

# Geostatystyka w R

*Jakub Nowosad*

*2016-09-03*



# Spis treści

<b>1 Wprowadzenie</b>	<b>5</b>
1.1 Wymagania wstępne . . . . .	5
<b>2 R a dane przestrzenne</b>	<b>7</b>
2.1 R a dane przestrzenne . . . . .	7
2.2 Import danych . . . . .	8
2.3 Eksport danych . . . . .	12
2.4 Wizualizacja danych 2D . . . . .	12
2.5 Tworzenie siatek . . . . .	16
<b>3 Eksploracyjna analiza danych nieprzestrzennych</b>	<b>21</b>
3.1 Cele eksploracyjnej analizy danych . . . . .	21
3.2 Dane . . . . .	21
3.3 Statystyki opisowe . . . . .	23
3.4 Wykresy . . . . .	25
3.5 Porównanie zmiennych . . . . .	28
3.6 Transformacje danych . . . . .	32
<b>4 Eksploracyjna analiza danych przestrzennych</b>	<b>41</b>
4.1 Mapy . . . . .	41
4.2 Próbkowanie . . . . .	41
4.3 Dane lokalnie odstające . . . . .	45
4.4 Rozgrupowanie danych . . . . .	47
<b>5 Metody interpolacji</b>	<b>59</b>
5.1 Modele deterministyczne . . . . .	59
5.2 Modele statystyczne . . . . .	66
<b>6 Geostatystyka - prolog</b>	<b>67</b>
6.1 Geostatystyka . . . . .	67
6.2 Przestrzenna kowariancja i korelacja i semiwariancja . . . . .	69
6.3 Anizotropia . . . . .	79
<b>7 Modelowanie matematycznie autokorelacji przestrzennej</b>	<b>83</b>
7.1 Modelowanie matematycznie autokorelacji przestrzennej . . . . .	83
7.2 Modele podstawowe . . . . .	83
7.3 Metody modelowania . . . . .	86
7.4 Modelowanie izotropowe . . . . .	87
7.5 Modelowanie anizotropowe . . . . .	105
<b>8 Estymacje jednozmienne</b>	<b>107</b>
8.1 Kriging . . . . .	107
8.2 Kriging prosty . . . . .	107
8.3 Kriging zwykły . . . . .	110
8.4 Kriging z trendem . . . . .	112
8.5 Porównanie wyników SK, OK i KZT . . . . .	115

<b>9 Estymacja lokalnego rozkładu prawdopodobieństwa</b>	<b>117</b>
9.1 Kriging danych kodowanych . . . . .	117
9.2 Kriging danych kodowanych   Przykłady . . . . .	117
<b>10 Estymacje wielozmienne</b>	<b>125</b>
10.1 Kokriging . . . . .	125
10.2 Krossemiwariogramy . . . . .	125
10.3 Modelowanie krossemiwariogramów . . . . .	128
10.4 Kokriging . . . . .	129
<b>11 Wykorzystanie do estymacji danych uzupełniających</b>	<b>133</b>
11.1 Kriging stratyfikowany . . . . .	133
11.2 Prosty kriging ze zmiennymi średnimi lokalnymi (LVM) . . . . .	141
11.3 Kriging uniwersalny . . . . .	143
<b>12 Ocena jakości estymacji</b>	<b>151</b>
12.1 Statystyki jakości estymacji . . . . .	151
12.2 Walidacja wyników estymacji . . . . .	154
<b>13 Symulacje</b>	<b>161</b>
13.1 Symulacje geostatystyczne . . . . .	161
13.2 Typy symulacji . . . . .	161
13.3 Symulacje bezwarunkowe . . . . .	162
13.4 Symulacje warunkowe . . . . .	163
13.5 Sekwencyjna symulacja danych kodowanych . . . . .	170
<b>14 Źródła wiedzy</b>	<b>175</b>
14.1 Podstawy R . . . . .	175
14.2 Analizy przestrzenne w R . . . . .	175
14.3 Geostatystyka . . . . .	175

# Rozdział 1

## Wprowadzenie

Masz przed sobą skrypt zawierający materiały do ćwiczeń z geostatystyki. Składa się ona z kilkunastu rozdziałów pokazujących jak: dodawać i wizualizować dane przestrzenne w R (rozdział 2.1), wykonywać wstępna eksplorację danych nieprzestrzennych (rozdział 3), wstępnie analizować dane przestrzenne (rozdział 4), wykorzystywać deterministyczne metody interpolacji (rozdział 5), rozumieć i tworzyć przestrzenne miary podobieństwa i niepodobieństwa (rozdział 6), modelować semiwariogramy bezkierunkowe i kierunkowe (rozdział 7.1), tworzyć estymacje jednozmienne (rozdział 8), estymacje danych kodowanych (rozdział 9), estymacje wielozmienne (rozdział 10), estymacje wykorzystujące dane uzupełniające (rozdział 11), oceniać jakość wykonanych estymacji (rozdział 12) oraz budować symulacje przestrzenne (rozdział 13). Dodatkowo w rozdziale 14 można znaleźć odnośniki do innych materiałów związanych z geostatystyką i R. Wszystkie zaprezentowane przykłady zawierają również kod w języku R. Skrypt został stworzony w R (R Core Team, 2016) z wykorzystaniem pakietów `bookdown` (Xie, 2016a), `rmarkdown` (Allaire et al., 2016), `knitr` (Xie, 2016b) oraz programu Pandoc. Aktualna wersja skryptu znajduje się pod adresem <https://bookdown.org/nowosad/Geostatystyka/>.

Zachęcam do zgłoszania wszelkich uwag, błędów, pomysłów oraz komentarzy na adres mailowy nowosad@amu.edu.pl.

### 1.1 Wymagania wstępne

#### 1.1.1 Oprogramowanie

Do odtworzenia przykładów użytych w poniższym skrypcie wystarczy podstawowa znajomość R. Aby zainstalować R oraz RStudio można skorzystać z poniższych odnośników:

- R - <https://cloud.r-project.org/>
- RStudio - <https://www.rstudio.com/products/rstudio/download/>

Dodatkowo, użyte zostały poniższe pakiety R (from Jed Wing et al., 2016; Wei and Simko, 2016; Hijmans et al., 2016; Nychka et al., 2016; Wickham and Chang, 2016; Auguie, 2016; Pebesma and Graeler, 2016; Giraudeau, 2016; Hijmans, 2016; Perpinan Lamigueiro and Hijmans, 2016; Bivand et al., 2016; Bivand and Rundel, 2016; Pebesma and Bivand, 2016).

```
pakiety <- c('caret', 'corrplot', 'dismo', 'fields', 'ggplot2', 'gridExtra',
      'gstat', 'pgirmess', 'raster', 'rasterVis', 'rgdal', 'rgeos', 'sp')
```

Pakiety R używane w tym skrypcie można również zainstalować poprzez funkcję `install.packages()`:

```
install.packages(pakiety)
```

Lub też za pomocą pakietu `geostatbook` (Nowosad, 2016), który automatycznie zainstaluje wszystkie wymagane pakiety:

```
# install.packages("devtools")
devtools::install_github("nowosad/geostatbook")
```

### 1.1.2 Dane

Dane wykorzystywane w tym skrypcie można pobrać w postaci spakowanego archiwum (dla rozdziału 2.1) oraz korzystając z pakietu `geostatbook` (dla kolejnych rozdziałów). Dodatkowo, przy instalacji pakietu `geostatbook` pobierane są wszystkie inne pakiety potrzebne do pełnego korzystania z materiałów zawartych w skrypcie.

- Archiwum zawierające dane do rozdziału drugiego
- Dane do kolejnych rozdziałów są zawarte w pakiecie `geostatbook`:

```
# install.packages("devtools")
devtools::install_github("nowosad/geostatbook")
```

Aby ułatwić korzystanie ze skryptu, rozdziały od 3 do 13 rozpoczynają się od wczytania wymaganych pakietów oraz zbiorów danych.

## Rozdział 2

# R a dane przestrzenne

### 2.1 R a dane przestrzenne

#### 2.1.1 Pakiety

R zawiera wiele funkcji pozwalających na przetwarzanie, wizualizację i analizowanie danych przestrzennych. Zawarte są one w szeregu pakietów (zbiorów funkcji), między innymi:

- GIS - `sp`, `rgdal`, `raster`, `rasterVis`, `rgeos`, `maptools`, `GeoXp`, `deldir`, `pgirmess`, `spatstat`
- Geostatystyka - `gstat`, `geoR`, `geoRglm`, `fields`, `spBayes`, `RandomFields`, `vardiag`

Więcej szczegółów na ten temat pakietów R służących do analizy przestrzennej można znaleźć pod adresem <https://cran.r-project.org/web/views/Spatial.html>.

#### 2.1.2 Reprezentacja danych nieprzestrzennych

- Wektory (ang. *vector*):
  - liczbowe (ang. *integer*, *numeric*) - `c(1, 2, 3)` i `c(1.21, 3.32, 4.43)`
  - znakowe (ang. *character*) - `c('jeden', 'dwa', 'trzy')`
  - logiczne (ang. *logical*) - `c(TRUE, FALSE)`
  - czynnikowe (ang. *factor*) - `c('jeden', 'dwa', 'trzy', 'jeden')`
- Ramki danych (ang. *data.frame*) - to zbiór zmiennych (kolumn) oraz obserwacji (wierszy) zawierających różne typy danych
- Macierze (ang. *matrix*)
- Listy (ang. *list*)

#### 2.1.3 Reprezentacja danych przestrzennych

- Obiekty klasy `Spatial*` z pakietu `sp` - wszystkie z nich zawierają dwie dodatkowe informacje:
  - bounding box (`bbox`) - obwiednia - określa zasięg danych
  - CRS (`proj4string`) - układ współrzędnych
- Najczęściej stosowane obiekty klasy `Spatial*` to `SpatialPointsDataFrame`, `SpatialPolygonsDataFrame` oraz `SpatialGridDataFrame`
- Obiekty klasy `Raster*` z pakietu `raster`, tj. `RasterLayer`, `RasterStack`, `RasterBrick`
- Inne

#### 2.1.4 GDAL/OGR

- <http://www.gdal.org/>
- GDAL to biblioteka zawierająca funkcje służące do odczytywania i zapisywania danych w formatach rastrowych

- OGR to biblioteka służąca do odczytywania i zapisywania danych w formatach wektorowych
- Pakiet `rgdal` pozwala na wykorzystanie bibliotek GDAL/OGR w R

### 2.1.5 PROJ.4

- Dane przestrzenne powinny być zawsze powiązane z układem współrzędnych
- PROJ.4 - to biblioteka pozwalająca na identyfikację oraz konwersję pomiędzy różnymi układami współrzędnych
- Strona <http://www.spatialreference.org/> zawiera bazę bazę danych układów współrzędnych

### 2.1.6 EPSG

- Kod EPSG (ang. *European Petroleum Survey Group*) pozwala na łatwe identyfikowanie układów współrzędnych
- Przykładowo, układ PL 1992 może być określony jako:

```
'+proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000 +y_0=-5300000 +ellps=GRS80
+towgs84=0,0,0,0,0,0,0 +units=m +no_defs'
```

- ...lub też za pomocą kodu EPSG:

```
'+init=epsg:2180'
```

### 2.1.7 Układ geograficzny

- Proporcje pomiędzy współrzędną oznaczającą długość geograficzną (X) a oznaczającą szerokość geograficzną (Y) nie są równe 1:1
- Wielkość oczka siatki jest zmienna
- Nie pozwala to na proste określanie odległości czy powierzchni
- Jednostka mapy jest abstrakcyjna
- Powyższe cechy układów geograficznych powodują, że do większości algorytmów w geostatystyce wykorzystywane są układy współrzędnych prostokątnych płaskich

## 2.2 Import danych

R pozwala na odczytywanie danych przestrzennych z wielu formatów. Do najpopularniejszych należą dane z plików .csv, dane z plików .shp, oraz dane z plików w formacie geotiff.

### 2.2.1 Format .csv (dane punktowe)

Dane z plików tekstowych (np. .csv) można odczytać za pomocą uogólnionej funkcji `read.table()` lub też funkcji szczegółowych - `read.csv()` lub `read.csv2()`.

```
dane_punktowe <- read.csv('dane/punkty.csv')
```

```
head(dane_punktowe)
```

```
##      srtm  clc      temp      ndvi      savi       x       y
## 1 175.7430 1 13.852222 0.6158061 0.4189449 750298.0 716731.6
## 2 149.8111 1 15.484209 0.5558816 0.3794864 753482.9 717331.4
## 3 272.8583 NA 12.760814 0.6067462 0.3745572 747242.5 720589.0
## 4 187.2777 1 14.324648 0.3756170 0.2386246 755798.9 718828.1
## 5 260.1366 1 15.908549 0.4598393 0.3087599 746963.5 717533.5
## 6 160.1416 2 9.941118 0.5600288 0.3453627 756801.6 720474.1
```

Po wczytaniu za pomocą funkcji `read.csv()`, nowy obiekt (np. `dane_punktowe`) jest reprezentowany za pomocą klasy nieprzestrzennej `data.frame`. Aby obiekt został przetworzony do klasy przestrzennej, konieczne jest nadanie mu współrzędnych. W tym wypadku współrzędne znajdowały się w kolumnach `x` oraz `y`. Nadanie układu współrzędnych odbywa się poprzez funkcję `coordinates()`.

```
library('sp')
coordinates(dane_punktowe) <- ~x+y
summary(dane_punktowe)

## Object of class SpatialPointsDataFrame
## Coordinates:
##       min     max
## x 745592.5 756967.8
## y 712642.4 721238.7
## Is projected: NA
## proj4string : [NA]
## Number of points: 248
## Data attributes:
##      srtm          clc          temp          ndvi
## Min.   :146.5   Min.   :1.000   Min.   : 7.883   Min.   :0.2024
## 1st Qu.:191.5   1st Qu.:1.000   1st Qu.:12.003   1st Qu.:0.4636
## Median :217.9   Median :1.000   Median :14.941   Median :0.5154
## Mean   :214.9   Mean   :1.481   Mean   :15.273   Mean   :0.5047
## 3rd Qu.:239.5   3rd Qu.:2.000   3rd Qu.:17.630   3rd Qu.:0.5742
## Max.   :278.4   Max.   :4.000   Max.   :24.945   Max.   :0.6597
## NA's    :3       NA's    :5       NA's    :1       NA's    :1
##      savi
## Min.   :0.0824
## 1st Qu.:0.2935
## Median :0.3256
## Mean   :0.3176
## 3rd Qu.:0.3594
## Max.   :0.4404
## NA's    :1
```

Ważne, ale nie wymagane, jest także dodanie informacji o układzie przestrzennym danych za pomocą funkcji `proj4string()`.

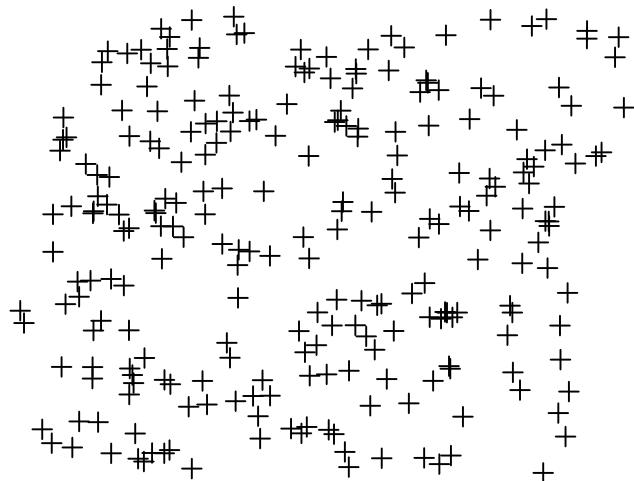
```
proj4string(dane_punktowe) <- '+init=epsg:2180'
summary(dane_punktowe)

## Object of class SpatialPointsDataFrame
## Coordinates:
##       min     max
## x 745592.5 756967.8
## y 712642.4 721238.7
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993
## +x_0=500000 +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0
## +units=m +no_defs]
## Number of points: 248
## Data attributes:
##      srtm          clc          temp          ndvi
## Min.   :146.5   Min.   :1.000   Min.   : 7.883   Min.   :0.2024
## 1st Qu.:191.5   1st Qu.:1.000   1st Qu.:12.003   1st Qu.:0.4636
## Median :217.9   Median :1.000   Median :14.941   Median :0.5154
## Mean   :214.9   Mean   :1.481   Mean   :15.273   Mean   :0.5047
## 3rd Qu.:239.5   3rd Qu.:2.000   3rd Qu.:17.630   3rd Qu.:0.5742
## Max.   :278.4   Max.   :4.000   Max.   :24.945   Max.   :0.6597
```

```
##  NA's    :3      NA's    :5      NA's    :1      NA's    :1
##      savi
##  Min.   :0.0824
##  1st Qu.:0.2935
##  Median :0.3256
##  Mean   :0.3176
##  3rd Qu.:0.3594
##  Max.   :0.4404
##  NA's    :1
```

Proste wyświetlenie uzyskanych danych klasy przestrzennej, np. `SpatialPointsDataFrame`, można uzyskać za pomocą funkcji `plot()`.

```
plot(dane_punktowe)
```



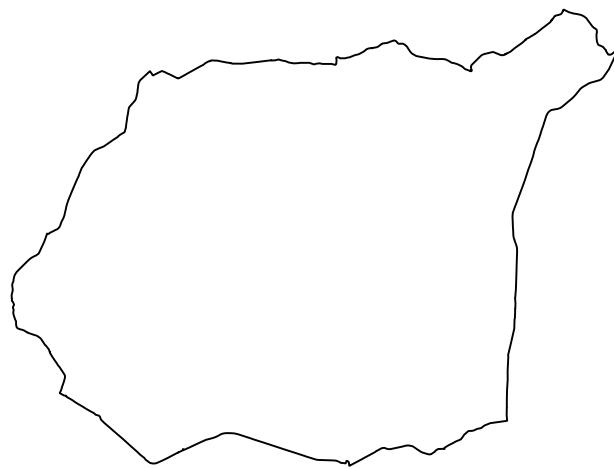
## 2.2.2 Dane poligonalne (formaty gisowe)

Dane wektorowe (np. shapefile) można odczytać za pomocą funkcji `readOGR()` z pakietu `rgdal`. Dla danych w formacie shapefile, przyjmuje ona co najmniej dwa argumenty - `dsn` oraz `layer`. Argument `dsn` określa folder, w którym znajdują się dane. W przypadku, gdy dane znajdują się w folderze roboczym należy ten argument określić za pomocą znaku kropki ('.'). Argument `layer` wymaga podania nazwy pliku bez jego rozszerzenia. Przykładowo, gdy pliki nazywają się `granica.dbf`, `granica.prj`, `granica.shp`, oraz `granica.shx` - konieczne jest podanie w argumencie `layer` jedynie nazwy `granica`.

```
library('rgdal')
granica <- readOGR(dsn='dane', layer='granica')
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "dane", layer: "granica"
## with 1 features
## It has 3 fields
```

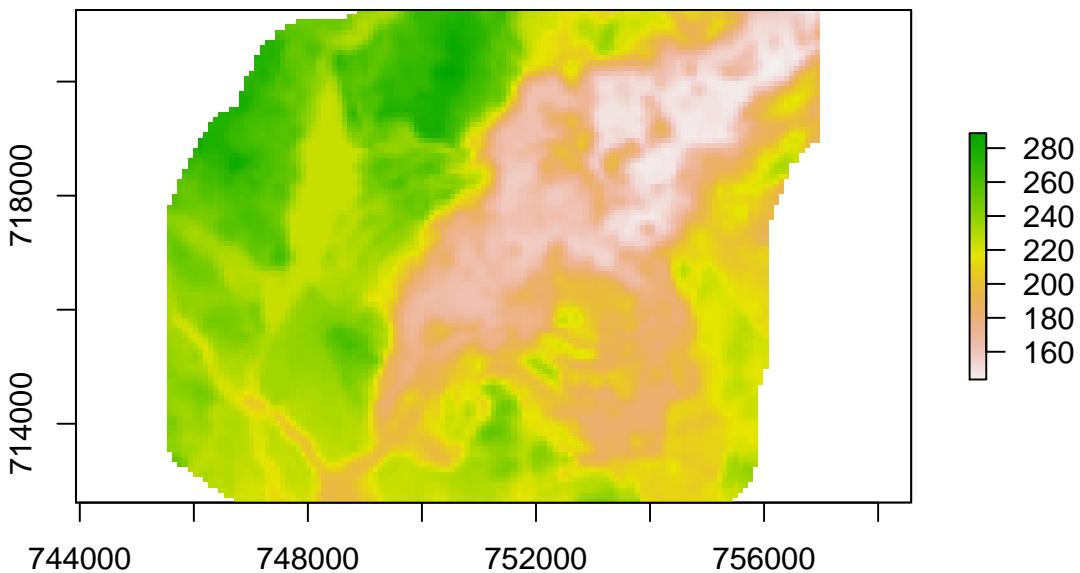
```
plot(granica)
```



### 2.2.3 Rastry

Istnieje kilka sposobów odczytu danych rasterowych w R. Do najpopularniejszych należą funkcje `readGDAL()` z pakietu `rgdal` oraz `raster()` z pakietu `raster`. W tym drugim przypadku należy jedynie podać ścieżkę do pliku rasterowego.

```
library('raster')
siatka_raster <- raster('dane/siatka.tif')
plot(siatka_raster)
```



## 2.3 Eksport danych

### 2.3.1 Zapisywanie danych wektorowych

R pozwala również na zapisywanie danych przestrzennych. W przypadku zapisu danych wektorowych za pomocą funkcji `writeOGR()` konieczne jest podanie nazwy zapisywanej obiektu (np. `poligon`), folderu w którym chcemy zapisać plik (np. `nazwa_folderu`), nazwę zapisywanych plików bez rozszerzenia (np. `nowy_poligon`), oraz sterownika - w przypadku danych shapefile jest to `ESRI Shapefile`.

```
writeOGR(poligon, dsn='nazwa_folderu', layer='nowy_poligon', driver='ESRI Shapefile')
```

### 2.3.2 Zapisywanie danych rastrowych

Najprostszym sposobem zapisania danych rastrowych jest użycie funkcji `writeRaster()`. Wymaga ona podania dwóch argumentów - nazwy zapisywanej obiektu (np. `siatka_raster`) oraz ścieżki i nazwy nowego pliku wraz z rozszerzeniem (np. `nazwa_folderu/nowy_raster.tif`).

```
writeRaster(siatka_raster, filename='nazwa_folderu/nowy_raster.tif')
```

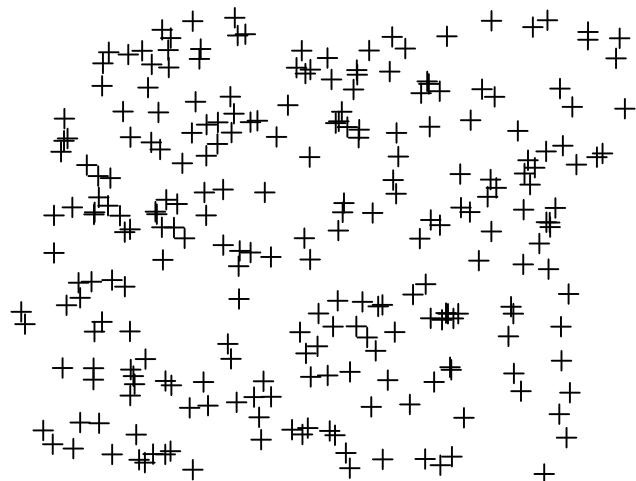
## 2.4 Wizualizacja danych 2D

Do wizualizacji danych przestrzennych w R służy co najmniej kilkanaście różnych pakietów. Poniżej pokazane są przykłady kilku najprostszych funkcji - `plot()` oraz `spplot()` z pakietu `sp` oraz `levelplot()` z pakietu `rasterVis`.

### 2.4.1 Dane punktowe

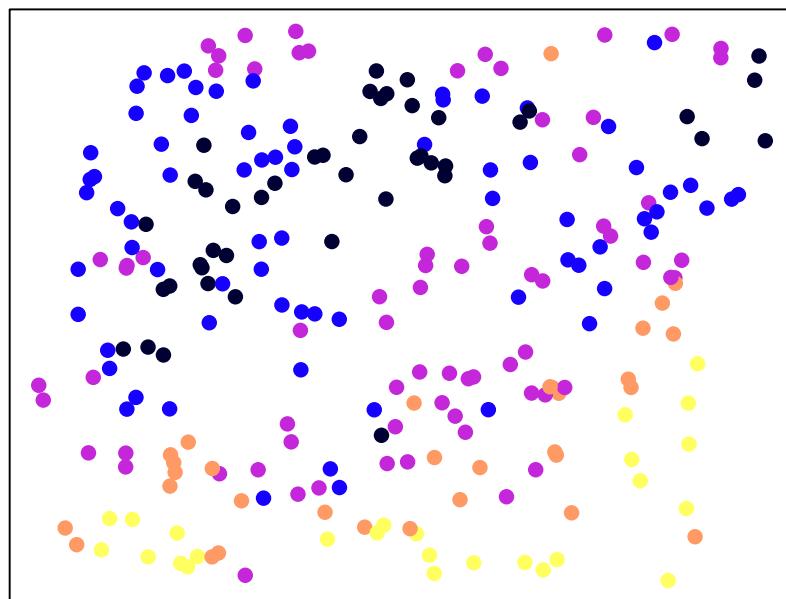
Funkcja `plot()` idealnie nadaje się do szybkiego przyjrzenia się, np. rodzajowi próbkowania danych.

```
plot(dane_punktowe)
```



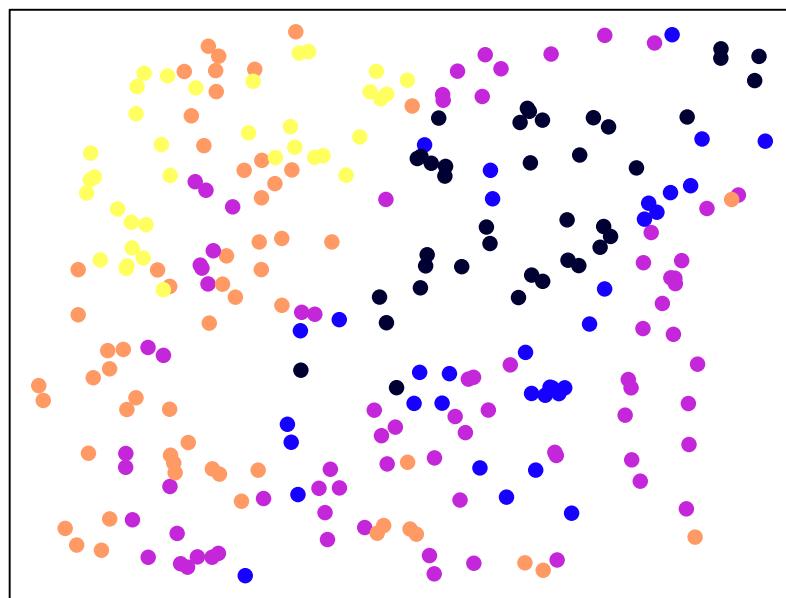
Funkcja `spplot()` w prosty sposób pozwala na obejrzenie rozkładu wartości wybranej zmiennej. Należy w niej podać nazwę obiektu oraz nazwę wyświetlanej zmiennej. Poniżej można zobaczyć przykłady dla zmiennych `temp` oraz `srtm`.

```
spplot(dane_punktowe, 'temp')
```



- [7.883, 11.3]
- (11.3, 14.71]
- (14.71, 18.12]
- (18.12, 21.53]
- (21.53, 24.94]

```
spplot(dane_punktowe, 'srtm')
```

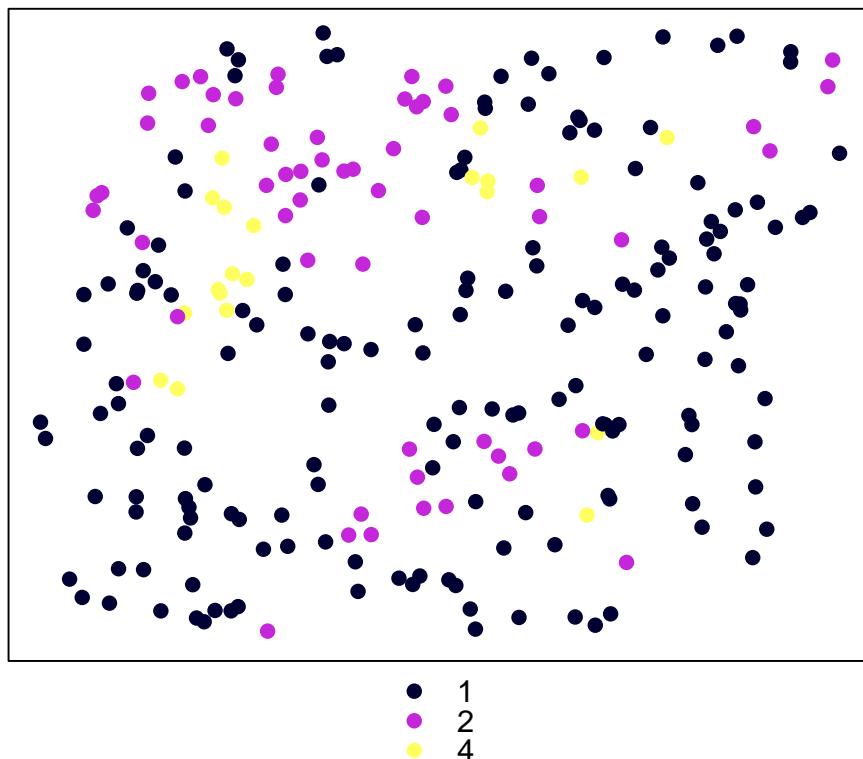


- [146.5, 172.9]
- (172.9, 199.3]
- (199.3, 225.6]
- (225.6, 252]
- (252, 278.4]

### 2.4.2 Dane punktowe - kategorie

Nie zawsze dane mają postać ciągły wartości - bywają one również określeniami różnych klas. W takich sytuacjach należy wcześniej przetworzyć typ danych do postaci kategorycznej (`as.factor()`). Następnie można je wyświetlić za pomocą funkcji `spplot()`.

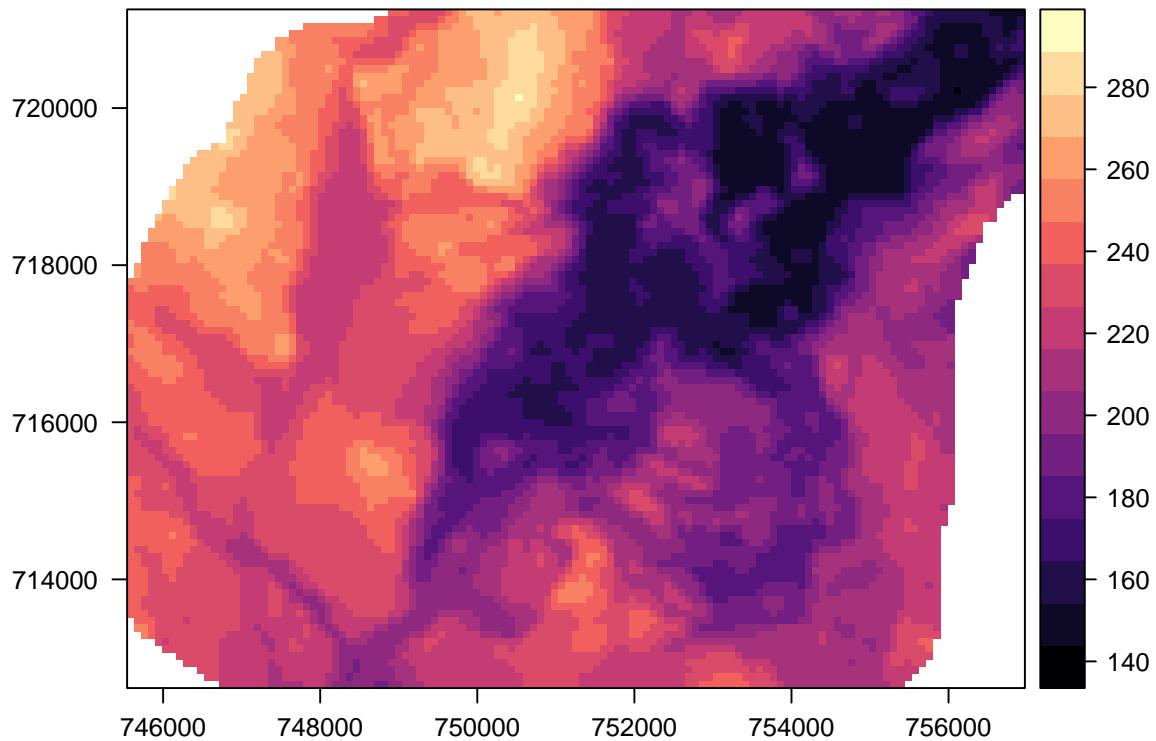
```
dane_punktowe@data$clc <- as.factor(dane_punktowe@data$clc)
spplot(dane_punktowe, 'clc')
```



### 2.4.3 Rastry

Wyświetlanie danych w formacie rastrowym może odbyć się z użyciem funkcji `levelplot()`. Wymaga ona jedynie zdefiniowania obiektu do wizualizacji. W poniższym przypadku dodatkowo ustawiono argument `margin=FALSE` co wyłącza wyświetlanie tzw. histogramów marginalnych (ang. *marginal histogram*).

```
library('rasterVis')
levelplot(siatka_raster, margin=FALSE)
```



## 2.5 Tworzenie siatek

W większości przypadków analiz geostatystycznych konieczne jest stworzenie siatki interpolacyjnej. Istnieją dwa podstawowe rodzaje takich siatek - siatki regularne oraz siatki nieregularne.

### 2.5.1 Siatki regularne

Siatki regularne mają kształt prostokąta obejmującego cały analizowany obszar. Określenie granic obszaru można wykonać na podstawie zasięgu danych punktowych za pomocą funkcji `bbox()` z pakietu `sp` lub funkcji `extent()` z pakietu `raster`.

```
bbox(dane_punktowe)
```

```
##      min      max
## x 745592.5 756967.8
## y 712642.4 721238.7
```

```
extent(dane_punktowe)
```

```
## class       : Extent
## xmin       : 745592.5
## xmax       : 756967.8
## ymin       : 712642.4
## ymax       : 721238.7
```

Do stworzenia siatki można użyć funkcji `expand.grid()`. Wymaga ona określenia dwóch argumentów - `x` oraz `y` (taka ich nazwa nie jest obowiązkowa). Oba argumenty przyjmują trzy wartości: (i) `from` oznaczający wartość początkową współrzędnej, (ii) `to` określający wartość końcową współrzędnej, oraz (iii) `by` określający rozdzielczość. Przy ustalaniu wartości początkowej i końcowej konieczne jest ich rozszerzenie względem wartości z funkcji `bbox()` lub `extent()`, aby wszystkie analizowane punkty znalazły się na badanym obszarze.

```
siatka <- expand.grid(x = seq(from = 745050, to = 757050, by = 500),
                      y = seq(from = 712650, to = 721650, by = 500))
```

Utworzony w ten sposób obiekt wymaga określenia współrzędnych (funkcja `coordinates()`), potwierdzenia że dane mają być siatką (funkcja `gridded()`), oraz przypisania układu współrzędnych z obiektu punktowego (funkcja `proj4string()`).

```
coordinates(siatka) <- ~x + y
gridded(siatka) <- TRUE
proj4string(siatka) <- proj4string(dane_punktowe)
```

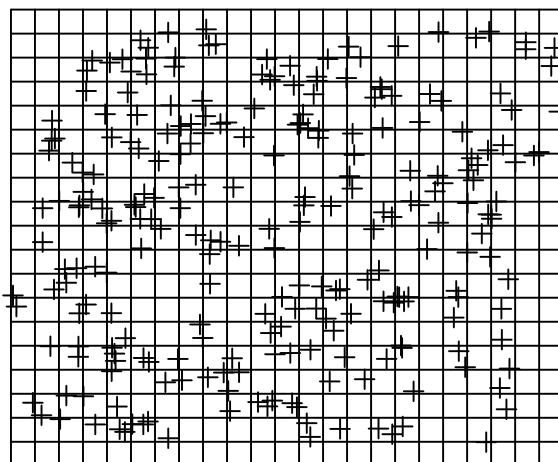
Alternatywnie, do stworzenia siatki można wykorzystać funkcję `makegrid()`. Tworzy ona nowy obiekt na podstawie istniejącego obiektu punktowego oraz zadanej rozdzielczości.

```
siatka <- makegrid(dane_punktowe, cellsize=500)
names(siatka) <- c('x', 'y')
coordinates(siatka) <- ~x + y
gridded(siatka) <- TRUE
proj4string(siatka) <- proj4string(dane_punktowe)
```

## 2.5.2 Siatki regularne

Sprawdzenie, czy uzyskana siatka oraz dane punktowe się na siebie nakładają można sprawdzić za pomocą funkcji `plot()`. W poniższym przykładzie, pierwszy wiersz służy wyświetleniu siatki, a drugi dodaje dane punktowe z użyciem argumentu `add`.

```
plot(siatka)
plot(dane_punktowe, add=TRUE)
```



### 2.5.3 Siatki nieregularne - klasa `RasterLayer`

Siatki nieregularne mają zazwyczaj kształt wieloboku obejmującego analizowany obszar. Mogą one powstać, np. w oparciu o wcześniej istniejące granice. Siatki nieregularne w R mają zazwyczaj klasę `RasterLayer` lub `SpatialPixelsDataFrame`. Pierwsza z klas jest elastyczniejsza i prostsza w użyciu, podczas gdy druga jest lepiej wspierana przez funkcje geostatystyczne, np. z pakietu `gstat`.

W poniższym przypadku odczytywana jest granica badanego obszaru z pliku shapefile. Taki obiekt można np. stworzyć za pomocą oprogramowania gisowego takiego jak QGIS. Następnie na podstawie tego obiektu tworzony jest obiekt klasy `RasterLayer`, a za pomocą funkcji `res()` definiowana jest jego rozdzielcość. W kolejnym kroku ustala się wartość wszystkich oczek siatki na zero, oraz poprzez funkcję `proj4string()` ujednolica się definicję układu współrzędnych siatki. Ostatnim krokiem jest przycięcie siatki do nieregularnego obszaru z użyciem funkcji `mask()`.

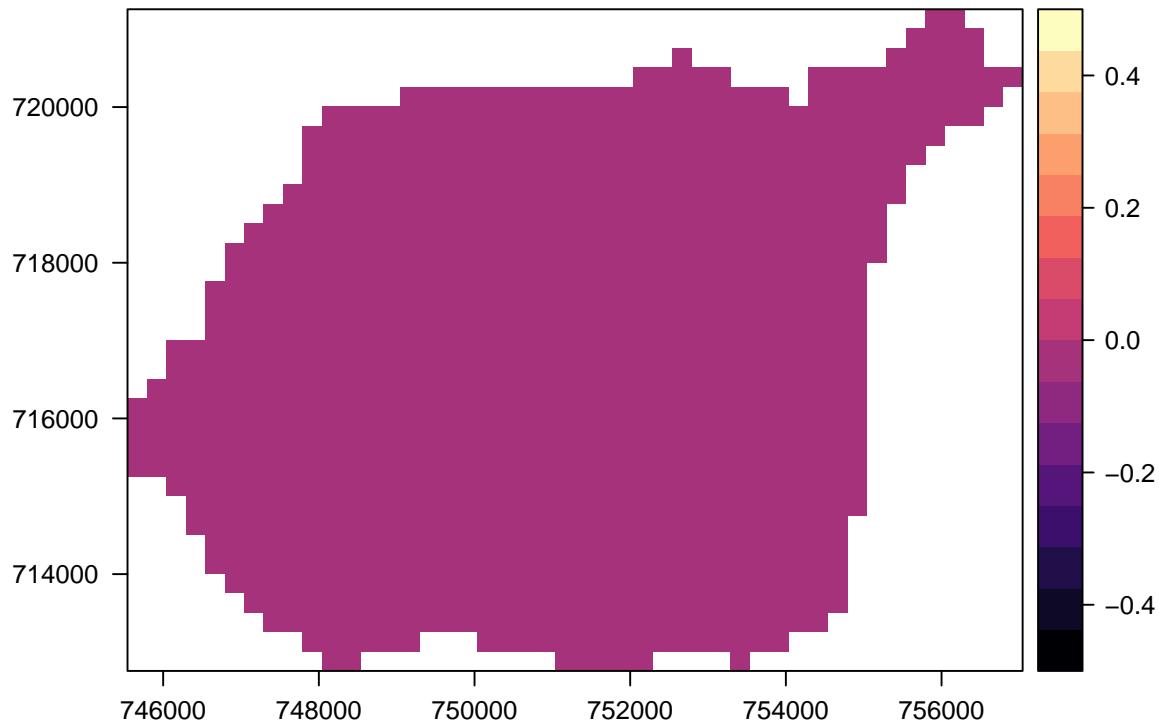
```
granica <- readOGR(dsn='dane', layer='granica')
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "dane", layer: "granica"
## with 1 features
## It has 3 fields

siatka_n <- raster(extent(granica))
res(siatka_n) <- c(250, 250)
siatka_n[] <- 0
proj4string(siatka_n) <- proj4string(granica)
siatka_n <- mask(siatka_n, granica)
```

Wynik przetworzenia można zobaczyć z użyciem funkcji `levelplot`.

```
levelplot(siatka_n, margin=FALSE)
```



#### 2.5.4 Siatki nierregularne - klasa SpatialPixelsDataFrame

Nieregularną siatkę klasy RasterLayer łatwo jest przetworzyć do klasy SpatialPixelsDataFrame.

```
siatka_n <- as(siatka_n, 'SpatialPointsDataFrame')
siatka_n <- siatka_n[!is.na(siatka_n@data$layer), ]
gridded(siatka_n) <- TRUE
plot(siatka_n)
```





# Rozdział 3

## Eksploracyjna analiza danych nieprzestrzennych

```
library('sp')
library('ggplot2')
library('corrplot')
library('geostatbook')
data(punkty)
data(granica)
```

### 3.1 Cele eksploracyjnej analizy danych

Zazwyczaj przed przystąpieniem do analiz geostatystycznych koniecznie jest wykonanie eksploracyjnej analizy danych nieprzestrzennych. Jej ogólne cele obejmują:

- Stworzenie ogólnej charakterystyki danych oraz badanego zjawiska
- Określenie przestrzennego/czasowego typu próbkowania
- Uzyskanie informacji o relacji pomiędzy lokalizacją obserwacji a czynnikami wpływającymi na zmienność przestrzenną badanych cech

### 3.2 Dane

#### 3.2.1 Dane

Nie istnieje jedyna, optymalna ścieżka eksploracji danych. Proces ten różni się w zależności od posiadanej zbioru danych, jak i od postawionego pytania. Warto jednak, aby jednym z pierwszych kroków było przyjrzenie się danym wejściowym. Pozwala na to, między innymi, funkcja `str()`. Przykładowo, dla obiektu klasy `SpatialPointsDataFrame` wyświetla ona szereg ważnych informacji. Oprócz klasy można również odczytać, że obiekt `punkty` zawiera pięć elementów (ang. *slots*) rozpoczynających się od symbolu `@`. Pierwszy z nich, `@data` jest obiektem klasy `data.frame` zawierającym informacje o kolejnych punktach w tym zbiorze. Element `@coords.nrs` określa numer kolumn zawierających współrzędne w oryginalnym zbiorze danych. W poniższym przypadku żadna kolumna nie ma takich informacji. Kolejny element, `@coords` definiuje współrzędne kolejnych punktów w obiekcie `punkty`. Ostatnie dwa elementy, `@bbox` i `@proj4string` określają kolejno zasięg przestrzenny danych oraz definicję ich układu współrzędnych.

```
str(punkty)
```

```
## Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
##   ..@ data      : 'data.frame': 250 obs. of 5 variables:
##     ...$ srtm: num [1:250] 231 185 269 212 209 ...
```

```

## ...$ clc : num [1:250] 1 1 1 1 2 2 2 2 1 1 ...
## ...$ temp: num [1:250] 22.5 16 16.1 17.1 22.3 ...
## ...$ ndvi: num [1:250] 0.589 0.546 0.509 0.529 0.491 ...
## ...$ savi: num [1:250] 0.357 0.382 0.342 0.345 0.22 ...
## ..@ coords.nrs : num(0)
## ..@ coords      : num [1:250, 1:2] 753638 749498 750131 751764 753676 ...
## ...- attr(*, "dimnames")=List of 2
## ... .$. : NULL
## ... .$. : chr [1:2] "x" "y"
## ..@ bbox       : num [1:2, 1:2] 745547 712619 756937 721193
## ...- attr(*, "dimnames")=List of 2
## ... .$. : chr [1:2] "x" "y"
## ... .$. : chr [1:2] "min" "max"
## ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
## ...@ projargs: chr "+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000 +y_0=--"

```

Oczywiście, każdy z elementów można obejrzeć indywidualnie, poprzez połączenie nazwy obiektu, znaku @, oraz nazwy elementu. Przykładowo `punkty@data` pozwala na obejrzenie tabeli atrybutów zbioru punktowego.

```
str(punkty@data)
```

```

## 'data.frame':   250 obs. of  5 variables:
## $ srtm: num  231 185 269 212 209 ...
## $ clc : num  1 1 1 1 2 2 2 2 1 1 ...
## $ temp: num  22.5 16 16.1 17.1 22.3 ...
## $ ndvi: num  0.589 0.546 0.509 0.529 0.491 ...
## $ savi: num  0.357 0.382 0.342 0.345 0.22 ...

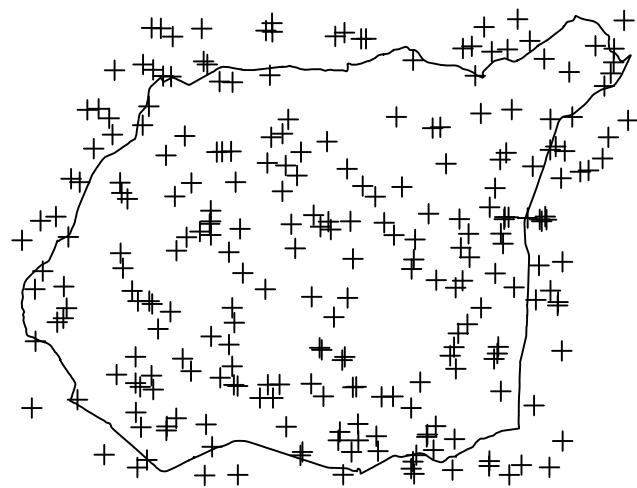
```

Istnieją dwa sposoby uzyskania wartości wybranej zmiennej z tabeli atrybutów.

```
punkty@data$temp #1
punkty$temp      #2
```

Z uwagi na to, że zajmujemy się danymi przestrzennymi - warto też się upewnić czy dane zostały poprawnie wczytane i obejmują analizowany obszar. Najszybciej można to wykonać z pomocą funkcji `plot()`.

```
plot(punkty)
plot(granica, add=TRUE)
```



## 3.3 Statystyki opisowe

### 3.3.1 Statystyki opisowe

Podstawową funkcją w R służącą wyliczaniu podstawowych statystyk jest `summary()`. Dla zmiennych numerycznych wyświetla ona wartość minimalną, pierwszy kwartyl, medianę, średnią, trzeci kwartyl, oraz wartość maksymalną.

```
summary(punkty@data)
```

```
##      srtm          clc          temp          ndvi
##  Min.   :146.2   Min.   :1.000   Min.   : 8.706   Min.   :0.2102
##  1st Qu.:191.0   1st Qu.:1.000   1st Qu.:13.284   1st Qu.:0.4567
##  Median :218.2   Median :1.000   Median :15.309   Median :0.5037
##  Mean   :213.4   Mean   :1.268   Mean   :15.950   Mean   :0.5039
##  3rd Qu.:236.4   3rd Qu.:1.000   3rd Qu.:18.273   3rd Qu.:0.5521
##  Max.   :278.8   Max.   :4.000   Max.   :26.139   Max.   :0.6848
##      savi
##  Min.   :0.08343
##  1st Qu.:0.29017
##  Median :0.32212
##  Mean   :0.31847
##  3rd Qu.:0.35292
##  Max.   :0.48354
```

### 3.3.2 Statystyki opisowe | średnia i mediana

Do określenia wartości przeciętnej zmiennych najczęściej stosuje się medianę i średnią.

```
median(punkty$temp, na.rm=TRUE)
## [1] 15.30944
mean(punkty$temp, na.rm=TRUE)
## [1] 15.95036
```

- W wypadku symetrycznego rozkładu te dwie cechy są równe
- Średnia jest bardziej wrażliwa na wartości odstające
- Mediana jest lepszą miarą środka danych, jeżeli są one skośne

Po co używać średniej?

- Bardziej przydatna w przypadku małych zbiorów danych
- Gdy rozkład danych jest symetryczny
- (Jednak) często warto podawać obie miary

### 3.3.3 Statystyki opisowe | minimum i maksimum

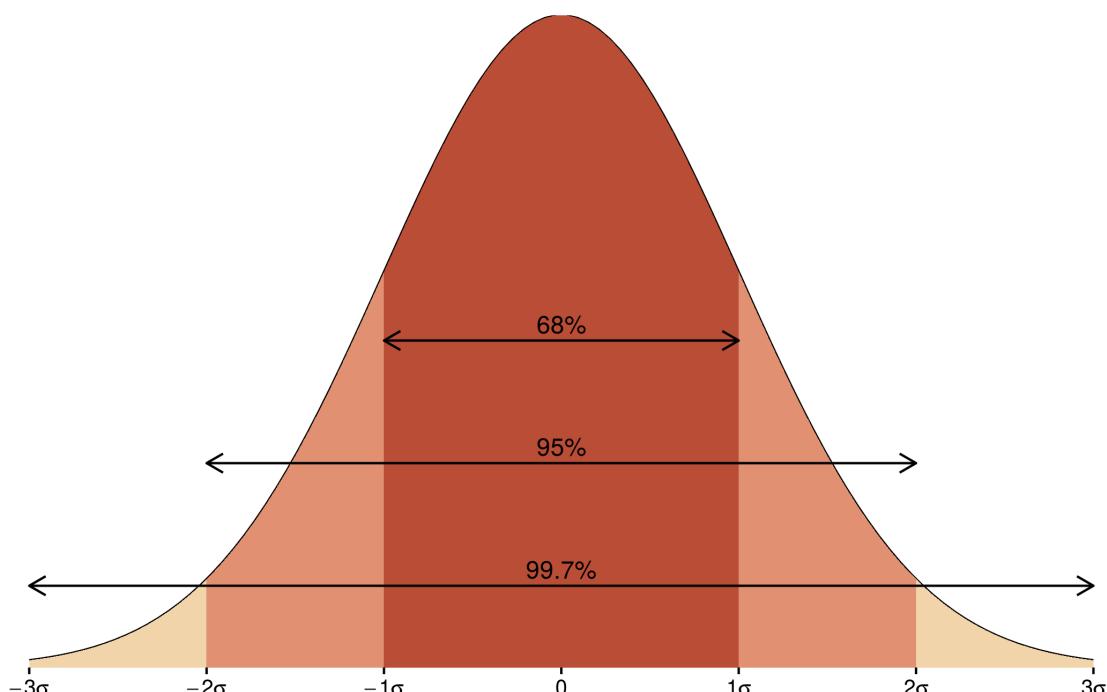
Minimalna i maksymalna wartość zmiennej służy do określenia ekstremów w zbiorze danych, jak i sprawdzenia zakresu wartości.

```
min(punkty$temp, na.rm=TRUE)
## [1] 8.706485
max(punkty$temp, na.rm=TRUE)
## [1] 26.13947
```

### 3.3.4 Statystyki opisowe | odchylenie standardowe

Dodatkowo, często używaną statystyką jest odchylenie standardowe. Wartość ta określa w jak mocno wartości zmiennej odstają od średniej. Dla rozkładu normalnego ta wartość ma znane własności:

- 68% obserwacji mieści się w granicach jednego odchylenia standardowego od średniej
- 95% obserwacji mieści się w granicach dwóch odchyleń standardowych od średniej
- 99,7% obserwacji mieści się w granicach trzech odchyleń standardowych od średniej



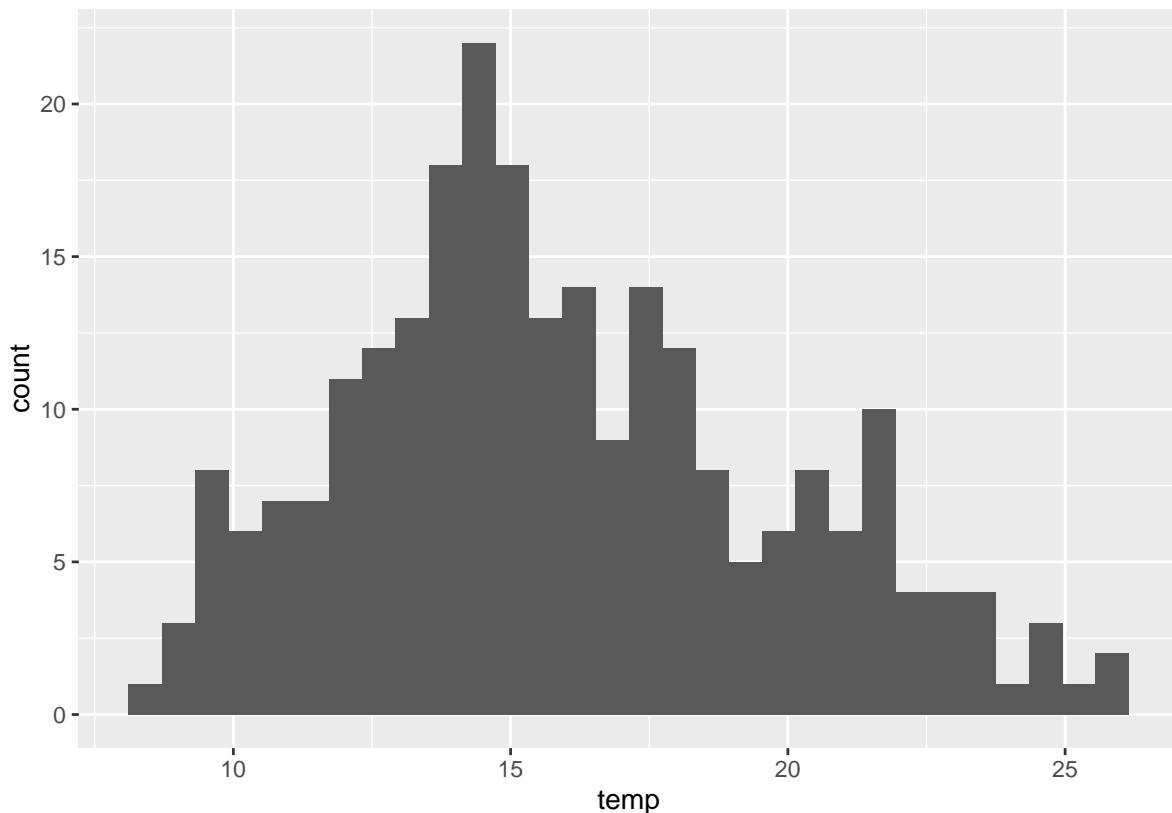
```
sd(punkty$temp, na.rm=TRUE)
## [1] 3.839596
```

## 3.4 Wykresy

### 3.4.1 Histogram

Histogram należy do typów wykresów najczęściej używanych w eksploracyjnej analizie danych.

```
ggplot(punkty@data, aes(temp)) + geom_histogram()
```

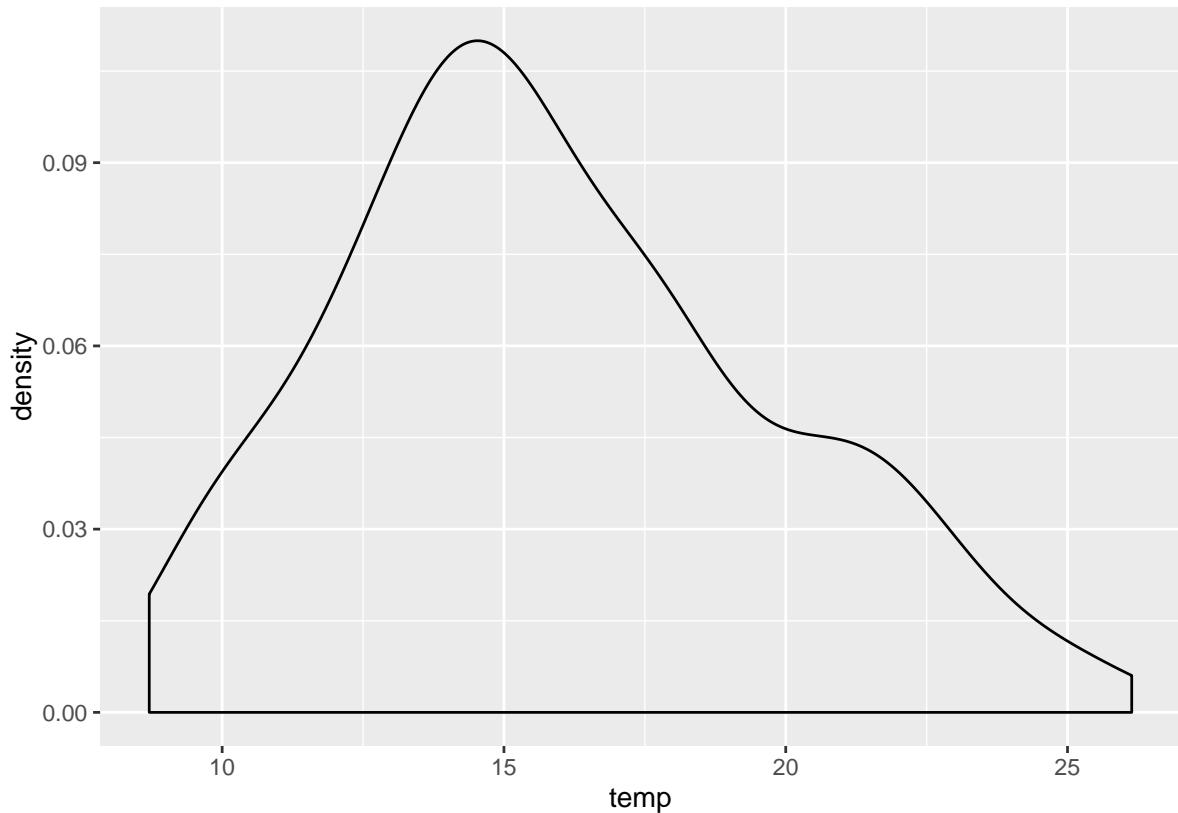


- Stworzony przez Karla Pearsona
- Jest graficzną reprezentacją rozkładu danych
- Wartości danych są łączone w przedziały (na osi poziomej) a na osi pionowej jest ukazana liczba punktów (obserwacji) w każdym przedziale
- Różny dobór przedziałów może dawać inną informację
- W pakiecie ggplot2, domyślnie przedział jest ustalany poprzez podzielenie zakresu wartości przez 30

### 3.4.2 Estymator jądrowy gęstości (ang. *kernel density estimation*)

Podobną funkcję do histogramu spełnia estymator jądrowy gęstości. Przypomina on wygładzony wykres histogramu i również służy graficznej reprezentacji rozkładu danych.

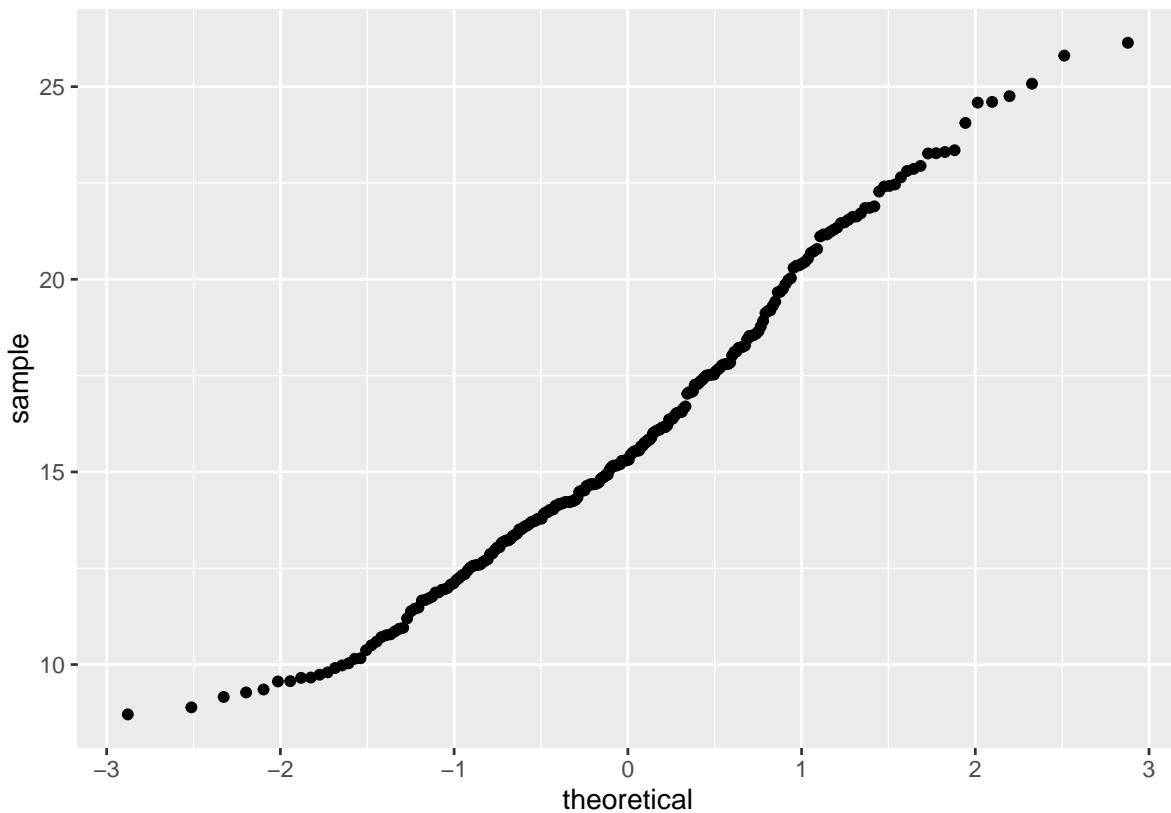
```
ggplot(punkty@data, aes(temp)) + geom_density()
```



### 3.4.3 Wykres kwantyl-kwantyl (ang.*quantile-quantile*)

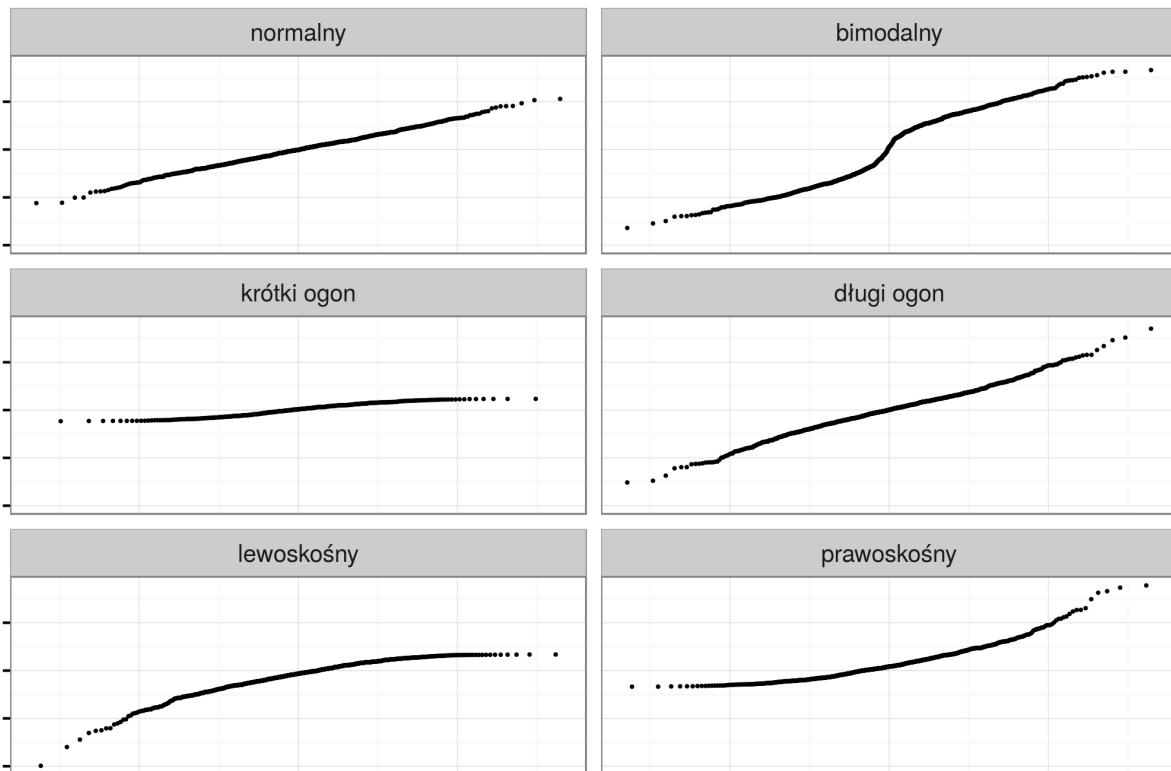
Wykres kwantyl-kwantyl ułatwia interpretację rozkładu danych.

```
ggplot(punkty@data, aes(sample=temp)) + stat_qq()
```



Na poniższej rycinie można zobaczyć przykłady najczęściej spotykanych cech rozkładu danych w wykresach kwantyl-kwantyl.

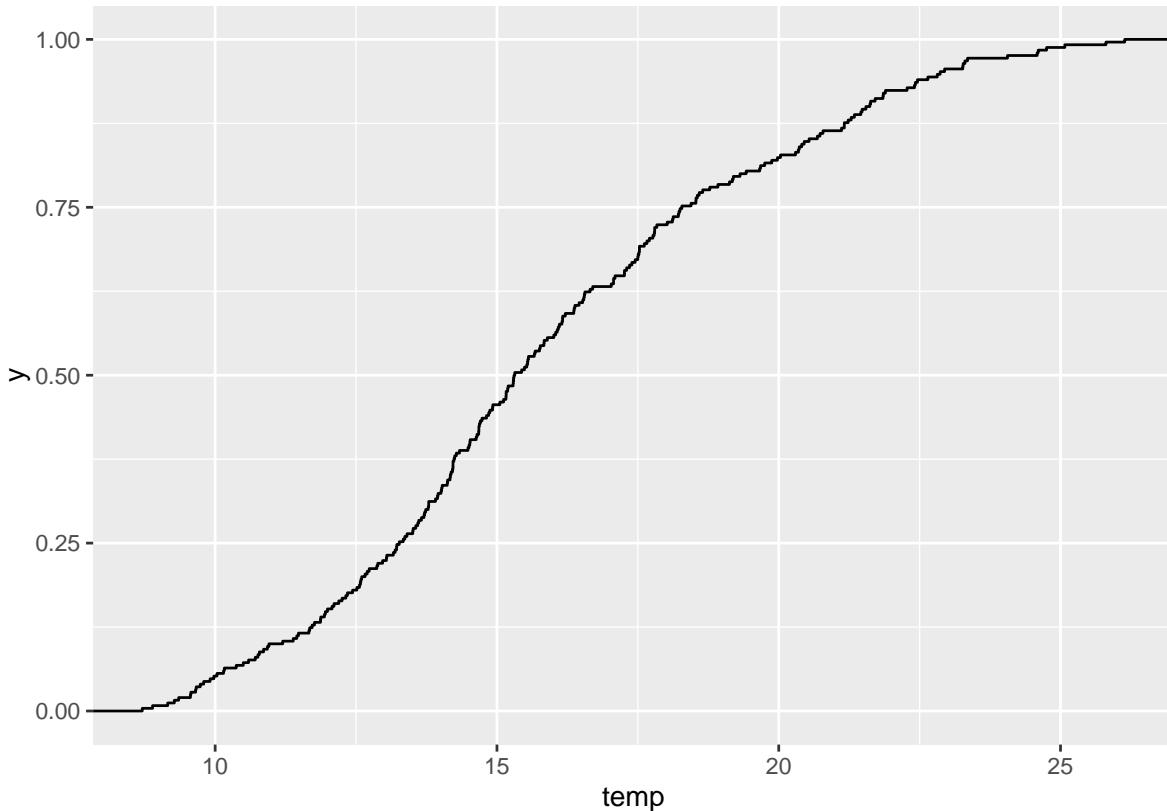
#### Interpretacja wykresu kwantyl-kwantyl



### 3.4.4 Dystrybuanta (CDF)

Dystrybuanta (ang. *conditional density function* - CDF) wyświetla prawdopodobieństwo, że wartość zmiennej przewidywanej jest mniejsza lub równa określonej wartości

```
ggplot(punkty@data, aes(temp)) + stat_ecdf()
```



## 3.5 Porównanie zmiennych

Wybór odpowiedniej metody porównania zmiennych zależy od szeregu cech, między innymi liczby zmiennych, ich typu, rozkładu wartości, etc.

### 3.5.1 Kowariancja

Kowariancja jest nieunormowaną miarą zależności liniowej pomiędzy dwiema zmiennymi. Kowariancja dwóch zmiennych  $x$  i  $y$  pokazuje jak dwie zmienne są ze sobą liniowo powiązane. Dodatnia kowariancja wskazuje na pozytywną relację liniową pomiędzy zmiennymi, podczas gdy ujemna kowariancja świadczy o odwrotnej sytuacji. Jeżeli zmienne nie są ze sobą liniowo powiązane, wartość kowariacji jest bliska零. Inaczej mówiąc, kowariancja stanowi miarę wspólnej zmienności dwóch zmiennych. Wielkość samej kowariacji uzależniona jest od przyjętej skali zmiennej (jednostki). Inne wyniku uzyskamy (przy tej samej zależności pomiędzy parą zmiennych), gdy będziemy analizować wyniki np. wysokości terenu w metrach i temperatury w stopniach Celsjusza a inne dla wysokości terenu w metrach i temperatury w stopniach Fahrenheita. Do wyliczania kowariacji w R służy funkcja `cov()`.

