CPC Merged Analysis of Precipitation (CMAP)

2022-05-30

```r
library(ncdf4)

# set path and filename
ncpath <- "/Users/hwingren/Downloads/Blackwell/Blackwell_Scholars_2022/"
ncname <- "precip.mon.mean.nc"
ncfname <- paste(ncpath, ncname, sep="")
dname <- "precip"
```

open a netCDF file

```r
ncin <- nc_open(ncfname)
print(ncin)
```

```
## File /Users/hwingren/Downloads/Blackwell/Blackwell_Scholars_2022/precip.mon.mean.nc (NC_FORMAT_NETCDF
##
##      1 variables (excluding dimension variables):
##         float precip[lon,lat,time]   (Chunking: [144,72,1])  (Compression: shuffle,level 2)
##             long_name: Average Monthly Rate of Precipitation
##             valid_range: 0
##              valid_range: 70
##             units: mm/day
##             add_offset: 0
##             scale_factor: 1
##             missing_value: -9.96920996838687e+36
##             precision: 2
##             least_significant_digit: 2
##             var_desc: Precipitation
##             dataset: CPC Merged Analysis of Precipitation Standard
##             level_desc: Surface
##             statistic: Mean
##             parent_stat: Mean
##             actual_range: 0
##              actual_range: 144.490005493164
##
##      3 dimensions:
##         lon  Size:144
##             units: degrees_east
##             long_name: Longitude
##             actual_range: 1.25
##              actual_range: 358.75
##             standard_name: longitude
##             axis: X
##         lat  Size:72
##             units: degrees_north
```

```
##             actual_range: 88.75
##              actual_range: -88.75
##             long_name: Latitude
##             standard_name: latitude
##             axis: Y
##         time  Size:520   *** is unlimited ***
##             units: hours since 1800-01-01 00:00:0.0
##             long_name: Time
##             delta_t: 0000-01-00 00:00:00
##             avg_period: 0000-01-00 00:00:00
##             standard_name: time
##             axis: T
##             actual_range: 1569072
##              actual_range: 1948176
##
##     11 global attributes:
##         Conventions: COARDS
##         title: CPC Merged Analysis of Precipitation (excludes NCEP Reanalysis)
##         platform: Analyses
##         source: ftp ftp.cpc.ncep.noaa.gov precip/cmap/monthly
##         dataset_title: CPC Merged Analysis of Precipitation
##         documentation: https://www.esrl.noaa.gov/psd/data/gridded/data.cmap.html
##         date_modified: 26 Feb 2019
##         References: https://www.psl.noaa.gov/data/gridded/data.cmap.html
##         version: V2205
##         history: update 05/2022 V2205
##         data_modified: 2022-05-09
```

get longitude and latitude

```
lon <- ncvar_get(ncin,"lon")
nlon <- dim(lon)
head(lon)
```

```
## [1]  1.25  3.75  6.25  8.75 11.25 13.75
```

```
lat <- ncvar_get(ncin,"lat")
nlat <- dim(lat)
head(lat)
```

```
## [1] 88.75 86.25 83.75 81.25 78.75 76.25
```

```
print(c(nlon,nlat))
```

```
## [1] 144  72
```

get time

```
time <- ncvar_get(ncin,"time")
time
```

```
##   [1] 1569072 1569816 1570488 1571232 1571952 1572696 1573416 1574160 1574904
##  [10] 1575624 1576368 1577088 1577832 1578576 1579272 1580016 1580736 1581480
##  [19] 1582200 1582944 1583688 1584408 1585152 1585872 1586616 1587360 1588032
```

```
##  [28] 1588776 1589496 1590240 1590960 1591704 1592448 1593168 1593912 1594632
##  [37] 1595376 1596120 1596792 1597536 1598256 1599000 1599720 1600464 1601208
##  [46] 1601928 1602672 1603392 1604136 1604880 1605552 1606296 1607016 1607760
##  [55] 1608480 1609224 1609968 1610688 1611432 1612152 1612896 1613640 1614336
##  [64] 1615080 1615800 1616544 1617264 1618008 1618752 1619472 1620216 1620936
##  [73] 1621680 1622424 1623096 1623840 1624560 1625304 1626024 1626768 1627512
##  [82] 1628232 1628976 1629696 1630440 1631184 1631856 1632600 1633320 1634064
##  [91] 1634784 1635528 1636272 1636992 1637736 1638456 1639200 1639944 1640616
## [100] 1641360 1642080 1642824 1643544 1644288 1645032 1645752 1646496 1647216
## [109] 1647960 1648704 1649400 1650144 1650864 1651608 1652328 1653072 1653816
## [118] 1654536 1655280 1656000 1656744 1657488 1658160 1658904 1659624 1660368
## [127] 1661088 1661832 1662576 1663296 1664040 1664760 1665504 1666248 1666920
## [136] 1667664 1668384 1669128 1669848 1670592 1671336 1672056 1672800 1673520
## [145] 1674264 1675008 1675680 1676424 1677144 1677888 1678608 1679352 1680096
## [154] 1680816 1681560 1682280 1683024 1683768 1684464 1685208 1685928 1686672
## [163] 1687392 1688136 1688880 1689600 1690344 1691064 1691808 1692552 1693224
## [172] 1693968 1694688 1695432 1696152 1696896 1697640 1698360 1699104 1699824
## [181] 1700568 1701312 1701984 1702728 1703448 1704192 1704912 1705656 1706400
## [190] 1707120 1707864 1708584 1709328 1710072 1710744 1711488 1712208 1712952
## [199] 1713672 1714416 1715160 1715880 1716624 1717344 1718088 1718832 1719528
## [208] 1720272 1720992 1721736 1722456 1723200 1723944 1724664 1725408 1726128
## [217] 1726872 1727616 1728288 1729032 1729752 1730496 1731216 1731960 1732704
## [226] 1733424 1734168 1734888 1735632 1736376 1737048 1737792 1738512 1739256
## [235] 1739976 1740720 1741464 1742184 1742928 1743648 1744392 1745136 1745808
## [244] 1746552 1747272 1748016 1748736 1749480 1750224 1750944 1751688 1752408
## [253] 1753152 1753896 1754592 1755336 1756056 1756800 1757520 1758264 1759008
## [262] 1759728 1760472 1761192 1761936 1762680 1763352 1764096 1764816 1765560
## [271] 1766280 1767024 1767768 1768488 1769232 1769952 1770696 1771440 1772112
## [280] 1772856 1773576 1774320 1775040 1775784 1776528 1777248 1777992 1778712
## [289] 1779456 1780200 1780872 1781616 1782336 1783080 1783800 1784544 1785288
## [298] 1786008 1786752 1787472 1788216 1788960 1789656 1790400 1791120 1791864
## [307] 1792584 1793328 1794072 1794792 1795536 1796256 1797000 1797744 1798416
## [316] 1799160 1799880 1800624 1801344 1802088 1802832 1803552 1804296 1805016
## [325] 1805760 1806504 1807176 1807920 1808640 1809384 1810104 1810848 1811592
## [334] 1812312 1813056 1813776 1814520 1815264 1815936 1816680 1817400 1818144
## [343] 1818864 1819608 1820352 1821072 1821816 1822536 1823280 1824024 1824720
## [352] 1825464 1826184 1826928 1827648 1828392 1829136 1829856 1830600 1831320
## [361] 1832064 1832808 1833480 1834224 1834944 1835688 1836408 1837152 1837896
## [370] 1838616 1839360 1840080 1840824 1841568 1842240 1842984 1843704 1844448
## [379] 1845168 1845912 1846656 1847376 1848120 1848840 1849584 1850328 1851000
## [388] 1851744 1852464 1853208 1853928 1854672 1855416 1856136 1856880 1857600
## [397] 1858344 1859088 1859784 1860528 1861248 1861992 1862712 1863456 1864200
## [406] 1864920 1865664 1866384 1867128 1867872 1868544 1869288 1870008 1870752
## [415] 1871472 1872216 1872960 1873680 1874424 1875144 1875888 1876632 1877304
## [424] 1878048 1878768 1879512 1880232 1880976 1881720 1882440 1883184 1883904
## [433] 1884648 1885392 1886064 1886808 1887528 1888272 1888992 1889736 1890480
## [442] 1891200 1891944 1892664 1893408 1894152 1894848 1895592 1896312 1897056
## [451] 1897776 1898520 1899264 1899984 1900728 1901448 1902192 1902936 1903608
## [460] 1904352 1905072 1905816 1906536 1907280 1908024 1908744 1909488 1910208
## [469] 1910952 1911696 1912368 1913112 1913832 1914576 1915296 1916040 1916784
## [478] 1917504 1918248 1918968 1919712 1920456 1921128 1921872 1922592 1923336
## [487] 1924056 1924800 1925544 1926264 1927008 1927728 1928472 1929216 1929912
## [496] 1930656 1931376 1932120 1932840 1933584 1934328 1935048 1935792 1936512
## [505] 1937256 1938000 1938672 1939416 1940136 1940880 1941600 1942344 1943088
```

```
## [514] 1943808 1944552 1945272 1946016 1946760 1947432 1948176
```

```
tunits <- ncatt_get(ncin,"time","units")
nt <- dim(time)
nt
```

```
## [1] 520
```

```
tunits
```

```
## $hasatt
## [1] TRUE
##
## $value
## [1] "hours since 1800-01-01 00:00:0.0"
```

get precipitation

```
precip_array <- ncvar_get(ncin,dname)
dlname <- ncatt_get(ncin,dname,"long_name")
dunits <- ncatt_get(ncin,dname,"units")
fillvalue <- ncatt_get(ncin,dname,"_FillValue")
dim(precip_array)
```

```
## [1] 144  72 520
```

get global attributes

```
title <- ncatt_get(ncin,0,"title")
institution <- ncatt_get(ncin,0,"institution")
datasource <- ncatt_get(ncin,0,"source")
references <- ncatt_get(ncin,0,"references")
history <- ncatt_get(ncin,0,"history")
Conventions <- ncatt_get(ncin,0,"Conventions")

nc_close(ncin)
```

load some packages

```
library(chron)
library(lattice)
library(RColorBrewer)
```

convert time – split the time units string into fields

```
tustr <- strsplit(tunits$value, " ")
tdstr <- strsplit(unlist(tustr)[3], "-")
tmonth <- as.integer(unlist(tdstr)[2])
tday <- as.integer(unlist(tdstr)[3])
tyear <- as.integer(unlist(tdstr)[1])
chron(time,origin=c(tmonth, tday, tyear))
```

```
##   [1] 12/22/95 01/04/98 11/07/99 11/21/01 11/11/03 11/24/05 11/14/07 11/27/09
##   [9] 12/11/11 11/30/13 12/14/15 12/03/17 12/17/19 12/30/21 11/26/23 12/09/25
##  [17] 11/29/27 12/12/29 12/02/31 12/15/33 12/29/35 12/18/37 01/01/40 12/21/41
##  [25] 01/04/44 01/17/46 11/20/47 12/03/49 11/23/51 12/06/53 11/26/55 12/09/57
##  [33] 12/23/59 12/12/61 12/26/63 12/15/65 12/29/67 01/11/70 11/14/71 11/27/73
##  [41] 11/17/75 11/30/77 11/20/79 12/03/81 12/17/83 12/06/85 12/20/87 12/09/89
##  [49] 12/23/91 01/05/94 11/08/95 11/21/97 11/11/99 11/25/01 11/15/03 11/28/05
##  [57] 12/12/07 12/01/09 12/15/11 12/04/13 12/18/15 12/31/17 11/27/19 12/10/21
##  [65] 11/30/23 12/13/25 12/03/27 12/16/29 12/30/31 12/19/33 01/02/36 12/22/37
##  [73] 01/05/40 01/18/42 11/21/43 12/04/45 11/24/47 12/07/49 11/27/51 12/10/53
##  [81] 12/24/55 12/13/57 12/27/59 12/16/61 12/30/63 01/12/66 11/15/67 11/28/69
##  [89] 11/18/71 12/01/73 11/21/75 12/04/77 12/18/79 12/07/81 12/21/83 12/10/85
##  [97] 12/24/87 01/06/90 11/09/91 11/22/93 11/12/95 11/25/97 11/15/99 11/29/01
## [105] 12/13/03 12/02/05 12/16/07 12/05/09 12/19/11 01/01/14 11/28/15 12/11/17
## [113] 12/01/19 12/14/21 12/04/23 12/17/25 12/31/27 12/20/29 01/03/32 12/23/33
## [121] 01/06/36 01/19/38 11/22/39 12/05/41 11/25/43 12/08/45 11/28/47 12/11/49
## [129] 12/25/51 12/14/53 12/28/55 12/17/57 12/31/59 01/13/62 11/16/63 11/29/65
## [137] 11/19/67 12/02/69 11/22/71 12/05/73 12/19/75 12/08/77 12/22/79 12/11/81
## [145] 12/25/83 01/07/86 11/10/87 11/23/89 11/13/91 11/26/93 11/16/95 11/29/97
## [153] 12/13/99 12/02/01 12/16/03 12/05/05 12/19/07 01/01/10 11/28/11 12/11/13
## [161] 12/01/15 12/14/17 12/04/19 12/17/21 12/31/23 12/20/25 01/03/28 12/23/29
## [169] 01/06/32 01/19/34 11/22/35 12/05/37 11/25/39 12/08/41 11/28/43 12/11/45
## [177] 12/25/47 12/14/49 12/28/51 12/17/53 12/31/55 01/13/58 11/16/59 11/29/61
## [185] 11/19/63 12/02/65 11/22/67 12/05/69 12/19/71 12/08/73 12/22/75 12/11/77
## [193] 12/25/79 01/07/82 11/10/83 11/23/85 11/13/87 11/26/89 11/16/91 11/29/93
## [201] 12/13/95 12/02/97 12/16/99 12/06/01 12/20/03 01/02/06 11/29/07 12/12/09
## [209] 12/02/11 12/15/13 12/05/15 12/18/17 01/01/20 12/21/21 01/04/24 12/24/25
## [217] 01/07/28 01/20/30 11/23/31 12/06/33 11/26/35 12/09/37 11/29/39 12/12/41
## [225] 12/26/43 12/15/45 12/29/47 12/18/49 01/01/52 01/14/54 11/17/55 11/30/57
## [233] 11/20/59 12/03/61 11/23/63 12/06/65 12/20/67 12/09/69 12/23/71 12/12/73
## [241] 12/26/75 01/08/78 11/11/79 11/24/81 11/14/83 11/27/85 11/17/87 11/30/89
## [249] 12/14/91 12/03/93 12/17/95 12/06/97 12/20/99 01/03/02 11/30/03 12/13/05
## [257] 12/03/07 12/16/09 12/06/11 12/19/13 01/02/16 12/22/17 01/05/20 12/25/21
## [265] 01/08/24 01/21/26 11/24/27 12/07/29 11/27/31 12/10/33 11/30/35 12/13/37
## [273] 12/27/39 12/16/41 12/30/43 12/19/45 01/02/48 01/15/50 11/18/51 12/01/53
## [281] 11/21/55 12/04/57 11/24/59 12/07/61 12/21/63 12/10/65 12/24/67 12/13/69
## [289] 12/27/71 01/09/74 11/12/75 11/25/77 11/15/79 11/28/81 11/18/83 12/01/85
## [297] 12/15/87 12/04/89 12/18/91 12/07/93 12/21/95 01/03/98 11/30/99 12/14/01
## [305] 12/04/03 12/17/05 12/07/07 12/20/09 01/03/12 12/23/13 01/06/16 12/26/17
## [313] 01/09/20 01/22/22 11/25/23 12/08/25 11/28/27 12/11/29 12/01/31 12/14/33
## [321] 12/28/35 12/17/37 12/31/39 12/20/41 01/03/44 01/16/46 11/19/47 12/02/49
## [329] 11/22/51 12/05/53 11/25/55 12/08/57 12/22/59 12/11/61 12/25/63 12/14/65
## [337] 12/28/67 01/10/70 11/13/71 11/26/73 11/16/75 11/29/77 11/19/79 12/02/81
## [345] 12/16/83 12/05/85 12/19/87 12/08/89 12/22/91 01/04/94 12/01/95 12/14/97
## [353] 12/04/99 12/17/01 12/07/03 12/20/05 01/03/08 12/23/09 01/06/12 12/26/13
## [361] 01/09/16 01/22/18 11/25/19 12/08/21 11/28/23 12/11/25 12/01/27 12/14/29
## [369] 12/28/31 12/17/33 12/31/35 12/20/37 01/03/40 01/16/42 11/19/43 12/02/45
## [377] 11/22/47 12/05/49 11/25/51 12/08/53 12/22/55 12/11/57 12/25/59 12/14/61
## [385] 12/28/63 01/10/66 11/13/67 11/26/69 11/16/71 11/29/73 11/19/75 12/02/77
## [393] 12/16/79 12/05/81 12/19/83 12/08/85 12/22/87 01/04/90 12/01/91 12/14/93
## [401] 12/04/95 12/17/97 12/07/99 12/21/01 01/04/04 12/24/05 01/07/08 12/27/09
## [409] 01/10/12 01/23/14 11/26/15 12/09/17 11/29/19 12/12/21 12/02/23 12/15/25
## [417] 12/29/27 12/18/29 01/01/32 12/21/33 01/04/36 01/17/38 11/20/39 12/03/41
## [425] 11/23/43 12/06/45 11/26/47 12/09/49 12/23/51 12/12/53 12/26/55 12/15/57
```

```
## [433] 12/29/59 01/11/62 11/14/63 11/27/65 11/17/67 11/30/69 11/20/71 12/03/73
## [441] 12/17/75 12/06/77 12/20/79 12/09/81 12/23/83 01/05/86 12/02/87 12/15/89
## [449] 12/05/91 12/18/93 12/08/95 12/21/97 01/04/00 12/25/01 01/08/04 12/28/05
## [457] 01/11/08 01/24/10 11/27/11 12/10/13 11/30/15 12/13/17 12/03/19 12/16/21
## [465] 12/30/23 12/19/25 01/02/28 12/22/29 01/05/32 01/18/34 11/21/35 12/04/37
## [473] 11/24/39 12/07/41 11/27/43 12/10/45 12/24/47 12/13/49 12/27/51 12/16/53
## [481] 12/30/55 01/12/58 11/15/59 11/28/61 11/18/63 12/01/65 11/21/67 12/04/69
## [489] 12/18/71 12/07/73 12/21/75 12/10/77 12/24/79 01/06/82 12/03/83 12/16/85
## [497] 12/06/87 12/19/89 12/09/91 12/22/93 01/05/96 12/25/97 01/08/00 12/29/01
## [505] 01/12/04 01/25/06 11/28/07 12/11/09 12/01/11 12/14/13 12/04/15 12/17/17
## [513] 12/31/19 12/20/21 01/03/24 12/23/25 01/06/28 01/19/30 11/22/31 12/05/33
```

replace netCDF fill values with NA's

```r
precip_array[precip_array==fillvalue$value] <- NA

length(na.omit(as.vector(precip_array[,,1])))
```

```
## [1] 8809
```

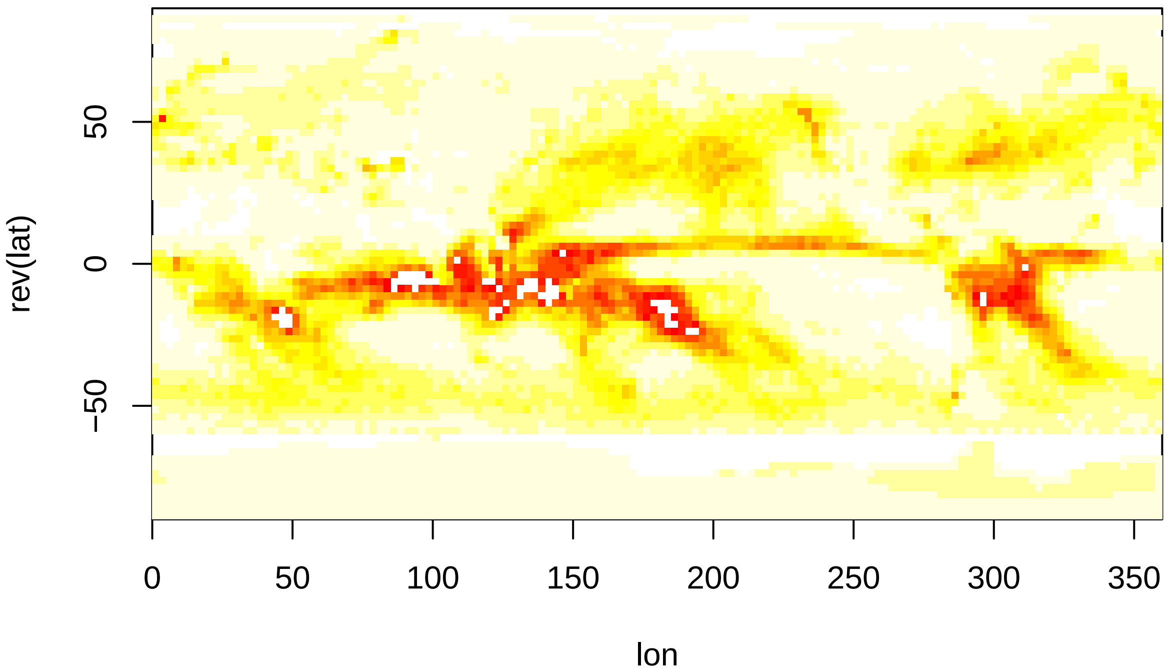
get 12 months slice of layer (Jan-Dec 1982 and Jan-Dec 1999)

```r
#m <- 1
precip_slice <- precip_array[,,c(37:48, 241:252)]
dim(precip_slice)
```

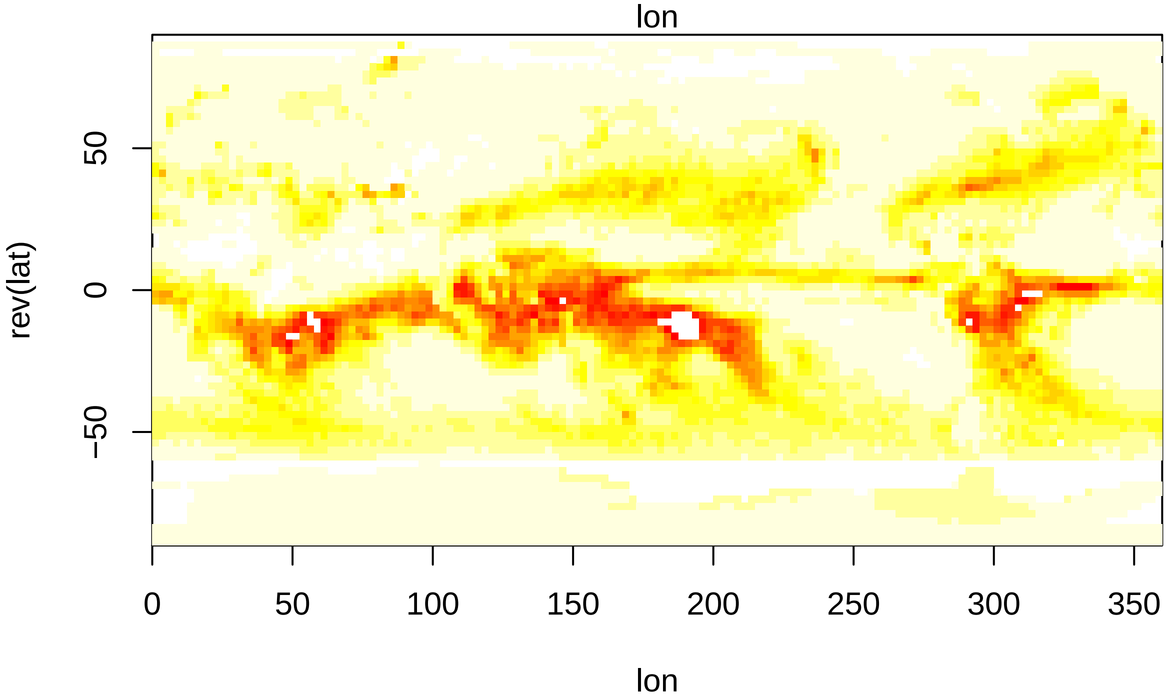
```
## [1] 144  72  24
```

quick map

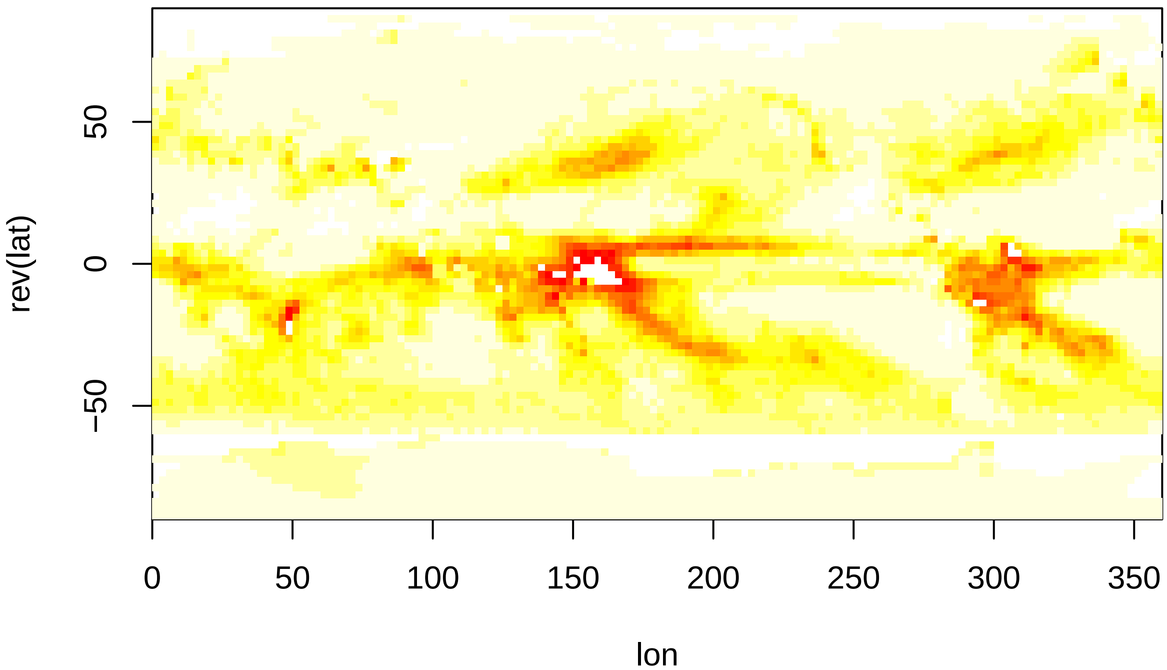
```r
for(m in 1:24){
image(lon,rev(lat),precip_slice[, ncol(precip_slice):1,m], col=rev(heat.colors(16)), breaks=0:16)}
```



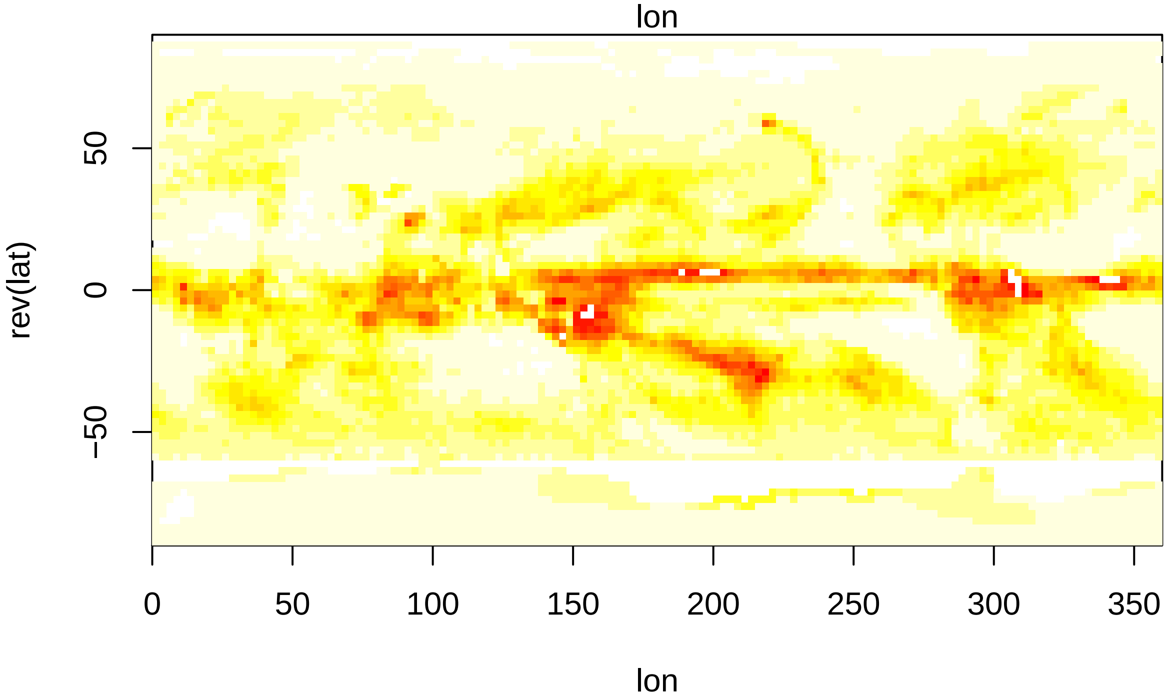
rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon



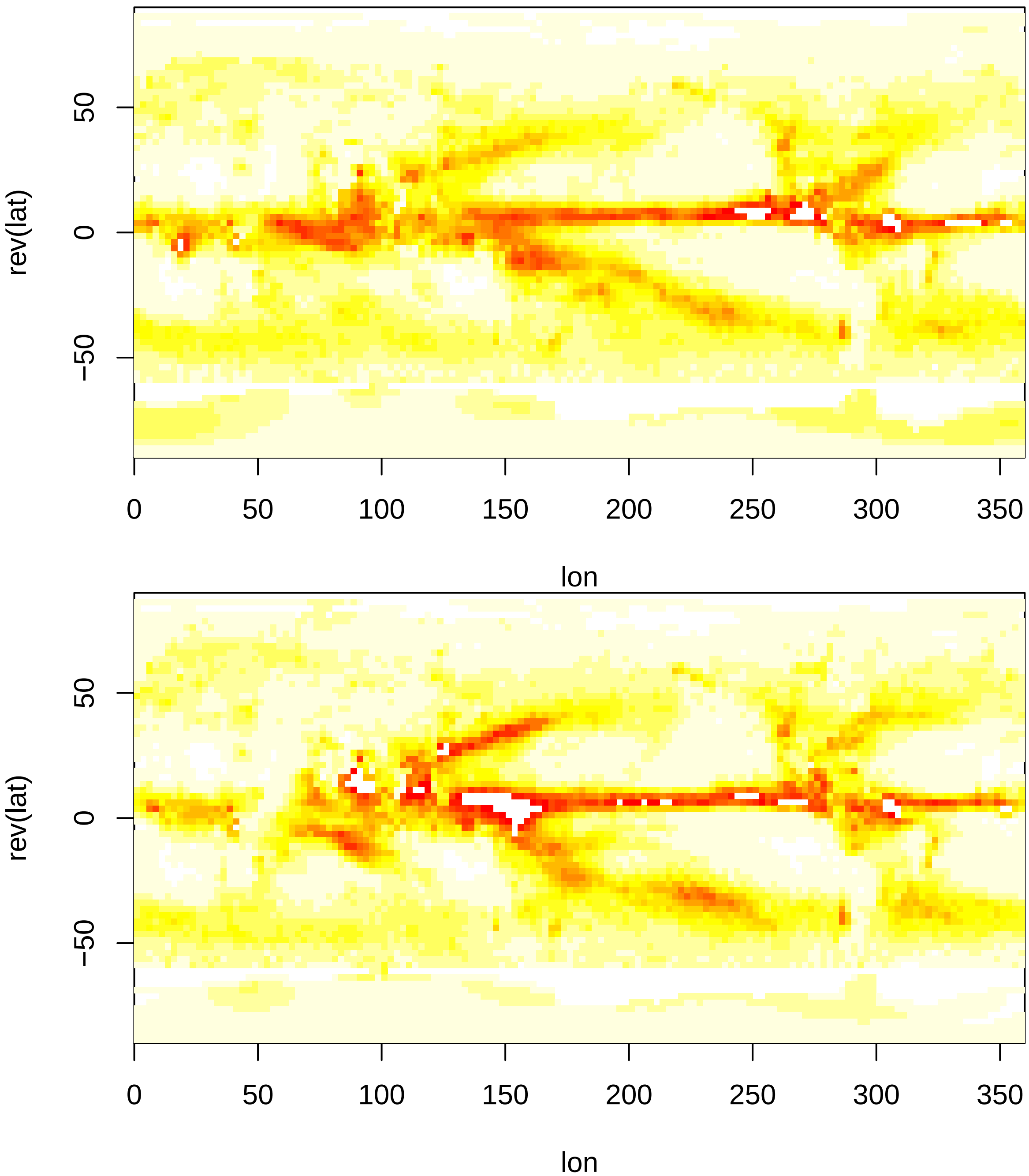
rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

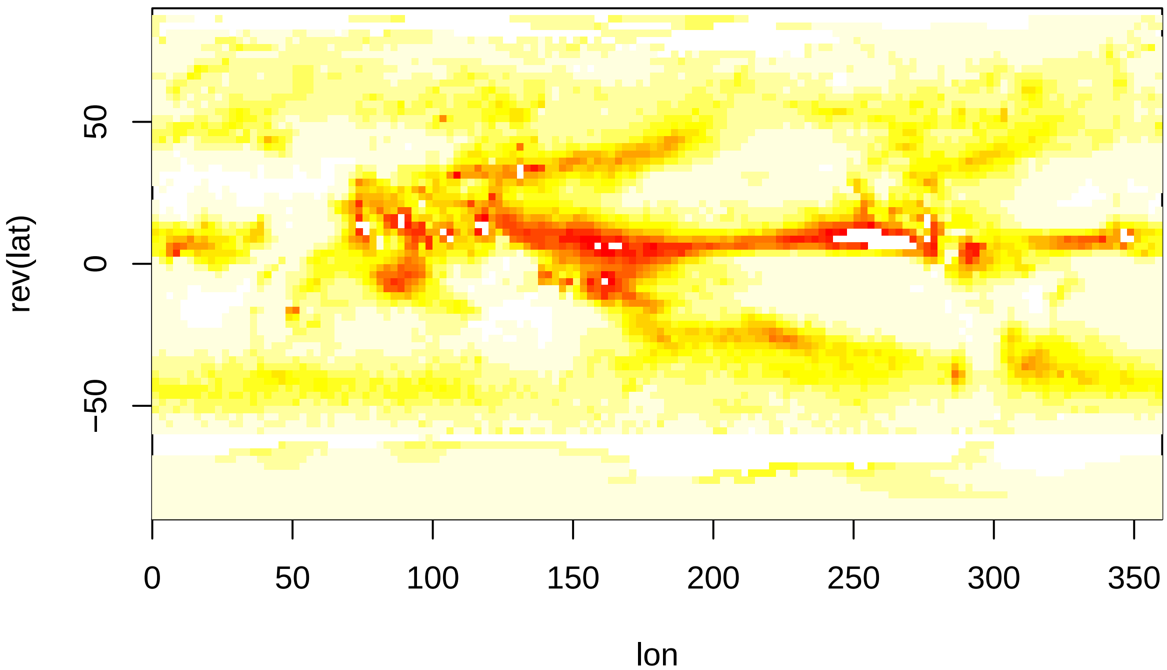


rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

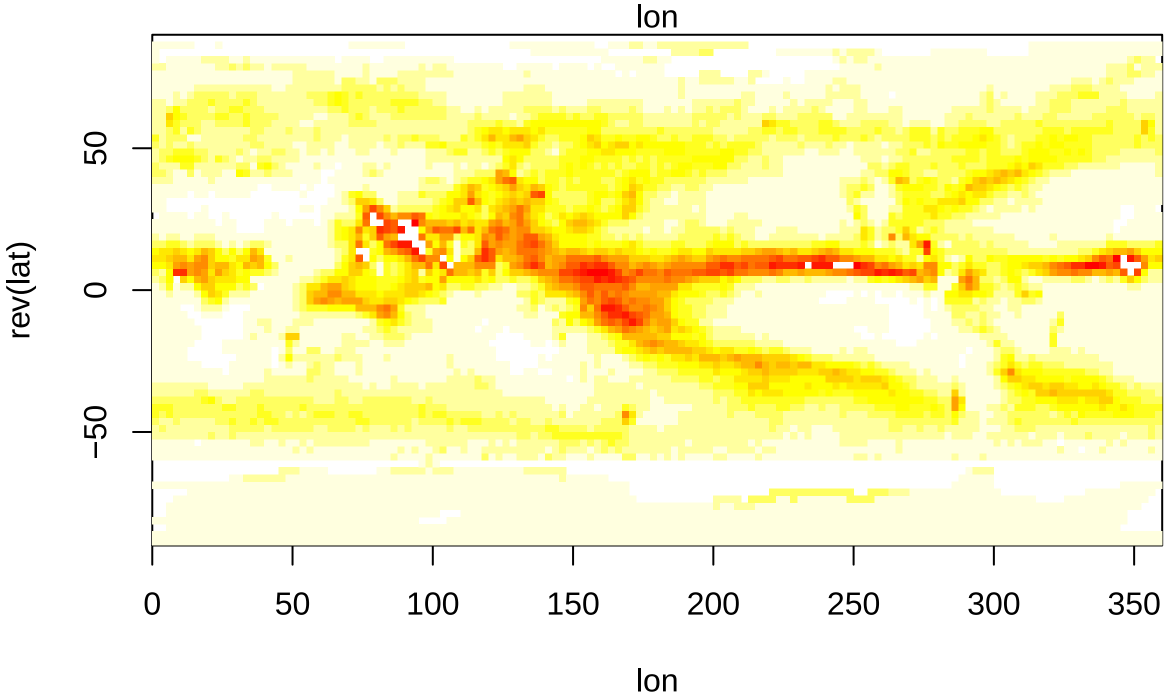


rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

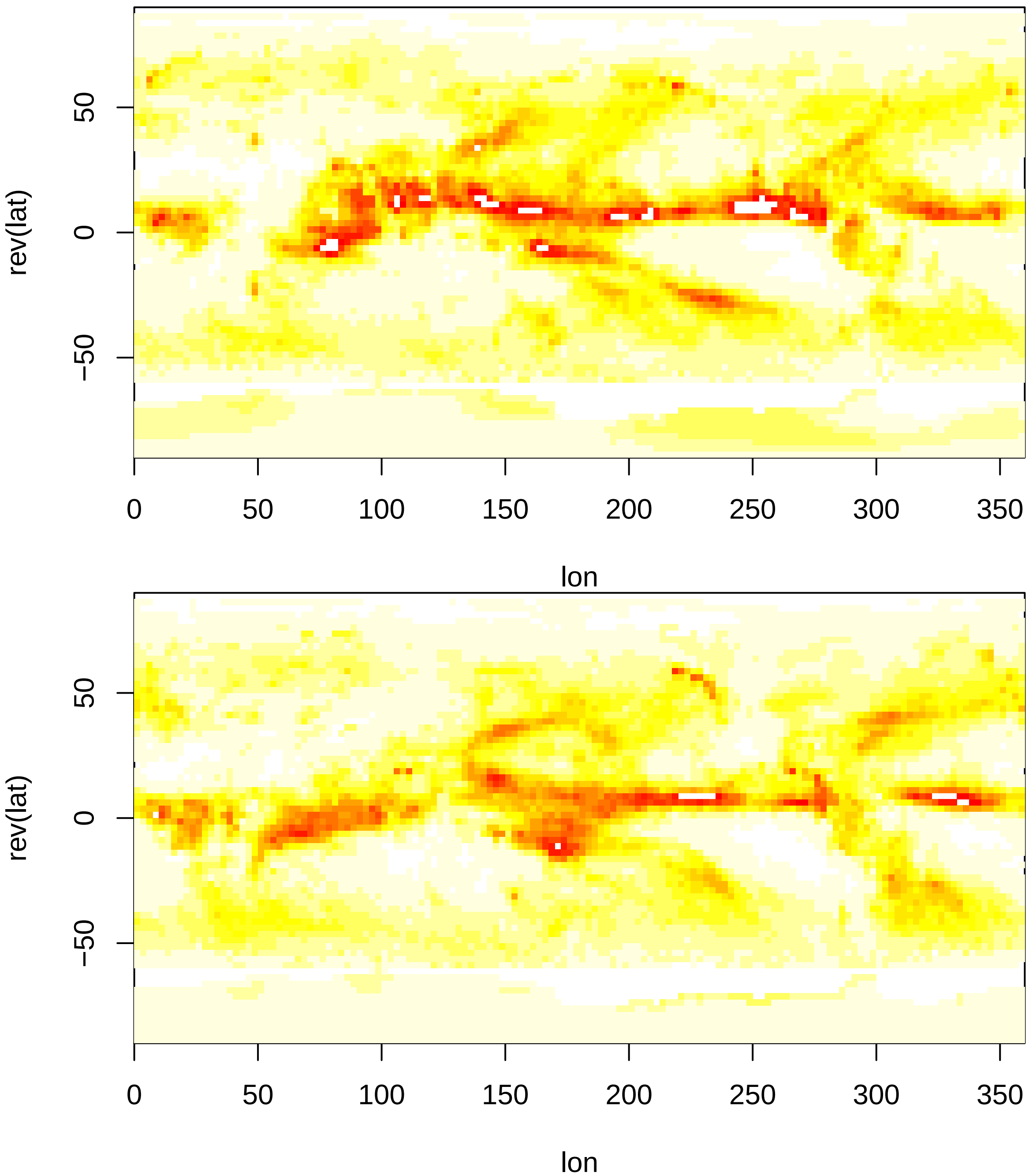

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon
rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

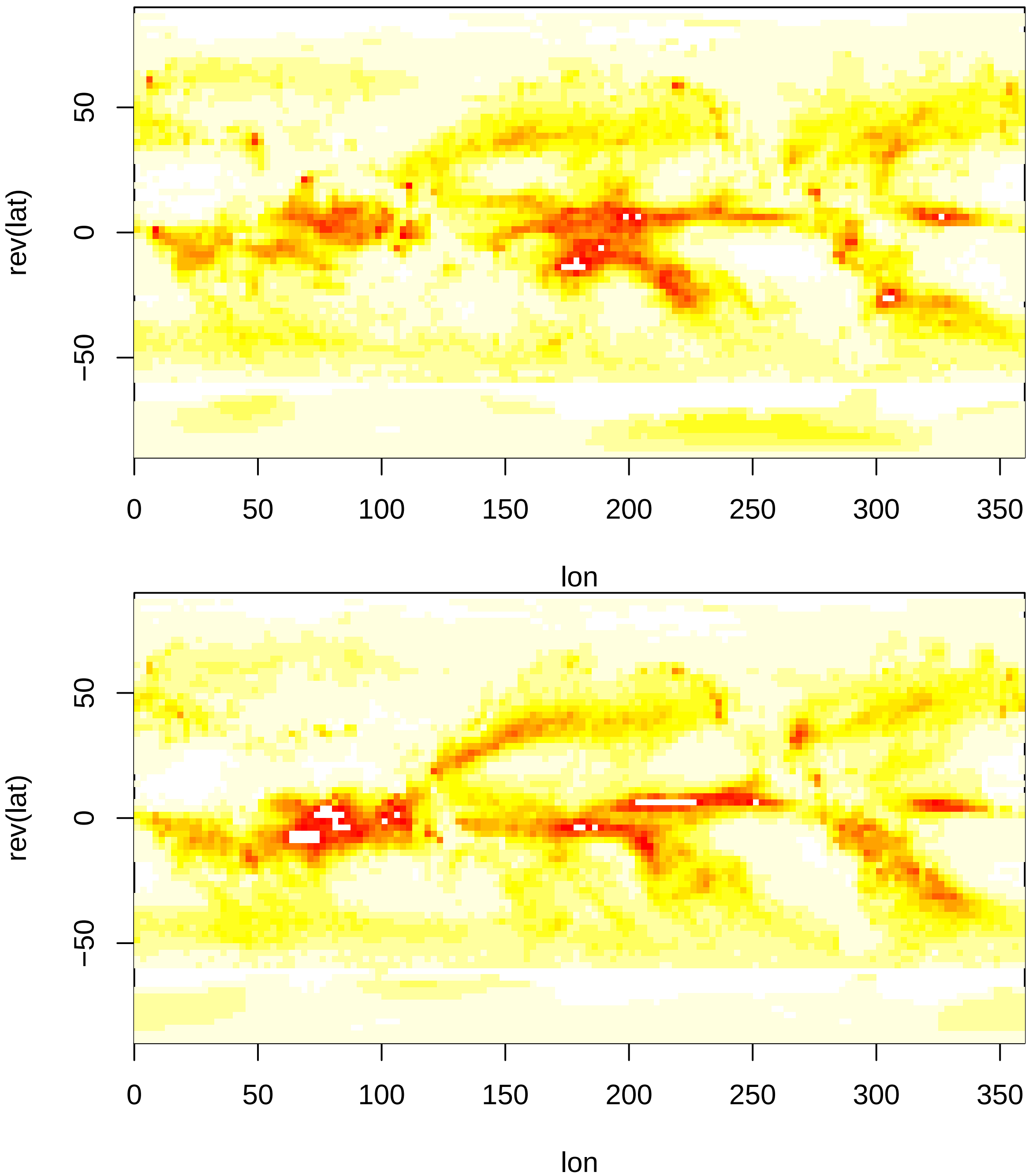


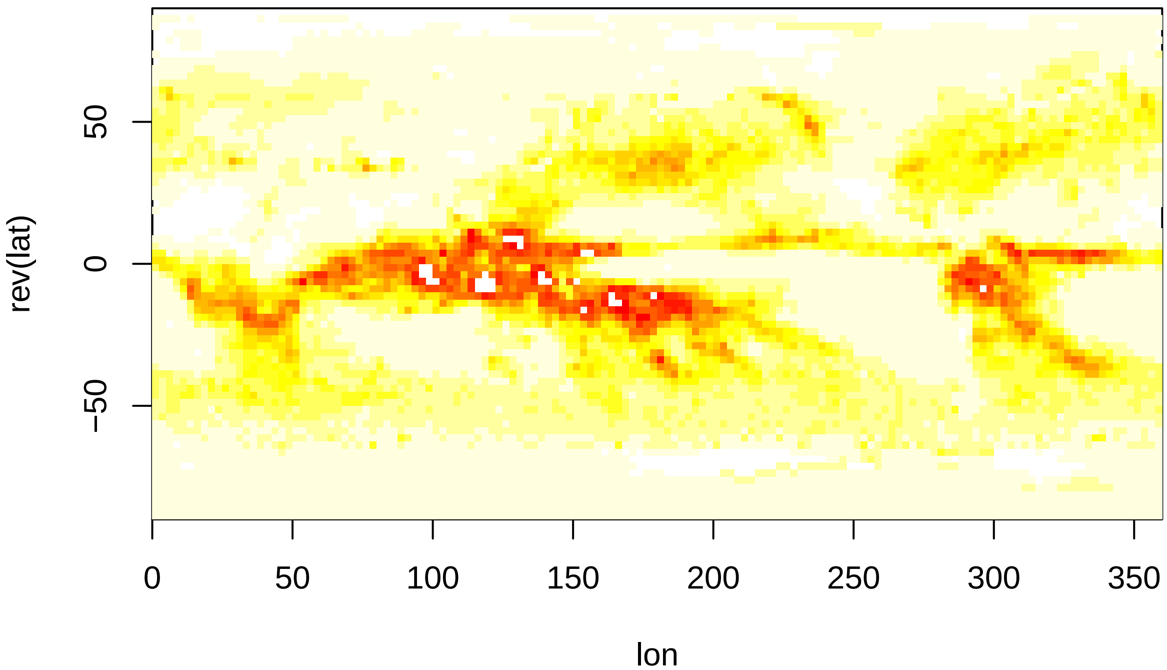
rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

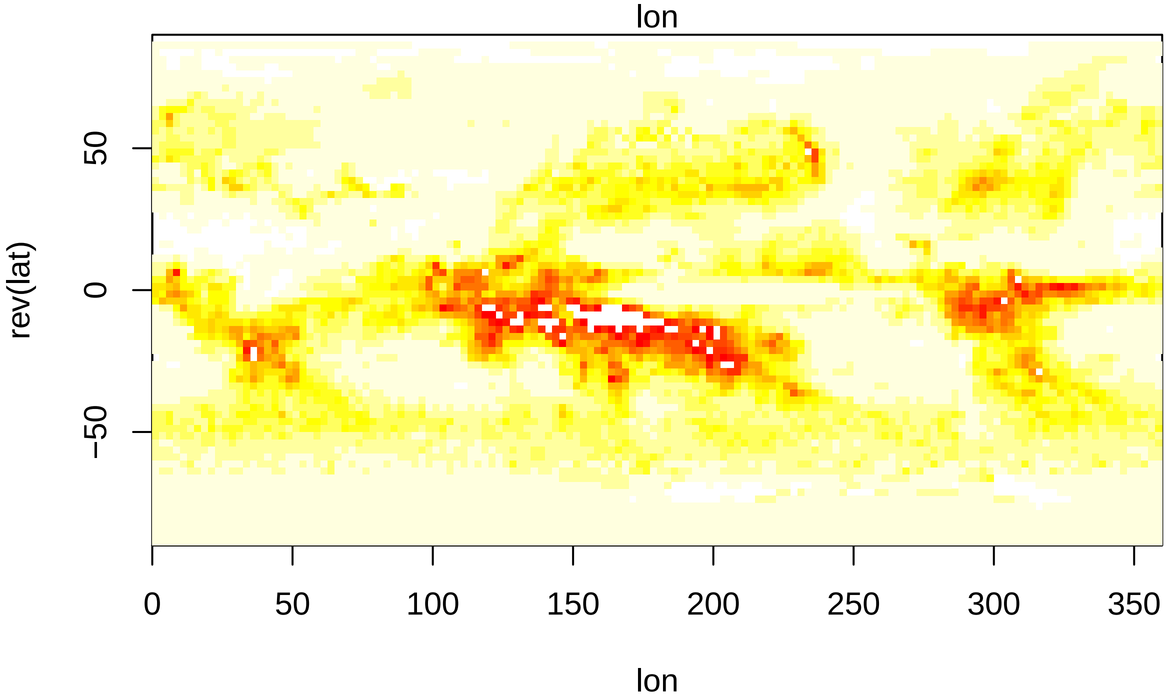


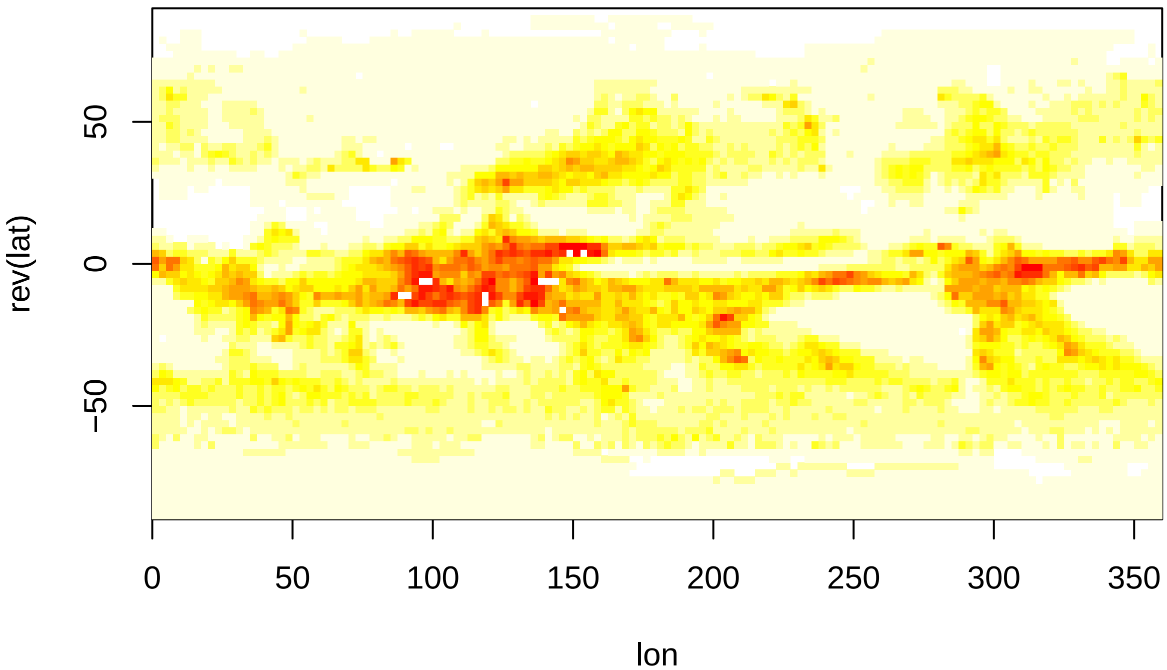
rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

