-0.34992	-0.05734	0.01306	0.41104	2.10552

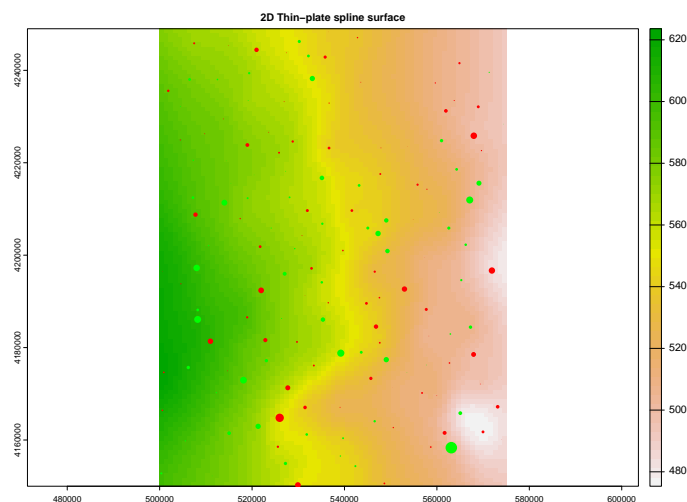
**TASK 45 :** Plot a histogram of these residuals:

```
hist(aq.sf$resid.tps, main="Thin-plate spline residuals", breaks=16)
rug(aq.sf$resid.tps)
```



**TASK 46 :** Display the gridded prediction, with the residuals over-printed.

```
plot(grid1km.tps,
     main = "2D Thin-plate spline surface")
points(st_coordinates(aq.sf)[,2] ~ st_coordinates(aq.sf)[,1], pch=16,
       col=ifelse(aq.sf$resid.tps < 0, "red", "green"),
       cex=2*abs(aq.sf$resid.tps)/max(abs(aq.sf$resid.tps)))
```



This adjusts very closely to the data points. Notice how many points hardly show at all, i.e., residual almost zero, and no residuals are very

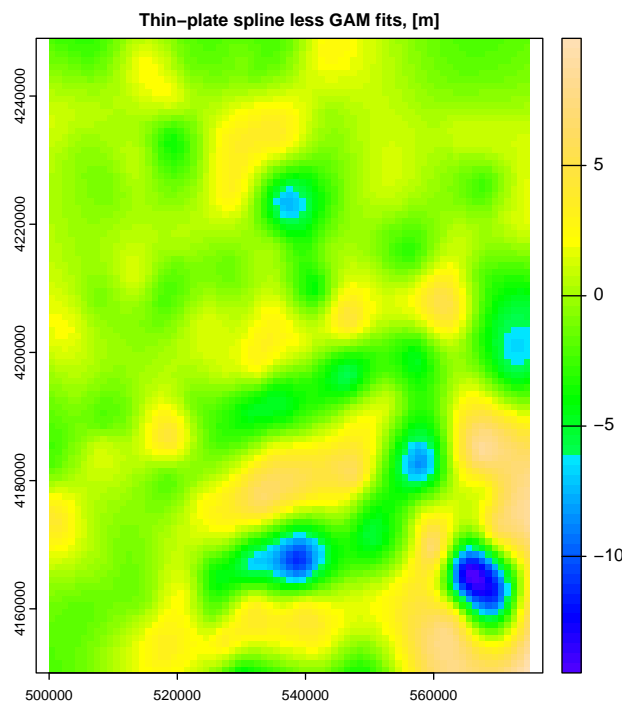
large.

---

**TASK 47 :** Compute and display the difference between the thin-plate spline and the GAM predictions. •

The differences are by direct subtraction of the gridded maps.

```
grid1km.diff.tps.gam <- grid1km
grid1km.diff.tps.gam <- grid1km.tps - grid1km.gam
values(grid1km.diff.tps.gam) <- (values(grid1km.tps) - values(grid1km.gam))
plot(grid1km.diff.tps.gam,
     col = topo.colors(64),
     main = "Thin-plate spline less GAM fits, [m]",
     xlab = "East", ylab = "North")
```



There are some differences, even though GAM allows some local deviations from a trend.

## 9 Spatial correlation of trend surface residuals

In trend surface analysis of §5.1 (1<sup>st</sup> order) and §5.2 (2<sup>nd</sup> order) we showed that the residuals from the OLS trend surface models are not spatially independent – there are local clusters of similar values as revealed by the bubble plot. This implies the trend surface should in fact be fit not by OLS but by Generalized Least Squares (GLS), taking into account the spatial auto-correlation of the residuals. We pursue this further in §10. Here we investigate the spatial structure of the residuals

The spatial structure of the **residuals** can be modelled with a **variogram**; this structure can then be used to adjust the trend surface with GLS

(§10.1, below). In this section we examine the empirical variogram of the residuals, and later use it to initialize the GLS estimate.

---

**TASK 48 :** Add the second-order trend-surface **predictions** and **residuals** as fields to the `aq` data frame. •

The `fitted` method extracts fitted values from a linear model object; the `residuals` method extracts the residuals.

```
aq$fit.ts2 <- fitted(model.ts2)
aq.sf$fit.ts2 <- fitted(model.ts2)
aq$res.ts2 <- residuals(model.ts2)
aq.sf$res.ts2 <- residuals(model.ts2)
```

## 9.1 The empirical variogram

To visualize the spatial autocorrelation of the trend surface residuals, we compute an **empirical variogram**, which show the relation between separation distance in **geographic space** between pairs of points and a measure of their separation distance in **attribute (feature) space**. This measure is called the **semivariance**:  $\gamma$  of one point-pair  $(\mathbf{s}_i, \mathbf{s}_j)$ :

$$\gamma(\mathbf{s}_i, \mathbf{s}_j) \equiv \frac{1}{2} [z(\mathbf{s}_i) - z(\mathbf{s}_j)]^2 \quad (9)$$

Because there a large number  $((n(n-1))/2)$  of point-pairs, the separation distances are usually grouped into ranges, and the average semivariance  $\bar{\gamma}(\mathbf{h})$  is computed as:

$$\bar{\gamma}(\mathbf{h}) = \frac{1}{2m(\mathbf{h})} \sum_{i=1}^{m(\mathbf{h})} [z(\mathbf{s}_i) - z(\mathbf{s}_i + \mathbf{h})]^2 \quad (10)$$

where:

- $m(\mathbf{h})$  is the number of point-pairs separated by vector  $\mathbf{h}$ , in practice some range of separations (“bin”);
- these are indexed by  $i$ ;
- the notation  $z(\mathbf{s}_i + \mathbf{h})$  means the “tail” of point-pair  $i$ , i.e., separated from the “head”  $\mathbf{s}_i$  by the separation vector  $\mathbf{h}$ .

---

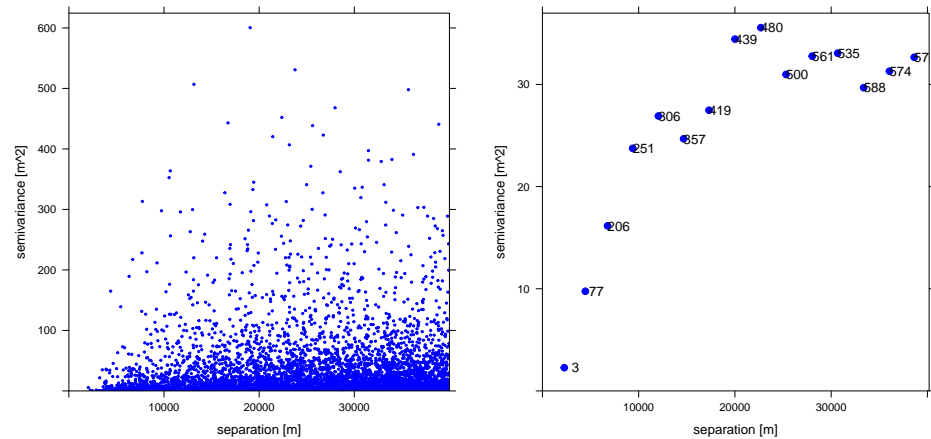
**TASK 49 :** Compute and plot the **empirical variogram** of the residuals from the second-order surface, with a cutoff of 40 km. •

The `variogram` function of the `gstat` package computes the empirical variogram. We show both the *variogram cloud* and the *summarized variogram*, which averages the points in the variogram cloud over some separation ranges; these are called **variogram bins**.

```

vr.c <- variogram(res.ts2 ~ 1, loc = aq.sf, cutoff = 40000, cloud = T)
vr <- variogram(res.ts2 ~ 1, loc = aq.sf, cutoff = 40000)
p1 <- plot(vr.c, col = "blue", pch = 20, cex = 0.5,
           xlab = "separation [m]", ylab = "semivariance [m^2]")
p2 <- plot(vr, plot.numbers = T, col = "blue", pch = 20, cex = 1.5,
           xlab = "separation [m]", ylab = "semivariance [m^2]")
print(p1, split = c(1,1,2,1), more = T)
print(p2, split = c(2,1,2,1), more = F)

```



**Note:** The code to print two variograms side-by-side uses the **split** and **more** optional arguments to the **print** method for Lattice graphics plots, which are the default for **gstat** variogram objects.

**Note:** Notice the large difference in the semivariance scale for the two kinds of variograms. This is because the summarized variogram averages within the bins, and the few very large individual semivariances are averaged out.

The empirical variogram can be characterized by three parameters; here they are just estimated by eye. These will be defined precisely as part of variogram modelling, see below.

- $c_0$  the **nugget** parameter: the semi-variance at zero separation; this represents a combination of measurement error, sampling error, and spatial variation at range shorter than the sample support;
- $c$  is the **sill** parameter, i.e., the maximum variance in the attribute when point-pairs are widely-separated;
- $a$  is the **range** parameter, the separation at which there is no more spatial autocorrelation, so that the semivariance reaches the sill.

---

**Q28 :** *What are the estimated sill, range, and nugget of this variogram?*  
[Jump to A28](#) •

In later sections (§11.2) we will see how to model the variogram and use it in spatial prediction. For now, we continue with a method that takes spatial correlation of the residuals into account when computing

the trend; the eyeball estimates of the spatial correlation parameters will be sufficient for this method.

## 10 Trend surface analysis by Generalized Least Squares

As explained in §5, the OLS solution is only valid for independent residuals. The previous § shows that in this case the residuals are *not* spatially independent, and we were able to model that dependence with a variogram model. Thus, using OLS may result in an incorrect trend surface equation, although the OLS estimate is unbiased. A large number of close-by points with similar values will “pull” a trend surface towards them. Furthermore, the OLS  $R^2$  (goodness-of-fit) may be over-optimistic. This is discussed by Fox [4, §14.1].

The solution is to use **Generalised Least Squares** (GLS) to estimate the trend surface. This allows a **covariance structure between residuals** to be included directly in the least-squares solution of the regression equation.

The GLS estimate of the regression coefficients is [2]:

$$\hat{\beta}_{\text{glS}} = (X^T C^{-1} X)^{-1} X^T C^{-1} \mathbf{y} \quad (11)$$

where  $X$  is the design matrix,  $C$  the **covariance matrix** of the (spatially-correlated) **residuals**, and  $\mathbf{y}$  the vector of **observations**. If there is no spatial dependence among the errors,  $C$  reduces to  $I\sigma^2$  and the estimate to OLS as in Equation 3.

The covariance matrix  $C$  gives the covariance between the residuals at each pair of points used to determine the  $\hat{\beta}_{\text{glS}}$ . Clearly, there is no way to know the covariance between all the point-pairs, since we only have one realization of the random field. So we model the covariance as a function of the **separation** (usually the distance) between point pairs, similar to what we did in §9.1, to fit a variogram model. However, we instead fit a **spatial covariance** model. This leads us to a further difficulty: the covariance structure refers to the residuals, but we can’t compute these until we fit the trend ...but we need the covariance structure to fit the trend ...and so on. This is a classic “which came first: the chicken or the egg?” problem.

One method to compute the GLS model is iterative:

1. make a first estimate of the trend surface with OLS;
2. compute the residuals;
3. model the covariance structure of the OLS residuals as a function of their separation;
4. use this covariance structure to determine the weights to compute the GLS trend surface;
5. repeat steps (2)–(4) until the covariance structure does not change between iterations.

In many cases only one iteration is necessary. However, theoretically this is not optimal, because the estimates of the covariance parameters are biased.

A more elegant solution is to fit the covariance structure at the same time the trend surface coefficients are computed. The theory and mathematics of this are explained in §D.

