공간데이터사이언스 개론

R의 활용

이상일

2025-08-08

목차

# 서장

데이터사이언스는 주어진 데이터를 바탕으로 질문에 대한 해답을 찾고, 그 과정을 서로 소통하는 학문이다. 데이터사이언스에서의 소통은 단순히 결과를 보여주는 것에 그치지 않고, 사용된 데이터를 공유하고, 답을 도출하는 과정을 포괄적이고 재현 가능한 방식으로 제시하는 것을 포함한다. 또한 데이터사이언스는 주어진 데이터가 질문에 답하기에 충분치 않을 수 있음을, 그리고 비록 답이 도출되었다 하더라도 데이터 수집이나 샘플링 방식에 따라 답이 달라질 수 있음을 인정한다.

이 책은 *공간적* 데이터의 기본 개념을 소개하고 설명한다. 포인트, 라인, 폴리곤, 래스터, 커버리지, 지오메트리 속성, 데이터 큐브, 참조계와 같은 기초 개념에서부터 속성이 지오메트리와 어떻게 연결되는지와 이것이 분석에 어떤 영향을 끼치는지 등과 관련된 보다 고차원적인 개념도 다룬다. 속성과 지오메트리의 관계를 서포트(support)라고 하는데, 서포트가 변경되면 속성의 특성도 변화한다. 어떤 데이터는 공간적 연속성에 기반해 생성되기 때문에 모든 지점에서 관찰될 수 있다. 그런데 또 다른 데이터는 공간적 이산성에 기반해 생성되기 때문에 특정한 구획 체계를 통해서만 관찰된다. 현대 공간데이터분석에서는 이러한 구획 체계 개념이 빈번하게 사용되며, 포인트 데이터, 지구통계학적 데이터, 에어리어 데이터를 가리지 않고 폭넓게 적용되고 있다. 공간적 재현의 중요성을 뒷받침하는 것이 바로 서포트(그리고 서포트에 대한 이해)이다. 이 책은 공간데이터를 자신의 분석에 활용하고자 하는 데이터 과학자를 대상으로 한다. 이 책을 통해 공간데이터분석이 어떻게 이루어지는 지를 보여주고자 하며, 예시를 위한 프로그래밍 언어로 R를 사용한다. 향후에 파이썬(Python)(예: Bivand 2022a)과 줄리아(Julia)를 사용한 예시도 추가할 예정이다.

보통 사람들이 공간데이터에 대해 흔히 가지는 생각이 있다. 공간데이터란 관측 개체의 경위도값을 속성으로 가지는 데이터 정도이고, 경위도값을 여타의 변수와 동일하게 취급하기만 하면 아무런 문제가 없다는 것이다. 이는 더 풍성한 연구 결과를 얻을 기회를 놓치게 하고, 잘못된 결론에 도달할 위험성을 높여줄 수 있다. 다음을 보자.

* 좌표 쌍은 말그대로 쌍이기 때문에, 반드시 함께 다루어져야 한다.
* 관측 개체는 포인트, 라인, 폴리곤, 그리드 셀 등으로 다양할 수 있으며, 각각의 특성이 반드시 고려되어야만 한다.
* 관측 개체 간의 공간적 거리는 단순히 직선 거리 뿐만 아니라, 상황에 따라 대원(great circle) 거리, 네트워크 거리, 또는 비용 거리 등으로 표현되어야 한다.

이 책을 통해 공간데이터, 좌표참조계, 공간분석과 관련된 다양한 개념 뿐만 아니라 **sf**(Pebesma 2018, 2022a), **stars**(Pebesma 2022b), **s2**(Dunnington, Pebesma, and Rubak 2023) and **lwgeom**(Pebesma 2023)와 같은 다양한 R 패키지도 다루어진다. 또한 공간적 **tidyverse**(Wickham et al. 2019; Wickham 2022) 확장 패키지 및 이들 패키지와 함께 사용할 수 있는 공간분석과 시각화 패키지들인 **gstat**(Pebesma 2004; Pebesma and Graeler 2022), **spdep**(Bivand 2022b), **spatialreg**(Bivand and Piras 2022), **spatstat**(Baddeley, Rubak, and Turner 2015; Baddeley, Turner, and Rubak 2022), **tmap**(Tennekes 2018, 2022), **mapview**(Appelhans et al. 2022)도 함께 다루어진다.

데이터사이언스와 마찬가지로, 공간데이터사이언스는 특정 과학 분야의 하위 분야로 발전한 것이 아니라 공간데이터의 적용과 관련된 수많은 학문 혹은 산업 영역로부터 상향식으로 발전해 온 분야이다. 학술 컨퍼런스, 심포지움, 학회, 연구 프로그램 등을 통해 공간데이터사이언스를 규정하려는 다양한 시도들이 이루어지고는 있지만, 공간데이터의 적용 영역이 너무나 광범위하고 다양하기 때문에 이러한 활동이 결실을 맺기는 쉽지 않아 보인다. 이 책에 “공간데이터사이언스”라는 제목을 붙인 것은 그것을 통해 이 분야의 경계를 획정하려는 의도 때문이 아니라, 지난 30~40년간 연구의 모든 것(기본 아이디어, 데이터, 소프트웨어 기반 연구 과정)을 기꺼이 공유한 수 많은 분야의 수 많은 연구자들과 함께 한 경험을 이 책을 통해 정리함으로써 이 연구 분야의 발전에 조금이나마 기여하고자 하는 의도 때문이다. 따라서 이 책에서 다루는 주제의 선택은 우리 자신의 연구 관심 분야와 경험에 의해 어느 정도 편향되어 있을 수 밖에 없다. 우리가 경험한 오픈 연구 커뮤니티를 만드는 데 여러 플랫폼이 도움을 주었다. ai-geostats 및 r-sig-geo 메일링 리스트, sourceforge, r-forge, GitHub, 그리고 2006년부터 매년 개최되는 OpenGeoHub 여름 학교가 그러한 플랫폼이다. 데이터사이언스의 언어 장벽을 넘고자 한 수 많은 노력의 결과 현재 우리는 새롭고도 가슴뛰는 퍼스펙티브가 열리고 있음을 목도하고 있다. 이 분야에 기여하고자 하는 것은 오픈사이언스가 더 나은 과학으로 이어지며, 더 나은 과학이 보다 지속가능한 세상에 기여할 수 있다고 믿기 때문이다.

## 감사의 글

우리는 r-spatial 커뮤니티 전체에 감사하며, 특히 다음에 열거한 분들께 특별한 감사의 마음을 전한다.

* r-spatial 패키지를 개발하거나 그 개발에 기여한 분들
* 트위터 #rspatial 또는 GitHub에서 디스커션에 참여한 분들
* 강좌, 여름 학교, 또는 컨퍼런스에서 의견을 개진하거나 질문을 해 준 분들

우리는 **s2** 패키지를 구현한 Dewey Dunnington, 적극적인 도움을 준 Sahil Bhandari, 6장의 그래프 제작을 도와준 Jonathan Bahlmann, 그리고 Claus Wilke와 Jakub Nowosad에게 특별한 감사의 마음을 전한다. 2021년과 2022년에 있었던 “R을 활용한 공간데이터사이언스(Spatial Data Science with R)” 강좌로부터 큰 도움을 받았다. 다음의 GitHub 리포지터리에 이슈, 풀 리퀘스트 또는 디스커션을 통해 적극적으로 기여한 분들께도 감사의 마음을 표한다.

* 이 책의 리포지터리(Nowosad, jonathom, JaFro96, singhkpratham, liuyadong, hurielreichel, PPaccioretti, Robinlovelace, Syverpet, jonas-hurst, angela-li, ALanguillaume, florisvdh, ismailsunni, andronaco)
* sf 리포지터리(aecoleman, agila5, andycraig, angela-li, ateucher, barryrowlingson, bbest, BenGraeler, bhaskarvk, Bisaloo, bkmgit, christophertull, chrisyeh96, cmcaine, cpsievert, daissi, dankelley, DavisVaughan, dbaston, dblodgett-usgs, dcooley, demorenoc, dpprdan, drkrynstrng, etiennebr, famuvie, fdetsch, florisvdh, gregleleu, hadley, hughjonesd, huizezhang-sherry, jeffreyhanson, jeroen, jlacko, joethorley, joheisig, JoshOBrien, jwolfson, kadyb, karldw, kendonB, khondula, KHwong12, krlmlr, lambdamoses, lbusett, lcgodoy, lionel-, loicdtx, marwahaha, MatthieuStigler, mdsumner, MichaelChirico, microly, mpadge, mtennekes, nikolai-b, noerw, Nowosad, oliverbeagley, Pakillo, paleolimbot, pat-s, PPaccioretti, prdm0, ranghetti, rCarto, renejuan, rhijmans, rhurlin, rnuske, Robinlovelace, robitalec, rubak, rundel, statnmap, thomasp85, tim-salabim, tyluRp, uribo, Valexandre, wibeasley, wittja01, yutannihilation, Zedseayou)
* stars 리포지터리(a-benini, ailich, ateucher, btupper, dblodgett-usgs, djnavarro, ErickChacon, ethanwhite, etiennebr, flahn, floriandeboissieu, gavg712, gdkrmr, jannes-m, jeroen, JoshOBrien, kadyb, kendonB, mdsumner, michaeldorman, mtennekes, Nowosad, pat-s, PPaccioretti, przell, qdread, Rekyt, rhijmans, rubak, rushgeo, statnmap, uribo, yutannihilation)
* s2 리포지토리(kylebutts, spiry34, jeroen, eddelbuettel)

# 1. 시작하기

이 장에서는 공간데이터와 시공간데이터를 다루기 위한 관련 개념들에 대해 소개한다. 이 개념들은 이후의 장들에서 좀 더 상세하게 다루어질 것이다. 이 외에도 모든 공간데이터사이언스 언어 실행의 토대가 되는 몇 가지 오픈소스 테크놀로지에 대해 소개한다.

## 1.1 첫 번째 지도

공간데이터를 표현하는 전형적인 방법은 지도를 그리는 것이다. 그림 1.1에는 단순한 지도가 나타나 있다.

|  |
| --- |
| 그림 1.1: 첫 번째 지도: 미국 노스캐롤라이나의 카운티별 출생아수, 1974~78년 |

이 지도에는 몇 가지 그래픽 요소들이 나타나 있다.

* 검은색 외곽선을 가진 폴리곤이 나타나 있고, 폴리곤의 내부는 BIR74라는 변수(제목)의 값에 따라 상이한 컬러로 채워져 있다.
* 범례는 서로 다른 컬러가 무엇을 의미하는지를 설명하고 있는데, 특정한 *컬러 팔레트*가 적용되어 있고, 컬러가 변하는 지점에 *컬러 단절값*이 나타나 있다.
* 경위선망(그래티큘)이 지도의 배경에 나타나 있다.
* 축의 눈금은 특정한 경도값과 위도값이다.

*폴리곤*은 특정한 형태의 *지오메트리*이다. 공간적 지오메트리(포인트, 라인, 폴리곤, 픽셀)에 대해서는 3장에서 자세히 다룬다. 폴리곤은 여러 개의 포인트들로 구성되어 있고, 포인트들은 선분으로 서로 이어져 있다. 공간데이터의 포인트 위치가 어떻게 표현되고 측정되는지에 대해서는 2장에서 다룬다. 그림 1.1에서 볼 수 있는 것처럼, 모든 경위선이 직선으로 표현되는 것이 아니다. 이것은 지도에 특정한 형태의 투영법이 적용되어 있음을 의미한다. 지도투영에 대해서는 2장과 8.1절에서 다룬다.

그림 1.1에서 컬러로 표현되어 있는 것은 BIR74라는 변수의 값이다. 값 하나는 지오메트리 혹은 피쳐(feature) 하나와 연결되어 있다. 5장은 이러한 피처 속성 및 그것들이 피처 지오메트리와 어떻게 관련되어 있는지에 대해서 다룰 것이다. BIR74 변수는 출생아수를 나타내며, 이는 *지역별* 빈도값이다. 지역별 빈도값이라는 말은 이 값이 지역 내의 모든 지점과 관련되어 있는 것이 아니라는 것을 의미하는데, 지도의 컬러가 연속적인 값을 취하고 있기 때문에 이렇게 오해할 수 있지만, 사실 빈도값은 폴리곤에 전체와 연결되어 있는 일종의 적분값을 의미한다.

그림 1.1을 그리기 위해서는 당연히 데이터가 필요한데, 7.1절에서 사용된 파일을 읽어 들였다. 세 개의 속성 변수에 대한 첫 세 개 레코드의 데이터 요약을 출력하면 다음과 같다.

# Simple feature collection with 100 features and 3 fields  
# Geometry type: MULTIPOLYGON  
# Dimension: XY  
# Bounding box: xmin: -84.3 ymin: 33.9 xmax: -75.5 ymax: 36.6  
# Geodetic CRS: NAD27  
# # A tibble: 100 × 4  
# AREA BIR74 SID74 geom  
# <dbl> <dbl> <dbl> <MULTIPOLYGON [°]>  
# 1 0.114 1091 1 (((-81.5 36.2, -81.5 36.3, -81.6 36.3, -81.6 36…  
# 2 0.061 487 0 (((-81.2 36.4, -81.2 36.4, -81.3 36.4, -81.3 36…  
# 3 0.143 3188 5 (((-80.5 36.2, -80.5 36.3, -80.5 36.3, -80.5 36…  
# # ℹ 97 more rows

이 데이터 요약은 다음의 사항을 알려준다:

* 데이터셋은 100개의 피처(레코드)와 3개의 필드(속성)으로 구성되어 있다.
* 지오메트리 유형은 MULTIPOLYGON (3장)이다.
* 디멘션(dimension)은 XY이다. 즉, 개별 포인트는 두 개의 좌표값으로 구성되어 있다.
* CRS(coordinate reference system, 좌표참조계)는 측지(geodetic) 좌표계이며, NAD27 데이텀(datum)에 의거한 경위도값을 가지고 있다(2장).
* 세 개의 속성 변수 바로 다음에 MULTIPOLYGON 유형의 geom 변수가 있는데, 이것은 폴리곤 정보를 도(degree) 형식으로 담고있다.

패싯(facet) 플롯을 활용하면 그림 1.2에서 보는 것과 같은 보다 복잡한 형태의 지도를 그릴 수 있다.

|  |
| --- |
| 그림 1.2: “미국 노스캐롤라이나 카운티별 영아돌연사증후군에 의한 사망아수의 패싯 지도, 1974~78년과 1979~84년” |

리플릿(leaflet)을 활용하면 [그림 1.3](#fig-1-3) 에서 보는 것과 같은 상호작용형 지도를 만들 수 있다.

|  |
| --- |
| 그림 1.3: **mapview**로 그린 상호작용형 지도: 팬과 줌을 이용해 지도 스케일에 변화를 줄 수 있고 카운티를 클릭하면 해당 카운티의 속성을 보여주는 팝업 윈도우가 뜬다. |

## 1.2 좌표참조계

그림 1.1의 배경에 그어진 회색선은 경위선망, 즉 *그래티큘*(graticule)이다. 경위선이 두 축과 직교하는 직선이 아니라는 점이 명백하다. 이것은 데이터에 특정한 *투영법*이 적용되어 있음을 의미한다. 그런데 그림 1.3에는 노스케롤라이나의 북쪽 경계가 곡선이 아닌 직선으로 표현되어 있는데, 이것은 또 다른 투영법이 적용되었음을 의미한다.

그림 1.1에 나타나 있는 경위도 좌표는 특정한 *데이텀*(여기서는 NAD27)과 연결되어 있다(역자주: 경위도 좌표는 절대적인 것이 아니라 데이텀에 따라 달라지는 상대적인 것이라는 점을 반드시 이해해야 한다. 동일한 지점이 데이텀에 따라 다른 경위도 좌표값을 부여받고, 동일한 경위도 좌표값이 데이텀에 따라 지표 상의 다른 지점을 의미한다). 데이텀은 지구를 모델화하기 위해 어떤 지구타원체가 선택되고, 그 지구타원체를 지구와 어떠한 방식으로 일치시킬 것인가(지구타원체의 원점을 지구의 어떤 지점과 어떠한 방향으로 일치시킬 것인가)와 관련된 몇 가지 사항들을 규정한다. GPS 수신기(예: 모바일 폰)를 통해 획득된 좌표값은 WGS84(World Geodetic System 1984) 데이텀에 의거한 것인데, 만일 NAD27(North American Datum 1927)에 의거한 것이라면 동일한 좌표값에 대해 대략 30m 정도의 편차를 나타낼 것이다.

투영법은 두 개의 좌표값을 연결하는 함수이다.

* **타원체 좌표**(ellipsoidal coordinates): 지구에 대한 수학적 모델(지구타원체 혹은 지구구체) 상의 3차원 좌표로, 경도와 위도로 나타낸 것이다.
* **투영 좌표**(projected coordinates): 지도 상의 2차원 평면 좌표로, x 좌표와 y 좌표 혹은 동거(easting)와 북거(northing)로 나타낸 것이다.

한 데이텀을 다른 데이텀으로 바꾸는 것을 데이텀 변환이라고 한다. 투영과 좌표계는 **공간참조계**(spatial reference system)의 설정과 관련되어 있고, 2장에서 상세히 다룬다.

## 1.3 래스터 데이터와 벡터 데이터

포인트, 라인, 폴리곤 지오메트리는 *벡터*(vector) 데이터의 예시이다. 벡터 지오메트리를 구성하는 좌표값은 지표 상의 “정확한” 위치를 의미한다. 이에 반해 래스터 데이터는 주로 정사각형 픽셀로 구성된 격자망(이것을 *래스터*라고 부름)에 속성값이 부여되어 있는 데이터이다. 래스터 데이터의 예가 그림 1.4에 나타나 있다.

|  |
| --- |
| 그림 1.4: “브라질의 대서양 연안 도시 올린다에 대한 래스터 지도: (a) Landsat-7의 블루 탐지대를 타나낸 것으로 서로 다른 컬러는 속성값의 차이를 나타냄. (b) 좌상의 10X10 픽셀만 확대하여 나타낸 것임. (c) 3개의 샘플 포인트로 구성된 벡터 데이터를 중첩하여 나타냄. (d) 샘플 포인트로부터 반경 500m를 나타낸 3개의 폴리곤으로 구성된 벡터 데이터를 중첩하여 나타냄.” |

벡터 데이터와 래스터 데이터를 여러 가지 방식으로 결합될 수 있다. 예를 들어, 그림 1.4(c)에 나타나 있는 세 포인트에 해당하는 래스터 값만을 추출할 수도 있고, 그림 1.4(d)에 나타나 있는 원과 결부된 모든 래스터 값을 추출할 수도 있다.

래스터-투-벡터 전환은 7.6절에서 다루어지는데, 다음과 같은 내용이 포함된다.

* 래스터 픽셀값을 포인트 속성값으로 전환하기
* 래스터 픽셀값을 폴리곤 속성값으로 전환한 후, 동일한 속성값을 가진 폴리곤을 머지하기(“폴리곤 생성”)
* 특정한 범위의 값을 가진 연속적인 픽셀 연속체를 나타내는 라인이나 폴리곤을 생성하기(“등치선 생성”)

|  |
| --- |
| 그림 1.5: 그림 1.1에 나타나 있는 카운티별 출생아수(1974~78)를 래스터화하여 나타낸 지도 |

벡터-투-래스터 전환은 그림 1.5에 나타나 있는 것(폴리곤의 래스터화)처럼 매우 간단한 것일 수 있다. 그러나 다른 형태의 벡터-투-래스터 전환은 통계적 모델링을 수반할 수 있다.

* 포인트 속성값을 인터폴레이션을 통해 그리드 셀에 할당하기 (12장)
* 포인트의 밀도 분포를 추정하여 그리드 셀에 할당하기 (11장)
* 폴리곤의 속성값을 에어리어-가중 인터폴레이션을 통해 그리드 셀에 할당하기 (5.3절)
* 포인트, 라인, 폴리곤을 래스터로 직접 변환하기 (7.6절)

## 1.4 래스터 유형

래스터 디멘션은 행과 열이 공간 좌표계와 어떻게 관련되어 있는가에 의해 결정된다. 그림 1.6은 다양한 가능성을 예시로 보여주고 있다.

|  |
| --- |
| 그림 1.6: 다양한 래스터 지오메트리 유형 |

그림 1.6에 나타나 있는 레귤러(regular) 래스터는 일정한 모양(반드시 정사각형일 필요는 없다)의 그리드 셀로 이루어져 있고, 가로추과 세로축이 x-축(동거축)과 y-축(북거축)과 일치한다. 그러나 다른 래스터 유형도 존재할 수 있는데, 가로축과 세로축이 더 이상 x-축 및 y-축과 일치하지 않거나(*rotated*), 가로축과 세로축이 직교하지 않거나(sheared), 셀 크기가 동일 축 상에서 조차 다르게 나타날 수 있다(rectilinear). 심지어 curvilinear 래스터의 경우는 셀의 크기 및 방향 속성이 더 이상 또 다른 래스터 디멘션으로부터 독립적이지 않다(?).

특정한 좌표참조계에 의거해 regular한 지오메트리 유형을 가진 래스터가 있다고 하자. 이것을 셀 구조는 그대로둔 상태로 다른 투영법으로 전환하게 되면, rectilinear 하게 될 수도 있고(예를 들어, 그림 1.3에서처럼, 측지좌표를 메르카토르 도법으로 전환하는 경우), curvilinear 해 질 수도 있다(예를 들어, 그림 1.1에서처럼, 측지좌표를 람베르트 정형원추 도법으로 전환하는 경우). 이 전환을 역으로 수행하게 되면 정확히 동일한 래스터를 회복할 수 있다.

새로운 투영법이 적용된 새로운 레귤러 그리드를 생성하는 것을 래스터(혹은 이미지) *재투영*(reprojection) 혹은 *워핑*(warping)이라고 한다(7.8절). 워핑은 정보 손실이 발생하고, 불가역적이며, 다양한 옵션의 설정이 필수적이다. 예를 들어, 새로운 셀 값을 생성하기 위해 인터폴레이션을 적용할지, 평균값을 산출할지, 합산값을 산출할지를 결정해야 하며, 이웃값을 활용한 재표집(resampling)이 적용되어야 할지의 여부를 결정해야 한다. 이러한 선택은 래스터 셀 값이 범주형인지 연속형인지에 따라 달라질 수 있다(1.6절).

## 1.5 시계열, 어레이, 데이터 큐브

많은 공간데이터는 *단지* 공간적이기만 한 것이 아니라 시간적이기도 하다. 모든 관측치는 그것이 관측된 특정한 지점과 결부되어 있을 뿐만 아니라 관측이 이루어진 특정한 시간과도 결부되어 있다. 노스캐롤라이나 카운티 데이터셋은 그림 1.2에서 보는 것처럼 두 기간의 관측값을 가지고 있다. 원 데이터셋에서는 두 기간이 두 개의 변수로 저장되어 있었겠지만, 그림 1.2처럼 두 개의 패싯 지도로 표현되기 위해서는 지오메트리의 반복을 통해 두 변수가 하나의 컬럼에 길게 이어져 있는 형태로 변형되어야 한다. 이러한 형태를 위컴(Wickham)(2014)은 타이디(tidy) 형태라고 부른바 있다. 그런데 지오메트리와 연결된 긴 시계열(time series) 데이터를 가지고 있을 때, 두 개의 옵션(시간을 여러 개의 컬럼에 할당하거나 지오메트리의 반복을 통해 시간을 하나의 컬럼에 길게 배열하는 것) 중 어는 것도 적절해 보이지는 않는다. 이런 경우 보다 효과적인 방법은 메트릭스나 어레이(array) 형식을 취해, 하나의 차원에는 시간을 할당하고 나머지 차원에는 공간을 할당하는 것이다. 이미지나 래스터 데이터는 이미 이러한 방식대로 매트릭스 형태로 저장되어 있고, 여기에 시간이 첨가되면 3차원의 어레이 형태가 된다. 그러한 데이터를 부를는 일반 용어가 바로 (시공간적) **데이터 큐브**이다. 큐브는 차원의 개수에 구애받지 않는 어레이를 의미한다. 데이터 큐브는 벡터 데이터와 래스터 데이터 모두를 지칭하는데 사용될 수 있고, 예시들이 6장에서 제시될 것이다.

## 1.6 서포트

단순한 포인트 지오메트리가 아니라 포인트 집합체의 지오메트리(다중-포인트, 라인, 폴리곤, 픽셀)를 가지 공간데이터의 경우 결부되어 있는 속성값은 지오메트리와 몇 가지 상이한 방식의 관련성을 갖는다.

* 지오메트리의 모든 포인트에 공통적으로 적용되는 **동일값**
* 지오메트리의 모든 포인트를 집합적으로 대표하는 **합산값**
* 개별 지오메트리의 고유성을 표현하는 **식별값**

동일값의 예로 폴리곤의 토지이용 속성이나 기반암 유형 속성을 들 수 있고, 합산값의 예로 카운티의 출생아수를 들 수 있고, 식별값으로 카운티 이름을 들 수 있다(역자주: 폴리곤의 토지이용은 폴리곤 내의 모든 지점에 공통적으로 적용될 수 있고, 카운티의 출생아수는 카운티 내 모든 지점의 출생아수를 합산했다는 의미이므로 카운티 내의 모든 지점에 그 값이 적용될 수는 없고, 카운티 전체를 집합적으로 대표한다).

한 속성값과 결부되어 있는 공간적 개체를 속성값의 **서포트**라고 한다. 합산값은 “블록” (폴리곤 혹은 라인) 서포트를 갖고 동일값은 “포인트” 서포트를 갖는다 (동일값은 모든 포인트에 적용된다). 예를 들어, 그림 1.5는 폴리곤 서포트를 갖는 변수(카운티별 출생아수)로부터 도출된 것으로, 카운티별 속성값을 카운티를 구성하는 픽셀의 속성값으로 할당한 것이다. 이러한 래스터화를 통해 생성된 지도는 무의미한다. 즉, 속성값인 카운티별 “총출생아수”는 래스터 셀과 무관하며, 속성값과 결부되어 있는 카운티의 경계는 표시 조차되어 있지 않다. 이 지도로부터 노스캐롤라이나 주 전체의 출생아수나 출생아 밀도를 재계산하는 것은 불가능하다. 이처럼 서포트를 무시하는 것은 무의미한 지도의 산출로 귀결된다. 5장에서 좀 더 심도있게 다룰 것이다.

래스터 셀 속성은 포인트 서포트를 가질 수도 있고 블록 스포트를 가질 수도 있다. 포인트 서포트의 예로 고도를 들 수 있는데, 디지털고도모델(digital elevation model)의 경우 보통 셀 중심점의 고도값을 셀의 속성값으로 저장한다. 블록 서포트 (혹은 셀 서포트)의 예로 위성영상을 들 수 있는데, 이미지 픽셀의 속성값은 주로 픽셀(혹은 픽셀을 중심으로 한 특정 영역) 내부의 값들의 평균값이다. 대부분의 파일 포맷은 이러한 정보를 제공하지 않는다. 그러나 래스터 데이터를 애그리게이팅(aggregating), 리그리딩(regridding), 워핑하거나(7.8절), 포인트별 값을 추출할 때 매우 중요한 사안이다.

## 1.7 공간데이터사이언스를 위한 소프트웨어

이 책의 기본 프로그래밍 언어는 R이고, 공간데이터사이언스를 위한 다양한 R 패키지를 활용한다. 그런데 우리가 사용할 R 패키지 중 많은 것들이 R만을 위해서 개발되지는 않은 다양한 소프트웨어 라이브러리를 활용하고 있다. 하나의 예로서, 그림 1.7은 sf 패키지의 의존계(dependency), 즉 sf 패키지가 어떤 R 패키지나 어떤 시스템 라이브러리에 의존하고 있는지를 보여주고 있다.

|  |
| --- |
| 그림 1.7: sf의 의존계: 직선은 강한 의존성을 점선은 약한 의존성을 나타낸다. |

C 혹은 C++ 라이브러리(GDAL, GEOS, PROJ, liblwgeom, s2geometry, NetCDF, udunits2)는 모두 R 커뮤니티와는 직접 관련성이 없는 (공간) 데이터사이언스 커뮤니티에 의해 개발, 유지 및 사용되고 있다. 이러한 라이브러리를 활용함으로써 R 사용자들은 이러한 다른 커뮤니티와 무엇을 공유하고 협업하고 있는지를 이해하게 된다. R, 파이썬, 줄리아는 상호작용형 인터페이스를 제공하고 있기 때문에 동일한 라이브러리를 활용하는 다른 소프트웨어의 사용자들과는 달리 이러한 라이브러리에 보다 가깝게 접근할 수 있다. 이 책의 첫 번째 파트에서는 이러한 라이브러리에 어떤한 개념이 녹아들어 있는지를 설명하고자 하는데, 공간데이터사이언스 일반을 이해하는데 도움이 될 것이다.

### 1.7.1 GDAL

GDAL(Geospatial Data Abstraction Library)는 공간데이터에 대한 스위스 아미 나이프(Swiss army knife)정도로 생각할 수 있다. GDAL는 R, 파이썬, PostGIS 외에 100개가 넘는 다른 소프트웨어 프로젝트에서 사용되고 있다.

GDAL은 공간데이터를 읽고 쓸수 있게 해주는 라이브러리 중의 라이브러리로, 수많은 다른 라이브러리를 필요로 한다. 대략 100개가 넘는 라이브러리와 연결되어 있는데, 개별 라이브러리는 특정한 데이터 파일 포맷, 특정한 데이터베이스, 특정한 웹서비스, 특정한 압축 코덱을 다룰 수 있다.

CRAN(역자주: CRAN은 The Comprehensive R Archive Network의 약자로서, R의 수많은 패키지의 저장소로 이해하면 된다. R 언어 자체의 과거와 현재의 버전들 뿐만 아니라 현재 대략 20,000개 이상의 R 패키지가 모여 있다. 1997년 Kurt Hornik와 Friedrich Leisch에 의해 처음 만들어졌으며 현재에도 Hornik와 많은 자원봉사자들에 의해 운영되고 있다)에서 배포되는 바이너리 형식의 R 패키지에는 스태틱 링크 코드(statically linked code)만 포함되어 있는데, 이는 CRAN이 패키지의 호스트 시스템에 서드파티(third-party) 라이브러리가 존재하는지 그렇지 않은지에 대한 가정을 하지 않으려 하기 때문이다. 그 결과 CRAN으로부터 바이너리 형식의 sf 패키지를 인스톨하면, sf 패키지의 의존계 뿐만 아니라 모든 외부 라이브러리도 함께 다운로드되기 때문에 용량이 100 Mb에 달한다.

### 1.7.2 PROJ

PROJ(혹은 PRJ)는 지도투영과 데이텀변환을 위한 라이브러리이다. 공간 좌표를 하나의 CRS로부터 다른 CRS로 바꾸어 준다. PROJ 속에는 현재까지 알려져 있는 수많은 투영법에 대한 데이터베이스가 포함되어 있으며, 데이터 그리드(데이텀변환을 위한 고정밀 계수값)에 접근할 수 있게 해준다. CRS의 국제 표준에 맞추어져 있다(Lott 2015). 2장에서 좌표계와 PROJ를 자세히 다룬다.

### 1.7.3 GOES와 s2geometry

GOES(Geometry Engine Open Source)와 s2geometry는 지오메트리 오퍼레이션을 위한 라이브러리이다. 이 라이브러리를 활용해 기하학적 측정(길이, 면적, 거리), 논리적 판단(두 지오메트리가 포인트를 공유하고 있는지의 여부), 새로운 지오메트리의 생성(두 지오메트리가 공유하고 있는 포인트) 등의 오퍼레이션을 수행할 수 있다. GEOS는 이러한 오퍼레이션을 2차원 평면()에서 수행할 수 있게 해주고, s2geometry는 3차원 구면()에서 수행할 수 있게 해준다. 2장에서 CRS를 다루며, 4장에서는 2차원 공간과 3차원 공간을 다루는 데 있어 차이점에 대해 더 자세히 논의한다.

### 1.7.4 NetCDF, udunits2, liblwgeom

NetCDF(UCAR 2020)은 파일 형식이자 NetCDF 파일을 읽고 쓰기 위한 C 라이브러리를 의미한다. NetCDF를 통해 모든 차원의 어레이를 정의할 수 있으며, 특히 (기후) 모델링 커뮤니티에서 공간 및 시공간 정보를 다루는데 널리 사용된다. Udunits2(UCAR 2014; Pebesma, Mailud, and Hiebert 2016; Pebesma et al. 2022)는 측정 단위와 관련된 데이터베이스이자 소프트웨어 라이브러리이다. Udunits2를 통해 측정 단위간 전환 및 파생 단위의 처리를 할 수 있고, 사용자-정의 단위에 대한 지원을 받을 수 있다. liblwgeom “라이브러리”는 POSTGIS(Obe and Hsu 2015)의 소프트웨어 요소로서 GDAL이나 GEOS에서는 다루지 않는 몇 가지 루틴을 포함하고 있다. 예를 들어 PROJ를 장착하고 있는 GeographicLib 루틴에 쉽게 접근할 수 있게 해준다.

## 1.8 연습문제

1. 래스터 데이터와 백터 데이터의 차이점 다섯 가지를 나열하라.
2. 그림 1.1 아래에 나열되어 있는 것 외에, 지도의 그래픽 요소 다섯 개를 더 나열하라.
3. 그림 1.5에 나타나 있는 수치 정보가 왜 오해를 불러일으키는지(혹은 무의미한지)에 대해 얘기해 보라.
4. 지오메트리 오퍼레이션을 에서 행하는 것과 에서 행하는 것의 차이가 가장 극명하게 드러나는 상황을 예를 들어 설명해 보라.

# 2. 좌표계

“데이터는 단순한 숫자가 아니라 맥락을 가진 숫자이다.”: “데이터 분석에서 맥락은 의미를 부여한다.”(Cobb and Moore 1997)

포인트, 라인 , 폴리곤, 커버리지, 그리드와 같은 지오메트리를 이해하기 전에, 좌표계에 대해 우선적으로 검토하는 것이 유용하다. 이를 통해 좌표값이 정확히 무엇을 나타내는지에 대한 개념을 갖출 수 있다. 공간데이터에서 관측개체의 위치는 좌표값으로 주어지고, 좌표값은 좌표계에 의해 규정된다. 이를 위해 다양한 좌표계가 사용될 수 있으며, 가장 중요한 차이점은 2차원 혹은 3차원 공간 상의 좌표값이 직교하는 두 축에 의거해 결정되는지(데카르트 좌표계) 혹은 거리와 방향에 의거해 결정되는지(극 좌표계, 구체 혹은 타원체 좌표계)이다. 모든 관측치는 관측의 위치 뿐만 아니라 관측의 시간과도 결부되어 있기 때문에 시간좌표계 역시 간단히 언급될 것이다. 단위와 데이텀이 무엇인지를 배우기에 앞서 퀀티티(quantity)에 대해 간략히 검토하고자 한다.

## 2.1 퀀티티, 단위, 데이텀

VIM(International Vocabulary of Metrology, BIPM et al., 2012)는 콴티티에 대해 “현상, 물체 또는 물질의 속성으로, 이 속성은 숫자로 표현할 수 있는 크기와 준거를 가진다.” 여기서 “준거는 측정 단위, 측정 절차, 기준 물질 또는 이러한 것들의 조합일 수 있다.”라고 기술한다. 모든 데이터가 퀀티티로 구성되어 있는지에 대해 논의할 수 있지만, 적절한 데이터 처리를 위해서는 숫자(또는 기호)와 자체 뿐만 아니라 숫자가 무엇을 의미하는지, 특히 숫자가 어떤 준거에 기반하고 있는지에 대한 정보가 함께 제공되어야 한다는 점에 대해서는 논의할 필요가 없다.

측정 시스템은 기본 퀀티티에 대한 기본 단위와 파생 퀀티티에 대한 파생 단위로 구성된다. 예를 들어, SI 단위계(Bureau International des Poids et Mesures 2006)는 일곱개의 기본 단위로 구성된다: 길이(미터, m), 질량(킬로그램, kg), 시간(초, s), 전류(암페어, A), 열역학적 온도(켈빈, K), 물질의 양(몰, mol), 그리고 광도(칸델라, cd). 파생 단위는 기본 단위의 정수 거듭제곱의 곱으로 구성되며, 예를 들어 속도()나 밀도(), 면적( ) 등이 있다. 특별한 경우에는 무단위 측정이 있을 수 있는데, 단위가 서로 소거되는 경우(예: 질량 분율: kg/kg 또는 라디안으로 측정된 각: m/m)나 물체나 사건의 빈도를 세는 경우(예: “5개의 사과”)이다. 각도와 사과의 개수를 더하는 것은 의미가 없지만, 5개의 사과와 3개의 오렌지를 더하는 것은 의미가 있을 수 있으며, 이 경우 결과를 상위 클래스(즉, *과일*의 개수)로 재해석할 수 있다. Hand(2004)는 측정 단위의 맥락에서 다양한 측정 스케일에 대해 논의한 바 있는데, 예를 들어 사회과학에서 지능과 같은 변수를 측정하는데 사용되는 측정 스케일과 같은 것이다.

많은 퀀티티의 경우 값의 자연적인 원점은 0이다. 이러한 자연적 원점의 개념은 양(amount)이라는 퀀티티에 잘 들어맞는다. 두 양의 차는 의미 있는 음의 값일 수 있다. 위치와 시간의 차도 이러한 자연적 원점 개념으로 해석될 수 있다. 즉, 위치의 차는 거리이고, 시간의 차는 지속기간이다. 절대적인 위치와 시간은 고정된 원점이 필요하며, 이를 기준으로 다른 절대적인 시공간적 지점을 의미 있게 측정할 수 있다. 이러한 기준을 우리는 데이텀이라고 부른다. 공간의 경우, 데이텀은 하나 이상의 차원을 포함한다. 데이텀과 측정 단위(스케일)의 조합이 하나의 참조계를 구성한다.

이제 공간적 위치를 타원체 좌표 또는 데카르트 좌표로 표현하는 방법에 대해 자세하게 설명하고자 한다. 다음 섹션에서는 시간 및 공간 참조계과 R에서 이러한 참조계를 처리하는 방법에 대해 다룰 것이다.

## 2.2 타원좌표계

|  |
| --- |
| 그림 2.1: 2차원 극 좌표계와 데카르트 좌표계 |

그림 2.1은 2차원 극 좌표계와 데카르트 좌표계를 보여주고 있다. 해당 지점의 데카르트 좌표는 으로 주어지고, 극 좌표는 으로 주어지는데 는 대략 라디안 혹은 이다. 여기서 , , 은 모두 길이 단위이고 는 각도 단위(무단위 길이/길이 비)라는 점에 유의할 필요가 있다. 데카르트 좌표와 극 좌표 간의 변환은 매우 간단하다.

여기서 이 대신 사용되었는데, 오른쪽 일사분면에 위치가 있기 때문이다.

### 2.2.1 구체 혹은 타원체 좌표계

3차원의 경우, 데카르트 좌표계는 로 주어지고, 극 좌표계는 로 주어진다.

* 은 구체의 반지름이다.
* 는 경도로, 평면에서 양의 축으로부터 반시계방향으로 측정된다.
* 는 위도로, 평면과 해당 벡터가 이루는 각도이다.

그림 2.2는 데카르트 지심 좌표계와 타원체 좌표계를 보여준다.

|  |
| --- |
| 그림 2.2: 세 개의 거리로 표현되는 데카르트 지심 좌표계(왼편)와 두 개의 각도와 하나의 타원체고로 표현되는 타원체 좌표계(오른편) |

는 에서 사이의 값(혹은 에서 사이의 값)을 갖고, 는 에서 사이의 값을 갖는다. 타원체가 아니라 고정된 반지름 갖는 구체 *상*의 위치만을 상정할 경우 위의 을 생략한 만으로도 모든 위치를 규정하기에 충분하다.

이것은 단지 *하나의* 정의일 뿐이며, 예를 들어 위도 대신 해당 벡터와 축 사이의 각도인 극각을 측정할 수도 있다는 점에 유의해야 한다. 좌표값을 의 형태로 나타내는 오랜 전통이 있지만 이 책에서는 경도-위도 형식()을 사용할 것이다. 그림 2.2에 표시된 지점은 각도값으로 구성된 혹은 타원체 좌표값을 가지며 도 단위로 주어진다.

# POINT (60 47)

지심좌표값은 미터 단위로 주어진다.

# POINT Z (2178844 3773868 4641765)

타원체 상의 지점에 대해서는 각도를 나타내는 두 가지 방법이 있다(그림 2.3). 하나는 타원체의 중심을 기준으로 측정된 각도(), 또는 해당 지점을 지나는 접선에 수직으로 측정된 각도()이다.

|  |
| --- |
| 그림 2.3: 타원체 상의 각도: 측지 위도(푸른색)와 지심 위도(붉은색) |

### 2.2.2 투영좌표계, 거리

종이 지도와 컴퓨터 화면이 지구본보다 훨씬 더 실용적이고 널리 사용되기 때문에, 우리는 공간데이터를 보통 2차원 평면 상에 투영된 형태로 보게 된다. 이차원 공간에서 위치를 계산한다는 것은 우리가 투영 좌표를 사용한다는 것을 의미한다. 타원체 좌표를 투영한다는 것은 형태, 방향, 면적 또는 이 세 가지 모두가 왜곡된다는 것을 의미한다(Iliffe and Lott 2008).

데카르트 좌표계에서 두 지점 와 간의 거리는 유클리드 거리로 계산되며, 2차원의 경우 이므로 다음의 수식으로 주어진다.

3차원의 경우는 이므로, 다음의 수식으로 주어진다.

이 거리는 지점 와 지점 사이의 직선 거리의 길이를 의미한다.

원 상의 두 지점의 경우, 두 지점 와 사이의 호의 길이는 다음과 같이 주어진다.

여기서 는 과 사이의 각도를 라디안으로 나타낸 것이다. 의 값이 매우 작다면 호가 직선에 가깝기 때문에 가 성립한다.

반지름이 인 구체 상의 두 지점 과 의 경우, 두 지점 과 를 통과하는 원(이 원의 중심은 구체의 중심과 일치) 상에서의 두 지점 사이의 호의 길이를 *대권거리*(great circle distance)라고 하며, 로 주어진다. 따라서 과 사이의 각도(라디안)는 다음과 같이 주어진다.

타원체 상의 두 지점 간의 호의 길이를 계산하는 것은 훨씬 더 복잡하다. Karney (2013)은 이와 관련하여 심도 있는 논의를 제공하였으며, PROJ 라이브러리의 일부인 GeographicLib에서 구현된 방법에 대한 자세한 설명을 제시한다.

이 거리 측정 방식들이 실제로 서로 다른 값을 산출함을 보여주기 위해, 우리는 베를린과 파리 간의 거리를 계산했다. WGS84 타원체와 완전 구체에 각각에 대해 거리를 계산했는데, gc\_는 대권거리를, str\_은 지심좌표값을 이용한 직선거리를 나타낸다.

# Units: [km]  
# gc\_ellipse str\_ellipse gc\_sphere str\_sphere   
# 879.70 879.00 877.46 876.77

### 2.2.3 한정 공간과 비한정 공간

2차원 및 3차원 유클리드 공간(와 )는 비한정 공간이다. 이 공간의 모든 선은 무한한 길이를 가지며, 면적이나 부피는 자연적인 상한이 없다. 이와는 대조적으로 구() 혹은 구체()의 공간은 한정 공간이다. 무한히 많은 점이 있을 수는 있지만, 원의 길이와 면적, 구의 반지름, 면적 및 부피는 유한하다.

이것은 사소하게 들릴 수 있지만, 공간데이터를 처리할 때 흥미로운 도전 과제로 이어진다. 상의 폴리곤은 명확하게 내부와 외부를 가진다. 공간인 구체 상에서 모든 폴리곤은 구체를 두 부분으로 나누며, 이 두 부분 중 어느 쪽을 내부로 간주하고 어느 쪽을 외부로 간주할지는 애매하며, 이는 탐색 방향(traversal direction)에 의해 정의되어야 한다(역자주: 탐색 방향은 보통 시계 방향과 반시계 방향으로 나누는데, 예를 들어, 삼각형의 변을 시계 방향으로 탐색하면 왼쪽에 있는 영역이 내부로 간주되고, 반시계 방향으로 탐색하면 오른쪽에 있는 영역이 내부로 간주된다.). 4장에서 지오메트리를 다룰 때의 결과에 대해 추가로 논의할 것이다.

## 2.3 CRS

Lott(2015)를 따라 다음과 같은 개념 정의를 사용할 것이다(이탤릭체는 Lott의 정의를 그대로 옮겨 쓴 것이다)

* **좌표계**는 *지점에 좌표를 할당하는 방법을 지정하는 수학적 규칙의 집합*이다.
* **데이텀**은 *좌표계의 원점, 축척, 방향을 정의하는 파라미터 또는 파라미터의 집합*이다.
* **측지데이텀**은 *2차원 또는 3차원 좌표계와 지구와의 관계를 설명하는 데이텀*이다(역자주: 다시 말해, 지구에 부여된 2차원 또는 3차원 좌표계를 설명하는 데이텀이다).
* **CRS**(좌표참조계)는 특정한 데이텀에 의거해 특정한 객체에 부여된 좌표계이다. 측지데이텀과 수직데이텀의 경우, 객체는 지구이다(역자주: 측지데이텀은 지구 상의 위치를 규정하는 수평데이텀이고 수직데이텀은 말그대로 지표 상의 높이를 규정하는 데이텀이다).

이 개념을 더 자세히 설명하는 읽기 쉬운 텍스트는 Iliffe and Lott(2008)이다.

지구는 규칙적인 형태를 따르지 않는다. 지표 기복이 매우 불규칙적인 형상을 띤다는 것을 잘 알려져 있지만, 평균해수면과 관련하여 일정한 중력에 의해 형성된 표면인 지오이드(geoid) 또한 불규칙적이다. 지오이드에 대한 모델 중 가장 일반적으로 사용되는 것은 회전타원체로, 이는 두 개의 동일한 반단축을 가진 타원이다. 이런 회전타원체를 지구와 어떠한 방식으로 일치시킬 것인가가 데이텀을 규정한다. 그런데 타원체를 지구의 어떤 부분에 일치시킬지, 혹은 어떤 준거 지점을 사용하여 일치시킬지에 따라 타원체의 적합도는 다양할 수 있고, 이런 연유로 인해 다양한 데이텀이 존재할 수 있는 것이다. 어떤 데이텀은 특정한 지각판에 대한 적합도에 집중하며(ETRS89), 어떤 데이텀은 전세계의 평균적인 적합도를 지향한다(WGS84). 국지적인 적합도에 치중할수록 해당 지역에서의 근사 오류는 낮아지게 된다.

위에서 제시된 정의를 살펴보면, 경도와 위도로 주어진 좌표값은 해당 데이텀이 주어질 경우에만 지구 좌표계로서의 의미를 부여받을 수 있고, 해석이 모호성이 사라질 수 있음을 함축하고 있다.

특정한 투영법이 적용된 데이터는 반드시 특정한 참조타원체(데이텀)와 결부되어 있음을 주의해야 한다. 데이텀 전환 없이 투영법만 바꾸는 것을 좌표전환(coordinate conversion)이라고 하는데, 이 좌표변환은 해당 데이텀과 결부되어 있는 특정 타원체의 좌표값에 의거해 이루어진다. 이 좌표변환의 과정은 정보의 비손실적이며 가역적이다. 변환을 위한 파라미터와 함수식은 불변이다.

새로운 데이텀에 의거해 좌표를 재계산하는 것을 좌표변환(coordinate transformation)이라고 한다. 좌표전환과 달리 좌표변환은 근사적으로 이루어진다. 왜냐하면 데이텀이 모델 적합의 결과이기 때문에 데이텀 간 변환 또한 적합된 모델이기 때문이다. 변환 함수도 경험적으로 얻어지는 것이며, 적합도와 정확성에 대한 설정을 달리하는 다양한 변환 경로가 가능할 수 있다.

판구조론은 글로벌 데이텀을 위한 고정된 객체의 위치가 시간이 흐름에 따라 변할 수 있음을 우리들에게 알려준다. 이는 데이텀 간 좌표변환이 시간 의존적일 수 있음을 시사하는 것이다. 지진으로 인해 특정 지역의 좌표가 갑작스럽게 변화할 수 있다. 국지 데이텀을 특정한 지각판에 고정시킬 수도 있지만(ETRS89), 보다 역동적으로 변화시킬 수도 있다.

## 2.4 PROJ와 지도 정확도

현재 오픈소스 지리공간 소프트웨어에서 활동하는 사람들 중 PROJ 이전의 시간을 기억하는 사람은 거의 없다. PROJ(Evenden 1990)는 1970년대에 포트란 프로젝트로 시작되어 1985년에 지도투영을 위한 C 라이브러리로 출시되었다. 이 라이브러리는 직접 투영 및 역 투영을 위한 명령줄 인터페이스를 제공하였으며, 소프트웨어와 연결하여 투영과 재투영을 곧바로 실행하는 것이 가능했다. 데이텀은 그냥 주어지는 것으로 간주되었으며 데이텀 변환은 허용되지 않았다.

2000년대 초 PROJ는 PROJ.4가 되었는데, 불변의 버전 번호가 붙은 것이다. GPS의 대두 등 여러 요인으로 인해 데이터 변환에 대한 필요성이 증가하였고, PROJ.4는 기본적인 데이텀 지원 기능을 장착하게 되었다. PROJ는 CRS를 다음과 같은 형식으로 정의한다.

+proj=utm +zone=33 +datum=WGS84 +units=m +no\_defs

“키=값”쌍은 + 기호로 시작하고 공백으로 구분된다. 이 형식은 PROJ 프로젝트가 수십 년 동안 4.x 버전으로 유지되었기 때문에 ’PROJ.4 문자열’로 알려지게 되었다. 다른 예를 들면 다음과 같다.

+ellps=bessel +towgs84=565.4,50.3,465.6,-0.399,0.344,-1.877,4.072

이 문자열은 해당 데이텀이 다른 타원체(Bessel)를 사용하며, 이 타원체를 WGS84(GPS때문에 널리 알려짐)로 변환하기 위해서는 7개(또는 3개)의 파라미터가 필요하다는 점을 잘 나타낸다.

PROJ.4 외에 널리 사용되고 있는 투영법에 대한 데이터베이스가 구축되었는데, 그것들 중 가장 널리 알려진 것인 EPSG(European Petroleum Survey Group) 레지스트리이다. 국가지도제작기관은 자국의 CRS에 대한 +towgs84 파라미터(WGS84로 변환하기 위한 파라미터)에 대한 최선의 추정치를 계산 및 업데이트하여, EPSG 등록부를 통해 배포하였다. 일부 변환의 경우, 데이텀 그리드가 제공되었는데 PROJ.4의 일부로 배포되었다. 이러한 그리드는 결국 래스터 지도를 의미하는데, 데이텀 변환이 야기하는 경도, 위도, 또는 고도의 변화값을 모든 지점에 대해 미리 계산해 둔 것이다.

PROJ.4에서는 모든 좌표변환이 WGS84로의 변환을 거쳐야 했다. 전혀 다른 데이텀을 가진 데이터를 재투영할 때도 WGS84로의 변환을 거쳐야 했다. 최대 100m의 오류는 비교적 큰 지역의 지도 제작 목적으로는 허용 가능했지만, 일부 응용 프로그램은 더 높은 정확도의 변환이 필요하다. 예를 들어, 정밀 농업, UAV(역자주: Unmanned Aerial Vehicle의 약자로 무인항공기로 번역될 수 있지만, 보통 드론을 지칭할 때 사용된다.) 비행 계획 또는 객체 추적이 있다.

2018년, 성공적인 ‘GDAL 좌표계 공동 개발’ 이니셔티브 이후, 오픈소스 지리공간 소프트웨어 스택에서 이익을 얻는 여러 회사들이 PROJ에서 보다 현대적이고 진보된 좌표변환 시스템 개발을 지원했다. 몇 년 동안 PROJ.4는 5, 6, 7, 8, 9 버전을 거치며 발전하였고, 따라서 PROJ(또는 PRJ)로 이름이 변경되었다.

가장 주목할만한 변화에 다음의 것들이 있다.

* 비록 PROJ.4 문자열이 새로운 CRS를 정의하는데 여전히 사용될 수 있지만, 모든 CRS를 포괄하기에 충분하지 않다는 점이 드러났다. 새로운 형식인 WKT-2(다음 섹션에서 설명됨)가 이를 대체한다.
* 허브 데이텀으로서의 WGS84의 지위를 박탈한다. 좌표변환이 더 이상 특정 데이텀을 거칠 필요가 없다.
* CRS A에서 CRS B로 가는 여러 변환 또는 전환 경로(소위 파이프라인)가 가능하며, 정확도 정보만 제시된다면 등재될 수 있다. PROJ는 디폴트로 가장 정확한 경로를 사용하지만 사용자가 제어할 수 있다.
* 변환 파이프라인은 축 교환 및 단위 변환을 포함하여 다수의 기본 변환 작업을 연결할 수 있다.
* 데이텀 그리드는 라이브러리와 함께 배포되지 않고 CDN(content delivery network)에서 접근할 수 있다. PROJ는 이러한 그리드에 대한 네트워크 접근을 활성화하거나 비활성화할 수 있도록 하며, 실제로 필요한 그리드 섹션만 다운로드하여 사용자의 기기에 캐시로 저장하여 향후 사용할 수 있게 한다.
* 에포케(epoche) 좌표변환, 즉 시간-의존적 좌표변환이 가능하다. 따라서 소스 및 타깃 시간을 포함하는 4차원 자표계에 대한 변환이 가능하다(역자주: 지구의 좌표계가 시간이 지남에 따라 변화해 왔으므로 특정 시대(에포케)에 의거한 변환임으로 하나의 정보로 포함시킬 수 있게 되었다는 점을 의미한다.).
* 기등록 CRS를 가진 데이터셋의 경우는 SQLite 데이터베이스에서 관리된다.
* 축 순서(경도, 위도)를 다르게 정의하는 것이 가능하다.

많은 개선이 이루어졌으며, 변환의 정확도는 이제 1미터 이하일 수 있다. 흥미로운 점은 마지막 부분이다. 수십년 동안 타원체 좌표를 가진 공간데이터의 축 순서(경도, 위도)를 자명한 것으로 가정할 수 있었지만, 이제는 더 이상 그렇지 않다. 섹션 7.7.6에서 이를 처리하는 방법을 살펴볼 것이다.

|  |
| --- |
| 그림 2.4: 영국의 OSGB 1936(EPSG:4277)를 ETRS89(EPSG:4258)로 변환하는데 사용되는 수평 데이텀 그리드 |

|  |
| --- |
| 그림 2.5: 영국의 ETRS89(EPSG:4937를 ODN 고도(EPSG:5701)로 변환하는데 사용되는 수직 데이텀 그리드 |

그림 2.4에 나타나 있는 수평 데이텀 그리드와 그림 2.5에 나타나 있는 수직 데이텀 그리드의 예는 cdn.proj.org로부터 다운로드 받았다. 데이텀 그리드는 픽셀별 정화도 값을 포함할 수 있다.

## 2.5 WKT-2

Lott (2015)는 CRS의 인코딩과 CRS 간의 변환을 WKT(well-known text) 통해 나타내는 표준을 정리한 바 있다. 이 표준(및 포맷)을 비공식적으로 WKT-2라고 부른다. 앞에서 언급했듯이, GDAL과 PROJ는 이 표준을 지원한다. 예로서 특정 CRS(EPSG:4326)은 WKT-2로 다음과 같이 표시된다.

GEOGCRS["WGS 84",  
 ENSEMBLE["World Geodetic System 1984 ensemble",  
 MEMBER["World Geodetic System 1984 (Transit)"],  
 MEMBER["World Geodetic System 1984 (G730)"],  
 MEMBER["World Geodetic System 1984 (G873)"],  
 MEMBER["World Geodetic System 1984 (G1150)"],  
 MEMBER["World Geodetic System 1984 (G1674)"],  
 MEMBER["World Geodetic System 1984 (G1762)"],  
 MEMBER["World Geodetic System 1984 (G2139)"],  
 ELLIPSOID["WGS 84",6378137,298.257223563,  
 LENGTHUNIT["metre",1]],  
 ENSEMBLEACCURACY[2.0]],  
 PRIMEM["Greenwich",0,  
 ANGLEUNIT["degree",0.0174532925199433]],  
 CS[ellipsoidal,2],  
 AXIS["geodetic latitude (Lat)",north,  
 ORDER[1],  
 ANGLEUNIT["degree",0.0174532925199433]],  
 AXIS["geodetic longitude (Lon)",east,  
 ORDER[2],  
 ANGLEUNIT["degree",0.0174532925199433]],  
 USAGE[  
 SCOPE["Horizontal component of 3D system."],  
 AREA["World."],  
 BBOX[-90,-180,90,180]],  
 ID["EPSG",4326]]

이 예시는 축 순서가 *위도*, *경도*로 설정된 좌표계를 보여주고 있다. 물론 실제로 사용되고 있는 대부분의 좌표계는 축 순서가 *경도*, *위도*이다. WGS84 타원체에 대한 *앙상블*(ENSEMBLE)은 다양한 업데이트를 나열하고 있다. 이 앙상블 중 *어떤* 것를 사용하는냐에 따라 수 미터의 오차가 발생할 수 있다. OGS:CRS84는 축 순서를 경도, 위도의 순서를 명확히 하였기 때문에 GRS84에 대한 대안으로 권장된다. 그러나 데이텀 앙상믈 문제는 해결하지 못한다.

PROJ의 역사와 최근 변화에 대한 자세한 소개는 Knudsen and Evers(2017), Evers and Knudsen(2017)의 연구를 토대로 정리한 Bivand(2020)에 잘 나타나 있다.

## 2.6 연습문제

R을 활용하여 아래의 연습문제를 풀되 패키지의 사용은 금지이다. 적절한 함수를 찾도록 노력하라.

1. 자연적 원점 0을 갖지 않는 *지리적* 측도 세가지를 나열하라.
2. 다음의 좌표, , , , 을 극 좌표로 전환하라.
3. 다음의 좌표, , , 를 데카르트 좌표로 전환하라.
4. 지구를 반지름이 6371km인 완전한 구체라고 가정하고, 다음의 두 지점 간의 대권거리를 구하라. 과 , 과 , 과 , 과 (단위: 각도)의 대권거리는 몇 도인가?

# 3. 지오메트리

2장에서 좌표계에 대해 배웠으므로 이제 이 장에서는 좌표계를 이용하여 지오메트리를 어떻게 정의할 수 있는지에 대해 배울 것이다. 다음의 사항이 다루어진다.

* 심플 피처(simple feature): 포인트, 라인, 폴리곤 지오메트리를 다루는 표준
* 지오메트리에 적용되는 다양한 오퍼레이션
* 커버리지(coveratge): 공간 혹은 시공간의 함수
* 테셀레이션(tessellation): 하나의 지역을 여러개의 하위지역으로 세분한 것
* 네트워크

구면 지오메트리는 4장에서 논의되며, 래스터와 공간 혹은 시공간을 직사각형 형태로 분할한 또 다른 세분 체계는 6장에서 다룬다.

## 3.1 심플 피쳐 지오메트리

심플 피처 지오메트리는 피처의 지오메트리를 설명하는 방법이다. 여기서 피처란 지오메트리를 가진 사물을 의미하는데, 암묵적으로 시간 속성을 가질 수 있고, 사물 자체를 묘사하기 위한 라벨이나 사물을 정량적으로 측정한 값과 관련된 속성을 가질 수 있다. 단순 피처 지오메트리의 주된 용도는 2차원 공간의 기하학적 형태를 포인트, 라인, 또는 폴리곤에 의거해 설명하는 것이다. “심플”이라는 형용사가 붙은 이유는 라인이나 폴리곤 지오메트리 역시 포인트 지오메트리와 그것을 연결한 직선으로 표현될 수 있기 때문이다.

심플 피처 억세스(access)는 심플 피처 지오메트리를 설명하는 표준으로(Herring 2011, 2010; ISO 2004), 다음의 사항을 포함한다.

* 클래스 위계
* 오퍼레이션의 집합
* 이항 엔코딩과 텍스트 엔코딩

먼저 가장 일반적으로 사용되는 7가지 심플 피처 지오메트리 유형에 대해 논의할 것이다.

### 3.1.1 7개의 대표 지오메트리

단일(single) 피처를 나타내기 위해 사용되는 가장 일반적인 심플 피처 지오메트리 유형은 다음과 같다.

| 유형 | 설명 |
| --- | --- |
| POINT | 단일 포인트 지오메트리 |
| MULTIPOINT | 포인트의 집합 |
| LINESTRING | 단일 라인스트링(두개 이상의 포인트가 직선으로 연결되어 있음) |
| MULTILINESTRING | 라인스트링의 집합 |
| POLYGON | 내부 링(구멍을 의미)을 가질 수 있는 외부 링 |
| MULTIPOLYGON | 폴리곤의 집합 |
| GEOMETRYCOLLECTION | 위에서 언급된 모든 지오메트리의 집합 |

|  |
| --- |
| 그림 3.1: 심플 피처 지오메트리의 주요 유형 |

그림 3.1은 이러한 기본 지오메트리 유형의 예를 보여준다. 지오메트리를 표현한 사람이 읽을 수 있는 WKT 표기법은 다음과 같다.

POINT (0 1)  
MULTIPOINT ((1 1), (2 2), (4 1), (2 3), (1 4))  
LINESTRING (1 1, 5 5, 5 6, 4 6, 3 4, 2 3)  
MULTILINESTRING ((1 1, 5 5, 5 6, 4 6, 3 4, 2 3), (3 0, 4 1, 2 1))  
POLYGON ((2 1, 3 1, 5 2, 6 3, 5 3, 4 4, 3 4, 1 3, 2 1),  
 (2 2, 3 3, 4 3, 4 2, 2 2))  
MULTIPOLYGON (((2 1, 3 1, 5 2, 6 3, 5 3, 4 4, 3 4, 1 3, 2 1),  
 (2 2, 3 3, 4 3, 4 2, 2 2)), ((3 7, 4 7, 5 8, 3 9, 2 8, 3 7)))  
GEOMETRYCOLLECTION (  
 POLYGON ((2 1, 3 1, 5 2, 6 3, 5 3, 4 4, 3 4, 1 3, 2 1),  
 (2 2 , 3 3, 4 3, 4 2, 2 2)),  
 LINESTRING (1 6, 5 10, 5 11, 4 11, 3 9, 2 8),  
 POINT (2 5),  
 POINT (5 4)  
)

좌표는 공백으로 구분되고, 포인트는 쉼표로 구분된다. 집합은 괄호로 묶여 있으며, 쉼표로 구분된다. 폴리곤은 외부 링과 (만일 있다면) 내부 링(구멍을 나타냄)으로 구성된다.

지오메트리의 개별 포인트는 최소한 두 개의 좌표를 포함하며, 이 좌표는 x와 y 순서로 나열된다. 이 좌표가 타원체 좌표를 나타내는 경우, x는 일반적으로 경도를, y는 위도를 의미하지만, 경우에 따라 위도와 경도의 순서로 나열될 수도 있다(섹션 2.4 및 섹션 7.7.6 참조).

### 3.1.2 심플 지오메트리, 타당한 지오메트리, 링 디렉션

라인리스팅은 자기 교차하지 않을 때 심플이라고 부를 수 있다.

# LINESTRING (0 0, 1 1, 2 2, 0 2, 1 1, 2 0)

# is\_simple   
# FALSE

타당한 폴리곤과 멀티폴리곤은 다음의 모든 프로프티를 보유한다.

* 폴리곤 링은 닫혀있다(첫번째 포인트와 마지막 포인트는 동일하다)
* 폴리곤 구멍(내부 링)은 외부 링의 안쪽에 있다.
* 폴리곤 내부 링은 외부 링과 한 지점에서 만날 수는 있지만 라인을 공유할 수는 없다.
* 폴리곤 링은 자신의 경로를 반복하지 않는다.
* 멀티폴리곤에서 외부 링은 다른 외부 링과 한 지점에서 만날 수는 있지만 라인을 공유할 수는 없다.

이것들 어느 하나라도 만족시키지 못한다면 해당 지오메트리는 타당하지 않다. 타당하지 않는 지오메트리는 일반적으로 오퍼레이션 과정에서 오류를 발생시키지만, 보통 사전에 타당한 지오메트리로 수정된다.

또 다른 규칙은 폴리곤의 외부 링은 반시계 방향으로 감겨야 하며, 구멍은 시계 방향으로 감겨야 한다는 것이다. 그러나 이러한 규칙을 따르지 않는 폴리곤도 여전히 타당한 것으로 간주된다. 이러한 ‘시계 방향’ 개념은 구체에서의 그리 유용하지 않다. 예를 들어, 적도를 폴리곤으로 간주할 경우, 북반구와 남반구 중 어느 쪽이 ’내부’라고 할 수 있는가? 여기서 채택된 규칙은 폴리곤을 변을 시계 방향으로 탐색할 때 왼쪽에 있는 영역을 폴리곤의 내부로 간주하는 것이다(7.3절 참조).

### 3.1.3 Z 좌표와 M 좌표

심플 피처 지오메트리의 단일 포인트(버텍스)는 X 좌표와 Y 좌표 외에 다음을 가질 수 있다.

* Z 좌표: 고도
* M 좌표: 측정치

M 속성이 버텍스의 프로퍼티라는 점을 염두에 두면 M 좌표는 유용하게 사용될 수 있을 것으로 보인다. 예를 들어, LINESTRING을 통해 일종의 경로 데이터를 엔코딩한다고 하면, 개별 버텍스에 시간 속성을 할당할 수 있다. 그러나 경로가 자기-교차하게 되면 비타당 혹은 비심플(non-simple)이 된다. 자기-교차성을 검토하기 위해 X 좌표와 Y 좌표만을 고려할 때는 드러나지 않는다.

Z와 M은 자주 사용되지 않으며, 이를 유용하게 활용할 수 있는 소프트웨어 지원도 (아직) 드물다. 그래도 Z와 M의 WKT 표현은 비교적 쉽게 이해할 수 있다.

# POINT Z (1 3 2)

# POINT M (1 3 2)

# LINESTRING ZM (3 1 2 4, 4 4 2 2)

### 3.1.4 엠프티 지오메트리

피쳐 지오메트리 프레임워크에서 매우 중요한 개념이 엠프티(empty) 지오메트리이다. 엠프티 지오메트리는 기하학적 오퍼레이션(3.2절)을 실행할 때 자연스럽게 생성된다. 예를 들어 POINT (0 0)과 POINT (1 1)의 인터섹션 여부를 검토한다고 하자.

# GEOMETRYCOLLECTION EMPTY

두 포인트는 교차하지 않으므로 엠프티 집합이 도출된다. 엠프티 포인트를 비어있지 않은 지오메트리와 결합하면(합집합을 구하면) 엠프티 포인트는 소실된다.

모든 지오메트리 유형은 엠프티 지오메트리를 나타내는 특정한 값을 가지고 있다.

# POINT EMPTY

# LINESTRING M EMPTY

엠프티 집합이 생성된다는 점에서는 동일하고 단지 디멘션만 다를 뿐이다(3.2절 참조).

### 3.1.5 10개의 부수적인 지오메트리

다음의 10가지 지오메트리는 매우 드물게 사용되지만, 사용 빈도가 증가하는 경향이 있다.

| 유형 | 설명 |
| --- | --- |
| CIRCULARSTRING | 서큘러스트링은 기본적인 곡선 유형으로 직선 유형의 LINESTRING과 유사하다. 단일 세그먼트는 세 점이 필요하며, 시작점과 끝점(첫 번째와 세 번째 점) 및 호 상의 다른 점이 필요하다. 예외는 닫힌 원의 경우로, 이 경우 시작점과 끝점이 동일하다. 이 경우 두 번째 점은 반드시 호의 중심, 즉 원의 반대편이어야 한다. 호를 연결하려면 이전 호의 마지막 점이 다음 호의 첫 번째 점이 되며, 이는 LINESTRING에서도 동일하다. 따라서 타당한 서큘러 문자열은 1보다 큰 홀수 개의 점을 가져야 한다. |
| COMPOUNDCURVE | 컴파운트커브는 곡선(서큘러) 세그먼트와 선형 세그먼트를 모두 포함하는 단일 연속 곡선이다. 구성 요소를 잘 갖추고 있어야 하며, 모든 구성 요소의 끝점(마지막을 제외하고)은 다음 구성 요소의 시작점과 일치해야 한다. |
| CURVEPOLYGON | 커브폴리곤 속에 컴파운트커브가 포함된 예: CURVEPOLYGON( COMPOUNDCURVE( CIRCULARSTRING(0 0,2 0, 2 1, 2 3, 4 3),(4 3, 4 5, 1 4, 0 0)), CIRCULARSTRING(1.7 1, 1.4 0.4, 1.6 0.4, 1.6 0.5, 1.7 1)) |
| MULTICURVE | 멀티커브는 곡선으로 구성된 1차원 지오메트리 컬렉션이다. 이 컬렉션은 라인 문자열, 서큘러 문자열 또는 컴파운드 문자열을 포함할 수 있다. |
| MULTISURFACE | 멀티서피스는 동일한 CRS를 사용하는 서피스로 구성된 2차원 지오메트리 컬렉션이다. |
| CURVE | 커브는 일반적으로 몇 개의 점의 연결로 정의되는 1차원 지오메트리 객체이며, 점들 간의 인터폴레이션 형태에 따라 커브의 하위 유형이 정의된다. |
| SURFACE | 서피스는 2차원 지오메트리 객체이다. |
| POLYHEDRALSURFACE | 폴리헤드럴서피스는 공통 경계 세그먼트를 공유하는 연속적인 폴리곤의 컬렉션이다. |
| TIN | 불규칙삼각망(triangulated irregular network)은 오직 삼각형으로만 구성된 폴리헤드럴서피스이다. |
| TRIANGLE | 트라이앵글은 세 개의 비공선적(non-collinear) 버텍스로 구성된(역자주: 세 점이 일직선 상에 놓이지 않음을 의미한다), 내부 경계가 없는 폴리곤이다. |

CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON은 SFA 표준에 언급되지 않지만 SQL-MM 파트3 표준에는 포함되어 있다. 위 표의 설명 부분은 PostGIS 매뉴얼에서 그대로 가져온 것이다.