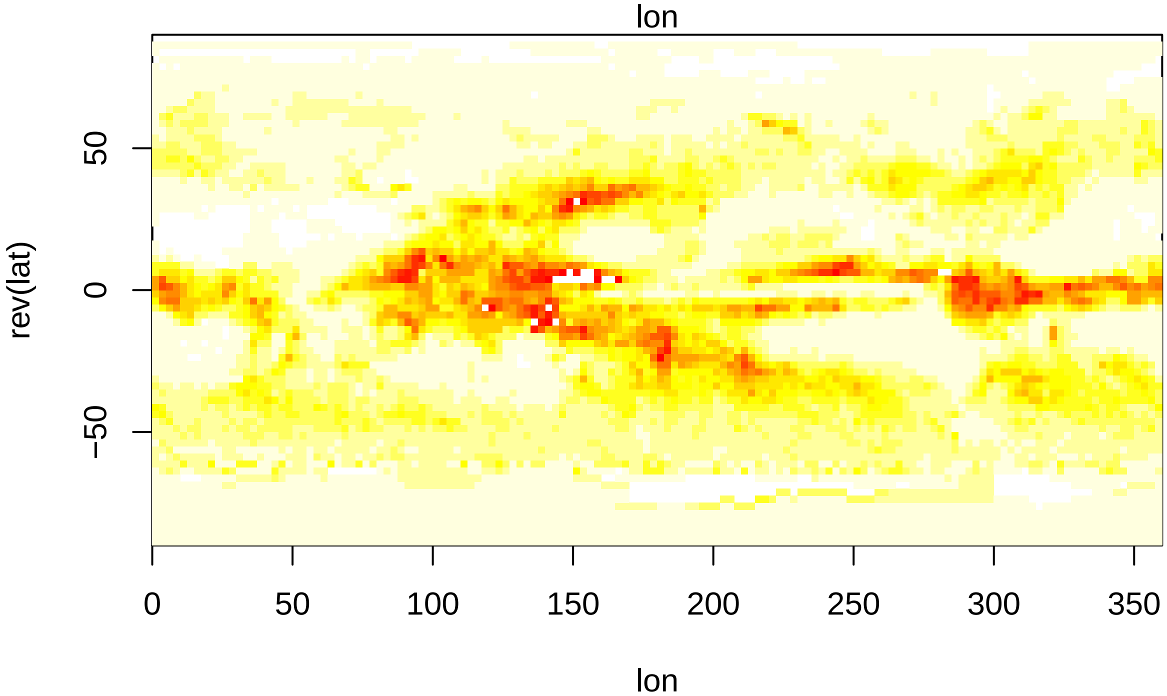

50
0
−50
rev(lat)
0
50
100
150
200
250
300
350
lon
50
0
−50
rev(lat)
0
50
100
150
200
250
300
350
lon

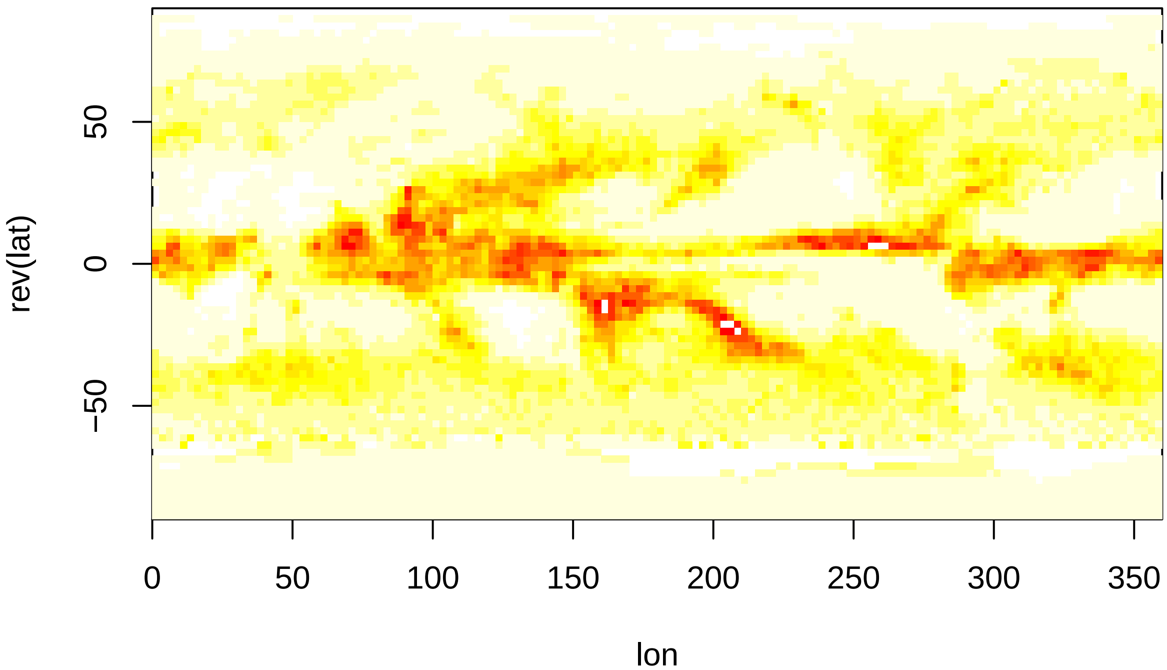

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon
rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

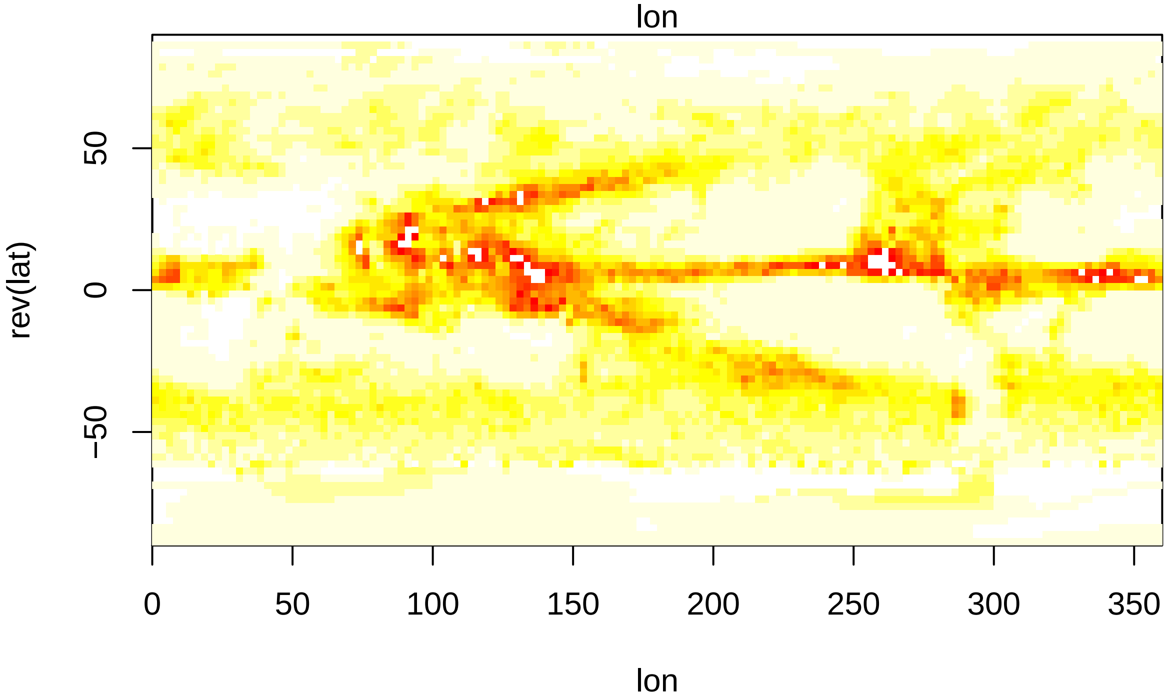

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

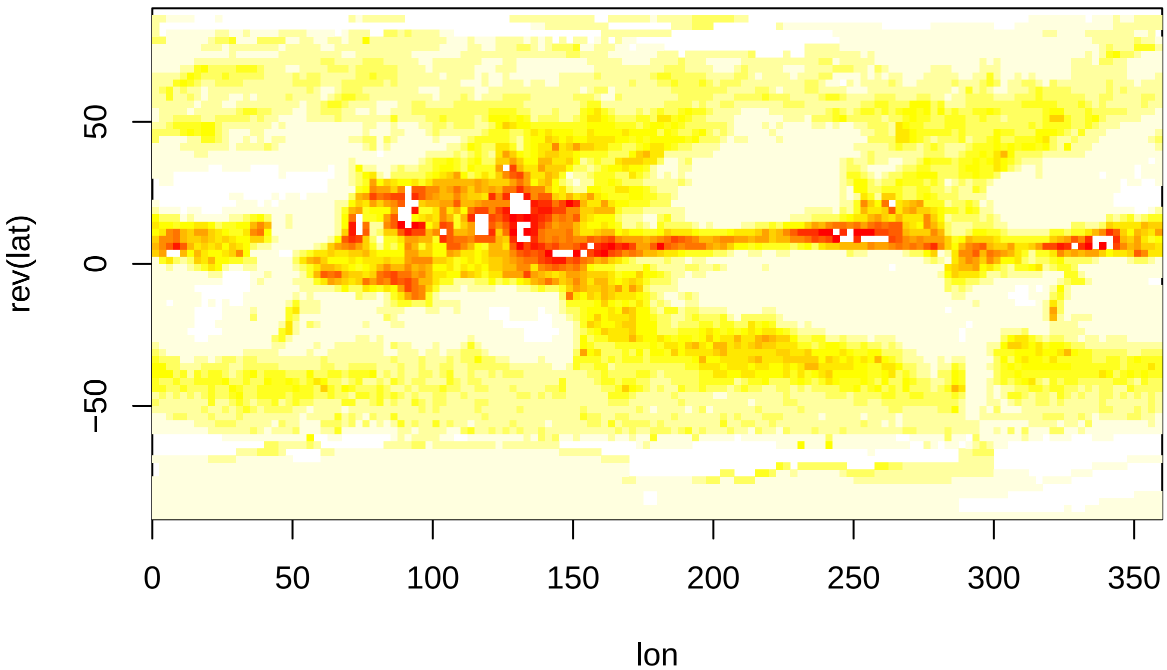

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

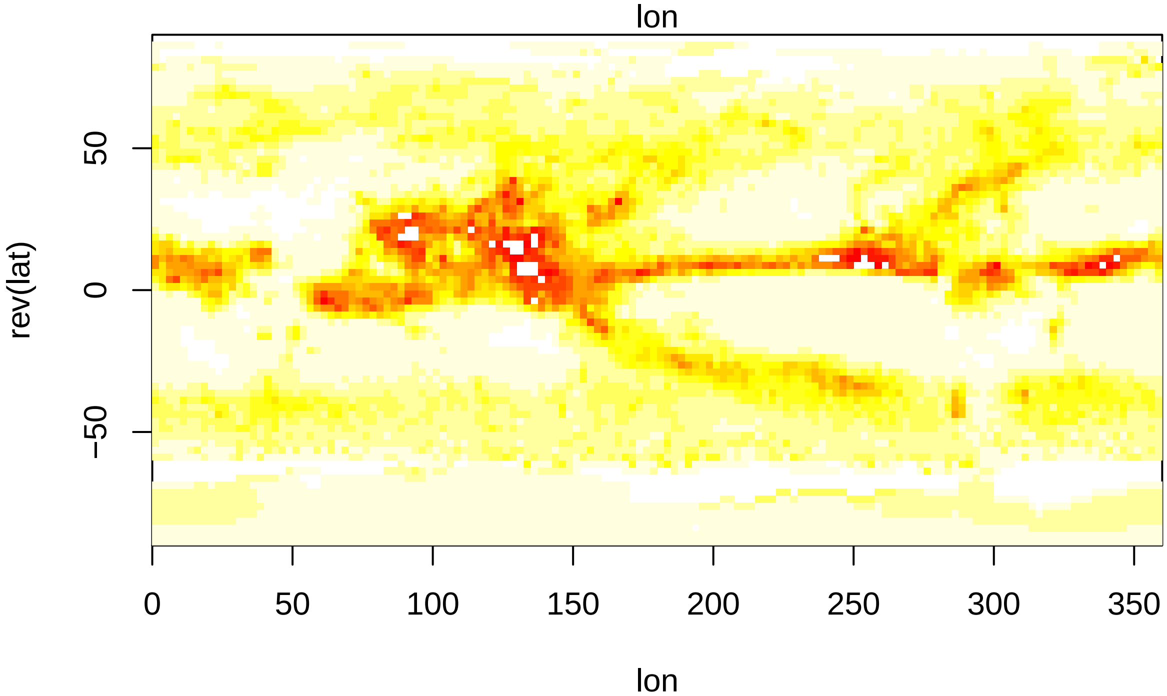

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

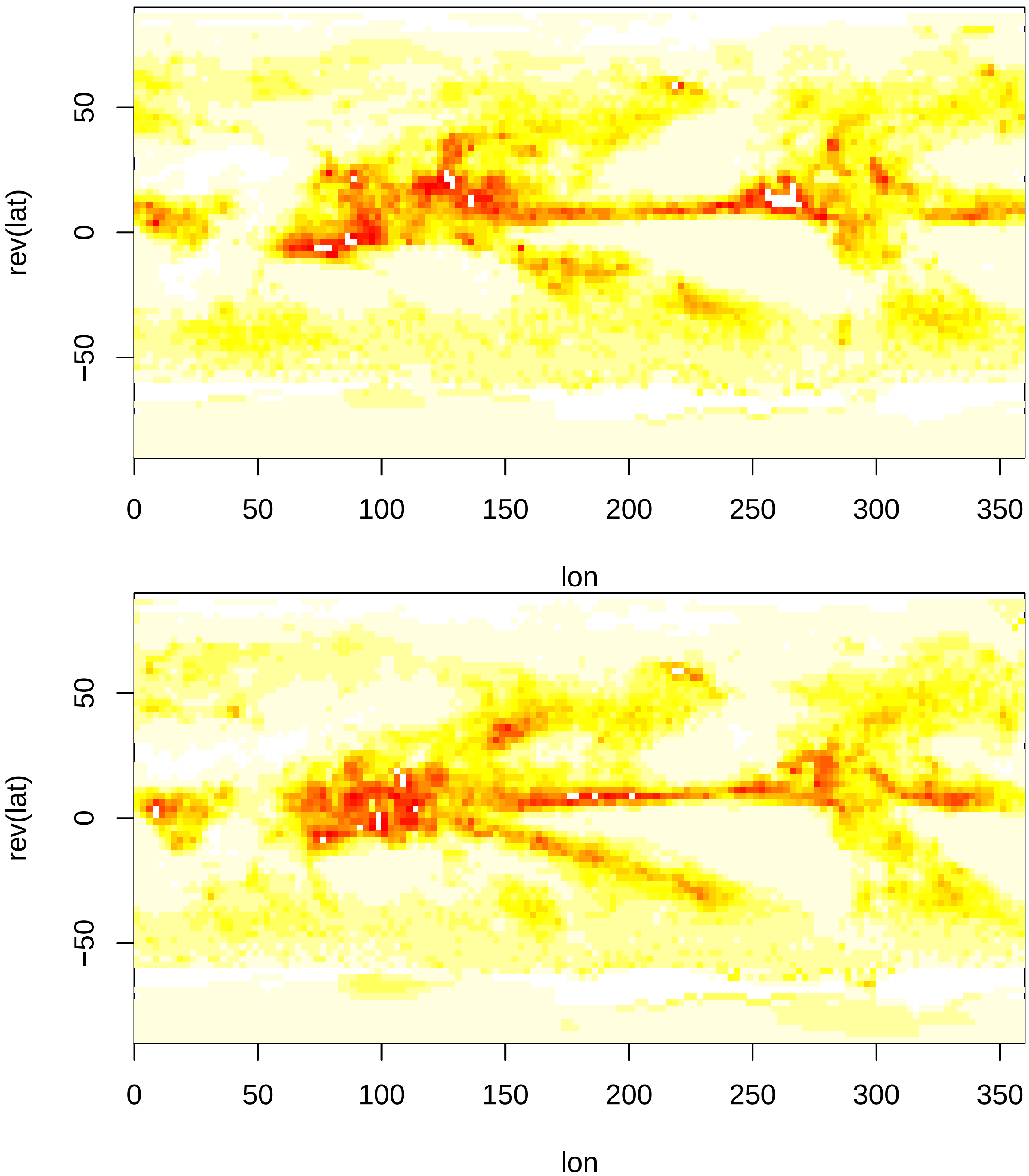

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

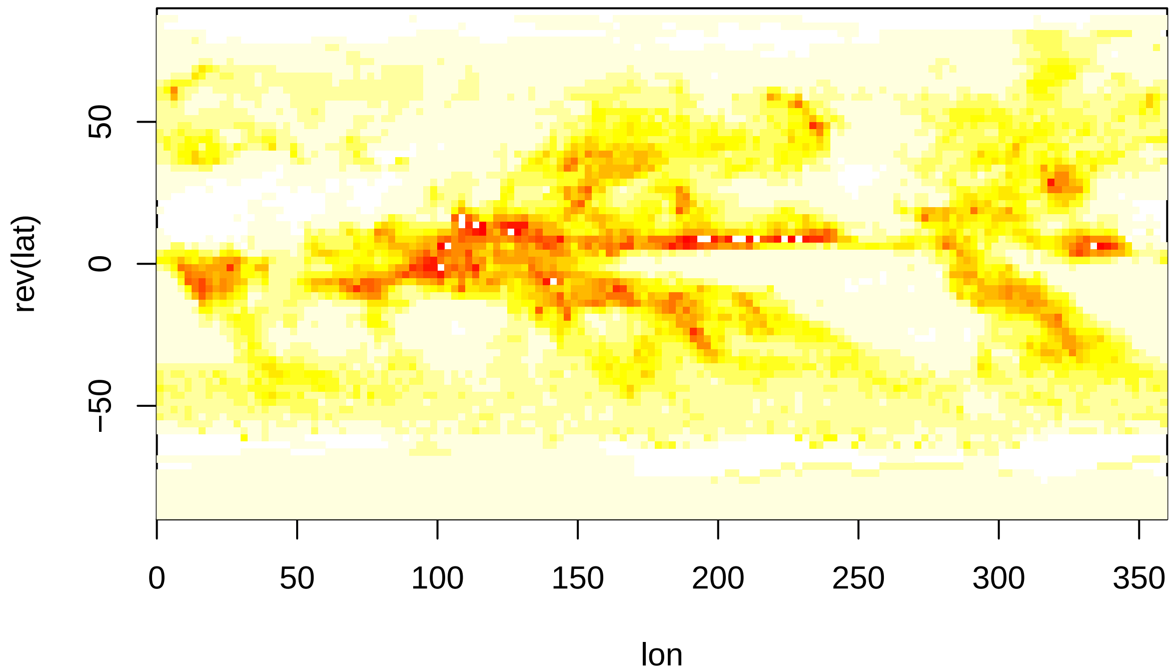

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

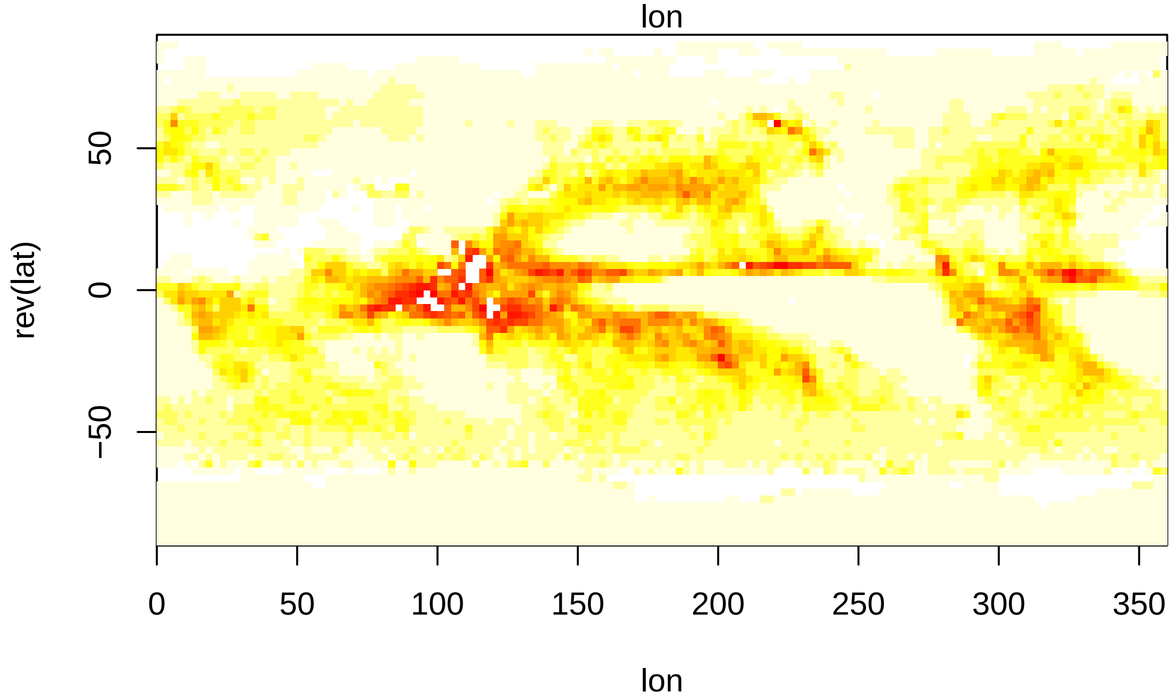

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon


rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon


rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon

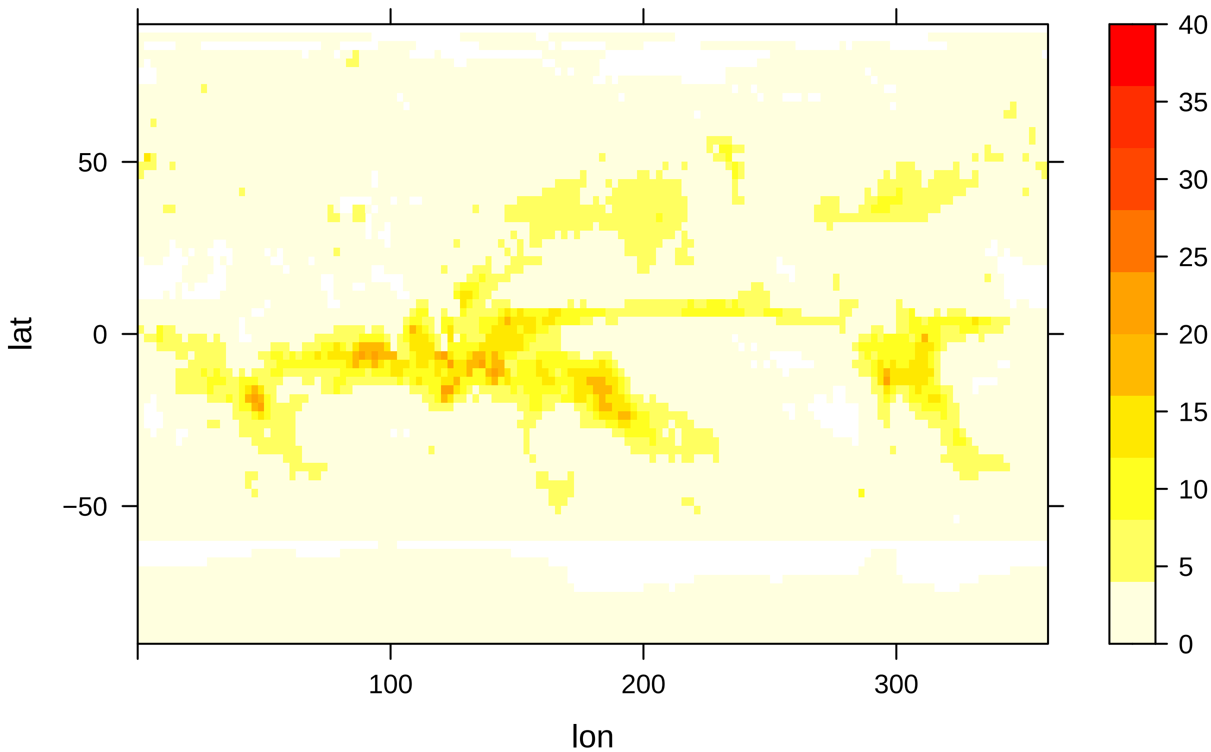

rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon
rev(lat)
50
0
−50
0
50
100
150
200
250
300
350
lon





levelplot of the slice

```
grid <- expand.grid(lon=lon, lat=lat)
cutpts <- seq(0, 40, length.out=11)
m<-1
levelplot(precip_slice[,,m] ~ lon * lat, data=grid, at=cutpts, cuts=11, pretty=T,
  col.regions=(rev(heat.colors(16))))
```



create new lon and lat vectors

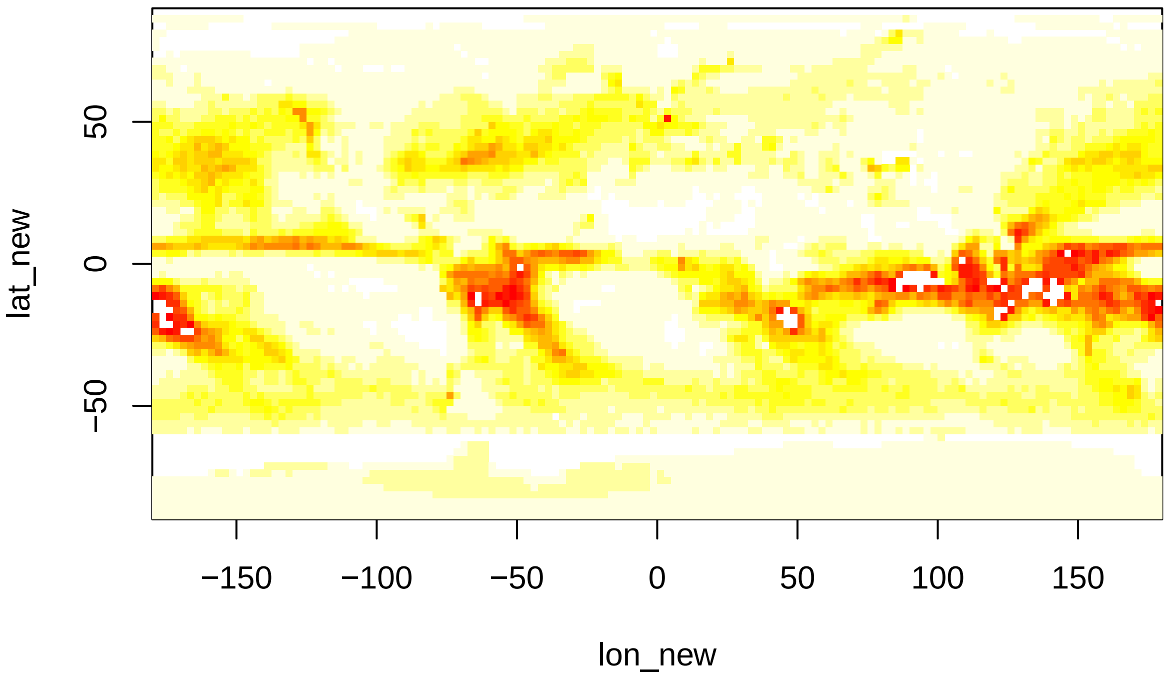
```
lon_new <- c(lon[73:144]-360, lon[1:72])
lat_new <- rev(lat)
```

create new precipitation matrix

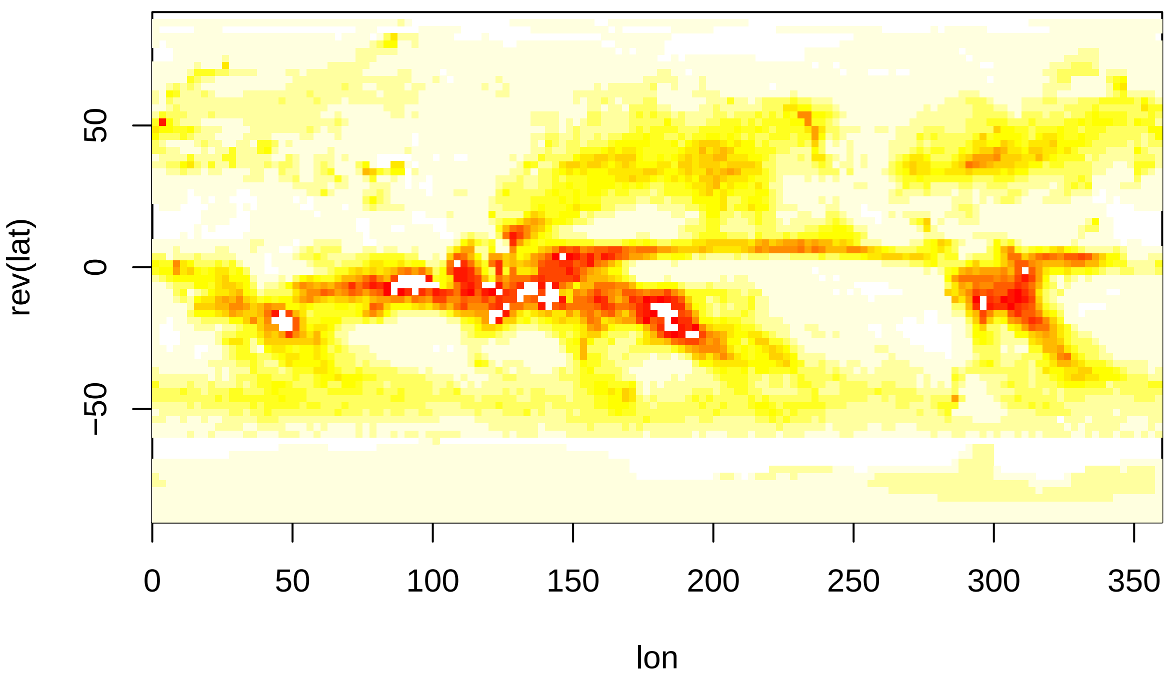
```
precip_slice_new <- precip_slice[c(73:144, 1:72), ncol(precip_slice):1,]
```

plot whole globe

```
m<-1 #January
image(lon_new, lat_new, precip_slice_new[,,m], col=rev(heat.colors(16)), breaks=0:16)
```




```
#Compare with previous plot
image(lon,rev(lat),precip_slice[, ncol(precip_slice):1,m], col=rev(heat.colors(16)), breaks=0:16)
```

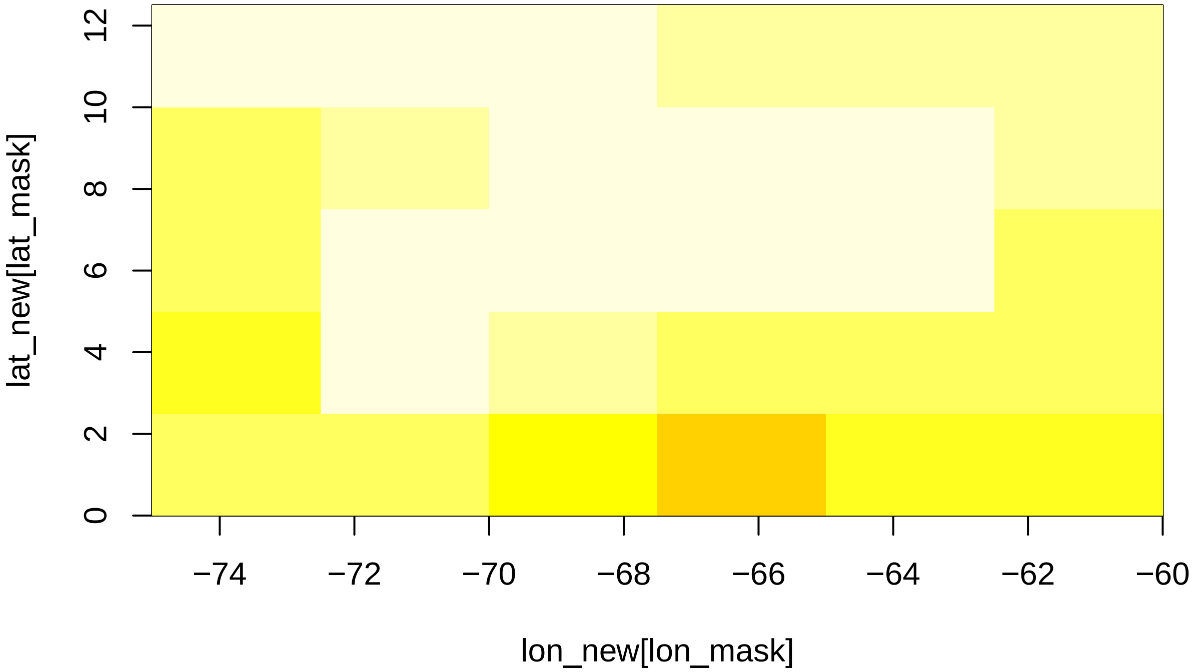


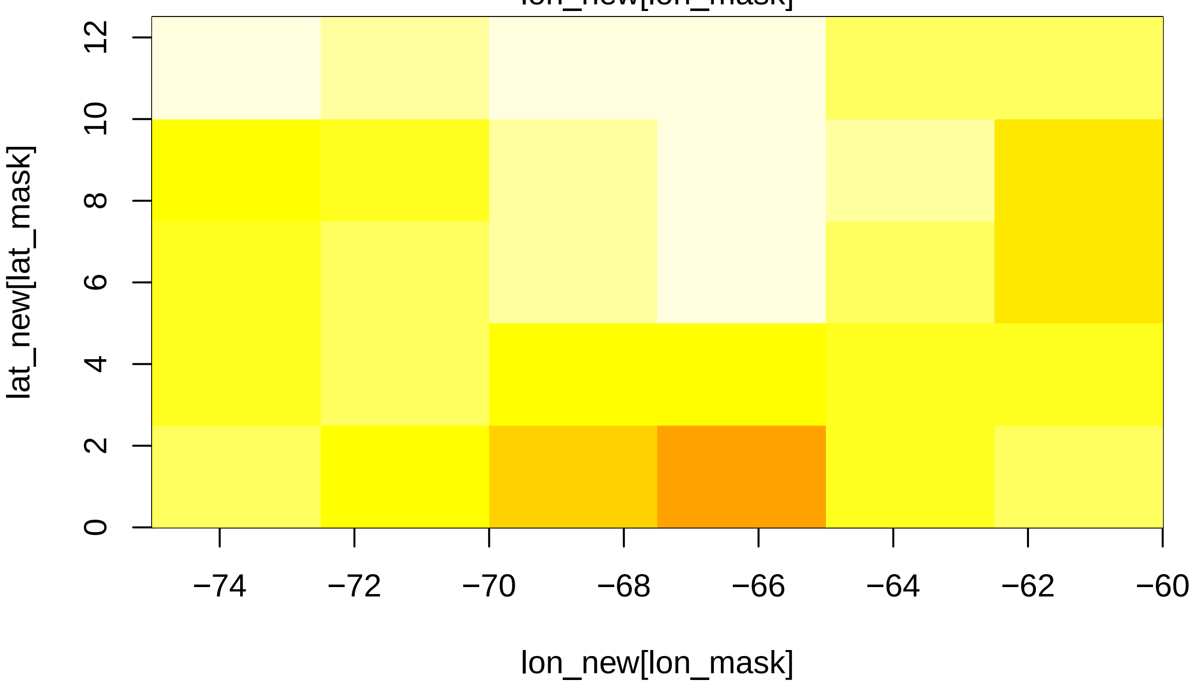
create boolean masks corresponding to region of interest

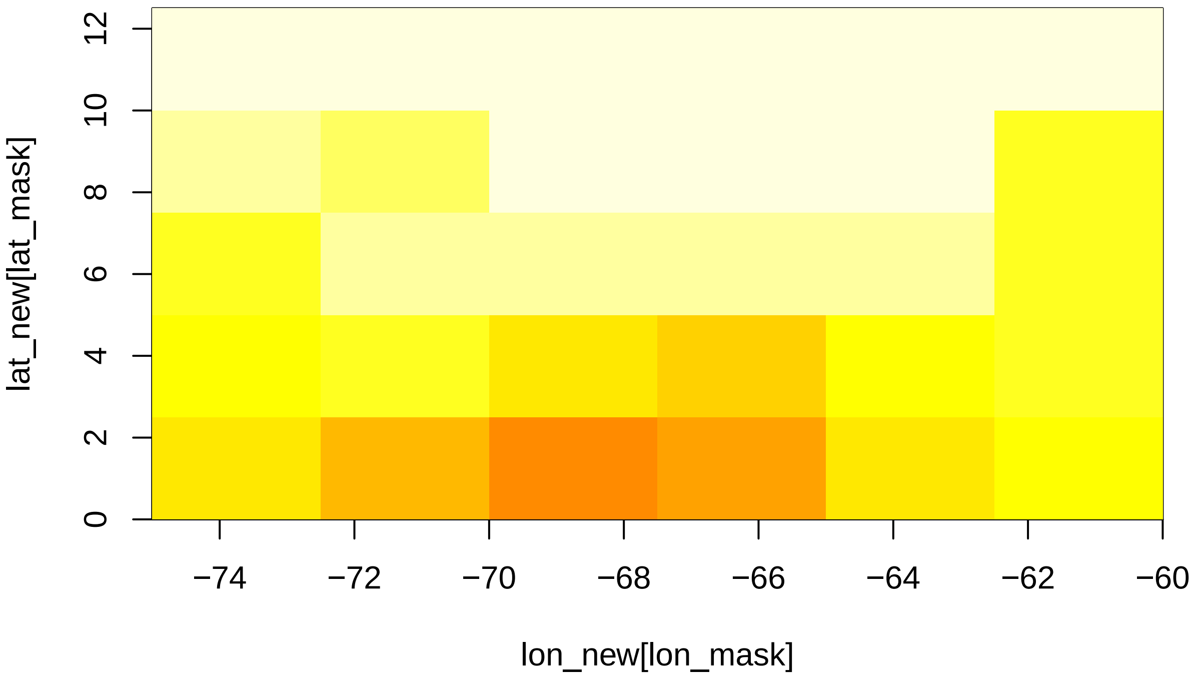
```
lon_mask <- -75<=lon_new & lon_new <=(-60) # need parentheses here because <- is the assignment operato
lat_mask <- 0<=lat_new & lat_new<=12
```

plot region of interest

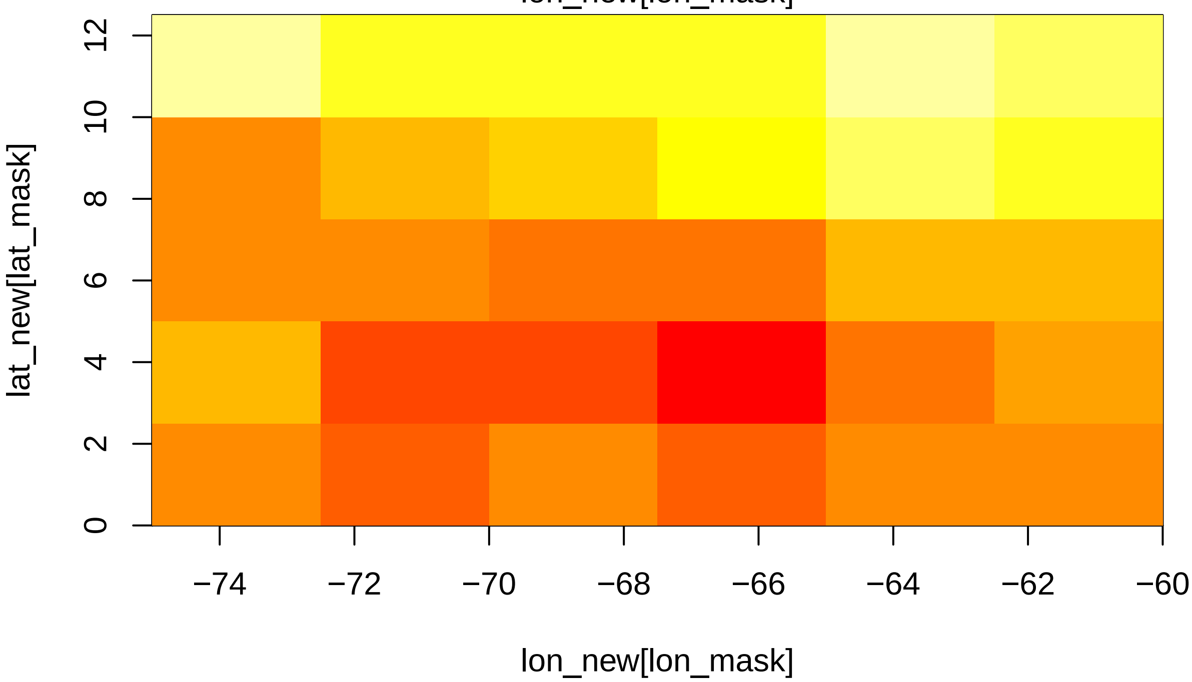
```
m<-1
for(m in 1:24){
image(lon_new[lon_mask], lat_new[lat_mask], precip_slice_new[lon_mask, lat_mask,m], col=rev(heat.colors
```



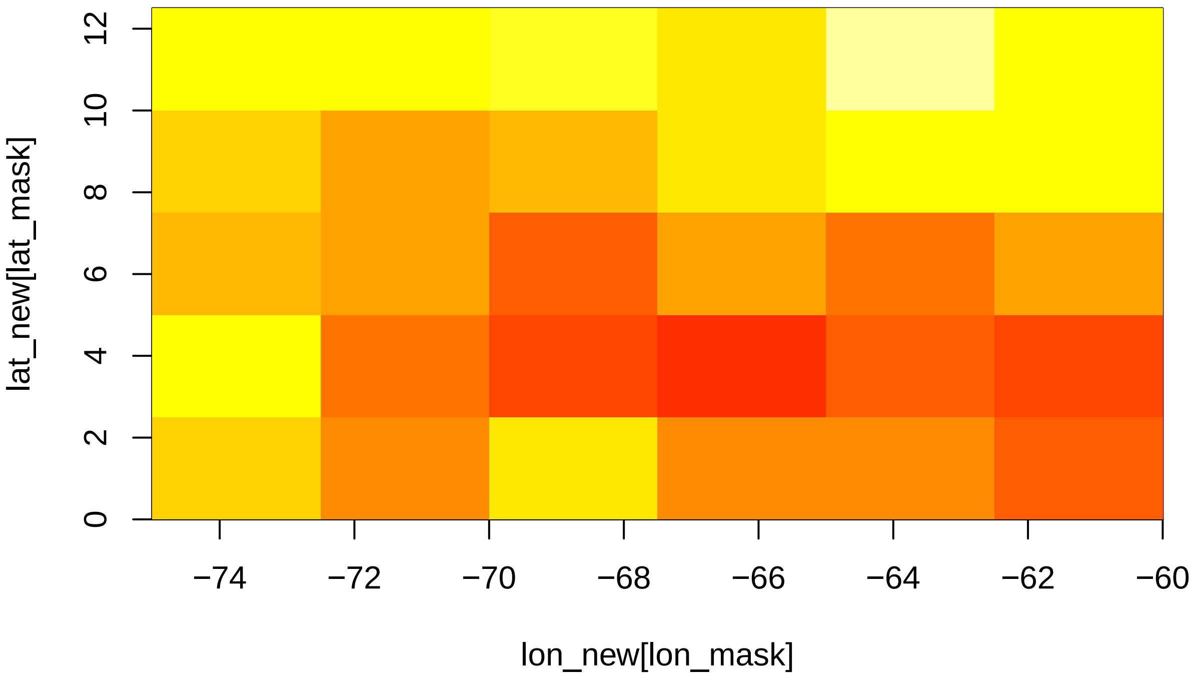




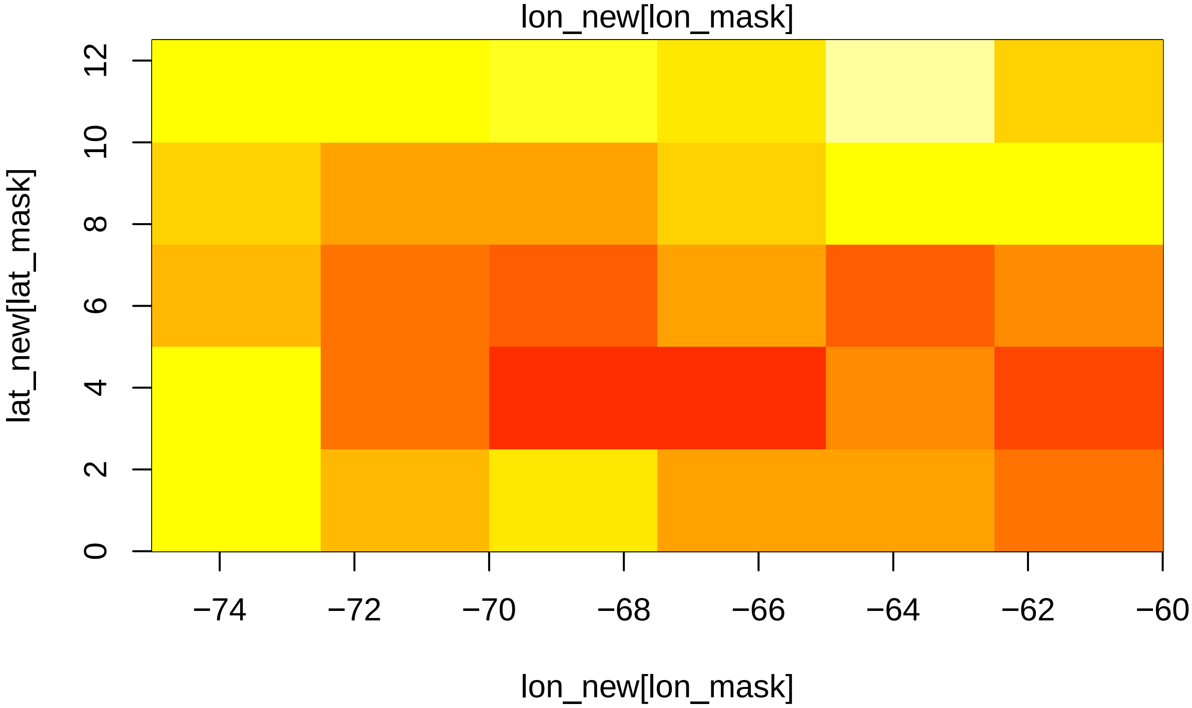
lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]



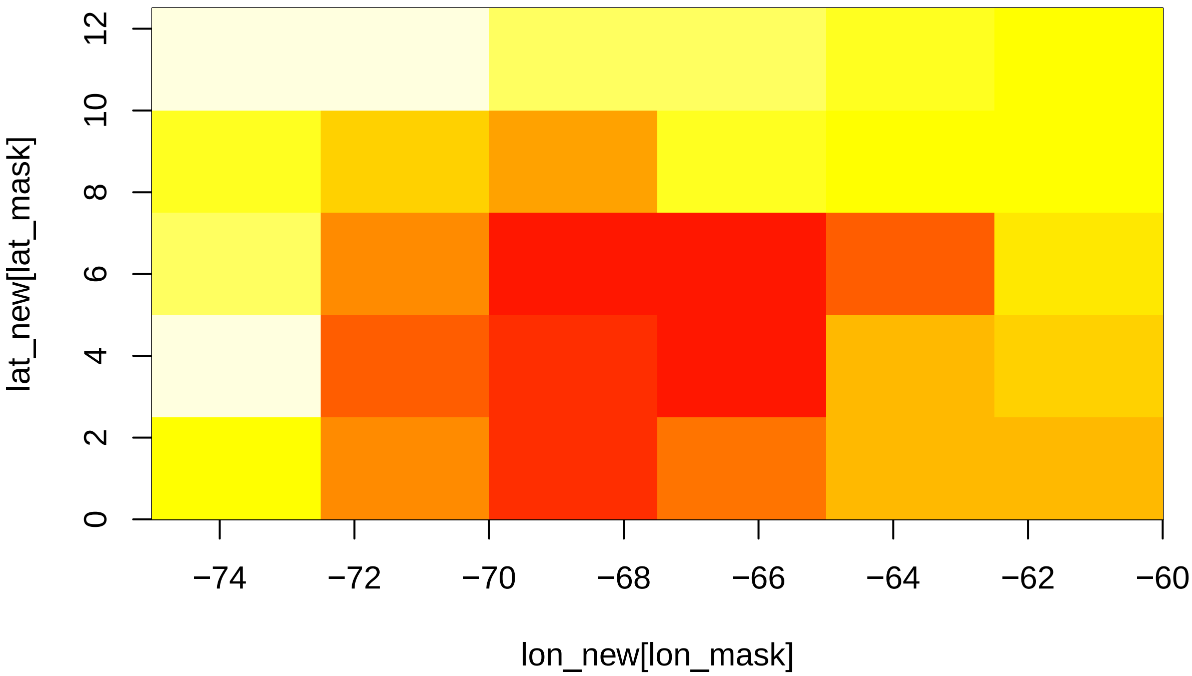
lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

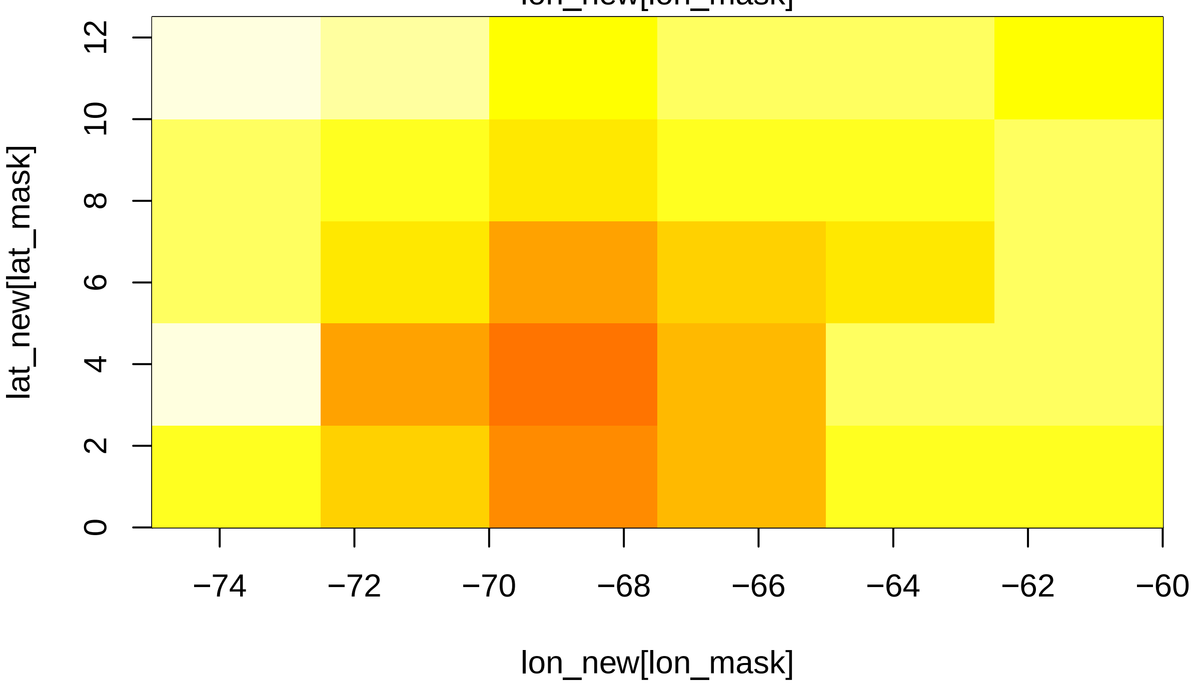


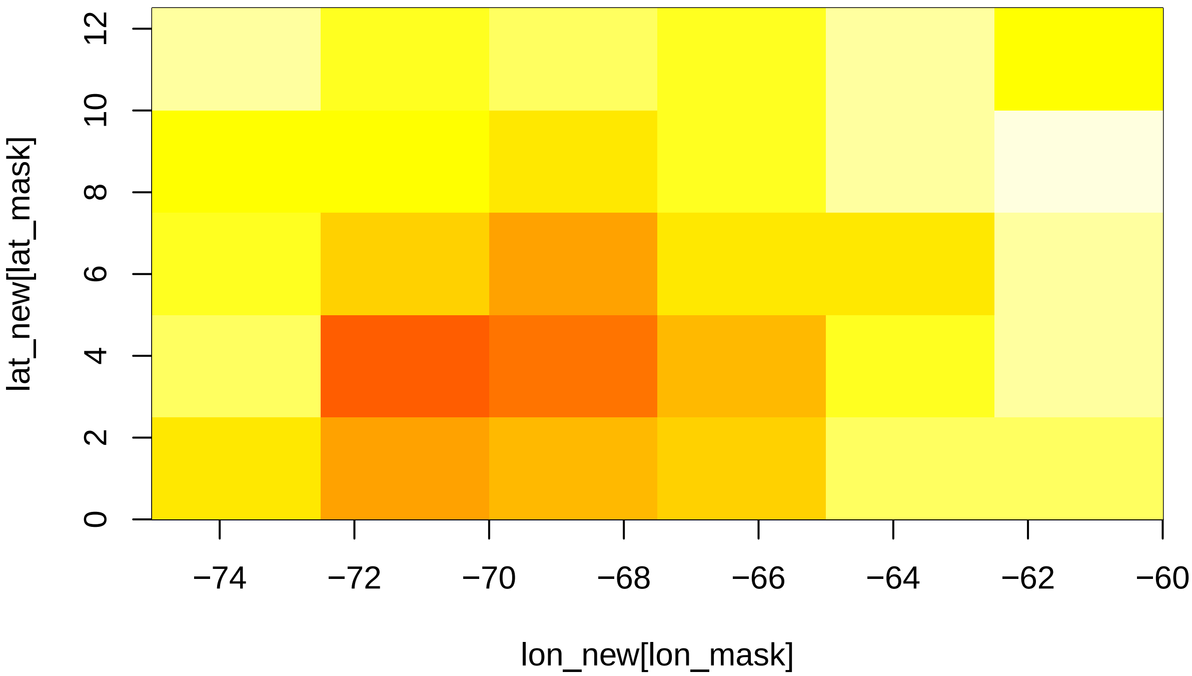
lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]