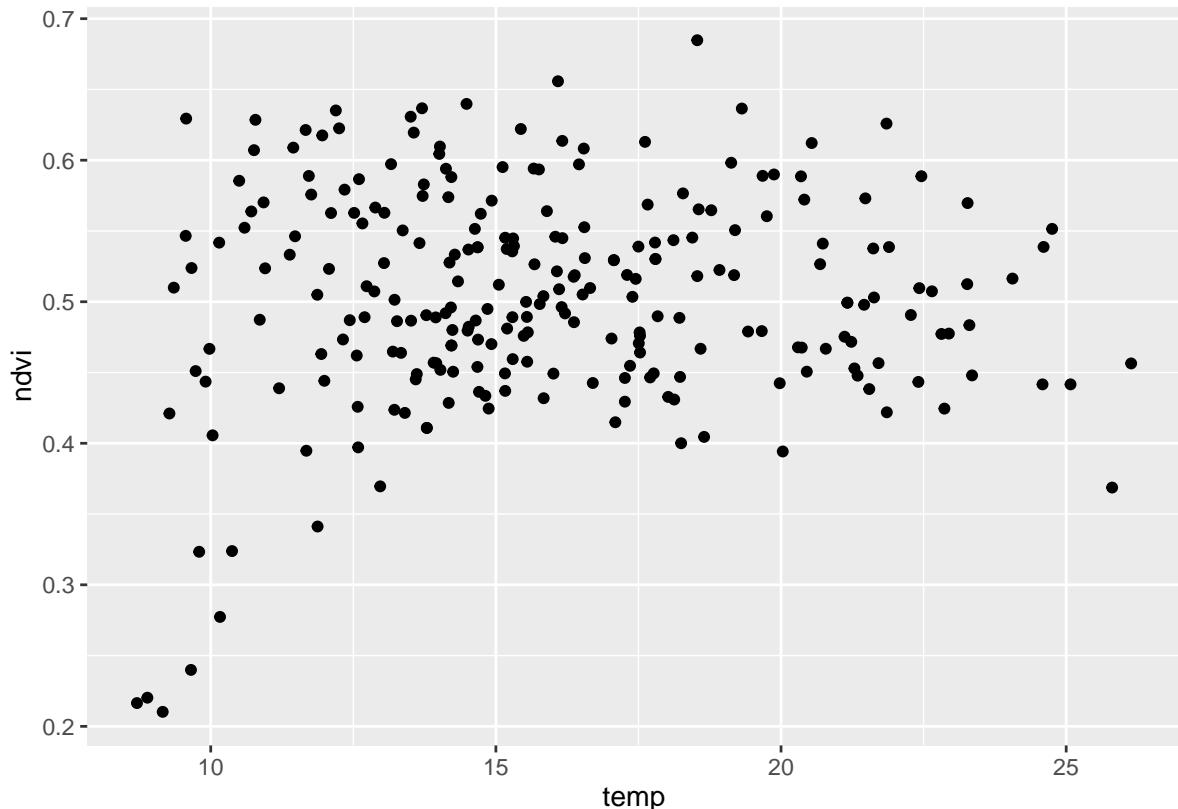
```
cov(punkty$temp, punkty$ndvi)
```

```
## [1] 0.02047509
```

### 3.5.2 Współczynnik korelacji

Współczynnik korelacji to unormowana miara zależności pomiędzy dwiema zmiennymi, przyjmująca wartości od -1 do 1. Ten współczynnik jest uzyskiwany poprzez podzielenie wartości kowariancji przez odchylenie standardowe wyników. Z racji unormowania nie jest ona uzależniona od jednostki. Korelację można wyliczyć dzięki funkcji `cor()`. Działa ona zarówno w przypadku dwóch zmiennych numerycznych, jak i całego obiektu zawierającego zmienne numeryczne.

```
ggplot(punkty@data, aes(temp, ndvi)) + geom_point()
```



```
cor(punkty$temp, punkty$ndvi)
```

```
## [1] 0.07049136
```

```
cor(punkty@data[c(1, 3:5)])
```

```
##           srtm      temp      ndvi      savi
## srtm  1.00000000 -0.060458939 0.05230326  0.035760652
## temp  -0.06045894  1.000000000 0.07049136 -0.007748702
## ndvi   0.05230326  0.070491358 1.000000000  0.889037014
## savi   0.03576065 -0.007748702 0.889037011 1.000000000
```

Dodatkowo funkcja `cor.test()` służy do testowania istotności korelacji. Za pomocą argumentu `method` można również wybrać jedną z trzech dostępnych miar korelacji.

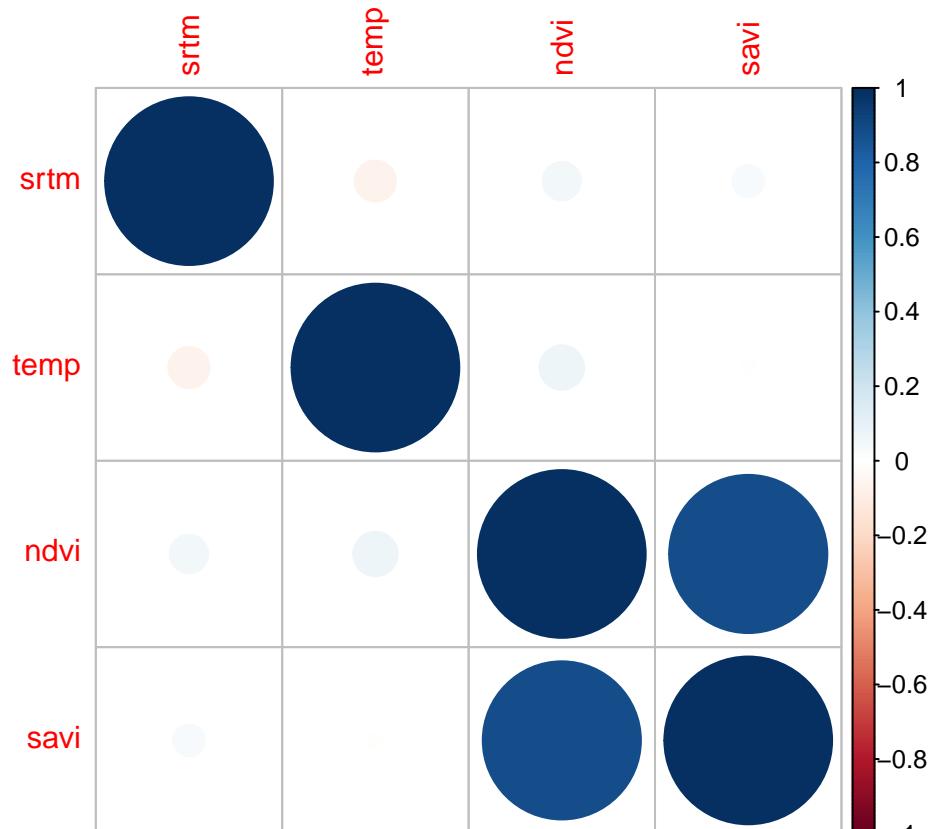
```
cor.test(punkty$temp, punkty$ndvi)
```

```
##
##  Pearson's product-moment correlation
##
## data:  punkty$temp and punkty$ndvi
## t = 1.1129, df = 248, p-value = 0.2668
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.05404834  0.19287158
```

```
## sample estimates:
##      cor
## 0.07049136
```

W przypadku posiadania co najmniej kilku par zmiennych, można również skorzystać z funkcji `corrplot()` z pakietu `corrplot`. Wyświetla ona wykres pokazujący wartości korelacji pomiędzy kolejnymi zmiennymi.

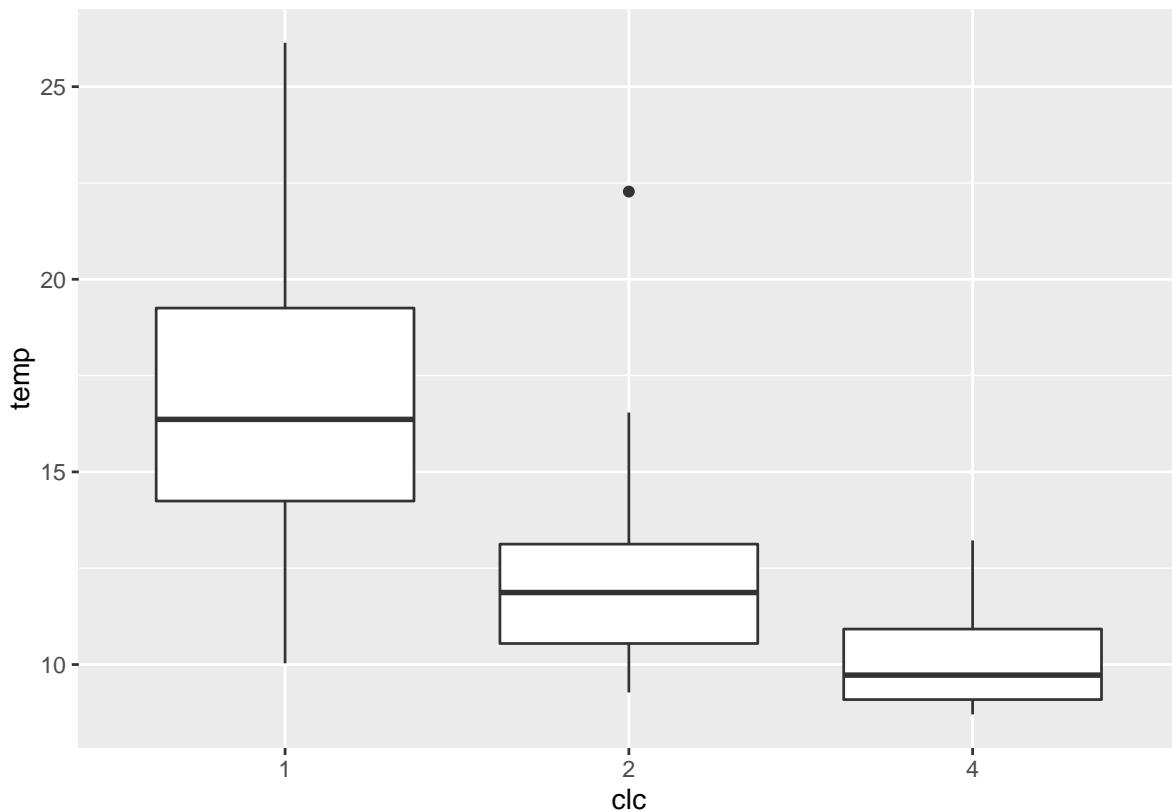
```
corrplot(cor(punkty@data[c(1, 3:5)]))
```



### 3.5.3 Wykres pudełkowy

Wykres pudełkowy obrazuje pięć podstawowych statystyk opisowych oraz wartości odstające. Pudełko to zakres międzykwantylowy (IQR), a linie oznaczają najbardziej ekstremalne wartości, ale nie odstające. Góra z nich to  $1,5 \times \text{IQR}$  ponad krawędź pudełka, dolna to  $1,5 \times \text{IQR}$  poniżej wartości dolnej krawędzi pudełka. Linia środkowa to mediana.

```
punkty$clc <- as.factor(punkty$clc)
ggplot(punkty@data, aes(clc, temp)) + geom_boxplot()
```



### 3.5.4 Testowanie istotności różnic średniej pomiędzy grupami

Analiza wariancji (ang. *Analysis of Variance* - ANOVA) służy do testowania istotności różnic między średnimi w wielu grupach. Metoda ta służy do oceny czy średnie wartości cechy  $Y$  różnią się istotnie pomiędzy grupami wyznaczonymi przez zmienną  $X$ . ANOVA nie pozwala na stwierdzenie między którymi grupami występują różnice. Aby to stwierdzić konieczne jest wykonanie porównań wielokrotnych (*post-hoc*). ANOVĘ można wykonać za pomocą funkcji `aov()` definiując zmienną zależną oraz zmienną grupującą oraz zbiór danych.

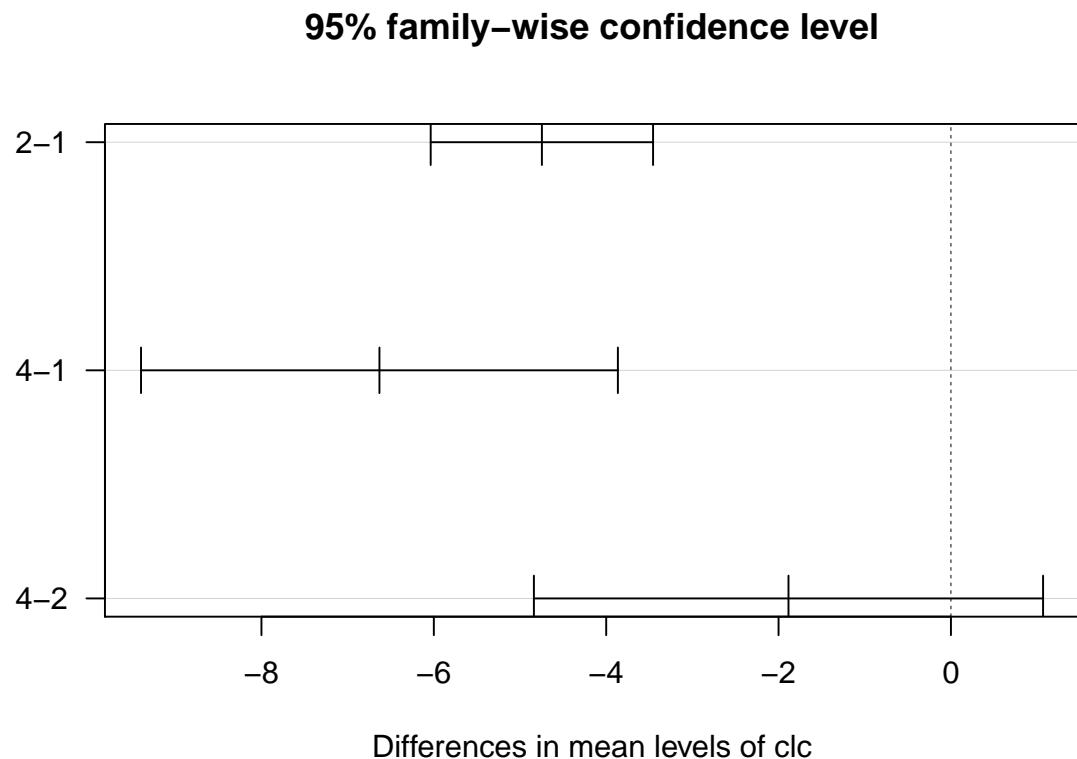
```
punkty$clc <- as.factor(punkty$clc)
aov_test <- aov(temp~clc, data=punkty)
summary(aov_test)

##           Df Sum Sq Mean Sq F value      Pr(>F)
## clc        2   1056   528.0   49.87 <0.000000000000002 ***
## Residuals 247   2615    10.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## null device
##               1
```

Do wykonania porównań wielokrotnych służy funkcja `TukeyHSD()`. Dodatkowo wyniki można wizualizować za pomocą funkcji `plot()`.

```
tukey <- TukeyHSD(aov_test, 'clc')
plot(tukey, las=1)
```



## 3.6 Transformacje danych

### 3.6.1 Transformacje danych

Transformacja danych może mieć na celu ułatwienie porównywania różnych zmiennych, zniwelowanie skośności rozkładu lub też zmniejszenie wpływu danych odstających.

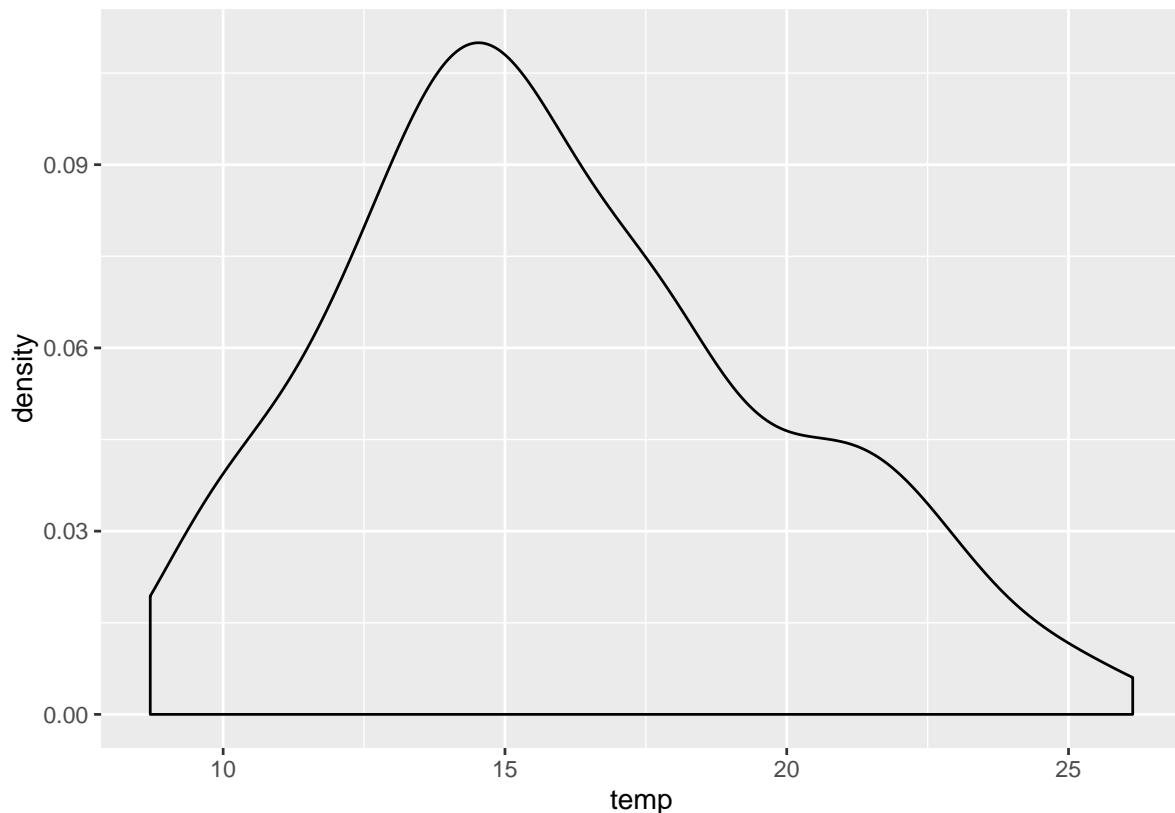
### 3.6.2 Transformacja danych | Standaryzacja

Centrowanie i skalowanie (standaryzacja):

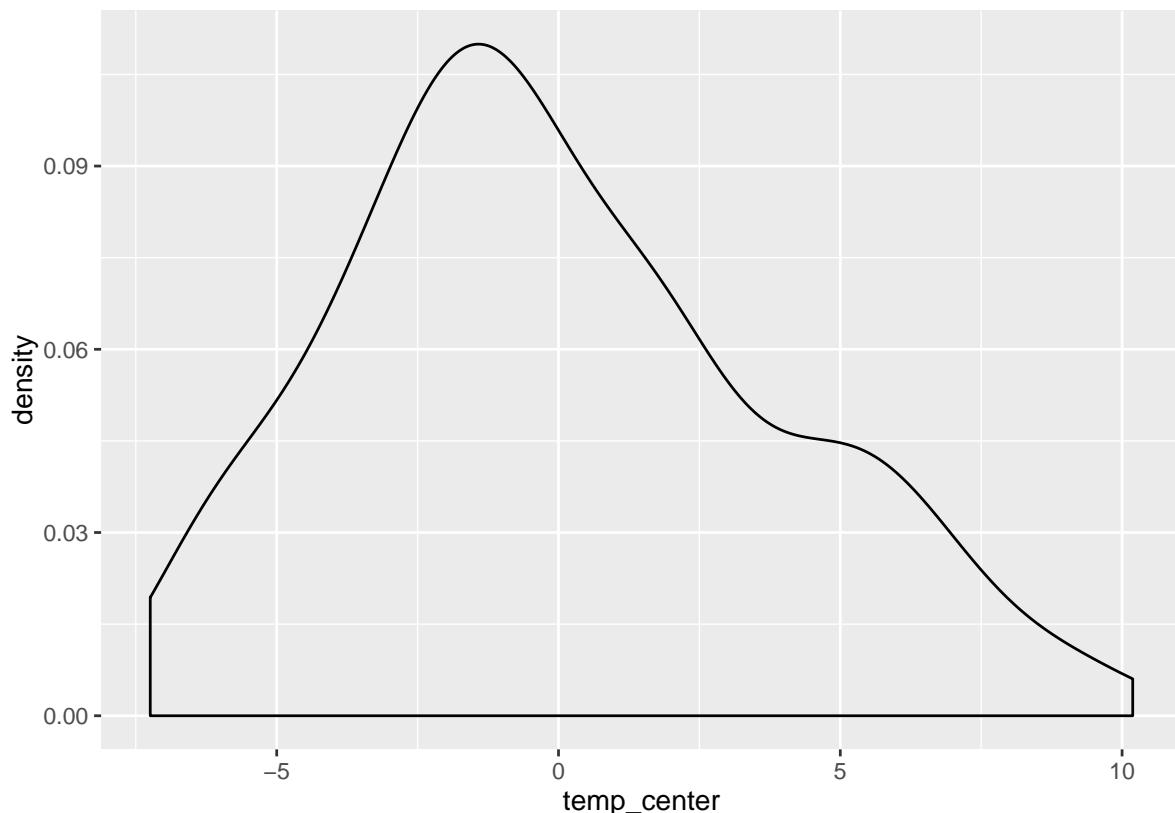
- Centrowanie danych - wybierana jest przeciętna wartość predyktora, a następnie od wszystkich wartości predyktorów odejmowana jest wybrana wcześniej wartość
- Skalowanie danych - dzielenie każdej wartości predyktora przez jego odchylenie standardowe
- Wadą tego podejścia jest główne zmniejszenie interpretowalności pojedynczych wartości

### 3.6.3 Transformacja danych | Centrowanie

```
ggplot(punkty@data, aes(temp)) + geom_density()
```

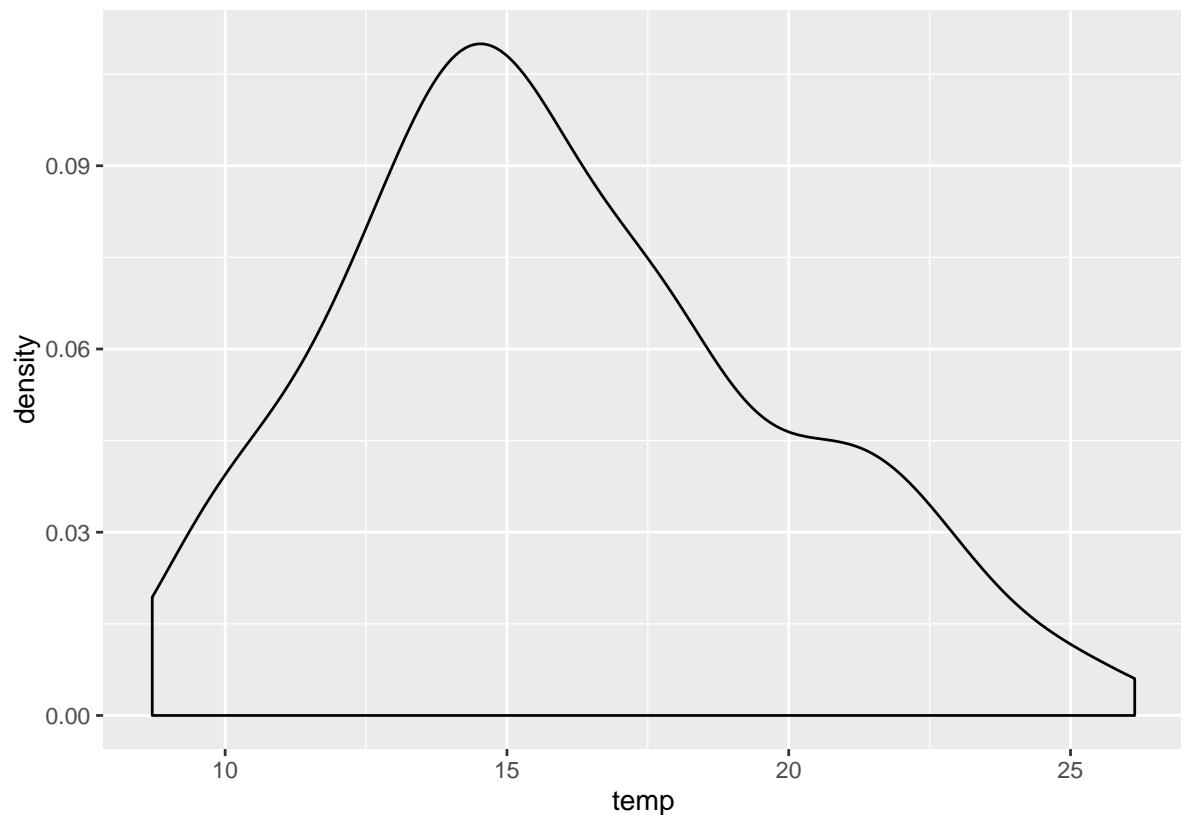


```
punkty$temp_center <- punkty$temp - mean(punkty$temp)
ggplot(punkty@data, aes(temp_center)) + geom_density()
```

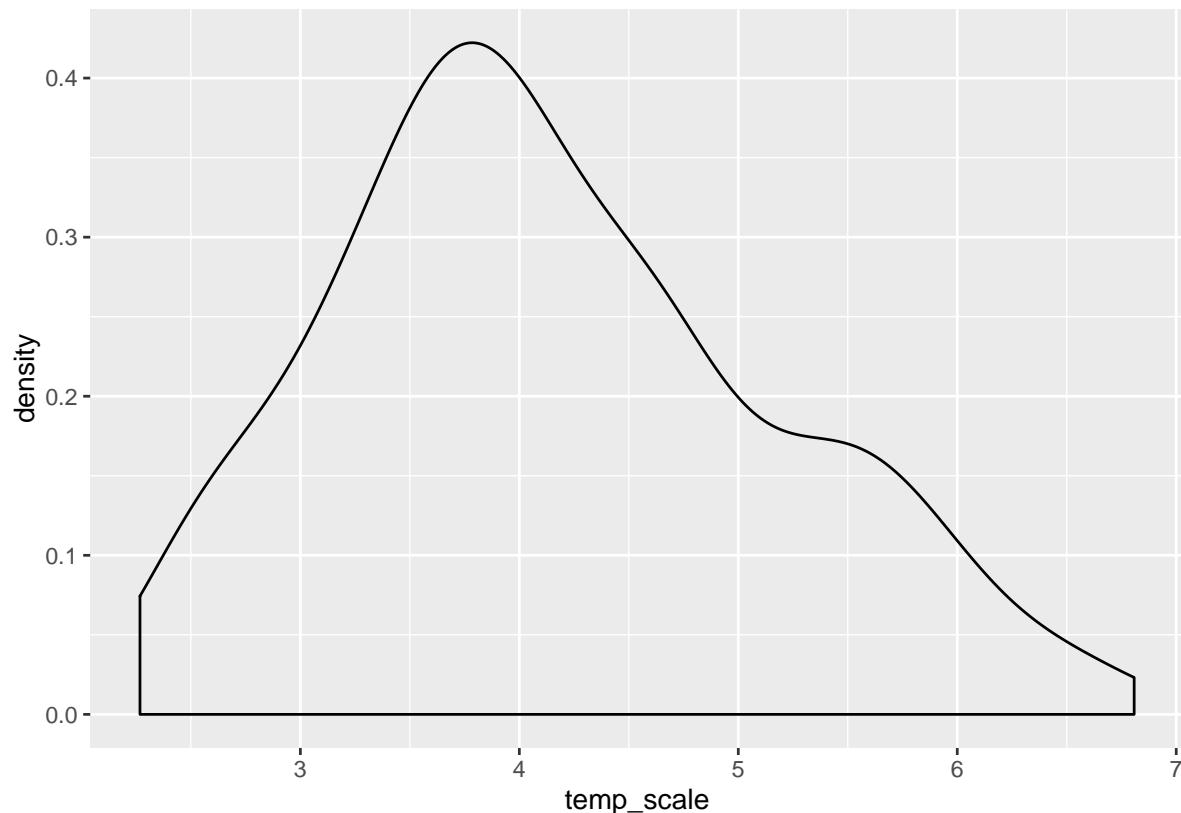


### 3.6.4 Transformacja danych | Skalowanie

```
ggplot(punkty@data, aes(temp)) + geom_density()
```



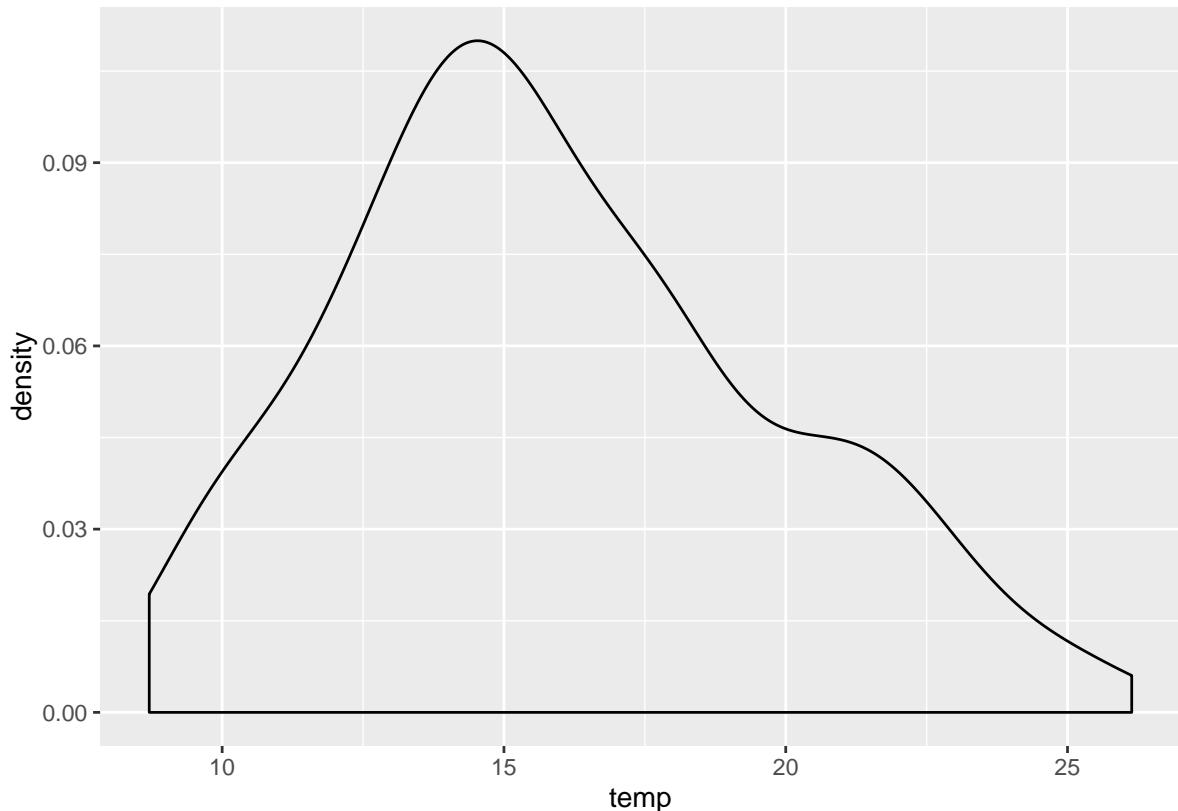
```
punkty$temp_scale <- punkty$temp / sd(punkty$temp)
ggplot(punkty@data, aes(temp_scale)) + geom_density()
```



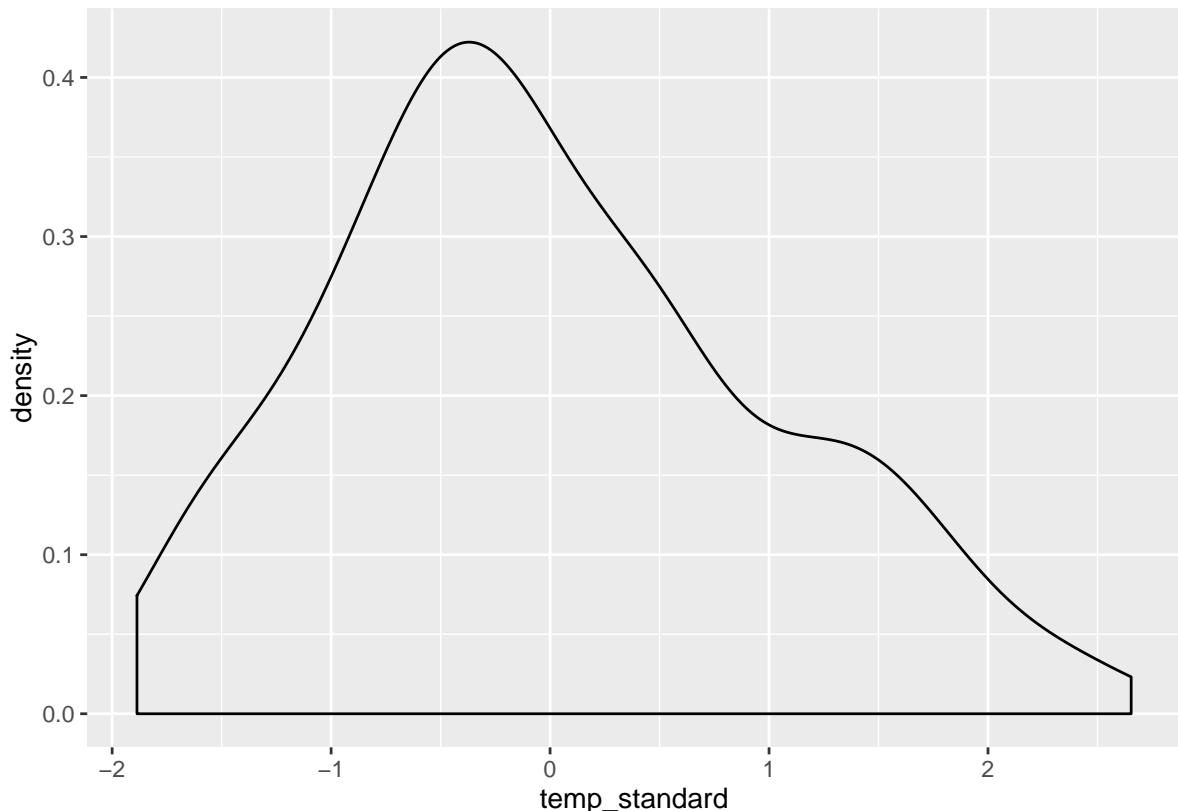
### 3.6.5 Transformacja danych | Standaryzacja

Do standaryzacji (centrowanie i skalowanie) służy funkcja `scale()`.

```
ggplot(punkty@data, aes(temp)) + geom_density()
```



```
punkty$temp_standard <- scale(punkty$temp)
ggplot(punkty@data, aes(temp_standard)) + geom_density()
```



### 3.6.6 Transformacja danych | Redukcja skośności

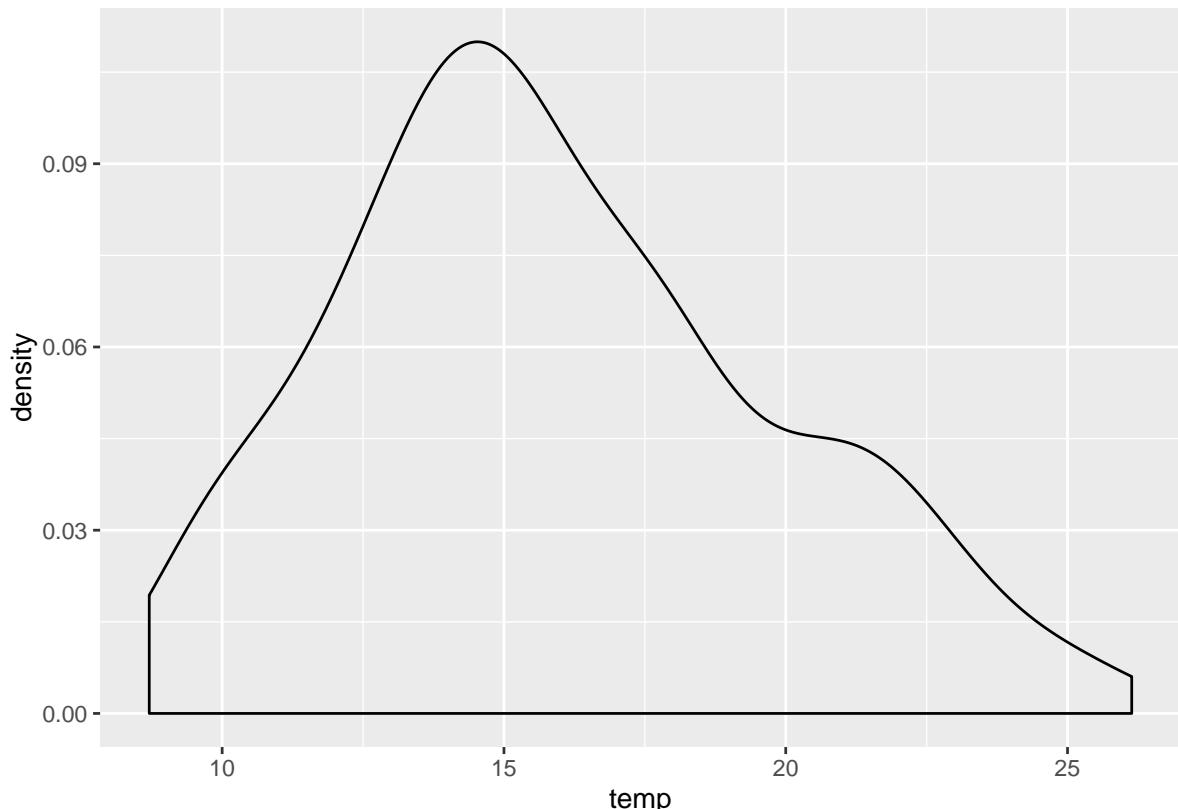
Redukcja skośności:

- Logarytmizacja
- Pierwiastkowanie
- Inne

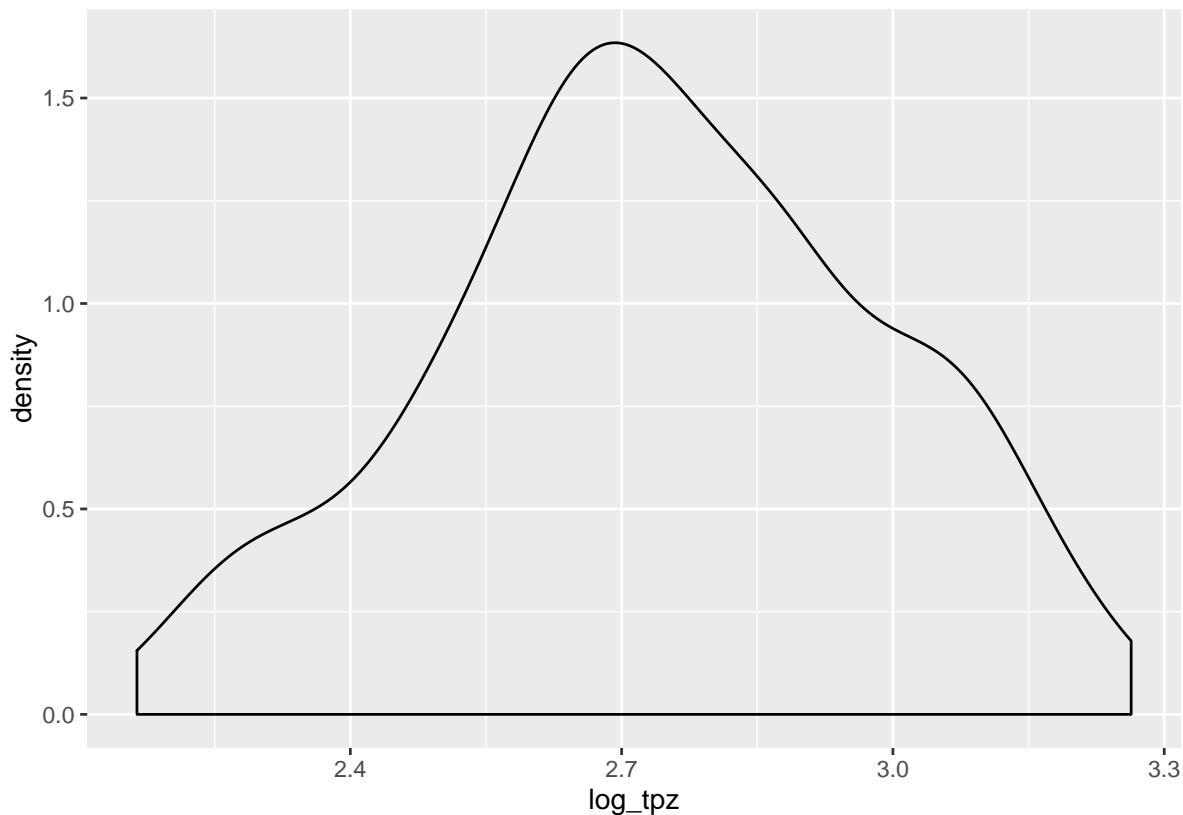
### 3.6.7 Transformacja danych | Logarytmizacja

Logarytmizacja w R może odbyć się za pomocą funkcji `log()`. Transformację logarytmiczną można odwrócić używając funkcji `exp()`.

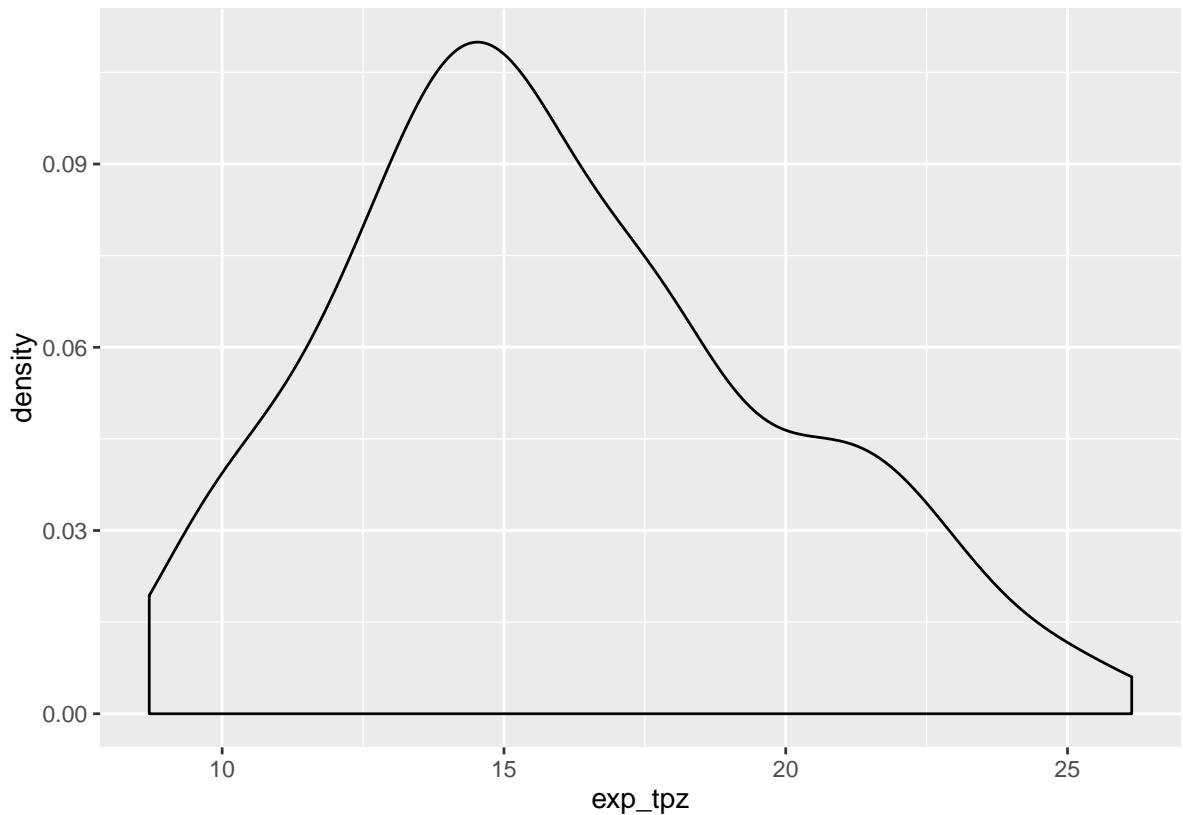
```
ggplot(punkty@data, aes(temp)) + geom_density()
```



```
punkty$log_tpz <- log(punkty$temp)
ggplot(punkty@data, aes(log_tpz)) + geom_density()
```



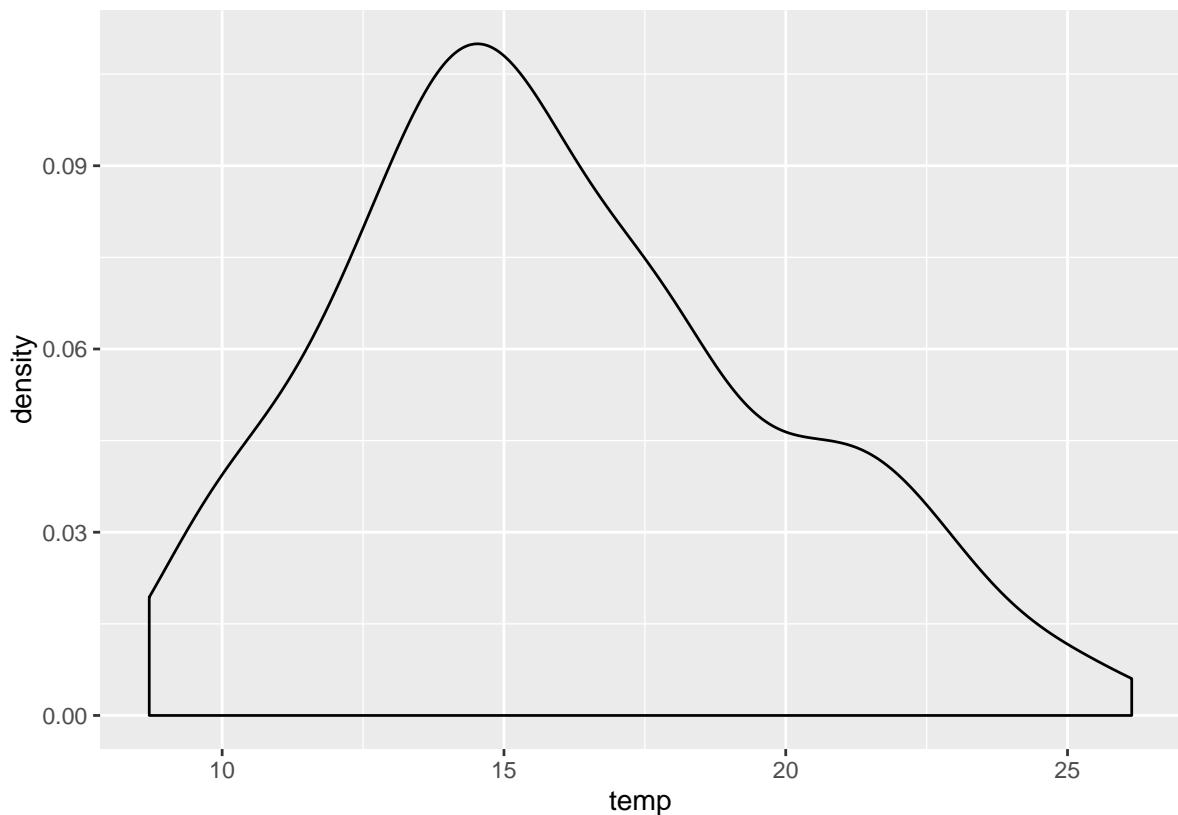
```
punkty$exp_tpz <- exp(punkty$log_tpz)
ggplot(punkty@data, aes(exp_tpz)) + geom_density()
```



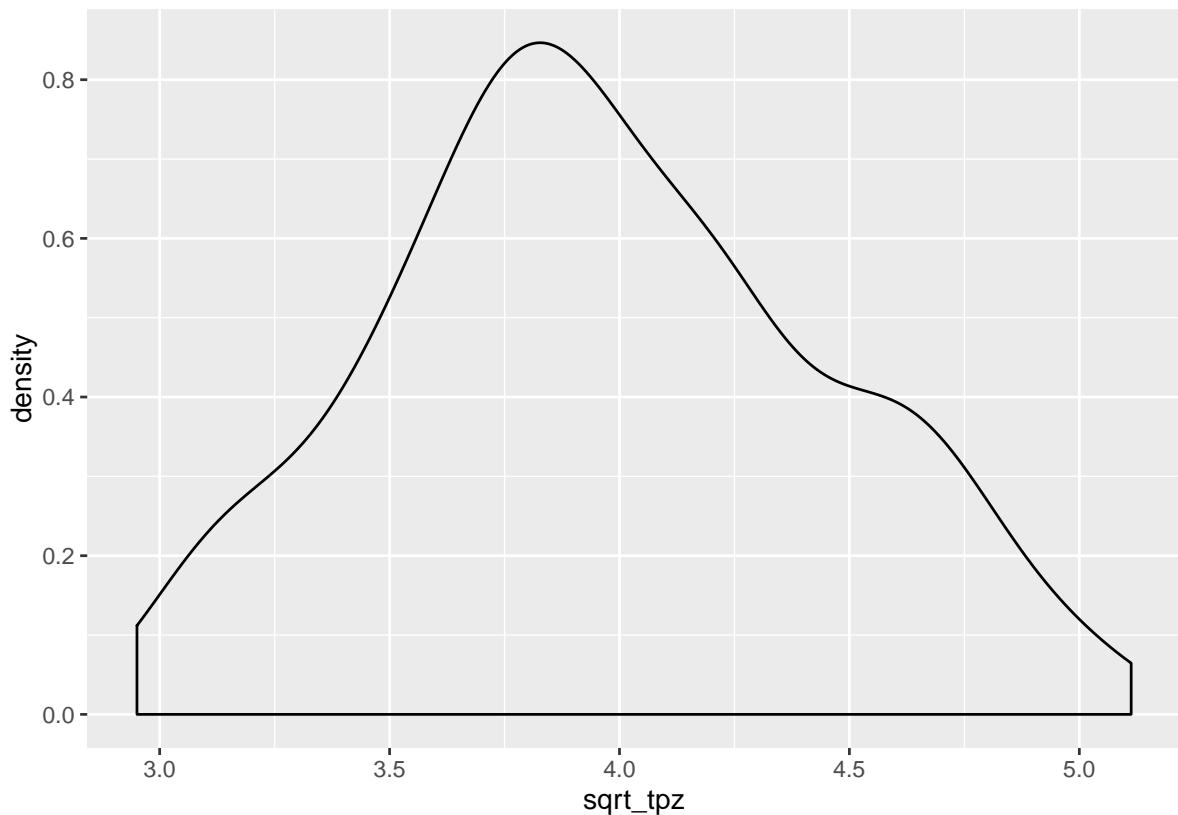
### 3.6.8 Transformacja danych | Pierwiastkowanie

Pierwiastkowanie w R może odbyć się za pomocą funkcji `sqrt()`. Odwrócenie tej transformacji odbywa się poprzez podniesienie wartości do potęgi (^2).

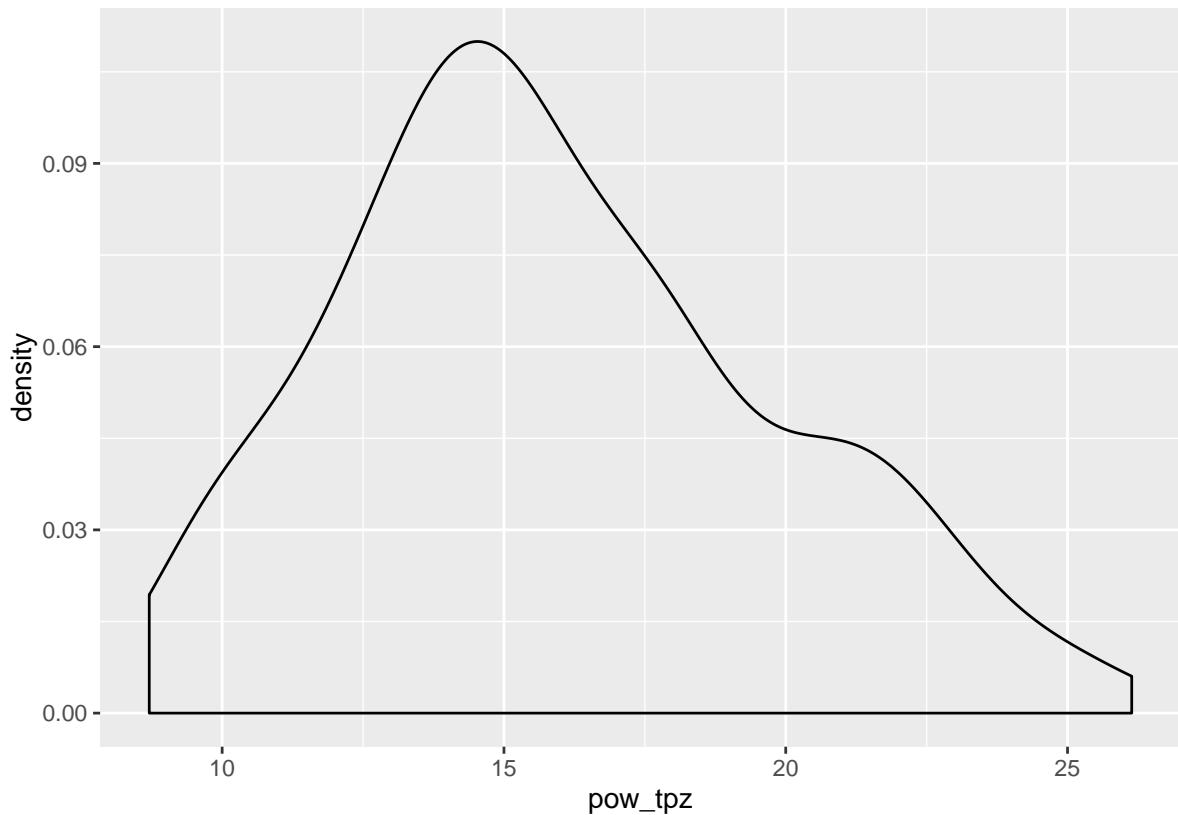
```
ggplot(punkty@data, aes(temp)) + geom_density()
```



```
punkty$sqrt_tpz <- sqrt(punkty$temp)
ggplot(punkty@data, aes(sqrt_tpz)) + geom_density()
```



```
punkty$pow_tpz <- punkty$sqrt_tpz^2  
ggplot(punkty@data, aes(pow_tpz)) + geom_density()
```



## Rozdział 4

# Eksploracyjna analiza danych przestrzennych

```
library('sp')
library('raster')
library('gstat')
library('dismo')
library('rgeos')
library('geostatbook')
data(punkty)
data(granica)
```

## 4.1 Mapy

### 4.1.1 Podstawowe terminy | Kontekst przestrzenny

- Populacja - cały obszar, dla którego chcemy określić wybrane właściwości
- Próba - zbiór obserwacji, dla których mamy informacje. Inaczej, próba to podzbiór populacji. Zazwyczaj niemożliwe (lub bardzo kosztowne) jest zdobycie informacji o całej populacji. Z tego powodu bardzo cenne jest odpowiednie wykorzystanie informacji z próby.

### 4.1.2 Mapy punktowe

Eksploracyjna analiza danych przestrzennych w przypadku danych punktowych ma na celu:

- Wgląd w typ próbkowania danych
- Sprawdzenie poprawności współrzędnych
- Sprawdzenie poprawności danych, w tym między innymi określenie danych odstających lokalnie
- Identyfikacja głównych cech struktury przestrzennej zjawiska (np. trend)

## 4.2 Próbkowanie

### 4.2.1 Podstawowe typy próbowania

Istnieje cały szereg typów próbkowania danych przestrzennych. Funkcja `spsample()` z pakietu `sp` pozwala na stworzenie kilku typów próbkowania (argument `type`), między innymi:

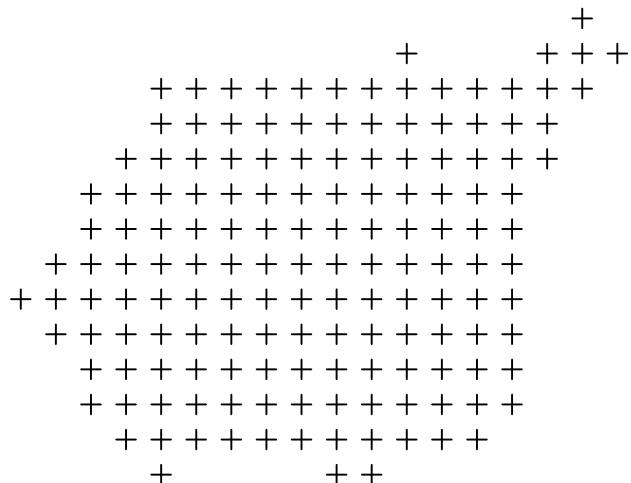
- Regularny (ang.*regular*)
- Losowy (ang.*random*)

- Losowy stratyfikowany (ang.*stratified*)
- Preferencyjny (ang.*clustered*)

#### 4.2.2 Typ próbowania | Regularny

W regularnym typie próbkowania, kolejne punkty rozłożone są równomiernie na badanym obszarze.

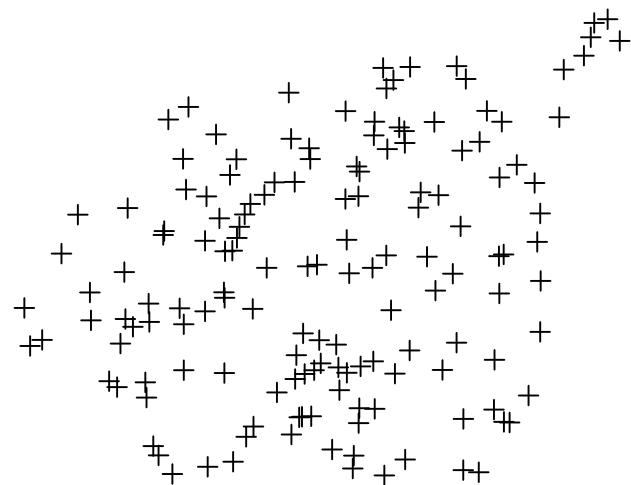
```
set.seed(225)
regularny <- spsample(granica, 150, type = 'regular')
plot(regularny)
```



#### 4.2.3 Typ próbowania | Losowy

W losowym typie próbkowania każda lokalizacja ma takie samo prawdopodobieństwo wystąpienia. Dofinansowanie, każdy punkt jest losowany niezależnie od pozostałych.

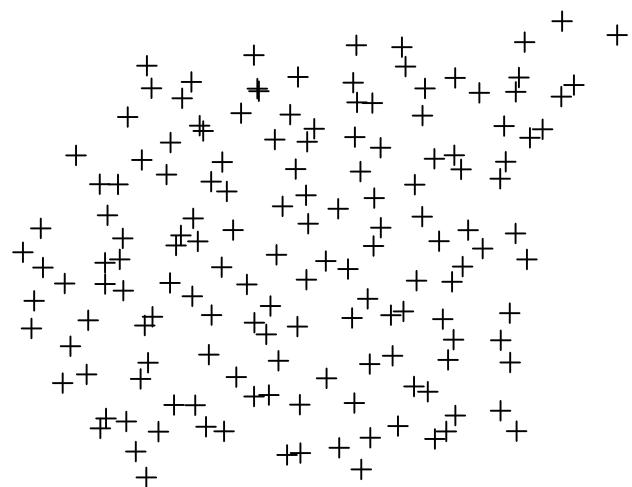
```
set.seed(301)
losowy <- spsample(granica, 150, type = 'random')
plot(losowy)
```



#### 4.2.4 Typ próbowania | Losowy stratyfikowany

Losowy stratyfikowany typ próbkowania polega na podzieleniu analizowanego obszaru na regularne komórki, a następnie dla każdej komórki losowana jest lokalizacja punktu.

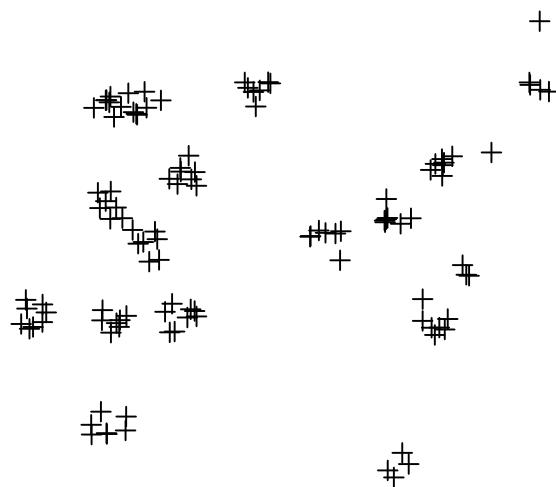
```
set.seed(125)
strat <- spsample(granica, 150, type = 'stratified')
plot(strat)
```



#### 4.2.5 Typ próbowania | Preferencyjny

W preferencyjnym typie próbkowania istnieją obszary, które z jakiegoś powodu (np. specyficzne wartości analizowanej cechy) są znacznie częściej opróbkowane niż inne.

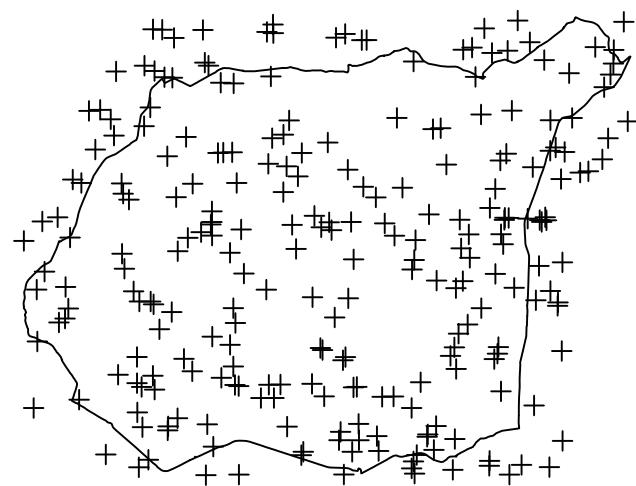
```
set.seed(425)
pref <- spsample(granica, 150, type = 'clustered', nclusters=30)
plot(pref)
```



### 4.3 Dane lokalnie odstające

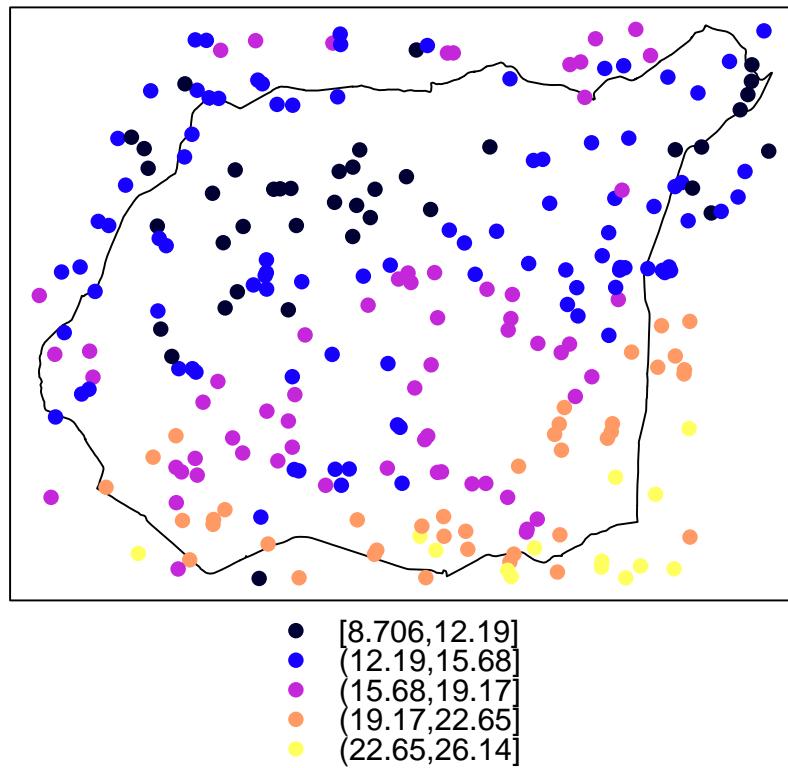
Wstępne sprawdzenie poprawności współrzędnych można wykonać poprzez wizualizację danych przestrzennych za pomocą funkcji `plot()`.

```
plot(granica)
plot(punkty, add=TRUE)
```



Dane lokalnie odstające oznaczają nietypowe przestrzennie wartości danej cechy. Inaczej mówiąc, może to być niska wartość otoczona wysokimi wartościami lub też wysoka wartość otoczona niskimi wartościami. Dane lokalnie ostające mogą oznaczać zarówno błąd w danych albo wpływ innego czynnika na analizowaną cechę. Przyjrzenie się wartościom analizowanej cechy można wykonać z użyciem funkcji `spplot()`. Na poniższym przykładzie wyświetlona jest zmienność `temp` oznaczająca średnią temperaturę dobową.

```
spplot(punkty, 'temp', sp.layout = granica)
```



Dodatkowo, używając argumentu `identify = TRUE` można interaktywnie określić numer punktu (numer wiersza w tabeli atrybutów). Działanie tej funkcji polega na wybraniu punktów za pomocą lewego przycisku myszy, a po wybraniu serii punktów klawisza Esc.

```
spplot(punkty, 'temp', identify = TRUE)
```

## 4.4 Rozgrupowanie danych

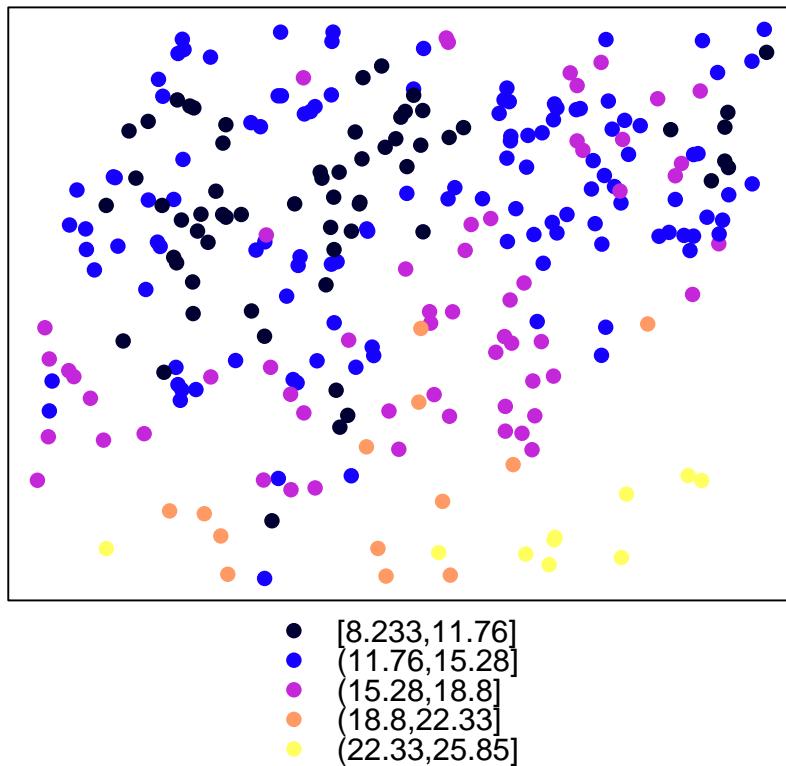
### 4.4.1 Rozgrupowanie danych

Istnieje szereg metod rozgrupowywania danych, między innymi jest to rozgrupowywanie komórkowe oraz rozgrupowywanie poligonalne. Celem tych metod jest nadanie wag obserwacjom w celu zapewnienia reprezentatywności przestrzennej danych. Poniżej znajdują się przykłady użycia dwóch wersji rozgrupowania komórkowego (rozgrupowywanie komórkowe I oraz rozgrupowywanie komórkowe II) i jedna wersja rozgrupowania poligonowego.