## 10.1 Computing the GLS trend surface

GLS trend surfaces can be computed in several R packages. The trend and the covariance must be computed at the same time. This is implemented in the `gls` function of the `nlme` package, using Residual Maximum Likelihood (REML)<sup>18</sup>.

---

**TASK 50 :** Compute the coefficients of a full second-order trend, using GLS. •

The `gls` function fits two models at once: the linear model, specified with the `model` argument, and the autocorrelation structure, specified with the `correlation` argument.

The `model` argument is the same model formula we use for OLS.

The `correlation` argument specifies the form of the correlation, and its initial parameters. Here we specify an exponential correlation with the `corExp` function. The form is two-dimensional in the coordinates, and from examination of the empirical variogram we see there is no nugget, so we specify `nugget` as `FALSE`. The `value` argument to `corExp` specifies starting values for this correlation structure, here the estimated range parameter. Recall that this is  $1/3$  of the separation at which there is no longer any spatial autocorrelation. From the empirical variogram (§9.1) we estimate this as  $30\,000/3 = 10\,000$  m.

```
model.ts2.gls <- gls(
  model = drop_units(zm) ~ N + E + I(N^2) + I(E^2) + I(E * N),
  data = aq,
  method = "ML",
  correlation = corExp(form = ~E + N,
    nugget = FALSE,
    value = 10000) # initial value of the range parameter
)
class(model.ts2.gls)

## [1] "gls"

summary(model.ts2.gls)

## Generalized least squares fit by maximum likelihood
## Model: drop_units(zm) ~ N + E + I(N^2) + I(E^2) + I(E * N)
## Data: aq
##      AIC      BIC    logLik
## 939.389 964.0403 -461.6945
##
## Correlation Structure: Exponential spatial correlation
## Formula: ~E + N
```

---

<sup>18</sup> See §D.3 for the theory and mathematics of REML.



```
## Parameter estimate(s):
## range
## 14421.58
##
## Coefficients:
##          Value Std.Error   t-value p-value
## (Intercept) -84955.46 30058.379 -2.826349  0.0053
## N            0.04     0.014  3.059245  0.0026
## E           -0.02     0.008 -2.282998  0.0238
## I(N^2)        0.00     0.000 -3.236521  0.0015
## I(E^2)        0.00     0.000 -0.511079  0.6100
## I(E * N)      0.00     0.000  2.325550  0.0213
##
## Correlation:
##          (Intr) N      E      I(N^2) I(E^2)
## N          -0.997
## E          -0.149  0.075
## I(N^2)       0.989 -0.997 -0.005
## I(E^2)       0.111 -0.094 -0.248  0.099
## I(E * N)     0.118 -0.047 -0.950 -0.026 -0.066
##
## Standardized residuals:
##          Min      Q1      Med      Q3      Max
## -3.4096923 -0.5485337  0.1276973  0.6513997  2.0752493
##
## Residual standard error: 6.385513
## Degrees of freedom: 161 total; 155 residual
```

Notice that the `gls` method also estimates the range of spatial correlation.

---

**Q29 :** *What is the range of spatial correlation of the exponential model, as estimated by `gls`?* [Jump to A29](#) •

This gives different coefficients than the OLS fit.

---

**TASK 51 :** Compare the coefficients from the GLS and OLS fits, as absolute differences and as percentages of the OLS fit. •

The generic `coef` method extracts coefficients from model objects.

```
coef(model.ts2.gls) - coef(model.ts2)

## (Intercept)          N          E          I(N^2)          I(E^2)
## 3.113971e+04 7.142363e-02 -7.859060e-02 2.031070e-09 3.950790e-10
## I(E * N)
## -2.170533e-09

round(100*(coef(model.ts2.gls) - coef(model.ts2))
      /coef(model.ts2),1)

## (Intercept)          N          E          I(N^2)          I(E^2)
## -26.8        -255.2        -132.4        -27.1        -24.0
## I(E * N)
## -32.4
```

---

**Q30 :** *Why are the GLS coefficients different than the OLS coefficients?* [Jump to A30](#) •

---

**TASK 52 :** Display the 90% confidence intervals for the GLS model parameters. •

The generic `intervals` method has a specific method for a fitted GLS model; internally this is the `intervals.gls` function of the `nlme` package.

```
intervals(model.ts2.gls, level=0.90)

## Approximate 90% confidence intervals
##
## Coefficients:
##           lower      est.      upper
## (Intercept) -1.346944e+05 -8.495546e+04 -3.521654e+04
## N           1.994295e-02  4.343916e-02  6.693538e-02
## E           -3.315918e-02 -1.922481e-02 -5.290439e-03
## I(N^2)       -8.264555e-09 -5.468607e-09 -2.672659e-09
## I(E^2)       -5.307723e-09 -1.252487e-09  2.802750e-09
## I(E * N)     1.306509e-09  4.529404e-09  7.752298e-09
##
## Correlation structure:
##           lower      est.      upper
## range 8129.732 14421.58 25582.87
##
## Residual standard error:
##           lower      est.      upper
## 5.003777  6.385513  8.148799
```

---

**TASK 53 :** Compute the residuals from this surface, and compare them to those from the OLS surface. •

```
summary(res.ts2.gls <- residuals(model.ts2.gls))

##      Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
## -21.77263  -3.50267   0.81541  -0.05721   4.15952  13.25153

summary(res.ts2)

##      Min. 1st Qu.  Median     Mean 3rd Qu.    Max.
## -19.847  -3.366   0.822   0.000   3.538   14.807
```

---

**Q31 :** *What are the main differences between these sets of residuals? Which surface, in this case, most closely fits the points? Why?* [Jump to A31](#) •

## 10.2 Predicting from the GLS trend surface

Now that we have a trend surface model based on the known points, we can apply this model to create a full surface.

---

**TASK 54 :** Predict over the grid with the GLS trend. •

The `predict` generic method has a specific method for a fitted GLS model; internally this is the `predict.gls` function of the `nlme` package.

```
pred.ts2.gls <- predict(model.ts2.gls, newdata=grid1km.df)
summary(pred.ts2.gls)
```

##	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	475.2	518.9	548.1	547.7	576.4	610.7

---

**TASK 55 :** Make a `SpatRast` grid with the model predictions (fits) and summarize them. •

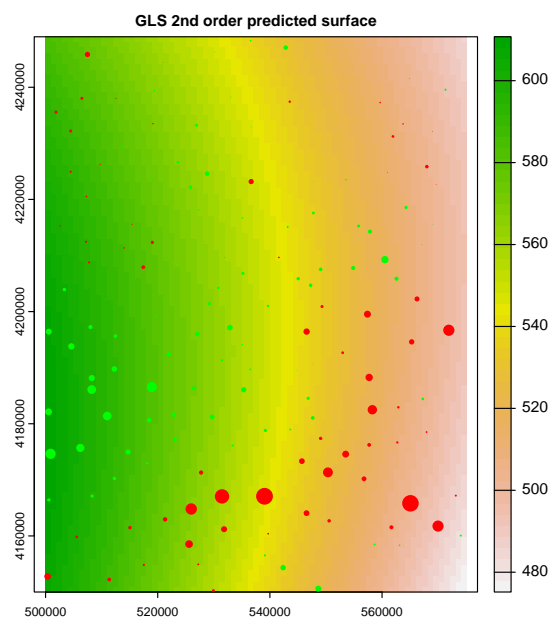
```
grid1km.gls <- grid1km
values(grid1km.gls) <- pred.ts2.gls
summary(values(grid1km.gls))
```

```
##           z
## Min.      :475.2
## 1st Qu.:518.9
## Median :548.1
## Mean   :547.7
## 3rd Qu.:576.4
## Max.    :610.7
```

---

**TASK 56 :** Display the interpolated surface, with the residuals at the data points superimposed. •

```
plot(grid1km.gls, main = "GLS 2nd order predicted surface")
points(st_coordinates(aq.sf)[,2] ~ st_coordinates(aq.sf)[,1], pch=16,
       col = ifelse((res.ts2.gls < 0), "red", "green"),
       cex=2*abs(res.ts2.gls)/max(abs(res.ts2.gls)))
)
```

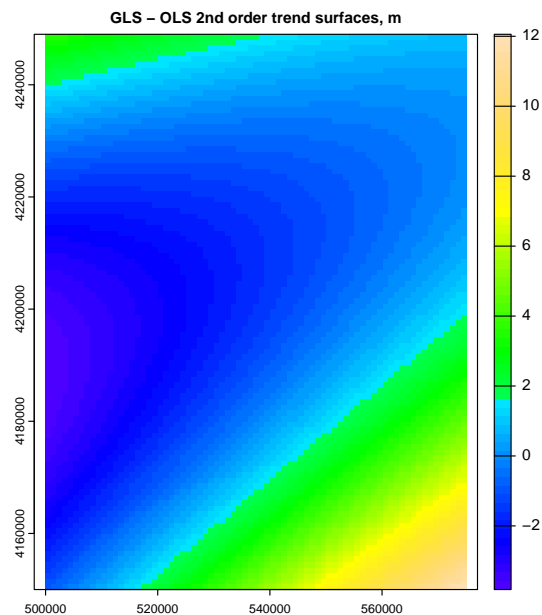


This looks quite similar to the OLS <sup>nd</sup>-order trend surface. In a dataset like this one with a good spread of points there is likely to be little difference. An obvious question is how much these differ, and where.

---

**TASK 57 :** Compute the difference between the OLS and GLS trend surfaces, and map this difference. •

```
grid1km.gls.ols.diff <- (grid1km.gls - grid1km)
plot(grid1km.gls.ols.diff, col = topo.colors(64),
     main = "GLS - OLS 2nd order trend surfaces, m")
```




---

**Q32 :** *Where are the largest differences between the OLS and GLS trend surfaces? Explain why.* Jump to A32 •

## 11 Local interpolation of the residuals

The trend surface fits an overall trend, but of course does not fit every observation exactly. The GLS fit, while correcting the trend surface parameters for local spatial dependence of the residuals, does not remove the local deviations from a surface expressed by one equation. This lack of fit can be pure noise, but it can also have a spatially-correlated component which can be modelled and used to improve the predictions.

### 11.1 Visualizing the residuals

---

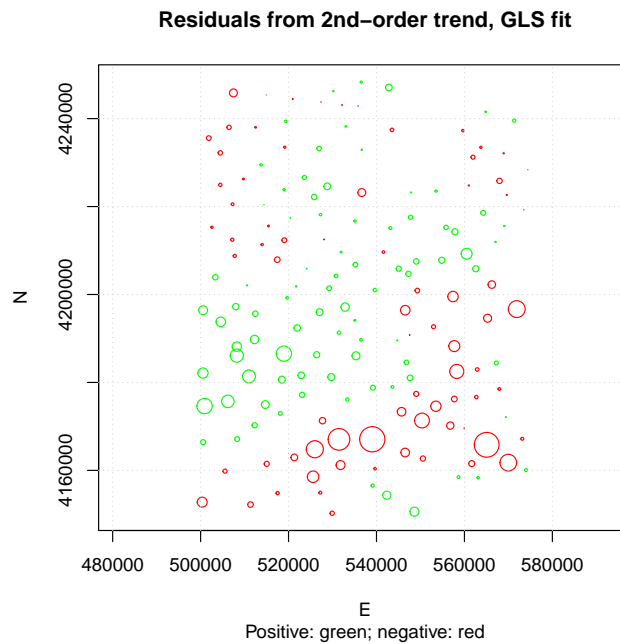
**TASK 58 :** Display the residuals from the GLS trend surface as a post-plot. •

```
summary(res.ts2.gls)

##      Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
## -21.77263 -3.50267  0.81541  -0.05721  4.15952  13.25153

plot(aq$N ~ aq$E, cex=3*abs(res.ts2.gls)/max(abs(res.ts2.gls)),
     col=ifelse(res.ts2.gls > 0, "green", "red"),
     xlab="E", ylab="N",
     main="Residuals from 2nd-order trend, GLS fit",
     sub="Positive: green; negative: red", asp=1)
```

`grid()`



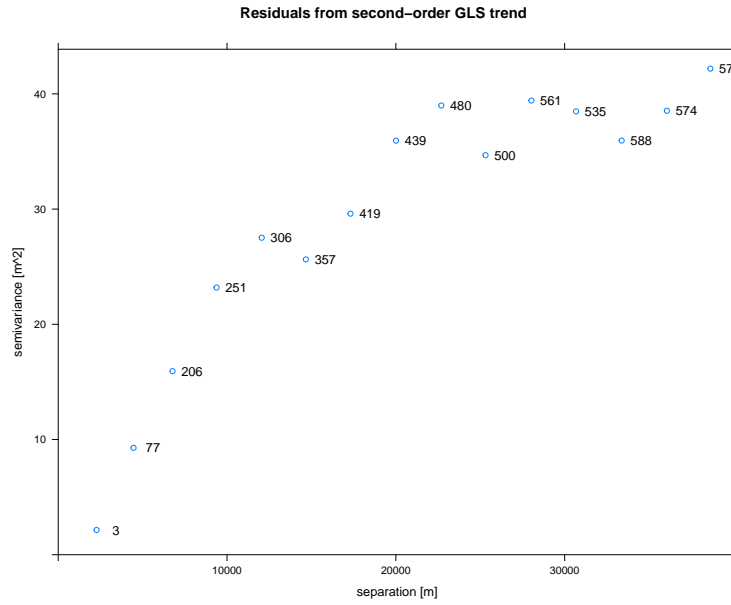
We can see from this post-plot of the residuals that there is local spatial correlation. The GLS fit optimized the estimates of the trend surface coefficients, and correctly estimated the spatial correlation of the residuals, but did not correct for this in mapping.