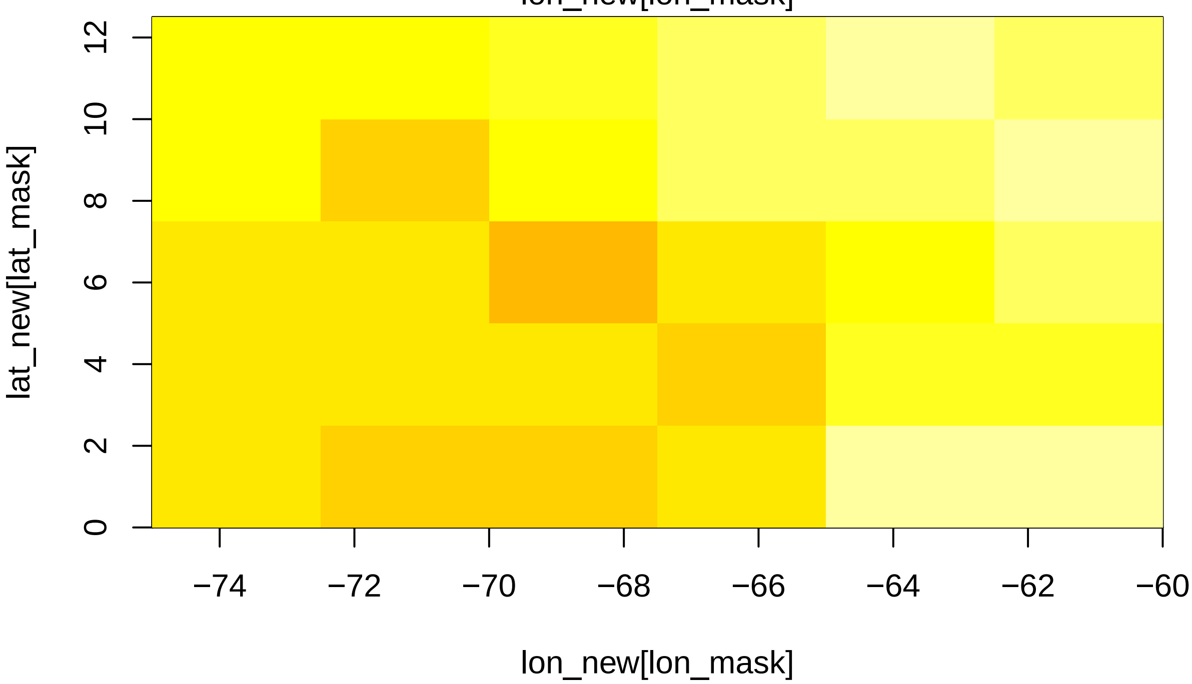
lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]


lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

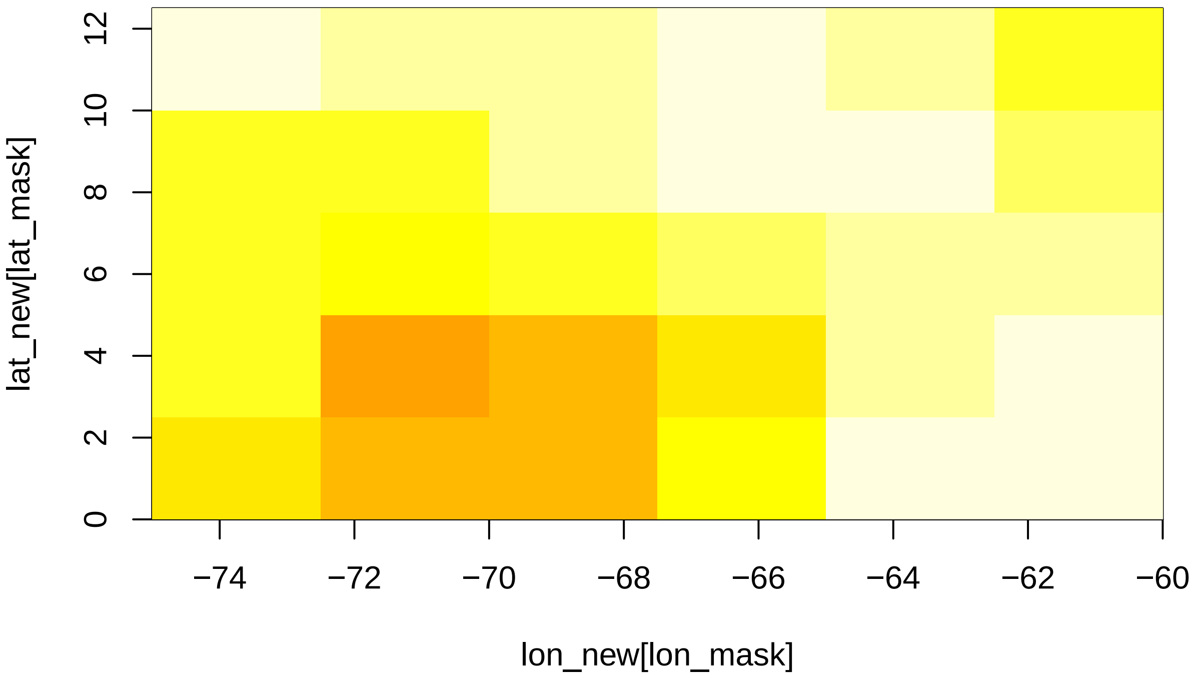

lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

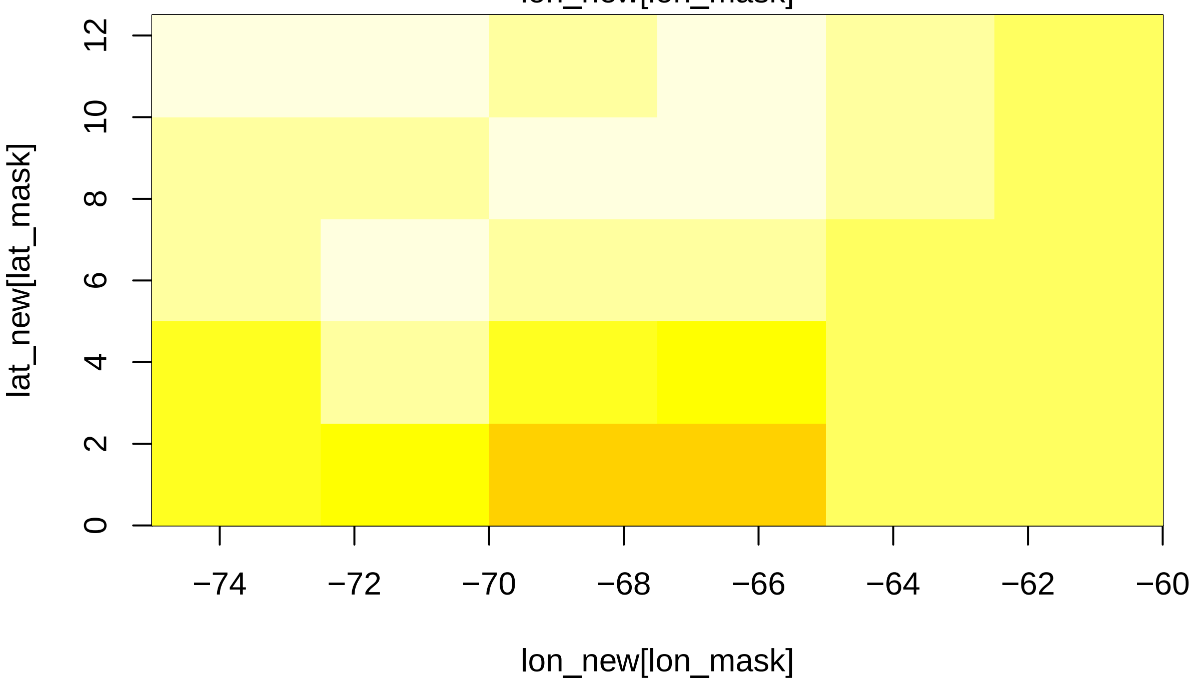


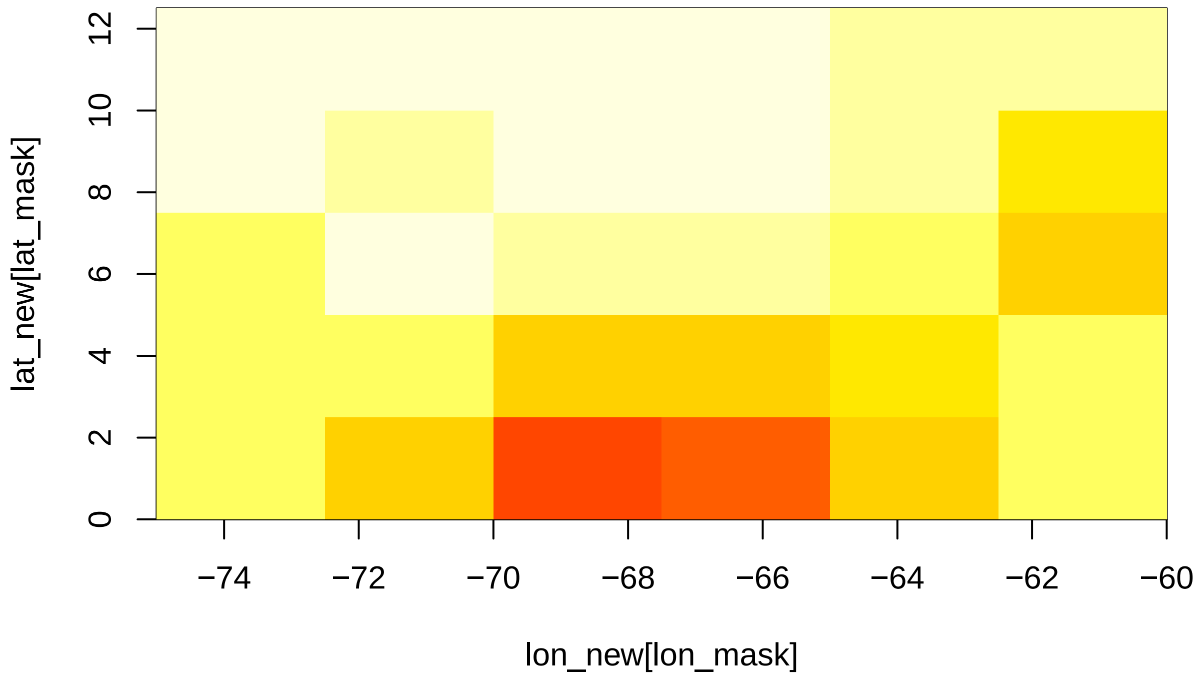
12
10
8
6
4
2
0
lat_new[lat_mask]
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

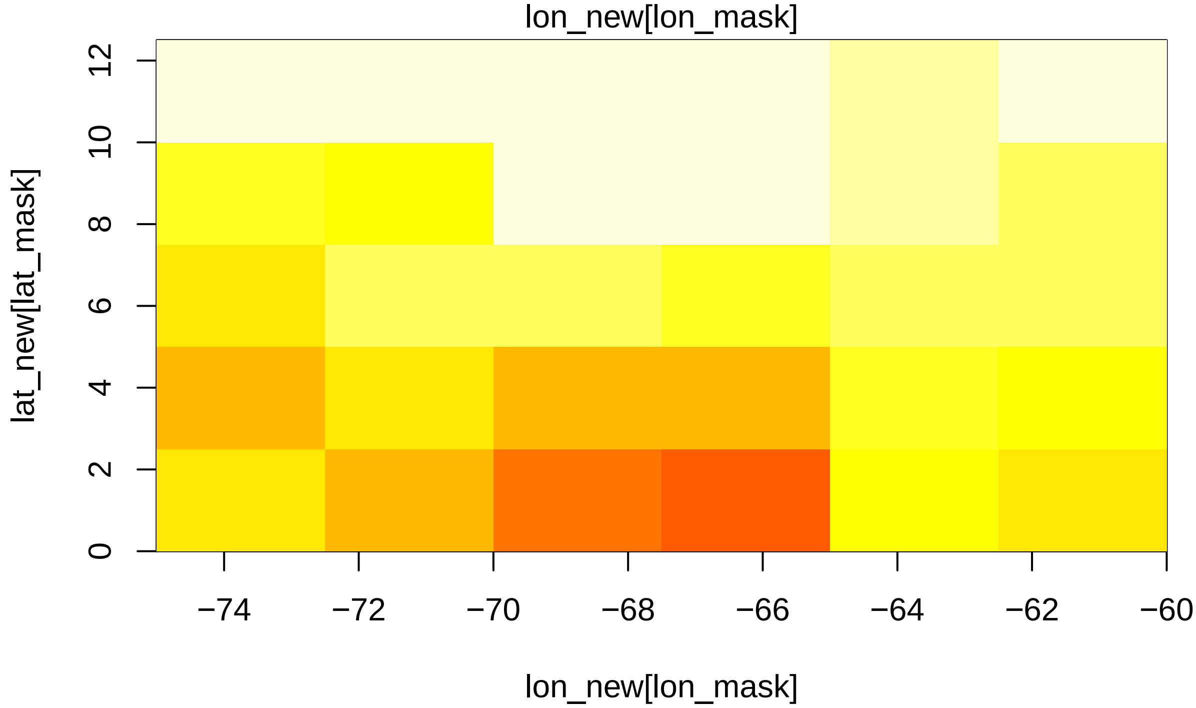


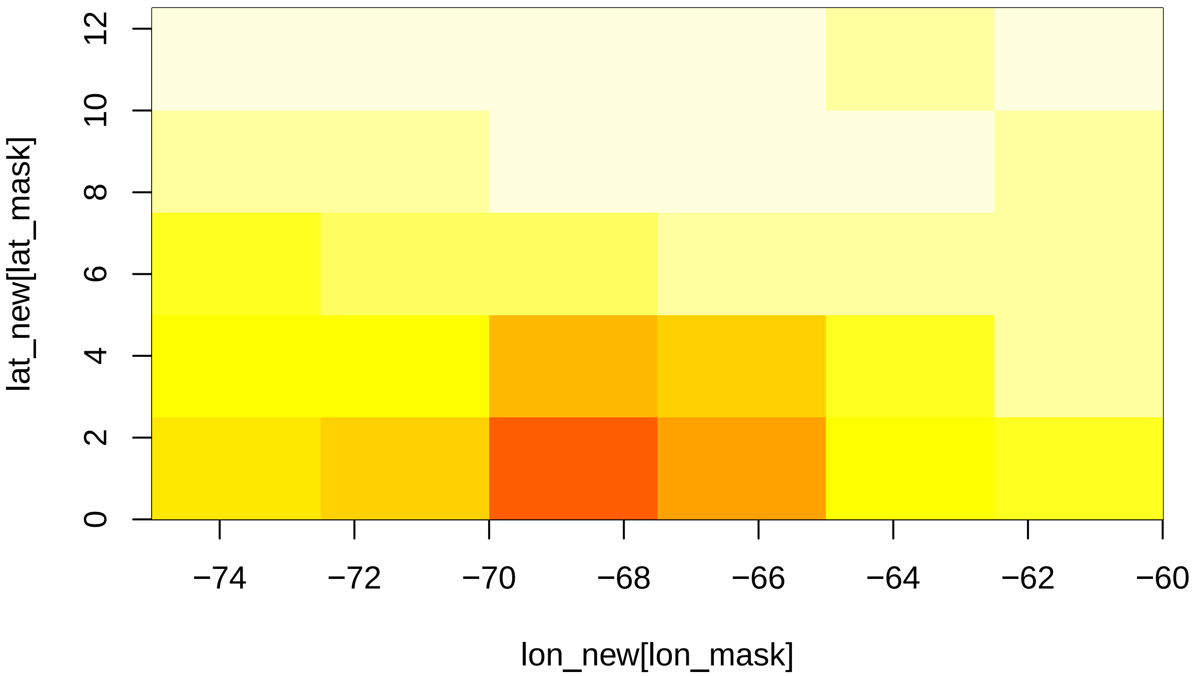
12
10
8
6
4
2
0
lat_new[lat_mask]
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

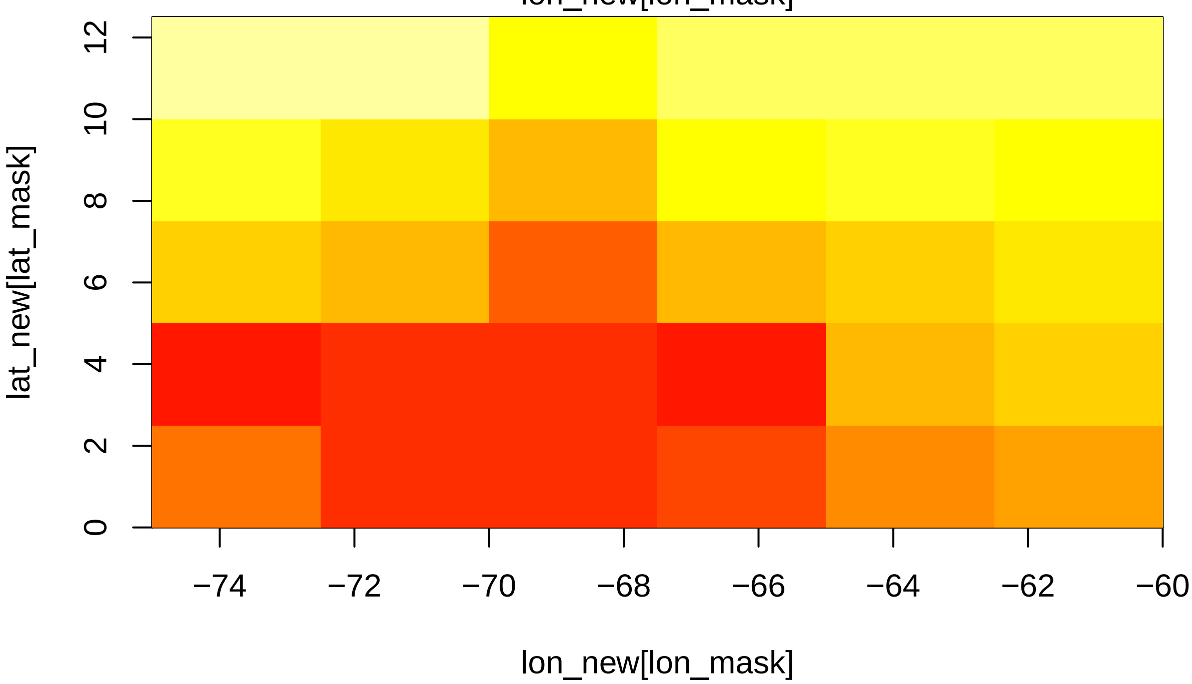

lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

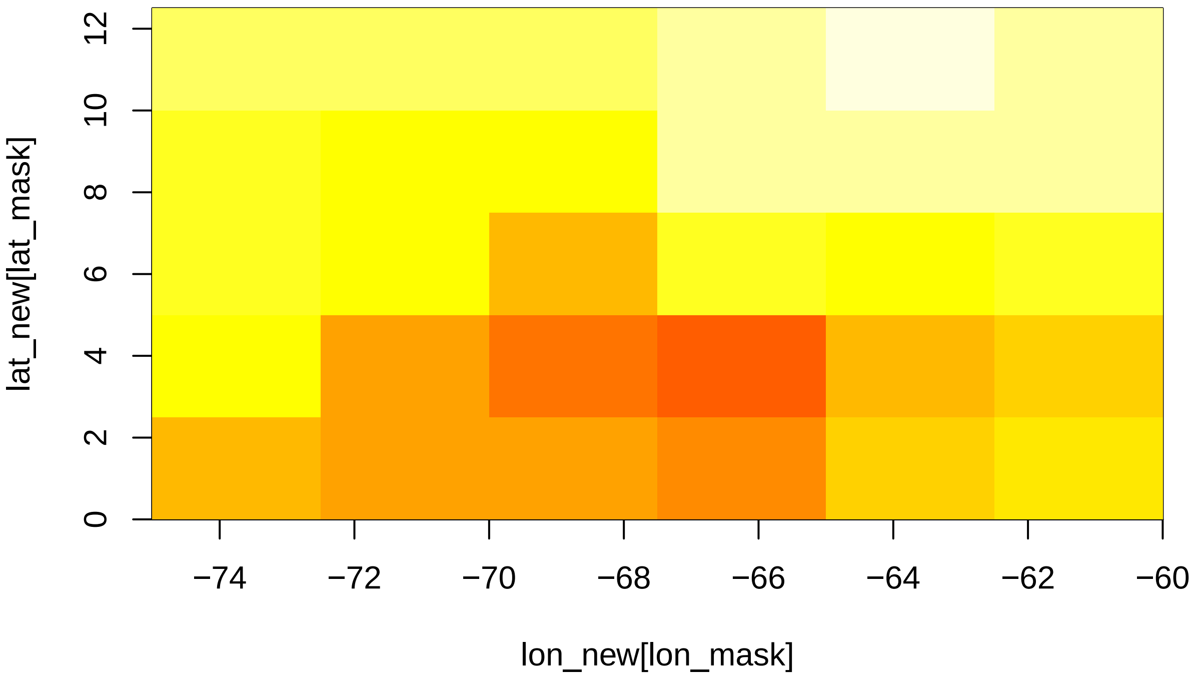

lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]


lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

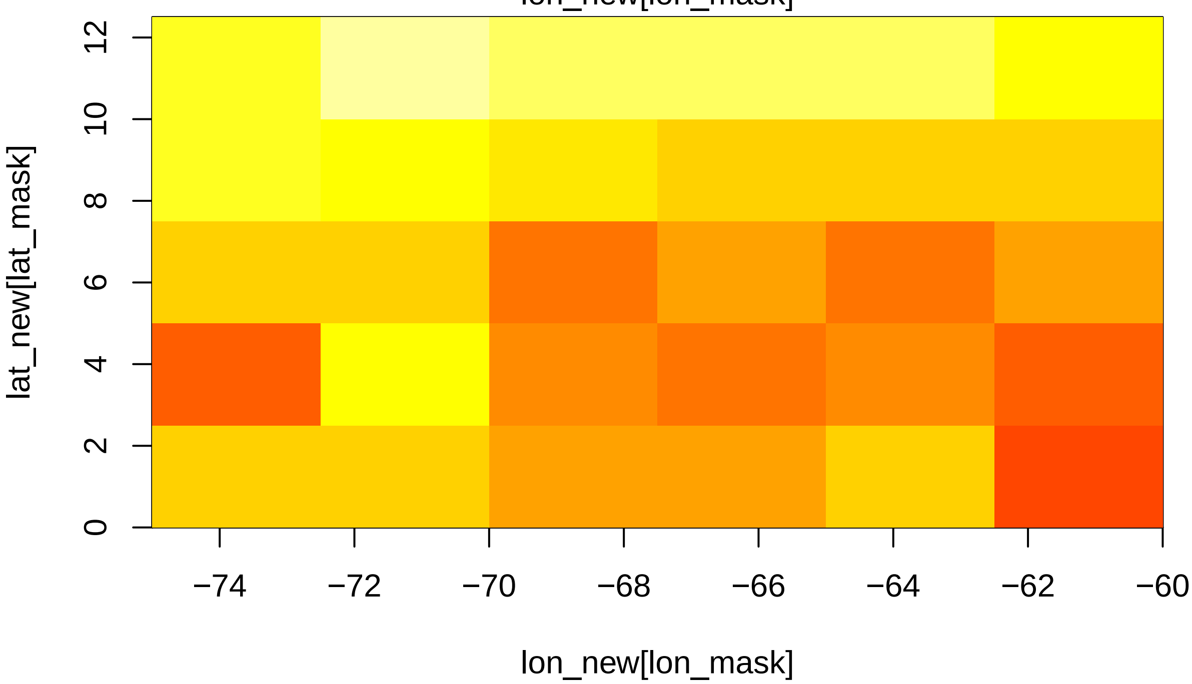

lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]


lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

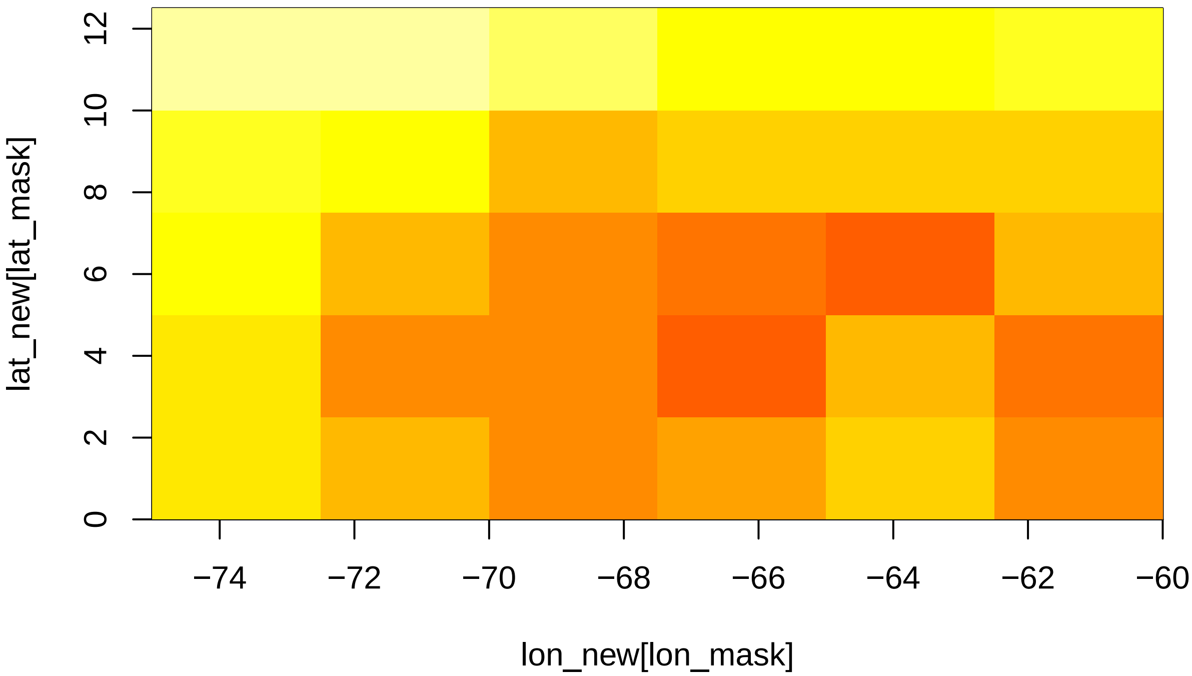

lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]



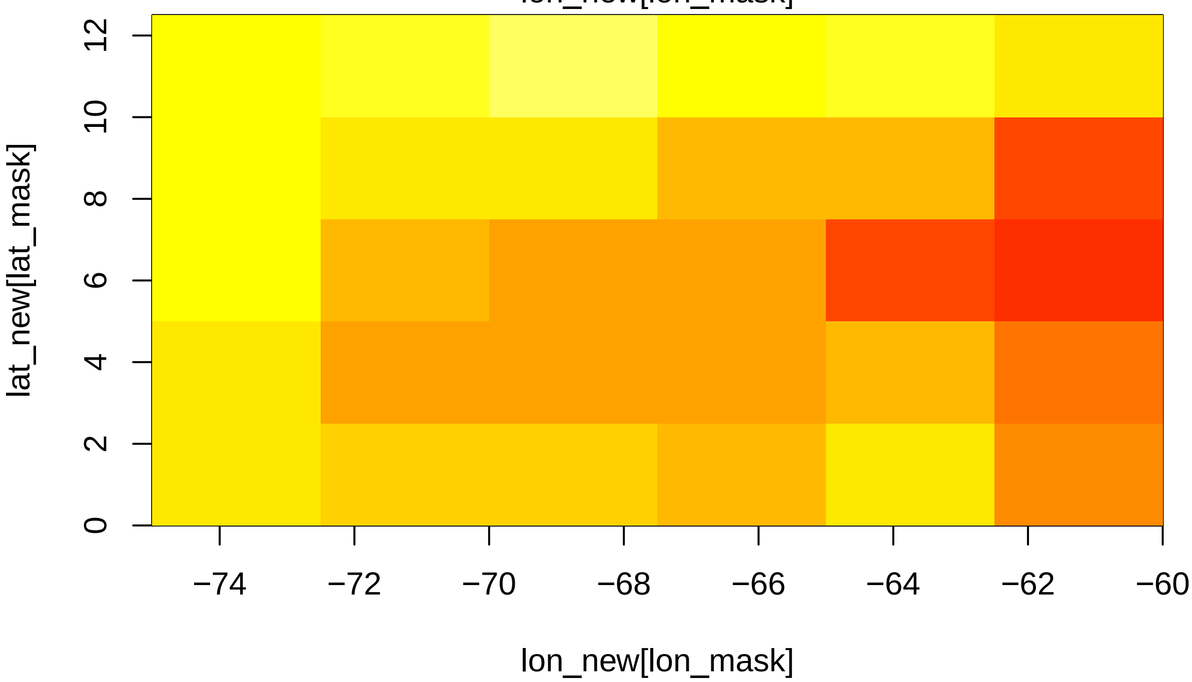
12
10
8
6
4
2
0
lat_new[lat_mask]
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]



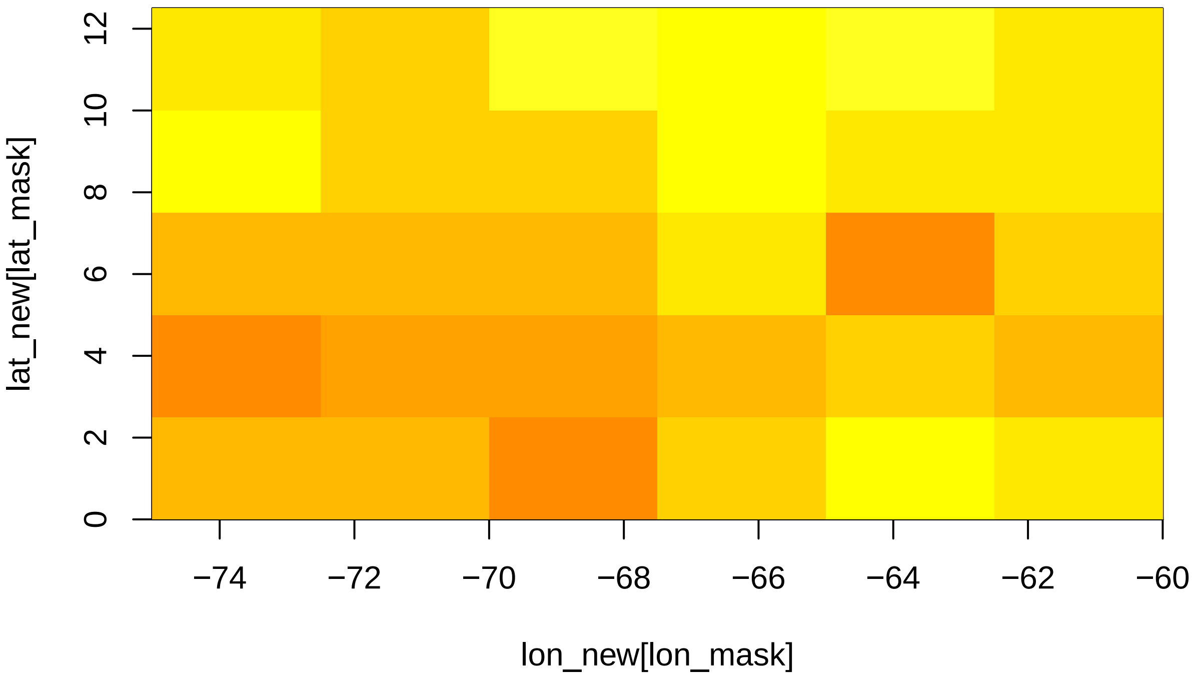
12
10
8
6
4
2
0
lat_new[lat_mask]
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

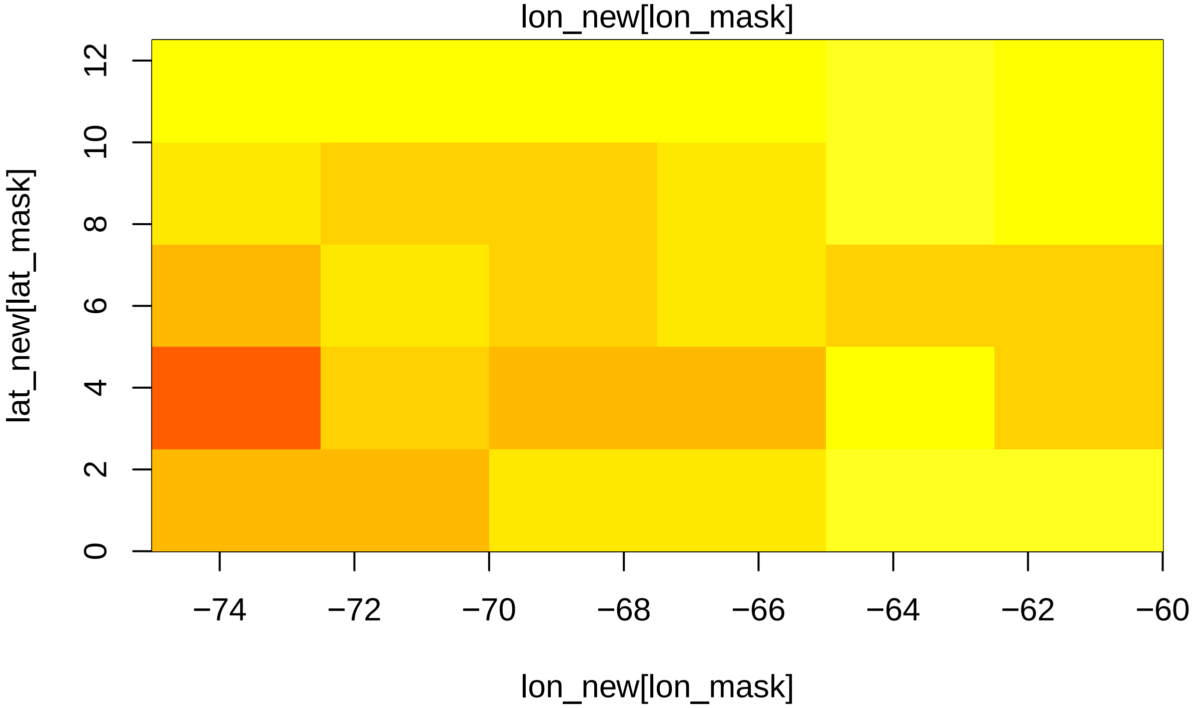


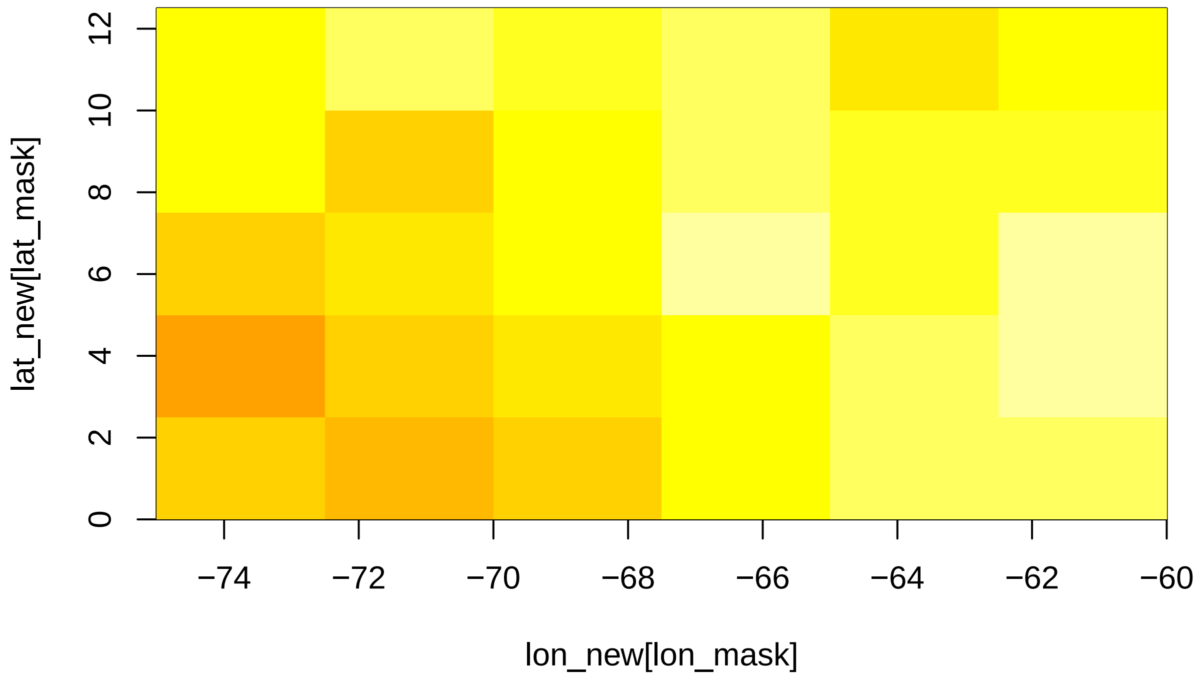
lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

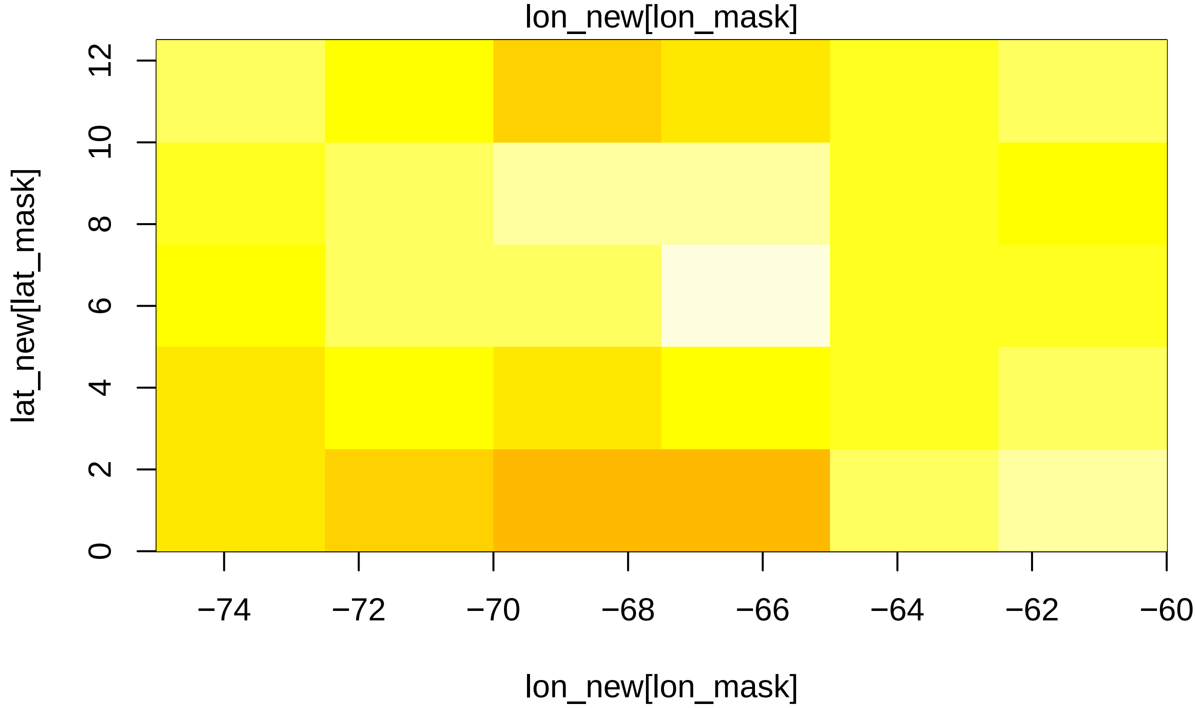


lat_new[lat_mask]
0
2
4
6
8
10
12
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]

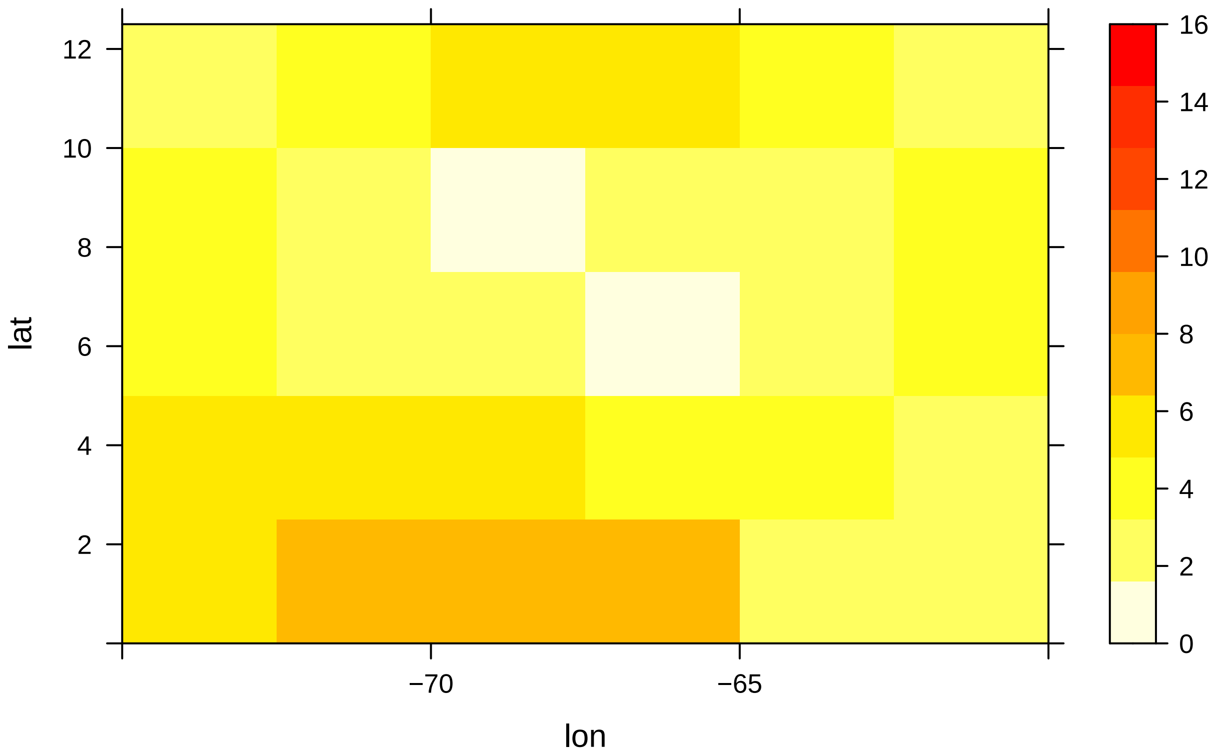

0
2
4
6
8
10
12
lat_new[lat_mask]
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]


0
2
4
6
8
10
12
lat_new[lat_mask]
−74
−72
−70
−68
−66
−64
−62
−60
lon_new[lon_mask]






```
#
m<-24
grid <- expand.grid(lon=lon_new[lon_mask], lat=lat_new[lat_mask])
grid$z<-as.vector(precip_slice_new[lon_mask, lat_mask,m])
cutpts <- seq(0, 16, length.out=11) #watch out for the limits here
levelplot(z ~ lon * lat, data=grid, at=cutpts, cuts=11, pretty=T,
  col.regions=(rev(heat.colors(16))))
```



create dataframe – reshape data

```
# matrix (nlon*nlat rows by 2 cols) of lons and lats
lonlat <- as.matrix(expand.grid(lon_new[lon_mask],lat_new[lat_mask]))
dim(lonlat)
```

```
## [1] 30  2
```

vector of precip values

```
prec_mat<-lonlat
for (m in 1:24) {
precip_vec <- as.vector(as.vector(precip_slice_new[lon_mask, lat_mask,m]))
prec_mat<-cbind(prec_mat,precip_vec)
}
colnames(prec_mat) <- c("lon","lat","Jan_1982","Feb_1982","Mar_1982","Apr_1982","May_1982","Jun_1982",
  "Jul_1982","Aug_1982","Sep_1982","Oct_1982","Nov_1982","Dec_1982","Jan_1999","Feb_1999","Mar_1999","Ap
  "Jul_1999","Aug_1999","Sep_1999","Oct_1999","Nov_1999","Dec_1999")
```

```
mat_stat<-apply(prec_mat[,3:26],2,summary)
mat_stat
```

```
##          Jan_1982 Feb_1982 Mar_1982  Apr_1982  May_1982  Jun_1982  Jul_1982
## Min.     0.090000 0.060000 0.010000  1.230000  1.790000  1.850000  0.670000
## 1st Qu.  0.867500 1.375000 0.920000  4.020000  4.847500  5.035000  3.282500
## Median   1.780000 2.640000 2.750000  8.705000  7.900000  7.685000  6.375000
## Mean     1.876667 2.946333 3.125667  7.765667  7.753333  7.790333  7.008333
## 3rd Qu.  2.565000 4.022500 4.622500 10.222500 10.025000 10.122500 10.227500
```

```
## Max.    6.100000 8.180000 9.240000 15.830000 13.520000 13.460000 14.430000
##          Aug_1982  Sep_1982 Oct_1982 Nov_1982 Dec_1982 Jan_1999 Feb_1999
## Min.     0.390000  0.780000 1.410000   0.1200 0.150000   0.0200   0.1800
## 1st Qu.  2.325000  2.635000 2.792500   1.0950 1.227500   1.0450   1.1450
## Median   3.765000  3.935000 4.690000   2.2300 2.090000   2.4250   3.1300
## Mean     4.411333  4.554333 4.270667   3.0440 2.321333   3.2960   3.8780
## 3rd Qu.  6.275000  6.010000 5.357500   4.1225 2.942500   5.4525   5.0875
## Max.    10.230000 11.250000 7.280000   8.4200 6.220000  12.1600  11.3100
##          Mar_1999  Apr_1999 May_1999  Jun_1999 Jul_1999 Aug_1999 Sep_1999
## Min.     0.280000  1.060000   0.2000  1.470000    1.540   2.6900   3.4700
## 1st Qu.  1.015000  4.370000   2.6475  4.742500    4.295   5.0950   5.1875
## Median   1.700000  6.960000   4.1150  6.680000    7.100   6.9250   6.2400
## Mean     3.033667  7.514667   5.0030  6.921333    6.821   7.1180   6.4470
## 3rd Qu.  4.130000 11.052500   7.1275  9.387500    9.010   8.5725   7.4350
## Max.    11.090000 14.030000  11.1200 12.140000   11.610  13.6400   9.7000
##         Oct_1999 Nov_1999 Dec_1999
## Min.      3.3100 1.330000 0.570000
## 1st Qu.   4.7050 2.960000 2.647500
## Median    5.7250 4.395000 3.700000
## Mean      5.8120 4.380667 3.887333
## 3rd Qu.   6.6975 5.950000 5.055000
## Max.     11.2600 8.770000 7.260000
```

```
x = prec_mat[,1]
y = prec_mat[,2]
trend_one = lm(prec_mat[,3]~x+y)
trend_two = lm(prec_mat[,3]~x+y+I(x^2)+I(y^2)+I(x*y))
summary(trend_one)
```

```
##
## Call:
## lm(formula = prec_mat[, 3] ~ x + y)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.81472 -0.62842 -0.07786  0.54036  2.94961
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.94884    3.03229   1.632    0.114
## x            0.02245    0.04456   0.504    0.619
## y           -0.24913    0.05381  -4.630 8.23e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.042 on 27 degrees of freedom
## Multiple R-squared:  0.4455, Adjusted R-squared:  0.4044
## F-statistic: 10.85 on 2 and 27 DF,  p-value: 0.000349
```

```
summary(trend_two)
```

```
##
## Call:
## lm(formula = prec_mat[, 3] ~ x + y + I(x^2) + I(y^2) + I(x *
##     y))
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4678 -0.4733 -0.1420  0.4805  2.5614
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 48.979927  52.513681   0.933   0.3603
## x            1.269103   1.555038   0.816   0.4225
## y           -1.090080   0.831506  -1.311   0.2023
## I(x^2)       0.009006   0.011501   0.783   0.4413
## I(y^2)       0.040590   0.017145   2.367   0.0263 *
## I(x * y)    -0.004942   0.011879  -0.416   0.6811
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9821 on 24 degrees of freedom
## Multiple R-squared:  0.5621, Adjusted R-squared:  0.4709
## F-statistic: 6.161 on 5 and 24 DF,  p-value: 0.00083
```

calculating statistics for full year

```
summary(as.vector(prec_mat[, 3:14]))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.010   2.040   3.880   4.739   6.897  15.830
```

```
summary(as.vector(prec_mat[, 15:26]))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.020   2.908   5.075   5.343   7.162  14.030
```

```
sd(as.vector(prec_mat[, 3:14]))
```

```
## [1] 3.456582
```

```
sd(as.vector(prec_mat[, 15:26]))
```

```
## [1] 3.119523
```

calculate an empirical semi-variogram

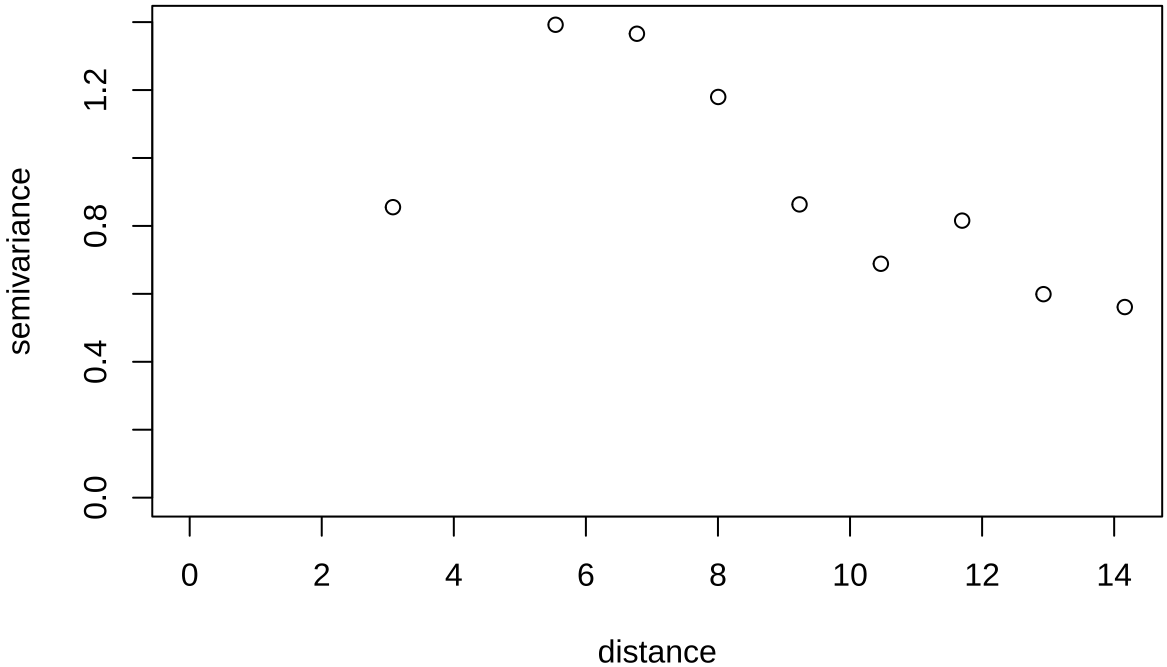
```
library(geoR)
```

```
## --------------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.8-1 (built on 2020-02-08) is now loaded
## --------------------------------------------------------------
```