### 4.4.2 Rozgrupowanie danych

Na potrzeby przykładów związanych z rozgrupowaniem danych w pakiecie `geostatbook` znajduje się zbiór danych `punkty_pref`. W tym zbiorze gęściej opróbkowane są niskie wartości temperatury, co powoduje, że średnia dla całego obszaru jest znacznie niższa niż w rzeczywistości.

```
data(punkty_pref)
spplot(punkty_pref, 'temp')
```



```
summary(punkty_pref$temp)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##    8.233 12.070 14.260 14.280 15.540 25.850
```

#### 4.4.3 Rozgrupowanie komórkowe I | (ang. *cell declustering*)

Pierwszy rodzaj rozgrupowania komórkowego polega na:

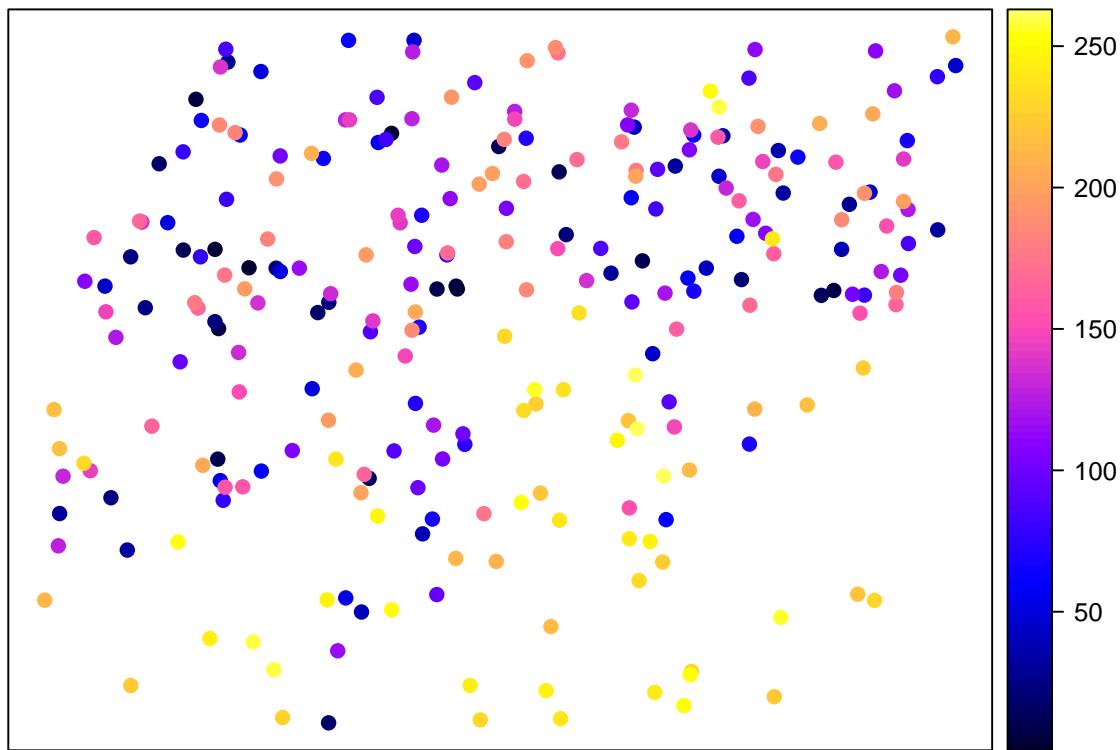
1. Stworzeniu siatki dla badanego obszaru
2. Policzeniu liczby obserwacji w każdym oczku siatki
3. Nadanie wagi dla każdego punktu, zgodnie ze wzorem:

$$w'_j = \frac{\frac{1}{n_i}}{\text{liczba komórek z danymi}} \cdot n$$

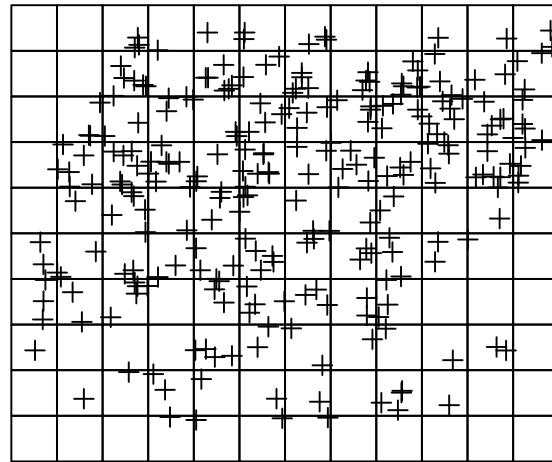
, gdzie  $n_i$  to liczba obserwacji w komórce, a  $n$  to łączna liczba obserwacji

#### 4.4.4 Rozgrupowanie komórkowe II | (ang. *cell declustering*)

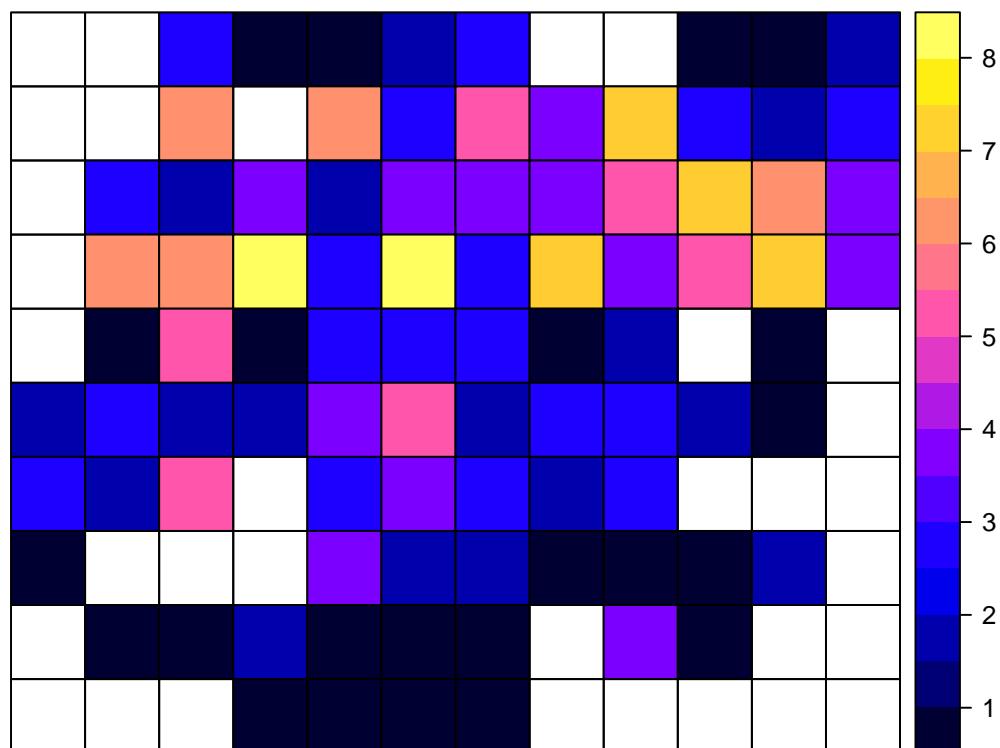
```
data(punkty_pref)
punkty_pref$id <- 1:nrow(punkty_pref)
spplot(punkty_pref, 'id', colorkey=TRUE)
```



```
data(granica)
siatka_n <- raster(extent(gBuffer(granica, width = 500)))
res(siatka_n) <- c(1000, 1000)
siatka_n[] <- 0
proj4string(siatka_n) <- CRS(proj4string(punkty_pref))
siatka_n <- as(siatka_n, 'SpatialPolygonsDataFrame')
siatka_n <- siatka_n[!is.na(siatka_n@data$layer), ]
plot(siatka_n)
plot(punkty_pref, add=TRUE)
```

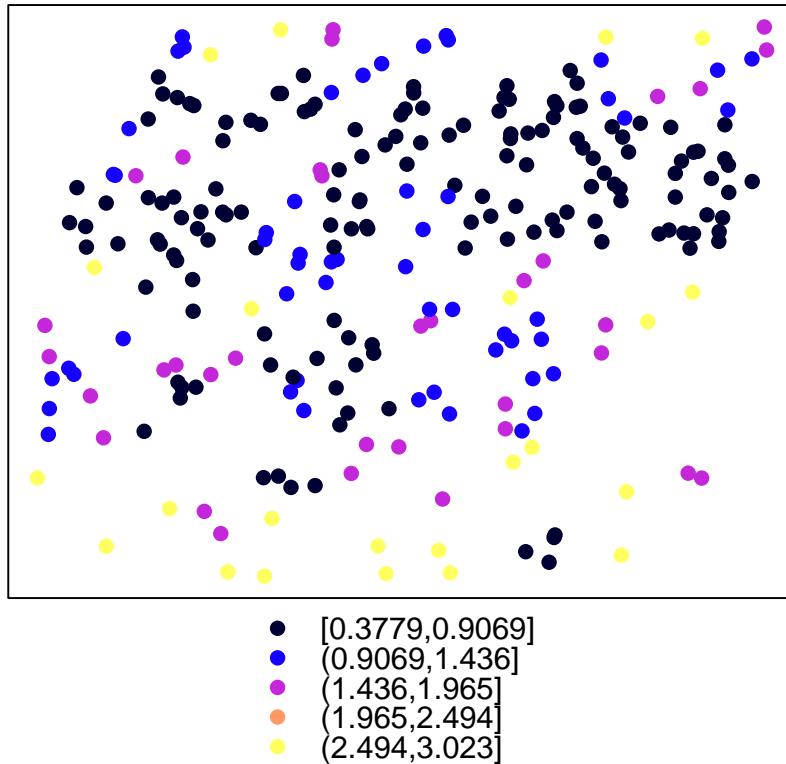


```
punkty_pref$liczebnosc <- rep(0, length(punkty_pref))
siatka_nr <- aggregate(punkty_pref['liczebnosc'], by = siatka_n, FUN = length)
spplot(siatka_nr, 'liczebnosc')
```



```
liczba <- over(punkty_pref, siatka_nr)
punkty_pref$waga <- ((1/liczba$liczebnosc)/sum(!is.na(siatka_nr$liczebnosc))) * length(punkty_pref)

spplot(punkty_pref, 'waga')
```



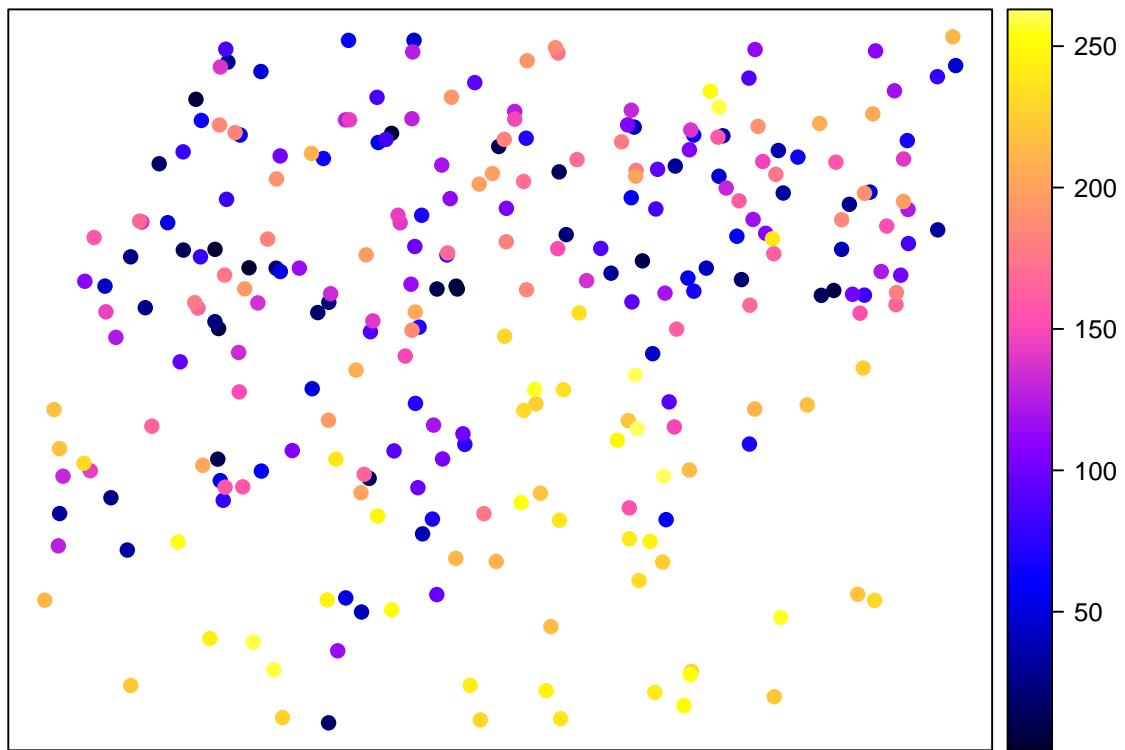
```
srednia_arytmetyczna <- mean(punkty_pref$temp)
srednia_wazona_c1 <- mean(punkty_pref$temp * punkty_pref$waga, na.rm=TRUE)
```

#### 4.4.5 Rozgrupowanie komórkowe II | (ang. *cell declustering*)

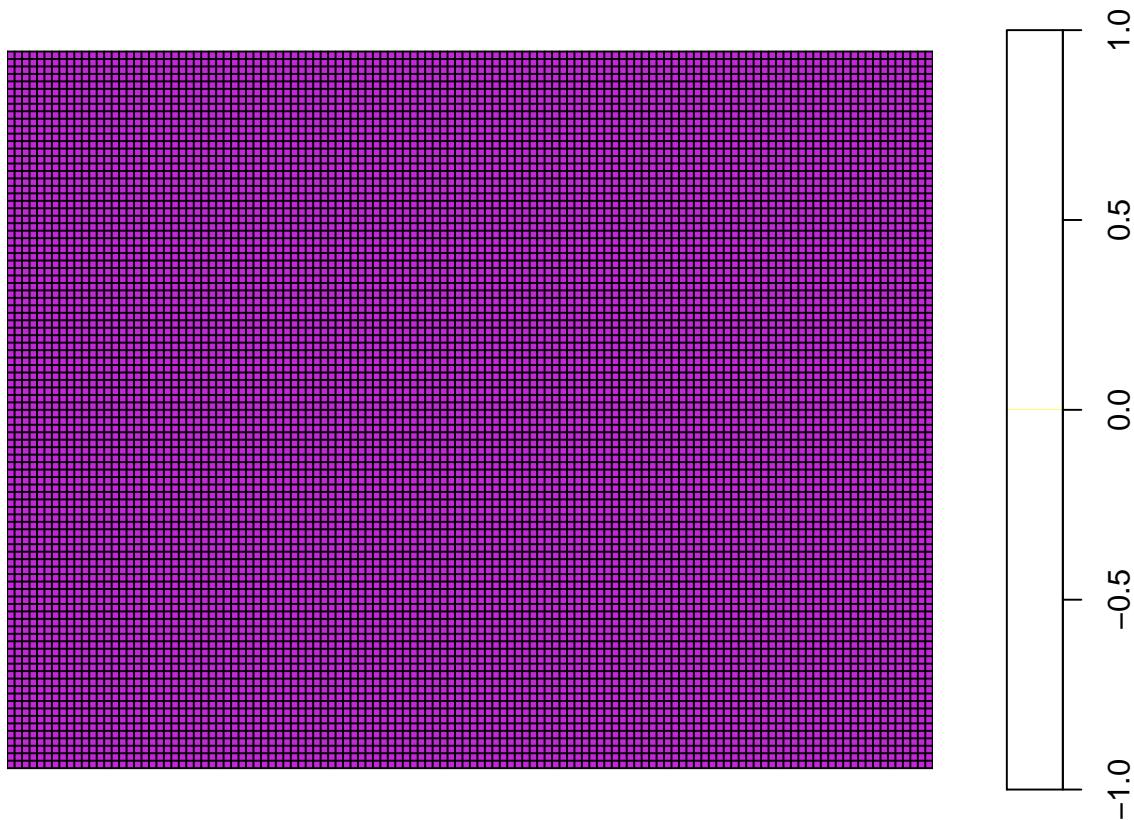
Drugi rodzaj rozgrupowania komórkowego polega na:

1. Stworzeniu siatki dla badanego obszaru
2. Wykonaniu interpolacji z użyciem funkcji `krige()` z pakietu `gstat`. W tym wypadku konieczne jest użycie argumentu `nmax = 1`, który przypisuje wartość najbliższej obserwacji do każdego oczka siatki.
3. Waga dla każdego punktu nadawana jest poprzez zliczenie liczby oczek siatki dla konkretnej wartości, a następnie podzielenie tego przez liczbę oczek siatki.

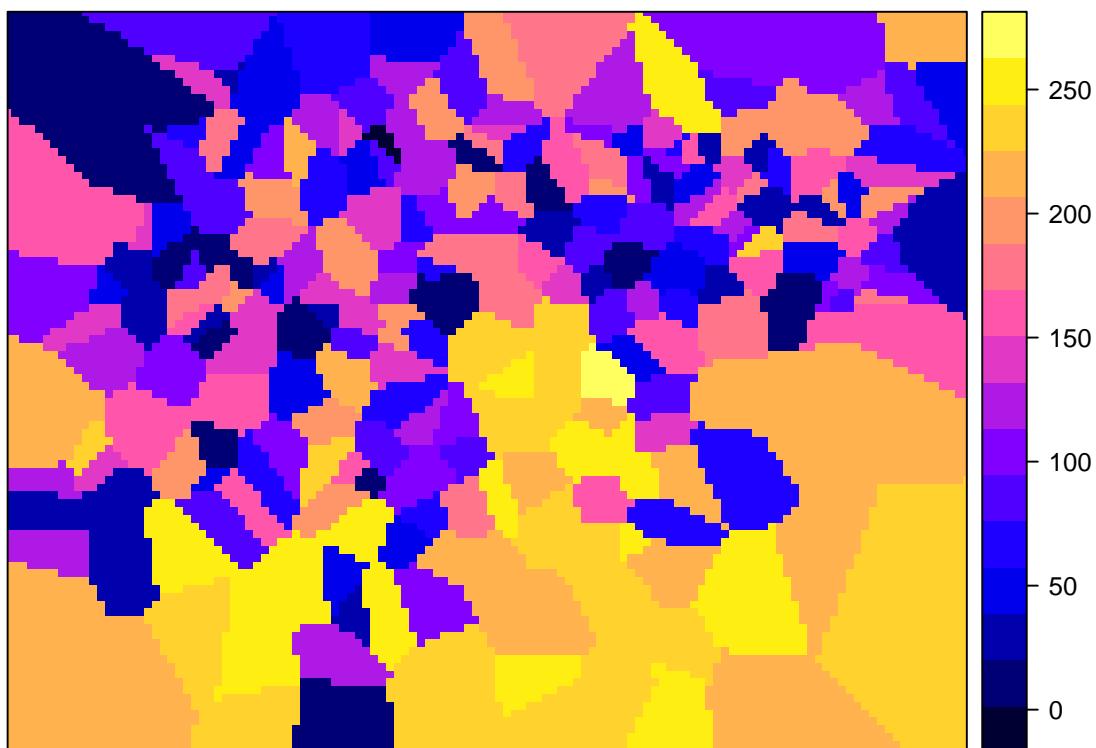
```
data(punkty_pref)
punkty_pref$id <- 1:nrow(punkty_pref)
spplot(punkty_pref, 'id', colorkey=TRUE)
```



```
data(granica)
siatka_n <- raster(extent(gBuffer(granica, width = 500)))
res(siatka_n) <- c(100, 100)
siatka_n[] <- 0
proj4string(siatka_n) <- CRS(proj4string(punkty_pref))
siatka_n <- as(siatka_n, 'SpatialPointsDataFrame')
siatka_n <- siatka_n[!is.na(siatka_n@data$layer), ]
gridded(siatka_n) <- TRUE
plot(siatka_n, border = TRUE)
```



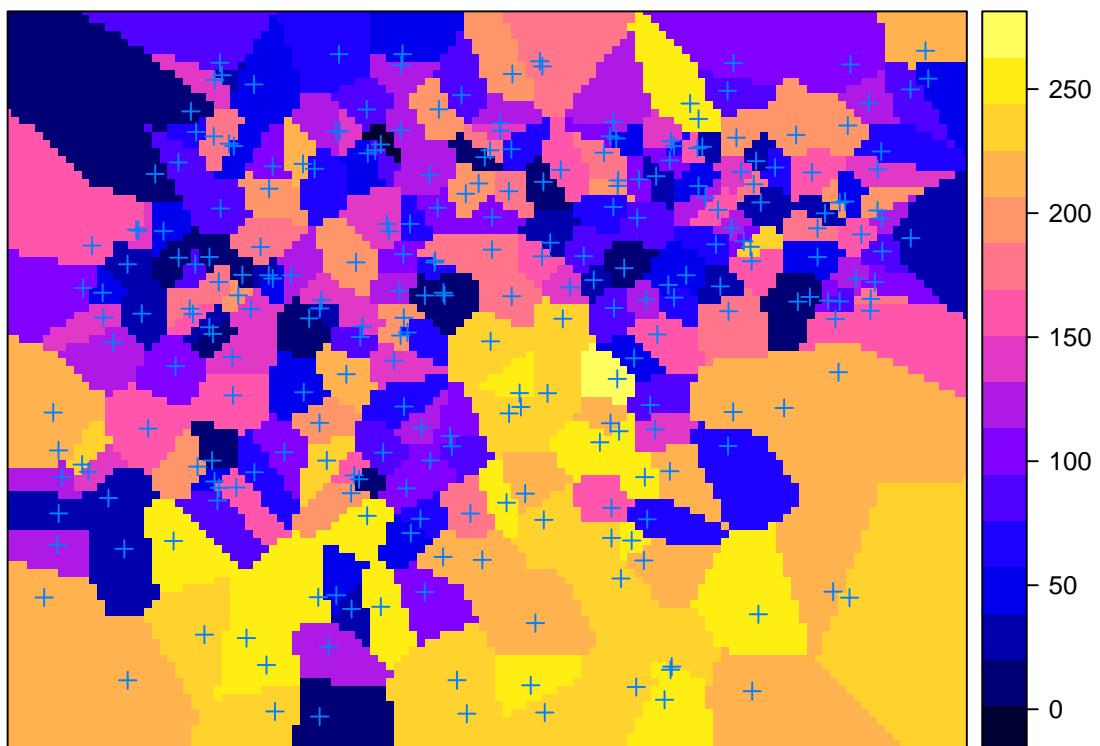
```
out <- krige(id~1, punkty_pref, siatka_n, nmax=1)  
  
## [inverse distance weighted interpolation]  
spplot(out, 'var1.pred')
```



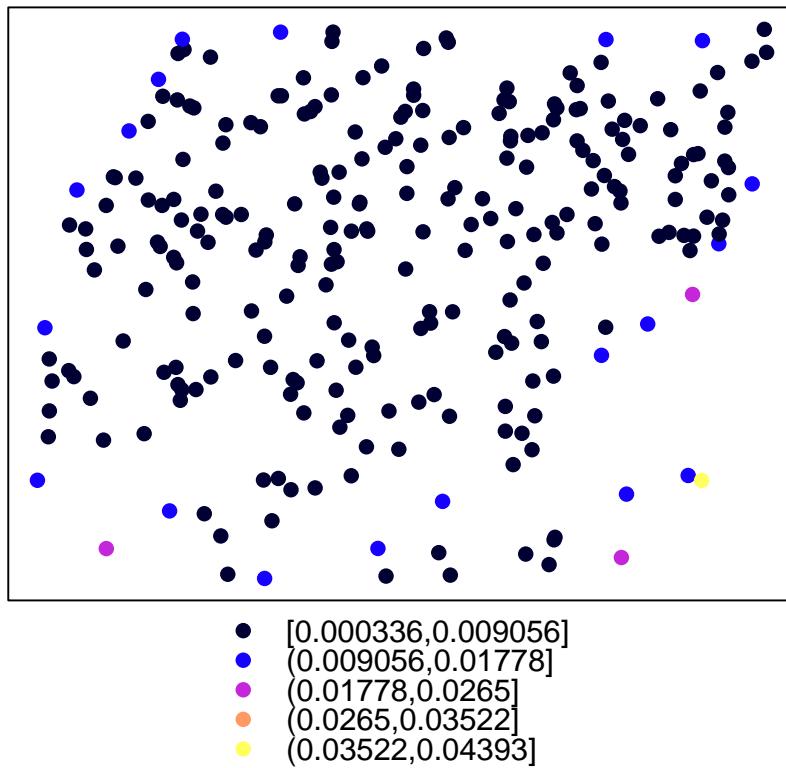
```
df <- as.data.frame(table(out[[1]]))
df$waga <- df$Freq/sum(df$Freq)
punkty_pref <- merge(punkty_pref, df, by.x='id', by.y='Var1')
summary(punkty_pref$waga)
```

```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
## 0.000336 0.001386 0.002520 0.003802 0.004326 0.043930
```

```
spplot(out, 'var1.pred', sp.layout=list('sp.points', punkty_pref))
```



```
spplot(punkty_pref['waga'])
```



```
srednia_arytmetyczna <- mean(punkty_pref$temp)
srednia_wazona_c2 <- sum(punkty_pref$temp * punkty_pref$waga, na.rm=TRUE)
```

#### 4.4.6 Rozgrupowanie poligonowe | (ang. *polygon declustering*)

Rozgrupowanie poligonowe polega na zastosowaniu jednej z metod triangulacji, np. poligonów Voronoi'a:

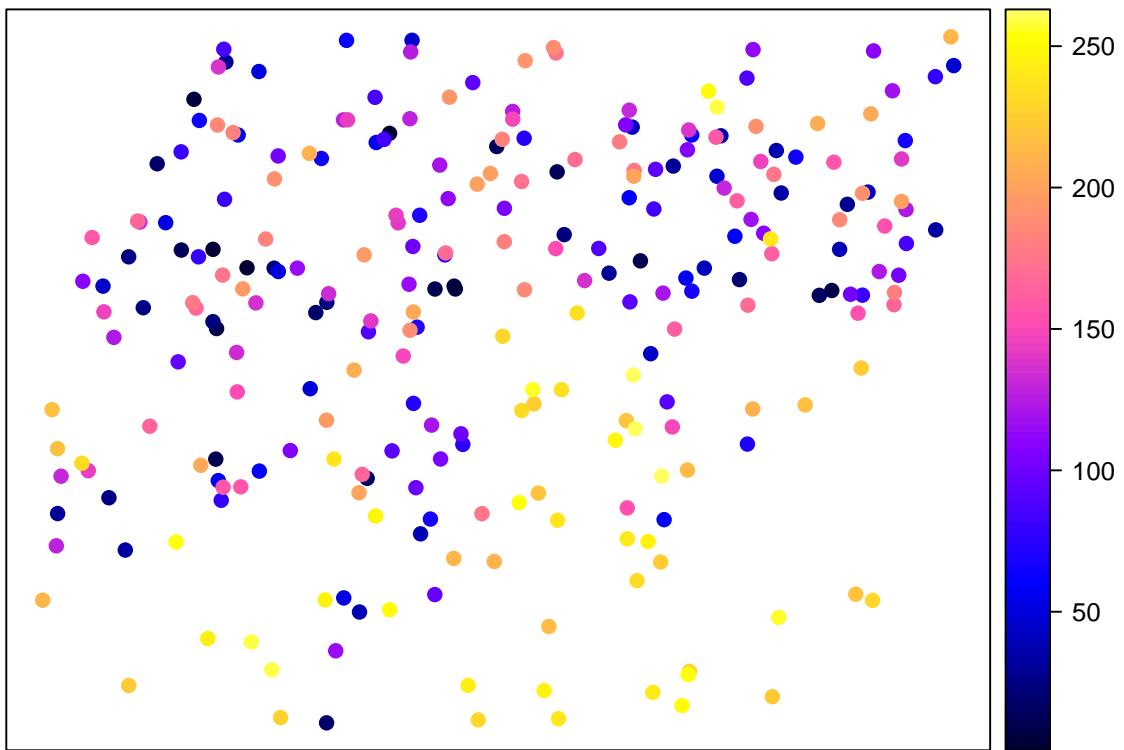
1. Dla każdego punktu określany jest poligon.
2. Wyliczana jest powierzchnia poligona.
3. Waga każdego punktu wyliczana jest poprzez podzielenie powierzchni indywidualnych przez powierzchnię całego obszaru, a następnie pomnożenie przez liczbę punktów.

$$w'_j = \frac{area_j}{\sum_{j=1}^n area_j} \cdot n$$

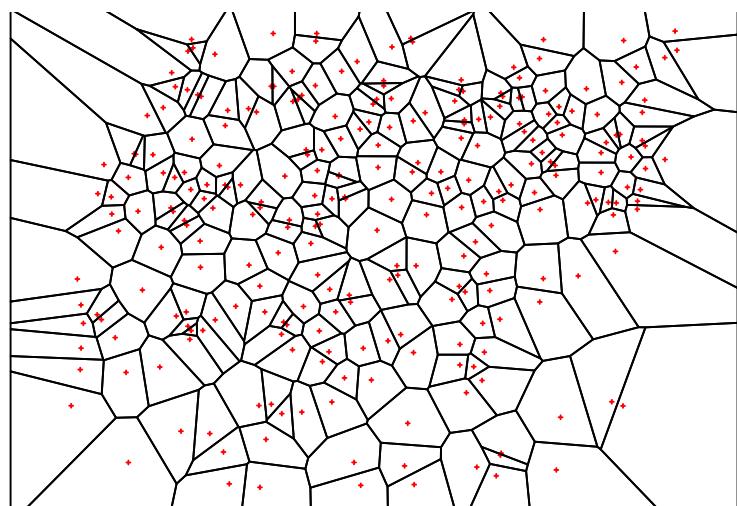
, gdzie  $area_j$  powierzchnia dla wybranej obserwacji, a  $n$  to łączna liczba obserwacji

#### 4.4.7 Rozgrupowanie poligonowe | (ang. *polygon declustering*)

```
data(punkty_pref)
punkty_pref$id <- 1:nrow(punkty_pref)
spplot(punkty_pref, 'id', colorkey=TRUE)
```

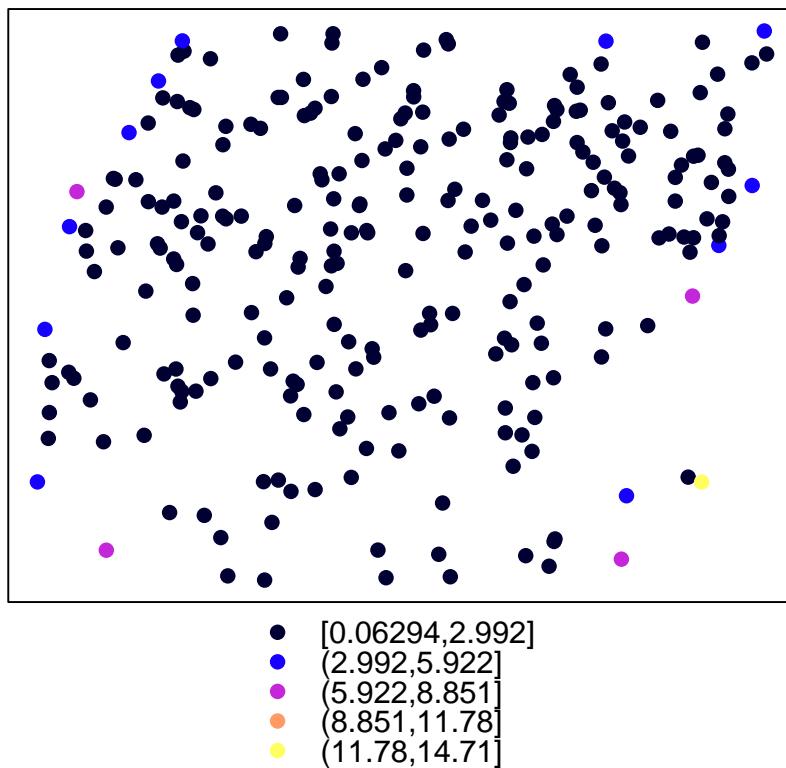


```
v <- voronoi(punkty_pref)
plot(punkty_pref, cex=0.2, col='red')
plot(v, add=TRUE)
```



```
v$pow <- area(v)
v$waga <- v$pow/sum(v$pow) * length(punkty_pref)

punkty_pref <- merge(punkty_pref, v[c('id', 'waga')], by='id')
spplot(punkty_pref, 'waga')
```



```
srednia_arytmetyczna <- mean(punkty_pref$temp, na.rm=TRUE)
srednia_wazona_p <- mean(punkty_pref$temp*punkty_pref>waga, na.rm=TRUE)
```

#### 4.4.8 Porównanie metod rozgrupowania

Średnia wartość temperatury dla badanego obszaru wynosiła 15,59 stopni Celsjusza, jednak w preferencyjnej próbie ta wartość wynosiła 14,28 stopni Celsjusza. Porównując trzy zastosowane metody próbkoowania warto zauważyc, że najbliższy wynik uzyskano korzystając z pierwszego rodzaju rozgrupowania komórkowego, która nieznacznie zaniżyła rzeczywistą wartość temperatury. Rozgrupowanie komórkowe II oraz rozgrupowanie poligonalne były w tym przypadku mniej dokładne, wyraźnie zawyżając wartości temperatury.

	Średnia arytmetyczna
Populacja	15.5913
Próba	14.2815
Rozgrupowanie komórkowe I	15.3789
Rozgrupowanie komórkowe II	16.0734
Rozgrupowanie poligonalne	16.2511

W przypadku metod rozgrupowania należy jednak pamiętać, że ich wynik zależy od szeregu wprowadzonych parametrów, w szczególności granic badanego obszaru oraz zastosowanej wielkości oczka siatki.



## Rozdział 5

# Metody interpolacji

```
library('dismo')
library('raster')
library('sp')
library('gstat')
library('fields')
library('geostatbook')
data(punkty)
data(siatka)
data(granica)
```

Przez przejściem do interpolacji geostatystycznych warto zdać sobie sprawę, że nie jest to jedyna możliwa droga postępowania. Można wyróżnić dwie główne grupy modeli przestrzennych - modele deterministyczne oraz modele statystyczne.

### 5.1 Modele deterministyczne

Modele deterministyczne charakteryzują się tym, że ich parametry są zazwyczaj ustalane w oparciu o funkcję odległości lub powierzchni. W tych modelach brakuje szacunków na temat oceny błędu modelu. Do zalet tych modeli należy szczególnie krótki czas obliczeń. Do modeli deterministycznych należą, między innymi:

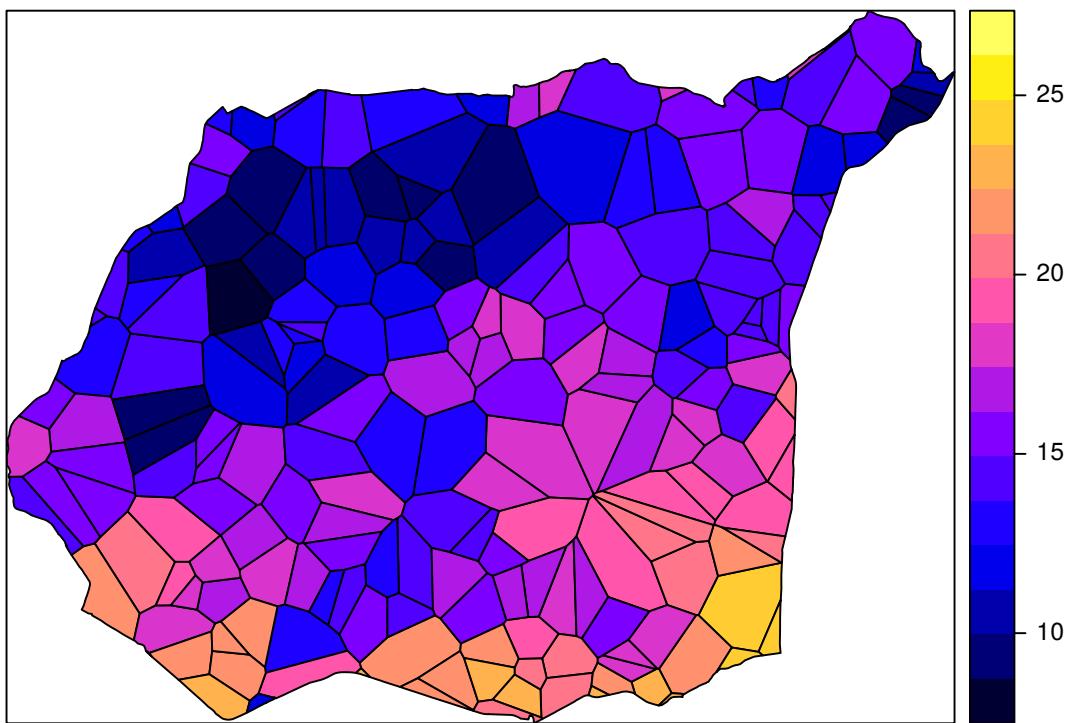
- Metoda diagramów Voronoi'a (ang. *Voronoi diagram*)
- Metoda średniej ważonej odległością (ang. *Inverse Distance Weighted - IDW*)
- Funkcje wielomianowe (ang. *Polynomials*)
- Funkcje sklejane (ang. *Splines*)

#### 5.1.1 Modele deterministyczne | Voronoi

Metoda diagramów Voronoi'a polega na stworzeniu nieregularnych poligonów na podstawie analizowanych punktów, a następnie wpisaniu w każdy poligon wartości odpowiadającego punktu. Na poniższym przykładzie ta metoda stosowana jest z użyciem funkcji `voronoi()` z pakietu `dismo`. Wyniki następnie można przyciąć do badanego obszaru z użyciem funkcji `intersect()` z pakietu `raster`.

```
voronoi_interp <- voronoi(punkty)
voronoi_interp <- intersect(granica, voronoi_interp)
spplot(voronoi_interp, 'temp', contour=TRUE, main='Poligony Voronoia')
```

## Poligony Voronoi

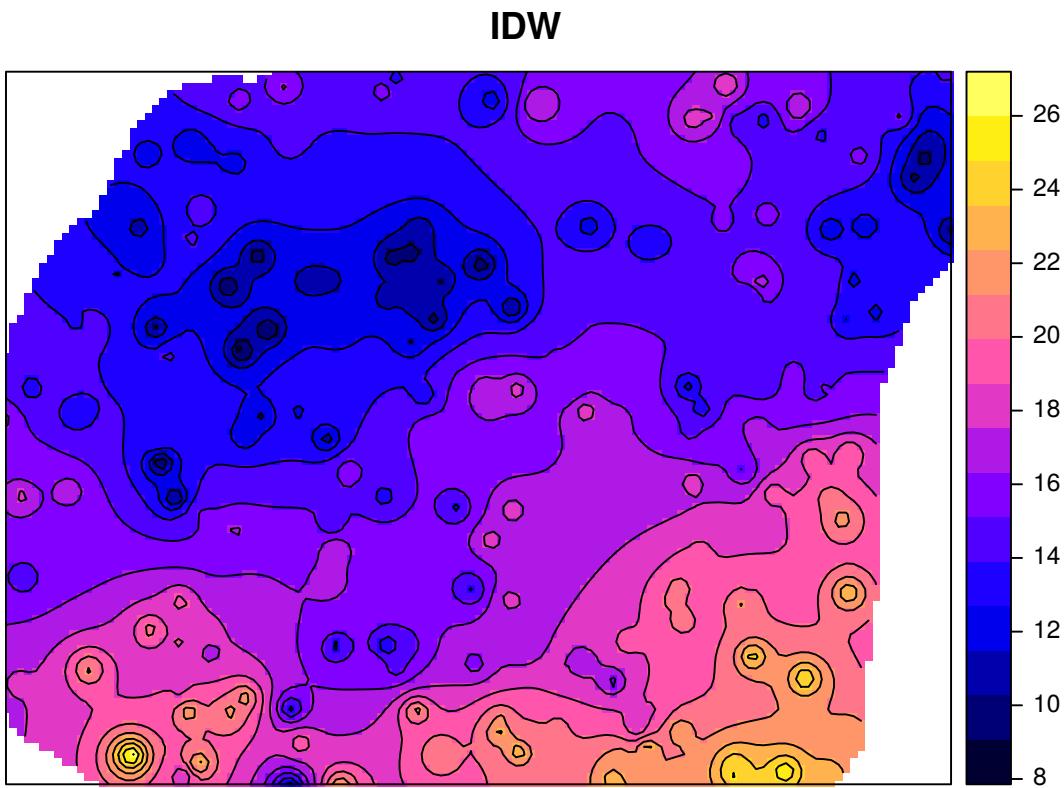


### 5.1.2 Modele deterministyczne | IDW

Metoda średniej ważonej odległością (IDW) wylicza wartość dla każdej komórki na podstawie wartości punktów obokległych ważonych odwrotnością ich odległości. W efekcie, czym punkt jest bardziej oddalony, tym mniejszy jest jego wpływ na interpolowaną wartość. Wagę punktów ustala się z użyciem argumentu wykładnika potęgowego (ang. *power*). W pakiecie `gstat` istnieje do tego celu funkcja `idw()`, która przyjmuje analizowaną cechę (`temp~1`), zbiór punktowy, siatkę, oraz wartość wykładnika potęgowego (argument `idp`).

```
idw_wolin <- idw(temp~1, punkty, siatka, idp=2)
```

```
## [inverse distance weighted interpolation]
spplot(idw_wolin, 'var1.pred', contour=TRUE, main='IDW')
```



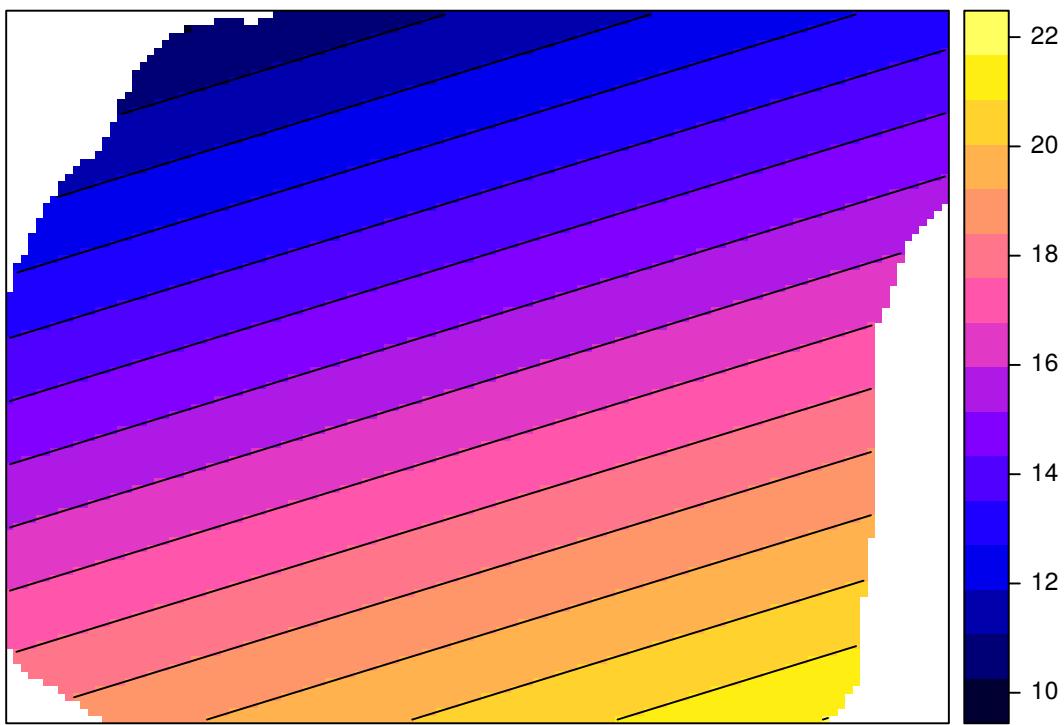
### 5.1.3 Modele deterministyczne | Funkcje wielomianowe

Stosowanie funkcji wielomianowych w R może odbyć się z wykorzystaniem funkcji `gstat()` z pakietu `gstat`. Wymaga ona podania trzech argumentów: `formula` określającego naszą analizowaną cechę (`temp~1` mówi, że chcemy interpolować wartość temperatury zależnej od samej siebie), `data` określający analizowany zbiór danych, oraz `degree` określającą stopień wielomianu. Następnie funkcja `predict()` przenosi nowe wartości na wcześniej stworzoną siatkę.

```
wielomian_1 <- gstat(formula=temp~1, data=punkty, degree=1)
wielomian_1_pred <- predict(wielomian_1, newdata=siatka)
```

```
## [ordinary or weighted least squares prediction]
spplot(wielomian_1_pred[1], contour=TRUE, main='Powierzchnia trendu - wielomian pierwszego stopnia')
```

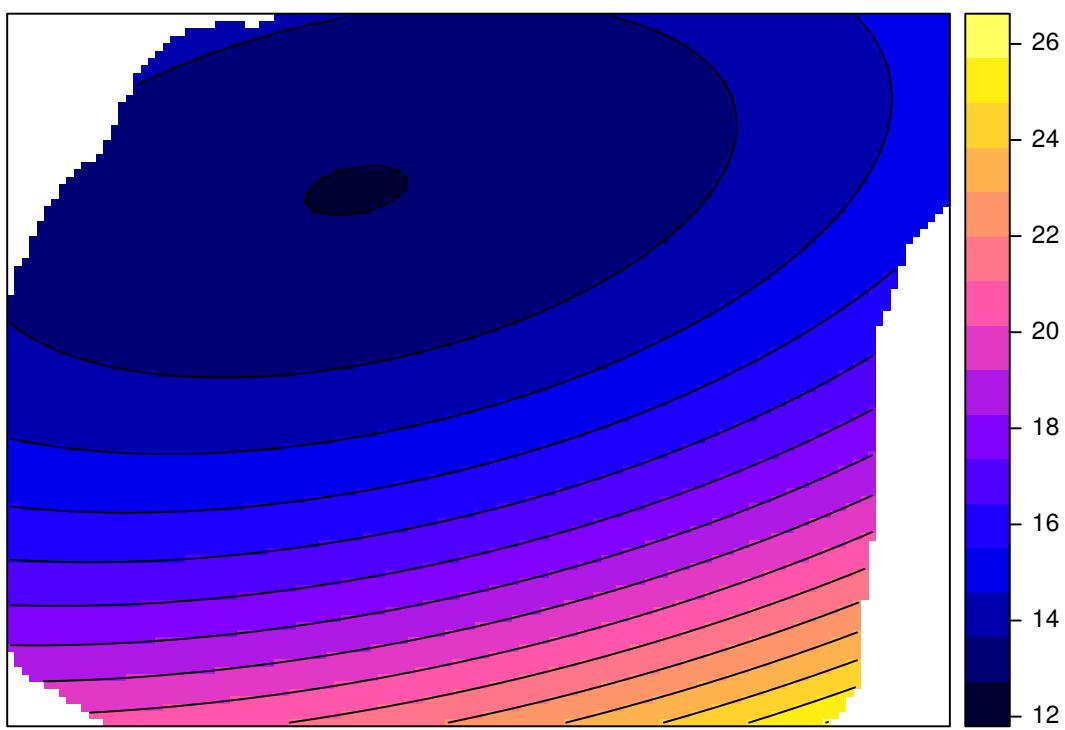
### Powierzchnia trendu - wielomian pierwszego stopnia



```
wielomian_2 <- gstat(formula=temp~1, data=punkty, degree=2)
wielomian_2_pred <- predict(wielomian_2, newdata=siatka)
```

```
## [ordinary or weighted least squares prediction]
spplot(wielomian_2_pred[1], contour=TRUE, main='Powierzchnia trendu - wielomian drugiego stopnia')
```

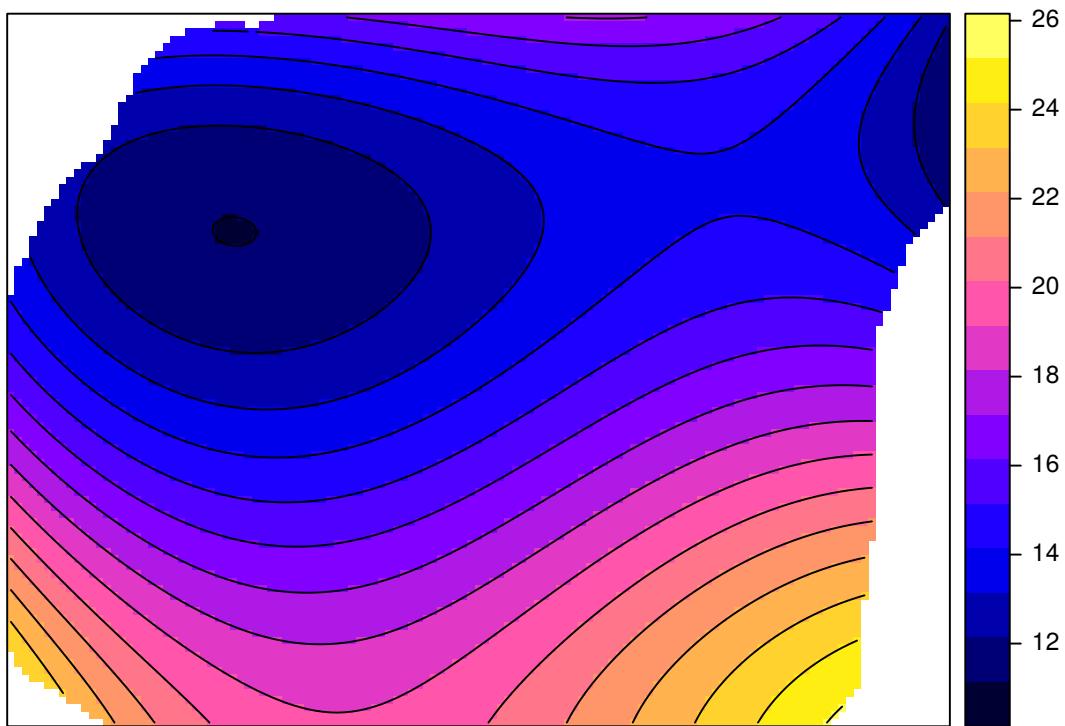
### Powierzchnia trendu - wielomian drugiego stopnia



```
wielomian_3 <- gstat(formula=temp~1, data=punkty, degree=3)
wielomian_3_pred <- predict(wielomian_3, newdata=siatka)
```

```
## [ordinary or weighted least squares prediction]
spplot(wielomian_3_pred[1], contour=TRUE, main='Powierzchnia trendu - wielomian trzeciego stopnia')
```

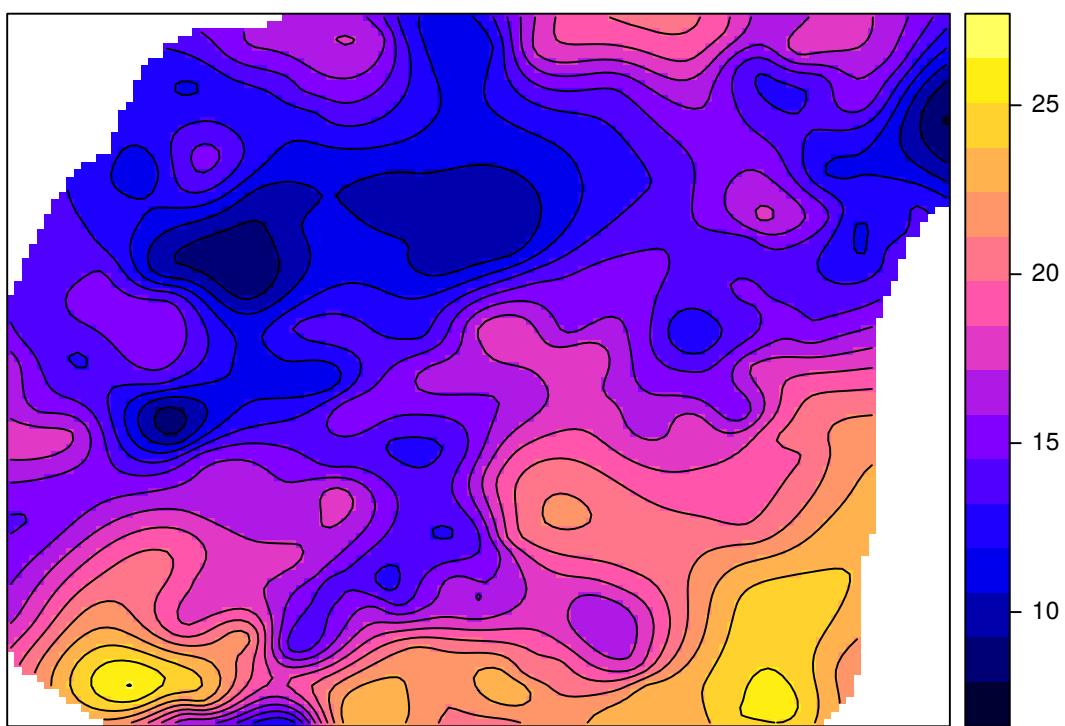
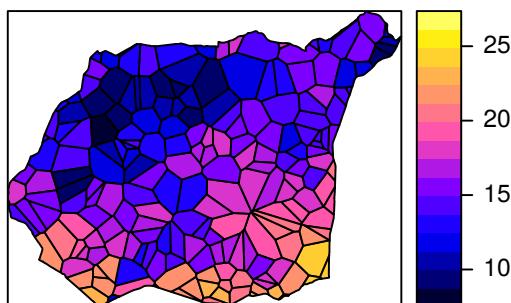
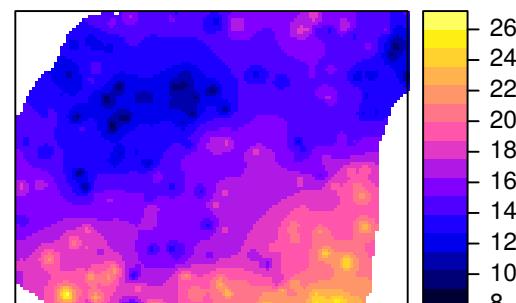
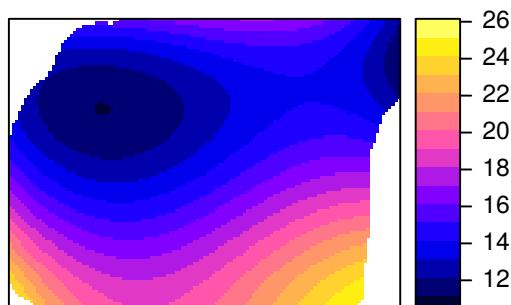
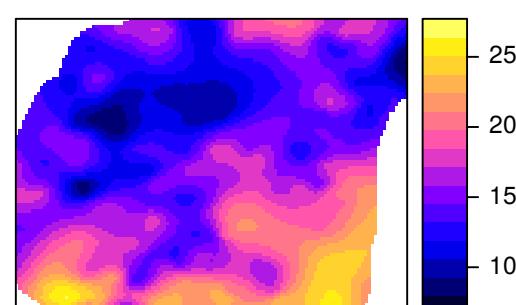
## Powierzchnia trendu - wielomian trzeciego stopnia



### 5.1.4 Modele deterministyczne | Funkcje sklejane

Interpolacja z użyciem funkcji sklejanych (funkcja `Tps()` z pakietu `fields`) dopasowuje krzywą powierzchnię do wartości analizowanych punktów.

```
tps <- Tps(coordinates(punkty), punkty@data$temp)
ras <- raster(siatka)
spline <- interpolate(ras, tps)
spline <- mask(spline, ras)
spplot(spline, contour=TRUE , main='Funkcje sklejane')
```

**Funkcje sklejane****5.1.5 Modele deterministyczne | Porównanie****Poligony Voronoi****IDW****Wielomiany****Funkcje sklejane**

## 5.2 Modele statystyczne

Modele deterministyczne charakteryzują się tym, że ich parametry określane są w oparciu o teorię prawdopodobieństwa. Dodatkowo wynik estymacji zawiera także oszacowanie błędu. Te metody zazwyczaj wymagają większych zasobów sprzętowych. Do modeli deterministycznych należą, między innymi:

- Kriging
- Modele regresyjne
- Modele bayesowe
- Modele hybrydowe

W kolejnych rozdziałach można znaleźć omówienie kilku podstawowych typów pierwszej z tych metod - krigingu.

# Rozdział 6

## Geostatystyka - prolog

```
library('gstat')
library('sp')
library('pgirmess')
library('ggplot2')
library('raster')
library('rasterVis')
library('geostatbook')
data(punkty)
data(siatka)
data(granica)
```

### 6.1 Geostatystyka

#### 6.1.1 Geostatystyka

Geostatystyka jest to zbiór narzędzi statystycznych uwzględniających w analizie danych ich przestrenną i czasową lokalizację, a opartych o teorię funkcji losowych.

#### 6.1.2 Geostatystyka | Funkcje

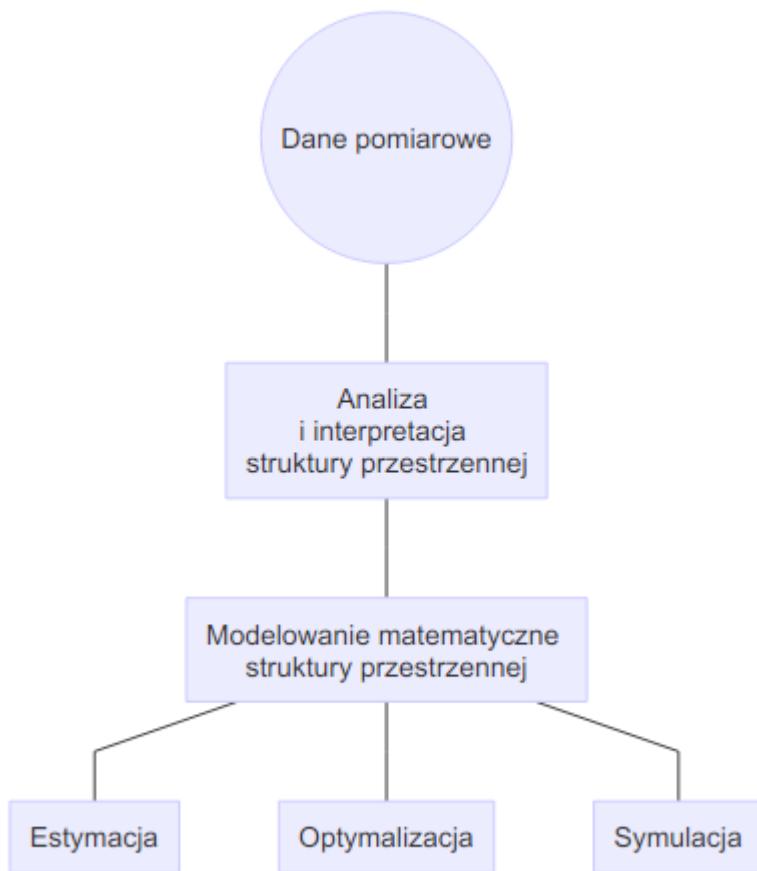
Istnieją cztery główne funkcje geostatystyki:

1. Identyfikacja i modelowanie struktury przestrzennej/czasowej zjawiska
2. Estymacja - szacowanie wartości badanej zmiennej w nieopróbowanym miejscu i/lub momencie czasu
3. Symulacja - generowanie alternatywnych obrazów, które honorują wyniki pomiarów i strukturę przestrenną/czasową zjawiska
4. Optymalizacja próbkowania/sieci pomiarowej

Inaczej mówiąc, celem geostatystyki nie musi być tylko interpolacja (estymacja) przestrzenna, ale również zrozumienie zmienności przestrzennej lub czasowej zjawiska, symulowanie wartości, oraz optymalizacja sieci pomiarowej.

#### 6.1.3 Geostatystyczna analiza danych

Geostatystyczna analiza danych może przyjmować różną postać w zależności od postawionego celu analizy. Poniższa rycina przedstawia uproszczoną ścieżkę postępowania geostatystycznego.



#### 6.1.4 Geostatystyka | Podstawowe etapy

W przypadku estymacji geostatystycznej, zwanej inaczej interpolacją geostatystyczną, pełna ścieżka postępowania składa się z siedmiu elementów:

1. Zaprojektowanie sposobu (typu) próbkowania oraz organizacji zadań
2. Zebranie danych, analiza laboratoryjna
3. Wstępna eksploracja danych, ocena ich jakości
4. Modelowanie semiwariogramów na podstawie dostępnych danych
5. Estymacja badanej cechy
6. Porównanie i ocena modeli
7. Stworzenie wynikowego produktu i jego dystrybucja

#### 6.1.5 Geostatystyka | Dane wejściowe

Jedną z najważniejszych ograniczeń stosowania metod geostatystycznych jest spełnienie odpowiednich założeń dotyczących danych wejściowych. Muszą one:

1. Zawierać wystarczająco dużą liczbę punktów (minimalnie  $>30$ , ale zazwyczaj więcej niż 100/150)
2. Być reprezentatywne
3. Być niezależne
4. Być stworzone używając stałej metodyki
5. Być wystarczająco dokładne

## 6.2 Przestrzenna kowariancja i korelacja i semiwariancja

### 6.2.1 Przestrzenna kowariancja, korelacja i semiwariancja | Założenia

Oprócz ograniczeń dotyczących danych wejściowych, istnieją również założenia dotyczące analizowanej cechy (analizowanego zjawiska):

1. Przestrzennej ciągłości - przestrzenna korelacja między zmienną w dwóch lokalizacjach zależy tylko od ich odległości (oraz czasem kierunku), lecz nie od tego gdzie są one położone
2. Stacjonarności - średnia i wariancja są stałe na całym badanym obszarze

### 6.2.2 Przestrzenna kowariancja, korelacja i semiwariancja | Symbole

Podstawowe symbole w określaniu przestrzennej zmienności to:

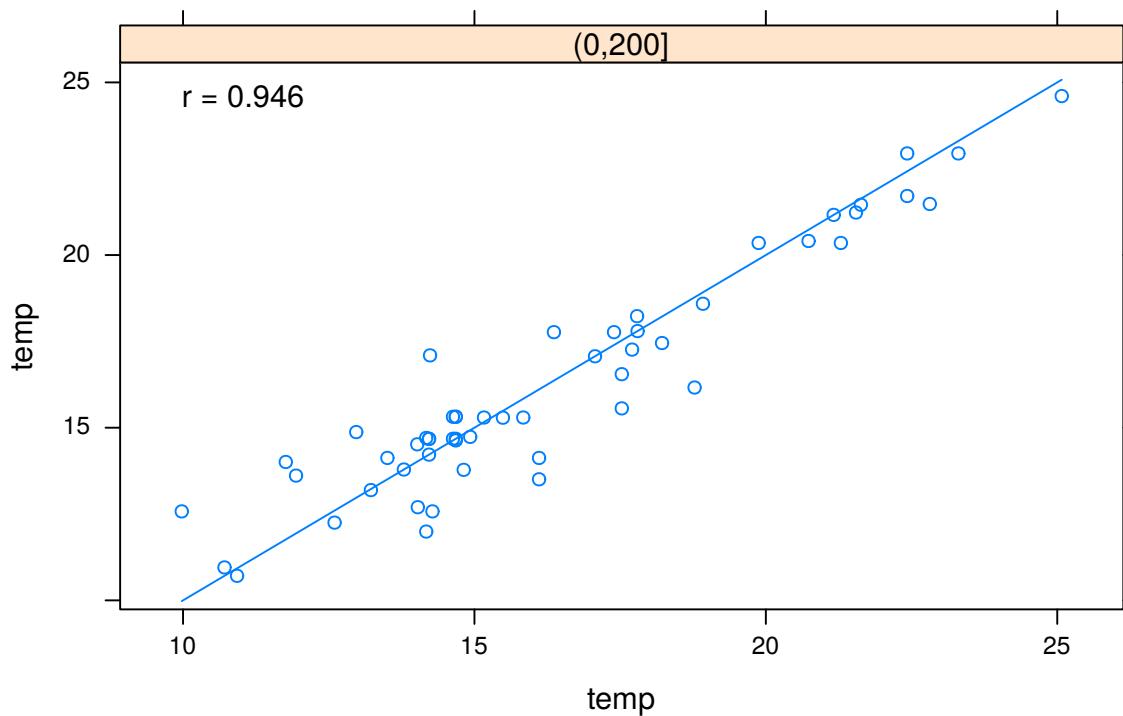
- $u$  - wektor współrzędnych
- $z(u)$  - badana zmienna jako funkcja położenia - inaczej określany jako ogon (ang. *tail*)
- $h$  - lag - odstęp pomiędzy dwoma lokalizacjami
- $z(u + h)$  - wartość badanej zmiennej odległej o odstęp  $h$  - inaczej określany jako głowa (ang. *head*)

### 6.2.3 Przestrzenna kowariancja, korelacja i semiwariancja

Przestrzenna kowariancja, korelacja i semiwariancja to miary określające przestrzenną zmienność analizowanej cechy.

- Kowariancja i korelacja to miary podobieństwa pomiędzy dwoma zmiennymi
- Przenosząc to na aspekt przestrzenny, badamy jedną zmienną, ale pomiędzy dwoma punktami odległymi od siebie o pewien dystans (określany jako lag)
- W efekcie otrzymujemy miarę podobieństwa pomiędzy wartością głową i ogona
- Trzecią miarę charakteryzującą relację między obserwacjami odległymi o kolejne odstępy jest semiwariancja
- Z praktycznego punktu widzenia, semiwariogram jest preferowaną miarą relacji przestrzennej, ponieważ wykazuje tendencję do lepszego wygładzania danych niż funkcja kowariancji
- Dodatkowo, semiwariogram jest mniej wymagający obliczeniowo
- Jednocześnie, dla potrzeb interpretacji relacji, kowariancja i korelacja przestrzenna nadaje się nie gorzej niż semiwariancja

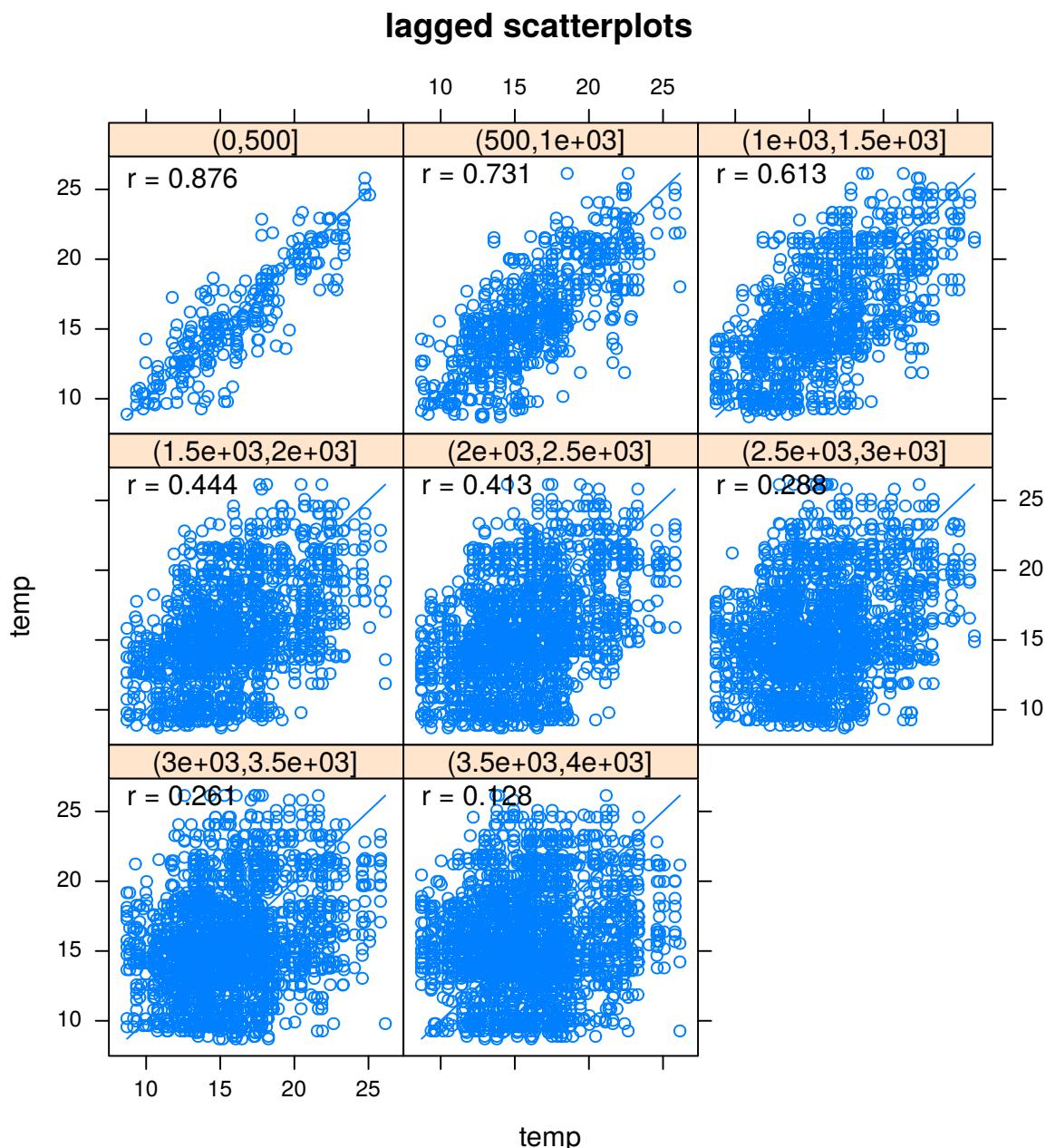
## lagged scatterplots



### 6.2.4 Wykres rozrzutu z przesunięciem

Wykres rozrzutu z przesunięciem pokazuje korelację pomiędzy wartościami analizowanej cechy w pewnych grupach odległości. Taki wykres można stworzyć używając funkcji `hscat()` z pakietu `gstat`. Przykładowo, na poniższym wykresie widać wartość cechy `temp` z kolejnymi przesunięciami - od 0 do 500 metrów, od 500 metrów do 1000 metrów, itd. W pierwszym przedziale wartość cechy `temp` z przesunięciem wykazuje korelację na poziomie 0,876, a następnie z każdym kolejnym przedziałem (odlegością) ta wartość maleje. W przedziale 3500 do 4000 metrów osiąga jedynie 0,128. Pozwala to na stwierdzenie, że cecha `temp` wykazuje zmienność przestrzenną - podobieństwo jej wartości zmniejsza się wraz z odlegością.