---

**TASK 59 :** Compute the empirical variogram model residuals from the GLS trend surface model. •

First extract the residuals into the point observations object, compute the empirical variogram, and display it to estimate the variogram model parameters.

```
aq.sf$res.ts2.gls <- residuals(model.ts2.gls)
vr.gls <- variogram(res.ts2.gls ~ 1, loc=aq.sf,
                    cutoff = 40000)
plot(vr.gls, plot.numbers=T,
     main="Residuals from second-order GLS trend",
     xlab = "separation [m]", ylab = "semivariance [m^2]")
```




---

Q33 : What are the approximate variogram parameters? [Jump to A33](#) •

## 11.2 Variogram modelling

In the kriging formula (see below, §11.3), we need to compute the semi-variance at *any* separation distance. Therefore, we need to fit a **variogram function** to the empirical variogram. This function represents the structure of the spatial autocorrelation of the attribute, in this case the trend surface residual.

There are many **authorized** variogram functions that will ensure that the kriging system can be solved. One of the most common is the **exponential** function:

$$\gamma(h) = c \left( 1 - e^{\left(-\frac{h}{a}\right)} \right) \quad (12)$$

Where:

- $h$  is the separation distance between a point-pair; this is the argument to the function which changes with each point-pair;
- $c$  is the fitted **sill** parameter, i.e., the maximum variance in the attribute when point-pairs are widely-separated'
- $a$  is the fitted **range** parameter.

The **effective range**  $3a$  is the separation distance at which  $\gamma = 0.95c$ .

In this case the **shape** of the empirical variogram suggests that an **exponential** model would fit well.

---

**TASK 60 :** Fit an exponential variogram function and display the fitted model on the empirical variogram. •

To fit a variogram function to an empirical variogram, one method is to estimate the parameters by eye, and adjust them until they seem to match the empirical variogram. A more objective way is to use the initial estimates as a starting point for the `fit.variogram` function, which adjusts the parameters by weighted least-squares.

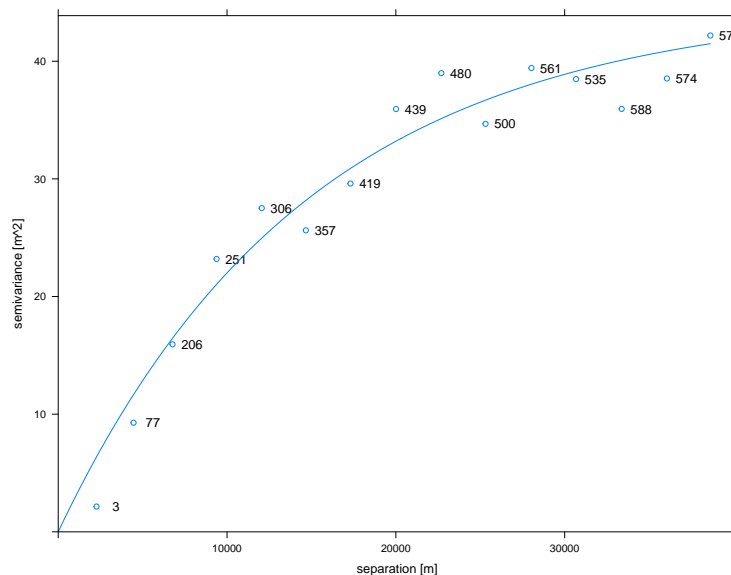
**Note:** A commonly-used empirical weighting is proportional to the number of point-pairs in a bin (giving more weight to bins with more evidence) and inversely proportional to the average separation in the bin (giving more weight to the close-range portion of the variogram, where most of the kriging weights are determined).

The `vgm` function specifies a variogram model and its parameters. We then adjust it with `fit.variogram`. Recall, we estimated the parameters by eye in the previous answer. We must divide our estimate of the range (see previous question) by 3 to obtain the  $a$  parameter of the exponential model.

```
vr.gls.m <- vgm(psill=40, model="Exp", range=22000/3, nugget=0)
(vr.gls.m.f <- fit.variogram(vr.gls, vr.gls.m))

##      model      psill      range
## 1      Nug 0.00000 0.00
## 2      Exp 44.76969 14784.35

plot(vr.gls, model=vr.gls.m.f, plot.numbers=T,
     xlab = "separation [m]", ylab = "semivariance [m^2]")
```



---

**TASK 61 :** Compare the range parameter of this fitted variogram with the range parameter estimate from the GLS fit. •

```
print(vr.gls.m.f)

##      model      psill      range
## 1   Nug  0.00000      0.00
## 2   Exp 44.76969 14784.35

intervals(model.ts2.gls)$corStruct[2]

## [1] 14421.58
```

---

**Q34 :** Does the range parameter of this fitted model agree with the estimate from the GLS fit? Jump to A34 •

### 11.3 The Ordinary Kriging system

Once we know the structure of the residuals, their values, and their locations, we can predict their values at all locations (e.g., over the grid), by Ordinary Kriging interpolation.

To do this, we first need to understand OK.

Kriging is a form of linear prediction of the attribute value at an unknown point  $\hat{z}(\mathbf{s}_0)$ , as a weighted sum of the attribute values at the known points  $z(\mathbf{s}_i)$ :

$$\hat{z}(\mathbf{s}_0) = \sum_{i=1}^N \lambda_i z(\mathbf{s}_i) \quad (13)$$

The weights  $\lambda_i$  must sum to 1, and are determined by solving the **kriging system** of equations. This system ensures that the prediction at each point has the least possible **prediction variance**, i.e., uncertainty, among all the predictions made with linear weights. Therefore OK is called the **Best Linear Unbiased Predictor** (BLUP).

Here we do not derive the system, but present it. The weights are the solution of the linear equation  $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b}$  where:

$$\mathbf{A} = \begin{bmatrix} \gamma(\mathbf{s}_1, \mathbf{s}_1) & \gamma(\mathbf{s}_1, \mathbf{s}_2) & \cdots & \gamma(\mathbf{s}_1, \mathbf{s}_N) & 1 \\ \gamma(\mathbf{s}_2, \mathbf{s}_1) & \gamma(\mathbf{s}_2, \mathbf{s}_2) & \cdots & \gamma(\mathbf{s}_2, \mathbf{s}_N) & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \gamma(\mathbf{s}_N, \mathbf{s}_1) & \gamma(\mathbf{s}_N, \mathbf{s}_2) & \cdots & \gamma(\mathbf{s}_N, \mathbf{s}_N) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \\ \psi \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \gamma(\mathbf{s}_1, \mathbf{s}_0) \\ \gamma(\mathbf{s}_2, \mathbf{s}_0) \\ \vdots \\ \gamma(\mathbf{s}_N, \mathbf{s}_0) \\ 1 \end{bmatrix}$$



All of the semivariances  $\gamma$  in these formulas are computed from the fitted variogram model, by substituting the separation between the known points  $(\mathbf{s}_i, \mathbf{s}_j)$  in the  $\mathbf{A}$  matrix, and between the known points and the prediction point  $(\mathbf{s}_i, \mathbf{s}_0)$  in the  $\mathbf{b}$  vector.

The kriging system is solved by matrix inversion and multiplication as:

$$\boldsymbol{\lambda} = \mathbf{A}^{-1}\mathbf{b}$$

These weights can then be used in the prediction formula Equation 13. They also can be used to compute the prediction variance as  $\hat{\sigma}^2 = \mathbf{b}^Y \boldsymbol{\lambda}$ .

## 11.4 OK predictions

---

**TASK 62 :** Predict over the grid by OK, using the fitted variogram model.

The `krige` function computes kriging predictions and their variances. This uses the location (and implicitly the data) of the known points (argument `loc`), a set of locations to be predicted (argument `newdata`, and of course the fitted variogram model (argument `model`).

The `krige` function does not understand spatial objects of class `SpatRast`. These must be converted to `sf` before use. We already have the point dataset in this format, but we now need to convert the prediction grid before using it as the new data points. We do this with the `st_as_sf` “convert to Simple Features” method of the `sf` package.

```
grid1km.sf <- st_as_sf(grid1km.df, coords = c("E", "N"))
st_crs(grid1km.sf) <- st_crs(grid1km)
kr <- krige(res.ts2.gls ~ 1,
            loc = aq.sf,
            newdata = grid1km.sf,
            model=vr.gls.m.f)

## [using ordinary kriging]

summary(kr)

##      var1.pred      var1.var      geometry
## Min.   :-20.50210  Min.    : 0.4129  POINT      :7425
## 1st Qu.: -3.14297  1st Qu.: 7.7124  epsg:26914   : 0
## Median : -0.04301  Median : 9.9055  +proj=utm ...: 0
## Mean    : -0.37462  Mean    :10.4527
## 3rd Qu.:  2.86436  3rd Qu.:12.0501
## Max.    : 12.78381  Max.    :31.9149

class(kr)

## [1] "sf"          "data.frame"
```

Kriging results in two fields: the predictions themselves, and their prediction variances – this is a byproduct of the design of kriging to minimize this variance, thus it must be computed as part of the process.

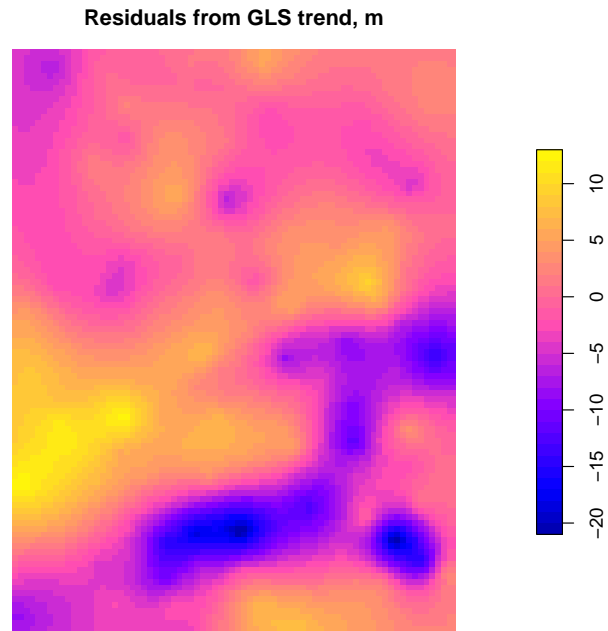
**Note:** Notice that the mean kriging prediction is not zero.