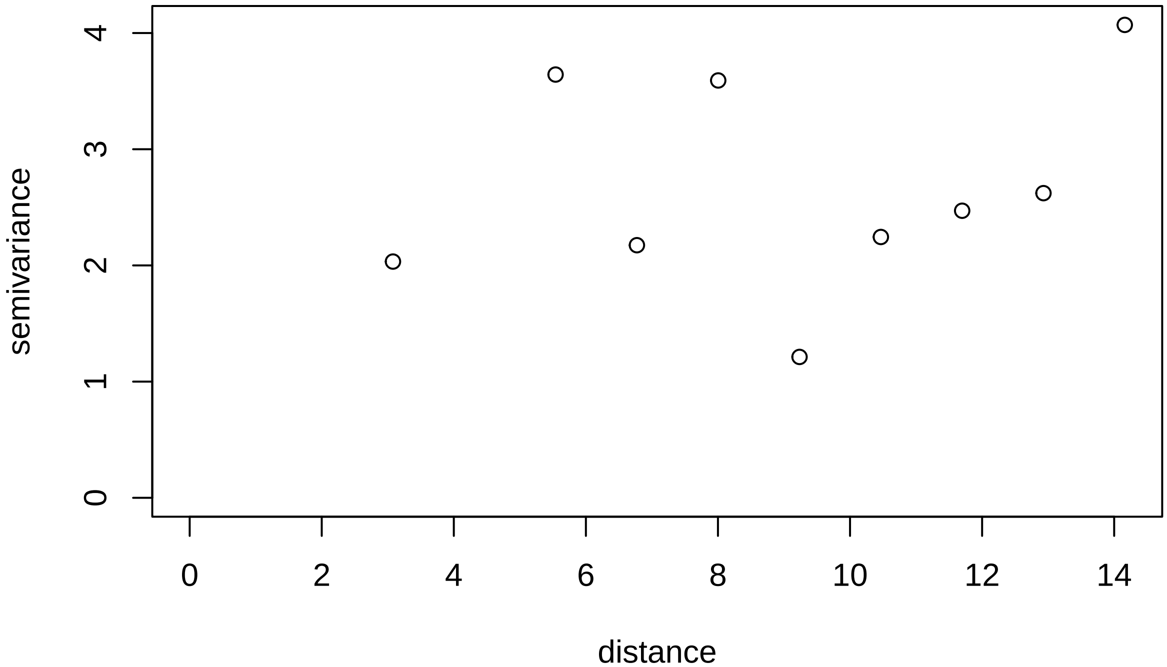
```
geo_precip_list = list()
empirical = list()
for(m in 3:26) {
    atrend="1st"
    geo_precip = as.geodata(prec_mat, coords.col = 1:2, data.col = m)
```

```
    geo_precip_list[[m-2]] = geo_precip
    if (m == 24) atrend="2nd"
    empirical[[m-2]] = variog(geo_precip, trend = atrend)
    plot(empirical[[m-2]])
}
```

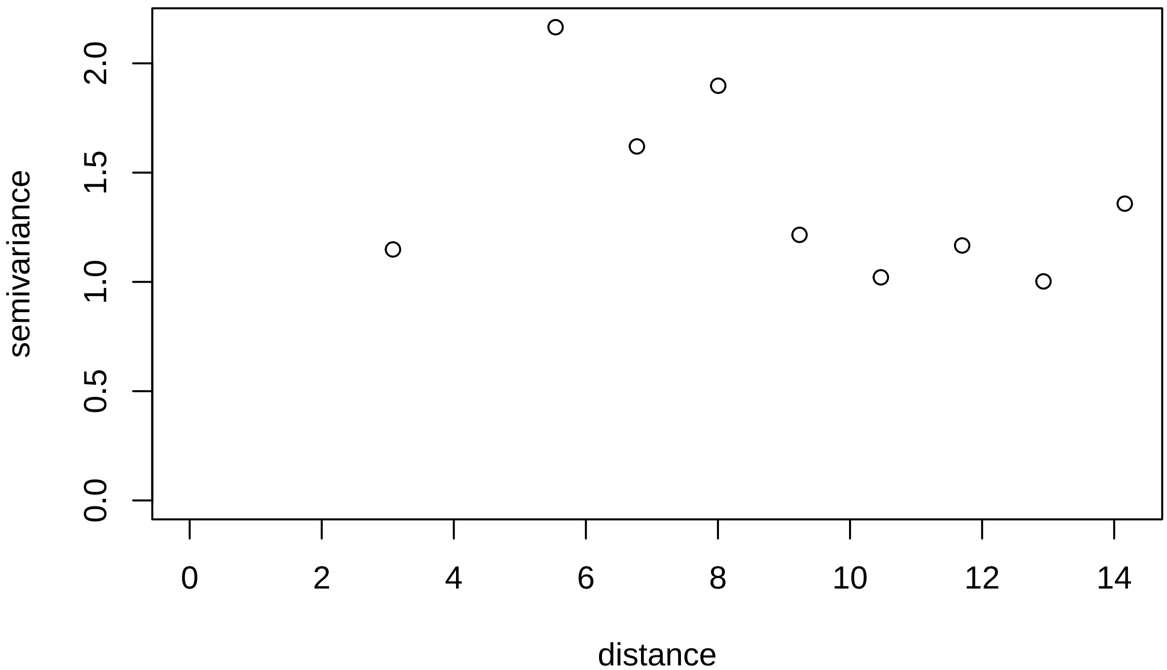
```
## variog: computing omnidirectional variogram
```



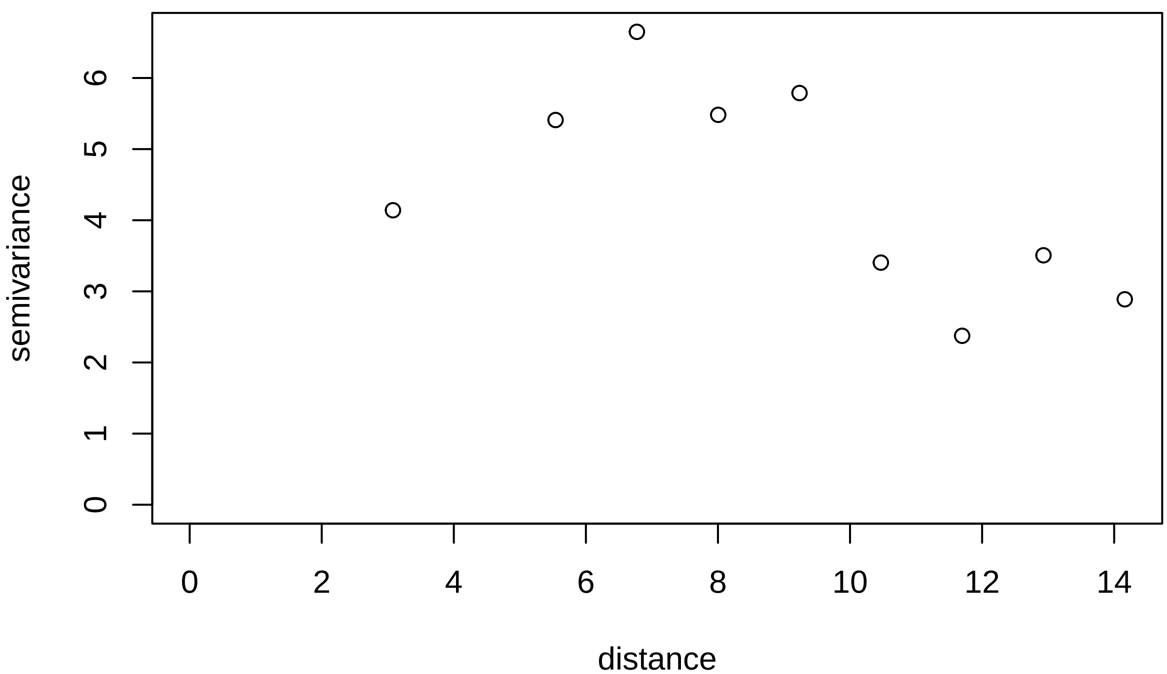

```
## variog: computing omnidirectional variogram
```



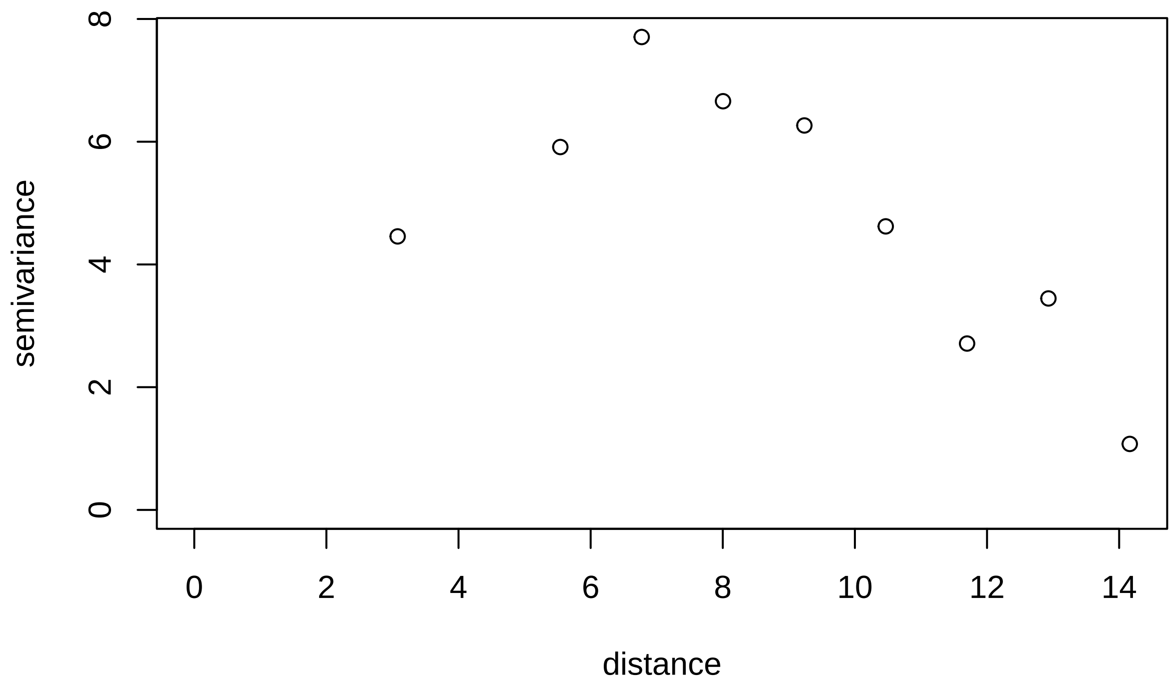

```
## variog: computing omnidirectional variogram
```



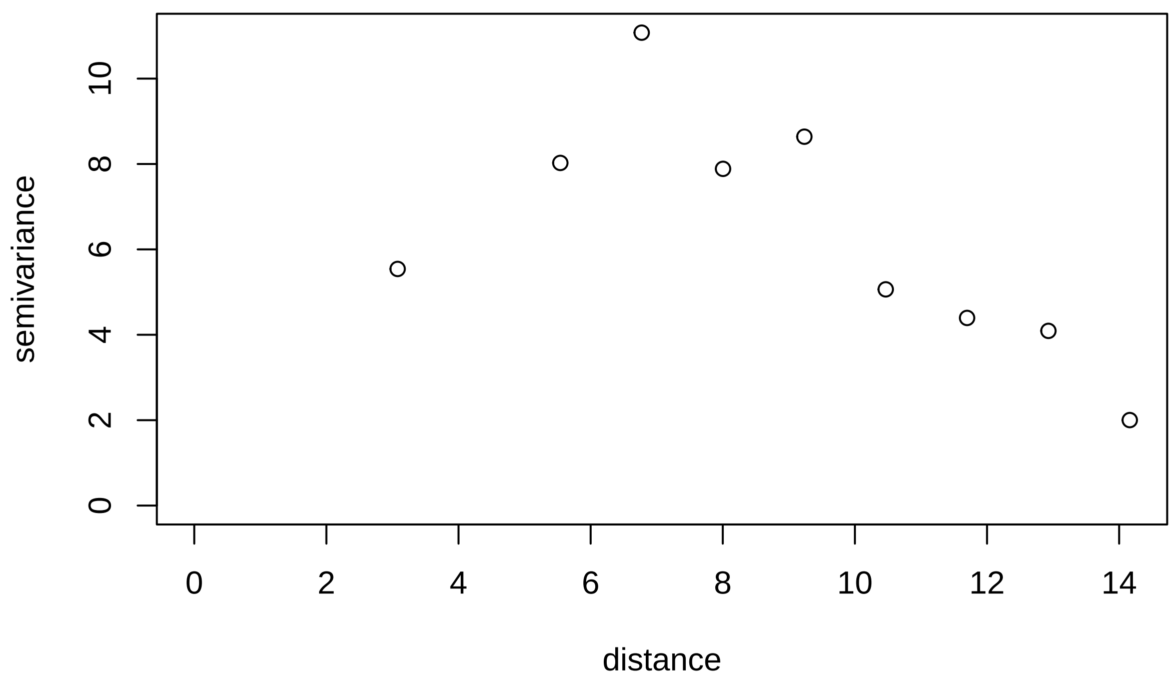

```
## variog: computing omnidirectional variogram
```



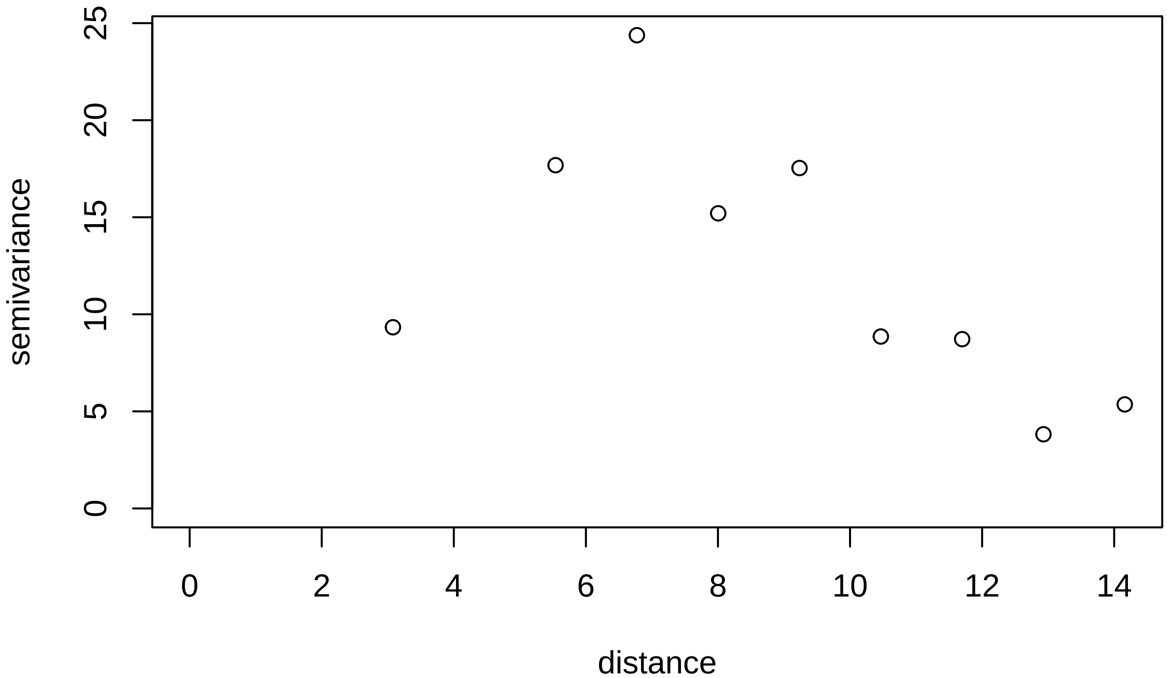

```
## variog: computing omnidirectional variogram
```



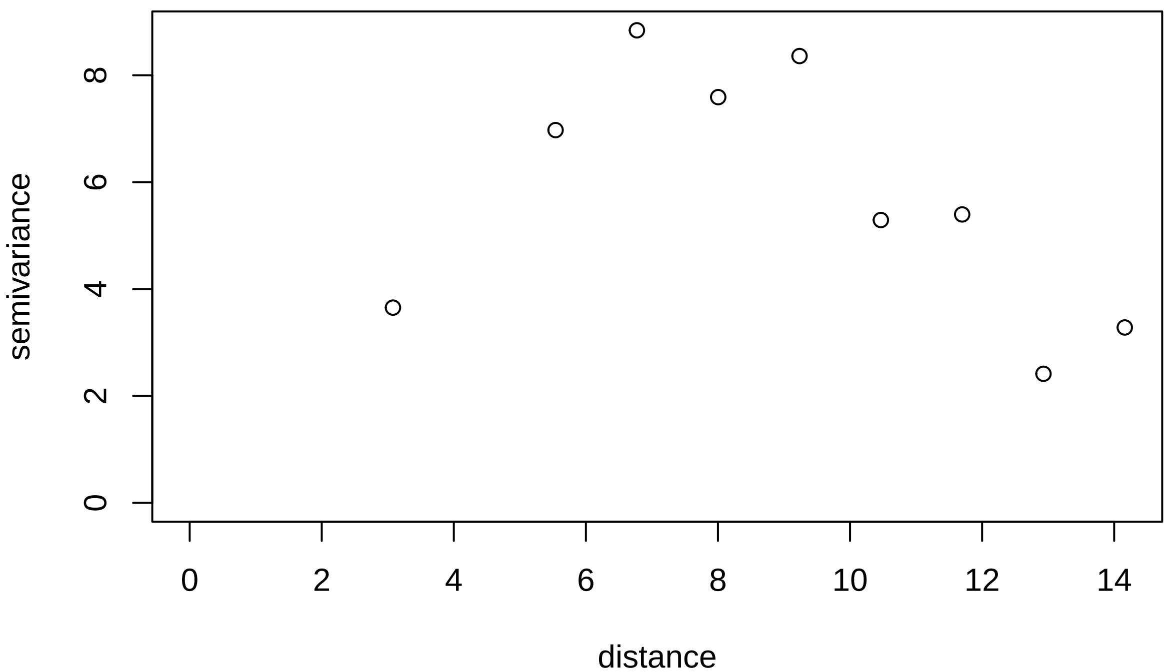

```
## variog: computing omnidirectional variogram
```



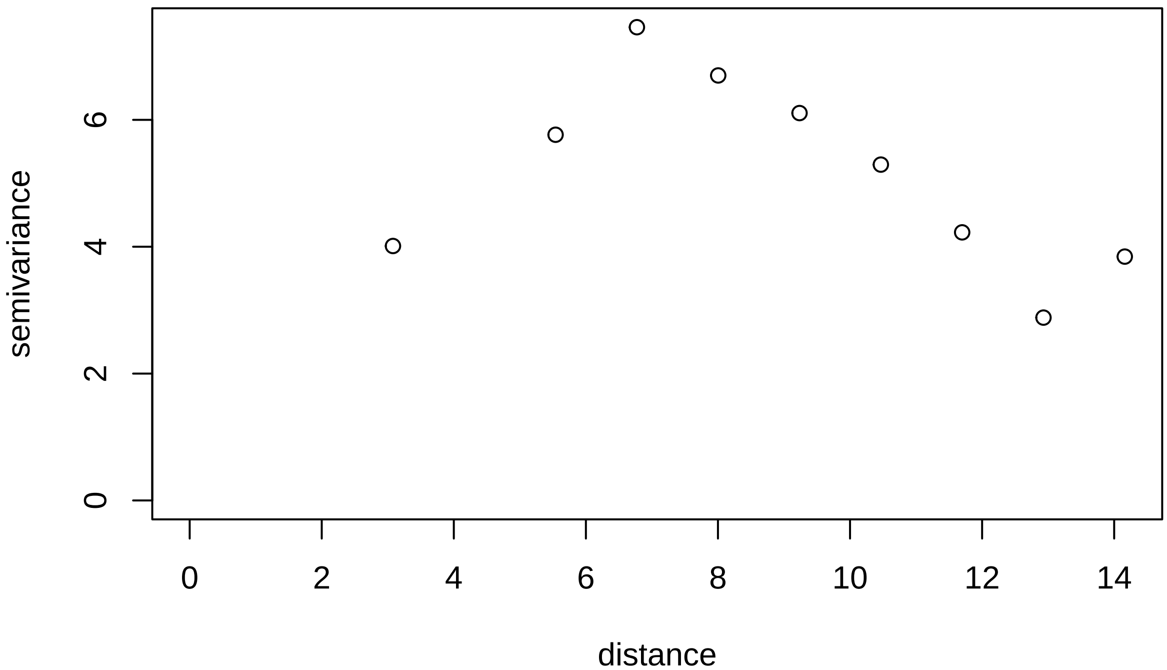

```
## variog: computing omnidirectional variogram
```



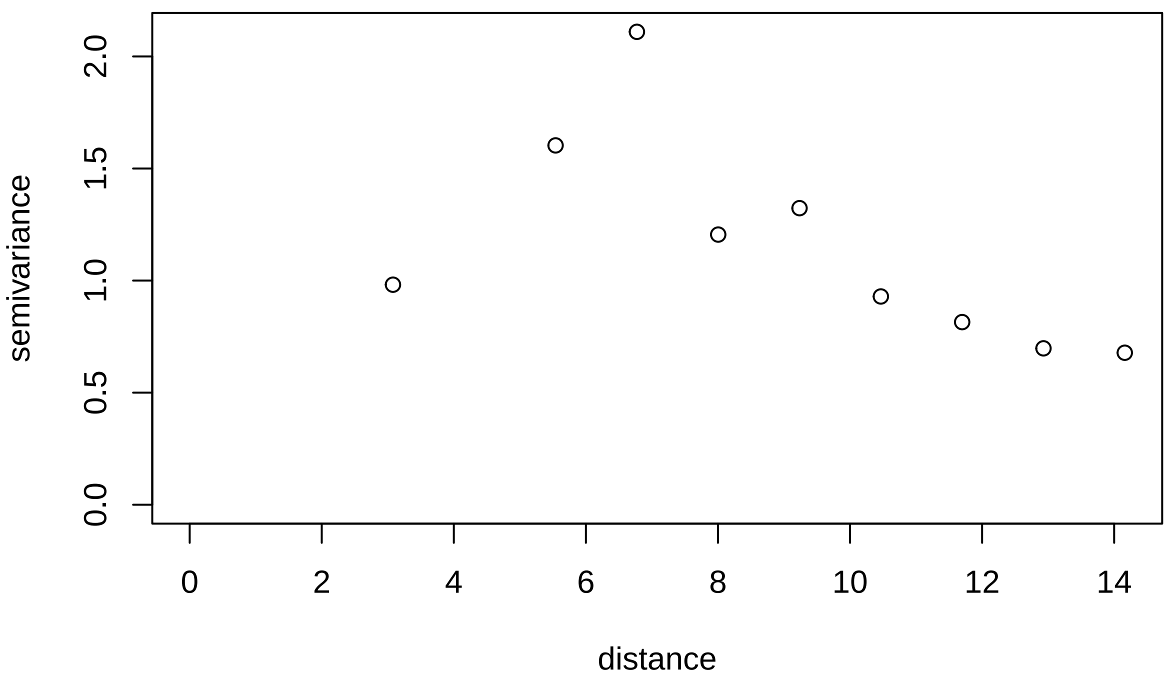

```
## variog: computing omnidirectional variogram
```



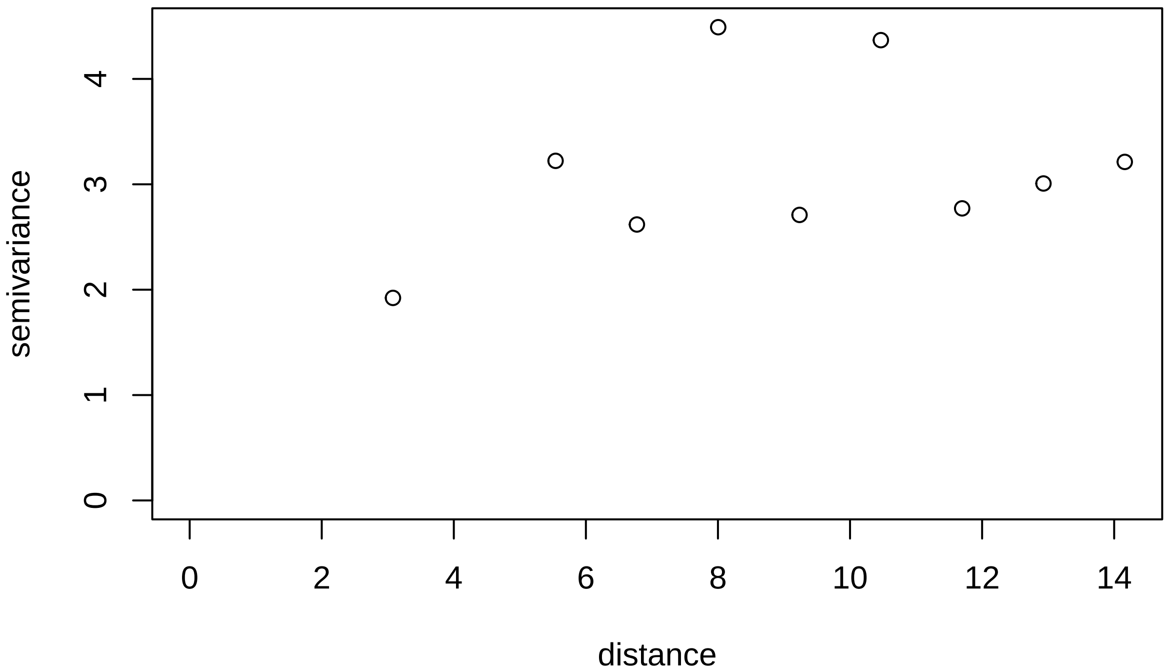

```
## variog: computing omnidirectional variogram
```



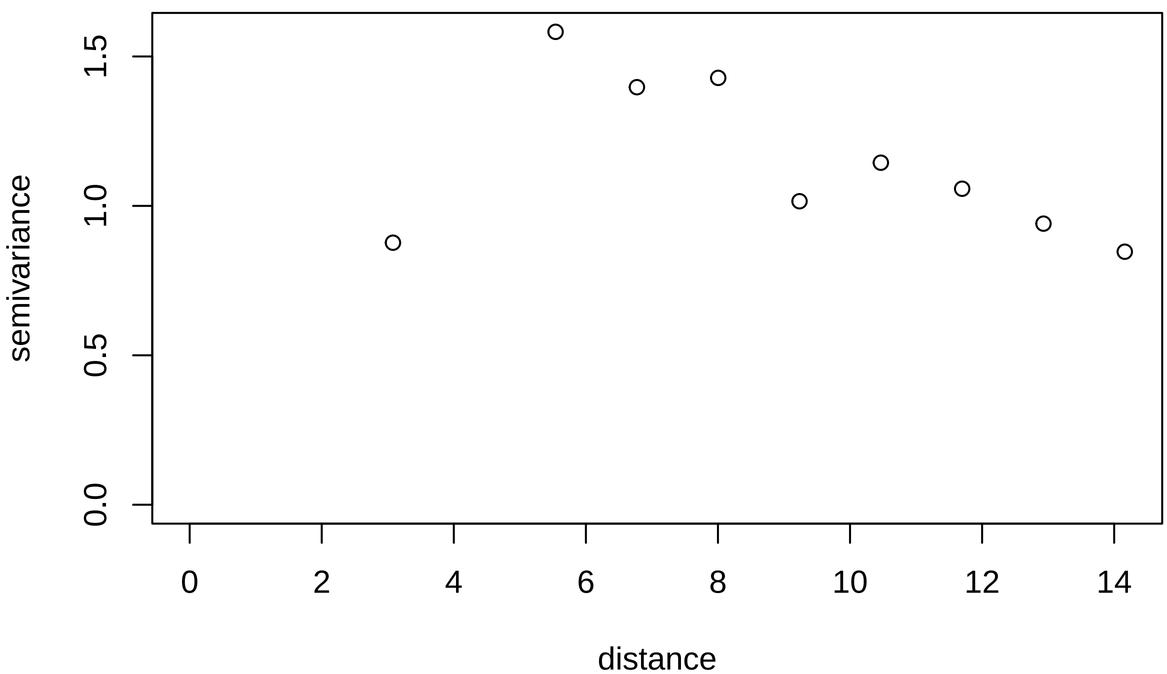

```
## variog: computing omnidirectional variogram
```



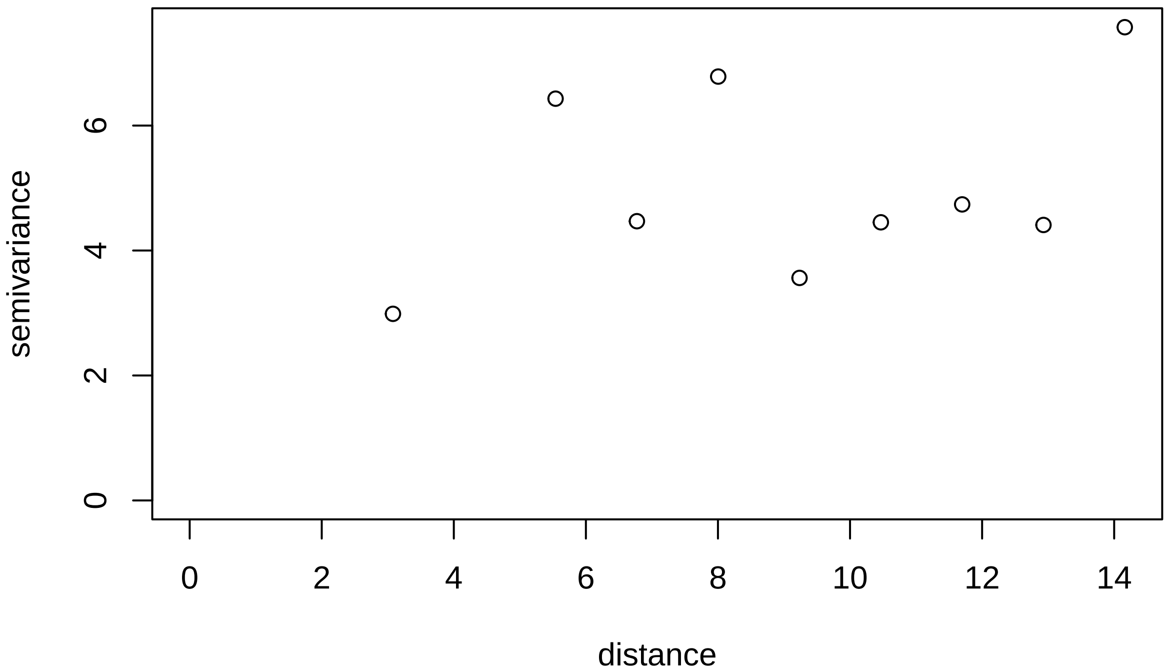

```
## variog: computing omnidirectional variogram
```



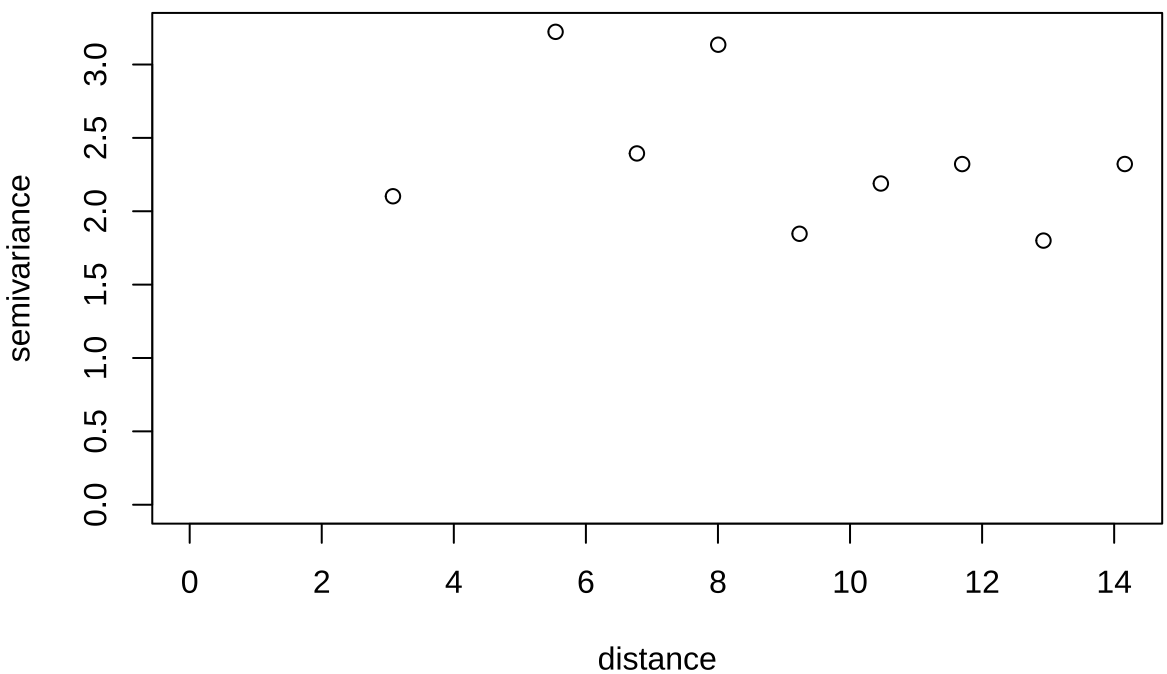

```
## variog: computing omnidirectional variogram
```



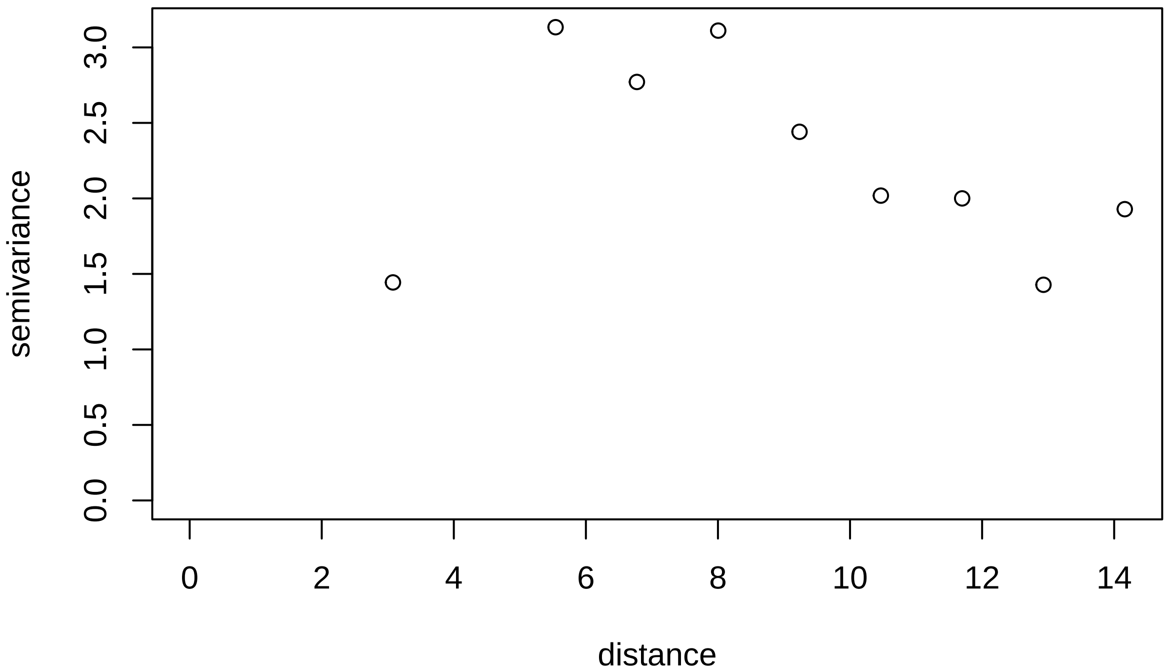

```
## variog: computing omnidirectional variogram
```



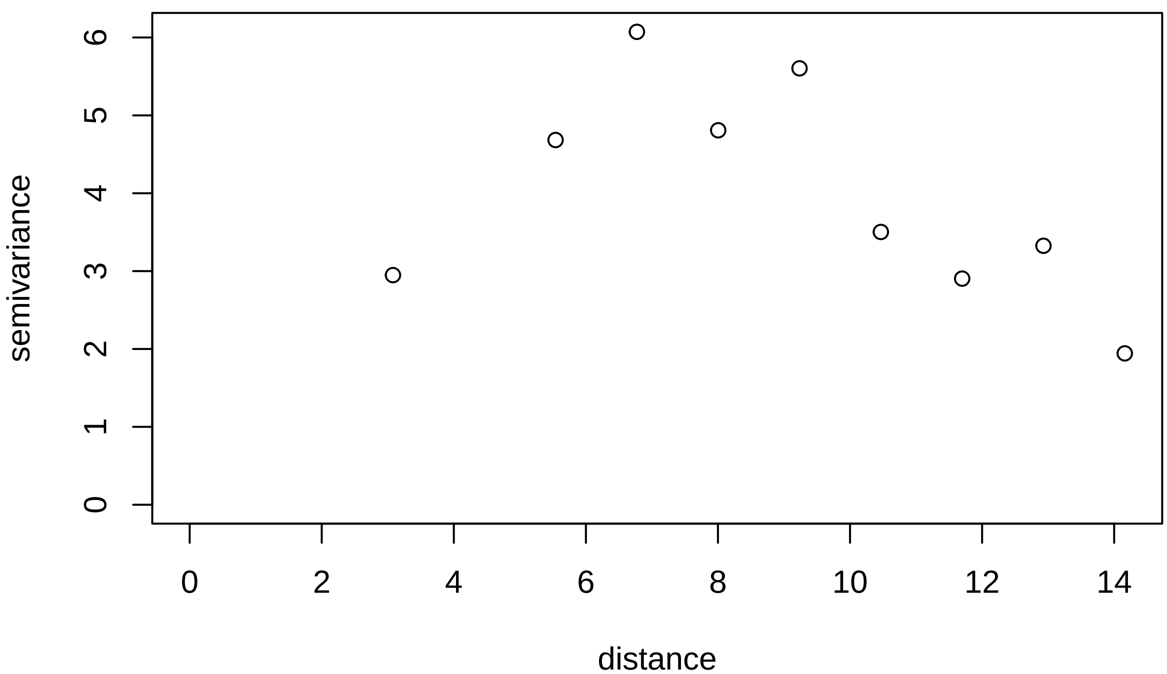

```
## variog: computing omnidirectional variogram
```



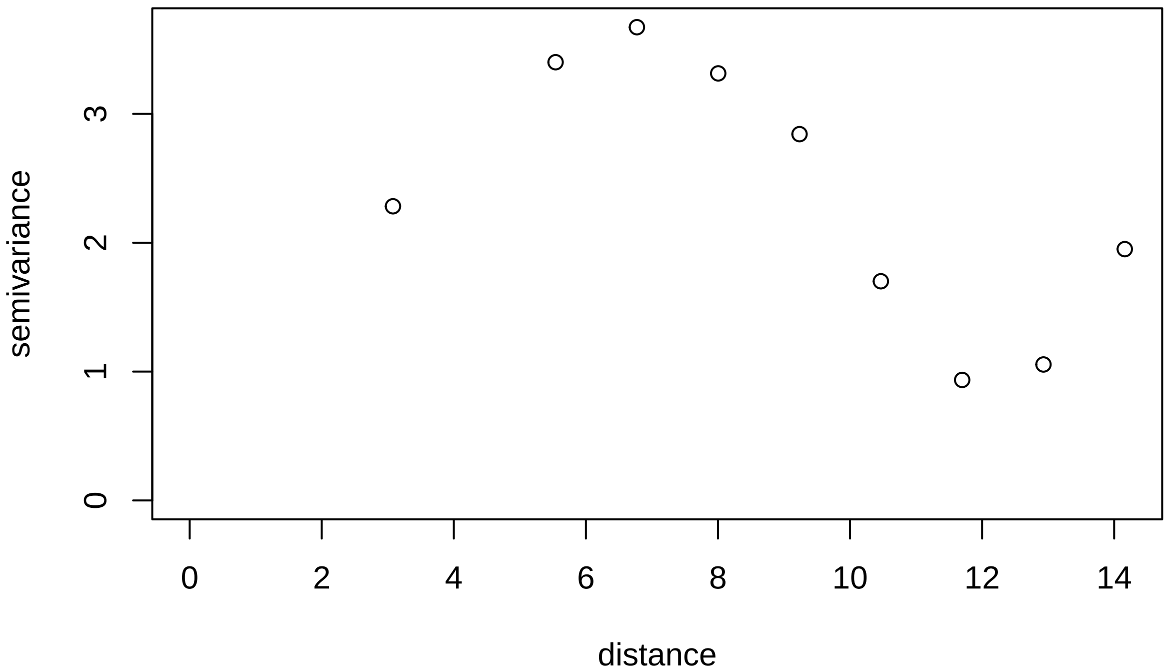

```
## variog: computing omnidirectional variogram
```



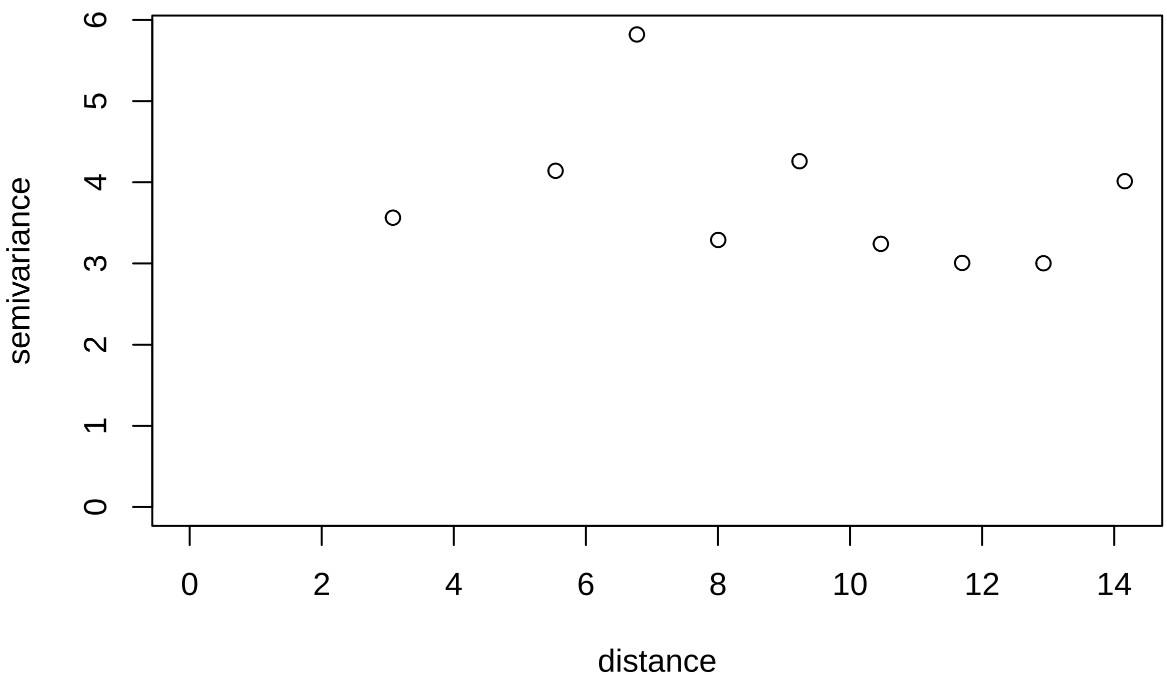

```
## variog: computing omnidirectional variogram
```



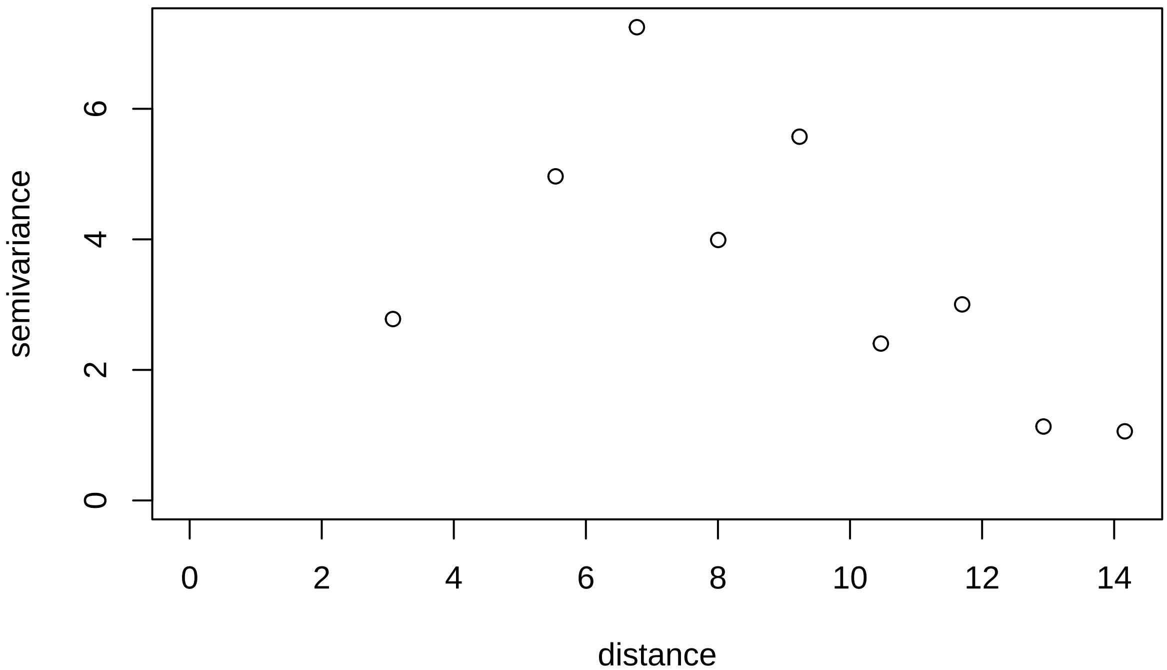

```
## variog: computing omnidirectional variogram
```



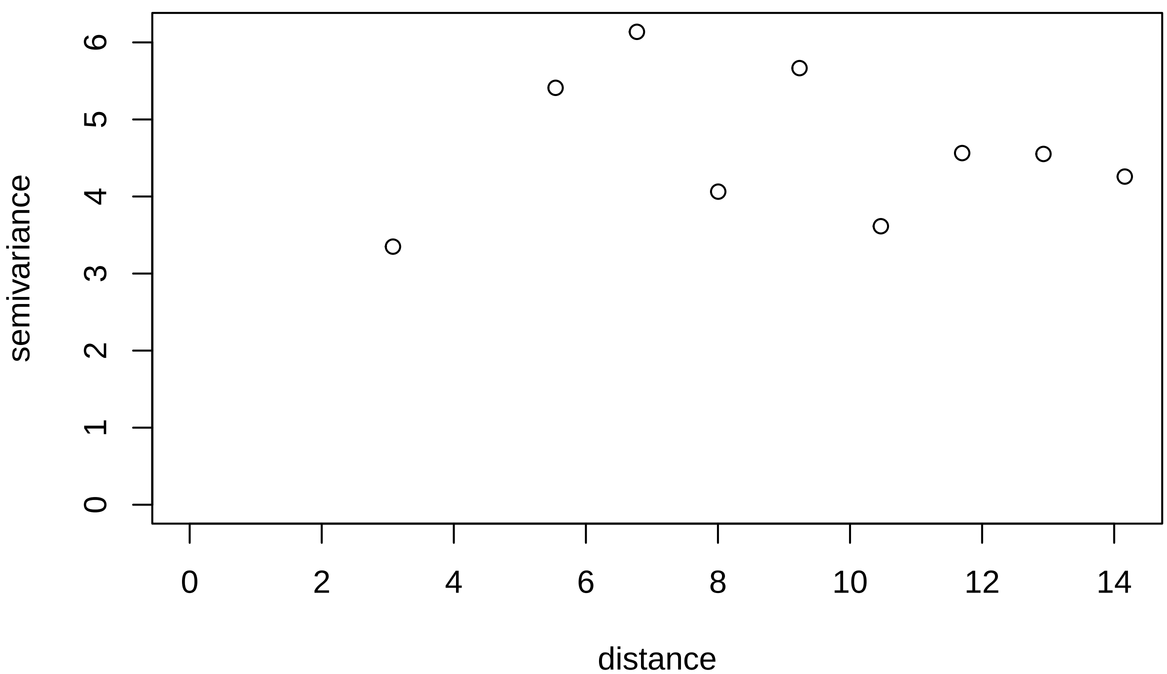

```
## variog: computing omnidirectional variogram
```



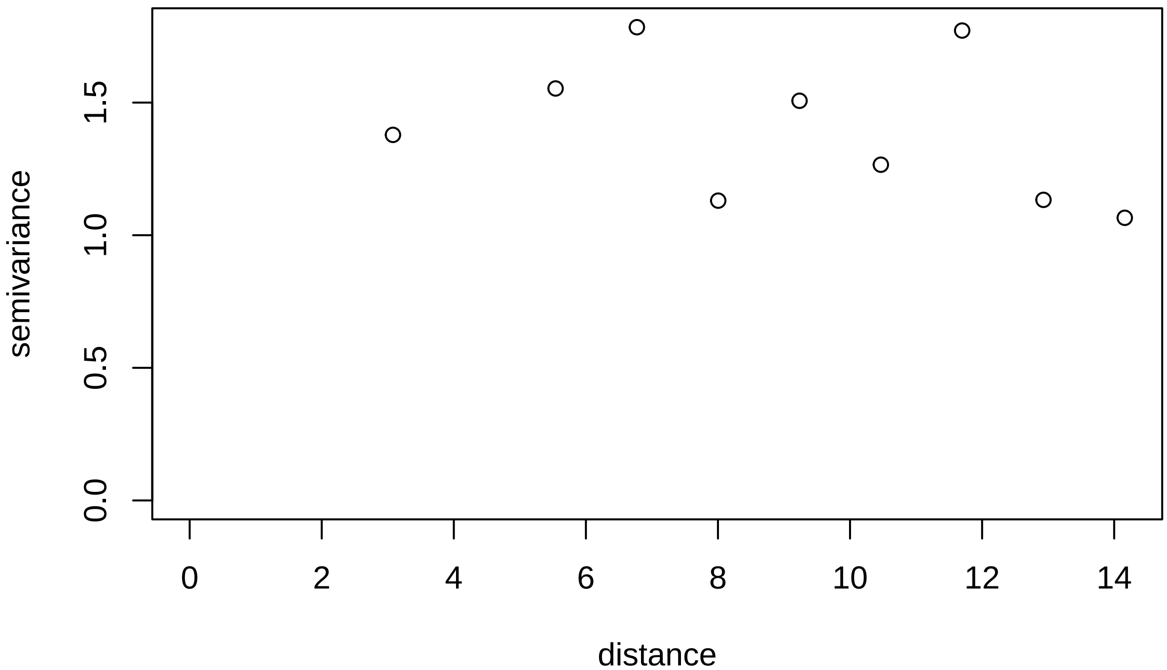

```
## variog: computing omnidirectional variogram
```



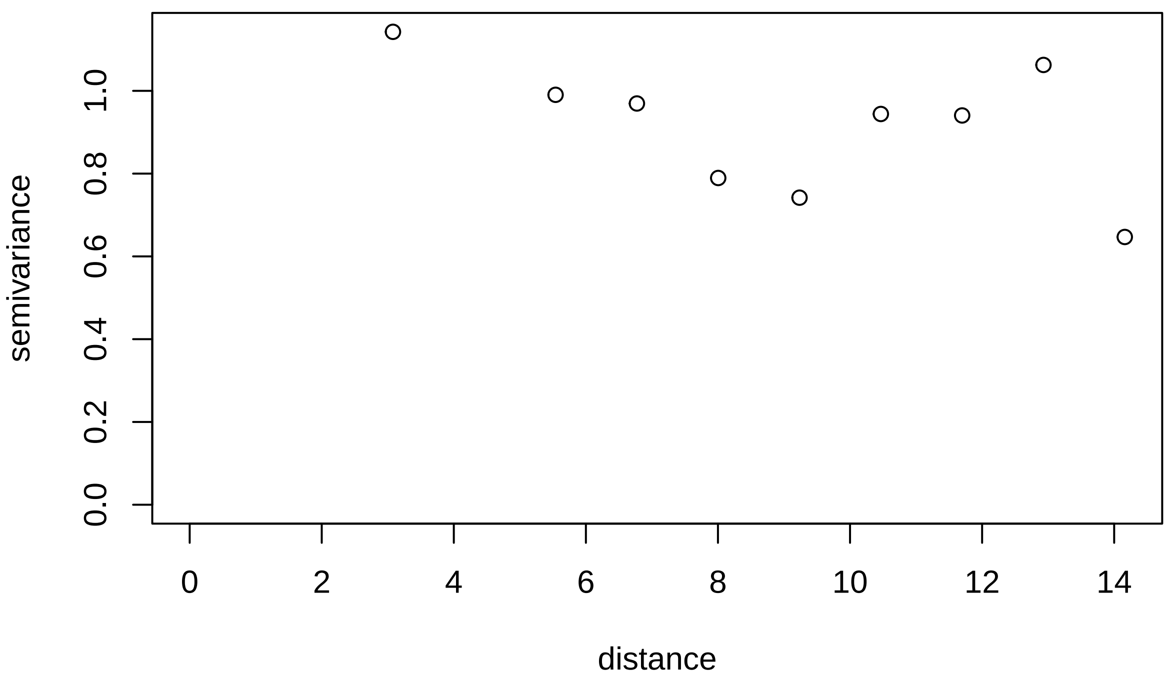

```
## variog: computing omnidirectional variogram
```