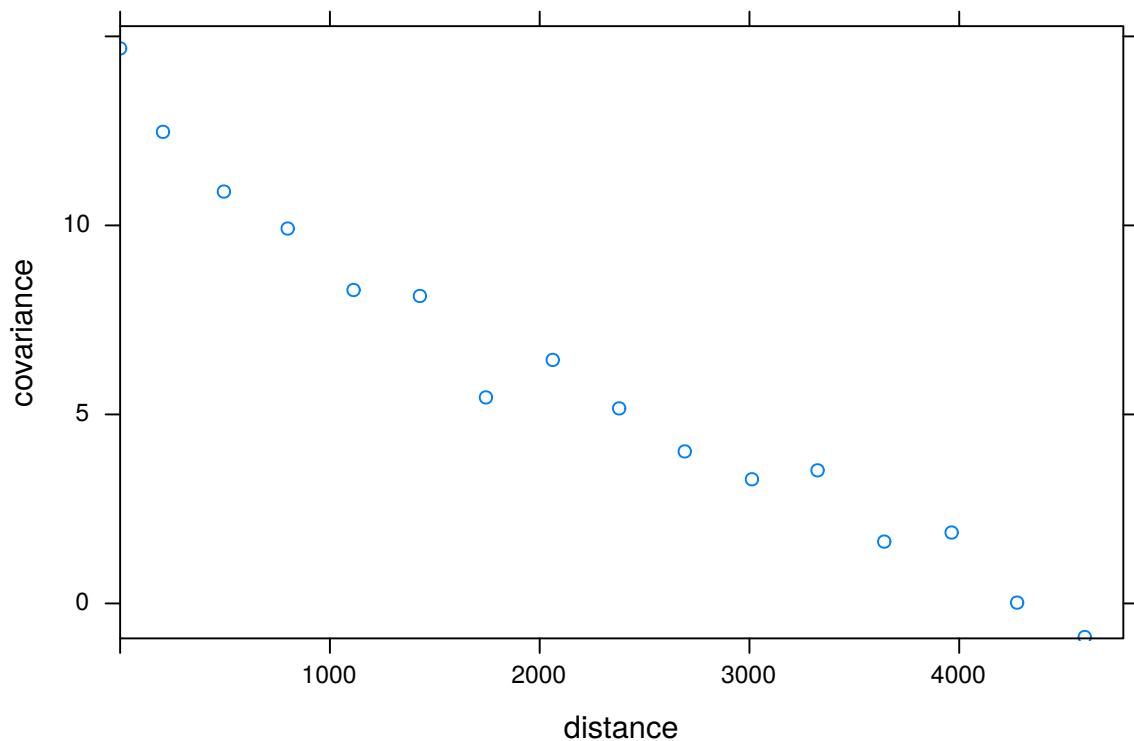
```
hscat(temp~1, punkty, breaks=seq(0, 4000, by=500))
```



### 6.2.5 Autokowariancja

Podobną informację jak wykres rozrzutu z przesunięciem daje autokowariancja. Pokazuje ona jak mocno przestrzennie powiązane są wartości par obserwacji odległych od siebie o kolejne przedziały. Jej wyliczenie jest możliwe z użyciem funkcji `variogram()` z pakietu `gstat`, gdzie definiuje się analizowaną zmienną, zbiór punktowy, oraz ustala się argument `covariogram` na TRUE.

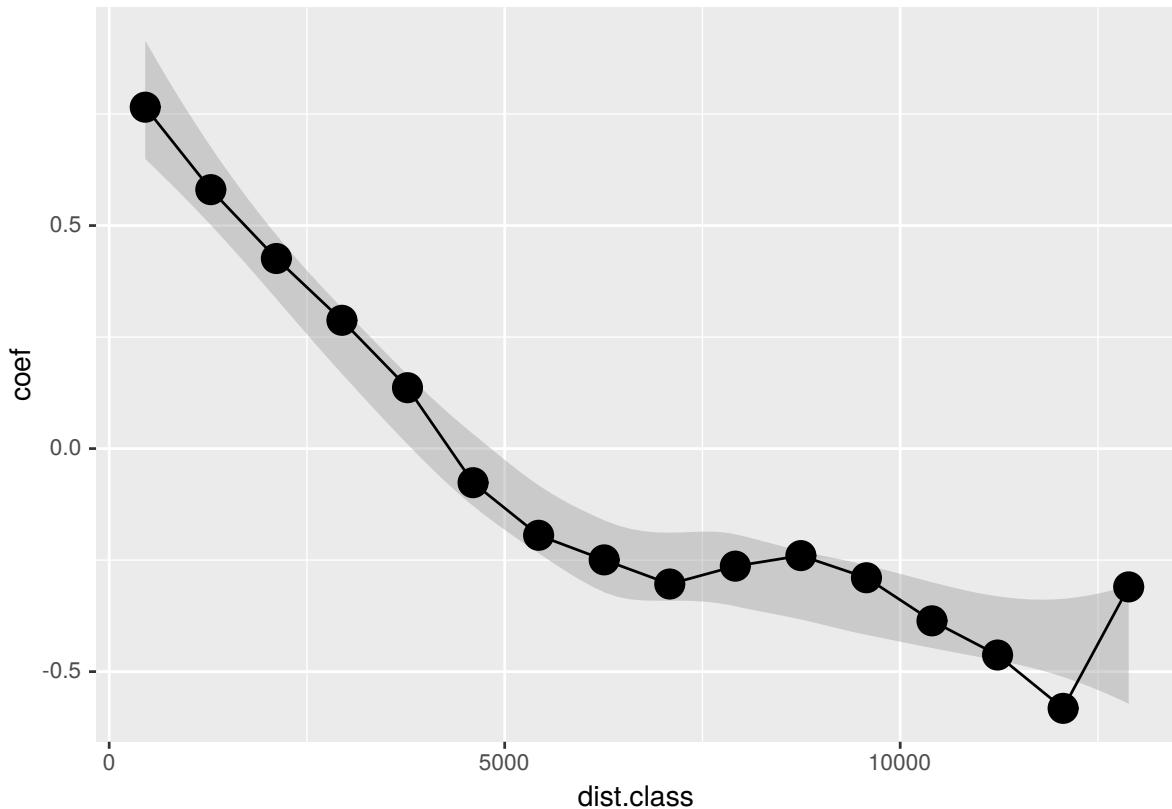
```
kowario <- variogram(temp~1, punkty, covariogram = TRUE)
plot(kowario)
```



### 6.2.6 Autokorelacja

Kolejną miarą przestrzennego podobieństwa jest autokorelacja. Jej wykres (autokorelogram) pokazuje wartość jednej z miar autokorelacji (np. I Morana lub C Geary'ego) w stosunku do odległości. Na poniższym przykładzie, wartość statystyki I Morana jest wyliczana poprzez funkcję `correlog()` z pakietu `pgirmess`.

```
wsp <- coordinates(punkty)
kor <- correlog(wsp, punkty$temp, method='Moran')
kor <- as.data.frame(kor)
ggplot(kor, aes(dist.class, coef)) + geom_smooth(linetype=0) + geom_line() + geom_point(size=5)
```



### 6.2.7 Semiwariancja

Zmienna przestrzenna analizowanej cechy może być określona za pomocą semiwariancji. Jest to połowa średniej kwadratu różnicy pomiędzy wartościami badanej zmiennej w dwóch lokalizacjach odległych o wektor  $h$ :

$$\gamma(h) = \frac{1}{2} E[(z(s) - z(s + h))^2]$$

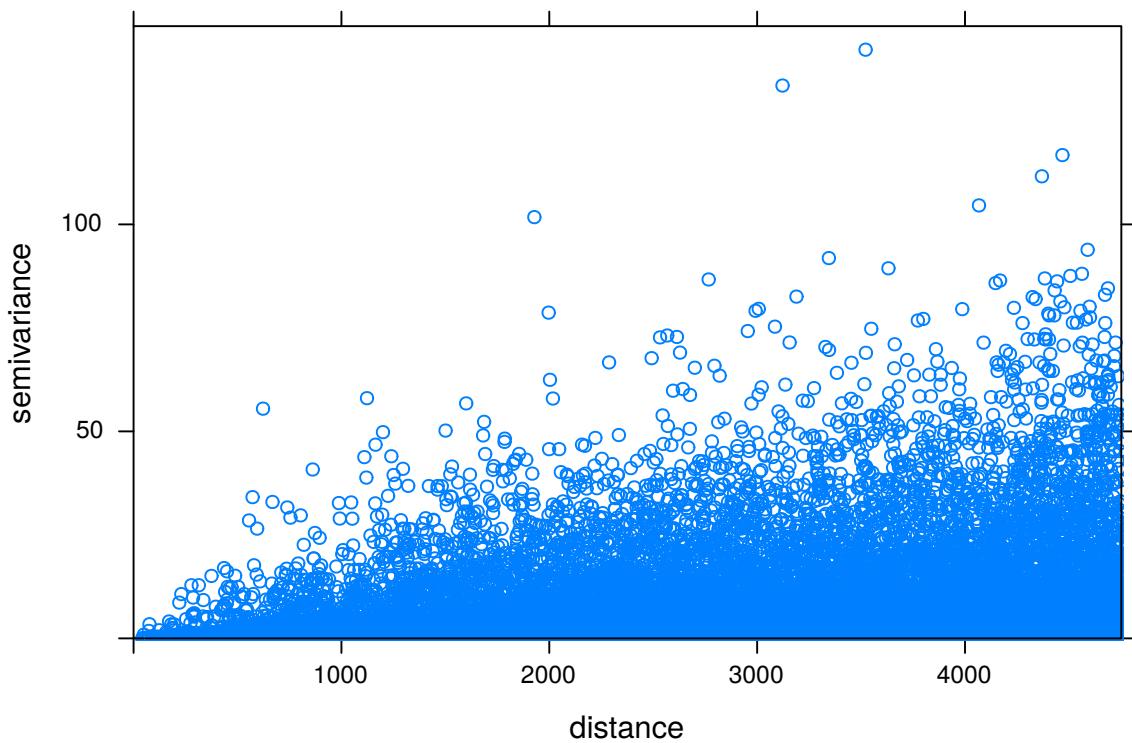
Przykładowo, aby wyliczyć wartość semiwariancji (`gamma`) pomiędzy dwoma punktami musimy znać wartość pierwszego z nich (w przykładzie jest to ok. 22,46 stopni Celsjusza) oraz drugiego z nich (ok. 16,04 stopni Celsjusza). Korzystając z wzoru na semiwariację otrzymujemy wartość równą ok. 20,60. Znamy również odległość między punktami (ok. 4576,59 metra), więc możemy w uproszczeniu stwierdzić, że dla tej pary punktów odległych o 4577 metry wartość semiwariancji wynosi około 20,6.

```
odl <- dist(coordinates(punkty)[c(1, 2), ])
gamma <- 0.5 * (punkty$temp[1] - punkty$temp[2])^2
gamma
## [1] 20.60122
```

### 6.2.8 Chmura semiwariogramu

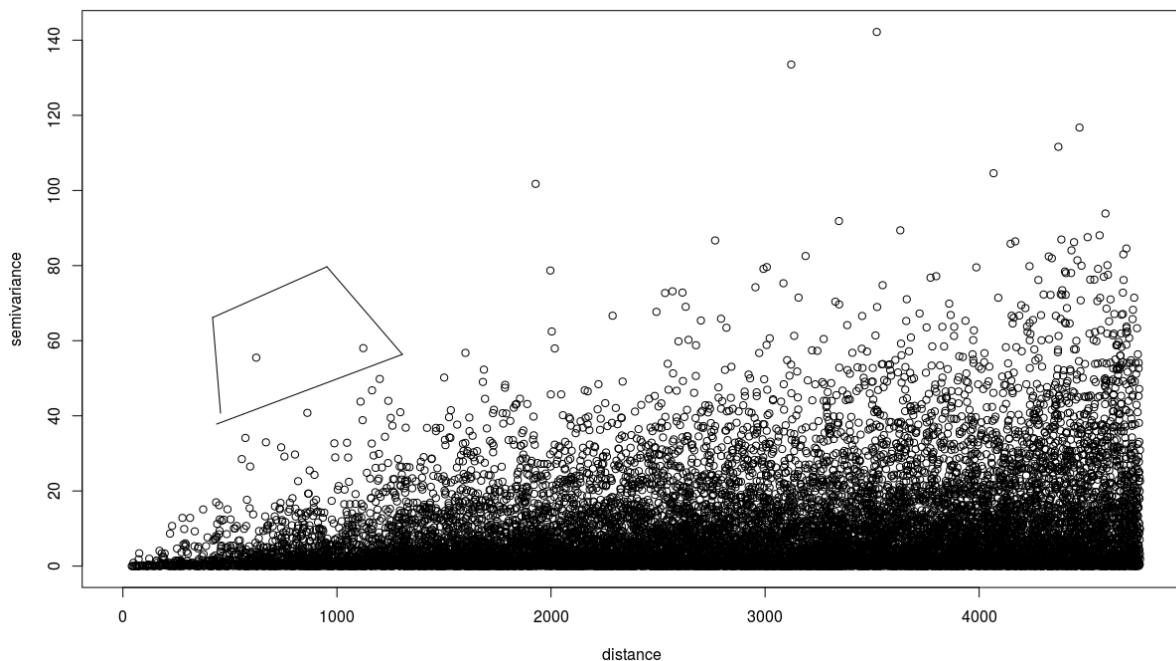
Jeżeli w badanej próbie mamy  $n$  obserwacji oznacza to, że możemy zaobserwować  $\frac{1}{2}n(n-1)$  par obserwacji. Każda z tych par obserwacji daje nam informację o semiwariancji występującej wraz z odległością. Tę semiwariancję można zaprezentować na wykresie zwanym chmurą semiwariogramu. Do jej wyliczenia służy funkcja `variogram()` z argumentem `cloud=TRUE`.

```
vario_cloud <- variogram(temp~1, punkty, cloud=TRUE)
plot(vario_cloud)
```



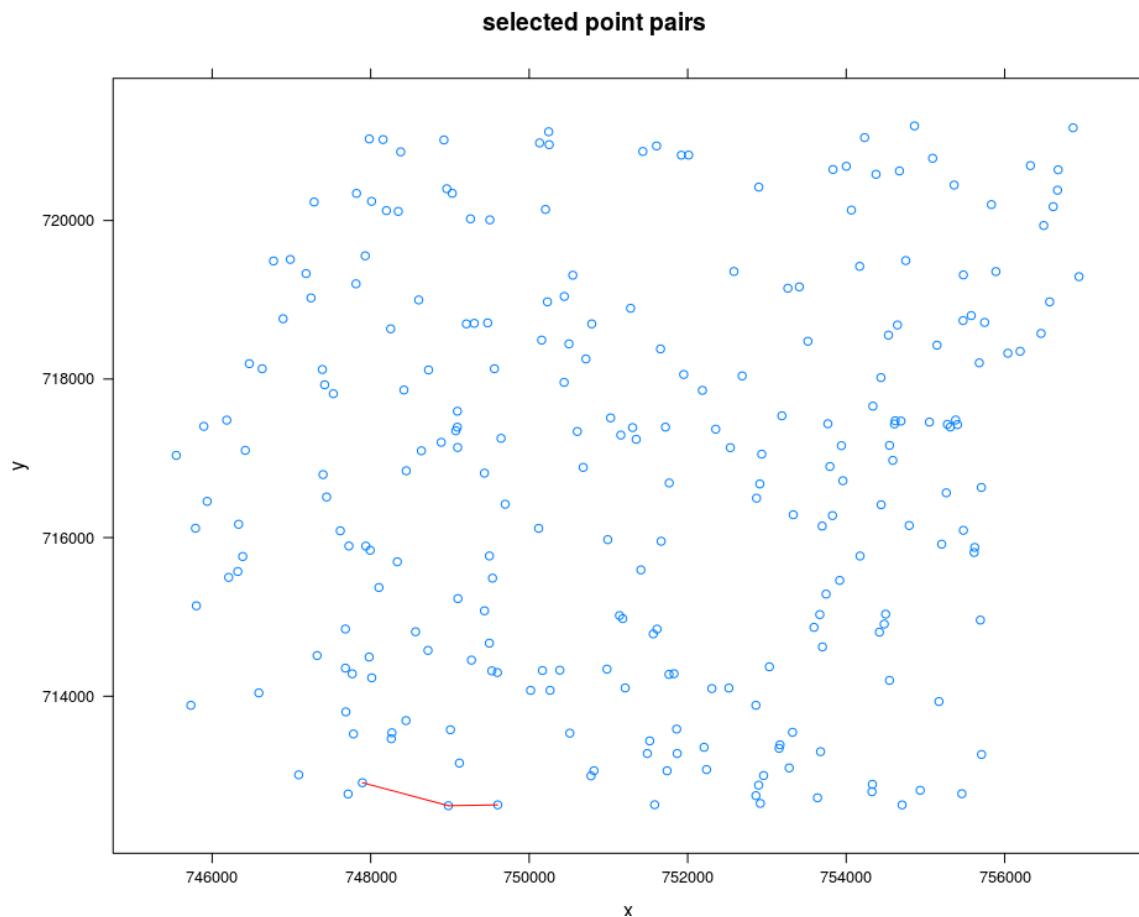
Chmura semiwariogramu pozwala także na zauważenie par punktów, których wartości znaczająco odstają. Aby zlokalizować te pary punktów, można zastosować funkcję `plot()` z argumentem `digitize=TRUE`, a następnie za pomocą kilku kliknięć określić obszar zawierający nietypowe wartości. Po skończonym wyborze należy nacisnąć klawisz Esc.

```
vario_cloud_sel <- plot(variogram(temp~1, punkty, cloud=TRUE), digitize=TRUE)
```



Następnie można wyświetlić wybrane pary punktów z użyciem funkcji `plot()`.

```
plot(vario_cloud_sel, punkty)
```



### 6.2.9 Semivariogram | Charakterystyka struktury przestrzennej

Semivariogram jest wykresem pokazującym relację pomiędzy odległością a semiwariancją. Inaczej mówiąc, dla kolejnych odstępów (lagów) wartość semiwariancji jest uśredniana i przedstawiana w odniesieniu do odległości.

$$\hat{\gamma}(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} (z(s_i) - z(s_i + h))^2$$

, gdzie  $N(h)$  oznacza liczbę par punktów w odstępie  $h$ .

W oparciu o semivariogram empiryczny (czyli oparty na danych) możemy następnie dopasować do niego model/e.

### 6.2.10 Semivariogram | Obliczenia pomocnicze

- Liczba par obserwacji:

```
0.5 * length(punkty) * (length(punkty) - 1)
```

```
## [1] 31125
```

- Połowa pierwiastka powierzchni:

```
pow <- area(granica)
as.vector(0.5 * sqrt(pow))
```

```
## [1] 3980.472
• Powierzchnia zajmowana przez jedną próbkę:
pow_pr <- area(granica) / length(punkty)
pow_pr

## [1] 253506.6
• Średnia odległość między punktami :
sqrt(pow_pr)

## [1] 503.4944
```

### 6.2.11 Semiwariogram

Semiwariogram składa się z trzech podstawowych elementów. Są to:

- **Nugget** - efekt nuggetowy - pozwala na określenie błędu w danych wejściowych oraz zmienności na dystansie krótszym niż pierwszy odstęp
- **Sill** - semiwariancja progowa - oznacza wariancję badanej zmiennej
- **Range** - zasięg - to odległość do której istnieje przestrzenna korelacja

### 6.2.12 Semiwariogram | Rules of thumb

Przy ustalaniu parametrów semiwariogramu powinno się stosować do kilku utartych zasad (tzw. *rules of thumb*):

- W każdym odstępie powinno się znaleźć co najmniej 30 par punktów
- Maksymalny zasięg semiwariogramu (ang. *cutoff distance*) to 1/2 pierwiastka z badanej powierzchni (inne źródła mówią o połowie z przekątnej badanego obszaru/jednej trzeciej)
- Liczba odstępów powinna nie być mniejsza niż 10
- Optymalnie maksymalny zasięg semiwariogramu powinien być dłuższy o 10-15% od zasięgu zjawiska
- Optymalne odstępy powinny być jak najbliżej siebie i jednocześnie nie być chaotyczne
- Warto metodą prób i błędów określić optymalne parametry semiwariogramu
- Należy określić czy zjawisko wykazuje anizotropię przestrzenną

### 6.2.13 Semiwariogram

Stworzenie podstawowego semiwariogramu w pakiecie **gstat** odbywa się z użyciem funkcji **variogram()**. Należy w niej zdefiniować analizowaną zmienną (w tym przykładzie **temp~1**) oraz zbiór punktowy (**punkty**).

```
vario_par <- variogram(temp~1, punkty)
```

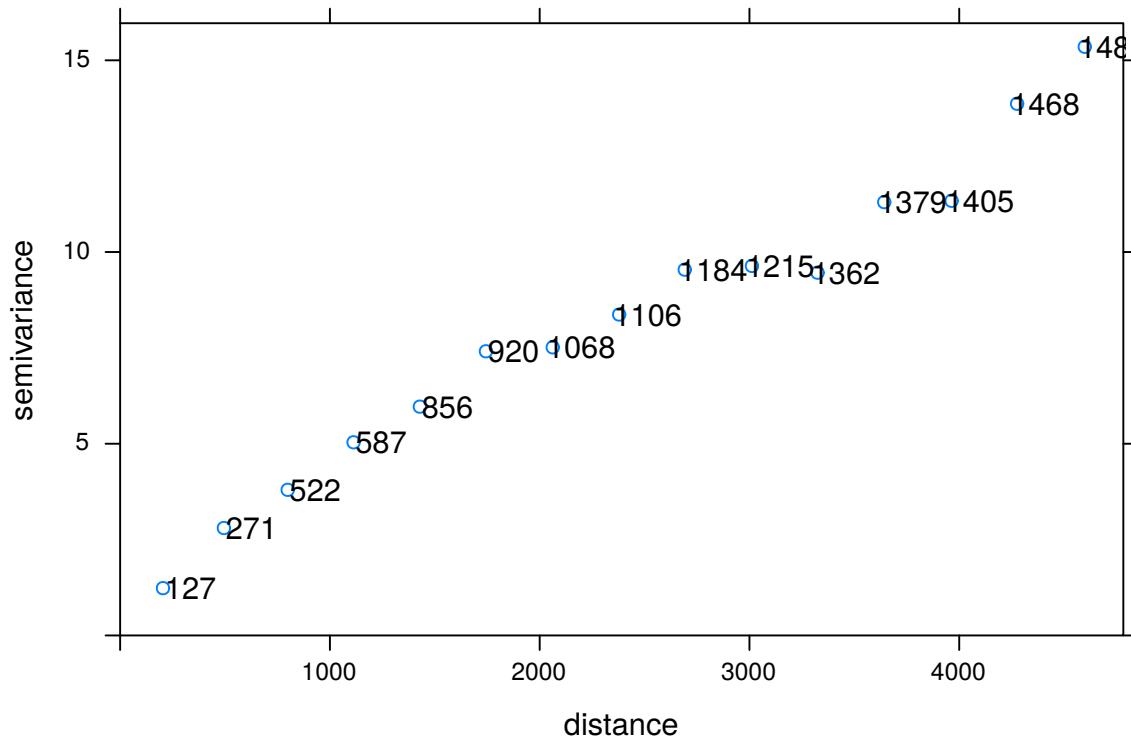
```
vario_par
```

	np	dist	gamma	dir.hor	dir.ver	id
## 1	127	204.2244	1.233999	0	0	var1
## 2	271	494.5572	2.802019	0	0	var1
## 3	522	798.7911	3.797612	0	0	var1
## 4	587	1112.7783	5.037545	0	0	var1
## 5	856	1428.7866	5.967276	0	0	var1
## 6	920	1743.7864	7.411276	0	0	var1
## 7	1068	2062.3041	7.514067	0	0	var1
## 8	1106	2378.9333	8.369087	0	0	var1
## 9	1184	2691.5206	9.539870	0	0	var1
## 10	1215	3011.7729	9.636891	0	0	var1
## 11	1362	3324.6705	9.461429	0	0	var1
## 12	1379	3642.5616	11.301515	0	0	var1
## 13	1405	3963.6776	11.335589	0	0	var1

```
## 14 1468 4276.0078 13.866888      0      0 var1
## 15 1482 4598.4144 15.353206      0      0 var1
```

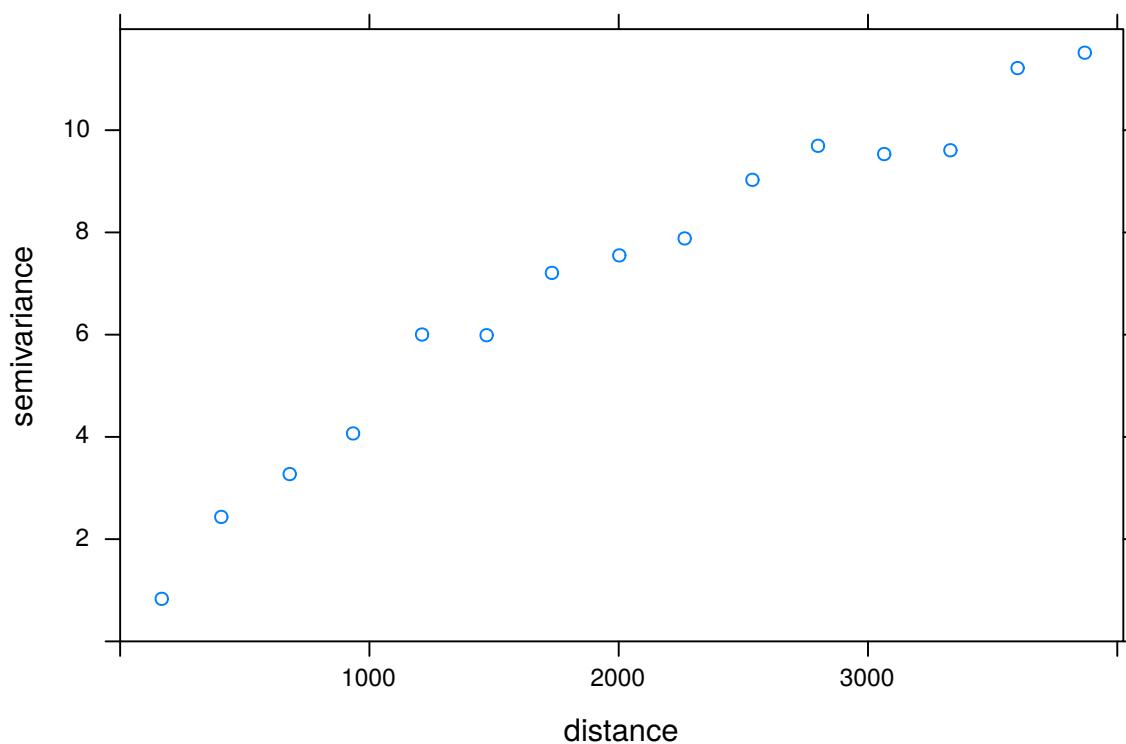
Do wyświetlenia semiwariogramu służy funkcja `plot()`. Można również dodać informację o liczbie par punktów, jaka posłużyła do wyliczenia semiwariancji dla kolejnych odstępów poprzez argument `plot.numbers=TRUE`.

```
plot(vario_par, plot.numbers=TRUE)
```



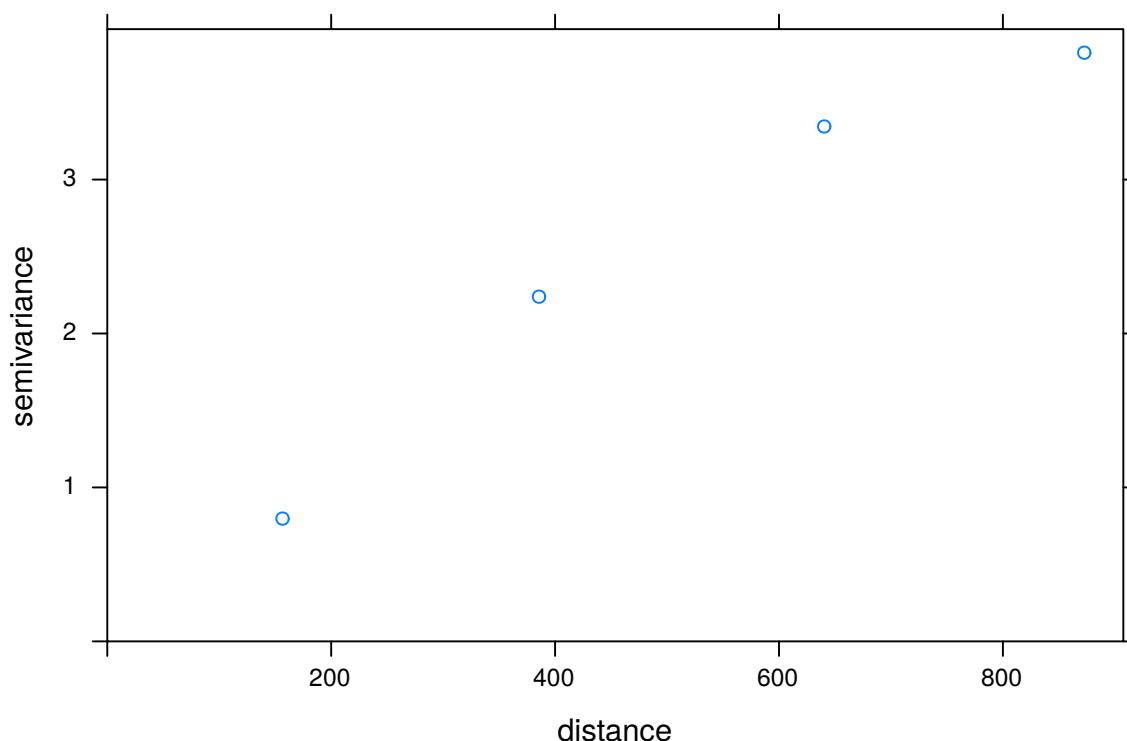
Maksymalny zasięg semiwariogramu (ang. *cutoff distance*) jest domyślnie wyliczany w pakiecie `gstat` jako  $1/3$  z najdłuższej przekątnej badanego obszaru. Można jednak tę wartość zmodyfikować używając argumentu `cutoff`.

```
vario_par <- variogram(temp~1, punkty, cutoff = 4000)
plot(vario_par)
```



Dodatkowo, domyślnie w pakiecie `gstat` odległość między przedziałami (ang. *interval width*) jest wyliczana jako maksymalny zasięg semiwariogramu podzielony przez 15. Można to oczywiście zmienić używając zarówno argumentu `cutoff`, jak i argumentu `width` mówiącego o szerokości odstępów.

```
vario_par <- variogram(temp~1, punkty, cutoff = 1000, width = 250)
plot(vario_par)
```



## 6.3 Anizotropia

### 6.3.1 Anizotropia struktury przestrzennej

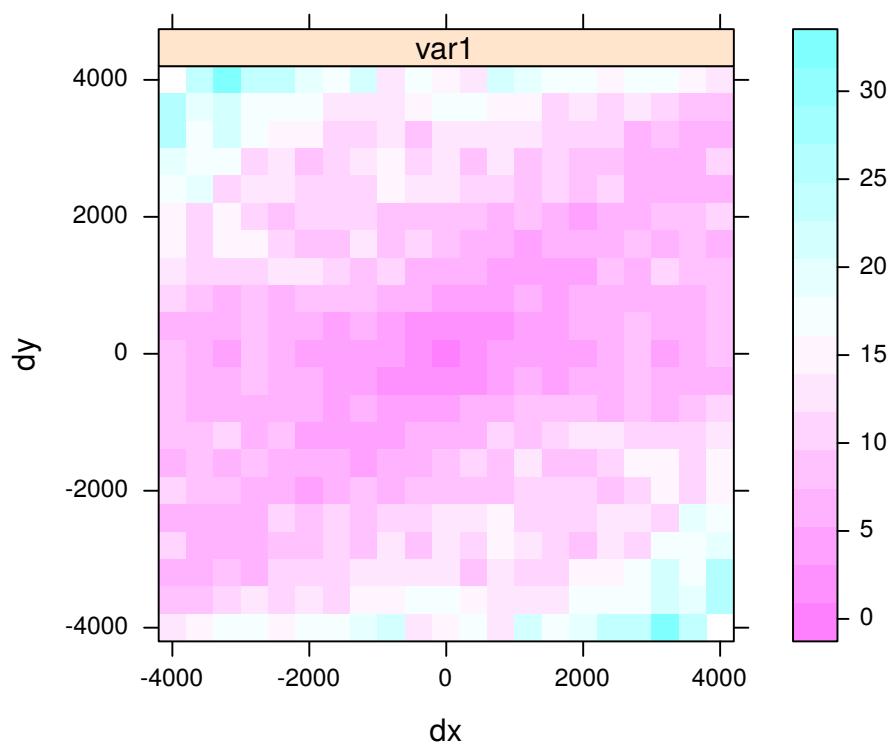
W wielu sytuacjach, wartość cechy zależy nie tylko od odległości, ale także od kierunku. O takim zjawisku mówimy, że wykazuje ono anizotropię struktury przestrzennej.

### 6.3.2 Mapa semiwariogramu

Mapa semiwariogramu (zwana inaczej powierzchnią semiwariogramu) służy do określenia czy struktura przestrzenna zjawiska posiada anizotropię, a jeżeli tak to w jakim kierunku. Na podstawie mapy semiwariogramu identyfikuje się także parametry potrzebne do zbudowania semiwariogramów kierunkowych.

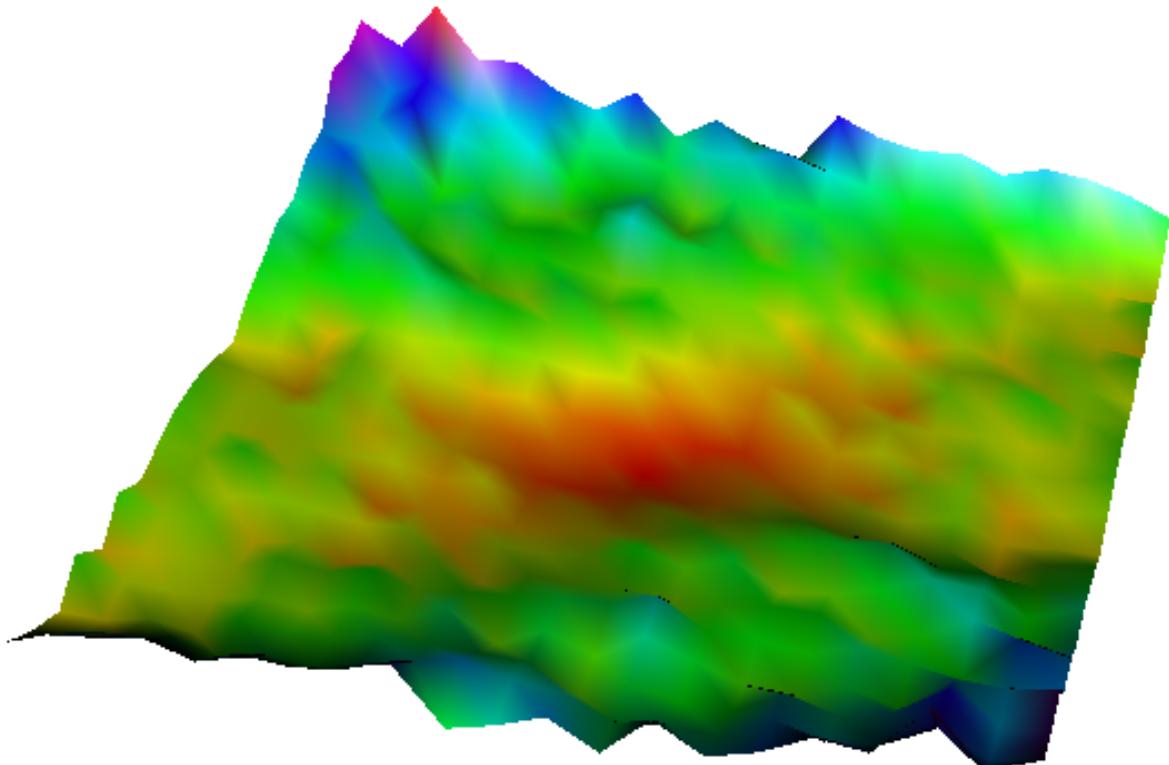
Stworzenie mapy semiwariogramu odbywa się poprzez dodanie kilku argumentów do funkcji `variogram()`: oprócz argumentu zmiennej i zbioru punktowego, jest to `cutoff`, `width`, oraz `map=TRUE`. Następnie można ją wyświetlić używając funkcji `plot()`. Warto w tym wypadku użyć dodatkowego argumentu `threshold`, który powoduje, że niepewna wartość semivariancji (wyliczona na małej liczbie punktów) nie jest wyświetlana.

```
vario_map <- variogram(temp~1, punkty, cutoff=4000, width=400, map=TRUE)
plot(vario_map, threshold=30) # co najmniej 30 par punktów
```



Mapie semiwariogramu można również się przyjrzeć interaktywnie używając funkcję `plot3D()` z pakietu `rasterVis`.

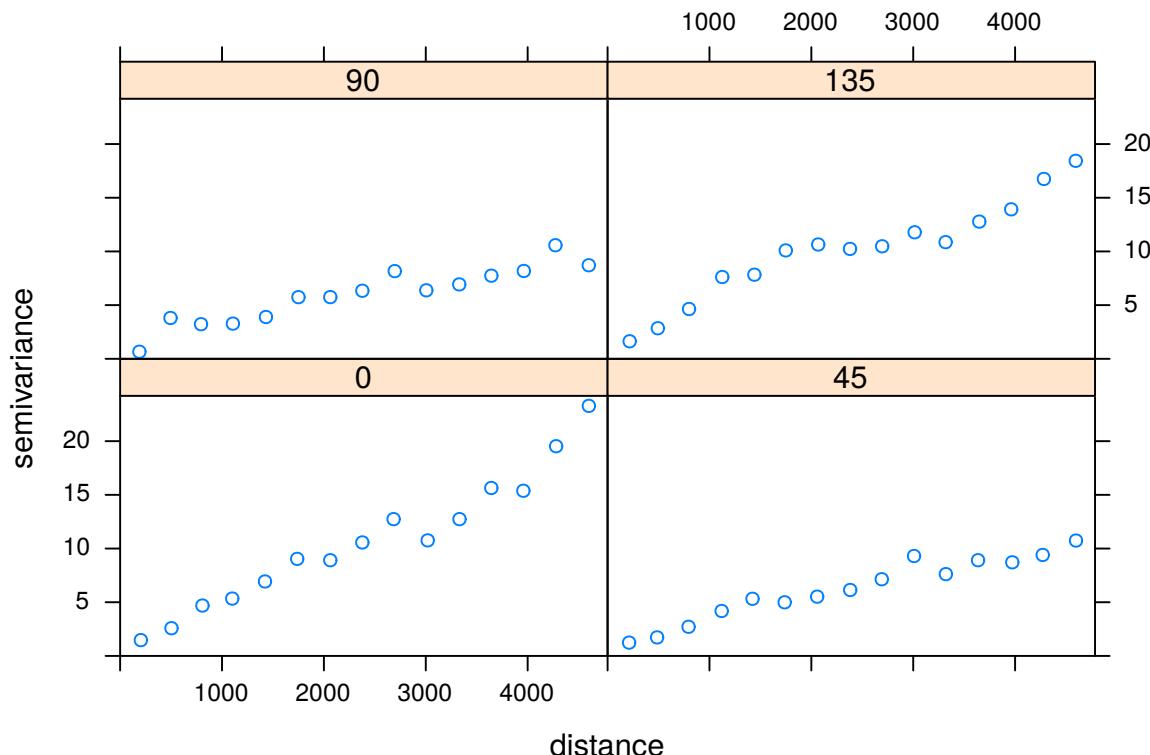
```
plot3D(raster(vario_map$map), col=rainbow)
```



### 6.3.3 Semiwariogramy kierunkowe | Kierunki

W przypadku, gdy zjawisko wykazuje anizotropię przestrzenną, możliwe jest stworzenie semiwariogramów dla różnych wybranych kierunków. Dla argumentu `alpha = c(0, 45, 90, 135)` wybrane cztery główne kierunki to 0, 45, 90 i 135 stopni. Oznacza to, że przykładowo dla kierunku 45 stopni brane pod uwagę będą wszystkie pary punktów pomiędzy 22,5 a 67,5 stopnia.

```
vario_kier <- variogram(temp~1, punkty, alpha = c(0, 45, 90, 135))
plot(vario_kier)
```





## Rozdział 7

# Modelowanie matematycznie autokorelacji przestrzennej

```
library('gstat')
library('geostatbook')
data(punkty)
```

## 7.1 Modelowanie matematycznie autokorelacji przestrzennej

### 7.1.1 Modelowanie matematycznie autokorelacji przestrzennej

Semiwariogram empiryczny (wyliczony z danych punktowych) jest:

- Nieciągły - wartości semiwariancji są średnimi przedziałowymi
- Chaotyczny - badana próba jest jedynie przybliżeniem rzeczywistości, dodatkowo obciążonym błędami

Estymacje i symulacje przestrzenne wymagają modelu struktury przestrzennej analizowanej cechy, a nie tylko wartości empirycznych. Dodatkowo, matematycznie modelowanie wygładza chaotyczne fluktuacje danych empirycznych.

## 7.2 Modele podstawowe

### 7.2.1 Modele podstawowe

Pakiet `gstat` zawiera 20 podstawowych modeli geostatystycznych, w tym najczęściej używane takie jak:

- Nuggetowy (ang. *Nugget effect model*)
- Sferyczny (ang. *Spherical model*)
- Gaussowski (ang. *Gaussian model*)
- Potęgowy (ang. *Power model*)
- Wykładniczy (ang. *Exponential model*)
- Inne

Do wyświetlenia listy nazw modeli i ich skrótów służy funkcja `vgm()`.

```
vgm()
```

```
##   short                      long
## 1   Nug                         Nug (nugget)
## 2   Exp                         Exp (exponential)
## 3   Sph                         Sph (spherical)
```

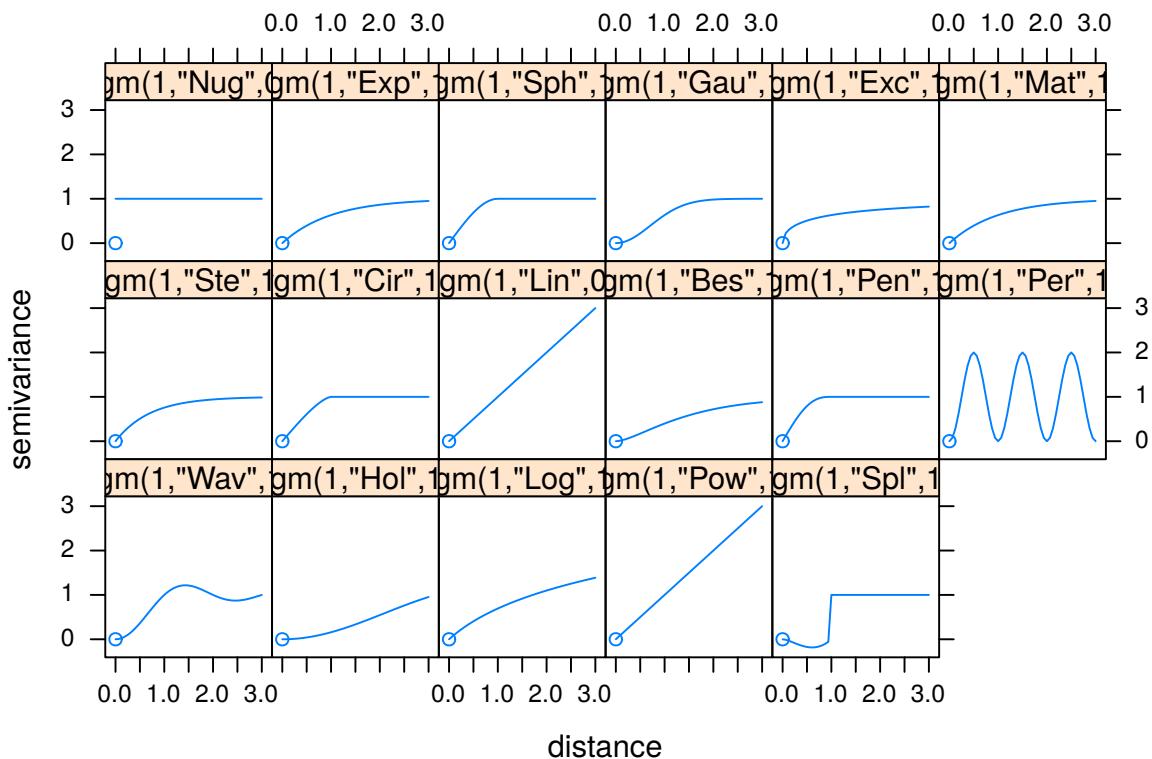
```

## 4   Gau          Gau (gaussian)
## 5   Exc          Exclass (Exponential class/stable)
## 6   Mat          Mat (Matern)
## 7   Ste Mat     (Matern, M. Stein's parameterization)
## 8   Cir          Cir (circular)
## 9   Lin          Lin (linear)
## 10  Bes          Bes (bessel)
## 11  Pen          Pen (pentaspherical)
## 12  Per          Per (periodic)
## 13  Wav          Wav (wave)
## 14  Hol          Hol (hole)
## 15  Log          Log (logarithmic)
## 16  Pow          Pow (power)
## 17  Spl          Spl (spline)
## 18  Leg          Leg (Legendre)
## 19  Err          Err (Measurement error)
## 20  Int          Int (Intercept)

```

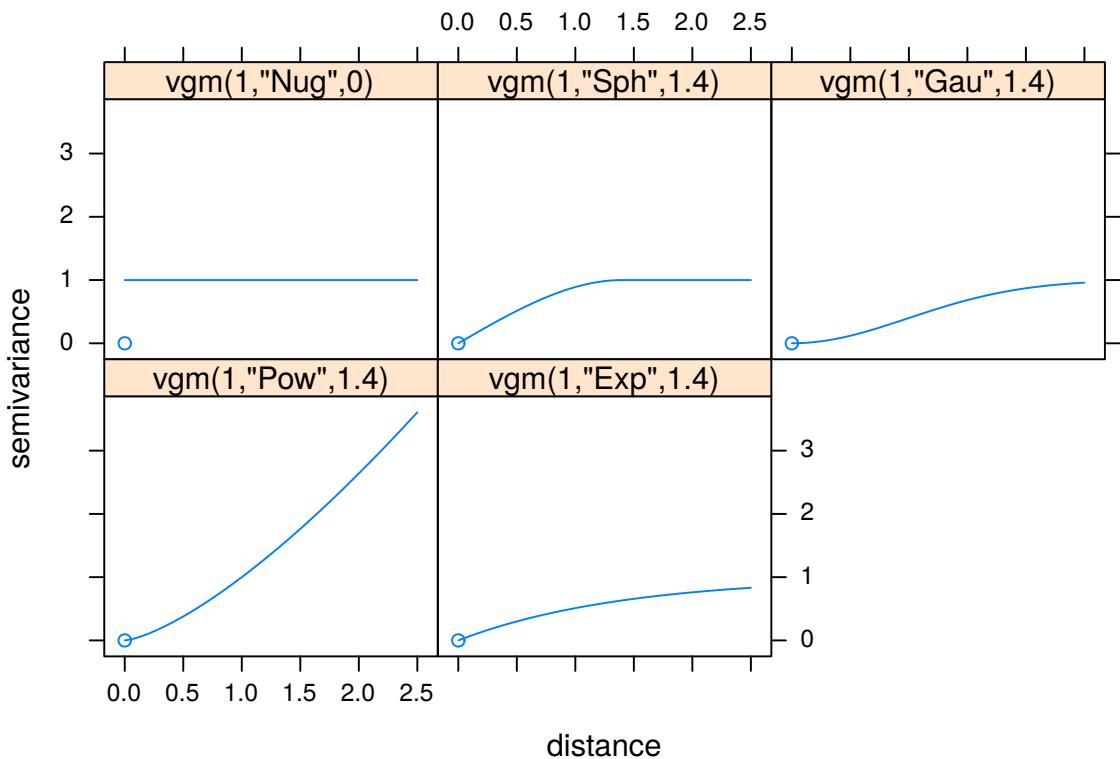
Można się również im przyjrzeć używając funkcji `show.vgms()`.

```
show.vgms()
```



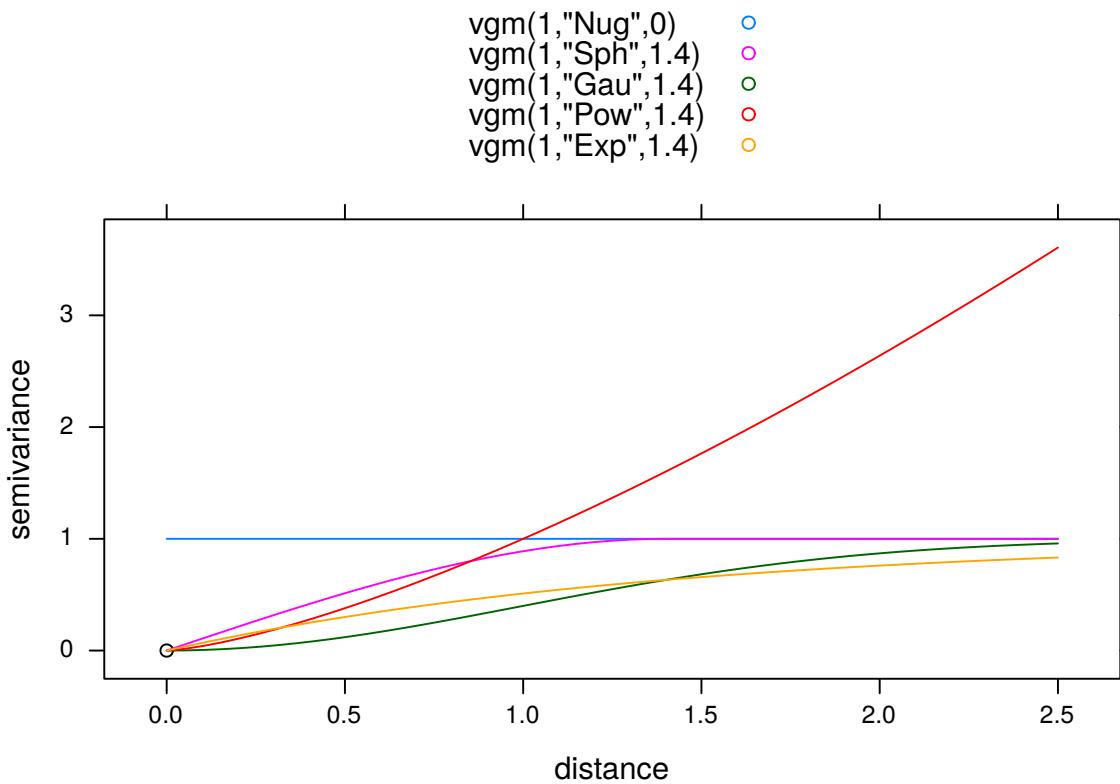
Istnieje możliwość wyświetlania tylko wybranych modeli podstawowych poprzez argument `models`.

```
show.vgms(models=c('Nug', 'Sph', 'Gau', 'Pow', 'Exp'), range=1.4, max=2.5)
```



Dodatkowo, można je porównać na jednym wykresie poprzez argument `as.groups = TRUE`.

```
show.vgms(models=c('Nug', 'Sph', 'Gau', 'Pow', 'Exp'), range=1.4, max=2.5, as.groups = TRUE)
```



## 7.3 Metody modelowania

### 7.3.1 Metody modelowania

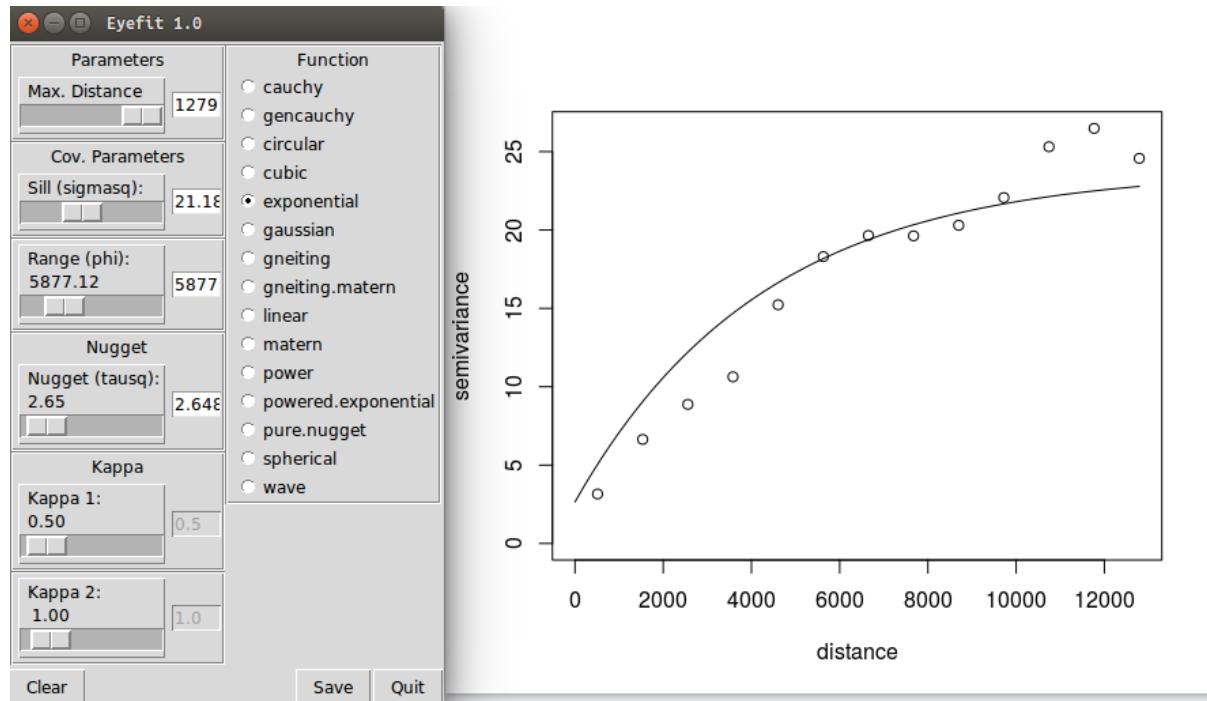
Istnieją trzy najczęściej spotykane metody modelowania geostatystycznego:

- Ustawianie “ręczne” parametrów modelu, np. funkcja `vgm()` z pakietu `gstat`
- Ustawianie “wizualne” parametrów modelu, np. funkcja `eyefit()` z pakietu `geoR`
- Automatyczny wybór parametrów na podstawie różnych kryteriów statystycznych, np. funkcja `fit.variogram()` z pakietu `gstat()`, `variofit()` z pakietu `geoR`, `autofitVariogram()` z pakietu `automap`

Odpowiednie określenie modelu matematycznego często nie jest proste. W efekcie automatyczne metody nie zawsze są w stanie dać lepszy wynik od modelowania “ręcznego”. Najlepiej, gdy wybór modelu oparty jest o wiedzę na temat zakładanego procesu przestrzennego.

### 7.3.2 Metody modelowania | funkcja `eyefit()`

```
v_eye <- eyefit(variog(as.geodata(punkty, 'temp')))
```



### 7.3.3 Metody modelowania | funkcja `fit.variogram()`

Funkcja `fit.variogram()` z pakietu `gstat` dopasowuje zasięg oraz semiwariancję progową w oparciu o ustalone “ręcznie” parametry modelu

### 7.3.4 Metody modelowania | Liniowy model regionalizacji

W przypadku, gdy analizowane zjawisko jest złożone, odwzorowanie kształtu semiwariogramu empirycznego wymaga połączenia dwóch lub większej liczby modeli podstawowych. W takiej sytuacji konieczne jest spełnienie dwóch warunków:

- Wszystkie zastosowane modele muszą być dopuszczalne (`vgm()`)
- Wariancja progowa każdego podstawowego modelu musi być dodatnia

## 7.4 Modelowanie izotropowe

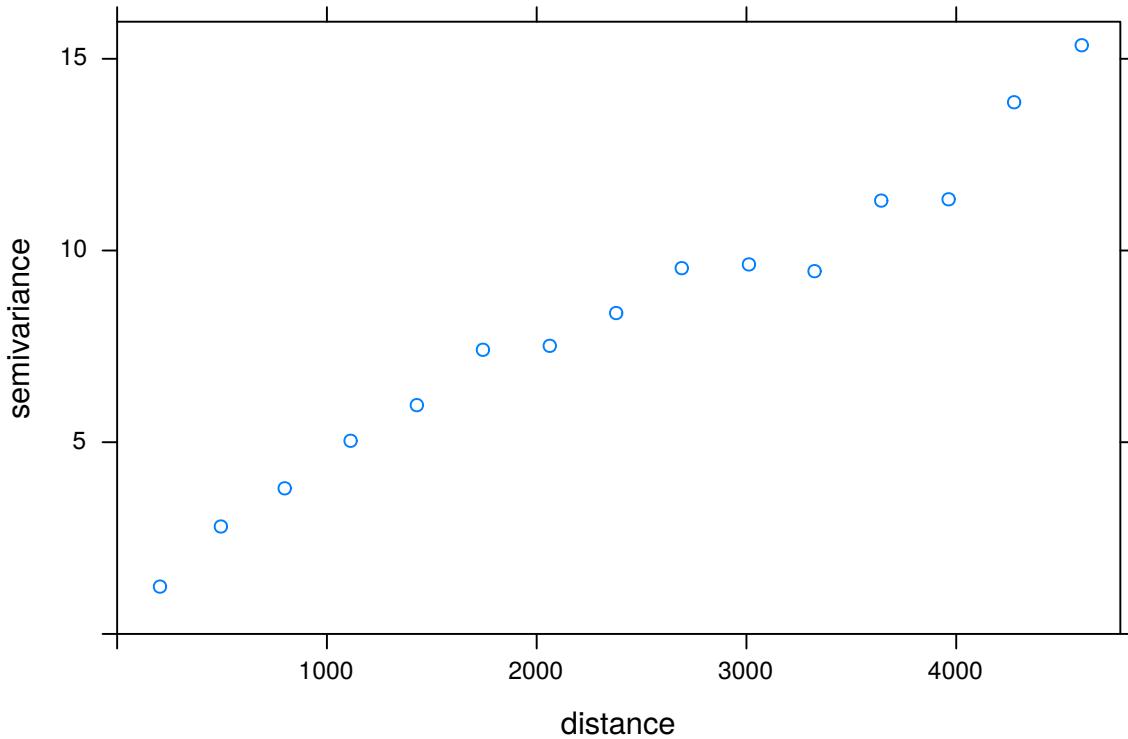
Do zbudowania modelu semiwariogramu należy wykonać szereg kroków:

1. Stworzyć i wyświetlić semiwariogram empiryczny analizowanej zmiennej z użyciem funkcji `variogram()` oraz `plot()`
2. Zdefiniować parametry semiwariogramu, tj. semiwariancja cząstkowa (`psill`), skrócona nazwa używanej funkcji (`model`) oraz jej zasięg (`range`) w funkcji `vgm()`. Uzyskany model można przedstawić w funkcji `plot()` podając nazwę obiektu zawierającego semiwariogram empiryczny oraz obiektu zawierającego model
3. Dopasować parametry modelu używając funkcji `fit.variogram()`. To dopasowanie można również zwizualizować używając funkcji `plot()`

### 7.4.1 Modelowanie izotropowe | Model nuggetowy

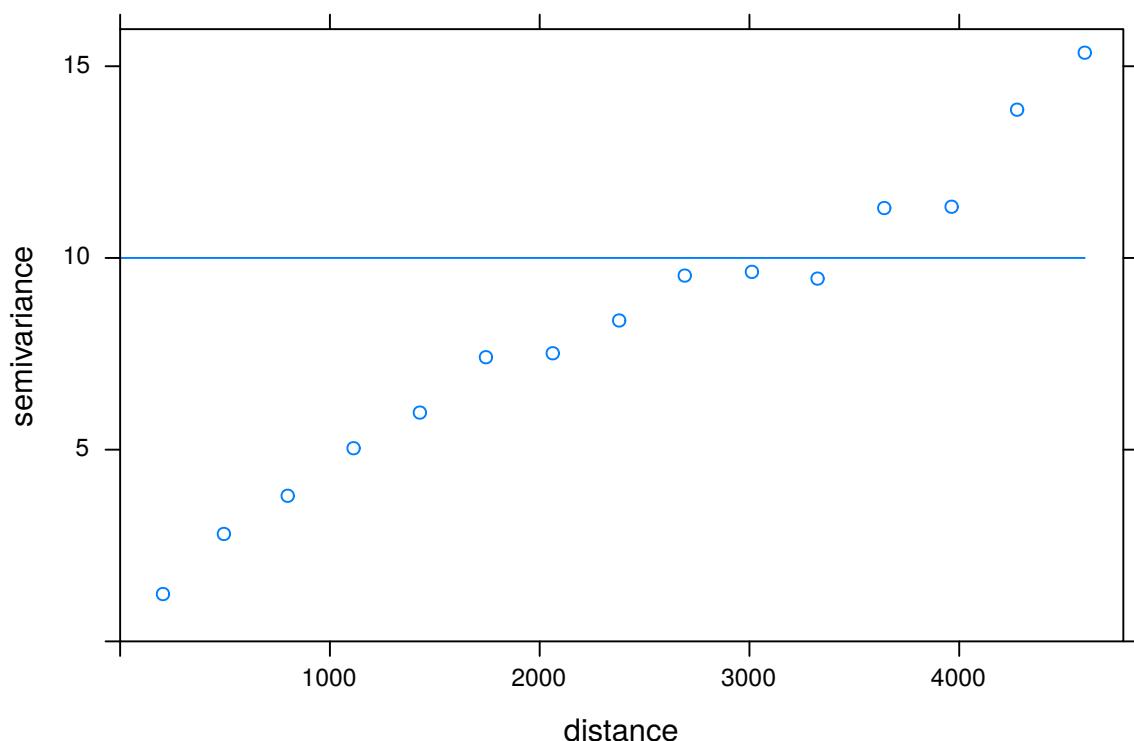
Model nuggetowy (Nug) określa sytuację, w której analizowana zmienna nie wykazuje autokorelacji. Inaczej mówiąc, niepodobieństwo jej wartości nie wzrasta wraz z odległością. Ten typ modelu najczęściej używany jest w modelach złożonych. W takich wypadkach służy on do określania, między innymi, błędu pomiarowego czy zmienności na krótkich odstępach.

```
vario <- variogram(temp~1, punkty)
plot(vario)
```



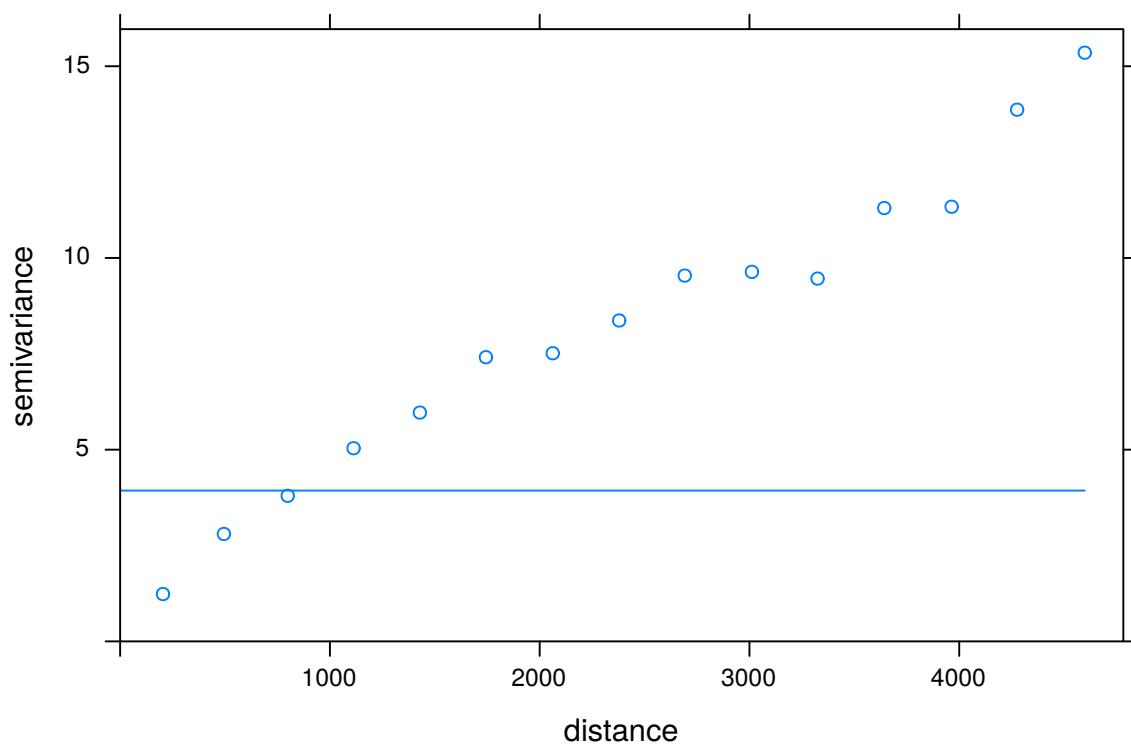
```
model_nug <- vgm(psill=10, model='Nug', range=0)
model_nug
```

```
##   model psill range
## 1   Nug     10     0
plot(vario, model=model_nug)
```



```
fitted_nug <- fit.variogram(vario, model_nug)
fitted_nug
```

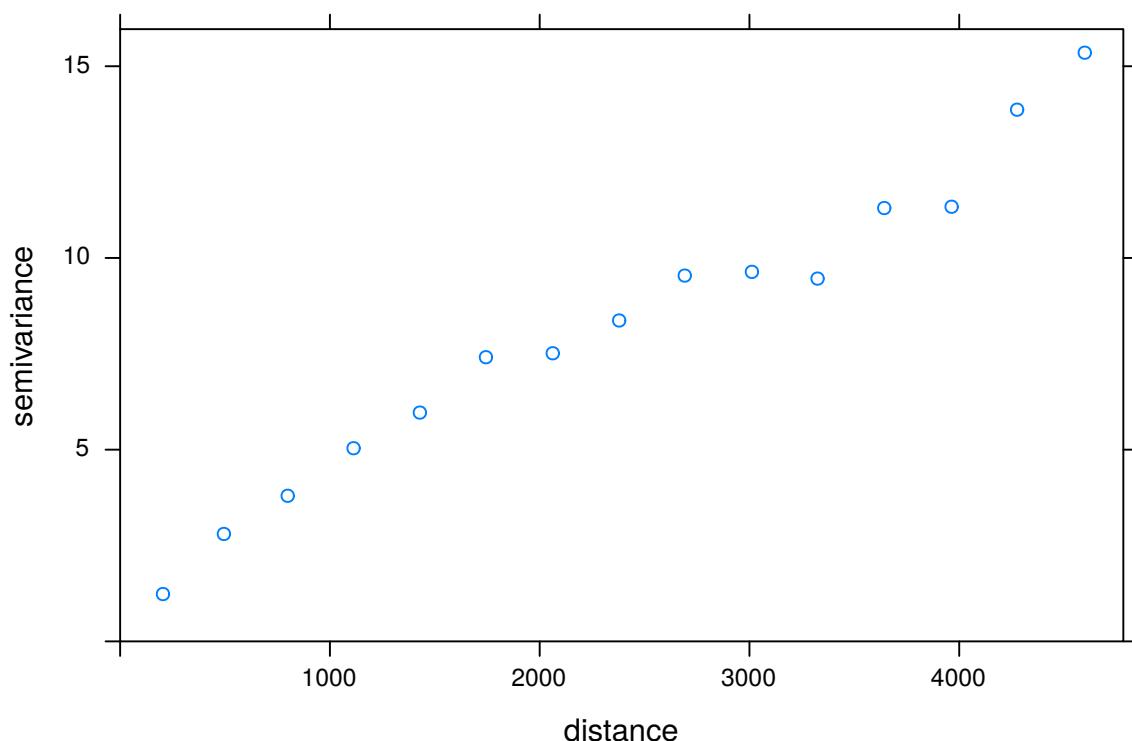
```
##   model    psill range
## 1   Nug 3.933029    0
plot(vario, model=fitted_nug)
```



#### 7.4.2 Modelowanie izotropowe | Model sferyczny

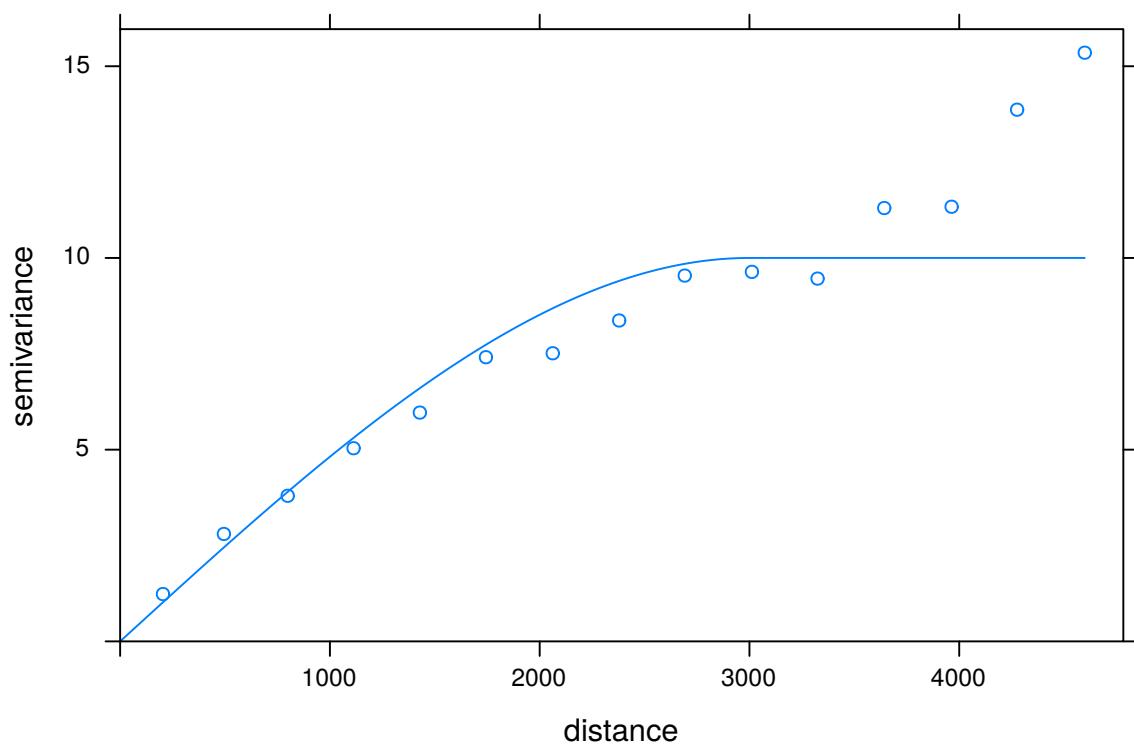
Model sferyczny (Sph) jest jednym z najczęściej stosowanych modeli geostatystycznych. Reprezentuje on cechę, której zmienność wartości ma charakter naprzemiennej płytaw niskich i wysokich wartości. Średnio te płyty mają średnicę określona przez zasięg (`range`) modelu.

```
vario <- variogram(temp~1, punkty)
plot(vario)
```