---

**TASK 63 :** Display the kriging predictions. •

Predictions:

```
plot(kr["var1.pred"], pch=15, nbreaks=24,  
     main="Residuals from GLS trend, m")
```



---

**Q35 :** Which areas were most changed by interpolating the residuals?  
Why? Jump to A35 •

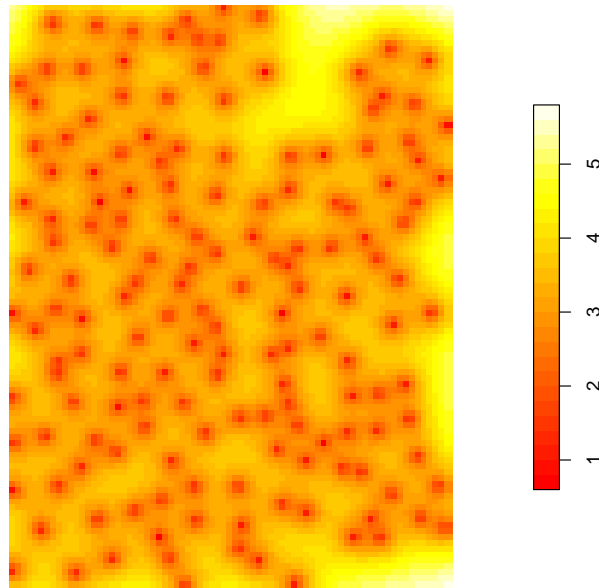
---

**TASK 64 :** Display the kriging prediction standard deviations. •

Prediction standard deviations (i.e., the square root of the variances, so they are on the same scale as the predictions):

```
kr$var1.sd <- sqrt(kr$var1.var)  
summary(kr)  
  
##      var1.pred      var1.var      geometry  
## Min.   :-20.50210  Min.    : 0.4129  POINT      :7425  
## 1st Qu.: -3.14297  1st Qu.: 7.7124  epsg:26914  : 0  
## Median : -0.04301  Median : 9.9055  +proj=utm ...: 0  
## Mean   : -0.37462  Mean    :10.4527  
## 3rd Qu.:  2.86436  3rd Qu.:12.0501  
## Max.    :12.78381  Max.     :31.9149  
##      var1.sd  
## Min.     :0.6425  
## 1st Qu.: 2.7771  
## Median   :3.1473  
## Mean     :3.1609  
## 3rd Qu.: 3.4713  
## Max.     :5.6493  
  
plot(kr["var1.sd"], pch=15, nbreaks=24, pal = heat.colors,  
     main="Standard errors of residuals from GLS trend, m")
```

Standard errors of residuals from GLS trend, m



---

Q36 : Which areas have the most and least uncertainty? Why? [Jump to A36](#) •

## 12 GLS-Regression Kriging

Now we have both parts of a universal model: a global trend and local deviations from it. We can combine these for a “best” prediction, taking into account both causes of spatial variation.

To understand this, we introduce the so-called **universal model of spatial variation**:

$$Z(\mathbf{s}) = Z^*(\mathbf{s}) + \varepsilon(\mathbf{s}) + \varepsilon'(\mathbf{s}) \quad (14)$$

where:

- $(\mathbf{s})$  is a location in space, designated by a **vector** of coördinates;
- $Z(\mathbf{s})$  is the **true** (unknown) value of some property at the location;
- $Z^*(\mathbf{s})$  is the **deterministic** component, due to some **non-stochastic** process, i.e., the trend surface;
- $\varepsilon(\mathbf{s})$  is the **spatially-autocorrelated stochastic** component of the deviations from the trend;
- $\varepsilon'(\mathbf{s})$  is the pure (“white”) **noise** with no structure; this can not be modelled.

We have seen how to model  $Z^*(\mathbf{s}) + \varepsilon'(\mathbf{s})$  with an OLS polynomial trend surface in §5. We have seen how to model the local spatial structure as

$\varepsilon(\mathbf{s}) + \varepsilon'(\mathbf{s})$  by Ordinary Kriging (OK) in §11. Both are modelled together in GLS (§10), but that trend surface still leaves residuals  $\varepsilon(\mathbf{s}) + \varepsilon'(\mathbf{s})$  to be accounted for.

When these two are combined, the method is called **Generalized Least Squares Trend – Regression Kriging** (GLS-RK).

---

**TASK 65 :** Add the OK predictions of the GLS residuals to the data frame version of prediction grid object, and then add this to the GLS trend surface prediction to obtain a final prediction. Make a SpatRast version for plotting. •

The kriging prediction object was built from the spatial grid, so it has the same dimensions.

```
grid1km.df$kr <- kr$var1.pred
grid1km.df$pred.ts2.gls <- pred.ts2.gls
grid1km.df$rkgl <- grid1km.df$pred.ts2.gls + grid1km.df$kr
summary(grid1km.df)
```

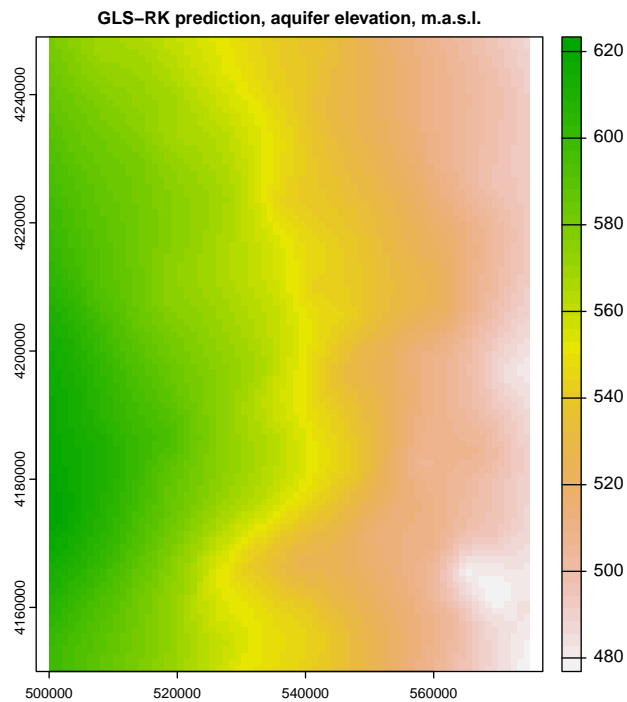
##	E	N	ts2.fit	ts2.lwr
##	Min. :500500	Min. :4150500	Min. :463.1	Min. :451.1
##	1st Qu.:518500	1st Qu.:4174500	1st Qu.:517.5	1st Qu.:506.3
##	Median :537500	Median :4199500	Median :547.9	Median :536.7
##	Mean :537500	Mean :4199500	Mean :547.4	Mean :536.1
##	3rd Qu.:556500	3rd Qu.:4224500	3rd Qu.:577.3	3rd Qu.:566.0
##	Max. :574500	Max. :4248500	Max. :614.4	Max. :603.0
##	ts2.upr	ts2.diff.range	pred.gam	pred.gam.se
##	Min. :475.2	Min. :22.30	Min. :480.0	Min. :1.181
##	1st Qu.:528.8	1st Qu.:22.33	1st Qu.:517.4	1st Qu.:1.321
##	Median :559.2	Median :22.45	Median :547.2	Median :1.404
##	Mean :558.7	Mean :22.56	Mean :547.4	Mean :1.521
##	3rd Qu.:588.6	3rd Qu.:22.71	3rd Qu.:576.4	3rd Qu.:1.611
##	Max. :625.8	Max. :24.21	Max. :621.4	Max. :3.607
##	diff.gam.ols	pred.tps	kr	
##	Min. :-10.766013	Min. :475.3	Min. :-20.50210	
##	1st Qu.: -3.105651	1st Qu.:517.1	1st Qu.: -3.14297	
##	Median : -0.369673	Median :546.8	Median : -0.04301	
##	Mean : 0.004091	Mean :547.4	Mean : -0.37462	
##	3rd Qu.: 2.472404	3rd Qu.:576.3	3rd Qu.: 2.86436	
##	Max. : 19.735232	Max. :623.6	Max. : 12.78381	
##	pred.ts2.gls	rkgl		
##	Min. :475.2	Min. :476.9		
##	1st Qu.:518.9	1st Qu.:516.9		
##	Median :548.1	Median :547.0		
##	Mean :547.7	Mean :547.3		
##	3rd Qu.:576.4	3rd Qu.:576.3		
##	Max. :610.7	Max. :623.4		

```
grid1km.rkgl <- grid1km
values(grid1km.rkgl) <- grid1km.df$rkgl
```

---

**TASK 66 :** Plot the final prediction. •

```
plot(grid1km.rkgl,
      main="GLS-RK prediction, aquifer elevation, m.a.s.l.")
```



The GAM prediction (§7.2) also considered both global and local components of the spatial variation. An obvious question is how similar they are.

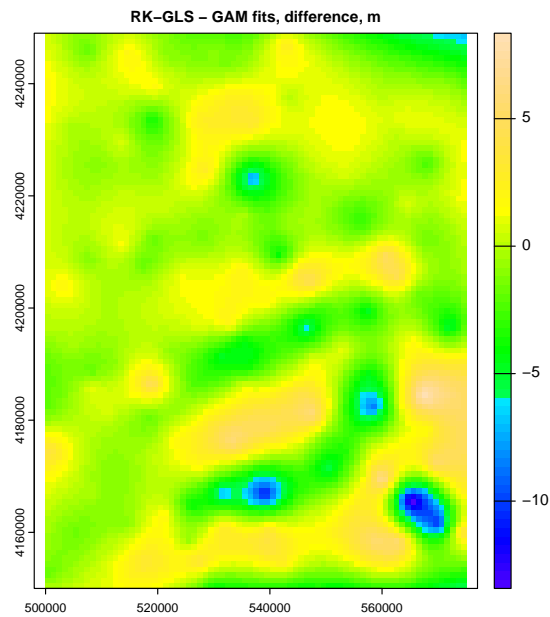
---

**TASK 67 :** Compare this predicted surface with the GAM prediction. •

```
summary(grid1km.rkgls.gam <- (grid1km.rkgls - grid1km.gam))

##           z
## Min.      :-13.426032
## 1st Qu.:  -1.056207
## Median :  -0.005607
## Mean      :-0.074158
## 3rd Qu.:   1.112371
## Max.      :   8.353024

plot(grid1km.rkgls.gam, main = "RK-GLS - GAM fits, difference, m",
     col = topo.colors(64))
```




---

**Q37 :** *Where are the largest differences between these two trend surface predictions? Explain why, considering how the two surfaces are computed.* Jump to A37 •

### 13 Universal Kriging (UK)

We showed in §9 that the residuals from the OLS fit are not spatially independent. We used this fact in §10 to produce a correct trend surface by GLS. We then modelled the spatial structure of the residuals from the GLS (not OLS) surface and interpolated these, to make a final map of both the trend and local variations with GLS-RK (§11).

There is another method of fitting the trend and the local deviations from it in one step, called “Universal Kriging”, abbreviated as UK. This is not completely correct theoretically, as we will explain, but if observation points are well-distributed over the area (as is the case in this exercise) so that the GLS and OLS trend surfaces are not too different, it provides a very similar map to GLS-RK, and in one step.

**Note:** If spatially-complete covariates are used instead of coördinates in the OLS model, this is sometimes called “Kriging with External Drift”, abbreviated KED. The mathematics are exactly the same, the difference is in functions added to the OK system to make the UK system, either the coördinates (UK) or covariates (KED).

Recall the **universal model of spatial variation**:

$$Z(\mathbf{s}) = Z^*(\mathbf{s}) + \varepsilon(\mathbf{s}) + \varepsilon'(\mathbf{s}) \quad (15)$$

where:

- $(\mathbf{s})$  is a location in space, designated by a **vector** of coördinates;
- $Z(\mathbf{s})$  is the **true** (unknown) value of some property at the location;
- $Z^*(\mathbf{s})$  is the **deterministic** component, due to some **non-stochastic** process, i.e., the trend surface;
- $\varepsilon(\mathbf{s})$  is the **spatially-autocorrelated stochastic** component of the deviations from the trend;
- $\varepsilon'(\mathbf{s})$  is the pure (“white”) **noise** with no structure; this can not be modelled.

We have seen how to model  $Z^*(\mathbf{s}) + \varepsilon'(\mathbf{s})$  with a polynomial trend surface in §5. We have seen how to model the local spatial structure as  $\varepsilon(\mathbf{s}) + \varepsilon'(\mathbf{s})$  by Ordinary Kriging (OK) in §11. UK is an extension of OK that models the entire Equation 15 in one step.

### 13.1 Residual variogram

The residual variogram is computed from the OLS surface as in §9, but directly from the definition of the trend. Recall that here we do not have a GLS surface.

---

**TASK 68:** Compute and display an empirical variogram of the residuals from a 2<sup>nd</sup> order OLS trend surface. •

The `variogram` function requires an `sf` spatial object with the locations as coördinates and the target variable.