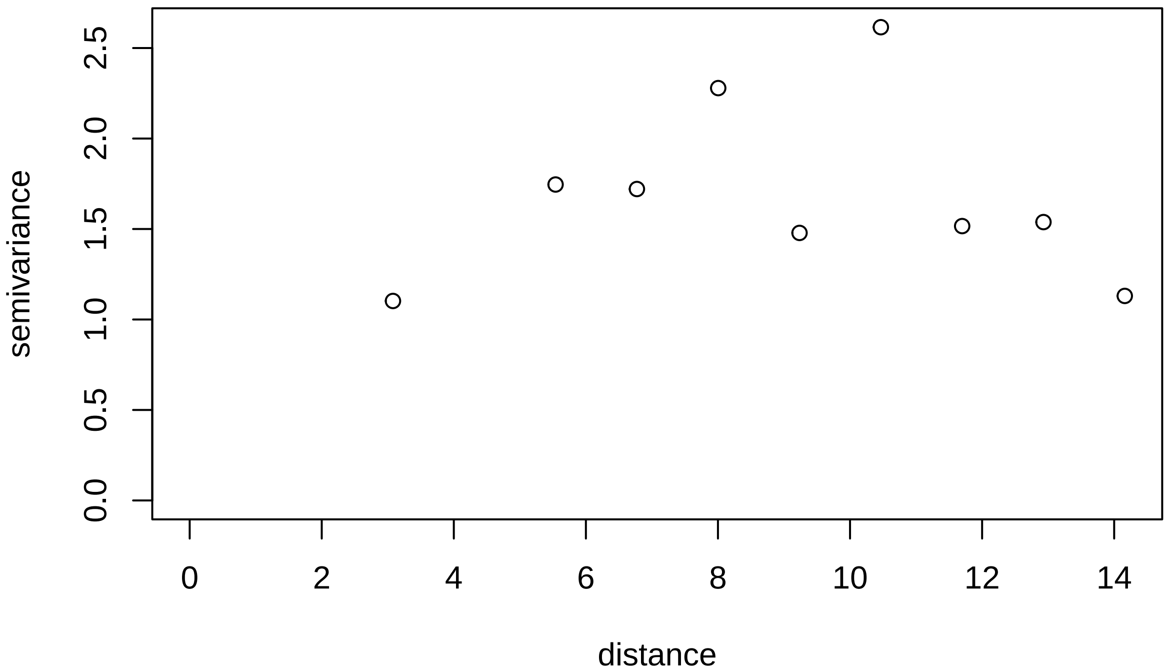

```
## variog: computing omnidirectional variogram
```



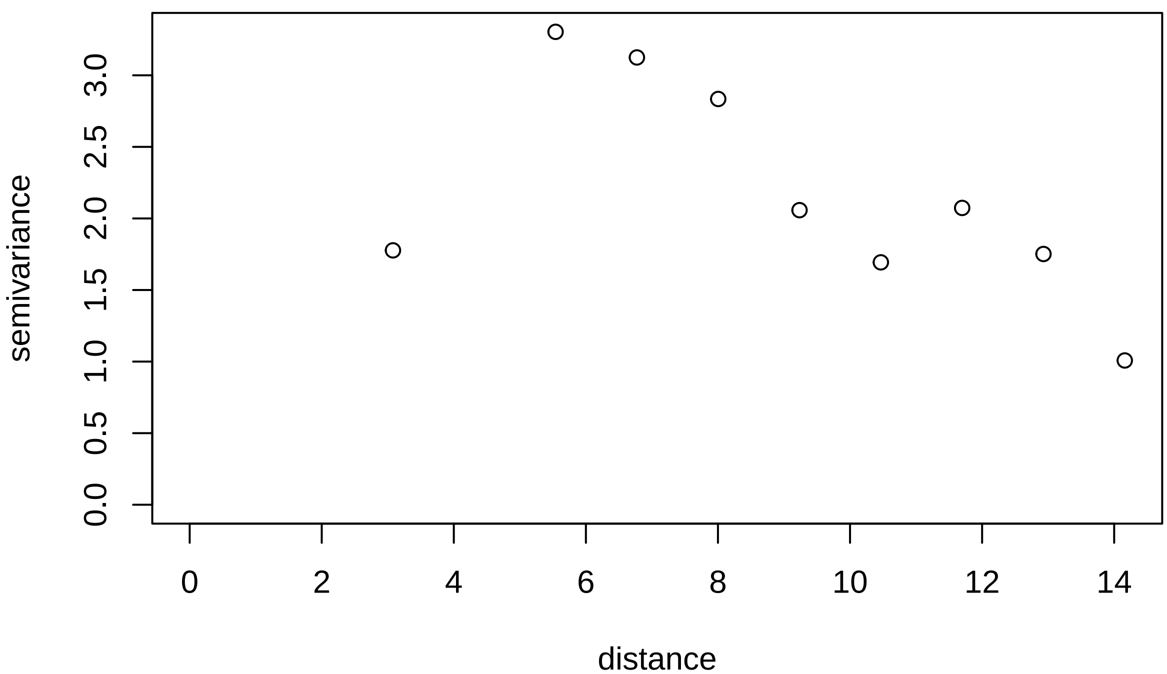

```
## variog: computing omnidirectional variogram
```




```
## variog: computing omnidirectional variogram
```




```
## variog: computing omnidirectional variogram
```

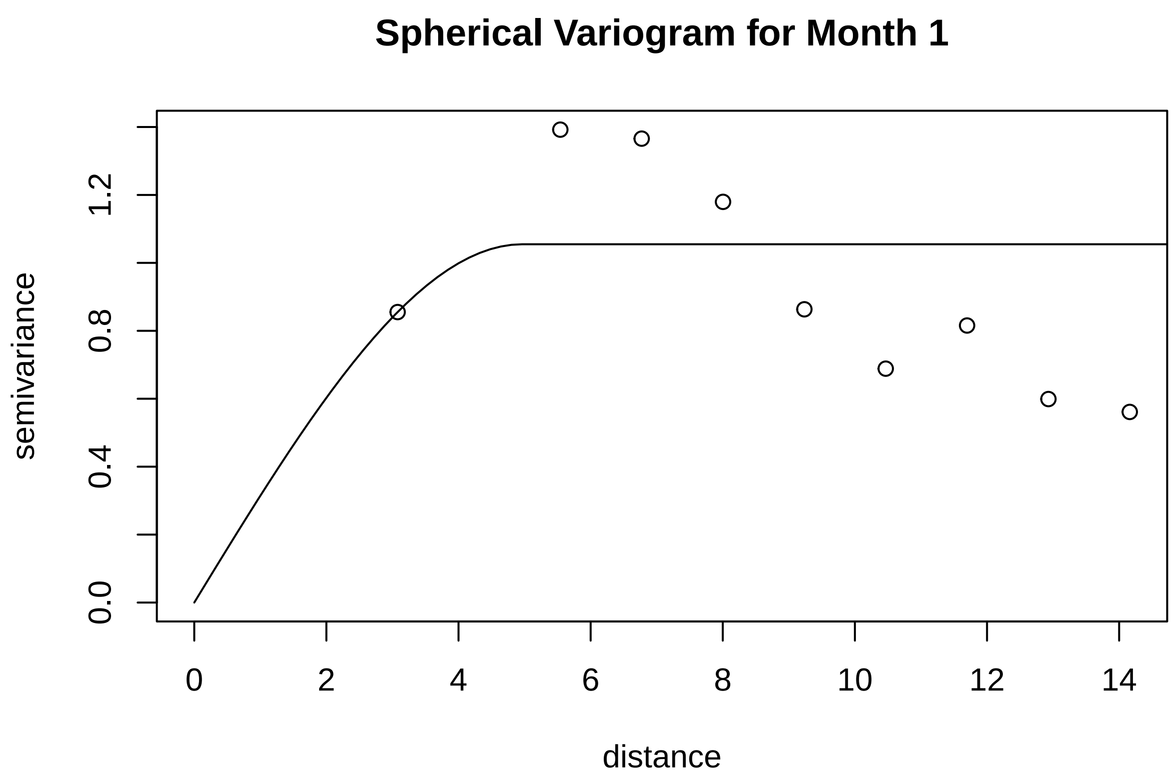


fitting a model

```
for(m in 1:24) {
    if(m %in% c(3, 4, 6, 10, 14, 19, 20, 23, 24)) {
        fit = variofit(empirical[[m]], cov.model = "gaussian", fix.nugget = FALSE, fix.kappa = TRUE)
        my_title = paste0("Gaussian Variogram for Month ", m)
    } else if(m %in% c(1, 2, 5, 7, 8, 9, 11, 12, 13, 15, 16, 17)) {
        fit = variofit(empirical[[m]], cov.model = "spherical", fix.nugget = FALSE, fix.kappa = TRUE)
```

```
        my_title = paste0("Spherical Variogram for Month ", m)
    } else {
        fit = variofit(empirical[[m]], cov.model = "exponential", fix.nugget = FALSE, fix.kappa = TRUE)
        my_title = paste0("Exponential Variogram for Month ", m)
    }
    plot(empirical[[m]], main=my_title)
    lines(fit)
}
```

```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search

## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "1.04"  "4.53" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 33.4676581471448

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```

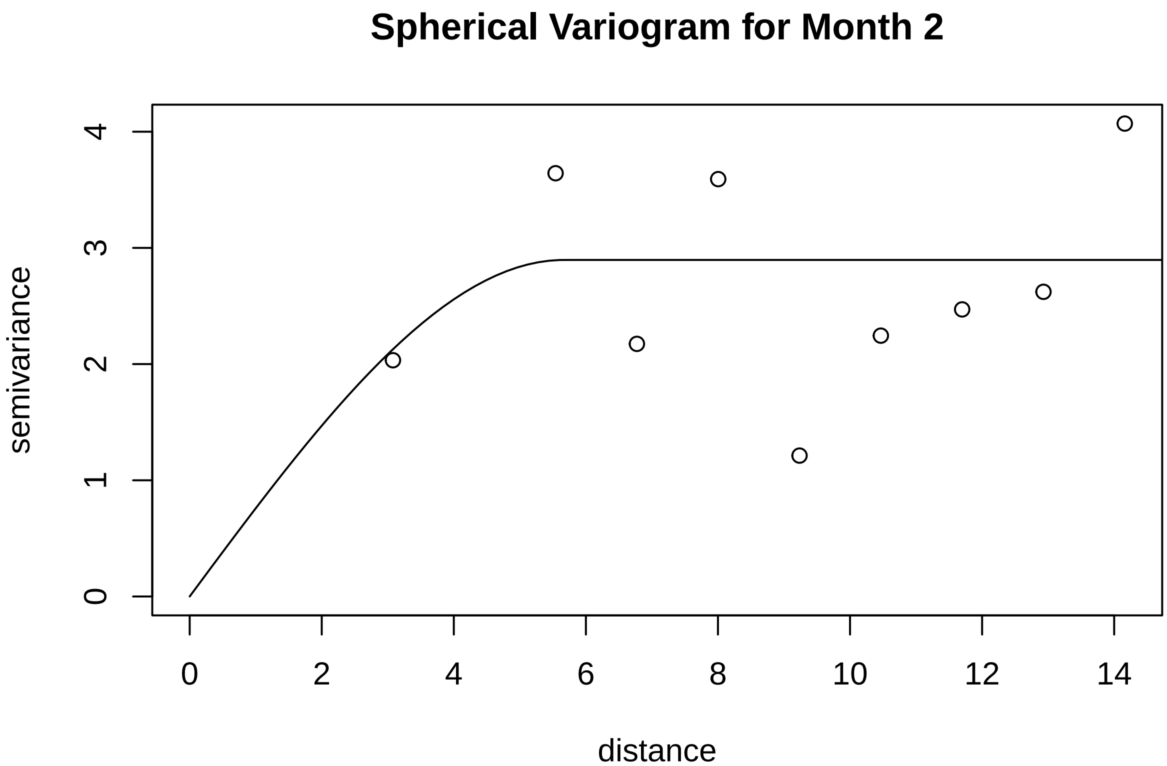



```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "2.03"  "6.8" "1.02" "0.5"
```

```
## status        "est"   "est" "est"  "fix"
## loss value: 264.621675415381
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

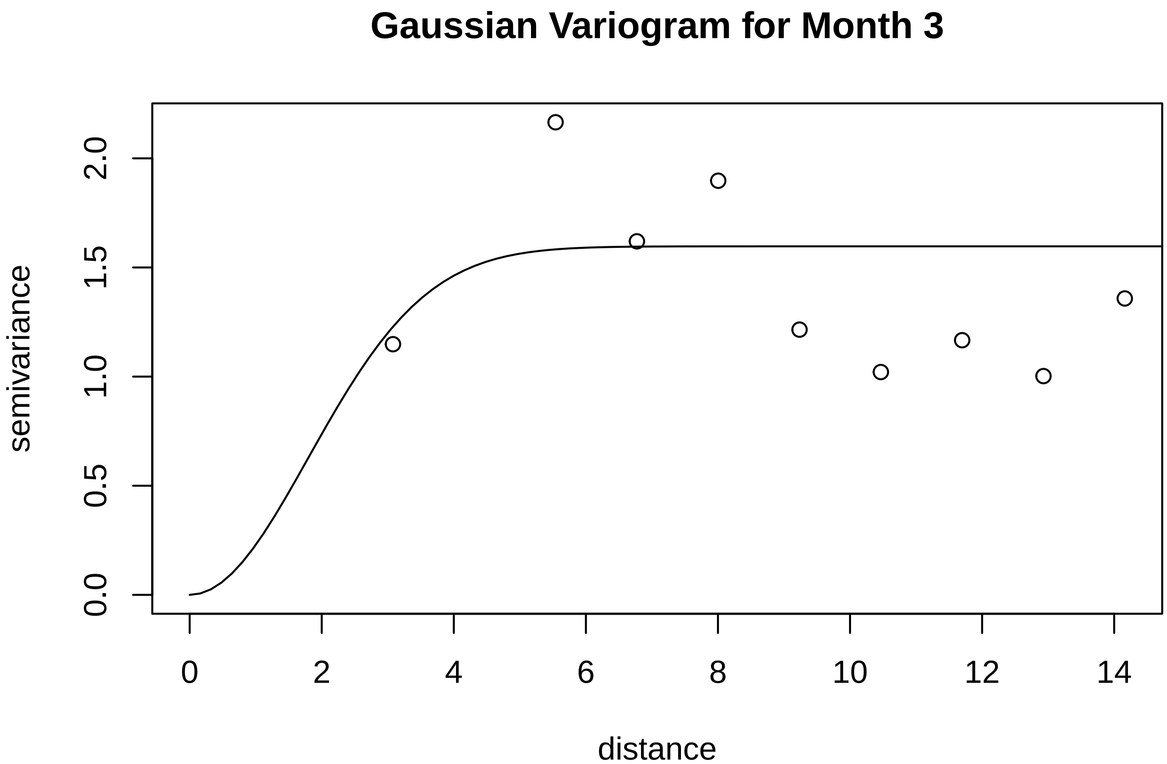
```
## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



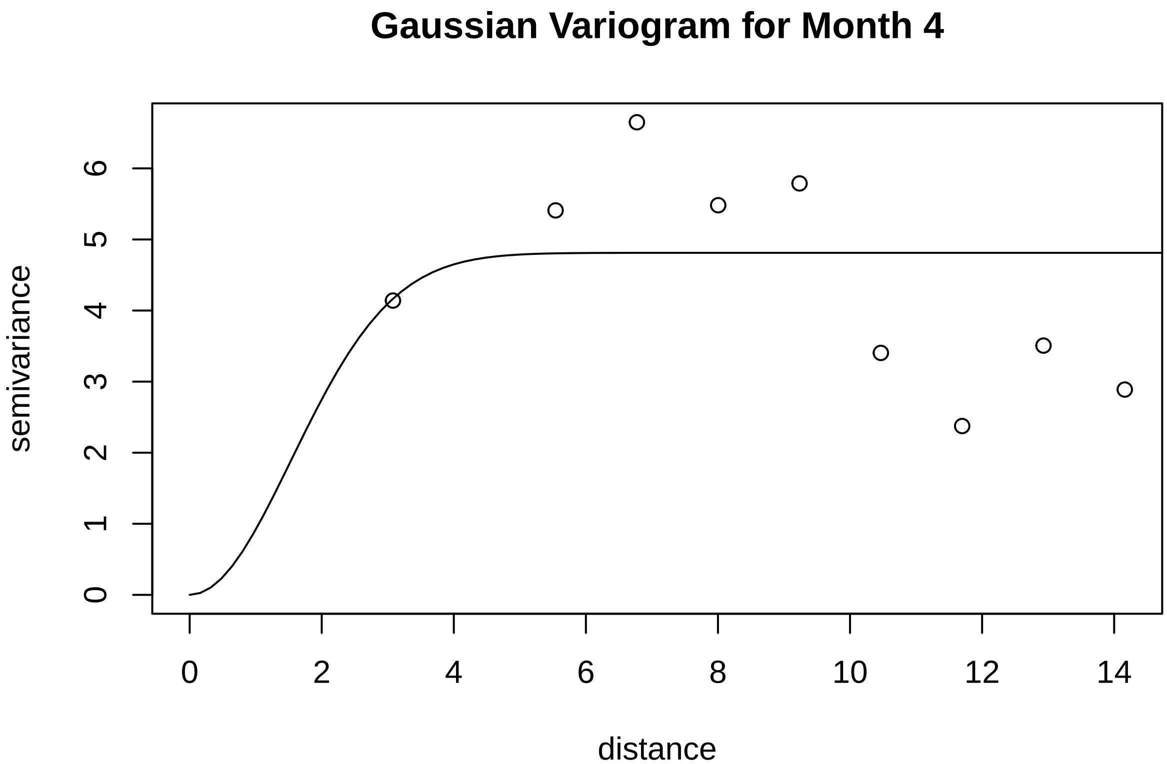

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "1.62"  "2.27" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 82.1529004948056
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

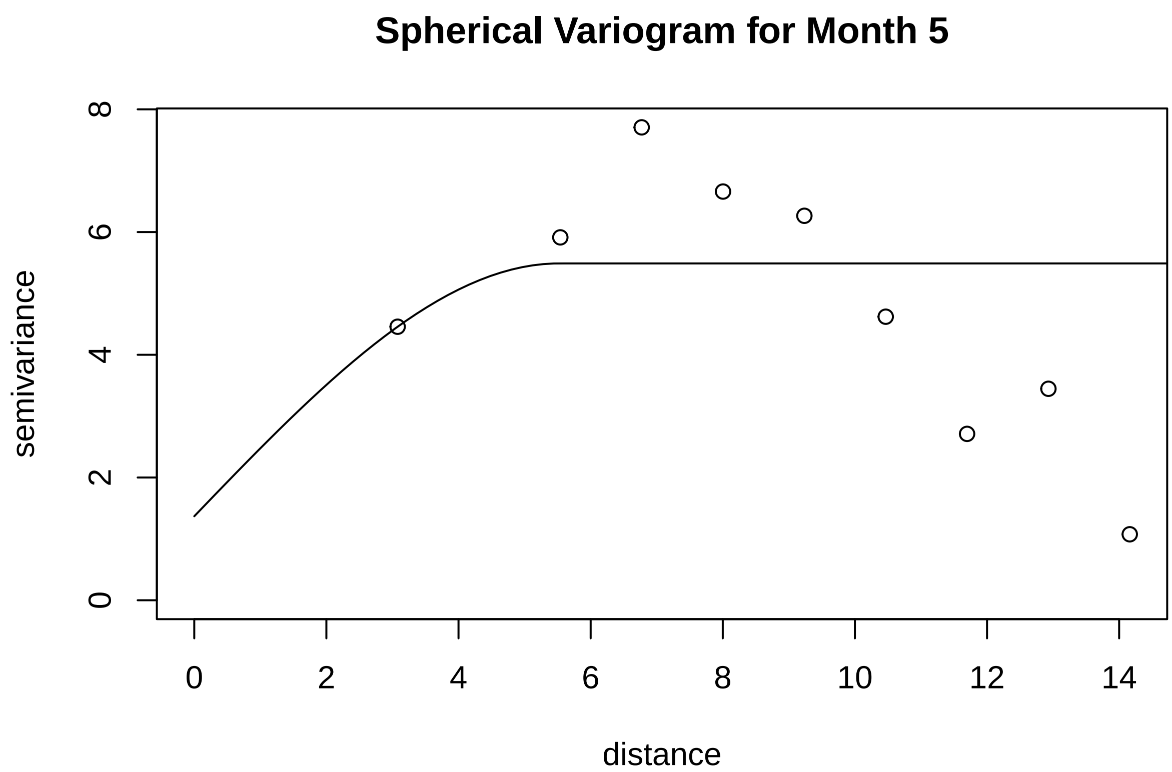
```
## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



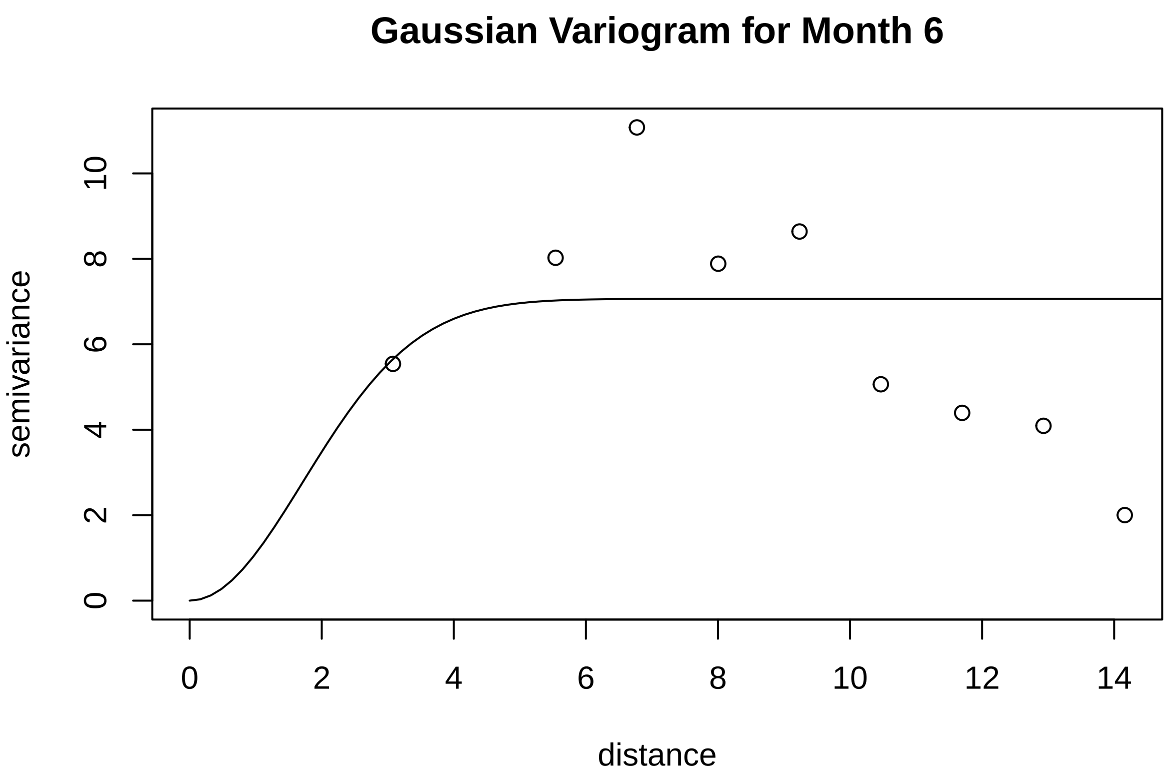

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "4.99"  "2.27" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 504.048900885206

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



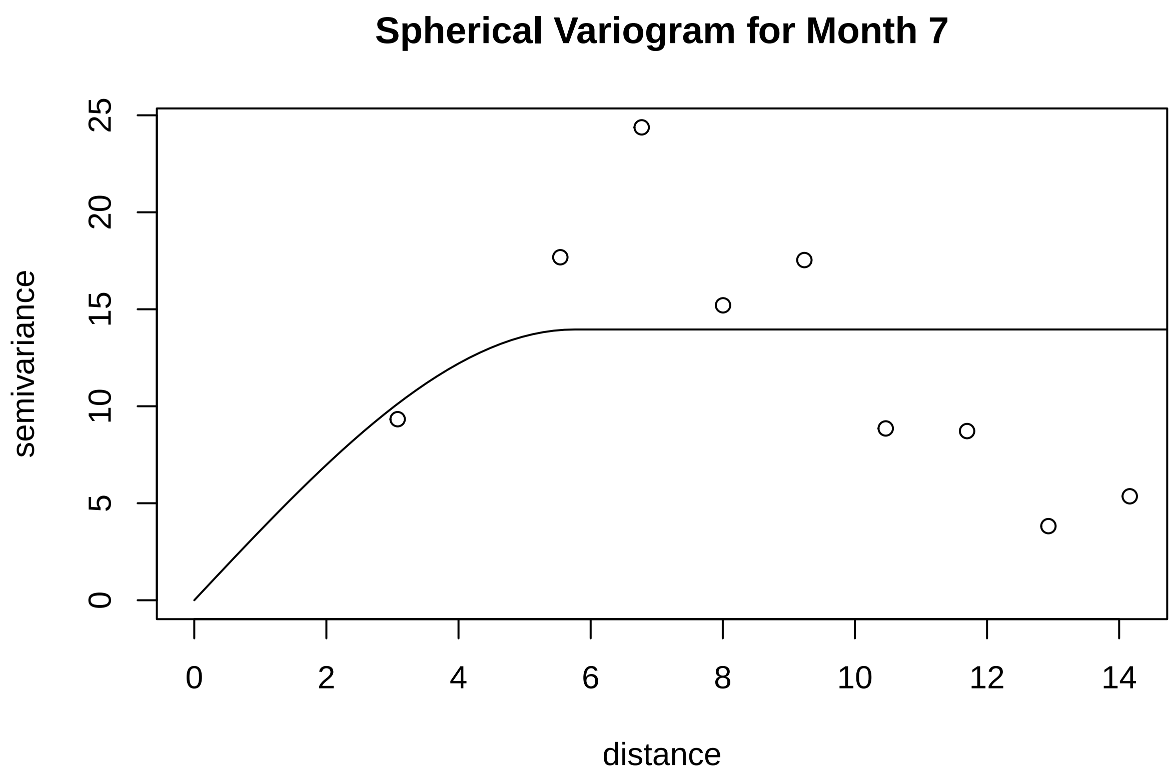

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "3.85"  "6.8" "1.93" "0.5"
## status        "est"   "est" "est"  "fix"
## loss value: 780.676676442668

## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



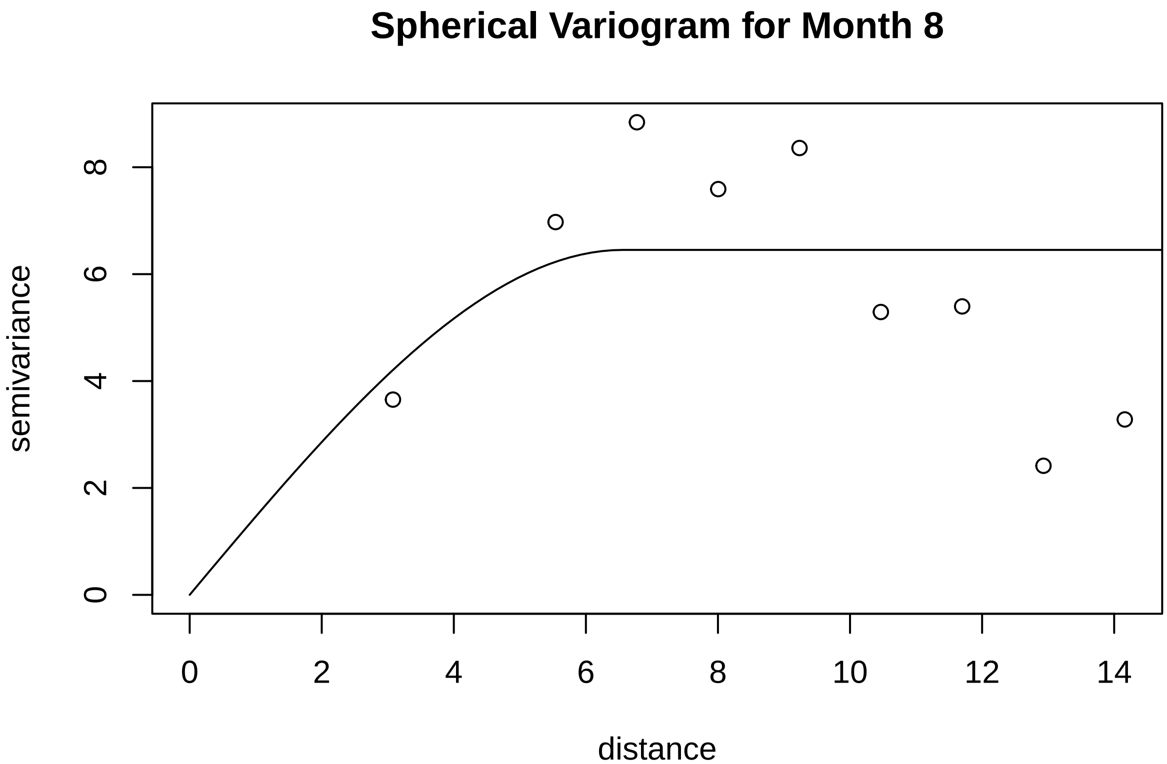

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq  kappa
## initial.value "5.54"  "2.27" "1.11" "0.5"
## status        "est"   "est"  "est"  "fix"
## loss value: 1538.80522476774

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



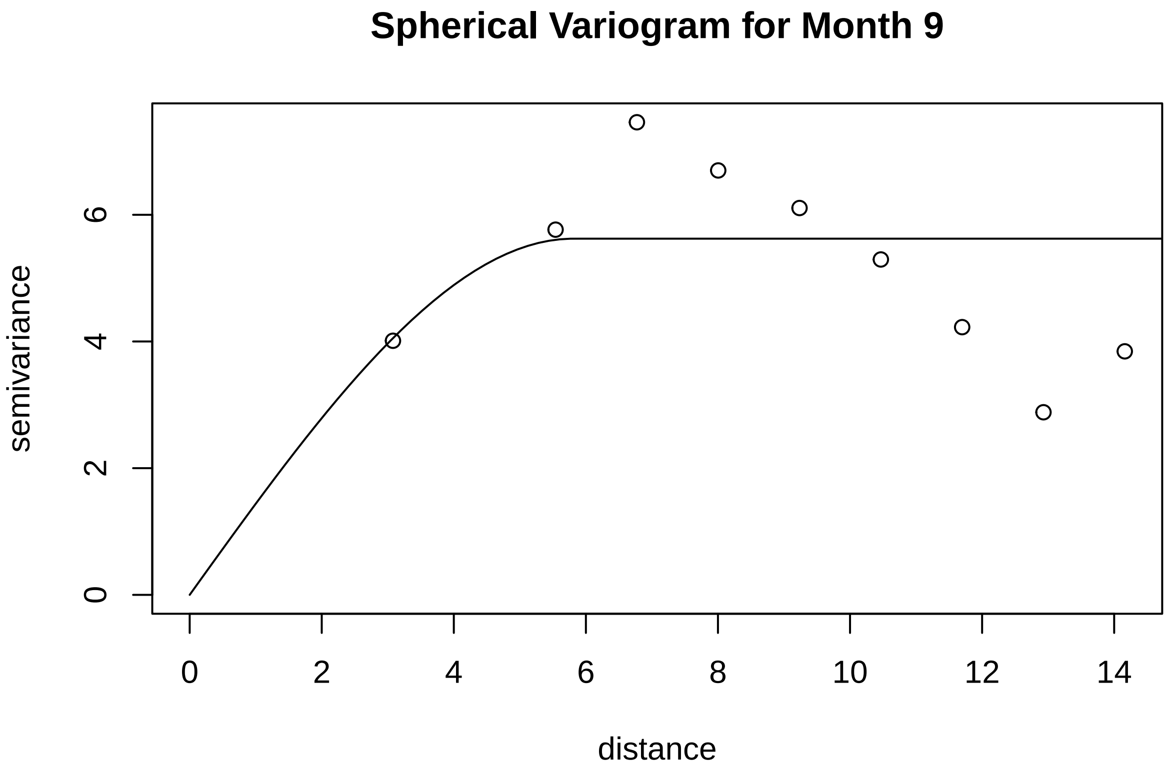

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "12.19" "6.8" "2.44" "0.5"
## status        "est"   "est" "est"  "fix"
## loss value: 11015.8679253109

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



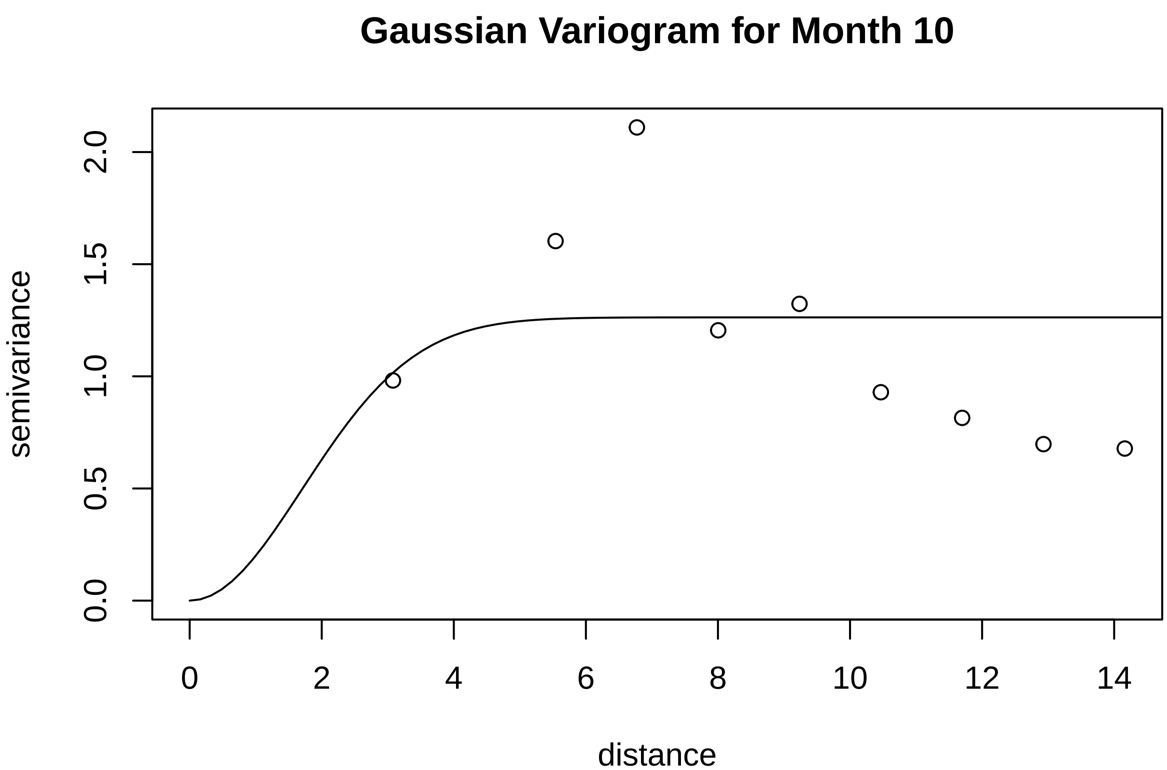

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq kappa
## initial.value "6.63"  "6.8" "0"   "0.5"
## status        "est"   "est" "est" "fix"
## loss value: 1158.13811684915

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



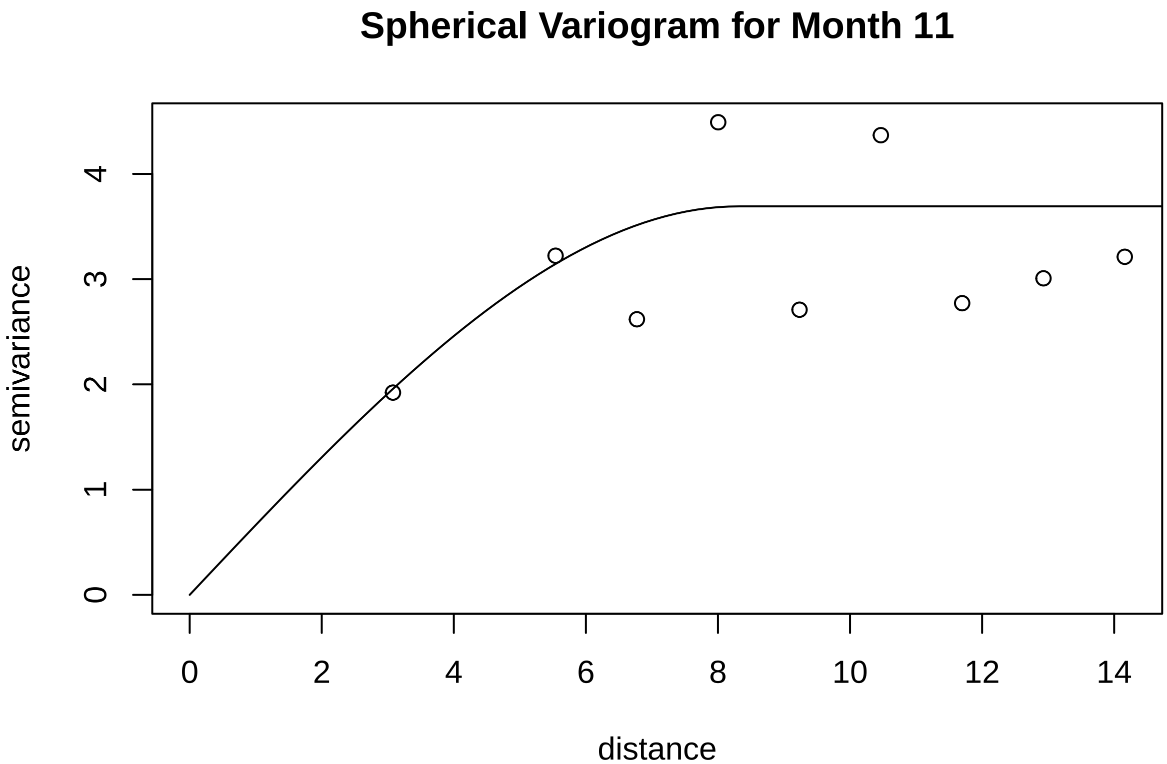

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "3.73"  "6.8" "1.87" "0.5"
## status        "est"   "est" "est"  "fix"
## loss value: 505.105340122813

## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



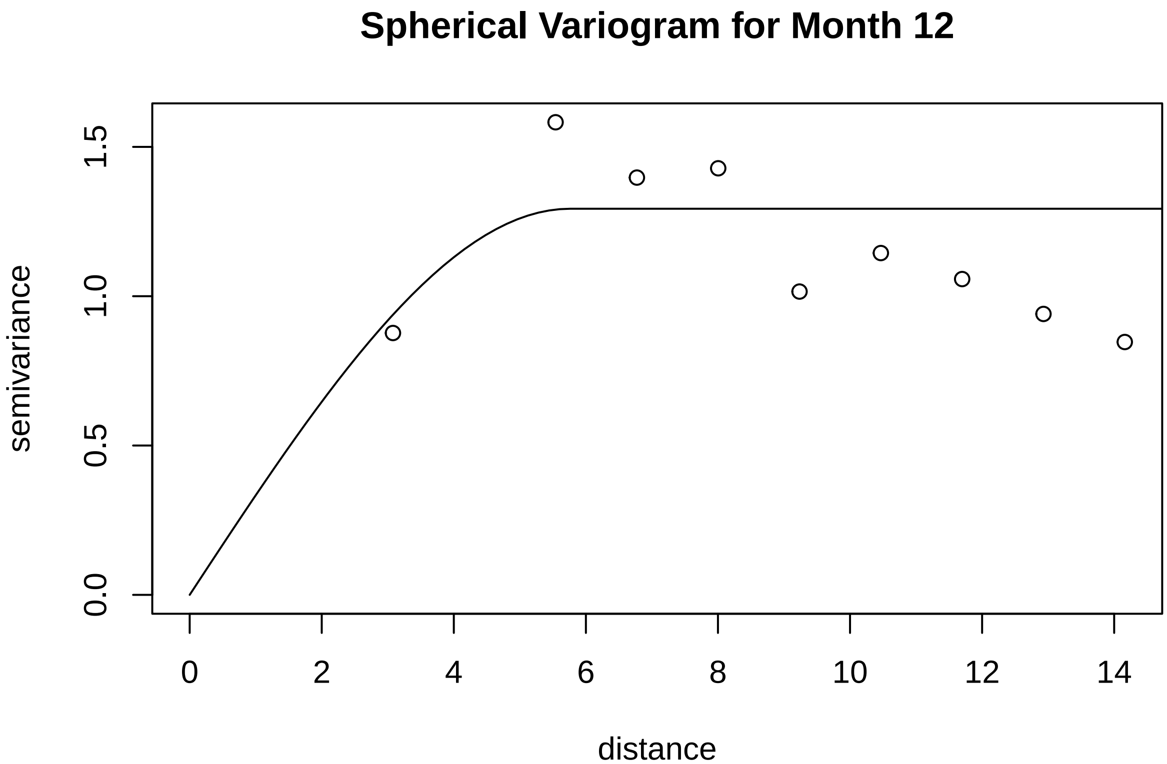

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq  kappa
## initial.value "1.05"  "2.27" "0.21" "0.5"
## status        "est"   "est"  "est"  "fix"
## loss value: 53.8659617637344

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



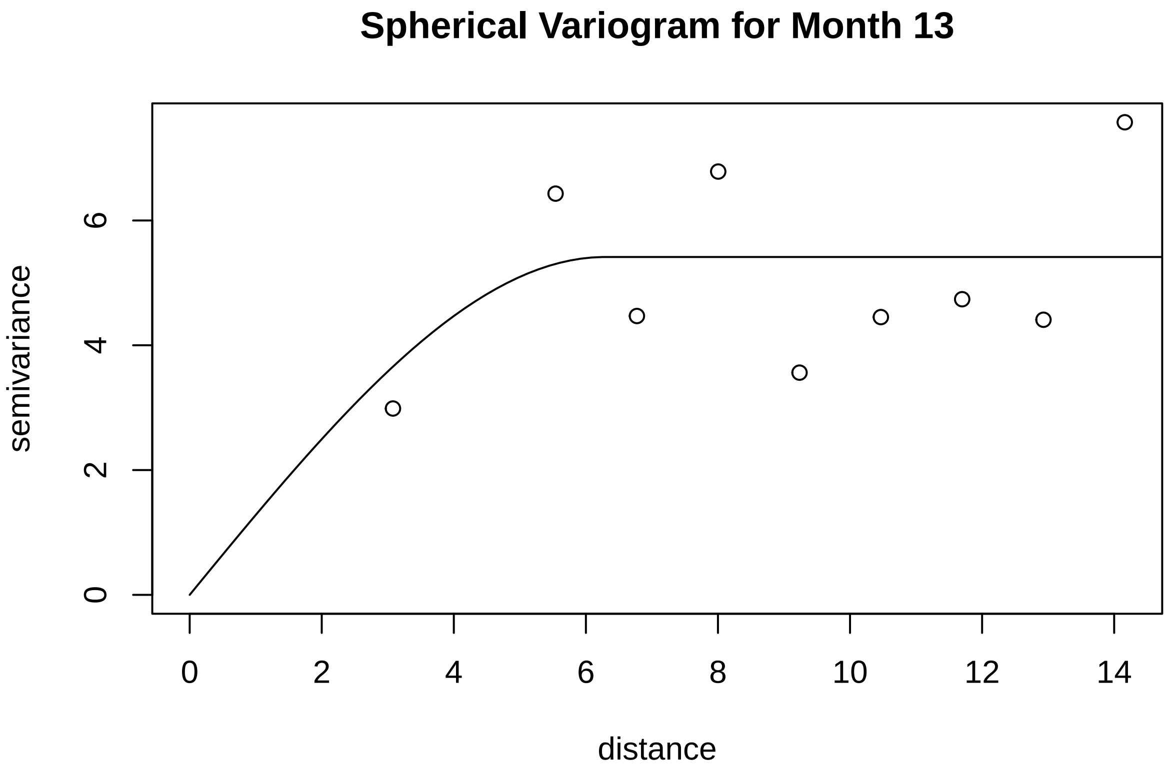

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq  kappa
## initial.value "3.37"  "9.06" "0.45" "0.5"
## status        "est"   "est"  "est"  "fix"
## loss value: 166.215479479821

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



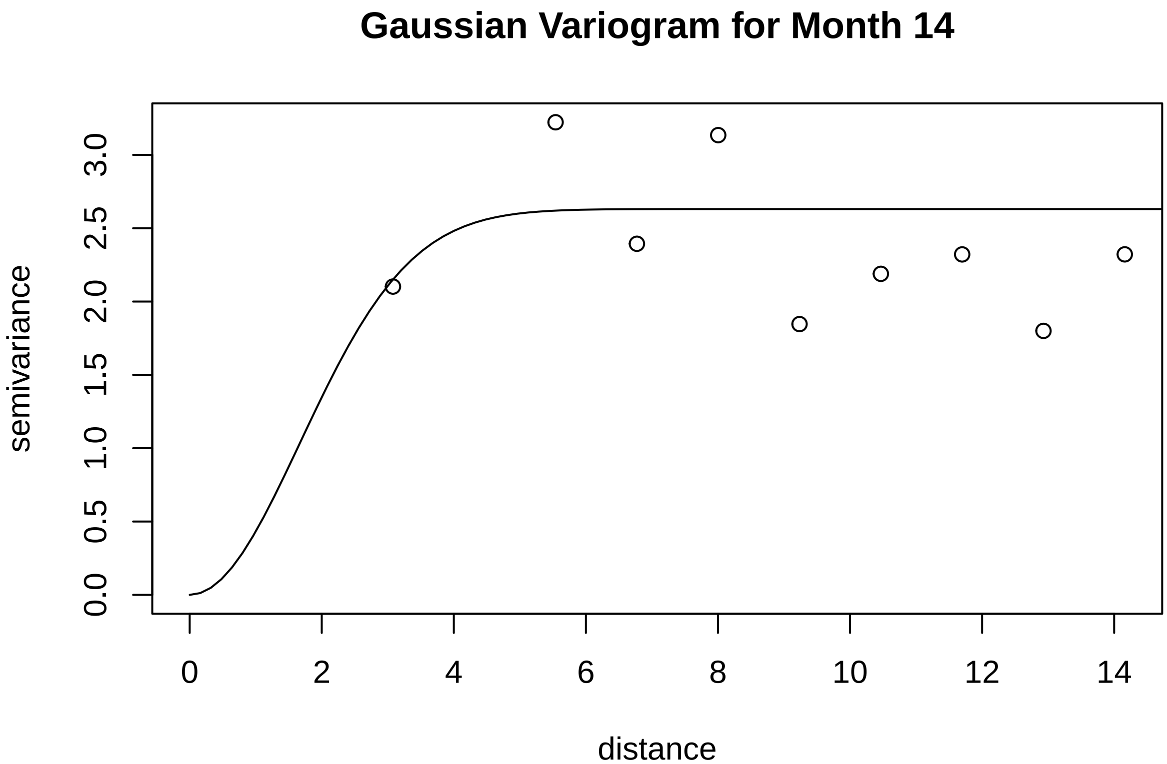

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "1.19"  "6.8" "0.16" "0.5"
## status        "est"   "est" "est"  "fix"
## loss value: 24.2230884931414

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



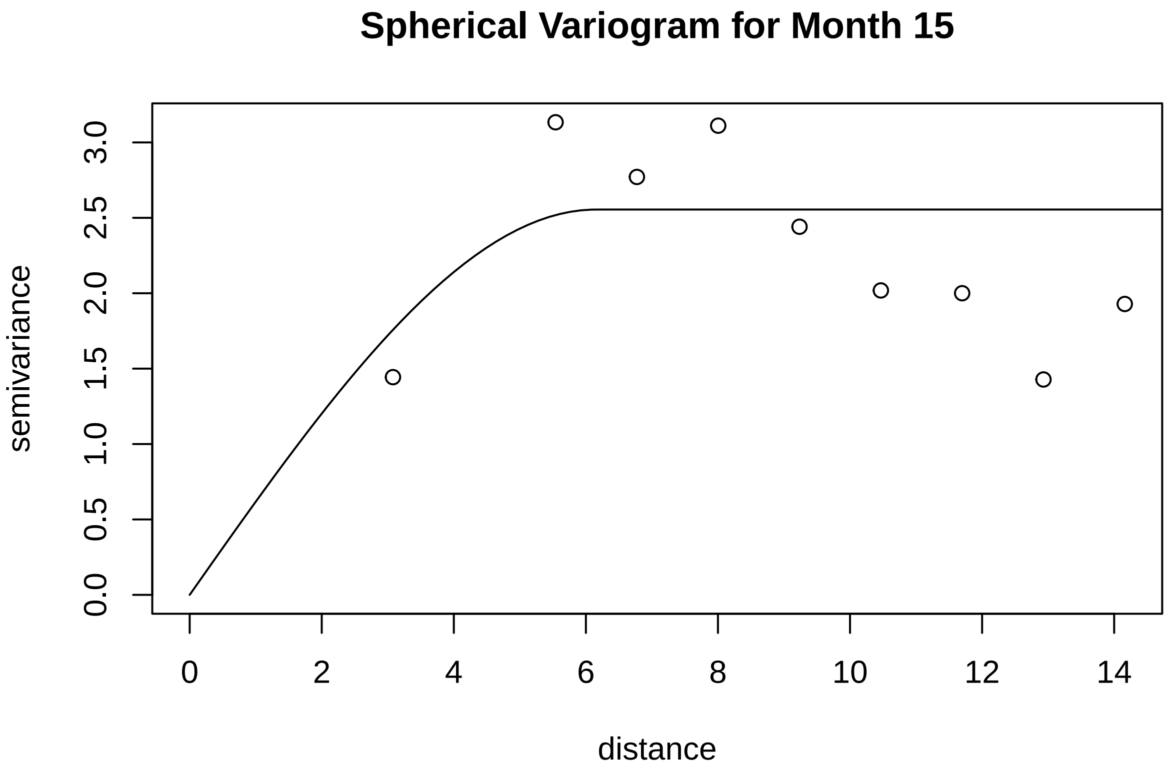

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq kappa
## initial.value "5.68"  "6.8" "0"   "0.5"
## status        "est"   "est" "est" "fix"
## loss value: 593.255624490829

## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



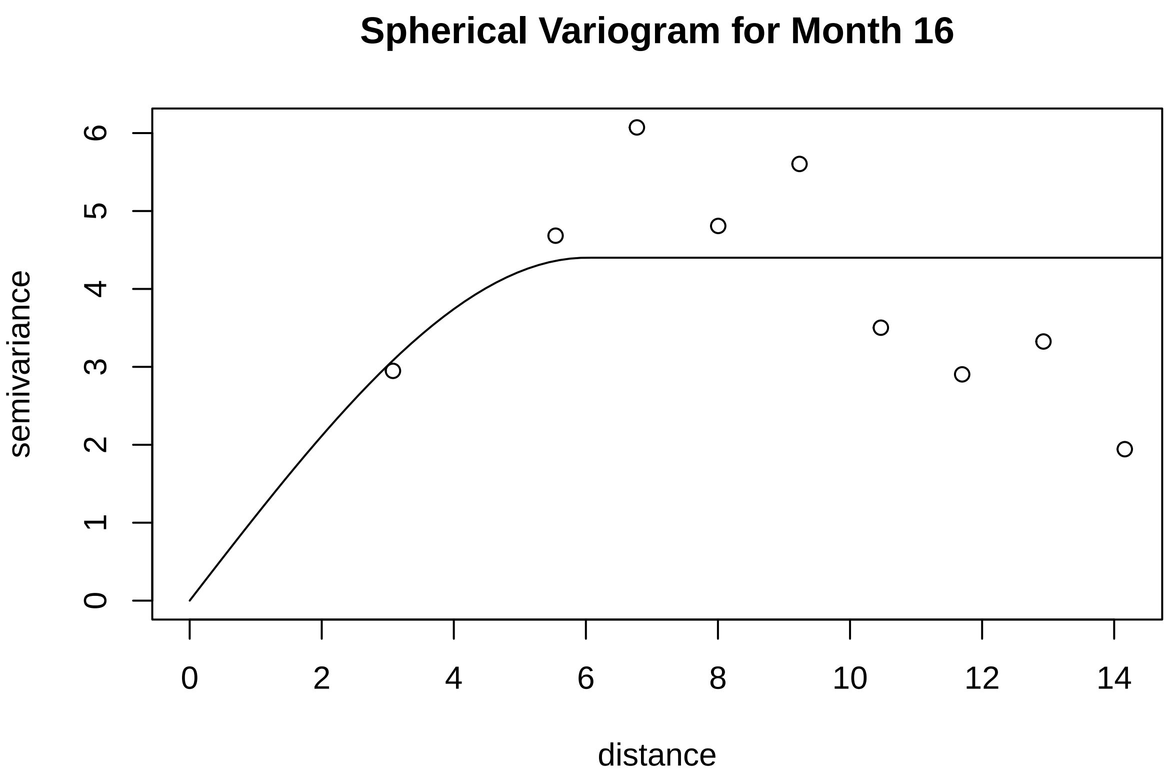

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq  kappa
## initial.value "2.42"  "2.27" "0.32" "0.5"
## status        "est"   "est"  "est"  "fix"
## loss value: 121.196044140946