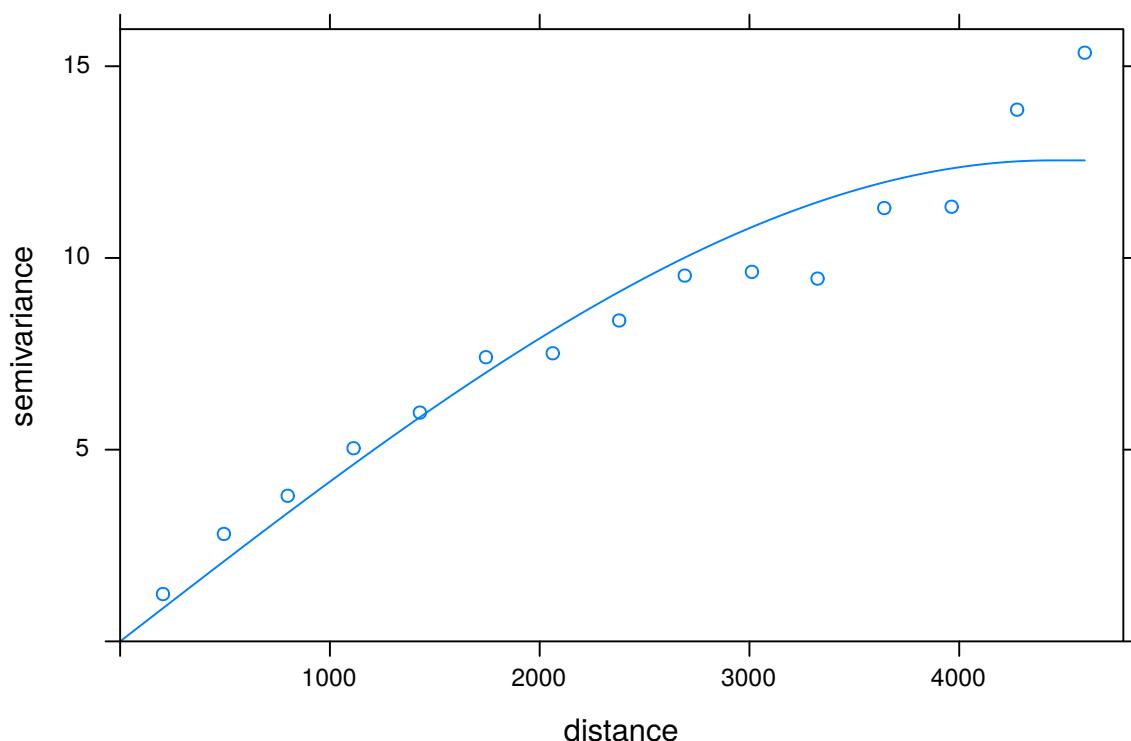
```
model_sph <- vgm(psill=10, model = 'Sph', range=3000)
model_sph
```

```
##   model psill range
## 1   Sph    10  3000
plot(vario, model=model_sph)
```



```
fitted_sph <- fit.variogram(vario, model_sph)
fitted_sph
```

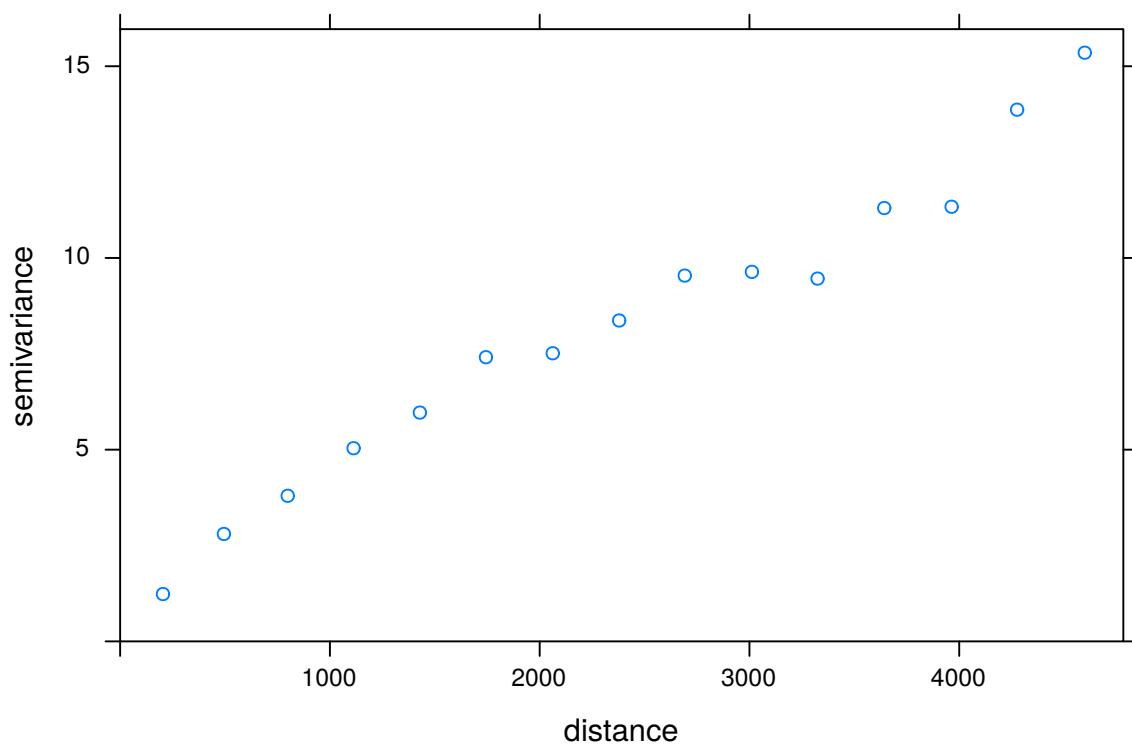
```
##   model   psill    range
## 1   Sph 12.5445 4440.768
plot(vario, model=fitted_sph)
```



#### 7.4.3 Modelowanie izotropowe | Model wykładniczy

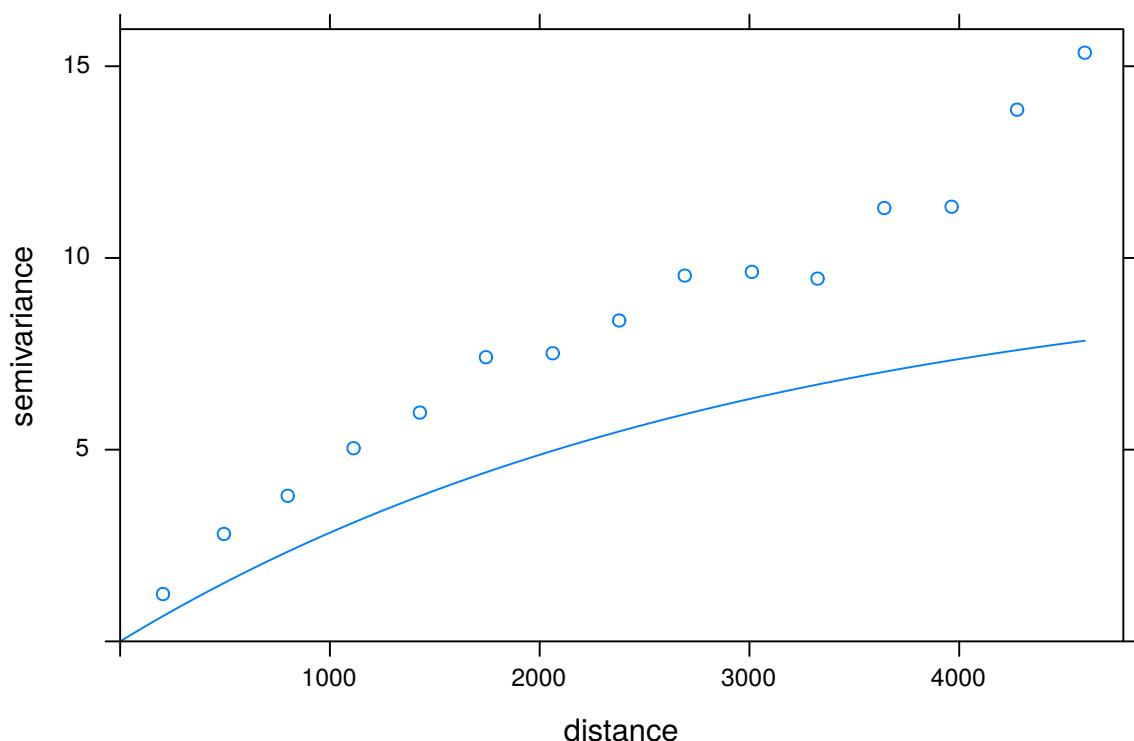
Model wykładniczy (Exp) również jest jednym z najczęściej używanych w geostatystyce. Od modelu sferycznego różni go szczególnie to, że nie ma on skończonego zasięgu. W jego przypadku, zamiast zasięgu podaje się tzw. zasięg praktyczny. Oznacza on odległość na jakiej model osiąga 95% wartości wariancji progowej.

```
vario <- variogram(temp~1, punkty)
plot(vario)
```



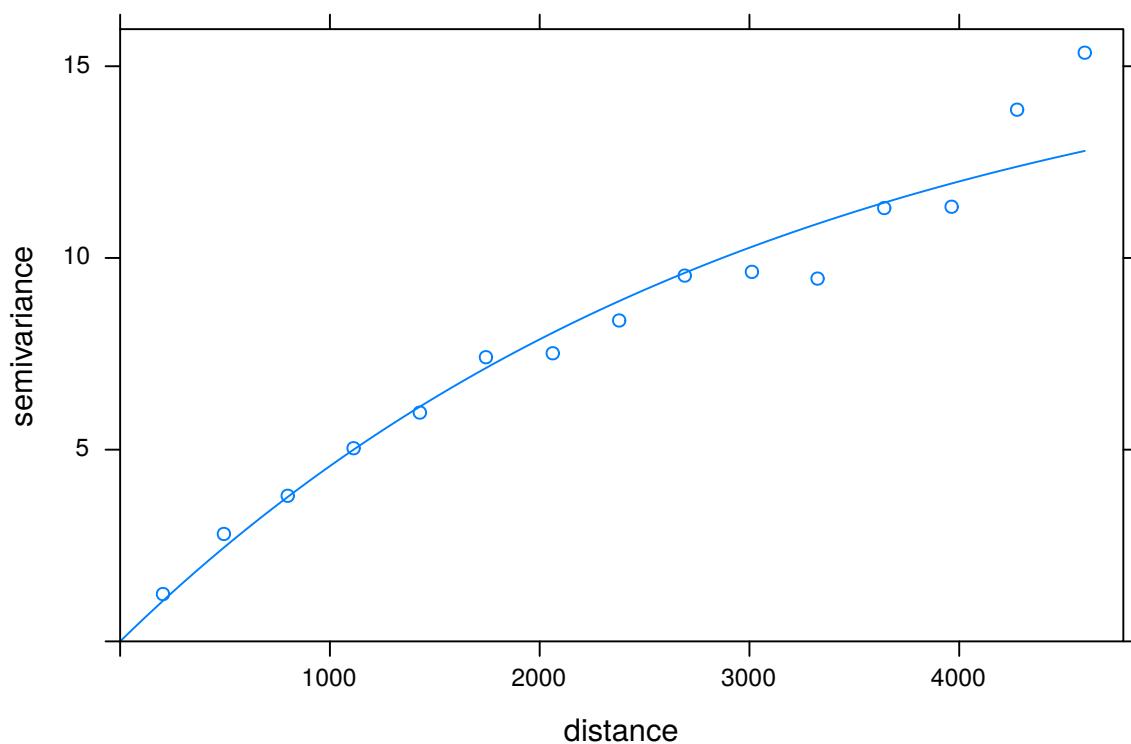
```
model_exp <- vgm(psill=10, model = 'Exp', range=3000)
model_exp
```

```
##   model psill range
## 1   Exp    10   3000
plot(vario, model=model_exp)
```



```
fitted_exp <- fit.variogram(vario, model_exp)
fitted_exp
```

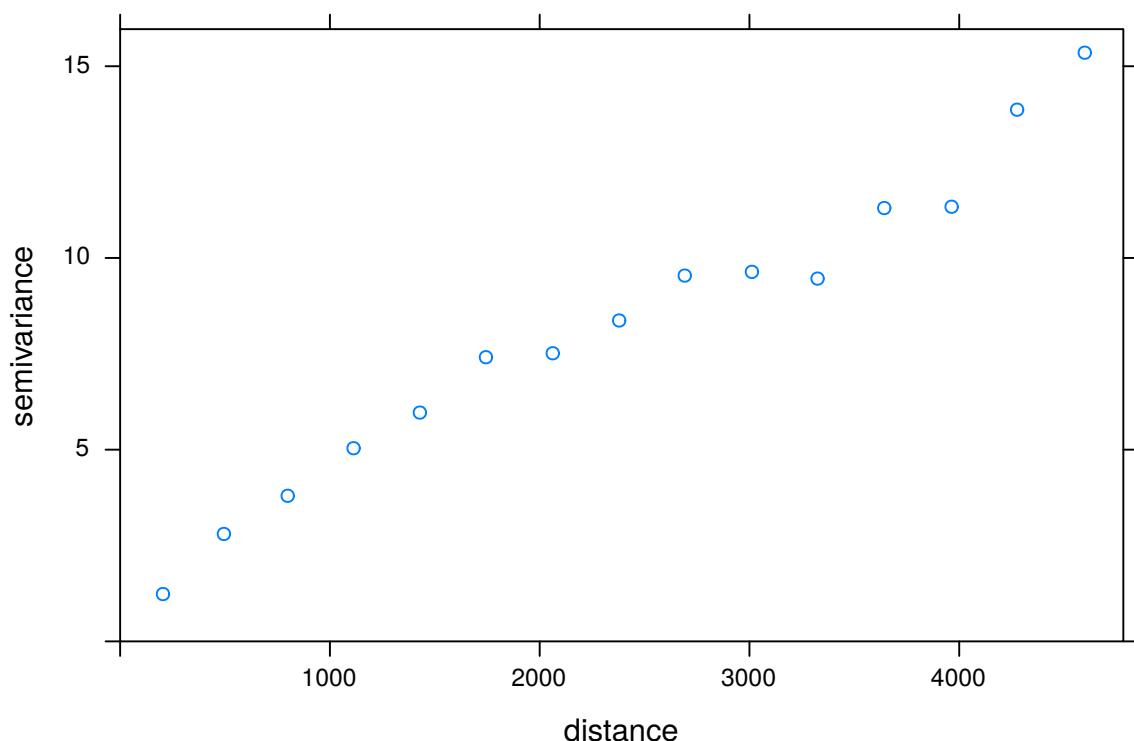
```
##   model    psill    range
## 1   Exp 16.50871 3084.309
plot(vario, model=fitted_exp)
```



#### 7.4.4 Modelowanie izotropowe | Model gaussowski

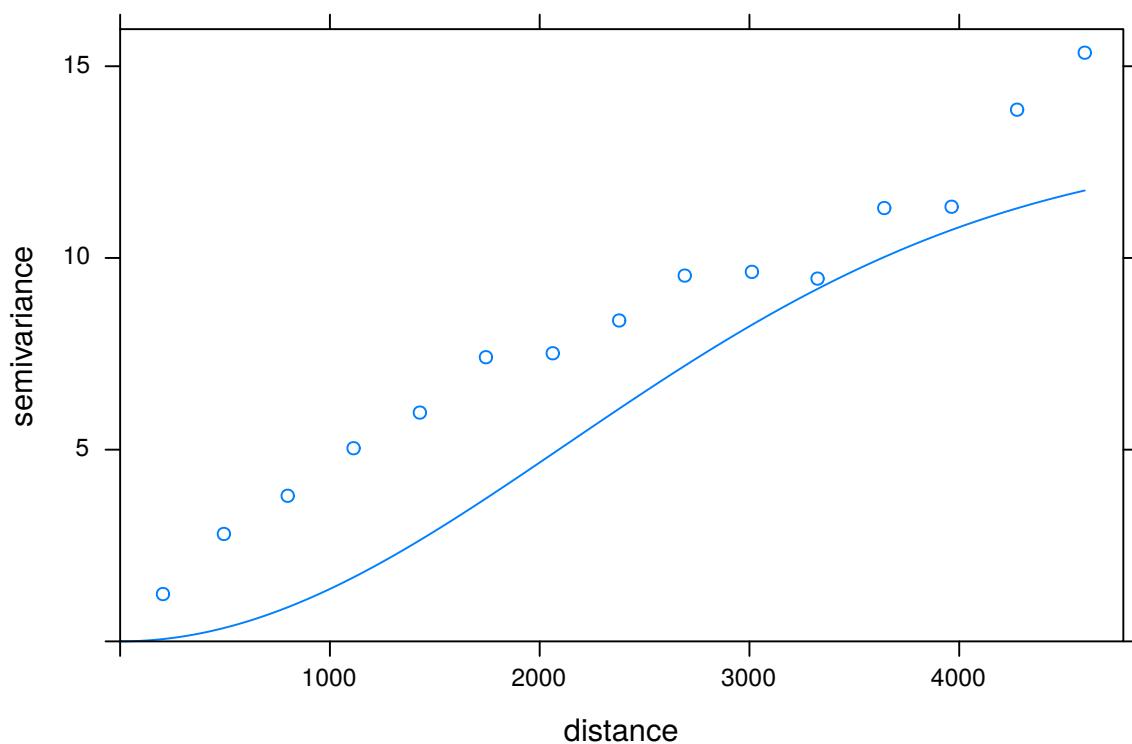
Model gaussowski (Gau) również posiada zasięg praktyczny definiowany jako 95% wartości wariancji progowej. Jego cechą charakterystyczną jest paraboliczny kształt na początkowym odcinku. Jest on najczęściej używany do modelowania cech o regularnej i łagodnej zmienności przestrzennej. Model gaussowski z uwagi na swoje cechy zazwyczaj nie powinien być stosowany samodzielnie, lecz jako element modelu złożonego.

```
vario <- variogram(temp~1, punkty)
plot(vario)
```



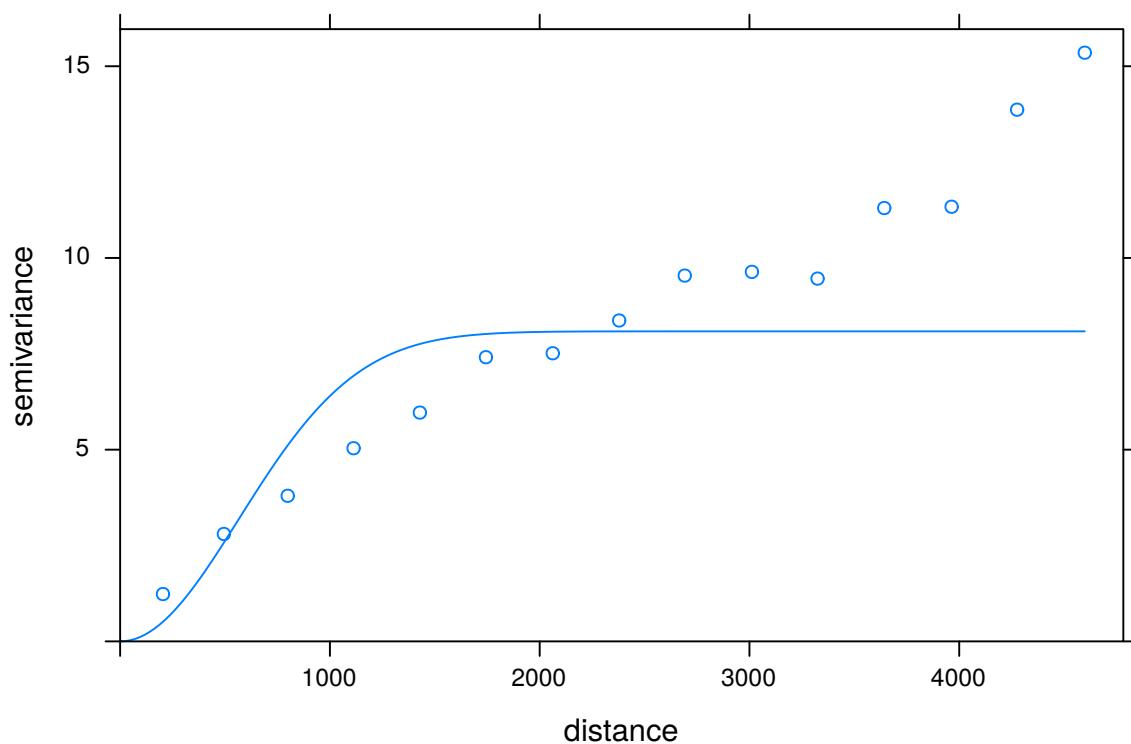
```
model_gau <- vgm(psill=13, model = 'Gau', range=3000)
model_gau
```

```
##   model psill range
## 1   Gau    13   3000
plot(vario, model=model_gau)
```



```
fitted_gau <- fit.variogram(vario, model_gau)
fitted_gau
```

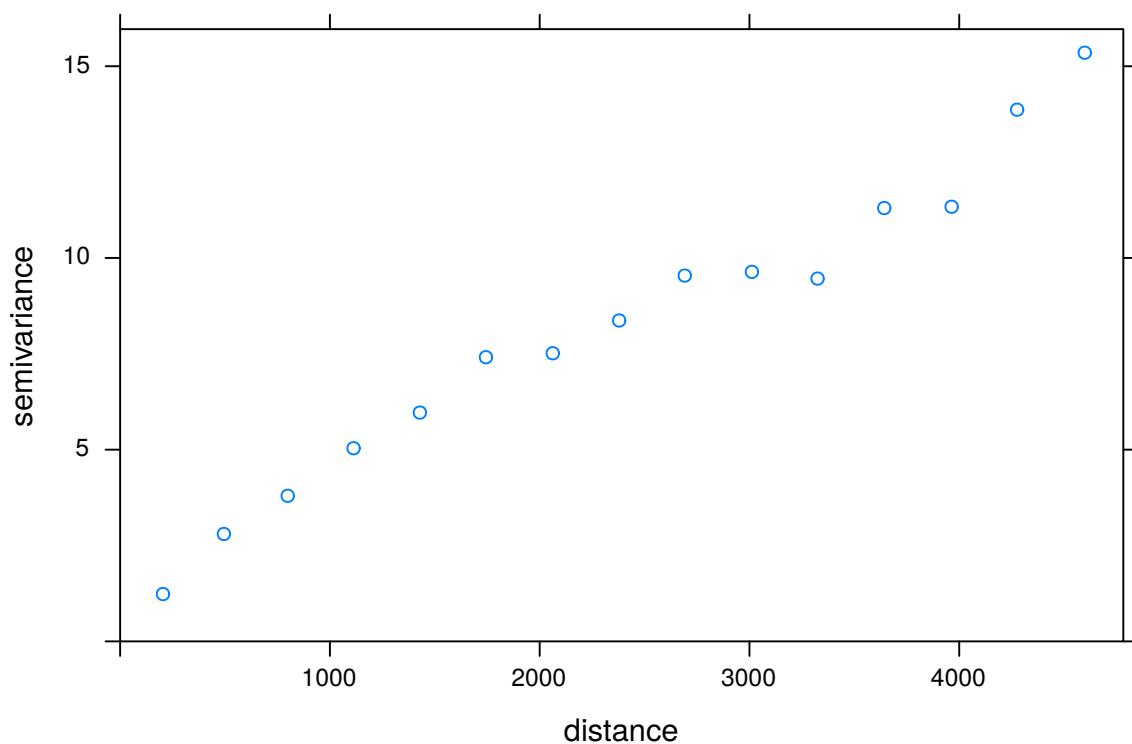
```
##   model    psill    range
## 1   Gau 8.085963 798.7454
plot(vario, model=fitted_gau)
```



#### 7.4.5 Modelowanie izotropowe | Model potęgowy

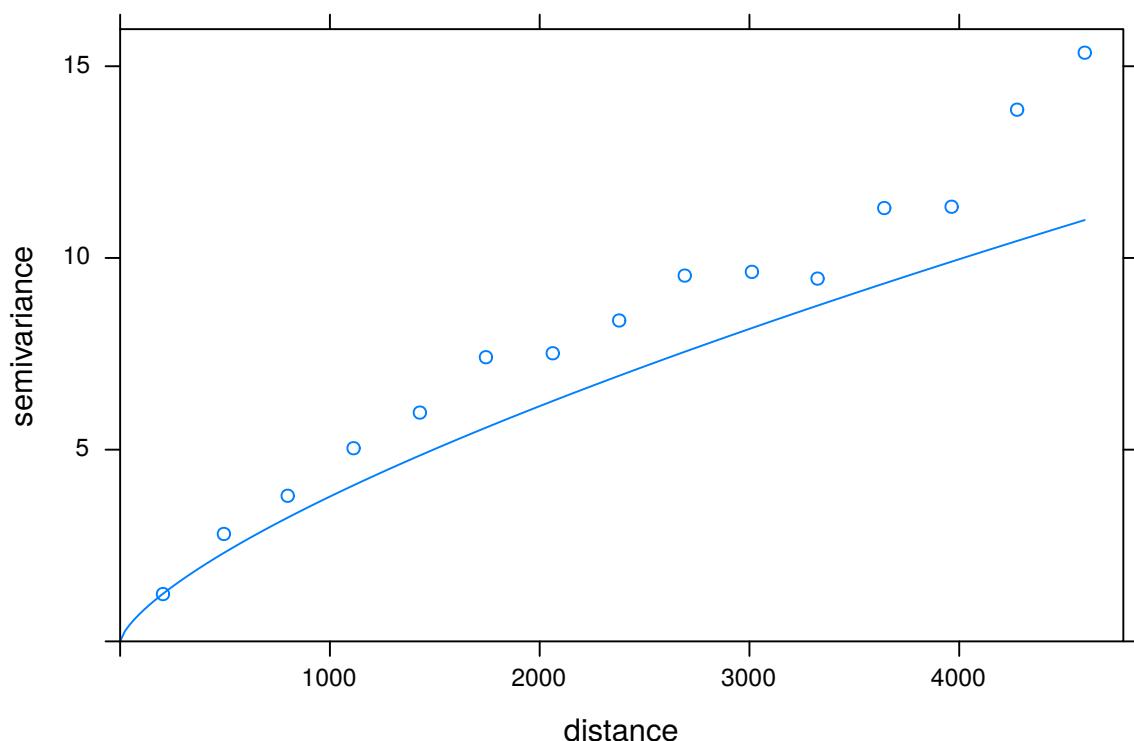
Model potęgowy (Pow) to przykład tzw. modelu nieograniczonego. Jego wartość rośnie w nieskończoność, dlatego niemożliwe jest określenie jego zasięgu. W przypadku modelu potęgowego, parametr `range` oznacza wykładnik potęgowy.

```
vario <- variogram(temp~1, punkty)
plot(vario)
```



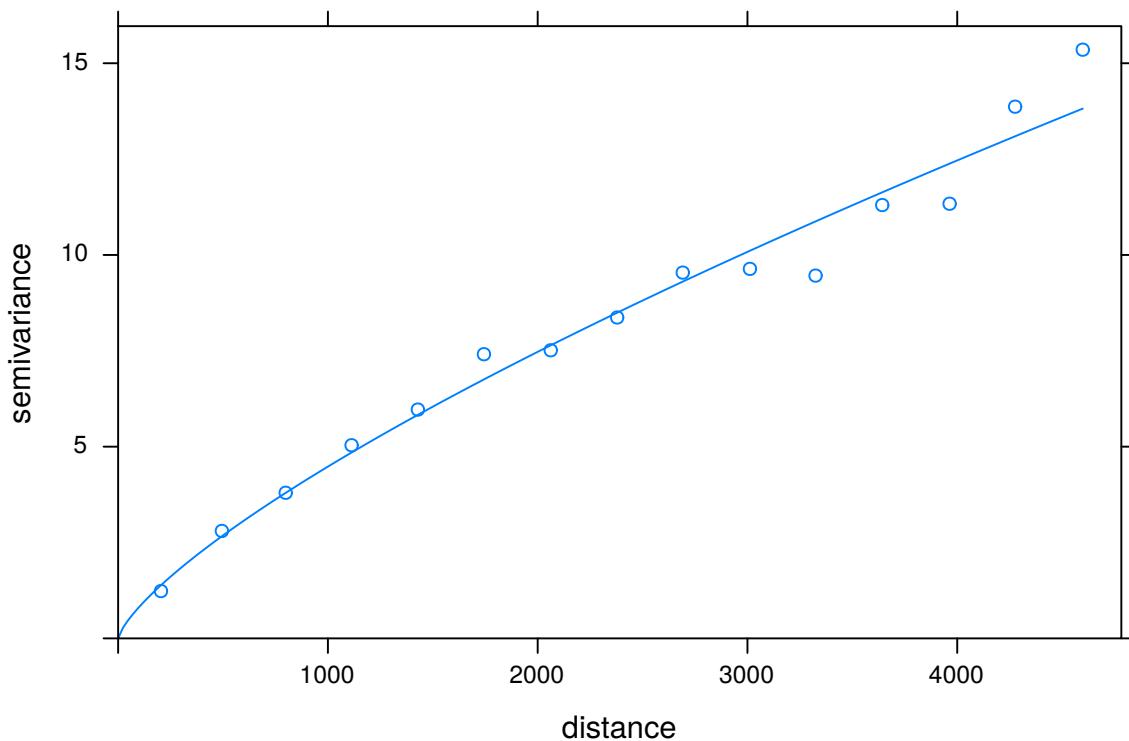
```
model_pow <- vgm(psill=0.03, model = 'Pow', range=0.7)
model_pow
```

```
##   model psill range
## 1 Pow  0.03    0.7
plot(vario, model=model_pow)
```



```
fitted_pow <- fit.variogram(vario, model_pow)
fitted_pow
```

```
##   model      psill      range
## 1 Pow 0.02732273 0.7382382
plot(vario, model=fitted_pow)
```

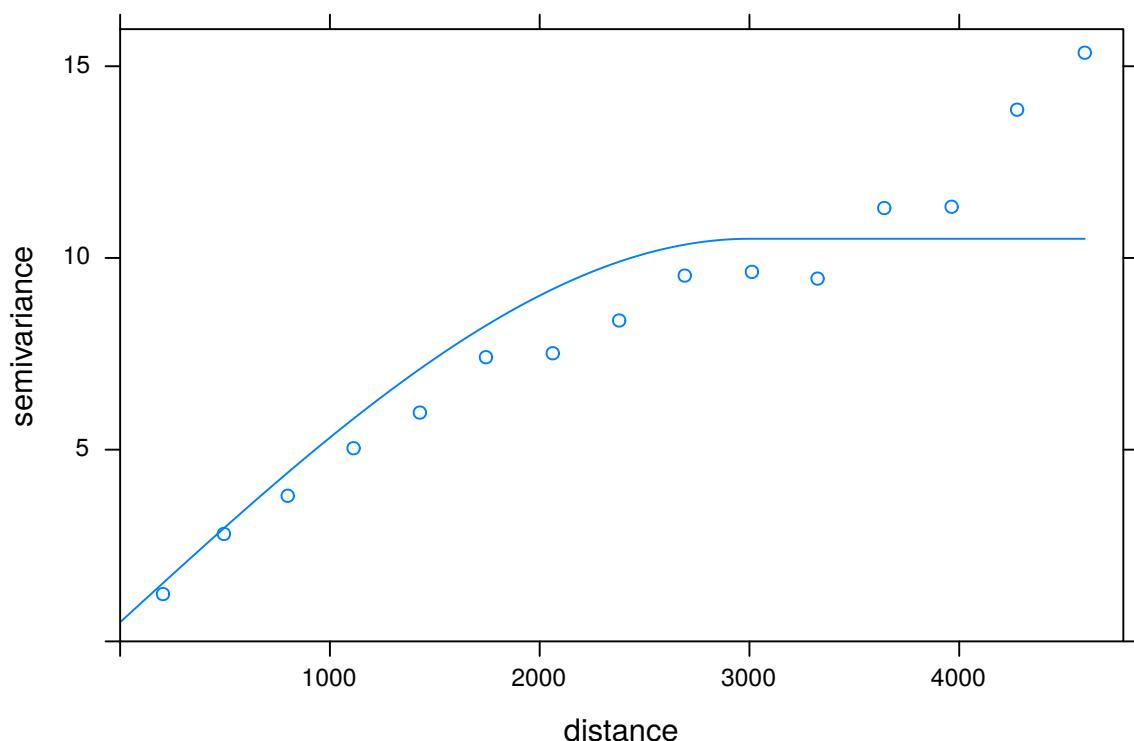


#### 7.4.6 Modelowanie izotropowe | Modele złożone I

Najczęściej pojedynczy model nie jest w stanie odwzorować dokładnie zmienności przestrzennej analizowanej cechy. W takich sytuacjach konieczne jest połączenie dwóch lub więcej modeli podstawowych. Najbardziej powszechny model złożony składa się z funkcji nuggetowej (dla odległości zero) oraz drugiej funkcji (dla dalszej odległości). Zdefiniowanie takiej funkcji odbywa się poprzez dodanie argumentu `nugget` w funkcji `vgm()`.

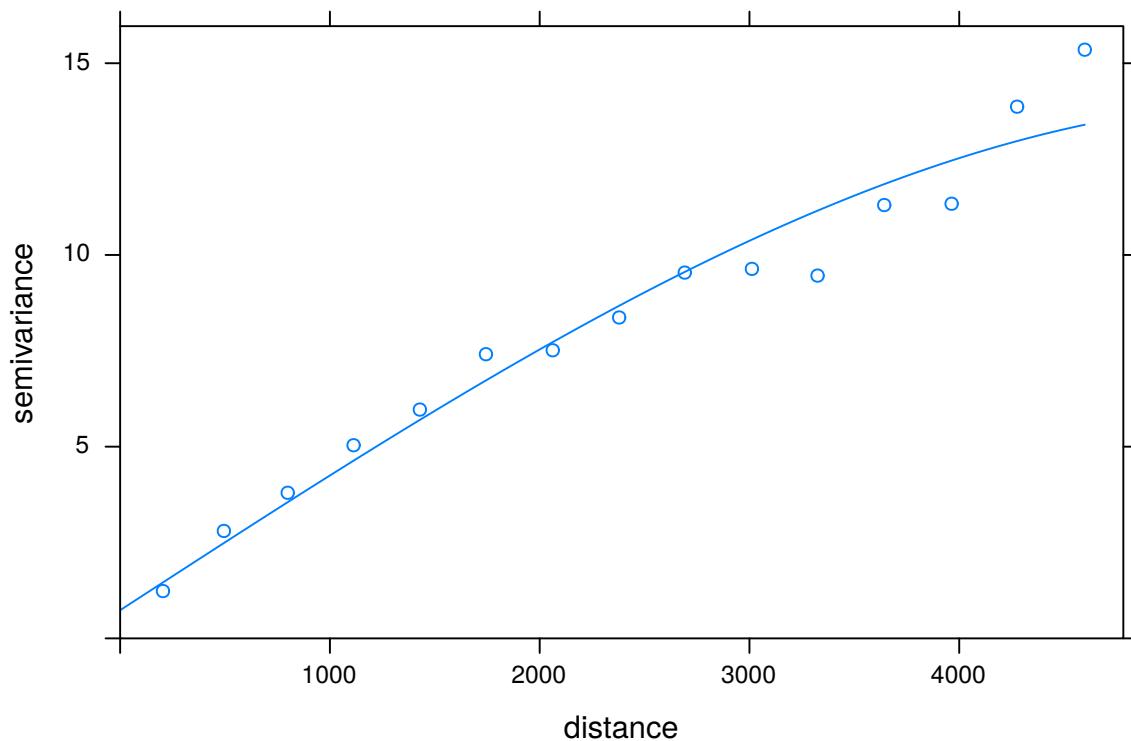
```
vario <- variogram(temp~1, punkty)
model_zl1 <- vgm(psill=10, model = 'Sph', range = 3000, nugget = 0.5)
model_zl1
```

```
##   model psill range
## 1   Nug    0.5     0
## 2   Sph   10.0   3000
plot(vario, model=model_zl1)
```



```
fitted_zl1 <- fit.variogram(vario, model_zl1)
fitted_zl1
```

```
##   model      psill     range
## 1   Nug  0.7346354  0.000
## 2   Sph 13.2621876 5602.027
plot(vario, model=fitted_zl1)
```



#### 7.4.7 Modelowanie izotropowe | Modele złożone II

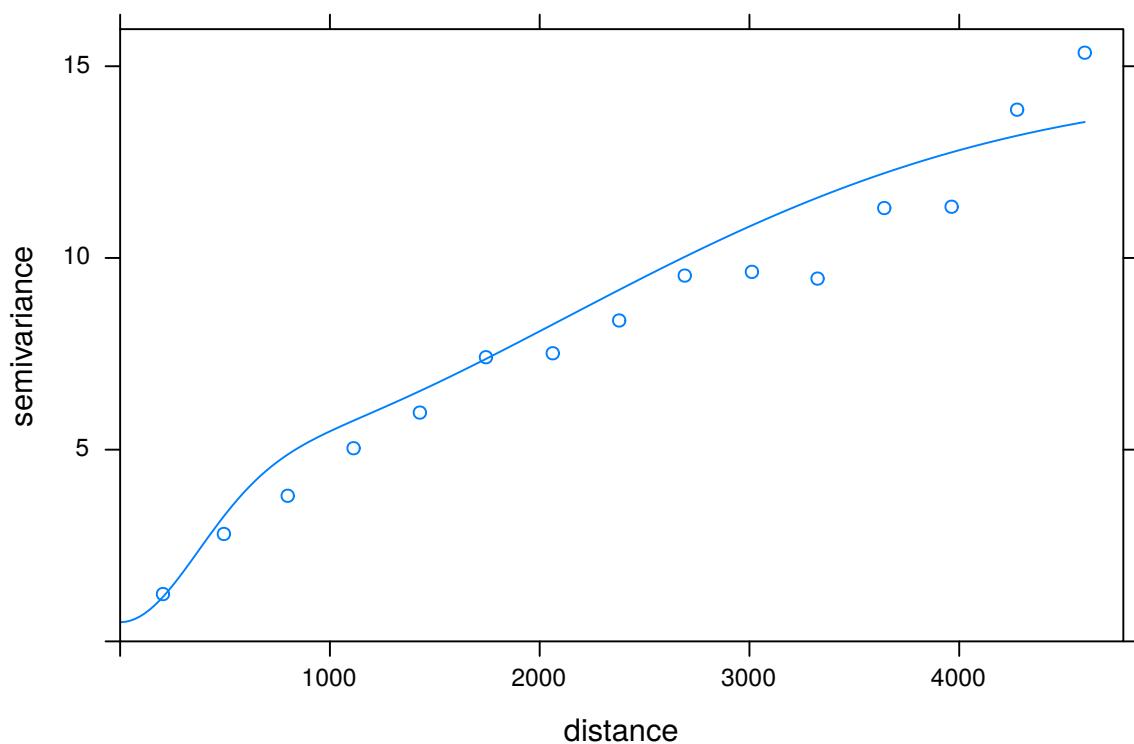
Bardziej złożone modele można tworzyć z pomocą argumentu `add.to`. Przyjmuje on kolejny obiekt funkcji `vgm()` i poprzez połączenie tych dwóch obiektów otrzymuje model złożony. Na poniższym przykładzie stworzony został model złożony składający się z modelu nuggetowego oraz dwóch modeli gaussowskich.

```

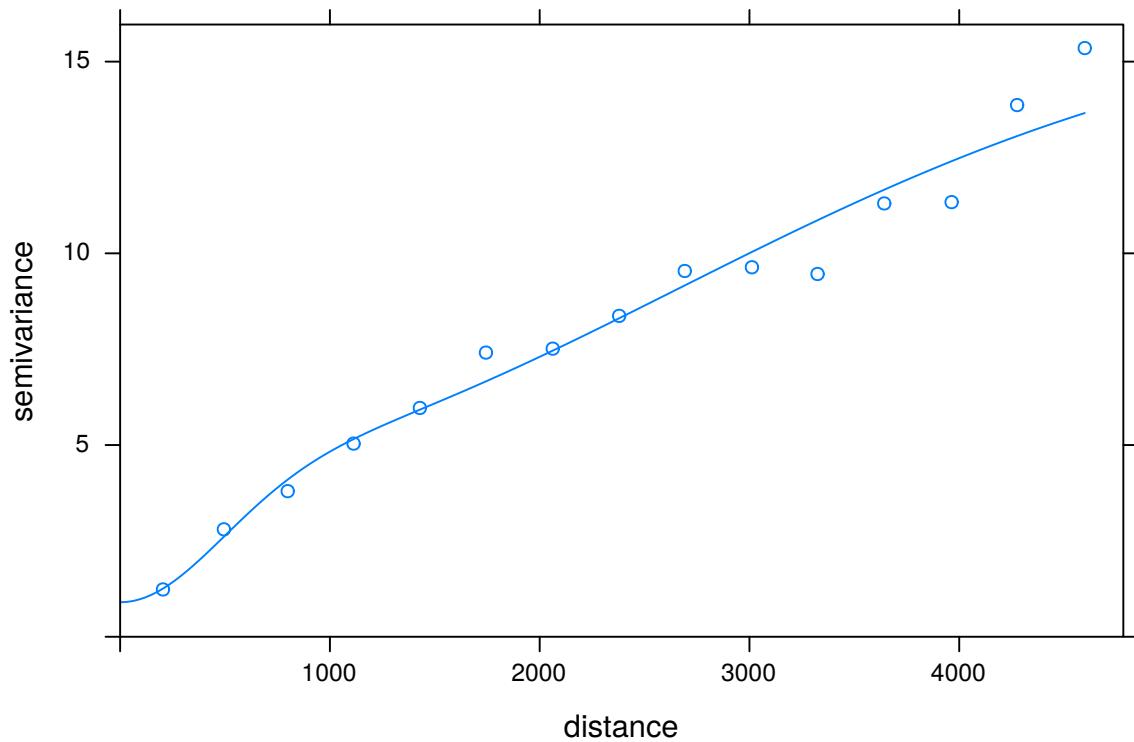
vario <- variogram(temp~1, punkty)
model_zl2 <- vgm(10, 'Gau', 3000, add.to = vgm(4, model = 'Gau', range = 500, nugget = 0.5))
model_zl2

##   model psill range
## 1   Nug    0.5     0
## 2   Gau    4.0   500
## 3   Gau   10.0  3000
plot(vario, model=model_zl2)

```



```
fitted_z12 <- fit.variogram(vario, model_z12)
plot(vario, model=fitted_z12)
```



## 7.5 Modelowanie anizotropowe

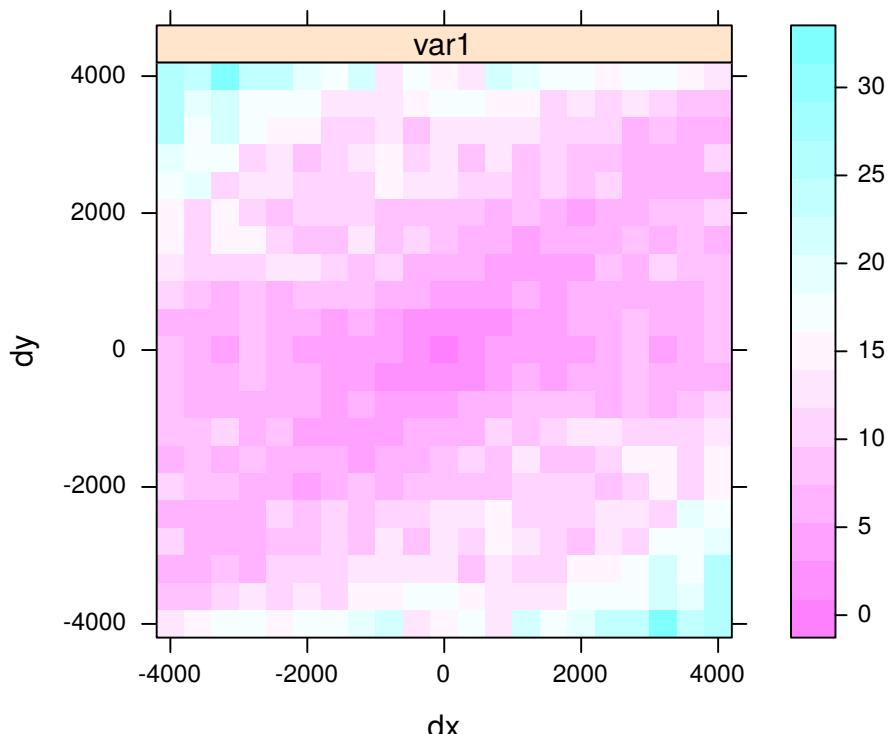
### 7.5.1 Anizotropia

Uwzględnienie anizotropii wymaga zamiany parametru zasięgu na trzy inne parametry:

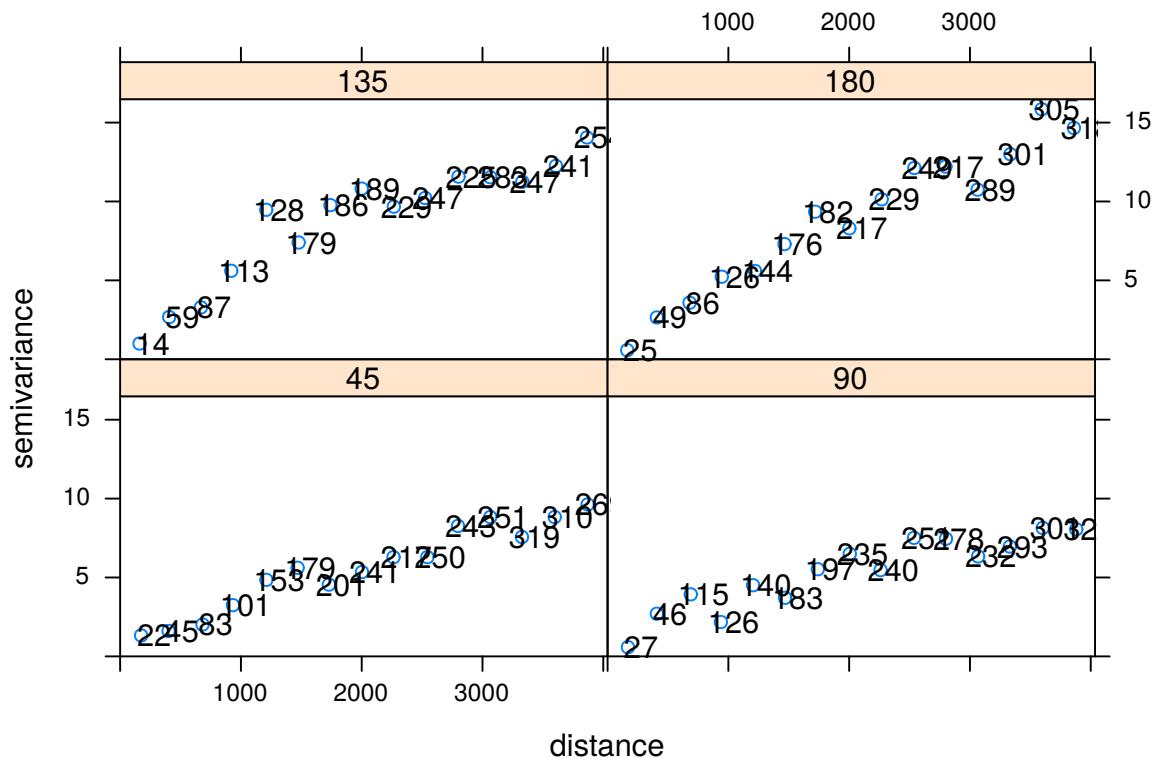
- Zasięg w dominującym kierunku
- Kąt określający dominujący kierunek
- Proporcję anizotropii, czyli relację pomiędzy zasięgiem w dominującym kierunku a zasięgiem w przeciwnieległym kierunku

W pakiecie `gstat` odbywa się to poprzez dodanie argumentu `alpha` do funkcji `variogram()`. Należy w niej zdefiniować analizowane kierunki, które zostały określone na podstawie mapy semiwariogramu. Następnie w funkcji `vgm()` należy podać nowy argument `anis`. Przyjmuje on dwie wartości. Pierwsza z nich (45 w przykładzie poniżej) oznacza dominujący kierunek anizotropii, druga zaś (0.4) mówi o tzw. proporcji anizotropii. Proporcja anizotropii jest to relacja pomiędzy zmiennością na głównym kierunku a kierunku prostopadłym. Na poniższym przykładzie zasięg ustalony dla głównego kierunku wynosi 4000 metrów. Wartość proporcji anizotropii, 0.4, w tym wypadku oznacza że dla prostopadłego kierunku zasięg będzie wynosił 1600 metrów (4000 metrów x 0.4).

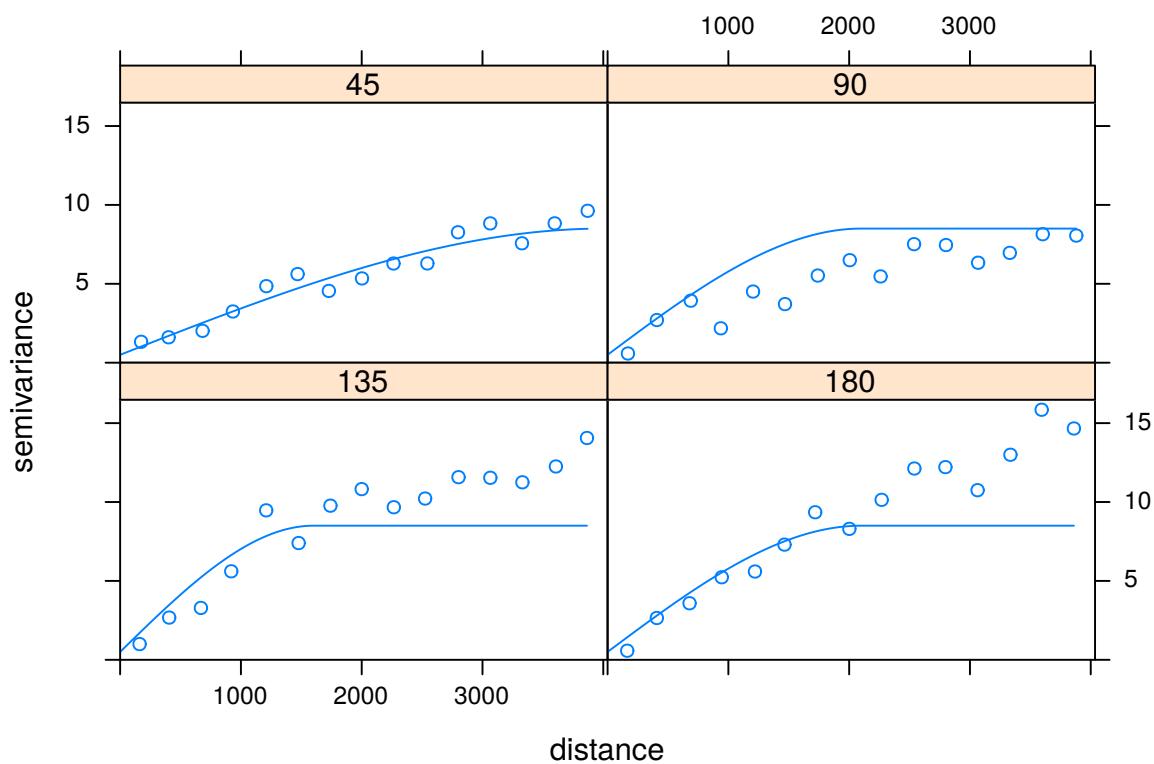
```
vario_map <- variogram(temp~1, punkty, cutoff=4000, width=400, map=TRUE)
plot(vario_map)
```



```
vario_kier <- variogram(temp~1, punkty, alpha = c(45, 90, 135, 180), cutoff=4000)
plot(vario_kier, plot.numbers=TRUE)
```



```
vario_kier_fit <- vgm(psill=8, model='Sph', range=4000, nugget=0.5, anis = c(45, 0.4))
plot(vario_kier, vario_kier_fit, as.table=TRUE)
```



# Rozdział 8

## Estymacje jednozmienne

```
library('gstat')
library('sp')
library('geostatbook')
data(punkty)
data(siatka)
```

### 8.1 Kriging

#### 8.1.1 Kriging | Interpolacja geostatystyczna

Kriging (interpolacja geostatystyczna) to grupa metod estymacji zaproponowana w latach 50. przez Daniela Krige. Główna zasada mówi, że prognoza w danej lokalizacji jest kombinacją obokległych obserwacji. Waga nadawana każdej z obserwacji jest zależna od stopnia (przestrzennej) korelacji - stąd też bierze się istotna rola semiwariogramów.

#### 8.1.2 Metod krigingu

Istnieje szereg metod krigingu, w tym:

- Kriging prosty (ang. *Simple kriging*)
- Kriging zwykły (ang. *Ordinary kriging*)
- Kriging z trendem (ang. *Kriging with a trend*)
- Kriging danych kodowanych (ang. *Indicator kriging*)
- Kriging stratyfikowany (ang. *Kriging within strata* – KWS)
- Kriging prosty ze zmiennymi średnimi lokalnymi (ang. *Simple kriging with varying local means* – SKlm)
- Kriging z zewnętrznym trendem/Uniwersalny kriging (ang. *Kriging with an external trend/Universal kriging*)
- Kokriging (ang. *Co-kriging*)
- Inne

### 8.2 Kriging prosty

#### 8.2.1 Kriging prosty (ang. *Simple kriging*)

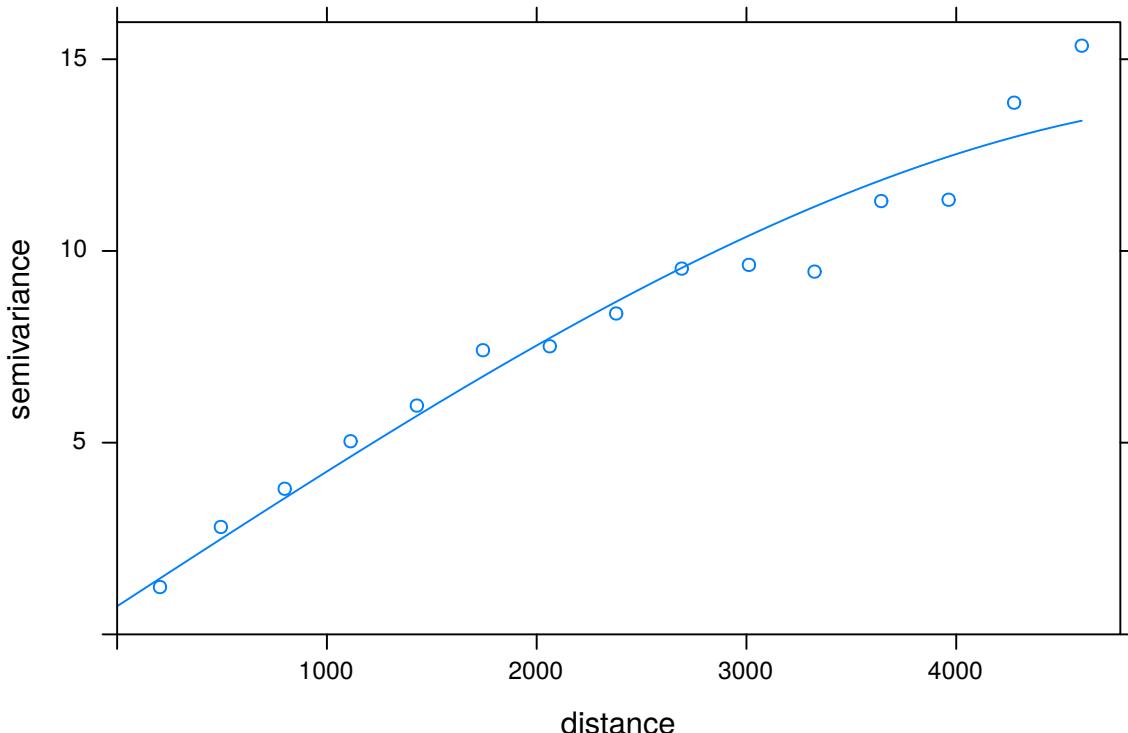
Kriging prosty zakłada, że średnia jest znana i stała na całym obszarze. W poniższym przykładzie po stworzeniu semiwariogramu empryczneg, dopasowano model semiwariogramu składający się z funkcji sferycznej o zasięgu 4000 metrów i wartości nuggetu równej 0,5.

```

vario <- variogram(temp~1, punkty)
model <- vgm(10, model = 'Sph', range = 4000, nugget = 0.5)
model

##   model psill range
## 1   Nug  0.5    0
## 2   Sph 10.0 4000
fitted <- fit.variogram(vario, model)
plot(vario, model=fitted)

```



Następnie nastąpiła estymacja wartości z użyciem metody krigingu prostego. W funkcji `krige()` z pakietu `gstat`, użycie tej metody wymaga ustalenia średniej wartości cechy za pomocą argumentu `beta`.

```
sk <- krige(temp~1, punkty, siatka, model=fitted, beta=15.32)
```

```
## [using simple kriging]
```

Wynik krigingu prostego, jak i każdy inny uzyskany z użyciem pakietu `gstat`, można podejrzeć używając funkcji `summary()`. Szczególnie ważne są dwie, nowe zmienne - `var1.pred` oraz `var1.var`. Pierwsza z nich oznacza wartość estymowaną dla każdego oczka siatki, druga zaś mówi o wariancji estymacji.

```
summary(sk)
```

```

## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min     max
## x 745586.7 756926.7
## y 712661.2 721211.2
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 10993

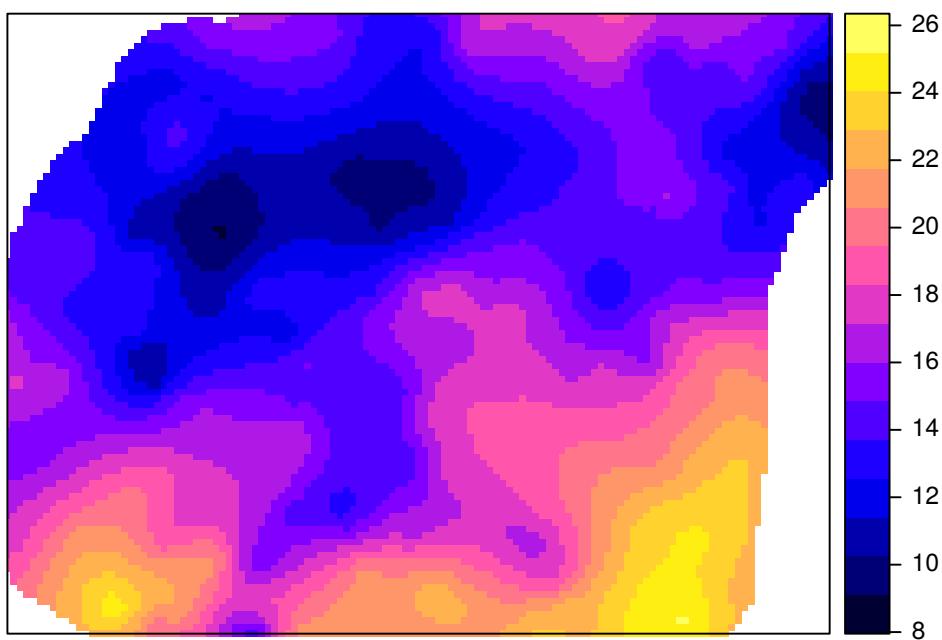
```

```
## Grid attributes:  
##   cellcentre.offset cellsize cells.dim  
##   s1           745586.7    90      127  
##   s2           712661.2    90      96  
## Data attributes:  
##   var1.pred      var1.var  
##   Min.    : 9.083  Min.    :1.062  
##   1st Qu.:13.208  1st Qu.:1.875  
##   Median  :15.224  Median  :2.183  
##   Mean    :15.862  Mean    :2.253  
##   3rd Qu.:18.081  3rd Qu.:2.574  
##   Max.    :25.218  Max.    :4.615
```

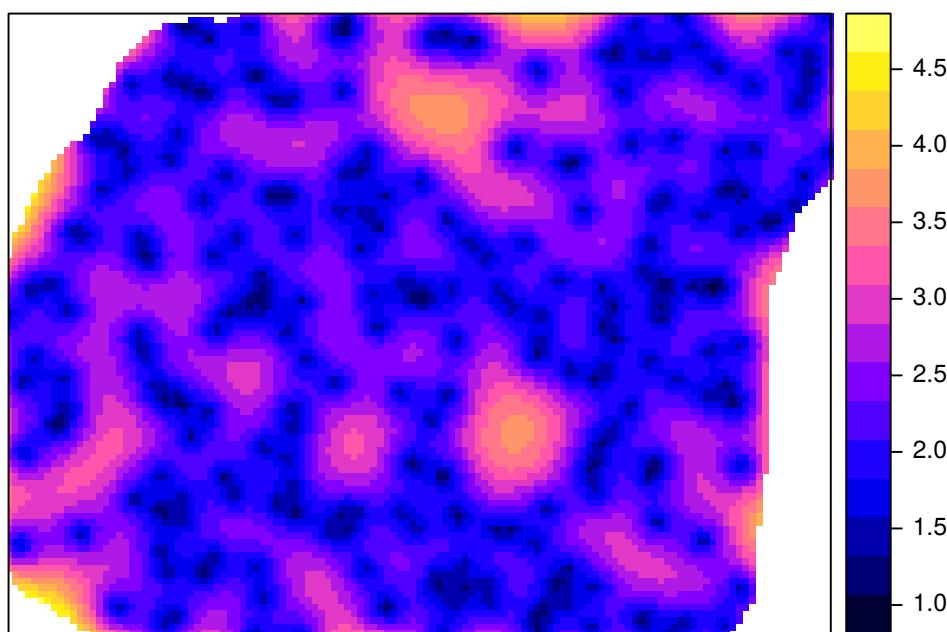
Obie uzyskane zmienne można wyświetlić z użyciem funkcji `spplot()`.

```
spplot(sk, 'var1.pred')  
spplot(sk, 'var1.var')
```

### Predukcja SK



### Wariancja predukcji SK



## 8.3 Kriging zwykły

### 8.3.1 Kriging zwykły (ang. *Ordinary kriging*)

W krigingu zwykłym średnia traktowana jest jako wartość nieznana. Metoda ta uwzględnia lokalne fluktuacje średniej poprzez stosowanie ruchomego okna. Parametry ruchomego okna można określić za pomocą jednego z dwóch argumentów:

- `nmax` - użytu zostanie określona liczba najbliższych obserwacji
- `maxdist` - użytu zostaną jedynie obserwacje w zadanej odległości

```
# ok <- kriging(temp~1, punkty, siatka, model=fitted, nmax=30)
ok <- kriging(temp~1, punkty, siatka, model=fitted, maxdist=1500)
```

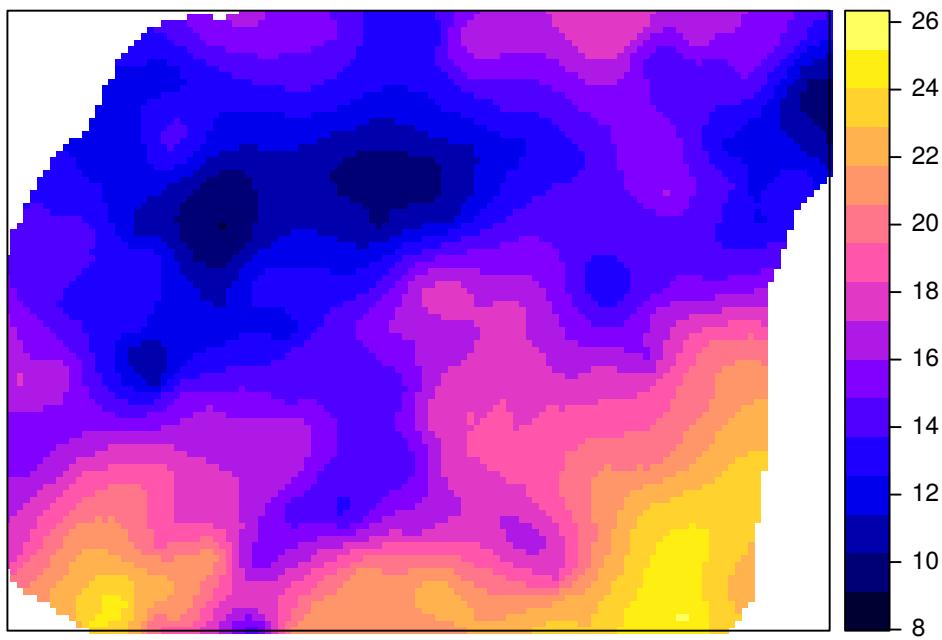
```
## [using ordinary kriging]
```

Podobnie jak w przypadku krigingu prostego, można przyjrzeć się wynikom estymacji używając funkcji `summary()` oraz wyświetlić je używając funkcji `spplot()`.

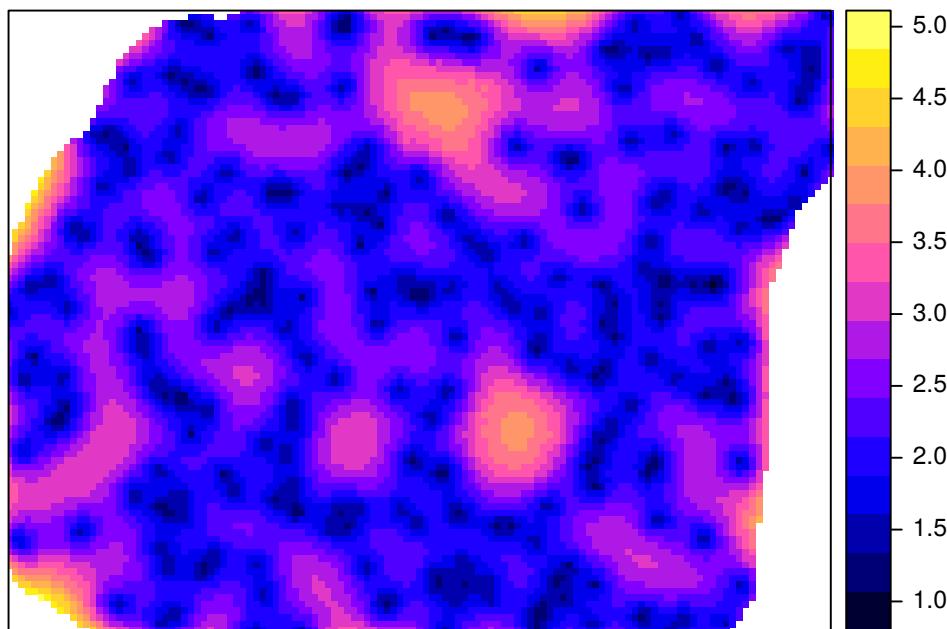
```
summary(ok)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min     max
## x 745586.7 756926.7
## y 712661.2 721211.2
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 10993
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## s1          745586.7      90      127
## s2          712661.2      90      96
## Data attributes:
##   var1.pred      var1.var
##   Min. : 9.096  Min. :1.062
##   1st Qu.:13.248  1st Qu.:1.880
##   Median :15.225  Median :2.194
##   Mean   :15.880  Mean   :2.278
##   3rd Qu.:18.028  3rd Qu.:2.602
##   Max.   :25.210  Max.   :4.847
spplot(ok, 'var1.pred')
spplot(ok, 'var1.var')
```

### Preidykja OK



### Wariancja preidykji OK

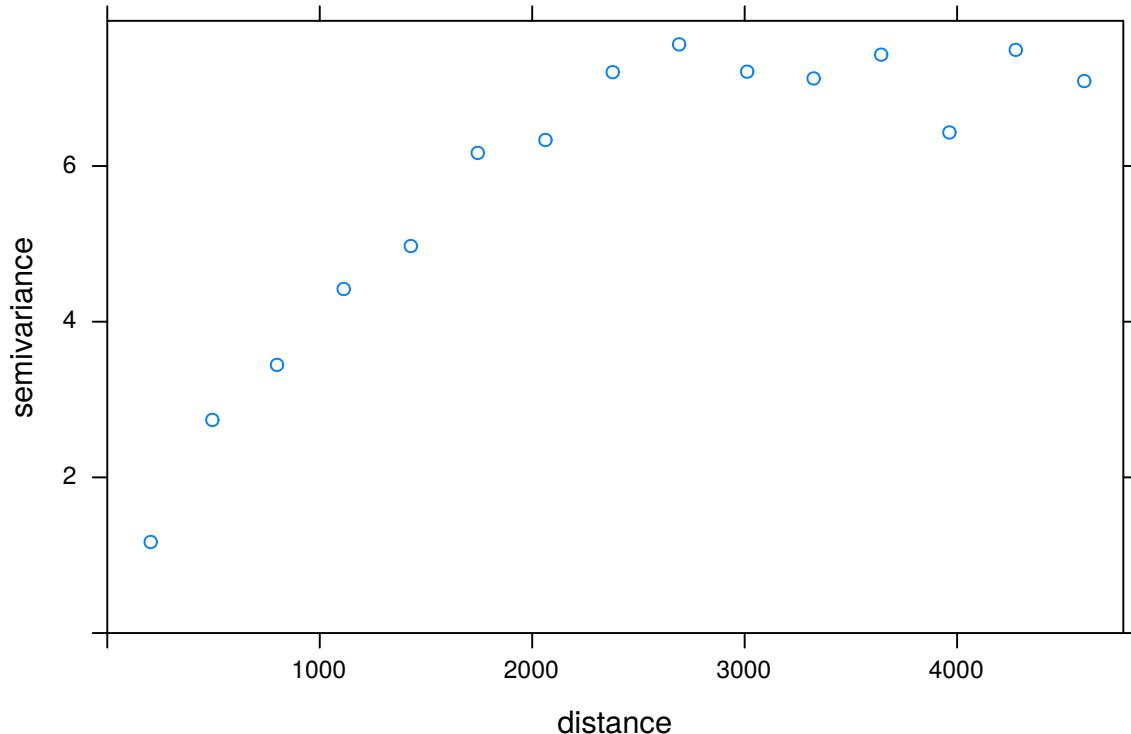


## 8.4 Kriging z trendem

### 8.4.1 Kriging z trendem (ang. *Kriging with a trend*)

Kriging z trendem, określany również jako kriging z wewnętrznym trendem, do estymacji wykorzystuje (oprócz zmienności wartości wraz z odległością) położenie analizowanych punktów. Na poniższym przykładzie w funkcji `variogram()` pierwszy z argumentów przyjął postać `temp~x+y`, co oznacza, że uwzględniamy liniowy trend zależny od współrzędnej  $x$  oraz  $y$ .