### 3.1.6 텍스트 엔코딩과 바이너리 엔코딩

심플 피처 표준에는 도 가지 엔코딩 방식이 포함되어 있다. 하나는 텍스트 인코딩이고 다른 하나는 바이너리 인코딩이다. 위에서 사용된 KWT는 인간-해독가능(human-readable)이고, KWB(well-known binary) 인코딩은 기계-해동가능(machine-readable)이다. WKB 인코딩은 정보 손실이 없으며 일반적인 텍스트 인코딩(및 디코딩)보다 작업 속도가 빠르다. 이는 R 패키지 **sf**와 GDAL, GEOS, liblwgeom, s2geometry 라이브러리 간의 모든 통신에서 사용된다(그림 1.7).

## 3.2 지오메트리에 적용되는 오퍼레이션

심플 피처 지오메트리의 프로퍼티를 추출하는 것이 가능하며, 심플 피처 지오메트리의 결합을 통해 새로운 지오메트리가 생성된 경우, 그것의 프로퍼티를 추출하는 것 역시 가능하다. 이 섹션에서는 기하학적 프로퍼티에 전적으로 초점을 맞춘 오퍼레이션에 대해서만 간략하게 다룬다. 5장에서는 비기하학적 피처 속성 분석에 초점을 맞춘다. 이 섹션의 일부 내용은 Pebesma (2018)로부터 가져온 것이다.

기하학적 프로퍼티에 대한 오퍼레이션을 투입과 산출에 의거해 분류하는 것이 가능하다. 우선 산출의 관점에서, 다음과 같이 구분할 수 있다.

* 프레디케이트(predicate): 특정 프로퍼티가 TRUE임을 주장하는 논리값
* 측정치(measure): 양(수치, 측정 단위가 있을 수 있음)
* 변형(transformation): 새롭게 생성된 지오메트리

오퍼레이션이 어떤 지오메트리에 적용되느냐에 따라 다음과 같이 분류하는 것이 가능하다.

* 유너리(unary)단일 지오메트리에 적용
* 바이너리(binary): 지오메트리의 쌍에 적용
* 에너리(n-ary): 지오메트리의 집합에 적용

### 3.2.1 유너리 프리디케이트: 단항 조건식(unary predicate)

유너리 프리디케이트는 하나의 지오메트리의 하나의 프로퍼티를 설명한다. is\_simple, is\_valid, is\_empty와 같는 프리디케이트는 각각 지오메트리가 심플한지, 타당한지, 엠프티한지의 여부에 대한 논리값을 반환한다. is\_longlat 프리디케이트는 주어진 CRS가 경위도 좌표계에 기반하고 있는지 평면 좌표계에 기반하고 있는지의 여부를 반환한다. is(geometry, class)는 지오메트리가 특정한 클래스에 속하는지의 여부를 확인한다.

### 3.2.2 바이너리 프리디케이트와 DE-9IM: 이항 조건식(binary predicate)과 DE-9IM

DE-9IM(Dimensionally Extended Nine-Intersection Model)(Clementini, Di Felice, and Oosterom 1993; Egenhofer and Franzosa 1991)은 2차원 공간()에서 두 지오메트리간의 정성적 관계를 설명하는 모델이다. 모든 지오메트리는 디멘션 값을 가진다.

* 포인트는 0
* 라인 지오메트리는 1
* 폴리곤 지오메트리는 2
* 엠프티 지오메트리는 F(거짓)

모든 지오메트리는 내부(I), 경계(B), 외부(E)를 가지며, 이것들이 어떤 역할은 하는지는 폴리곤의 경우에 보다 명확히 드러난다.

* 라인의 경계는 종점에 의해 형성되고, 선상의 모든 비종점이 내부를 구성한다.
* 포인트는 0차원의 내부를 가지지만, 경계는 없다.

|  |
| --- |
| 그림 3.2: DE-9IM: 폴리곤의 내부, 경계, 외부(행)와 라인스트링의 내부, 경계, 외부(열) 간의 인터섹션 |

그림 3.2는 폴리곤과 라인스트링의 I, B 및 E 간의 인터섹션을 빨간색으로 표시한 것이다. 개별 그래프의 제목에 인터섹션의 결과 도출되는 디멘션(0, 1, 2, 또는 F)이 표시되어 있다. 폴리곤 지오메트리와 라인 지오메트리의 관계는 바로 이러한 디멘션의 연속(concatenation)이다.

# [,1]   
# [1,] "1020F1102"

첫 세 문자는 *첫 번째* 지오메트리(폴리곤)의 내부와 관련이 있다. 그림 3.2에는 첫 번째 지오메트리가 행에 배열되어 있다. 더 나아가 지오메트리의 쌍에 대해 *마스크(mask) 문자열*로 표현된 특정한 조건의 만족 여부도 질의할 수 있다. 예를 들어, 문자열 "0\*\*\*\*\*\*"는 두 번째 지오메트리가 첫 번째 지오메트리의 내부와 하나 이상의 경계 점을 공유할 때 TRUE로 평가된다. 여기서 기호 \*는 디멘션(0, 1, 2 또는 F)을 의미하는 것으로 네 개 중 어는 것도 올 수 있다. 마스크 문자열 "T\*\*\*\*\*\*\*\*"은 내부가 서로 교차하는 지오메트리 쌍을 찾아내느데, 여기서 기호 T는 차원 디멘션이 0, 1, 2이면서 비엠프티 인터섹션인 경우를 의미한다.

더 나아가 바이너리 프리디케이트는 DE-9IM에서 규정된 정의를 활용함으로써 일반 언어 동사로 표현될 수 있다. 예를 들어, 프리디케이트 equals는 관계 "T\*F\*\*FFF\*"와 동일하다. 어떤 두 지오메트리가 이러한 관계를 만족한다면, 그 둘은 (위상적으로)는 동일하지만, 노드의 순서가 다를 수 있다.

바이너리 프리디케이트를 나열하면 다음과 같다.

| 프리디케이트 | 의미 | 역프리디케이트 |
| --- | --- | --- |
| contains | A 포인트 어느 것도 B의 외부에 있지 않다. | within |
| contains\_properly | A는 B를 포함하며, B의 어떤 포인트도 A의 경계 상에 있지 않다. |  |
| covers | B의 어떤 포인트도 A의 외부에 있지 않다. | covered\_by |
| covered\_by | covers의 반대 |  |
| crosses | A와 B는 일부 내부 포인트를 공유하지만 모든 포인트를 공유하는 것은 아니다. |  |
| disjoint | A와 B는 어떤 포인트도 서로 공유하지 않는다. | intersects |
| equals | A와 B는 위상적으로 동일하다. 노드 순서나 노드 수가 다를 수 있으며, A가 B를 포함하고 A가 B의 내부에 있는 것과 동일하다. |  |
| equals\_exact | A와 B는 기하학적으로 동일하며, 노드 순서도 동일하다. |  |
| intersects | A와 B가 완전 분리의 관계를 가지지 않는다. | disjoint |
| is\_within\_distance | A가 주어진 거리보다 B에 더 가깝게 위치해 있다. |  |
| within | B의 어떤 포인트도 A의 외부에 있지 않다. | contains |
| touches | A와 B는 최소한 한 개의 경계 포인트를 공유한다. 내부 포인트를 공유하는 것은 아니다. |  |
| overlaps | A와 B가 다수의 포인트를 공유한다. 디멘션은 A와 B의 디멘션과 동일하다. |  |
| related | A와 B가 주어진 마스크 패턴을 준수하는지의 여부를 반환한다. |  |

위키피디어의 DE-9IM 페이지를 가면, 개별 동사에 해당하는 relate 마스크 패턴을 확인할 수 있다. 왜냐하면 *covers*와 *contains*와 같은 동사(그 역도 마찬가지)는 그 의미를 직관적으로 파악하기 어려울 때가 있다.

* 만일 A가 B를 포함한다면, B는 A의 외부나 경계와 어떠한 공유 포인트도 없다.
* 만일 A가 B를 덮으면, B는 A의 외부와 어떠한 공유 포인트도 없다.

### 3.2.3 유너리 측도

유너리 측도는 지오메트리의 퍼로퍼티를 설명하는 측정값이나 양을 반환한다.

| 측도 | 반환값 |
| --- | --- |
| dimension | 포인트는 0, 라인은 1, 폴리곤은 2, 엠프티 지오메트리에 대해서는 NA |
| area | 지오메트리의 면적 |
| length | 라인 지오메트리의 길이 |

### 3.2.4 바이너리 측도

distance는 지오메트리 간의 거리를 반환한다. 질적 측도로서의 (마스크가 없는) relate는 관계 패턴을 제공한다. 두 지오메트리간의 기하학적 관련성에 대한 설명은 섹션 3.2.2에 나타나 있다.

### 3.2.5 유너리 변환자

유너리 변환자는 지오메트리별로 작동하며, 각 지오메트리에 대해 새로운 지오메트리를 반환한다.

| 변환자 | 반환 지오메트리 |
| --- | --- |
| centroid | 투입 지오메트리의 센트로이드로 구성된 POINT 유형의 지오메트리 |
| buffer | 투입 지오메트리보다 더 큰(혹은 더 작은) 지오메트리: 버퍼 사이즈에 따라 산출 지오메트리의 크기가 달라짐 |
| jitter | 이변량 균등 분포를 이용하여 조금 전위된 지오메트리 |
| wrap\_dateline | 날짜 변경선(데이라인)을 더 이상 덮거나 교차하지 않는 조각으로 분할된 지오메트리 |
| boundary | 투입 지오메트리의 경계를 가진 지오메트리 |
| convex\_hull | 투입 지오메트리의 컨벡스헐을 가진 지오메트리(그림 3.3) |
| line\_merge | MULTILINESTRING 내의 LINESTRING 요소들을 결합하여 더 긴 LINESTRING을 형성한 지오메트리 |
| make\_valid | 타당하게 교정된 지오메트리 |
| node | 노드가 없는 교차점에 노드를 추가한 라인 지오메트리. 개별적인 라인 지오메트리에만 적용됨 |
| point\_on\_surface | 서피스에 임의의 포인트를 가진 지오메트리 |
| polygonize | 폐쇄 링을 형성하는 라인으로부터 생성된 폴리곤 지오메트리 |
| segmentize | 주어진 밀도 또는 최소 거리를 만족하는 노드로 구성된 라인 지오메트리 |
| simplify | 버텍스/노드를 제거함으로써 단순화된 라인 혹은 폴리곤 지오메트리 |
| split | 라인스트링에 의해 분할된 지오메트리 |
| transform | 새로운 CRS로 변형 혹은 전환된 지오메트리(2장) |
| triangulate | 들로네(Delauney) 삼각망으로 구성된 지오메트리(그림 3.3) |
| voronoi | 투입 지오메트리로부터 형성된 보로노이(Voronoi) 테셀레이션(그림 3.3) |
| zm | Z 좌표 및 M 좌표가 수정된(일부 좌표의 삭제 혹은 새로운 좌표의 첨가) 지오메트리 |
| collection\_extract | 특정 유형의 GEOMETRYCOLLECTION으로부터 일부를 추출한 지오메트리 |
| cast | 유형이 전환된 지오메트리 |
| + | 주어진 벡터만큼 전위된 지오메트리 |
| \* | 스칼라 또는 매트릭스가 곱해진 지오메트리 |

|  |
| --- |
| 그림 3.3: 포인트의 집합, 왼편은 컨벡스 헐(빨간색), 가운데는 보로노이 폴리곤, 오른편은 들로네 삼각망 |

### 3.2.6 바이너리 변환자

바이러니 변환자는 지오메트리 쌍에 적용되어 새로운 지오메트리를 생성하는 함수이다. 다음과 같은 것이 있다.

| 함수 | 반환 지오메트리 | 인픽스 오퍼레이터 |
| --- | --- | --- |
| intersection | 두 지오메트리 겹치는 부분에 대한 지오메트리 | & |
| union | 두 지오메트리를 결합한 지오메트리로, 내부 경계를 제거하고, 중복되는 포인트, 노드, 또는 라인을 삭제한다. | | |
| difference | 두 번째 지오메트리와 중복되는 부분을 제거한 첫 번째 지오메트리 | / |
| sym\_difference | 중복되는 부분을 제거한 이후에 두 지오메트리를 결한한 지오메트리로 intersection의 반대 | %/% |

### 3.2.7 에너리 변환자

에너리 변환자는 지오메트리의 집합에 작용한다. union을 적용하면 모든 지오메트리의 결합 결과를 얻을 수 있다. 이런 방식이 아니라면, 동일한 차원을 가진 지오메트리 집합은 MULTI-유형 지오메트리로 결합하거나 GEOMETRYCOLLECTION으로 결합할 수 있다. 이 경우, union을 적용하지 않았기 때문에 두 폴리곤 링이 공유 경계선을 가질 때와 같이 타당하지 않은 지오메트리가 생성될 수 있다.

에너리 intersection과 difference의 경우 형식적으로는 단일 아규먼트를 취하지만 모든 쌍, 세 쌍, 네 쌍 등에 대해 순차적으로 작동한다. 그림 3.4를 살펴보자. 세 개의 상자가 모두 겹치는 영역을 어떻게 식별할 수 있을까? 바이너리 intersection을 사용하면 모든 쌍(1-1, 1-2, 1-3, 2-1, 2-2, 2-3, 3-1, 3-2, 3-3)에 대한 intersection을 얻을 수 있다. 하지만 두 개 이상의 지오메트리가 교차하는 영역을 식별할 수는 없다. 그림 3.4의 오른쪽 그림은 에너리 intersection을 보여주는데, 한 개, 두 개, 또는 *더 많은* 지오메트리의 교차를 통해 생성된 7개의 겹치지 않는 고유한 지오메트리를 확인할 수 있다.

|  |
| --- |
| 그림 3.4: 왼쪽: 세 개의 겹치는 정사각형이 있는데, 모든 사각형이 겹치는 부분을 어떻게 확인할 수 있을까 오른쪽: 서로 겹치지 않는 고유한 에너리 교차 |

유사하게, 집합 에 에너리 difference를 적용하여 를 생성할 수 있다. 결과가 그림 3.5에 나타나 있는데, 왼편에는 원래 집합을 보여주고 있고, 오른편에는 투입 지오메트리의 순서를 바꾼 후의 집합을 보여주고 있다. 이는 결과가 입력 지오메트리의 순서에 의존한다는 사실을 명확히 하기 위함이다. 결과로 얻어진 지오메트리는 서로 겹치지 않는다.

|  |
| --- |
| 그림 3.5: 박스를 서로다른 순서에 따라 differece를 적용한 결과: 왼쪽은 원리 순서이고 오른쪽은 반대 순서 |

## 3.3 정밀도

기하학적 오퍼레이션, 예를 들어 특정 점이 선 위에 있는지 여부를 찾는 오퍼레이션은 R에서 사용되는 8바이트 더블과 같은 배정밀도 부동소수점 수로 좌표가 표현될 때 실패할 수 있다. 자주 선택되는 해결책은 연산 전에 좌표의 정밀도를 제한하는 것이다. 이를 위해 정밀도 모델이 채택된다. 가장 일반적인 방법은 하나의 계수 를 선택하고, 원래 좌표 에에서 반올림된 좌표 를 계산하는 것이다.

이러한 종류의 반올림은 좌표를 간격이 인 규칙적인 그리드 상의 점으로 변환하며, 이는 기하학적 연산에 유익하다. 물론, 이러한 반올림은 면적이나 거리와 같은 모든 계산에도 영향을 미치며, 유효한 기하를 무효하게 만들 수 있다. 어떤 정밀도 값이 특정 애플리케이션에 가장 적합한지는 일반적으로 상식과 시행착오를 결합해 결정된다.

## 3.4 커버리지: 테셀레이션과 래스터

OGC(Open Geospatial Consortium, 오픈지리공간컨소시움)는 *커버리지*(coverage)를 “시공간적 도메인 내의 임의의 직접 위치에 대해 그 범위에서 값을 반환하는 함수로 작용하는 피처”라고 정의한다 (Baumann, Hirschorn, and Masó 2017). *함수*가 존재한다는 것은 시공간적 도메인 내의 모든 “포인트”, 즉 특정한 지점과 특정한 시점의 모든 조합마다 범위에 대해 *단일* 값을 우리가 얻을 수 있다는 것을 의미한다. 이는 시공간적 현상에서 매우 일반적인 상황이며, 몇 가지 예를 들 수 있다.

* 경계 분쟁을 제외하고 얘기한다면, 특정 시점의 특정 지역 내 모든 지점(도메인)은 단일 행정 단위(범위)에 속한다.
* 특정 시점의 특정 지역 내 모든 지점(도메인)은 특정한 토지피복 유형(범위)을 갖는다.
* 특정 지역(도메인)의 모든 지점은 단일한 고도값(범위)을 가지며, 이는 주어진 평균 해수면을 기준으로 측정될 수 있다.
* 3차원 기체의 모든 시공간적 지점(도메인)은 온도(범위)에 대해 단일 값을 갖는다.

여기서 주의해야 할 점은 관찰이나 측정은 항상 시간과 공간을 소요하기 때문에, 측정된 값은 항상 시공간적 부피(volume)에 대한 평균값일 수 밖에 없다는 점이다. 따라서 범위 변수를 무부피의 “포인트”에 대해 측정하는 경우는 극히 드물다. 그러나 많은 실제 사례에서 측정된 부피는 “포인트”로 간주될 만큼 충분히 작기는 하다. *토지 피복* 유형과 같은 변수의 경우, 구별되는 유형이 측정된 면적 단위와 관련하여 의미가 있도록 부피를 선택해야 한다.

주어진 예의 처음 두 가지에서 범위 변수는 *범주형*이며, 마지막 두 가지에서 범위 변수는 *연속형*이다. 범주형 범위 변수의 경우, 넓은 지역이 동일한 범위 값을 가지면, 이러한 데이터를 효율적으로 표현하는 방법은 동일한 값을 가진 지역의 경계를 저장하는 것이다. 예를 들어, 국가 경계와 같은 경계가 이에 해당한다. 이는 단순 피처 지오메트리(폴리곤 또는 멀티폴리곤)을 사용하여 수행할 수 있지만, 이로 인해 몇 가지 도전 과제가 발생할 수 있다.

* 단순 피처 폴리곤들이 서로 겹치지 않고 틈새도 없어야 하지만, 이러한 점이 반드시 보장된다고 말하기 어렵다.
* 단순 피처는 인접한 두 폴리곤의 경계 상에 놓인 포인트를 하나의 폴리곤에만 할당하지 못하는데, 이는 커버리지로서의 해석과 충돌한다.

### 3.4.1 토폴로지 모델

폴리곤 커버리지의 겹침과 틈새가 없도록 보장된 데이터 모델을 *토폴로지* 모델(topological model)이라고 하는데, GRASS GIS나 ArcGIS와 같은 지리정보시스템(GIS)에서 그 예를 찾을 수 있다. 토폴로지 모델은 폴리곤 경계를 한 번만 저장하고, 경계의 양쪽에 어떤 폴리곤이 있는지를 등록한다.

토폴로지 모델에서 동일한 범위 값을 가진 지역들에 대해 (멀티)폴리곤 집합을 도출하는 것은 간단하다. 폴리곤의 세트로부터 토폴로지를 재정의하는 역의 과정을 수행하기 위해서는 오류에 대한 임계값을 설정해야하고 겹침과 틈새를 처리하는 특정한 방법을 고려해야 한다.

### 3.4.2 래스터 테셀레이션

테셀레이션은 공간(2차원 또는 3차원)을 폴리곤을 이용해 더 작은 요소로 세분화한 것을 말한다. 규칙(정규) 테셀레이션은 정규 폴리곤(삼각형, 사각형 또는 육각형)으로 구성되는 테셀레이션을 의미한다. 정사각형을 사용하는 테셀레이션이 공간데이터에 일반적으로 사용되는 이를 래스터 데이터라고 부른다. 래스터 데이터는 공간적 디멘션 를 정규 셀로 세분하는데, 각 셀 는 좌측 닫힘 및 우측 개방 범위(구간)로 형성된다(역자주: 어떤 범위가 좌측 닫힘 치 우측 개방의 형식으로 규정되었다는 것은 범위의 왼쪽 끝점은 포함되지만, 범위의 오른쪽 끝점은 포함되지 않는다는 것을 의미한다.)

여기서 는 오프셋값, 는 구간(셀 또는 픽셀) 크기, 셀 인덱스인 는 임의의 값을 취할 수 있지만 연속적인 정수값이어야한다. 는 -축(북거)에 대해서는 보통 음의 값을 취하는데 래스터의 행은 남쪽으로 갈수록 증가하는 반면 -좌표값은 감소하는 것과 보조를 맞추기 위함이다.

일반적인 폴리곤 테셀레이션에서는 두 개의 폴리곤이 공유하는 경계에 위치한 점의 할당이 모호하지만, 정규 테셀레이션에서는 좌측 닫힘 “[” 및 우측 개방 “)” 구간을 사용하기 때문에 이러한 모호성이 해소된다. 이는 -좌표에 음의 값이 -좌표에 양의 값이 적용된 적용된 래스터의 경우, 각 셀의 모서리 점들 중 좌측 상단의 것만이 해당 셀의 부분으로 간주된다는 것을 의미한다. 이로 인해 발생할 수 있는 예기치 못한 결과중 한 예가 그림 3.6에 나타나 있다.

|  |
| --- |
| 그림 3.6: 래스터화의 예기치 못한 결과: 각 셀의 좌측 상단 지점만 셀의 내부로 간주되기 때문에 대각선 아래에서 붉은 선에 접촉한 셀들도 래스터로 전환되었다. |

시간 차원을 좌측 닫힘 및 우측 개방 구간으로 세분하는 것은 매우 일반적이며, 이는 R의 **xts** 패키지와 같은 시계열 소프트웨어의 암묵적인 기본 가정이다. 즉, 시간 스탬프(시점)는 시간 간격의 시작을 나타낸다. 다른 모델이 결합될 수도 있다. 예를 들어, 심플 피처 폴리곤을 사용하여 공간을 세분하고 이를 정규 시간 테셀레이션과 결합하여 시공간 벡터 데이터 큐브 개념을 구현할 수 있다. 래스터 및 벡터 데이터 큐브에 대한 논의는 6장에서 다룬다.

위에서 언급한 것처럼, 에 대한 정규 테셀레이션을 생성하는데 활용될 수 있는 폴리곤으로 정사각형 외에 삼각형과 육각형도 있다. 3차원 구체로 확장하면 더 많아지는데, 큐브, 정팔면체, 정이십면체, 정십이면체 등을 포함한다. 큐브를 기반으로 하는 공간 인덱스는 s2geometry이며, H3 라이브러리는 정이십면체를 사용하는데, 밀집화를 위해 (주로) 육각형을 사용한다. 지구 전체를 포괄하는 모자이크 역시 이산 글로벌 그리드라고 불린다.

## 3.5 네트워크

공간적 네트워크느 일반적으로 라인(LINESTRING) 요소로 구성되지만, 네트워크로서의 완결성을 위한 추가적인 위상적 특성을 가진다.

* 라인스트링의 시작점과 끝점은 다른 라인스트링의 시작점이나 끝점에 연결될 수 있으며, 이를 통해 노드와 엣지의 집합이 형성된다.
* 엣지는 방향성을 가질 수 있으며, 이 경우 연결(흐름, 수송)은 한 방향으로만 가능하다.

**osmar**(Schlesinger and Eugster 2013), **stplanr**(Lovelace, Ellison, and Morgan 2022), **sfnetworks**(van der Meer et al. 2022)와 같은 R 패키지는 네트워크 객체를 구성하고 이를 다루는 기능을 제공하며, 네트워크를 통한 최단 또는 최속 경로를 계산할 수 있게 해준다. **spatstat** 패키지(Baddeley, Turner, and Rubak 2022; Baddeley, Rubak, and Turner 2015)는 포인트 패턴 분석을 선형 네트워크 상에서도 할 수 있는 기능을 제공한다(11장). Lovelace, Nowosad, and Muenchow (2019)의 12장은 네트워크 데이터를 사용한 교통 애플리케이션에 대해 다루고 있다.

## 3.6 연습문제

다음의 연습문제를 풀되, 적절한 곳에서 R을 활용하라.

1. 2차원(평면) 공간에서 심플 피처 지오메트리로 표현될 수 없는 지오메트리의 두 가지 예를 제시하고, 그림으로 표현하시오.
2. 좌표 10.542, 0.01, 45321.6789를 정밀도 값 1, 1e3, 1e6, 및 1e-2를 사용하여 재계산하시오.
3. 에너리 인터섹션이 요구되는 현실 문제를 제시하시오.
4. 지점별로 하나의 폐쇄 폴리곤을 가지는 보로노이 다이어그램(그림 3.3)을 어떻게 만들 수 있는지 설명하시오.
5. 다음의 지오메트리에 대해 유너리 측도 dimension을 계산하시오. POINT Z (0 1 1), LINESTRING Z (0 0 1,1 1 2), POLYGON Z ((0 0 0,1 0 0,1 1 0,0 00))
6. LINESTRING(0 0,1 0)과 LINESTRING(0.5 0,0.5 1)의 DE-9IM 관계를 설명하시오. 개별 문자의 의미도 함께 설명하시오.
7. 심플 피처 폴리곤의 집합을 가지고 커버리지 하나를 만들 수 있을지의 여부에 대해 답하시오. 만약 답이 그렇다이면 어떤 제약조건 하에서 그것이 가능한지 설명하시오.
8. **sf** 패키지에 들어 있는 nc 카운티 데이터를 가지고, 네 개의 카운티가 동시에 접촉하고 있는 포인트들을 추출하시오.
9. -축을 위한 값이 양수라면, 그림 3.6은 어떻게 달라질지 설명하시오.

# 4. 구체 지오메트리

“국지적 데카르트 투영 좌표계로 세팅이 되어 있는 GIS 소프트웨어를 글로벌 스케일의 애플리케이션을 위해 확장하는 순간, 오류투성이의 결론이 도출되고 어이없는 측정이 이루어지기 시작한다.” (Chrisman 2012)

3장에서는 평면인 에에서 정의되는 지오메트리에 대해 논의했다. 이 장에서는 평면이 아니라 3차원 구체() 상의 지오메트리를 다룰 때 어떠한 변화가 발생하는지에 대해 논의할 것이다.

우리는 제2장에서 지구의 형태가 일반적으로 구체가 아니라 타원체로 근사된다는 것을 배웠지만, 그림 1.7에서 녹색으로 표시된 라이브러리 중 어느 것도 타원체를 전제한 포괄적인 계산 함수의 세트에 접근할 수 없다. 오직 s2geometry (Dunnington, Pebesma, and Rubak 2023; Veach et al. 2020) 라이브러리만이 이러한 접근을 가능케 하는데, 타원체 대신 구체를 사용한다. 그러나 이전 장에서 사용한 평면(투영된) 공간과 비교할 때, 구는 타원체에 대한 *훨씬* 더 나은 근사값이다.

## 4.1 직선

3장에서 다룬 심플 피처의 기본 전제는 지오메트리가 *직선으로 연결된* 포인트들의 시퀀스로 표현된다는 것이다. (또는 데카르트 공간)에서는 이는 자명하지만, 구체 상에서는 직선이 존재하지 않는다. 두 지점을 연결하는 가장 짧은 선은 두 지점을 지나는 원의 호로, 이를 *대권호*(grate circle segment)라고도 한다. 이로 인해, 구의 반대편에 있는 두 점을 연결하는 *가장* 짧은 거리의 선은 존재하지 않으며, 이들을 연결하는 모든 대원호가 같은 길이를 갖는다. GeoJSON 표준(Butler et al. 2016)은 측지 좌표에서 직선이 무엇인지에 대한 독자적인 정의를 제시하고 있다(이 장의 끝에 있는 연습문제 1 참조).

## 4.2 링 디렉션과 완전 폴리곤

구체 상에 존재하는 폴리곤은 구 표면을 유한한 면적을 가진 두 부분, 즉 내부와 외부로 나눈다. 의 경우에 적용된 “반시계 방향 규칙”은 잘 작동하지 않는데, 이는 방향 해석이 무엇을 내부로 정의할 것이냐에 따라 달라지기 때문이다. 통상적인 방식은, 폴리곤의 점을 순서대로 순회할 때 다각형 경계의 왼쪽(또는 오른쪽)을 내부로 정의하는 것이다. 노드 순서를 반대로 하면 내부와 외부가 뒤바뀐다.

3장에서 빈 폴리곤에 대해 배웠는데, 정반대로 지표면 전체를 포괄하는 *완전 폴리곤* 개념을 생각해 볼 수 있다. 이 개념은, 예를 들어, 완전 폴리곤과 육지부의 합집합의 기하학적 차이를 통해 해양부를 정의할 때, 유용할 수 있다(그림 8.1과 그림 11.6 참고).

## 4.3 바운딩 박스, 바운딩 직사각형, 바운딩 캡

에서는 와 좌표의 범위로 바운딩 박스를 쉽게 정의할 수 있지만, 타원체 좌표의 경우 지오메트리가 반대자오선(antimeridian)(경도 )(역자주: 반대자오선은 본초자오선의 반대편에 위치한 자오선을 의미하는 것으로 동경 180도 혹은 서경 180도 경선을 지칭한다.) 이나 극점을 가로지를 때 이러한 범위는 그다지 유용하지 않다.

낮은 값이 높은 값의 서쪽에 있다는 에서의 가정은 반대 자오선을 넘을 때 성립하지 않는다. 구체 상의 영역을 획정하는 보다 자연스러운 대안은 바운딩 캡(bounding cap)인데, 영역의 중심 좌표와 반지름으로 만으로 정의할 수 있다. 남극대륙의 경우, 그림 4.1의 (a)와 (c)에 나타나 있는 것처럼, 바운딩 박스는 다음의 좌표 범위를 통해 정의된다.

# xmin ymin xmax ymax   
# -180.0 -85.2 179.6 -60.5

이 바운딩 박스는 명백히 ymin이 -90이고 xmax가 180인 지역을 포함하지 않는다. 바운딩 캡은 해당 지역을 포함한다.

# lng lat angle  
# 1 0 -90 29.5

해당 지역을 포함하는 또 다른 지오메트리에 바운딩 *직사각형*(rectangle)이 있다.

# lng\_lo lat\_lo lng\_hi lat\_hi  
# 1 -180 -90 180 -60.5

반대자오선을 가로지르는 지역의 예로 피지 제도를 들 수 있는데, 피지 제도의 바운딩 박스는 다음과 같이 주어진다.

# xmin ymin xmax ymax   
# -179.9 -21.7 180.2 -12.5

지구를 한바퀴 도는 정도의 크기이다. 동일한 지역에 대한 바운딩 직사각형은 다음과 같이 주어진다.

# lng\_lo lat\_lo lng\_hi lat\_hi  
# 1 175 -21.7 -178 -12.5

여기서 lng\_lo가 lng\_hi 보다 더 *큰* 값을 갖는데, 이는 바운딩 직사각형이 반대자오선을 가로지른다는 것을 나타낸다. 좌표 범위를 가지고 이러한 위치 관계를 추로하는 것은 불가능하다.

## 4.4 구체 상의 유효성 개념

많은 글로벌 데이터셋은 타원체 좌표로 제공되지만 [-180,180][-90,90]의 에서도 잘 작동하도록 사전 조치되어 있다. 이것은 다음과 같은 사항을 의미한다.

* 반대자오선(경도 )을 가로지르는 지오메트리는 양쪽으로 분할되어 횡단은 발생하지 않는다(하지만 둘은 거의 서로 닿아 있다).
* 극점을 포함하는 지오메트리, 예를 들어 남극은 를 기준으로 분할되어 -180,-90과 180,-90가 다른 것으로 취급되지만 실질적으로는 둘 다 지리적 남극(Geographic South Pole)을 나타낸다.

그림 4.1은 남극의 두 가지 다른 표현을 보여준다. 위쪽은 을 상정한 타원체 좌표로 나타낸 것이고, 아래쪽은 극평사도법(Polar Stereographic projection)으로 나타낸 것이다. 왼쪽은 지리적 남극을 중심으로한 분할이 없는 경우이고, 오른쪽은 지리적 남극을 중심으로한 분할이 있는 경우이다.

폴리곤 (b)와 (c)는 유효하지만, 폴리곤 (a)는 자기-교차하기 때문에 유효하지 않고, 폴리곤 (d)는 남극을 향한 같은 변을 두 번 지나기 때문에 유효하지 않다. 구()에서는 폴리곤 (a)는 유효하지만, (b)는 (d)와 같은 이유로 유효하지 않다.

|  |
| --- |
| 그림 4.1: 남극 폴리곤으로 (a)와 (c)는 POINT(-180 90)을 통과하지 않고, (b)와 (d)는 POINT(-180 -90)과 POINT(180 -90)을 통과한다. |

## 4.5 연습문제

다음의 연습문제를 풀되, 적절한 곳에서 R을 활용하라.

1. GeoJSON 형식(Butler et al. 2016)은 타원체 좌표 간의 “직선”을 어떻게 정의하는가(섹션 3.1.1)? 이 직선 정의를 사용했을 때, LINESTRING(0 85, 180 85)는 극투영법에서 어떻게 나타나는가? 북극을 통과하게 하려면 이 지오메트리를 어떻게 수정해야 할까?
2. 의 전형적인 폴리곤의 링 디렉션을 어떻게 확인할 수 있을까?
3. 바운딩 박스 대신 바운딩 캡을 사용하는 것의 이점이 있는가? 이점이 있다면 열거해 보시오.
4. 왜 작은 지역에 대해 해당 지역을 중심으로 한 정사도법이 에서 다루어진 지오메트리의 좋은 근사가 되는가?
5. rnaturalearth::ne\_countries(country = "Fiji", returnclass = "sf")를 사용하여 피지의 기하학이 에서 유효한지, 피지를 중심에 둔 정사도법에서 유효한지, 에서 유효한지 확인하시오. 에서 지오메트리를 유효하게 하기 위해서는 어떻게 해야 하는가? 결과 지오메트리를 로 되돌려 지도로 그려보시오. 피지의 센트로이드를 와 각각에 대해 구하고 두 센트로이드 간의 거리를 구하시오.
6. **giscoR** 패키지의 gisco\_countries 데이터셋에서 NAME\_ENGL == "Fiji"인 국가를 선택하시오. 이 국가의 지오메트리가 구체 상에서 유효한가? 그렇다면, 어떻게 그렇게 되었는지 설명하시오.

# 5. 속성과 서포트

피처 *속성*은 피처(“사물”)의 특성을 의미한다. 피처의 지오메트리 특성 그 자체는 속성으로 간주되지 않지만 어떤 속성은 지오메트리로부터 *도출*된다(예를 들어, LINESTRING의 길이나 POLYGON의 면적과 같은 속성). 그러나 많은 속성은 다음과 같은 비도출 속성이다.

* 스트리트나 카운티의 이름
* 카운티의 인구수
* 도로의 유형
* 토양 유형
* 가게의 영업 개시 시간
* 동물의 체중 혹은 심박수
* 대기질 측정소의 이산화질소 농도

어떤 경우에는 시간적 특성이 피처의 속성으로 간주될 수 있으며, 예를 들어 사람의 출생일이나 도로의 건설 연도가 이에 해당한다. 대기질과 같은 속성이 공간과 시간의 함수일 때, 시간은 기하학적 특성과 동등한 중요성을 가지는 방식으로 다루는 것이 가장 적합하다(6장에서 데이터 큐브에서와 관련하여 이러한 사항이 다루어진다).

심플 피처를 구현하는 공간데이터사이언스 소프트웨어는 일반적으로 피처의 기하학적 특성과 속성을 모두 포함하는 테이블 형태로 데이터를 구성한다. 이는 파이썬의 Geopandas, PotsgreSQL의 PostGIS 테이블, R의 sf 객체 모두에 해당된다. 3.2절에서 설명된 기하학적 오퍼레이션은 *오직* 지오메트리에만 적용되며, 경우에 따라 새로운 속성(프리디케이트, 측정, 변환)을 생성할 수 있지만, 기존 속성에는 적용되지 않는다.

지오메트리의 수정이 있음에도 속성 *값*이 그대로 유지되는 경우에는 서포트 문제가 발생할 수 있다. 예를 들어 카운티 폴리곤을 해당 폴리곤의 센트로이드로 대체하는 간단한 사례를 살펴보면, R 패키지 **sf**는 다음과 같은 경고 메시지를 출력한다.

Warning: st\_centroid assumes attributes are constant over geometries

이러한 경고 메시지가 표출되는 이유는 데이터셋에 전체 폴리곤과 연관된 값(역자주: 폴리곤 내부의 모든 지점이 해당 값을 가진다고 볼 수 없고, 폴리곤 전체의 합산값 혹은 총빈도의 의미를 갖는 값)들을 가진 변수들이 포함되어 있기 때문이다. 이 경우, 인구수와 같은 값들은 폴리곤 전체에 관련된 것이므로, 폴리곤을 대체하는 POINT 지오메트리와는 연관되지 않는다.

1.6절에서 이미 언급했듯이, 라인 및 폴리곤 지오메트리에서는 피처 속성 값이 *포인트 스포트* 또는 *블록 스포트*를 가질 수 있다. 포인트 스포트는 해당 값이 지오메트리의 *어떤 지점에도*(*모든 지점*에) 적용되는 경우이고, 블록 서포트는 지오메트리 내의 *모든 지점을 요약*하는 경우를 의미한다(해당 지오메트리보다 더 작거나 더 큰 서포트 역시 가능한다).

이 장에서는 속성이 지오메트리와 관련되는 방식과 서로 다른 방식이 데이터 분석에 미치는 영향, 그리고 서로 다른 지오메트리로부터 속성을 도출하는 방법(상향 스케일링과 하향 스케일링)에 대해 설명할 것이다.

## 5.1 속성-지오메트리 관계와 서포트

피처 속성은 변경하지 않고 피처의 기하 특성만 변경해도 *피처*는 변화한 것이다. 왜냐하면 피처는 지오메트리와 속성의 조합으로 이루어지기 때문이다. 예를 들어, 해당 지오메트리를 컨벡스 헐이나 센트로이로 대체할 때, 산출되는 피처가 여전히 속성 값과 의미 있게 관계를 맺을 것인지를 미리 예측할 수 있을까? 이건 그때마다 다르다.

LINESTRING의 지오메트리를 갖는 도로망을 예를 들어보자. 데이터셋에 도로폭이라는 속성이 포함되어 있고, 어떤 도로의 속성값이 10m라고 하자. 이 때 해당 도로의 특정 부분의 도로폭에 대해 우리는 뭐라고 얘기할 수 있을까? 이는 속성인 도로폭이 도로의 모든 지점의 도로폭을 의미하는지, 즉 도로폭이 일정하다는 것을 의미하는지, 아니면 최소 또는 평균 도로폭과 같은 집합적 속성을 의미하는지에 따라 달라진다. 최소의 경우를 좀 더 따져보자. 최소 도로폭은 해당 도로에서 임의로 특정 구간을 선택했을 때 그것의 최소 도로폭이 주어진 최소 도로폭보다 작지 않을 것이라는 점을 함축하고 있지만, 그 하위 구간의 최소 도로폭이 아닐 수도 있다. 이는 두 가지 *유형*의 속성-지오메트리 관계(AGR)가 있음을 함축한다.

* 고정(constant) AGR: 속성값이 지오메트리의 모든 부분에 적용된다. 피처가 무수히 많은 지점으로 구성되어 있고, 모든 지점이 동일한 속성값을 가지는 경우라고 생각하면 된다. 지구통계학적 용어로는 이것을 *포인트 서포트*를 보유한 변수라고 부른다.
* 집계(aggregate) AGR: 속성값이 지오메트리의 전체에 대한 요약값이다. 피처가 하나의 관측값을 갖고 그 값은 지오메트리 전체에 적용되는 경우라고 생각하면 된다. 이러한 변수를 *블록 서포트*를 보유한 변수라고 부른다.

폴리곤 데이터의 경우, 고정 AGR(포인트 서포트)의 예로 다음을 변수를 들 수 있다.

* 토지이용도에서의 토지이용
* 지질도에서의 암석 단위 또는 지질층
* 토양도에서의 토양 유형
* 기복도에서의 고도 클래스
* 기후구분도에서의 기후 지역

이러한 변수의 전형적인 특성은 지오메트리가 인위적으로 생성된 것이 아니며, 센서 장치(예: 원격탐사의 이미지 픽셀 경계)와도 관련이 없다는 것이다. 대신, 지오메트리는 관찰된 변수를 매핑함으로써 결정된다. 집계 AGR의 예로는 다음과 같은 변수를 들 수 있다.

* 인구: 인구수 혹은 인구밀도
* 지역별로 요약된 사회경제적 변수
* 원격탐사 픽셀별 평균 반사율
* 지역별 총 오염물질 배출량
* 이산화질소 농도에 대한 블록 평균: 보통 정사각형 블록에 대한 블로 크리깅(12.5절) 또는 구역 평균값을 예측하는 분산 모델을 통해 획득된다.

이러한 변수의 전형적인 특성은 지오메트리가 법률적 규정, 관측 장치 또는 분석 선택과 같은 것으로부터 비롯된 것으로, 관찰 변수 그 자체와 본질적으로 연결되어 있지는 않다는 점이다.

세 번째 유형의 AGR는 속성이 피처 기하의 식별자 역할을 하는 경우에 발생한다. 개별 지오메트리가 변수의 특정 값과 고유한 관련성을 맺고 있을 때, 우리는 이 속성을 식별 변수(identity variable)라고 부른다(같은 값을 가진 다른 지오메트리가 없다). 카운티 이름의 예를 들어보자. 이름이 해당 카운티를 식별하고, 카운티 내의 어떤 하위-지역에 대해서도 동일 이름이 적용될 수 있다(포인트 서포트). 그러나 임의의 하위-구역을 염두에 두면 해당 변수는 더 이상 식별자로서의 역할을 할 수 없는 고정 속성값으로 변모한다. 하나의 예를 들면 다음과 같다.

* 카운티 내부의 임의 점(또는 지역)은 여전히 그 카운티의 일부이며 카운티 이름이라는 변수에 대해 동일한 변수값을 가져야 한다. 하지만 해당 지점(또는 지역)은 더 이상 해당 카운티의 (총체적인) 지오메트리에 대한 식별자 구실을 할 수 없다.

여기서 중요한 것은 공간정보(단순화를 위해 시간을 무시할 경우)가 서로 다른 현상 유형으로 나뉘어질 수 있다는 점이다(Scheider et al. 2016).

* 필드: *연속적* 공간 상의 모든 지점이 특정한 속성 값을 갖는 경우(예를 들어, 고도, 대기질 또는 토지이용)
* 객체: 위치의 *이산적* 집합으로 규정되는 경우(예를 들어, 주택, 나무 또는 사람)
* 집계값: 필드별 총합 혹은 평균, 라인 혹은 폴리곤 객체의 총빈도 혹은 밀도로 계산되는 값

그런데 이러하 현상 유형이 지오메트리 유형(포인트, 라인, 폴리곤, 래스터 셀)과 일대일 대응의 관계에 있는 것은 아니다.

* 포인트는 필드 상의 표집 위치일 수도 있고(대기질) 객체의 위치일 수도 있다.
* 라인은 객체(도로와 강)일수도 있고, 필드를 위한 등치선일 수 있고, 행정구역의 경계일 수도 있다.
* 래스터 픽셀과 폴리곤은 토지이용(*커버리지*)과 같은 범주형 필드와 연관될 수도 있고, 인구밀도와 같은 집계값과도 관련될 수 있다.
* 래스터 혹은 다른 종류의 메시 삼각망은 노드(포인트), 엣지(라인), 페이스(에어리어, 셀)와 관련된 서로 다른 변수를 가질 수 있다. 스태거드(staggered) 그리드를 이용해 편미분 방정식을 근사하는 상황(Haltiner and Williams 1980; Collins et al. 2013)이 이러한 예가 될 수 있다.

속성-지오메트리 관계를 적절히 정의하고, 속성-지오메트리 관계에 대한 정보가 부존하거나 지오메트리의 변화(서포트의 변화)가 정보의 변화를 야기하는 경우에 대해 경고 메시지를 표출하는 것은 공간데이터의 *서포트*와 관련된 일반적인 공간데이터 분석 오류(Stasch et al. 2014)를 피하는 데 도움이 될 수 있다.

## 5.2 애그리게이션과 서머라이즈

테이블 레코드의 합형은 다음의 두 간계를 통해 이루어진다.

* 그룹 프리디케이트에 기반해 레코드를 분류하기
* 합형 함수를 적용해 그룹별로 단일한 요약 속성값을 계산하기

SQL에서 합형 과정은 다음의 예시처럼 이루어진다.

SELECT GroupID, SUM(population) FROM table GROUP BY GroupID;

여기서 합형 함수는 SUM이고 그룹 프리디케이트는 GroupID이다.

R 패키지인 dplyr은 이를 수행하기 위해 두 단계를 사용한다. 함수 group\_by는 레코드의 그룹 멤버십을 지정하고, summarise는 각 그룹에 대한 데이터 요약(예: 합계 또는 평균)을 계산한다. 베이스 R의 aggregate 함수는 테이블, 그룹화 조건, 및 집계 함수를 아규먼트로 받아 두 가지 작업을 단일 함수로 해결한다.

노스캐롤라이나 카운티의 예는 그림 5.1에 나타나 있다. 타원 좌표 POINT(-79, 35.5)를 기준으로 사분면을 설정하고 카운티의 센트로이드가 어느 사분면에 위치하느냐에 따라 카운티를 그룹화하고, 각 그룹별로 질병 사례 수를 합산했다. 그 결과, 그룹별로 통합된 지오메트리가 생성되었음을 알 수 있다(3.2.6절 참조). 이러한 그룹별 합형은 필수적인데, 만일 카운티 지오메트리를 단순히 한데 묶어 MULTIPOLYGON을 생성했다면 수 많은 중복 경계가 생겨나 유효하지 않은 지오메트리가 생성되었을 것이기 때문이다(3.1.2절 참조).

|  |
| --- |
| 그림 5.1: SID74가 네 개의 지역별로 합산되었다. |

결합된 카운티 폴리곤을 지도로 나타내는 것은 기술적으로 아무런 문제가 없지만, 그룹 합계가 그룹화된 카운티에 관련된 것이 아니라 개별 카운티와 관련된 것이라는 잘못된 암시를 줄 수 있다. 이러한 방식의 합형의 특징은 각 레코드가 하나의 그룹에만 할당된다는 점이다. 이렇게 하면 그룹별 합계의 총합이 그룹화 이전 데이터의 총합과 동일하다는 장점이 있다. 즉, 정량적인 변수의 경우, 정보가 손실되거나 추가되는 일이 없다. 새로 형성된 기오메트리는 원 레코드의 지오메트리를 그룹에 의거해 유니온한 결과이다.

|  |
| --- |
| 그림 5.2: 노스케롤라이나 카운티 상의 타깃 블 |

지오메트리의 결합 없이 그룹별 합산값을 산출할 필요가 있을 수 있다. 이 경우, 우리는 공간적 프리디케이트를 사용하는데, 하나의 레코드가 여러개의 그룹과 관련될 수 있다. 그림 5.2의 직사각형을 타깃 에어리어로 하여 직사각형별로 교차하는 카운티의 유병자를 합산하게 되면 전체 합계는 훨씬 더 큰 값을 갖게 될 것이다.

# sid74\_sum\_counties sid74\_sum\_rectangles   
# 667 2621

반대로 contains 또는 covers와 같은 다른 프리디케이트를 사용하면 훨씬 더 적은 값이 산출된다. 왜냐하면 많은 카운티가 직사각형 속에 완전히 포함되지 않기 때문이다. 그러나 이러한 결과가 좋을 수 있는 상황도 충분히 존재할 수 있다.

* POINT 지오메트리를 폴리곤에 의거해 합형하고자 하는 경우. 이 때 모든 포인트는 해당 포인트를 완전 포함하는 폴리곤에 할당된다. 만일 경계 상에 위치하게 되면 양쪽 폴리곤 모두에 할당된다(DE-9IM-기반 GEOG 라이브러리가 이러한 방식을 취한다. s2geometry 라이브러리는 “반-개방” 폴리곤을 지정할 수 있게 해주는데, 폴리곤의 중첩만 없다면, 반드시 하나의 폴리곤에 포인트를 할당해준다.
* 매우 작은 폴리곤이나 래스터 픽셀을 더 큰 에어리어로 합역하는 경우. 예를 들어 노스케롤라이나에 대한 30m 해상도의 고도 데이터를 카운티별로 평균내는 경우가 해당될 수 있는데 다중 대응으로 인한 데이터 오류는 무시할 만하다.
* 다대다 대응에 최대면적대응이 적용된 경우(그림 7.4 참조)

보다 작은 에어리어를 보다 큰 에어리어로 합역하는 보다 포괄적인 접근 방법은 면적-가중 인터폴레이션(area-weighted interpolation)을 적용하는 것이다.

## 5.3 면적-가중 인터폴레이션

두 개의 데이터셋의 지오메트리와 속성을 결합함으로써 소스 데이터셋의 속성 값을 타겟 데이터셋의 지오메트리에 의거한 요약값으로 전환하고자 한다면, 면적-가중 인터폴레이션(area-weighted interpolation)이 가장 간단한 접근 방법이 될 수 있다. 이 방법은 소스와 타겟의 지오메트리가 겹치는 면적을 고려하여 소스 속성 값을 타겟 속성 값으로 전환하기 위한 가중치로 사용한다(Goodchild and Lam 1980; Do, Thomas-Agnan, and Vanhems 2015a, 2015b; Do, Laurent, and Vanhems 2021). 이 기법은 보수적 합역(conservative region aggregation) 또는 재그리드(regridding)로도 알려져 있다(Jones 1999). 여기서는 Do, Thomas-Agnan, and Vanhems(2015b)의 표기법을 따르도록 한다.

면적-가중 인터폴레이션은 타깃 에어리어 에 대한 가중평균값을 산출해 주는데, 해당 타깃 에어리어와 결부되어 있는 개의 타깃 에어리어 의 속성값 에 대한 가중평균값이다.

여기서 는 와 의 겹침의 정도에 따라 달라지는 데, 겹침의 정도는 로 표현된다. 와 의 관계는 아래에서 자세히 설명하고자 한다.

가중치를 계산하는 다양한 옵션이 있으며, 외부 변수를 활용하는 방법(대시메트릭 매핑은 Mennis 2003 참조)도 그 중 하나이다. 외부 변수를 사용하지 않고 가중치를 계산하는 두 가지 단순한 방식이 존재하는데, 변수 가 공간적으로 *외연적인*(extensive) 변수인지 *내포적인*(intensive) 변수인지에 따라 달라진다.

### 5.3.1 공간적으로 외연적인 변수와 공간적으로 내포적인 변수

공간적으로 외연적인 변수는 길이, 면적, 부피, 또는 카운트와 같은 물리적 크기와 관련된 양을 나타낸다. 외연적인 변수의 예로 인구 수를 들 수 있다. 인구 수는 특정 크기의 영역과 관련된 값이며, 그 영역이 더 작은 영역으로 분할되면 인구 수 또한 분할되어야 한다. 인구가 공간적으로 균일하지 분포하지 않은 경우가 많기 때문에, 이 분할이 반드시 면적에 비례하여 이루어질 필요는 없지만, 더 작은 영역의 인구 수의 합계는 전체 영역의 인구 수와 일치해야 한다.

공간적으로 내포적인 변수는 영역의 면적에 비례적으로 작동하지 않는 변수이다. 즉, 영역이 분할되더라도 평균적인 의미에서는 값이 그대로 유지된다. 내포적인 변수의 예로 인구밀도를 들 수 있다. 영역이 더 작은 영역으로 분할되더라도 연구밀도 값이 면적에 비례하여 할당되지는 않는다. 더 작은 영역의 인구밀도의 합산값은 아무런 의미가 없으며, 작은 영역의 인구밀도의 평균의 전체 영역의 인구밀도와 유사할 것이다.

외연적인 변수 가 공간상에 균등하게 분포한다고 했을 때, 소스 데이터의 지역 의 변수값 로부터 타깃 지역과의 겹침을 통해 생성된 구역()의 속성값 를 다음의 공식에 의거해 구할 수 있다.

여기서 는 면적을 의미한다. 를 추청하려면, 와 겹쳐서 생성되는 모든 하위 영역의 값을 합산해야 한다.

내포적인 변수의 경우, 해당 변수값이 개별 소스 구역 내에서 일정한 값을 나타낸다고 가정하기 때문에, 겹침에 의해 생성된 하위 구역의 추정값은 해당 소스 구역의 전체 값과 동일하다.

따라서 소스 구역의 값에 대해 면적-가중 평균을 구하면 타깃 구역 에 대한 추정값을 구할 수 있다.

### 5.3.2 대시메트릭 매핑

다시메트릭 매핑은 보다 큰 크기의 구역 체계의 변수값(예: 인구 수)을 보다 크기가 작은 구역 체계의 변수값으로 분배하는 방법이다. 이때 인구 분포와 연관된 다른 변수들, 예를 들어 토지 이용, 건물 밀도, 도로 밀도 등을 활용한다. 대시매트릭 매핑을 위한 가장 간단한 방식은 식 5.2에서 나타나 있는 비 대신 또 다른 외연적인 변수와 관련된 비 를 사용하는 것이다. 이 방법을 사용하기 위해서는 소스 구역과 겹침 구역 모두에 대해 변수 값을 알고 있어야 한다.

### 5.3.3 파일 포맷에서의 서포트

GDAL의 벡터 API는 필드 도메인(field domains)이라고 불리는 것을 읽고 쓰는 기능을 지원하며, 이는 지오메트리가 분할되거나 결합될 때 속성 변수에 대해 어떤 처리를 해야 할지 나타내는 “분할 정책(split policy)”과 “병합 정책(merge policy)”을 가질 수 있다. 이러한 값은 공간적으로 내표적인 변수의 경우, 분할 시 “중복(duplicate)”이, 병합 시 “기하 가중(geometry weighted)”가 될 수 있으며, 공간적으로 외연적인 변수의 경우, 분할 시 “기하 비율(geometry ratio)”이, 병합 시 “합(sum)”이 될 수 있다. 이 기능을 지원하는 파일 형식으로는 GeoPackage와 FileGDB가 있다.

## 5.4 상향 스케일링과 하향 스케일링

상향 및 하향 스케일링(upscaling and downscaling)은 일반적으로 저해상도 데이터에서 고해상도 정보를 얻는 과정(하향 스케일링) 또는 고해상도 데이터에서 저해상도 정보를 얻는 과정(상향 스케일링)을 의미한다. 이 두 과정은 속성과 지오메트리의 관계에서의 변화, 즉 서포트에서의 변화를 야기한다. 상향 스케일링은 집계(aggregation), 하향 스케일링은 분해(disaggregation)와 동의어로 사용할 수 있다. 하향 스케일링의 가장 단순한 형태는 폴리곤, 라인, 또는 격자 셀의 값을 주어진 포인트에 대해 샘플링(또는 추출)하는 것이다. 이는 포인트-서포트를 가진 변수(“고정” AGR)에는 적합하지만, 변수 값이 합산값일 경우에는 대략적인 결과만을 제공할 수 있다. 하향 스케일링의 도전 과제로는 저해상도 기상 예측 모델 또는 기후 변화 모델을 통해 얻은 변수를 고해상도로 예측하는 것, 그리고 다른 시공간적 해상도를 가진 센서의 융합을 기반으로 위성 이미지에서 파생된 변수를 고해상도로 예측하는 작업이 포함된다.

식 5.1과 이에 기반한 식 5.2(공간적으로 외연적인 변수)와 식5.3(공간적으로 내포적인 변수)는 소스 구역 와 타깃 구역 간에 겹침이 존재하기만 하면 두 구역 사이에서 정보를 이동시키는 것이 가능하다는 점을 보여준다. 이는 임의로 더 큰 구역 단위로 이동(집계)하거나 더 작은 구역 단위로 이동(분해)할 수 있음을 의미한다. 물론 이러한 방식의 유의성은 다음의 가정이 성립하는 정도에 의존적이다. 즉, 소스 구역에서 외연적인 변수는 균등 분포해야 하고, 내포적인 변수는 일정한 값을 가져야 한다.

디스애그리게이션은 라인이나 폴리곤 데이터로부터 포인트 값을 추출하는 것에서 시작된다. 포인트는 면적을 갖지 않기 때문에() 식 5.2와 5.3은 적용할 수 없다. 그러나 지오메트리 내부의 값이 일정하다는 가정을 할 수 있다면, 내포적인 변수인 경우에는 이 포인트에 할당될 수 있다. 단, 모든 지점이 고유한 방식으로 단일한 소스 지점 에 할당될 수 있어야 한다. 폴리곤 데이터를 예를 들자면, 가 커버리지 변수(3.4절)여야만 한다는 것을 함축한다. 외연적인 변수의 경우, 포인트에 대한 변수값을 추출한다는 것이 무의미하다. 항상 0이 추출될 것이기 때문이다.

영역과 관련된 값이 해당 영역의 집계 값인 경우, 면적-가중 인터폴레이션이나 대시매트릭 매핑에서 가정하는 균일 분포 또는 일정한 값이라는 가정은 매우 비현실적이다. 이러한 경우에라 하더라도 이러한 단순한 접근법이 합리적인 근사치를 제공할 수 있으며, 예를 들어 다음과 같은 상황에서 가능하다.

* 소스 구역과 타깃 구역이 거의 동일하다.
* 소스 구역 내의 변동성이 매우 작아서, 변수값이 거의 균등 분포하거나 일정한 값을 보인다.

다른 경우에는 이러한 방법을 사용하여 얻은 결과는 그저 타당성이 결여된 가정의 결과일 뿐이다. 점 또는 더 작은 구역으로부터 더 큰 구역에 대한 퀀티티를 추정할 수 있는 통계적 애그리게이션 방법에는 다음이 포함된다.

* 디자인-기반(design-based) 방법: 타깃 지역에서 확률 샘플이 이용 가능하고, 포함 확률(inclusion probability)이 알려져 있어야 한다(Brus 2021, 10.4 절).
* 모델-기반(model-based) 방법: 공간적 자기상관을 수용하는 랜덤 필드 모델을 가정한다(블록 크리깅, 12.5절).

다른 디스애그리게이션 기법에 다음과 같은 것들이 있다.

* 결정론적, 평활화-기반 접근: 커널-기반 또는 스플라인-기반의 평활화 기법 포함
* 통계적, 모델-기반 접근: 에어리어-투-에어리어 크리깅과 에어리어-투-포인트 크리깅

## 5.5 연습문제

다음의 연습문제를 풀되, 적절한 곳에서 R을 활용하라.

1. 노스케롤라이나 데이터셋(nc)에 nc$State = "North Carolina"와 같은 방식으로 변수를 첨가한다고 할 때(모든 카운티에 동일한 주 이름이 할당), 속성-지오메트리 관계(AGR)을 위해 이 변수에 어떤 값을 첨가할 수 있을까?
2. st\_union(nc)으로 얻은 지오메트리로부터 새로운 sf 객체를 생성하고, State라는 변수에 "North Carolina"라는 값을 할당하라. 이 변수에 어떤 대한 속성-지오메트리 관계(AGR)을 속성 변수에 할당할 수 있을까?
3. nc 데이터셋에 st\_area를 사용하여 이름이 area인 변수를 추가하시오. area 변수와 AREA 변수를 비교하시오. AREA의 단위는 무엇인가? 두 변수는 선형적인 관련성이 있는가? 불일치가 존재한다면, 그 원인은 무엇일까?
4. area는 내포적인 변수인가 외연적인 변수인가? area의 AGR은 constant인가, identity인가, 아니면 aggregate인가?
5. 그림 5.3에서 5.3.1 절에 나타나 있는 방정식을 이용하여 (a) 점선 셀과 (b)네 개의 실선 셀을 모두 포함하는 정사각형에 대해 면적-가중 인터폴레이션의 결과를 구하시오. 후자에 대해서는 (i) 외연적인 변수인 경우와 (ii) 내포적인 변수인 경우로 나누어 계산하시오. 빨간 숫자는 소스 구역의 데이터 값이다.

|  |
| --- |
| 그림 5.3: 면적-가중 인터폴레이션의 예제 데이터 |

# 6. 데이터 큐브

데이터 큐브는 시간에 걸쳐 일련의 지오메트리의 특성을 반복적으로 관찰할 때 자연스럽게 발생한다. 시간 정보는 때때로 건물의 건축 연도나 사람의 생년월일을 등록할 때와 같이 피처의 속성으로 간주될 수 있다(5장). 다른 경우에는 속성을 관찰하는 시간이나 속성에 대한 예측이 이루어진 시점을 나타낼 수 있다. 이러한 경우, 시간은 공간과 동등한 위상을 가지며, 시간과 공간은 함께 우리가 관찰, 모델링, 예측 또는 예보하는 물리적 차원을 설정한다.

우리 세상을 바라보는 한 가지 방법은 세 가지 공간 차원과 하나의 시간 차원을 포함하는 네 차원 공간을 상정하는 것이다. 그런 관점에서 사건은 시간 차원에서의 크기를 지속 시간으로 갖는 “사물”이나 “객체”가 된다(Galton 2004). 이러한 관점은 우리가 세상을 경험하고 묘사하는 방식과 잘 맞지 않지만, 데이터 분석의 관점에서는 네 개의 숫자(와 참조 체계) 만으로 관측의 시공간 좌표를 설명하는 데 충분하다.

우리는 데이터 큐브를 공간 및/또는 시간과 관련된 하나 이상의 배열 차원을 가진 배열 데이터로 정의한다(Lu, Appel, and Pebesma 2018). 이는 래스터 데이터, 속성을 가진 피처, 그리고 시계열 데이터가 모두 데이터 큐브의 특별한 경우임을 의미한다. 우리는 3차원 구조에 한정될 이유가 없으므로 실제로는 큐브보다는 *하이퍼큐브*(hypercube)를 의미하며, 서로 다른 차원의 큐브 범위가 동일할 필요는 없고 비교 가능한 단위를 가질 필요도 없기 때문에 더 적절한 용어는 *하이퍼-직사각형*(hyper-retangle)이 될 것이다. 간편하게 우리는 데이터 큐브라고 부른다.

데이터 큐브의 표준 형식은 그림 6.1에 나타나 있으며, 이는 서로 다른 시간 단계에서 수집(관찰 또는 모델링)된 동일한 지역의 래스터 레이어 세트를 원근법 플롯으로 보여준다. 세 개의 큐브 차원인 경도, 위도, 시간은 서로 직교 관계를 이룬다. 임의의 2차원 큐브 슬라이스는 특정 값에서 하나의 차원을 고정함으로써 얻어지고, 1차원 슬라이스는 특정 값에서 두 개의 차원을 고정하여 얻어지며, 스칼라는 세 개의 차원을 특정 값에서 고정함으로써 얻어진다.

|  |
| --- |
| 그림 6.1: 위도, 경도, 시간의 세 차원을 가지 래스터 데이터 큐브 |

## 6.1 4차원 데이터 큐브

|  |
| --- |
| 그림 6.2: x, y, 밴드, 시간의 네 차원을 가진 4차원 래스터 데이터 큐브 |

그림 6.2는 4차원 래스터 데이터 큐브를 보여준다(Appel and Pebesma 2019). 여기서 스펙트럼 차원(“밴드”)을 가진 3차원 래스터 데이터 큐브가 네 번째 차원인 시간 축을 따라 조직된다. 컬러 이미지 데이터는 항상 세 개의 밴드(파란색, 녹색, 빨간색)를 가지며, 이 예제는 스펙트럼 원격탐사 데이터에서 일반적으로 발견되는 네 번째 밴드(근적외선, NIR)를 포함하고 있다.

그림 6.3은 정확히 동일한 데이터를 보여주지만, 평면적으로 배열한 패싯 플롯(또는 산점도 행렬)으로 나타내고 있다. 플롯의 두 차원( 및 )은 기본적으로는 *밴드*와 *시간* 차원을 의미하지만, 세부 플롯 내부에서는 실제 좌표계를 의미한다.

|  |
| --- |
| 그림 6.3: 두 차원에 의거해 평면적으로 배열된 4차원 래스터 데이터 |

## 6.2 차원, 속성, 서포트

시공간 상의 현상은 정의역이 공간과 시간이며, 치역이 하나 이상의 관찰된 속성인 함수로 생각할 수 있다. 명확하게 식별 가능한 이산적인 사건이나 객체의 경우, 치역은 일반적으로 이산적이며, 이산성은 시작과 끝의 정밀한 좌표를 설정하는 것으로 부터 확보될 수 있고, 이는 벡터 지오메트리에 잘 들어맞는다. 반면에 기온이나 토지이용 유형처럼 어느 곳에서나 값을 가지는 연속적인 현상에서는 무한히 많은 값을 나타내야 하며, 보통 해당 시공간 도메인(범위)을 일정하게 이산화하는 전략이 사용된다. 이러한 고려는 널리 알려져 있는 여러 데이터 구조로 이어진다.

* 시계열: 시간의 함수로서 타임라인으로 표시되는 데이터 구조
* 이미지 혹은 래스터: 2차원 공간데이터를 위한 데이터 구조
* 이미지의 시간적 시퀀스: 동적인 공간데이터를 위한 데이터 구조

세 번째의 데이터 구조는 변수 가 에 의존한다.

이는 시공간 어레이 또는 데이터 큐브의 전형적인 형태이다. 정의역을 규칙적으로 이산화한 지점들이 형성하는 입체적 모양이 큐브 형태를 띠기 때문이다. 치역을 형성하는 변수(여기서는 )를 큐브 차원이라고 부른다. 데이터 큐브는 여러 속성을 가질 수 있다.

만약 가 함수형이라면, 예를 들어 전자기 스펙트럼에서 측정된 반사율 값이라면, 스펙트럼 파장 가 추가적인 차원을 형성할 수 있다(). 6.5절에서는 컬러 밴드를 속성으로 표현하는 대안적 방식을 다룬다.

다중 시간 차원도 가능하다. 여러 다른 시간 에 대해 미래의 여러 다른 시점 에 대한 예측을 할 때나, 시간이 여러 차원으로 나누어질 때(연도, 연중 일자, 하루 중 시간 등)이다. 데이터 큐브의 가장 일반적인 정의는 개의 차원에서 개의 속성으로의 함수적 매핑이다.

여기에서는 1개 이상의 공간 차원과 0개 이상의 시간 차원을 가진 모든 데이터셋을 데이터 큐브로 간주한다. 이렇게 하면 다음의 모든 것을 포괄할 수 있다.

* 심플 피처(3.1절)
* 피처 집합에 대한 시계열
* 래스터 데이터
* 다분광 래스터 데이터(이미지)
* 다분광 래스터 데이터의 시계열(비디오)

### 6.2.1 정규 차원, GDAL’의 지오트랜스폼

데이터 큐브는 보통 다차원 어레이에 저장되며, 1-기반 배열 인덱스 와 관련된 규칙적으로 이산화된 차원 변수 사이의 일반적인 관계는 다음과 같다.

여기서 여기서 는 원점이고, 이 차원의 그리드 간격이다.

그림 1.6 b-c와 같은 보다 일반적인 경우에서 와 및 배열 인덱스 와 사이의 관계는 다음과 같다.