We have to refer to the coördinates within the Simple Features object to build the full 2<sup>nd</sup> order trend surface expression. In order to use these in a variogram formula, these must be explicitly include as covariates in the observations objects:

```
str(st_coordinates(aq.sf))

##  num [1:161, 1:2] 569464 573151 559974 553514 550350 ...
##  - attr(*, "dimnames")=List of 2
##  ..$ : chr [1:161] "1" "2" "3" "4" ...
##  ..$ : chr [1:2] "X" "Y"

names(aq.sf)

## [1] "z"          "zm"          "geometry"    "resid.tps"
## [5] "fit.ts2"    "res.ts2"     "res.ts2.gls"

aq.sf$E <- st_coordinates(aq.sf)[ , "X"]
aq.sf$N <- st_coordinates(aq.sf)[ , "Y"]
names(aq.sf)

## [1] "z"          "zm"          "geometry"    "resid.tps"
## [5] "fit.ts2"    "res.ts2"     "res.ts2.gls" "E"
## [9] "N"
```

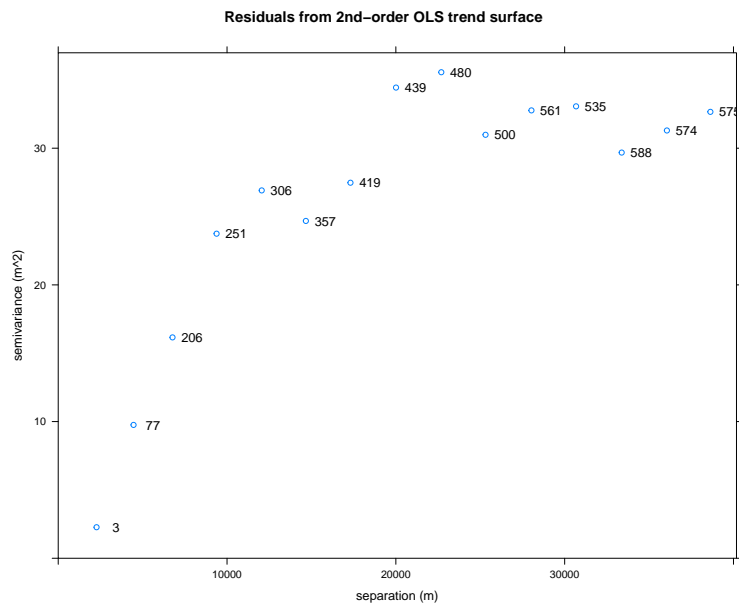
Now these names can be used in the variogram formula:

```
# summary(apply(st_coordinates(aq.sf), MARGIN = 1, FUN = prod))
vr <- variogram(zm ~ E + N + I(E^2) + I(N^2) + I(E*N),
```

```

locations = aq.sf,
cutoff = 40000)
plot(vr, plot.numbers = TRUE,
     main = "Residuals from 2nd-order OLS trend surface",
     xlab = "separation (m)",
     ylab = "semivariance (m^2)")

```



This variogram of the OLS trend surface residuals is then modelled as before (§11.2), using eyeball estimates of the exponential model parameters:

```

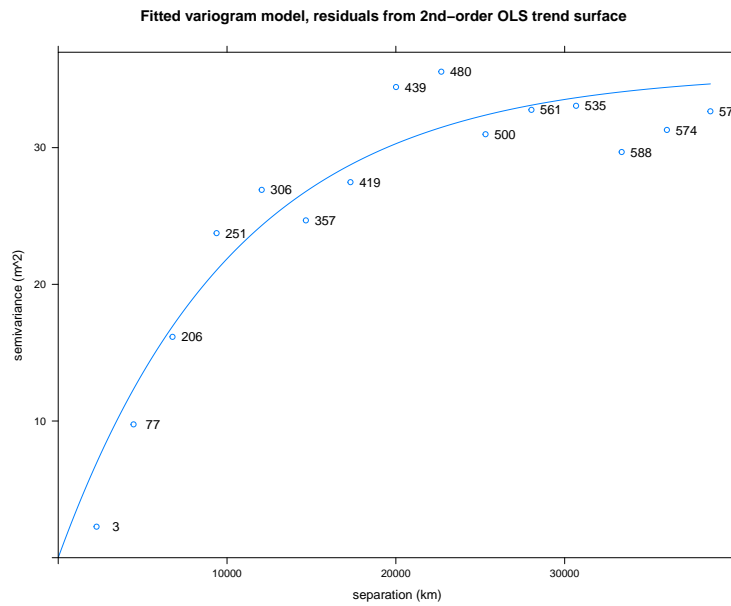
(vr.m.f <- fit.variogram(vr, vgm(35, "Exp", 22000/3, 0)))

## model psill range
## 1 Nug 0.0000 0.00
## 2 Exp 35.5515 10471.76

plot(vr, plot.numbers=TRUE,
     xlab="separation (km)", ylab="semivariance (m^2)",
     model=vr.m.f,
     main="Fitted variogram model, residuals from 2nd-order OLS trend surface")

```





**TASK 69 :** Compare this fitted variogram from the OLS trend surface residuals with the estimate from the GLS fit. •

```
print(vr.m.f)          # OLS trend residuals

##   model  psill  range
## 1  Nug  0.0000   0.00
## 2  Exp 35.5515 10471.76

print(vr.gls.m.f)     # GLS trend residuals

##   model  psill  range
## 1  Nug  0.00000  0.00
## 2  Exp 44.76969 14784.35
```

**Q38 :** How do these fitted variogram parameters compare to those from the GLS trend surface residuals (§11.2)? Why are they different? [Jump to A38](#) •

## 13.2 The Universal Kriging system

Recall from §11.3 that Kriging is a form of linear prediction of the attribute value at an unknown point  $\mathbf{s}_0$ , as a weighted sum of the attribute values at the known points  $z(\mathbf{s}_i)$ :

$$\hat{z}(\mathbf{s}_0) = \sum_{i=1}^N \lambda_i z(\mathbf{s}_i) \quad (16)$$

where the weights  $\lambda_i$  must sum to 1, and are determined by solving the **kriging system** of equations, which ensures that the prediction has the least possible **prediction variance**, i.e., uncertainty, among all the possible weights.

In OK the weights only take into account local spatial autocorrelation. In UK the weights  $\lambda_i$  take into account both the global trend and the local spatial autocorrelation of the trend residuals.

Here we do not derive the system, but present it.

The weights are the solution of the linear equation  $\mathbf{A}_U \lambda_U = \mathbf{b}_U$  where:

$$\mathbf{A}_U = \begin{bmatrix} \gamma(\mathbf{s}_1, \mathbf{s}_1) & \cdots & \gamma(\mathbf{s}_1, \mathbf{s}_N) & 1 & f_1(\mathbf{s}_1) & \cdots & f_k(\mathbf{s}_1) \\ \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \gamma(\mathbf{s}_N, \mathbf{s}_1) & \cdots & \gamma(\mathbf{s}_N, \mathbf{s}_N) & 1 & f_1(\mathbf{s}_N) & \cdots & f_k(\mathbf{s}_N) \\ 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ f_1(\mathbf{s}_1) & \cdots & f_1(\mathbf{s}_N) & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f_k(\mathbf{s}_1) & \cdots & f_k(\mathbf{s}_N) & 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\lambda_U = \begin{bmatrix} \lambda_1 \\ \cdots \\ \lambda_N \\ \psi_0 \\ \psi_1 \\ \cdots \\ \psi_k \end{bmatrix} \quad \mathbf{b}_U = \begin{bmatrix} \gamma(\mathbf{s}_1, \mathbf{s}_0) \\ \vdots \\ \gamma(\mathbf{s}_N, \mathbf{s}_0) \\ 1 \\ f_1(\mathbf{x}_0) \\ \vdots \\ f_k(\mathbf{x}_0) \end{bmatrix}$$

All of the semivariances  $\gamma$  in these formulas are computed from the fitted variogram model, by substituting the separation between the known points  $(\mathbf{s}_i, \mathbf{s}_j)$  in the  $\mathbf{A}_U$  matrix, and between the known points and the prediction point  $(\mathbf{s}_i, \mathbf{s}_0)$  in the  $\mathbf{b}_U$  vector.

Then the kriging system is solved by matrix inversion and multiplication as:

$$\lambda_U = \mathbf{A}_U^{-1} \mathbf{b}_U$$

These weights can then be used in the prediction formula Equation 16. They also can be used to compute the prediction variance as  $\hat{\sigma}^2 = \mathbf{b}_U^T \lambda_U$ .

### 13.3 Prediction

Now we have the model, the **krige** function can compute the UK prediction at any location, for example at all the grid points. Note that the formula given in **krige** **must** match that given in **variogram**.

---

**TASK 70** : Predict over the grid with the UK model. •

The predictor variables, i.e., the coördinates which were used in the residual variogram estimation, are only implicitly in the **sf** version of the prediction grid:

```

names(grid1km.sf)

## [1] "ts2.fit"          "ts2.lwr"          "ts2.upr"
## [4] "ts2.diff.range"  "pred.gam"         "pred.gam.se"
## [7] "diff.gam.ols"    "pred.tps"         "geometry"

str(grid1km.sf$geometry)

## sfc_POINT of length 7425; first list element: 'XY' num [1:2] 500500 4248500

names(aq.sf)

## [1] "z"          "zm"          "geometry"    "resid.tps"
## [5] "fit.ts2"    "res.ts2"     "res.ts2.gls" "E"
## [9] "N"

str(aq.sf$geometry)

## sfc_POINT of length 161; first list element: 'XY' num [1:2] 569464 4172115

```

In order to use a UK formula, these must be explicitly include as covariates in both the observation and prediction grid objects

```

str(st_coordinates(grid1km.sf))

## num [1:7425, 1:2] 500500 501500 502500 503500 504500 ...
## - attr(*, "dimnames")=List of 2
## ..$ : chr [1:7425] "1" "2" "3" "4" ...
## ..$ : chr [1:2] "X" "Y"

grid1km.sf$E <- st_coordinates(grid1km.sf)[ , "X"]
grid1km.sf$N <- st_coordinates(grid1km.sf)[ , "Y"]
names(grid1km.sf)

## [1] "ts2.fit"          "ts2.lwr"          "ts2.upr"
## [4] "ts2.diff.range"  "pred.gam"         "pred.gam.se"
## [7] "diff.gam.ols"    "pred.tps"         "geometry"
## [10] "E"                "N"

```

Now these names can be used in the UK formula.

```

k.uk <- krige(zm ~ E + N + I(E^2) + I(N^2) + I(E*N),
              locations = aq.sf,
              newdata = grid1km.sf,
              model=vr.m.f)

## [using universal kriging]

summary(k.uk)

##   var1.pred   var1.var   geometry
## Min.   :475.1   Min.    : 0.4626   POINT      :7425
## 1st Qu.:516.9   1st Qu.: 8.5469   epsg:26914   : 0
## Median :546.9   Median :10.9385   +proj=utm    ...: 0
## Mean   :547.3   Mean    :11.5170
## 3rd Qu.:576.3   3rd Qu.:13.2251
## Max.   :623.3   Max.    :41.6235

```

---

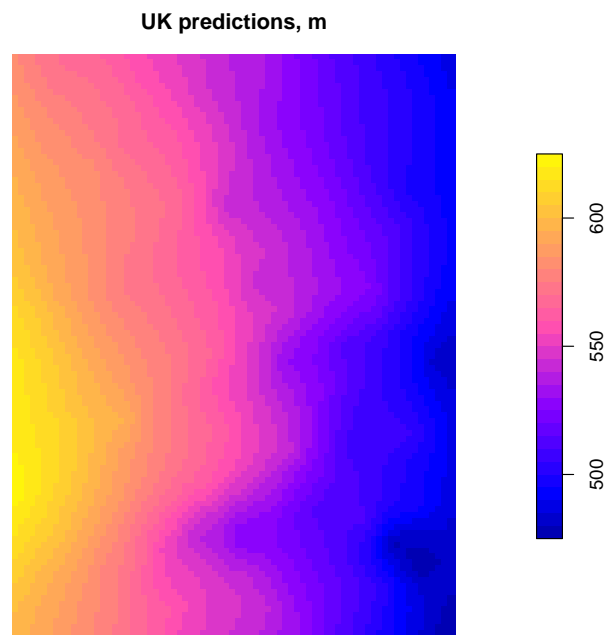
**TASK 71 :** Display the UK predictions and their prediction standard deviations

Predictions:

```

plot(k.uk["var1.pred"], pch=15, nbreaks=24,
     main="UK predictions, m")

```



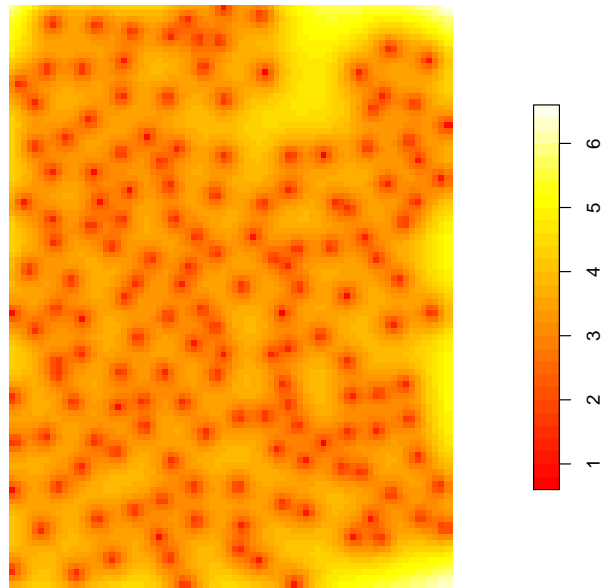
Prediction standard deviations (i.e., the square root of the variances, so they are on the same scale as the predictions):

```
k.uk$var1.sd <- sqrt(k.uk$var1.var)
summary(k.uk)
```

```
##      var1.pred      var1.var      geometry
## Min.   :475.1    Min.    : 0.4626    POINT      :7425
## 1st Qu.:516.9    1st Qu.: 8.5469    epsg:26914   : 0
## Median :546.9    Median :10.9385    +proj=utm ...: 0
## Mean   :547.3    Mean    :11.5170
## 3rd Qu.:576.3    3rd Qu.:13.2251
## Max.   :623.3    Max.    :41.6235
##      var1.sd
## Min.   :0.6802
## 1st Qu.:2.9235
## Median :3.3073
## Mean   :3.3189
## 3rd Qu.:3.6366
## Max.   :6.4516
```

```
plot(k.uk["var1.sd"], pch=15, nbreaks=24, pal = heat.colors,
      main="Standard errors of UK predictions, m")
```

Standard errors of UK predictions, m



**TASK 72 :** Compare these prediction standard deviations to those from the kriging of the GLS residuals. •

```
summary(k.uk$var1.sd)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.6802  2.9235   3.3073   3.3189  3.6366   6.4516
```

```
summary(kr$var1.sd)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.6425  2.7771   3.1473   3.1609  3.4713   5.6493
```

The UK deviations are greater because they include the uncertainty of the trend surface.