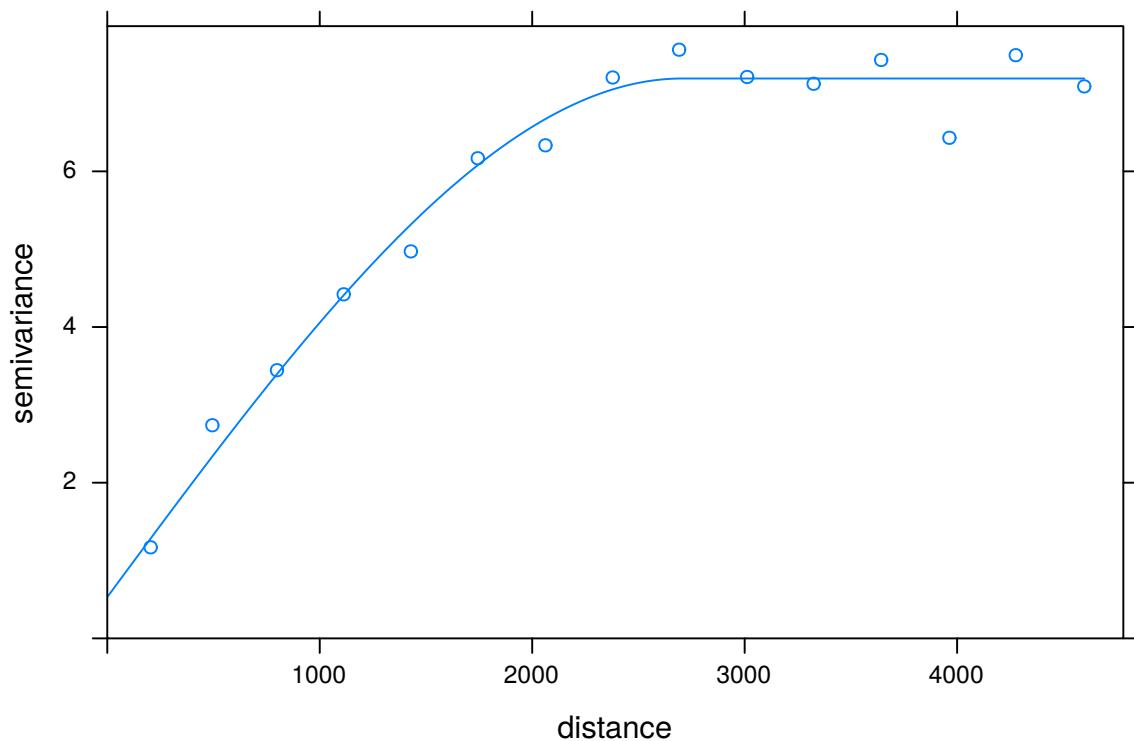
```
vario_kzt <- variogram(temp~x+y, data=punkty)
plot(vario_kzt)
```



Dalszym etapem jest dopasowanie modelu semiwariancji, a następnie wyliczenie estymowanych wartości z użyciem funkcji `krige()`. Należy tutaj pamiętać, aby wzór (w przykładzie `temp~x+y`) był taki sam podczas budowania semiwariogramu, jak i interpolacji.

```
model_kzt <- vgm(psill = 5, model = 'Sph', range = 2500, nugget = 1)
fitted_kzt <- fit.variogram(vario_kzt, model_kzt)
fitted_kzt
```

```
##   model      psill      range
## 1   Nug 0.5308819    0.000
## 2   Sph 6.6620981 2705.967
plot(vario_kzt, fitted_kzt)
```



```

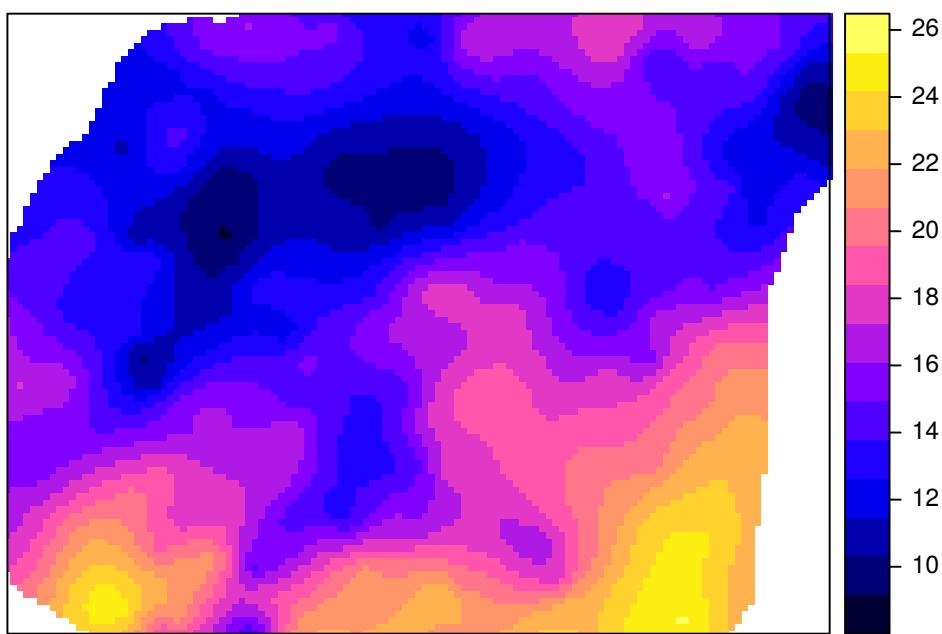
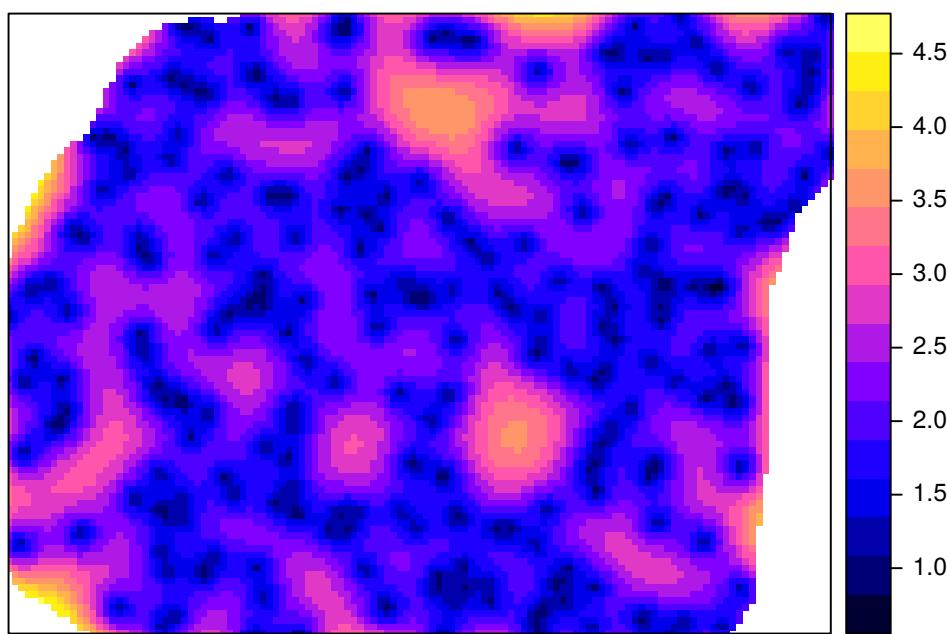
kzt <- krige(temp~x+y, punkty, siatka, model=fitted_kzt)

## [using universal kriging]
summary(kzt)

## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min     max
## x 745586.7 756926.7
## y 712661.2 721211.2
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 10993
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## s1          745586.7      90      127
## s2          712661.2      90      96
## Data attributes:
##   var1.pred      var1.var
##   Min. : 9.14  Min. :0.8141
##   1st Qu.:13.20  1st Qu.:1.6472
##   Median :15.20  Median :1.9750
##   Mean   :15.86  Mean   :2.0481
##   3rd Qu.:18.10  3rd Qu.:2.3847
##   Max.   :25.35  Max.   :4.5134

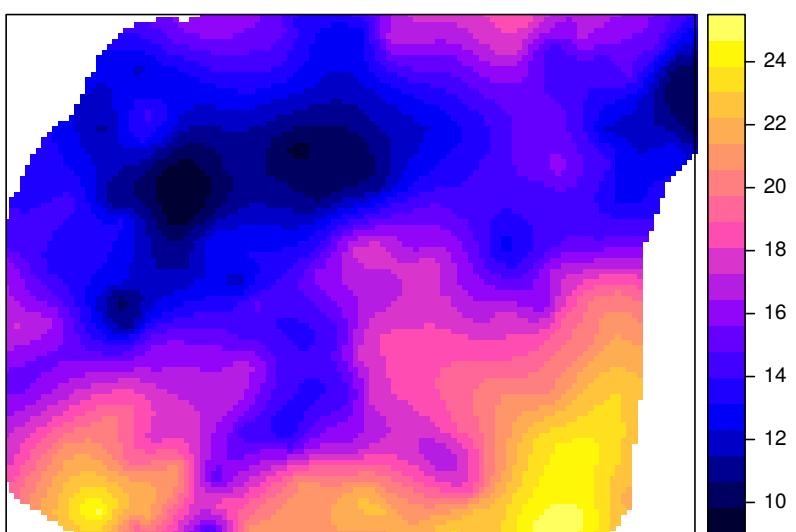
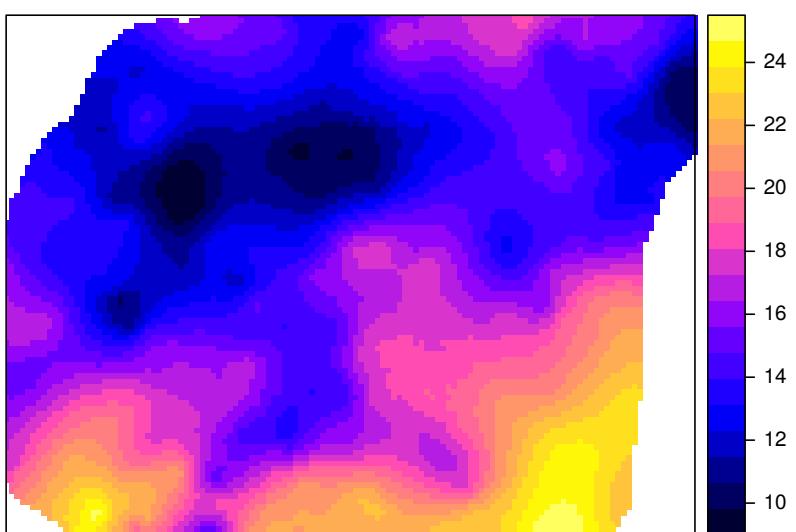
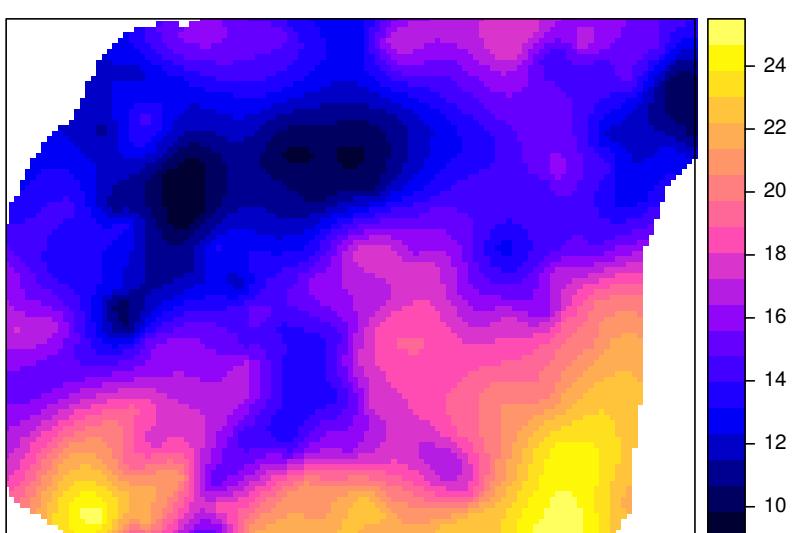
spplot(kzt, 'var1.pred')
spplot(kzt, 'var1.var')

```

**Preidykcja OK****Wariancja preidykcji OK**

## 8.5 Porównanie wyników SK, OK i KZT

Poniższe porównanie krigingu prostego (SK), zwykłego (OK) i z trendem (KZT) wykazuje niewielkie różnice w uzyskanych wynikach. W rozdziałach 10 oraz 11 pokazane będą uzyskane wyniki interpolacji temperatury powietrza korzystając z innych metod krigingu.

**SK****OK****KZT**

# Rozdział 9

## Estymacja lokalnego rozkładu prawdopodobieństwa

```
library('ggplot2')
library('gstat')
library('sp')
library('geostatbook')
data(punkty)
data(siatka)
```

### 9.1 Kriging danych kodowanych

#### 9.1.1 Kriging danych kodowanych (ang. *Indicator kriging*)

Kriging danych kodowanych to metoda krigingu oparta o dane kategoryzowane lub też dane przetworzone z postaci ciągłej do binarnej. Jest ona zazwyczaj używana jest to oszacowania prawdopodobieństwa przekroczenia zdefiniowanej wartości progowej, może być również używana do szacowania wartości z całego rozkładu. Wartości danych wykorzystywane do krigingu danych kodowanych są określone jako 0 lub 1, co reprezentuje czy wartość danej zmiennej jest powyżej czy poniżej określonego progu.

#### 9.1.2 Wady i zalety krigingu danych kodowanych

Zalety:

- Możliwość zastosowania, gdy nie interesuje nas konkretna wartość, ale znalezienie obszarów o wartości przekraczającej dany poziom
- Nie jest istotny kształt rozkładu

Wady:

- Potencjalnie trudne do modelowania semiwariogramy (szczególnie skrajnych przedziałów)
- Czasochłonność/pracochłonność

### 9.2 Kriging danych kodowanych | Przykłady

#### 9.2.1 Binaryzacja danych

Pierwszym krokiem w krigingu danych kodowanych jest stworzenie zmiennej binarnej. Na poniższym przykładzie tworzona jest nowa zmienna `temp_ind`. Przyjmuje ona wartość TRUE (czyli 1) dla pomiarów

temperatury wyższych niż 20 stopni Celsjusza, a dla pomiarów równych i niższych niż 20 stopni Celsjusza jej wartość wynosi FALSE (czyli 0).

```
summary(punkty$temp)
```

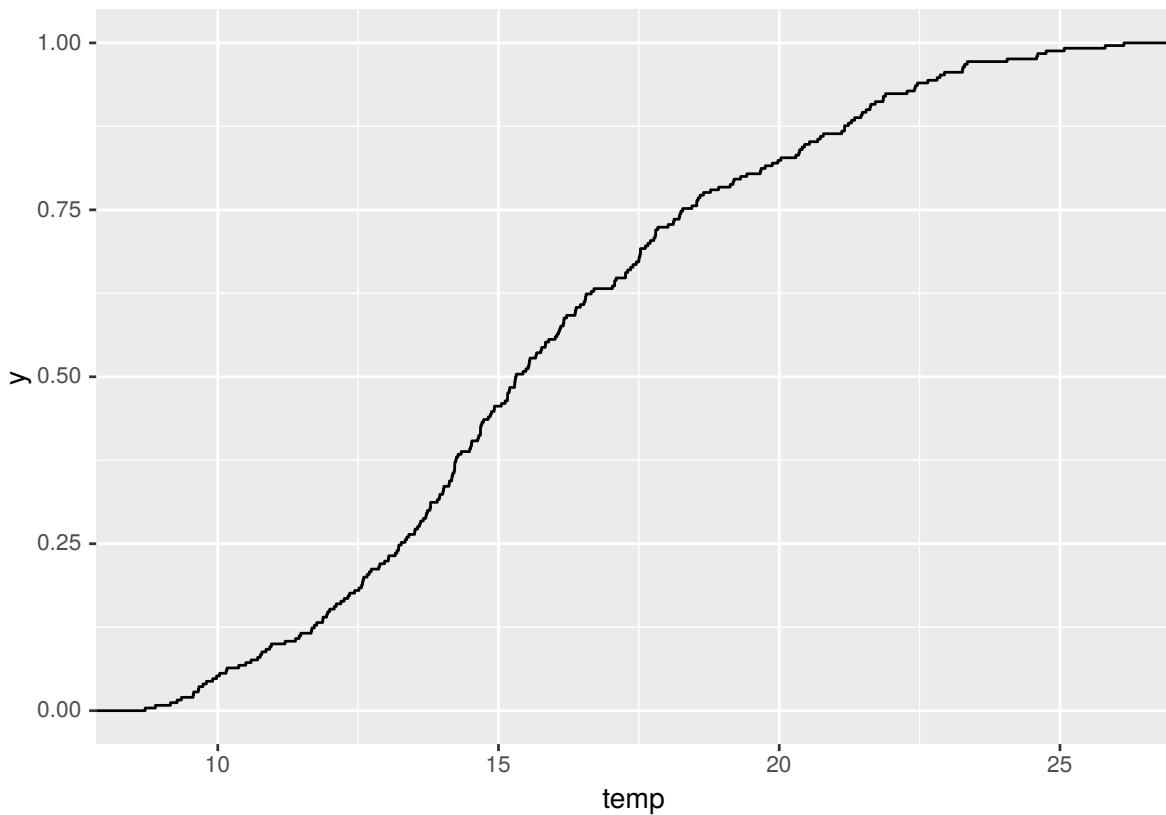
```
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
##    8.706 13.280 15.310 15.950 18.270 26.140
```

```
punkty$temp_ind <- punkty$temp > 20
summary(punkty$temp_ind)
```

```
##   Mode FALSE  TRUE NA's
## logical 206     44     0
```

W przykładzie, próg został wyznaczony arbitralnie. Istnieje oczywiście szereg innych możliwości wyznaczania progu. Można wykorzystać wiedzę zewnętrzną (np. toksyczne stężenie analizowanej substancji) lub też posłużyć się wykresem dystrybuantu do określenia istotnej zmiany wartości.

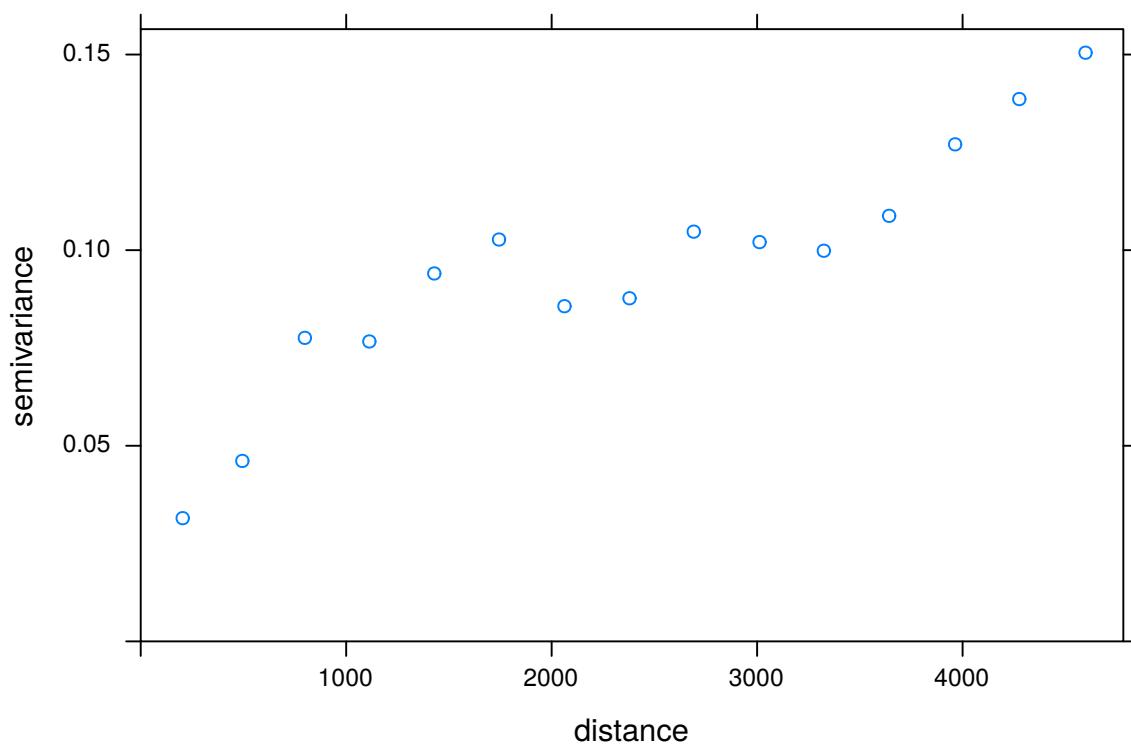
```
ggplot(punkty@data, aes(temp)) + stat_ecdf()
```



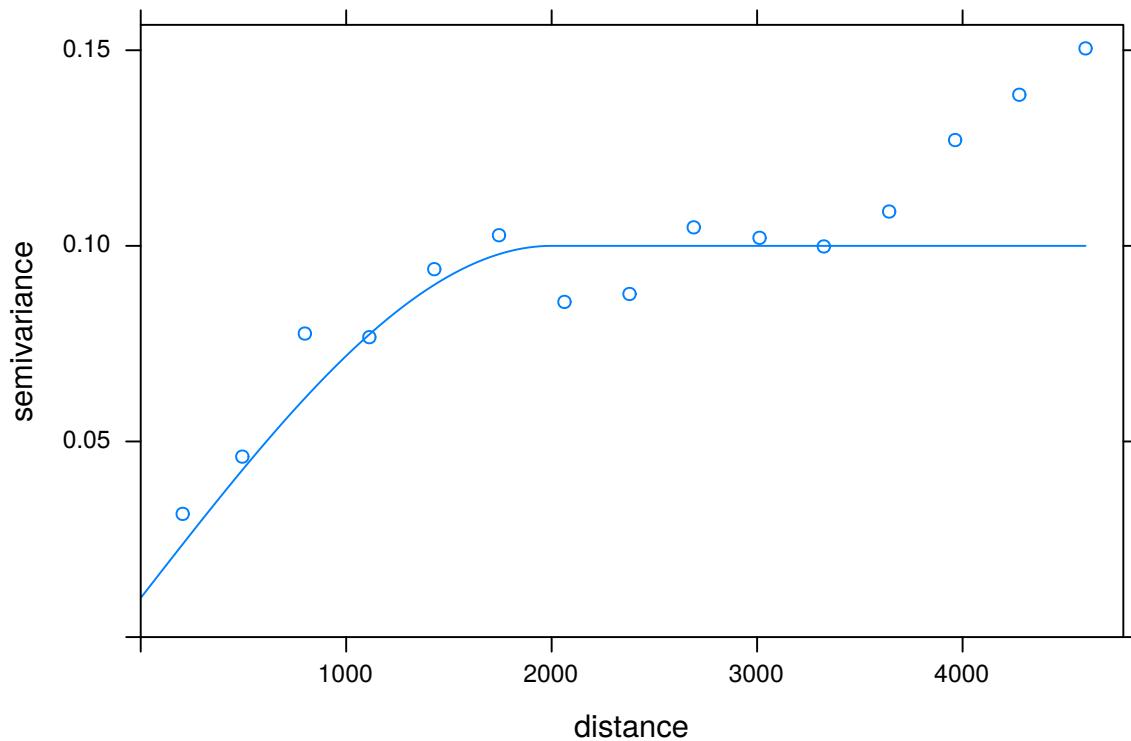
### 9.2.2 Kriging danych kodowanych (ang. *Indicator kriging*) | Modelowanie

Tworzenie i modelowanie semivariogramu empirycznego w krigingu danych kodowanych wygląda tak samo jak, np. w przypadku krigingu zwykłego. Korzystając z funkcji `variogram()` tworzony jest semivariogram empiryczny, używając `vgm()` tworzony jest model “ręczny”, który następnie jest dopasowywany z użyciem funkcji `fit.variogram()`.

```
vario_ind <- variogram(temp_ind~1, punkty)
plot(vario_ind)
```

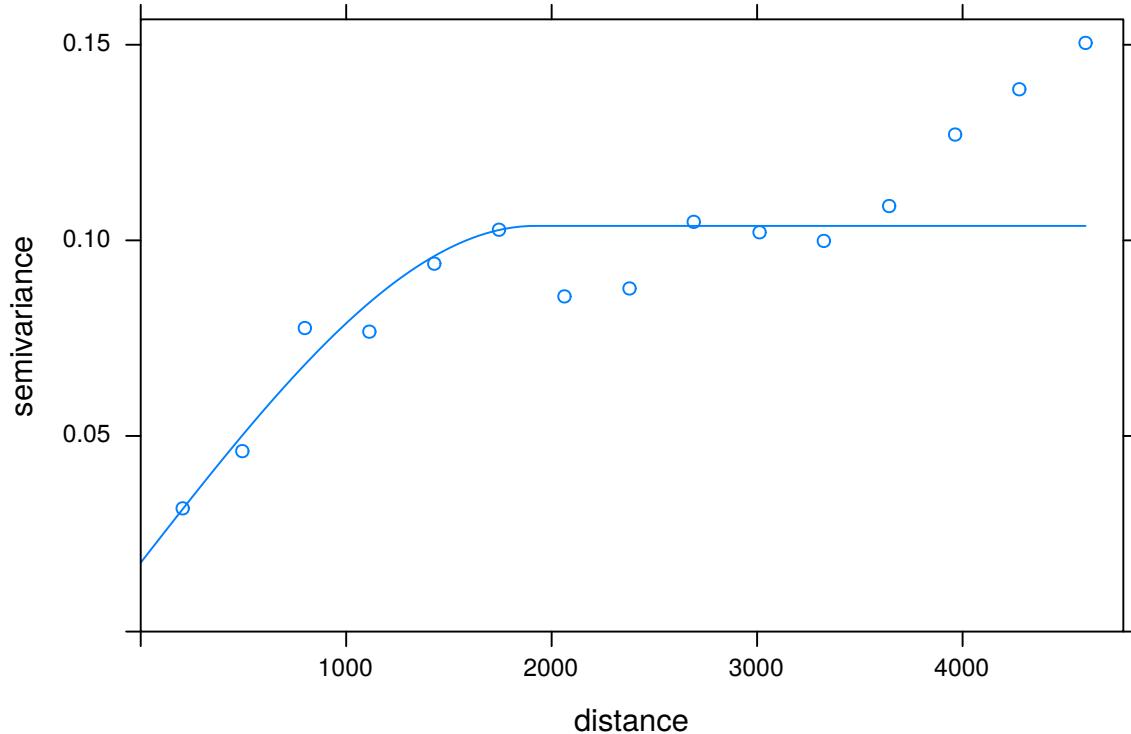


```
model_ind <- vgm(0.09, model = 'Sph', range = 2000, nugget = 0.01)
plot(vario_ind, model=model_ind)
```



```
fitted_ind <- fit.variogram(vario_ind, model=model)
fitted_ind
```

```
##   model      psill    range
## 1 Nug 0.01761559  0.00
## 2 Sph 0.08607730 1919.46
plot(vario_ind, model=fitted_ind)
```



### 9.2.3 Kriging danych kodowanych (ang. *Indicator kriging*) | Interpolacja

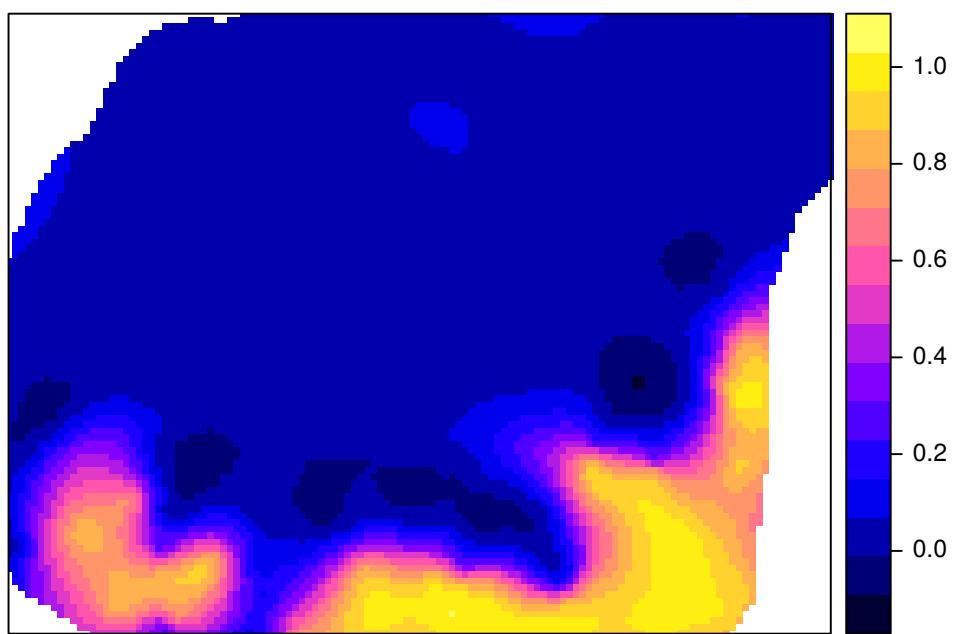
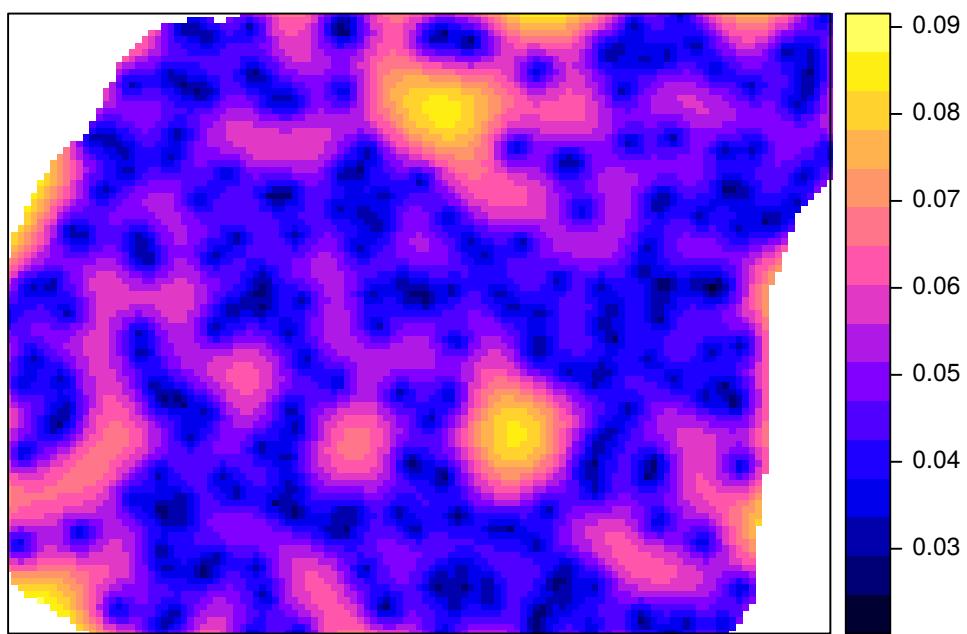
Ostatnim etapem jest stworzenie interpolacji geostatystycznej z pomocą funkcji `krige`. Wymaga ona czterech argumentów - wzoru (`temp_ind~1`), zbioru punktowego (`punkty`), siatki do interpolacji (`siatka`) oraz modelu (`fitted_ind`).

```
ik <- krige(temp_ind~1, punkty, siatka, model=fitted_ind)
```

```
## [using ordinary kriging]
```

W wyniku estymacji otrzymuje się mapę przedstawiającą prawdopodobieństwo przekroczenia zadanej wartości (w tym wypadku jest to 20 stopni Celsjusza) oraz uzyskaną wariancję predykcji.

```
spplot(ik, 'var1.pred')
spplot(ik, 'var1.var')
```

**Prawdopodobieństwo Temp > 20****Wariancja predykcji IK**

#### 9.2.4 Kriging danych kodowanych (ang. *Indicator kriging*)

Alternatywnie, zamiast tworzenia nowej zmiennej (takiej jak `temp_ind`), można wykorzystać funkcję `I`. Z jej użyciem można definiować przyjęte progi bezpośrednio do funkcji `variogram` i `krige`. Na poniższych przykładach w ten sposób ustalono trzy progi - poniżej 20, poniżej 16, oraz poniżej 12 stopni Celsjusza.

```
vario_ind20 <- variogram(I(temp<20)~1, punkty)
fitted_ind20 <- fit.variogram(vario_ind20, vgm(0.08, 'Sph', 3500, nugget=0.01))
vario_ind16 <- variogram(I(temp<16)~1, punkty)
fitted_ind16 <- fit.variogram(vario_ind16, vgm(0.18, 'Sph', 3500, nugget=0.03))
```

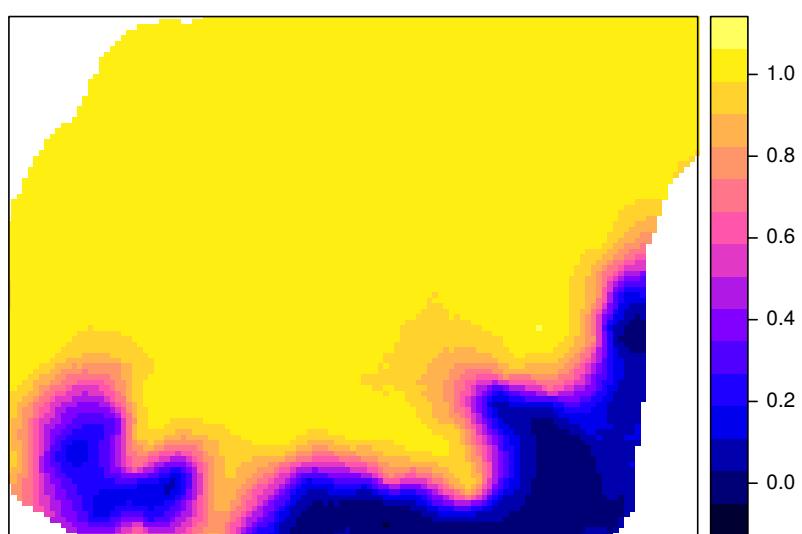
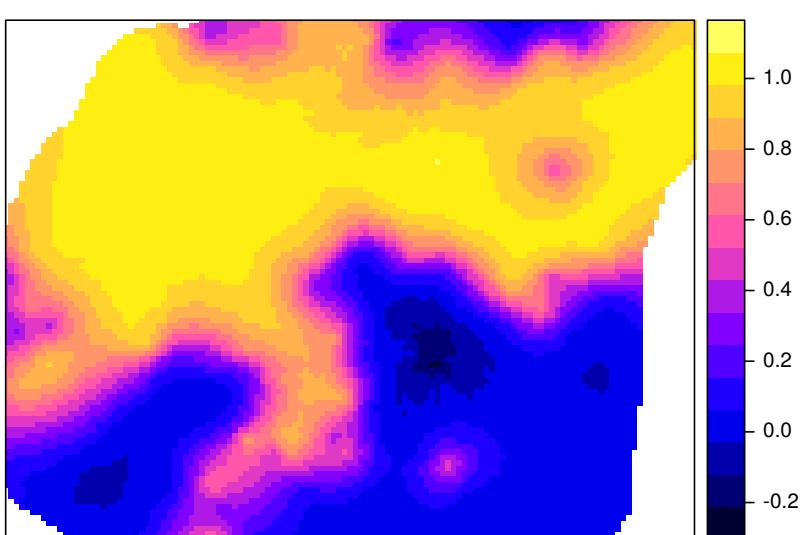
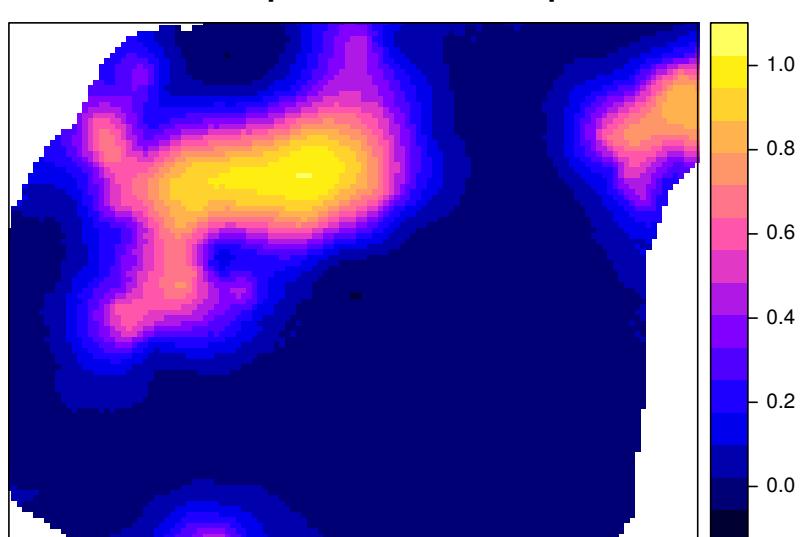
```
vario_ind12 <- variogram(I(temp<12)~1, punkty)
fitted_ind12 <- fit.variogram(vario_ind12, vgm(0.13, 'Sph', 2000, nugget=0.03))

ik20 <- krig(I(temp<20)~1, punkty, siatka, model=fitted_ind20, nmax=30)
```

```
## [using ordinary kriging]
ik16 <- krig(I(temp<16)~1, punkty, siatka, model=fitted_ind16, nmax=30)
```

```
## [using ordinary kriging]
ik12 <- krig(I(temp<12)~1, punkty, siatka, model=fitted_ind12, nmax=30)
```

```
## [using ordinary kriging]
```

**Prawdopodobieństwo Temp < 20****Prawdopodobieństwo Temp < 16****Prawdopodobieństwo Temp < 12**



# Rozdział 10

## Estymacje wielozmienne

```
library('gstat')
library('sp')
library('geostatbook')
data(punkty)
data(punkty_ndvi)
data(siatka)
```

### 10.1 Kokriging

#### 10.1.1 Kokriging (ang. *co-kriging*)

Kokriging pozwala na wykorzystanie dodatkowej zmiennej (ang. *auxiliary variable*), zwanej inaczej kozmienią (ang. *co-variable*), która może być użyta do prognozowania wartości badanej zmiennej w nieoprowadowanej lokalizacji. Zmienna dodatkowa może być pomierzona w tych samych miejscowościach, gdzie badana zmienność, jak też w innych niż badana zmienność. Możliwa jest też sytuacja, gdy zmienność dodatkowa jest pomierzona w dwóch powyższych przypadkach. Kokriging wymaga, aby obie zmienności były istotnie ze sobą skorelowane. Najczęściej kokriging jest stosowany w sytuacji, gdy zmienność dodatkowa jest łatwiejsza (tańsza) do pomierzenia niż zmienność główna. W efekcie, uzyskany zbiór danych zawiera informacje o badanej zmienności oraz gęściej opróbowane informacje o zmienności dodatkowej. Jeżeli informacje o zmienności dodatkowej są znane dla całego obszaru wówczas bardziej odpowiednią techniką będzie kriging z zewnętrznym trendem (KED).

#### 10.1.2 Kokriging | Wybór dodatkowej zmiennej

Wybór zmienności dodatkowej może opierać się na dwóch kryteriach:

- Teoretycznym
- Empirycznym

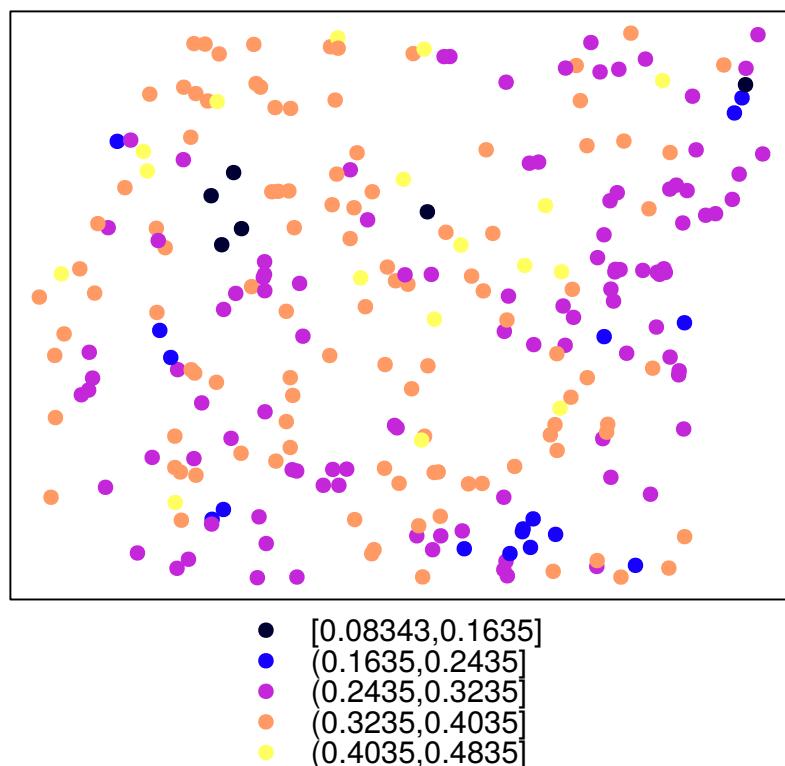
### 10.2 Krossemiwariogramy

#### 10.2.1 Krossemiwariogramy

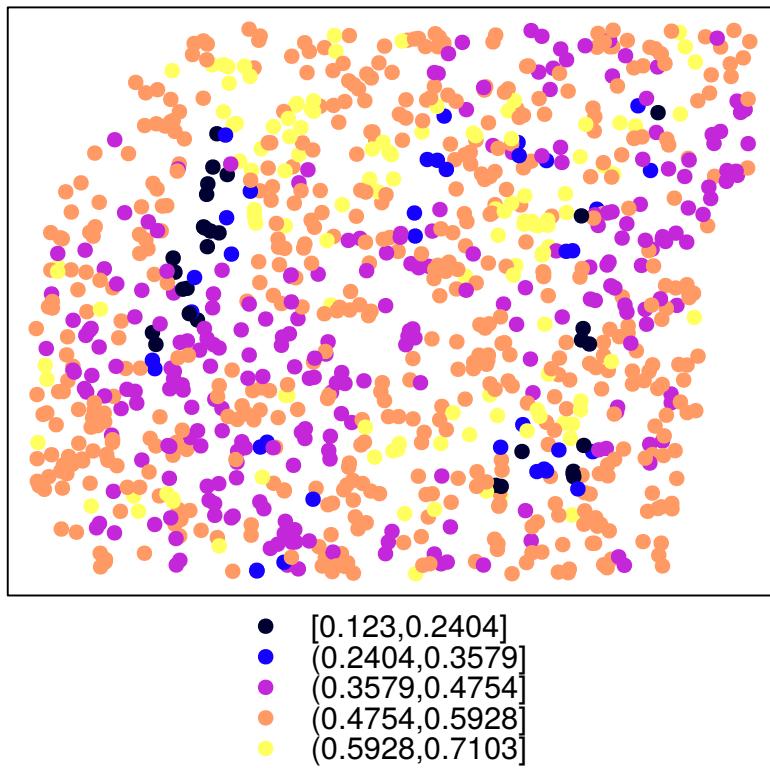
Metoda kokrigingu opiera się nie o semiwariogram, lecz o krossemiwariogramy. Krossemiwariogram jest to wariancja różnicy pomiędzy dwiema zmiennościami w dwóch lokalizacjach. Wyliczając krossemiwariogram otrzymujemy empiryczne semiwariogramy dla dwóch badanych zmienności oraz krosvariogram dla kombinacji dwóch zmienności.

W poniższym przykładzie istnieją dwie zmienne, `savi` ze zbioru `punkty` pomierzona w 250 lokalizacjach oraz `ndvi` ze zbioru `punkty_ndvi` pomierzona w 997 punktach.

```
spplot(punkty, 'savi')
```



```
spplot(punkty_ndvi, 'ndvi')
```



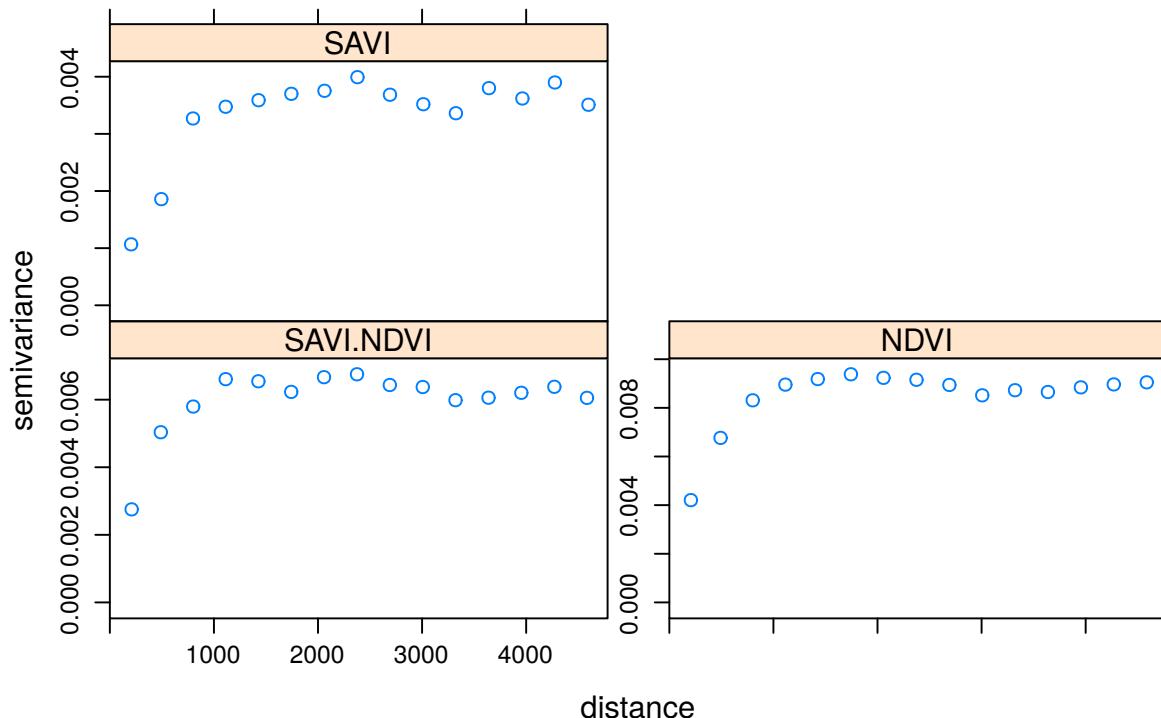
Tworzenie krossemiwariogramów odbywa się z użyciem funkcji `gstat()`. Na początku definiujemy pierwszy obiekt `g`. Składa się on z obiektu pustego (`NULL`), nazwy pierwszej zmiennej (nazwa może być dowolna), wzoru (w przykładzie `savi~1`), oraz pierwszego zbioru punktowego. Następnie do pierwszego obiektu `g` dodajemy nowe informacje również poprzez funkcję `gstat()`. Jest to nazwa obiektu (`g`), nazwa drugiej zmiennej, wzór, oraz drugi zbiór punktowy.

```
g <- gstat(NULL, id='SAVI', form = savi~1, data = punkty)
g <- gstat(g, id='NDVI', form = ndvi~1, data = punkty_ndvi)
g
```

```
## data:
## SAVI : formula = savi`~`1 ; data dim = 250 x 5
## NDVI : formula = ndvi`~`1 ; data dim = 997 x 1
```

Z uzyskanego w ten sposób obiektu tworzymy krossemiwariogram (funkcja `variogram()`), a następnie go wizualizujemy używając funkcji `plot()`.

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



## 10.3 Modelowanie krossemiwariogramów

### 10.3.1 Modelowanie krossemiwariogramów

Modelowanie krossemiwariogramów, podobnie jak ich tworzenie, odbywa się używając funkcji `gstat()`. Podaje się w niej wcześniejszy obiekt `g`, model, oraz argument `fill.all=TRUE`. Ten ostatni parametr powoduje, że model dodawany jest do wszystkich elementów krossemiwariogramu.

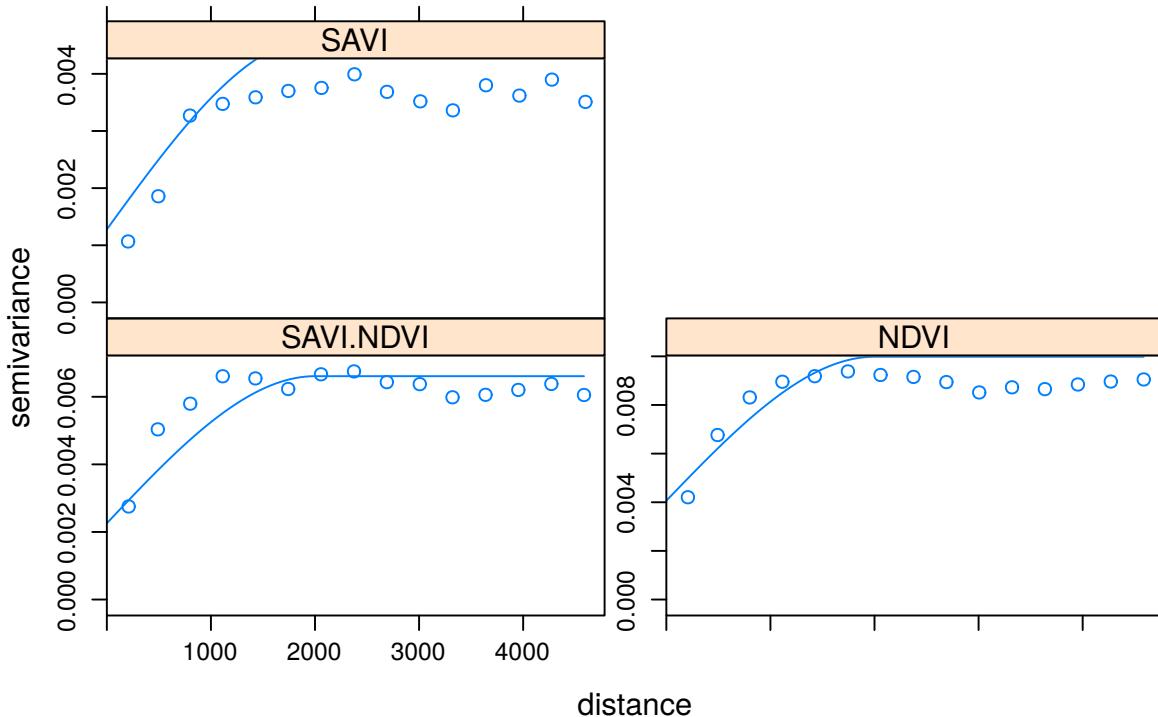
```
g <- gstat(g, model=vgm(0.006, 'Sph', 2000, 0.001), fill.all=TRUE)
```

W przypadku semiwariogramów funkcja `fit.variogram()` służyła dopasowaniu parametrów modelu do semiwariogramu empirycznego. Podobną rolę w krossemiwariogramach spełnia funkcja `fit.lmc()`. Dopasowuje ona liniowy model koregionalizacji do semiwariogramów wielozmiennych. Funkcja `fit.lmc()` oczekuje co najmniej dwóch elementów, krossemiwariogramu oraz modelów krossemiwariancji. W poniższym przykładzie dodatkowo użyto parametru `correct.diagonal = 1.01`, z uwagi na to że analizowane zmienne wykazywały bardzo silną korelację.

```
# zmienne są bardzo mocno skorelowane - dlatego użyto argumentu correct.diagonal = 1.01
g_fit <- fit.lmc(v, g, correct.diagonal = 1.01)
g_fit

## data:
## SAVI : formula = savi`~`1 ; data dim = 250 x 5
## NDVI : formula = ndvi`~`1 ; data dim = 997 x 1
## variograms:
##           model      psill range
## SAVI[1]    Nug 0.001278509     0
## SAVI[2]    Sph 0.003338946 2000
## NDVI[1]    Nug 0.004065525     0
## NDVI[2]    Sph 0.005918453 2000
## SAVI.NDVI[1] Nug 0.002257298     0
```

```
## SAVI.NDVI[2]    Sph 0.004352821  2000
plot(v, g_fit)
```



## 10.4 Kokriging

### 10.4.1 Kokriging

Posiadając dopasowane modele oraz siatkę można uzyskać wynik używając funkcji `predict()`.

```
ck <- predict(g_fit, siatka)
```

```
## Linear Model of Coregionalization found. Good.
## [using ordinary cokriging]
```

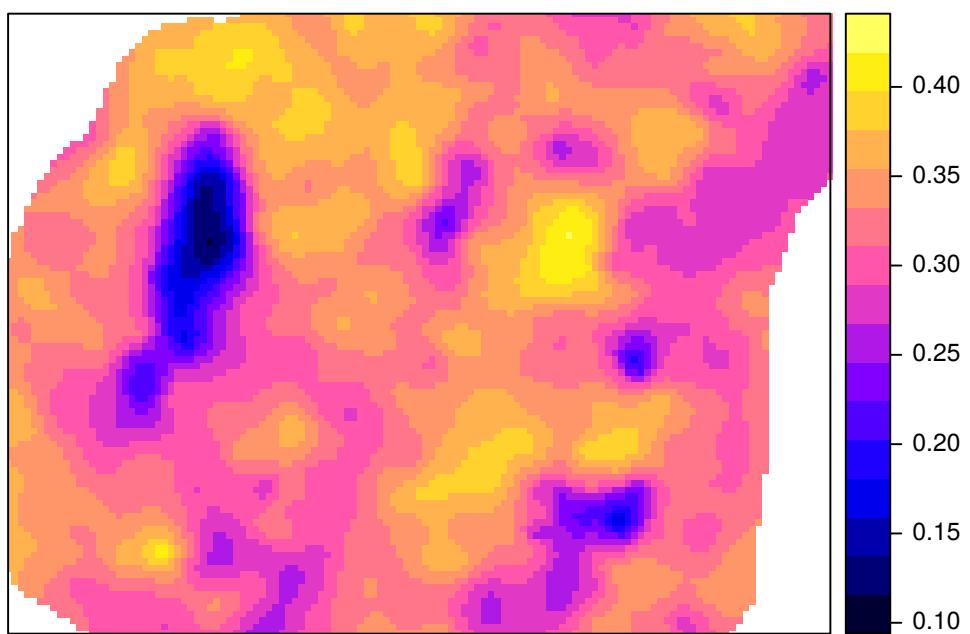
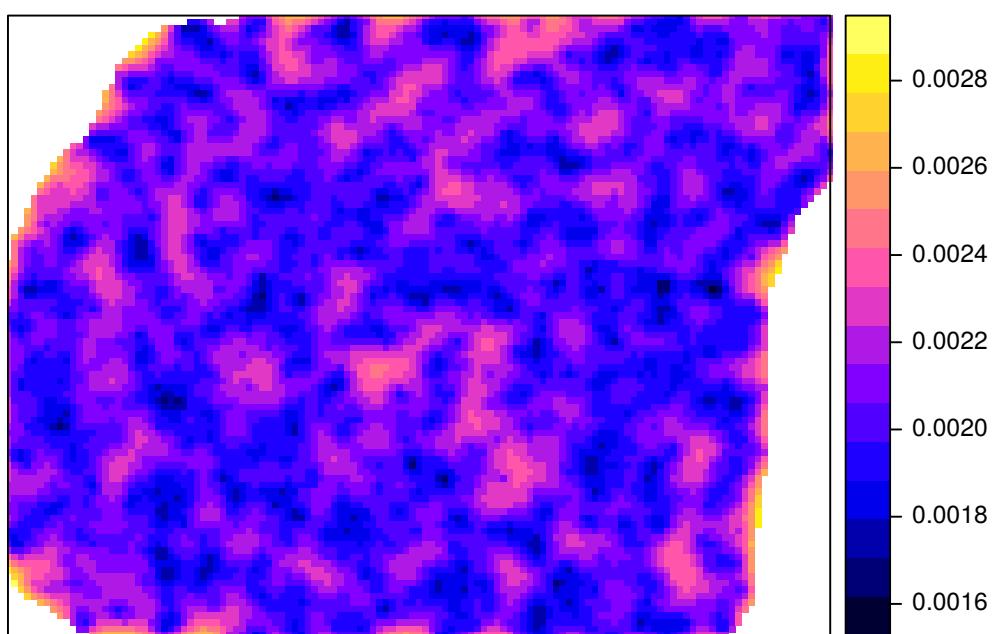
W efekcie otrzymujemy pięć zmiennych:

1. `SAVI.pred` - estymacja zmiennej `savi`
2. `SAVI.var` - wariancja zmiennej `savi`
3. `NDVI.pred` - estymacja zmiennej `ndvi`
4. `NDVI.var` - wariancje zmiennej `ndvi`
5. `cov.SAVI.NDVI` - kowariancje zmiennych `savi` oraz `ndvi`

```
summary(ck)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min     max
## x 745586.7 756926.7
## y 712661.2 721211.2
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
```

```
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 10993
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## s1          745586.7      90      127
## s2          712661.2      90      96
## Data attributes:
##   SAVI.pred      SAVI.var      NDVI.pred      NDVI.var
## Min.    :0.1152  Min.    :0.001615  Min.    :0.2336  Min.    :0.004750
## 1st Qu.:0.3001  1st Qu.:0.001958  1st Qu.:0.4715  1st Qu.:0.005188
## Median  :0.3255  Median  :0.002046  Median  :0.5056  Median  :0.005336
## Mean    :0.3205  Mean    :0.002060  Mean    :0.4996  Mean    :0.005365
## 3rd Qu.:0.3480  3rd Qu.:0.002149  3rd Qu.:0.5365  3rd Qu.:0.005513
## Max.    :0.4197  Max.    :0.002862  Max.    :0.6280  Max.    :0.006780
## cov.SAVI.NDVI
## Min.    :0.002658
## 1st Qu.:0.003045
## Median  :0.003157
## Mean    :0.003178
## 3rd Qu.:0.003289
## Max.    :0.004232
spplot(ck, 'SAVI.pred')
spplot(ck, 'SAVI.var')
```

**Preidykcja CK****Wariancja CK**



## Rozdział 11

# Wykorzystanie do estymacji danych uzupełniających

```
library('gstat')
library('sp')
library('geostatbook')
data(punkty)
data(siatka)
```

W wielu przypadkach, oprócz konkretnych pomiarów, istnieje również informacja na temat zmienności innych cech na analizowanym obszarze. W sytuacji, gdy dodatkowe zmienne są skorelowane ze zmienną analizowaną można wykorzystać jedną z metod krigingu wykorzystującą dane uzupełniające, tj. kriging stratyfikowany, prosty kriging ze zmiennymi średnimi lokalnymi, czy kriging uniwersalny.

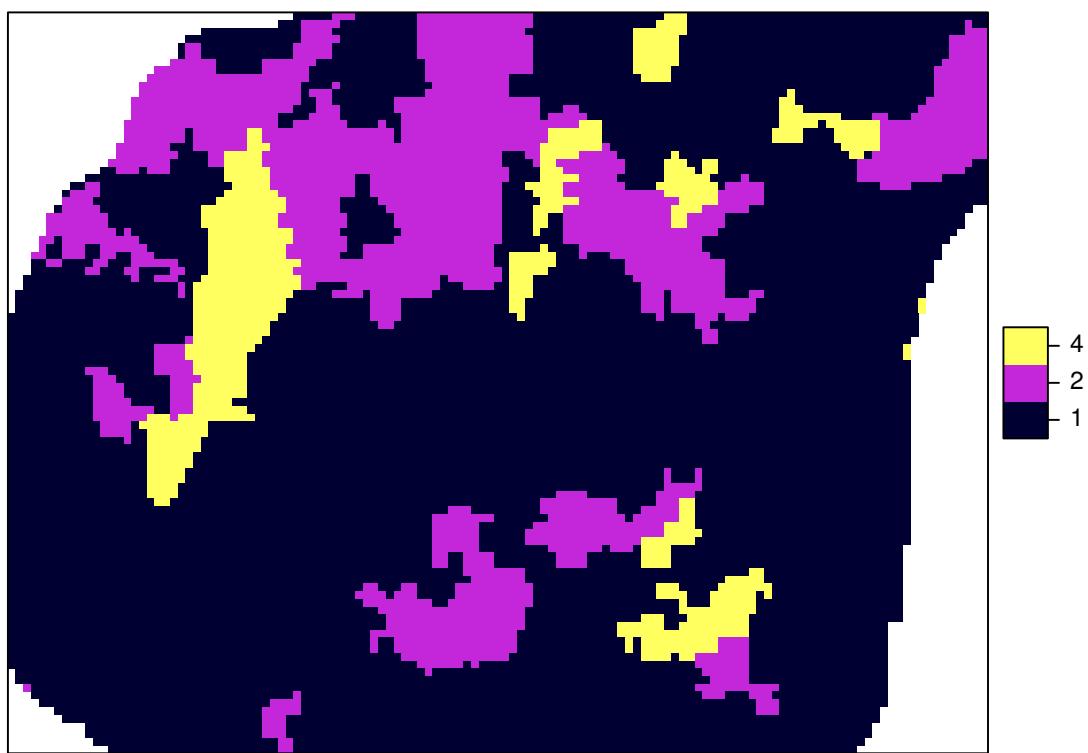
### 11.1 Kriging stratyfikowany

#### 11.1.1 Kriging stratyfikowany (ang. *Kriging within strata*)

Kriging stratyfikowany zakłada, że zmienna badanego zjawiska zależy od cechy jakościowej (kategoryzowanej). Przykładowo, wartość badanej zmiennej jest różna w zależności od pokrycia terenu. Kriging stratyfikowany wymaga posiadania danych zmiennej jakościowej (kategoryzowanej) na całym badanym obszarze.

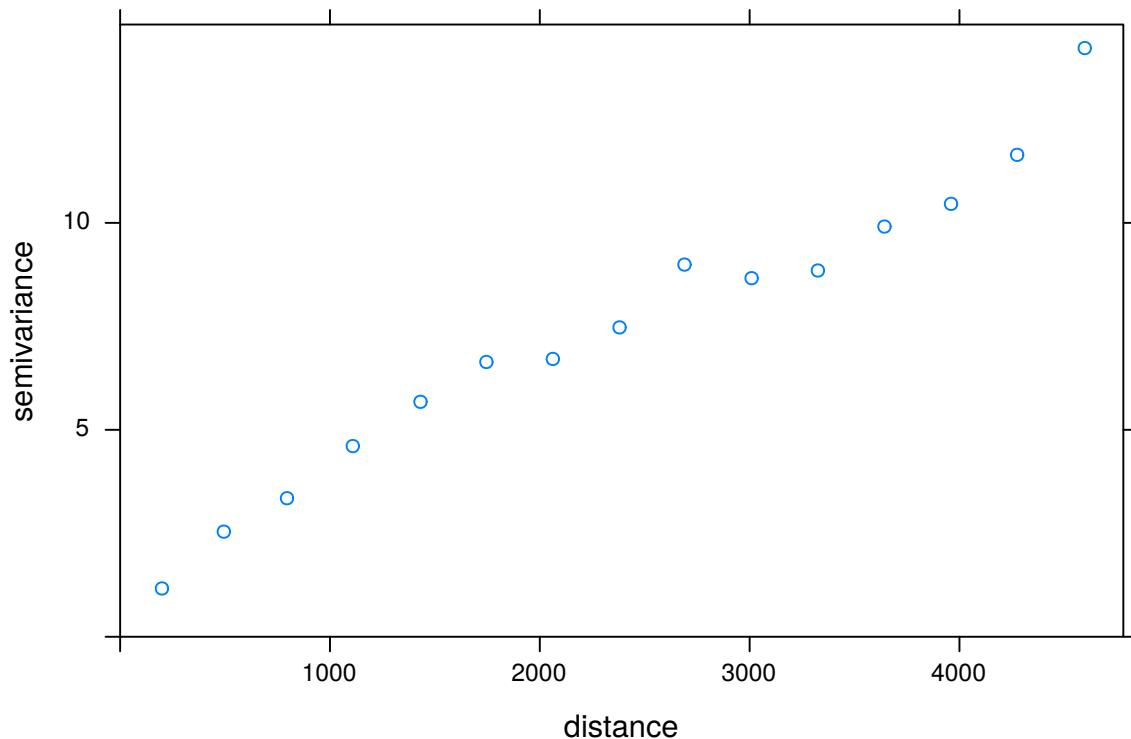
W poniższym przykładzie zmienną jakościową jest uproszczone pokrycie terenu ze zmiennej `clc`. Przyjmuje ono jedno z trzech wartości. 1 oznacza obszary rolnicze, 2 oznacza obszary leśne, a 4 oznacza wody powierzchniowe.

```
siatka$clc <- as.factor(siatka$clc)
spplot(siatka, 'clc')
```

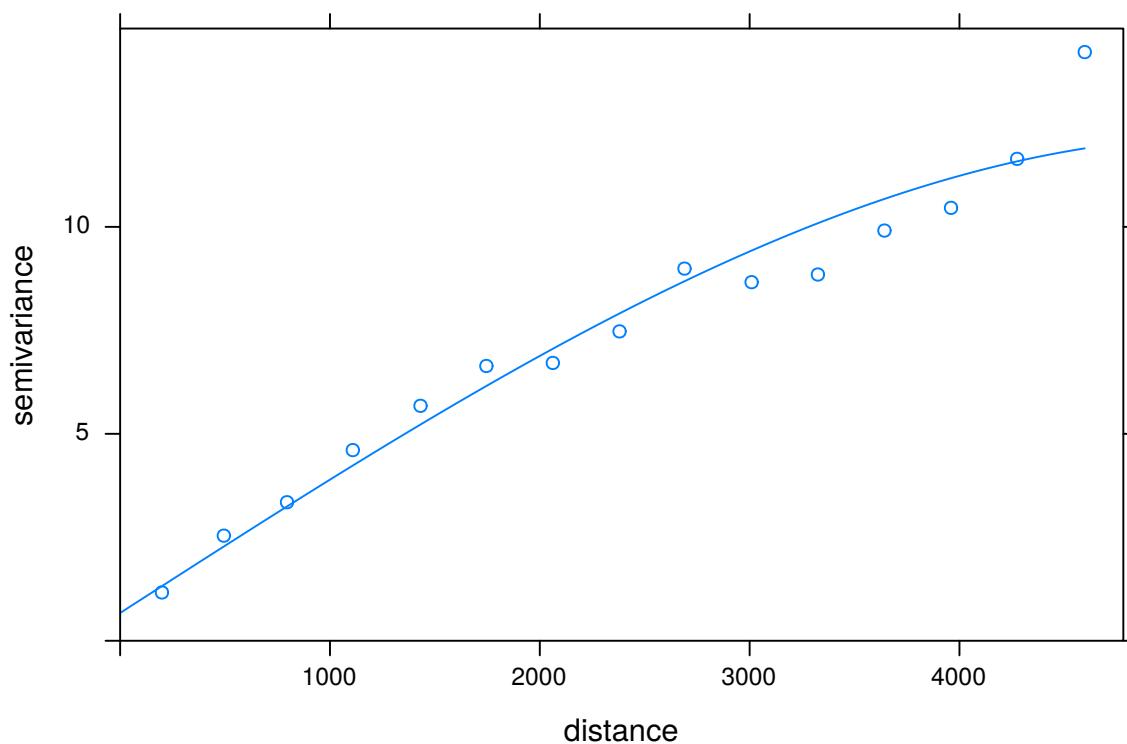


Kriging stratyfikowany polega na niezależnym tworzeniu i modelowaniu semiwariogramów dla każdej z kategorii.