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



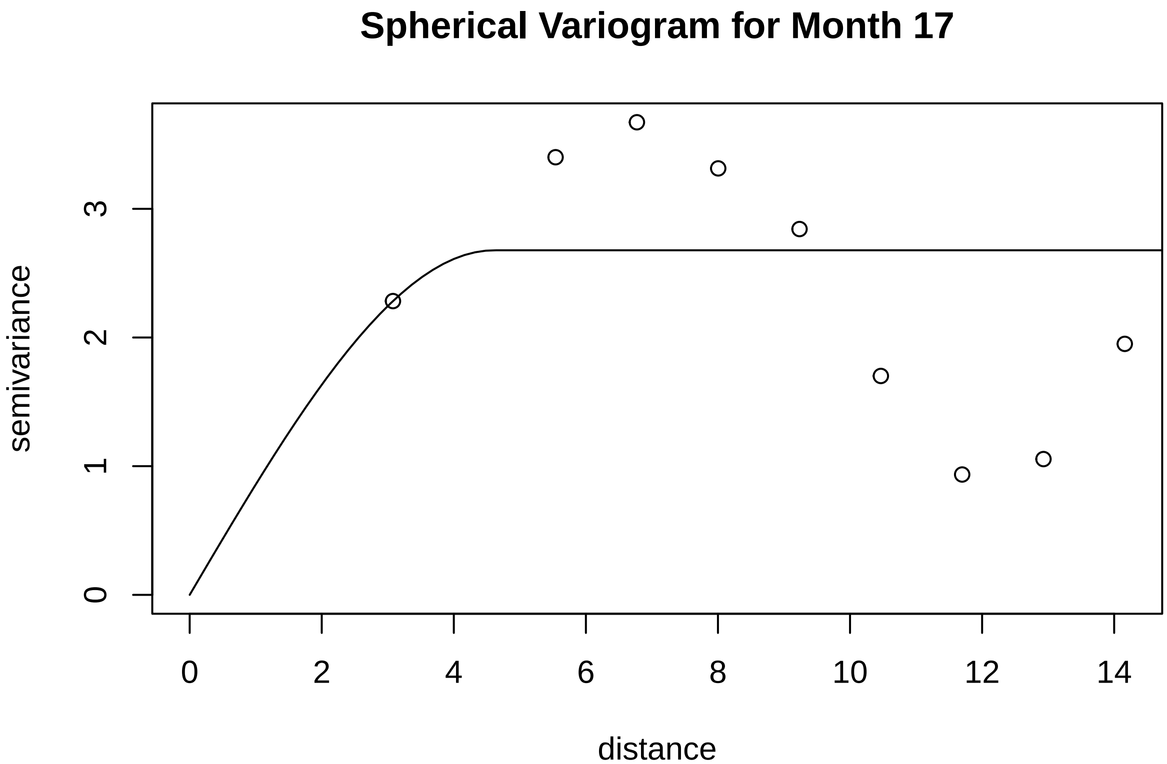

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "2.35"  "6.8" "0.31" "0.5"
## status        "est"   "est" "est"  "fix"
## loss value: 147.493988887621

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



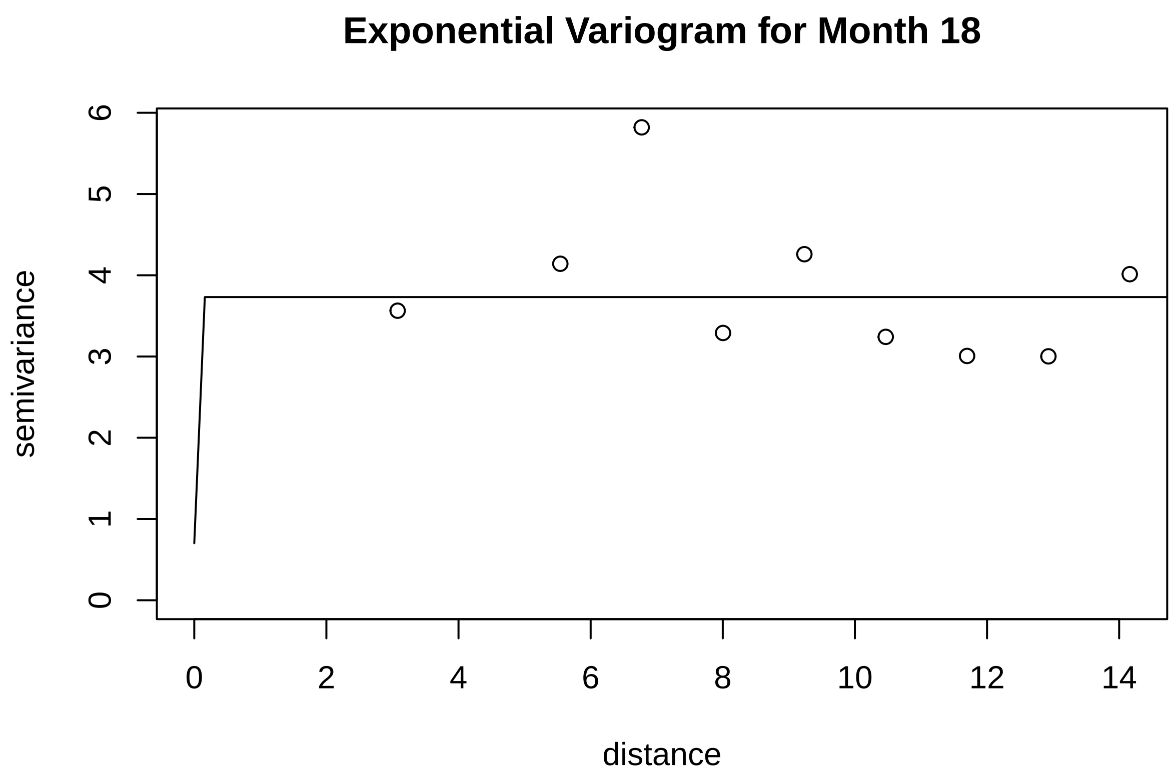

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq kappa
## initial.value "4.55"  "6.8" "0"   "0.5"
## status        "est"   "est" "est" "fix"
## loss value: 328.443817690152

## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



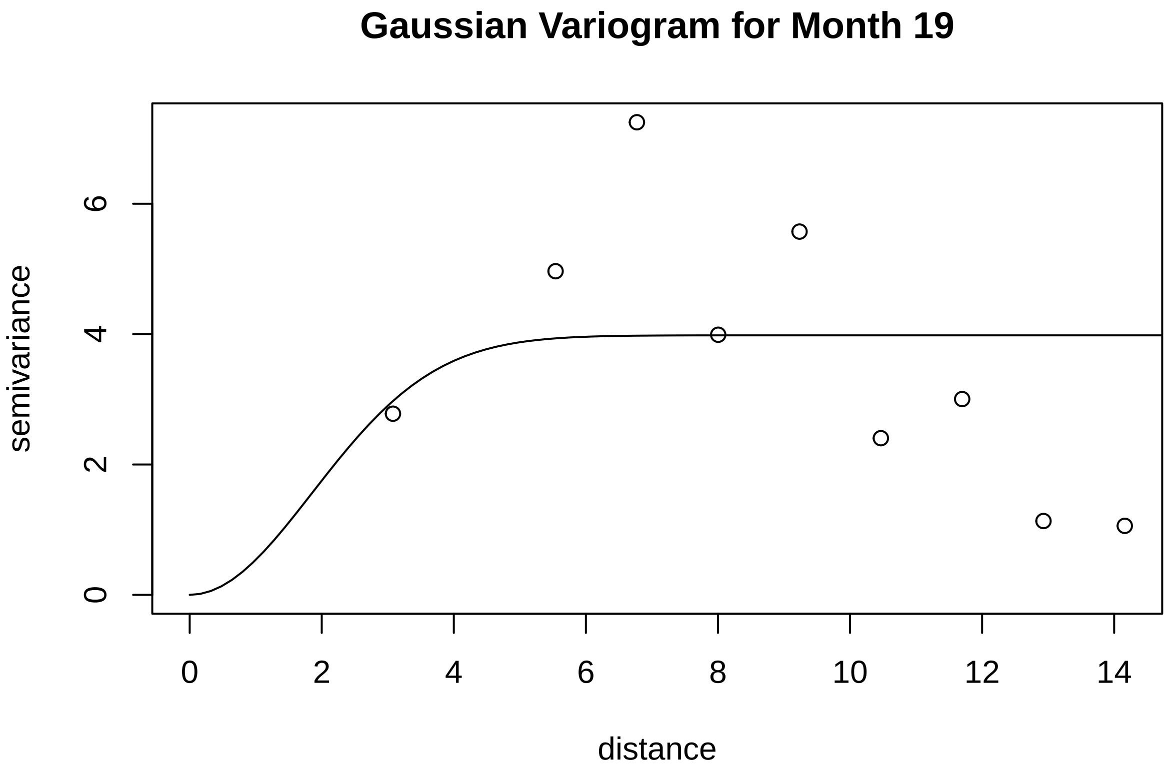

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "2.75"  "4.53" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 311.873198723778

## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "exponential", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



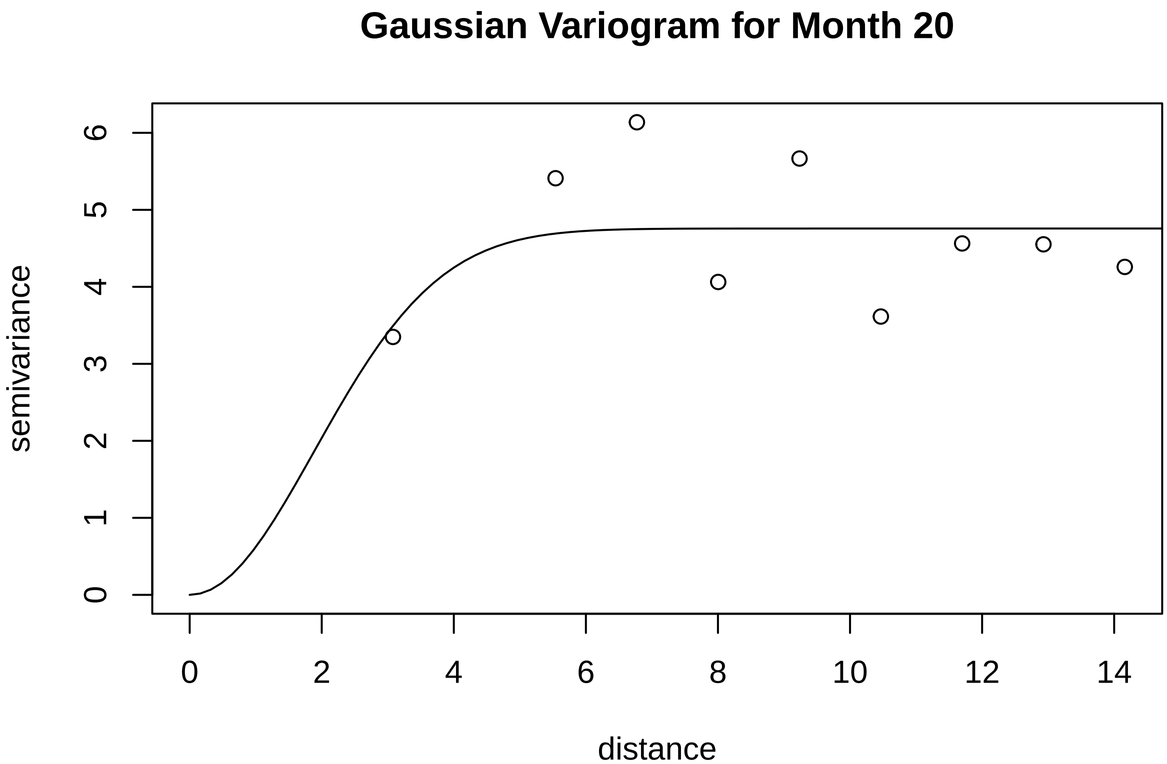

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "2.91"  "0"   "0.58" "0.5"
## status        "est"   "est" "est"  "fix"
## loss value: 213.827165023613

## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



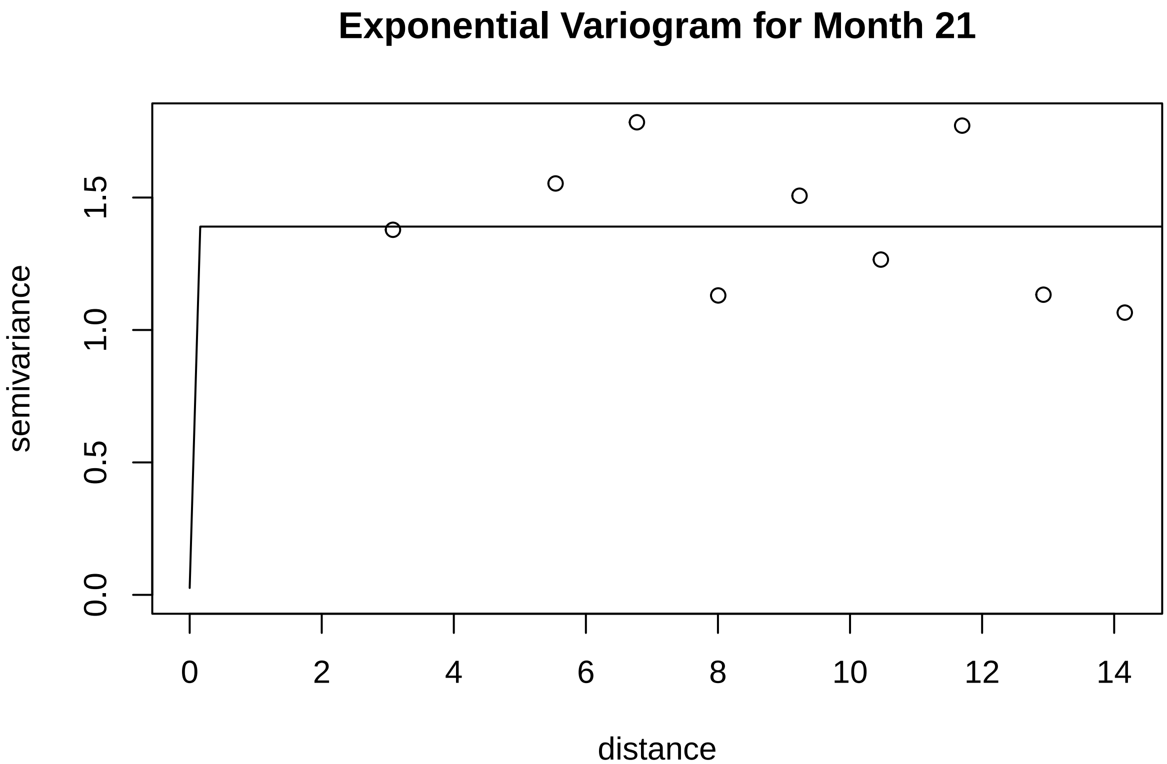

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "3.63"  "2.27" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 988.609853804012

## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



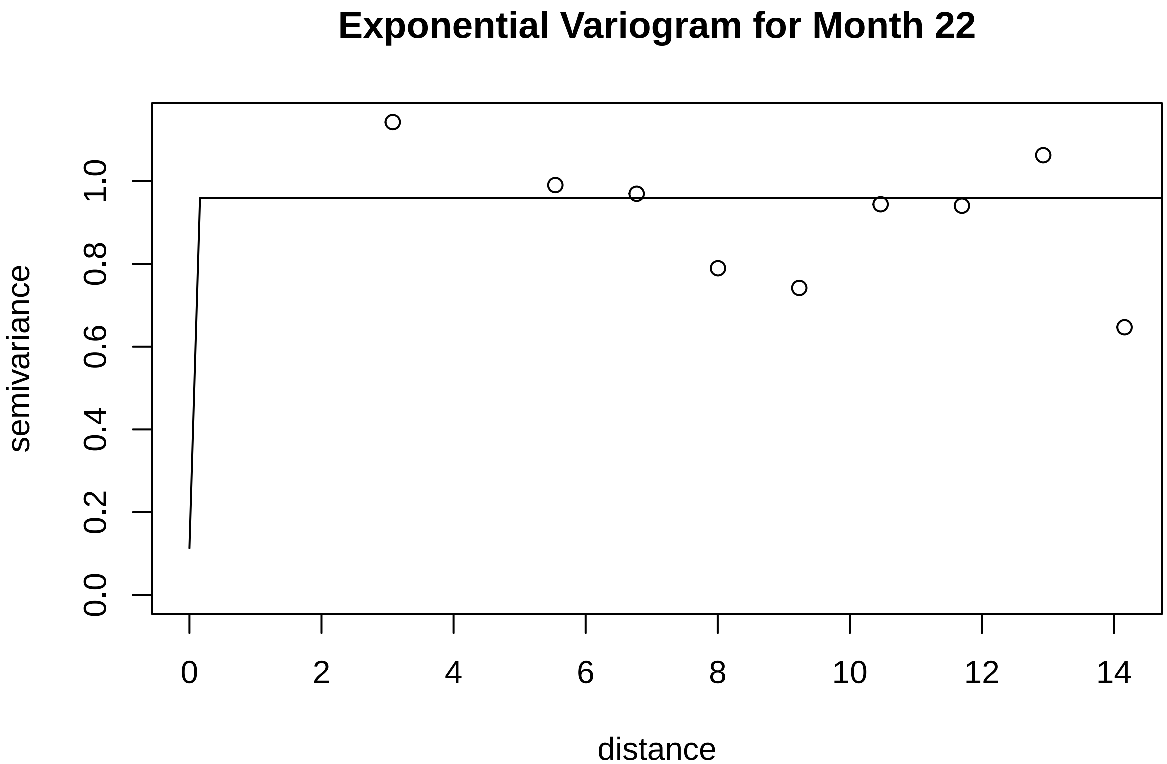

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "4.6"   "2.27" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 261.265093356613

## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "exponential", fix.nugget =
## FALSE, : initial values not provided - running the default search
```




```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq kappa
## initial.value "1.34"  "0"   "0"   "0.5"
## status        "est"   "est" "est" "fix"
## loss value: 19.5953905480498

## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "exponential", fix.nugget =
## FALSE, : initial values not provided - running the default search
```

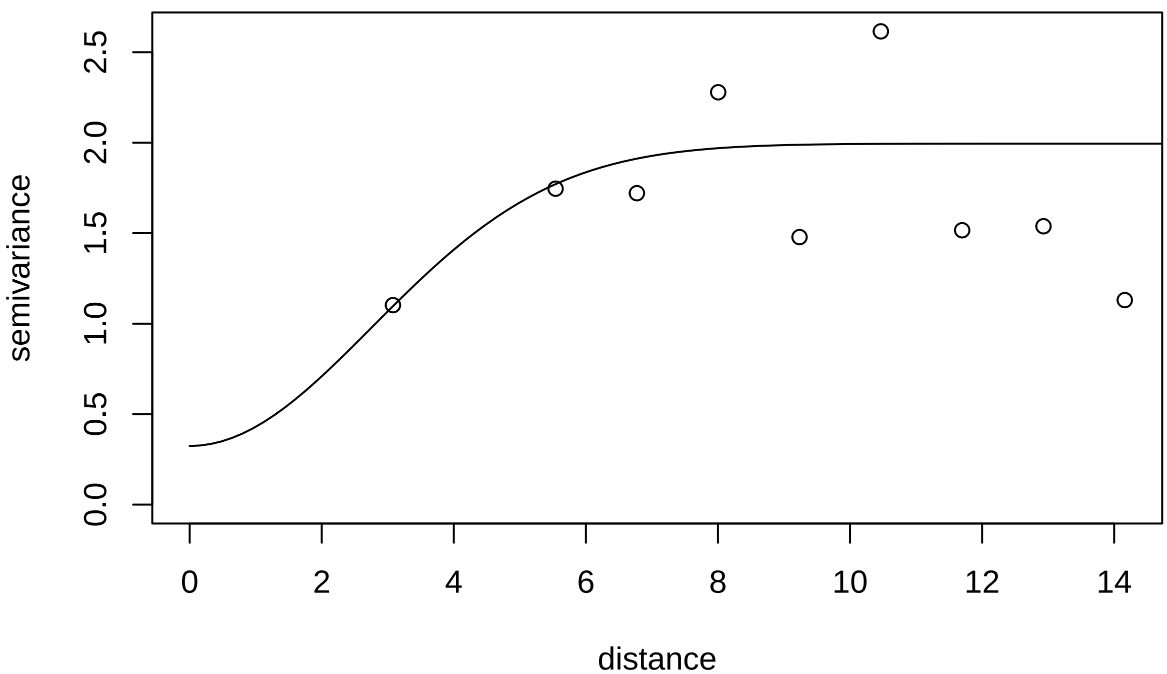



```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi   tausq  kappa
## initial.value "0.86"  "0"   "0.11" "0.5"
## status        "est"   "est" "est"  "fix"
## loss value: 7.95912216372283

## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```



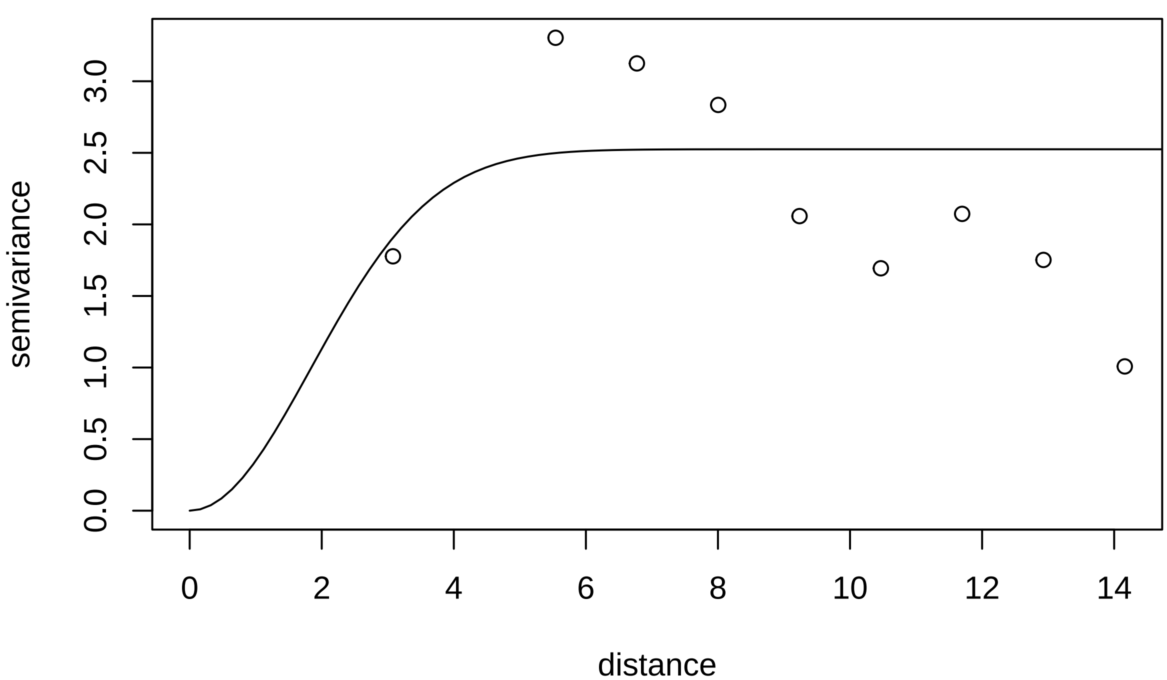

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq  kappa
## initial.value "1.31"  "4.53" "0.65" "0.5"
## status        "est"   "est"  "est"  "fix"
## loss value: 57.0999413939858

## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim

## Warning in variofit(empirical[[m]], cov.model = "gaussian", fix.nugget =
## FALSE, : initial values not provided - running the default search
```

Gaussian Variogram for Month 23




```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "2.48"  "2.27" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 174.050273616137
```

Gaussian Variogram for Month 24



```r
fit = variofit(empirical[[1]], cov.model = "spherical", fix.nugget = FALSE, fix.kappa = TRUE)
```

```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
## Warning in variofit(empirical[[1]], cov.model = "spherical", fix.nugget =
## FALSE, : initial values not provided - running the default search
```

```
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "1.04"  "4.53" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 33.4676581471448
```

```r
#Months 18, 21, all 4 equal
#Month 22, matern and exponential equal
#gaussian = 3, 4, 6, 10, 14, 19, 20, 23, 24
#spherical = 1, 2, 5, 7, 8, 9, 11, 12, 13, 15, 16, 17
```

#Load Libraries

```r
library(atakrig)
library(raster)
```

```
## Loading required package: sp
```

```r
library(terra)
```

```
## terra 1.5.21
```

```
##
## Attaching package: 'terra'
```

```
## The following objects are masked from 'package:chron':
##
##     origin, origin<-
```

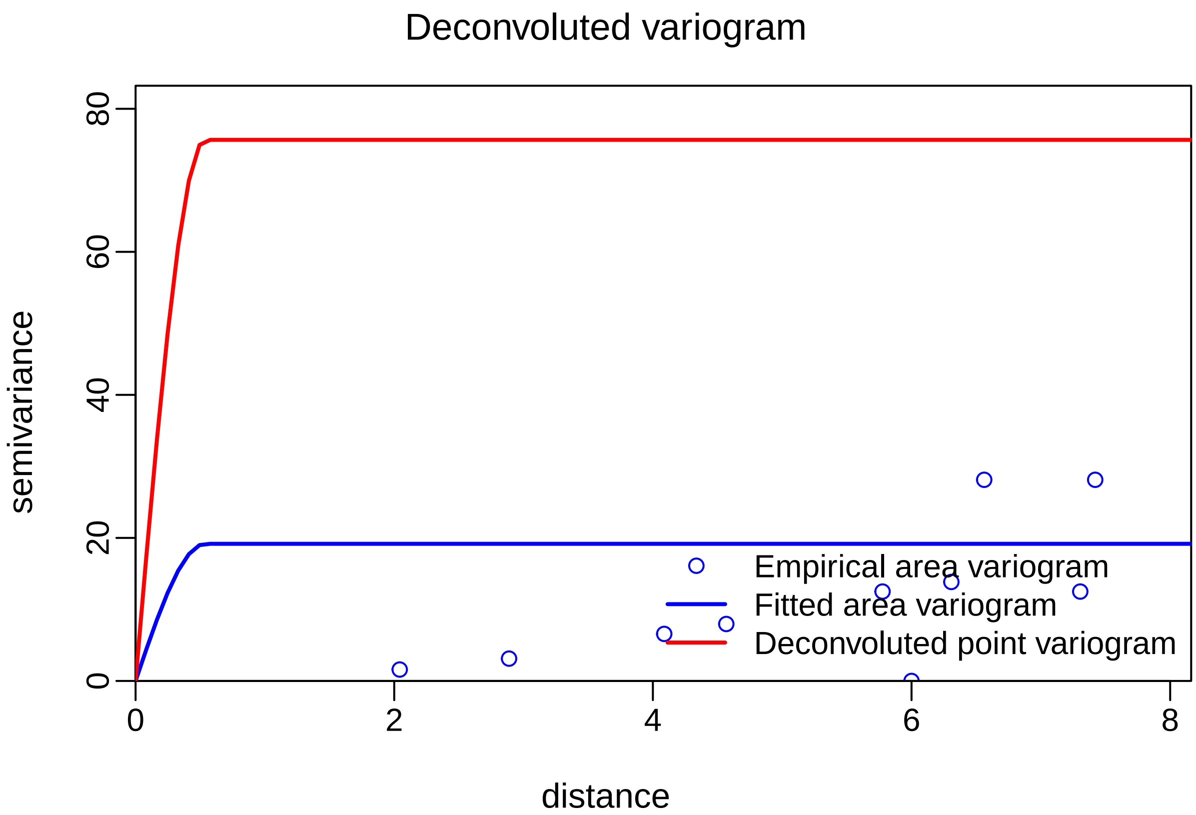
#Calculate Residuals

```r
resid_mat <- prec_mat[, 1:2]
for(m in 1:24) {
    mod <- lm(prec_mat[, 2+m] ~ prec_mat[, 1] + prec_mat[, 2])
    resid_mat <- cbind(resid_mat, resid(mod))
}
```

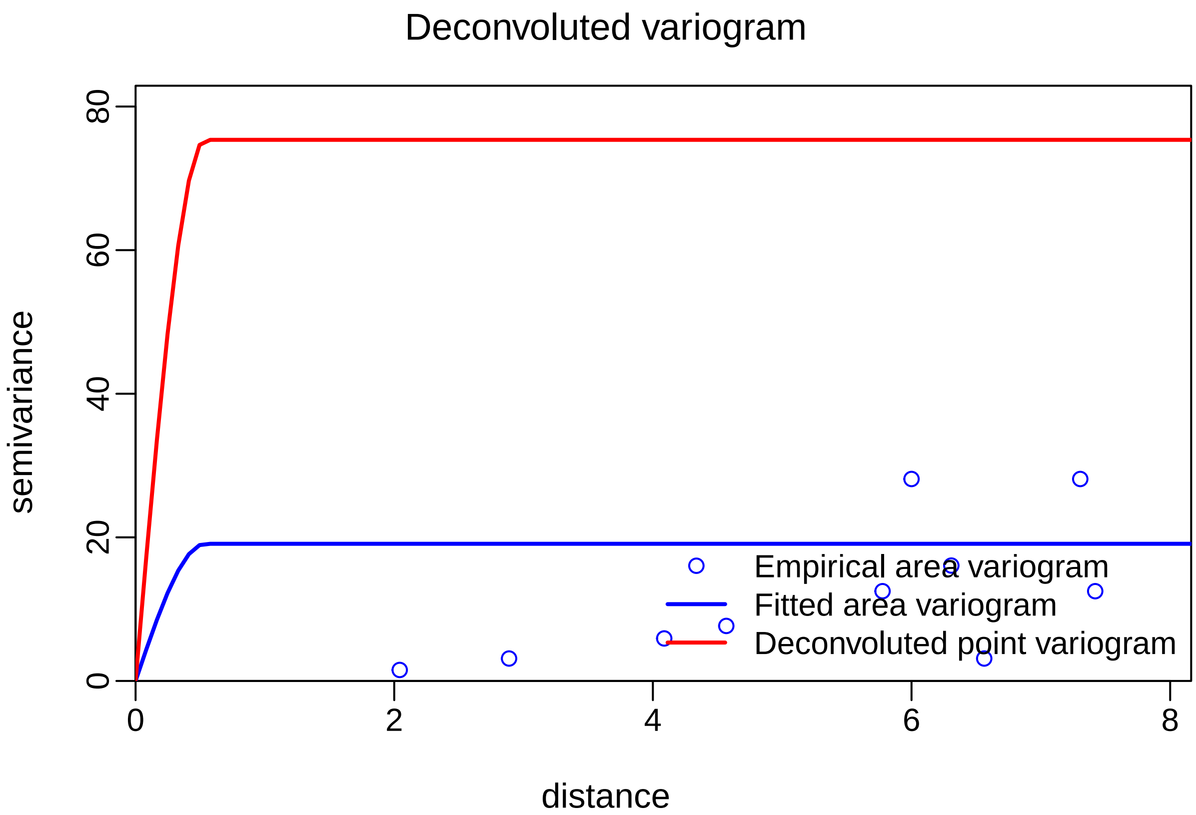
#Discretize raster to points

```r
deconv_precip<-list()
for(m in 1:24) {
  prec_ras_resid = raster(apply(matrix(resid_mat[,m], nrow=5, ncol=6, byrow=TRUE), 2, rev), xmn=-73.75,
  obs.discrete = discretizeRaster(prec_ras_resid, cellsize=480, psf = "gau", sigma = 2)
    if(m %in% c(3, 4, 6, 10, 14, 19, 20, 23, 24)) {
      deconv_precip[[m]] = deconvPointVgm(obs.discrete, model = "Gau", maxIter = 100, fig=TRUE, ngroup =
    } else if(m %in% c(1, 2, 5, 7, 8, 9, 11, 12, 13, 15, 16, 17)) {
        deconv_precip[[m]] = deconvPointVgm(obs.discrete, model = "Sph", maxIter = 100, fig=TRUE, ngrou
    } else {
        deconv_precip[[m]] = deconvPointVgm(obs.discrete, model = "Exp", maxIter = 100, fig=TRUE, ngrou
    }
}
```

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```

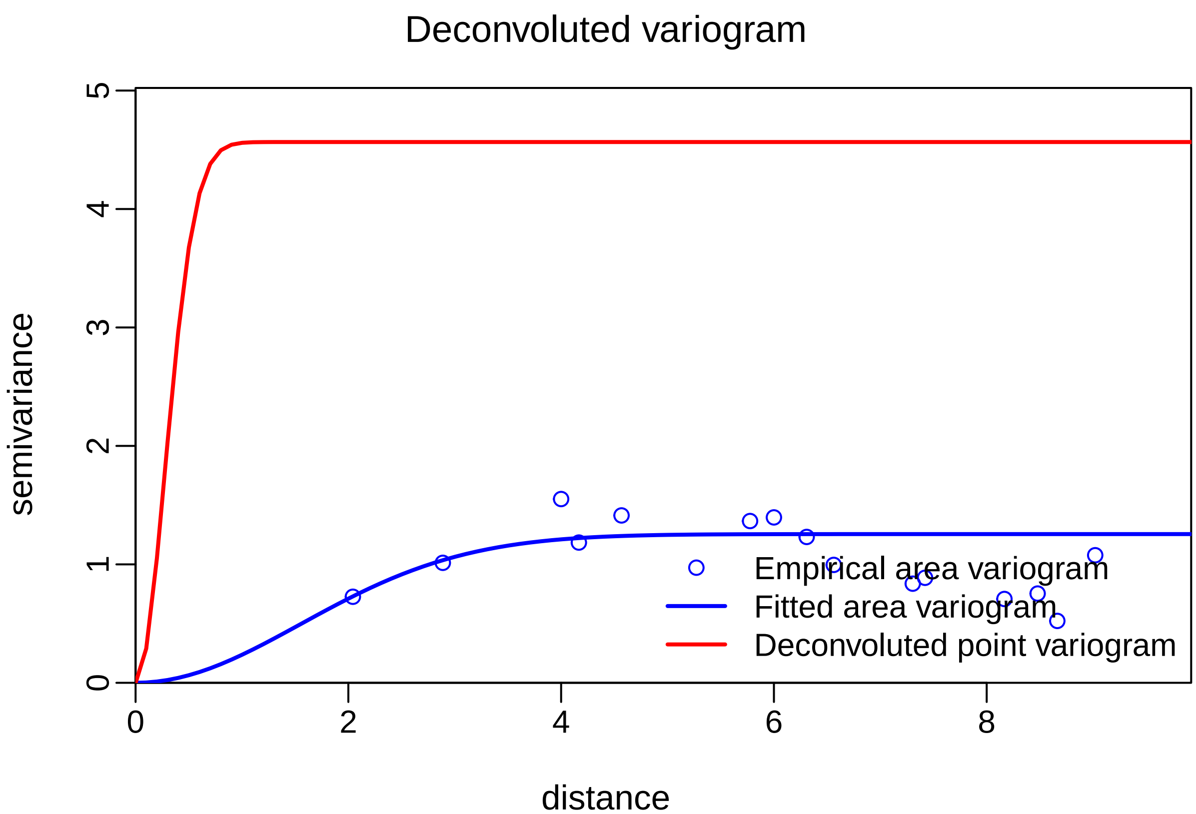



```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
## iterating: 16
## iterating: 17
## iterating: 18
```



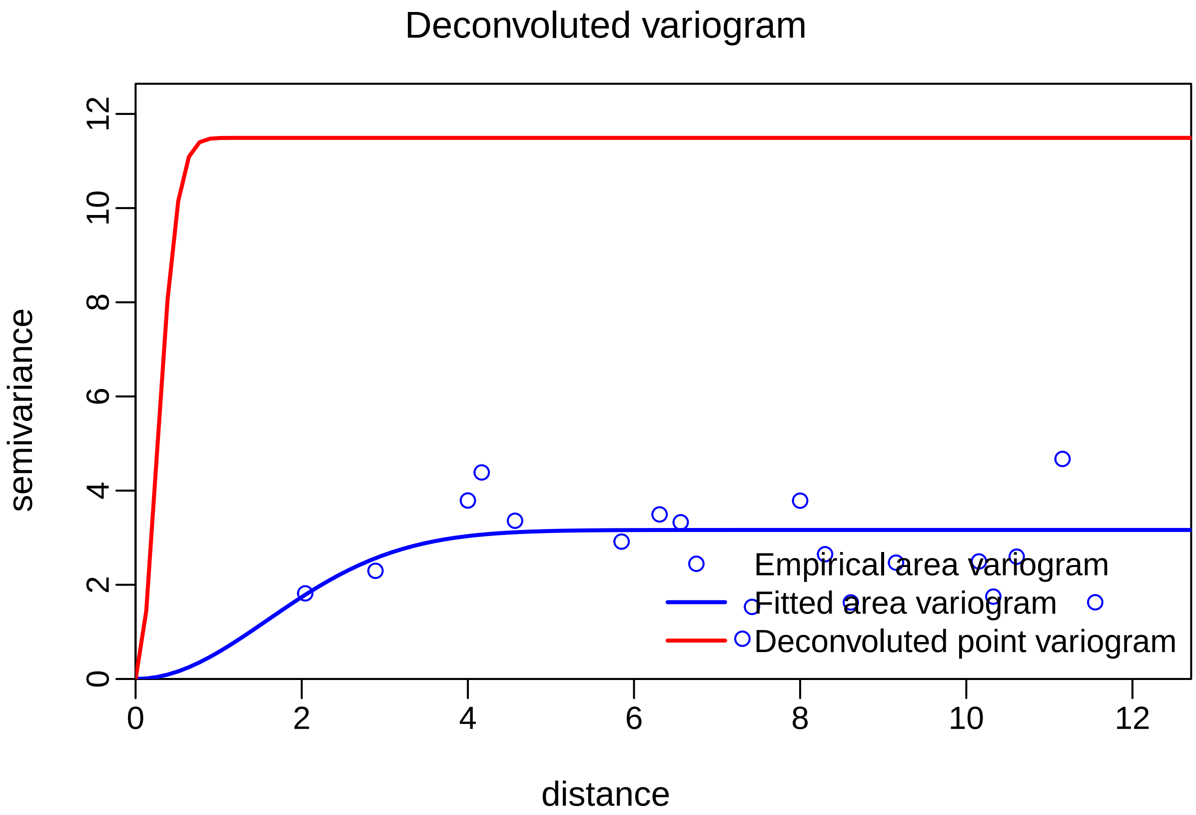

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
## iterating: 16
## iterating: 17
## iterating: 18
## iterating: 19
```

```
## iterating: 20
## iterating: 21
## iterating: 22
## iterating: 23
## iterating: 24
## iterating: 25
## iterating: 26
## iterating: 27
## iterating: 28
## iterating: 29
## iterating: 30
```



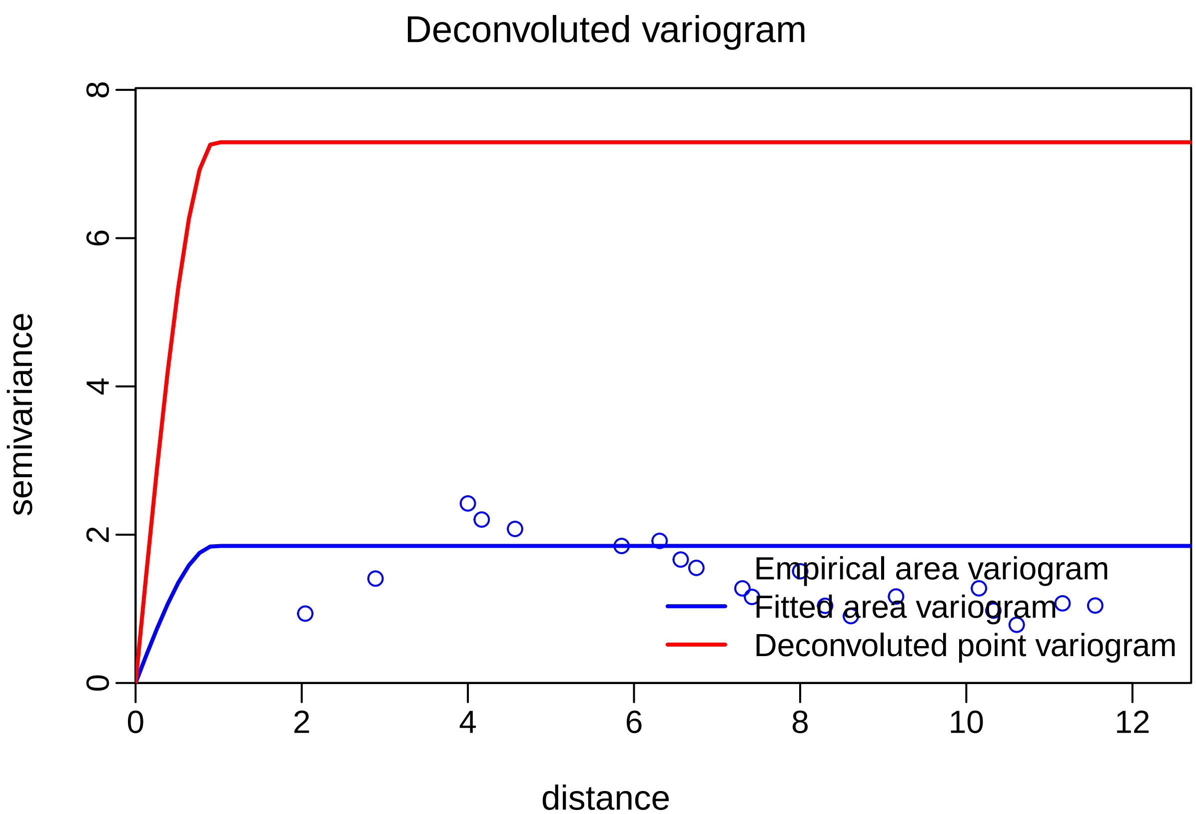

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
```

```
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```



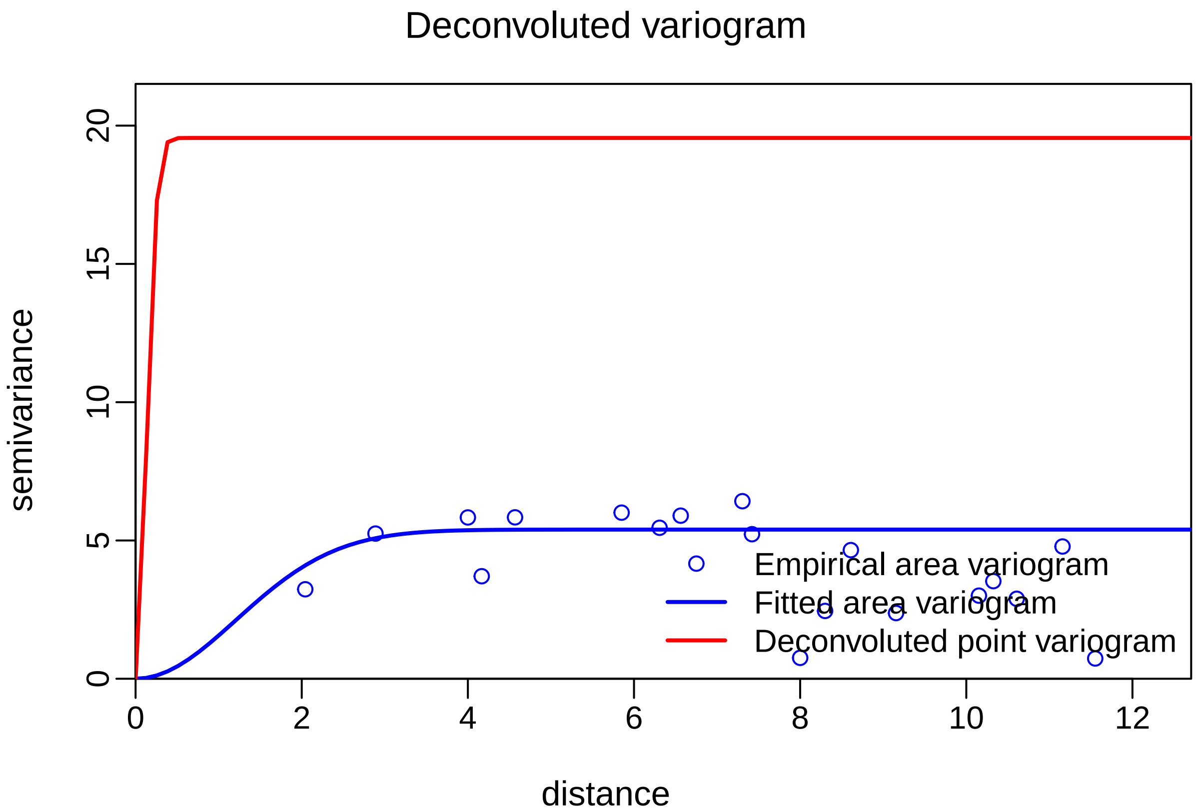

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
```

```
## iterating: 16
## iterating: 17
## iterating: 18
## iterating: 19
## iterating: 20
## iterating: 21
## iterating: 22
## iterating: 23
## iterating: 24
## iterating: 25
```

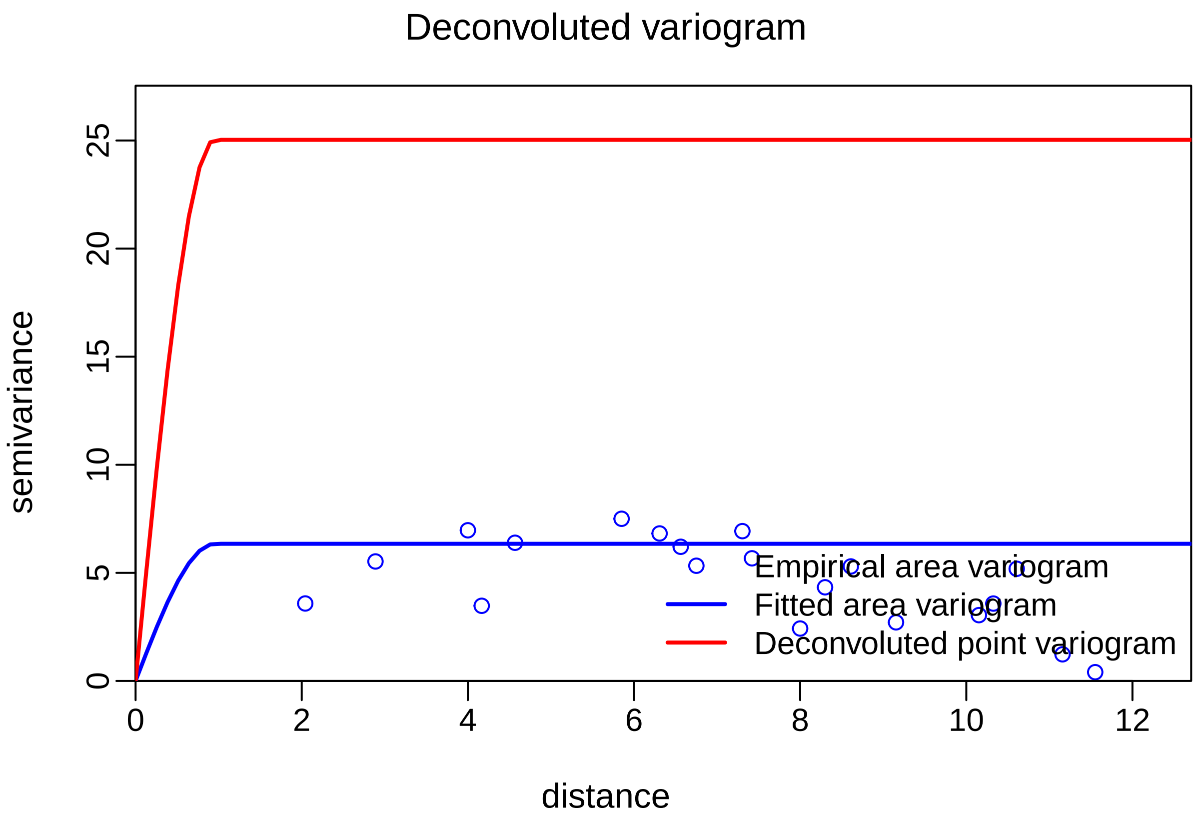



```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
```

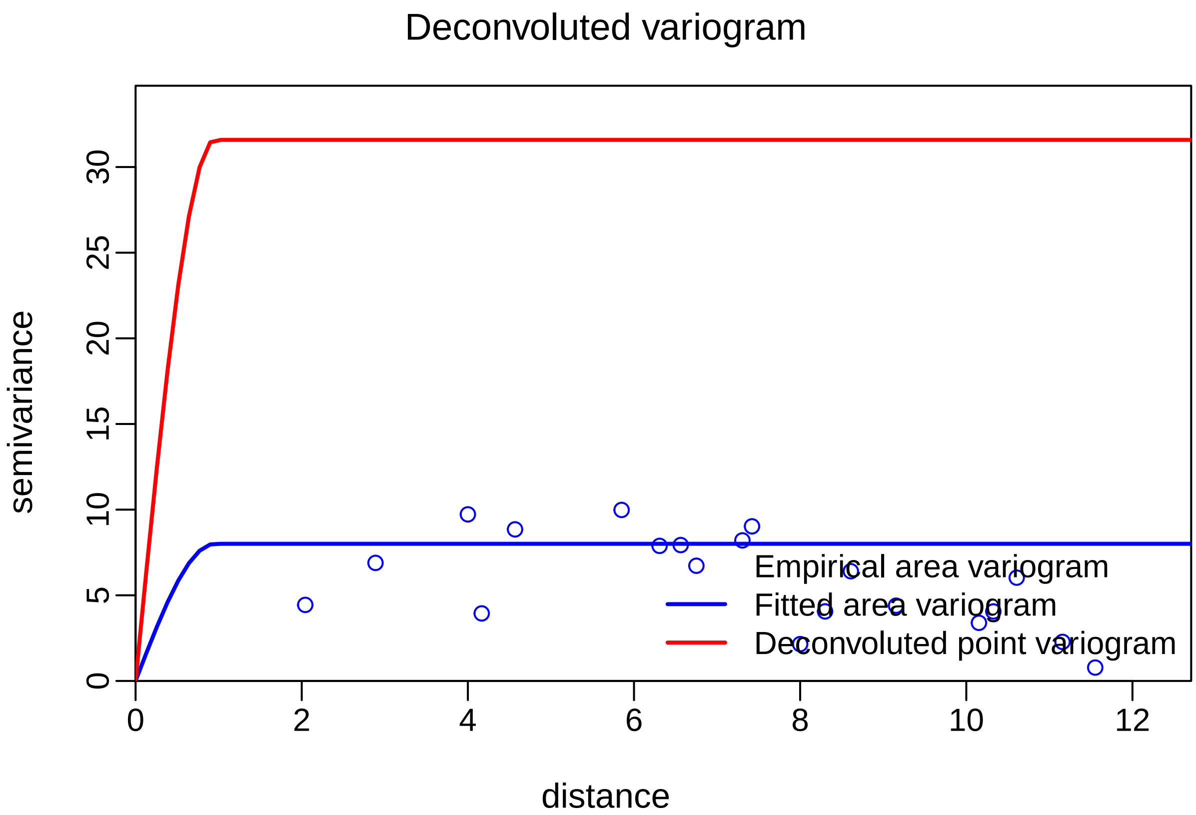
```
## iterating: 10