The obvious question is how close is this one-step procedure to the two-step procedure of GLS-RK (§12). Both methods take both global and local structure into account.

**TASK 73 :** Compute the difference between the UK and GLS-RK predictions, and display as a histogram and on a map. •

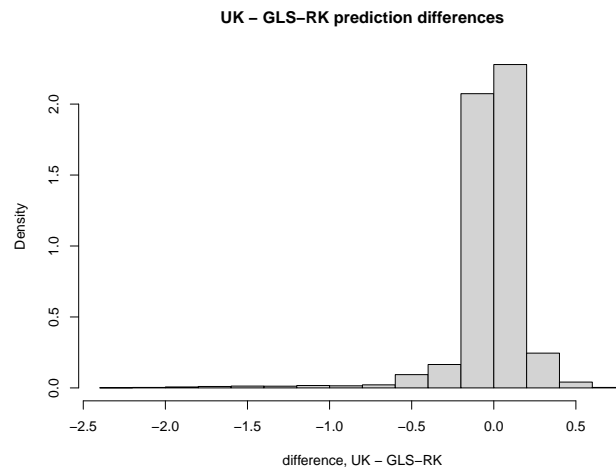
```
grid1km.uk <- grid1km
```

```
values(grid1km.uk) <- k.uk$var1.pred
```

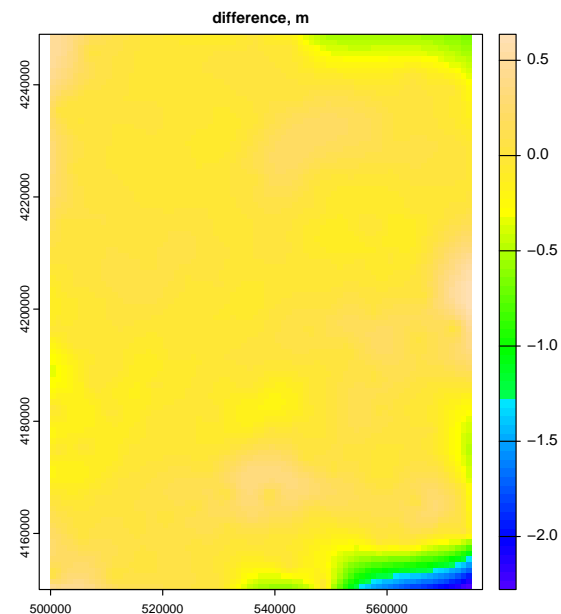
```
summary(grid1km.diff.uk.rkgls <- (grid1km.uk - grid1km.rkgls))
```

```
##      z
## Min.  :-2.27833
## 1st Qu.: -0.04609
## Median : 0.00146
## Mean   :-0.01812
## 3rd Qu.: 0.06089
## Max.   : 0.63664
```

```
hist(grid1km.diff.uk.rkgls, main = "UK - GLS-RK prediction differences",
     freq = FALSE, xlab = "difference, UK - GLS-RK")
```



```
plot(grid1km.diff.uk.rkgls, sub="UK - GLS-RK predictions",
     main="difference, m", xlab="East", ylab="North",
     col = topo.colors(64))
```




---

**Q39 :** *How large are differences between the UK and GLS-RK trend surface predictions? Where are the largest differences? Explain why there is a difference.* Jump to A39 •

## 14 Discussion

In this exercise we have compared several methods of predicting an attribute over space, from a set of geo-referenced observations.

**Discussion question:** In this study area, which of the prediction methods would you recommend, and why?

**Discussion question:** For each method introduced, in what situations would you prefer it to the other methods?

## 15 Answers

---

**A1 :** The map of aquifer elevations, along with a map of the elevation of the land surface, can be used by well-drillers, to estimate the cost of drilling a well to reach the aquifer at any location. [Return to Q1 •](#)

---

**A2 :** There are 161 observations (wells); for each we know the coördinates (E and N) and the elevation of aquifer (z); we also have the transformed elevation in meters and the reduced coördinates. [Return to Q2 •](#)

---

**A3 :** UTM East from  $5.003613 \times 10^5$  m ...  $5.744296 \times 10^5$  m (range 74.068 km); UTM North from  $4.1502482 \times 10^6$  m ...  $4.2483125 \times 10^6$  m (range 98.064 km); total area 7263 km<sup>2</sup>. [Return to Q3 •](#)

---

**A4 :** Elevations are from 475.5 to 623.2 m.a.s.l., a range of 147.7 m. [Return to Q4 •](#)

---

**A5 :** Nearby points tend to be similar; there appears to be trend from E to W, but there are portions of the map that do not follow this strictly. [Return to Q5 •](#)

---

**A6 :** (1) The text postplot has the advantage of showing the actual values, but it is not very graphical and difficult to read; (2) the size postplot clearly shows the relative data values; (3) the size and colour postplot gives two ways to visualize; it seems especially good for seeing the E-W increasing first-order trend. [Return to Q6 •](#)

---

**A7 :** The aquifer has a flat surface, tilted towards some direction, by some regional uplift. In this case, the uplift of the Rocky Mountains about 650 km to the west has tilted the aquifer. [Return to Q7 •](#)

---

**A8 :** The trend surface equation is:  $z = 1557 + -0.001617 E + -3.3 \times 10^{-5} N$ . The intercept term gives the estimated aquifer elevation at the centroid of the area. Then the two coefficients give the change in elevation per unit change of the target variable. That is, for each km E the elevation decreases by -1.62 m, for each km N it decreases by -0.03 m. The relation is highly-significant; it explains 94.1% of the variability in the observations; however the N coördinate is not needed - it is not statistically different from zero. [Return to Q8 •](#)

---

**A9 :** Residuals range from -25.4 to 16.7 m; compare this to the median elevation 552.8 m; the maximum calibration error is 4.6%. [Return to Q9 •](#)

---

**A10 :**



1. No relation between fitted values and residuals; but ...
2. a slight relation between spread of residuals and the fitted values: a “cone” shape with a wider spread of residuals at the higher fitted values; but more seriously ...
3. the residuals are not normally-distributed, especially in the high tail. That is, the largest positive residuals (under-predictions) are not as extreme as would be expected. The largest negative residuals (over-predictions) are a bit too extreme.

Conclusion: this OLS fit does not satisfy the assumptions of independent residuals. [Return to Q10](#)

•

---

**A11 :** There is a spatial pattern. Large residuals tend to be near each other, and vice-versa. Positive residuals (above the trend surface) are found almost exclusively in the middle third of the map. Dependence seems to be stronger along a SW-NE axis (range about 50 to 70 km) than the NW-SE axis (range about 10 to 20 km). This implies a higher-order trend surface or a periodic surface superimposed on the linear trend. [Return to Q11](#) •

---

**A12 :** The tilted structure has local warping as either a dome or a basin. [Return to Q12](#) •

---

**A13 :** The model explains 97.5% of the variance in the observations, compared to 94.1% for the first-order significance. [Return to Q13](#) •

---

**A14 :** The probability that the higher-order surface is this much better just by chance is almost zero, so the second-order surface is statistically superior to the first-order surface. [Return to Q14](#) •

---

**A15 :** Residuals range from -19.8 to 14.8 m; compare this to the median elevation 552.8 m; the maximum calibration error is 3.6%. This range is somewhat narrower than for the first-order surface: -25.4 to 16.7 m. [Return to Q15](#) •

---

**A16 :**

1. No relation between fitted values and residuals;
2. no slight relation between spread of residuals and the fitted values; but ...
3. The residuals are closer to normally-distributed. The largest negative residuals (over-predictions) are still too extreme, by up to -20 m. However, the problem with the largest positive residuals (under-predictions) from the first-order surface has been solved.

*Conclusion: this 2<sup>nd</sup>-order OLS fit is much closer to being a valid linear model than the 1<sup>st</sup>order OLS fit.*

*[Return to Q16](#) •*

---

**A17 :** *These residuals form local clusters of positive, negative, and near-zero; there does not appear to be any overall spatial pattern. So, a higher-order trend surface is not indicated. Instead, some local interpolation of the residuals would seem to improve the model. There is no evidence of anisotropy. All of the largest over-predictions are in the SE portion of the map; three are near neighbours.*

*[Return to Q17](#) •*

---

**A18 :** *The fit is generally good but some clusters of points stand out from the background; their values are not that well matched.*

*[Return to Q18](#) •*

---

**A19 :** *The prediction errors are from 22.3 to 24.2 m; this is about 4.1% of the predicted value. This much uncertainty in the prediction corresponds to uncertainty in the expense of drilling a well at the location.*

*[Return to Q19](#) •*

---

**A20 :** *They are least at the centre of gravity of the regression in both E and N; they increase away from this in both directions; the largest uncertainties are in the corners of the grid.*

*[Return to Q20](#) •*

---

**A21 :** *The relation with East seems almost linear, and a very tight relation. However, the relation with North is much more scattered, and seems to have higher elevations towards the middle of the range.*

*[Return to Q21](#) •*

---

**A22 :** *Extremely well.*

*[Return to Q22](#) •*

---

**A23 :** *The GAM has a much smaller spread of residuals, much more concentrated towards zero.*

*[Return to Q23](#)*

•

---

**A24 :** *There is short-range spatial autocorrelation, to about 10 km. This is about 1/3 the range of the spatial autocorrelation of the OLS trend surface residuals.*

*[Return to Q24](#) •*

---

**A25 :** *The GAM trend shows local warping.*

*[Return to Q25](#) •*

---

**A26 :** *The fit is very good, the residuals are smaller than for OLS or GLS. A few large negative residuals (over-predictions) are in the south-central and southeast.*

*[Return to Q26](#) •*

---

**A27 :** *The GAM predicts higher elevations of the aquifer in the NE and especially the SE, and some areas in the W. OLS predicts higher elevations in the NW*

and a band from SW to E. The predictions are the same at the map centroid. The differences are because the GAM adjust locally, especially to the higher values in the SE that do not fit the overall trend (see the map of trend surface residuals). [Return to Q27](#) •

---

**A28** : The sill is estimated as 35 m<sup>2</sup>, there is no nugget, the sill is reached at a range of about 20 km. [Return to Q28](#) •

---

**A29** : The range parameter of spatial correlation of the exponential model, as estimated by *gls*, is 14.4 km. [Return to Q29](#) •

---

**A30** : The GLS coefficients take into account the spatial correlation of the trend surface residuals, i.e., the fit uses a variance-covariance matrix of the residuals to adjust the least-squares fit. [Return to Q30](#) •

---

**A31** : (1) The mean OLS residual is by definition zero, whereas the mean GLS residual is slightly biased; (2) The IQR of the OLS residuals is narrower than for the GLS residuals.