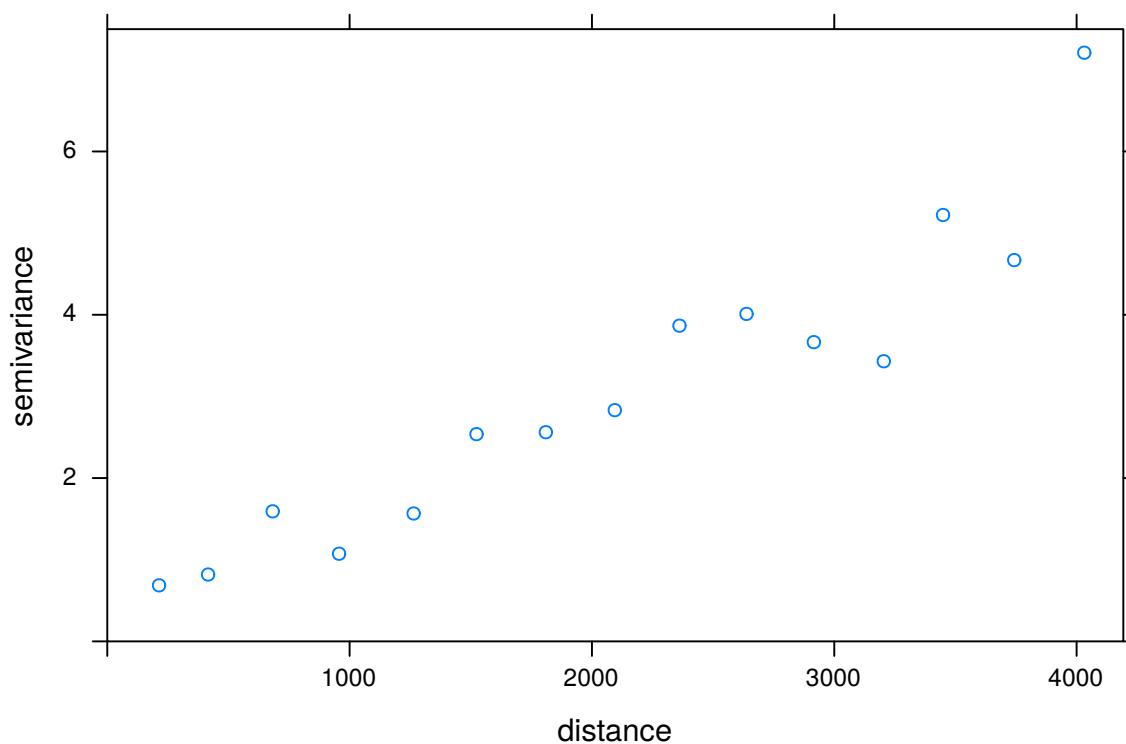
```
vario_kws1 <- variogram(temp~1, punkty[punkty$clc==1, ])
plot(vario_kws1)
```



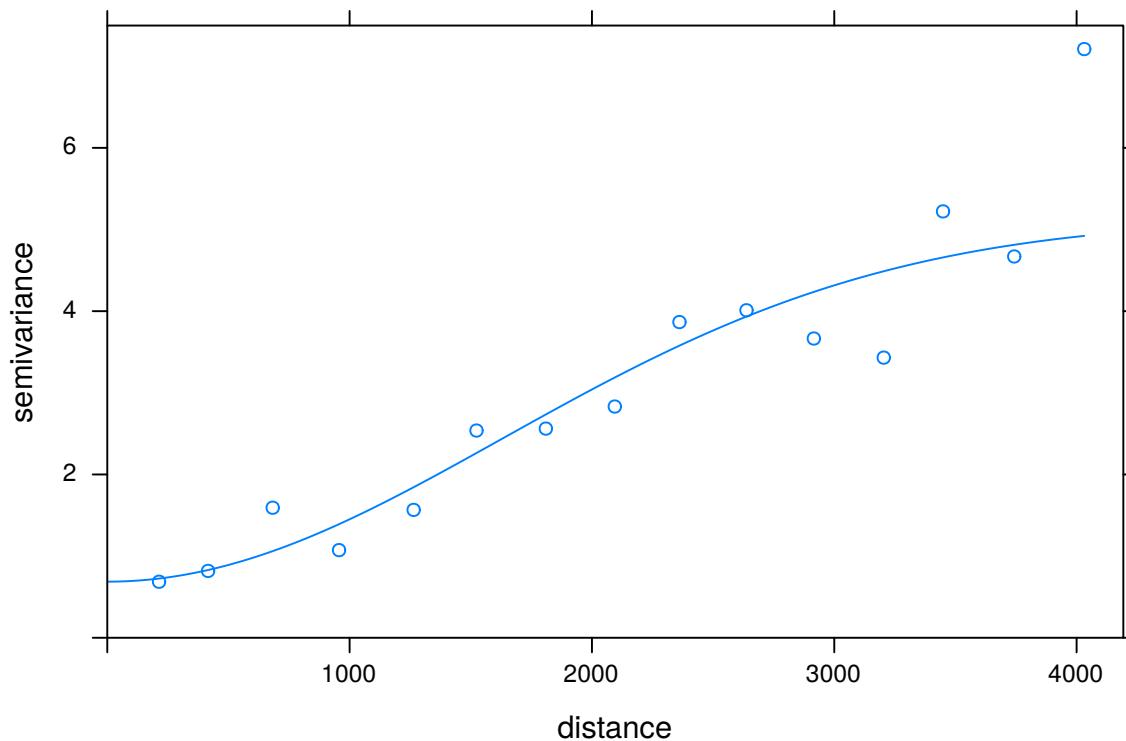
```
fitted_kws1 <- fit.variogram(vario_kws1, vgm(10, model = 'Sph', range = 4500, nugget = 0.5))
plot(vario_kws1, fitted_kws1)
```



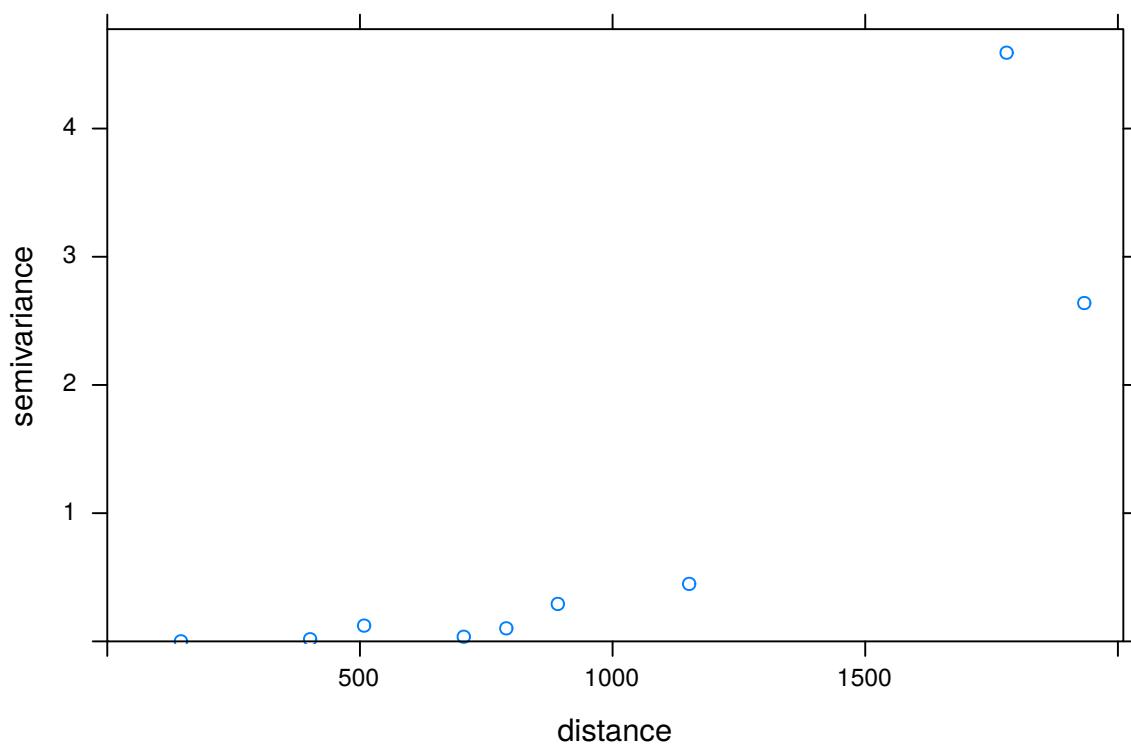
```
vario_kws2 <- variogram(temp~1, punkty[punkty$clc==2, ])
plot(vario_kws2)
```



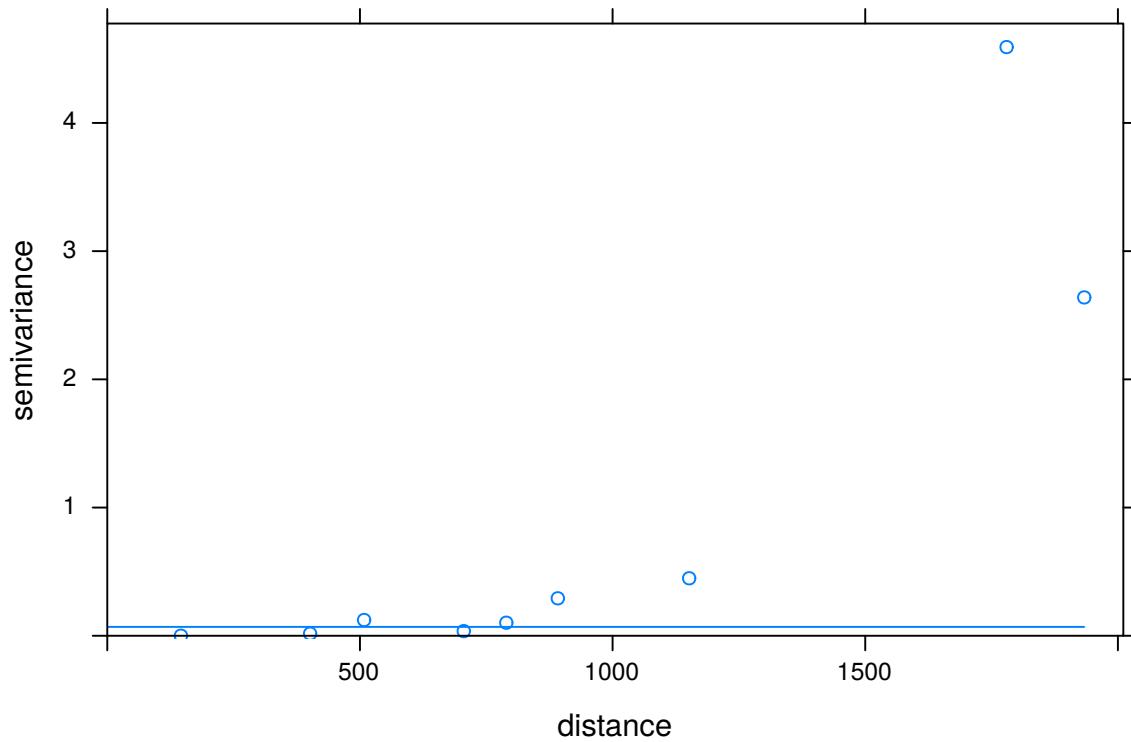
```
fitted_kws2 <- fit.variogram(vario_kws2, vgm(5, model = 'Gau', range = 4500, nugget = 0.1))
plot(vario_kws2, fitted_kws2)
```



```
vario_kws4 <- variogram(temp~1, punkty[punkty$clc==4, ])
plot(vario_kws4)
```



```
fitted_kws4 <- fit.variogram(vario_kws4, vgm(0.5, model = 'Nug', range = 0))
plot(vario_kws4, fitted_kws4)
```

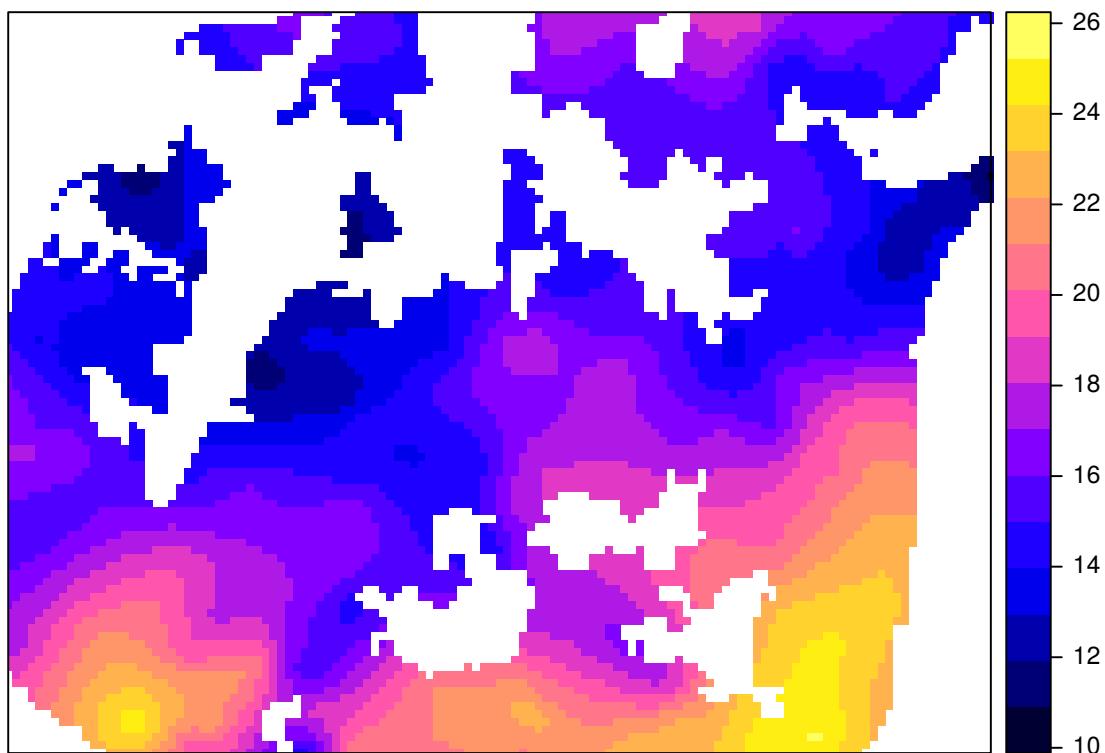


Następnie dla każdego obszaru przeprowadzona jest niezależna estymacja wartości analizowanej cechy. Należy jedynie wcześniej zadbać, by w siatce nie było elementów NA dotyczących zmiennych jakościowych.

W przykładzie tworzona jest nowa siatka (`siatka2`) nie zawierająca braków wartości dla zmiennej `clc`.

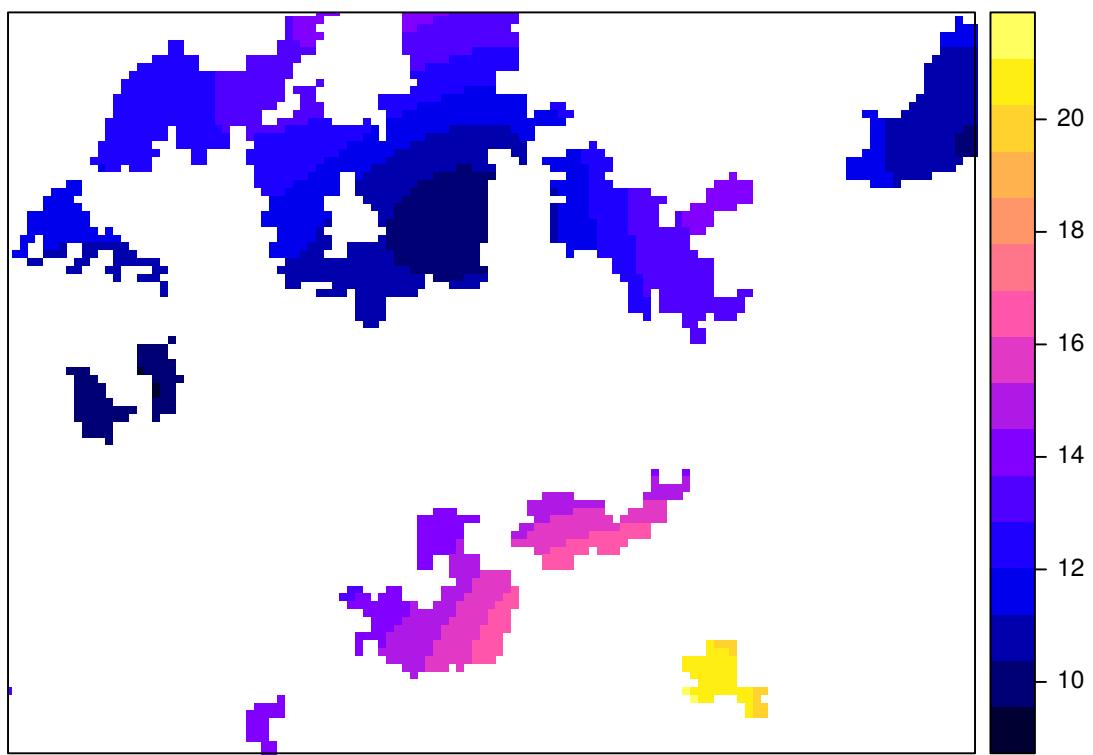
```
siatka2 <- siatka[!is.na(siatka$clc), ]  
kws1 <- krige(temp~1, punkty[punkty$clc==1, ], siatka2[na.omit(siatka2$clc==1), ], model = fitted_kw
```

```
## [using ordinary kriging]  
spplot(kws1, 'var1.pred')
```

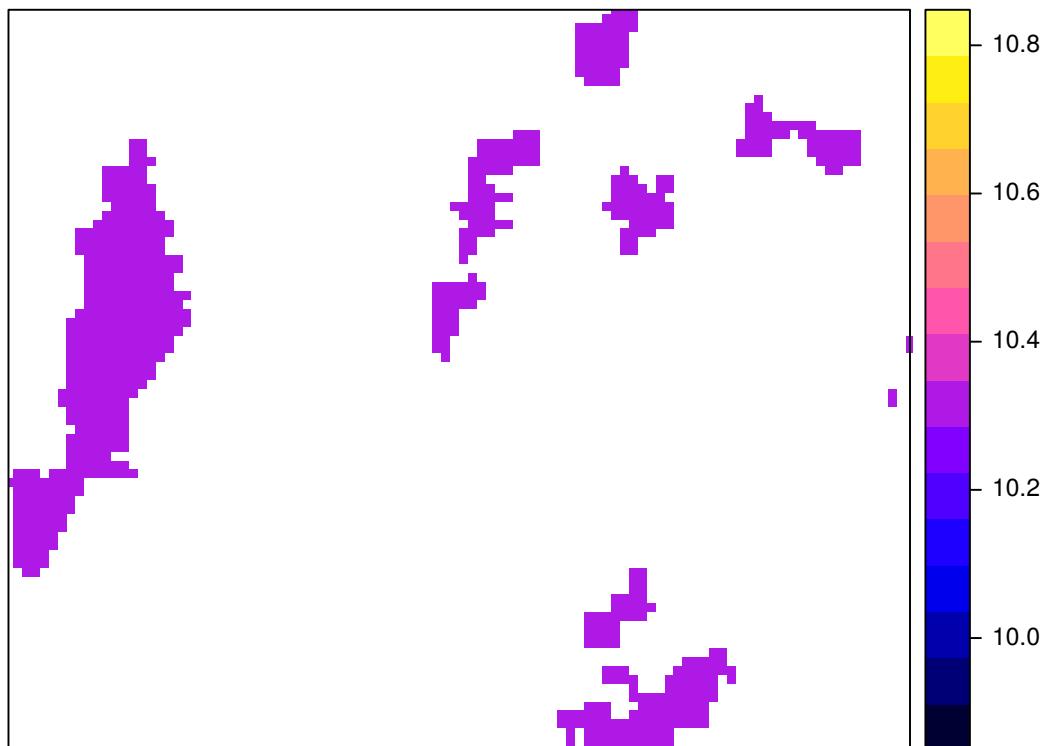


```
kws2 <- krige(temp~1, punkty[punkty$clc==2, ], siatka2[na.omit(siatka2$clc==2), ], model = fitted_kw
```

```
## [using ordinary kriging]  
spplot(kws2, 'var1.pred')
```



```
kws4 <- kriging(temp~1, punkty[punkty$clc==4, ], siatka2[na.omit(siatka2$clc==4), ], model = fitted_kw  
## [using ordinary kriging]  
spplot(kws4, 'var1.pred')
```



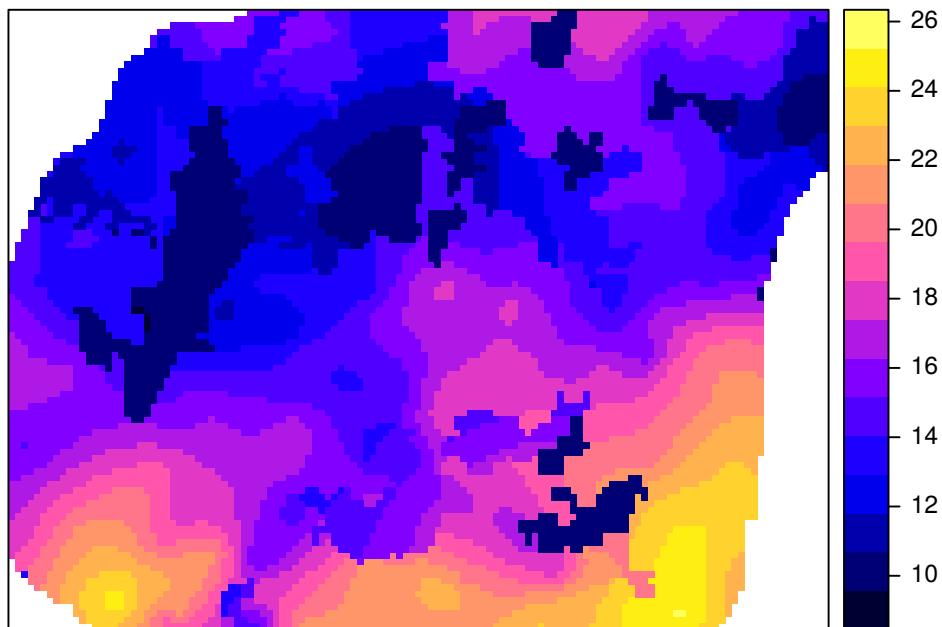
Ostatnim etapem jest połączenie cząstkowych wyników w jeden obiekt klasy `SpatialPixelsDataFrame`.

```
kws <- rbind(as.data.frame(kws1), as.data.frame(kws2), as.data.frame(kws4))
coordinates(kws) <- ~x+y
kws <- as(kws, 'SpatialPixelsDataFrame')
```

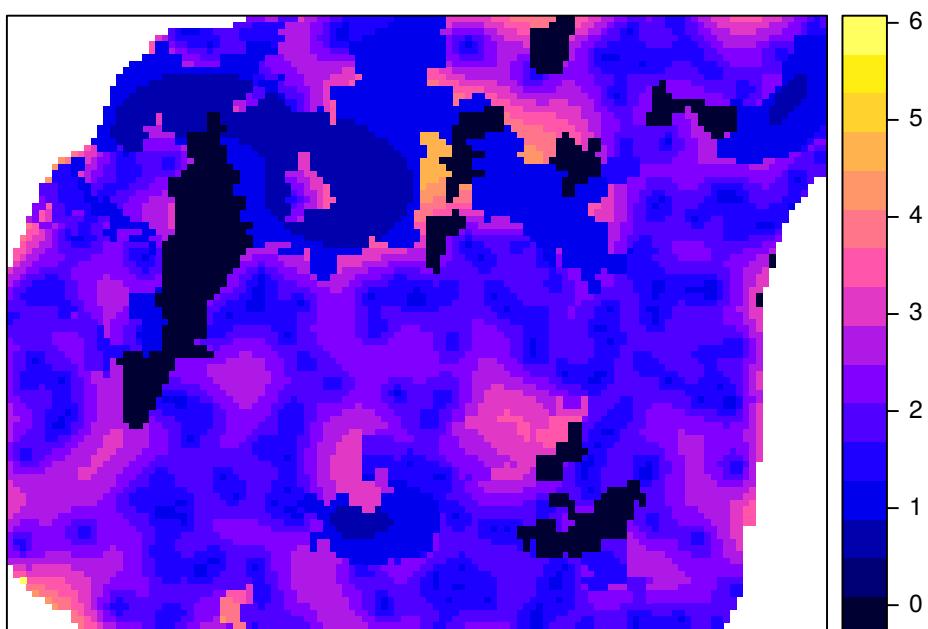
Uzyskane w ten sposób wyniki znacznie różnią się od estymacji krigingem prostym czy zwykłym, wykazując odrębność zmienności w poszczególnych kategoriach pokrycia/użytkowania terenu.

```
spplot(kws, 'var1.pred', sp.layout=punkty)
spplot(kws, 'var1.var', sp.layout=punkty)
```

**Preidykcja KWS**



**Wariancja preidykcji KWS**



## 11.2 Prosty kriging ze zmiennymi średnimi lokalnymi (LVM)

### 11.2.1 Prosty kriging ze zmiennymi średnimi lokalnymi (LVM) (ang. *Simple kriging with varying local means*)

Prosty kriging ze zmiennymi średnimi lokalnymi zamiast znanej (stałej) stacjonarnej średniej wykorzystuje zmienne średnie lokalne uzyskane na podstawie innej informacji. Lokalna średnia może być uzyskana za pomocą wyliczenia regresji liniowej pomiędzy zmienną badaną a zmienną dodatkową. W takiej sytuacji konieczne jest użycie funkcji `lm()`. W poniższym przykładzie budowany jest model liniowy relacji pomiędzy temperaturą powietrza (`temp`), a wysokością nad poziomem morza (`srtm`).

```
coef <- lm(temp~srtm, punkty)$coef
coef
```

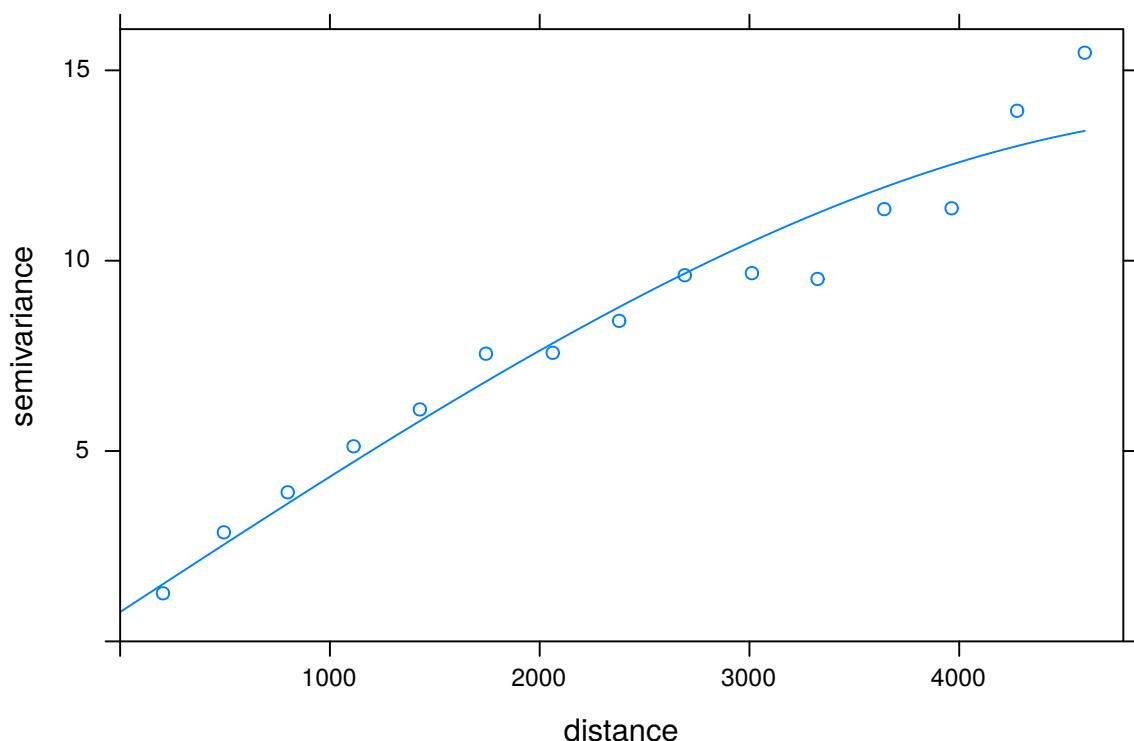
```
## (Intercept)          srtm
## 17.506469957 -0.007291269
```

Wykorzystując relację pomiędzy tymi dwoma zmiennymi tworzony jest semiwariogram empiryczny, który następnie jest modelowany.

```
vario <- variogram(temp~srtm, punkty)
model_sim <- vgm(10, model = 'Sph', range = 4000, nugget = 1)
model_sim
```

```
##   model psill range
## 1   Nug     1    0
## 2   Sph    10  4000
fitted_sim <- fit.variogram(vario, model_sim)
fitted_sim
```

```
##   model      psill      range
## 1   Nug  0.7705839  0.000
## 2   Sph 13.1155524 5474.628
plot(vario, model=fitted_sim)
```



Ostatnim krokiem jest estymacja geostatystyczna, w której oprócz czterech podstawowych argumentów, definiujemy także parametr **beta**. W tym wypadku jest to wypadku obiekt uzyskany na podstawie regresji liniowej.

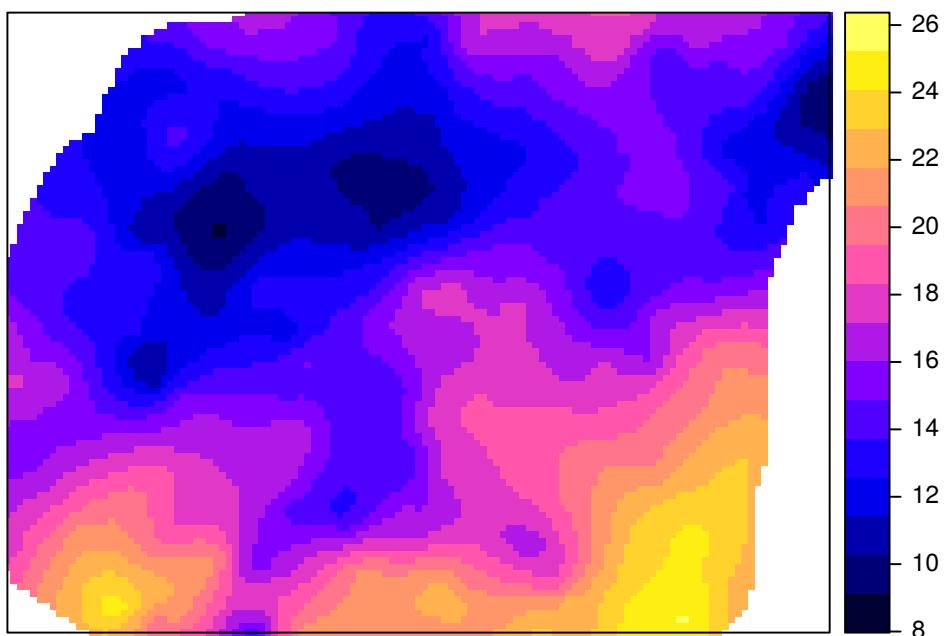
```
sk_lvm <- krige(temp~srtm, punkty, siatka, model=fitted_sim, beta = coef)
```

```
## [using simple kriging]
summary(sk_lvm)
```

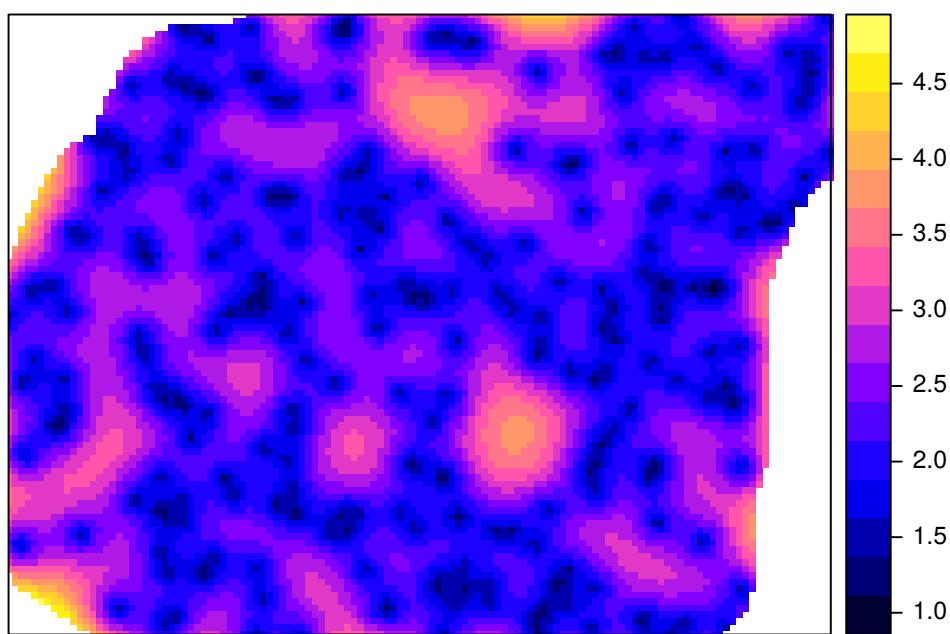
```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min     max
## x 745586.7 756926.7
## y 712661.2 721211.2
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 10993
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## s1          745586.7      90      127
## s2          712661.2      90      96
## Data attributes:
##   var1.pred      var1.var
## Min. : 9.098  Min. :1.108
## 1st Qu.:13.215 1st Qu.:1.931
## Median :15.237  Median :2.241
## Mean   :15.864  Mean   :2.312
## 3rd Qu.:18.049 3rd Qu.:2.637
## Max.  :25.240  Max.  :4.705
## NA's   :43       NA's   :43
```

```
spplot(sk_lvm, 'var1.pred')
spplot(sk_lvm, 'var1.var')
```

**Predykcja SK LVM**



**Wariancja predykcji SK LVM**



## 11.3 Kriging uniwersalny

### 11.3.1 Kriging uniwersalny (ang. *Universal kriging*)

Kriging uniwersalny, określany również jako kriging z trendem (ang. *Kriging with a trend model*) zakłada, że nieznana średnia lokalna zmienia się stopniowo na badanym obszarze. W krigingu uniwersalnym

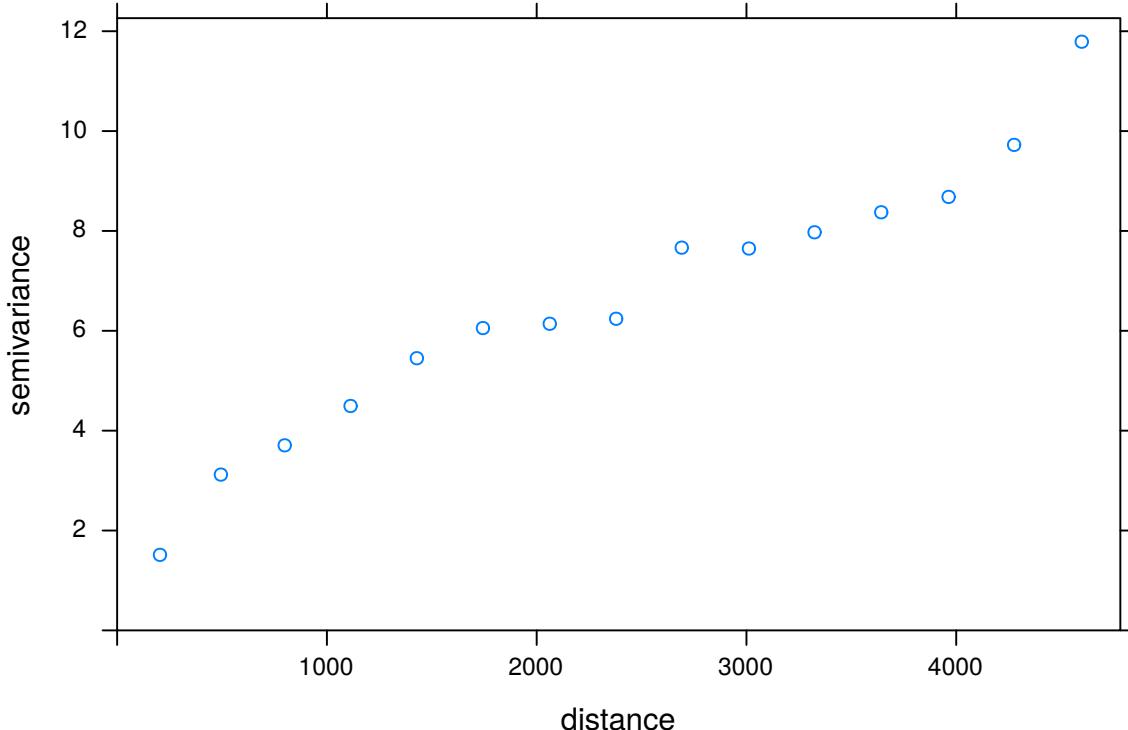
możemy stosować zarówno zmienne jakościowe, jak i ilościowe.

W pierwszym przykładzie, kriging uniwersalny służy stworzeniu semiwariogramu, modelowaniu oraz estymacji temperatury powietrza z użyciem zmiennej pokrycia terenu.

```
punkty$clc <- as.factor(punkty$clc)
vario_uk1 <- variogram(temp~clc, punkty)
vario_uk1
```

```
##      np      dist     gamma dir.hor dir.ver   id
## 1  127  204.2244  1.513128     0      0 var1
## 2  271  494.5572  3.120279     0      0 var1
## 3  522  798.7911  3.706435     0      0 var1
## 4  587 1112.7783  4.494623     0      0 var1
## 5  856 1428.7866  5.451369     0      0 var1
## 6  920 1743.7864  6.054877     0      0 var1
## 7 1068 2062.3041  6.140934     0      0 var1
## 8 1106 2378.9333  6.241295     0      0 var1
## 9 1184 2691.5206  7.665422     0      0 var1
## 10 1215 3011.7729  7.647765     0      0 var1
## 11 1362 3324.6705  7.973022     0      0 var1
## 12 1379 3642.5616  8.374149     0      0 var1
## 13 1405 3963.6776  8.681657     0      0 var1
## 14 1468 4276.0078  9.723868     0      0 var1
## 15 1482 4598.4144 11.789961     0      0 var1
```

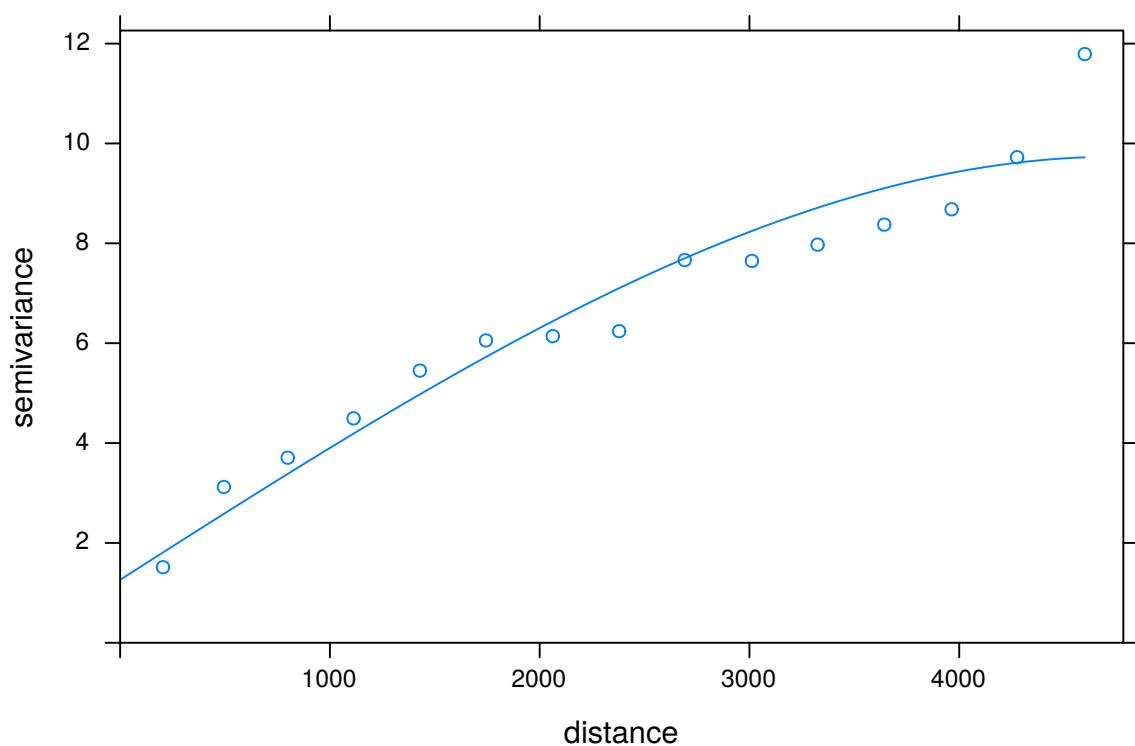
```
plot(vario_uk1)
```



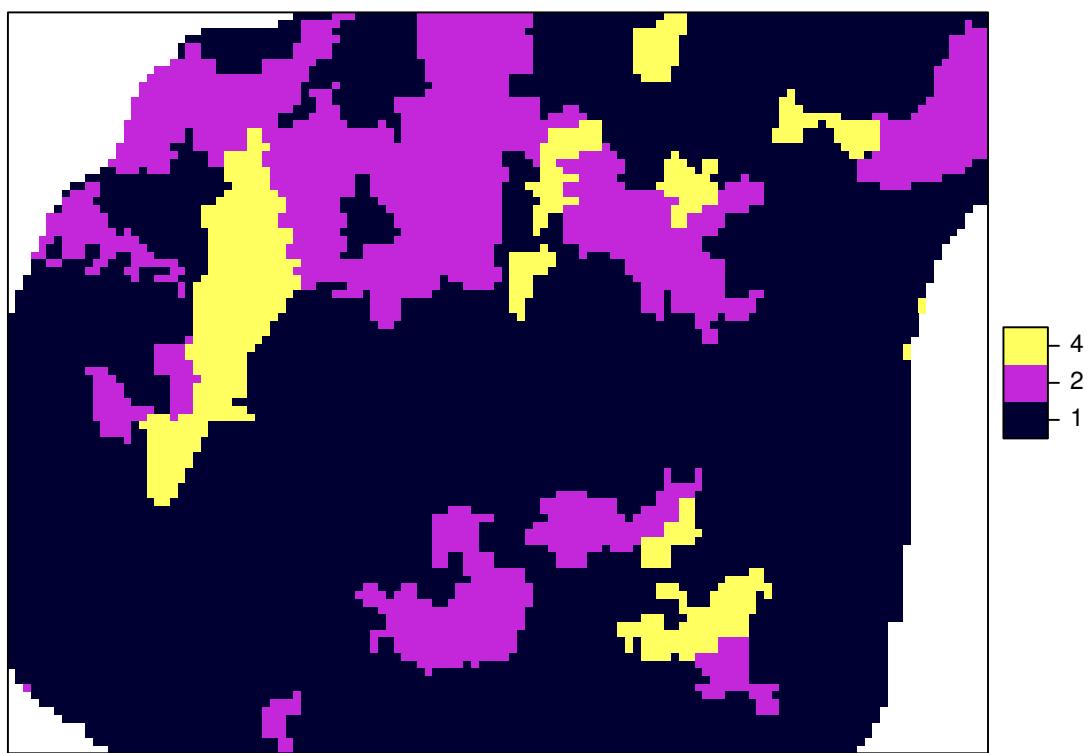
```
model_uk1 <- vgm(8, model = 'Sph', range = 3000, nugget = 1)
vario_fit_uk1 <- fit.variogram(vario_uk1, model=model_uk1)
vario_fit_uk1
```

```
##    model    psill    range
## 1    Nug 1.261541  0.000
```

```
## 2 Sph 8.472224 4742.211  
plot(vario_uk1, vario_fit_uk1)
```



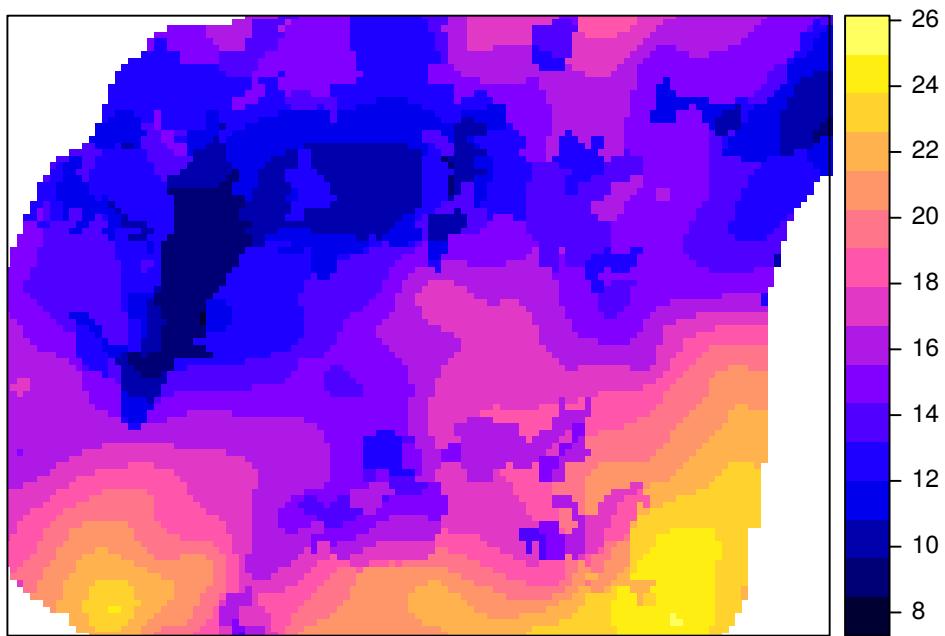
```
siatka$clc <- as.factor(siatka$clc)  
spplot(siatka, 'clc')
```



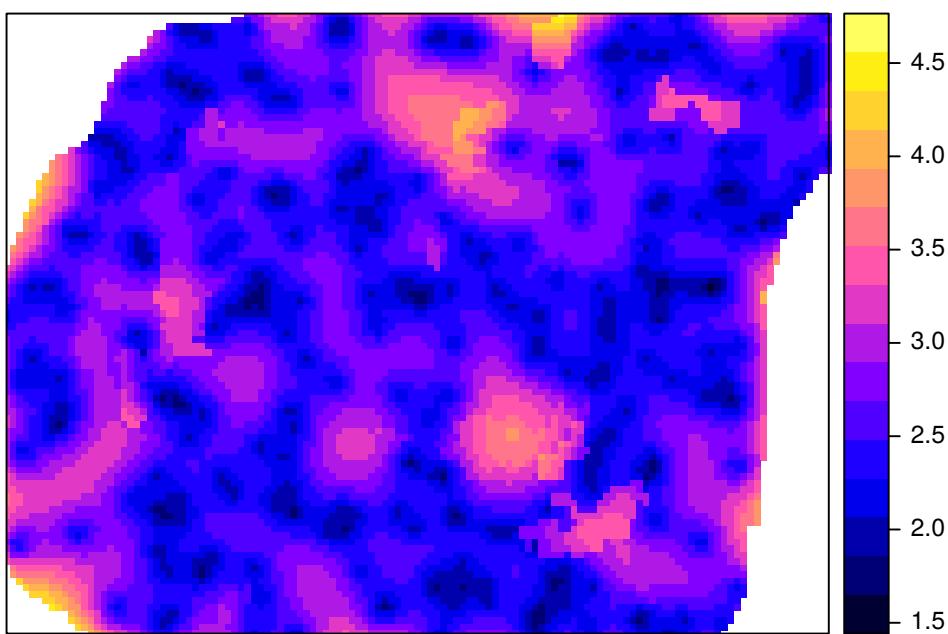
```
uk1 <- krige(temp~clc, locations = punkty, newdata=siatka, model=vario_fit_uk1)
```

```
## [using universal kriging]
spplot(uk1, 'var1.pred')
spplot(uk1, 'var1.var')
```

### Preidykcja KED



### Wariancja preidykcji KED



W kolejnym przykładzie zastosowane są już dwie zmienne uzupełniające - wartość wskaźnika wegetacji (ndvi) oraz wysokość nad poziomem morza (srtm).

```
vario_uk2 <- variogram(temp~ndvi+srtm, punkty)
vario_uk2
```

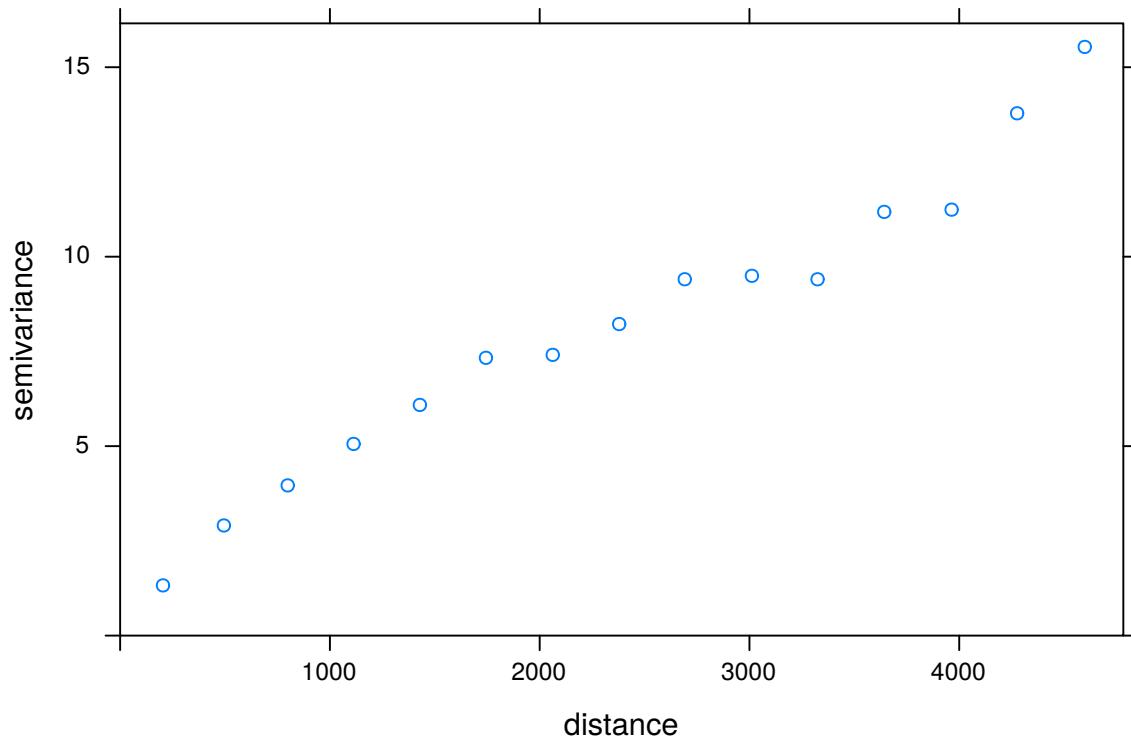
##	np	dist	gamma	dir.hor	dir.ver	id
## 1	127	204.2244	1.324959	0	0	var1
## 2	271	494.5572	2.906522	0	0	var1
## 3	522	798.7911	3.964307	0	0	var1
## 4	587	1112.7783	5.058684	0	0	var1
## 5	856	1428.7866	6.088010	0	0	var1

```

## 6   920 1743.7864 7.331287      0      0 var1
## 7  1068 2062.3041 7.409558      0      0 var1
## 8  1106 2378.9333 8.222238      0      0 var1
## 9  1184 2691.5206 9.404511      0      0 var1
## 10 1215 3011.7729 9.495409      0      0 var1
## 11 1362 3324.6705 9.403714      0      0 var1
## 12 1379 3642.5616 11.182666      0      0 var1
## 13 1405 3963.6776 11.241615      0      0 var1
## 14 1468 4276.0078 13.783975      0      0 var1
## 15 1482 4598.4144 15.536012      0      0 var1

```

```
plot(vario_uk2)
```



```

model <- vgm(8, model = 'Sph', range = 3000, nugget = 1)
vario_fit_uk2 <- fit.variogram(vario_uk2, model=model)
vario_fit_uk2

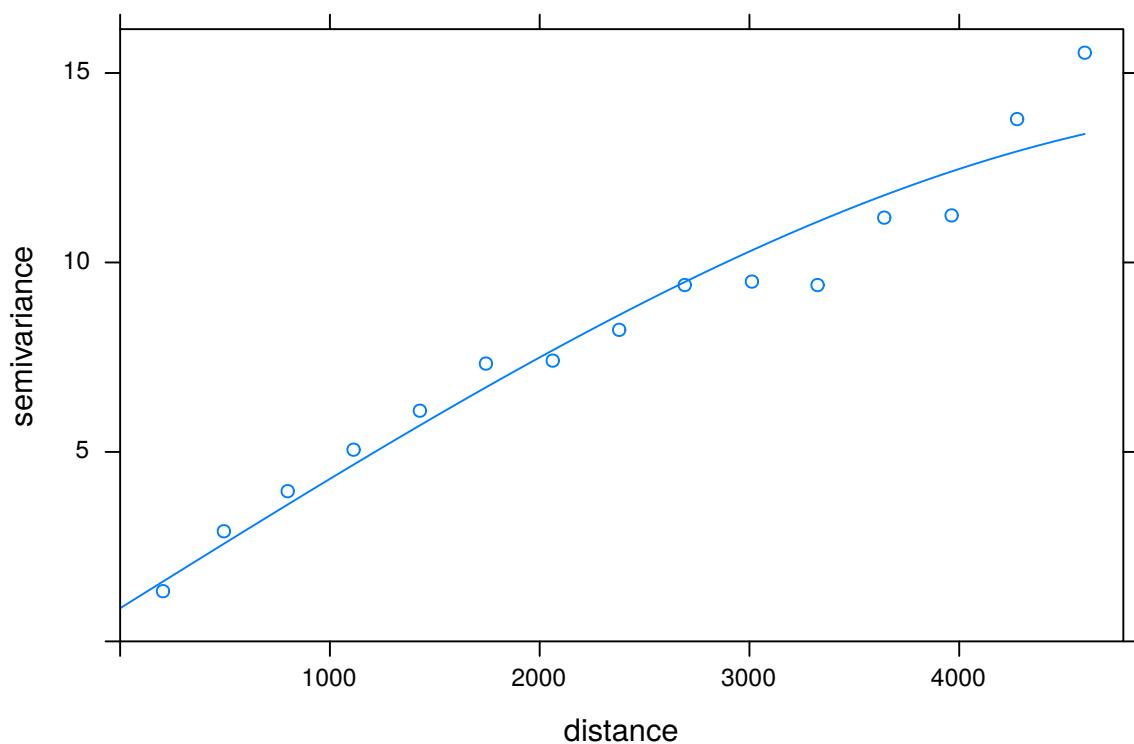
```

```

##    model      psill     range
## 1   Nug  0.8757491  0.000
## 2   Sph 13.3016445 5789.379

```

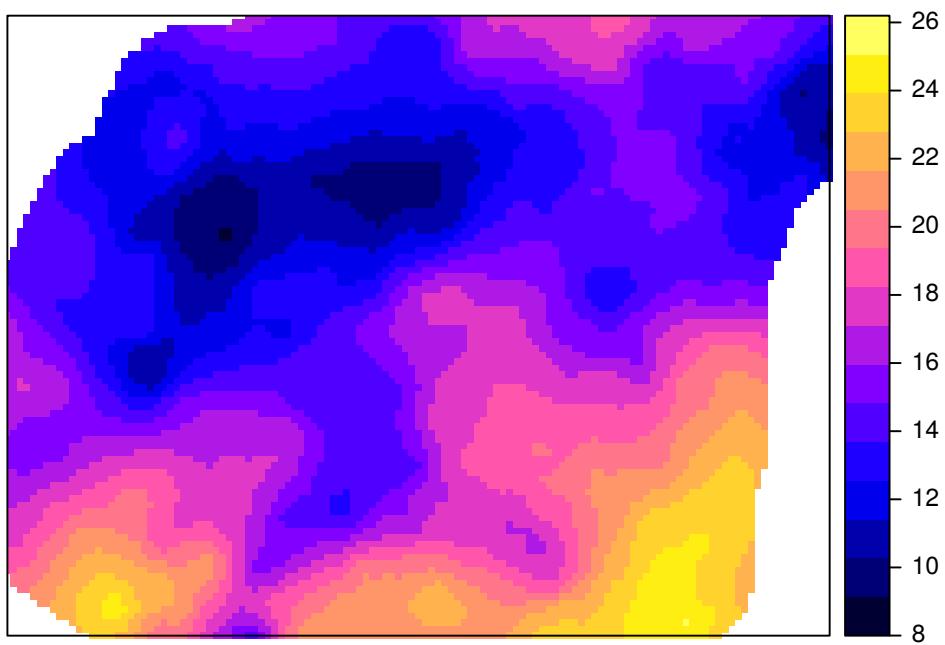
```
plot(vario_uk2, vario_fit_uk2)
```



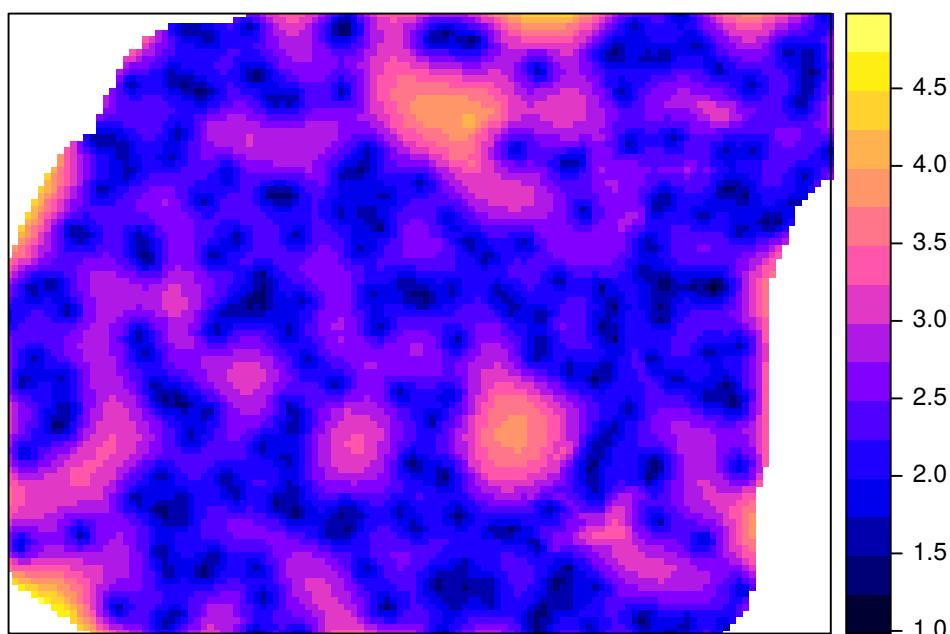
```
uk2 <- kriging(temp~ndvi+srtm, locations = punkty, newdata=siatka, model=vario_fit_uk2)
```

```
## [using universal kriging]
spplot(uk2, 'var1.pred')
spplot(uk2, 'var1.var')
```

### Predkcja KED



### Wariancja predkcji KED



# Rozdział 12

## Ocena jakości estymacji

```
library('gstat')
library('sp')
library('ggplot2')
library('caret')
library('geostatbook')
data(punkty)
data(siatka)
```

### 12.1 Statystyki jakości estymacji

#### 12.1.1 Statystyki jakości estymacji

W momencie, gdy trzeba określić jakość estymacji lub porównać wyniki pomiędzy estymacjami należy zastosować tzw. statystyki jakości estymacji. Do podstawowych statystyk ocen jakości estymacji należą:

- Średni błąd predykcji (MPE, ang. *mean prediction error*)
- Pierwiastek średniego błędu kwadratowego (RMSE, ang. *root square prediction error*)
- Rozkład błędu (np. 5. percentyl, mediana, 95. percentyl)

Idealna estymacja dawałaby brak błędu oraz współczynnik korelacji pomiędzy pomiarami (całą populacją) i szacunkiem równy 1. Wysokie, pojedyncze wartości błędu mogą świadczyć, np. o wystąpieniu wartości odstających.

#### 12.1.2 Średni błąd estymacji

Średni błąd estymacji (MPE) można wyliczyć korzystając z poniższego wzoru:

$$MPE = \frac{\sum_{i=1}^n (\hat{v}_i - v_i)}{n}$$

, lub używając funkcji `mean()` w R.

```
MPE <- mean(estymacja - obserwowane)
```

Optymalnie wartość średniego błędu estymacji powinna być jak najbliżej 0.

#### 12.1.3 Pierwiastek średniego błędu kwadratowego

Pierwiastek średniego błędu kwadratowego (RMSE) jest możliwy do wyliczenia poprzez wzór:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (v_i - \hat{v}_i)^2}{n}}$$

, jak i proste obliczenie w R.

```
RMSE <- sqrt(mean((obserwowane - estymacja)^2))
```

Optymalnie wartość pierwiastka średniego błędu kwadratowego powinna być jak najmniejsza.

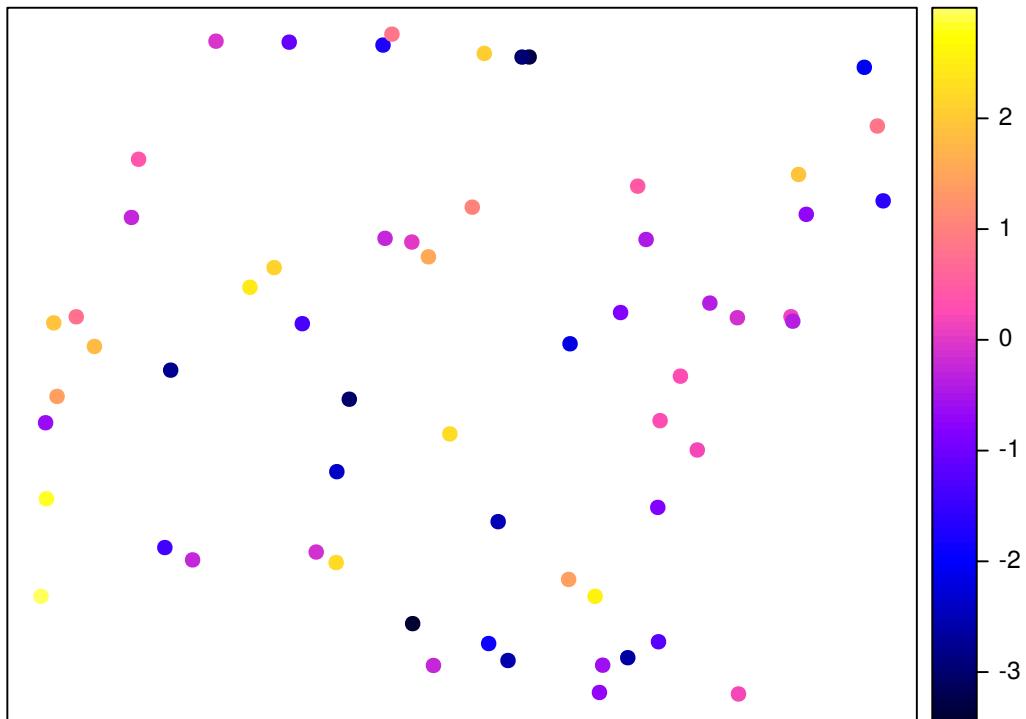
#### 12.1.4 Statystyki jakości estymacji | Mapa

Do oceny przestrzennej jakości predykcji można również zastosować mapę przedstawiającą błędy predykcji (błędy estymacji). Wyliczenie błędów estymacji odbywa się poprzez odjęcie od predykcji wartości obserwowanej.

```
blad_predykcji <- estymacja - obserwowane
```

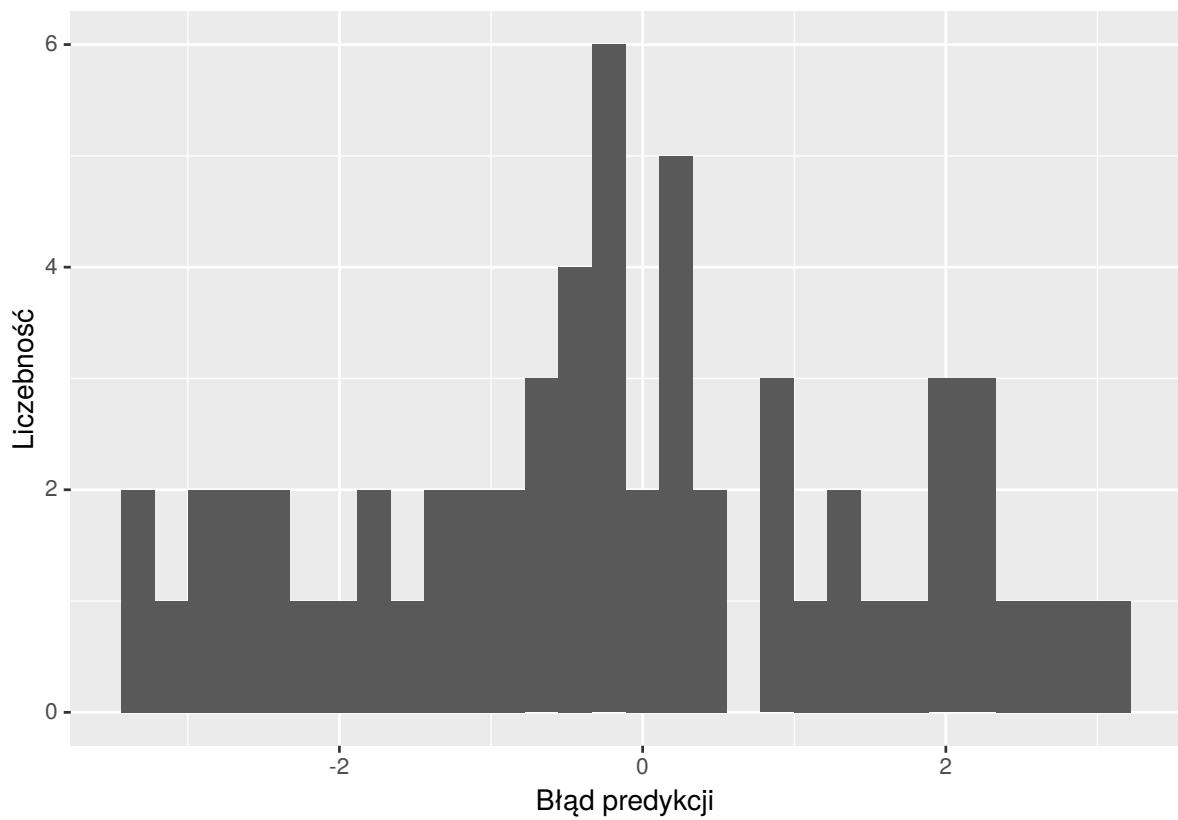
```
## [using simple kriging]
```

**Błąd predykcji**



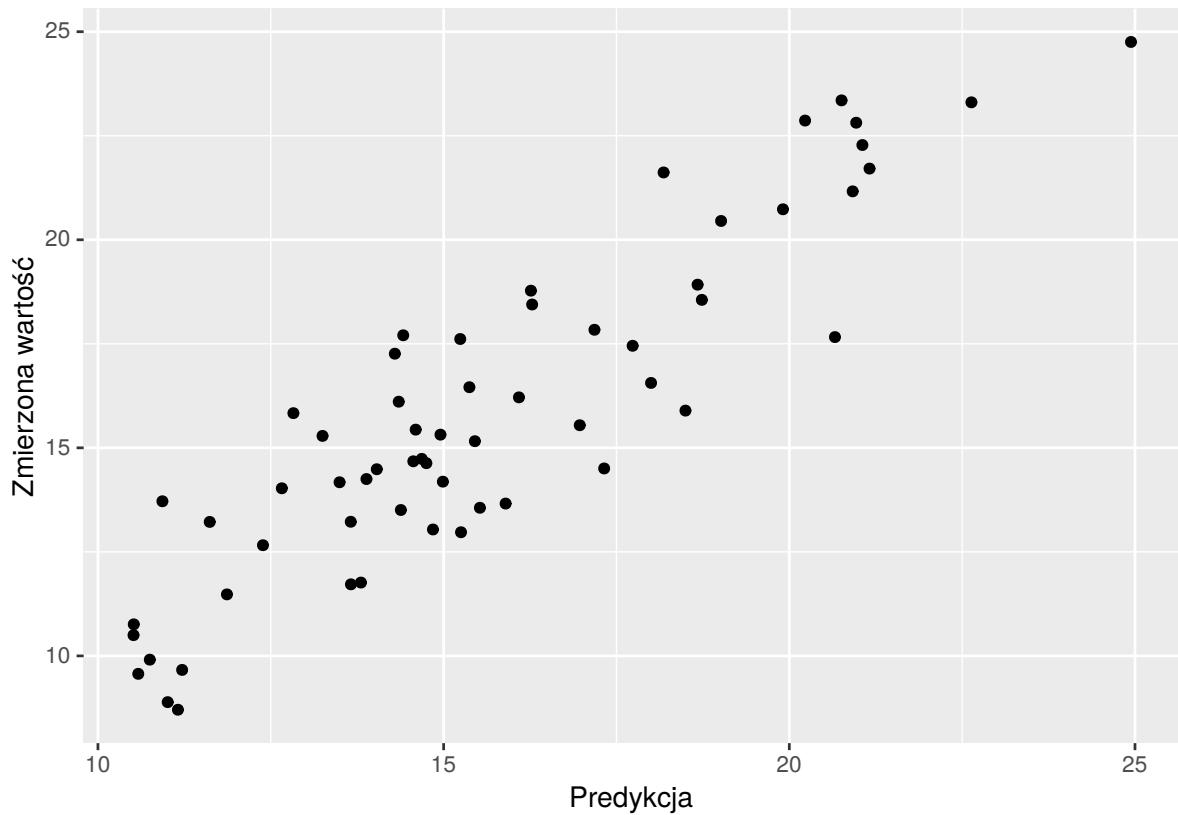
#### 12.1.5 Statystyki jakości estymacji | Histogram

Błąd predykcji można również przedstawić na wykresach, między innymi, na histogramie.



#### 12.1.6 Statystyki jakości estymacji | Wykres rozrzutu

Do porównania pomiędzy wartością estymowaną a obserwowaną może również posłużyć wykres rozrzutu.



## 12.2 Walidacja wyników estymacji

### 12.2.1 Walidacja wyników estymacji

Dokładne dopasowanie modelu do danych może w efekcie nie dawać najlepszych wyników. Szczególnie będzie to widoczne w przypadku modelowania, w którym dane obarczone są znacznym szumem (zawierają wyraźny błąd) lub też posiadają kilka wartości odstających. W efekcie ważne jest stosowanie metod pozwalających na wybór optymalnego modelu. Do takich metod należy, między innymi, walidacja podzbiorem (ang. *jackknifing*) oraz kroswalidacja (ang. *crossvalidation*).