여기서 과 를 아핀(affine) 파라미터라고 하면, 이것은 GDAL에서 사용되는 소위 *지오변환*(geostransform)이다. 일 경우, 수식은 인 정사각형 셀을 가진 그림 1.6a의 규칙적 래스터로 축소된다. 정수 인덱스의 경우, 좌표는 그리드 셀의 시작 가장자리의 좌표이며, 셀 면적(픽셀)은 인덱스 값이 (포함)에서 (제외)로 범위가 설정된 셀을 차지한다. 가장 일반적인 이미지 형식의 경우, 는 음수로, 이는 이미지 행 인덱스가 값이 감소함에 따라(남쪽으로) 증가함을 나타낸다. 왼쪽 상단 그리드 셀의 *중심* 좌표를 얻기 위해서는(가 음수인 경우) 와 를 사용한다.

직교(rectilinear) 래스터의 경우(역자주: 그림 1.6의 네 번째 사례), 어레이 인덱스를 차원 값에 매핑하기 위한 테이블이 필요하다. 예를 들어 NetCDF 파일은 공간 차원(좌표) 변수의 모든 값을 항상 저장하는데, 주로 공간 그리드 셀의 중심 좌표값이거나 오프셋 값을 저장한다. 추가로, 그리드 셀의 경계를 저장하여 직교 차원을 정의하거나 좌표 변수 값과 셀 경계 간의 관계를 명확히 나타낼 수도 있다.

곡선형(curvilinear) 래스터의 경우(역자주: 그림 1.6의 다섯 번째 사례), 모든 조합을 쌍에 매핑하는 어레이가 필요하거나, 이를 수행하는 파라미터 함수(투영 또는 역투영 함수)가 필요하다. NetCDF 파일은 보통 두 가지를 모두 제공한다. GDAL은 이러한 어레이를 지리위치 (geolocation) 어레이라고 부르며, 이와 관련된 변환에 대해 광범위한 지원을 제공한다.

### 6.2.2 큐브 차원과 서포트

5.1절에서는 속성 변수의 공간적 서포트를 특정 관측 또는 예측과 관련된 지오메트리의 크기(길이, 면적, 부피)로 정의하였다. 동일한 개념이 시간적 서포트에도 적용된다. 시간은 시작 및 종료 시간이 명시된 기간으로 보고되는 경우가 드물지만, 많은 경우 타임스탬프 자체가 기간을 암시하거나(ISO-8601에서는 “2021”는 해당 연도의 한해 전체를, “2021-01”은 해당 연월의 전체를 의미하는 것으로 정의한다) 현재 레코드의 타임스탬프에서 다음 레코드의 타임스탬프까지(타임스템프 그 자체는 제외)를 기간으로 간주된다.

예를 들어, MODIS 위성 이미지는 16일 합성의 식생 지수(NDVI 및 EVI)를 제공하는데, 이는 16일 기간 동안 모든 사용 가능한 이미지를 단일 이미지로 집계한 것이다. 이러한 합성 이미지는 시간적 “블록 서포트”를 가진다. 반면에 Sentinel-2 또는 Landsat-8 데이터는 “스냅샷” 이미지를 제공하는데, 시간적 “포인트 서포트”를 가진다. 시간적 포인트 서포트 데이터를 월별 값으로 집계할 때는, 목표 시간 구간에 해당하는 모든 이미지를 선택한다. MODIS의 16일 합성과 같은 시간적 블록 서포트 이미지를 집계할 때는, 16일 합성 기간과 목표 기간이 겹치는 양에 따라 이미지에 가중치를 부여해야 하며, 이는 5.3절의 면적-가중 인터폴레이션과 유사한 것으로 시간-가중을 하는 것이다.

## 6.3 데이터 큐브에 작용하는 오퍼레이션

### 6.3.1 큐브의 분할: 필터

데이터 큐브는 특정 값으로 한 차원을 고정하여 다수의 서브-큐브로 분할할 수 있다. 그림 6.4는 시간, 스펙트럼 또는 공간 차원에서 이러한 방식으로 얻어진 서브-큐브를 보여준다. 이 그림에서 공간 필터링은 특정 공간 차원을 단일 값으로 고정하는 것이 아니라 특정 하위-영역을 선택함으로써 이루어는데, 이것이 보다 일반적인 작업이다. 또는 를 고정하면 일정한 또는 의 횡단면을 따라 서브 큐브가 생성되며, 이는 하나의 공간 차원과 하나의 시간 차원에서 속성이 플롯(색상으로 표현)된 호브몰러 다이어그램(Hovmöller diagram)을 보여주는 데 사용할 수 있다.

|  |
| --- |
| 그림 6.4: 시간, 밴드 혹은 공간에 의거한 데이터 큐브 필터링 |

### 6.3.2 차원에 함수를 적용하기

잘 사용되는 또 다른 오퍼레이션에 하나 이상의 큐브 차원에 대해 함수를 적용하는 것이 있다. 간단한 경우는 abs, sin 또는 sqrt와 같은 함수를 큐브의 모든 값에 적용하거나, 큐브의 모든 값을 단일 스칼라로 반환하는 경우(예: 큐브 전체에 대한 평균 또는 최대값을 계산하는 경우)이다. 다른 옵션으로는 선택된 차원에 함수를 적용하는 것이 있는데, 예를 들어 그림 6.5에서 볼 수 있는 것처럼 개별(픽셀/밴드) 시계열에 시간적 로-패스 필터(low-pass filter)를 적용하거나, 그림 6.6에서 볼 수 있는 것처럼 모든 밴드/시간 조합에 대해 각 공간 슬라이스에 공간적 로패스 필터를 적용하는 경우가 있다.

|  |
| --- |
| 그림 6.5: 시계열 데이터에 로-패스 필터링 적용하기 |

|  |
| --- |
| 그림 6.6: 공간 슬라이스에 로-패스 필터링 적용하기 |

### 6.3.3 차원의 삭감

전체 데이터 큐브에 평균이라는 함수를 적용하면, 모든 차원이 사라지게 된다. 결과로 생성된 “데이터 큐브”는 차원 0을 갖는다. 또한, 특정한 차원 집합에 함수를 적용할 수도 있다. 이렇게 되면 해당 차원만 사라지거나 축소된다. 필터링이 이러한 차원 삭감의 특별한 경우라는 것을 이미 보았지만, 모든 시계열의 최대값을 계산하거나, 각 공간 슬라이스에 대한 평균을 구하거나, NDVI와 같은 밴드 지수를 계산하여 서로 다른 스펙트럼 값을 단일의 새로운 “밴드”로 요약하는 것도 모두 차원 삭감과 관련되어 있다. 그림 6.7은 이러한 옵션을 설명한다.

|  |
| --- |
| 그림 6.7: 데이터 큐브의 차원 삭감하기 |

## 6.4 래스터 큐브를 벡터 큐브로 애그리게이션 하기

그림 6.8은 4차원 래스터 데이터 큐브를 3차원 벡터 데이터 큐브로 애그리게이션 하는 과정을 보여준다. 래스터의 픽셀은 벡터 지오메트리와의 공간적 교차 양상에 의거해 그룹화되며, 각 그룹은 평균 또는 최대값과 같은 집계 함수를 통해 단일 값으로 축소된다. 예를 들어, 두 개의 공간적 차원 와 는 피처 지오메트리의 1차원적 시퀀스로 구성된 단일 차원으로 축소되는데, 피처 지오메트리는 와 의 좌표를 가진 공간 상에서 정의된다. POINT 지오메트리에 의거해 애그리게이팅을 실행할 수도 있으며, 이 경우 집계 함수는 필요하지 않다. POINT 위치에서의 속성값은 픽셀 값을 쿼리하거나 가장 가까운 픽셀에서 인터폴레이션함으로써 추출할 수 있다.

|  |
| --- |
| 그림 6.8: 래스터 데이터 큐브를 벡터 데이터 큐브로 애그리게이션 하기 |

또 다른 벡터 데이터 큐브의 사례로 대기질 데이터를 들 수 있는데, 2차원의 를 다음에 의거해 획득할 수 있다.

* 모니터링 스테이션
* 시간 간격

이와 유사하게, 인구 수로 구성된 시계열 인구 데이터나 환자 수로 구성된 시계열 역학 데이터의 예를 들 수 있는데, 다음에 의거해 정의될 수 있다.

* 지역: 개의 지역
* 연령 그룹: 개의 연령 그룹
* 연도: 개의 연도

이는 로 구성된 어레이를 형성한다.

공간데이터사이언스에서 벡터 및 래스터 데이터 큐브를 다루는 것은 매우 유용하다. 우선 많은 변수가 공간적으로 그리고 시간적으로 변동하기 때문이다. 또한 차원의 변경 및 애그리게이션과 같은 과업을 완전히 유연한 방식하고도 질서있는 방식으로 수행할 수 있게 도와주기 때문이다. 차원을 변경하는 예는 다음과 같다.

* 대기질의 측정값을 규칙 그리드(래스터) 상에 인터폴레이션하는 것(12장)
* 포인트나 라인으로부터 밀도 값을 추정하는 것, 예를 들어 주 1회 비행기 통과 수를 1km 탐색반경을 기준으로 추정하는 것(11장)
* 기후 모델의 예측값을 행정구역별로 요약 지표로 애그리게이션하는 것
* MODIS(250m 픽셀, 16일 주기)와 Sentinel-2(10m 픽셀, 5일 주기)와 같은 서로 다른 센서의 지구 관측 데이터를 결합하는 것

하나 이상의 차원을 완전히 애그리게이션하는 사례에는 다음과 같은 것이 있다.

* 어떤 대기 질 모니터링 스테이션이 나쁨 수준을 보이는지(시간)
* 어떤 지역이 질병 발생률에서 가장 높은 증가를 보이는지(공간, 시간)
* 지구 온난화(10년 당 지구 전체의 섭씨 온도 변화)

## 6.5 차원을 속성으로 교체하기

차원이 무순서의 범주적 성격을 띨 수도 있는데, 이럴 경우에는 속성 집합을 단일 차원으로 쉽게 교환할 수 있다. 즉, 아래의 첫 번째 수식을 두 번째 수식으로 대체할 수 있다.

여기서 는 기수 를 가지며 레이블로서 (라는 이름)을 갖는다. 그림 6.9는 대기질 스테이션에 대한 벡터 데이터 큐브를 보여주는데, 한 차원이 대기질 파라미터들로 이루어져 있음을 볼 수 있다. 그림 6.9에서처럼 가 전혀 다른 측정 단위를 가지고 있는 경우에는 “파라미터” 차원 을 축소할 때는 주의해야 한다. 즉, mean 또는 max와 같은 함수는 무의미할 것이다. 그러나 각 변수의 임계값을 초과하는 수를 세는 것은 의미가 있을 수 있다.

|  |
| --- |
| 그림 6.9: 대기질의 시계열 데이터를 가진 벡터 데이터 큐브 |

차원과 속성을 유연하게 교환할 수 있게 되면 매우 유연한 분석이 이루어질 가능성을 높여준다(Brown, 2010).

## 6.6 기타 동적 공간데이터

우리는 데이터 큐브 구조와 잘 맞는 여러 동적 래스터 및 벡터 데이터 예제를 보았다. 그러나 다른 데이터 예제는 덜 그러하다. 특히 시공간 포인트 패턴(11장)과 경로[이동 데이터; 최근 리뷰는 Joo et al. (2020) 참조]는 종종 데이터 큐브로 처리하기보다는 보다 간편한 방식으로 다루어진다. 시공간 포인트 패턴은 사건이나 물체의 시공간 좌표의 집합이다. 즉, 사고, 발병, 교통 체증, 번개와 같은 것들이다. 경로 데이터는 이동하는 물체(사람, 자동차, 위성, 동물)의 공간적 위치의 시간적 시퀀스이다. 경로 데이터의 주요 정보는 좌표값인데, 이러한 좌표값을 규칙적으로 분할된 그리드 셀의 체계에서의 위치값으로 치환하여 사용할 수 있다. 이러한 치환은 고밀도 지역에서 패턴을 빠르게 탐색하는 등 일부 분석에서 도움이 될 수 있지만, 정확한 좌표의 손실로 말미암아 거리, 방향 또는 속도 계산을 포함하는 여러 접근 방식을 어렵게 한다. 그럼에도 불구하고, 이러한 데이터를 데이터 큐브 형태로 전환하여 사용하는 경우가 많은데, 시간을 고정한 상태에서 공간을 이산화하거나 공간을 고정한 상태에서 시간을 이산화한다.

포인트 패턴이나 경로 데이터를 희소 어레이 형태로 재현하는 것이 가능한데, SciDB(Brown, 2010) 또는 TileDB(Papadopoulos et al., 2016)를 활용할 수 있다. 이러한 방법은 좌표 정확도 손실의 문제를 상당히 해결할 수 있는데, 극단적으로 조밀한 그리드 차원을 선택하거나 데이터 포인트를 포함하는 그리드 셀만을 선택하는 것이다. 경로 데이터의 경우, 개체 단위를 식별하거나 연속적인 이동 시퀀스의 개별 단위를 식별하기 위한 그룹화 차원을 추가해야 한다.

## 6.7 연습문제

다음의 연습문제에 대해 글로 대답하라. 필요한 경우 주장을 설명하기 위해 R 코드를 사용하라.

1. 데이터 큐브를 사용하여 이동하는 물체를 추적하여 얻은 (x, y, t)의 경로 및 시퀀스를 표현하는 것이 왜 어려운가? 이 장의 내용에 기반하여 답하시오.
2. 인구, 기대 수명, 국내 총생산과 같은 변수로 구성된 사회경제적 벡터 데이터 큐브에서, 국가와 연도를 차원으로 하여 정렬할 때, 어떤 변수들이 공간 차원에 대한 블록 지원을 가지며, 어떤 변수들이 시간 차원에 대한 블록 지원을 가지는가?
3. Sentinel-2 위성은 12개의 스펙트럼 밴드로 이미지를 수집한다. 이를 (i) 밴드별로 서로 다른 데이터 큐브로, (ii) 12개 밴드를 12개 속성으로 가진 데이터 큐브로, (iii) 스펙트럼 차원을 가진 단일 속성 데이터 큐브로 표현할 때의 장점과 단점을 나열하라.
4. 그림 1.6에 나타나 있는 곡선 래스터가 특별한 데이터 큐브로 간주될 수 있는 이유를 설명하라.
5. 다음 문제들이 데이터 큐브 오퍼레이션인 필터, 적용, 축소 및/또는 애그리게이트를 사용하여 어떻게 해결될 수 있는지, 또 어떤 순서로 적용해야하는지 설명하라. 또한 각 문제에 대해 적용되는 함수와 결과 데이터 큐브의 차원(있는 경우)을 언급하라.
   * 대기질 모니터링 스테이션의 시간별 측정치로부터, 스테이션별 일평균 50 를 초과하는 연간 일수를 계산하라.
   * 석유 유출의 항공 이미지 시퀀스를 사용하여, 석유 유출이 가장 넓은 범위를 가졌던 시점과 해당 범위를 찾으라.
   * 10년 동안의 전 세계 일일 해수면 온도(SST) 래스터 맵을 사용하여, SST 값의 시간적 추세에서 상위 10%에 해당하는 지역과 하위 10%에 해당하는 지역을 찾으라.

# 7. sf와 stars

이 장에서는 R 패키지인 **sf**와 **stars**를 소개한다. **sf**는 피처의 기하 정보를 리스트-컬럼에 저장하는 단순 피처의 테이블 포맷을 제공한다. **stars** 패키지는 래스터와 벡터 데이터 큐브(6장에서 다룸)를 지원하기 위해 작성되었으며, 래스터 레이어, 래스터 스택 및 피처 시계열 또한 지원한다. **sf**는 2016년에 CRAN에 처음 등장했고, **stars**는 2018년에 등장했다. 두 패키지의 개발은 R 컨소시엄의 지원과 강력한 커뮤니티 참여로 이루어졌다. 이 패키지들은 함께 작동하도록 설계되었다. **sf**나 **stars** 객체를 대상으로 하는 함수 또는 메서드는 st\_로 시작하는데, 쉽게 관련 함수를 인식할 수 있을 뿐만 아니라 명령어 완성 기능을 사용할 때 검색의 용이함이 있다.

## 7.1 sf 패키지

R 패키지 **sf**(Pebesma 2018)는 기존의 R 패키지인 **sp**, r**geos** 및 **rgdal**의 벡터 부분을 대체하고 성공적으로 이어받기 위해 개발되었으며, 산업계 및 오픈 소스 프로젝트에서 볼 수 있는 표준 기반 접근법에 더 가까이 다가가고, 최신 버전의 오픈소스 지리공간 소프트웨어 스택(그림 1.7)을 기반으로 하며, 필요 시 R 공간 소프트웨어와 타이디버스(Wickham et al. 2019)의 통합을 가능하게 한다.

이를 위해 **sf**는 R에 네이티브로 심플 피처 접근(Herring et al. 2011)을 제공한다. 이 패키지는 여러 타이디버스 패키지, 특히 **ggplot2**, **dplyr**, **tidyr**와의 인터페이스를 제공하며, GDAL을 통해 데이터를 읽고 쓰고, GEOS(투영 좌표의 경우) 또는 s2geometry(타원체 좌표의 경우)를 사용하여 기하학적 연산을 수행하며, PROJ를 이용해 좌표 변환 또는 좌표 전환 작업을 수행할 수 있다. 외부 C++ 라이브러리와의 인터페이싱(연결)은 **Rcpp** 패키지(Eddelbuettel 2013)를 통해 이루어진다.

**sf**는 sf 객체로 심플 피처를 나타내며, 이는 data.frame 또는 티블(tibble)의 하위 클래스이다. sf 객체는 최소 하나 이상의 sfc 클래스의 지오메트리 *리스트 열*(list-column)을 포함하는데, 각 요소는 sfg 클래스의 R 객체로서 지오메트리 정보를 담고 있다. 지오메트리 리스트 열은 data.frame 또는 티블 내에서 변수처럼 작동하지만, 숫자나 문자형 변수와 같은 기본 벡터보다 더 복잡한 구조를 가지고 있다(부록 B.3 참조).

sf 객체는 다음과 같은 메타데이터를 가진다.

* (활성화된) 지오메트리 열의 이름: sf\_column 속성에 저장되어 있음.
* 각 비기하 변수의 속성-지오메트리 관계 (5.1절 참조): agr 속성에 저장되어 있음.

sfc 지오메트리 리스트 열은 st\_geometry함수를 통해 sf 객체에서 추출되며, 다음과 같은 메타데이터를 가진다

* 좌표참조계: crs 속성에 저장되어 있음.
* 바운딩 박스: bbox 속성에 저장되어 있음.
* 정밀도: precision 속성에 저장되어 있음.
* 지오메트리 수: n\_empty 속성에 저장되어 있음.

이러한 속성들의 값을 확인하거나 수정하기 위해 st\_bbox, st\_crs, st\_set\_crs, st\_agr, st\_set\_agr, st\_precision, st\_set\_precision 같은 함수를 사용할 수 있다.

### 7.1.1 생성

다음과 같은 방식으로 sf 객체를 생성할 수 있다.

library(sf)  
# Linking to GEOS 3.11.1, GDAL 3.6.4, PROJ 9.1.1; sf\_use\_s2() is TRUE  
p1 <- st\_point(c(7.35, 52.42))  
p2 <- st\_point(c(7.22, 52.18))  
p3 <- st\_point(c(7.44, 52.19))  
sfc <- st\_sfc(list(p1, p2, p3), crs = 'OGC:CRS84')  
st\_sf(elev = c(33.2, 52.1, 81.2),   
 marker = c("Id01", "Id02", "Id03"), geom = sfc)  
# Simple feature collection with 3 features and 2 fields  
# Geometry type: POINT  
# Dimension: XY  
# Bounding box: xmin: 7.22 ymin: 52.2 xmax: 7.44 ymax: 52.4  
# Geodetic CRS: WGS 84  
# elev marker geom  
# 1 33.2 Id01 POINT (7.35 52.4)  
# 2 52.1 Id02 POINT (7.22 52.2)  
# 3 81.2 Id03 POINT (7.44 52.2)

|  |
| --- |
| 그림 7.1: sf 객체의 구조 |

그림 7.1은 출력된 구성 요소에 대한 설명을 제공한다. 객체를 처음부터 생성하는 대신, R에서의 공간데이터는 보통 외부 소스에서 읽어오며, 그 외부 소스에는 다음과 같은 것이 있다.

* 외부 파일
* 데이터베이스 내의 테이블(또는 테이블 집합)
* 웹서비스에서 호출을 통해 획득된 데이터셋
* R 패키지에 포함되어 있는 데이터셋

### 7.1.2 읽기와 쓰기

외부 “데이터 소스”(파일, 웹서비스 또는 문자열)로부터 데이터셋을 읽어오는 것은 st\_read() 함수를 사용하여 수행된다.

library(sf)  
(file <- system.file("gpkg/nc.gpkg", package = "sf"))  
# [1] "/home/edzer/R/x86\_64-pc-linux-gnu-library/4.3/sf/gpkg/nc.gpkg"  
nc <- st\_read(file)  
# Reading layer `nc.gpkg' from data source   
# `/home/edzer/R/x86\_64-pc-linux-gnu-library/4.3/sf/gpkg/nc.gpkg'   
# using driver `GPKG'  
# Simple feature collection with 100 features and 14 fields  
# Geometry type: MULTIPOLYGON  
# Dimension: XY  
# Bounding box: xmin: -84.3 ymin: 33.9 xmax: -75.5 ymax: 36.6  
# Geodetic CRS: NAD27

여기서 파일 이름과 경로는 sf 패키지에서 읽어오는 것인데, sf 패키지를 설치하는 과정에서 설정된 것이므로 어떤 컴퓨터에서건 예외없이 읽어들여진다.

st\_read() 명령어는 두 개의 아규먼트(*데이터 소스 이름*(dsn)과 *레이어*(layer))를 가진다. 위의 예에서 *geopackage*(GPKG) 파일은 단일 레이어만 포함하고 있으며, 해당 레이어가 불러들여진 것이다. 만약 여러 레이어가 포함되어 있었다면, 첫 번째 레이어가 읽히고 경고 메시지가 표출되었을 것이다. 데이터셋의 사용 가능한 레이어는 다음과 같이 조회할 수 있다.

st\_layers(file)  
# Driver: GPKG   
# Available layers:  
# layer\_name geometry\_type features fields crs\_name  
# 1 nc.gpkg Multi Polygon 100 14 NAD27

심플 피처 객체는 st\_write 함수를 사용하여 저장할수 있다.

(file = tempfile(fileext = ".gpkg"))  
# [1] "/tmp/Rtmpm9lGRF/file361e653fae4a9.gpkg"  
st\_write(nc, file, layer = "layer\_nc")  
# Writing layer `layer\_nc' to data source   
# `/tmp/Rtmpm9lGRF/file361e653fae4a9.gpkg' using driver `GPKG'  
# Writing 100 features with 14 fields and geometry type Multi Polygon.

여기서 파일 형식(GPKG)은 파일 이름 익스텐션에서 파생된다. st\_write() 함수의 append 아규먼트 설정을 통해 기존 레이어에 레코드를 추가하거나 아예 레이어를 교체할 수 있는데, append 아규먼트 설정되지 않으면 레이어가 이미 존재할 경우 오류가 발생한다. 타이디버스 스타일의 write\_sf() 함수는 append가 설정되지 않은 경우 오류 표출 없이 레이어를 교체한다. 또한, st\_delete() 함수를 사용하여 레이어를 삭제할 수 있으며, 이는 특히 데이터베이스의 테이블과 연결되어 있는 레이어를 다룰 때 편리하다.

WKT-2 좌표참조계를 지원하는 파일 형식의 경우, st\_read()와 st\_write()는 이를 읽고 쓸 수 있다. 그러나 csv와 같은 간단한 포맷에서는 이 기능이 작동하지 않는다. 또한, 셰이프파일(shapefile) 형식은 CRS에 대해 매우 제한된 인코딩만 지원한다.

### 7.1.3 부분을 골라내기

매우 일반적인 작업 중 하나는 객체의 일부분을 골라내기(subset)하는 것이며, 베이스 R에서는 이를 위해 대괄호 기호([) 를 사용한다. data.frame 객체에 적용되는 규칙을 sf 객체에 그대로 적용할 수 있다. 예를 들어, 다음과 같은 코드를 통해 레코드 2에서 5와 열 3에서 7을 선택할 수 있다.

nc[2:5, 3:7]

여기에 몇몇 옵션을 부가적으로 적용할 수 있다.

* drop 아규먼트가 디폴트로 FALSE로 설정되어 있는데, 지오메트리 열이 *항상* 선택되며 sf 객체가 반환된다. TRUE로 설정되면, 지오메트리열이 선택되지 않으면 해당 열이 제거된 data.frame이 반환된다.
* 공간(sf, sfc 또는 sfg) 객체를 첫 번째 아규먼트로 사용한 선택은 해당 객체와 공간적으로 *교차*하는 피처를 선택하는 결과를 가져온다(다음 절 참조). 다른 프레디케이트를 선택하고자 할 경우, **op** 아규먼트를 설정하여 st\_covers와 같은 함수 또는 3.2.2절에 나열된 다른 이항 프레디케이트 함수로 지정할 수 있다.

### 7.1.4 바이너리 프레디케이트

st\_intersects, st\_covers와 같은 이항 프레디케이트 함수(3.2.2절 참조)는 두 개의 피처 집합 또는 피처 지오메트리를 입력받아 모든 쌍에 대해 조건이 TRUE인지 FALSE인지를 반환한다. 대규모 집합의 경우, 이는 일반적으로 대부분 FALSE 값으로 채워진 거대한 행렬을 생성할 수 있으며, 이러한 이유로 기본적으로 희소 표현(sparse representation)이 반환된다(역자주: 희소 표현은 메모리 사용을 최적화하고, 데이터 처리를 더욱 효율적으로 만들어 준다. 희소 표현을 일반적으로 TRUE 값만을 저장하고, FALSE 값은 저장하지 않는다.)

nc5 <- nc[1:5, ]  
nc7 <- nc[1:7, ]  
(i <- st\_intersects(nc5, nc7))  
# Sparse geometry binary predicate list of length 5, where the  
# predicate was `intersects'  
# 1: 1, 2  
# 2: 1, 2, 3  
# 3: 2, 3  
# 4: 4, 7  
# 5: 5, 6

|  |
| --- |
| 그림 7.2: 노스케롤라이나의 첫 7개의 카운티 |

그림 7.2는 첫 다섯 개 카운티와 첫 일곱 개 카운티의 교차를 이해하는 방법을 보여준다. 다음과 같은 방식으로 희소 논리 행렬을 조밀한 행렬(dense matrix)로 변환할 수 있다.

as.matrix(i)  
# [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
# [1,] TRUE TRUE FALSE FALSE FALSE FALSE FALSE  
# [2,] TRUE TRUE TRUE FALSE FALSE FALSE FALSE  
# [3,] FALSE TRUE TRUE FALSE FALSE FALSE FALSE  
# [4,] FALSE FALSE FALSE TRUE FALSE FALSE TRUE  
# [5,] FALSE FALSE FALSE FALSE TRUE TRUE FALSE

nc5의 개별 카운티가 교차하는 nc7 카운티의 수는 다음과 같이 계산할 수 있다.

lengths(i)  
# [1] 2 3 2 2 2

역으로 nc7의 개별 카운티가 교차하는 nc5의 카운티의 수는 다음과 같이 계산할 수 있다.

lengths(t(i))  
# [1] 2 3 2 1 1 1 1

객체 i 가 sgbp(sparse geometrical binary predicate) 클래스의 한 객체라고 했을 때, 객체 i는 정수 벡터의 리스트로 주어지는데, 리스트의 각 요소는 논리 프레디케이트 행렬의 한 행을 나타내고, 논리 프레디케이트 행렬은 해당 행에 대해 TRUE 값을 갖는 열의 인덱스를 보유하고 있다. 이 객체는 사용된 프레디케이트 및 총 열 수와 같은 메타데이터도 포함하고 있다. sgbp 객체에 적용할 수 있는 메소드에 다음과 같은 것들이 있다.

methods(class = "sgbp")  
# [1] as.data.frame as.matrix coerce dim   
# [5] initialize Ops print show   
# [9] slotsFromS3 t   
# see '?methods' for accessing help and source code

sgbp 클래스의 객체에서 사용 가능한 유일한 Ops 메서드는 !(부정 연산자)이다.

### 7.1.5 타이디버스

**tidyverse** 패키지는 다양한 데이터사이언스 패키지를 함께 로드한다(Wickham and Grolemund 2017; Wickham et al. 2019). **sf** 패키지는 **tidyverse** 스타일의 읽기 및 쓰기 함수인 read\_sf()와 write\_sf()를 제공하며, 이 함수들은 다음과 같은 특징이 있다.

* data.frame 대신 tibble을 반환한다.
* 출력 내용을 인쇄하지 않는다.
* 기본적으로 기본 데이터를 덮어쓴다.

sf 객체에 사용될 수 있는 **tidyverse** 로 filter, select, group\_by, ungroup, mutate, transmute, rowwise, rename, slice, summarise, distinct, gather, pivot\_longer, spread, nest, unnest, unite, separate, separate\_rows, sample\_n, 및 sample\_frac등과 같은 것이 있다. 대부분의 함수는 sf 객체의 메타데이터를 관리하기만 할 뿐 지오메트리 정보를 건드리지 않는다. 사용자가 지오메트리를 제거하고자 할 경우, st\_drop\_geometry를 사용하거나 선택하기 전에 간단히 tibble 또는 data.frame으로 강제 변환(coerce)할 수 있다.

library(tidyverse) |> suppressPackageStartupMessages()  
nc |> as\_tibble() |> select(BIR74) |> head(3)  
# # A tibble: 3 × 1  
# BIR74  
# <dbl>  
# 1 1091  
# 2 487  
# 3 3188

sf 객체에 대한 summarise 함수는 두 가지 특별한 아규먼트를 가지고 있다

* do\_union (기본값: TRUE): 그룹화된 지오메트리가 반환 시 유니언(합집합)되는지 여부를 결정하여, 유효한 지오메트리가 형성하도록 한다.
* is\_coverage (기본값: FALSE): 그룹화된 지오메트리가 커버리지(겹침이 없는 경우)를 형성하는 경우, 이를 TRUE로 설정하면 유니언 과정이 빨라진다.

distinct 함수는 고유한 레코드를 선택하며, st\_equals 함수는 지오메트리의 고유성을 평가한다.

filter 함수는 일반적인 프레디케이트와 함께 사용할 수 있으며, 공간적 프레디케이트를 사용하고자 할 경우, 예를 들어 오렌지 카운티에서 50km 이내에 있는 모든 카운티를 선택하려면 다음과 같이 사용할 수 있다.

orange <- nc |> dplyr::filter(NAME == "Orange")  
wd <- st\_is\_within\_distance(nc, orange,   
 units::set\_units(50, km))  
o50 <- nc |> dplyr::filter(lengths(wd) > 0)  
nrow(o50)  
# [1] 17

(여기서 dplyr::filter를 사용하는 것은 베이스 R의 filter 함수와의 혼동을 피하기 위함이다.)

그림 7.3은 이 분석의 결과를 보여주며, 카운티 경계 주위에 버퍼도 추가되어 있다. 이 버퍼는 설명을 위한 것이며, 카운티를 선택하는 데 사용되지는 않았음을 주의하라.

|  |
| --- |
| 그림 7.3: 오렌지 카운티(오렌지색), 반경 50km 내의 카운티(검은색), 오랜지 카운티 주변의 버퍼(갈색), 나머지 카운티(회색) |

## 7.2 공간적 조인

일반적인 조인(왼쪽, 오른쪽 또는 내부 조인)에서는 두 테이블 사이에서 하나 이상의 속성이 일치할 때 조인이 이루어진다. 공간적 조인도 이와 유사하지만, 레코드를 조인하는 기준은 속성의 일치가 아니라 공간적 프레디케이트이다. 이로 인해 공간적으로 일치하는 레코드를 정의하는 다양한 선택지가 있으며, 이는 3.2.2절에 나열된 바이너리 프레디케이트를 사용하여 결정할 수 있다. “왼쪽,” “오른쪽,” “내부,” 또는 “전체” 조인의 개념은 비공간 조인과 동일하게 유지되는데, 이 경우는 공간적 일치를 상정하지 않은 상태에서 레코드의 조인을 처리할 때이다.

공간적 조인을 실행할 때, 각 레코드에 여러 일치하는 레코드가 있을 수 있어 결과 테이블이 매우 커질 수 있다. 이 복잡성을 줄이는 방법으로, 일치하는 레코드 중에서 타깃 지오메트리와 가장 넓은 면적이 겹치는 레코드 하나를 선택하는 방식이 있다. 이 방법의 시각적 예시는 그림 7.4에 나타나 있으며, st\_join 함수에서 largest = TRUE 아규먼트를 사용하여 이를 수행할 수 있다.

|  |
| --- |
| 그림 7.4: largest = TRUE 아규먼트를 적용한 st\_join 함수의 예: 아래쪽 그림의 폴리곤과 가장 넓은 면적이 겹치는 위쪽 그림의 폴리곤 라벨이 아래쪽 폴리곤에 할당되어 있다. |

결과의 복잡성을 줄이는 또 다른 방법은 조인 후에 aggregate 함수를 사용하여 모든 일치하는 레코드를 결합함과 동시에 지오메트리도 병합하는 것이다. 이에 대한 자세한 내용은 5.4절을 참고하라.

### 7.2.1 샘플링, 그리딩, 인터폴레이팅

sf 패키지가 제공하는 유용한 함수 몇 가지를 소개하고자 한다. st\_sample 함수는 타깃 지오메트리로부터 임의의 샘플링 포인트를 생성해 주는데, 타깃 지오메트리는 점, 선, 또는 폴리곤 등 다양할 수 있다. 샘플링 방식으로 완전 무작위 방식, 규칙적 방식, 또는 폴리곤의 경우 삼각형 방식이 선택될 수 있다. 11장에서 spatstat 패키지에서 제공하는 공간적 샘플링(또는 포인트 패턴 시뮬레이션) 방법들이 어떻게 st\_sample 함수를 통해 이루어지는지 설명한다.

st\_make\_grid 함수는 특정 영역 위에 정사각형, 직사각형, 또는 육각형의 그리드를 생성해준다. 옵션을 다르게 적용하면 그리드 자체가 아니라 그리드의 중심점 혹은 구석점을 생성해준다. 이 함수는 그림 7.4에서 직사각형 그리드를 생성하는 데 사용되었다.

함수 st\_interpolate\_aw는 5.3절에서 설명된 대로 공간적으로 내포적인 변수와 공간적으로 외연적인 변수를 새로운 영역으로 “인터폴레이션”하는 기능을 제공한다.

## 7.3 타원 좌표

비투영 데이터는 경위도로 표현된 타원체 좌표를 가진다. 4.1절에서 설명된 대로, 포인트간 “직선”은 최단 곡선 경로(“측지선”)이다. 기본적으로 **sf** 패키지는 s2geometry 라이브러리의 기하학적 연산을 사용하며, 이는 **s2** 패키지를 통해 이루어진다(Dunnington, Pebesma, and Rubak 2023). 예를 들어, 아래의 지점은 특정 폴리곤 *내부*에 존재한다(그림 7.5의 왼쪽 그림: 정사 도법).

"POINT(50 50.1)" |> st\_as\_sfc(crs = "OGC:CRS84") -> pt

"POLYGON((40 40, 60 40, 60 50, 40 50, 40 40))" |>  
 st\_as\_sfc(crs = "OGC:CRS84") -> pol  
st\_intersects(pt, pol)  
# Sparse geometry binary predicate list of length 1, where the  
# predicate was `intersects'  
# 1: 1

|  |
| --- |
| 그림 7.5: 인터섹션의 결과는 측지선 혹은 대권호를 사용하느냐(왼쪽) 데카르트 좌표계를 사용하느냐에 따라 달라진다. |

**sf** 패키지가 타원체 좌표를 마치 데카르트 좌표처럼 사용하도록 하려면, **s2** 사용을 비활성화하면 된다.

old <- sf\_use\_s2(FALSE)  
# Spherical geometry (s2) switched off  
st\_intersects(pol, pt)  
# although coordinates are longitude/latitude, st\_intersects assumes  
# that they are planar  
# Sparse geometry binary predicate list of length 1, where the  
# predicate was `intersects'  
# 1: (empty)  
sf\_use\_s2(old) # restore  
# Spherical geometry (s2) switched on

이렇게 하면, 그림 7.5의 오른쪽 그림(정거원통 도법)처럼, 엠프티 인터섹션이 리턴된다. 타원체 좌표를 평면 좌표로 취급한다는 점이 경고 메시지에 명시되어 있다.

성능이나 레거시 구현과의 호환성 이유로 **s2** 사용을 비활성화할 수 있다. 데카르트 기하학을 위한 **GEOS** 라이브러리와 구체 기하학을 위한 **s2geometry** 라이브러리(그림 1.7)는 서로 다른 동기로 개발되었으며, **sf**를 통해 사용될 때 그 방식에서 몇 가지 차이가 있다.

* 특정 오퍼레이션에서 속도차가 크게 날 수 있다.
* 특정 함수는 오로지 특정 라이브러리에만 존재한다(예를 들어 st\_relate 함수는 **GEOS** 라이브러리에만 존재)
* 변환자(transformer)를 사용할 때, **GEOS**는 외부 폴리곤 링을 시계 방향으로 노드로 반환하며, 이를 반시계 방향으로 되돌리기 위해 st\_sfc(..., check\_ring\_dir = TRUE)를 사용한다. 반면, s2geometry는 외부 폴리곤 링을 반시계 방향으로 반환한다.

## 7.4 stars 패키지

sp 패키지가 래스터 데이터에 대한 지원이라는 측면에서 정체되어 있는 동안, raster 패키지(Hijmans 2023a)가 지난 10여년 동안 래스터 분석을 위한 강력하고 유연하며 확장 가능한 패키지로서 지배적인 위치를 공고히 하였다. raster 패키지(및 그 후속인 terra 패키지(Hijmans 2023b))는 2D 규칙 래스터 또는 래스터 레이어 집합(“래스터 스택”)이라는 래스터 데이터 모델에 기반하고 있다. 이는 세상이 수많은 레이어로 구성되어 있고 개별 레이어는 특정한 주제를 반영하고 있다는 고전적인 정적 “GIS 뷰”와 일치한다. 그러나 오늘날의 많은 데이터는 동적이며, 시계열 래스터 또는 시계열 래스터 스택으로 제공된다. 기존의 래스터 스택은 이러한 동적 특성을 제대로 반영하지 못하며, 사용자늘 어떤 레이어가 무엇을 나타내는지를 기록하고 있어야만 한다.

또한, raster 패키지와 그 후속인 terra 패키지는 데이터 크기가 로컬 저장소(컴퓨터의 하드 드라이브)보다 크지 않을 때에만 훌륭한 전산 처리 능력을 발휘한다. 그러나 최근 데이터셋, 예를 들어 위성 이미지, 기후 모델 또는 기상 예보 데이터는 로컬 저장소의 용량으로는 더 이상 감당하기 어려운 수준에 도달했다(9장 참조). spacetime 패키지(Pebesma 2012, 2022)는 벡터 지오메트리 또는 래스터 그리드 셀의 시계열 분석을 어느 정도 다룰 수 있다. 하지만 더 높은 차원의 어레이나 메모리가 감당하기 어려운 정도의 크기를 가진 데이터셋은 여전히 다루기 어렵다.

여기서 우리는 래스터 및 벡터 데이터 큐브 분석을 위한 패키지로 stars를 소개한다. 이 패키지는 다음과 같은 기능을 제공한다.

* 동적(시간 변화) 래스터 스택을 재현할 수 있다.
* 로컬 디스크의 크기에 한정되지 않는 확장 가능성을 목표로 한다.
* GDAL 라이브러리의 래스터 기능과 강력한 통합을 제공한다.
* 규칙 그리드 외에도 회전, 전단, 직선, 곡선 래스터를 처리할 수 있다(그림 1.6 참조).
* sf 패키지와의 긴밀한 통합을 제공한다.
* 비래스터(non-raster) 공간 차원을 가진 어레이 데이터(벡터 데이터 큐브)를 처리할 수 있다.
* 타이디버스 디자인 원리를 따른다.

벡터 데이터 큐브에는 심플 피처의 시계열이나 출발지-목적지 매트릭스(그것의 시계열 포함)와 같은 공간 그래프 데이터가 포함된다. 공간적 벡터 및 래스터 데이터 큐브의 개념은 6장에서 설명되었다. 불규칙 시공간 관측치는 sftime 패키지(Teickner, Pebesma, and Graeler 2022)에서 제공하는 sftime 객체로 재현될 수 있으며, 이는 시간 열을 추가하는 방식으로 sf 객체를 확장한 것이다(13.3절 참조).

### 7.4.1 래스터 데이터의 읽기와 쓰기

래스터 데이터는 일반적으로 파일을 불러온다. 우리는 브라질의 올린다 시에 대한 30m 해상도의 Landsat 7 데이터셋(밴드 1-5 및 7)을 사용한다. 패키지 **stars**에서 규칙 비회전 그리드에 대한 예제 GeoTIFF 파일을 읽을 수 있다.

tif <- system.file("tif/L7\_ETMs.tif", package = "stars")  
library(stars)  
# Loading required package: abind  
(r <- read\_stars(tif))  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 1 54 69 68.9 86 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA

여기서 우리는 오프셋, 셀 크기, 좌표참조계, 및 차원을 확인할 수 있다. 차원 테이블은 각 차원에 대해 다음과 같은 필드를 포함한다.

* from: 시작 인덱스 값
* to: 종료 인덱스 값
* offset: 첫 번째 픽셀의 시작(모서리)에서의 차원 값
* delta: 셀 크기로 음의 delta 값은 디멘션 값이 감소할수록 필셀 인덱스 값이 증가한다는 것을 의미
* refsys: 참조계
* point: 셀 값이 포인트 서포트인지, 셀 서포트인지를 명시하는 논리 값
* x/y: 디멘션이 래스터의 x- 축과 관련되는지 y-축과 관련되는지를 나타내는 값

여기에는 사용되지 않기 때문에 숨겨진 또 다른 필드인 values가 있다. 정규, 회전, 또는 전단 그리드와 같은 규칙적으로 이산화된 차원(예: 시간)의 경우, 오프셋과 델타는 NA가 아니다. 반면 불규칙한 경우에는 오프셋과 델타가 NA이며, values 속성은 다음 중 하나를 포함한다.

* 값 또는 구간의 시퀀스: 직선 공간 래스터 또는 불규칙 시간 차원의 경우
* 공간 차원과 연결된 지오메트리: 벡터 데이트 큐브의 경우
* 각 래스터 셀에 대한 좌표값이 포함된 매트릭스: 곡선 래스터의 경우
* 차원 값과 연결된 밴드 이름 또는 레이블: 이산 디멘션의 경우

stars 클래스의 객체 r은 길이가 1인 간단한 리스트로 구성되어 있으며, 3차원 어레이를 포함한다.

length(r)  
# [1] 1  
class(r[[1]])  
# [1] "array"  
dim(r[[1]])  
# x y band   
# 349 352 6

또한, 이 객체는 어레이 차원이 무엇을 나타내는지 알기 위해 필요한 모든 메타데이터를 포함한 디멘션 테이블을 속성으로 가지고 있다. 이는 다음을 통해 얻어진다.

st\_dimensions(r)

어레이의 공간적 범위에 대한 정보를 다음과 같이 얻을 수 있다.

st\_bbox(r)  
# xmin ymin xmax ymax   
# 288776 9110729 298723 9120761

write\_stars 함수를 통해 래스터 데이터를 로컬 디스크에 저장할 수 있다.

tf <- tempfile(fileext = ".tif")  
write\_stars(r, tf)

파일 확장자를 통해 데이터 형식(이 경우, GeoTIFF)이 지정된다. 단순 피처와 마찬가지로 읽기 및 쓰기 작업은 GDAL 라이브러리를 사용하며, 래스터 데이터에 사용할 수 있는 드라이버 목록은 다음을 통해 확인할 수 있다.

st\_drivers("raster")

### 7.4.2 stars 데이터 큐브로부터 일부분 골라내기

데이터 큐브는 [ 연산자를 사용하거나 타이디버스 동사를 사용하여 부분 집합을 만들 수 있다. 첫 번째 옵션인 [를 사용하는 방식은 다음의 아규먼트를 쉼표의 구분과 함께 순서대로 지정하는 것이다.

* 속성(이름, 인덱스, 또는 논리 벡터)
* 차원

예를 들어, r[1:2, 101:200,, 5:10]는 r에서 속성 1-2를 선택하고, 디멘션 1에 대해 인덱스 101-200, 차원 3에 대해 인덱스 5-10을 선택함을 의미한다. 차원 2를 통한 선택은 이루어지지 않는다. 속성의 경우, 속성 이름, 인덱스 또는 논리 벡터를 사용할 수 있다. 차원의 경우에는 논리 벡터가 지원되지 않는다. 불연속 범위 선택은 규칙 시퀀스일 때만 지원된다. 기본적으로 drop은 FALSE로 설정되어 있으며, TRUE로 설정하면 단일 값을 가진 차원은 모두 제거된다.

r[,1:100, seq(1, 250, 5), 4] |> dim()  
# x y band   
# 100 50 1  
r[,1:100, seq(1, 250, 5), 4, drop = TRUE] |> dim()  
# x y   
# 100 50

특정 범위의 차원 *값*을 선택하기 위해 filter 함수를 사용할 수 있는데, 이는 dplyr 패키지를 우선적으로 로드해야 한다.

library(dplyr, warn.conflicts = FALSE)  
filter(r, x > 289000, x < 290000)  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 5 51 63 64.3 75 242  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 35 289004 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 1 1 NA NA

이는 차원의 오프셋을 변경한다. 특정 큐브 슬라이스는 slice 함수를 사용하여 얻을 수 있다.

slice(r, band, 3)  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 21 49 63 64.4 77 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]

이는 단일 차원 band를 제거한다. mutate 함수는 stars 객체에서 기존 어레이를 기반으로 새로운 어레이를 추가하는 데 사용되며, transmute는 여기에 덧붙여 기존 어레이를 제거한다.

### 7.4.3 잘라내기

부분 집합을 생성하는 또 다른 방법에 sf, sfc 또는 bbox 클래스의 공간 객체를 사용하는 방법이 있다.

b <- st\_bbox(r) |>  
 st\_as\_sfc() |>  
 st\_centroid() |>  
 st\_buffer(units::set\_units(500, m))  
r[b]  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# L7\_ETMs.tif 22 54 66 67.7 78.2 174 2184  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 157 193 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 159 194 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA

예를 들어, 해당 지역에 대한 직경 500m의 원형 중심부를 추출할 수 있는데, 그림 7.6에는 첫 번째 밴드에 적용한 결과가 나타나 있다.

|  |
| --- |
| 그림 7.6: Landsat 7 (band 1) 이미지의 원형 중심부 |

원형 공간 객체의 외부에 존재하는 픽셀에는 NA 값이 할당되는 것을 볼 수 있다. 이 원형 객체는 여전히 r의 offset과 delta 값에 대한 차원 인덱스를 가지고 있다. 다음과 같은 방식으로 offset 값을 재설정할 수 있다.

r[b] |> st\_normalize() |> st\_dimensions()  
# from to offset delta refsys point x/y  
# x 1 37 293222 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 36 9116258 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA

기본적으로, 결과 래스터는 선택 객체의 범위로 잘린다. 입력 객체와 동일한 차원을 가진 객체는 다음과 같은 방식으로 얻을 수 있다.

r[b, crop = FALSE]  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# L7\_ETMs.tif 22 54 66 67.7 78.2 174 731280  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA

stars 객체의 잘라내기는 st\_crop 함수를 사용하여 직접 수행할 수도 있다.

st\_crop(r, b)

### 7.4.4 stars 객체의 차원재부여 및 결합

stars 패키지는 다양한 어레이 조작을 수행하기 위해 패키지 abind (Plate and Heiberger 2016)를 사용한다. 한 예로 어레이를 순열하여 전치하는 aperm 함수이다.

aperm(r, c(3, 1, 2))  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 1 54 69 68.9 86 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# band 1 6 NA NA NA NA   
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]

stars 객체에 대한 메서드가 제공되며, 결과 객체의 차원 순서를 순열한다.

속성과 차원을 교환할 수 있으며, 이는 split과 merge를 사용하여 수행된다.

(rs <- split(r))  
# stars object with 2 dimensions and 6 attributes  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# X1 47 67 78 79.1 89 255  
# X2 32 55 66 67.6 79 255  
# X3 21 49 63 64.4 77 255  
# X4 9 52 63 59.2 75 255  
# X5 1 63 89 83.2 112 255  
# X6 1 32 60 60.0 88 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
merge(rs, name = "band") |> setNames("L7\_ETMs")  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs 1 54 69 68.9 86 255  
# dimension(s):  
# from to offset delta refsys point values x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE NULL [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE NULL [y]  
# band 1 6 NA NA NA NA X1,...,X6

split 함수는 밴드 차원을 2차원 어레이의 여섯 개 속성에 분배하며, merge 함수는 거꾸로 수행한다. st\_redimension 함수는 단일 어레이 차원을 두 개의 새로운 차원으로 분할하는 것과 같은 보다 일반적인 작업에 사용된다.

st\_redimension(r, c(x = 349, y = 352, b1 = 3, b2 = 2))  
# stars object with 4 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 1 54 69 68.9 86 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# b1 1 3 NA NA NA NA   
# b2 1 2 NA NA NA NA

동일한 차원을 가진 여러 개의 stars 객체는 c를 사용하여 결합할 수 있다. 결합된 어레이는 추가적인 속성으로 취급하는 것이 디폴트 설정이지만, along 아규먼트를 지정하면 어레이를 새로운 차선을 따라 병합할 수도 있다.

c(r, r, along = "new\_dim")  
# stars object with 4 dimensions and 1 attribute  
# attribute(s), summary of first 1e+05 cells:  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 47 65 76 77.3 87 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA   
# new\_dim 1 2 NA NA NA NA

이의 사용 예시는 7.5.2절에서 설명된다.

### 7.4.5 포인트 샘플 추출하기, 애그리게이팅하기

래스터 데이터 큐브 분석의 매우 일반적인 사용 사례는 특정 위치에서 값을 추출하거나 특정 지오메트리에 대해 집계값을 계산하는 것이다. st\_extract 함수는 포인트 값을 추출한다. 우리는 r 객체의 바운딩 박스 내의 몇 개의 무작위로 샘플링 포인트에 대해 이 오퍼레이션을 수행할 것이다.

set.seed(115517)  
pts <- st\_bbox(r) |> st\_as\_sfc() |> st\_sample(20)  
(e <- st\_extract(r, pts))  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 12 41.8 63 61 80.5 145  
# dimension(s):  
# from to refsys point  
# geometry 1 20 SIRGAS 2000 / ... TRUE  
# band 1 6 NA NA  
# values  
# geometry POINT (293002 ...,...,POINT (290941 ...  
# band NULL

이것은 20개의 포인트와 6개의 밴드를 가진 벡터 데이터 큐브를 생성한다(시드를 설정하게 되면, 반복 실행에서 동일한 샘플의 사용이 보장된다. 따라서 포인트의 무작위 생성을 재실행하고자 하는 경우는 시드를 설정해서는 안된다).

데이터 큐브에서 정보를 추출하는 또 다른 방법은 집계값을 산출하는 것이다. 이를 수행하는 한 가지 방법은 공간 폴리곤 또는 라인에 의거해 값을 공간적으로 집계하는 것이다(6.4절 참조). 예를 들어, 그림 1.4(d)에 표시된 세 개의 원 각각에 대해 여섯 개 밴드의 최대 픽셀 값을 계산할 수 있다.

circles <- st\_sample(st\_as\_sfc(st\_bbox(r)), 3) |>  
 st\_buffer(500)  
aggregate(r, circles, max)  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 73 94.2 117 121 142 205  
# dimension(s):  
# from to refsys point  
# geometry 1 3 SIRGAS 2000 / ... FALSE  
# band 1 6 NA NA  
# values  
# geometry POLYGON ((2913...,...,POLYGON ((2921...  
# band NULL

이는 세 개의 지오메트리와 여섯 개의 밴드를 가진 (벡터) 데이터 큐브를 생성한다. 시간 차원에 대한 집계값의 산출은 aggregate 함수의 두 번째 아규먼트로 시간 변수를 설정하여 수행된다. 시간 변수에 다음과 같은 것이 있을 수 있다.

* 시간 간격의 시작을 나타내는 타임 스탬프의 집합
* make\_intervals 함수로 정의된 시간 간격 집합
* “주”, “5일” 또는 “년”과 같은 시간 기간

### 7.4.6 예측 모델

R에서의 일반적인 모델 예측 워크플로우는 다음과 같다.

* 응답 변수와 예측 변수(공변량)가 포함된 data.frame의 사용
* data.frame을 기반으로 모델 객체를 생성
* 모델 객체와 대상 예측 변수 값이 포함된 data.frame을 사용하여 predict를 호출

stars 패키지는 stars 객체에 대한 predict 메서드를 제공하며, 이는 본질적으로 위의 마지막 단계를 수행하는데, data.frame을 생성하고, predict 메서드를 호출한 후, 예측 값으로 stars 객체를 재구성한다.

이 과정을 설명하기 위해 Landsat 데이터셋에 위에서 추출한 샘플 포인트를 적용해 육지를 바다에서 분리하는 간단한 이진 클래스 예제를 사용할 것이다. 결과는 그림 7.7에 나타나 있다.

|  |
| --- |
| 그림 7.7: 트레이닝 데이터로 사용되는 무작위 샘플 포인트: 빨간색은 해양부이고 노란색을 육지부이다. |

이 그림에서 포인트 8, 14, 15, 18 및 19는 수부에 위치하고 있으며, 나머지는 육지부에 있다는 것을 “육안”으로 확인할 수 있다. 선형 판별(“최대 우도”) 분류기를 사용하여, 우리는 그림 7.8에 나타난 모델 예측 결과를 얻을 수 있다.

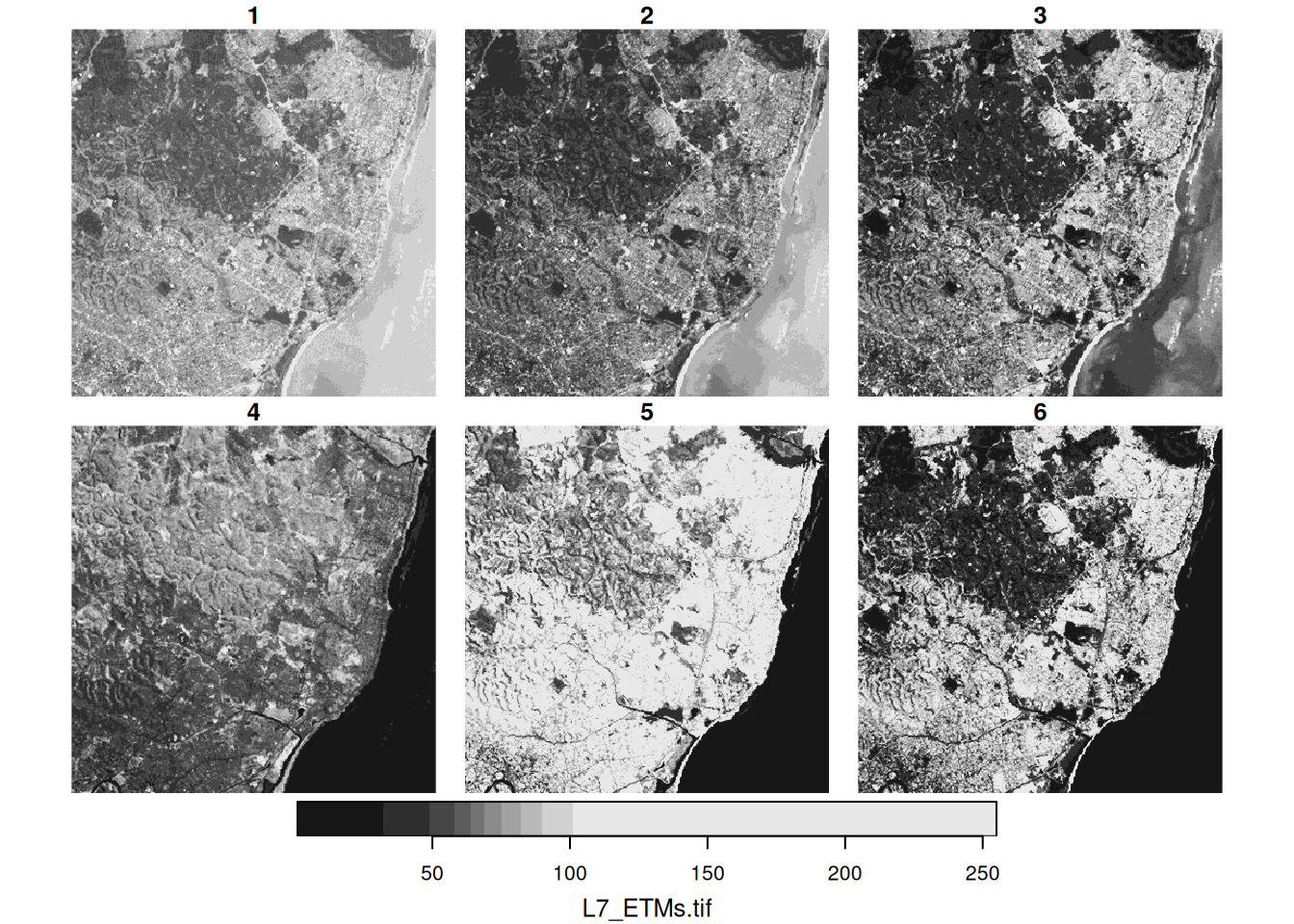
rs <- split(r)  
trn <- st\_extract(rs, pts)  
trn$cls <- rep("land", 20)  
trn$cls[c(8, 14, 15, 18, 19)] <- "water"  
model <- MASS::lda(cls ~ ., st\_drop\_geometry(trn))  
pr <- predict(rs, model)

여기서 우리는 MASS:: 접두사를 사용하여 **MASS** 패키지를 로드하지 않았으며, 이는 dplyr의 select 함수를 가리는 것을 방지하기 위함이다. split 단계는 밴드 차원을 속성으로 변환하여 예측 변수로 투입하기 위해 필요한 단계이다.

|  |
| --- |
| 그림 7.8: 육지부/수부 구분을 위한 선형 판별 분류기로 그림 7.7의 트레이닝 데이터에 기반한 결과임. |

우리는 또한 그림 7.8에 플로팅된 레이어가 클래스 레이블을 가진 범주형 변수임을 알 수 있다.

### 7.4.7 래스터 데이터 플로팅



30m 해상도의 Landsat 6개 밴드를 90m로 다운샘플한 결과, 브라질의 올린다

베이스 플롯 함수를 stars 객체에 적용할 수 있는데, plot(r)로 생성된 플롯은 그림 7.9에 나타나 있다. 기본 색상 스케일은 회색 톤을 사용하며, 모든 밴드의 데이터 분위수에 맞춰 명암 대비가 조정된다(“히스토그램 평활화”)(역자주: 히스토그램 평활화는 데이터의 분토를 재조정하여 시각적으로 더 유용하고 세부 사항을 더 잘 볼 수 있게 하는 기법을 말한다. 예를 들어, 어두운 이미지에서는 어두운 영역의 픽셀 값이 많이 몰려 있는데, 히스토그램 평활화를 적용하면 이 픽셀 값의 범위를 더 넓은 범위로 확장(stretch)시켜 이미지를 밝고 명확하게 보이도록 할 수 있다.) breaks = "equal"로 설정하면 급폭이 동일하게 설정되는 등간격 분류법이 적용되며, 계급 단절값을 임의로 지정할 수도 있다. 그런데 더 익숙한 플로팅 방식은 그림 7.10에 나타난 RGB 또는 폴스 컬러 합성일 것이다.

|  |
| --- |
| 그림 7.9: 컬러 합성의 두 가지 예 |

보다 자세한 사항은 8장에서 다루어진다.

### 7.4.8 래스터 데이터 분석하기

stars 객체의 개별 엘리먼트에 대해서는, 수리 함수의 적용이 어레이에 직접적으로 이루어진다. 이는 사용자가 함수를 호출하여 표현식(expression)을 생성하는 것이 가능하다는 것을 의미한다.

log(r)  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 0 3.99 4.23 4.12 4.45 5.54  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA  
r + 2 \* log(r)  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 1 62 77.5 77.1 94.9 266  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA

혹은 특정 값들을 마스킹 처리할 수도 있다.

r2 <- r  
r2[r < 50] <- NA  
r2  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# L7\_ETMs.tif 50 64 75 79 90 255 149170  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA

혹은 마스킹을 해제할 수도 있다.

r2[is.na(r2)] <- 0  
r2  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 0 54 69 63 86 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]  
# band 1 6 NA NA NA NA

stars 객체의 개별 차원에 대해서는, 선택된 어레이 차원에 함수를 적용하는 것이 가능하며, 이는 apply 함수가 어레이에 대해 수행하는 방식과 유사하다(6.3.3절). 예를 들어, 각 픽셀에 대해 6개 밴드 값의 평균을 계산할 수 있다.

st\_apply(r, c("x", "y"), mean)  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# mean 25.5 53.3 68.3 68.9 82 255  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]

더 유의미한 함수로는 예를 들어 NDVI(Normalized Difference Vegetation Index, 정규화 식생 지수 )를 계산하는 것이 있다.

ndvi <- function(b1, b2, b3, b4, b5, b6) (b4 - b3)/(b4 + b3)  
st\_apply(r, c("x", "y"), ndvi)  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# ndvi -0.753 -0.203 -0.0687 -0.0643 0.187 0.587  
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE [x]  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE [y]

혹은, 다음과 같이 정의할 수도 있다.

ndvi2 <- function(x) (x[4]-x[3])/(x[4]+x[3])

밴드 수가 많을 경우 이러한 방식은 더 편리하지만, 각 픽셀에 대해 호출해야 하므로 위에서 정의한 ndvi 함수보다 훨씬 느리다. 반면 ndvi 함수는 모든 픽셀이나 큰 픽셀 덩어리에 대해 한 번 호출할 수 있다. 전체 이미지에 대해 각 밴드의 평균은 다음과 같이 계산된다.

st\_apply(r, c("band"), mean) |> as.data.frame()  
# band mean  
# 1 1 79.1  
# 2 2 67.6  
# 3 3 64.4  
# 4 4 59.2  
# 5 5 83.2  
# 6 6 60.0

결과는 data.frame의 형태로 막바로 출력해 볼 수 있을 만큼 작다. 이 두 가지 예에서는 전체 디멘션이 사라진다. 그러나 그렇지 않은 경우도 종종 발생한다(6.3.2절). 예를 들어, 각 밴드에 대해 세 개의 사분위수를 계산할 수 있다.

st\_apply(r, c("band"), quantile, c(.25, .5, .75))  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 32 60.8 66.5 69.8 78.8 112  
# dimension(s):  
# from to values  
# quantile 1 3 25%, 50%, 75%  
# band 1 6 NULL

이렇게 하면 세 개의 값으로 이루어진 새로운 차원인 quantile이 *생성*된다. 또는 각 픽셀에 대해 여섯 개의 밴드 값에 대한 세 개의 사분위수를 다음과 같은 방식으로 얻을 수도 있다.

st\_apply(r, c("x", "y"), quantile, c(.25, .5, .75))  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 4 55 69.2 67.2 81.2 255  
# dimension(s):  
# from to offset delta refsys point  
# quantile 1 3 NA NA NA NA  
# x 1 349 288776 28.5 SIRGAS 2000 / ... FALSE  
# y 1 352 9120761 -28.5 SIRGAS 2000 / ... FALSE  
# values x/y  
# quantile 25%, 50%, 75%   
# x NULL [x]  
# y NULL [y]

### 7.4.9 곡선 래스터

비규칙 래스터가 발생하는 여러 가지 이유가 있다(그림 1.6). 우선, 데이터가 지구의 형태를 그대로 반영하고 있을 때, 규칙 래스터는 곡면인 지구의 표면에 맞지 않는다. 다른 이유에는 다음과 같은 것들이 있다.

* 규칙 래스터 데이터를 다른 좌표참조계로 전환하거나 변환할 경우, 리샘플링, 즉 워핑(7.8절)을 하지 않으면 곡선 형태가 된다. 그런데 워핑은 항상 데이터 손실을 초래하며, 이는 가역적이지 않다.
* 관측이 비규칙 래스터를 초래할 수 있다. 품질이 낮은 위성 영상의 경우, 위성의 진행 방향에서는 규칙 래스터가 되지만( 또는 와 정렬되지 않음), 그와 수직인 방향에서는 직각(rectilinear) 래스터가 된다(예: 센서가 시야*각*을 일정한 간격으로 분할하여 관측을 수행하는 경우).

### 7.4.10 GDAL 유틸리티

GDAL 라이브러리는 일반적으로 데이터 변환 및 처리를 위한 여러 실행 가능한 바이너리, 즉 GDAL 명령줄 유틸리티와 함께 제공된다. 이러한 유틸리티 중 여러 개(파이썬으로 작성된 것을 제외한 모든 유틸리티)는 “GDAL Algorithms C API”를 통해 GDAL 라이브러리의 C 함수로도 사용 가능하다. GDAL 라이브러리와 연동되어 있는 sf와 같은 R 패키지가 이러한 C API 알고리즘을 사용한다면, 사용자는 R 패키지 외에 추가로 GDAL 바이너리 명령줄 유틸리티를 설치할 필요가 없다는 의미이다.

sf 패키지는 gdal\_utils 함수를 통해 이러한 C API 알고리즘을 호출할 수 있으며, 첫 번째 인수는 gdal 접두사를 제거한 해당 유틸리티의 이름이다.

* info: GDAL(래스터) 데이터셋에 대한 정보를 출력한다.
* warp: 래스터를 새로운 래스터로 변환한다(CRS의 전환 포함).
* rasterize: 벡터 데이터셋을 래스터화한다.
* translate: 래스터 파일을 다른 형식으로 변환한다.
* vectortranslate: 벡터 파일을 다른 형식으로 변환한다(ogr2ogr에 해당).
* buildvrt: 가상 래스터 타일(여러 파일의 결합을 통해 생성된 단일 래스터)을 생성한다.
* demprocessing: DEM(digital data model, 디지털고도모델)에 대한 다양한 프로세싱을 수행한다.
* nearblack: 거의 검정색/흰색 경계를 검정색으로 변환한다.
* grid: 흩어진 데이터로부터 규칙 그리드를 생성한다.
* mdiminfo: 다차원 어레이에 대한 정보를 출력한다.
* mdimtranslate: 다차원 어레이를 다른 형식으로 변환한다.

이러한 유틸리티는 기본적으로 파일에서 작동하며, sf 또는 stars 객체에 직접적으로 작용하지는 않는다. 그러나 stars\_proxy 객체는 본질적으로 파일에 대한 포인터이며, 다른 객체들도 파일로 기록될 수 있다. 이러한 유틸리티 중 몇몇은 (항상 혹은 선택적으로) st\_mosaic, st\_warp, 또는 st\_write와 같은 함수를 통해 사용된다. R의 **gdalUtilities** (O’Brien 2022) 패키지는 sf::gdal\_utils에 덧붙여 사용자 편의성을 위한 래퍼 함수를 제공하는데, 함수의 아규먼트 이름이 바이너리 유틸리티의 명령줄 아규먼트와 정확히 일치한다.

## 7.5 벡터 데이터 큐브의 예제

### 7.5.1 예제: 대기질 시계열 데이터에 대한 애그리게이션 실행

유럽 대기질 데이터를 사례로 벡터 데이터 큐브에 대한 애그리게이션 작업을 설명한다. 동일한 데이터가 Gräler, Pebesma, Heuvelink (2016)에서 사용되었으며, 12장과 13장에서도 사용될 예정이다. 독일의 농촌 지역 관측소의 1998~2009년 데이터로부터 일평균 값을 계산하였다.

air 데이터 매트릭스, 날짜 벡터인 dates, 그리고 SpatialPoints 객체인 stations을 결합해 stars 객체를 생성할 수 있다.

load("data/air.rda") # this loads several datasets in .GlobalEnv  
dim(air)  
# space time   
# 70 4383  
stations |>  
 st\_as\_sf(coords = c("longitude", "latitude"), crs = 4326) |>  
 st\_geometry() -> st  
d <- st\_dimensions(station = st, time = dates)  
(aq <- st\_as\_stars(list(PM10 = air), dimensions = d))  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# PM10 0 9.92 14.8 17.7 22 274 157659  
# dimension(s):  
# from to offset delta refsys point  
# station 1 70 NA NA WGS 84 TRUE  
# time 1 4383 1998-01-01 1 days Date FALSE  
# values  
# station POINT (9.59 53.7),...,POINT (9.45 49.2)  
# time NULL

그림 7.11에서 시간 시계열이 상당히 길지만, 큰 결측값 간격도 있다는 것을 알 수 있다. 그림 7.12는 평균 값과 함께 측정소의 공간 분포를 보여준다.

|  |
| --- |
| 그림 7.10: 시간과 스테이션별로 계산된 PM10 값에 대한 시공간 다이어그램 |

|  |
| --- |
| 그림 7.11: 관측 스테이션별 PM10 평균값 |

간단한 실습 차원에서, 측정 스테이션별 시간 시계열 데이터를 지역 평균으로 집계할 수 있다. 이를 위해 stars 객체에 대한 aggregate 메소드를 사용한다.