The OLS fit is on average closer, because the clustered high-leverage points draw it closer to them. See the postplot of the residuals on the two trend surfaces in the SE of the map. [Return to Q31](#) •

---

**A32** : The GLS surface is higher than the OLS surface in the NW and especially SE, and lower in the NE and SW. This shows that some clustered observations affected the OLS fit, especially the local warping of the aquifer in the SE. However the absolute differences are not much, given the fits on the order of 460–600 m. [Return to Q32](#) •

---

**A33** : Sill about 38 m<sup>2</sup>, no nugget, range about 22 km. [Return to Q33](#) •

---

**A34** : Yes, this range parameter here is 14.8 km; the range estimated by *gls*, is 14.4 km. These are very close, despite being fit in two different ways. [Return to Q34](#) •

---

**A35** : The largest adjustments towards lower aquifer elevations are in a NE-SW band towards the SE of the map. The largest adjustments towards higher aquifer elevations are in a large spot at the SW-center side of the map. These correspond to local warping of the overall aquifer structure at the scale revealed by the variogram model. [Return to Q35](#) •

---

**A36** : The prediction uncertainty is least near observation points, especially near clusters of them. It is most away from points. This is because the semi-variance between observation points and prediction points increases with sep-

aration.

[Return to Q36](#)

- 

---

**A37 :** The differences here are mainly because in GLS-RK the kriging of the residuals finds local highs and lows that can not be fully accounted for by the smooth function of the GAM. This is most clearly seen with the negative residual “spots”, i.e., where the GAM over-predicts but the kriged residuals adjust locally to these lower-than-expected values. [Return to Q37](#) •

---

**A38 :** The range parameters are  $1.47843 \times 10^4$  (GLS) and  $1.04718 \times 10^4$  (OLS), so that the GLS range is about 40% longer. The partial sills are 44.8 (GLS) and 35.6 (OLS), so that the GLS sill is about 25% higher. This implies that GLS has removed less of the local variation in the residuals than OLS, or in other words, OLS incorrectly removed this variation. [Return to Q38](#) •

---

**A39 :** The differences are quite small, almost all  $< \pm 2.5$  m, with a range of  $\approx -2.28 \dots 0.64$  m. These are much smaller differences than between GAM and GLS-RK. Differences are skewed to the positive differences, i.e., UK  $>$  GLS-RK. This difference comes about because UK uses the fitted variogram from the 2<sup>nd</sup>-order OLS trend residuals, not from the 2<sup>nd</sup>-order OLS residuals. Because the two trend surfaces are different, so are the residuals, and so are the fitted models. The largest difference is in the SE and NE edges, where the GLS trend surface most diverges from the OLS trend surface. [Return to Q39](#) •

## A Derivation of the OLS solution to the linear model

Recall the equation for OLS from §5.1:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon \quad (17)$$

To solve Equation 17 we need an **optimization criterion**, i.e., what makes a particular solution (values of  $\beta$ ) better than any other. The obvious criterion is to minimize the total error (lack of fit) as some function of  $\varepsilon = \mathbf{y} - \mathbf{X}\beta$ ; the goodness-of-fit is then measured by the size of this error. A common way to measure the total error is by the sum of vector norms; in the simplest case the Euclidean distance from the expected value, which we take to be 0 in order to have an unbiased estimate. If we decide that both positive and negative residuals are equally important, and that larger errors are more serious than smaller, the vector norm is expressed as the sum of squared errors, which in matrix algebra can be written as:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \quad (18)$$

which expands to

$$\begin{aligned} S &= \mathbf{y}^T \mathbf{y} - \beta^T \mathbf{X}^T \mathbf{y} - \mathbf{y}^T \mathbf{X} \beta + \beta^T \mathbf{X}^T \mathbf{X} \beta \\ S &= \mathbf{y}^T \mathbf{y} - 2\beta^T \mathbf{X}^T \mathbf{y} + \beta^T \mathbf{X}^T \mathbf{X} \beta \end{aligned} \quad (19)$$

**Note:**  $\mathbf{y}^T \mathbf{X} \beta$  is a  $1 \times 1$  matrix, i.e., a scalar<sup>19</sup>, so it is equivalent to its transpose:  $\mathbf{y}^T \mathbf{X} \beta = [\mathbf{y}^T \mathbf{X} \beta]^T = \beta^T \mathbf{X}^T \mathbf{y}$ . So we can collect the two identical  $1 \times 1$  matrices (scalars) into one term.

This is minimized by finding the partial derivative with respect to the unknown coefficients  $\beta$ , setting this equal to 0, and solving:

$$\begin{aligned} \frac{\partial}{\partial \beta^T} S &= -2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X} \beta \\ \mathbf{0} &= -\mathbf{X}^T \mathbf{y} + \mathbf{X}^T \mathbf{X} \beta \\ (\mathbf{X}^T \mathbf{X}) \beta &= \mathbf{X}^T \mathbf{y} \\ (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X}) \beta &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ \hat{\beta}_{\text{OLS}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \end{aligned} \quad (20)$$

which is the OLS solution.

## B Standardized residuals

Standardized residuals<sup>20</sup> adjust the residuals from a linear regression model to residuals which should be distributed as  $\mathcal{N}(0, 1)$  with equal variance. These can then be compared to residuals drawn from that theoretical distribution, for example in a quantile-quantile (“QQ”) plot of the standardized residuals.

<sup>19</sup> The dimensions of the matrix multiplication are  $(1 \times n)(n \times p)(p \times 1)$

<sup>20</sup> This is the term used by `plot.lm`; some authors call this the “studentized” residuals.

The standardized residuals are computed as  $r_i / (s \cdot \sqrt{1 - h_{ii}})$ , where  $r_i$  are the unstandardized residuals,  $s$  is the sample standard deviation of the residuals, and the  $h_{ii}$  are the diagonal entries of the so-called “hat” matrix  $V = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ .

The sample standard deviation of the residuals  $s$  is computed as the square root of the estimated variance of the random error:

$$s = \sqrt{\frac{1}{(n - p)} \cdot \sum r_i^2}$$

where  $n$  is the number of observations and  $p$  the number of predictors. It is shown in the linear model summary as “Residual standard error”; it can be extracted as `summary(model_name)$sigma`. This is an overall measure of the variability of the residuals, and so can be used to standardize the residuals to  $\mathcal{N}(0, 1)$ .

The “hat” matrix  $V$  is another way to look at linear regression. This matrix multiplies the observed values to compute the fitted values. The hat value for an observation gives the overall leverage (i.e., importance when computing the fit) of that observation. So the term  $\sqrt{1 - h_{ii}}$  in the denominator shows that with low influence (small  $h_{ii}$ ) the ratio  $r_i/s$  (a simple standardization) is not affected much, but with a high influence (large  $h_{ii}$ ) the denominator is smaller and so the standardized residual is increased. Thus the standardized residuals are higher for points with high influence on the regression coefficients.

## C Theory of thin-plate splines

Hastie et al. [5, §5.7] explains the mathematics of multi-dimensional smoothing splines. A more thorough mathematical treatment is given by Wood [19] and Mitsova and Mitso [13]; these are developments from the “minimum curvature” methods of Briggs [1]. Applications include Hutchinson [7] and Mitsova and Hofierka [12].

Fitting a TPS depends on the  $k$  data points with known coördinates and attribute values. They can be described by  $2(k + 3)$  parameters, six of which are overall affine transformation parameters (to center the function in 2D) and  $2k$  of which link to the control points.

The general method is to minimize the residual sum of squares (RSS) of the fitted function, subject to a constraint that the function be “smooth” in some sense; this is expressed by a **roughness penalty** which balances the fit to the observations with smoothness. This is a minimization problem. If  $\mathbf{x}_i$  is one point in 2D space (i.e., it has two coördinates) and  $y_i$  is the attribute value at the same points, the aim is to minimize:

$$\min_f \sum_{i=1}^N \{y_i - f(\mathbf{x}_i)\}^2 + \lambda J[f] \quad (21)$$

where  $J$  is the penalty function and  $\lambda$  controls how important it is;  $\lambda = 0$  means there is no roughness penalty and the data will be fit exactly; as

$\lambda \rightarrow \omega$  the solution approximates the least-squares plane, i.e., the trend surface averaged over all the points.

In 2D an appropriate penalty is:

$$J[f] = \int_{\mathbb{R}} \int_{\mathbb{R}} \left[ \left( \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} \right)^2 + 2 \left( \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} \right)^2 + \left( \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} \right)^2 \right] dx_1 dx_2 \quad (22)$$

where  $(x_1, x_2)$  are the two coördinates of the vector  $\mathbf{x}$ . In practice the double integral is discretized over some grid known as **knots**; these may be defined by the observations or may be a different set, maybe an evenly-spaced grid.

This penalty can be interpreted as the “bending energy” of a thin plate represented by the function  $f(x)$ ; by minimizing this energy the spline function in over the 2D plane is a thin (flexible) plate which, according to the first term of Equation 21 would be forced to pass through data points, with minimum bending. However the second term of Equation 21 allows some smoothing: the plate does not have to bend so much, since it is allowed to pass “close to” but not necessarily through the data points. The higher the  $\lambda$ , the less exact is the fit.

This has two purposes: (1) it allows for measurement error; the data points are not taken as exact; (2) it results in a smoother surface. So cross-validation is used to determine the degree of smoothness.

The solution to Equation 22 is a linear function:

$$f(\mathbf{x}) = \beta_0 + \beta^T \mathbf{x} + \sum_{j=1}^N \alpha_j h_j(\mathbf{x}) \quad (23)$$

where the  $\beta$  account for the overall trend and the  $\alpha$  are the coefficients of the warping.

The set of functions  $h_j(\mathbf{x})$  is the **basis kernel**, also called a **radial basis function** (RBF), for thin-plate splines:

$$h_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_j\|^2 \log \|\mathbf{x} - \mathbf{x}_j\| \quad (24)$$

where the norm distance  $r = \|\mathbf{x} - \mathbf{x}_j\|$  is also called the **radius** of the basis function. The norm is usually the Euclidean (straight-line) distance.

## D Theory of GLS and REML

Here we present the theory of Generalized Least Squares (GLS) estimation of the parameters of the linear model (§D.1), the specific case of GLS with spatially-correlated residuals (§D.2), as well as the estimation of the covariance structure of the linear model residuals (§D.3).

## D.1 GLS

The difference between Generalized Least Squares (GLS) and Ordinary Least Squares (OLS) solutions to the linear model is that in the linear model fit by OLS, the residuals  $\varepsilon$  are assumed to be *independently* and *identically* distributed with the same variance  $\sigma^2$ :

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \quad (25)$$

Whereas, in GLS the residuals are themselves considered to be a random variable  $\eta$  that has a covariance structure:

$$\mathbf{y} = \mathbf{X}\beta + \eta, \quad \eta \sim \mathcal{N}(0, \mathbf{V}) \quad (26)$$

where  $\mathbf{V}$  is a positive-definite variance-covariance matrix of the model residuals.

In modelling terminology, the coefficients  $\beta$  are called **fixed** effects, because their effect on the response variable is fixed once the parameters are known. By contrast the covariance parameters  $\eta$  are called **random** effects, because their effect on the response variable is stochastic, depending on a random variable with these parameters. Models with the form of Equation 26 are called **mixed** models: some effects are fixed (in the example of this tutorial, the relation between coördinates and the aquifer elevation) and others are random (here, the error variances) but follow a known structure. These models have many applications and are extensively discussed in Pinheiro and Bates [16].

Lark and Cullis [10, Appendix] point out that the error vectors can now not be assumed to be spherically distributed in feature space around the 0 expected value, but rather that error vectors in some directions are longer than in others. So, the measure of distance (the vector norm) is now a so-called “generalized” distance<sup>21</sup>, taking into account the covariance between error vectors:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T \mathbf{V}^{-1} (\mathbf{y} - \mathbf{X}\beta) \quad (27)$$

The OLS equivalent is simpler:

$$S = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \quad (28)$$

Comparing these equations, we see that the GLS formulation of Equation 27 includes the variance-covariance matrix of the residuals  $\mathbf{V} = \sigma^2 \mathbf{C}$ , where  $\sigma^2$  is the variance of the residuals and  $\mathbf{C}$  is the correlation matrix. This reduces to the OLS formulation of Equation 28 if there is no covariance, i.e.,  $\mathbf{V} = \mathbf{I}$ .

The covariances can be based on any relation, for example, time dependence (e.g., correlation between measurements that are close in time),

<sup>21</sup> This is closely related to the Mahalanobis distance



repeated measures of the same object, or spatial dependence (the case here, see next sub-section for details).