## iterating: 11

## iterating: 12
```




```
## iterating: 1

## iterating: 2

## iterating: 3

## iterating: 4

## iterating: 5

## iterating: 6

## iterating: 7

## iterating: 8

## iterating: 9

## iterating: 10

## iterating: 11

## iterating: 12
```

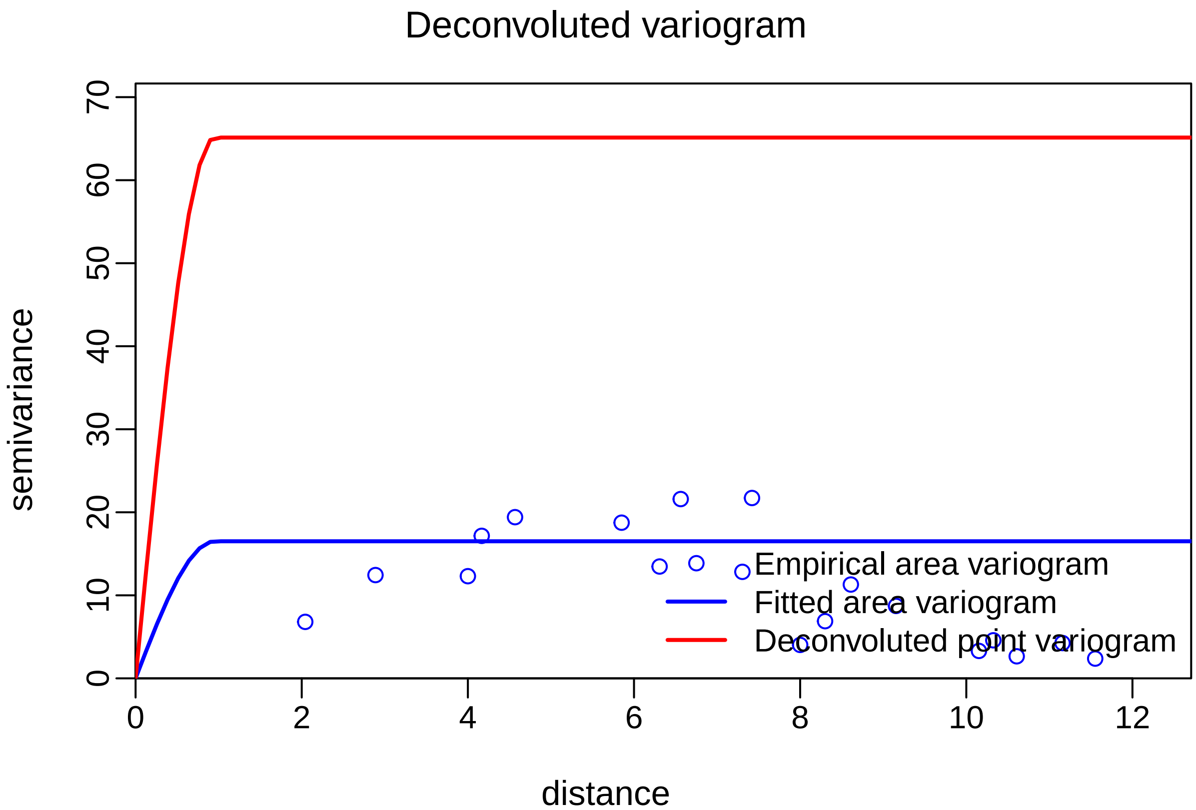



```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```



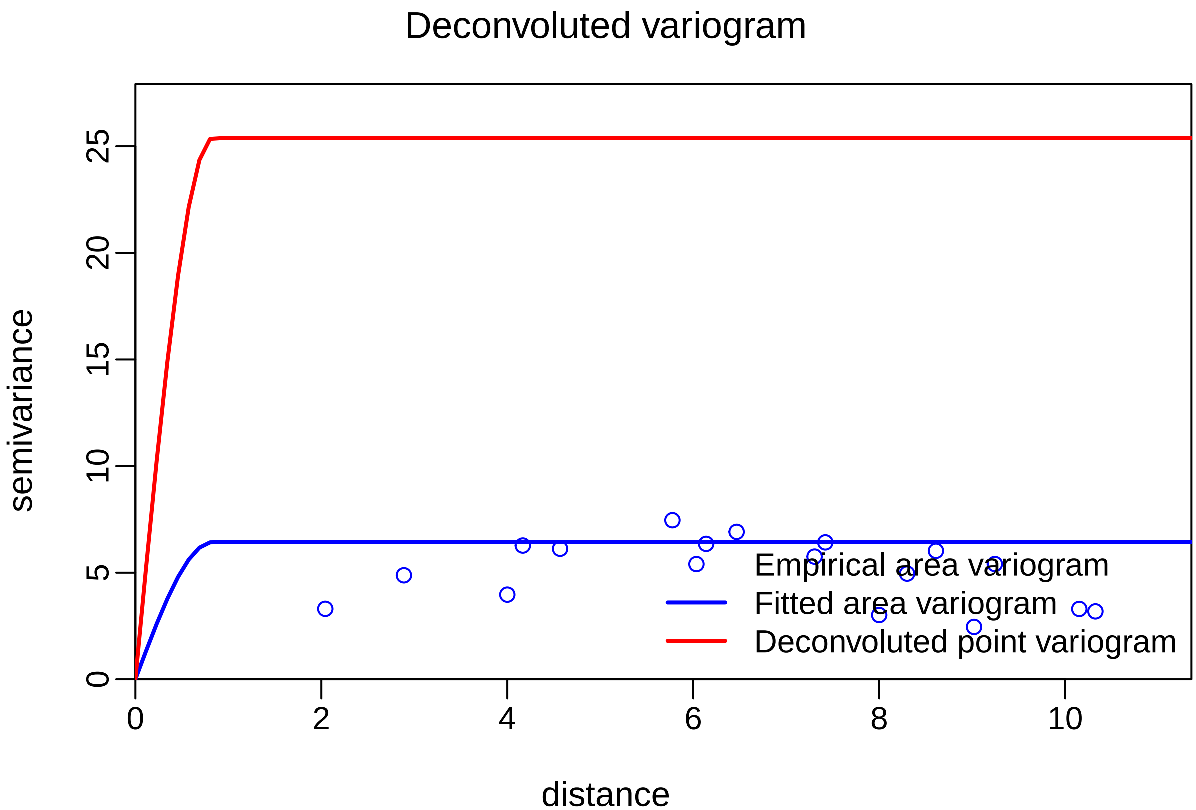

```
## iterating: 1

## iterating: 2

## iterating: 3

## iterating: 4

## iterating: 5

## iterating: 6

## iterating: 7

## iterating: 8

## iterating: 9

## iterating: 10

## iterating: 11

## iterating: 12

## iterating: 13

##   model    psill     range
## 1   Gau 2.185259 -27.31036

## iterating: 1

## iterating: 2

## iterating: 3

## iterating: 4
```

```
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```

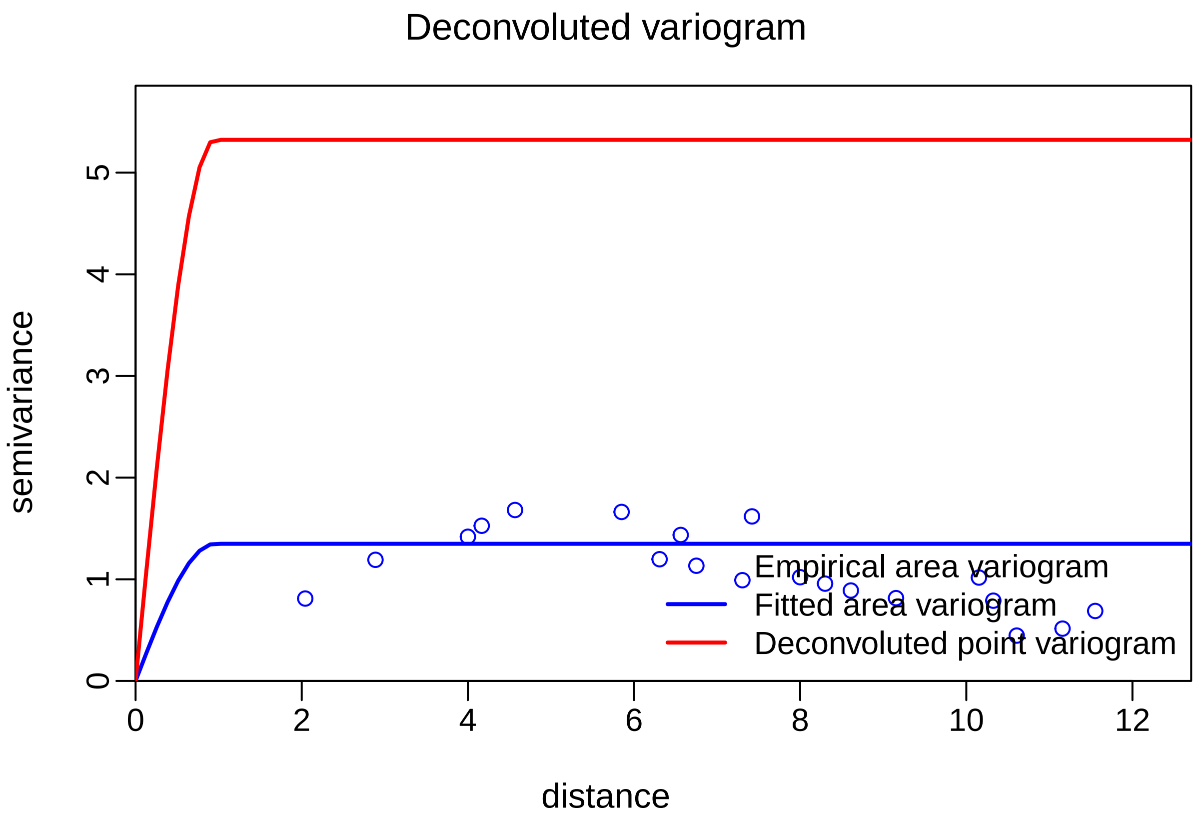



```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
```

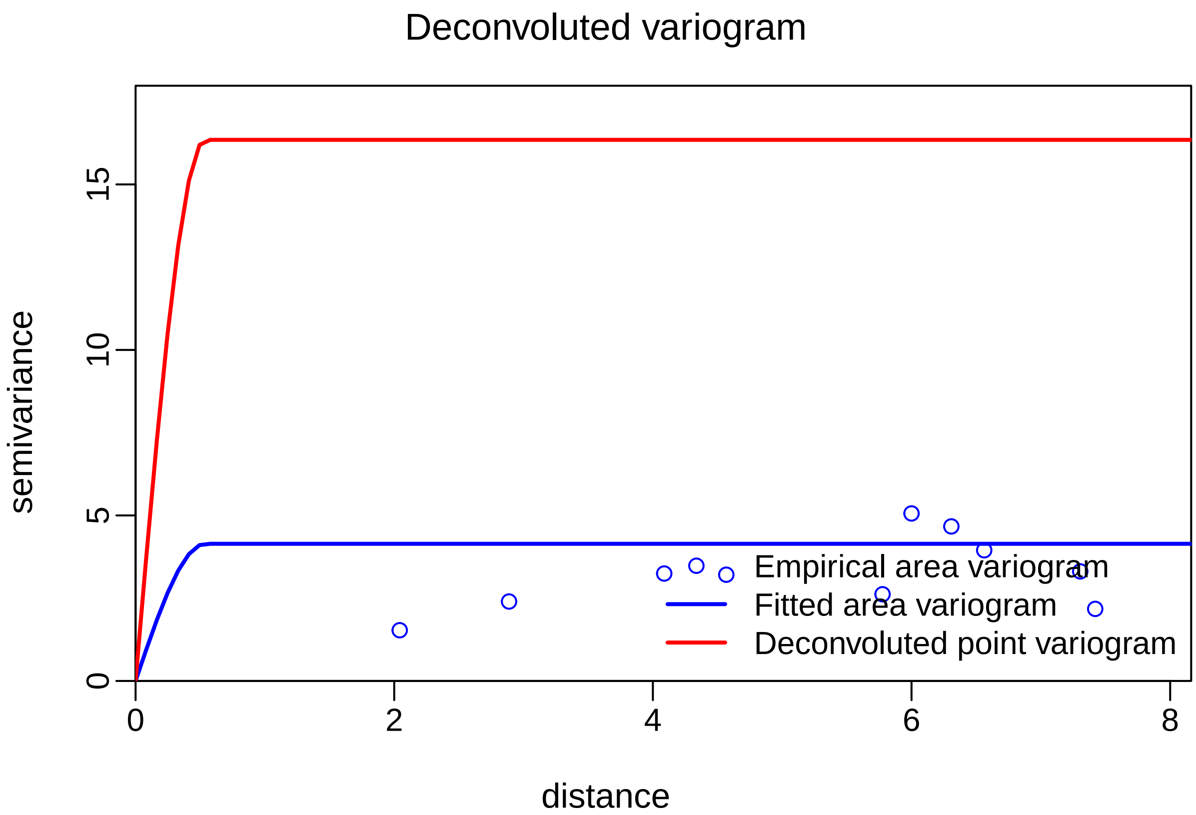
```
## iterating: 12
```



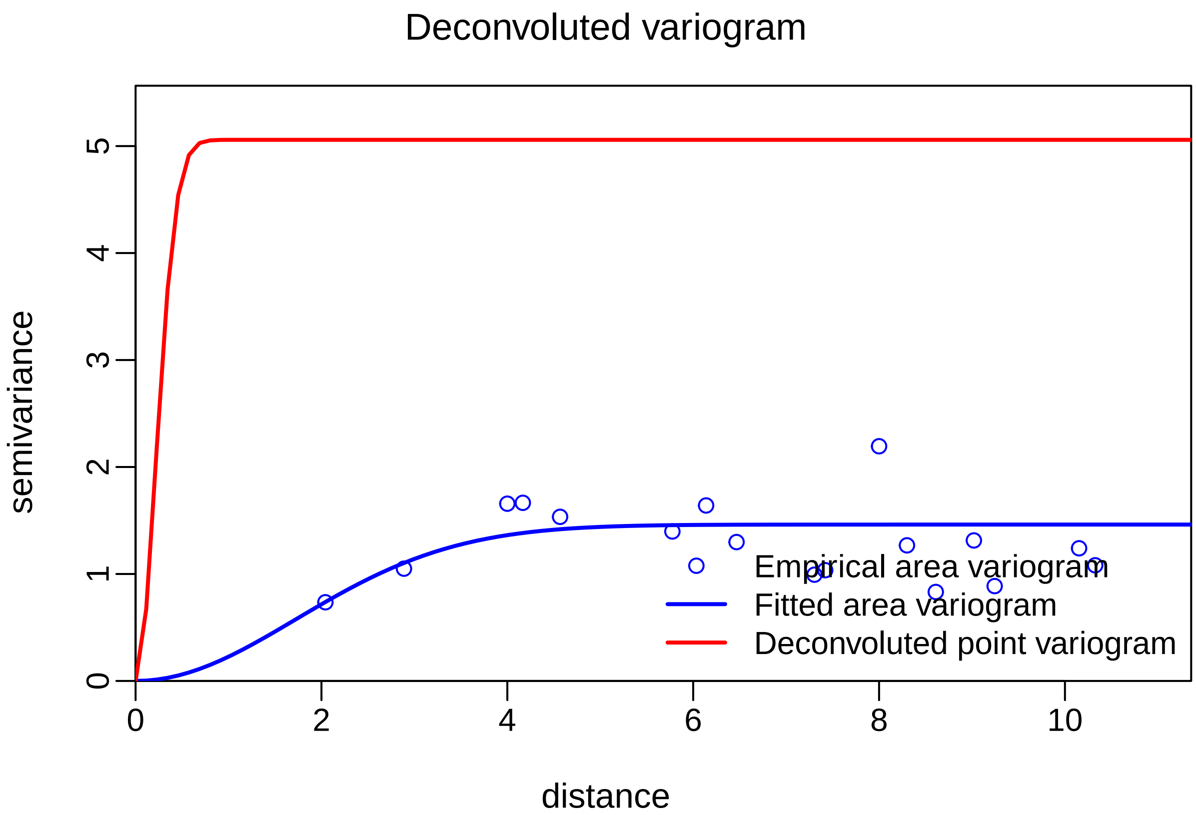

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```



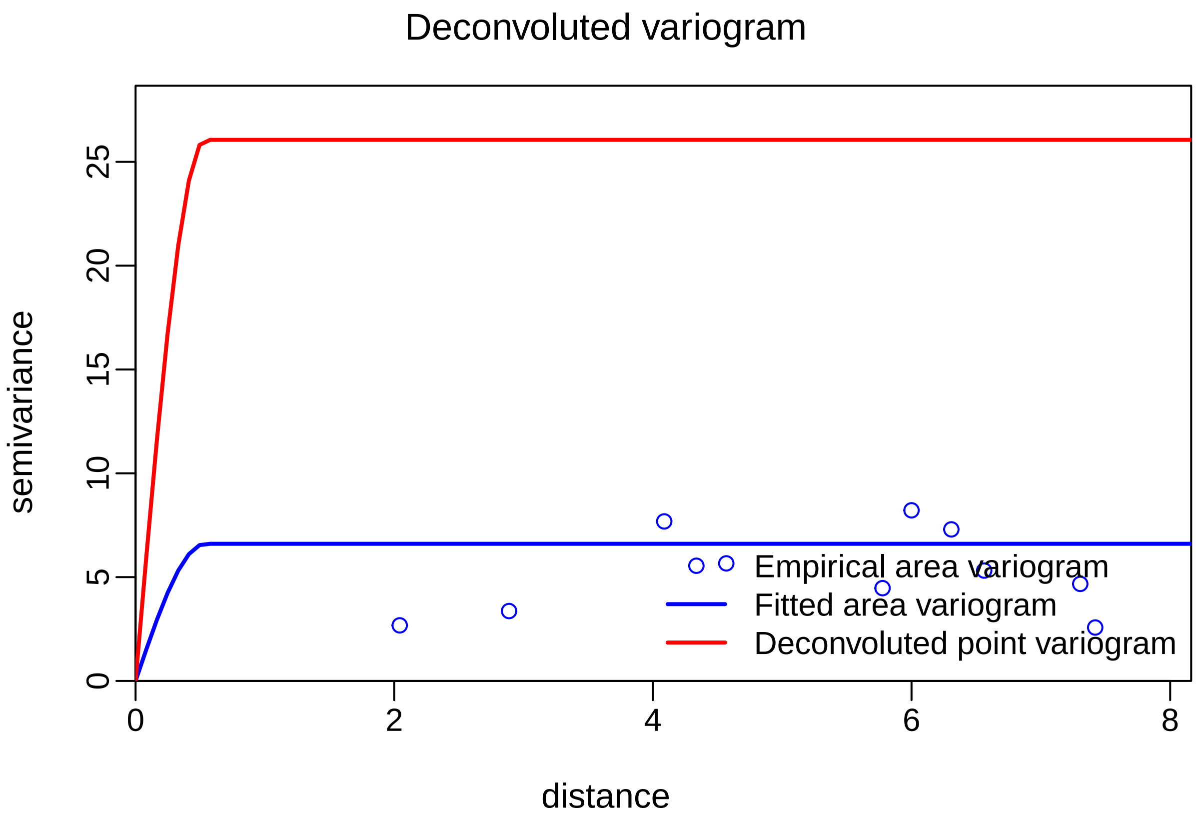

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
## iterating: 16
```



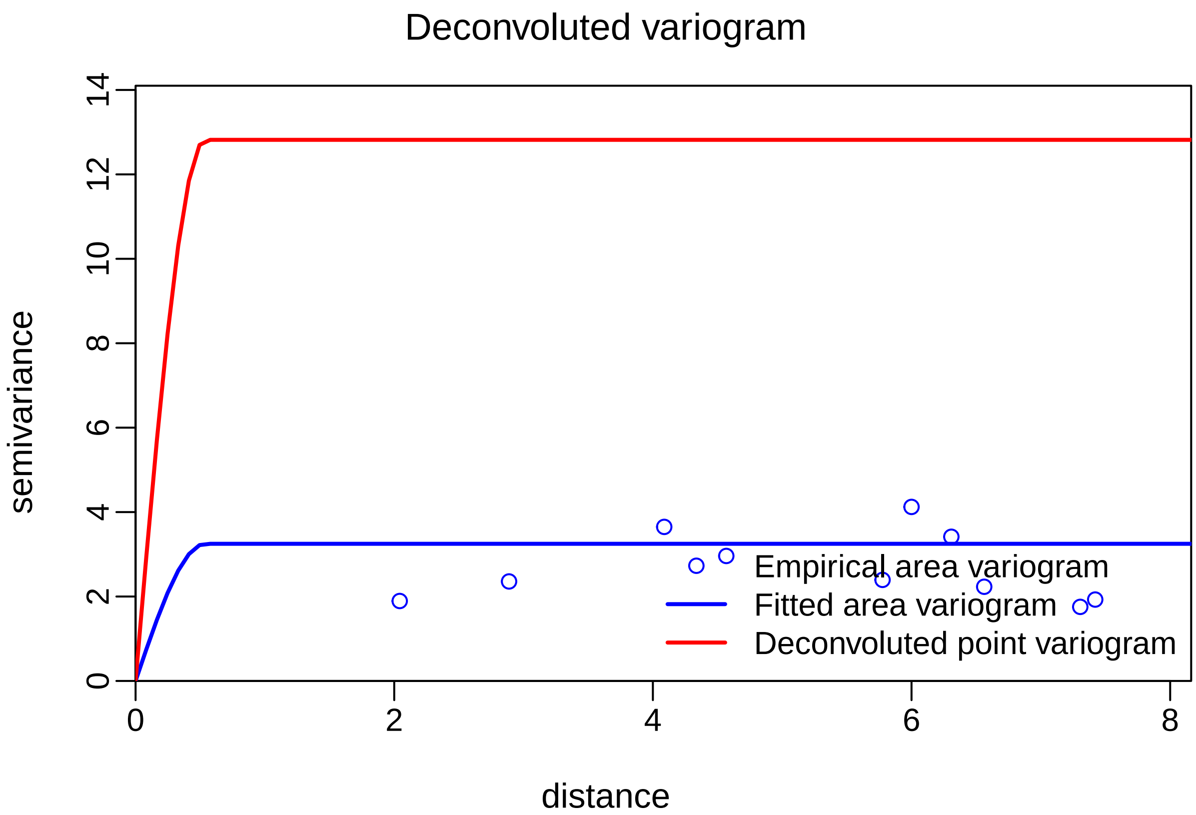

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```




```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```

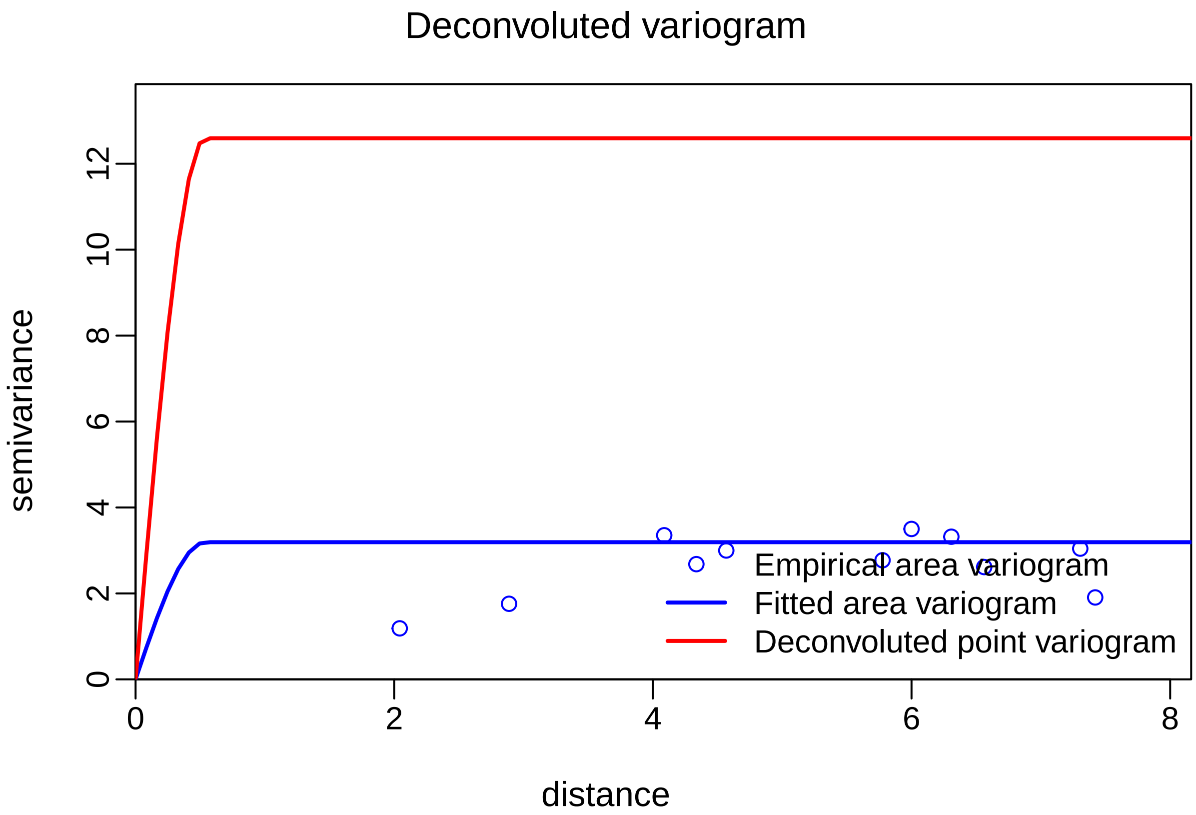



```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
```



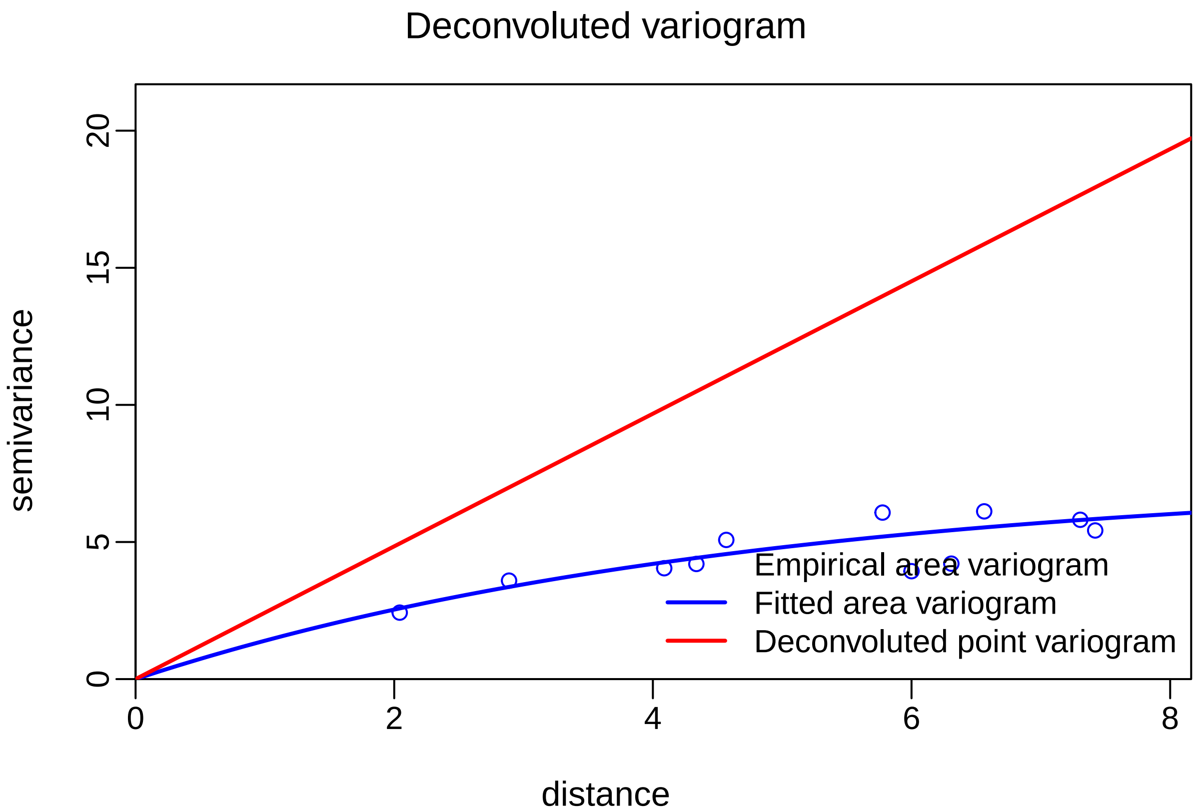

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
## iterating: 16
## iterating: 17
## iterating: 18
## iterating: 19
```

```
## iterating: 20
## iterating: 21
```

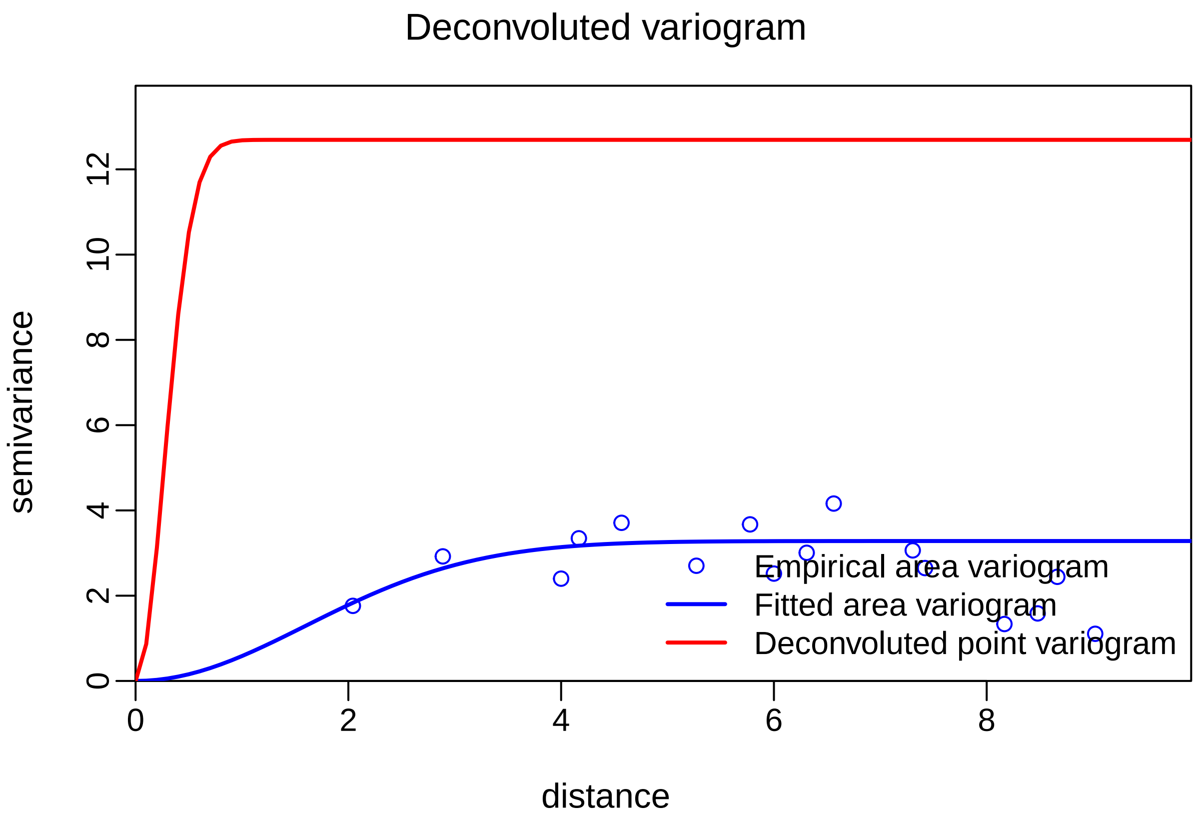



```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
## iterating: 16
## iterating: 17
```

```
## iterating: 18
```

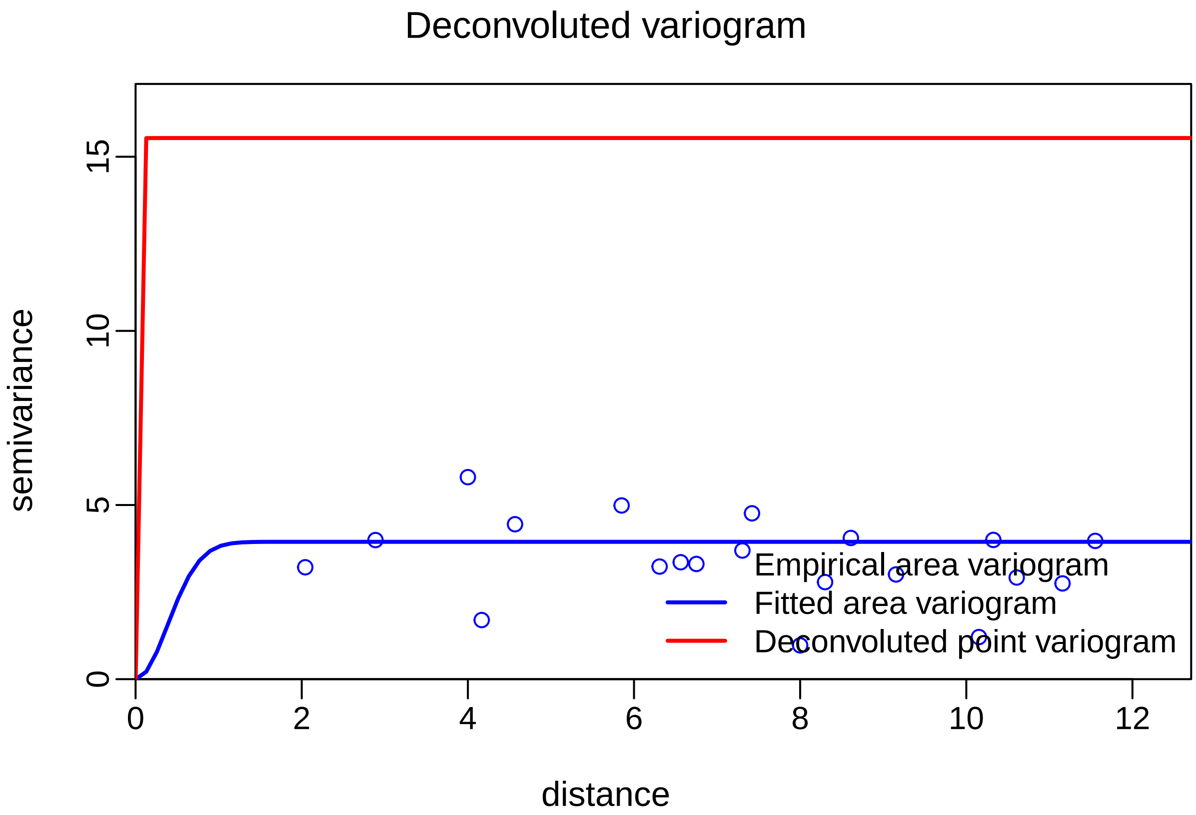



```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
```



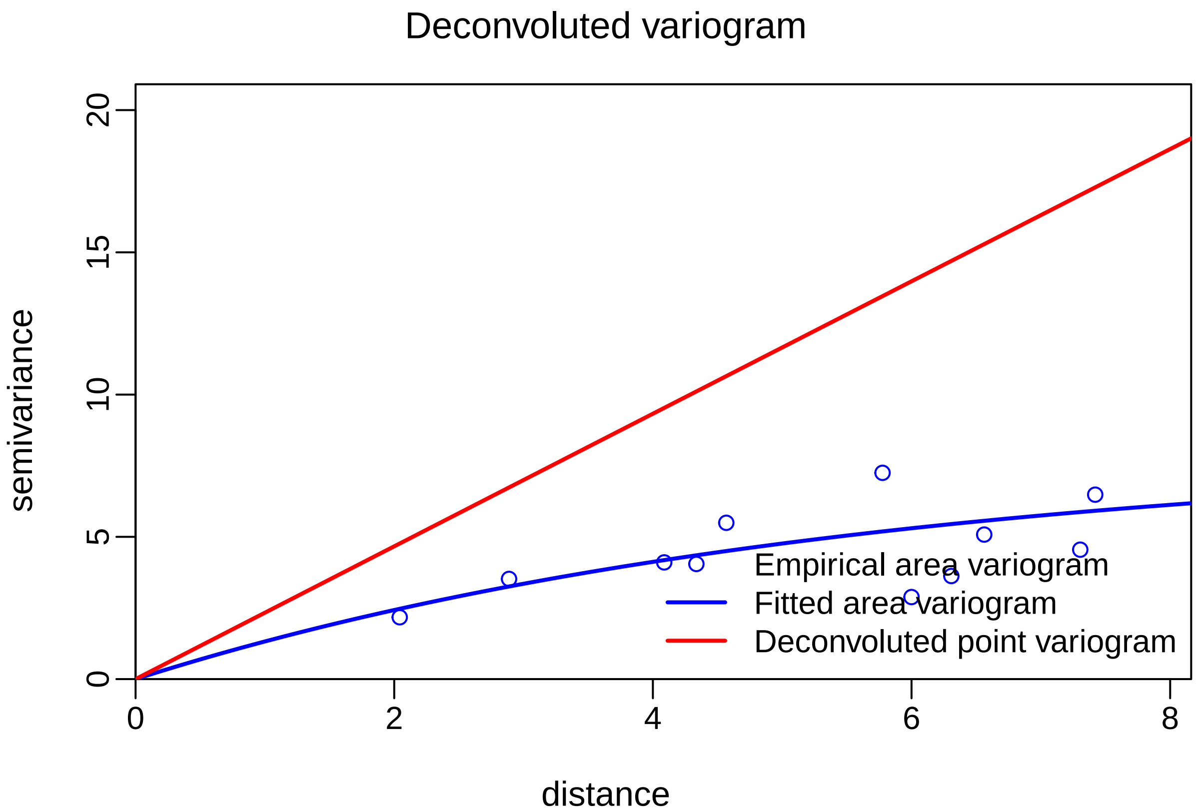

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
## iterating: 16
## iterating: 17
## iterating: 18
## iterating: 19
```

```
## iterating: 20

## iterating: 21

## iterating: 22
```

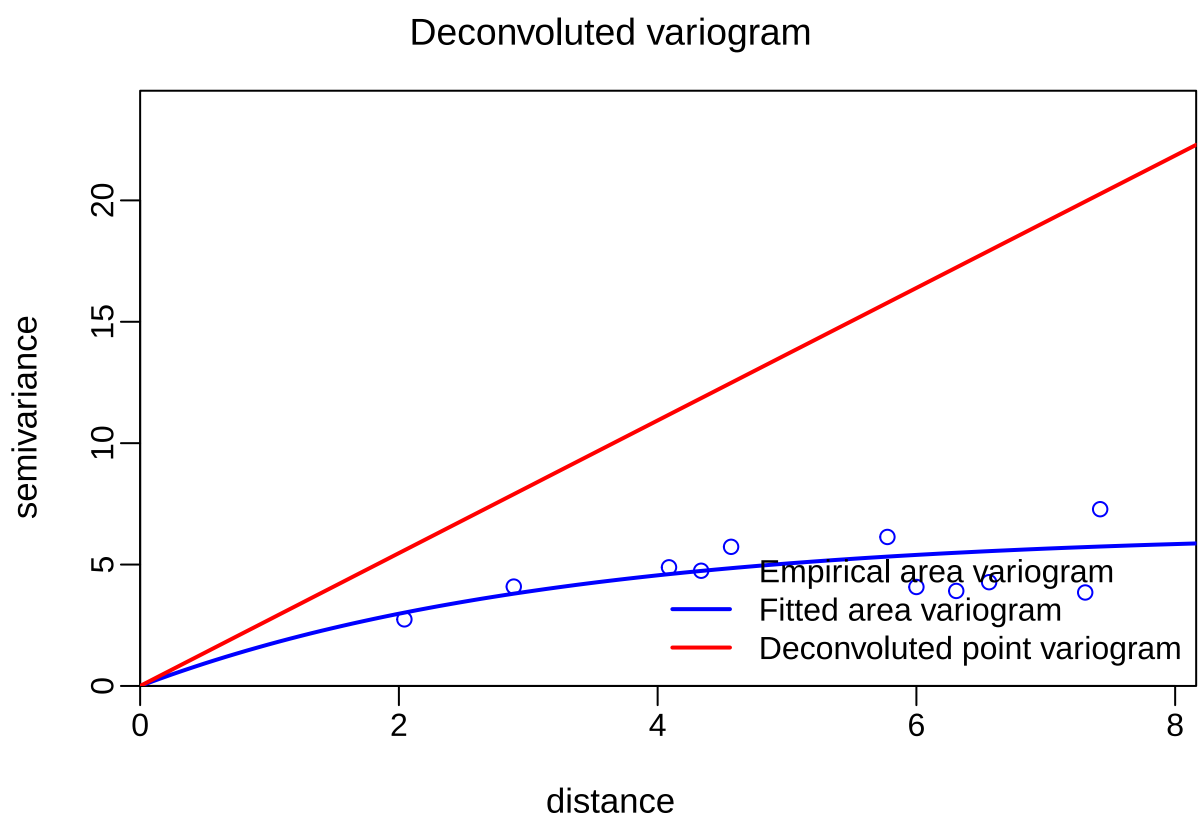



```
## iterating: 1

## iterating: 2

## iterating: 3

## iterating: 4

## iterating: 5

## iterating: 6

## iterating: 7

## iterating: 8

## iterating: 9

## iterating: 10

## iterating: 11

## iterating: 12

## iterating: 13

## iterating: 14

## iterating: 15

## iterating: 16
```

```
## iterating: 17
```

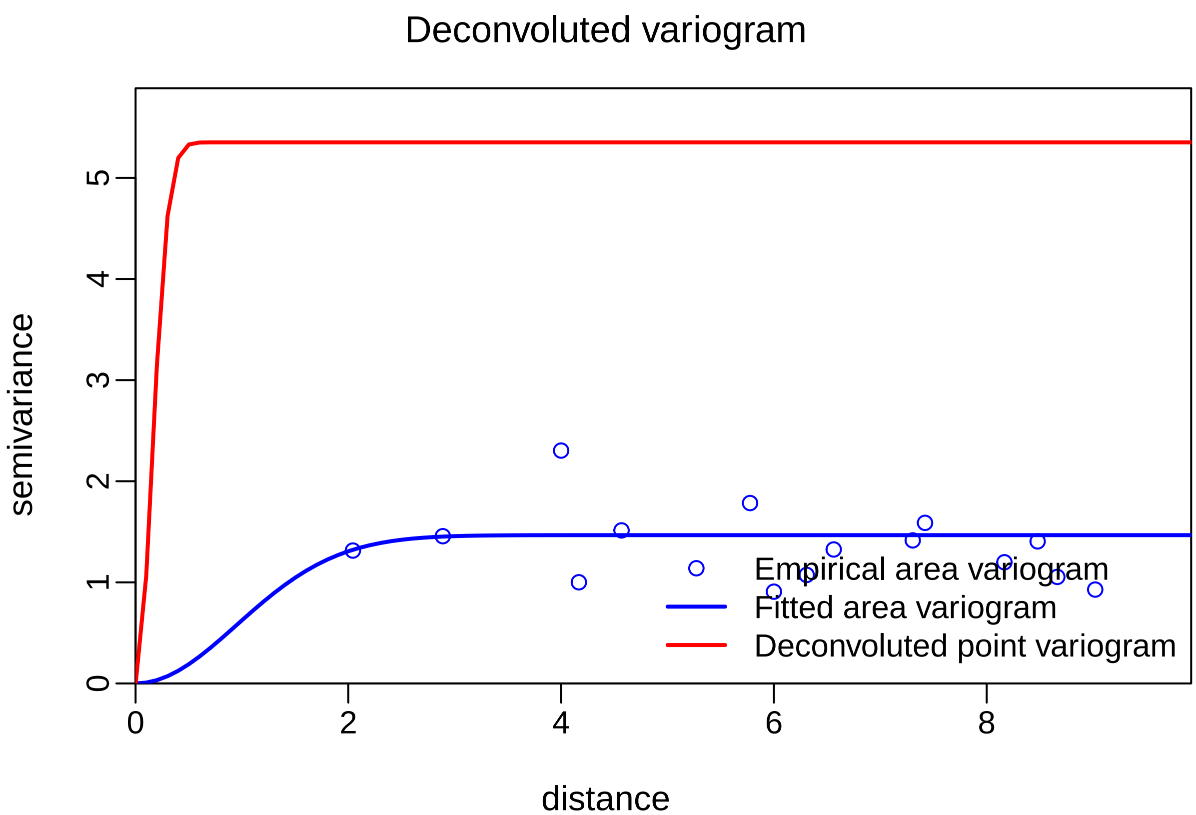


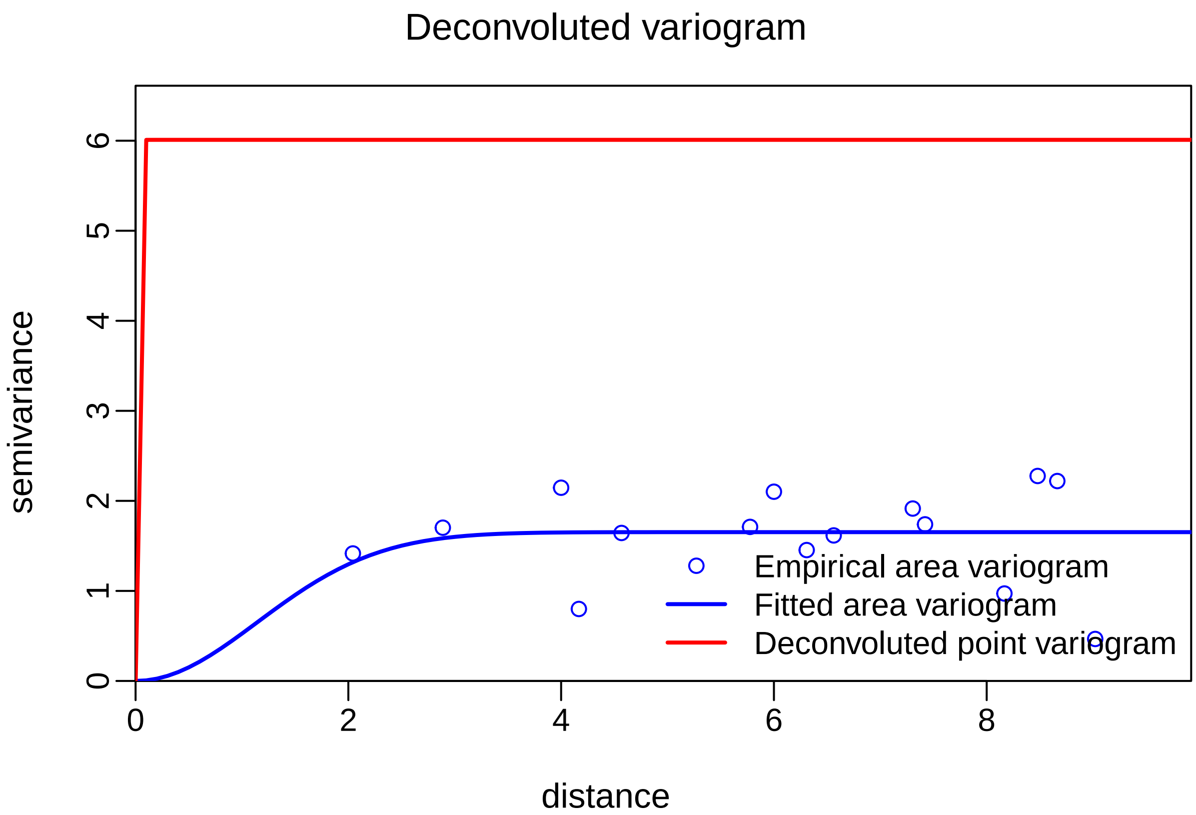

```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
```




```
## iterating: 1
## iterating: 2
## iterating: 3
## iterating: 4
## iterating: 5
## iterating: 6
## iterating: 7
## iterating: 8
## iterating: 9
## iterating: 10
## iterating: 11
## iterating: 12
## iterating: 13
## iterating: 14
## iterating: 15
## iterating: 16
## iterating: 17
## iterating: 18
## iterating: 19
```

```
## iterating: 20
## iterating: 21
## iterating: 22
## iterating: 23
## iterating: 24
## iterating: 25
## iterating: 26
```

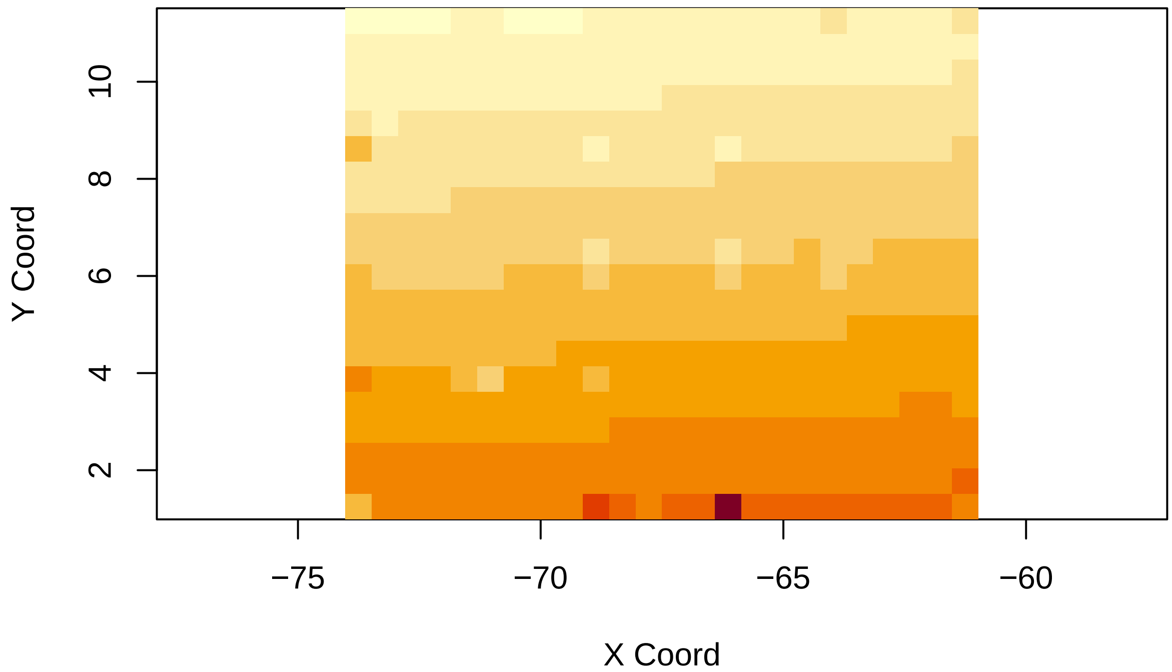




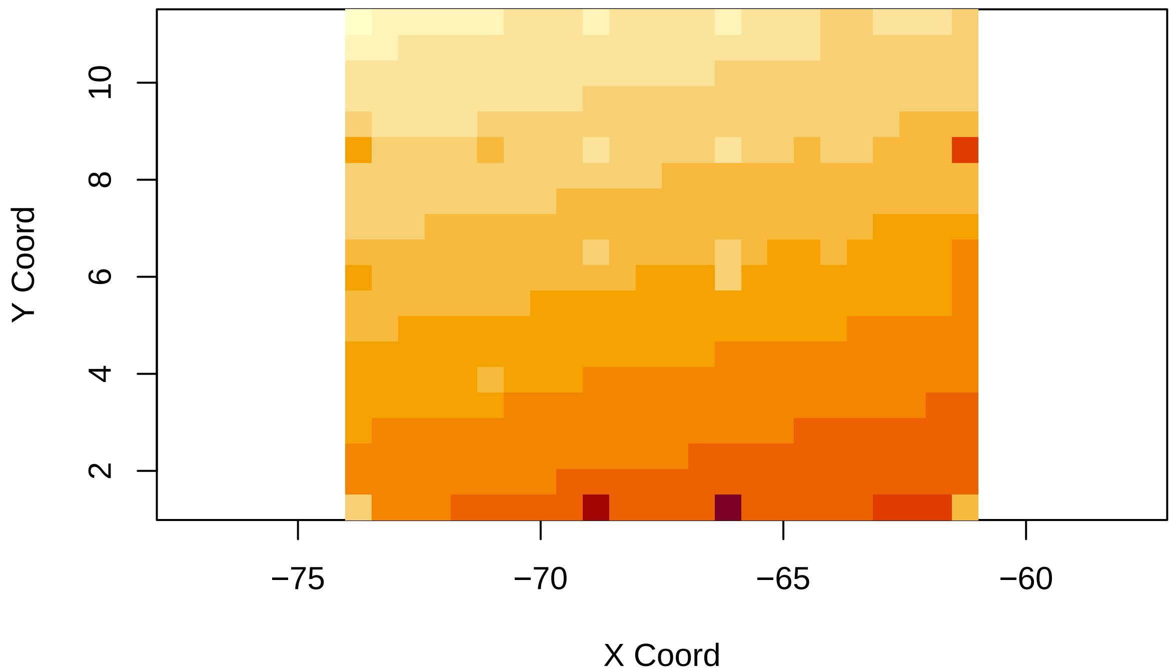
#Make Kriging predictions