(a <- aggregate(aq, de\_nuts1, mean, na.rm = TRUE))  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# PM10 1.08 10.9 15.3 17.9 21.8 172 25679  
# dimension(s):  
# from to offset delta refsys point  
# geom 1 16 NA NA WGS 84 FALSE  
# time 1 4383 1998-01-01 1 days Date FALSE  
# values  
# geom MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# time NULL

그리고, 아래의 코드를 통해 임의로 선택한 여섯 개의 날짜에 대한 지도를 보여줄 수 있다(그림 7.13).

library(tidyverse)  
a |> filter(time >= "2008-01-01", time < "2008-01-07") |>   
 plot(key.pos = 4)

|  |
| --- |
| 그림 7.12: 임의의 6일에 대한 지역 평균 PM10 |

또한 다음의 코드를 이용해 단일 주에 대한 평균 값의 시계열 플롯을 생성할 수 있다(그림 7.14).

library(xts) |> suppressPackageStartupMessages()  
plot(as.xts(a)[,4], main = de\_nuts1$NAME\_1[4])

|  |
| --- |
| 그림 7.13: 단일 스테이션에 대한 지역 평균 PM10의 시계열 |

### 7.5.2 예제: 브리스톨 출발지-도착지 데이터 큐브

이 예제에 사용된 데이터는 Lovelace, Nowosad, Muenchow (2019)에서 가져온 것으로, 출발지-목적지(OD) 매트릭스이다. 구체적으로 A 지역에서 B 지역으로 이동하는 사람의 수를 교통 수단별로 나타낸 것이다. 102개의 지역에 대한 피처 지오메트리는 sf 객체인 bristol\_zones에 포함되어 있다.

|  |
| --- |
| 그림 7.14: 영국 브리스톨의 102개 구역 현황(33번 구역(E02003043)이 빨간색으로 표시되어 있음) |

bristol\_od 테이블에 데이터가 저장되어 있는데, OD 쌍(이동량이 0인 경우는 제외)가 레코드로, 서로 다른 교통수단이 변수로 들어가 있다.

head(bristol\_od)  
# # A tibble: 6 × 7  
# o d all bicycle foot car\_driver train  
# <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
# 1 E02002985 E02002985 209 5 127 59 0  
# 2 E02002985 E02002987 121 7 35 62 0  
# 3 E02002985 E02003036 32 2 1 10 1  
# 4 E02002985 E02003043 141 1 2 56 17  
# 5 E02002985 E02003049 56 2 4 36 0  
# 6 E02002985 E02003054 42 4 0 21 0

제외된 무이동 OD 쌍의 갯수는 모든 OD 조합의 수에서 데이터에 포함된 OD 쌍의 갯수를 빼면 구할 수 있다.

nrow(bristol\_zones)^2 - nrow(bristol\_od)   
# [1] 7494

우리는 출발지, 목적지, 교통 수단을 차원으로 가지는 3차원 벡터 데이터 큐브를 형성할 것이다. 이를 위해 먼저 pivot\_longer를 사용하여 bristol\_od 테이블을 정리하여 출발지(o), 목적지(d), 교통 수단(mode), 및 수(count)라는 변수를 갖도록 한다.

# create O-D-mode array:  
bristol\_tidy <- bristol\_od |>   
 select(-all) |>   
 pivot\_longer(3:6, names\_to = "mode", values\_to = "n")  
head(bristol\_tidy)  
# # A tibble: 6 × 4  
# o d mode n  
# <chr> <chr> <chr> <dbl>  
# 1 E02002985 E02002985 bicycle 5  
# 2 E02002985 E02002985 foot 127  
# 3 E02002985 E02002985 car\_driver 59  
# 4 E02002985 E02002985 train 0  
# 5 E02002985 E02002987 bicycle 7  
# 6 E02002985 E02002987 foot 35

그리고 나서 0으로 채워진 3차원 어레이 a를 생성한다.

od <- bristol\_tidy |> pull("o") |> unique()  
nod <- length(od)  
mode <- bristol\_tidy |> pull("mode") |> unique()  
nmode = length(mode)  
a = array(0L, c(nod, nod, nmode),   
 dimnames = list(o = od, d = od, mode = mode))  
dim(a)  
# [1] 102 102 4

해당 어레이의 세 차원에 구역 이름(o, d)과 교통 수단 이름(mode)이 부여된 것을 확인할 수 있다. 이렇게 함으로써 bristol\_tidy의 각 행은 해당 어레이의 한 단위(엔트리)에 해당하게 된다. bristol\_tidy 테이블에 있는 인덱스(o, d 및 mode)와 값(n)을 이용해 해당 어레이(a)의 비제로 부분을 채울 수 있다.

a[as.matrix(bristol\_tidy[c("o", "d", "mode")])] <-   
 bristol\_tidy$n

bristol\_zones의 구역과 bristol\_tidy의 구역 이름 간에 순서 불일치가 발생할 수 있기 때문에 다음의 조치를 취한다.

order <- match(od, bristol\_zones$geo\_code)  
zones <- st\_geometry(bristol\_zones)[order]

순서가 이미 올바른 경우도 있지만, 이러한 가정을 배제한 채 위의 코드를 실행하는 것이 좋다. 다음으로 존과 교통수단을 이용해 stars 디멘션 객체를 생성한다.

library(stars)  
(d <- st\_dimensions(o = zones, d = zones, mode = mode))  
# from to refsys point values  
# o 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# d 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# mode 1 4 NA FALSE bicycle,...,train

a와 d로부터 최종적으로 stars 객체를 생성한다.

(odm <- st\_as\_stars(list(N = a), dimensions = d))  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# N 0 0 0 4.8 0 1296  
# dimension(s):  
# from to refsys point values  
# o 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# d 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# mode 1 4 NA FALSE bicycle,...,train

이 3차원 어레이로부터 단일 슬라이스를 추출할 수 있다. 예를 들어 구역 33에 대한 데이터를 odm[,,33]을 통해 추출한 후 플롯을 그릴 수 있다(그림 7.16).

plot(adrop(odm[,,33]) + 1, logz = TRUE)

|  |
| --- |
| 그림 7.15: 33번 존에 대한 OD 데이터를 추출한 후 교통수단별로 지도화하였다. |

이렇게 하여 서브셋을 생성하면, 첫 번째 아규먼트가 비어 있으므로 모든 속성(여기서는 하나만 존재함: N)을 선택하고, 두 번째 아규먼트가 비어 있어 모든 출발지를 선택하며, 세 번째 아규먼트로 목적지 구역 33을 선택하고, 네 번째 아규먼트는 비어 있어 모든 교통 수단을 선택하게 된다.

우리가 이 특정 구역을 목적지로 선택한 이유는 해당 구역이 가장 많은 여행자를 가지고 있기 때문이다. 이는 목적지별로 모든 출발지와 여행 수단을 합산하여 확인할 수 있다.

d <- st\_apply(odm, 2, sum)  
which.max(d[[1]])  
# [1] 33

다른 애그리게이션도 실행할 수 있다. 예를 들어 OD(102 x 102)의 총통행량을 다음과 같이 구할 수 있다.

st\_apply(odm, 1:2, sum)  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# sum 0 0 0 19.2 19 1434  
# dimension(s):  
# from to refsys point values  
# o 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# d 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...

교통 수단별로 출발지 총계를 구할 수 있다.

st\_apply(odm, c(1,3), sum)  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# sum 1 57.5 214 490 771 2903  
# dimension(s):  
# from to refsys point values  
# o 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# mode 1 4 NA FALSE bicycle,...,train

교통 수단별로 도착지 총계를 구할 수 있다.

st\_apply(odm, c(2,3), sum)  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# sum 0 13 104 490 408 12948  
# dimension(s):  
# from to refsys point values  
# d 1 102 WGS 84 FALSE MULTIPOLYGON (...,...,MULTIPOLYGON (...  
# mode 1 4 NA FALSE bicycle,...,train

모드별로 합산된 출발지 총계를 구할 수 있다.

o <- st\_apply(odm, 1, sum)

모드별로 합산된 도착지 총계를 구할 수 있다.

d <- st\_apply(odm, 2, sum)

o와 d를 결합한 후 함께 플롯할 수 있다(그림 7.17).

x <- (c(o, d, along = list(od = c("origin", "destination"))))  
plot(x, logz = TRUE)

|  |
| --- |
| 그림 7.16: 출발지별 총 통근(왼쪽) 또는 목적지별 총 통근(오른쪽) |

이 지도가 현상의 본질을 호도할 수 있다는 우려가 있을 수 있다. 왜냐하면 실질적인 값의 크기(컬러) 뿐만 아니라 구역의 면적 역시 시각적으로 느끼는 양의 크기에 영향을 주기 때문이다. 이러한 점을 감안하여 밀도값(카운트/)을 계산하여 나타낼 수 있다(그림 7.18)(역자주: 지도학적 원칙으로 보자면, 카운트를 코로플레스 맵으로 나타내는 것은 적절하지 않다. 따라서 보다 좋은 방법은 카운트를 도형표현도로 나타내는 것이다).

library(units)  
a <- set\_units(st\_area(st\_as\_sf(o)), km^2)  
o$sum\_km <- o$sum / a  
d$sum\_km <- d$sum / a  
od <- c(o["sum\_km"], d["sum\_km"], along =   
 list(od = c("origin", "destination")))  
plot(od, logz = TRUE)

|  |
| --- |
| 그림 7.17: 출발지별 총 통근 밀도(왼쪽) 또는 목적지별 총 통근 밀도(오른쪽) |

카운트를 정규화하는 또 다른 방법은 값을 면적이 아닌 인구수로 나누는 것이다.

### 7.5.3 타이디 어레이 데이터

Wickham(2014)의 타이디 데이터 논문은 3차원 데이터를 어레이 데이터 형식보다는 각 행이 (지역, 클래스, 연도, 값)으로 구성된 긴(비정규화된) 테이블 형식으로 처리되는 것이 더 좋다고 제안한다. 이는 가능하기만 하다면 늘 좋은 방법이다. 그러나 이 옵션이 불가능한 경우가 있는데, 기본 처리 및 저장의 목적에서 그러하다. 그 이유는 다음과 같다.

* 많은 어레이 데이터는 처음부터 어레이 데이터로 수집되거나 생성된다. 예를 들어, 원격탐사를 통해 수집된 데이터 혹은 기후 모델을 통해 생성된 데이터
* 어레이 형식을 긴 테이블 형태로 전환하는 것이 그 역보다 훨씬 쉽다.
* 긴 테이블 형식의 데이터가 훨씬 더 많은 메모리를 요구한다. 왜냐하면 가 디멘션 의 기수(크기)일 때, 디멘션 값이 차지하는 메모리 공간은 가 아니라 로 주어지기 때문이다.
* 결측값이 있는 셀이 삭제되면, 긴 테이블 형식은 어레이 형식에 내재된 인덱싱을 상실하게 된다.

이 주장을 극단적으로 표현하자면, 모든 이미지, 비디오 및 음성 데이터가 어레이 형식으로 저장된다고 가정해 보자. 실제로 이를 긴 테이블 형식으로 저장해야 한다고 주장하는 사람은 거의 없을 것이다. 그럼에도 불구하고 tsibble(Wang et al. 2022)과 같은 R 패키지는 긴 테이블 형식을 취하고 있는데, 동일한 시간 스텝을 가진 다수의 공간적 피처에 대해 순서를 매기는 것은 매우 모호한 작업임에도 불구하고 어쨋던 인덱싱을 해야한다는 문제점이 있을 수 있다. 이러한 문제는 stars에서 제공하는 어레이 형식을 사용함으로써 *자동*으로 해결된다. 물론 조밀한 어레이를 사용해야한다는 대가를 치루는 것이기도 하다.

stars 패키지는 어레이 집합을 처리하는 문제에 있어서는 타이디 데이터 원칙(tidy manifesto)을 따르려 하며, 특히 하나 이상의 차원이 공간 및/또는 시간을 참조하는 경우에 대해서는 특별히 더 그러하다.

### 7.5.4 벡터 데이터 큐브를 위한 파일 포멧

정규 테이블 형식(긴 테이블 형식을 포함)은 하나의 대안이지만 사용하기에는 불편하다. 위의 출발지-목적지 데이터 예제와 13장에서 다를 내용은 테이블 형식에서 벡터 데이터 큐브를 재구성하는 것이 매우 복잡하다는 점을 잘 보여준다. NetCDF나 Zarr와 같은 어레이 형식은 어레이 데이터를 저장하기 위해 설계되었다. 그러나 이러한 형식은 모든 데이터 구조에 사용할 수 있으며, 일단 작성된 파일은 재사용하기 어려운 위험이 있다. 포인트, (멀티)라인스트링 또는 (멀티)폴리곤으로 구성된 단일 지오메트리 디멘션을 가진 벡터 큐브의 경우, CF 규칙(Eaton et al. 2022)은 이러한 지오메트리를 인코딩하는 방법을 설명한다. stars::read\_mdim과 stars::write\_mdim은 이러한 규칙을 따르는 벡터 데이터 큐브를 읽고 쓸 수 있다.

## 7.6 래스터-벡터 전환과 벡터-래스터 전환

1.3절에서 래스터-벡터 변환과 벡터-래스터 변환에 대한 몇 가지 예제를 이미 다루었다. 이 절에서는 코드와 예제를 추가하고자 한다.

### 7.6.1 벡터-래스터 전환

st\_as\_stars 함수는 객체를 stars 객체로 변환하는 메소드로로 설계되었다. 그러나 모든 stars 객체가 래스터 객체인 것은 아니며, sf 객체에 이 메소들 적용하면 지오메트리를 공간적(벡터) 디멘션으로 하고 속성을 속성 디멘션으로 가지는 벡터 데이터 큐브가 생성된다. 피처 *지오메트리*(sfc) 객체가 주어지면, st\_as\_stars 함수는 이를 래스터한다(섹션 7.8과 그림 7.19).

file <- system.file("gpkg/nc.gpkg", package="sf")  
read\_sf(file) |>   
 st\_geometry() |>  
 st\_as\_stars() |>  
 plot(key.pos = 4)

|  |
| --- |
| 그림 7.18: st\_as\_stars 함수를 활용한 백터 지오메트리의 래스터화 |

st\_as\_stars 함수는 셀 크기, 셀 수 및/또는 범위를 제어하는 파라미터를 설정할 수 있다. 반환되는 셀 값은 지오메트리 밖에 중심점이 있는 셀의 경우는 0이고, 지오메트리 내부 또는 경계에 중심점이 있는 셀의 경우는 1이다. 기존 피처를 래스터화하는 것은 st\_rasterize 함수를 사용하여 수행되며, 이는 그림 1.5에서도 볼 수 있다.

library(dplyr)  
read\_sf(file) |>  
 mutate(name = as.factor(NAME)) |>  
 select(SID74, SID79, name) |>  
 st\_rasterize()  
# stars object with 2 dimensions and 3 attributes  
# attribute(s):  
# SID74 SID79 name   
# Min. : 0 Min. : 0 Sampson : 655   
# 1st Qu.: 3 1st Qu.: 3 Columbus: 648   
# Median : 5 Median : 6 Robeson : 648   
# Mean : 8 Mean :10 Bladen : 604   
# 3rd Qu.:10 3rd Qu.:13 Wake : 590   
# Max. :44 Max. :57 (Other) :30952   
# NA's :30904 NA's :30904 NA's :30904   
# dimension(s):  
# from to offset delta refsys point x/y  
# x 1 461 -84.3 0.0192 NAD27 FALSE [x]  
# y 1 141 36.6 -0.0192 NAD27 FALSE [y]

라인과 포인트 지오메트리고 이와 유사하게 래스터화할 수 있다(그림 7.20).

read\_sf(file) |>  
 st\_cast("MULTILINESTRING") |>  
 select(CNTY\_ID) |>  
 st\_rasterize() |>  
 plot(key.pos = 4)

|  |
| --- |
| 그림 7.19: 노스캐롤라이나 카운티 경계를 래스터로 전환하기 |

## 7.7 좌표변환 및 좌표전환

### 7.7.1 st\_crs 함수

sf 또는 stars 클래스의 공간 객체는 CRS(좌표참조계)를 포함하고 있다. st\_crs 함수를 사용하여 해당 객체의 CRS를 확인하거나 다른 CRS로 교체할 수 있다. 또한, st\_set\_crs함수를 사용하여 CRS를 설정하거나 교체할 수 있다. CRS는 EPSG 코드로 설정할 수 있으며, 예를 들어 st\_crs(4326)는 st\_crs('EPSG:4326')로 변환된다. 또는 “+proj=utm +zone=25 +south”와 같은 PROJ.4 문자열, “WGS84”와 같은 이름, 또는 “OGC”와 같이 기관명이 앞에 붙은 이름으로 설정할 수 있다. 대안으로는 WKT, WKT-2(섹션 2.5) 또는 PROJJSON 형식의 CRS 정의가 있다. st\_crs 함수에 의해 반환된 객체는 다음의 두 개의 필드를 포함한다.

* wkt: WKT-2 형식으로 표현된 CRS
* input: 사용자 입력(있는 경우), 또는 CRS에 대한 인간가독설명(가능한 경우)

PROJ.4 문자열은 일부 CRS을 정의하는 데 사용할 수 있지만, CRS를 대표하는 데는 사용할 수 없다. 예를 들어, crs 객체의 WKT-2를 $proj4string 메서드를 사용하여 proj4string으로 전환하려면 다음과 같이 할 수 있다.

x <- st\_crs("OGC:CRS84")  
x$proj4string  
# [1] "+proj=longlat +datum=WGS84 +no\_defs"

그런데, 이 과정이 성공적으로 이루어졌다 하더라도, 보통 정보의 손실이 수반되고 가역적으로 전환되지 않는다. PROJ.4 문자열을 CRS를 정의하기 위해 사용하는 경우(예를 들어 파라미터가 지정되어 있는 투영 CRS), 투영 CRS가 WGS84 데이텀과 관련되는 한은 문제가 없다.

### 7.7.2 st\_transform함수와 st\_project 함수

sf 또는 stars 객체에 대한 좌표 변환이나 좌표 전환은 st\_transform함수를 사용하여 수행된다. 이 함수의 첫 번째 아규먼트는 CRS가 설정된 sf 또는 stars 클래스의 공간 객체이고, 두 번째 아규먼트는 crs 객체(또는 st\_crs 함수로 변환 가능한 값)이다. 소스 crs에서 타깃 crs로 변환 혹은 전환할 수 있는 방법이 여러 가지인 경우, PROJ는 가장 높은 명시(declared) 정확도를 가진 방법을 선택한다. 더 세밀한 옵션은 7.7.5에서 설명된다. 규칙 래스터 디멘션을 가진 stars 객체의 경우, st\_transform 함수는 좌표만 변환하며 항상 곡선형 그리드를 생성한다. 새로운 CRS에서 규칙 래스터를 생성하려면 재그리딩(regridding)을 사용하는 st\_warp 함수를 활용하면 된다(Section 7.8).

sf나 stars 객체가 아닌 상황에서의 좌표 변환 혹은 좌표 전환은 저수준(lower-level) 함수인 sf\_project를 통해 이루어진다. 이 함수는 좌표가 담긴 행렬과 소스 및 타깃 CRS(crs)를 입력으로 받아, 변환된 혹은 전환된 좌표를 반환한다.

### 7.7.3 sf\_proj\_info 함수

sf\_proj\_info 함수는 PROJ 소프트웨어에서 사용 가능한 투영, 타원체, 단위 및 본초 자오선에 대한 정보를 조회하는 데 사용된다. 이 함수는 단일 매개변수 type을 받으며, 이 매개변수는 다음과 같은 값을 가질 수 있다.

* type = "proj": 사용 가능한 투영법의 짧은 이름과 긴 이름을 나열한다. 짧은 이름은 “+proj=name” 문자열에서 사용할 수 있다.
* type = "ellps": 사용 가능한 타원체를 나열하며, 이름, 긴 이름 및 타원체의 파라미터 정보를 포함한다.
* type = "units": 사용 가능한 길이 단위를 나열하고, 미터로의 변환 상수 정보를 포함한다.
* type = "prime\_meridians": 본초 자오선을 나열하고, 그리니치 자오선과의 상대적 위치 정보를 포함한다.

### 7.7.4 데이텀 그리드, proj.db, cdn.proj.org. 로컬 캐쉬

데이텀 그리드(2.4절)는 로컬에 설치하거나 PROJ 측량 그리드 CDN(https://cdn.proj.org/)에서 불러 올 수 있다. 로컬에 설치된 경우, 데이텀 그리드는 PROJ 검색 경로를 통해 불러올 수 있는데, 이 경로는 다음과 같이 표시된다.

sf\_proj\_search\_paths()  
# [1] "/home/edzer/.local/share/proj" "/usr/share/proj"

핵심 PROJ 데이터베이스는 proj.db로, 일반적으로 다음 위치에서 불러올 수 있는 sqlite3 데이터베이스이다.

paste0(tail(sf\_proj\_search\_paths(), 1), .Platform$file.sep,   
 "proj.db")  
# [1] "/usr/share/proj/proj.db"

각 PROJ 릴리스에 포함된 EPSG 데이터베이스 스냅샷의 버전은 proj.db의 "metadata" 테이블에 명시되어 있으며, **sf**에서 사용되는 PROJ 런타임의 버전은 다음과 같이 표시된다.

sf\_extSoftVersion()["PROJ"]  
# PROJ   
# "9.1.1"

특정 좌표 변환에 필요한 데이텀 그리드가 로컬에 없을 경우, PROJ는 PROJ CDN의 온라인 데이텀 그리드를 검색하게 된다. 단 아래의 결과가 TRUE인 경우이다.

sf\_proj\_network()  
# [1] FALSE

디폴트는 FALSE로 설정되어 있지만, 이를 TRUE로 바꾸면 해당 네트워크 리소스의 URL을 반환한다. 이 리소스는 더 빠르거나 제한이 덜한 다른 리소스로 설정할 수도 있다.

sf\_proj\_network(TRUE)  
# [1] "https://cdn.proj.org"

CDN에서 데이텀 그리드를 조회한 후, PROJ는 조회된 그리드의 *일부분*(디폴트는 전체 그리드가 아님)을 로컬 캐시에 기록한다. 이 캐시는 사용자 디렉토리의 또 다른 sqlite3 데이터베이스로 다음과 같이 표시된다.

list.files(sf\_proj\_search\_paths()[1], full.names = TRUE)  
# [1] "/home/edzer/.local/share/proj/cache.db"

차후의 데이텀 그리드 조회는 이 데이터베이스에 대해 우선적으로 이루어진다.

### 7.7.5 변환 파이프라인

내부적으로 PROJ는 소스 CRS에서 타겟 CRS로 가는 오퍼레이션 시퀀스를 나타내기 위해 이른바 *좌표 오퍼레이션 파이프라인*(coordinate operation pipeline)을 사용한다. 소스에서 타겟으로 가는 여러 옵션이 있을 경우, st\_transform 함수는 가장 높은 정확도를 가진 옵션을 선택한다. 사용 가능한 옵션을 조회하려면 sf\_proj\_pipelines 함수를 사용하면 된다.

(p <- sf\_proj\_pipelines("OGC:CRS84", "EPSG:22525"))  
# Candidate coordinate operations found: 5   
# Strict containment: FALSE   
# Axis order auth compl: FALSE   
# Source: OGC:CRS84   
# Target: EPSG:22525   
# Best instantiable operation has accuracy: 2 m  
# Description: axis order change (2D) + Inverse of Corrego Alegre  
# 1970-72 to WGS 84 (2) + UTM zone 25S  
# Definition: +proj=pipeline +step +proj=unitconvert +xy\_in=deg  
# +xy\_out=rad +step +inv +proj=hgridshift  
# +grids=br\_ibge\_CA7072\_003.tif +step  
# +proj=utm +zone=25 +south +ellps=intl

해당 변환에서 가장 높은 정확도를 보이는 파이프라인이 요약되어 있고 특정한 데이텀 그리드의 사용이 지정되어 있음을 알 수 있다. 만약 네트워크 검색을 활성화하지 않았다면, 다른 결과를 얻었을 것이다.

sf\_proj\_network(FALSE)  
# character(0)  
sf\_proj\_pipelines("OGC:CRS84", "EPSG:22525")  
# Candidate coordinate operations found: 5   
# Strict containment: FALSE   
# Axis order auth compl: FALSE   
# Source: OGC:CRS84   
# Target: EPSG:22525   
# Best instantiable operation has accuracy: 2 m  
# Description: axis order change (2D) + Inverse of Corrego Alegre  
# 1970-72 to WGS 84 (2) + UTM zone 25S  
# Definition: +

이 경우에는 데이텀 그리드에 대한 사항이 누락되어 있음을 볼 수 있다. sf\_proj\_pipelines이 반환하는 객체는 서브클래스화된 데이터 프레임으로, 다음과 같은 열을 포함한다.

names(p)  
# [1] "id" "description" "definition" "has\_inverse"   
# [5] "accuracy" "axis\_order" "grid\_count" "instantiable"  
# [9] "containment"

예를 들어 다음과 같이 정확도를 나열할 수 있다.

p |> pull(accuracy)  
# [1] 2 5 5 8 NA

여기서 NA는 “대략적인 정확도”를 의미하며, 이는 30~120m 범위 내의 한 값을 가진다.

p |> filter(is.na(accuracy))  
# Candidate coordinate operations found: 1   
# Strict containment: FALSE   
# Axis order auth compl: FALSE   
# Source: OGC:CRS84   
# Target: EPSG:22525   
# Best instantiable operation has only ballpark accuracy   
# Description: Ballpark geographic offset from WGS 84 (CRS84) to  
# Corrego Alegre 1970-72 + UTM zone 25S  
# Definition: +proj=pipeline +step +proj=unitconvert +xy\_in=deg  
# +xy\_out=rad +step +proj=utm +zone=25  
# +south +ellps=intl

st\_transform 함수가 선택한 가장 정확한 파이프라인이 디폴트이지만, pipeline 아규먼트를 지정하면 결과를 바꿀 수도 있다. 이 경우 p$definition에 있는 옵션 중 하나를 선택하면 된다.

### 7.7.6 축 순서와 방향

2.5절에서 언급한 것처럼, EPSG:4326은 첫 번째 축이 위도와 관련되고 두 번째 축이 경도와 관련되도록 정의한다. 이것은 수많은 다른 타원체 CRS에서도 마찬가지이다. 이러한 방식은 해당 기관(EPSG)이 규정한 것이지만, 현재 대부분의 데이터셋은 이러한 방식으로 저장되지 않는다. 대부분의 다른 소프트웨어와 마찬가지로, **sf** 패키지는 이를 무시하고 디폴트로 타원체 좌표 쌍을 (경도, 위도)로 해석한다. 하지만 해당 기관의 규정을 준수하는 데이터 원천(예를 들어 WFS 서비스)이 생산한 데이터를 읽어야 할 경우 다음과 같이 지정할 수 있다.

st\_axis\_order(TRUE)

이렇게 하면 sf가 GDAL과 PROJ 루틴을 호출할 때 규정 준수(위도, 경도 순서)가 항상 전제되게 할 수 있다. 이러한 규정 준수에 많은 문제가 발생할 수 있다고 예상되며, 예를 들어 데이터를 플로팅할 때 문제가 드러난다. sf 객체를 위한 플롯 메소드는 축 순서 규정을 준수하며 플로팅 전에 변환 파이프라인 "+proj=pipeline +step +proj=axisswap +order=2,1"을 사용하여 위경도의 순서를 바꾸지만, ggplot2 패키지의 geom\_sf는 이러한 수정 과정을 거치지 않는다. 앞서 언급한 바와 같이, EPSG:4326에서 발견되는 축 순서 상의 모호성은 OGC:CRS84로 교체하면 모두 해결된다.

축의 순서와는 다른 문제로, 모든 CRS가 북쪽과 동쪽 방향을 양의 값으로 지정하지는 않는다는 점 또한 매우 중요하다. R의 대부분의 플로팅 함수는 위와는 반대로 정의된 축을 가진 데이터와는 제대로 작동하지 않는다. 축의 방향과 단위에 대한 정보는 다음과 같이 검색할 수 있다.

st\_crs(4326)$axes  
# name orientation  
# 1 Geodetic latitude 1  
# 2 Geodetic longitude 3  
st\_crs(4326)$ud\_unit  
# 1 [°]  
st\_crs("EPSG:2053")$axes  
# name orientation  
# 1 Westing 4  
# 2 Southing 2  
st\_crs("EPSG:2053")$ud\_unit  
# 1 [m]

## 7.8 래스터 변환 및 워프

래스터 데이터셋에 대해 st\_transform 함수를 사용할 때는 다음과 같이 할 수 있다.

tif <- system.file("tif/L7\_ETMs.tif", package = "stars")  
read\_stars(tif) |>  
 st\_transform('OGC:CRS84')  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# L7\_ETMs.tif 1 54 69 68.9 86 255  
# dimension(s):  
# from to refsys point values x/y  
# x 1 349 WGS 84 FALSE [349x352] -34.9,...,-34.8 [x]  
# y 1 352 WGS 84 FALSE [349x352] -8.04,...,-7.95 [y]  
# band 1 6 NA NA NULL   
# curvilinear grid

이제 곡선형 그리드가 생성되는 것을 볼 수 있으며, 이는 새로운 CRS에 의거해 모든 그리드 셀의 좌표가 재계산되므로 더 이상 규칙 그리드로 존재할 수 없음을 의미한다. 이러한 데이터를 플로팅하면 속도가 극단적으로 느린데, 각 그리드 셀에 대해 작은 폴리곤이 우선적으로 계산되고 그 후에 플로팅되기 때문이다. 이것의 이점은 정보가 손실되지 않는다는 것으로 투영 이후에도 그리드 셀의 값은 동일하게 유지된다.

규칙 그리드를 투입하여 새로운 CRS에서도 규칙 그리드를 산출하려면 워프(warp) 오퍼레이션을 적용해야 한다. 즉, 새로운 위치에 그리드를 재생성하고 새로운 그리드 셀에 값을 할당하는 특정한 규칙을 사용해야 한다. 이러한 규칙에 가장 가까운 값을 사용하거나 어떤 형태의 보간법을 사용하는 것이 포함될 수 있다. 이 오퍼레이션은 정보의 손실이 발생하며 불가역적이다.

워프를 수행하는 가장 좋은 방법은 타깃 그리드를 stars 객체로 지정하는 것이다. 타깃 CRS만 지정할 경우, 문제에 전혀 적합하지 않은 기본 옵션이 선택될 수 있다. 타깃 CRS만 사용하는 예시 워크플로우는 다음과 같다.

read\_stars(tif) |>  
 st\_warp(crs = st\_crs('OGC:CRS84')) |>  
 st\_dimensions()  
# from to offset delta refsys x/y  
# x 1 350 -34.9 0.000259 WGS 84 [x]  
# y 1 352 -7.95 -0.000259 WGS 84 [y]  
# band 1 6 NA NA NA

이는 상당한 근사 래스터를 생성하지만, 변환도 상대적으로 작다. 소스 래스터와 정확히 동일한 행과 열을 갖는 타깃 래스터를 먼저 생성하려는 워크플로우의 경우 다음과 같이 하면 된다.

r <- read\_stars(tif)  
grd <- st\_bbox(r) |>  
 st\_as\_sfc() |>  
 st\_transform('OGC:CRS84') |>  
 st\_bbox() |>  
 st\_as\_stars(nx = dim(r)["x"], ny = dim(r)["y"])  
st\_warp(r, grd)  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# L7\_ETMs.tif 1 54 69 68.9 86 255 6180  
# dimension(s):  
# from to offset delta refsys x/y  
# x 1 349 -34.9 0.00026 WGS 84 [x]  
# y 1 352 -7.95 -0.000259 WGS 84 [y]  
# band 1 6 NA NA NA

여기서 와 의 방향의 그리드 해상도가 조금 달라진 점을 확인할 수 있다.

## 7.9 연습문제

R을 사용하여 다음 연습문제를 해결하시오.

1. nc 카운티 중 LINESTRING(-84 35, -78 35)와 교차하는 카운티의 이름을 찾아보라. 이를 위해 [를 사용하고, 대안으로 st\_join 함수를 사용하라.
2. sf\_use\_s2(FALSE)를 설정한 후에 이 작업을 반복하고, 차이를 *계산*하라(힌트: setdiff를 사용). 차이가 나타나는 카운티는 색상 ’#88000088’로 채색하라.
3. 두 지점 사이의 직선과 대권을 하나의 플롯에 그려보라. 현재 사용 중인 투영법에서는 R은 직선을 항상 직선으로 그린다는 점을 명심하라. st\_segmentize 함수를 사용하여 직선 상이나 타원 좌표의 대원(큰 원) 상에 포인트를 추가하라.
4. NDVI(정규화 차이 식생 지수)는 (NIR-R)/(NIR+R)로 계산되며, 여기서 NIR은 근적외선 밴드, R은 빨간색 밴드이다. L7\_ETMs.tif 파일을 객체 x에 읽고, split(x, "band")를 사용하여 밴드 차원을 속성으로 분리하라. 그런 다음 NIR(밴드 4)과 R(밴드 3) 속성을 직접 사용하는 표현식을 사용하여 이 객체에 NDVI 속성을 추가하라.
5. L7\_ETMs.tif 이미지에 대해 밴드 차원을 축소하여 NDVI를 계산하라. 이를 위해 st\_apply 함수와 ndvi = function(x) { (x[4]-x[3])/(x[4]+x[3]) } 함수를 사용하라. 결과를 플로팅하고, 결과를 GeoTIFF 형식으로 저장하라.
6. L7\_ETMs.tif에서 읽은 stars 객체를 st\_transform 함수를 사용하여 OGC:CRS84로 변환하라. 객체를 출력하라. 이것이 규칙 그리드인가? 첫 번째 밴드를 axes=TRUE 및 border=NA 아규먼트를 사용하여 플로팅하고, 왜 이렇게 시간이 오래 걸리는지 설명하라.
7. L7\_ETMs.tif 객체를 st\_warp 함수를 사용하여 OGC:CRS84로 변환하고, 결과 객체를 axes=TRUE로 플로팅하라. 왜 st\_transform 함수에 비해 플롯이 훨씬 더 빨리 생성되는지 설명하라.
8. 래스터 L7\_ETMs의 벡터 재현을 사용하여 POINT(293716 9113692)를 중심으로 반지름 75m인 원형 영역과의 인터센션을 플로팅하고, 이 원의 면적-가중 평균 픽셀 값을 계산하라. 벡터 데이터를 사용하여 애그리게이트한 값과 래스터 데이터를 사용하여 애그리게이트한 값을 (디폴트인 exact=FALSE 및 exact=TRUE 사용) 비교하라. 차이점을 설명하라.

# 8. 공간데이터의 플로팅

타임라인과 함께, 지도는 가장 강력한 그래프 중 하나일 것이다. 그건 아마도 지도 상의 공간과 현실 공간(현재 어디에 있고, 한때 어디에 있었는지)을 즉시 연관 지을 수 있기 때문일 것이다. 최근 발간된 시각화 관련 두 권의 책(Healy 2018; Wilke 2019)은 지리공간 데이터 또는 지도를 시각화하는 방법에 대한 장을 포함하고 있다. 여기에서는 어떤 지도가 좋고 나쁜지를 논하는 것이 아니라, 지도를 제작할 수 있는 다양한 가능성, 그 과정에서 마주할 수 있는 도전 과제들, 그리고 이를 해결할 수 있는 가능한 방법들을 살펴볼 것이다.

## 8.1 모든 지도는 투영법을 가지고 있다.

세상은 둥글지만, 플로팅 장치는 평평하다. 2.2.2절에서 언급했듯이, 세상을 평평한 장치에 어떤 방식으로든 플로팅한다는 것은 우리가 특정한 투영법을 적용하고 있는 것을 의미한다. 즉, 타원체 좌표를 특정한 방식으로 데카르트 좌표로 전환하고 있는 것이다. 이는 우리가 아무것도 하지 않았다고 생각하는 경우(역자주: 투영법을 적용하지 않았다고 생각하는 경우)(그림 8.1의 왼쪽)나, 우주에서 본 것처럼 세상을 “있는 그대로” 보여준다고 생각하는 경우(그림 8.1의 오른쪽)에도 마찬가지이다. 평면 상의 모든 지도는 투영법을 가지고 있다.

|  |
| --- |
| 그림 8.1: 국가 경계: 왼쪽은 경위도를 xy 좌표로 선형 변환한 것이고(플라트 카레 도법), 오른쪽은 무한 거리에서 지구를 바라본 것 같은 방식으로 나타낸 것(정사 도법)이다. |

왼쪽 지도는 다음의 코드로 그렸다.

library(sf)  
library(rnaturalearth)  
w <- ne\_countries(scale = "medium", returnclass = "sf")  
plot(st\_geometry(w))

타원 좌표로 지구 전체를 나타내고자 할 때, 해당 투영법이 디폴트 투영법이라는 점을 다음을 통해 알 수 있다.

st\_is\_longlat(w)  
# [1] TRUE

그림 8.1(왼쪽)에 사용된 투영법은 등장방형 도법(정거원통 도법)으로, 경도를 -축과 위도를 -축과 선형적으로 연결지어 나타냄으로써 지도의 가로세로의 거리 단위가 동일하게 유지되게 한 것이다(역자주: 지구의 동서 길이는 360도이고 남북 길이는 180도이므로, 이 지도의 가로세로비는 정확히 2:1로 나타난다). 그러므로 지구의 일부분에 대해 이 투영법을 적용하여 플롯할 경우, 지도의 동서 방향과 남북 방향의 거리 단위가 동일하게 유지되도록 플롯 비율을 선택해야 한다. 이는 비투영 sf 또는 stars 데이터셋에 대한 plot 메서드의 디폴트 동작이며, ggplot2::geom\_sf의 기본 설정이기도 하다(8.4절)(역자주: 비투영 객체는 경위도 좌표계를 가지고 있다는 것인데, 그것을 plot 메서드를 통해 플롯하게되면, 디폴트로 경위도 좌표를 마치 좌표처럼 나타낸다는 의미이다. 지구 전체에 대한 비투영 객체를 이러한 방식으로 플롯하면 시각적으로는 등장방형 도법과 동일하게 나타난다.).

플로팅 전에 투영법을 적용하여 데이터를 변형할 수도 있다. 예를 들어 독일을 플로팅하려고 한다면, 국가 경계를 불러온 후, st\_transform 함수를 사용하여 투영을 수행한다.

DE <- st\_geometry(ne\_countries(country = "germany",  
 returnclass = "sf"))  
DE |> st\_transform("+proj=eqc +lat\_ts=51.14 +lon\_0=90w") ->  
 DE.eqc

여기서 eqc는 PROJ의 “등장방형 도법”을 의미한다. lat\_ts는 투영 파라미터로 표준 위선(축척의 왜곡이 *없는* 위선)의 위치를 지정할 수 있다. 이 표준 위선 상에서 동서 방향과 남북 방향의 길이 단위가 동일해진다. 이 값은 지도의 바운딩 박스의 중간 지점에 해당한다.

# [1] 51.14

그림 8.2에 나타난 두 지도를 보면, 축의 값들을 제외하고는 두 지도는 동일하다는 점을 알 수 있다. 왼쪽 지도의 축 값은 타원 좌표(도 단위)이고, 오른쪽 지도의 축 값은 투영 좌표(데카르트 좌표)(미터 단위)이다.

|  |
| --- |
| 그림 8.2: 등장방형 도법이 적용된 독일. 왼쪽 지도의 단위는 도이고 정거원통 도법이 적용된 오른쪽 지도의 단위는 미터이다. |

### 8.1.1 데이터에 맞는 투영법 고르기

안타깝게도 여기에는 만병통치약이 없다. 모든 지점에서 모든 방향으로 축척이 동일하게 유지되는 투영법은 없다. 오직 지구본만이 이 속성을 보유한다. 널리 사용되고 있는 투영법들은 다음 중 하나를 보존하려고 한다:

* 면적: 정적 도법
* 형태: 정형 도법(예: *메르카토르 도법*)(역자주: 형태가 유지되기 위해서는 각도가 유지되어야 하기 때문에 정각 도법이라고도 부른다.)
* 거리의 일부 속성 (등장방형 도법은 모든 지점에 경선 방향으로 정거성을 보존하고, *정거방위* 도법은 투영 원점에서 모든 방향으로 거리를 보존한다.)

또는 어떤 도법은 주로 두 개 속성의 절충점을 찾으려고 한다(역자주: 주로 면적과 형태를 절충하는데, 이런 도법을 절충 도법이라 부르고 대표적인 것에 로빈슨 도법과 빈켈 트리펠 도법이 있다). 투영 패러미터는 지도의 중앙과 가장자리에 어떤 지역이 위치하는지, 어떤 지역이 위쪽에 있고 어떤 지역이 아래쪽에 있는지, 그리고 어떤 지역이 가장 크게 확대되는지를 결정한다. 이러한 모든 선택이 합리적으로 이루어지도록 도와주는 가이드라인이 있기는 하지만, 절대적인 규준은 존재할 수 없기 때문에, 어떤 맥락에서는 정치적 결단과 유사할 수도 있다.

다양한 투영법을 적용하고 결과를 비교해보는 것은 재밌기도 하고 교육적이기도 하다. 그러나 지도를 그리는 주된 목적이 투영법의 다양성에 대한 흥미 충족 및 지식 획득이 아니라면, 잘 알려져 있는 투영법 혹은 최소한 덜 생소한 투영법을 선택함으로써, 어떤 투영법을 선택할 것인가와 관련된 논의에 천착하기 보다는 선택된 투영법을 어떻게 적용할 것인가와 관련된 논의로 진전해 나가는 것이 더 바람직할 수 있다. 그러나 세계 지도를 위한 투영법으로 무엇인 좋은가에 대한 질문에는 어느 정도 합의된 대답이 존재한다. 거의 모든 경우에 있어, 정적 도법이 플라트 카레(Plate Carrée)(역자주: 등장방형 도법 혹은 적도를 표준위선으로 설정한 정거원통도법의 또 다른 이름이다.)나 웹 메르카토르 도법보다 선호된다.

## 8.2 포인트, 라인, 폴리곤, 그리드 셀의 플로팅

지도는 통계 데이터를 플로팅하는 특별한 형식으로 볼 수 있기 때문에, 플로팅의 일반적인 규칙이 지도에도 그대로 적용된다. 그렇지만 지도-특수적 문제는 존재하며, 여기에는 다음과 같은 것들이 있다.

* 매우 작은 폴리곤은 경우, 플로팅하면 사라질 수 있다.
* 데이터에 따라 지도 심볼이 서로 겹칠 수 있어서 어떤 심볼은 부분적으로만 보일 수 있다. 투명도를 조정하면 겹친 심볼을 식별하는 데 도움이 될 수 있다.
* 포인트 피처나 포인트 심볼을 플로팅하는 경우, 포인트들이 쉽게 겹치고 다른 포인트 뒤에 완전히 가려질 수 있다. 커널 밀도 지도(11장)가 더 유용할 수 있다.
* 라인 피처나 라인 심볼을 플로팅하는 경우, 라인에 부여된 색이 잘 분간되지 않을 수 있으며, 라인 너비와 상관없이 겹칠 수 있다.

### 8.2.1 컬러

컬러 심볼이 적용된 폴리곤을 플로팅하는 경우, 폴리곤 경계를 나타낼지 아니면 생략할지를 선택할 수 있다. 경계가 너무 눈에 띄면, 회색 톤이나 폴리곤 컬러와 충돌이 적은 다른 색상을 경계에 적용할 수 있다. 경계를 완전히 생략해 버리면, (거의) 동일한 컬러를 가진 폴리곤들을 시각적으로 구별할 수 없게 된다. 컬러가 서로 다른 토지 피복 유형을 표현한 것이라면, 경계의 생략이 큰 문제가 되지는 않는다. 그러나 컬러가 집계값(예를 들어 인구수)의 크기를 표현한 것이라면 지도 오독의 문제를 발생시킬 수 있다(역자주: 인구 1,000명 정도를 가진 폴리곤이 연접해 있을 때, 경계를 없애 버리면 마친 훨씬 넓은 지역의 인구가 여전히 1,000명인 것처럼 보여질 수 있다). 특히 인구 수와 같은 공간적으로 외연적인 속성의 경우, 이는 엄청난 오독의 가능성을 야기할 수 있다. 사실은, 공간적으로 외연적인 속성의 경우는 경계를 없애지 않는다 하더라도 폴리곤 내부를 컬러로 채우는 형태의 지도는 적절하지 않다. 왜냐하면 컬러가 폴리곤이 보유한 속성의 크기 뿐만 아니라 폴리곤의 면적도 함께 나타내기 때문이다(역자주: 이런 이유로 공간적으로 외연적인 속성은 코로플레스 맵이 아니라 도형표현도로 나타내는 것이 지도학적 합리성에 부합하는 것이다. 그러나 불행하게도 총합이나 총빈도를 코로플레스 맵으로 나타낸 지도가 인터넷에 넘쳐나고 있다.).

컬러의 단절이 없는 연속형 컬러 스킴은 연속형 공간 현상(역자주: 어디에나 존재하는 속성으로 예를 들어 기온)을 표현하는데 빈번히 사용되고 있는데, 지도학적 실용성보다는 시각적 매력도가 우선시되는 경우가 많다.

* 지도 상의 특정 컬러와 범례 상의 특정 값을 일치시키는 것은 인간의 시각 능력을 고려할 때 큰 실용성이 없다(역자주: 연속형 컬러 스킴을 적용한다는 것은 미세한 컬러의 차이에 의거해 미세한 값의 차이를 분간하라는 것인데, 효과적인 정보 전달이라는 측면에서 큰 지도학적 실효성이 없다).
* 컬러의 범위와 값의 범위가 비선형적으로 연결되는 경우가 많기 때문에, 값의 상대적인 차이를 분간하기 어렵게 만든다.

값의 식별보다는 공간적 현상의 연속성의 재현이 더 중요한 경우에 한하여 연속형 컬러 스킴의 적용이 정당성을 획득할 수 있다. 대표적인 예가 고해상도 디지털 지형 모델을 채색을 통해 시각화하는 경우이다. 좋은 컬러 스킴과 팔레트는 hcl.colors 또는 palette.colors 함수에서 찾을 수 있으며, **RColorBrewer**(Neuwirth 2022), **viridis**(Garnier 2021), 또는 **colorspace**(Ihaka et al. 2023; Zeileis et al. 2020)와 같은 패키지에서도 찾아볼 수 있다.

### 8.2.2 컬러 단절값: classInt

연속적인 공간적 속성을 제한된 컬로(또는 기호)를 사용해 플로팅하려고 하면, 데이터를 몇 개의 계급으로 구분하여야 한다. R 패키지인 **classInt** (Bivand 2022)는 이를 수행할 수 있는 여러 방법을 제공한다. 디폴트는 “등개수분류법(quantile)”이다.

library(classInt)  
# set.seed(1) if needed ?  
r <- rnorm(100)  
(cI <- classIntervals(r))  
# style: quantile  
# one of 1.49e+10 possible partitions of this variable into 8 classes  
# [-2.29,-1.27) [-1.27,-0.698) [-0.698,-0.426) [-0.426,-0.147)   
# 13 12 13 12   
# [-0.147,0.129) [0.129,0.47) [0.47,1.06) [1.06,2.1]   
# 12 13 12 13  
cI$brks  
# [1] -2.290 -1.272 -0.698 -0.426 -0.147 0.129 0.470 1.059 2.105

이 함수의 n 아규먼트를 통해 계급의 수를 설정할 수 있고, style 아규먼트를 통해 서로다른 계급구분 방식을 선택할 수 있다. 사용가능한 옵션에 “fixed”, “sd”, “equal”, “pretty”, “quantile”, “kmeans”, “hclust”, “bclust”, “fisher”, 또는 “jenks”가 있다. n이 지정되었다 하더라도 “pretty”를 선택하면 무효화될 수 있고, n이 제공되지 않으면 nclass.Sturges가 사용된다. 자동으로 n을 선택할 수 있는 두 가지 다른 방법도 있다. 관측값의 수가 3000개를 초과하는 경우, “fisher”와 “jenks”의 경우 10% 샘플이 사용되어 계급을 생성한다.

### 8.2.3 그래티큘 및 관련 요소

그래티큘은 일정한 위도 또는 경도를 따라 지도상에 그으진 선의 네트워크이다. 그림 1.1에서는 회색으로 그려져 있고, 그림 1.2에서는 흰색으로 그려져 있다. 그래티큘은 기본적으로 위치에 대한 참조물로서 지도에 그려진다. 그림 1.1의 첫 번째 지도에서는 플로팅된 지역이 북위 35도, 서경 80도 근처에 있음을 읽을 수 있다. 투영 좌표에 기반한 그래티큘은 모든 선이 직선으로 나타나고 좌표값이 특정 지점으로부터의 거리를 나타내므로 경위도 그래티큘에 비해 쓰임새가 크지 않다(역자주: 통상적으로 투영 좌표에 기반한 격자망은 그래티큘이라 부르지 않고 그리드(grid)라고 부른다). 그래도 독도자가 이러한 그래티큘에 익숙하고 좌표값의 단위가 주어져 있다면, 크기나 거리를 해석하는데 도움을 줄 수 있을 것이다. 그래티큘의 형태는 사용된 투영법의 특성을 반영한다. 따라서 그래트큘을 통해 투영법에 대한 정보를 얻을 수 있다. 등장방형 도법이나 메르카토르 도법은 수직선 및 수평선을 가지며, 원추 도법은 직선이지만 간격이 넓어지는 경선(meridian)을 가지며, 많은 정적 도법에서 경선은 곡선으로 나타난다(역자주: 이것은 지나친 일반화이다. 세계 전체를 나타내는 타원형 형태의 정적 도법(예를 들어: 에케르트 IV 도법)에서 경선이 곡선으로 표현되긴 하지만, 원통 도법(정축의 경우)에서는 경위선이 모두 직선으로 나타난다).

그림 8.1과 대부분의 다른 지도들에서, 실제적인 참조물 구실을 하는 것은 그래티큘이라기 보다는 주 경계, 국가 경계, 해안선, 강, 도로, 철도 등과 같은 지리적 사상들이다. 이러한 요소들이 지도상에 적절히 배치된다면 그래티큘은 생략하는 것이 좋다. 그래티큘이 생략되면 일반적인 플롯의 중요 구성요소인 축, 눈금, 레이블이 사라지므로, 실질적인 지도 데이터로 채울 수 있는 많은 플로팅 공간이 남겨지게 된다.

## 8.3 베이스 plot

sf 및 stars 객체에 적용되는 plot 메서드는 유용한 데이터 탐색용 플롯을 빠르고 간편하게 만들 수 있게 해준다. 더 높은 품질과 더 높은 사용자 자율성을 원한다면, **ggplot2**(Wickham et al. 2022), **tmap**(Tennekes 2022, 2018) 또는 **mapsf**(Giraud 2022)와 같은 패키지를 활용하면 된다.

plot 메서드의 디폴트는 주어진 “모든 것”을 플로팅한다는 것이다. 이는 다음을 의미한다.

* 지오메트리만 주어지면(sfc), 색상 없이 지오메트리만 플로팅된다.
* 지오메트리와 속성이 함께 주어지면, 속성 값에 따라 지오메트리에 컬러가 부여된다. factor 또는 logical 속성에는 질적인 컬러 스킴이 적용되고, 그 외에는 연속적 컬러 스킴이 적용되며 컬러에 대한 범례가 추가된다.
* 여러 속성이 주어지면, 여러 개의 지도가 플로팅된다. 색상 할당이 각 하위 지도별로 이루어지기 때문에 지도마다 다른 컬러 스킴이 적용된다. 디폴트로 범례는 생략된다.
* 여러 속성을 가진 stars 객체의 경우, 첫 번째 속성만 플로팅되며, 3차원 래스터 큐브의 경우, 세 차원 상의 모든 슬라이스가 하위 플롯으로 플로팅된다.

### 8.3.1 플롯에 범례 첨가하기

stars 및 sf 객체의 plot 메서드는 플롯 영역의 한쪽에 컬러 범례를 나타낼 수 있다(그림 1.1). base::plot 함수는 이를 위해 플롯 영역을 두 개로 나누고 두 개의 플롯을 생성한다. 당연히 하나는 지도이고, 다른 하나는 범례이다. plot 함수는 디폴트로 그래픽 장치를 초기화하는데(layout(matrix(1)) 이는 이후에 생성되는 플롯이 장치가 둘로 나누어지는 것에 의해 방해받지 않도록 하려는 것이다. 하지만, 이것은 만들어진 플롯에 그래픽 요소를 *추가*하는 것을 불가능하게 한다. 색상 범례가 있는 기존 플롯에 추가하려면 plot 명령에서 reset = FALSE를 사용하여 장치 초기화를 방지하고, 이후 plot 호출에서는 add = TRUE를 사용해야 한다. 그림 8.3에 예시가 나타나 있다.

library(sf)  
nc <- read\_sf(system.file("gpkg/nc.gpkg", package = "sf"))  
plot(nc["BIR74"], reset = FALSE, key.pos = 4)  
plot(st\_buffer(nc[1,1], units::set\_units(10, km)), col = 'NA',   
 border = 'red', lwd = 2, add = TRUE)

|  |
| --- |
| 그림 8.3: 범례가 있는 베이스 플롯에 주석 달기 |

단일한 stars 레이어가 표시되는 경우, stars 플롯에 주석을 추가하는 것은 동일한 방식으로 이루어진다. 여러 큐브 슬라이스가 있는 stars 패싯 플롯에 주석을 추가하려면, “후크(hook)” 함수를 추가하면 되는데, 이 후크 함수는 모든 슬라이스에서 개별적으로 호출되어야 한다. 이는 다음과 같이 수행할 수 있으며 결과가 그림 8.4에 나타나 있다.

library(stars)  
# Loading required package: abind  
system.file("tif/L7\_ETMs.tif", package = "stars") |>  
 read\_stars() -> r  
st\_bbox(r) |> st\_as\_sfc() |> st\_sample(5) |>   
 st\_buffer(300) -> circ  
hook <- function() {   
 plot(circ, col = NA, border = 'yellow', add = TRUE)  
}  
plot(r, hook = hook, key.pos = 4)  
# downsample set to 1

|  |
| --- |
| 그림 8.4: 다중 슬라이스를 가진 stars 플롯에 주석 달기 |

후크 함수는 패싯 파라미터, 패싯 레이블 및 바운딩 박스에 접근할 수 있다.

베이스 plot 메서드는 화면 장치의 해상도에 접근할 수 있다. 따라서 stars 및 stars\_proxy 객체에 대해 고밀도 래스터는 다운샘플링하여 가용 장치에 적합한 수준의 밀도로만 픽셀을 플로팅한다.

### 8.3.2 베이스 플롯의 투영법

베이스 plot 메서드는 타원체 좌표를 가진 데이터의 경우 등장방형 도법을 이용해 데이터를 플로팅하는데(그림 8.2), 표준 위선의 파라미터 값으로 경계 상자의 중간 위도를 디폴트로 사용한다. 이 파라미터 값을 제어하려면, 플로팅 전에 다른 파라미터 값을 가진 등장방형 도법을 적용하거나, 파라미터 asp를 직접 설정함으로써 디폴트를 해제하면 된다(asp=1은 플라트 카레 도법(그림 8.1 왼쪽)의 지도를 생성하게 된다). 기존 플롯에 차후의 플롯을 중첩하는 것이 의미가 있으려면, 후속 플롯에 동일한 CRS가 적용되어 있어야 한다. 베이스 plot 메서드는 동일 CRS의 여부를 체크하지 않는다.

### 8.3.3 컬러와 컬러 단절값

베이스 plot에서, nbreaks 아규먼트는 컬러 단절값의 수를 설정하는 데 사용되며, breaks 아규먼트는 실제 컬러 단절값을 포함한 수치 벡터를 설정하거나 classInt::classIntervals의 style 아규먼트에 대한 스타일 값을 지정하는데 사용된다.

## 8.4 ggplot2 패키지로 지도 그리기

**ggplot2** 패키지(Wickham 외, 2022; Wickham, 2016)는 더 복잡하고 보기 좋은 그래프를 만들 수 있다. 이 패키지에는 sf의 개발과 함께 발전된 geom\_sf라는 지오메트리 객체가 있으며, 아름다운 지도를 만드는 데 도움을 준다. 이에 대한 소개는 Moreno와 Basille(2018)에서 찾을 수 있다. 첫 번째 예시는 그림 1.2에 나와 있으며, 이 플롯에 사용된 코드는 다음과 같다.

library(tidyverse) |> suppressPackageStartupMessages()  
nc.32119 <- st\_transform(nc, 32119)   
year\_labels <-   
 c("SID74" = "1974 - 1978", "SID79" = "1979 - 1984")  
nc.32119 |> select(SID74, SID79) |>   
 pivot\_longer(starts\_with("SID")) -> nc\_longer

ggplot() + geom\_sf(data = nc\_longer, aes(fill = value), linewidth = 0.4) +   
 facet\_wrap(~ name, ncol = 1,   
 labeller = labeller(name = year\_labels)) +  
 scale\_y\_continuous(breaks = 34:36) +  
 scale\_fill\_gradientn(colours = sf.colors(20)) +  
 theme(panel.grid.major = element\_line(colour = "white"))

코드를 살펴보면, 패싯(facet) 형태의 플롯팅을 위해 사전에 두 개의 속성을 스택(pivot\_longer) 했다는 것을 알 수 있다. 이것이 “타이디” 데이터의 핵심 개념이며, sf 객체에 대한 pivot\_longer 메서드는 지오메트리 열도 자동으로 스택한다.

ggplot2는 그래픽 객체를 먼저 생성한 후 플롯팅하기 때문에, 모든 요소의 CRS를 제어할 수 있으며, 이후의 모든 객체를 첫 번째 객체의 CRS로 변환하거나 전환한다. 또한, 회색 배경 위의 얇은 흰색 선(디폴트)으로 그래피큘을 나타내며, 특정한 데이텀(디폴트는 WGS84)를 사용한다. geom\_sf는 다른 geom과 결합할 수 있으며, 이를 통해 플롯에 주석을 추가하는 등의 작업을 할 수 있다.

**stars** 패키지의 경우, geom\_stars가 존재하기는 하지만 책을 쓰고 있는 현 시점을 기준으로 말하자만 활용성이 다소 제한적인 상태에 있다. geom\_stars는 지도 레이아웃과 벡터 데이터 큐브에 대해서는 geom\_sf를 사용하며, 규칙 래스터에 대해서는 geom\_raster를, 직교 래스터에 대해서는 geom\_rect를 추가적으로 사용한다. 사용자가 다운샘플링 비율을 지정하면 다운샘플링을 수행하지만, 화면 크기에 접근하여 자동으로 다운샘플링 비율을 선택하는 기능은 없다. 그러나 이 정도 기능만으로도 충분할 수 있으며, 예를 들어 그림 8.5는 다음 명령어로 생성되었다.

library(ggplot2)  
library(stars)  
r <- read\_stars(system.file("tif/L7\_ETMs.tif", package = "stars"))  
ggplot() + geom\_stars(data = r) +  
 facet\_wrap(~band) + coord\_equal() +  
 theme\_void() +  
 scale\_x\_discrete(expand = c(0,0)) +   
 scale\_y\_discrete(expand = c(0,0)) +  
 scale\_fill\_viridis\_c()

|  |
| --- |
| 그림 8.5: ggplot2와 geom\_stars로 제작된 단순 패싯 래스터 플롯 |

더 정교한 ggplot2 기반의 stars 객체 플롯은 **ggspatial** 패키지(Dunnington, 2022)를 사용하여 제작할 수 있다. 보다 고품질의 지도를 만드는 옵션에 **tmap** 패키지가 있다. 이를 이용하면 ggplot2와 호환되지는 않지만 형식적으로 ggplot2과 유사한 스타일의 지도를 생성할 수 있다(8.5절).

## 8.5 tmap 패키지로 지도 그리기

**tmap** 패키지(Tennekes, 2022, 2018)는 R에서 공간데이터를 플로팅하는 신선한 접근을 시도한다. 이 패키지는 grid 패키지를 기반으로 그래픽 객체를 먼저 구성한 후 출력하며, 지도 요소를 + 기호로 연결하는 방식에서 ggplot2와 유사하지만, 그 외에는 ggplot2와 완전히 독립적이고 호환되지 않는다. 이 패키지는 매우 전문적인 지도를 만들 수 있는 다양한 옵션을 제공하며, 많은 디폴트 설정도 신중하게 선택되었다. 두 개의 유사한 속성을 가진 지도를 한꺼번에 만들 수 있는데, tm\_polygons 함수에 두 속성을 동시에 지정하면 된다.

library(tmap)  
system.file("gpkg/nc.gpkg", package = "sf") |>  
 read\_sf() |> st\_transform('EPSG:32119') -> nc.32119  
tm\_shape(nc.32119) +   
 tm\_polygons(c("SID74", "SID79"), title = "SIDS") +  
 tm\_layout(legend.outside = TRUE,   
 panel.labels = c("1974-78", "1979-84")) +  
 tm\_facets(free.scales=FALSE)

|  |
| --- |
| 그림 8.6: tmap: tm\_polygon() 함수에 두 개의 속성을 동시에 지정하여 플로팅하기 |

또는, pivot\_longer로 얻은 긴 테이블 형태의 데이터를 + tm\_polygons("SID") + tm\_facets(by = "name")를 적용하면 동일한 지도를 생성할 수 있다.

**tmap** 패키지는 또한 stars 객체를 지원하며, 아래에 예시가 그림 8.7에 나타나 있다.

tm\_shape(r) + tm\_raster()

|  |
| --- |
| 그림 8.7: tmap을 활용하여 제작된 단순한 래스터 플롯 |

**tmap** 패키지를 활용해 제작된 더 많은 사례 지도가 14~16장에 제시되어 있다.

## 8.6 인터렉티브 지도: leaflet, mapview, tmap

그림 1.3에 나타나 있는 것과 같은 인터렉티브 지도는 R 패키지인 **leaflet**, **mapview** 또는 **tmap**을 사용하여 생성할 수 있다. **mapview** 패키지는 **leaflet** 패키지가 제공하는 기본 기능에 여러 기능을 추가하는데, 맵 범례, 피처 클릭 팝업 창의 조정, 래스터 데이터 지원, FlatGeobuf 파일 형식의 대용량 피처 세트를 위한 축척조정형(scalable) 맵, 그리고 줌 및 팬 작업에 동기화하여 반응하는 패싯 맵과 같은 기능들이다. tmap 패키지는 다음을 제공하는 옵션이 있다.

**tmap** 패키지는 두 가지 모드를 제공하는 데, 다음과 같이 “view” 지정을 하면 모든 tmap 명령이 상호작용형 html/leaflet 위젯에 적용된다.

tmap\_mode("view")

그런데 “plot” 지정을 하면, 모든 산출물이 원래대로 R의 (정적인) 그래픽 장치로 보내진다.

tmap\_mode("plot")

## 8.7 연습문제

1. 인도네시아와 캐나다에 대해 등장방형(equirectangular) 도법, 정사(orthographic) 도법, 그리고 람베르트 정적원추(Lambert azimuthal equal area) 도법이 적용된 지도 플롯을 생성하라. 각 투영법에 대해 해당 국가에 적절한 투영 파라미터를 선택하라.
2. 그림 8.3의 플롯을 **ggplot2** 패키지와 **tmap** 패키지를 각각 사용하여 재생성하라.
3. 그림 8.7의 플롯을 viridis 색상 램프를 사용하여 재성성하라.
4. tmap의 “view” (상호작용) 모드를 사용하여 그림 8.7을 인터랙티브 플롯으로 재생성하고 어떤 상호작용이 가능한지 탐색하라. 또한 + tm\_facets(as.layers=TRUE)를 추가한 후, 레이어를 켜고 끄는 것을 해보라. 투명도 값을 0.5로 설정해 보라.

# 9. 대용량 데이터와 클라우드 네이티브

이 장에서는 R을 사용하여 대용량 공간 및 시공간 데이터셋을 처리하는 방법을 설명하며, **sf**와 **stars** 패키지의 활용에 초점을 맞춘다. 여기서 대용량의 의미를 다음의 세가지로 분류하고자 한다.

* 작업 메모리에 맞추기 어려울 만큼 크다.
* 로컬 하드 드라이브에 맞추기 어려울 만큼 크다.
* 로컬 관리 인프라(네트워크 부착 스토리지)에 다운로드하기 어려울 만큼 크다.

이 세 가지 범주는 (현재) 대략적으로 기가바이트(Gigabyte), 테라바이트(Terabyte), 페타바이트(Petabyte) 크기의 데이터셋에 해당한다고 볼 수 있다. 크기 고려 사항 외에도 접근 및 처리 속도도 중요한 역할을 하며, 특히 초대용량 데이터셋이나 인터랙티브 애플리케이션의 경우 더욱 그렇다. 클라우드 네이티브 지리공간 포맷은 클라우드 인프라에서의 처리에 최적화된 포맷으로, 컴퓨팅 및 저장 비용의 측면에서 최적화될 필요가 있다.

다음과 같은 방식으로 비용을 절감할 수 있다.

* 압축. 클라우드-최적화 GeoTIFF에 사용되는 LERC(제한 오류 래스터 압축) 알고리즘이나 ZARR 어레이 데이터에 사용되는 BLOSC 압축기와 같은 압축 기술이다.
* 공간 하위 영역에 대한 빠른 접근 또는 컬럼-지향 데이터 접근. 전자는 클라우드-최적화 GeoTIFF에 대해 HTTP 래인지 리퀘스트를 통해 이루어지며, 후자는 GeoParquet 및 GeoArrow 포맷에 주로 적용된다.
* 점진적인 해상도로 데이터에 접근 및 보기(우선 저해상도 데이터를 제공하고, 이후 필요에 따라 해상도를 점진적으로 높여가는 방식). 점진적 JPEG 포맷이나 이미지 피라미드(또는 오버뷰) 같은 것이 실행 예시이다.
* 데이터 접근의 최적화. 해당 클라우드 스토리지의 구조와 동작 방식에 맞추거나 해당 객체 스토리지 프로토콜이 제공하는 기능을 활용하는 등의 방식을 통해 최적화를 수행한다.

이 분야에는 만능 해결책이 없다는 점에 유의해야 한다. 특정 접근 패턴에 맞춰 스토리지를 최적화하면 다른 방식으로의 접근 속도가 느려질 수 있다. 예를 들어, 래스터 데이터가 서로 다른 공간 해상도에서 공간 영역에 대한 최적 접근을 위해 저장되면, 픽셀 시계열로 데이터를 읽는 속도가 매우 느릴 수 있다. 압축은 스토리지 및 대역폭(전송) 비용을 낮추지만, 읽을 때는 압축 해제 과정이 필요하기 때문에 처리 비용이 증가한다.

## 9.1 벡터 데이터: sf

### 9.1.1 로컬 디스크로부터 불러오기

st\_read 함수는 GDAL을 사용하여 디스크에서 벡터 데이터를 읽은 후, 해당 데이터를 작업 메모리에 유지한다. 파일이 너무 커서 작업 메모리에 모두 로드할 수 없을 경우, 파일의 일부만 읽는 여러 가지 옵션이 존재한다. 첫 번째 방법은 wkt\_filter 아규먼트에 지오메트리 정보를 포함한 WKT 텍스트 문자열을 설정하는 것이다. 이 경우, 해당 지오메트리와 교차하는 타깃 파일의 지오메트리 정보만 반환된다. 아래에 사례가 있다.

library(sf)  
# Linking to GEOS 3.11.1, GDAL 3.6.4, PROJ 9.1.1; sf\_use\_s2() is TRUE  
file <- system.file("gpkg/nc.gpkg", package = "sf")  
c(xmin = -82,ymin = 36, xmax = -80, ymax = 37) |>  
 st\_bbox() |> st\_as\_sfc() |> st\_as\_text() -> bb  
read\_sf(file, wkt\_filter = bb) |> nrow() # out of 100  
# Re-reading with feature count reset from 17 to 16  
# [1] 16

여기에서는 st\_read 대신 read\_sf가 사용되었는데, 출력을 억제하기 위해서이다.