Expanding Equation 27, taking the partial derivative with respect to the parameters, setting equal to zero and solving we obtain:

$$\begin{aligned}\frac{\partial}{\partial \beta} S &= -2\mathbf{X}^T \mathbf{V}^{-1} \mathbf{y} + 2\mathbf{X}^T \mathbf{V}^{-1} \mathbf{X} \beta \\ 0 &= -\mathbf{X}^T \mathbf{V}^{-1} \mathbf{y} + \mathbf{X}^T \mathbf{V}^{-1} \mathbf{X} \beta \\ \hat{\beta}_{\text{GLS}} &= (\mathbf{X}^T \mathbf{V}^{-1} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{V}^{-1} \mathbf{y}\end{aligned}\quad (29)$$

This reduces to the OLS estimate  $\hat{\beta}_{\text{OLS}}$  if there is no covariance, i.e.,  $\mathbf{V} = \mathbf{I}$ .

## D.2 GLS with spatially-correlated residuals

In the case that the residuals are **spatially-correlated**, the covariances (off-diagonals) in the  $\mathbf{V}$  matrix are typically based on the distance between observations, using some model of spatial correlation. We ensure positive-definiteness (i.e., always a real-valued solution) by using an **authorized spatial covariance function**  $C$  and assuming that the entries are completely determined by the **vector distance**<sup>22</sup> between points  $\mathbf{x}_i - \mathbf{x}_j$ :

$$C_{i,j} = C(\mathbf{x}_i - \mathbf{x}_j) \quad (30)$$

In this formulation  $C$  has a three-parameter vector  $\theta$ , as does the corresponding variogram model: the range  $a$ , the total sill  $\sigma^2$ , and the proportion of total sill due to pure error, not spatial correlation  $s$ <sup>23</sup>.

Here the random effect  $\eta$  represents both the spatial structure of the residuals from the fixed-effects model, and the unexplainable (short-range) noise. This latter corresponds to the noise  $\sigma^2$  of the linear model of Equation 25.

To solve Equation 29 we first need to compute  $\mathbf{V}$ , i.e., estimate the covariance parameters  $\theta = [\sigma^2, s, a]$ , use these to compute  $C$  with equation 30 and from this  $\mathbf{V}$ , after which we can use equation 29 to estimate the fixed effects  $\beta$ . But  $\theta$  is estimated from the residuals of the fixed-effects regression, which has not yet been computed. How can this “chicken-and-egg”<sup>24</sup> computation be solved?

The answer is to use **residual** (sometimes called “restricted”) **maximum likelihood** (REML) to maximize the likelihood of the random effects  $\theta$  independently of the fixed effects  $\beta$ .

Here we fit the fixed effects (regression coefficients) at the same time as we estimate the spatial correlation.

<sup>22</sup> so this distance measure can take into account angles, i.e., anisotropy

<sup>23</sup> In variogram terms, this is the nugget variance  $c_0$  as a proportion of the total sill  $(c_0 + c_1)$ .

<sup>24</sup> from the question “which came first, the chicken or the egg?”

Lark and Cullis [10, Eq. 12] show that the likelihood of the parameters in Equation 25 can be expanded to include the spatial dependence implicit in the variance-covariance matrix  $\mathbf{V}$ , rather than a single residual variance  $\sigma^2$ . The log-likelihood is then:

$$\ell(\beta, \theta | \mathbf{y}) = c - \frac{1}{2} \log |\mathbf{V}| - \frac{1}{2} (\mathbf{y} - \mathbf{X}\beta)^T \mathbf{V}^{-1} (\mathbf{y} - \mathbf{X}\beta) \quad (31)$$

where  $c$  is a constant (and so does not vary with the parameters) and  $\mathbf{V}$  is built from the variance parameters  $\theta$  and the distances between the observations. By assuming **second-order stationarity**<sup>25</sup>, the structure can be summarized by the covariance parameters  $\theta = [\sigma^2, s, a]$ , i.e., the total sill, nugget proportion, and range.

However, maximizing this likelihood for the random-effects covariance parameters  $\theta$  also requires maximizing in terms of the fixed-effects regression parameters  $\beta$ , which in this context are called *nuisance parameters* since at this point we don't care about their values; we will compute them after determining the covariance structure.

### D.3 REML estimation of the covariance parameters

Both the covariance and the nuisance parameters  $\beta$  must be estimated, it seems at the same time (“chicken and egg” problem) but in fact the technique of REML can be used to first estimate  $\theta$  without having to know the nuisance parameters. Then we can use these to compute  $\mathbf{C}$  with equation 30 and from this  $\mathbf{V}$ , after which we can use equation 29 to estimate the fixed effects  $\beta$ .

The maximum likelihood estimate of  $\theta$  is thus called “restricted”, because it only estimates the covariance parameters (random effects). Conceptually, REML estimation of the covariance parameters  $\theta$  is ML estimation of both these and the nuisance parameters  $\beta$ , with the later integrated out:

$$\ell(\theta | \mathbf{y}) = \int \ell(\beta, \theta | \mathbf{y}) d\beta \quad (32)$$

Pinheiro and Bates [16, §2.2.5] show how this is achieved, given a likelihood function, by a change of variable to a statistic sufficient for  $\beta$ .

<sup>25</sup> that is, the covariance structure is the same over the entire field, and only depends on the vector distance between pairs of points

## References

- [1] I. C. Briggs. Machine contouring using minimum curvature. *Geophysics*, 39(1):39–48, January 1974. ISSN 0016-8033, 1942-2156. doi: 10.1190/1.1440410. 80
- [2] N Cressie. *Statistics for spatial data*. John Wiley & Sons, revised edition, 1993. 49
- [3] J. C. Davis. *Statistics and data analysis in geology*. John Wiley & Sons, New York, 3rd edition, 2002. 3, 6, 25
- [4] J Fox. *Applied regression, linear models, and related methods*. Sage, Newbury Park, 1997. 49
- [5] T Hastie, R Tibshirani, and J H Friedman. *The elements of statistical learning data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2nd ed edition, 2009. ISBN 9780387848587. 33, 80
- [6] Jan M. Hoem. The reporting of statistical significance in scientific journals: A reflexion. *Demographic Research*, 18(15):437–442, Jun 2008. ISSN 1435-9871. doi: 10.4054/DemRes.2008.18.15. 3
- [7] M. F. Hutchinson. Interpolating mean rainfall using thin plate smoothing splines. *International Journal of Geographical Information Science*, 9(4):385 – 403, 1995. 80
- [8] R Ihaka and R Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. 1
- [9] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning: with applications in R*. Number 103 in Springer texts in statistics. Springer, 2013. ISBN 9781461471370. 33
- [10] R. M. Lark and B. R. Cullis. Model based analysis using REML for inference from systematically sampled data on soil. *European Journal of Soil Science*, 55(4):799–813, 2004. 82, 84
- [11] Virginia L. McGuire. Water-level and recoverable water in storage changes, High Plains aquifer, predevelopment to 2015 and 2013–15. USGS Numbered Series 2017-5040, U.S. Geological Survey, Reston, VA, 2017. 6
- [12] H. Mitsova and J. Hofierka. Interpolation by regularized spline with tension: II. Application to terrain modeling and surface geometry analysis. *Mathematical Geology*, 25(6):657–669, 1993. doi: 10.1007/BF00893172. 80
- [13] H. Mitsova and L. Mitas. Interpolation by regularized spline with tension: I. Theory and implementation. *Mathematical Geology*, 25(6):641–655, 1993. doi: 10.1007/BF00893171. 80

- [14] R. A. Olea and J. C. Davis. Sampling analysis and mapping of water levels in the High Plains aquifer of Kansas. Technical Report KGS Open File Report 1999-11, Kansas Geological Survey, May 1999. URL [http://www.kgs.ku.edu/Hydro/Levels/OFR99\\_11/](http://www.kgs.ku.edu/Hydro/Levels/OFR99_11/). 4
- [15] R. A. Olea and J. C. Davis. Optimization of the high plains aquifer water-level observation network. Technical Report KGS Open File Report 1999-15, Kansas Geological Survey, May 1999. URL [http://www.kgs.ku.edu/Hydro/Levels/OFR99\\_15/](http://www.kgs.ku.edu/Hydro/Levels/OFR99_15/). 4
- [16] J C Pinheiro and D M Bates. *Mixed-effects models in S and S-PLUS*. Springer, 2000. ISBN 0387989579. 82, 84
- [17] R Development Core Team. *R Data Import/Export*. The R Foundation for Statistical Computing, version 3.6.2 (2019-12-12) edition, 2015. URL <http://cran.r-project.org/doc/manuals/R-data.pdf>. 6
- [18] W N Venables, D M Smith, and R Development Core Team. *An introduction to R; notes on R: a programming environment for data analysis and graphics; Version 3.6.2*. R Foundation for Statistical Computing, Dec 2019. ISBN ISBN 3-900051-12-7. URL <http://www.R-project.org>. 1
- [19] S. N. Wood. Thin plate regression splines. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 65:95–114, 2003. doi: 10.1111/1467-9868.00374. 80

## Index of R Concepts

`^` formula operator, 21  
`~` formula operator, 14  
`*` formula operator, 21

`adj` argument (text function), 13  
`anova`, 21  
array class, 40, 44  
`as.numeric`, 40  
`asp=1` argument (plot function), 13

base package, 1  
`bpy.colors` (sp package), 13, 14

`ceiling`, 26  
`cex` argument (plot function), 13, 19  
`coef`, 51  
`col` argument (plot function), 19  
`col` argument to `terra::plot` function, 31  
`col2rgb`, 36  
`colors`, 36  
`colours`, 36  
`corExp` argument (nlme function), 50  
correlation argument (gls function), 50

`data.frame` class, 7, 8, 16, 27, 40, 44  
`data.frame` dataset, 11  
datasets package, 1  
`diff`, 11  
`dim`, 11  
`drop_units` (units package), 16

`expand.grid`, 26

fields package, 2, 42-44  
`fit.variogram` (gstat package), 57  
fitted, 47  
`floor`, 26  
form argument (`corExp` function), 50

`gam` (mgcv package), 34  
`geom_smooth` (ggplot2 package), 33  
geometry field, 16  
ggplot2 package, 2, 33, 36  
gls (nlme package), 50, 51, 77  
graphics package, 1  
grDevices package, 1  
`grid.arrange` (gridExtra package), 33  
gridExtra package, 2, 33

`gstat` class, 48  
gstat package, 2, 47

`head`, 15  
`hist`, 17

`I`, 21  
`ifelse`, 19, 30  
`interval` argument (`predict.lm` function), 28  
intervals, 52  
`intervals.gls` (nlme package), 52

`krige` (gstat package), 59, 68

`level` argument (`predict.lm` function), 28  
library, 2  
`lm`, 15, 16, 21, 34  
`load`, 10  
`loc` argument (`krige` function), 59  
`loess`, 33  
`log`, 34

`matrix`, 43  
`max`, 26  
methods package, 1  
mgcv package, 2, 34, 37, 38  
`min`, 26  
`model` argument () function, 59  
`model` argument (gls function), 50  
`model.matrix`, 15

`newdata` argument () function, 59  
nlme package, 2, 50, 52  
nugget argument (`corExp` function), 50

options, 3

`par`, 18  
`plot`, 12, 18  
`plot` (sf package), 30  
`plot.gam` (mgcv package), 37  
`plot.lm`, 18  
`predict`, 52  
`predict.gam` (mgcv package), 39, 40, 44  
`predict.gls` (nlme package), 52  
`predict.Krig` (fields package), 44  
`predict.lm`, 28

- range, 11
- rank, 13
- rast (terra package), 26
- read.table, 6
- require, 2
- residuals, 17, 47
- rstandard, 24
  
- s (mgcv package), 34
- save, 10
- scheme argument (plot.gam function), 37
- se.fit argument (predict.gam function), 39
- select argument (plot.gam function), 37
- seq, 26
- set\_units (units package), 7
- setwd, 5
- sf class, 8, 16, 59, 65, 68
- sf package, 2, 7, 8, 59
- skip argument (read.table function), 6
- sp package, 14
- span argument (loess function), 33
- SpatRast class, 53, 59, 62
- SpatRaster class, 26, 30
- sqrt, 34
- st\_as\_sf (sf package), 8, 59
- st\_bbox (sf package), 11
- st\_coordinates (sf package), 8, 16
- st\_crs (sf package), 9
- stats package, 1
- summary, 11
  
- terra (extract package), 44
- terra package, 2, 26, 29, 40
- text, 13
- theta argument (plot.gam function), 37
- Tps (fields package), 43
  
- units class, 33
- units package, 2, 7, 16
- utils package, 1
  
- value argument (corExp function), 50
- values (terra package), 29
- variogram (gstat package), 47, 65, 68
- vgm (gstat package), 57
- vis.gam (mgcv package), 38
  
- which argument (plot.lm function), 18