### 12.2.2 Walidacja podzbiorem

Walidacja podzbiorem polega na podziale zbioru danych na dwa podzbiory - treningowy i testowy. Zbiór treningowy służy do stworzenia semiwariogramu empirycznego, budowania modelu oraz estymacji wartości. Następnie wynik estymacji porównywany jest z rzeczywistymi wartościami ze zbioru testowego. Zaletą tego podejścia jest stosowanie danych niezależnych od estymacji do oceny jakości modelu. Wadą natomiast jest konieczność posiadania (relatywnie) dużego zbioru danych.

Na poniższym przykładzie zbiór danych dzielony jest używając funkcji `createDataPartition()` z pakietu `caret`. Użycie tej funkcji powoduje stworzenie indeksu zawierającego numery wierszy dla zbioru treningowego. Ważną zaletą funkcji `createDataPartition()` jest to, iż w zbiorze treningowym i testowym zachowane są podobne rozkłady wartości. W przykładzie użyto argumentu `p=0.75`, który oznacza, że 75% danych będzie należało do zbioru treningowego, a 25% do zbioru testowego. Następnie korzystając ze stworzonego indeksu, budowane są dwa zbiory danych - treningowy (`train`) oraz testowy (`test`).

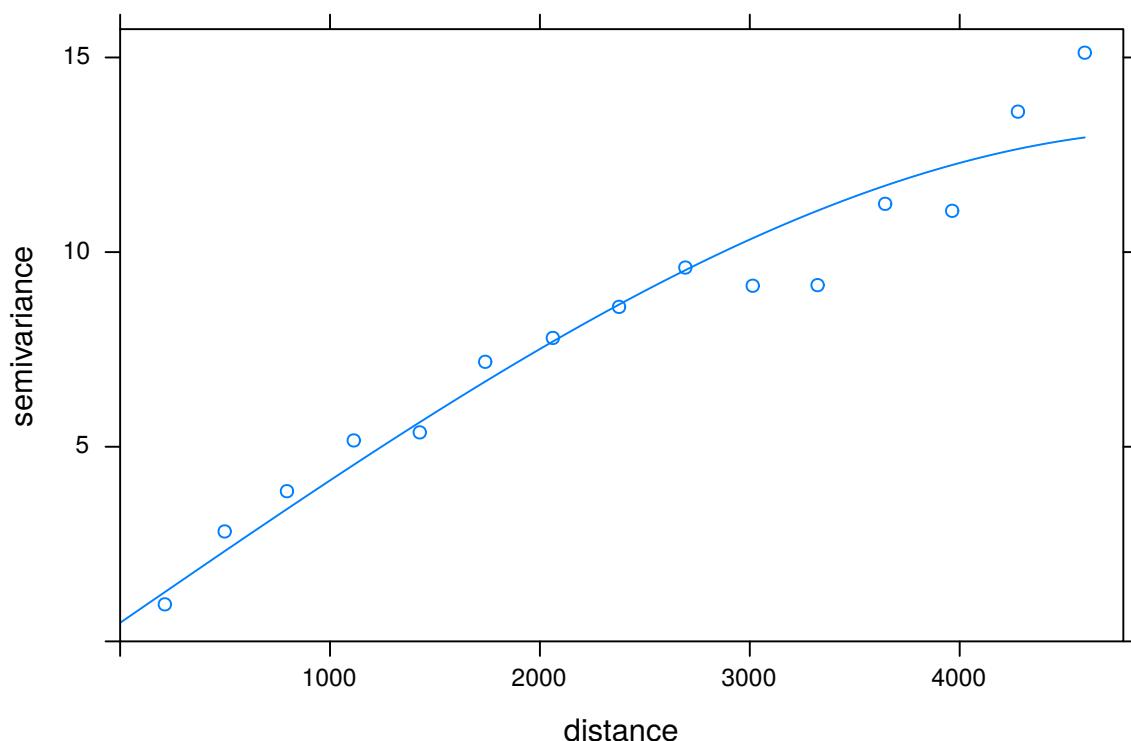
```
set.seed(124)
indeks <- as.vector(createDataPartition(punkty$temp, p=0.75, list=FALSE))
indeks
```

```
## [1] 1 2 4 7 8 9 12 13 14 15 16 17 18 19 22 23 25 26 27 29
## [21] 30 32 34 35 36 37 38 40 41 42 43 45 49 51 52 53 54 55 56 58
## [41] 59 60 61 62 64 66 67 69 70 73 75 76 78 79 81 82 83 84 85 86
## [61] 87 88 89 90 91 94 96 97 98 100 101 102 103 106 107 108 109 110 111 112
## [81] 113 114 115 117 118 120 122 124 125 127 128 129 130 132 133 134 135 137 138 139
## [101] 140 141 142 143 144 146 148 149 150 153 154 155 156 157 158 159 160 161 162 163
## [121] 167 168 169 171 173 174 175 176 177 180 181 183 184 185 186 187 188 189 190 191
## [141] 193 194 195 196 199 200 201 202 204 205 208 209 210 212 213 214 215 216 217 218
## [161] 219 220 221 223 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
## [181] 241 242 243 244 245 246 247 248 249 250
```

```
train <- punkty[indeks, ]
test <- punkty[-indeks, ]
```

Dalszym krokiem jest stworzenie semiwariogramu empirycznego oraz jego modelowanie w oparciu o zbiór treningowy.

```
vario <- variogram(temp~1, data=train)
model <- vgm(10, model = 'Sph', range = 4000, nugget = 0.5)
fitted <- fit.variogram(vario, model)
plot(vario, model=fitted)
```



Do porównania wyników estymacji w stosunku do zbioru testowego posłuży funkcja `krige()`. Wcześniej wymagała ona podania wzoru, zbioru punktowego, siatki oraz modelu. W tym przypadku jednak chcemy porównać wynik estymacji i testowy zbiór punktowy. Dlatego też, zamiast obiektu siatki definiujemy obiekt zawierający zbiór testowy (`test`).

```
test_sk <- krige(temp~1, train, test, model=fitted, beta=15.324)

## [using simple kriging]
summary(test_sk)

## Object of class SpatialPointsDataFrame
## Coordinates:
##      min      max
## x 745731 756567.0
## y 712629 721118.7
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 60
## Data attributes:
##      var1.pred      var1.var
## Min.   :10.51   Min.   :1.002
## 1st Qu.:13.62   1st Qu.:1.654
## Median :14.97   Median :2.084
## Mean   :15.66   Mean   :2.187
## 3rd Qu.:18.05   3rd Qu.:2.608
## Max.   :24.94   Max.   :5.602
```

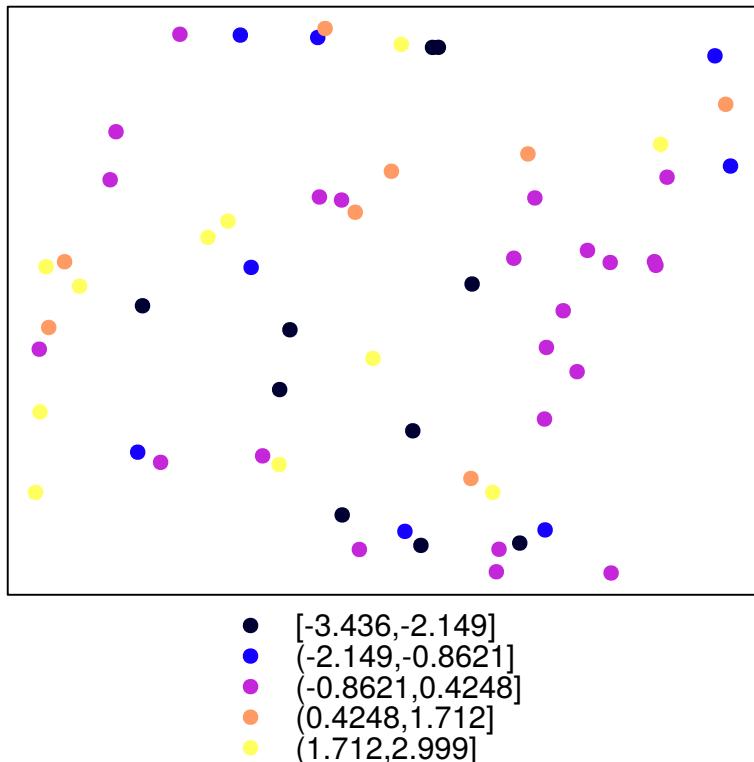
Uzyskane w ten sposób wyniki możemy określić używając statystyk jakości estymacji lub też wykresów.

```
blad_predykcji_sk <- test_sk$var1.pred - test$temp
summary(blad_predykcji_sk)
```

```
##      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
## -3.4360 -1.3860 -0.2442 -0.1954  0.9098  2.9990
MPE <- mean(test_sk$var1.pred - test$temp)
MPE
```

```
## [1] -0.1953769
RMSE <- sqrt(mean((test$temp-test_sk$var1.pred)^2))
RMSE
```

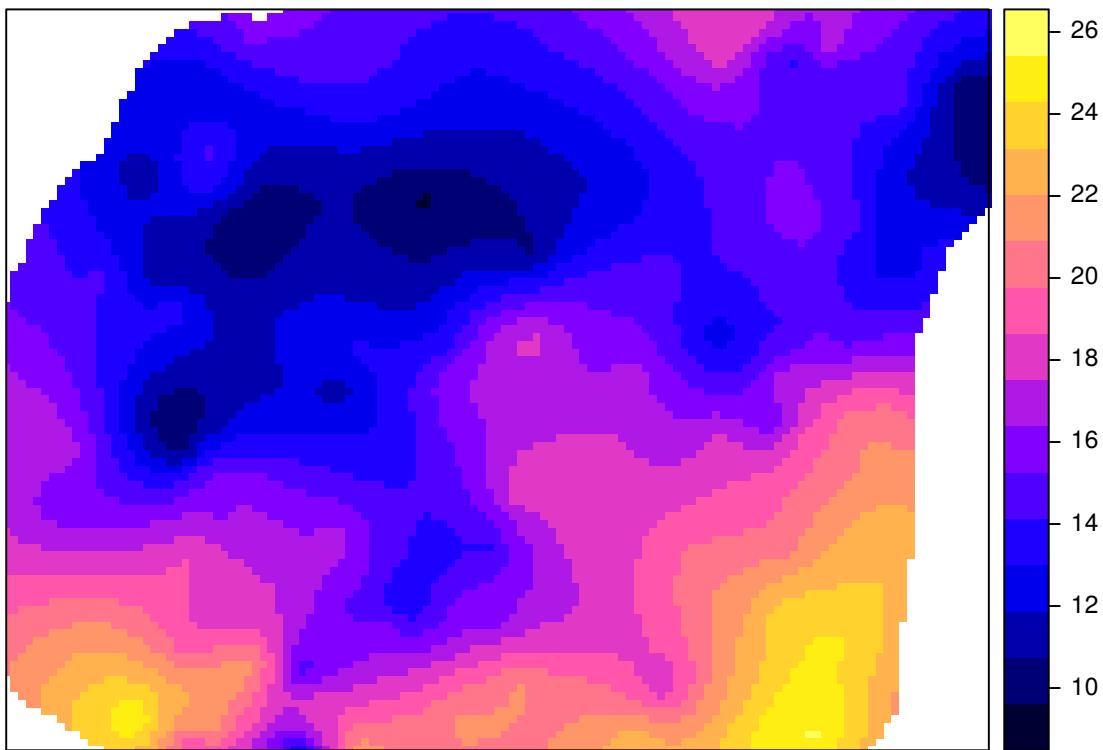
```
## [1] 1.692512
test_sk$blad_predykcji_sk <- blad_predykcji_sk
spplot(test_sk, 'blad_predykcji_sk')
```



W sytuacji, gdy uzyskany model jest wystarczająco dobry, możemy również uzyskać estymację dla całego obszaru z użyciem funkcji `krige()`, tym razem jednak podając obiekt siatki.

```
test_sk <- krige(temp~1, train, siatka, model=fitted, beta=15.324)
```

```
## [using simple kriging]
spplot(test_sk, 'var1.pred')
```



### 12.2.3 Kroswalidacja

W przypadku kroswalidacji te same dane wykorzystywane są do budowy modelu, estymacji, a następnie do oceny prognozy. Procedura kroswalidacji LOO (ang. *leave-one-out cross-validation*) składa się z poniższych kroków:

1. Zbudowanie matematycznego modelu z dostępnych obserwacji
2. Dla każdej znanej obserwacji następuje:
  - Usunięcie jej ze zbioru danych
  - Użycie modelu do wykonania predykcji w miejscu tej obserwacji
  - Wyliczenie reszty (ang. *residual*), czyli różnicy pomiędzy znaną wartością a estymacją
3. Podsumowanie otrzymanych wyników

W pakiecie **gstat**, kroswalidacja LOO jest dostępna w funkcjach **krige.cv()** oraz **gstat.cv()**. Działają one bardzo podobnie jak funkcje **krige()** oraz **gstat()**, jednak w przeciwieństwie do nich nie wymagają podania obiektu siatki.

```

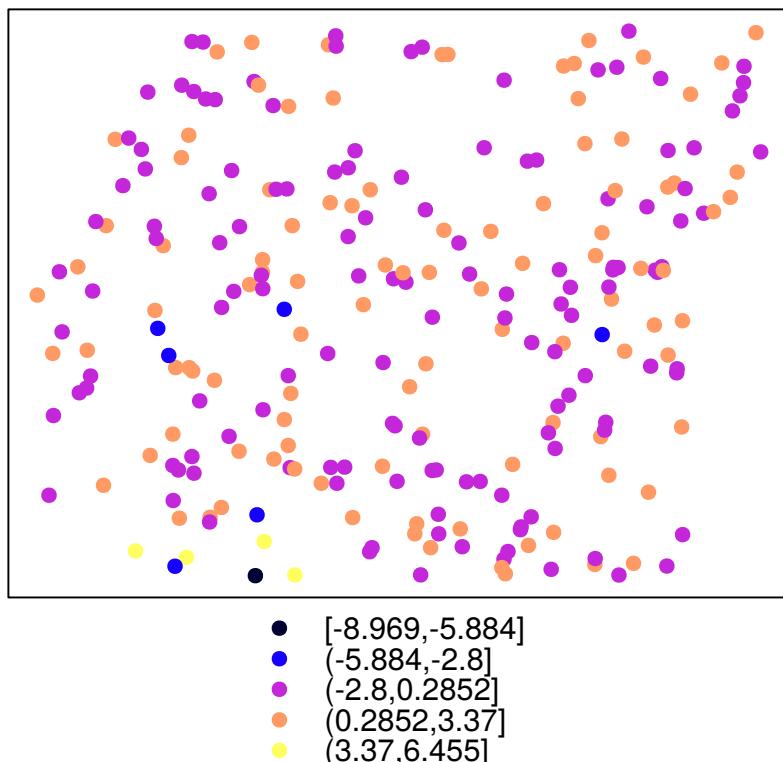
vario <- variogram(temp~1, data=punkty)
model <- vgm(10, model = 'Sph', range = 4000, nugget = 0.5)
fitted <- fit.variogram(vario, model)

cv_sk <- krige.cv(temp~1, punkty, model=fitted, beta=15.324, verbose=FALSE)
summary(cv_sk)

## Object of class SpatialPointsDataFrame
## Coordinates:
##       min     max
## x 745546.9 756937.4
## y 712618.8 721192.6
## Is projected: NA
## proj4string : [NA]
## Number of points: 250

```

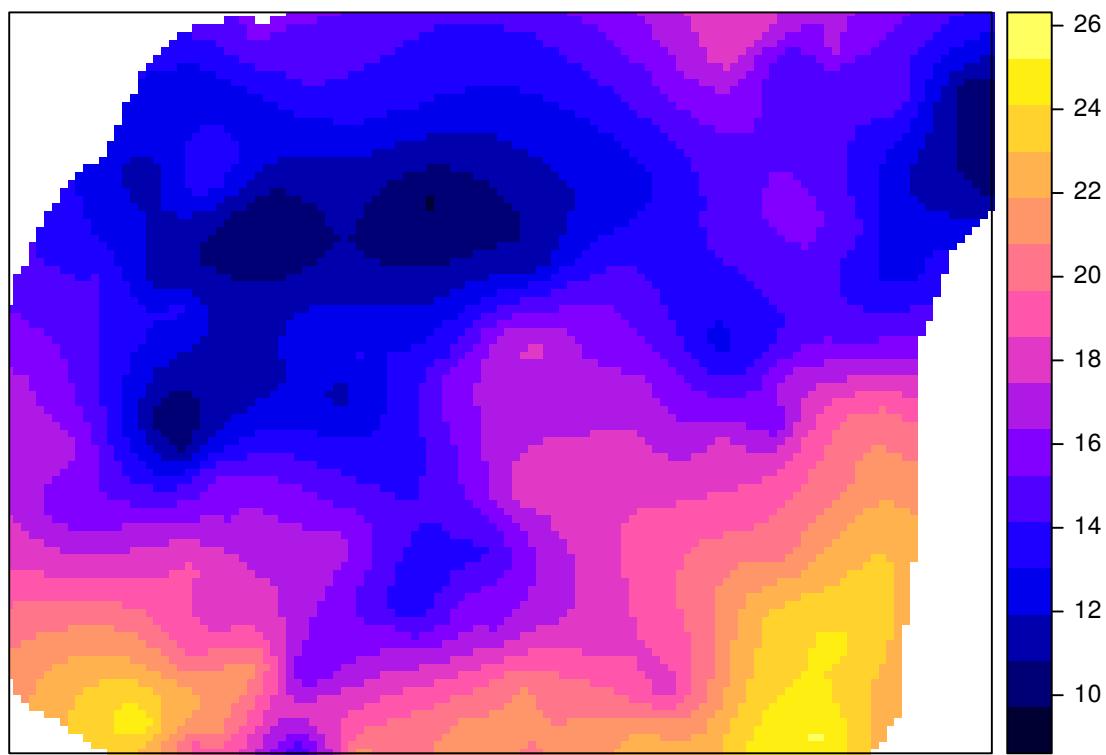
```
## Data attributes:
##   var1.pred      var1.var     observed      residual
##   Min. : 9.786  Min. :1.219  Min. : 8.706  Min. :-8.96908
## 1st Qu.:13.333 1st Qu.:1.806 1st Qu.:13.284 1st Qu.:-0.95234
## Median :15.298 Median :2.093 Median :15.309 Median : 0.04980
## Mean   :15.936 Mean  :2.220 Mean  :15.950 Mean  : 0.01423
## 3rd Qu.:18.062 3rd Qu.:2.477 3rd Qu.:18.273 3rd Qu.: 0.85669
## Max.  :24.994  Max. :5.320  Max. :26.139 Max. : 6.45468
##   zscore        fold
##   Min. :-4.930132  Min. : 1.00
## 1st Qu.:-0.647920 1st Qu.: 63.25
## Median : 0.036433 Median :125.50
## Mean   : 0.008082 Mean  :125.50
## 3rd Qu.: 0.598859 3rd Qu.:187.75
## Max.  : 3.330852 Max. :250.00
spplot(cv_skk, 'residual')
```



Podobnie jak w walidacji podzbiorem, gdy uzyskany model jest wystarczająco dobry, estymację dla całego obszaru uzyskuje się z użyciem funkcji `krige()`.

```
cv_skk <- krige(temp~1, train, siatka, model=fitted, beta=15.324)
```

```
## [using simple kriging]
spplot(cv_skk, 'var1.pred')
```





# Rozdział 13

## Symulacje

```
library('gstat')
library('sp')
library('ggplot2')
library('raster')
library('geostatbook')
data(punkty)
data(siatka)
```

### 13.1 Symulacje geostatystyczne

#### 13.1.1 Symulacje geostatystyczne

Kriging daje optymalne predykcje, czyli wyznacza najbardziej potencjalnie możliwą wartość dla wybranej lokalizacji. Dodatkowo, efektem krigingu jest wygładzony obraz. W konsekwencji wyniki estymacji różnią się od danych pomiarowych. Uzyskiwana jest tylko (aż?) predykcja, a prawdziwa wartość jest niepewna... Korzystając z symulacji geostatystycznych nie tworzymy predykcji, ale generujemy równie prawdopodobne możliwości poprzez symulację z rozkładu prawdopodobieństwa (wykorzystując generator liczb losowych).

#### 13.1.2 Symulacje geostatystyczne

Właściwości symulacji geostatystycznych:

- Efekt symulacji ma bardziej realistyczny przestrzenny wzór (ang. *pattern*) niż kriging, którego efektem jest wygładzona reprezentacja rzeczywistości
- Każda z symulowanych map jest równie prawdopodobna
- Symulacje pozwalają na przedstawianie niepewności interpolacji
- Jednocześnie - kriging jest znacznie lepszy, gdy naszym celem jest jak najdokładniejsza predykcja

### 13.2 Typy symulacji

#### 13.2.1 Typy symulacji

Istnieją dwa typy symulacji geostatystycznych:

- Symulacje bezwarunkowe (ang. *Unconditional Simulations*) - wykorzystujące semiwariogram, żeby włączyć informację przestrzenną, ale wartości ze zmierzonych punktów nie są w niej wykorzystywane

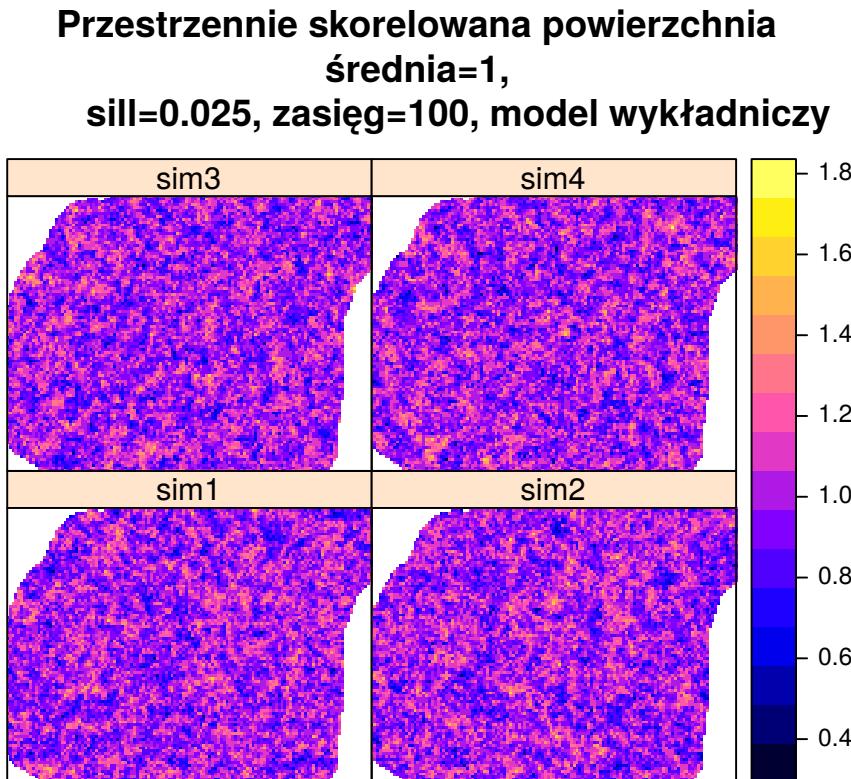
- Symulacje warunkowe (ang. *Conditional Simulations*) - opiera się ona o średnią wartość, strukturę kowariancji oraz obserwowane wartości

### 13.3 Symulacje bezwarunkowe

Symulacje bezwarunkowe w pakiecie `gstat` tworzy się z wykorzystaniem funkcji `krige()`. Podobnie jak w przypadku estymacji geostatystycznych, należy tutaj podać wzór, model, siatkę, średnią globalną (`beta`), oraz liczbę sąsiednich punktów używanych do symulacji (w przykładzie poniżej `nmax=30`). Należy wprowadzić również informacje, że nie korzystamy z danych punktowych (`locations=NULL`) oraz że chcemy stworzyć dane sztuczne (`dummy=TRUE`). Ostatni argument (`nsim=4`) informuje o liczbie symulacji do przeprowadzenia.

```
sym_bezw1 <- krige(formula=z~1, model=vgm(psill=0.025, model='Exp', range=100), newdata=siatka, beta=0.025, nmax=30, locations=NULL, dummy=TRUE, nsim=4)
```

```
## [using unconditional Gaussian simulation]
spplot(sym_bezw1, main='Przestrzennie skorelowana powierzchnia \nśrednia=1,
       sill=0.025, zasięg=100, model wykładniczy')
```

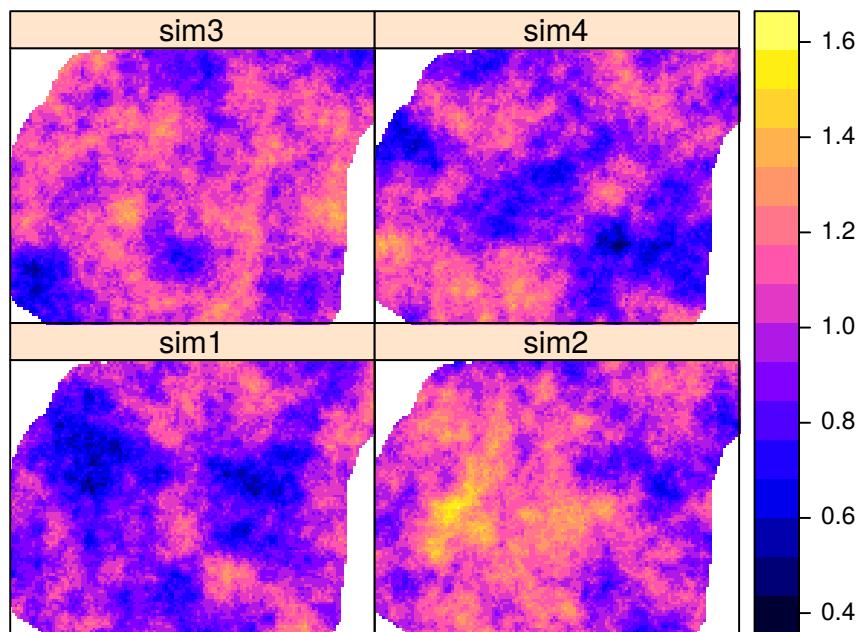


```
sym_bezw2 <- krige(formula=z~1, , model=vgm(psill=0.025, model='Exp', range=1500), newdata=siatka, beta=0.025, nmax=30, locations=NULL, dummy=TRUE, nsim=4)

## [using unconditional Gaussian simulation]
```

```
spplot(sym_bezw2, main='Przestrzennie skorelowana powierzchnia \nśrednia=1,
       sill=0.025, zasięg=1500, model wykładniczy')
```

### Przestrzennie skorelowana powierzchnia średnia=1, sill=0.025, zasięg=1500, model wykładniczy



## 13.4 Symulacje warunkowe

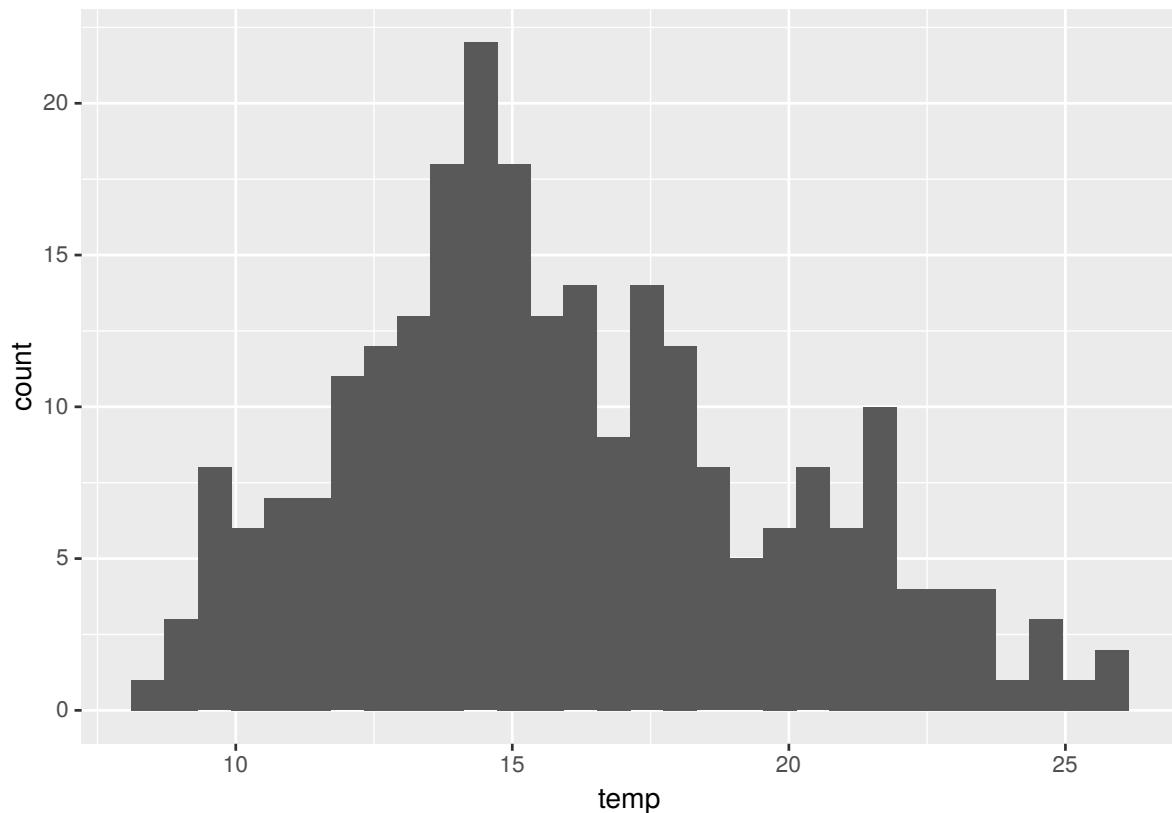
### 13.4.1 Sekwencyjna symulacja gaussowska (ang. *Sequential Gaussian simulation*)

Jednym z podstawowych typów symulacji warunkowych jest sekwencyjna symulacja gaussowska. Polega ona na:

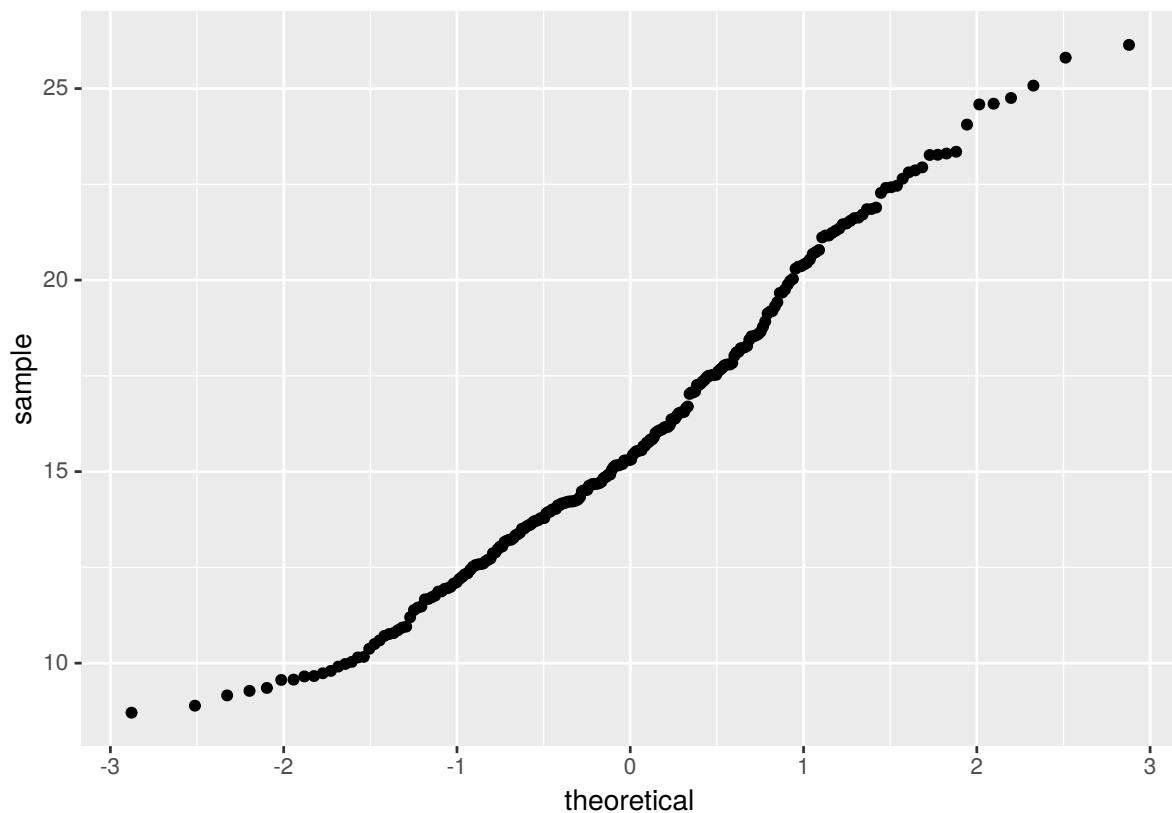
1. Wybraniu lokalizacji nie posiadającej zmierzonej wartości badanej zmiennej
2. Krigingu wartości tej lokalizacji korzystając z dostępnych danych, co pozwala na uzyskanie rozkładu prawdopodobieństwa badanej zmiennej
3. Wylosowaniu wartości z rozkładu prawdopodobieństwa za pomocą generatora liczba losowych i przypisaniu tej wartości do lokalizacji
4. Dodaniu symulowanej wartości do zbioru danych i przejściu do kolejnej lokalizacji
5. Powtórzeniu poprzednich kroków, aż do momentu gdy nie pozostanie już żadna nieokreślona lokalizacja

Sekwencyjna symulacja gaussowska wymaga zmiennej posiadającej wartości o rozkładzie zbliżonym do normalnego. Można to sprawdzić poprzez wizualizacje danych (histogram, wykres kwantyl-kwantyl) lub też test statystyczny (test Shapiro-Wilka). Zmienna `temp` nie ma rozkładu zbliżonego do normalnego.

```
ggplot(punkty@data, aes(temp)) + geom_histogram()
```



```
ggplot(punkty@data, aes(sample=temp)) + stat_qq()
```



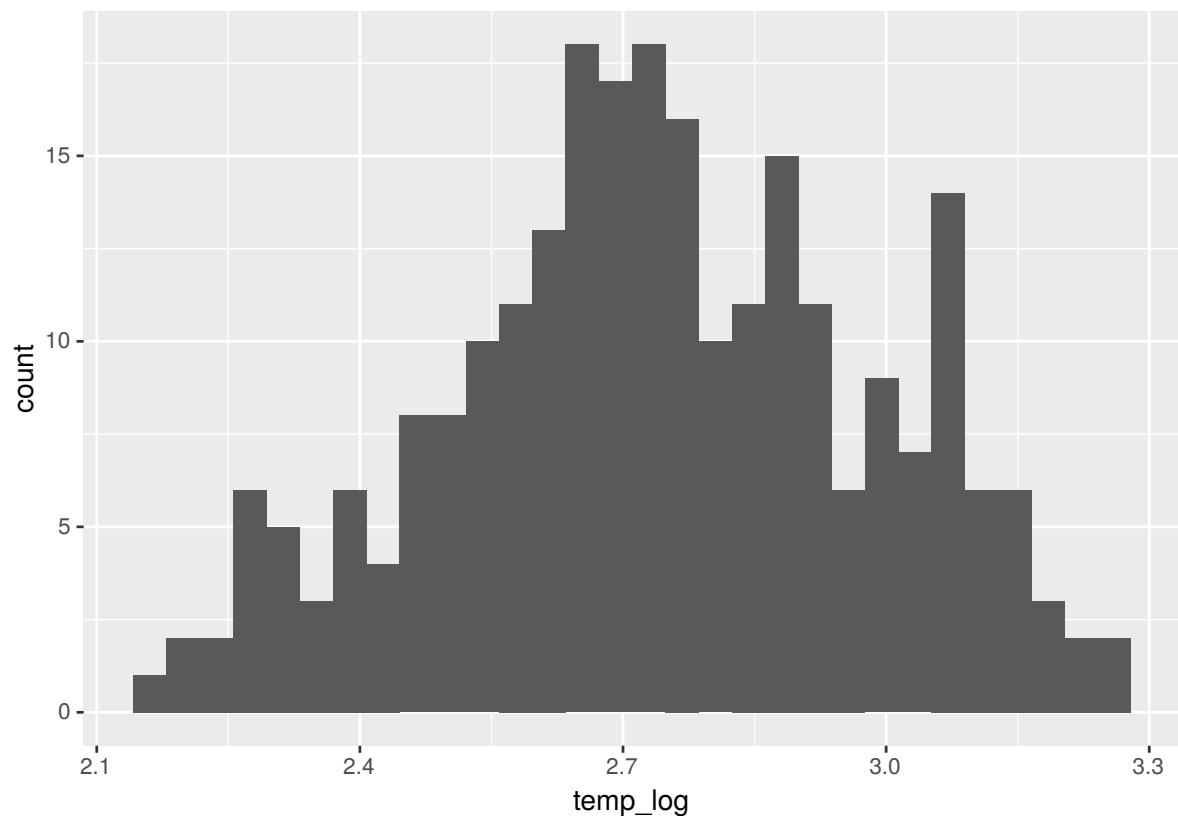
```
shapiro.test(punkty$temp)
```

```
##
```

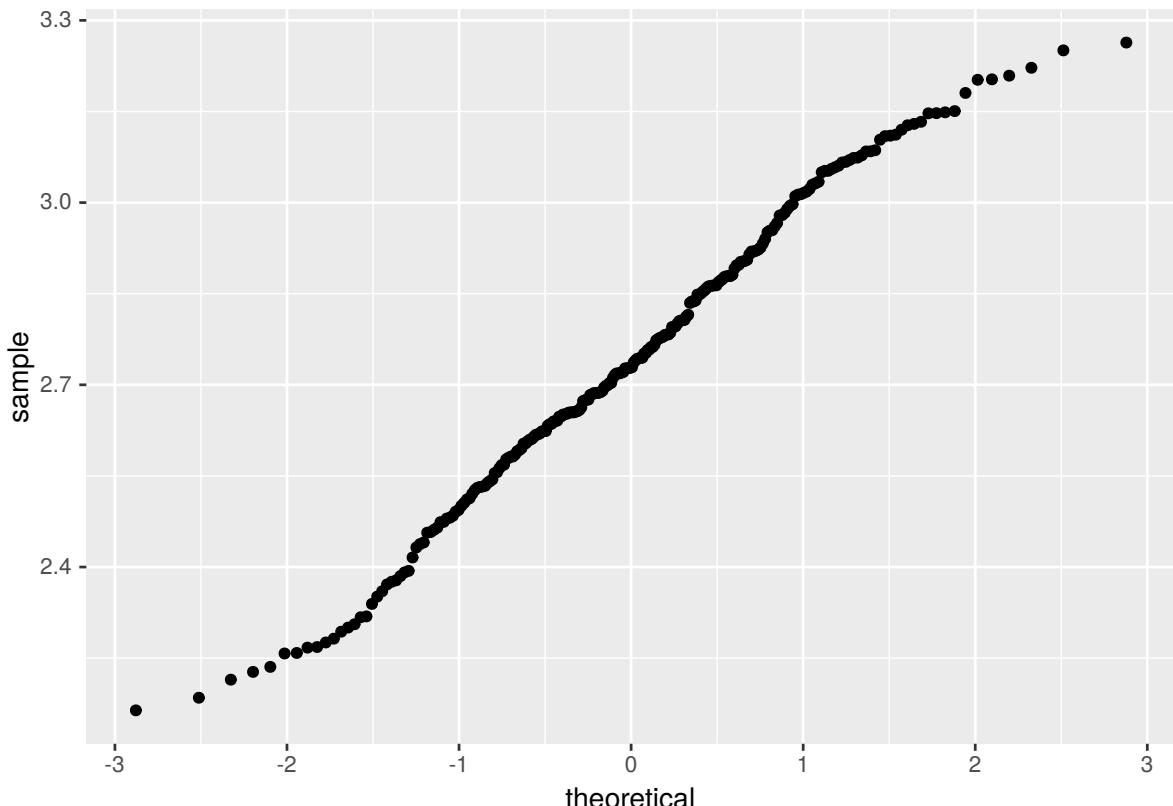
```
## Shapiro-Wilk normality test
##
## data: punkty$temp
## W = 0.97683, p-value = 0.0004194
```

Na potrzeby symulacji zmienna `temp` została zlogarytmizowana.

```
punkty$temp_log <- log(punkty$temp)
ggplot(punkty@data, aes(temp_log)) + geom_histogram()
```



```
ggplot(punkty@data, aes(sample=temp_log)) + stat_qq()
```

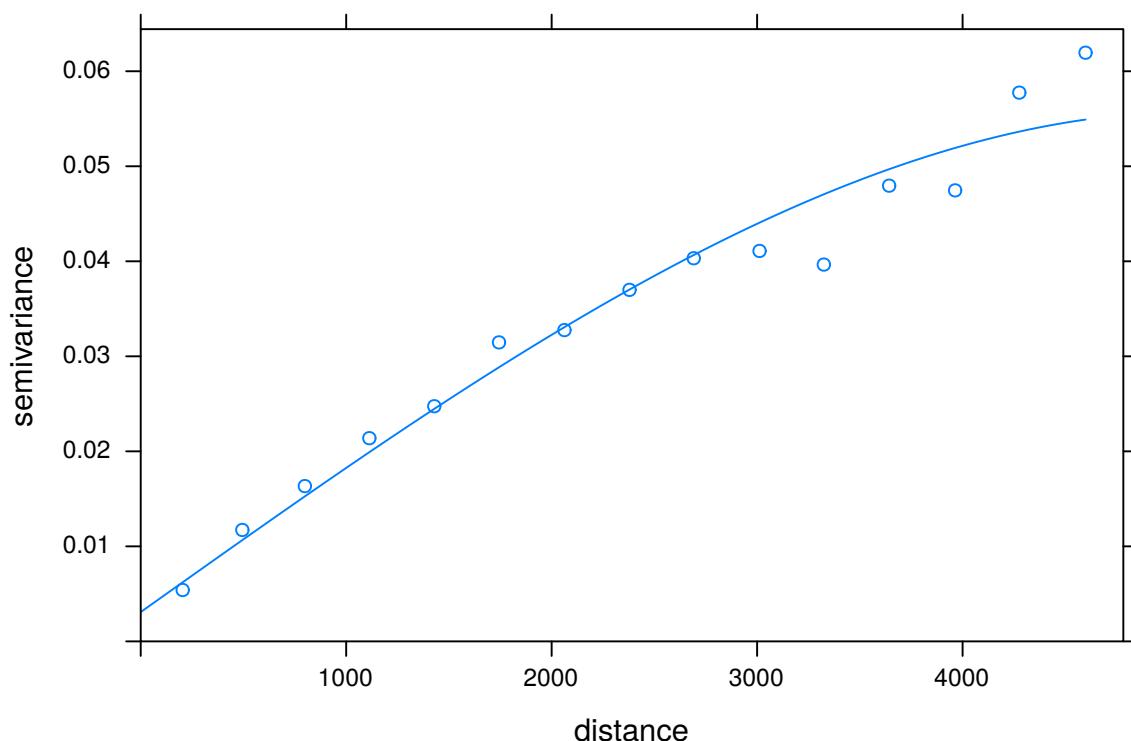


```
shapiro.test(punkty$temp_log)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: punkty$temp_log  
## W = 0.98907, p-value = 0.05568
```

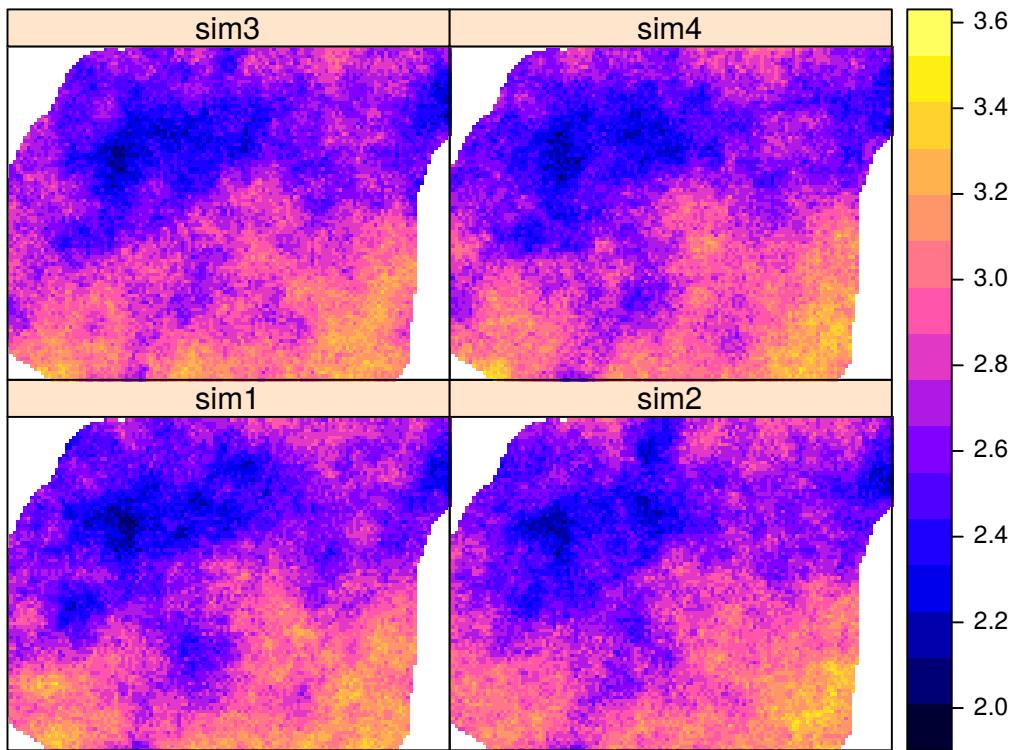
Dalsze etapy przypominają przeprowadzenie estymacji statystycznej, jedynym wyjątkiem jest dodanie argumentu mówiącego o liczbie symulacji do przeprowadzenia (`nsim` w funkcji `krige()`).

```
vario <- variogram(temp_log~1, punkty)  
model <- vgm(0.05, model = 'Sph', range = 4500, nugget=0.005)  
fitted <- fit.variogram(vario, model)  
plot(vario, model=fitted)
```



```
sym_ok <- kriging(temp_log~1, punkty, siatka, model=fitted, nmax=30, nsim=4)
```

```
## drawing 4 GLS realisations of beta...
## [using conditional Gaussian simulation]
spplot(sym_ok)
```



Wyniki symulacji można przetworzyć do pierwotnej jednostki z użyciem funkcji wykładniczej (`exp`).

```
summary(sym_ok)
```

```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min     max
## x 745586.7 756926.7
## y 712661.2 721211.2
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 10993
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## s1      745586.7      90      127
## s2      712661.2      90      96
## Data attributes:
##       sim1        sim2        sim3        sim4
## Min. :2.008    Min. :2.067    Min. :2.008    Min. :2.014
## 1st Qu.:2.548  1st Qu.:2.563  1st Qu.:2.567  1st Qu.:2.548
## Median :2.710  Median :2.730  Median :2.730  Median :2.715
## Mean   :2.729  Mean   :2.740  Mean   :2.734  Mean   :2.730
## 3rd Qu.:2.915  3rd Qu.:2.919  3rd Qu.:2.894  3rd Qu.:2.912
## Max.   :3.425  Max.   :3.506  Max.   :3.361  Max.   :3.525
sym_ok@data <- as.data.frame(apply(sym_ok@data, 2, exp))
summary(sym_ok)
```

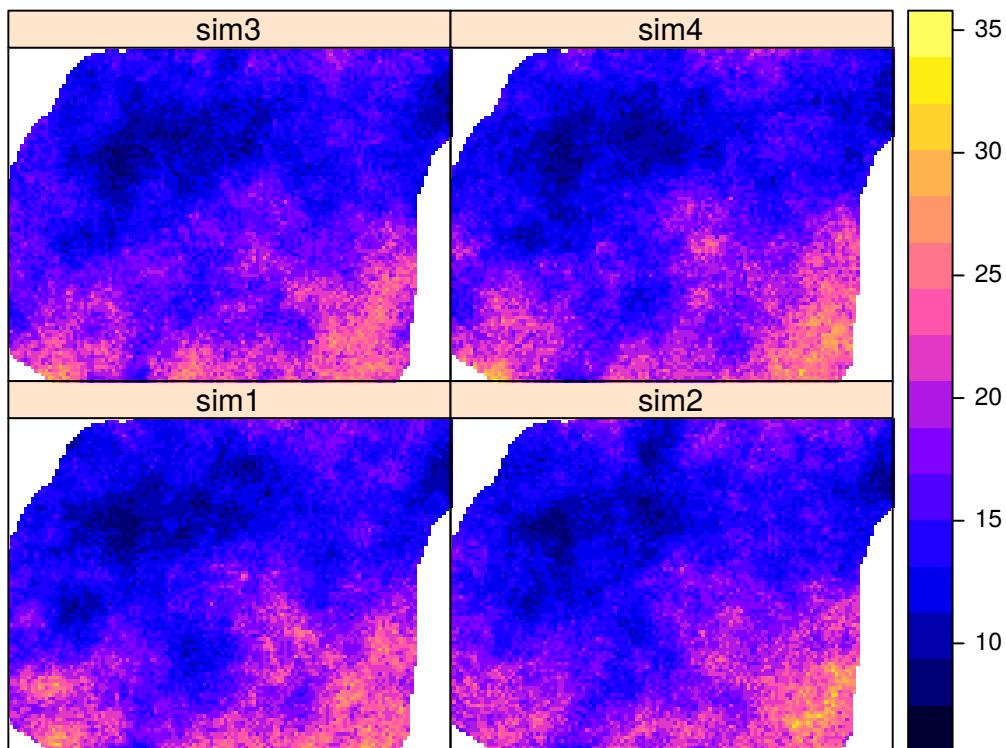
```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##      min     max
## x 745586.7 756926.7
## y 712661.2 721211.2
```

```

## x 745586.7 756926.7
## y 712661.2 721211.2
## Is projected: TRUE
## proj4string :
## [+init=epsg:2180 +proj=tmerc +lat_0=0 +lon_0=19 +k=0.9993 +x_0=500000
## +y_0=-5300000 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs]
## Number of points: 10993
## Grid attributes:
##   cellcentre.offset cellsize cells.dim
## s1      745586.7      90      127
## s2      712661.2      90      96
## Data attributes:
##   sim1          sim2          sim3          sim4
## Min. : 7.445  Min. : 7.902  Min. : 7.45  Min. : 7.493
## 1st Qu.:12.787 1st Qu.:12.973 1st Qu.:13.02 1st Qu.:12.785
## Median :15.035 Median :15.329 Median :15.33  Median :15.107
## Mean   :15.814 Mean   :15.961 Mean   :15.83  Mean   :15.819
## 3rd Qu.:18.443 3rd Qu.:18.520 3rd Qu.:18.07 3rd Qu.:18.392
## Max.   :30.714  Max.   :33.311  Max.   :28.82  Max.   :33.940

spplot(sym_ok)

```



Symulacje geostatystyczne pozwalają również na przedstawianie niepewności interpolacji. W tym wypadku należy wykonać znacznie więcej powtórzeń (np. `nsim=100`). Uzyskane wyniki należy przeliczyć do oryginalnej jednostki, a następnie wyliczyć odchylenie standardowe ich wartości (używając funkcji `stack()` i `calc()` z pakietu `raster`).

```
sym_sk <- krige(temp_log~1, punkty, siatka, model=fitted, beta=2.7, nsim=100, nmax=30)
```

```

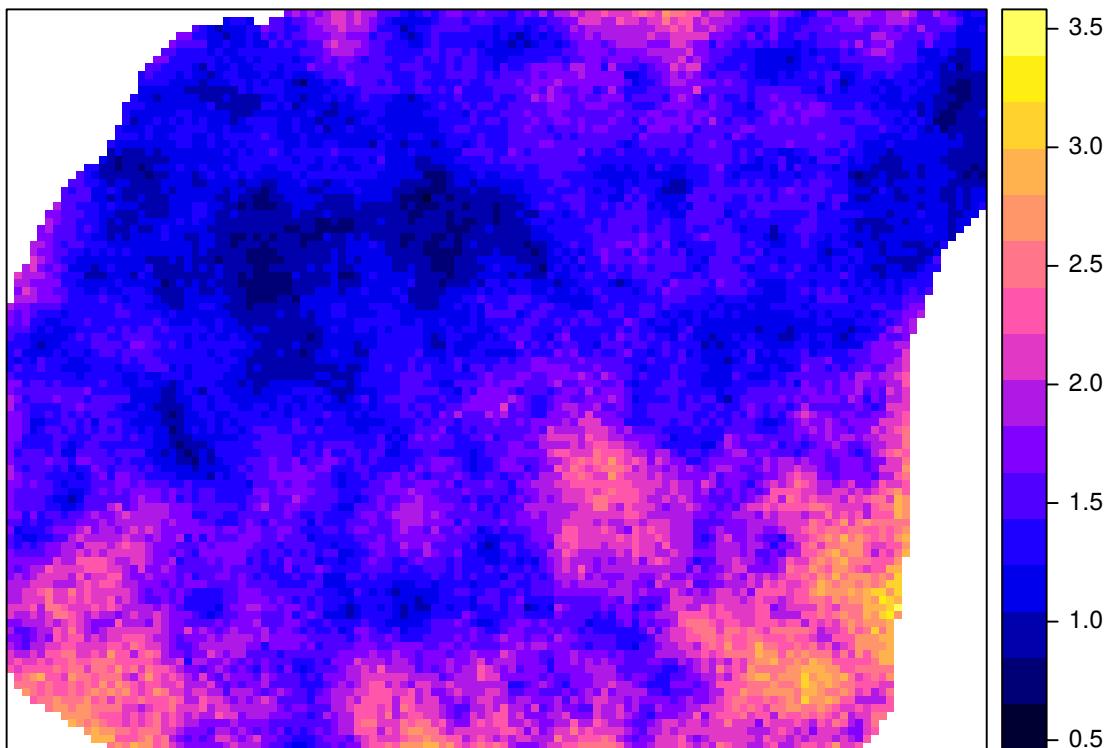
## [using conditional Gaussian simulation]
sym_sk@data <- as.data.frame(apply(sym_sk@data, 2, exp))

```

```
sym_sk <- stack(sym_sk)
sym_sk_sd <- calc(sym_sk, fun = sd)
```

W efekcie otrzymujemy mapę odchylenia standardowego symulowanych wartości. Można na niej odczytać obszary o najpewniejszych (najmniej zmiennych) wartościach (niebieski kolor) oraz obszary o największej zmienności cechy (kolor żółty).

```
spplot(sym_sk_sd)
```



## 13.5 Sekwencyjna symulacja danych kodowanych

### 13.5.1 Sekwencyjna symulacja danych kodowanych (ang. *Sequential indicator simulation*)

Symulacje geostatystyczne można również zastosować do danych binarnych. Dla potrzeb przykładu tworzone jest nowa zmienna `temp_ind` przyjmująca wartość TRUE dla pomiarów o wartościach temperatury niższych niż 12 stopni Celsjusza oraz FALSE dla pomiarów o wartościach temperatury równych lub wyższych niż 12 stopni Celsjusza.

```
summary(punkty$temp)
```

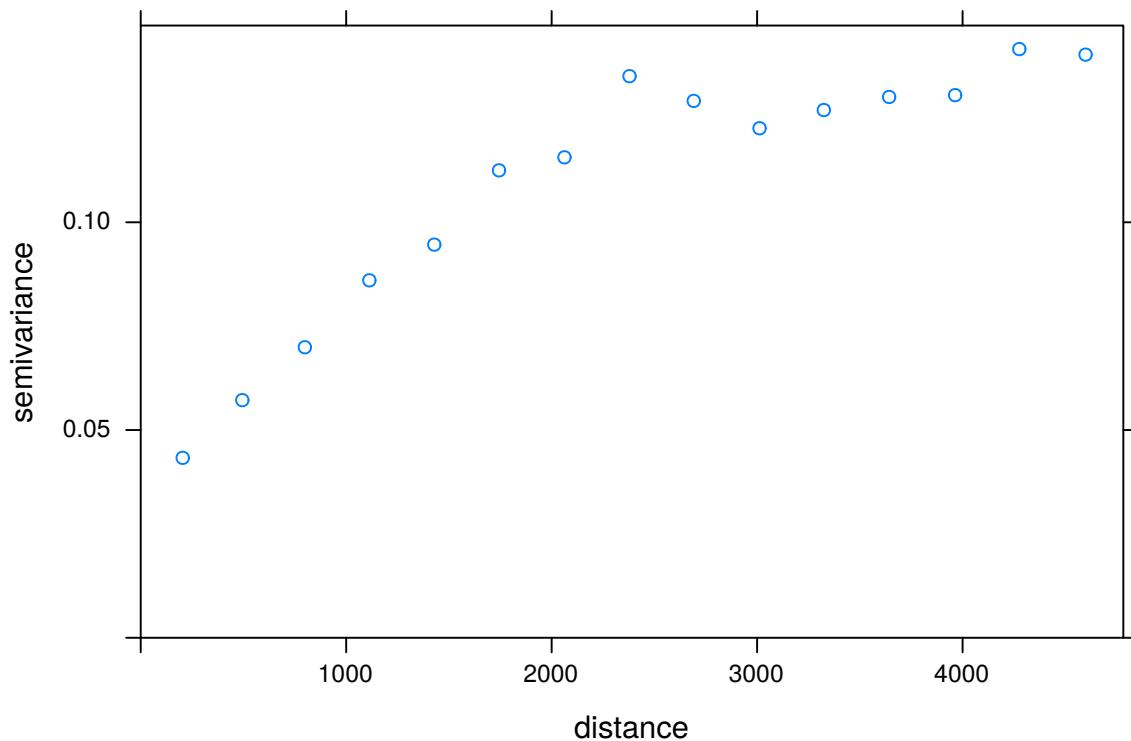
```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##  8.706 13.280 15.310 15.950 18.270 26.140
```

```
punkty$temp_ind <- punkty$temp < 12
summary(punkty$temp_ind)
```

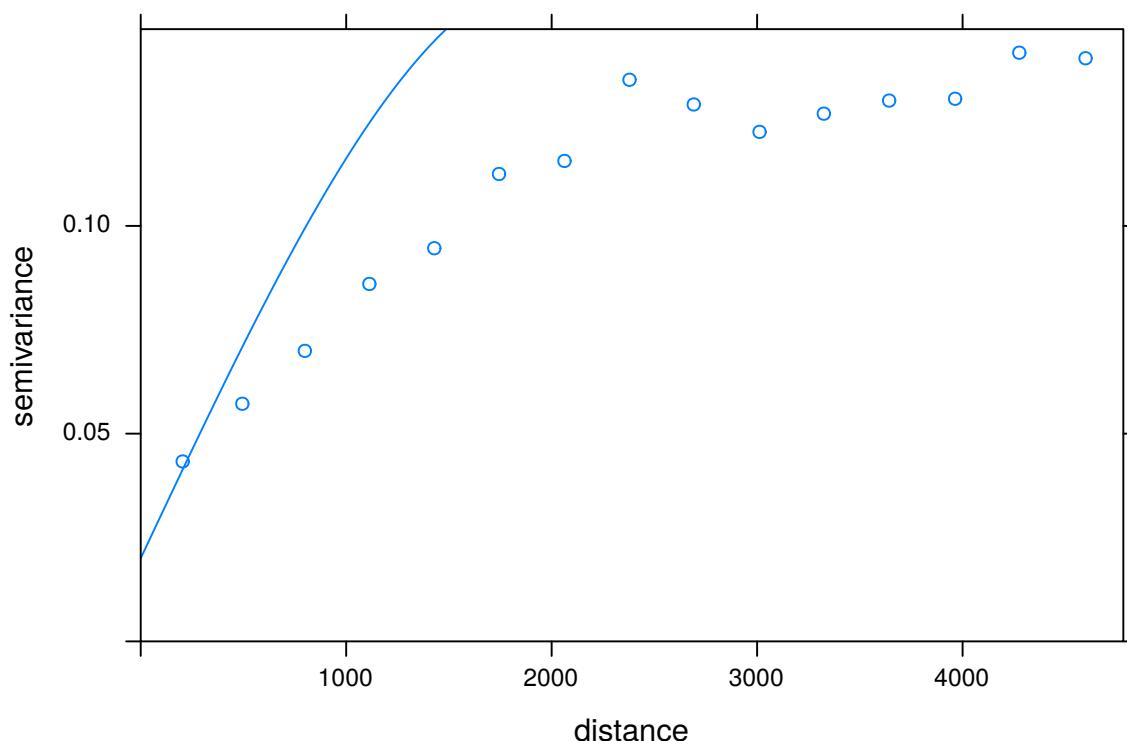
```
##    Mode   FALSE   TRUE   NA's
## logical  212     38      0
```

W tej metodzie kolejne etapy przypominają przeprowadzenie krigingu danych kodowanych. Jedynie w funkcji `krige()` należy dodać argument mówiący o liczbie symulacji do przeprowadzenia (`nsim`).

```
vario_ind <- variogram(temp_ind~1, punkty)
plot(vario_ind)
```

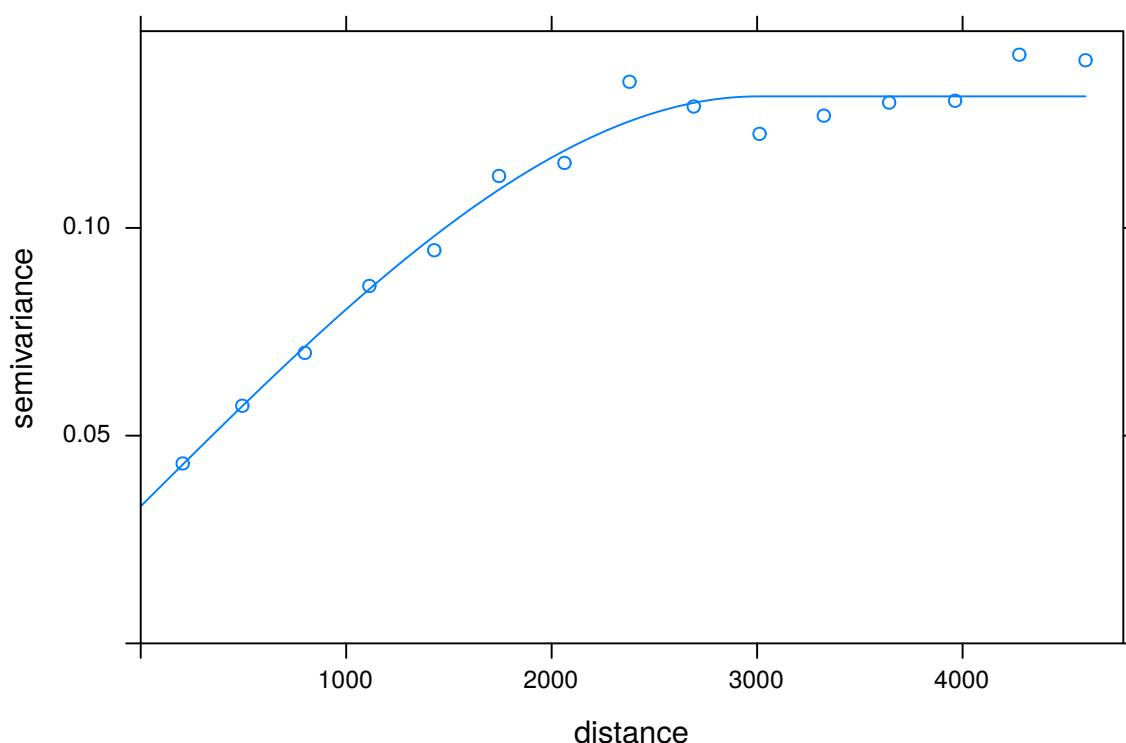


```
model_ind <- vgm(0.14, model = 'Sph', range = 2000, nugget = 0.02)
plot(vario_ind, model=model_ind)
```



```
fitted_ind <- fit.variogram(vario_ind, model_ind)
fitted_ind
```

```
##   model      psill     range
## 1   Nug 0.03299461 0.000
## 2   Sph 0.09866551 3006.403
plot(vario_ind, model=fitted_ind)
```



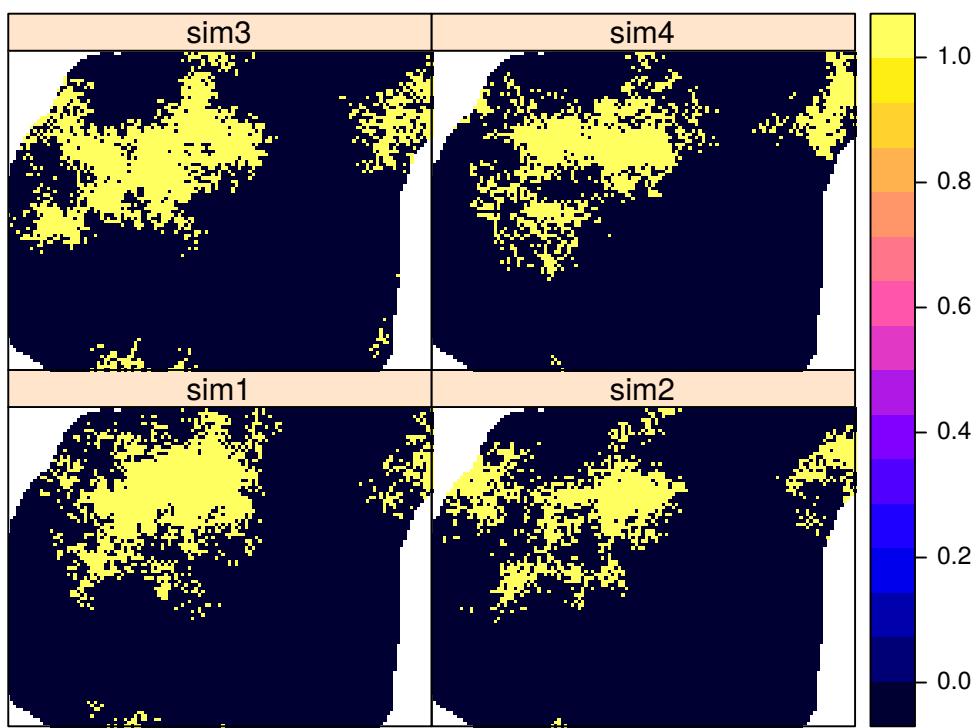
```
sym_ind <- krige(temp_ind~1, punkty, siatka, model=fitted_ind, indicators=TRUE, nsim=4, nmax=30)
```

```
## drawing 4 GLS realisations of beta...
## [using conditional indicator simulation]
```

Wynik symulacji danych kodowanych znacząco różni się od wyniku krigingu danych kodowanych. W przeciwieństwie do tej drugiej metody, w rezultacie symulacji nie otrzymujemy prawdopodobieństwa zajścia danej klasy, ale konkretne wartości 1 lub 0.

```
spplot(sym_ind, main='Symulacje warunkowe')
```

### Symulacje warunkowe



## Rozdział 14

# Źródła wiedzy

### 14.1 Podstawy R

- An Introduction to R - oficjalne wprowadzenie do R
- Przewodnik po pakiecie R
- Programowanie w języku R
- Statystyczna analiza danych z wykorzystaniem programu R

### 14.2 Analizy przestrzenne w R

- Applied Spatial Data Analysis with R
- CRAN Task View: Analysis of Spatial Data

### 14.3 Geostatystyka

- Praktyczny poradnik - jak szybko zrozumieć i wykorzystać geostatystykę w pracy badawczej
- A Practical Guide to Geostatistical Mapping
- gstat user's manual
- Applied Geostatistics
- Statistics for spatial data
- Geostatistics for Natural Resources Evaluation
- Geostatistics for Environmental Scientists



# Bibliografia

- Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wickham, H., Atkins, A., and Hyndman, R. (2016). *rmarkdown: Dynamic Documents for R*. R package version 1.0.9010.
- Auguie, B. (2016). *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R package version 2.2.1.
- Bivand, R., Keitt, T., and Rowlingson, B. (2016). *rgdal: Bindings for the Geospatial Data Abstraction Library*. R package version 1.1-10.
- Bivand, R. and Rundel, C. (2016). *rgeos: Interface to Geometry Engine - Open Source (GEOS)*. R package version 0.3-19.
- from Jed Wing, M. K. C., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., the R Core Team, Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., and Candan., C. (2016). *caret: Classification and Regression Training*. R package version 6.0-71.
- Giraudoux, P. (2016). *pgirmess: Data Analysis in Ecology*. R package version 1.6.4.
- Hijmans, R. J. (2016). *raster: Geographic Data Analysis and Modeling*. R package version 2.5-8.
- Hijmans, R. J., Phillips, S., Leathwick, J., and Elith, J. (2016). *dismo: Species Distribution Modeling*. R package version 1.1-1.
- Nowosad, J. (2016). *geostatbook: Geostatystyka w R*. R package version 0.2.2.
- Nychka, D., Furrer, R., Paige, J., and Sain, S. (2016). *fields: Tools for Spatial Data*. R package version 8.4-1.
- Pebesma, E. and Bivand, R. (2016). *sp: Classes and Methods for Spatial Data*. R package version 1.2-3.
- Pebesma, E. and Graeler, B. (2016). *gstat: Spatial and Spatio-Temporal Geostatistical Modelling, Prediction and Simulation*. R package version 1.1-3.
- Perpinan Lamigueiro, O. and Hijmans, R. (2016). *rasterVis: Visualization Methods for Raster Data*. R package version 0.40.
- R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Wei, T. and Simko, V. (2016). *corrplot: Visualization of a Correlation Matrix*. R package version 0.77.
- Wickham, H. and Chang, W. (2016). *ggplot2: An Implementation of the Grammar of Graphics*. R package version 2.1.0.
- Xie, Y. (2016a). *bookdown: Authoring Books with R Markdown*. R package version 0.1.
- Xie, Y. (2016b). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.14.