두 번째 옵션은 st\_read의 query 아규먼트를 사용하는 것으로, 이는 “OGR SQL”로 작성된 어떤 쿼리도 다 가능하다. 예를 들어, 특정 레이어에서 피처를 선택하거나 필드를 제한하는데 사용될 수 있다. 아래에 사례가 있다.

q <- "select BIR74,SID74,geom from 'nc.gpkg' where BIR74 > 1500"  
read\_sf(file, query = q) |> nrow()  
# [1] 61

nc.gpkg는 레이어 이름이며, 이는 st\_layers를 사용하여 파일에서 확인할 수 있다. 레코드의 시퀀스는 LIMIT과 OFFSET을 사용하여 읽을 수 있으며, 51-60번째 레코드를 읽으려면 다음과 같이 한다.

q <- "select BIR74,SID74,geom from 'nc.gpkg' LIMIT 10 OFFSET 50"  
read\_sf(file, query = q) |> nrow()  
# [1] 10

추가적인 쿼리 옵션으로는 지오메트리 유형 또는 폴리곤 면적을 기준으로 선택하는 것이 있다. 쿼리되는 데이터셋이 공간 데이터베이스인 경우, 해당 쿼리는 GDAL이 해석하지 않고 데이터베이스로 전달된다. 이는 더 강력한 기능을 사용할 수 있음을 의미한다. 자세한 정보는 GDAL 문서의 “OGR SQL 방언(dialect)” 부분에서 확인할 수 있다.

대용량 파일이나 디렉터리가 압축되어 있는 경우, /vsizip(Zip 파일), /vsigzip(Gzip 파일), 또는 /vsitar(Tar 파일) 접두사를 파일 경로 앞에 붙여 압축을 풀지 않고도 파일을 읽을 수 있다. 압축 파일의 종류에 따라 접두어를 쓰고, 그 다음에 zip 파일의 경로를 쓰고, 그 다음에 압축 파일 내 해당 파일의 이름을 쓰면 된다. 이런 방식으로 파일을 읽으면 컴퓨팅 자원을 더 많이 소모할 수 있다.

### 9.1.2 데이터베이스로부터 불러오기, dbplyr

GDAL은 여러 공간 데이터베이스를 지원하며, 앞서 언급한 바와 같이 query 아규먼트에 포함된 SQL 문을 데이터베이스에 전달하지만, DBI R 데이터베이스 드라이버를 사용하여 공간 데이터베이스에 직접 읽고 쓰는 것이 유리한 경우도 있다. 이러한 예시는 다음과 같다.

pg <- DBI::dbConnect(  
 RPostgres::Postgres(),  
 host = Sys.getenv("DB\_HOST"),  
 user = Sys.getenv("DB\_USERNAME"),  
 dbname = "postgis")  
read\_sf(pg, query =   
 "select BIR74,wkb\_geometry from nc limit 3") |> nrow()  
# [1] 3

여기에서, 데이터베이스 호스트와 사용자 이름은 환경 변수에서 가져오고, 데이터베이스 이름은 postgis이다.

공간 쿼리는 보통 아래와 같은 모습이다.

q <- "SELECT BIR74,wkb\_geometry FROM nc WHERE \  
ST\_Intersects(wkb\_geometry, 'SRID=4267;POINT (-81.50 36.43)');"  
read\_sf(pg, query = q) |> nrow()  
# [1] 1

여기서, 인터섹션 연산은 데이터베이스 내에서 수행되며, 공간 인덱스가 존재할 경우 이를 사용한다. 데이터베이스 백엔드를 사용하여 dplyr을 활용할 때도 동일한 메커니즘이 작동한다.

library(dplyr, warn.conflicts = FALSE)  
nc\_db <- tbl(pg, "nc")

dplyr 문법 속에서 공간 쿼리가 작성될 수 있으며, 해당 쿼리는 데이터베이스로 전달된다.

nc\_db |>   
 filter(ST\_Intersects(wkb\_geometry,   
 'SRID=4267;POINT (-81.50 36.43)')) |>  
 collect()  
# # A tibble: 1 × 16  
# ogc\_fid area perimeter cnty\_ cnty\_id name fips fipsno cress\_id  
# <int> <dbl> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <int>  
# 1 1 0.114 1.44 1825 1825 Ashe 37009 37009 5  
# # ℹ 7 more variables: bir74 <dbl>, sid74 <dbl>, nwbir74 <dbl>,  
# # bir79 <dbl>, sid79 <dbl>, nwbir79 <dbl>,  
# # wkb\_geometry <pq\_gmtry>  
nc\_db |> filter(ST\_Area(wkb\_geometry) > 0.1) |> head(3)  
# # Source: SQL [3 x 16]  
# # Database: postgres [edzer@localhost:5432/postgis]  
# ogc\_fid area perimeter cnty\_ cnty\_id name fips fipsno cress\_id  
# <int> <dbl> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <int>  
# 1 1 0.114 1.44 1825 1825 Ashe 37009 37009 5  
# 2 3 0.143 1.63 1828 1828 Surry 37171 37171 86  
# 3 5 0.153 2.21 1832 1832 North… 37131 37131 66  
# # ℹ 7 more variables: bir74 <dbl>, sid74 <dbl>, nwbir74 <dbl>,  
# # bir79 <dbl>, sid79 <dbl>, nwbir79 <dbl>,  
# # wkb\_geometry <pq\_gmtry>

(참고로, PostGIS의 ST\_Area 함수는 nc의 AREA 필드와 동일한 면적을 계산하는데, 사실 아무런 의미가 없는 값이다. 왜냐하면 경위도 좌표를 마치 투영 좌표로 취급하여 계산한 면적이기 때문이다.)

### 9.1.3 온라인 리소스 혹은 웹 서비스로부터 불러오기

GDAL 드라이버는 URL이 https://로 시작하는 온라인 리소스에서 읽는 것을 지원하며, 이때 URL 앞에 /vsicurl/을 붙인다. 특정 클라우드에 특화된 유사한 드라이버도 여러 가지가 있으며, Amazon S3의 경우 /vsis3/, Google Cloud Storage의 경우 /vsigs/, Azure의 경우 /vsiaz/, Alibaba Cloud의 경우 /vsioss/, OpenStack Swift Object Storage의 경우 /vsiswift/를 사용한다. /vsicurl/을 사용하는 예시는 9.3.2절에 있다.

위의 접두사와 /vsizip/를 결합하면 압축된 온라인 리소스에서 파일을 읽을 수 있다. 파일 형식에 따라 다르겠지만, 보통 이런 방식으로 정보를 읽으면 파일 전체를 읽어야 하거나 심지어 여러 번 읽어야 할 수 있으며, 항상 리소스를 처리하는 가장 효율적인 방법이 아닐 수도 있다. 클라우드 네이티브 포맷은 HTTP 리퀘스트에 효과적으로 작동하도록 최적화되어 있다.

### 9.1.4 API, OpenStreetMap

지리공간 데이터를 위한 웹 서비스는 사용자의 요청에 따라 데이터를 실시간으로 생성하고 이를 API를 통해 접근할 수 있게 해준다. 예를 들어, OpenStreetMap의 데이터는 GDAL 벡터 드라이버와 같은 것을 통해 일괄 다운로드하여 로컬에서 활용하는 것이 가능하다. 그러나 사용자는 보통 일부 데이터만을 얻고자 하거나 소규모 쿼리에 데이터를 사용하고자 한다. OpenStreetMap 데이터를 쿼리하는 여러 R 패키지가 존재한다.

* **OpenStreetMap** 패키지(Fellows와 Jan Peter Stotz의 JMapViewer 라이브러리, 2019)는 데이터를 래스터 타일로 다운로드하며, 보통 다른 피처를 플로팅하는 배경 또는 참조로 사용된다.
* **osmdata** 패키지(Mark Padgham 외, 2017)는 **sf** 또는 **sp** 형식의 포인트, 라인 또는 폴리곤으로 벡터 데이터를 다운로드한다.
* **osmar** 패키지(CRAN 아카이브에서 다운로드 가능)는 벡터 데이터를 반환하지만, 도로 요소가 어떻게 연결되어 있는지를 포함한 네트워크 토폴로지(igraph 객체 형태)도 제공하며, 최단 경로를 계산하는 함수도 포함되어 있다.

올바르게 구성된 API 호출이 URL의 형태로 주어지면, 세세한 옵션 설정이 가능한 GDAL OSM 드라이버(st\_read 함수에서 사용 가능)가 “.osm” 파일(xml)을 읽고 다음과 같은 다섯 개의 레이어가 포함된 데이터 세트를 반환한다. 중요 태그를 가진 points, 면적이 없는 “way” 피처를 가진 lines, “relation” 피처를 가진 multilinestrings, “relation” 피처를 가진 miltipolygons, 그리고 other\_relations. 다음은 매우 작은 영역에 대한 단순한 OpenStreetMap에 대한 쿼리의 사례를 보여준다.

download.file(paste0("https://openstreetmap.org/api/0.6/map?",  
 "bbox=7.595,51.969,7.598,51.970"),  
 "data/ms.osm", method = "auto")

다운로드된 파일로부터 lines 레이어를 읽어 첫 번째 속성을 아래와 같이 플롯할 수 있다(그림 9.1).

o <- read\_sf("data/ms.osm", "lines")  
p <- read\_sf("data/ms.osm", "multipolygons")  
st\_bbox(c(xmin = 7.595, ymin = 51.969,   
 xmax = 7.598, ymax = 51.970), crs = 'OGC:CRS84') |>  
 st\_as\_sfc() |>  
 plot(axes = TRUE, lwd = 2, lty = 2, cex.axis = .5)  
plot(o[,1], lwd = 2, add = TRUE)  
plot(st\_geometry(p), border = NA, col = '#88888888', add = TRUE)

|  |
| --- |
| 그림 9.1: OpenStreetMap 벡터 데이터 |

Overpass API는 OpenStreetMap 데이터에 대해 더 일반적이고 강력한 쿼리 기능을 제공한다.

### 9.1.5 GeoParquet와 GeoArrow

클라우드 네이티브 분석에 특화된 두 가지 포맷은 Apache 프로젝트인 Parquet와 Arrow에서 파생되었다. 두 형식 모두 테이블 데이터에 대한 컬럼 지향 스토리지를 제공하는데, 대부분의 다른 데이터베이스가 취하고 있는 레코드 지향 스토리지에 비해 많은 레코드에서 단일 필드를 읽어내는 속도가 빠르다. 두 형식의 Geo 확장에는 다음이 포함된다.

* 지오메트리 컬럼을 저장하는 방법으로, WKB(well-known binary) 또는 WKT(well-known text)로 저장할 수 있으며, 서브 지오메트리가 미리 인덱스된 보다 효율적인 형식으로 저장할 수도 있다.
* CRS의 저장

이 책을 쓰고 있는 시점 기준으로, 두 포맷 모두 활발히 개발 중이며, GDAL 버전 3.5부터는 이들을 읽거나 생성하기 위한 드라이버가 제공된다. 두 포맷 모두 압축 저장을 지원한다. 차이점은 (Geo)Parquet가 퍼시스턴트 스토리지(역자주: 데이터를 장기적으로 훼손없이 저장하는 방식)에 더 중점을 두고 있는 반면, (Geo)Arrow는 빠른 접근과 빠른 계산에 더 중점을 두고 있다는 것이다. 예를 들어, Arrow는 인메모리 포맷(역자주: 데이터를 주기억장치에 저장하여 처리하는 방식)과 온디스크 포맷(역자주: 데이터를 저장 매체에 저장하는 방식) 모두로 사용할 수 있다.

## 9.2 래스터 데이터: stars

래스터 데이터셋을 다룰 때 발생하는 가장 일반적인 문제는 파일의 용량이 너무 크다는 점 뿐만 아니라(단일 Sentinel-2 타일은 약 1GB), 관심 있는 지역과 기간에 대한 데이터를 얻기 위해서는 수천 또는 수백만 개의 이러한 파일이 필요하다는 점이다. 2022년, Sentinel 위성 운영 프로그램인 Copernicus는 매일 160TB의 이미지 데이터를 산출했다. 이는 데이터를 로컬 디스크에 다운로드하고, 메모리에 로드한 다음, 분석하는 전통적인 R 사용 패턴은 더 이상 작동하지 않을 것임을 의미한다.

Google Earth Engine (Gorelick et al. 2017), Sentinel Hub, 또는 openEO.cloud와 같은 클라우드 기반 지구 관측 프로세싱 플랫폼은 이러한 점을 인식하고 사용자가 페타바이트 범위의 데이터셋을 쉽게 다루고 상호작용할 수 있도록 해준다. 이러한 플랫폼은 다음과 같은 특성을 공유한다.

* 계산은 가능한 한 늦게 수행된다(지연 평가)
* 사용자가 요청한 데이터만 계산되고 반환되며, 그 이상은 하지 않는다.
* 중간 결과를 저장하지 않고 즉석(on-the-fly) 계산을 선호한다.
* 유용한 결과를 가진 지도가 신속하게 생성되고 표시되어 인터렉티브 모델 개발을 가능하게 한다.

이것은 데이터베이스 및 클라우드 기반 분석 환경에 대한 dbplyr 인터페이스와 유사하지만, 우리가 *무엇*을 신속하게 보고자하는가에서 차이가 난다. dbplyr의 지연 테이블에서는 처음 *n*개의 레코드를 빨리 보고 싶어 하지만, 여기서는 결과에 대한 빠른 *개요*를 보고 싶어한다. 그것도 전체 지역 또는 일부 지역에 대한 지도 형식의 개요를 보고 싶어한다. 이를 위해 해상도는 손해본다(원본(관측) 해상도가 아닌 화면 해상도로 디스플레이 된다).

예를 들어, 화면에서 미국의 결과를 “보고자” 할 때 1000 x 1000 픽셀의 해상도가 필요하다면, 이만큼의 픽셀에 대한 결과만 계산하면 된다. 이는 대략 3000m x 3000m 그리드 셀로 구성된 데이터에 해당한다. Sentinel-2 데이터의 경우 해상도가 10m이므로, 300배로 다운샘플링하여 3km x 3km 해상도로 작업할 수 있다. 이 경우, 기본 10m x 10m 해상도로 작업할 때와 비교하여 처리, 저장 및 네트워크 요구 사항이 배 줄어든다. 언급된 플랫폼에서는 지도를 확대하면 더 세밀한 해상도와 더 작은 범위에서 추가 계산이 촉발된다.

이러한 원리에 따른 간단한 최적화 중 하나는 stars 객체의 플롯 방법이 작동하는 방식이다. 대용량 래스터를 플로팅할 경우, 플롯하기 전에 어레이를 다운샘플링하여 시간을 대폭 절약한다. 다운샘플링 정도는 플로팅 영역의 크기와 플로팅 해상도(픽셀 밀도)에 의거해 결정된다. PDF와 같은 벡터 장치의 경우, R은 플롯 해상도를 75 dpi로 설정하며, 이는 픽셀당 0.3mm에 해당한다. 플롯을 확대하면 이 사실이 드러날 수 있지만, 확대된 장치에 다시 플롯하면 목표 밀도로 플롯이 생성된다. geom\_stars의 경우, 사용자가 downsample 비율을 지정해야 하는데, 이는 **ggplot2**의 해당 함수를 통해 장치 크기를 설정할 수 있는 방법이 없기 때문이다.

### 9.2.1 stars 프록시 객체

작업 메모리에 맞추기 어려울 만큼 큰 데이터셋을 처리하기 위해 **stars**는 stars\_proxy 객체를 제공한다. 어떻게 사용하는지를 보여주기 위해, 대용량 데이터셋(총 약 1GB)을 포함한 R 데이터 패키지인 **starsdata** 패키지를 사용할 것이다. 이는 다음과 같이 설치할 수 있다.

options(timeout = 600) # or larger in case of slow network  
install.packages("starsdata",   
 repos = "http://cran.uni-muenster.de/pebesma/", type = "source")

다음과 같이 **starsdata** 패키지에서 Sentinel-2 이미지를 “불러올” 수 있다.

library(stars) |> suppressPackageStartupMessages()  
f <- paste0("sentinel/S2A\_MSIL1C\_20180220T105051\_N0206",  
 "\_R051\_T32ULE\_20180221T134037.zip")  
granule <- system.file(file = f, package = "starsdata")  
file.size(granule)  
# [1] 7.69e+08  
base\_name <- strsplit(basename(granule), ".zip")[[1]]  
s2 <- paste0("SENTINEL2\_L1C:/vsizip/", granule, "/", base\_name,   
 ".SAFE/MTD\_MSIL1C.xml:10m:EPSG\_32632")  
(p <- read\_stars(s2, proxy = TRUE))  
# stars\_proxy object with 1 attribute in 1 file(s):  
# $EPSG\_32632  
# [1] "[...]/MTD\_MSIL1C.xml:10m:EPSG\_32632"  
#   
# dimension(s):  
# from to offset delta refsys values x/y  
# x 1 10980 3e+05 10 WGS 84 / UTM z... NULL [x]  
# y 1 10980 6e+06 -10 WGS 84 / UTM z... NULL [y]  
# band 1 4 NA NA NA B4,...,B8  
object.size(p)  
# 12576 bytes

이 과정에서 실제로는 픽셀 값을 전혀 불러오지 않으며, 데이터셋에 대한 참조를 유지하고 디멘션 테이블을 채울 뿐이다. (복잡한 s2 이름을 통해 .zip 파일 내의 115개 파일 중 무엇이 올바른 파일인지를 GDAL이 찾을 수 있도록 한다.)

프록시 객체의 아이디어에 기반해 다음과 같은 표현식을 생성할 수 있다.

p2 <- p \* 2

하지만 이 경우 계산은 미루어진다. 실제로 데이터가 필요할 때만 p \* 2가 평가된다. 데이터가 필요할 때는 다음과 같다.

* 데이터를 플롯할 때
* write\_stars를 사용하여 객체를 디스크에 쓸 때
* st\_as\_stars를 사용하여 객체를 명시적으로 메모리에 불러올 때

전체 객체를 담기에 메모리가 부족한 경우, plot과 write\_stars는 이를 처리하기 위해 서로 다른 전략을 선택한다.

* plot은 모든 픽셀이 아닌 화면에 표시될 수 있는 픽셀만 다운샘플링하여 가져온다.
* write\_stars는 데이터를 청크 단위로 읽고 처리한 후, 기록한다.

|  |
| --- |
| 그림 9.2: Sentinel-2의 10m 밴드를 다운샘플링한 경우 |

고밀도 공간 이미지에 대해서는 다운샘플링과 청크 처리가 구현되지만, 고밀도 시계열이나 기타 고밀도 차원에 대해서는 구현되지 않는다. 예를 들어, 그림 9.2에 표시된 plot(p)의 출력은 각 밴드에서 사용할 수 있는 10,980 x 10,980 픽셀이 아닌, 플롯 장치에서 볼 수 있는 픽셀만 가져온다. 적용된 다운샘플링 비율은 다음과 같이 구할 수 있다.

# [1] 19

숫자가 의미하는 바는 원본 이미지의 19 x 19 서브 이미지마다 단 하나의 픽셀만 읽고 플롯되었다는 것이다.

### 9.2.2 프록시 객체에 대한 오퍼레이션

stars\_proxy 객체를 위해 여러 전용 메소드가 제공된다.

methods(class = "stars\_proxy")  
# [1] [ [[<- [<- adrop   
# [5] aggregate aperm as.data.frame c   
# [9] coerce dim droplevels filter   
# [13] hist image initialize is.na   
# [17] Math merge mutate Ops   
# [21] plot predict print pull   
# [25] rename select show slice   
# [29] slotsFromS3 split st\_apply st\_as\_sf   
# [33] st\_as\_stars st\_crop st\_dimensions<- st\_downsample   
# [37] st\_mosaic st\_normalize st\_redimension st\_sample   
# [41] st\_set\_bbox transmute write\_stars   
# see '?methods' for accessing help and source code

우리는 이미 plot과 print의 작동은 살펴보았고 dim은 차원 메타데이터 테이블에서 차원을 읽어온다.

실제로 데이터를 가져오는 세 가지 메소드는 st\_as\_stars, plot, 그리고 write\_stars이다. st\_as\_stars는 실제 데이터를 stars 객체로 읽어오며, 이때 downsample 아규먼트가 다운샘플링 비율을 제어한다. plot도 이 작업을 수행하며, 장치 해상도에 적합한 다운샘플링 값을 선택하고 객체를 플롯한다. write\_stars는 stars\_proxy 객체를 디스크에 기록한다.

stars\_proxy 객체를 위한 다른 모든 메소드는 래스터 데이터에 실질적으로 작용하는 것을 아니고, 객체에 연결된 작업 목록에 오퍼레이션을 추가하지만 한다. 실제 래스터 데이터가 불어와지면, 예를 들어 plot 또는 st\_as\_stars가 호출되면, 그 때 이 목록의 명령이 실행된다.

st\_crop는 읽힐 래스터의 범위(영역)를 제한한다. c는 stars\_proxy 객체를 결합한다. 역시 실제 데이터를 결합하는 것은 아니다. adrop은 빈 차원을 제거하고, aperm은 차원 순서를 변경한다.

write\_stars는 입력을 청크 단위로 읽고 처리한다. 이 메소드는 사용자가 공간 청크의 크기를 제어할 수 있도록 해주는 chunk\_size 아규먼트를 가진다.

### 9.2.3 원격 래스터 리소스

“클라우드 최적화 GeoTIFF”(COG)와 같은 포맷은 효율성과 자원 친화성을 염두에 두고 설계되었다. 예를 들어, 메타데이터만 읽거나 (전체 이미지를 낮은 해상도로 표시한) 오버뷰만 읽거나 /vsixxx/ 메커니즘을 사용하여 특정 공간 영역만을 읽는 데 유용하다(9.1.3절 참조). COG는 GDAL의 GeoTIFF 드라이버를 사용하여 생성할 수 있으며, write\_stars 호출에서 적절한 데이터셋 생성 옵션을 설정하면 된다.

## 9.3 대용량 데이터 큐브

어떤 단계에서는 다운로드가 더 이상 실행 불가능할 정도로 큰 데이터셋을 분석해야 할 필요가 있다. 로컬 저장소가 충분하더라도 네트워크 대역폭이 제한적일 수 있다. 예를 들어, Landsat 및 Copernicus(Sentinel-x)의 위성 이미지 아카이브나, 1950년부터의 전 세계 대기, 육지 표면, 해양 파도를 모델링한 ERA5(기후 재분석 모델)와 같은 모델 재분석(reanalysis)이 있다(Hersbach et al. 2020). 이러한 경우, 해당 데이터가 제공되는 클라우드의 가상 머신에 접근하거나, 사용자가 가상 머신과 스토리지에 대해 걱정하지 않고 계산을 수행할 수 있도록 해주는 시스템을 사용하는 것이 가장 유용할 수 있다. 두 가지 옵션에 대해 논의할 것이다.

### 9.3.1 검색과 처리

클라우드의 가상 머신에서 작업할 때, 첫 번째 작업은 일반적으로 작업할 자산(파일)을 찾는 것이다. 파일 목록을 얻고, 그런 다음 다음과 같은 파일 이름을 파싱하는 것이 매력적으로 보인다.

S2A\_MSIL1C\_20180220T105051\_N0206\_R051\_T32ULE\_20180221T134037.zip

파싱은 획득 날짜와 공간 타일 코드 정보를 가지고 있는 메타데이터에 대해 이루어진다. 그러나 이러한 파일 목록을 얻는 것은 일반적으로 전산 부담이 매우 크며, 타일 수가 수백만 개에 달할 경우 결과를 처리하는 것도 마찬가지로 부담이 된다.

이 문제에 대한 한가지 해결책은 카탈로그를 사용하는 것이다. 최근 개발되고 점점 더 많이 배포되는 STAC(*Spatiotemporal Asset Catalogue*의 약자)는 바운딩 박스, 날짜, 밴드, 구름 커버리지와 같은 속성으로 이미지 컬렉션을 쿼리하는 데 사용할 수 있는 API를 제공한다. R 패키지인 **rstac**(Simoes, Carvalho, Brazil Data Cube Team 2023)는 쿼리를 생성하고 반환된 정보를 관리하기 위한 R 인터페이스를 제공한다.

결과 파일을 처리하는 과정에는 서로 다른 CRS(예: 여러 UTM 존)을 가진 이미지로부터 더 낮은 공간 및/또는 시간 해상도의 데이터 큐브를 생성하는 것이 포함될 수 있다. 이러한 이미지 컬렉션에서 규칙 데이터 큐브를 생성할 수 있는 R 패키지로 **gdalcubes**(Appel 2023; Appel and Pebesma 2019)가 있으며, STAC(Appel, Pebesma, Mohr 2021)를 직접 사용하여 이미지를 식별한다.

### 9.3.2 클라우드 네이티브 스토리지: Zarr

COG가 래스터 이미지에 대한 클라우드 네이티브 스토리지를 제공하는 반면, Zarr은 대용량 다차원 어레이의 클라우드 네이티브 스토리지 포맷이다. Zarr은 NetCDF의 후속 포맷으로 볼 수 있으며, 기후 및 예측 커뮤니티에서 유사한 관례를 따르는 것으로 보인다(Eaton et al. 2022). Zarr “파일”은 실제로 데이터의 압축 청크를 포함하는 하위 디렉토리가 있는 디렉토리이다. 압축 알고리즘과 청크 전략은 특정 하위 큐브를 읽거나 쓰는 속도에 영향을 미칠 수 있다.

stars::read\_mdim 함수를 통해 전체 데이터 큐브를 읽을 수도 있지만, 옵션을 통해 하위 큐브를 읽을 수도 있다. 옵션은 각 차원별로 오프셋, 픽셀 수 및 낮은 해상도로 차원을 읽기 위한 스텝 사이즈 등을 지정할 수 있다(Pebesma 2022). 이와 유사하게, stars::write\_mdim 함수를 통해 다차원 어레이를 다양한 포맷으로 쓸 수 있는데, Zarr 또는 NetCDF 파일, 또는 GDAL C++ 다차원 배열 API를 지원하는 다른 포맷이 가능하다.

원격(클라우드 기반) Zarr 파일을 읽기 위해서는 URL 앞에 형식 및 접근 프로토콜에 대한 지시자를 추가해야 한다.

dsn = paste0('ZARR:"/vsicurl/https://ncsa.osn.xsede.org',  
 '/Pangeo/pangeo-forge/gpcp-feedstock/gpcp.zarr"')

그 후, 다음을 통해 첫 10개의 시간 스텝을 읽을 수 있다.

library(stars)  
bounds = c(longitude = "lon\_bounds", latitude = "lat\_bounds")  
(r = read\_mdim(dsn, bounds = bounds, count = c(NA, NA, 10)))  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max.  
# precip [mm/d] 0 0 0 2.25 1.6 109  
# dimension(s):  
# from to offset delta refsys x/y  
# longitude 1 360 0 1 NA [x]  
# latitude 1 180 -90 1 NA [y]  
# time 1 10 1996-10-01 1 days Date  
st\_bbox(r)  
# xmin ymin xmax ymax   
# 0 -90 360 90

이 예에서

* count의 NA 값은 해당 차원에 대해 사용 가능한 모든 값을 가져오도록 한다는 것을 의미한다.
* 여기서 bounds 변수는 데이터 소스가 더 최근의 CF (1.10) 규칙을 따르지 않았기 때문에 명시적인 지정이 필요했으며, bounds를 무시하면 [] 위도 범위를 벗어나는 바운딩 박스를 가진 래스터가 생성될 수도 있다.

### 9.3.3 데이터 API: GEE, openEO

클라우드에 *존재하는* 가상 머신에 대한 관리 및 프로그래밍 없이 이미지에 직접 접근할 수 있는 플랫폼으로는 GEE(구글 어스 엔진), openEO, 기후 데이터 스토어가 있다.

구글 어스 엔진(GEE)은 대용량의 지구 관측 데이터와 모델링 제품에 사용자가 접근할 수 있게 해주는 클라우드 플랫폼이다(Gorelick et al. 2017). 이 플랫폼은 6.3절에서 설명된 데이터 큐브 작업을 포함한 강력한 분석 기능을 제공한다. GEE는 JavaScript와 Python 언어를 위한 인터페이스를 제공한다. GEE의 코드는 오픈소스가 아니며, Python이나 R과 같은 언어를 통한 사용자 정의 함수의 추가가 불가능하다. R 패키지인 **rgee**(Aybar 2022)는 GEE에 대한 R 클라이언트 인터페이스를 제공한다.

온전히 오픈소스 소프트웨어로만 구축된 클라우드 기반 데이터 큐브 처리 플랫폼이 등장하고 있으며, 그 중 몇몇은 openEO API(Schramm et al. 2021)를 사용하고 있다. 이 API는 Python 또는 R로 작성된 사용자 정의 함수(UDF)를 허용하며, 이러한 함수는 API를 통해 전달되어 픽셀 수준에서 실행된다. 예를 들어, 사용자 정의 리듀서를 사용하여 차원을 애그리게이트하거나 축소할 수 있다. R의 UDF는 처리할 데이터 청크를 stars 객체로 표현하며, Python에서는 xarray 객체가 사용된다.

기타 플랫폼으로는 Copernicus 기후 데이터 스토리지(Raoult et al. 2017)나 대기 데이터 스토리지가 있으며, 이들 플랫폼을 통해 ERA5를 포함한 ECMWF의 대기 및 기후 데이터를 처리할 수 있다. 두 데이터 스토리지에 대한 인터페이스를 제공하는 R 패키지는 **ecmwfr**(Hufkens 2023)이다.

## 9.4 연습문제

다음 연습문제를 R을 사용하여 해결하시오.

1. S2 이미지(위의 경우)에 대해 st\_get\_dimension\_values()를 사용하여 밴드의 순서를 확인하고, 각 스펙트럴 밴드/색상이 무엇인지 알아보라.
2. S2 이미지에 대해 st\_apply함수와 적절한 ndvi 함수를 사용하여 NDVI를 계산하라. 결과를 화면에 플롯한 후, 결과를 GeoTIFF로 저장하라. 플로팅과 쓰기의 실행 시간 차이에 대해 설명하라.
3. S2 이미지를 RGB 합성하여 플롯하라. 먼저 plot()의 rgb 아규먼트를 사용하고, 그 다음 st\_rgb() 함수를 사용하라.
4. S2의 바운딩 박스에서 무작위로 다섯 개의 점을 선택하고, 이 점에서 밴드 값을 추출하라. 반환된 객체를 sf 객체로 변환하라.
5. POINT(390000 5940000) 중심으로 반경 10km 원에서 aggregate를 사용하여 S2 이미지의 평균 픽셀 값을 계산하라. 이미지를 30배 다운샘플링한 결과와 원래 해상도에서의 결과를 비교하라. 두 결과 간의 상대적 차이를 계산하라.
6. 다운샘플링된 S2 이미지에서 hist를 사용하여 히스토그램을 계산하라. 각 밴드에 대해서도 동일한 작업을 수행하라. ggplot2를 사용하여 네 개의 히스토그램을 모두 포함하는 단일 플롯을 만들어라.
7. st\_crop 함수를 사용하여 S2 이미지를 10km 원이 포함하는 영역으로 자른 다음 결과를 플롯하라. 아규먼트 crop = FALSE를 설정했을 때의 어떤 변화가 있는지 살펴보라.
8. 다운샘플링된 이미지를 사용하여 네 개의 밴드에서 모든 픽셀 값이 1000보다 큰지의 여부를 나타내는 논리 레이어를 계산하라. 네 개의 밴드에 대해 래스터 대수 표현식을 사용하거나 st\_apply 함수를 사용하라.

# 10. 공간데이터의 통계적 모델링

지금까지 이 책에서는 주로 데이터를 기술하거나 탐색하는 문제를 다뤘다. 이는 기하학적 측정, 프리디케이트, 지오메트리 변환, 속성의 통계적 요약, 플로팅 등의 방법을 통해 이루어졌다. 특히 플로팅은 지오메트리와 속성 각각의 변동성을 보여줄 뿐만 아니라 두 가지 모두의 변동성도 보여준다.

통계적 모델링은 단순히 데이터를 기술하거나 탐색하는 것을 넘어서, 데이터를 모집단에서 추출된 표본으로 간주하고, 모집단에 대한 평가(추론)를 시도하는 것을 목표로 한다. 예를 들어 변수들 간의 관계를 정량화하거나, 모집단의 모수를 추정하거나, 인터폴레이션 문제처럼 실제로 관측되지 않은 결과를 예측하는 등의 방법이 이에 해당한다.

이는 보통 데이터에 대한 모델 수립을 통해 수행되며, 예를 들어 특정 모델은 관측값을 다음과 같이 분해한다.

여기서 “설명된 부분”은 일반적으로 관측된 변수와 관련된 외부 변수(예측 변수, 공변량, 머신러닝에서는 혼동되게도 피처라고도 함)와, 관측된 변수의 변동성을 설명하기 위한 회귀 모델 등을 사용하여 얻는다. “잔여 부분”은 설명되지 않은 나머지 변동성을 의미한다. 이러한 통계적 모델링의 목적은 예측 변수와 관측 변수 간 관계의 특성 및 크기를 파악하려는 것일 수도 있고, 새로운 관측값을 예측하려는 것일 수도 있다.

통계 모델과 표본 추출은 확률 개념에 토대를 두고 있다. 그런데 공간데이터사이언스에서확률은 자명한 것이 아니라 특정한 방식으로 가정되어야 한다. (공간적) 무작위 표본 추출에서 얻은 데이터가 주어지고 평균이나 총합을 추정하는 것이 목표라면, 표본 위치의 무작위성을 가정하는 *디자인 기반* 접근법이 가장 좋은 분석 방법이다(10.4절에서 더 자세히 설명한다). 만약 관측값이 무작위로 추출되지 않았거나 특정 위치에서의 값을 예측하는 것(즉, 지도화하는 것)이 목적이라면, *모델 기반* 접근법이 필요하다. 이 파트의 나머지 장들은 모델 기반 접근법을 다룬다.

## 10.1 비공간적 회귀분석과 머신러닝 모델을 통한 지도화

회귀 모델이나 기타 머신러닝(ML) 모델은 비공간 문제에서 새로운 관측값을 예측하는 데 사용되는 방식과 동일하게 공간 및 시공간 데이터에도 적용할 수 있다.

1. **추정**: 일련의 관측값에 대해, 해당 관측값에 대응하는 예측 변수 값을 사용하여 회귀 또는 머신러닝 모델을 적합시킨다(머신러닝 용어로는 이 단계를 “훈련”이라고도 함).
2. **예측**: 새로운 상황에 대해 알려진 예측 변수 값을 적합된 모델과 결합하여 관측 변수 값을 예측하며, 가능하다면 예측 오차 또는 예측 구간도 함께 제시한다.

sf 클래스 객체는 그 자체가 data.frame 객체이므로 특별한 처리가 필요하지 않다. 예측 결과를 지도에 표현하려면, 예측된 값을 sf 객체에 추가해야 하며, 1장에서 불러온 nc 데이터셋을 사용해 다음과 같이 할 수 있다.

nc |> mutate(SID = SID74/BIR74, NWB = NWBIR74/BIR74) -> nc1  
lm(SID ~ NWB, nc1) |>  
 predict(nc1, interval = "prediction") -> pr  
bind\_cols(nc, pr) |> names()  
# [1] "AREA" "PERIMETER" "CNTY\_" "CNTY\_ID" "NAME"   
# [6] "FIPS" "FIPSNO" "CRESS\_ID" "BIR74" "SID74"   
# [11] "NWBIR74" "BIR79" "SID79" "NWBIR79" "geom"   
# [16] "fit" "lwr" "upr"

여기에서 우리는 다음의 사항을 알 수 있다.

* lm 함수는 선형 모델을 추정하며 sf 객체에 직접 적용할 수 있다.
* 모델의 결과가 predict 모델에 투입되는데, 이 모델은 nc와 마찬가지로 sf 객체인 nc1에 대한 예측값을 생성한다.
* predict는 세 개의 열을 생성하는데, fit은 예측값을, lwr과 upr은 95% 예측 구간을 나타낸다.
* 이 세 열은 bind\_cols 함수를 사용하여 최종 객체에 추가되었다.

많은 회귀 및 머신러닝 유형 문제들이 동일한 구조를 공유하기 때문에, caret (Kuhn 2022) 또는 tidymodels (Kuhn and Wickham 2022)와 같은 패키지는 광범위한 모델 대안에 대한 자동 평가 및 비교를 가능하게 하며, 다양한 모델 평가 기준과 교차 검증(cross-validation) 전략을 제공한다. 이러한 교차 검증 접근법은 기본적으로 독립 관측 가정에 기반한다. 그런데 공간데이터와 관련하여서는 이것이 반드시 합리적인 가정이라고 전제하기 어려운 경우가 상당히 많다. 이는 공간적 자기상관(Ploton et al. 2020) 때문이거나 표본 데이터에서의 강한 공간 클러스터링(Meyer and Pebesma 2022) 때문이거나, 둘 다 때문이다. 여러 R 패키지는 이러한 단순한 교차 검증을 대체하기 위한 방법을 제공하는데, 여기에는 spatialsample (Silge and Mahoney 2023), CAST (Meyer, Milà, and Ludwig 2023), mlr3spatial (Becker and Schratz 2022), 그리고 mlr3spatiotempcv (Schratz and Becker 2022)가 포함된다.

다양한 이유로 표본의 강한 공간 클러스터링이 나타날 수 있다. 하나는 서로 다른 데이터베이스를 결합하여 하나의 표본 데이터를 구성하는 경우이다. 이는 각 데이터베이스는 서로 다른 샘플링 밀도를 가지고 있기 때문인데, 보통 글로벌 데이터셋에서 나타난다(Meyer and Pebesma 2022). 또 다른 강한 클러스터링의 예는, 토지 피복 클래스를 샘플링하기 위해 폴리곤을 디지타이징하고 이들 폴리곤 내에서 위성 이미지의 픽셀 해상도로 포인트를 샘플링할 때 발생한다.

공간적 자기상관은 모델의 “잔여 부분”에서 나타날 수 있는데, 공간 좌표 또는 공간 좌표의 함수들을 예측 변수로 추가 투입하면 감소시킬 수 있다. 그러나 이는 인터폴레이션에서 지나치게 낙관적인 예측의 위험성을 수반하며, (교차) 검증 및 모델 평가에서도 동일한 문제를 야기한다. 이와 관련된 내용은 10.5절에서 더 논의된다.

## 10.2 서포트와 통계적 모델링

데이터의 서포트(1.6절; 5장) 개념은 공간 데이터의 통계 분석에서 중요한 역할을 한다. 에어리어 데이터에 대한 기법(14-17장)은 에어리어 서포트를 가진 데이터, 즉 에어리어들의 집합이 지역 전체를 포괄하는 데이터를 위한 것이다.

공간적으로 외연적인 변수를 코로플레스맵으로 나타내는 것은(1.1 그림과 같이) 상당히 위험한 선택일 수 있다. 왜냐하면 정보가 폴리곤 크기와 관련되면서 컬러로 표현된 속성값이 해당 폴리곤의 전역에서 나타나는 것처럼 보여질 수 있다. 해결하는 방법은 공간적으로 외연적인 변수를 공간적으로 내포적인 변수로 전환하는 것인데, 예를 들어 *인구 수*와 같은 변수의 경우 폴리곤 면적으로 나누어 *인구 밀도*를 산출하고 그것을 지도로 나타내는 것이다. 그럼 1.1에 나타나 있는 연도별 발병자 수와 같은 보건 데이터의 경우는, 폴리곤 면적으로 나누어 공간적 밀도를 계산하기보다는 관측값을 해당 폴리곤의 인구 수로 나누어 확률이나 *발병률*로 변환하는 것이 일반적이다. 이 경우, 여전히 폴리곤 면적과 연관되지만, 해당 변수의 서포트는 인구 총수와 관련되어 있다. 이러한 총수는 후속 모델링, 예를 들어 CAR 유형 모델(16장)에서 사용되는 (포아송) 변동성을 알려주는 데 중요한 역할을 한다.

11장은 원칙적으로 포인트 서포트 관측개체를 다루지만, 그렇다고 해서 관측개체가 면적이 전혀 없는 0차원인 것은 아니라는 점을 인식해야 한다. 예를 들어 나무 줄기 “포인트들”의 경우 나무 지름보다 작은 거리에서는 또 다른 포인트가 나타날 수가 없다. 또한, 포인트 패턴 분석에서 포인트는 *관측 창* 개념에 의거해 측정된 것으로, 해당 관측 창은 완벽한 포인터 데이터셋을 구성해야 한다(즉 관측 창 외부에는 포인트가 없다). 관측 창 개념은 많은 분석 도구에 영향을 미친다. 만약 포인트가 라인 네트워크 상에서 관측된다면, 관측 창은 관측된 라인의 집합으로 구성되며, 포인트 간 거리는 해당 네트워크 상에서의 거리로 측정된다.

지구통계학적 데이터(12장 및 13장)는 일반적으로 포인트 서포트 관측값에 기반하여 연구 대상 지역 내 비관측 지점의 값을 예측하거나(공간적 인터폴레이션), 세부 지역에 대한 평균값을 예측한다(블록 크리깅; 12.5절). 관측값이 포인트 값이 아니라 구역 전체의 집계값일 수도 있다(Skøien et al. 2014). 원격탐사 데이터에서 픽셀 값은 보통 픽셀 전체에 대한 집계값이다. 이 경우 구름으로 인한 이미지의 결함을 인접한 공간 및 시간의 픽셀로부터 메우는 것이 중요한 도전 과제이다(Militino et al. 2019).

서로 다른 공간 지원을 가진 데이터를 결합할 때, 모든 데이터를 가장 높은 해상도를 가진 데이터에 “맞추는” 경우가 종종 발생한다. 예를 들어 행정구역 폴리곤과 래스터 레이어를 결합하는 경우, 래스터 레이어의 개별 픽셀 위치에서 폴리곤의 속성 값을 단순 추출하고, 이렇게 생성된 “관측값”을 가지고 분석을 진행해나가는 방식이다. 당연히 이러한 방식으로 생성된 “데이터”를 분석하면 비합리적인 결과를 초래할 위험성이 훨씬 커진다. 불확실성에 대처하기 위해 시뮬레이션을 사용할 수 있는 적절한 다운샘플링 전략이 더 나은 대안이 될 것이다. 초보 사용자가 주어진 소프트웨어를 경각심 없이 사용하는 상황은 매우 우려스럽다. 왜냐하면 어떤 소프트웨어는 구역과 관련된 속성값의 서포트 개념을 잘 다루지 못하거나, 나이브한 다운샘플링에 대한 경고를 보여주지 않기 때문이다.

## 10.3 예측 모델에서의 시간

Schabenberger와 Gotway (2005)는, 많은 시공간 데이터에 대한 통계 분석이 시간 차원을 축소(삭감)한 후 해당 문제를 공간적으로 다루거나(시간 먼저, 공간 나중) 공간 차원을 축소한 후 문제를 시간적으로 다루는(공간 먼저, 시간 나중) 방식으로 진행된다는 점을 오래 전에 지적한 바 있다. 첫 번째 접근 방식의 예는 12장에서 제시되는데, 1년 동안의 시간 단위 측정값 데이터셋(13장에서 자세히 설명됨)을 먼저 측정소의 평균 값으로 축소한 후(시간 먼저), 이 평균 값을 공간적으로 인터폴레이션하는(공간 나중) 과정을 보여준다.

원격탐사 분야의 예시는 다음과 같다.

* Simoes et al. (2021)은 지도 학습 기법과 시계열 딥 러닝을 사용하여 픽셀 시계열을 토지 이용 시퀀스로 분할한 후(시간 먼저), 결과로 생성된 맵 시퀀스를 평활화하여 독특한 필셀이 보여주는 불가능한 전환을 제거(공간 나중)한다.
* Verbesselt et al. (2010)은 픽셀 시계열에서 급변점을 찾기 위해 (비지도) 구조적 변화 알고리즘을 사용하고(시간 먼저), 이러한 급변점을 이후에 산림 파괴의 맥락에서 해석한다.

원격탐사 분야에서 공간 먼저, 시간 나중 접근 방식의 예시는 단일 레이어(scene) 또는 단일 시즌에 속하는 레이어들을 우선 분류한 후, 분류된 레이어의 시간 순서를 비교하여 토지 이용이나 토지 피복의 다년간 변화를 평가하는 경우이다. Brown et al. (2022)가 그 예이다. 공간과 시간을 *함께* 고려하는 예시로는 13장에서 다루는 시공간 인터폴레이션(spatiotemporal interpolation)과 Lu et al. (2016)이 원격탐사 맥락에서 제시한 연구가 있다.

## 10.4 디자인-기반 추정과 모델-기반 추정

통계적 추론은 샘플 데이터를 통해 모집단에 대한 파라미터(모수)를 추정하는 행위를 의미한다. 위치 에서 측정된 속성값을 라고 하면 변수는 로 주어지고, 우리는 샘플 데이터 에서 도메인 의 면적 에 대한 |의 평균값을 추정하는 것에 관심이 있다고 하자.

이 때 우리 앞에는 모델-기반 접근법(model-based)과 디자인-기반 접근법(design-based)이라는 두 가지 가능성이 놓여 있다. 모델-기반 접근법은 를 초모집단 의 실현으로 간주하며(확률 변수를 나타내기 위해 대문자를 사용함), 예를 들어 다음과 같은 공간 변동성을 설명하는 모델을 가정할 수 있다.

여기서 은 고정(constant) 평균이고 는 평균이 0이고 공분산 행렬이 인 잔차를 의미한다. 이는 공분산 함수 를 선택하고 로부터 그 매개변수 를 추정한 다음, 블록 크리깅 예측값 를 계산해야 한다(12.5절). 이 접근법은 가 공간적으로 어떻게 샘플링되었는지에 대한 가정을 하지 않지만, 물론 공분산 함수를 선택하고 그 파라미터를 추정할 수 있어야 한다. 추론은 가정된 모델의 유효성에 따라 달라진다.

디자인-기반 접근법(De Gruijter and Ter Braak 1990; Brus 2021a; Breidt, Opsomer 등 2017)은 초모집단 모델이 아니라 위치에서의 무작위성을 가정한다. 따라서 이 접근법은 무작위 샘플링을 사용할 때만 정당화된다. 샘플 데이터는 반드시 확률 샘플링을 통해 획득되어야 하는데, 의 모든 요소가 샘플에 포함될 (양수) 확률이 수학적으로 규정되어 있는 특정한 종류의 공간적 무작위 샘플링이 사용되어야 한다. 무작위 프로세스는 샘플링 프로세스이다. 즉, 은 무작위 과정 의 실현으로 *무작위 샘플링을 반복하여 얻은* 첫 번째 관측치이다. 디자인-기반 추정량은 이러한 포함 확률만 있으면 평균 값을 표준 오차와 함께 추정할 수 있다. 이는 예를 들어 단순 무작위 샘플이 주어졌을 때, 비가중 샘플 평균이 그대로 모집단 평균을 추정하는 데 사용되며, 모델 파라미터를 적합시킬 필요가 없다.

이제 질문은 과 가 가까이 있을 때 과 가 서로 상관될 것으로 기대할 수 있는지 여부이다. 과 가 단지 두 개의 숫자에 불과할 때는 이 질문이 성립하지 않는다. 상관성을 논하기 위해서는 확률 변수와 같은 일종의 프레임워크가 필요한데, 이 상황을 재현할 수 있는 두 세트의 숫자를 만들어 내야한다. 여기서의 오해는, Brus (2021a)가 설명한 바와 같이, 두 세트의 숫자가 항상 공간적으로 상관되어 있다고 생각하는 것이다. 그러나 이는 과 가 강한 상관성을 가지는 모델을 전제한 경우에만 성립한다(“모델-의존”). 특정 무작위 샘플(실현)에서 과 는 공간적으로 가까울 *수* 있지만, 반복적 무작위 샘플링을 통한 확률 변수 과 는 공간적으로 가깝지 않고, 디자인에 독립적이다. 이 두 상황은 서로 모순 없이 공존할 수 있으며, 어떤 추론 틀을 선택하느냐에 달린 문제일 뿐이다.

디자인-기반 프레임워크를 선택할지 모델-기반 프레임워크를 선택할지는 연구 목적과 데이터 수집 과정에 따라 달라진다. 모델-기반 프레임워크는 다음과 같은 경우에 가장 적합하다.

* 개별 위치에 대한 예측이 필요하거나, 샘플링하기에는 너무 좁은 지역에 대한 예측이 필요한 경우
* 해당 데이터가 무작위 샘플링 방식으로 수집되지 않은 경우(즉, 샘플 포함 확률이 알려져 있지 않거나 특정 지역이나 시간대에 대해 포함 확률이 0인 경우)

디자인-기반 접근법은 다음과 같은 경우에 가장 적합하다.

* 관측값이 공간 무작위 샘플링 과정을 통해 수집된 경우
* 전체 샘플 지역(또는 하위 지역)에 대한 집계 속성이 필요한 경우
* 규제나 법적 목적 등을 위해 모델의 오지정 문제에 민감하지 않은 추정치가 필요한 경우

샘플링 절차를 계획해야 하는 경우(De Gruijter et al. 2006), 공간 무작위 샘플링 방식을 고려하는 것은 매우 바람직한데, 이는 두 가지 추론 프레임워크를 모두 활용할 가능성을 열어주기 때문이다.

## 10.5 좌표값을 활용한 예측 모델

데이터사이언스 프로젝트를 수행할 때, 좌표값을 예측 변수(또는 피처, 공변량) 중 하나로 간주하고 다른 변수와 동일한 방식으로 취급하는 경우가 있다. 그런데, 여기에는 몇 가지 위험 요소가 도사리고 있다.

예측 변수를 다룰 때와 마찬가지로, 원점 이동이나 단위(스케일) 변화에 민감하지 않은 예측 방법을 선택하는 것이 좋다. 2차원 문제를 가정할 때, 예측 모델은 -축과 -축 또는 위도와 경도 축의 임의 회전에 민감해서는 안 된다. 투영된 2차원(데카르트) 좌표의 경우, 예를 들어 과 같은 차 다항식을 사용해 이러한 특성을 보장할 수 있으며, 이는 형태 대신 사용된다. 2차 다항식의 경우 항을 포함시켜 타원형 경향면이 반드시 -축이나 -축에 정렬될 필요가 없도록 한다. 스플라인 요소를 포함하는 GAM 모델의 경우에도 상호작용을 허용하지 않는 독립적 스플라인 와 대신, 2차원에서의 스플라인 를 사용한다. 이러한 ’규칙’에 당연히 예외가 있을 수 있는데, 예를 들어 연간 총 태양 에너지 유입을 설명하기 위해 순전히 위도의 효과만 필요한 경우이다.

데이터가 넓은 지역에 걸쳐 있는 경우, 타원 좌표와 투영 좌표를 사용하는 것의 차이가 당연히 커지기 때문에 둘 중 어느 좌표를 선택하느냐가 예측 모델링의 결과에 큰 영향을 미칠 수 있다. 매우 넓은 범위나 전 지구적 모델의 경우, 위도와 경도 좌표를 활용한 다항식이나 스플라인은 경도의 순환 특성과 극지점에서의 좌표 특이성을 무시하기 때문에 잘 작동하지 않는다. 이때, 구면 조화 함수(spherical harmonics)를 다항식에 대한 대체 함수나 스플라인 기저 함수로 활용할 수 있는데, 이는 구면 위에서 연속성을 가지며 공간 주파수가 높아질수록 더 세밀한 변화를 표현할 수 있는 함수이다.

많은 경우, 표본이 수집된 공간 좌표는 예측이 이루어질 공간도 정의하게 되기 때문에, 좌표 변수는 다른 예측 변수와는 선명하게 구별되는 특성을 가진다. 대부분의 간단한 예측 접근법, 특히 많은 머신러닝 기법은 표본 데이터를 독립적이라고 가정한다. 표본이 공간 대상 지역에서 공간 무작위 샘플링(표본추출)에 의해 수집된 경우, 디자인-기반 모델의 맥락에서는 이 가정이 정당화될 수 있다(Brus 2021b). 그러나 디자인-기반 접근에서는 좌표 공간을 우리가 무작위화의 대상 변수로 간주하기 때문에 새롭게 무작위로 선택된 위치에 대한 예측이 가능하지만 고정된 위치에 대한 예측은 불가능하다. 즉, 표본이 수집된 지역의 평균 값을 구할 수는 있지만, 해당 지역에 대한 공간적 인터폴레이션 값을 구할 수는 없다. 고정된 위치에 대한 예측이 필요하거나 데이터가 공간 무작위 표본추출로 수집되지 않은 경우, 모델-기반 접근(Chapter 12에서 설명)과 더불어 잔차의 공간 및/또는 시간적 자기상관을 가정하는 방법을 고려해야만 한다.

일반적인 경우는 샘플 데이터가 기회적으로 수집되고(“찾을 수 있는 것 모두”), 이후 가중치를 부여하지 않은 채 예측 프레임워크에서 사용되는 경우이다. 이로 인해 결과 모델은 과대표집된 영역(예측 변수 공간 및/또는 공간 좌표 공간)으로 편향될 수 있으며, 단순한(무작위) 교차 검증 통계가 공간 예측의 성능 척도로 사용될 때 지나치게 낙관적인 결과가 도출될 수 있다 (Meyer and Pebesma 2021, 2022; Mila et al. 2022). 공간적 교차 검증(spatial cross-validation)과 같은 적응형 교차 검증 측도가 예측 성능에 대한 유관성 높은 평가값을 얻는 데 도움이 될 수 있다.

## 10.6 연습문제

R을 활용하여 다음의 연습문제를 해결하라.

1. 10.1절의 lm 예제를 참고하여 랜덤 포레스트 모델을 사용해 SID 값을 예측하고(예: **randomForest** 패키지 사용), 랜덤 포레스트 예측값을 관측값과 함께 플로팅하되 선도 함께 표시하시오.
2. nc 데이터셋에서 1,000개의 포인트를 무작위로 샘플링하여 새로운 데이터셋을 만들고, 이 데이터셋에 10.1절의 선형 회귀 모델을 다시 실행한다. 적합된 모델의 summary를 살펴보라. 특히 추정 계수, 표준 오차, 잔차 표준 오차를 주목하라. 원 모델과 비교했을 때 무엇이 달라졌는가?
3. 7.4.6절의 수역-육지 분류를 lda 대신 class::knn을 사용하여 k=5의 값을 설정하여 다시 수행하고, 예측값을 lda의 예측값과 비교하라.
4. nc 데이터셋을 사용하는 선형 모델과 이전 연습문제의 knn 예제에 대해 1차 및 2차 공간 좌표 변수를 추가한 다항 선형 모델을 실행하고 결과를 비교하라. 이를 위해 st\_centroid를 사용하여 폴리곤의 중심점을 얻고, st\_coordinates를 사용하여 x 및 y 좌표를 행렬 형태로 추출하라.

# 11. 포인트 패턴 분석

포인트 패턴 분석은 공간 상의 포인트 패턴을 기술하거나 그러한 패턴을 야기한 프로세스에 대한 추론을 행한다. 여기서 핵심적인 사항은 포인트들의 위치 정보인데, 이러한 위치는 샘플링을 위한 통제 지점이 아니라, 특정한 프로세스의 산물(동물 목격 지점, 사고 지점, 발병 지점, 수목 지점 등)이다. 이것은 모든 지점에서 해당 현상이 발생하지만 관측값은 우리가 통제한 관측 지점에서만 획득된 것이라고 보는 지구통계학적 프로세스(12장)와 대조된다. 그러므로, 지구통계학적 연구의 주된 관심은 관측 지점에 있는 것이 아니라 비관측 지점의 값을 추정하는 데 있다. 포인트 패턴 분석은 해당 지역의 모든 지점에서 관측이 이루어질 수 있다고 가정한다. 이는 특정 지점에 포인트가 없다면 해당 지점에서 관측이 이루어지지 않았다는 것이 아니라(지구통계학적 프로세스) 관측은 이루어졌지만 현상이 발생하지 않았다는 것을 의미한다. 확률 과정(random process)의 용어로 말하자면, 포인트 패턴 분석에서 위치는 확률 변수이지만, 지구통계학적 분석에서 측정 변수는 고정 위치를 가진 랜덤 필드이다.

이 장에서는 **spatstat** 패키지(Baddeley, Turner, Rubak 2022)를 활용한 포인트 패턴 분석의 기초만을 다룬다. 포인트 패턴 이론과 **spatstat** 및 관련 패키지의 사용에 대한 포괄적인 내용은 Baddeley, Rubak, Turner(2015)의 **spatstat** 책에 잘 나타나 있다. 이 장에서는 특정 주제만을 다루게 되는데, 해당 주제가 누락된 주제에 비해 더 중요하다는 의미는 전혀 아니다. **spatstat** 패키지와 보다 공간데이터사이언스 지향 패키지인 **sf** 및 **stars** 간의 인터페이스를 설명하는데 공간을 할애하려 한다. 포인트 패턴 분석을 소개하는 또 다른 책으로는 Stoyan et al.(2017)가 있다. 시공간 포인트 프로세스 분석용 R 패키지인 **stpp**(Gabriel et al. 2022)에 대한 내용은 Gabriel, Rowlingson, Diggle(2013)를 참고할 수 있다.

포인트 패턴 분석에서 중요한 개념은 포인트 *패턴*과 포인트 *프로세스*를 구분하는 것이다. 포인트 프로세스는 포인트 패턴을 생성하는 확률 과정(stochastic process)이다. 데이터셋은 항상 포인트 패턴이며, 추론은 우리가 관찰한 패턴을 생성할 수 있는 프로세스의 특성을 파악하는 것이다. 공간 포인트 프로세스의 특성에는 다음이 포함된다.

* 1차 특성: 강도 함수는 단위 면적당 포인트의 수를 측정하며, 이 함수는 *이질적*(inhomogeneous) 포인트 프로세스의 경우 공간적으로 변동한다.
* 2차 특성: 주어진 강도 함수(고정 혹은 변동) 하에서, 포인트들이 *서로* 독립적으로 분포하는지, 서로 끌어당기는 경향이 있는지(군집화), 또는 서로 밀어내는 경향이 있는지(완전 공간 무작위성(complete spatial randomness)보다 더 규칙적으로 분포하는지)를 묘사한다.

## 11.1 관측 윈도우

포인트 패턴을 관측 윈도우를 가진다. 다음의 코드를 통해 무작위 포인트 패턴을 생성한다.

library(sf)  
# Linking to GEOS 3.11.1, GDAL 3.6.4, PROJ 9.1.1; sf\_use\_s2() is TRUE  
n <- 30  
set.seed(13531) # remove this to create another random sequence  
xy <- data.frame(x = runif(n), y = runif(n)) |>   
 st\_as\_sf(coords = c("x", "y"))

이 포인트들은 영역 내에 고르게 분포하고 있는데, 완전히 공간적으로 무작위로 분포하고 표현할 수도 있다. 그런데 영역의 크기가 확대되면 고르게 분포하고 있다는 말을 하기 어려워 진다. 예를 들어 아래의 코드로 두 개의 정사각형 영역(w1과 w2)을 생성한다(그림 11.1).

w1 <- st\_bbox(c(xmin = 0, ymin = 0, xmax = 1, ymax = 1)) |>   
 st\_as\_sfc()   
w2 <- st\_sfc(st\_point(c(1, 0.5))) |> st\_buffer(1.2)

|  |
| --- |
| 그림 11.1: 관찰 윈도우(회색)에 따라 동일한 포인트 패턴이 공간적으로 완전히 무작위로 나타날 수도 있고(왼쪽), 군집을 이루고 있는 것으로 나타날 수도 있다(오른쪽). |

**spatstat** 패키지는 포인트 패턴을 ppp 클라스의 객체로 저장한다. 모든 ppp 객체는 포인트 위치 정보와 관측 윈도우(owin 클라스의 객체)를 가진다. 다음과 같이 포인트로부터 ppp 객체로 생성할 수 있다.

library(spatstat) |> suppressPackageStartupMessages()  
as.ppp(xy)  
# Planar point pattern: 30 points  
# window: rectangle = [0.009, 0.999] x [0.103, 0.996] units

여기서 윈도우가 지정되지 않으면 포인트들의 바운딩 박스가 관측 윈도우로 사용됨을 볼 수 있다. 데이터셋의 첫 번째 피처로 폴리곤 지오메트리를 추가하면, 해당 폴리곤이 관측 윈도우로 사용된다.

(pp1 <- c(w1, st\_geometry(xy)) |> as.ppp())  
# Planar point pattern: 30 points  
# window: polygonal boundary  
# enclosing rectangle: [0, 1] x [0, 1] units  
c1 <- st\_buffer(st\_centroid(w2), 1.2)  
(pp2 <- c(c1, st\_geometry(xy)) |> as.ppp())  
# Planar point pattern: 30 points  
# window: polygonal boundary  
# enclosing rectangle: [-0.2, 2.2] x [-0.7, 1.7] units

포인트 패턴의 등질성(homogeneity)(역자주: 기대 밀도가 어는 지점에서나 동일한가의 여부)을 검정하기 위해 적절한 방격 레이아웃(그림 11.2에는 3 3 레이아웃이 나타나 있다)를 사용하여 방격 빈도(quadrat count)(방격별 포인트 수)를 구할 수 있다.

par(mfrow = c(1, 2), mar = rep(0, 4))  
q1 <- quadratcount(pp1, nx=3, ny=3)  
q2 <- quadratcount(pp2, nx=3, ny=3)  
plot(q1, main = "")  
plot(xy, add = TRUE)  
plot(q2, main = "")  
plot(xy, add = TRUE)

|  |
| --- |
| 그림 11.2: 두 포인트 패턴의 3 X 3 방격 빈도 |

이 방격 빈도 정보를 활용해 다음과 같이 검정을 수행한다.

quadrat.test(pp1, nx=3, ny=3)  
# Warning: Some expected counts are small; chi^2 approximation may be  
# inaccurate  
#   
# Chi-squared test of CSR using quadrat counts  
#   
# data: pp1  
# X2 = 8, df = 8, p-value = 0.9  
# alternative hypothesis: two.sided  
#   
# Quadrats: 9 tiles (irregular windows)  
quadrat.test(pp2, nx=3, ny=3)  
# Warning: Some expected counts are small; chi^2 approximation may be  
# inaccurate  
#   
# Chi-squared test of CSR using quadrat counts  
#   
# data: pp2  
# X2 = 43, df = 8, p-value = 2e-06  
# alternative hypothesis: two.sided  
#   
# Quadrats: 9 tiles (irregular windows)

이는 두 번째 사례가 CSR(완전공간무작위성) 패턴이 아님을 나타낸다. 경고 메시지에서 언급된 것처럼 기대 빈도 수가 너무 적기 때문에 유의 확률(p-값)을 매우 신중하게 해석해야 한다.

density 함수를 통해 커널 밀도를 계산할 수 있으며 커널의 형태와 탐색반경에 조정할 수 있다. 탐색반경 파라미터인 sigma는 bw.diggle 함수의 교차 검증을 통해 지정되며, 그림 11.3에 생성된 밀도면이 나타나 있다.

den1 <- density(pp1, sigma = bw.diggle)  
den2 <- density(pp2, sigma = bw.diggle)

|  |
| --- |
| 그림 11.3: 두 포인트 패턴에 대한 커널 밀도면 |

이러한 방식으로 생성된 밀도 맵은 명백히 래스터 이미지이며, 이를 stars 객체로 전환할 수 있다.

library(stars)  
# Loading required package: abind  
s1 <- st\_as\_stars(den1)  
(s2 <- st\_as\_stars(den2))  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# v 6.28e-15 0.000153 0.304 6.77 13.1 42.7 3492  
# dimension(s):  
# from to offset delta x/y  
# x 1 128 -0.2 0.0187 [x]  
# y 1 128 1.7 -0.0187 [y]

밀도면 하부 체적(역자주: 각 셀별로 면적과 밀도값을 곱하여 셀 체적을 구하고 모든 셀에 대한 이 체적값을 합한 값을 의미한다)을 계산하면 샘플 크기(30)와 유사해진다는 것을 다음의 코드를 통해 확인할 수 있다.

s1$a <- st\_area(s1) |> suppressMessages()  
s2$a <- st\_area(s2) |> suppressMessages()  
with(s1, sum(v \* a, na.rm = TRUE))  
# [1] 29  
with(s2, sum(v \* a, na.rm = TRUE))  
# [1] 30.7

여기에 밀도면을 외부 변수의 함수로 나타내는 모델링을 적용하면 좀 더 흥미로운 결과를 얻을 수 있다. 예를 들어, pp2의 밀도를 푸아송 포인트 패턴 프로세스으로 모델링하고자 한다고 가정해 보자(즉, 포인트 간에 상호작용이 없음을 가정). 여기서 밀도 강조는 “클러스터” 중심으로부터의 거리의 함수이며, 이 거리값은 stars 객체에 포함되어 있다.

pt <- st\_sfc(st\_point(c(0.5, 0.5)))  
st\_as\_sf(s2, as\_points = TRUE, na.rm = FALSE) |>  
 st\_distance(pt) -> s2$dist

그런 다음 ppm을 사용하여 밀도를 모델링할 수 있으며, 포인트 패턴 객체의 *이름*이 formula의 왼쪽에 사용된다.

(m <- ppm(pp2 ~ dist, data = list(dist = as.im(s2["dist"]))))  
# Nonstationary Poisson process  
# Fitted to point pattern dataset 'pp2'  
#   
# Log intensity: ~dist  
#   
# Fitted trend coefficients:  
# (Intercept) dist   
# 4.54 -4.24   
#   
# Estimate S.E. CI95.lo CI95.hi Ztest Zval  
# (Intercept) 4.54 0.341 3.87 5.21 \*\*\* 13.32  
# dist -4.24 0.700 -5.62 -2.87 \*\*\* -6.06

반환된 객체는 ppm 클래스이며, 이를 플롯할 수 있다. 그림 11.4는 예측 밀도면을 보여준다. 예측 표준 오차도 플롯할 수 있다.

|  |
| --- |
| 그림 11.4: ppm 모델에 의거한 예측 밀도면 |

모델에는 예측 메소드도 있어, 이를 통해 im 객체를 반환하며, 이 객체는 다음과 같이 stars 객체로 전환될 수 있다.

predict(m, covariates = list(dist = as.im(s2["dist"]))) |>  
 st\_as\_stars()  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# v 0.0698 0.529 2.13 6.62 7.3 89.7 3492  
# dimension(s):  
# from to offset delta x/y  
# x 1 128 -0.2 0.0187 [x]  
# y 1 128 1.7 -0.0188 [y]

## 11.2 CRS

**spatstat** 패키지의 모든 루틴은 데카르트 좌표계를 가진 2차원 데이터를 위해 설계되어 있다. 타원 좌표계를 가진 객체를 전환하려고 하면 오류가 발생한다.

system.file("gpkg/nc.gpkg", package = "sf") |>   
 read\_sf() |>  
 st\_geometry() |>  
 st\_centroid() |>  
 as.ppp()  
# Error: Only projected coordinates may be converted to spatstat  
# class objects

**spatstat** 패키지의 데이터 구조로 전환되고 나면, 처음 사용했던 CSR이 손실된다. 이를 sf 또는 stars 객체로 되돌리기 위해서는 st\_set\_crs를 사용하면 된다.

## 11.3 마크 포인트 패턴, 선형 네트워크 상의 포인트

**spatstat** 패키지에서는 몇 가지 추가 데이터 유형을 서로 전환할 수 있다. 마크(marked) 포인트 패턴은 각 포인트에 대해 범주형 레이블 또는 숫자 레이블을 가진 포인트 패턴이다. **spatstat** 패키지에 포함되어 있는 마크 포인트 패턴으로 longleaf 소나무 데이터셋이 있는데, 이 데이터셋에는 가슴 높이에서의 나무 직경이 숫자 마크로 포함되어 있다.

longleaf  
# Marked planar point pattern: 584 points  
# marks are numeric, of storage type 'double'  
# window: rectangle = [0, 200] x [0, 200] metres  
ll <- st\_as\_sf(longleaf)  
print(ll, n = 3)  
# Simple feature collection with 585 features and 2 fields  
# Geometry type: GEOMETRY  
# Dimension: XY  
# Bounding box: xmin: 0 ymin: 0 xmax: 200 ymax: 200  
# CRS: NA  
# First 3 features:  
# spatstat.geom..marks.x. label geom  
# NA NA window POLYGON ((0 0, 200 0, 200 2...  
# 1 32.9 point POINT (200 8.8)  
# 2 53.5 point POINT (199 10)

값은 다음과 같이 ppp객체로 다시 전환할 수 있다.

as.ppp(ll)  
# Warning in as.ppp.sf(ll): only first attribute column is used for  
# marks  
# Marked planar point pattern: 584 points  
# marks are numeric, of storage type 'double'  
# window: polygonal boundary  
# enclosing rectangle: [0, 200] x [0, 200] units

**spatstat** 패키지의 psp 클래스의 선분은 LINESTRING 지오메트리를 가진 sf 피처로 전환 및 재전환이 가능한데, 관측 윈도우를 가진 POLYGON 피처 하나를 우선적으로 포함하게 된다.

print(st\_as\_sf(copper$SouthLines), n = 5)  
# Simple feature collection with 91 features and 1 field  
# Geometry type: GEOMETRY  
# Dimension: XY  
# Bounding box: xmin: -0.335 ymin: 0.19 xmax: 35 ymax: 158  
# CRS: NA  
# First 5 features:  
# label geom  
# 1 window POLYGON ((-0.335 0.19, 35 0...  
# 2 segment LINESTRING (3.36 0.19, 10.4...  
# 3 segment LINESTRING (12.5 0.263, 11....  
# 4 segment LINESTRING (11.2 0.197, -0....  
# 5 segment LINESTRING (6.35 12.8, 16.5...