```
new_lonlat = expand.grid(x = seq(-73.75,-61.25,length.out = 24), y = seq(1.25,11.25,length.out = 20))

kriging_precip = list()
for(m in 1:24) {
    if(m %in% c(3, 4, 6, 10, 14, 19, 20, 23, 24)) {
      krige_para = krige.control(type.krige = "ok", trend.d = "1st", trend.l = "1st", obj.model = NULL,
      kriging_precip[[m]] = krige.conv(geo_precip_list[[m]], coords=geo_precip_list[[m]]$coords, data=ge
    } else if(m %in% c(1, 2, 5, 7, 8, 9, 11, 12, 13, 15, 16, 17)) {
        krige_para = krige.control(type.krige = "ok", trend.d = "1st", trend.l = "1st", obj.model = NULL
      kriging_precip[[m]] = krige.conv(geo_precip_list[[m]], coords=geo_precip_list[[m]]$coords, data=ge
    } else {
        krige_para = krige.control(type.krige = "ok", trend.d = "1st", trend.l = "1st", obj.model = NULL
      kriging_precip[[m]] = krige.conv(geo_precip_list[[m]], coords=geo_precip_list[[m]]$coords, data=ge
    }
    image(kriging_precip[[m]], locations = new_lonlat)
}
```

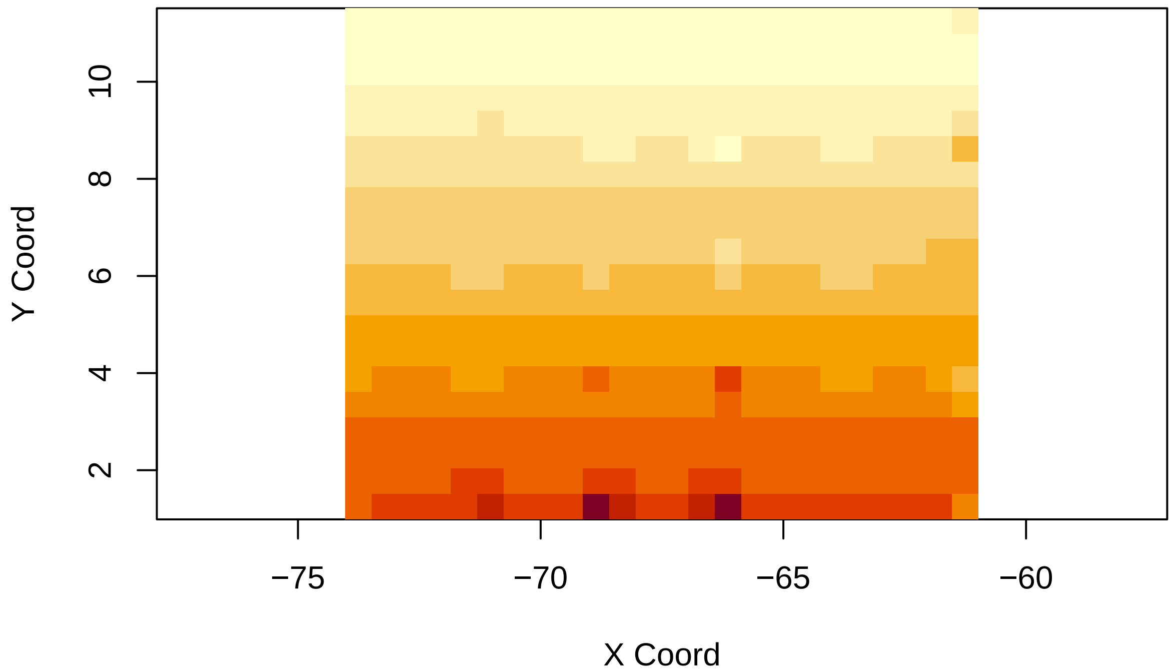
```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



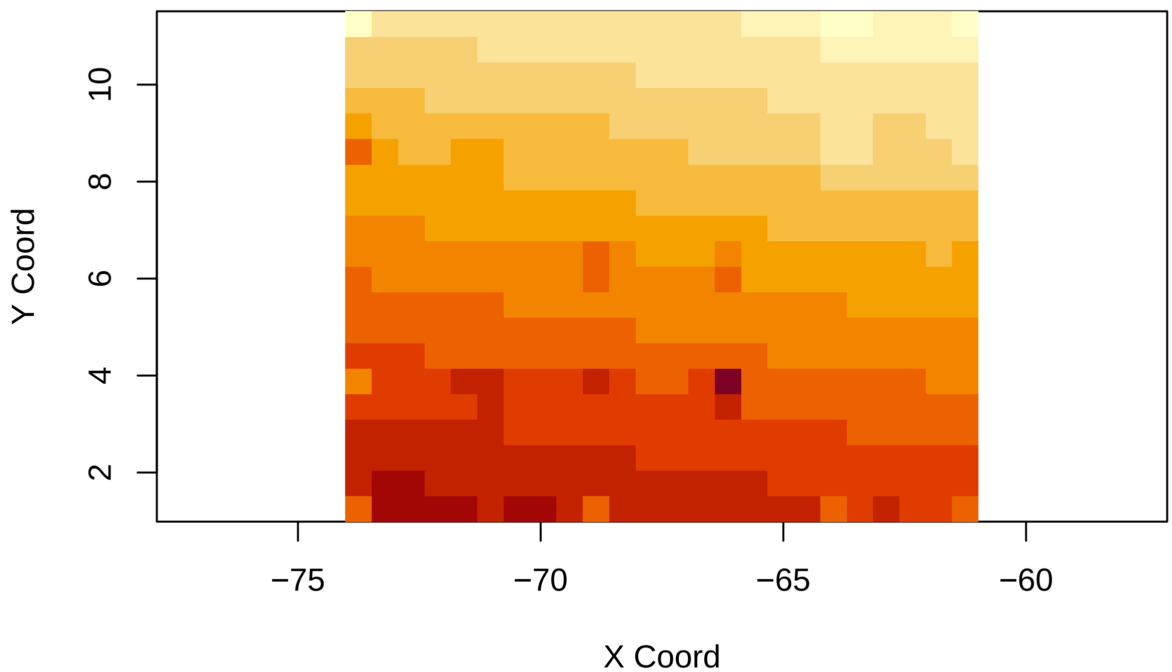

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



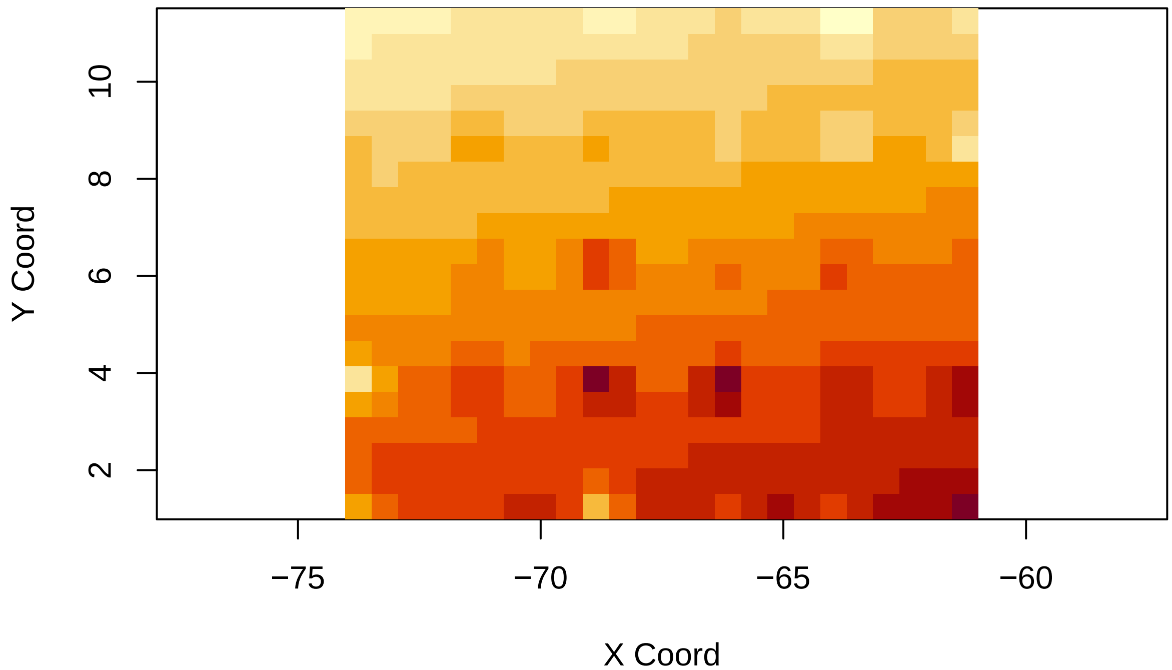

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



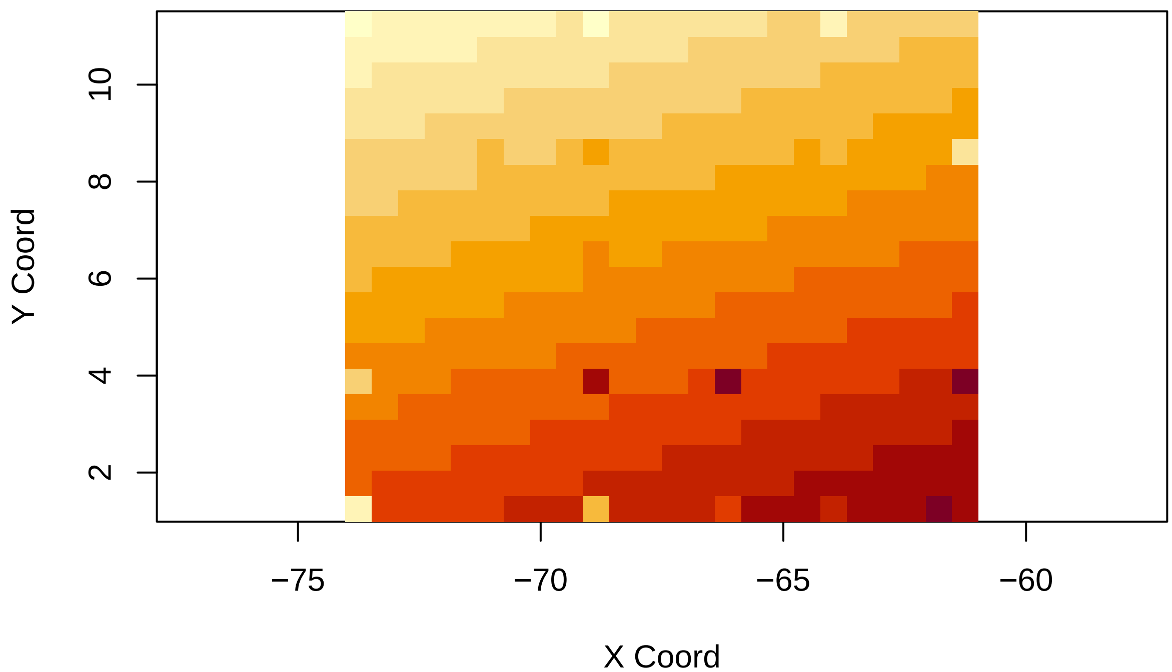

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



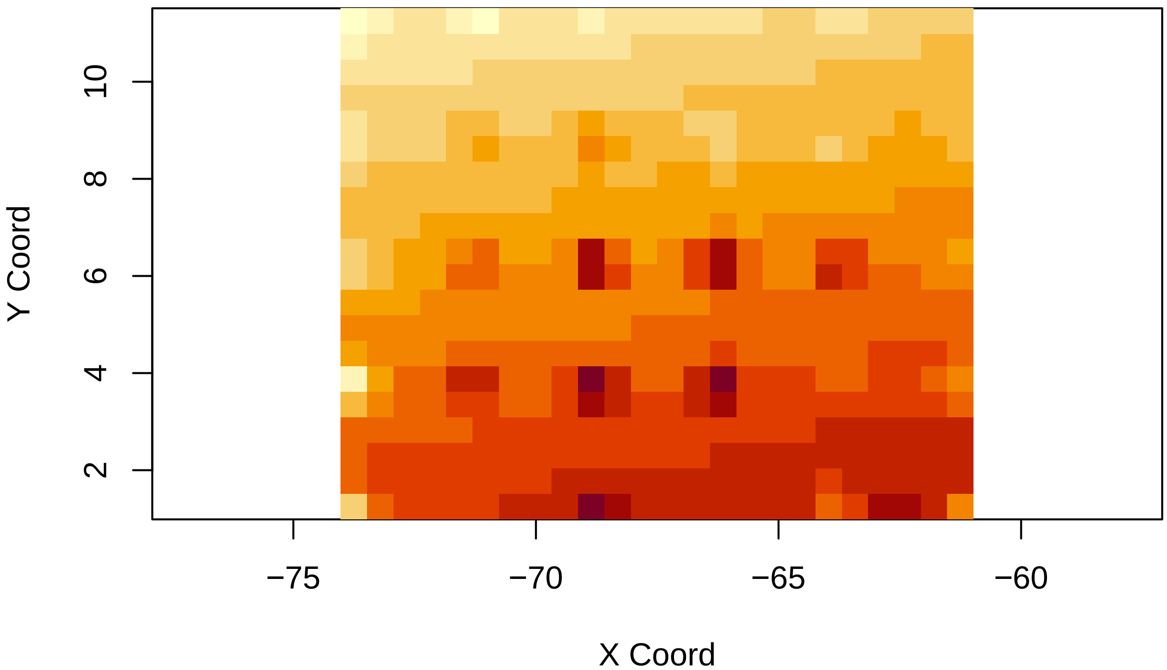

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



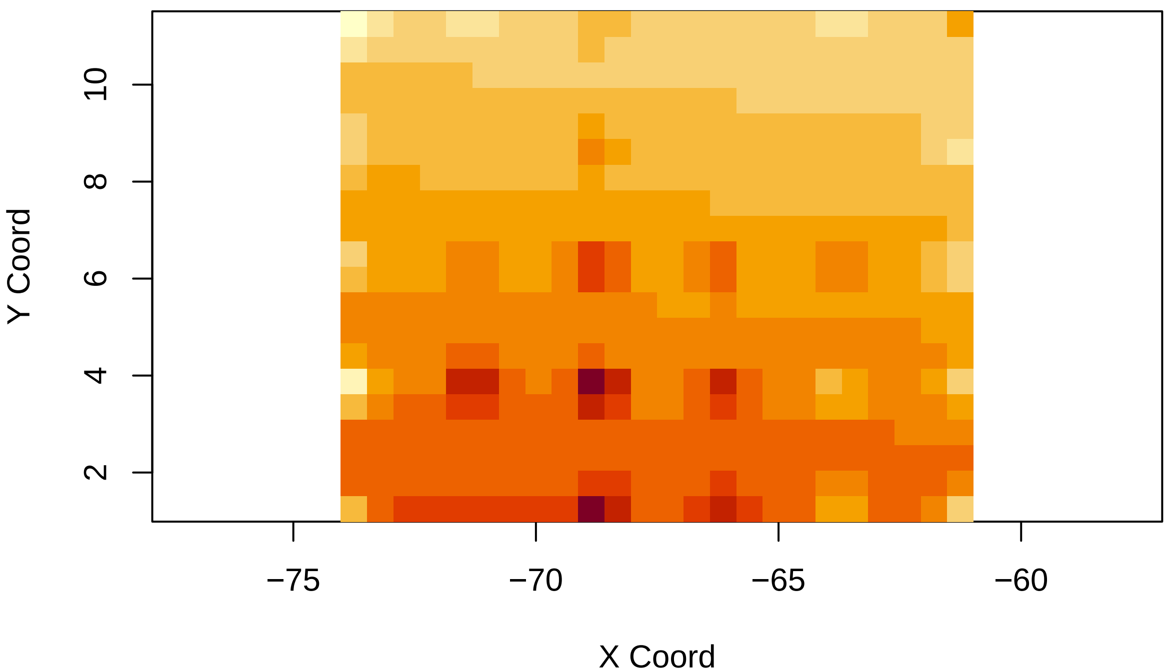

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



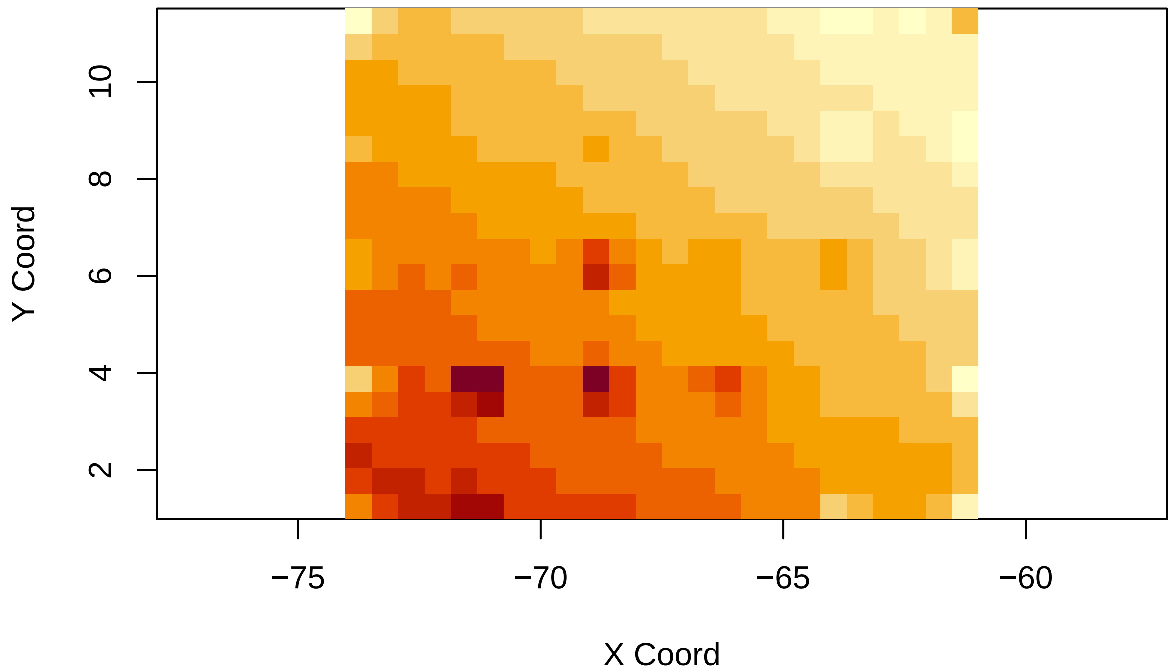

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



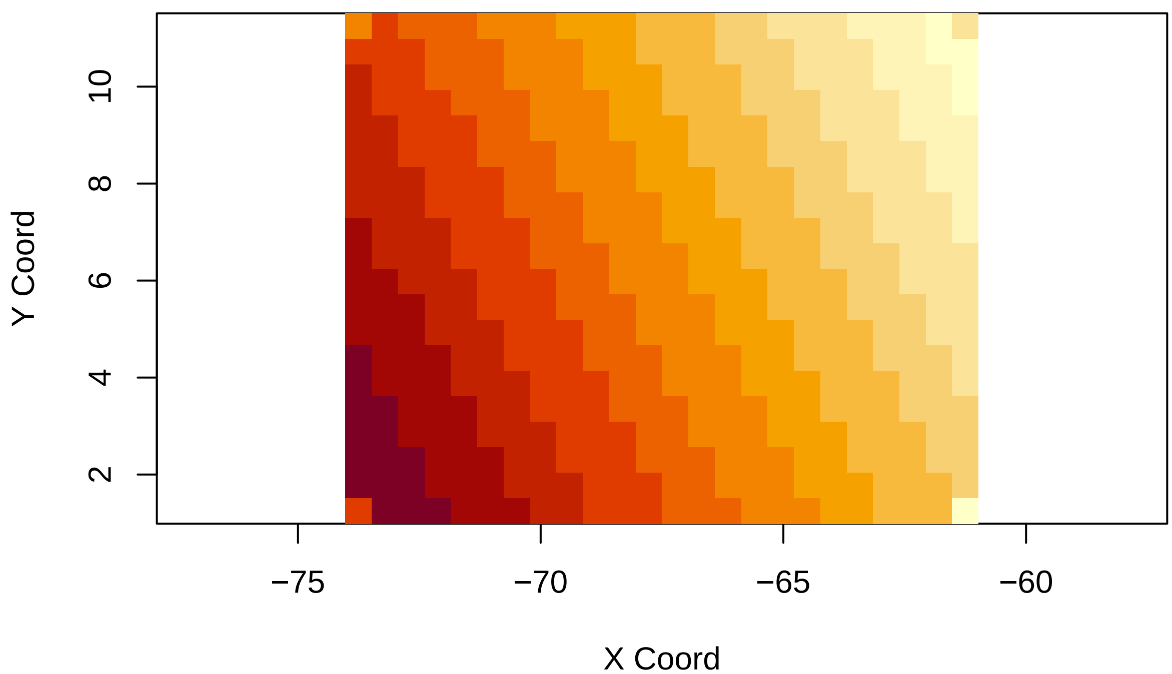

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



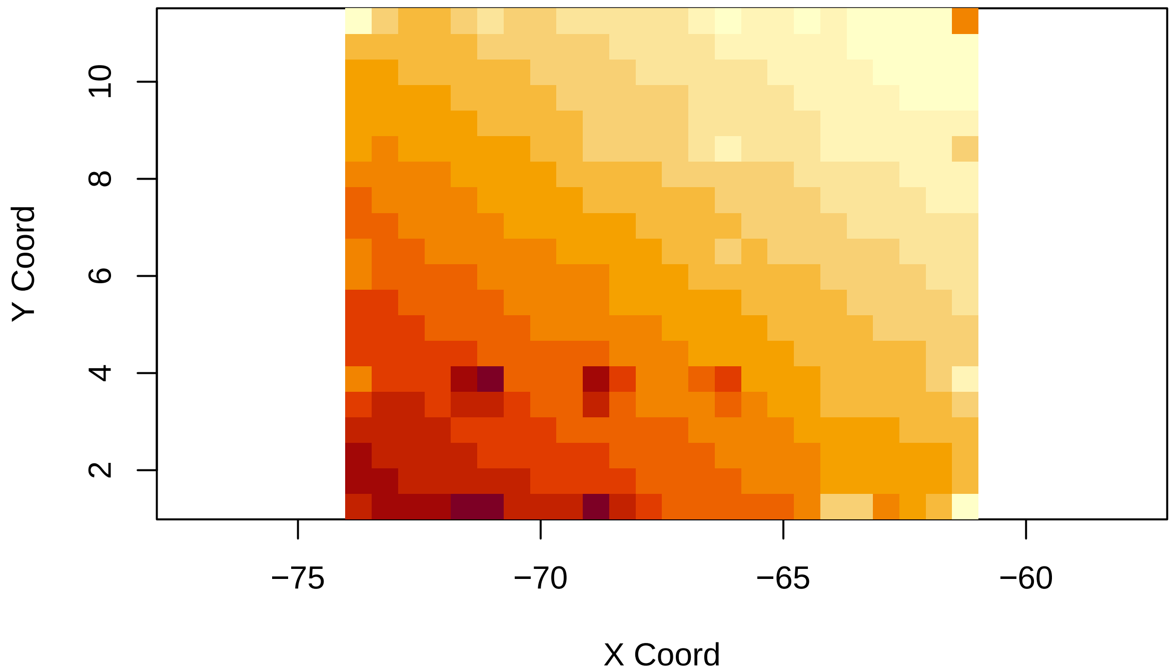

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



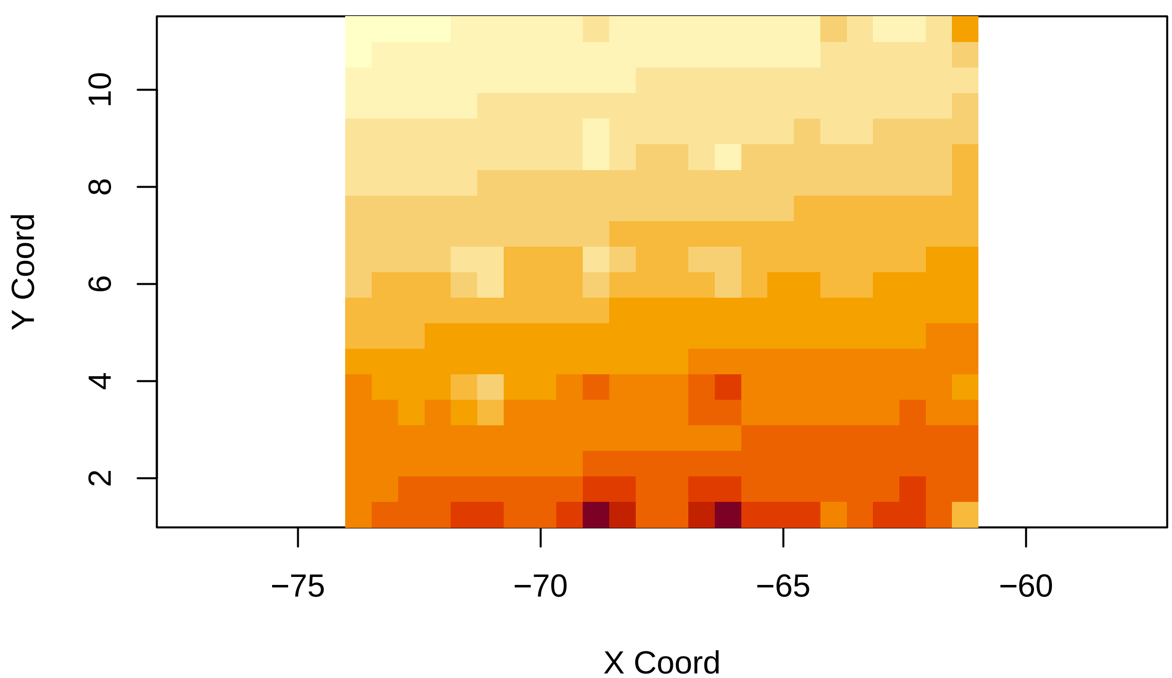

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



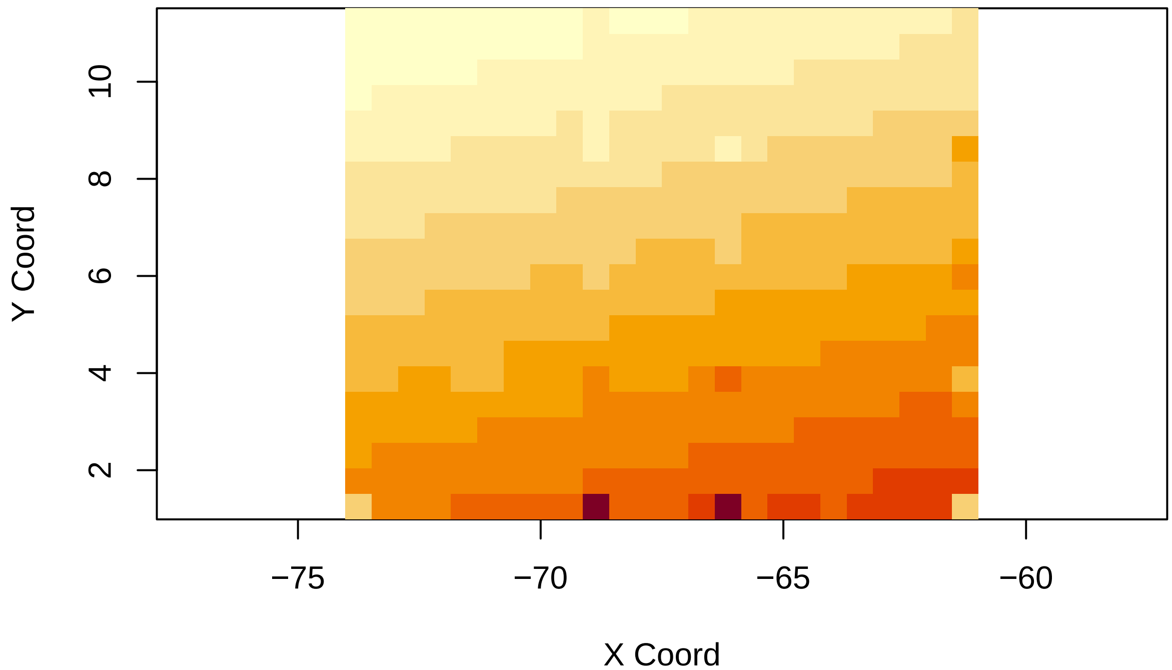

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



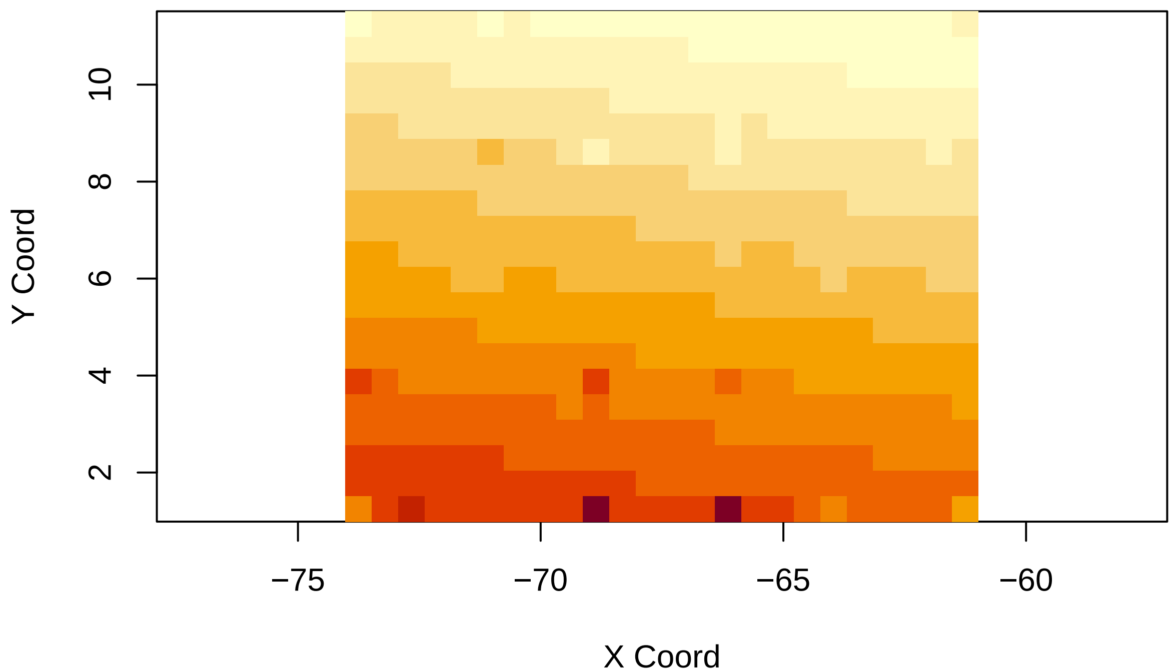

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



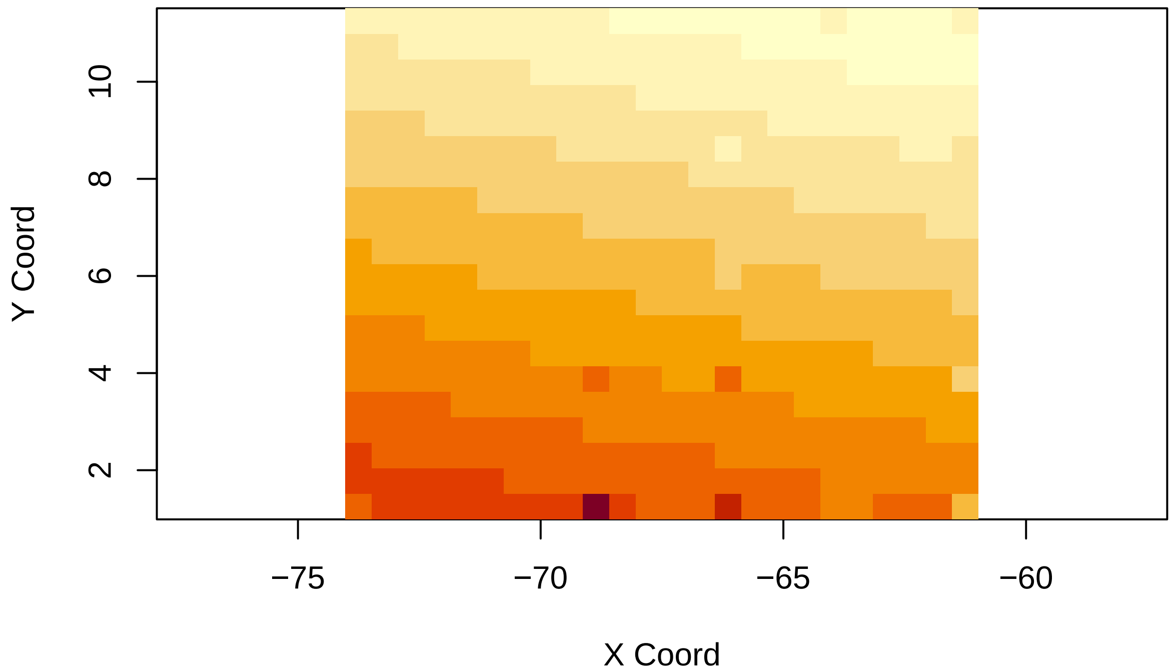

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



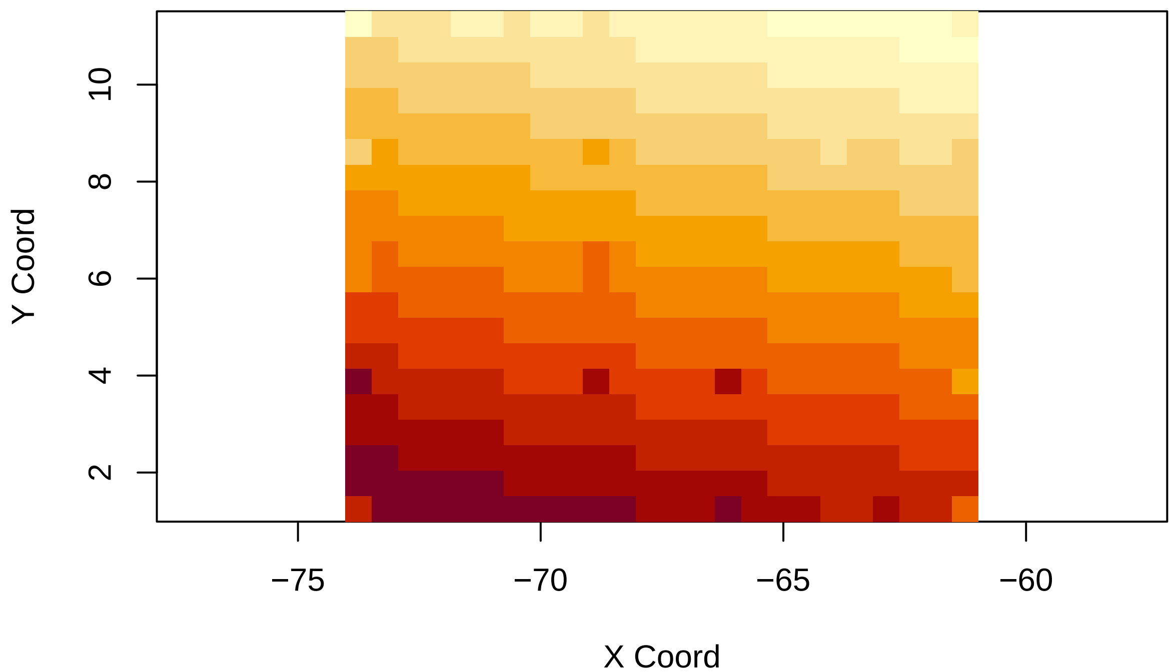

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



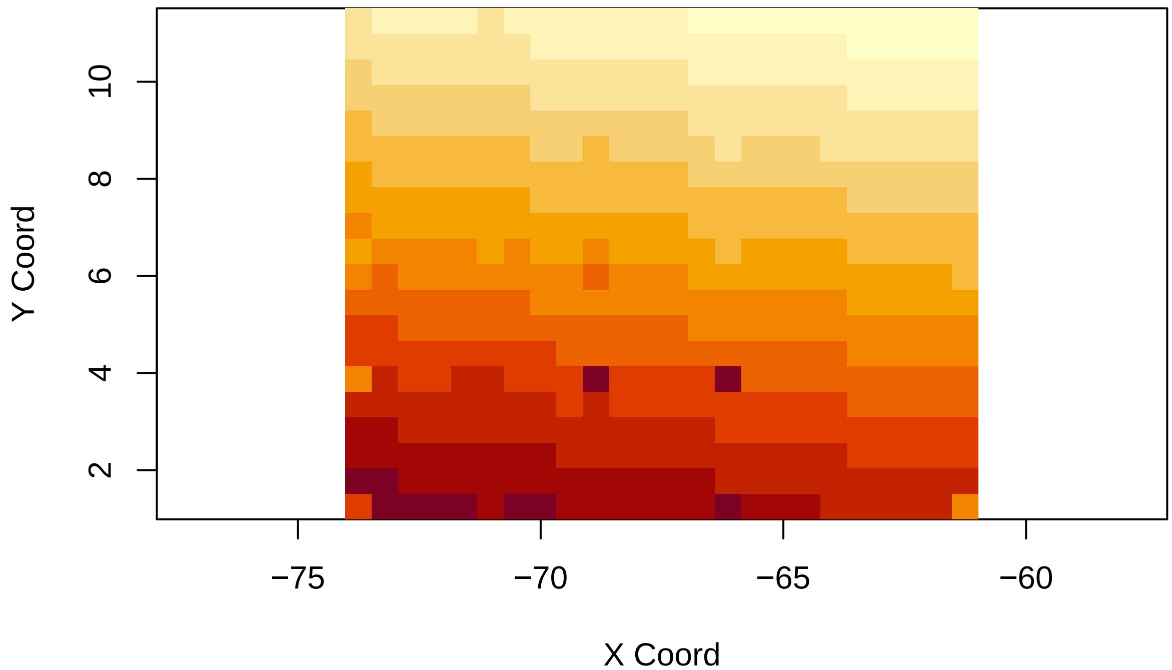

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



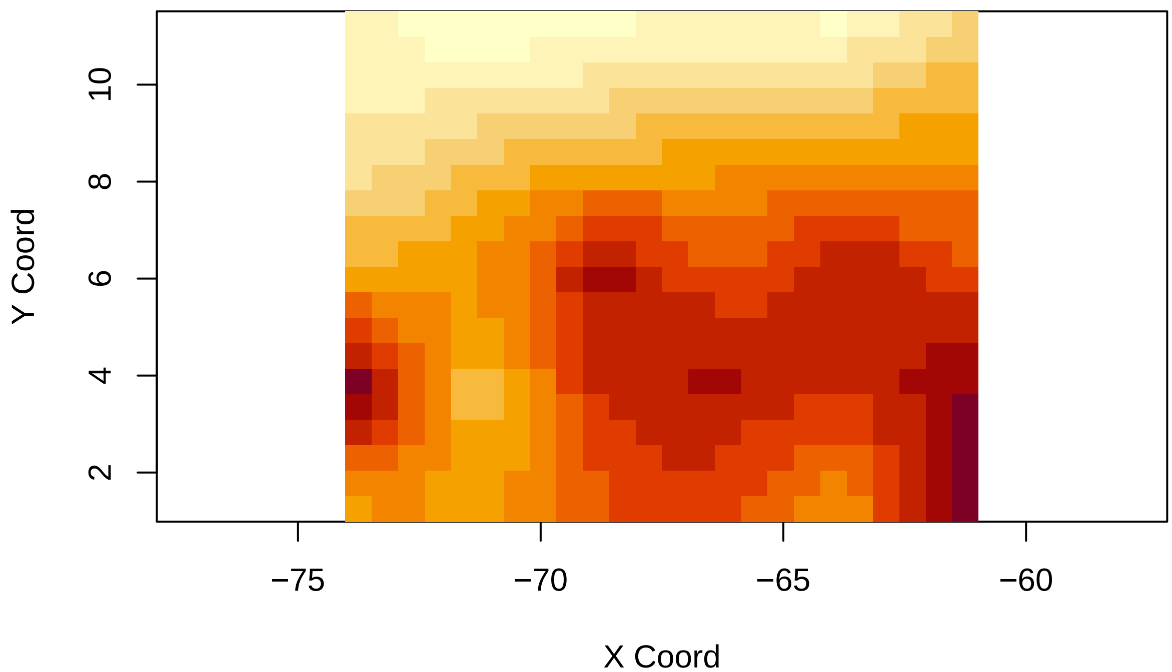

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



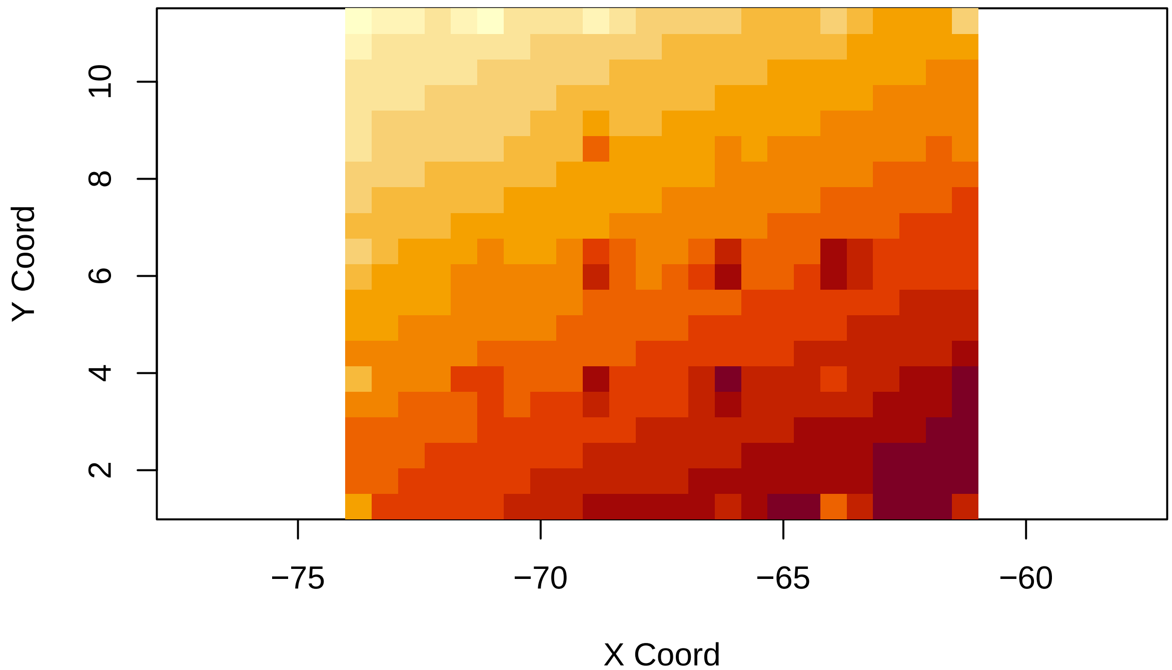

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



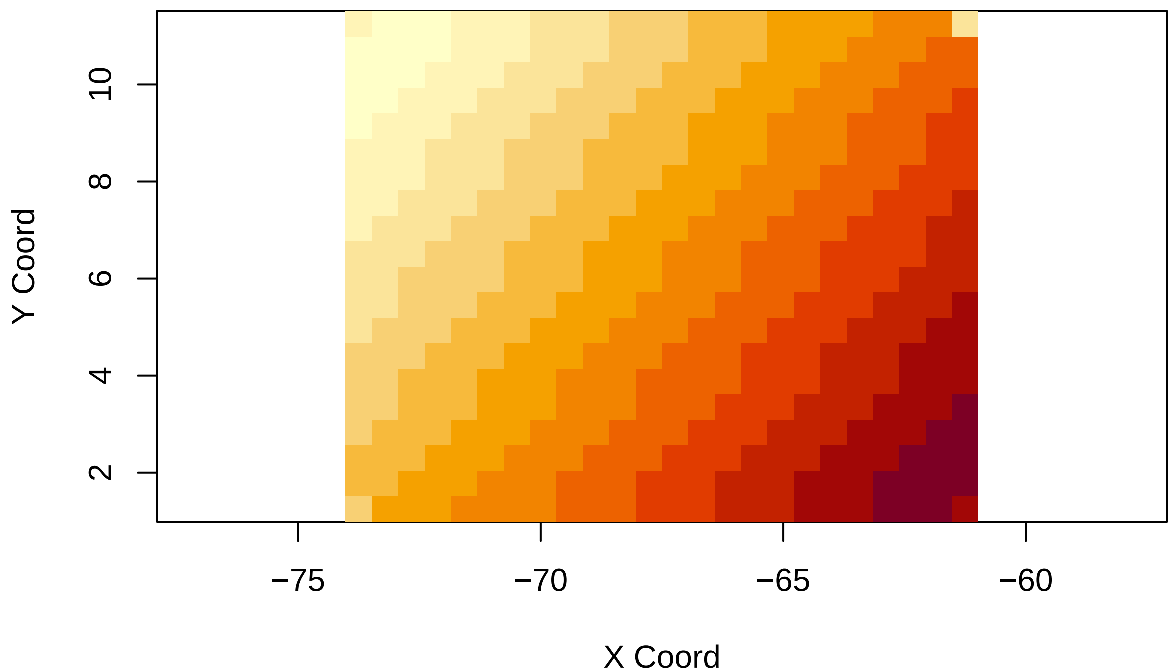

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



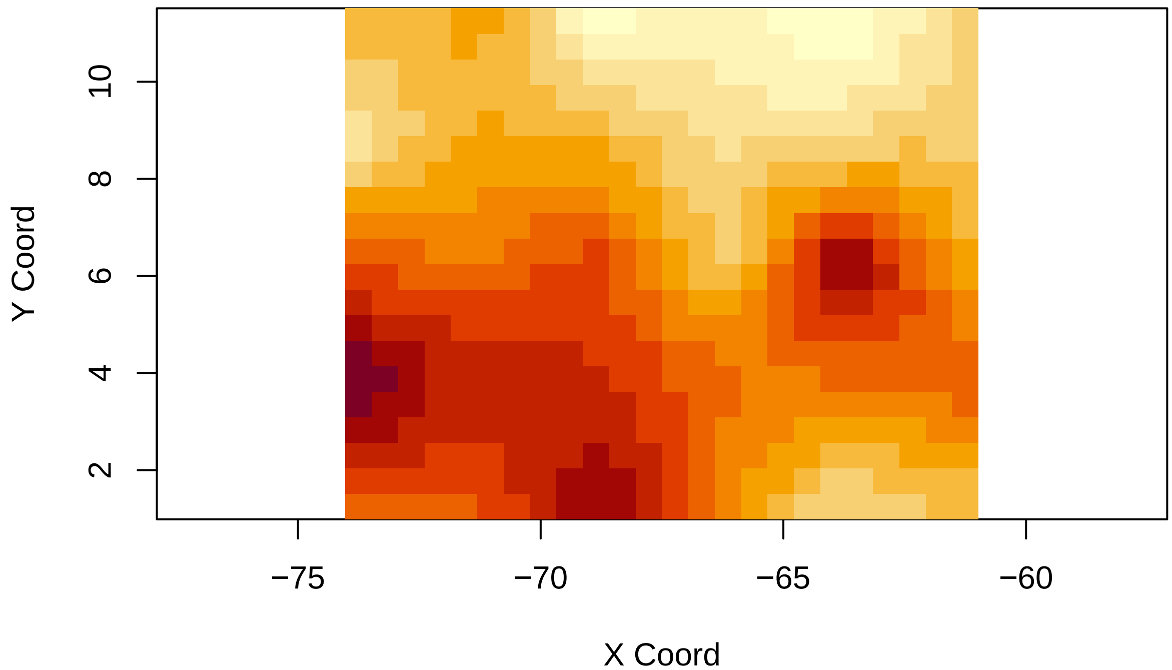

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



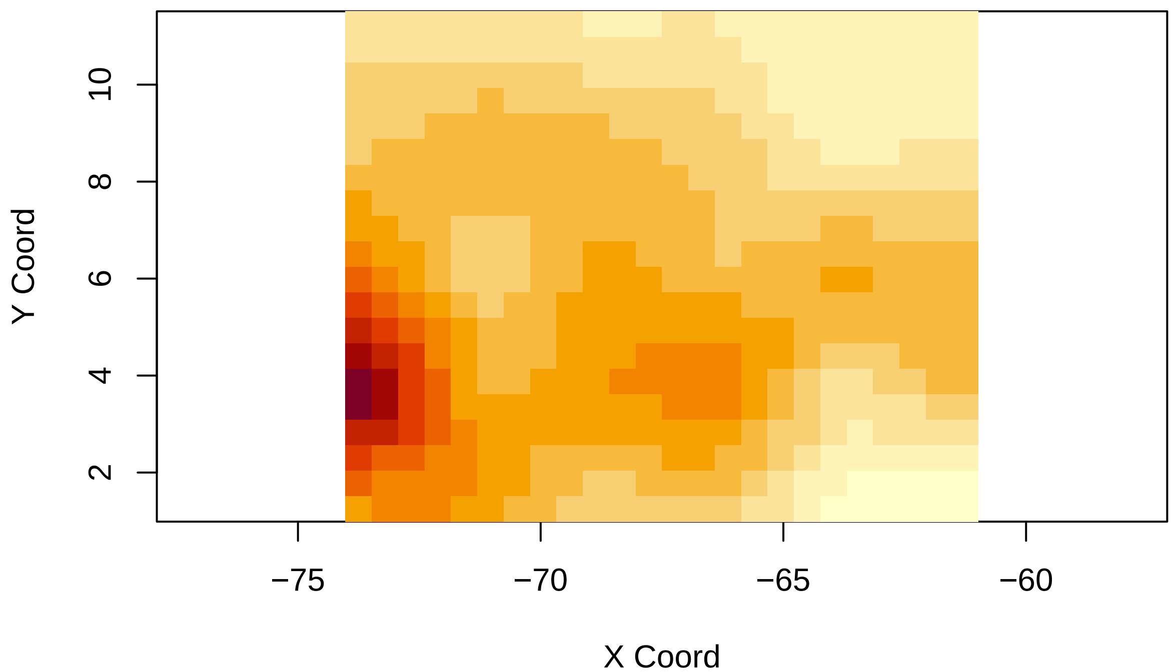

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```



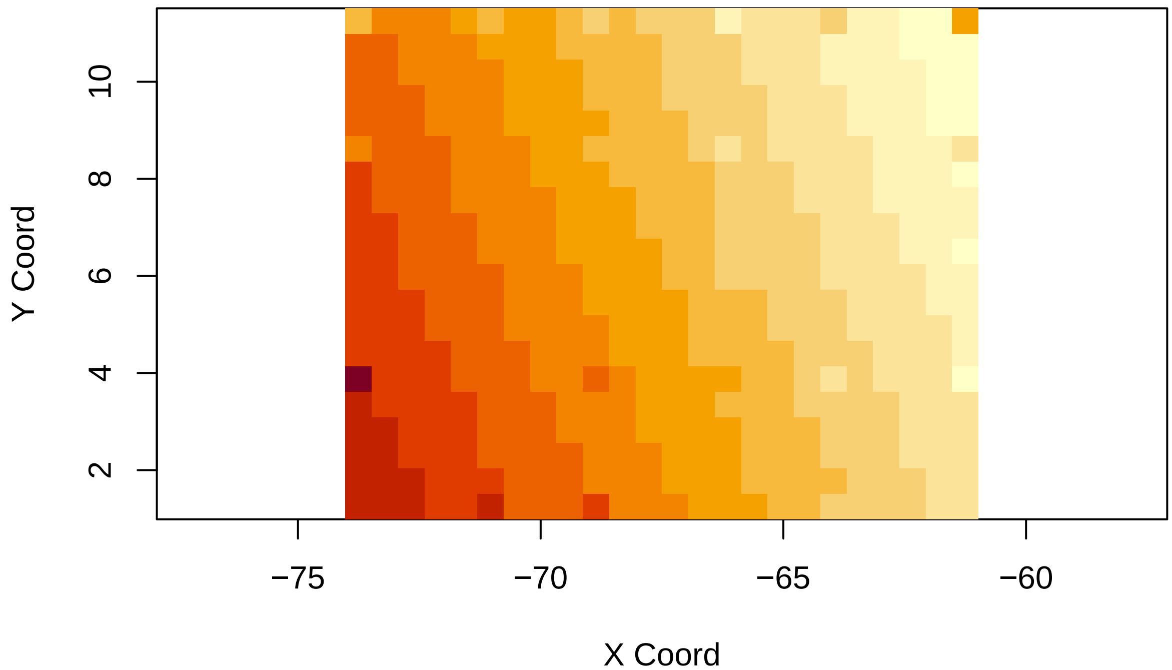

```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```




```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```




```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```




```
## krige.conv: model with mean given by a 1st order polynomial on the coordinates
## krige.conv: Kriging performed using global neighbourhood
```