마지막으로, 선형 네트워크상의 포인트 패턴은 **spatstat** 패키지에서 lpp 객체로 표현되는데 다음과 같이 sf 객체로 전환할 수 있다.

print(st\_as\_sf(chicago), n = 5)  
# Simple feature collection with 620 features and 4 fields  
# Geometry type: GEOMETRY  
# Dimension: XY  
# Bounding box: xmin: 0.389 ymin: 153 xmax: 1280 ymax: 1280  
# CRS: NA  
# First 5 features:  
# label seg tp marks geom  
# 1 window NA NA <NA> POLYGON ((0.389 153, 1282 1...  
# 2 segment NA NA <NA> LINESTRING (0.389 1254, 110...  
# 3 segment NA NA <NA> LINESTRING (110 1252, 111 1...  
# 4 segment NA NA <NA> LINESTRING (110 1252, 198 1...  
# 5 segment NA NA <NA> LINESTRING (198 1277, 198 1...

여기서는 첫 다섯 개 피처만 나타나 있어 금방 알아챌 수는 없지만, label 변수를 통해 알 수 있듯이 포인트들도 이 객체에 포함되어 있다.

table(st\_as\_sf(chicago)$label)  
#   
# point segment window   
# 116 503 1

네트워크의 *구조*에 대한 정보(역자주: 네트워크 연결성에 대한 위상 정보), 즉 LINESTRING 지오메트리가 어떻게 연결되어 있는지는 sf 객체에 포함되어 있지 않다. 이런 측면에서 **sfnetworks** 패키지(van der Meer et al. 2022)는 좋은 대안일 수 있는데, 네트워크 위상 정보를 다룰 수 있을 뿐만 아니라 OpenStreetMaps에서 가져온 네트워크 데이터를 **spatstat** 패키지로 전달할 수도 있다.

## 11.4 공간 샘플링과 포인트 프로세스를 시뮬레이션하기

**sf** 패키지에는 MULTIPOINT, 선형 또는 폴리곤 지오메트리에서 포인트를 샘플링하는 st\_sample 메소드가 포함되어 있으며, 다양한 공간 샘플링 전략을 사용한다. 이 메소드는 기본적으로 “random”, “hexagonal”, “Fibonacci”(11.5절), 및 “regular” 옵션을 지원하는데, 여기서 “regular”는 정사각형 격자에서 샘플링하는 것을 의미하고 “hexagonal”은 본질적으로 삼각형 격자를 제공한다. “random” 유형만 요청된 포인트의 수를 정확히 반환하고, 다른 유형에서는 근사값만을 반환한다.

st\_sample 함수는 **spatstat** 패키지의 포인트 프로세스 시뮬레이션 함수와도 인터페이스를 제공하는데, 샘플링 유형에 대해 다른 값을 선택하는 경우가 해당된다. 예를 들어, type = Thomas로 설정하면 **spatstat** 함수 rThomas가 호출된다(그림 11.5).

kappa <- 30 / st\_area(w2) # intensity  
th <- st\_sample(w2, kappa = kappa, mu = 3, scale = 0.05,   
 type = "Thomas")  
nrow(th)  
# [1] 90

|  |
| --- |
| 그림 11.5: mu =3, scale = 0.05로 설정한 Thomas 프로세스 |

?rThomas를 실행하며 파라미터 kappa, mu, 및 scale의 의미에 대한 자세한 설명을 얻을 수 있다. 포인트 프로세스 시뮬레이션을 수행할 때는 샘플 크기가 아닌 강도가 주어진다. 이 방법으로 얻어진 관측 윈도우 내의 샘플 크기는 하나의 확률 변수가 된다.

## 11.5 구체 상의 포인트를 시뮬레이션하기

sf에서 기본적으로 지원하는 또 다른 공간 무작위 샘플링 유형은 구면에서의 무작위 포인트 시뮬레이션이다. 그림 11.6에 한 예시가 나타나 있으며, 포인트들이 모두 해양에만 존재한다. 구면에 규칙적 포인트 패턴을 시뮬레이션 하려면 st\_sample 함수에서 아규먼트로 type = "Fibonacci"로 지정하면 된다(González 2010).

|  |
| --- |
| 그림 11.6: 구체의 해양부에 한정된 샘플 포인트: 무작위 패턴(왼쪽)과 규칙 패턴(오른쪽)이 정사 도법의 지도 상에 나타나 있다. |

## 11.6 연습문제

1. **spatstat** 패키지에서 plot(longleaf)로 얻은 플롯과 동일한 것을 ggplot2 패키지의 geom\_sf() 함수와 sf::plot()을 사용하여 생성하라.
2. 12장에서 사용된 NO 데이터의 샘플 위치를 적절한 관측 윈도우와 함께 ppp 객체로 전환하라.
3. NO 데이터셋을 가지고 밀도를 계산하고 밀도면을 플롯하라. 밀도면을 stars 객체로 전환하고, 표면 하부 체적을 계산하라.

# 12. 공간적 인터폴레이션

공간적 인터폴레이션은 관측된 위치를 바탕으로 관측되지 않은 공간 위치에서 공간적으로 연속적인 변수(필드)의 값을 추정하는 활동이다. 공간적 인터폴레이션을 위한 통계적 방법론을 지구통계학(geostatistics)이라고 부르며, 공간적으로 연속적인 현상의 모델링, 예측, 시뮬레이션에 중점을 둔다. 공간적 인터폴레이션의 전형적인 상황은 결측값 문제이다. 즉, 한정된 샘플 위치 에서 특정 현상의 속성 가 관측되었다고 했을 때, 이를 바탕으로 연구 지역 내의 모든 위치 에서 해당 속성 값을 알아내고자 한다. 결국 관측되지 않은 위치에 대해 속성값을 예측해야 하는 상황이다. 이러한 과정을 크리깅(kriging) 또는 가우시안 프로세스 예측(Gaussian Process prediction)이라고도 부른다. 에 화이트 노이즈 성분인 가 포함된 경우 (측정 에러 포함)로 주어질 수 있는데, 이 경우 대신 를 예측하거나 시뮬레이션하는 것이 대안적 목표가 될 수 있다. 이것을 보통 *공간 필터링*(spatial filtering) 또는 *평활화*(smoothing)라고 부른다.

이 장에서는 지구통계학적 데이터를 다루는 간단한 접근법, 간단한 인터폴레이션 기법, 그리고 공간적 자기상관, 공간 예측 및 시뮬레이션 모델링에 대해 다룬다. 13장에서는 다변량 및 시공간 지구통계하적 모델과 같은 보다 더 복잡한 모델에 중점을 둔다. 우리는 **gstat** 패키지(Pebesma and Graeler 2022; Pebesma 2004)를 주로 사용하게 되는데, 이 패키지는 비교적 다양한 비베이지안 지구통계적 분석 모델과 옵션을 제공하는 사용한다. R로 구현된 베이지안 방법은 Diggle, Tawn, and Moyeed (1998), Diggle and Ribeiro Jr. (2007), Blangiardo and Cameletti (2015), Wikle, Zammit-Mangion, and Cressie (2019)에서 찾을 수 있다. 대용량 데이터셋에 활용되는 기법의 개요와 비교는 Heaton et al. (2018)에서 제공된다.

## 12.1 첫 번째 데이터셋

**gstat** 패키지에 포함되어 있는 평균 NO₂ 데이터셋을 읽어 들인다. 이것은 13장에서 준비된다.

library(tidyverse) |> suppressPackageStartupMessages()  
no2 <- read\_csv(system.file("external/no2.csv",   
 package = "gstat"), show\_col\_types = FALSE)

이 데이터셋에 UTM 투영을 적용하여 sf 객체로 전환한다.

library(sf)  
# Linking to GEOS 3.11.1, GDAL 3.6.4, PROJ 9.1.1; sf\_use\_s2() is TRUE  
crs <- st\_crs("EPSG:32632")  
st\_as\_sf(no2, crs = "OGC:CRS84", coords =   
 c("station\_longitude\_deg", "station\_latitude\_deg")) |>  
 st\_transform(crs) -> no2.sf

그리고 나서, 국가 행정 경계를 불러 들이고, ggplot 패키지를 이용해 지도를 그린다(그림 12.1)

read\_sf("data/de\_nuts1.gpkg") |> st\_transform(crs) -> de

|  |
| --- |
| 그림 12.1: 독일의 농촌 지역 관측소들에 높은 평균 NO 값이 집중해 있다. |

이 데이터를 바탕으로 공간적 인터폴레이션을 수행하려면 먼저 인터폴레이션을 할 위치를 결정해야 한다. 이는 보통 연구 대상 지역 전체를 덮는 규칙적 그리드를 사용하여 수행된다. 객체 de의 국가 윤곽에 맞추어 독일 전역을 커버하는 10 km × 10 km의 규칙적 그리드를 생성한다.

library(stars) |> suppressPackageStartupMessages()  
st\_bbox(de) |>  
 st\_as\_stars(dx = 10000) |>  
 st\_crop(de) -> grd  
grd  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# values 0 0 0 0 0 0 2076  
# dimension(s):  
# from to offset delta refsys x/y  
# x 1 65 280741 10000 WGS 84 / UTM z... [x]  
# y 1 87 6101239 -10000 WGS 84 / UTM z... [y]

그리드 셀의 크기를 너무 세밀하게 하지 않은 것은 플롯 상에서 눈으로 확인해 보려는 의도이다.

가장 간단한 공간적 인터폴레이션 기법은 역거리 가중법(inverse distance weighted)일 것이다. 인터폴레이션이 적용되는 지점으로부터의 거리의 역비례에 따라 가중치를 부여하여 가중 평균을 산출하는 기법이다.

여기서 으로 주어진다. 역거리를 위한 지수()는 보통 2가 사용되는데, 교차 검증을 통해 최적화할 수도 있다. **gstat** 패키지의 idw 함수를 활용하면 역거리 가중 인터폴레이션을 수행할 수 있다.

library(gstat)  
i <- idw(NO2~1, no2.sf, grd)  
# [inverse distance weighted interpolation]

결과는 그림 12.2에 나타나 있다.

|  |
| --- |
| 그림 12.2: 독일 NO 집중도에 대한 역거리 가중 인터폴레이션의 결과 |

## 12.2 샘플 베리오그램

지구통계학적 방법을 사용하여 공간 예측을 수행하려면 먼저 평균과 공간적 자기상관을 위한 모델을 수립해야 한다. 가장 단순한 모델은 로 주어질 수 있는데, 평균은 알려지지 않은 고정값 이고 공간적 자기상관은 의 형태로 주어지는 베리오그램(variogram)을 통해 모델화된다. 유한 분산 를 가진 프로세스의 경우는 가 성립하는데, 이는 베리오그램이 코베리오그램(covariogram) 또는 공분산 함수(covariance function)와 관련되어 있음을 보여준다.

샘플 베리오그램은 거리 인터벌 별 에 대한 추정값을 계산함으로써 획득된다.

여기에서 는 거리 구간 에 해당하는 모든 샘플 쌍의 개수를 의미한다(역자주: 샘플 쌍의 거리가 해당 거리 구간 내에 포함되는 모든 샘플 쌍의 개수). **gstat** 패키지의 variogram 함수를 통해 샘플 베리오그램을 계산할 수 있다.

v <- variogram(NO2~1, no2.sf)

그림 12.3은 베리오그램의 결과를 플롯으로 나타낸 것이다.

|  |
| --- |
| 그림 12.3: 샘플 베리오그램 플롯 |

variogram 함수는 내장된 디폴트 값을 최대 거리(cutoff: 경계 상자의 대각선 길이의 삼분의 일)와 (상수) 구간 너비(width: cutoff를 15로 나눈 값)에 적용하는데. 디폴트 값은 다음을 통해 변경할 수 있다.

v0 <- variogram(NO2~1, no2.sf, cutoff = 100000, width = 10000)

새로운 값의 결과는 그림 12.4에 나타나 있다.

|  |
| --- |
| 그림 12.4: 디폴트 값을 조정해 생성된 샘플 베리오그램 |

공식 NO₂ ~ 1은 데이터 파일(NO₂)에서 관심 변수를 선택하고 평균 모델을 지정하는 데 사용된다. 여기서 ~1은 절편만 있는(알려지지 않은 고정 평균) 모델을 지정한다.

## 12.3 베리오그램 모델 적합

공간 예측을 진행하기 위해서는 위에서 도출한 거리 구간별 추정값이 아니라 이론적으로 모든 거리 에 적용될 수 있는 베리오그램 *모델* 가 필요하다. 위의 구간별 추정값을 하나의 직선으로 연결하거나 구간별 추정값을 해당 구간에 대한 고정값이라고 가정하게 되면 문제가 발생한다. 즉, 이것은 비양수 정의 공분산 행렬(non-positive definite covariance matrices)을 갖는 통계 모델을 전제하는 것이 되고, 이것을 예측에 사용하는 것은 불가능한다.

이를 피하기 위해 우리는 파라메트릭 모델 를 적합하여 추정값 를 구한다. 여기서 는 를 추정하는 데 관여한 모든 값의 평균값으로 설정한다. 예를 들어, 다음과 같은 지수 베리오그램 모델을 적합할 수 있다.

v.m <- fit.variogram(v, vgm(1, "Exp", 50000, 1))

결과는 그림 12.5에 나타나 있다.

|  |
| --- |
| 그림 12.5: 샘플 베리오그램(가운데가 비어 있는 점) 상에 가중 최소 제곱법(실선) 및 최대 우도 추정법(점선)을 사용하여 적합한 모델이 표시되어 있다. |

라인의 적합에 다음을 최소화하는 가중 최소 제곱법이 적용되었다.

여기에서 가중치의 디폴트값은 으로 주어진다. fit.method 아규먼트를 통해 다른 가중치 옵션을 선택할 수 있다.

가중 최소 제곱법 적합의 대안으로 최대 우도(ML) 또는 제한 최대 우도(parameter estimation) 모수 추정법을 사용할 수 있으며(Kitanidis and Lane 1985), 이 사례에서는 그림 12.5에서 점선으로 나타난 것처럼 비교적 유사한 적합 모델이 도출되었다. ML 유형 접근법의 장점은 방정식 12.1에서 거리 구간 나 방정식 12.2에서 가중치 를 선택할 필요가 없다는 것이다. 단점으로는 데이터가 다변량 정규 분포를 보인다는 강한 가정에 기반하고 있고, 더 큰 데이터셋의 경우 관측 수만큼의 선형 시스템을 반복적으로 풀어야 한다는 점 등이 있다. Heaton et al. (2018)은 대규모 데이터셋에 모델을 적합하는 데 특화된 접근법을 비교한다.

## 12.4 크리깅 인터폴레이션

일반적으로 인터폴레이션은 연구 대상 지역을 덮는 규칙적인 그리드 상의 지점들을 대상으로 이루어진다. 먼저 해당 지역을 커버하면서 그 외부는 NA로 채워진 래스터(stars 객체)를 생성한다.

크리깅은 연구 대상 지역 내의 임의의 지점 에 대한 속성값을 예측한다. **gstat** 패키지의 krige 함수를 통해 NO₂를 크리깅할 수 있으며, 이때 경향면 모델, 데이터, 예측 그리드, 베리오그램 모델을 아규먼트로 전달한다(그림 12.6).

k <- krige(NO2~1, no2.sf, grd, v.m)  
# [using ordinary kriging]

ggplot() + geom\_stars(data = k, aes(fill = var1.pred, x = x, y = y)) +   
 xlab(NULL) + ylab(NULL) +  
 geom\_sf(data = st\_cast(de, "MULTILINESTRING")) +   
 geom\_sf(data = no2.sf) +  
 coord\_sf(lims\_method = "geometry\_bbox")

|  |
| --- |
| 그림 12.6: 독일 NO 농도에 대한 크리깅의 결과 |

## 12.5 에어리어 평균: 블록 크리깅

에어리어 평균을 계산하는 방법은 여러 가지가 있다. 가장 간단한 방법은 대상 폴리곤 내부에 포함된 포인트 샘플의 평균을 취하는 것이다.

a <- aggregate(no2.sf["NO2"], by = de, FUN = mean)

더 복잡한 방법으로는 블록 크리깅(block kriging, Journel and Huijbregts 1978)이 있는데, 타깃 지역의 평균값을 추정하기 위해 모든 데이터를 활용한다. krige 함수에서 newdata 인수에 타깃 지역(폴리곤)을 전달하여 이를 수행한다.

b <- krige(NO2~1, no2.sf, de, v.m)  
# [using ordinary kriging]

두 지도를 하나의 객체로 병합하여 단일 플롯을 생성한다(그림 12.7).

b$sample <- a$NO2  
b$kriging <- b$var1.pred

b |> select(sample, kriging) |>   
 pivot\_longer(1:2, names\_to = "var", values\_to = "NO2") -> b2  
b2$var <- factor(b2$var, levels = c("sample", "kriging"))  
ggplot() + geom\_sf(data = b2, mapping = aes(fill = NO2)) + facet\_wrap(~var) +  
 scale\_fill\_gradientn(colors = sf.colors(20))

|  |
| --- |
| 그림 12.7: 단순 평균계산(왼쪽)과 블록 크리깅(오른쪽)을 통해 계산된 NO 농도의 집계값 |

패턴은 유사하지만, 단순 평균계산을 통한 샘플 평균이 블록 크리깅의 값보다 변동성이 더 크다는 것을 알 수 있다. 이는 크리깅의 평활화 효과 때문일 수 있는데, 집계 영역 외부의 데이터 포인트에도 가중치가 부여되기 때문이다.

표준 오차에 대한 대략적인 추정치는 를 통해 얻을 수 있다.

SE <- function(x) sqrt(var(x)/length(x))  
a <- aggregate(no2.sf["NO2"], de, SE)

샘플이 공간적으로 무작위 표본 추출을 통해 얻어진 경우, 디자인-기반 추론(10.4절)에서 실제 추정치가 이와 같았을 것이다. 블록 크리깅 분산은 모델-기반 추정치로, 크리깅의 부산물로 계산된다. 그림 12.8에서 두 값을 비교해 보면, 단순 평균계산 접근법이 블록 크리깅에 비해 구역 평균의 예측 오차가 더 큰 변동성을 보일 뿐만 아니라 값도 더 크다는 것을 확인할 수 있다.

|  |
| --- |
| 그림 12.8: 단순 평균계산법(왼쪽)과 블록 크리깅(오른쪽)을 통해 획득된 평균 NO 농도의 표준 오차 |

## 12.6 조건부 시뮬레이션

필드 의 조건부 평균이 아닌 하나 혹은 여러개의 조건부 실현이 필요할 경우, 조건부 시뮬레이션을 활용할 수 있다. 조건부 실현이 필요한 경우는 비선형 함수인 을 통해 의 구역 평균 값, 즉 를 추정해야 하는 상황이 해당된다. 간단한 예로는 가 특정 임계값을 초과하는 지역이 존재하는 경우이다.

gstat 패키지의 기본 접근 방식은 이를 위해 순차 시뮬레이션 알고리즘을 사용하는 것이다. 이 알고리즘은 예측이 이루어지는 위치들을 무작위로 순회하며 각 위치에서 다음과 같은 작업을 수행한다.

* 크리깅 예측을 수행한다.
* 크리깅 분산과 동일한 평균과 분산을 가진 정규 분포에서 난수를 생성한다.
* 이 값을 조건부 데이터셋에 추가한다.
* 새로운 무작위 시뮬레이션 위치를 찾는다.

이것을 모든 위치에 반복 수행한다.

gstat 패키지의 krige 함수가 이를 수행하며 nsim 아규먼트는 양수로 설정된다.

set.seed(13341)  
(s <- krige(NO2~1, no2.sf, grd, v.m, nmax = 30, nsim = 6))  
# drawing 6 GLS realisations of beta...  
# [using conditional Gaussian simulation]  
# stars object with 3 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# var1 -5.7 6.12 8.68 8.88 11.5 23.9 12456  
# dimension(s):  
# from to offset delta refsys values x/y  
# x 1 65 280741 10000 WGS 84 / UTM z... NULL [x]  
# y 1 87 6101239 -10000 WGS 84 / UTM z... NULL [y]  
# sample 1 6 NA NA NA sim1,...,sim6

set.seed()를 설정한 것은 시뮬레이션 결과가 달라지지 않도록 하기 위해서이다.

nmax 아규먼트의 설정을 통해 크리깅 추정 시 포함할 최근접 이웃의 (최대) 개수를 제한하는 것이 필요하다. 왜냐하면 단계가 진해되면서 데이터셋이 계속 증가하기 때문에 계산 시간이 길어지고 메모리 요구량이 커지기 때문이다. 조건부 시뮬레이션의 결과가 그림 12.9에 나타나 있다.

|  |
| --- |
| 그림 12.9: NO 농도에 대한 6가지의 조건부 시뮬레이션 결과 |

gstat 패키지에는 최근 조건부 시뮬레이션의 대체 기법이 추가되었으며, 원형 임베딩(circular embedding) 기법을 구현한 krigeSimCE(Davies와 Bryant, 2013)와 터닝 밴드(turning band) 기법을 구현한 krigeSTSimTB(Schlather, 2011)가 있다. 이러한 기법들은 대규모 데이터셋이나 시공간 데이터셋의 조건부 시뮬레이션에 유용하다.

## 12.7 경향면 모델

이 장에서 사용된 크리깅과 조건부 시뮬레이션은 모든 공간 변동성이 공간 공분산 모델로 특징지어지는 무작위 과정이라고 가정한다. 만약 타깃 변수와 의미 있게 상관된 다른 변수가 있는 경우, 이러한 변수를 경향면을 위한 선형 회귀 모델에 사용할 수 있다.

여기서 , 는 절편, 는 변수에 대한 회귀계수이다. 변수를 추가하면 보통 잔차 의 공간적 자기상관과 분산이 모두 감소하여, 더 정확한 예측과 유사한 조건부 시뮬레이션 결과를 얻을 수 있다. NO의 변동에 대해 인구 밀도 변수를 사용하는 예를 살펴볼 것이다.

### 12.7.1 인구 밀도 그리드

대기의 NO에 대한 예측 변수로 인구 밀도를 사용한다. NO는 주로 교통에 의해 발생하며, 교통은 인구 밀도가 높은 지역에서 더 집중된다. 인구 밀도는 2011년 인구 조사에서 얻어진 데이터로, 100m × 100m 그리드 셀당 거주자 수가 CSV 파일에 포함되어 있다. 우리는 이 데이터를 타깃 그리드 셀에 맞춰 합산함으로써 새로운 집계 데이터를 생성할 수 있다.

v <- vroom::vroom("aq/pop/Zensus\_Bevoelkerung\_100m-Gitter.csv")  
v |> filter(Einwohner > 0) |>   
 select(-Gitter\_ID\_100m) |>  
 st\_as\_sf(coords = c("x\_mp\_100m", "y\_mp\_100m"), crs = 3035) |>  
 st\_transform(st\_crs(grd)) -> b  
a <- aggregate(b, st\_as\_sf(grd, na.rm = FALSE), sum)

위의 코드를 통해 집계된 타깃 그리드 셀의 인구 수가 a에 저장된다. 인구 밀도를 구하기 위해서는 각 셀의 면적이 필요한데, 국경에 걸쳐 있는 그리드 셀의 경우 면적이 10km × 10km보다 작을 것이다.

grd$ID <- 1:prod(dim(grd)) # to identify grid cells  
ii <- st\_intersects(grd["ID"],  
 st\_cast(st\_union(de), "MULTILINESTRING"), as\_points = FALSE)  
grd\_sf <- st\_as\_sf(grd["ID"], na.rm = FALSE)[lengths(ii) > 0,]  
st\_agr(grd\_sf) = "identity"  
iii <- st\_intersection(grd\_sf, st\_union(de))  
grd$area <- st\_area(grd)[[1]] +   
 units::set\_units(grd$values, m^2)  
grd$area[iii$ID] <- st\_area(iii)

위의 두 단계 과정, 즉 먼저 국경에 걸쳐 있는 셀을 찾고 그 후에 면적을 계산하는 과정을 거치지 말고, 모든 셀에 바로 st\_intersection 함수를 적용하는 방법도 있겠지만 이는 오퍼레이션 시간이 긴 단점이 있다. 인구 수와 면적을 통해 인구 밀도를 계산하고(그림 12.10), 총계가 맞는지 확인해 본다.

grd$pop\_dens <- a$Einwohner / grd$area  
sum(grd$pop\_dens \* grd$area, na.rm = TRUE) # verify  
# 80323301 [1]  
sum(b$Einwohner)  
# [1] 80324282

두 값의 일치도가 상당히 높다는 것을 알 수 있다. st\_interpolate\_aw 함수를 사용했다면 정확히 일치하는 결과를 얻었을 것이다.

|  |
| --- |
| 그림 12.10: 100 m X 100 m 그리드 셀별 인구 밀도 값 |

인구 수를 인구 밀도로 전환하기 위해서는 해당 인구 수를 100m × 100m 격자 셀의 수로 나누어야 한다.

모니터링 네트워크 스테이션에서 인구 밀도 값을 얻기 위해 st\_extract 함수를 사용한다.

grd |>  
 select("pop\_dens") |>  
 st\_extract(no2.sf) |>  
 pull("pop\_dens") |>   
 mutate(no2.sf, pop\_dens = \_) -> no2.sf

그런 다음 모니터링 스테이션 위치에서 NO와 인구 밀도 간의 선형 관계를 조사할 수 있다.

summary(lm(NO2~sqrt(pop\_dens), no2.sf))  
#   
# Call:  
# lm(formula = NO2 ~ sqrt(pop\_dens), data = no2.sf)  
#   
# Residuals:  
# Min 1Q Median 3Q Max   
# -7.990 -2.052 -0.505 1.610 8.095   
#   
# Coefficients:  
# Estimate Std. Error t value Pr(>|t|)   
# (Intercept) 4.537 0.685 6.62 5.5e-09 \*\*\*  
# sqrt(pop\_dens) 326.154 49.366 6.61 5.8e-09 \*\*\*  
# ---  
# Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
#   
# Residual standard error: 3.13 on 72 degrees of freedom  
# Multiple R-squared: 0.377, Adjusted R-squared: 0.369   
# F-statistic: 43.7 on 1 and 72 DF, p-value: 5.82e-09

해당 선형 관계는 그림 12.11의 산점도에 나타나 있다.

|  |
| --- |
| 그림 12.11: 농촌 지역 대기질 관측소의 2017년 연평균 NO 농도와 인구 밀도의 산점도 |

이 새로운 모델에서의 예측을 수행하기 위해서는 먼저 잔차 베리오그램을 모델링해야 한다(그림 12.12).

no2.sf <- no2.sf[!is.na(no2.sf$pop\_dens),]  
vr <- variogram(NO2~sqrt(pop\_dens), no2.sf)  
vr.m <- fit.variogram(vr, vgm(1, "Exp", 50000, 1))

|  |
| --- |
| 그림 12.12: 인구 밀도 경향성을 제거하고 남은 잔차에 대한 베리오그램 |

그리고 나서 아래의 코드로 크리깅 예측이 이루어진다. 여기서, 중요한 것은 예측 위치에 대해서도 pop\_dens 값이 계산되어 새로이 생성되는 객체인 grd 속에 포함되다는 점이다. 예측 결과는 12.13에 나타나 있다.

kr <- krige(NO2 ~ sqrt(pop\_dens), no2.sf,   
 grd["pop\_dens"], vr.m)  
# [using universal kriging]

|  |
| --- |
| 그림 12.13: 인구 밀도를 경향성 변수로 사용한 크리깅의 NO 농도 예측값 |

(오디네리) 크리깅과 비교할 때 몇 가지 뚜렷한 차이를 볼 수 있다. 인구 밀도를 경향면으로 사용한 크리깅의 결과 지도는 측정 스테이션의 극단적인 값이 아니라 인구 밀도의 극단적인 값을 따르는 경향이 있으며, 값의 범위도 오디네리 크리깅의 범위 보다 넓다. 그러나 “농촌 배경”의 범주에 속하는 측정 스테이션만 사용되어 인구 밀도가 전체적으로 낮은 특성이 있다는 점에서 해석에 신중을 기할 필요가 있다. 그림 12.11의 산점도의 x-축을 보면 측정 스테이션의 인구 밀도 값이 인구 밀도 지도에 나타나 있는 값의 범위에 비해 훨씬 제한적이라는 사실을 알 수 있으며, 따라서 오른쪽 지도는 그림 12.11에 나타난 관계를 강하게 인터폴레이션한 결과이다.

## 12.8 연습문제

1. 그림 12.13과 같은 플롯을 생성하되, 왼쪽에 그림 12.2의 역거리 인터폴레이션 맵을 추가하라.
2. IDW와 크리깅, 그리고 IDW와 잔차 크리깅 간의 관계를 나타내는 산점도를 생성하라.
3. 그리드 셀을 중심으로 하는 블록 평균을 예측하기 위해 *블록 크리깅*을 수행하되, krige() 함수의 block 아규먼트를 통해 블록 크기가 10km(그리드 셀 크기), 50km, 200km인 블록 크리깅을 수행하라. 이 세 가지 블록 크기에 대한 추정치의 결과 맵을 포인트 크리깅으로 얻은 맵과 비교하고, 관련된 모든 크리깅 표준 오차에 대해서도 동일하게 수행하라.
4. 위에서 얻은 잔차 크리깅 결과를 기반으로, 크리깅 오차가 정규 분포를 따른다고 가정할 때 95% 신뢰 구간의 하한선과 상한선 맵을 계산하고, 이를 단일(결합) 범례가 있는 플롯에 표시하라.
5. 크리깅 오차에 대한 정규 분포 가정 하에서, NO의 포인트 값이 15 ppm을 초과할 확률 값을 계산하고 그것을 지도로 표현하라.

# 13. 다변량 및 시공간 지구통계학

12장에 제시된 간단한 인터폴레이션 기법을 바탕으로, 이 장에서는 다변량 지구통계학과 시공간 지구통계학을 다룬다. 다변량 지구통계학은 Bivand, Pebesma, & Gómez-Rubio(2013)에 잘 나타나 있기 때문에 여기서는 간략하게만 다룬다. 시공간 지구통계학은 NO₂ 대기 질 데이터의 사례 연구를 통해 다루고자 한다. 인구 밀도를 공변량으로 사용한 시공간 인터폴레이션 기법이 제시된다.

## 13.1 대기질 데이터셋 준비

여기서 사용할 데이터셋은 유럽 환경청(EEA)에서 얻은 대기 질 데이터셋이다. 유럽 회원국들은 이 기관에 대기 질 측정 결과를 보고한다. 이른바 *검증된* 데이터는 회원국에 의해 품질이 관리되며, 연단위로 보고된다. 이 데이터는 정책 준수 평가와 (대응) 조치의 기초가 된다.

EEA의 대기 질 전자 보고(e-reporting) 웹사이트를 통해 유럽 회원국들이 보고한 데이터에 접근할 수 있다. 우리는 기본 측정 데이터인 시간별(시계열) 데이터를 다운로드하기로 결정했다. 웹 양식을 통해 우리의 데이터 선택 규준이 손쉽게 HTTP GET 요청으로 변환된다. 독일(CountryCode=DE)의 2017년(Year\_from, Year\_to)의 모든 검증된(Source=E1a) (Pollutant=8) 데이터를 선택하는 URL이 생성되었다. 이 URL은 여러개의 CSV 파일과 연결된 URL 정보가 들어있는 텍스트 파일을 반환한다. 개별 CSV 파일에는 개별 측정 측정 스테이션에 대해 전 기간에 대한 시간별 측정값이 들어 있다. 파일을 다운로드하고 dos2unix 명령줄 유틸리티를 활용하여 인코딩 변환하였다.

다음으로, 스테이션 메타데이터가 포함된 단일 파일을 제외하고 나머지 모든 파일을 리스트 형태로 읽어들인다.

files <- list.files("aq", pattern = "\*.csv", full.names = TRUE)  
files <- setdiff(files, "aq/AirBase\_v8\_stations.csv") # metadata file  
r <- lapply(files, function(f) read.csv(f))

그런 다음 시간 변수를 POSIXct 변수로 변환하고, 다음과 같이 시간 순서로 정렬한다.

Sys.setenv(TZ = "UTC") # don't use local time zone  
r <- lapply(r, function(f) {  
 f$t = as.POSIXct(f$DatetimeBegin)   
 f[order(f$t), ]   
 }  
)

이 데이터셋에서 시간별 데이터가 없는 작은 하위 데이터셋을 제거한다.

r <- r[sapply(r, nrow) > 1000]  
names(r) <- sapply(r,  
 function(f) unique(f$AirQualityStationEoICode))  
length(r) == length(unique(names(r)))  
# [1] TRUE

그런 다음 **xts** 패키지의 cbind 함수를 사용하여 모든 파일을 병합하여 시간을 기준으로 매칭 레코드를 결합한다.

library(xts) |> suppressPackageStartupMessages()  
r <- lapply(r, function(f) xts(f$Concentration, f$t))  
aq <- do.call(cbind, r)

이 데이터셋에 추가적인 선택을 실행하였는데, 측정된 시간별 값이 75%가 유효한 스테이션을 선택하는 것이다. 즉, 25% 이상의 결측 시간별 값을 가진 스테이션은 제외한다. mean(is.na(x)) 함수가 벡터 x에서 결측 값의 비율을 계산해 주기 때문에, 이 함수를 열(스테이션)에 적용하면 된다.

sel <- apply(aq, 2, function(x) mean(is.na(x)) < 0.25)  
aqsel <- aq[, sel]

다음으로, 스테이션 메타데이터를 읽어 들이고, 독일(“DE”)의 농촌 배경 스테이션에 해당하는 내용만을 골라낸다.

library(tidyverse) |> suppressPackageStartupMessages()  
read.csv("aq/AirBase\_v8\_stations.csv", sep = "\t") |>  
 as\_tibble() |>   
 filter(country\_iso\_code == "DE",  
 station\_type\_of\_area == "rural",  
 type\_of\_station == "Background") -> a2

포함되어 있던 좌표값을 이용해, (정적) 스테이션 메타데이터를 가진 sf 객체를 생성한다.

library(sf) |> suppressPackageStartupMessages()  
a2.sf <- st\_as\_sf(a2, crs = 'OGC:CRS84',  
 coords = c("station\_longitude\_deg", "station\_latitude\_deg"))

이제 위에서 정리한 대기 질 측정 데이터에서 농촌 배경 유형의 스테이션에 해당 되는 것만을 골라내야 하는데, 스테이션의 코드 정보는 메타데이터를 정리한 a2에 저장되어 있었다.

sel <- colnames(aqsel) %in% a2$station\_european\_code  
aqsel <- aqsel[, sel]  
dim(aqsel)  
# [1] 8760 74

스테이션별로 평균을 계산하고 그것을 스테이션 위치 객체와 조인한다.

tb <- tibble(NO2 = apply(aqsel, 2, mean, na.rm = TRUE),   
 station\_european\_code = colnames(aqsel))  
crs <- st\_crs('EPSG:32632')  
right\_join(a2.sf, tb) |> st\_transform(crs) -> no2.sf   
read\_sf("data/de\_nuts1.gpkg") |> st\_transform(crs) -> de

그림 12.1에는 이렇게 계산된 스테이션별 평균 NO 농도와 국가 경계가 나타나 있다.

## 13.2 다변량 지구통계학

다변량 지구통계학은 여러 변수를 결합하여 모델링, 예측 및 시뮬레이션하는 것을 의미한다. 이것을 수식으로 표현하면 다음과 같다.

이러한 모델을 위해서는 각 변수별로 관측치, 경향면 모델, 베리오그램이 필요할 뿐만 아니라 각 변수쌍별로 잔차의 교차-베리오그램이 필요하다. 교차-베리오그램은 와 간의 공분산을 나타낸다. 이 교차-공분산이 0이 아니라면, 의 정보는 를 예측(또는 시뮬레이션)하는 데 도움이 될 수 있다. 이는 특히 가 보다 더 조밀하게 샘플링된 경우에 그러하다. 이러한 방식의 예측과 시뮬레이션을 코크리깅(cokriging) 및 코시뮬레이션(cosimulation)이라고 부른다. 데모 스크립트를 실행하면 **gstat** 패키지를 사용한 예제를 살펴볼 수 있다.

library(gstat)  
demo(cokriging)  
demo(cosimulation)

좀 더 자세한 사항은 Bivand, Pebesma, & Gómez-Rubio(2013)를 참고하라.

다양한 변수가 동일한 위치에서 관측되는 경우, 예를 들어 다양한 대기 질 변수가 스테이션에서 함께 측정되는 경우, 코크리깅(cokriging)을 사용하는 통계적 이익이 미미한 것으로 드러나는 경우가 있다. 그러나 진정한 다변량 모델링을 원한다면 코크리깅을 수행하는 것이 맞다. 코크리깅을 통해 예측 벡터 를 얻을 뿐만 아니라 예측 오차의 전체 공분산 행렬도 얻을 수 있다(Ver Hoef and Cressie 1993). 이 예측 오차 공분산 행렬을 이용하면, 의 어떠한 선형 조합, 예를 들어 에 대한 표준 오차를 얻을 수 있다.

베리오그램 및 교차-베리오그램을 자동으로 계산하고 적합할 수 있지만, 변수의 수가 많아질수록 다변량 지구통계 모델링은 관리하기 어려워진다. 필요한 베리오그램 및 교차-베리오그램의 수가 로 늘어나기 때문이다.

여러 변수라는 의미가 동일한 변수에 대한 여러 시점의 변수를 의미하는 경우라도 다변량(코크리깅) 예측 접근 방식을 사용할 수 있다. 하지만 이 경우 두 시점 사이의 특정 시점에 대해 인터폴레이션 하는 것은 불가능하다. 이런 경우, 혹은 여러 시간 인스턴스에서 관측된 데이터를 처리해야 하는 경우, 처럼 연속적인 시공간 *결합* 함수를 통해 변동성을 모델링할 수 있다. 다음 절에서 이를 다룰 것이다.

## 13.3 시공간 지구통계학

시공간 지구통계 프로세스는 시공간의 모든 위치에서 변수 값이 존재한다는 전제에 기반한다. 이것을 로 나타낼 수 있고, 여기서 와 는 시공간 상에서 연속적으로 정의되는 인덱스이다. 관측치 와 베리오그램(공변동) 모델 가 주어지면, 표준 가우시안 프로세스 이론을 활용해 임의의 시공간 위치 의 속성값 를 예측할 수 있다.

최근에 시공간 지구통계 데이터의 처리 및 모델링에 관한 현대적인 접근을 다룬 몇 권의 책이 출간되었다. 예를 들어 Wikle, Zammit-Mangion, & Cressie(2019)와 Blangiardo & Cameletti(2015)가 있다. 여기서는 Gräler, Pebesma, & Heuvelink(2016)를 참고하고, 이전 장에서 사용한 데이터셋을 활용한 간단한 예제를 제시할 것이다.

### 13.3.1 시공간 베리오그램 모델

이 장의 서두에서 NO의 시공간 매트릭스 데이터를 aq 객체에 저장했고, 이것으로부터 완전한 레코드를 보유한 농촌 배경 관측소만을 선택해 aqsel 객체로 만들었다. 최종 74개 관측소의 공간 위치를 다음과 같이 선택할 수 있다.

sfc <- st\_geometry(a2.sf)[match(colnames(aqsel),  
 a2.sf$station\_european\_code)] |>  
 st\_transform(crs)

그리고 나서 마침내 시간과 스테이션을 디멘션으로 가지는 stars 벡터 큐브를 구축한다.

library(stars)  
# Loading required package: abind  
st\_as\_stars(NO2 = as.matrix(aqsel)) |>  
 st\_set\_dimensions(names = c("time", "station")) |>  
 st\_set\_dimensions("time", index(aqsel)) |>  
 st\_set\_dimensions("station", sfc) -> no2.st  
no2.st  
# stars object with 2 dimensions and 1 attribute  
# attribute(s):  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
# NO2 -8.1 3.02 5.66 8.39 10.4 197 16134  
# dimension(s):  
# from to offset delta refsys point  
# time 1 8760 2017-01-01 UTC 1 hours POSIXct NA  
# station 1 74 NA NA WGS 84 / UTM z... TRUE  
# values  
# time NULL  
# station POINT (439814 ...,...,POINT (456668 ...

이 데이터를 바탕으로 다음과 같이 시공간 베리오그램을 구성한다.

library(gstat)

v.st <- variogramST(NO2~1, no2.st[,1:(24\*31)], tlags = 0:48,   
 cores = getOption("mc.cores", 2))

결과는 그림 13.1에 나타나 있다.

|  |
| --- |
| 그림 13.1: 2017년 독일의 농촌 배경 스테이션의 시간별 NO 농도에 대한 시공간 샘플 베리오그램으로 오른쪽의 컬러는 시간 지체를 나타내며(노란색이 더 늦은 시간을 나타냄) 거리의 단위는 미터이다. |

이 샘플 베리오그램에 대해 특정한 베리오그램 모델을 적합할 수 있다. 여기에서는 비교적 유연한 모델인 곱-합(product-sum) 모델(Gräler, Pebesma, 및 Heuvelink 2016)을 적용하고자 하는데, 다음과 같이 적합된다.

# product-sum  
prodSumModel <- vgmST("productSum",  
 space = vgm(150, "Exp", 200000, 0),  
 time = vgm(20, "Sph", 6, 0),  
 k = 2)  
#v.st$dist = v.st$dist / 1000  
StAni <- estiStAni(v.st, c(0,200000))  
(fitProdSumModel <- fit.StVariogram(v.st, prodSumModel,  
 fit.method = 7, stAni = StAni, method = "L-BFGS-B",  
 control = list(parscale = c(1,100000,1,1,0.1,1,10)),  
 lower = rep(0.0001, 7)))  
# space component:   
# model psill range  
# 1 Nug 0.0166 0  
# 2 Exp 152.7046 83590  
# time component:   
# model psill range  
# 1 Nug 0.0001 0.00  
# 2 Sph 25.5736 5.77  
# k: 0.00397635996859073

그림 13.2에 결과가 나타나 있으며, 와이어 프레임으로도 플롯할 수 있다(그림 13.3). 이 모델의 적합은 선택된 파라미터에 다소 민감한데, 사용 가능한 관측 네트워크 관측소의 수가 상대적으로 적은 수(74개)이기 때문일 수 있다.

|  |
| --- |
| 그림 13.2: 샘플 베리오그램에 적합된 곱-합 모델 |

|  |
| --- |
| 그림 13.3: 적합된 시공간 베리오그램 모델에 대한 와이어 프레임 플롯 |

시공간 베리오그램을 적합하는 전략과 대체 모델에 대해서는 Gräler, Pebesma, & Heuvelink(2016)을 참고할 수 있다.

이 적합 모델과 주어진 관측치를 바탕으로, 우리는 시공간의 임의의 위치에 대한 크리깅 또는 시뮬레이션을 수행할 수 있다. 예를 들어, 누락된 시계열 값을 추정(또는 시뮬레이션)하는데 사용할 수 있다. 이런 경우는 흔히 발생하며, 이에 대한 대처로서 12.4절에서는 관측치의 최대 25%를 배제한 시계열 평균을 계산한 바 있다. 보다 더 합리적인 옵션은, 해당 결측치를 (시공간적) 이웃의 관측치에 기반한 추정값 혹은 시뮬레이션한 값으로 대체한 후 연간 평균 값을 계산하는 것이다.

보다 일반적인 관점에서 임의의 시공간 위치에서 추정을 수행하는 것이 가능하며, 이러한 과정을 특정 위치에서 시계열 값을 예측하는 것과 공간 슬라이스를 예측하는 것을 통해 설명할 것이다(Gräler, Pebesma, and Heuvelink 2016). 이를 위해 두 개의 공간 지점을 무작위로 선택하고 그 두 지점에 대한 모든 시간 인스턴스를 가진 stars 객체를 생성한다.

set.seed(1331)  
pt <- st\_sample(de, 2)  
t <- st\_get\_dimension\_values(no2.st, 1)  
st\_as\_stars(list(pts = matrix(1, length(t), length(pt)))) |>  
 st\_set\_dimensions(names = c("time", "station")) |>  
 st\_set\_dimensions("time", t) |>  
 st\_set\_dimensions("station", pt) -> new\_pt

그리고 krigeST 함수를 사용하여 이 두 지점에서의 시공간 예측값을 얻는다.

no2.st <- st\_transform(no2.st, crs)  
new\_ts <- krigeST(NO2~1, data = no2.st["NO2"], newdata = new\_pt,  
 nmax = 50, stAni = StAni, modelList = fitProdSumModel,  
 progress = FALSE)

결과는 그림 13.4에 나타나 있다.

|  |
| --- |
| 그림 13.4: 선택된 두 지점에 대한 시공간 예측의 시계열 플롯 |

또한, 2017년 한 해 동안 일정한 시간 간격으로 생성된 일련의 래스터 지도들에 대한 시공간 예측을 생성할 수 있으며, 이는 다음과 같이 수행된다.

st\_bbox(de) |>  
 st\_as\_stars(dx = 10000) |>  
 st\_crop(de) -> grd  
d <- dim(grd)  
t4 <- t[(1:4 - 0.5) \* (3\*24\*30)]  
st\_as\_stars(pts = array(1, c(d[1], d[2], time = length(t4)))) |>  
 st\_set\_dimensions("time", t4) |>  
 st\_set\_dimensions("x", st\_get\_dimension\_values(grd, "x")) |>  
 st\_set\_dimensions("y", st\_get\_dimension\_values(grd, "y")) |>  
 st\_set\_crs(crs) -> grd.st

그리고 예측은 다음과 같이 수행된다.

new\_int <- krigeST(NO2~1, data = no2.st["NO2"], newdata = grd.st,  
 nmax = 200, stAni = StAni, modelList = fitProdSumModel,  
 progress = FALSE)  
names(new\_int)[2] = "NO2"

결과는 그림 13.5에 나타나 있다.

|  |
| --- |
| 그림 13.5: 네 개의 시점에 대해 수행된 시공간 예측의 결과 |

여기서는 nmax 아규먼트의 값을 크게 설정할 필요가 있었는데, 이는 이산적인 이웃의 선택에서 시간과 공간을 *모두* 고려해야 하고 이로 인해 발생할 수 있는 시각적 왜곡(날카로운 경계)을 줄이기 위한 조치이다.

### 13.3.2 불규칙 시공간 데이터

관측 지점이 계속해서 변화하거나 고정된 관측 지점이라 하더라도 시간 프레임이 일관성이 없는 경우, stars 객체(벡터 데이터 큐브)는 이러한 데이터를 잘 다루지 못한다. 이러한 불규칙 시공간 관측치는 **sftime** 패키지(Teickner, Pebesma, and Graeler 2022)에서 제공하는 sftime 객체로 표현할 수 있다. sftime 객체는 기본적으로 sf 객체인데, 지정된 시간 컬럼을 가지고 있다. 이 사용 예시는 **gstat** 패키지에서 제공되는 demo(sftime)에서 찾을 수 있다.

## 13.4 연습문제

1. 섹션 13.1에서 “관측 스테이션이 최소 75%의 완전한 데이터를 가져야 한다”는 기준을 적용할 때, 스테이션의 몇 %가 제거되는가?
2. no2.st의 시간당 시계열 데이터에서 aggregate 함수를 사용하여 일별 평균 농도를 계산하고, 이에 대한 시공간 베리오그램을 계산하시오. 이를 시간별 베리오그램과 비교하시오.
3. 그림 13.5에 표시된 날짜의 일별 평균 값에 대한 시공간 인터폴레이션을 수행하고 그 결과를 비교하시오.
4. 13.2절에서 소개된 데모 스크립트의 예를 따라, 그림 13.5에 표시된 네 날의 일별 평균 스테이션 데이터를 사용하여 코크리깅을 수행하시오.
5. 시공간 크리깅에 대한 위의 접근법이 가지는 차별점은 무엇인가?

# 14. 근접성과 에어리어 데이터

관측개체가 에어리어 객체인 경우는 매우 흔하다. 연구대상 전체에서 관측이 동시적으로 이루어지고 그러한 관측값이 비중첩 구역단위로 합산되는 경우가 여기에 해당한다. 이러한 구역단위는 행정구역인 경우가 많다. 물론 통근 플로와 같이 기저 공간 프로세스를 직접적으로 반영한 경우도 있지만, 보통은 프로세스와 관계없이 임의적으로 주어진다. 이러한 구역단위와 기저 공간 프로세스 간의 불일치는 동일한 공간적 프로세스가 인접한 구역단위 사이에서 부분적으로 공유된다는 것을 의미하며, 이러한 불일치가 해당 데이터셋 내의 단 하나의 변수에서라도 발생한다면, 전체적으로 공간적 자기상관의 문제가 야기된다고 할 수 있다. 여기서 근접성(proximity)이란 데이터 생성 프로세스라는 의미에서의 *가까움*(closeness)을 의미한다. 포인트 스포트를 사용하는 횡단 지구통계학 분석에서는 포인트 간 거리가 전형적인 데이터 생성 프로세스에 적합하다. 그런데, 에어리어 데이터의 경우는 거리보다는 공유 경계가 더 적합할 수 있다. 공유 경계는 우리가 확실히 측정할 수 있는 것이지만, 동일한 확실성 하에서 에어리어간 거리를 측정할 수 있는 방법은 없기 때문이다.

데이터의 스포트란 개별 관찰단위(관측개체)의 물리적 크기(길이, 면적, 부피)를 의미한다(5장 참조). 에어리어 데이터는 기본적으로 폴리곤 스포트를 가지지만 포인트 서포트를 가지는 것으로 재현할 수도 있다. 개별 폴리곤의 센트로이드 혹은 멀티폴리곤의 경우는 최대 폴리곤의 센트로이드를 대표 지점으로 삼을 수 있다. 반대로, 포인트 스포트를 가지는 데이터를 에어리어 데이터로 취급할 수도 있다. 이 경우 포인트를 비중첩 테셀레이션으로 전환해야 하는데, 들로네 삼각망을 통해 보로노이 다이어그램(디리클레 테셀레이션 혹은 티센 폴리곤)을 생성한다. 다른 메트릭이 선택될 수 있으며, 평면이 아닌 네트워크 상의 거리를 측정할 수도 있다. 국지적 공간 분석에서는 가중 보로노이 다이어그램이 사용되기도 한다(Boots and Okabe 2007; Okabe et al. 2008; She et al. 2015 참조).

데이터의 원 스포트가 포인트이지만, 기저의 공간 프로세스가 관측개체간의 거리가 아니라 오로지 인접성에 의거해 발생하는 경우라면, 데이터는 총빈도 혹은 총계(투표소의 개수, 총매출액)이거나 관측개체의 직접적 속성(예: 투표소의 개방 시간)일 수 있다. 이러한 모든 경우에서 기저의 공간 프로세스를 잘못 재현할 위험성이 상존함은 분명한 사실이다. 특히, 연구지역 전체를 하나의 테셀레이션 체계로 재현하는 경우, 관측값이 기저의 공간적 프로세스를 모두 포착해 낸 것으로 상정되기 때문에 더욱 그러하다. 이러한 측면에서 에어리어 데이터는 지구통계학적 데이터와는 다르다. 후자는 전체 지역에 대해 특정한 방식의 샘플링이 적용되었다는 것을 전제하지만 전자는 그렇지 않다. 또한 생태 및 환경 분석에서 사용되는 에어리어 샘플링과도 다른데, 그러한 샘플링은 지역 전체가 아니라 일부 영역에 대해 선택적으로 이루어진다.

15~17장에 걸쳐 에어리어 데이터를 탐색 및 분석하기 위한 기법들이 다루어진다. 이를 위해서는 관측개체 간의 근접성을 나타내는 특정한 방식이 요구된다. 이 장에서는 근접성을 연접성(contiguity)에 의거해 규정하는 방식에 집중한다. 여기서 연접성은 이웃으로 정의된 관측개체들을 서로 연결한 그래프를 통해 규정된다. 이 그래프는 보통 방향성과 가중치가 없지만, 특정 설정에서는 방향과 가중치가 부여될 수도 있으며, 이 경우 대칭성에 대한 추가적인 이슈가 야기될 수 있다. 원칙적으로, 근접성은 공간 상에서 대칭적으로 작동하는 것으로 간주한다. 즉, 가 에 미치는 영향과 가 에 미치는 영향은 동일한 것으로 간주한다. 일반적으로 에지 효과(edge effect)는 고려하지 않는다.

## 14.1 근접성의 재현: spdep 패키지의 경우

공간적 자기상관을 그래프 상의 이웃들의 관계를 통해 다루는 접근의 경우, 해당 그래프는 주어진 것이고, 그 그래프를 연구자가 선택한 것으로 간주한다. 이것은 지구통계학적 접근과는 다른 것인데, 연구자는 경험적 베리오그램의 거리를 어떻게 나눌 것인지, 어떤 함수를 적용할 것인지, 베리오그램 적합을 어떻게 수행할 것인지를 모두 선택한다. 두 접근법 모두 사전 선택을 포함하지만, 기저의 상관성을 재현하는 방식에서는 서로 다른 것이다(Wall 2004). 그래프에 기반한 이웃 규정 방식을 보다 넓은 맥락에서 설명하는 시도도 존재한다(Bavaud, 1998).

이웃 관계를 나타내는 객체를 생성할 때 이웃이 없는 구역단위의 존재는 문제를 야기한다(Bivand and Portnov 2004). 섬이나 강으로 분리된 구역단위가 이러한 비이웃 구역단위일 수 있는데, 구역단위에 에어리어 스포터가 적용되어 있고 공유 경계와 같은 위상 관계가 사용되는 경우이다. 예를 들어, mgcv::gam과 같은 모델 적합 함수에서 mrf(마르코프 랜덤 필드) 항을 사용할 때, 방향이 없는 필요하지만, 그래프가 분리된 하위 그래프들로 구성되는 경우 에러가 발생한다.

이러한 무이웃 문제는 포인트 간의 거리로 이웃을 규정하는 경우에도 발생할 수 있는데, 거리 임계값이 최근린 이웃 거리보다 작은 경우가 여기에 해당된다. 공유 경계에 기반한 연접성 규정은 비투영 경위도 좌표를 사용하더라도 영향을 받지 않지만, 모든 포인트 기반 접근법은 어쨌든 거리를 사용하며, 적절한 방식으로 거리를 계산할 필요가 있다.

**spdep** 패키지는 이웃을 규정하는 nb 클래스를 제공한다. nb 클래스는 관측개체의 수를 길이로 갖는 리스트로 정수 벡터를 구성 요소로 갖는다. 이웃이 없는 경우는 0L이 단일 요소로 포함된 정수 벡터로 인코딩된다. 이웃이 있는 경우는 1L:n 범위 내의 값이 포함된 정수 벡터로 인코딩되는데, 그 값들은 이웃으로 규정된 관측개체의 인덱스값이다. 이것은 소위 행-기반 희소 재현(row-oriented sparse representation) 방식이다. **spdep** 패키지는 nb 객체를 생성하는 다양한 방법을 제공하며, 이 표현과 생성 함수는 다른 패키지에서도 널리 사용된다.

**spdep** 패키지는 nb 클래스(무방향 혹은 유방향 그래프)를 통해 listw 객체를 구성한다. listw 객체는 세 가지 구성 요소를 갖는 리스트로, nb 객체, 가중치 리스트, 그리고 가중치 계산 방식을 나타내는 단일 요소 문자 벡터가 포함된다. 사회과학에서 가장 흔히 사용되는 방법은 행 표준화를 통해 가중치를 계산하는 것인데, 개별 관측개체의 한 이웃의 가중치는 해당 관측개체의 이웃의 수(즉 카드널리티)의 역수(즉, 1/card(nb)[i])로 변환된다.

이 장에서는 2015년 폴란드 대통령 선거 데이터를 사용하는데, 지역 전체가 2,495개의 지방자치단체와 바르샤바 구역으로 구성되어 있다(그림 14.1 참조). 이 지도는 **tmap** 패키지(8.5절)를 활용해 그려진 것으로 지방자치단체 유형이 표시되어 있다. 구역단위는 sf 객체이며, 투표소 단위의 결과를 구역단위로 집계한 데이터이다.

library(sf)  
# Linking to GEOS 3.11.1, GDAL 3.6.4, PROJ 9.1.1; sf\_use\_s2() is TRUE

data(pol\_pres15, package = "spDataLarge")  
pol\_pres15 |>  
 subset(select = c(TERYT, name, types)) |>  
 head()  
# Simple feature collection with 6 features and 3 fields  
# Geometry type: MULTIPOLYGON  
# Dimension: XY  
# Bounding box: xmin: 235000 ymin: 367000 xmax: 281000 ymax: 413000  
# Projected CRS: ETRS89 / Poland CS92  
# TERYT name types  
# 1 020101 BOLESŁAWIEC Urban  
# 2 020102 BOLESŁAWIEC Rural  
# 3 020103 GROMADKA Rural  
# 4 020104 NOWOGRODZIEC Urban/rural  
# 5 020105 OSIECZNICA Rural  
# 6 020106 WARTA BOLESŁAWIECKA Rural  
# geometry  
# 1 MULTIPOLYGON (((261089 3855...  
# 2 MULTIPOLYGON (((254150 3837...  
# 3 MULTIPOLYGON (((275346 3846...  
# 4 MULTIPOLYGON (((251770 3770...  
# 5 MULTIPOLYGON (((263424 4060...  
# 6 MULTIPOLYGON (((267031 3870...

library(tmap, warn.conflicts = FALSE)  
# Breaking News: tmap 3.x is retiring. Please test v4, e.g. with  
# remotes::install\_github('r-tmap/tmap')  
tm\_shape(pol\_pres15) + tm\_fill("types")

|  |
| --- |
| 그림 14.1: 폴란드 구역단위 유형(2015년) |

sf 객체의 위상 구조의 유효성을 검토한다.

if (!all(st\_is\_valid(pol\_pres15)))  
 pol\_pres15 <- st\_make\_valid(pol\_pres15)

2002년 초부터 2019년 4월까지, spdep는 이웃 및 공간 가중치 객체를 생성하고 처리하는 함수, 공간적 자기상관을 검정하는 함수, 모델 적합과 관련된 함수 등을 포함하고 있었다. 모델 적합과 관련된 함수는 spatialreg로 분리되었으며, 이후 장에서 다룰 예정이다. 현재 spdep(Bivand 2022)는 sf 클래스와 sp 클래스 객체 모두를 지원한다.

library(spdep) |> suppressPackageStartupMessages()

## 14.2 이웃의 규정: 연접성 기반

**spdep** 패키지의 poly2nb 함수는 pl 아규먼트를 통해 투입된 객체에서 폴리곤 경계를 구성하고 있는 경계 포인트들을 이용한다. 투입 객체는 일반적으로 "POLYGON" 또는 "MULTIPOLYGON" 지오메트리를 가진 "sf" 또는 "sfc" 객체이다. 각 관측개체에 대해, 최소한 하나의 포인트(디폴트 퀸 방식, queen=TRUE) 또는 최소 두 개의 포인트(루크 방식, queen=FALSE)가 snap 거리 내에 또 다른 포인트(역자주: 연접 폴리곤의 경계 포인트)를 가지는지를 확인한다. 거리 계산은 투영법에 관계없이 원 길이 단위에 기반한 평면 거리로 이루어진다. 필요한 수의 충분히 가까운 점을 찾으면, 검색이 중단된다.

args(poly2nb)

# function (pl, row.names = NULL, snap = sqrt(.Machine$double.eps),  
# queen = TRUE, useC = TRUE, foundInBox = NULL)

**spdep** 패키지 1.1-7부터, poly2nb 함수는 후보 이웃을 찾고 foundInBox를 내부적으로 채우기 위해 **sf** 패키지의 GEOS 인터페이스를 사용한다. 이 경우, **sf** 패키지를 통한 GEOS의 공간 인덱싱(STRtree 쿼리 사용)이 기본값으로 설정된다.

pol\_pres15 |> poly2nb(queen = TRUE) -> nb\_q

print 메서드는 이웃 객체의 요약 구조를 표시한다.

nb\_q  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 14242   
# Percentage nonzero weights: 0.229   
# Average number of links: 5.71

**sf** 패키지 버전 1.0-0부터는 구체 지오메트리를 위해 기본적으로 **s2** 패키지(Dunnington, Pebesma, and Rubak 2023)가 사용된다. 이는 poly2nb 함수에서 사용되는 st\_intersects 함수가 계산을 s2::s2\_intersects\_matrix로 전달하기 때문이다(4장 참조). **spdep** 패키지 버전 1.1-9부터는 sf\_use\_s2()가 TRUE일 경우 구체 인터섹션을 사용하여 후보 이웃을 찾는다. GEOS와 마찬가지로, s2 라이브러리는 빠른 공간 인덱싱을 사용한다.

old\_use\_s2 <- sf\_use\_s2()

sf\_use\_s2(TRUE)

(pol\_pres15 |> st\_transform("OGC:CRS84") -> pol\_pres15\_ll) |>   
 poly2nb(queen = TRUE) -> nb\_q\_s2

예시의 경우, 구면 교차와 평면 교차가 동일한 인접 이웃을 생성한다. 두 경우 모두 투입 지오메트리의 유효성이 담보되어야 한다.

all.equal(nb\_q, nb\_q\_s2, check.attributes=FALSE)  
# [1] TRUE

nb 객체는 대칭적인 이웃 관계인 에서 , 에서를 모두 기록하는데, 이는 nb 객체가 비대칭적인 관계도 허용하기 때문이다. 그러나 객체 생성에는 이러한 중복이 큰 의미는 없다.

대부분의 **spdep** 패키지 함수는 이웃 객체를 생성할 때 row.names 아규먼트를 사용하며, 이 값은 region.id 속성으로 저장된다. 만약 row.names 아규먼트가 지정되지 않으면, 첫 번째 아규먼트의 row.names()에서 가져온다. region.id 속성은 nb 객체가 원 데이터와 동일한 순서로 정리되어 있는지를 확인하는 데 사용된다. nb 객체의 일부만 추출할 경우 인덱스는 1:length(subsetted\_nb) 범위 내의 값으로 재설정되지만, region.id 속성 값을 통해 원본 객체와 정확한 연결 정보를 확인할 수 있다. 이는 17.4절에서 간략하게 논의될 공간적 회귀 모델의 샘플 외 예측에서 사용된다.

n.comp.nb 함수를 사용하여 이 비방향 그래프의 연결성을 확인할 수도 있다. 일부 모델 추정 기법은 비연결 그래프를 지원하지 않지만, 비연결 그래프가 야기할 가능한 문제를 인지하고 있는 것은 중요하다(Freni-Sterrantino, Ventrucci, and Rue 2018).

(nb\_q |> n.comp.nb())$nc  
# [1] 1

이 접근법은 이웃 객체를 그래프처럼 취급하고 해당 그래프에 대해 그래프 분석을 수행하는 것과 동일하다(Csardi and Nepusz 2006; Nepusz 2022). 먼저 이웃 객체를 이진 희소 행렬로 변환한 후 그래프 분석을 수행한다(Bates, Maechler, and Jagan 2022).

library(Matrix, warn.conflicts = FALSE)  
library(spatialreg, warn.conflicts = FALSE)  
nb\_q |>   
 nb2listw(style = "B") |>   
 as("CsparseMatrix") -> smat  
library(igraph, warn.conflicts = FALSE)  
(smat |> graph.adjacency() -> g1) |>   
 count\_components()  
# [1] 1

다른 소프트웨어와의 호환성을 위해 이웃 객체를 GAL 형식으로 내보내고 가져올 수 있다. 이를 위해 write.nb.gal함수와 read.gal 함수가 사용된다.

tf <- tempfile(fileext = ".gal")  
write.nb.gal(nb\_q, tf)

## 14.3 이웃의 규정: 그래프 기반

구역단위가 적합한 재현이지만 평면 상의 포인트로 관찰된 경우, 연접성은 그래프 기반 이웃을 사용하여 근사할 수 있다. 이 경우, 평면은 폴리곤 테셀레이션(역자주: 티센폴리곤과 같은 것)으로 분할되는데, 각 폴리곤 내의 모든 지점은 해당 포인트를 가장 가까운 포인트로 갖는다. 가장 간단한 형태는 삼각망 형성(triangulation)을 사용하는 것인데, 여기서는 **deldir** 패키지의 deldir 함수를 사용한다. 이 함수는 와 식별자를 반환하므로, 세로(긴) 형식으로 listw 객체를 구성하는 것이 용이하다. 이는 S-Plus SpatialStats 모듈에서 사용되었던 방식이고, nb 객체(가로 형식)를 구성하기 위해 내부적으로 sn2listw 함수에서 사용되는 방식이기도 하다. GEOS와 같은 다른 대안들은 이웃의 식별을 위한 충분한 정보를 반환하지 못한다.

이러한 함수들이 리턴한 결과는 graph2nb 함수를 통해 nb 객체로 전환된다. 이 때 sym 아규먼트를 사용하여 이웃 관계의 대칭성을 규정할 수 있다. 그래프 기반 방식을 적용하기 위해 폴리곤의 센트로이드(다중폴리곤의 경우 가장 큰 폴리곤의 센트로이드)를 포인트 재현으로 활용한다. 물론 인구 가중 센트로이드를 구할 수 있다면 더 좋다.

pol\_pres15 |>   
 st\_geometry() |>   
 st\_centroid(of\_largest\_polygon = TRUE) -> coords   
(coords |> tri2nb() -> nb\_tri)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 14930   
# Percentage nonzero weights: 0.24   
# Average number of links: 5.98

평균 이웃수의 측면에서 보면 퀸 방식의 경계 연접성과 유사한 결과가 나왔다. 그러나 nbdists() 함수를 사용하여 엣지 길이의 분포를 살펴보면, 상위 4분위 수는 약 15km 정도이지만 최대값은 거의 300km에 달한다는 것을 알 수 있다. 이는 전체 지역을 포괄하는 컨벡스 헐의 한쪽 면의 길이에 버금가는 것이다. 최소 거리 역시 중요하다. 왜냐하면 많은 도시 구역의 센트로이드가 주변 농촌 구역의 센트로이드와 매우 근접해 있기 때문이다.

nb\_tri |>   
 nbdists(coords) |>   
 unlist() |>   
 summary()  
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 247 9847 12151 13485 14994 296974

삼각망에 의거한 이웃 규정도 연결 그래프(역자주: 그래프 상의 모든 지점이 직간접적으로 연결되어 있는 그래프)를 생성한다.

(nb\_tri |> n.comp.nb())$nc  
# [1] 1

그래프 기반 접근법에는 soi.graph, relativeneigh, gabrielneigh와 같은 것들이 있다. 여기서는 soi.graph만 살펴본다.

soi.graph 함수에서 soi는 영향권(sphere of influence)의 약자이다. 이 함수는 삼각망 이웃에서 비정상적으로 긴 엣지로 표현된 이웃 관계는 제거함으로써 실질적인 의미를 가진 이웃 관계만 남긴다. 이러한 비정상적으로 긴 엣지는 컨벡트 헐의 가장 자리에서 흔히 나타나는 것이다(Avis and Horton, 1985).

(nb\_tri |>   
 soi.graph(coords) |>   
 graph2nb() -> nb\_soi)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 12792   
# Percentage nonzero weights: 0.205   
# Average number of links: 5.13

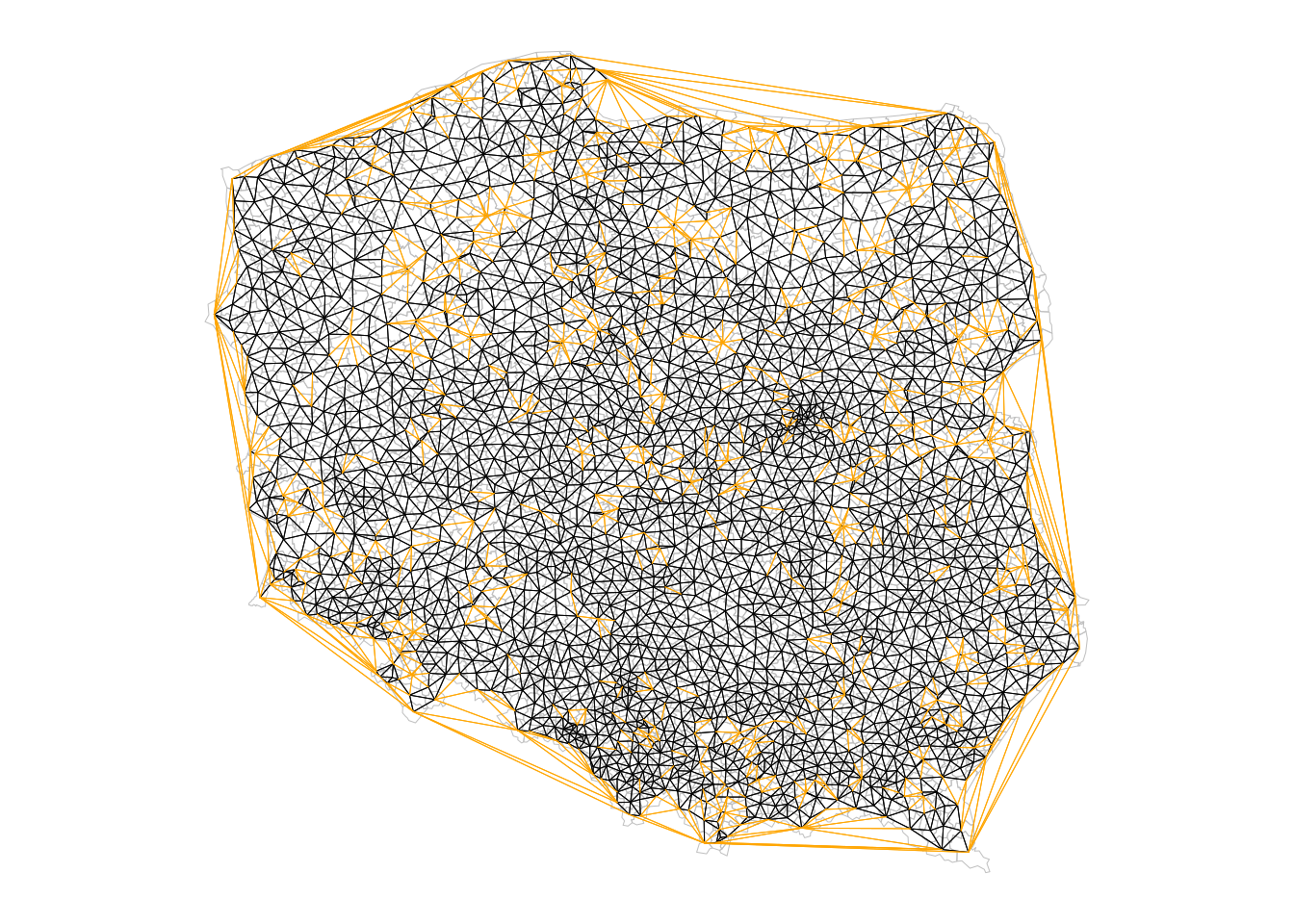
그러나 삼각망에 의거한 이웃 관계의 일부를 해체하면, 연결 그래프로서의 전체적인 성격은 사라지게 된다.

(nb\_soi |> n.comp.nb() -> n\_comp)$nc  
# [1] 16

이 알고리즘은 비정상적으로 긴 엣지를 제거하도록 설계되었지만, 농촌 구역들이 하나의 도시 구역을 완전히 둘러싸고 있는 경우 매우 가까운 도시-농촌 쌍의 엣지도 잘못 삭제될 수 있다. 이러한 결과로 15개의 도시-농촌 쌍이 메인 그래프로부터 분리되는 결과가 초래된 것이다.

table(n\_comp$comp.id)  
#   
# 1 2 3 4 5 6 7 8 9 10 11 12 13   
# 2465 2 2 2 2 2 2 2 2 2 2 2 2   
# 14 15 16   
# 2 2 2

컨벡스 헐에서 가장 긴 엣지들이 제거되었지만, 연결되지 않은 이웃 쌍들이 나타나면서 “구멍”이 생겼다. nb\_tri와 nb\_soi의 차이는 그림 14.2에서 주황색으로 표시되었다.



삼각망 이웃(오렌지색과 검은색)과 영향권 이웃(검은색)의 비교. 군데군데 구멍이 형성되어 있는데, 모두 도시 구역이 농촌 구역으로 완전히 둘러싸여 있는 경우(그림 14.1 참조)에 해당한다.

## 14.4 이웃의 규정: 거리 기반

거리 기반 이웃은 dnearneigh를 사용하여 생성할 수 있다. bounds 아규먼트를 통해 거리 구간을 설정할 수 있는데, d1과 d2는 각각 거리의 하한값과 상한값이다. 경위도 좌표계가 사용되고, 좌표 객체 x가 주어지고, longlat=TRUE로 설정된 경우, WGS84 기준 타원체를 가정하여 킬로미터 단위의 대권 거리를 계산한다. use\_s2=TRUE(기본값)로 설정되면 구체를 상정한 거리 계산을 한다(4장 참조). dwithin이 FALSE이고 **s2** 버전이 1.0.7보다 크면 s2\_closest\_edges가 사용될 수 있으며, dwithin이 TRUE이고 use\_s2=TRUE이면 s2\_dwithin\_matrix가 사용된다. 이 두 방법 모두 빠른 구형 공간 인덱싱을 사용하긴 하지만, s2\_closest\_edges의 경우는 최소 및 최대 경계를 지정하기 때문에 dnearneigh의 R 코드에서 한 번의 실행만으로 충분하다.

**dbscan** 패키지(Hahsler and Piekenbrock 2022)에 새로운 아규먼트를 추가함으로써 2차원 또는 3차원에서 평면 공간 인덱싱을 사용하여 이웃을 찾는 기능이 보강되었고 대칭성 을 확인하는 절차가 필요없게 되었다. 또한, 구면 기하학적 거리 측정을 사용하는 세 가지 아규먼트도 추가되었다.

-최근린 이웃을 위한 knearneigh 함수는 knn 객체를 반환하며, knn2nb를 사용해 nb 객체로 전환된다. 이 함수는 구면 거리도 사용할 수 있는데, 평면 거리와는 다른 최근린 이웃이 산출될 수 있기 때문이다. 는 작은 숫자여야 한다. 투영 좌표의 경우 dbscan 패키지를 사용하여 최근접 이웃을 더 효율적으로 계산한다. 이렇게 생성된 nb 객체는 대개 대칭적이지 않으므로, knn2nb는 대칭성을 강제할 수 있는 sym 아규먼트를 제공한다. 대칭성을 강제하면 모든 단위가 최소 개의 이웃을 가지게 되지만, 모든 단위가 정확히 개의 이웃을 갖는 것은 아니다. sf\_use\_s2()가 TRUE인 경우, 입력 객체가 "sf" 또는 "sfc" 클래스일 때 knearneigh는 빠른 구형 공간 인덱싱을 사용한다.

nbdists 함수는 투영 좌표의 경우 좌표 단위로 이웃 관계 엣지의 길이를 반환하고, 그렇지 않으면 킬로미터 단위로 반환한다. 거리 밴드의 상한을 설정하려면 먼저 첫 번째 최근린 이웃 거리의 최대값을 찾아야 하며, 이때 반환된 객체의 리스트 구조를 제거하기 위해 unlist를 사용할 수 있다. sf\_use\_s2()가 TRUE인 경우, nbdists는 입력 객체가 "sf" 또는 "sfc" 클래스일 때 빠른 구형 거리 계산을 사용한다.

coords |>   
 knearneigh(k = 1) |>   
 knn2nb() |>   
 nbdists(coords) |>   
 unlist() |>   
 summary()  
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 247 6663 8538 8275 10124 17979

여기서 첫 번째 최근린 이웃 거리의 최대값은 약 18km로, 이를 상한선으로 사용하면 모든 단위가 최소한 하나의 이웃을 가지게 된다.

coords |> dnearneigh(0, 18000) -> nb\_d18

사례에서 보는 것처럼 관측개체의 수가 그리 많지 않은 경우에는 공간 인덱싱을 사용해도 실행 시간에서 큰 이점을 얻지 못한다.

coords |> dnearneigh(0, 18000, use\_kd\_tree = FALSE) -> nb\_d18a

그리고 산출 객체도 동일하다.

all.equal(nb\_d18, nb\_d18a, check.attributes = FALSE)  
# [1] TRUE

nb\_d18  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 20358   
# Percentage nonzero weights: 0.327   
# Average number of links: 8.16

그런데 이웃이 없는 관측값은 없지만(이들은 nb 객체의 print 메소드에서 보고됨), 그래프는 완전 연결 상태가 아니다. 왜냐하면 한 쌍의 관측개체가 서로의 유일한 이웃이기 때문이다.

(nb\_d18 |> n.comp.nb() -> n\_comp)$nc  
# [1] 2

table(n\_comp$comp.id)  
#   
# 1 2   
# 2493 2

임계값에 300m를 추가하면 비이웃 관측단위가 없는 이웃 객체가 생성되고, 모든 관측단위는 그래프를 통해 다른 모든 관측단위에 도달할 수 있게 된다.

(coords |> dnearneigh(0, 18300) -> nb\_d183)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 21086   
# Percentage nonzero weights: 0.339   
# Average number of links: 8.45

(nb\_d183 |> n.comp.nb())$nc  
# [1] 1

거리 기반 이웃의 한 특징은 면적이 작은 단위들이 밀집된 지역일수록 더 많은 이웃을 갖는다는 것이다(예를 들어, 바르샤바 구역은 평균적으로 훨씬 작지만, 이 거리 기준으로 약 30개의 이웃을 가진다). 많은 이웃을 갖는 것은 이웃 관계를 더 많은 이웃을 통해 부드럽게 만든다.

나중에 사용할 수 있도록, 16km의 임계값을 사용하여 비이웃 단위가 포함된 이웃 객체도 생성한다.

(coords |> dnearneigh(0, 16000) -> nb\_d16)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 15850   
# Percentage nonzero weights: 0.255   
# Average number of links: 6.35   
# 7 regions with no links:  
# 569 1371 1522 2374 2385 2473 2474

-최근린 이웃을 사용하여 이웃의 수를 직접적으로 제어할 수 있으며, 이때 비대칭 이웃을 허용할 수도 있다.

((coords |> knearneigh(k = 6) -> knn\_k6) |> knn2nb() -> nb\_k6)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 14970   
# Percentage nonzero weights: 0.24   
# Average number of links: 6   
# Non-symmetric neighbours list

또는 대칭성을 강제할 수도 있다.

(knn\_k6 |> knn2nb(sym = TRUE) -> nb\_k6s)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 16810   
# Percentage nonzero weights: 0.27   
# Average number of links: 6.74

여기서 의 크기는 완전 연결성을 보장하기에 충분하지만, 그래프가 반드시 평면성을 가지는 것은 아니다. 엣지가 노드가 아닌 다른 위치에서 교차하기 때문이다. 이는 연접성 기반 이웃이나 그래프 기반 이웃에서는 발생하지 않는 상황이다.

(nb\_k6s |> n.comp.nb())$nc  
# [1] 1

구체 상의 포인트들인 경우(4장에서 참조), st\_centroid의 출력은 달라지므로, 포인트들을 역 투영하는 대신 역 투영된 폴리곤 지오메트리에서 경위도 좌표로 포인트들을 추출한다.

old\_use\_s2 <- sf\_use\_s2()

sf\_use\_s2(TRUE)

pol\_pres15\_ll |>   
 st\_geometry() |>   
 st\_centroid(of\_largest\_polygon = TRUE) -> coords\_ll

구면 좌표의 경우, 거리 경계는 킬로미터 단위로 표시된다.

(coords\_ll |> dnearneigh(0, 18.3, use\_s2 = TRUE,   
 dwithin = TRUE) -> nb\_d183\_ll)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 21140   
# Percentage nonzero weights: 0.34   
# Average number of links: 8.47

이 이웃들은 예상대로 구면 18.3 km 이웃들과 다르다.

isTRUE(all.equal(nb\_d183, nb\_d183\_ll, check.attributes = FALSE))  
# [1] FALSE

**s2**가 더 빠른 거리 이웃 인덱싱을 제공할 수 있다면, 디폴트로 s2\_closest\_edges가 경위도 좌표에 대해 사용된다.

(coords\_ll |> dnearneigh(0, 18.3) -> nb\_d183\_llce)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 21140   
# Percentage nonzero weights: 0.34   
# Average number of links: 8.47

이 경우에는 두 **s2** 기반 이웃 객체가 동일하다.

isTRUE(all.equal(nb\_d183\_llce, nb\_d183\_ll,  
 check.attributes = FALSE))  
# [1] TRUE

**s2**에서 빠른 구형 공간 인덱싱을 사용하여 개의 최근린 이웃을 찾는다.

(coords\_ll |> knearneigh(k = 6) |> knn2nb() -> nb\_k6\_ll)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 14970   
# Percentage nonzero weights: 0.24   
# Average number of links: 6   
# Non-symmetric neighbours list

이 이웃들은 예상대로 평면 k=6 최근접 이웃과 다르지만, 전통적인 브루트포스(brute-force) 타원체 거리와도 약간 다를 것이다.

isTRUE(all.equal(nb\_k6, nb\_k6\_ll, check.attributes = FALSE))  
# [1] FALSE

nbdists 함수 역시 구면 상의 거리를 계산하기 위해 **s2**를 사용하는데, "sf" 또는 "sfc" 투입 객체가 경위도 좌표값을 가지는 경우에 그러하다(반환된 거리는 킬로미터 단위로 표시).

nb\_q |> nbdists(coords\_ll) |> unlist() |> summary()  
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 0.2 9.8 12.2 12.6 15.1 33.0

평면 좌표가 사용될 때와 구형 혹은 타원체 지오메트리가 사용할 때, 동일한 가중치 객체인 경우에도 계산되는 거리 값이 약간 다르다(평면 지오메트리의 경우 포인트의 단위로 거리가 반환되고, 타원체 및 구형 지오메트리의 경우 킬로미터 단위로 거리가 반환된다).

nb\_q |> nbdists(coords) |> unlist() |> summary()  
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 247 9822 12173 12651 15117 33102

sf\_use\_s2(old\_use\_s2)

## 14.5 가중치 부여

이웃 객체를 기반으로 가중치 객체를 지정한다. 여기에 몇 가지 선택이 이루어져야 한다. nb2listw 함수는 nb 객체에 기반하여 listw 가중치 객체를 생성한다. 가중치 객체는 가중치 벡터 리스트와 스타일 선택값으로 구성된다. 비이웃 관측개체의 처리가 중요한 사안인데 zero.policy 아규먼트가 이를 담당한다. 디폴트는 FALSE로 설정되어 있으며, 이는 비이웃 관측개체의 존재가 오류를 발생시킨다는 것을 의미한다. 관측개체가 이웃을 가지지 않으면 공간지체값(spatially lagged values)을 사용할 수 없기 때문이다. 보통 비관측개체의 경우 공간지체값으로 제로가 부여되는데, 이는 제로 값의 가중치 벡터와 데이터 벡터의 교차곱과 같기 때문에 zero.policy라는 이름이 붙여졌다.

args(nb2listw)

# function (neighbours, glist = NULL, style = "W", zero.policy =  
# NULL)

우리는 스타일 선택을 변경했을 때의 결과를 보여주기 위해 아래에서 도우미 함수 spweights.constants를 사용할 것이다. 이 함수는 listw 객체에 대한 상수 값들을 반환한다. 여기서 은 관측개체의 수이고, n1부터 n3은 nn은 이며, , , 는 상수이다. 는 가중치의 합이다. 상수에 대한 자세한 논의는 Bivand와 Wong (2018)를 참고하면 된다.

args(spweights.constants)

# function (listw, zero.policy = NULL, adjust.n = TRUE)

"B" 바이너리 스타일은 이웃 관계에 대해 단위 값(1)을 부여하며, 이웃의 규정을 위한 경계가 존재하는 가장자리 구역단위에 비해 더 많은 이웃을 가질 수 있는 내부 구역에 더 높은 가중치를 부여한다.

(nb\_q |>   
 nb2listw(style = "B") -> lw\_q\_B) |>   
 spweights.constants() |>   
 data.frame() |>   
 subset(select = c(n, S0, S1, S2))  
# n S0 S1 S2  
# 1 2495 14242 28484 357280

"W" 행표준화 스타일은 연구 지역의 가장자리에 위치하여 반드시 더 적은 수의 이웃을 가질 수 밖에 없는 구역단위에 더 높은 가중치를 부여한다. 행표준화 스타일은 먼저 각 이웃 관계에 대해 단위 값을 가중치로 부여한 후, 이 가중치를 각 구역단위의 가중치 합으로 나눈다. 비이웃 구역단위에서는 0을 0으로 나누게 되므로 “부정(not-a-number)”의 결과가 발생하게 된다. 물론 zero.policy를 TRUE로 설정한 경우에는 문제가 없다. 행표준화 스타일에서는 는 과 같아진다.

(nb\_q |>   
 nb2listw(style = "W") -> lw\_q\_W) |>   
 spweights.constants() |>   
 data.frame() |>   
 subset(select = c(n, S0, S1, S2))  
# n S0 S1 S2  
# 1 2495 2495 958 10406

역거리 가중치는 많은 애플리케이션에서 사용된다. 밀집된 역거리 행렬이 사용되는 경우 많은 역거리가 거의 0에 가깝고, 특히 공간 프로세스 행렬 자체가 밀집되어 있는 경우 실제적인 의미는 미약하다. 역거리 가중치는 보통의 다음의 절차를 통해 생성된다. 우선 엣지의 길이를 산출하고, 단위를 바꾸고(예시의 경우 미터에서 킬로미터로 변환)어 가중치가 너무 크거나 작지 않도록 하고, 그 값을 역수로 바꾼 후, nb2listw의 glist 인자를 통해 전달한다.

nb\_d183 |>   
 nbdists(coords) |>   
 lapply(function(x) 1/(x/1000)) -> gwts  
(nb\_d183 |> nb2listw(glist=gwts, style="B") -> lw\_d183\_idw\_B) |>   
 spweights.constants() |>   
 data.frame() |>   
 subset(select=c(n, S0, S1, S2))  
# n S0 S1 S2  
# 1 2495 1841 534 7265

비이웃 단위의 경우 디폴트로 가중치 객체가 생성되지 않는다. 이를 통해 분석가가 이후 과정을 결정할 수 있게 한다.

try(nb\_d16 |> nb2listw(style="B") -> lw\_d16\_B)  
# Error in nb2listw(nb\_d16, style = "B") : Empty neighbour sets found

nb와 listw 객체와 관련된 많은 함수에서는 zero.policy 아규먼트를 활용할 수 있다.

nb\_d16 |>   
 nb2listw(style="B", zero.policy=TRUE) |>   
 spweights.constants(zero.policy=TRUE) |>   
 data.frame() |>   
 subset(select=c(n, S0, S1, S2))  
# n S0 S1 S2  
# 1 2488 15850 31700 506480

spweights.constants 함수의 adjust.n 아규먼트는 디폴트로 TRUE로 설정되어 있어, 비이웃 관측개체 수를 차감하기 때문에 이 작아지며, 통계적 추론에 영향을 미칠 수 있다. 원래 은 아규먼트를 다르게 지정하면 알 수 있다.

## 14.6 고차 이웃의 규정

앞에서 살펴본 것처럼 퀸 인접성 기반 이웃 객체의 특성은 다음과 같다.

nb\_q  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 14242   
# Percentage nonzero weights: 0.229   
# Average number of links: 5.71

가 의 이웃이고, 가 의 이웃인 경우, 즉 이웃 그래프에서 두 단계를 거쳐 에서 까지의 이웃을 나타내는 객체를 만들고자 한다면, nblag를 사용할 수 있다. 이 함수는 자동으로 에서 로 가는 자기 이웃을 제거한다.

(nb\_q |> nblag(2) -> nb\_q2)[[2]]  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 32930   
# Percentage nonzero weights: 0.529   
# Average number of links: 13.2

nblag\_cumul 함수는 주어진 모든 차원의 이웃 목록을 누적한다.

nblag\_cumul(nb\_q2)  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 47172   
# Percentage nonzero weights: 0.758   
# Average number of links: 18.9

union.nb 집합 연산은 두 개의 객체를 받아들이며, 여기서는 동일한 결과를 생성한다.

union.nb(nb\_q2[[2]], nb\_q2[[1]])  
# Neighbour list object:  
# Number of regions: 2495   
# Number of nonzero links: 47172   
# Percentage nonzero weights: 0.758   
# Average number of links: 18.9

앞에서 이웃 객체를 그래프 재현으로 전환하였는데, 해당 그래프 객체를 이용하면 그래프를 탐색하는 데 몇 단계가 필요한지에 대한 정보를 얻을 수 있다.

diameter(g1)  
# [1] 52

각 관측개체에서 그래프를 통해 가장 짧은 경로로 도달하는 데 필요한 단계 수를 계산하여 크기의 sps 행렬을 생성한다. 이를 통해 동일한 최대 값을 얻을 수 있다.

g1 |> shortest.paths() -> sps  
(sps |> apply(2, max) -> spmax) |> max()  
# [1] 52

최대 값을 가진 지방자치단체는 Lutowiska로, 남동부의 말단부에 위치해 우크라이나와 국경을 맞대고 있다.

mr <- which.max(spmax)  
pol\_pres15$name0[mr]  
# [1] "Lutowiska"

그림 14.3은 연접성 기반 이웃과 거리 기반 이웃이 매우 유사하다는 점을 잘 보여준다. 어떤 경우에는 거리 기반 이웃 방법이 선호된다. 왜냐하면 모든 관측개체가 다른 모든 관측개체와 관련된다는 점을 명확히 보여주기 때문이다. 그러나 공간적 자기상관 및 공간 회귀 모델을 위한 검정은 공간적 프로세스 모델의 역(inverse)을 사용하고, 이것은 회귀계수의 벡터와 공간가중행렬의 곱을 여러번 더한 합이므로, 결국 모든 관측개체와 다른 모든 관측개체 간의 관계가 이러한 과정 속에 포함될 수 밖에 없다. 희소 연접 이웃 객체를 사용하면 이러한 복잡한 관계를 명시적으로 만들 필요 없이 풍부한 의존성 구조를 표현할 수 있다.

|  |
| --- |
| 그림 14.2: Lutowiska에 이르는 최단 경로 수와 Lutowiska까지의 거리의 관련성. 왼쪽은 Lutowiska에 이르는 최단 경로 수를 나타낸 지도이고, 오른쪽은 Lutowiska에 이르는 최단 경로 수와 Lutowiska까지의 거리의 관계를 나타낸 그래프이다. |

## 14.7 연습문제

1. 어떤 종류의 지오메트리 스포트가 이웃 객체를 생성하는 함수에 적합한가?
2. 이웃 객체를 생성하는 함수 중 어떤 것이 평면 재현에만 적합한가?
3. 체스판에서 queen 연접성 대신 rook 연접성을 선택하면 어떤 차이가 발생하는가?
4. 이웃 집합의 카디널리티(이웃 수)와 행표준화 가중치 사이에는 어떤 관계가 있으며, 이것이 어떻게 엣지 효과(edge effect) 분석을 가능하게 하는가? 연습문제 3에서 만든 체스판을 사용하고, rook와 queen 이웃 각각에 대해 설명하라.

# 15. 공간적 자기상관 측도

에어리어 데이터를 분석할 때, 공간적 자기상관이 존재한다면, 독립적인 관찰이라는 기본 가정에 입각한 것과는 다른 추론 방식이 적용되어야 한다는 점은 오래전부터 인식되어 왔다. 공간적 자기상관이 존재하는 경우, 관측개체 의 값을 관측개체 의 값으로 예측할 수 있는데, 는 에 인접한 이웃들의 집합인 의 한 원소이다. 선구적인 연구 결과(Moran 1948; Geary 1954)가 점진적인 수용 과정을 거치고(사회과학 분야의 예: Duncan, Cuzzort, and Duncan 1961). 이후 이러한 결과들이 정리되고 확장되어 정량적 분석을 위한 기본 도구로 발전해 나갔다(Cliff and Ord 1973, 1981).

Cliff와 Ord(1973)는 다양한 공간적 통계량, 즉 조인-카운트 통계량, 모런 통계량(Moran’s ), 기어리 통계량(Geary’s )에 공통적으로 적용가능한 분포 이론을 확립하고자 했으며, 이러한 과정에 공간 가중치 행렬 개념을 일반화하고 확장하는 것이 힘을 쏟았다. 전체 연구 지역에 대해 하나의 공간적 자기상관 값을 반환한다는 의미에서 이러한 통계량을 전역적 측도라고 부르며, 이후에 개별 구역단위에 공간적 자기상관 값을 산출하는 국지적 측도(Getis and Ord 1992; Anselin 1995)의 발전으로 이어졌다.

**spdep** 패키지는 다양한 측도를 계산할 수 있게 해준다는 의미에서 뿐만 아니라 측도간 비교 분석을 진작한다는 의미로 설계되었다. 이러한 이유로 측도 함수는 컴파일된 코드가 아닌 R로 작성되었으며, 최근 선보인 **rgeoda** 패키지(Li and Anselin 2021; Anselin, Li, and Koschinsky 2021)보다 속도가 느리긴 하지만 더 유연하다.

## 15.1 측도와 프로세스 오지정

Tobler(1970)가 말한 지리학의 제1법칙, 즉 “모든 것은 다른 모든 것과 연관되어 있다. 그러나 가까이 있는 것은 멀리 떨어져 있는 것보다 더 많이 연관되어 있다”가 만고의 진리인 것으로 취급되어서는 안된다. 이는 지나치게 단순화된 개념으로 다른 잠재적 문제들, 즉 개체화(entitation)의 문제(역자주: 사용된 공간단위가 해당 공간적 현상을 분석하기에 적당한 것인가와 관련된 문제로 소위 ’공간단위 임의성의 문제(MAUP, modifiable areal unit problem)’와 관련된 것이다), 스포트의 문제, 기타 오지정(misspecification)의 문제(역자주: 통계 모델에서 해당 현상을 설명하기에 적절하지 않은 방식으로 변수 설정이 이루어진 경우를 통칭하는 용어인데, 주로 주요 변수의 누락이 야기하는 문제를 지적한다)를 덮어 버릴 수 있다. 관측 단위의 크기가 기저의 공간 프로세스의 스케일에 부합하는가? 주어진 관측 단위 하에서 표출된 관심 변수의 공간적 패턴이 다른 변수의 공간적 패턴으로 설명될 수 있는가?

토블러(Tobler, 1970)의 논문은 올슨(Olsson, 1970)의 논문과 함께 *Economic Geography*의 특별호에 실렸다. 하지만 올슨은 공간적 자기상관이 공간적 현상에 필연적으로 내재되어 있는 것이 아니라, 부적절한 개체화, 누락된 변수, 그리고/또는 부적절하게 설정된 함수 관계에 의해 발생하는 경우가 더 많다는 점을 간파했다. Olsson의 핵심 인용구는 228쪽에 있다.

이러한 자기상관의 존재는 Tobler(1970)의 “모든 것은 다른 모든 것과 연관되어 있다. 그러나 가까이 있는 것은 멀리 떨어져 있는 것보다 더 많이 연관되어 있다”는 주장에 동의하고 싶어지게 만든다. 그러나 한편으로 이러한 자기상관이 체계적인 오지정 오류를 감추는 것처럼 보인다는 사실은 이 주장을 ’지리학의 제1법칙’으로 격상시키는 것이 우선은 성급하다는 점을 시사한다. 최악의 경우, 이 주장은 사후 오류(post hoc fallacy)(단순한 우연을 인과 관계로 잘못 해석하는 오류)의 공간적 변형에 불과한 것일 수 있다.

“제1법칙”으로서의 지위는 존 스노(John Snow)가 콜레라의 원인이 수인성이라는 것을 지도로부터 유도해냈다는 믿음과 매우 유사하다. 이는 GIS를 홍보하는 좋은 방법일 수 있지만, 부정확한 것이다. Snow는 Soho를 방문하기 전에 이미 강력한 작업 가설을 가지고 있었고, 그 지도는 Broad Street 펌프가 차단된 후 그의 가설이 맞다는 것을 문서화하기 위해 작성된 것이었다(Brody et al. 2000).

불행히도 공간적 자기상관 측도는 우리가 데이터를 모델링하는 과정에서 저지른 다른 오지정 오류를 반영한다(Schabenberger and Gotway 2005; McMillen 2003). 참고로, 모런 통계량은 다음과 같이 주어진다(Cliff and Ord 1981, 17).

여기서 , 는 해당 변수의 개의 관측값 중 하나, , , 는 공간적 가중치, 이다. 먼저 우리는 무작위 확률 변수에 대해 공간적 자기상관을 검토하고자 하는데, 정규성 가정(아규먼트를 randomisation=FALSE로 지정) 하에서 모런 검정을 행한다. 검정통계량은 로, 산출된 z-값을 와 를 가진 정규분포와 대조한다. 아래는 무작위 확률 변수에 공간적 자기상관이 존재하지 않는다는 검정 결과이다.

library(spdep) |> suppressPackageStartupMessages()  
library(parallel)  
glance\_htest <- function(ht) c(ht$estimate,   
 "Std deviate" = unname(ht$statistic),   
 "p.value" = unname(ht$p.value))  
set.seed(1)  
(pol\_pres15 |>   
 nrow() |>   
 rnorm() -> x) |>   
 moran.test(lw\_q\_B, randomisation = FALSE,  
 alternative = "two.sided") |>   
 glance\_htest()  
# Moran I statistic Expectation Variance   
# -0.004772 -0.000401 0.000140   
# Std deviate p.value   
# -0.369320 0.711889

그런데 약간의 공간적 경향성을 가진 가상의 변수를 생성하고 그것을 또 다른 변수로 간주하는 대신 원 무작위 변수와 합산했다고 하자. 이 상태에서 동일한 검정을 적용하면 이번에는 강한 공간적 자기상관이 나타나게 된다. 공간적 경향성을 가진 변수를 모델에 포함시키지 않았다는 측면에서 변수 누락의 문제가 있는 것인데 그것이 공간적 자기상관의 문제로 전이된 것이다.

beta <- 0.0015  
coords |>   
 st\_coordinates() |>   
 subset(select = 1, drop = TRUE) |>   
 (function(x) x/1000)() -> t  
(x + beta \* t -> x\_t) |>   
 moran.test(lw\_q\_B, randomisation = FALSE,  
 alternative = "two.sided") |>   
 glance\_htest()  
# Moran I statistic Expectation Variance   
# 0.043403 -0.000401 0.000140   
# Std deviate p.value   
# 3.701491 0.000214

공간적 경향성을 가진 변수를 선형 모델에 독립적인 변수로 투입하면, 잔차에서 공간적 자기상관이 사라진 것을 확인할 수 있다.

lm(x\_t ~ t) |>   
 lm.morantest(lw\_q\_B, alternative = "two.sided") |>   
 glance\_htest()  
# Observed Moran I Expectation Variance Std deviate   
# -0.004777 -0.000789 0.000140 -0.337306   
# p.value   
# 0.735886

다양한 공간적 자기상관 측도가 다양한 R 패키지에서 이용가능하다. 그 중에서 **spdep** 패키지(Bivand 2022b)가 핵심적 역할을 담당하며, 패키지간 차이는 주로 실행 디자인과 관련되어 있다(Bivand and Wong 2018). **spdep** 패키지를 통해 회귀 잔차에 대한 전역적 및 국지적 모런 검정을 행할 수 있으며, 다른 패키지와 달리 정확(exact) 및 안장점 근사법(Tiefelsdorf 2002; Bivand, Müller, and Reder 2009)을 제공한다.

## 15.2 전역적 측도

### 15.2.1 범주 데이터를 위한 조인-카운트 통계량

먼저 조인-카운트(join-count) 통계량을 살펴본다. 여기서 joincount.test 함수는 "factor" 값 벡터 fx와 listw 객체를 입력받고, **stats** 패키지에서 정의된 htest(가설 검정) 객체들의 리스트를 반환한다. 반환되는 htest 객체는 fx 아규먼트의 각 수준에 대해 하나씩 생성된다. 관찰된 카운트는 동일한 범주 수준을 가진 이웃쌍의 수로, 이를 동일 색상 조인(same-colour joins)이라고 한다.

args(joincount.test)

# function (fx, listw, zero.policy = NULL, alternative = "greater",  
# sampling = "nonfree", spChk = NULL, adjust.n = TRUE)

이 함수는 가설 검정을 위한 alternative 아규먼트와 통계량의 분산을 구성하는 기준을 나타내는 sampling 아규먼트를 입력받는다. 기본값인 "nonfree"는 개념적으로 분석적 순열(analytical permutation)과 동일한 것이다. spChk 아규먼트는 이전 버전과의 호환성을 위해 유지된다. 참고로, 앞에서 살펴본 지방자치단체 및 바르샤바 데이터에 대한 범주 유형별 카운트는 다음과 같다.

(pol\_pres15 |>   
 st\_drop\_geometry() |>   
 subset(select = types, drop = TRUE) -> Types) |>   
 table()  
#   
# Rural Urban Urban/rural Warsaw Borough   
# 1563 303 611 18

네 개의 범주 유형, 즉 네 개의 수준이 있으므로, htest 객체 리스트를 재배열하여 추정된 결과를 나타내는 행렬을 생성한다. 관찰된 동일-컬러 조인 카운트는 입력된 범주 수준의 카운트를 기반으로 한 기대값과 함께 표로 정리된다. 예를 들어 바르샤바 구역 간에는 결합이 거의 없을 것으로 예상된다. 이는 구역의 수가 매우 적기 때문이다. 분산 계산은 선택된 listw 객체의 기본 상수와 입력 범주 수준의 카운트를 사용하여 수행된다. z-값은 관찰된 조인 카운트와 기대값의 차이를 분산의 제곱근으로 나누어 얻어진다.

조인-카운트 검정은 컬러 멀티-컬러 조인 카운트 상황으로의 확장을 위해 수정되었다(Upton and Fingleton 1985). **spdep**에서 joincount.multi 함수로 구현되었는데 비자유 샘플링(non-free sampling)을 기반으로 표를 반환하며, p-값은 보고하지 않는다.

Types |> joincount.multi(listw = lw\_q\_B)  
# Joincount Expected Variance z-value  
# Rural:Rural 3087.000 2793.920 1126.534 8.73  
# Urban:Urban 110.000 104.719 93.299 0.55  
# Urban/rural:Urban/rural 656.000 426.526 331.759 12.60  
# Warsaw Borough:Warsaw Borough 41.000 0.350 0.347 68.96  
# Urban:Rural 668.000 1083.941 708.209 -15.63  
# Urban/rural:Rural 2359.000 2185.769 1267.131 4.87  
# Urban/rural:Urban 171.000 423.729 352.190 -13.47  
# Warsaw Borough:Rural 12.000 64.393 46.460 -7.69  
# Warsaw Borough:Urban 9.000 12.483 11.758 -1.02  
# Warsaw Borough:Urban/rural 8.000 25.172 22.354 -3.63  
# Jtot 3227.000 3795.486 1496.398 -14.70

바이너리 가중치를 적용하는 상항에서는 조인-카운트의 합에 해당 조인에 대한 가중치를 곱한 값도 여전히 정수로 나타난다. 만약 행표준화 가중치를 적용하게 되면, 가중치가 대부분 1보다 작은 분수가 되기 때문에 카운트, 기대값, 분산은 달라진다. 그러나 최종적인 z-값에는 큰 변화가 없다.

그러나 역거리 기반의 listw 객체를 사용하면 z-값이 크게 변한다. 이는 가까운 센트로이드에 상대적으로 강한 가중치가 부여되기 때문이다.

Types |> joincount.multi(listw = lw\_d183\_idw\_B)  
# Joincount Expected Variance z-value  
# Rural:Rural 3.46e+02 3.61e+02 4.93e+01 -2.10  
# Urban:Urban 2.90e+01 1.35e+01 2.23e+00 10.39  
# Urban/rural:Urban/rural 4.65e+01 5.51e+01 9.61e+00 -2.79  
# Warsaw Borough:Warsaw Borough 1.68e+01 4.53e-02 6.61e-03 206.38  
# Urban:Rural 2.02e+02 1.40e+02 2.36e+01 12.73  
# Urban/rural:Rural 2.25e+02 2.83e+02 3.59e+01 -9.59  
# Urban/rural:Urban 3.65e+01 5.48e+01 8.86e+00 -6.14  
# Warsaw Borough:Rural 5.65e+00 8.33e+00 1.73e+00 -2.04  
# Warsaw Borough:Urban 9.18e+00 1.61e+00 2.54e-01 15.01  
# Warsaw Borough:Urban/rural 3.27e+00 3.25e+00 5.52e-01 0.02  
# Jtot 4.82e+02 4.91e+02 4.16e+01 -1.38

### 15.2.2 모런 통계량

**spdep** 패키지에서 모런 통계량은 moran.test 함수로 구현되어 있다. 이 함수는 joincount.test 함수와 유사한 아규먼트를 가지지만, 샘플링 대신 랜덤화(randomisation)를 사용하여 측정의 분산을 계산한다. 또한 수치 값 대신 순위를 사용할 수도 있다(Cliff and Ord 1981, 46). 일부 오래된 소프트웨어에서는 분산 항목의 마지막 구성 요소가 생략된채 나타나는데, drop.EI2 아규먼트는 그것을 재현해준다.

args(moran.test)

# function (x, listw, randomisation = TRUE, zero.policy = NULL,  
# alternative = "greater", rank = FALSE, na.action = na.fail,  
# spChk = NULL, adjust.n = TRUE, drop.EI2 = FALSE)

randomisation 아규먼트의 디폴트는 TRUE인데, FALSE를 지정하게 되면 정규성 가정 하에서의 검정을 수행하게 된다. 아래의 사례를 통해, 단일 변수에 정규성 가정을 적용한 검정 결과와 절편만이 포함된 회귀 모델의 잔차에 대한 랜덤화 가정 하에서의 검정 결과가 동일하게 나타난다는 점을 알게 될 것이다. 해당 변수는 2015년 폴란드 대통령 선거의 투표율이다. randomisation의 철자는 Cliff와 Ord (1973)의 방식에 따른 것이다.

pol\_pres15 |>   
 st\_drop\_geometry() |>   
 subset(select = I\_turnout, drop = TRUE) -> I\_turnout

I\_turnout |> moran.test(listw = lw\_q\_B, randomisation = FALSE) |>   
 glance\_htest()  
# Moran I statistic Expectation Variance   
# 0.691434 -0.000401 0.000140   
# Std deviate p.value   
# 58.461349 0.000000

lm.morantest 함수는 resfun 아규먼트도 가지고 있는데, 이를 통해 검정을 위한 잔차를 추출하는 함수를 설정할 수 있으며, 종속 변수의 다른 중요한 특징을 모델화할 수 있게 해준다(Cliff and Ord 1981, 203). 단일 변수에 대한 표준 테스트와 비교하기 위해 여기서는 절편만 포함된 회귀 모델을 사용하였으며, 그 결과가 동일함을 확인할 수 있다.

lm(I\_turnout ~ 1, pol\_pres15) |>   
 lm.morantest(listw = lw\_q\_B) |>   
 glance\_htest()  
# Observed Moran I Expectation Variance Std deviate   
# 0.691434 -0.000401 0.000140 58.461349   
# p.value   
# 0.000000

정규성과 랜덤화 가정 하에서의 검정 간 차이는 변수의 첨도가 정상 범위를 벗어나는 경우 추가 항목이 더해진다는 점이다. 이때 사용되는 척도는 고전적인 첨도 측정법이다. 디폴트인 랜덤화 가정 하에서는 결과가 크게 변화하지는 않는다.

(I\_turnout |>   
 moran.test(listw = lw\_q\_B) -> mtr) |>   
 glance\_htest()  
# Moran I statistic Expectation Variance   
# 0.691434 -0.000401 0.000140   
# Std deviate p.value   
# 58.459835 0.000000

1970년대 초반부터 몬테카를로(Monte Carlo) 검정, 즉 호프(Hope) 유형의 검정 혹은 순열 부트스트랩으로 알려진 검정 절차에 대한 관심이 제기되었다. 기본적으로 moran.mc 함수는 "htest" 객체를 반환하지만, 내부적으로는 boot::boot을 사용하기 때문에 return\_boot=TRUE로 설정하면 "boot" 객체를 반환할 수도 있다. 또한 시뮬레이션 횟수는 nsim으로 지정하는데, 이는 관측값들이 무작위로 섞이는 횟수를 나타낸다.

set.seed(1)  
I\_turnout |>   
 moran.mc(listw = lw\_q\_B, nsim = 999,   
 return\_boot = TRUE) -> mmc

부트스트랩 순열은 각 무작위 순열의 결과를 보존하며, 통계량의 관측값(여기서는 모런 통계값)과 랜덤화 시뮬레이션의 평균값(와 동일한 역할)의 차이, 그리고 램덤화 시뮬레이션의 표준 편차를 보고한다.

몬테카를로 시뮬레이션의 분산과 랜덤화 가정 하의 분석적 분산을 비교하면 보통 큰 차이가 없으며, 이는 몬테카를로 검정의 무용론을 주장할 근거가 된다.

c("Permutation bootstrap" = var(mmc$t),   
 "Analytical randomisation" = unname(mtr$estimate[3]))  
# Permutation bootstrap Analytical randomisation   
# 0.000144 0.000140

기어리의 전역적 통계량(Geary’s )는 moran.test 함수와 거의 동일한 아규먼트 구조를 따라 geary.test 함수로 구현되어 있다. 게티스-오드(Getis-Ord ) 테스트는 추가적인 아규먼트를 가지는데, 이는 Bivand와 Wong (2018)이 지적한 바처럼, 초기 통계량의 변종이 이후에 나타났고 주로 거리 기반 이웃 규정에서는 생성되는 무이웃 관측개체를 다루는 방식에서의 차이를 반영해야 하기 때문이다. Getis와 Ord(1992)의 194 페이지를 보면, 의 경우, 와는 달리 합산 제약이 완화되어 가 자기 자신을 이웃으로 포함할 수 있다(따라서 모든 관측개체가 적어도 하나의 이웃을 가지므로 무이웃 문제도 해결된다).

마지막으로, 경험적 베이즈 모런 통계량은 비율 데이터에서 공간적 자기상관을 평가할 때 분모를 고려할 수 있게 해준다(Assunção and Reis 1999). 지금까지 우리는 공간단위별로 유효 투표수와 투표권을 가진 인구수의 비율을 고려했으나, EBImoran.mc 함수를 사용하면 투표권을 가진 인구수가 적은 공간단위에서의 극단적인 비율값이 가지는 통계적 불확실성을 반영할 수 있게 된다. 그러나 결과에는 큰 영향이 없다.

지금까지 다룬 전역적 공간적 자기상관 통계량은 이웃 그래프에 기반한 공간 가중치를 활용한 측도들이다. 이러한 측도들을 매우 정교한 통계적 툴이라고 부르기는 어려울 것 같다. 왜냐하면 결과의 해석이 해당 변수에 대한 평균 모델을 어떻게 설정하느냐에 크게 의존하기 때문이다. 만약 평균 모델이 절편만을 가진 모델이라면, 전역적 통계량은 공간적 자기상관 뿐만 아니라 다른 모든 종류의 오지정 문제에 동시에 반응하게 된다(역자주: 공간적 자기상관 통계량이 측정하고 검정하는 것 속에는 다른 오지정 문제가 포함될 수 있기 때문에, 공간적 자기상관 통계량을 정교한 측도라고 평가할 수 없다는 의미이다). 공간단위의 선정과 관련된 개체화의 문제가 일반적으로는 오지정 문제의 핵심적인 원인이 된다.

## 15.3 국지적 측도

전역적 측도의 한계를 극복하기 위한 노력의 일환으로, 1990년대 초반부터 국지적 공간연관성 지표들이 등장하기 시작했다(Anselin 1995; Getis and Ord 1992, 1996).

또한, 모런 플롯을 통해 관심 변수와 관심 변수의 공간 지연값(spatially lagged values) 간의 관련성을 시각화할 수 있게 되었다. 일반적으로 행 표준화 가중치를 사용하면 축을 보다 더 직접적으로 비교할 수 있게 된다(Anselin 1996). moran.plot 함수는 또한 영향력 측정 객체(influence measures object)를 반환하는데, 전역적 모런 통계값을 나타내는 직선의 기울기에 상대적으로 큰 영향을 미치는 관측개체를 표시할 수 있게 해준다. 그림 15.1에서 높은 영향력을 미치는 관측개체가 상당히 많다는 점을 확인할 수 있다. 이러한 관측값과 공간지연 관측값의 쌍이 집합적으로 글로벌 측도를 구성하지만, 세부적으로도 탐색할 수 있다. 모런 플롯에서, 왼쪽 하단 사분면에는 낮은 값-낮은 값(low-low) 쌍이, 오른쪽 상단 사분면에는 높은 값-높은 값(high-high) 쌍이, 왼쪽 상단 사분면과 오른쪽 하단 사분면에는 낮은 값-높은 값(low-high) 및 높은 값-낮은 값(high-low) 쌍이 나타난다. 앞의 두 개의 사분면에 비해 상대적으로 적은 수가 포함되어 있다. moran.plot 함수에서 사분면은 변수값와 공간지연값의 평균을 기준으로 나뉘지만, 해당 값을 표준화한 경우라면 0을 기준으로 나뉘기도 한다.

|  |
| --- |
| 그림 15.1: 투표율에 대한 모런 플롯으로 행 표준화된 가중치가 사용되었다. |

반환된 객체에서 회귀 영향 측정값(hat influence measure)을 추출하여 지도를 그리면(그림 15.2) 경계부에 위치한 공간단위들이 높은 값을 나타내는 것을 볼 수 있다(아마도 행표준화 공간가중치행렬을 적용했기 때문일 수도 있음)(역자주: 경계부의 공간단위는 상대적으로 적은 수의 이웃을 가지며, 행 표준화가 적용되었을 때 상대적으로 높은 가중치를 부여받게 된다.). 대도시 혹은 그 주변 지역에서도 높은 값을 관찰할 수 있다.

library(tmap)  
# Breaking News: tmap 3.x is retiring. Please test v4, e.g. with  
# remotes::install\_github('r-tmap/tmap')  
pol\_pres15$hat\_value <- infl\_W$hat  
tm\_shape(pol\_pres15) + tm\_fill("hat\_value")

|  |
| --- |
| 그림 15.2: 모런 회귀 영향 측도값의 분포로 행 표준화 행렬을 적용함. |

### 15.3.1 국지적 모런 통계량

Bivand와 Wong(2018)은 국지적 모런 통계량(Moran’s )​과 국지적 게티스-오드 통계량(Getis-Ord )​와 같은 국지적 지표 사용에 영향을 미치는 문제들을 다루었다. 일부 문제는 국지적 지표 계산에 영향을 미치며, 다른 문제는 해당 값의 통계적 추론에 영향을 미친다. 개의 관측단위에 대해 개의 통계값이 산출되기 때문에, 다중 비교(multiple comparison) 문제를 해결할 필요가 있다. Caldas de Castro와 Singer(2006)는 전형적인 데이터셋과 시뮬레이션 실험을 바탕으로, 확률 값에 대한 오발견율(FDR, false discovery rate) 조정이 무조정의 결과보다 흥미로운 클러스터를 더 잘 보여줄 것이라고 결론지었다. 이를 바탕으로, Anselin(2019)은 FDR 조정을 재정의된 “유의성” 기준치(Benjamin et al., 2018)와 결합하는 방식을 제안했다. 예를 들어, 기존의 0.1, 0.05, 0.01 대신 0.01, 0.005, 0.001을 사용하는 방식이다. 또한 *유의적*(significant) 대신 *흥미로운*(interesting)이라는 용어를 사용할 것을 권고하였다. 이러한 논의는 Bivand(2022a)에서 더 자세히 다루어진다. 전역적 통계량과 마찬가지로, 오지정 문제는 여전히 혼란의 원인이 되며, 전역적 공간적 자기상관이 존재하는 상황에서 국지적 공간적 자기상관을 해석하는 것은 여전히 도전 과제이다(Ord and Getis 2001; Tiefelsdorf 2002; Bivand, Müller, and Reder 2009).

args(localmoran)

# function (x, listw, zero.policy = NULL, na.action = na.fail,  
# conditional = TRUE, alternative = "two.sided", mlvar = TRUE,  
# spChk = NULL, adjust.x = FALSE)

Bivand와 Wong(2018)는 국지적 모런 통계량의 표준 편차에 대해 분석적 공식에 기반한 방식과 조건부 순열(permutation)에 기반한 방식을 비교한 바 있다. 이에 대해 Sauer et al.(2021)은 그러한 비교가 오해에 기반하고 있음을 명확히 보여주었다. Sokal, Oden, 그리고 Thomson(1998)은 총괄(total) 순열과 조건부 순열에 기반한 국지적 모런 통계량의 표준 편차 공식을 제시한 바 있는데, Bivand와 Wong(2018)에서 사용된 분석 공식은 이전의 관행에 기반하여 총괄 순열만 사용하며, 따라서 시뮬레이션 조건부 순열과 일치하지 않는다. 적시에 제안된 풀 리퀘스트 덕분에, 이제 localmoran 함수는 Sokal, Oden, 그리고 Thomson(1998)의 부록에 수록되어 있는 대안적 공식을 사용한 conditional 논리 아규먼트(디폴트는 TRUE)를 포함한다. localmoran 함수의 mlvar와 adjust.x 아규먼트가 필요한 이유는 Bivand와 Wong(2018)에 나타나 있으며, 다른 패키지의 결과와 비교할 수 있게 해준다. "two.sided" 확률 값을 디포트로 설정하여 다음과 같은 결과를 얻을 수 있다.

I\_turnout |>   
 localmoran(listw = lw\_q\_W) -> locm

국지적 모런 통계량을 합산한 후 공간 가중치의 합으로 나누면 전역적 모런 통계량과 같아지며, 국지적 수준에서의 양 혹은 음의 공간적 자기상관의 존재를 확인할 수 있다.

all.equal(sum(locm[,1])/Szero(lw\_q\_W),   
 unname(moran.test(I\_turnout, lw\_q\_W)$estimate[1]))  
# [1] TRUE

stats::p.adjust를 사용하여 다중 비교를 조정한 결과, 조정을 적용하지 않을 경우 2,495개의 국지적 지표 중 15% 이상이 p-value < 0.005를 가지는 것으로 나타났다. 그러나 전체 오류율(FWER, family-wise error rate)(역자주: 다중 가설 검정에서 최소한 하나의 개별 검정에서 제1종 오류가 발생할 확률을 의미하는데, 예를 들어 전체 오류률을 0.05로 하는 경우 개별 검정에서의 오류률은 반드시 0.05보다 적은 값이어야 함.)을 제어하기 위해 Bonferroni 조정을 사용할 경우 이는 1.5%로 감소한다. 두 가지 다른 옵션을 통한 조정도 가능한데, "fdr"은 Benjamini와 Hochberg(1995)의 허위 발견율(FDR) 조정 방법으로 약 6%를 나타내고, "BY"는 Benjamini와 Yekutieli(2001)의 또 다른 FDR 조정 방법으로 약 2.5%를 보였다.

pva <- function(pv) cbind("none" = pv,   
 "FDR" = p.adjust(pv, "fdr"), "BY" = p.adjust(pv, "BY"),  
 "Bonferroni" = p.adjust(pv, "bonferroni"))  
locm |>   
 subset(select = "Pr(z != E(Ii))", drop = TRUE) |>   
 pva() -> pvsp  
f <- function(x) sum(x < 0.005)  
apply(pvsp, 2, f)  
# none FDR BY Bonferroni   
# 385 149 64 38

전역적 측도에서는 분석적 방법을 대체할 수 있는 추론 방식으로 부트스트랩 순열을 사용할 수 있다. 그런데 분석적 분산 도출 방식과 순열 방식 모두 모든 관측값을 뒤섞는 것에 기반하고 있다. 국지적 측도에서는 조건부 순열이 사용되어야 하며, 이는 관측개체 의 값을 고정한채로 나머지 개의 값을 무작위로 샘플링하여 해당 이웃의 값을 결정하는 방식이다. 조건부 순열은 localmoran\_perm 함수로 제공되며, 여러 컴퓨팅 노드가 있는 경우 병렬 샘플링이 가능하고, 각 컴퓨팅 노드에서 난수 생성기의 시드를 설정할 수도 있다. nsim 아규먼트로 시뮬레이션 수를 설정할 수 있는데, 시뮬레이션된 값 중 관측된 모런 통계값의 순위에 기반한 확률값 추정치의 정밀도를 결정한다.

library(parallel)  
invisible(spdep::set.coresOption(max(detectCores()-1L, 1L)))  
I\_turnout |>   
 localmoran\_perm(listw = lw\_q\_W, nsim = 9999,   
 iseed = 1) -> locm\_p

그 결과, 다중 비교 조정이 없는 경우 15% 이상의 관측값이 양측 p값이 0.005 미만으로 나타났고, Bonferroni 조정을 적용했을 때는 약 1.5%가 p값이 0.005 미만이었다. 여기서 p값은 순열 샘플의 표준 편차와 정규 분포를 사용하여 계산된 것이다.

locm\_p |>   
 subset(select = "Pr(z != E(Ii))", drop = TRUE) |>   
 pva() -> pvsp  
apply(pvsp, 2, f)  
# none FDR BY Bonferroni   
# 380 148 64 39

해당 변수가 반드시 정규 분포를 따른다고 가정할 수 없으므로, 모든 시뮬레이션 값 중에 관측값의 순위를 결정하고 그것에 기반하여 균등 분포에서의 확률 값을 계산하는 방식으로 p값을 구할 수도 있다.

locm\_p |>   
 subset(select = "Pr(z != E(Ii)) Sim", drop = TRUE) |>   
 pva() -> pvsp  
apply(pvsp, 2, f)  
# none FDR BY Bonferroni   
# 391 127 0 0

위의 결과는 "BY"와 본페로니 방식을 적용할 경우 9999개의 샘플의 경우에는 *흥미로운* 위치가 없다는 점을 말하고 있다. 하지만 샘플 수를 999999로 증가시킨다면 *흥미로운* 위치가 발견될 수 있다. FDR 조정과 흥미로운 컷오프 0.005를 적용하면 약 5%의 위치가 흥미로운 위치로 나타난다.

pol\_pres15$locm\_pv <- p.adjust(locm[, "Pr(z != E(Ii))"], "fdr")  
pol\_pres15$locm\_std\_pv <- p.adjust(locm\_p[, "Pr(z != E(Ii))"],   
 "fdr")  
pol\_pres15$locm\_p\_pv <- p.adjust(locm\_p[, "Pr(z != E(Ii)) Sim"],  
 "fdr")

|  |
| --- |
| 그림 15.3: 국지적 모런 통계량의 FDR 유의확률 값. 왼쪽 상단 패널에는 분석적 조건부 유의확률이, 오른쪽 상단 패널에는 순열 표준 편차를 활용한 조건부 유의확률이, 왼쪽 하단 패널에는 순열 순위 조건부 유의확률이 나타나 있다. 투표율 데이터에 행 표준화 이웃이 적용되었다. |

FDR 조정과 흥미로운 위치 컷오프 0.005를 적용하여 진행한 결과, 그림 15.3에서 볼 수 있듯이 분석적 조건부 접근법, 순열 샘플링에서 샘플된 값들의 적률(moments)을 사용하는 접근법, 순열 샘플 내에서 관측 값의 순위를 사용하는 접근법 모두 비슷한 지도 패턴을 보여준다. 이는 입력 변수의 분포가 정규 분포에 매우 가깝기 때문에 발생한 것이다.

국지적 모런 통계치를 제시할 때는 종종 “핫스팟” 맵이 사용된다. 국지적 모런 통계값은 입력 변수의 값이 낮든 높든 강한 양의 자기상관을 보이기만 하면 높은 값을 가지므로, 유사한 값을 가지는 이웃들의 “클러스터”가 어디에서 발생하는지를 낮은 값의 클러스터와 높은 값의 클러스터를 구분하여 보여주는 것이 어렵다. 이를 해결하기 위해 모런 플롯을 사용할 수 있는데, 변수값과 공간 지체값의 평균을 기준으로 범주형 사분면 변수를 생성한다. 그런 다음, 기준 확률 값과 조정 과정에 의거해 모런 통계값이 “흥미로운” 값으로 간주되지 않으면 해당 개체의 사분면 범주값은 NA로 설정된다. 아래의 사례에서는 FDR 조정된 조건부 분석 확률 값이 사용되었는데(그림 15.3, 왼쪽 상단 패널), 53개의 관측치가 “Low-Low” 클러스터에, 96개의 관측치는 “High-High” 클러스터에 속한다. 이는 표준 편차 기반의 순열 p-값(그림 15.3, 오른쪽 상단 패널)에서도 비슷하지만, 순위 기반의 순열 p-값은 “High-High” 클러스터의 수가 줄고 “Low-Low” 클러스터의 군집 수가 증가한다(그림 15.3, 왼쪽 하단 패널).

quadr <- attr(locm, "quadr")$mean  
a <- table(addNA(quadr))  
locm |> hotspot(Prname="Pr(z != E(Ii))", cutoff = 0.005,   
 droplevels=FALSE) -> pol\_pres15$hs\_an\_q  
locm\_p |> hotspot(Prname="Pr(z != E(Ii))", cutoff = 0.005,   
 droplevels=FALSE) -> pol\_pres15$hs\_ac\_q   
locm\_p |> hotspot(Prname="Pr(z != E(Ii)) Sim", cutoff = 0.005,  
 droplevels = FALSE) -> pol\_pres15$hs\_cp\_q  
b <- table(addNA(pol\_pres15$hs\_an\_q))  
c <- table(addNA(pol\_pres15$hs\_ac\_q))  
d <- table(addNA(pol\_pres15$hs\_cp\_q))  
t(rbind("Moran plot quadrants" = a, "Analytical cond." = b,   
 "Permutation std. cond." = c, "Permutation rank cond." = d))  
# Moran plot quadrants Analytical cond.  
# Low-Low 1040 53  
# High-Low 264 0  
# Low-High 213 0  
# High-High 978 96  
# <NA> 0 2346  
# Permutation std. cond. Permutation rank cond.  
# Low-Low 53 56  
# High-Low 0 0  
# Low-High 0 0  
# High-High 95 71  
# <NA> 2347 2368

pol\_pres15$hs\_an\_q <- droplevels(pol\_pres15$hs\_an\_q)  
pol\_pres15$hs\_ac\_q <- droplevels(pol\_pres15$hs\_ac\_q)  
pol\_pres15$hs\_cp\_q <- droplevels(pol\_pres15$hs\_cp\_q)

|  |
| --- |
| 그림 15.4: 국지적 모런 통계량의 FDR 조정 핫스팟 지도(유의수준은 0.005). 왼쪽 상단 패널에는 분석적 조건부 유의확률이, 오른쪽 상단 패널에는 순열 표준 편차를 활용한 조건부 유의확률이, 왼쪽 하단 패널에는 순열 순위 조건부 유의확률이 적용된 경우가 나타나 있다. 투표율 데이터에 행 표준화 이웃이 적용되었다. |

Figure 15.4는 세 가지 접근 방식인 분석적 조건부 표준 편차, 순열 기반 표준 편차, 순위 기반 확률 값에 대해 확률 값 컷오프를 선택한 경우 FDR 조정된 *흥미로운* 클러스터들 사이에 거의 차이가 없음을 보여준다. “High-High” 클러스터의 핵심부는 대도시 지역이다.

Tiefelsdorf (2002)는 국지적 모런 통계량의 표준 편차 계산을 위한 표준 접근 방식에 수치적 추정치를 추가해야 한다고 주장하며, 안장점 근사를 통해 이 목표를 달성하는 것이 계산적으로 효율적인 방법임을 보여준다. localmoran.sad 함수는 첫 번째 아규먼트로 적합된 선형 모델을 받으므로, 우선 기본 모델(절편만 있는 모델)을 적합시켜야 한다. 그런데 유권자 수가 구역별로 매우 상이하므로 그것의 효과를 통제하기 위해 결국 가중 선형 모델을 적합시킨다.

lm(I\_turnout ~ 1) -> lm\_null

안장점 근사 방식은 조건부 순열만큼 계산이 많이 소요된다. 많은 샘플에 대해 단일한 측정값을 계산하는 대신, 각 국지적 근사에 대해 상당한 수치 계산이 필요하기 때문이다.

lm\_null |> localmoran.sad(nb = nb\_q, style = "W",  
 alternative = "two.sided") |>  
 summary() -> locm\_sad\_null

안장점 근사법의 주요 장점은 단순히 수치 변수를 사용하는 대신, 적합된 선형 모델을 사용하여 잔차를 분석한다는 점이다. 절편만 있는 모델을 사용하면 결과는 국지적 모런 통계량과 유사하지만, 관찰값에 가중치를 부여할 수 있다. 여기서는 투표권이 있는 사람의 수를 기준으로 가중치를 부여하며, 이를 통해 작은 단위의 관찰값은 낮은 가중치를 받게 된다.

lm(I\_turnout ~ 1, weights = pol\_pres15$I\_entitled\_to\_vote) ->  
 lm\_null\_weights  
lm\_null\_weights |>  
 localmoran.sad(nb = nb\_q, style = "W",  
 alternative = "two.sided") |>  
 summary() -> locm\_sad\_null\_weights

다음으로, 농촌, 도시 및 기타 유형의 관찰 단위를 구분하는 범주형 변수를 추가한다.

lm(I\_turnout ~ Types, weights=pol\_pres15$I\_entitled\_to\_vote) ->  
 lm\_types  
lm\_types |> localmoran.sad(nb = nb\_q, style = "W",  
 alternative = "two.sided") |>  
 summary() -> locm\_sad\_types

locm\_sad\_null |> hotspot(Prname="Pr. (Sad)",  
 cutoff=0.005) -> pol\_pres15$locm\_sad0  
locm\_sad\_null\_weights |> hotspot(Prname="Pr. (Sad)",  
 cutoff = 0.005) -> pol\_pres15$locm\_sad1  
locm\_sad\_types |> hotspot(Prname="Pr. (Sad)",  
 cutoff = 0.005) -> pol\_pres15$locm\_sad2

|  |
| --- |
| 그림 15.5: 국지적 모런 통계량의 FDR 조정 하스팟 클러스터 지도로 양측검정이고 흥미로운 컷오프값으로 0.005가 적용되었다. 왼쪽 상단 패널에는 순열 표준 편차를 활용한 조건부 유의확률이, 오른쪽 상단 패널에는 기본(절편만 있는) 모델의 안장점 근사에 근거한 유의확률이 , 왼쪽 하단 패널에는 가중 기본(절편만 있는) 모델의 안장점 근사에 근거한 유의확률이, 오른쪽 하단 패널에는 가중 유형 모델의 안장점 근사에 근거한 유의확률이 적용된 경우가 나타나 있다. 투표율 데이터에 행 표준화 이웃이 적용되었다. |

rbind(null = append(table(addNA(pol\_pres15$locm\_sad0)),  
 c("Low-High" = 0), 1),  
 weighted = append(table(addNA(pol\_pres15$locm\_sad1)),  
 c("Low-High" = 0), 1),  
 type\_weighted = append(table(addNA(pol\_pres15$locm\_sad2)),  
 c("Low-High" = 0), 1))  
# Low-Low Low-High High-High <NA>  
# null 19 0 55 2421  
# weighted 9 0 52 2434  
# type\_weighted 13 0 81 2401

그림 15.5의 왼쪽 상단 패널에는 비교의 목적으로 순열 순위 기반 클러스터가 제시되어 있다. 안장점 근사법은 더 풍부한 평균 모델을 사용할 수 있게 해주고, 에서의 회귀 잔차 값을 그 이웃들의 값과 연결시킨다는 의미에서 본질적으로 국지적인 근사법이기 때문에, 안정접 근사법에 기반한 나머지 세 개의 패널은 제법 다른 패턴을 보여주고 있다. 절편만 있는 (기본) 모델은 표준적 결과와 매우 유사하지만, 유권자 수에 따른 가중치 부여로 말미암아 대부분의 “Low-Low” 클러스터가 제거된 모습을 보인다. 범주형 유형 변수를 추가하면 도시의 “High-High” 클러스터가 강화되지만, 바르샤바 구역들은 흥미로운 군집 핵으로서 제외된다. 높은 투표율을 보인 바르샤바의 중앙 구역들은 마찬가지로 높은 투표율을 보이는 다른 구역들에 의해 둘러싸여 있는데, 이것은 공간적 자기상관에 기인한 것이 아니라 모두 대도시 구역들로 이루어졌기 때문이다. 또한, 전역적 공간 프로세스가 통합된 경우 안장점 근사법을 사용할 수 있는데, 이는 표준 접근 방식에서의 전역 및 지역 공간 자가 상관의 결합을 제거한다.

같은 결과를 정확 접근법을 사용하여 달성할 수도 있지만, 수치 적분이 실패할 수 있어 표준 편차의 정확한 추정값 대신 NaN을 반환하는 경우가 있을 수 있으므로 더 많은 조정이 필요할 수 있다(Bivand, Müller, and Reder 2009).

lm\_types |> localmoran.exact(nb = nb\_q, style = "W",   
 alternative = "two.sided", useTP=TRUE, truncErr=1e-8) |>   
 as.data.frame() -> locm\_ex\_types

locm\_ex\_types |> hotspot(Prname = "Pr. (exact)",  
 cutoff = 0.005) -> pol\_pres15$locm\_ex

|  |
| --- |
| 그림 15.6: 국지적 모런 통계량의 FDR 조정 하스팟 클러스터 지도로 양측검정이고 흥미로운 컷오프값으로 0.005가 적용되었다. 왼쪽 패널에는 가중 유형 모델의 안장점 근사에 근거한 유의확률이, 오른편 패널에는 가중 유형 정확 접근법에 의거한 유의확률 의 결과가 나타나 있다. 투표율 데이터에 행 표준화 이웃이 적용되었다. |

그림 15.6에서 볼 수 있듯이, 정확 방법과 안장점 근사 방법은 동일한 회귀 잔차, 다중 비교 조정 방법, 그리고 기준값 수준을 적용했을 때 거의 동일한 클러스터 분류 결과를 산출한다. 아래에 나타나 있는 것처럼, 둘 사이의 불일치는 정확 방법이 4개의 부가적인 관측개체를 흥미로운 케이스로 탐지한다는 것이다.

table(Saddlepoint = addNA(pol\_pres15$locm\_sad2),  
 exact = addNA(pol\_pres15$locm\_ex))  
# exact  
# Saddlepoint Low-Low High-High <NA>  
# Low-Low 13 0 0  
# High-High 0 81 0  
# <NA> 2 2 2397

### 15.3.2 국지적 게티스-오드 통계량

국지적 게티스-오드 측도(Getis-Ord )(Getis and Ord 1992, 1996)는 표준 편차값으로 주어진다. include.self 아규먼트를 통해 이웃 객체에 자신을 포함하도록 설정하면 측도값을 계산할 수도 있다. return\_internals=TRUE를 설정하면 관측값, 대값, 그리고 그에 대한 분석적 분산값이 반환된다.

I\_turnout |>   
 localG(lw\_q\_W, return\_internals = TRUE) -> locG

순열에 기반한 가설검정도 가능하다.

I\_turnout |>   
 localG\_perm(lw\_q\_W, nsim = 9999, iseed = 1) -> locG\_p

분석적 표준편차에 기반한 유의확률, 순열-기반 표준편차에 기반한 유의확률(첫 번째 두 열과 행), 순열 순위-기반 유의확률간의 상관관계는 매우 강하다.

cor(cbind(localG=attr(locG, "internals")[, "Pr(z != E(Gi))"],   
 attr(locG\_p, "internals")[, c("Pr(z != E(Gi))",   
 "Pr(z != E(Gi)) Sim")]))  
# localG Pr(z != E(Gi)) Pr(z != E(Gi)) Sim  
# localG 1 1 1  
# Pr(z != E(Gi)) 1 1 1  
# Pr(z != E(Gi)) Sim 1 1 1

### 15.3.3 국지적 기어리 통계량

Anselin(2019)은 Anselin(1995)를 확장한 내용으로, Josiah Parry의 기여(풀 리퀘스트: <https://github.com/r-spatial/spdep/pull/66>) 덕분에 최근 **spdep**에 추가되었다. 와 에 사용된 조건부 순열 프레임워크가 국지적 기어리 통계량()에도 적용된다.

I\_turnout |>   
 localC\_perm(lw\_q\_W, nsim=9999, iseed=1) -> locC\_p

순열 표준편차 기반 유의확률과 순위-기반 유의확률값은 에 비해 상관관계가 그리 높지 않은데, 이는 부분적으로 에서는 값들의 유사성을 값들의 곱이 아닌 값들 간의 차이 함수로 표현하는데, 공간적 자기상관에 대한 이러한 개념화의 차이가 반영된 것으로 해석된다.

cor(attr(locC\_p, "pseudo-p")[, c("Pr(z != E(Ci))",  
 "Pr(z != E(Ci)) Sim")])  
# Pr(z != E(Ci)) Pr(z != E(Ci)) Sim  
# Pr(z != E(Ci)) 1.000 0.966  
# Pr(z != E(Ci)) Sim 0.966 1.000

locC\_p |> hotspot(Prname = "Pr(z != E(Ci)) Sim",  
 cutoff = 0.005) -> pol\_pres15$hs\_C  
locG\_p |> hotspot(Prname = "Pr(z != E(Gi)) Sim",  
 cutoff = 0.005) -> pol\_pres15$hs\_G

|  |
| --- |
| 그림 15.7: FDR 조정 하스팟 클러스터 지도로 양측검정이고 흥미로운 컷오프값으로 0.005가 적용되었다. 왼쪽 패널에는 국지적 모런 통계량에 의한 공간 클러스터, 가운데 패널에는 국지적 게티스-오드 통계량에 의한 공간 클러스터, 오른편 패널에는 국지적 기어리 통계량에 의한 공간 클러스터가 나타나 있다. 투표율 데이터에 행 표준화 이웃이 적용되었다. |

그림 15.7은 동일한 변수(투표율)와 동일한 공간 가중치를 사용하여 , , 를 통해 흥미로운 클러스터 핵심부를 식별한 결과가 매우 유사함을 보여준다. 통계적 추론을 위해 순위-기반 순열 FDR 조정 확률값을 사용했고 기준값이 적용되었다. 대부분의 경우, "High-High" 클러스터 핵심은 도시 지역이며, "Low-Low" 클러스터의 핵심은 북부의 인구 밀도가 낮은 농촌 지역과 남부 국경 근처의 독일 소수 민족 지역이다. 세 가지 측도는 클러스터 핵심을 명명하는 데 약간 다른 전략을 사용한다. 는 모런 산점도 플롯의 사분면을 사용하고, 는 입력 변수의 평균을 기준으로 "Low"와 "High"로 나눈다(이는 튜플의 첫 번째 요소와 동일). 는 입력 변수의 평균값 기준으로 하지만 공간 지체값에 대해서는 0을 기준으로 나눈다. 이전과 마찬가지로 해당 케이스를 갖지 않는 클러스터 범주는 제외된다.

비교를 위해, 다변량 로 넘어가기 전에 두 번째(최종) 라운드 투표율에 대한 단변량 를 살펴보자. 1차 투표에서 상위 두 후보 간의 결선 투표는 1차 투표에서 명확한 선호를 보이지 않았던 일부 유권자를 동원할 가능성이 있지만, 1차 투표에서 탈락한 후보에게 강한 충성심을 가졌던 일부 유권자들의 참여를 저해할 가능성도 있다.

pol\_pres15 |>   
 st\_drop\_geometry() |>   
 subset(select = II\_turnout) |>   
 localC\_perm(lw\_q\_W, nsim=9999, iseed=1) -> locC\_p\_II

locC\_p\_II |> hotspot(Prname = "Pr(z != E(Ci)) Sim",  
 cutoff = 0.005) -> pol\_pres15$hs\_C\_II

다변량 (Anselin 2019)는 단변량 의 합을 변수 개수로 나눈 값으로 계산되지만, 순열은 고정되어 변수 간 상관관계가 변하지 않도록 한다.

pol\_pres15 |>   
 st\_drop\_geometry() |>   
 subset(select = c(I\_turnout, II\_turnout)) |>  
 localC\_perm(lw\_q\_W, nsim=9999, iseed=1) -> locMvC\_p

다변량 가 단변량 의 평균값임을 다음과 같이 확인해 볼 수 있다.

all.equal(locMvC\_p, (locC\_p+locC\_p\_II)/2,  
 check.attributes = FALSE)  
# [1] TRUE

locMvC\_p |> hotspot(Prname = "Pr(z != E(Ci)) Sim",  
 cutoff = 0.005) -> pol\_pres15$hs\_MvC

|  |
| --- |
| 그림 15.8: FDR 조정 하스팟 클러스터 지도로 양측검정이고 흥미로운 컷오프값으로 0.005가 적용되었다. 왼쪽 패널에는 첫 번째 라운드 투표율에 대한 공간 클러스터, 가운데 패널에는 두 번째 라운드 투표율에 대한 공간 클러스터, 오른편 패널에는 두 라운드 모두의 투표율에 대한 공간 클러스터가 나타나 있다. 행 표준화 이웃이 적용되었다. |

그림 15.8은 다변량 측도가 개별 단변량 측도에서 흥미로운 케이스로 확인된 관측개체들을 기본적으로 결합하고 있음을 보여준다. 이를 세분화하기 위해 1차 및 2차 라운드 단변량 측도값을 결합하고, 이를 다변량 측정값과 대조하여 표로 정리할 수 있다.

table(droplevels(interaction(addNA(pol\_pres15$hs\_C),  
 addNA(pol\_pres15$hs\_C\_II), sep=":")),   
 addNA(pol\_pres15$hs\_MvC))  
#   
# Positive <NA>  
# High-High:High-High 81 0  
# NA:High-High 41 27  
# Low-Low:Low-Low 25 0  
# NA:Low-Low 43 11  
# NA:Other Positive 1 0  
# NA:Negative 0 1  
# High-High:NA 15 0  
# Low-Low:NA 11 3  
# NA:NA 36 2200

다변량 에서는 흥미로운 케이스로 나타난 관측개체들 중 47개는 단변량 어디에서도 흥미로운 케이스로 나타나지 않은 것이다(FDR, 기준값 0.005). 1차 및 2차 라운드에서 모두 흥미로운 것으로 나타난 관측값은 거의 모두 다변량에서도 흥미로운 것으로 간주되지만, 두 라운드 중 하나에서만 흥미로운 것으로 나타난 관측값의 경우는 보다 혼재된 결과가 도출되었다.

### 15.3.4 rgeoda 패키지

**rgeoda**(Li and Anselin 2022) 패키지는 Geoda의 R 래퍼 패키지로, **spdep** 패키지와 유사한 기능을 제공함으로써 에어리어 데이터에서의 공간적 자기상관을 탐색할 수 있게 해준다. 활성 객체는 컴파일된 작업 공간의 메모리 주소를 참조하는 포인터로 관리된다. 즉 모든 오퍼레이션이 컴파일된 코드를 통해 실행되기 때문에(GeoDa와 동일한 방식) **rgeoda** 패키지는 빠른 성능을 자랑한다. 그러나 수정이나 기능 확장이 필요할 때는 유연성이 떨어진다.

library(rgeoda)  
Geoda\_w <- queen\_weights(pol\_pres15)  
summary(Geoda\_w)  
# name value  
# 1 number of observations: 2495  
# 2 is symmetric: TRUE  
# 3 sparsity: 0.00228786229774178  
# 4 # min neighbors: 1  
# 5 # max neighbors: 13  
# 6 # mean neighbors: 5.70821643286573  
# 7 # median neighbors: 6  
# 8 has isolates: FALSE

비교를 위해, 2015년 폴란드 대통령 선거 투표율에 대한 다변량 를 지표를 살펴보자.

lisa <- local\_multigeary(Geoda\_w,   
 pol\_pres15[c("I\_turnout", "II\_turnout")],   
 cpu\_threads = max(detectCores() - 1, 1),  
 permutations = 99999, seed = 1)

연접 이웃을 비교 하면 위에서 poly2nb 함수를 통해 규정된 것과 동일하다.

all.equal(card(nb\_q), lisa\_num\_nbrs(lisa),   
 check.attributes = FALSE)  
# [1] TRUE

다변량 값도 위와 동일하다.

all.equal(lisa\_values(lisa), c(locMvC\_p),  
 check.attributes = FALSE)  
# [1] TRUE

한 가지 차이점은 **rgeoda**에서 사용하는 접힌(folded) 양측 순위-기반 순열 유의확률 값의 범위가 로 제한된다는 점이며, 이는 **spdep**에서도 계산되지만, 다르게 처리될 수 있다는 것이다.

apply(attr(locMvC\_p, "pseudo-p")[,c("Pr(z != E(Ci)) Sim",   
 "Pr(folded) Sim")], 2, range)  
# Pr(z != E(Ci)) Sim Pr(folded) Sim  
# [1,] 0.0002 0.0001  
# [2,] 0.9990 0.4995

이것은 범위에서 에 해당하는 컷오프 값이 범위에서는 에 해당한다는 의미이다.

locMvC\_p |> hotspot(Prname = "Pr(folded) Sim",  
 cutoff = 0.0025) -> pol\_pres15$hs\_MvCa

따라서 local\_multigeary 함수는 클러스터 핵심 클래스를 설정할 때 기본 컷오프 값인 를 사용했지만, 우리는 컷오프 값을 더 엄격하게 설정하고 lisa 객체의 출력 구성 요소에 FDR 조정을 적용하여 결과를 더 신뢰성 있게 만들 수 있다.

mvc <- factor(lisa\_clusters(lisa), levels=0:2,  
 labels = lisa\_labels(lisa)[1:3])  
is.na(mvc) <- p.adjust(lisa\_pvalues(lisa), "fdr") >= 0.0025  
pol\_pres15$geoda\_mvc <- droplevels(mvc)

regoda 순열 방식을 통해 약 80개의 추가 관측값이 흥미로운 케이스인 것으로 확인되었으며, 구현 세부 사항에 대한 추가 분석이 여전히 진행 중이다.

addmargins(table(spdep = addNA(pol\_pres15$hs\_MvCa),  
 rgeoda = addNA(pol\_pres15$geoda\_mvc)))  
# rgeoda  
# spdep Positive <NA> Sum  
# Positive 249 4 253  
# <NA> 75 2167 2242  
# Sum 324 2171 2495

|  |
| --- |
| 그림 15.9: FDR 조정 다변량 기어리 핫스팟 클러스터 지도로 양측검정이고 *흥미로운* 컷오프값으로 범위에서 0.0025가 .적용되었다. 왼편 패널은 합산 투표율을 **spdep**에 적용한 결과이고, 오른편 패널은 합산 투표율을 **rgeoda**에 적용한 결과이다. 행 표준화 이웃이 적용되었다. |

## 15.4 연습문제

1. 체스판의 조인-카운트 통계값이 rook과 queen 이웃 간에 매우 상이하게 나타나는 이유를 설명하라.
2. 15.1절에서 보여준 시뮬레이션을 체스판 다각형과 행 표준화된 queen 연접 이웃을 사용하여 반복하라. 공간적 자기상관이 보통 (피할 수 없는) 데이터의 모델 오지정을 나타낸다는 것을 이해하는 것이 중요한 이유를 설명하라.
3. 국지적 공간적 자기상관 통계량에 FDR 조정이 권장되는 이유를 설명하라.
4. 물음 2의 시뮬레이션 데이터에 대해 분석적 조건 접근법과 안장점 근사법을 사용한 의 표준편차 값(검정 통계값)을 비교하라. 안장점 근사법의 장단점을 설명하라.

# 16. 공간적 회귀분석

에어리어 서포트를 가지는 한 변수가 주어졌을 때 그것이 보여주는 공간적 패턴의 해석에 집중하는 것은 의미있는 일이다. 그러나 해당 변수의 공간적 패턴을 설명하고자 한다면, 즉 해당 변수를 반응 변수로 간주한다면 얘기는 좀 달라진다. 왜냐하면 반응 변수과 보여주는 공간적 패턴을 주로 공변량(및 그 함수 형태)에 기인한 것일 가능성이 크고, 개별 변수의 영향력이 작동하는 공간적 범위에 영향을 받기 때문이다. 이차원 공간적 자기회귀 모델은 공변량 없이 시작되었으며, 시계열 분석 모델과 명확한 연관성을 가지고 있다(Whittle, 1954). 이후 선형 모델 잔차에서 공간적 자기상관을 검정하는 방법과 자기회귀 성분을 반응 변수나 잔차에 적용하는 모델로 확장되었는데, 자기회귀 성분을 잔차에 적용하는 모델은 잔차에 대한 검정과 일치하는 것이었다(Cliff and Ord, 1972, 1973). 이러한 에어리어 데이터에 대한 “격자(lattice)” 모델은 일반적으로 연접성 행렬 형태의 이웃 관계 그래프를 사용하여 관측값 간의 공간적 의존성을 나타낸다.

물론, 일반화 최소제곱 모델이나 **nlme** 및 기타 여러 패키지에서 제공하는 (일반화) 선형 또는 비선형 혼합효과 모델에서 공간적 자기상관을 다루는 데 반드시 이웃 관계 그래프를 사용할 필요는 없다(Pinheiro and Bates, 2000). 관측값 간의 거리 함수를 사용하거나 공간적 자기상관을 모델링하기 위해 베리오그램을 적합하는 것 역시 공간적 회귀 모델의 일종이라고 말할 수 있다. 이러한 접근은 기저 과정에 대한 더 명확한 이해를 제공한다고 평가되었으며(Wall, 2004), 지구통계학에 기반을 두고 있다. 예를 들어, **glmmTMB** 패키지는 이 접근법을 활용해 공간적 회귀분석을 성공적으로 수행한다(Brooks et al., 2017). 하지만 본 장에서는 공간 가중치 행렬을 사용하는 공간적 회귀분석만을 고려할 것이다.

## 16.1 마르코프 랜덤 필드와 다수준 모델

질병 지도화(disease mapping) 연구에서는 공간적으로 구조화된 임의 효과를 다루기 위해 조건부(conditional) 자기회귀(CAR) 모델과 내재적(intrinsic) 자기회귀(ICAR) 모델을 사용하는 광범위한 연구가 존재한다. 이러한 모델은 다수준 모델로 확장될 수 있으며, 공간적으로 구조화된 임의 효과가 모델의 서로 다른 수준에서 적용될 수 있다(Bivand et al., 2017). 여러 변형을 시도하기 위해, 우리는 트랙트(tract) 수준과 모델 출력 구역 집계(model output zone aggregated) 수준을 두 단계로 나누어 이웃이 없는 관측값을 제거해야 한다. 트랙트 수준에서 이웃 관계가 줄어들면 모델 출력 구역 수준에서도 이웃이 없는 결과가 발생할 수 있기 때문이다. 모델 추정 함수의 대부분은 family 아규먼트를 사용하며, 이웃 간 관계를 마르코프 랜덤 필드로 표현한 공간 랜덤 효과를 통해 각 관측값에 대해 일반화 선형 혼합효과 모델을 적합한다. 다수준 모델의 경우, 랜덤 효과는 그룹 수준에서 모델화될 수 있으며, 아래의 예시에 적용되어 있다.

공간적 회귀분석을 혼합 효과 모델에 의거해 설명하는 Pinheiro and Bates (2000)와 McCulloch and Searle (2001)의 논리가 중요한데, Gómez-Rubio (2019)의 표현식을 딸라 설명하고자 한다. 반응 변수 , 고정 공변량 , 회귀계수 , 오차 항 이 주어진 상태에서 랜덤 효과 가 첨가되면, 가우시안 선형 혼합 효과 모델은 다음과 같이 정의된다.

여기서 는 랜덤 효과를 위한 고정 디자인 행렬이다. 만약 개의 랜덤 효과가 있다면, 단위 행렬이 되고, 만일 관측값이 개의 그룹으로 집계되어 의 랜덤 효과만 있다면 각 관측값이 어느 그룹에 속하는지를 나타내는 행렬이 된다. 랜덤 효과는 다변량 정규 분포 로 모델화되며, 은 랜덤 효과의 분산-공분산 행렬이다.

CAR 모델(Besag, 1974)과 SAR(simultaneous autoregressive) 모델(Ord, 1975; Hepple, 1976)이 분리되어 발전해 왔는데, 이러한 구분의 유익함은 사실 별로 없다. CAR 모델과 SAR 모델은 밀접하게 연관되어 있음에도 불구하고 , 이 두 분야는 유사한 모델을 적용하는 경험을 공유하는데 어려움을 겪고 있다. 이러한 어려움은 CAR과 SAR 모델을 정리한 핵심 저작(Ripley, 1981, 1988; Cressie, 1993)을 참고하는 경우에도 마찬가지이다. Ripley는 SAR의 분산을 제시한 바 있는데(Ripley, 1981, 89), 여기서는 (정밀도 행렬로도 알려짐)로 나타낸다.

여기서 는 공간적 자기상관 파라미터이고, 는 공간적 의존성을 나타내는 비단일(non-singular) 공간 가중치 행렬이다. CAR 분산을 다음과 같이 주어진다.

여기서 는 대칭적이고 엄격히 양의 정의인 공간 가중치 행렬이다. 내재적 CAR 모델의 경우, 공간적 자가상관 파라미터를 추정하지 않으며 다음과 같은 형태로 주어진다.

여기서 는 앞서 언급한 것처럼 대칭적이고 엄격히 양의 정의인 공간 가중치 행렬이며,  
은 의 행 합산값을 나타낸다. Besag-York-Mollié 모델은 내재적 CAR 공간 구조화된 랜덤 효과와 비구조화된 랜덤 효과를 포함한다. Leroux 모델은 비구조화된 랜덤 효과와 공간 구조화된 랜덤 효과의 행렬 구성 요소를 결합하며, 공간 구조화된 랜덤 효과는 내재적 CAR의 정의를 따르는 것으로 간주된다.

이 모델들의 정의에 대해서는 Gómez-Rubio (2020)를 참조할 수 있고, Besag-York-Mollié와 Leroux 모델에 영향을 미치는 추정 문제에 대해서는 Gerber and Furrer (2015)에서 다루어진다.

에어리어 데이터 모델링의 이론적 기초를 다루고 있는 최근의 서적(Gaetan and Guyon, 2010; Van Lieshout, 2019)은 SAR 모델과 CAR 모델의 유사성을 몇몇 장에서 지적하고 있다. 배경 정보에 관심이 있는 독자는 이 서적들을 참고하면 된다.

### 16.1.1 보스턴 주택가격 데이터셋

여기서는 보스턴 주택 데이터셋을 사용할 것이다. 이 데이터셋은 센서스 트랙트 경계에 맞추어 재구성되었다(Bivand 2017). 원 데이터셋은 506개의 센서스 트랙트로 구성되었으며, 깨끗한 공기에 대한 지불 의사를 추정하기 위해 헤도닉 모델을 사용했다. 반응 변수는 1970년 센서스의 주택 가치에 대한 서열 응답(클래스)에 대한 카운트로부터 생성되었다. 이 반응 변수는 센서스 원본에서 좌측 및 우측 검열되었으며, 가우시안 분포로 처리되었다. 주요 공변량은 연간 질소산화물(NOX) 수준을 보여주는 기상 모델에 기반하여 생성되었으며 소수의 모델 산출 구역 단위로 주어진다. 주택 응답 수는 트랙트와 모델 출력 존별로도 다르다. 다른 공변량도 여러 개 포함되어 있으며, 일부는 트랙트 수준에서 측정되었고, 일부는 타운 단위로만 측정되었는데, 이 타운들은 대체로 대기오염 모델 출력 존과 일치한다.

우리는 먼저 **spData** 패캐지(Bivand, Nowosad, and Lovelace 2022)에서 506개 트랙트 데이터셋을 불러온 뒤, 인접성 이웃 객체를 생성하고 이를 기반으로 행 표준화된 공간 가중치 객체를 생성한다.

library(sf)  
library(spData)  
boston\_506 <- st\_read(system.file("shapes/boston\_tracts.shp",  
 package = "spData")[1], quiet = TRUE)

nb\_q <- spdep::poly2nb(boston\_506)  
lw\_q <- spdep::nb2listw(nb\_q, style = "W")

중위 주택 가격 데이터를 살펴보면, 검열된 값들은 결측값으로 지정되었으며, 이로 인해 17개의 트랙트가 영향을 받았음을 알 수 있다.

table(boston\_506$censored)  
#   
# left no right   
# 2 489 15

summary(boston\_506$median)  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
# 5600 16800 21000 21749 24700 50000 17

다음으로, 비검열 주택 가격값을 가진 나머지 489개 트랙트로 서브셋을 만들고, 여기에 맞추어 이웃 객체도 조정한다. 이렇게 하면 이웃이 없는 관측값이 하나 존재하게 된다.

boston\_506$CHAS <- as.factor(boston\_506$CHAS)  
boston\_489 <- boston\_506[!is.na(boston\_506$median),]  
nb\_q\_489 <- spdep::poly2nb(boston\_489)  
lw\_q\_489 <- spdep::nb2listw(nb\_q\_489, style = "W",  
 zero.policy = TRUE)

상위 집계 수준을 정의하는 NOX\_ID 변수를 활용하여 트랙트 수준의 데이터를 대기오염 모델 출력 구역으로 집계한다. 이웃 객체와 행 표준화된 공간 가중치 객체를 생성하고, NOX 변수는 평균값을 사용해 재계산하고, 찰스강 상의 관측단위인지 아닌지에 대한 더미 변수 CHAS도 재계산한다. 여기서는 5.3.1절에서 설명한 원칙, 즉 공간적으로 외연적인 변수와 내포적인 변수를 다르게 취급하는 원칙을 따른다. NOX와 CHAS는 모두 카운트 변수가 아니기 때문에 합계를 통해 재계산할 수 없다.

agg\_96 <- list(as.character(boston\_506$NOX\_ID))  
boston\_96 <- aggregate(boston\_506[, "NOX\_ID"], by = agg\_96,  
 unique)  
nb\_q\_96 <- spdep::poly2nb(boston\_96)  
lw\_q\_96 <- spdep::nb2listw(nb\_q\_96)  
boston\_96$NOX <- aggregate(boston\_506$NOX, agg\_96, mean)$x  
boston\_96$CHAS <-  
 aggregate(as.integer(boston\_506$CHAS)-1, agg\_96, max)$x

반응 변수의 집계는 **matrixStats** 패키지의 weightedMedian 함수를 통해 이루어지는데, 주택 가격 클래스의 중간값을 기준으로 계산된다. 주택 가격 클래스별 주택 수를 산출하는 것이 매우 중요한데, 센서스 공표 데이터를 검증하는데 사용될 수도 있다. 트랙트 수준에서 weightedMedian을 적용하면 공표 값이 계산되어 나오는 것을 확인할 수 있다. 반응 변수에 대한 집계 결과, 두 개의 출력 구역의 가중 중위값이 주택 가격에 대한 최상위 클래스의 한계 값인 5만 달러는 초과하는 것으로 확인되어 최종적으로 제거되었다. 이렇게 큰 가중 중위값은 최상의 클래스의 중간값을 어떻게 설정하는냐와 관련된 문제이기도 하다. 주택 가격 클래스별 주택 수를 계산하는 것은 공간적으로 외연적인 집계에 해당하므로 재계산을 위한 함수로 합계를 적용하는 것이 적절하다.

nms <- names(boston\_506)  
ccounts <- 23:31  
for (nm in nms[c(22, ccounts, 36)]) {  
 boston\_96[[nm]] <- aggregate(boston\_506[[nm]], agg\_96, sum)$x  
}  
br2 <-   
 c(3.50, 6.25, 8.75, 12.5, 17.5, 22.5, 30, 42.5, 60) \* 1000  
counts <- as.data.frame(boston\_96)[, nms[ccounts]]  
f <- function(x) matrixStats::weightedMedian(x = br2, w = x,  
 interpolate = TRUE)  
boston\_96$median <- apply(counts, 1, f)  
is.na(boston\_96$median) <- boston\_96$median > 50000  
summary(boston\_96$median)  
# Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
# 9009 20417 23523 25263 30073 49496 2

나머지 공변량은 트랙트 수준의 센스서 인구 수를 활용하여 가중 평균을 산출함으로서(Bivand 2017) 집계 절차를 진행한다. 공변량들이 카운트 데이터가 아니라 공간적으로 내포적인 변수이기 때문에 이렇게 한다. 이 집계 과정을 이어서 서브세팅을 한다.

POP <- boston\_506$POP  
f <- function(x) matrixStats::weightedMean(x[,1], x[,2])  
for (nm in nms[c(9:11, 14:19, 21, 33)]) {  
 s0 <- split(data.frame(boston\_506[[nm]], POP), agg\_96)  
 boston\_96[[nm]] <- sapply(s0, f)  
}  
boston\_94 <- boston\_96[!is.na(boston\_96$median),]  
nb\_q\_94 <- spdep::subset.nb(nb\_q\_96, !is.na(boston\_96$median))  
lw\_q\_94 <- spdep::nb2listw(nb\_q\_94, style="W")

이제 두 개의 데이터셋이 두 개의 서로 다른 수준에서 존재한다. 즉, 하위 수준인 센서스 트랙트 수준과 상위 수준인 대기오염 모델 출력 구역 수준이다. 하나는 검열된 관측값을 포함하고 있고, 다른 하나는 이를 제외한 데이터이다.

boston\_94a <- aggregate(boston\_489[,"NOX\_ID"],   
 list(boston\_489$NOX\_ID), unique)  
nb\_q\_94a <- spdep::poly2nb(boston\_94a)  
NOX\_ID\_no\_neighs <-  
 boston\_94a$NOX\_ID[which(spdep::card(nb\_q\_94a) == 0)]  
boston\_487 <- boston\_489[is.na(match(boston\_489$NOX\_ID,  
 NOX\_ID\_no\_neighs)),]  
boston\_93 <- aggregate(boston\_487[, "NOX\_ID"],  
 list(ids = boston\_487$NOX\_ID), unique)  
row.names(boston\_93) <- as.character(boston\_93$NOX\_ID)  
nb\_q\_93 <- spdep::poly2nb(boston\_93,  
 row.names = unique(as.character(boston\_93$NOX\_ID)))

원래 모델은 트랙트별 중위 주택 가격의 로그값과 NOX 값의 제곱의 관련성에 관한 것이었는데, 주로 트랙트별 주택 가격과 관련된 다른 공변량(예: 총 방 수, 총 연령, 민족, 사회적 지위, 중심가까지 거리, 가장 가까운 방사형 도로까지 거리, 범죄율, 도시 수준 변수 등)을 포함하였다(Bivand 2017). 이 데이터를 활용하여 공간 회귀 모델을 적합할 때 발생할 수 있는 문제들에 대해 살펴볼 것이다. 이 데이터는 다수준 이슈를 다루는데도 유용하다.

## 16.2 보스턴 데이터셋에 대한 다수준 모델

ZN, INDUS, NOX, RAD, TAX, 및 PTRATIO 변수는 TASSIM 구역 내에서 변동이 거의 없음을 보여주며, 따라서 다수준 모델에서는 랜덤 효과가 이들의 영향을 흡수할 수 있다.

form <- formula(log(median) ~ CRIM + ZN + INDUS + CHAS +   
 I((NOX\*10)^2) + I(RM^2) + AGE + log(DIS) +  
 log(RAD) + TAX + PTRATIO + I(BB/100) +   
 log(I(LSTAT/100)))

### 16.2.1 IID 랜덤 효과: lme4 패키지의 활용

**lme4** 패키지(Bates et al. 2022)는 모델 출력 구역 수준에서 독립동일분포(IID)를 보이는 비구조적 랜덤 효과를 추가할 수 있게 해준다. 이는 모델 포뮬러에 랜덤 효과 항을 업데이트하는 방식으로 이루어진다.

library(Matrix)  
library(lme4)  
MLM <- lmer(update(form, . ~ . + (1 | NOX\_ID)),   
 data = boston\_487, REML = FALSE)

랜덤 효과를 "sf" 객체에 복사하여 나중에 지도로 표현할 수 있다.

boston\_93$MLM\_re <- ranef(MLM)[[1]][,1]

### 16.2.2 IID와 CAR 랜덤 효과: hglm 패키지의 활용

동일한 모델을 **hglm** 패키지를 사용하여 추정할 수 있는데(Alam, Ronnegard, and Shen 2019), 랜덤 효과 항을 나타내는 한쪽 방향 포뮬러(one-sided formula)를 추가하는 방식으로 이루어진다.

library(hglm) |> suppressPackageStartupMessages()  
suppressWarnings(HGLM\_iid <- hglm(fixed = form,  
 random = ~1 | NOX\_ID,  
 data = boston\_487,  
 family = gaussian()))  
boston\_93$HGLM\_re <- unname(HGLM\_iid$ranef)

**hglm** 패키지는 공간적으로 구조화된 SAR과 CAR 랜덤 효과도 다룰 수 있게 확장되었으며, 이 경우 희소 공간 가중치 행렬이 필요하다(Alam, Rönnegård, and Shen 2015). 여기서는 이진 공간 가중치를 사용한다.

library(spatialreg)  
W <- as(spdep::nb2listw(nb\_q\_93, style = "B"), "CsparseMatrix")

and.family 아규먼트를 사용하여 상위 수준에서 CAR 모델을 적합한다. 이 때 인덱싱 변수인 NOX\_ID의 값이 의 행 이름과 일치하도록 한다.

suppressWarnings(HGLM\_car <- hglm(fixed = form,  
 random = ~ 1 | NOX\_ID,  
 data = boston\_487,  
 family = gaussian(),  
 rand.family = CAR(D=W)))  
boston\_93$HGLM\_ss <- HGLM\_car$ranef[,1]

### 16.2.3 IID와 ICAR 랜덤 효과: R2BayesX 패키지의 활용

**R2BayesX** 패키지(Umlauf et al. 2022)는 공간 다수준 모델을 포함한 다양한 구조화된 가법(additive) 회귀 모델을 지원한다. 지원 모델 중 하나가 상위 레벨에서의 IID 비구조적 랜덤 효과 모델인데, sx 모델 항에 "re" 사양을 사용하면 된다(Umlauf et al. 2015). “MCMC” 메소드를 선택한다.

library(R2BayesX) |> suppressPackageStartupMessages()

BX\_iid <- bayesx(update(form, . ~ . + sx(NOX\_ID, bs = "re")),  
 family = "gaussian", data = boston\_487,  
 method = "MCMC", iterations = 12000,  
 burnin = 2000, step = 2, seed = 123)

boston\_93$BX\_re <- BX\_iid$effects["sx(NOX\_ID):re"][[1]]$Mean

상위 수준의 "nb" 객체를 토대로 "mrf" (Markov Random Field) 공간적으로 구조화된 내재적 CAR 랜덤 효과 사양을 선택한다. "nb" 객체의 "region.id" 속성은 sx 효과 항의 인덱싱 변수와 일치하는 값을 포함할 필요가 있는데, 이는 설계 행렬 의 내부 구조화를 용이하게 하려는 것이다.

RBX\_gra <- nb2gra(nb\_q\_93)  
all.equal(row.names(RBX\_gra), attr(nb\_q\_93, "region.id"))  
# [1] TRUE

위의 내재적 CAR 모델 정의에서 본 것처럼, 이웃 수는 대각선에 입력되지만, 현재 구현에서는 희소가 아닌 밀집 행렬을 사용한다.

all.equal(unname(diag(RBX\_gra)), spdep::card(nb\_q\_93))  
# [1] TRUE

sx 모델 항은 여전히 인덱싱 변수를 포함하며, 이제 내재적 CAR 정밀 행렬을 통과한다.

BX\_mrf <- bayesx(update(form, . ~ . + sx(NOX\_ID, bs = "mrf",  
 map = RBX\_gra)),   
 family = "gaussian", data = boston\_487,  
 method = "MCMC", iterations = 12000,  
 burnin = 2000, step = 2, seed = 123)

boston\_93$BX\_ss <- BX\_mrf$effects["sx(NOX\_ID):mrf"][[1]]$Mean

### 16.2.4 IID, ICAR, Lerouzx 랜덤 효과: INLA 패키지의 활용

Bivand, Gómez-Rubio, and Rue (2015) 및 Gómez-Rubio (2020)는 **INLA** 패키지(Rue, Lindgren, and Teixeira Krainski 2022)와 공간 회귀 모델에 대한 inla 모델 적합 함수의 사용을 설명한다.

library(INLA) |> suppressPackageStartupMessages()

세부적인 차이는 있지만, 고정 모델 공식을 무구조 랜덤 효과 항으로 업데이트하는 접근 방식은 위에서 본 것과 매우 유사하다.

INLA\_iid <- inla(update(form, . ~ . + f(NOX\_ID, model = "iid")),  
 family = "gaussian", data = boston\_487)

boston\_93$INLA\_re <- INLA\_iid$summary.random$NOX\_ID$mean

대부분의 구현과 마찬가지로, 공간 가중치와 인덱싱 변수를 일치시키는 데 주의가 필요하다. 여기서는 NOX\_ID 변수를 직접적으로 사용하는 대신, 1부터 93까지의 인덱스를 사용하는 방식을 취한다.

ID2 <- as.integer(as.factor(boston\_487$NOX\_ID))

동일한 희소 이진 공간 가중치 행렬이 사용되며, 내재적 CAR 표현은 내부적으로 생성된다.

INLA\_ss <- inla(update(form, . ~ . + f(ID2, model = "besag",  
 graph = W)),  
 family = "gaussian", data = boston\_487)

boston\_93$INLA\_ss <- INLA\_ss$summary.random$ID2$mean

Gómez-Rubio (2020)에 의해 제공된 희소 Leroux 표현은 다음과 같이 구성될 수 있다:

M <- Diagonal(nrow(W), rowSums(W)) - W  
Cmatrix <- Diagonal(nrow(M), 1) - M

이 모델은 지정된 정밀 행렬과 함께 "generic1" 모델을 사용하여 추정할 수 있다.

INLA\_lr <- inla(update(form, . ~ . + f(ID2, model = "generic1",  
 Cmatrix = Cmatrix)),  
 family = "gaussian", data = boston\_487)

boston\_93$INLA\_lr <- INLA\_lr$summary.random$ID2$mean

### 16.2.5 ICAR 랜덤 효과: mgcv 패키지의 gam() 함수의 활용

비슷한 방식으로, **mgcv** 패키지의 gam 함수(Wood 2022)는 "nb" 객체를 사용하여 "mrf" 항을 포함할 수 있다. 이 경우, "nb" 객체는 "region.id" 속성의 내용을 이웃 목록 구성 요소의 이름으로 복사해야 하며, 인덱싱 변수는 범주형이어야 한다(Wood 2017).

library(mgcv)  
names(nb\_q\_93) <- attr(nb\_q\_93, "region.id")  
boston\_487$NOX\_ID <- as.factor(boston\_487$NOX\_ID)

공간적으로 구조화된 항의 지정 방식은 위의 예들과 다소 차이가 있지만, 결과는 동일하다. bayesx의 "REML" 방법은 이 경우 gam의 "REML"을 사용하는 것과 동일한 결과를 산출한다.

GAM\_MRF <- gam(update(form, . ~ . + s(NOX\_ID, bs = "mrf",  
 xt = list(nb = nb\_q\_93))),  
 data = boston\_487, method = "REML")

상위 수준의 랜덤 효과는 예측을 통해 추출할 수 있다. 동일한 상위 수준의 공기 질 모델 출력 구역에 속하는 모든 하위 수준 트랙트의 값은 동일하다는 것을 확인할 수 있다.

ssre <- predict(GAM\_MRF, type = "terms",   
 se = FALSE)[, "s(NOX\_ID)"]  
all(sapply(tapply(ssre, list(boston\_487$NOX\_ID), c),  
 function(x) length(unique(round(x, 8))) == 1))  
# [1] TRUE

따라서 각 상위 수준 단위에 대해 첫 번째 값을 반환할 수 있다.

boston\_93$GAM\_ss <- aggregate(ssre, list(boston\_487$NOX\_ID),   
 head, n=1)$x

### 16.2.6 상위-수준 랜덤 효과: 요약

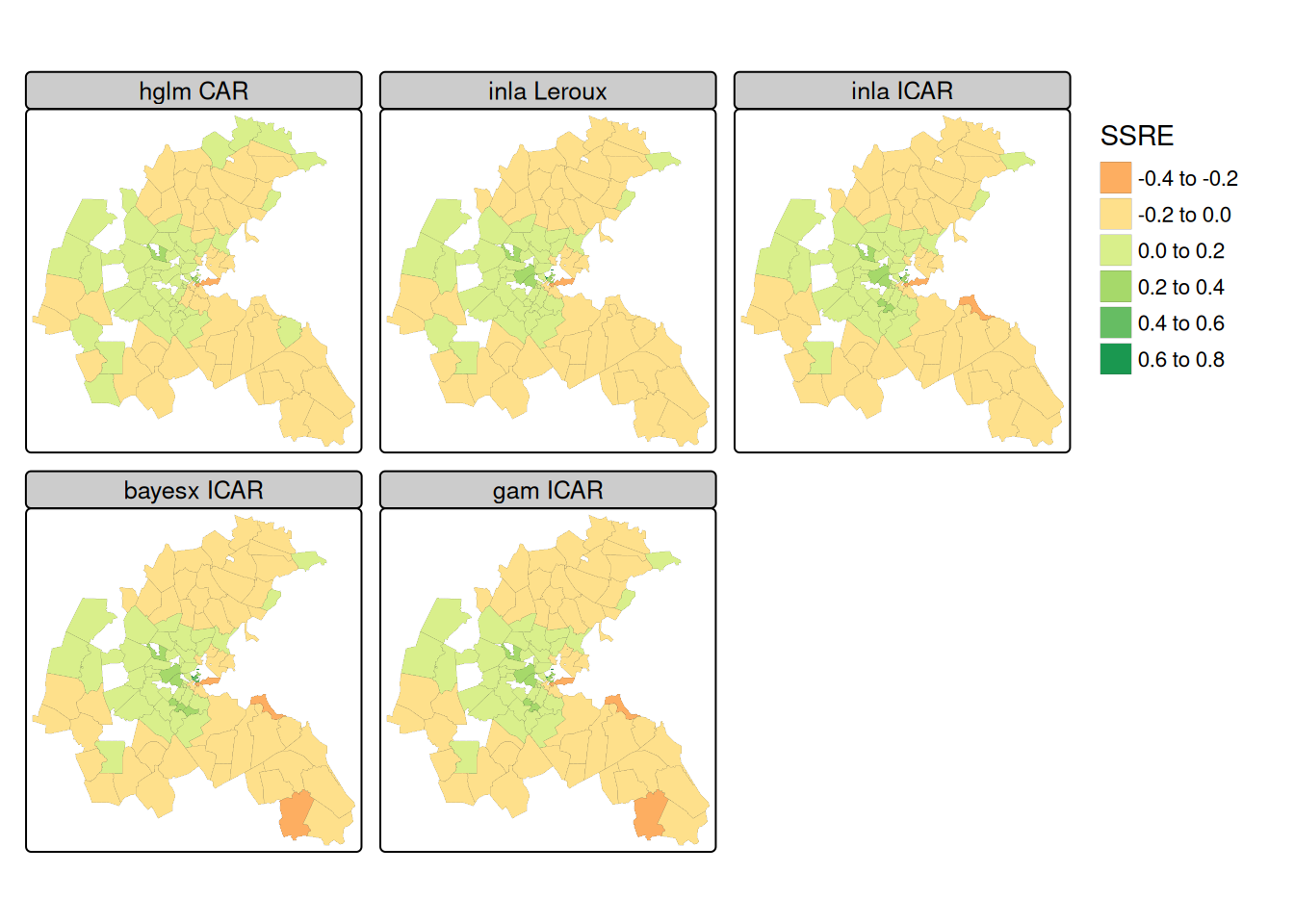
hglm, bayesx, inla, 그리고 gam의 경우, 이산적 반응 변수를 모델링하는 것 역시 가능하다. bayesx, inla, gam의 경우는 해당 공변량에 대한 기능 형태 적합의 일반화를 용이하게 한다.

안타깝게도 이 다수준 모델들이 추정한 공기 질 변수의 회귀계수는 별로 도움이 되지 않는다. 모두 음의 값을 보인 것은 예상대로이지만, 모델 출력 구역 수준 효과와, IID 또는 공간적으로 구조화된 효과가 포함되면서 관찰 스케일의 영향을 해당 스케일(상위 스케일)에서의 공변수의 영향으로부터 분리해 내는 것이 어렵게 되었다.

Figure 16.1은 공기 질 모델 출력 구역 수준의 IID 랜덤 효과가 네 가지 모델 적합 함수 모두에서 매우 유사하다는 것을 보여준다. 모든 지도에서 중앙 도심 구역은 강한 음의 랜덤 효과 값을 보인다. 그런데 중앙 도심 근방에서는 강한 양의 값이 나타나고, 교외 지역은 제로에 가까운 값이 나타난다.

|  |
| --- |
| 그림 16.1: 공기 질 모델 출력 구역 수준의 IID 랜덤 효과(**lme4**, **hglm**, **INLA** 및 **R2BayesX**를 사용하여 추정). 응답(log(median))의 범위는 2.1893이다. |

Figure 16.2는 공간적으로 구조화된 랜덤 효과가 서로 매우 유사하며, “SAR” 공간적 평활도가 랜덤 효과 항이 취하는 값의 범위를 고려할 때 “CAR” 평활도보다 약간 더 부드럽다는 것을 보여준다.



공기 질 모델 출력 구역 수준의 공간적으로 구조화된 랜덤 효과(**lme4**, **hglm**, **INLA**, **R2BayesX** 및 **mgcv**를 사용하여 추정).

다수준 데이터를 다룰 수 있는 공간 회귀 모델 적합 함수에 대한 보다 철저한 비교 연구가 여전히 필요하지만, 최근 몇 년간 상당한 진전이 있었다. Vranckx, Neyens, and Faes (2019) 질병 매핑 공간 회귀에 대한 비교 연구를 진행했는데, 주로 기대 빈도를 오프셋으로 설정한 포아송 회귀 프레임워크에 초점을 맞춘 것이다. Bivand and Gómez-Rubio (2021)는 공간 가중치 행렬을 사용하여 공간 생존 모델을 추정하는 방법과 공간 프로빗 모델 간의 비교를 다룬다.

## 16.3 연습문제

1. **HSAR** 패키지(<https://cran.r-project.org/src/contrib/Archive/HSAR/HSAR_0.5.1.tar.gz>)의 아테네 주택 데이터(**spData** 패키지 2.2.1 버전에 포함되어 있음)를 이용하여 다수준 데이터셋을 생성하라. 각 자치구 부서의 속성 값이 각 자치구 부서 내에 위치한 모든 포인트 관측 값에 복사되는 시점은 언제인가?
2. 상위 수준과 하위 수준 모두에서 이웃 객체를 생성하고, 상위 수준과 하위 수준 모두에서 greensp의 공간적 자기상관을 테스트하라. 자치구 부서의 녹지 면적(제곱 미터)을 포인트 서포트를 가지는 부동산 수준으로 복사한 것의 주요 결과는 무엇인가?
3. 위의 공식 객체를 사용하여 상위 수준의 변수를 추가하는 것이 합리적인지 평가하라. **mgcv** 패키지의 gam 함수를 사용하여 선형 혼합 효과 모델을 적합하라(자치구 부서에 대한 식별 변수인 num\_dep를 사용하여 IID 지정). 하위 수준 변수만 사용한 모델과 하위 및 상위 수준 변수를 모두 사용한 모델을 비교하라. 결론이 달라지는가?
4. IID 랜덤 효과를 "mrf" 마르코프 랜덤 필드와 위에서 생성한 연접 이웃 객체로 대체함으로써 분석을 완수하시오. 자치구 부서 수준 변수, 예를 들어 greensp와 같은 변수에 근거하여 결론을 내는 것이 합리적인지에 대해 견해를 밝히시오.

# 17. 공간계량경제학적 모델

공간 가중치 행렬을 활용하고 최대우도추정법을 적용하는 전형적인 형태의 공간적 자기회귀 모델은 이미 오래전에 정식화된 바 있다(Cliff and Ord 1973, 1981). 이러한 초기 연구를 확장하여 공간 계량경제학적 관점에서 일단의 모델이 발전해 왔으며, 많은 경우 동시적 자기회귀 프레임워크와 행 표준화 공간 가중치 행렬이 사용되었다(Anselin 1988). 동시적 자기회귀 프레임워크와 조건부 자기회귀 프레임워크가 두 개의 축을 이루고 있으며, 두 프레임워크 모두 보완적 차원에서 관측치의 상대적 중요성을 반영하기 위한 사례 가중치를 활용할 수 있다(Waller and Gotway 2004).

공간 회귀 모델을 적합시키는 과정에서 제기되는 이슈들을 논의하기에 앞서, 몇 가지 사항을 언급하고자 한다. 공간적 회귀분석을 공간계량경제학적 맥락에서의 접근하는 방식에 대한 최근 리뷰는 Kelejian과 Piras(2017)에 의해 제공되었다. 이 연구에서는 공간 지체 반응 변수의 계수를 , 공간 지체 잔차의 계수를 로 명명하고 있는데, 이는 다른 연구들(Anselin 1988; LeSage and Pace 2009)과는 거꾸로 이다. 여기서는 공간 지체 모델의 공간 계수는 로, 공간 오차 모델의 공간 계수는 로 표현하고자 한다. 흥미로운 발견 중 하나는 상대적으로 밀도가 높은 공간 가중치 행렬이 모델 추정치를 과소평가할 가능성이 있다는 점인데, 이는 희소 가중치 행렬이 더 적합하다는 점을 시사하는 것이다(Smith 2009). 또 다른 유용한 발견은 잔차에 공간적 자기상관이 존재하더라도, 공변량 자체가 공간적 자기상관을 나타내지 않는 한 편향된 회귀 계수 분산의 추정치가 산출되지는 않는다는 점이다(Smith and Lee 2012). 그러나 일반적으로 반응 변수와 공변량의 공간 과정이 일치하지 않을 가능성이 있으며, 공변량과 잔차가 공간적 자기상관을 보이는 경우 공간 과정을 모델링하려는 시도가 이루어지지 않는 한 회귀 계수 분산의 추정치가 과소평가될 가능성이 높다.

## 17.1 정의

공간 과정을 모델링하려는 시도 중 가장 초기의 공간계량경제학적 정식 중 하나는 잔차의 공간 자기상관을 모델링하는 것이다(공간 오차 모델, SEM):

여기서 는 위치에서 측정된 반응 변수로 관측치 벡터, 는 공변량 행렬, 는 계수 벡터, 는 공간적 자기상관을 가진 교란 벡터, 은 독립항등분포를 가진 교란 벡터, 은 스칼라 공간 파라미터이다.

이 모델과 유사한 다른 공간계량경제학 모델들은 혼합 모델 프레임워크에 부합하지 않는다. 여기서 모델링된 공간 과정은 반응 변수, 공변량, 그리고 계수와 직접적으로 상호작용한다. 이러한 종류의 모델링 프레임워크는 시계열 프레임워크를 단순히 2차원으로 확장한 오래된 전통에서 기원한 것으로 보인다.

공변량과 반응 변수에서의 과정이 일치한다면, 최소제곱법 계수와 SEM 계수 간의 차이는 거의 없을 것이다. 그러나 실제로는 두 개의 과정이 일치하지 않으며, 이를 고려하기 위해서는 Hausman 검정과 같은 부가적인 조치가 필요하다(Pace and LeSage 2008). 이는 공간 과정이 일치하는 경우 시계열 분석에서의 단위근과 공적분 개념을 공간적으로 확장하는 이전 논의와 관련이 있을 수 있다(Fingleton 1999).

반응 변수에만 공간 과정을 포함하는 모델은 공간 지체 모델(SLM, 종종 SAR - 공간적 자기회귀 모델)이라고 한다(LeSage and Pace 2009). Durbin 모델은 공변량에 공간 지체 공변량을 추가한 모델이다. 공간 Durbin 모델에 대한 리뷰는 Mur와 Angulo(2006)를 살펴보면 된다. 만약 반응 변수에 공간 과정 외에도 잔차에 공간 과정을 포함시키기로 선택한다면, 다시 두 가지 모델이 형성된다. 하나는 모든 모델을 포함하는 일반적인 중첩 모델(GNM, general nested model)이고, 다른 하나는 공간 지체 공변량을 포함하지 않는 모델(SAC, 또한 SARAR(Spatial AutoRegressive-AutoRegressive) 모델)이다. 만약 잔차나 반응 변수에 공간 과정을 모델링하지 않는다면, 공간 지체 공변량을 선형 모델에 추가할 수 있다. 이를 공간 지체 X 모델(SLX)이라고 한다(Elhorst 2010; Bivand 2012; LeSage 2014; Halleck Vega and Elhorst 2015). GNM은 다음과 같이 표현할 수 있다:

여기서 는 파라미터 벡터이다. 는 절편과 공변량의 일부를 정의하는데, 행표준화 공간 가중치를 사용하고 공간 지체 절편을 누락하는 경우 이다.

이 기본 모델에 제약을 가하면 다양한 변용 모델이 생성되는데, 으로 설정하면 이중 공간 계수 모델인 SAC/SARAR 모델이 되고, 으로 설정하면 오차 Durbin 모델(SDEM)이 된다. 더 많은 조건을 부여할 수 있는데, 와 로 설정하면 공간 지체 모델(SLM), 와 로 설정하며 공간 오차 모델(SEM), 과 로 설정하면 공간 지체 X 모델(SLX)을 얻을 수 있다.

공변량이 관측된 새로운 위치에 대해 예측을 행하는 것과 관련된 이슈들은 오래전에 인식되었지만, 가능성에 대한 검토에서 진전을 이루는 데는 많은 시간이 걸렸다(Bivand 2002; Goulard, Laurent, and Thomas-Agnan 2017; Laurent and Margaretic 2021). 최대우도추정법으로 적합된 SLM, SDM, SEM, SDEM, SAC, GNM 모델을 예측에 적용하는 방법은 Google Summer of Coding 프로젝트를 수행한 Martin Gubri의 공로가 크다. 이 프로젝트와 결측 데이터를 다루는 유사한 모델에 대한 연구(Suesse 2018)를 원용하면 보스턴 데이터셋에서 검열된 중앙값 주택 가격을 탐색하는 연구에 시사점을 얻을 수 있다. 예측 이슈를 다루다 보면 해당 모델의 축약 형태가 매우 중요하다는 점을 인식하게 되는데, SLM, SDM, SAC, GNM 모델에서처럼 반응 변수의 공간 과정이 회귀 계수와 상호작용하는 경우 보다 단순화된 형태가 도움이 될 수 있다.

반응 변수와 회귀 계수 간의 이러한 상호작용의 결과는, 공변량의 단위 변화가 회귀 계수의 값으로 반응 변수에 영향을 미치려면 공간 지체 반응 변수의 계수가 0이어야 한다는 것이다. 만약 공간 계수가 0이 아니라면, 글로벌 파급효과가 작용하게 되며, 이러한 파급효과의 크기는 회귀 계수 만큼이나 중요한 정보이다(LeSage and Pace 2009; Elhorst 2010; Bivand 2012; LeSage 2014; Halleck Vega and Elhorst 2015). SDEM 및 SLX 모델의 경우, 각 공변량의 총 파급효과(공변량 및 공간 지체 공변량 계수의 합)의 표준 오차를 계산하기 위해 선형 결합을 사용하여 로컬 파급효과를 나타낸다.

이것은 GNM 데이터 생성 과정에서 확인할 수 있다:

$$
(\textbf{I}-\rho\_{\text Err}\textbf{W})(\textbf{I}-\rho\_{\text Lag}\textbf{W})\textbf{y}=(\textbf{I}-\rho\_{\text Err}\textbf{W})(\textbf{X}\beta+\textbf{WX}\gamma)+\epsilon
$$

다음과 같이 다르게 나타낼 수 있다.

와 간의 상호작용(가 존재할 경우, 셋 간의 상호작용)이 존재한다. 이것은 다음의 편도함수로부터 확인할 수 있다. . 여기서 밀집 행렬 의 주대각 요소는 직접적인 영향(효과)을 나타내고, 비대각 요소는 간접적인 영향을 나타낸다.

Piras와 Prucha (2014)는 Raymond J. G. M. Florax, Folmer, Rey (2003)를 재검토하고 수정하였다(또한 Hendry (2006)와 Raymond J. G. M. Florax, Folmer, Rey (2006)의 논평 참조). 이들은 모델 선택을 위해 통상적으로 사용되는 사전 검정(pre-test) 전략보다는 해당 모델링에 적합한 가장 일반적인 모델을 추정하는 전략이 더 낳다고 결론지었다. 이 발견에 비추어, 여기서는 사전 검정 모델 선택을 사용하지 않는다.

현재 **spatialreg** 패키지가 주의를 기울이고 있는 것은 공간 지체 공변량을 처리하는 방법을 개선하는 것인데, Durbin 아규먼트를 사용하여 공간 지체 형식으로 추가할 공변량의 하위 집합을 제공하는 논리 값이나 수식을 받는다. 일부 공변량, 예를 들어 더미 변수와 같은 것들은 공간 지체 형식으로 추가되지 않아야 한다는 주장이 있다. 일단 공간 지체 공변량이 선택되면 다음의 이슈는 영향력을 계산할 때 이를 어떻게 적절하게 다룰지와 관련된 다. 이렇게 개선된 기능은 MCMC 또는 최대우도추정법으로 적합된 횡단면 모델에 적용되며, 공간 패널 모델에 적용될 예정이다.

거의 다루어지지 않은 이슈인 기능적 형태 가정(functional form assumption)을 언급하는 것은 충분한 가치가 있다. 이 이슈는 예를 들어 **McSpatial** 패키지에서 제시된 공간 분위수 회귀(McMillen 2013)와 같은 유연한 구조가 유용한 상황과 관련되어 있다. 또한 **McSpatial** 패키지의 일부 함수, 새로운 패키지 **spldv**(Sarrias and Piras 2022), 그리고 **spatialprobit** 패키지 및 **ProbitSpatial** 패키지(Wilhelm and Matos 2013; Martinetti and Geniaux 2017)에서 다루는 이산 반응 변수와 관련된 이슈들도 있다. **McSpatial** 패키지의 MCMC 실행은 LeSage와 Pace(2009)를 기반으로 한다. 마지막으로, Wagner와 Zeileis(2019)는 SLM 모델이 재귀적 분할 설정에서 어떻게 사용될 수 있는지 보여주며, 이를 **spatialreg** 패키지의 lagsarlm() 함수를 사용한 **lagsarlmtree** 패키지에서 구현하는 방법을 제시한다.

**spatialreg** 패키지(Bivand and Piras 2022)를 통해 횡단(cross-sectional) 최대우도추정과 일반화된 모멘트법(GMM) 추정을 행하는 것과 **sphet** 패키지를 통해 공간계량경제학 스타일의 공간 회귀 모델을 행하는 것에 대한 리뷰(Bivand and Piras 2015)는 여전히 대부분 유효하다. 이 리뷰에서는 R 패키지에서의 추정량을 다른 프로그래밍 언어에서 제공되는 대체 구현과 비교하였다. 그러나 베이지안 공간계량경제학 스타일의 공간 회귀는 다루지 않았다. Millo와 Piras (2012)에서 설명한 공간 패널 추정량에 대해서는 더 많은 변화가 있었지만, 여기서는 다루지 않는다.

Bivand, Millo, and Piras (2021)는 공간계량경제학적 분석을 위한 R 패키지의 많은 기능을 다루고 있으며, Bivand와 Piras (2015)를 업데이트하고 일반화된 모멘트법(GMM) 및 공간 패널 모델링의 최근 발전을 포함하고 있기 때문에, 이 장에서는 Boston 주택 가격 데이터셋을 사용한 Bivand (2017)의 예제 몇 가지만 제한적으로 다룰 것이다.

## 17.2 최대우도추정: spatialreg 패키지의 활용

단일 공간 계수를 가진 모델(SEM 및 SDEM은 errorsarlm() 함수 사용, SLM 및 SDM은 lagsarlm() 함수 사용)에는 Ord (1975)가 처음 설명한 방법이 사용된다. 다음 표는 최대우도추정법을 사용하여 위에서 설명한 모델을 추정할 수 있는 함수들을 보여준다.

| 모델 | 모델명 | 최대우도추정 함수 |
| --- | --- | --- |
| SEM | 공간에러 | errorsarlm(..., Durbin=FALSE) |
| SEM | 공간에러 | spautolm(..., family="SAR") |
| SDEM | 공간더빈에러 | errorsarlm(..., Durbin=TRUE) |
| SLM | 공간지체 | lagsarlm(..., Durbin=FALSE) |
| SDM | 공간더빈 | lagsarlm(..., Durbin=TRUE) |
| SAC | 공간자기회귀결합 | sacsarlm(..., Durbin=FALSE) |
| GNM | 일반 포함 | sacsarlm(..., Durbin=TRUE) |

추정 함수인 errorsarlm()과 lagsarlm()은 비슷한 인자를 받는데, 첫 번째와 두 번째 인가(formula와 data)는 대부분의 모델 추정 함수에서 공통적으로 사용된다. 세 번째 인자는 listw 공간 가중치 객체이고, na.action은 결측값을 가진 관측개체의 공간 가중치를 처리하는 방법을 정의하는데 다른 모델 추정 함수와 동일하게 동작한다. weights 아규먼트는 분산 항목에서 관측값별 변동성을 나타내는 가중치를 제공하는 데 사용된다. 단, lagsarlm() 함수에서는 이 옵션이 제공되지 않는다.

Durbin 아규먼트는 이전의 type과 etype 인자를 대체하며, 주어지지 않으면 기본값은 FALSE로 설정된다. 만약 제공된다면, FALSE나 TRUE일 수 있으며, 이 경우 모든 공간 지체 설명 변수가 포함된다. 또는, 일면 공식(one-sided formula)를 통해 어떤 공간 지체 설명 변수가 포함될지를 지정할 수 있다. method 아규먼트는 로그 우도 함수에서 로그 행렬식 항을 계산하는 방법을 지정하며, 기본값은 "eigen"으로, 이는 중간 크기의 데이터셋에 적합하다. interval 아규먼트는 **stats** 패키지의 optimize() 함수를 통해 공간 계수를 찾는 선형 탐색의 도메인 범위를 제공한다. tol.solve() 아규먼트는 **base** 패키지의 solve() 함수에 전달되며, 이는 회귀 계수 간에 수치적 스케일 차이가 있는 데이터셋에서 분산-공분산 행렬의 역행렬을 계산하는 데 필요하다. 디폴트 값은 base::solve()에서 훨씬 더 큰 값이다. control 아규먼트는 추정 함수 실행을 더 세밀하게 조정할 수 있게 해주는 제어 값의 리스트를 받는다.

sacsarlm() 함수는 SAC 및 GNM 형식에서 두 개의 공간적 프로세스를 모델링할 때 서로 다른 두 개의 공간 가중치를 사용하는 경우, 두 번째 공간 가중치 및 interval 아규먼트를 받을 수 있다. 디폴트는 동일한 공간 가중치를 사용하는 것이다. 기본적으로는 **stats** 패키지의 nlminb() 함수가 수치 최적화에 사용되며, 시작 값을 선택하는 데 휴리스틱이 사용된다. lagsarlm()함수와 마찬가지로, 이 함수는 weights 아규먼트를 받지 않는다.

대용량 데이터셋이 사용되는 경우, 계수의 분산-공분산 행렬을 계산하기 위해 분석적 점근적(analytic asymptotic) 접근 방식 대신 수치적 헤시안(Hessian) 접근 방식이 사용된다.

### 17.2.1 보스턴 주택 가격 데이터셋 예시

예제는 Bivand (2017)를 기반으로 16장에서 생성된 객체들을 사용한다.

library(spatialreg)  
eigs\_489 <- eigenw(lw\_q\_489)  
SDEM\_489 <- errorsarlm(form, data = boston\_489,   
 listw = lw\_q\_489, Durbin = TRUE, zero.policy = TRUE,  
 control = list(pre\_eig = eigs\_489))  
SEM\_489 <- errorsarlm(form, data = boston\_489,   
 listw = lw\_q\_489, zero.policy = TRUE,  
 control = list(pre\_eig = eigs\_489))

디폴트인 "eigen" 방법을 사용하기 위해 미리 계산된 고유값을 control 리스트 아규먼트를 사용하여 전달한다.

cbind(data.frame(model=c("SEM", "SDEM")),   
 rbind(broom::tidy(Hausman.test(SEM\_489)),   
 broom::tidy(Hausman.test(SDEM\_489))))[,1:4]  
# model statistic p.value parameter  
# 1 SEM 52.0 2.83e-06 14  
# 2 SDEM 48.7 6.48e-03 27

489개의 트랙트 데이터셋에 대한 두 가지 Hausman 검정 결과는 회귀 계수가 비공간적 회귀분석의 회귀계수와는 유의미하게 다르다는 것을 시사하며, 이는 공간 프로세스의 발자국이 일치하지 않음을 함축한다.

eigs\_94 <- eigenw(lw\_q\_94)  
SDEM\_94 <- errorsarlm(form, data=boston\_94, listw=lw\_q\_94,  
 Durbin = TRUE,  
 control = list(pre\_eig=eigs\_94))  
# Warning in RET$pfunction("adjusted", ...): Completion with error >  
# abseps  
  
# Warning in RET$pfunction("adjusted", ...): Completion with error >  
# abseps  
  
# Warning in RET$pfunction("adjusted", ...): Completion with error >  
# abseps  
  
# Warning in RET$pfunction("adjusted", ...): Completion with error >  
# abseps  
SEM\_94 <- errorsarlm(form, data = boston\_94, listw = lw\_q\_94,  
 control = list(pre\_eig = eigs\_94))

94개의 대기 오염 모델 출력 구역에 대한 Hausman 검정은 계수 간 큰 차이를 발견하지 못했다.

cbind(data.frame(model=c("SEM", "SDEM")),   
 rbind(broom::tidy(Hausman.test(SEM\_94)),   
 broom::tidy(Hausman.test(SDEM\_94))))[, 1:4]  
# model statistic p.value parameter  
# 1 SEM 15.66 0.335 14  
# 2 SDEM 9.21 0.999 27

이러한 결과는 SEM과 SDEM 모델이 대기 오염 모델 출력 구역 수준에서 최소 제곱법이나 SLX에 거의 추가적인 차이를 만들지 않았다는 점을 보여준다. 우도비 검정을 통해 이러한 사실을 확인할 수 있다.

cbind(data.frame(model=c("SEM", "SDEM")),  
 rbind(broom::tidy(LR1.Sarlm(SEM\_94)),  
 broom::tidy(LR1.Sarlm(SDEM\_94))))[,c(1, 4:6)]  
# model statistic p.value parameter  
# 1 SEM 2.593 0.107 1  
# 2 SDEM 0.216 0.642 1

**spatialreg** 패키지의 LR.Sarlm() 함수를 사용하면 중첩(nested) 모델 간에 우도비 검정을 실행할 수 있지만, 여기서는 **lmtest** 패키지의 lrtest() 함수를 사용하는데 결과는 동일하다. 트랙트와 모델 출력 구역 모두에서 공간 지체 공변량을 포함하는 모델이 보다 선호된다는 결과가 산출된다.

o <- lmtest::lrtest(SEM\_489, SDEM\_489)  
attr(o, "heading")[2] <- "Model 1: SEM\_489\nModel 2: SDEM\_489"  
o  
# Likelihood ratio test  
#   
# Model 1: SEM\_489  
# Model 2: SDEM\_489  
# #Df LogLik Df Chisq Pr(>Chisq)   
# 1 16 274   
# 2 29 311 13 74.4 1.2e-10 \*\*\*  
# ---  
# Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

o <- lmtest::lrtest(SEM\_94, SDEM\_94)  
attr(o, "heading")[2] <- "Model 1: SEM\_94\nModel 2: SDEM\_94"  
o  
# Likelihood ratio test  
#   
# Model 1: SEM\_94  
# Model 2: SDEM\_94  
# #Df LogLik Df Chisq Pr(>Chisq)   
# 1 16 59.7   
# 2 29 81.3 13 43.2 4.2e-05 \*\*\*  
# ---  
# Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

SLX 모델은 최소제곱법을 사용하여 적합되며, 로그 우도 값을 반환하여 잔차에 공간적 과정 요소가 포함될 필요가 있는지를 테스트할 수 있게 해준다. 트랙트 데이터셋에서는 분명히 그렇다는 결과가 도출되었다.

SLX\_489 <- lmSLX(form, data = boston\_489, listw = lw\_q\_489,  
 zero.policy = TRUE)  
o <- lmtest::lrtest(SLX\_489, SDEM\_489)  
attr(o, "heading")[2] <- "Model 1: SLX\_489\nModel 2: SDEM\_489"  
o  
# Likelihood ratio test  
#   
# Model 1: SLX\_489  
# Model 2: SDEM\_489  
# #Df LogLik Df Chisq Pr(>Chisq)   
# 1 28 231   
# 2 29 311 1 159 <2e-16 \*\*\*  
# ---  
# Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

그러나 산출 존 데이터에 대해서는 반대의 결과가 도출되었다.

SLX\_94 <- lmSLX(form, data = boston\_94, listw = lw\_q\_94)  
o <- lmtest::lrtest(SLX\_94, SDEM\_94)  
attr(o, "heading")[2] <- "Model 1: SLX\_94\nModel 2: SDEM\_94"  
o  
# Likelihood ratio test  
#   
# Model 1: SLX\_94  
# Model 2: SDEM\_94  
# #Df LogLik Df Chisq Pr(>Chisq)  
# 1 28 81.2   
# 2 29 81.3 1 0.22 0.64

이러한 결과는 주택 단위 수를 트랙트와 출력 구역별로 집계하여 사례 가중치를 사용할 때도 유지된다.

SLX\_489w <- lmSLX(form, data = boston\_489, listw = lw\_q\_489,  
 weights = units, zero.policy = TRUE)  
SDEM\_489w <- errorsarlm(form, data = boston\_489,  
 listw = lw\_q\_489, Durbin = TRUE,  
 weights = units, zero.policy = TRUE,  
 control = list(pre\_eig = eigs\_489))  
o <- lmtest::lrtest(SLX\_489w, SDEM\_489w)  
attr(o, "heading")[2] <- "Model 1: SLX\_489w\nModel 2: SDEM\_489w"  
o  
# Likelihood ratio test  
#   
# Model 1: SLX\_489w  
# Model 2: SDEM\_489w  
# #Df LogLik Df Chisq Pr(>Chisq)   
# 1 28 311   
# 2 29 379 1 136 <2e-16 \*\*\*  
# ---  
# Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

SLX\_94w <- lmSLX(form, data = boston\_94, listw = lw\_q\_94,  
 weights = units)  
SDEM\_94w <- errorsarlm(form, data = boston\_94, listw = lw\_q\_94,  
 Durbin = TRUE, weights = units,  
 control = list(pre\_eig = eigs\_94))  
o <- lmtest::lrtest(SLX\_94w, SDEM\_94w)  
attr(o, "heading")[2] <- "Model 1: SLX\_94w\nModel 2: SDEM\_94w"  
o  
# Likelihood ratio test  
#   
# Model 1: SLX\_94w  
# Model 2: SDEM\_94w  
# #Df LogLik Df Chisq Pr(>Chisq)  
# 1 28 97.5   
# 2 29 98.0 1 0.92 0.34

이 경우와 Bivand (2017)에서 제시된 논거를 바탕으로 가중치 사용은 정당화된다. 주택 단위 수가 트랙트의 경우는 5개에서 3,031개로 다양하고, 대기 오렴 모델 출력 구역의 경우는 25개에서 12,411개로 다양하기 때문이다. 이러한 이유로, 그리고 가중치 일반화 중첩 모델(GNM)이 개발되지 않았다는 이유로, GNM을 일반적인 모델에서 더 단순한 모델로 테스트하는 출발점으로 삼을 수 없다. 대신, SDEM 모델을 출발점으로 삼고 Hausman 테스트를 사용하여 관측 단위 선택의 지침을 구한다.

## 17.3 영향

글로벌 영향은 공간 지체 반응 변수를 포함하는 모델(SLM, SDM, SAC, GNM)의 적합 결과를 보고하는 데 있어 핵심적인 사항으로 이해되고 있다(LeSage와 Pace 2009). 글로벌 영향을 공간 지체 공변량(SLX, SDEM)을 포함하는 다른 모델로 확장하는 시도도 이루어졌다(Elhorst 2010; Bivand 2012; Halleck Vega와 Elhorst 2015). SLM, SDM, SAC, GNM 모델은 최대 우도 또는 GMM을 사용하여 적합시킬 때 계수의 분산-공분산 행렬을 사용할 수 있으며, 계수 값을 평균으로 삼고 추정된 분산-공분산 행렬로부터 분산을 가져와 다변량 정규 분포를 설정하고 그것으로부터 랜덤 샘플을 추출할 수 있다. 베이지안 방법을 사용하여 적합하는 경우라면, 샘플은 사전에 주어진다. SDEM의 경우, 비지체 공변량의 회귀 계수에 대한 샘플은 직접적인 영향을 나타내고, 공간 지체 공변량의 계수에 대한 샘플은 간접적인 영향을 나타내며, 이들의 합은 총 영향을 나타낸다.

SLX와 SDEM 모델에서는 추론을 위해 샘플링이 필요하지 않으므로 최대 우도를 사용하여 적합한 모델에 대해 선형 결합을 사용한다. 여기서는 공기 오염 변수에 대해서만 결과를 보여준다. 모델 출력 보고 방법에 대한 문제를 완전히 해결한 연구는 아직 발간되지 않았다. 개별 공변량에 대해 세 가지 영향력 값을 보고해야 하기 때문이다. 공간 지체 공변량이 포함될 경우, 두 개의 계수는 세 가지 영향으로 대체된다. 여기서는 공기 오염 변수에 대해 그 예를 보여준다.

sum\_imp\_94\_SDEM <- summary(impacts(SDEM\_94))  
rbind(Impacts = sum\_imp\_94\_SDEM$mat[5,],   
 SE = sum\_imp\_94\_SDEM$semat[5,])  
# Direct Indirect Total  
# Impacts -0.01276 -0.01845 -0.0312  
# SE 0.00235 0.00472 0.0053

SLX와 SDEM 모델에서 직접적인 영향은 같은 관측 단위에서 공기 오염 변화가 응답에 미치는 결과이고, 간접적인(지역적인) 영향은 인접한 관측 단위에서 공기 오염 변화가 응답에 미치는 결과이다.

sum\_imp\_94\_SLX <- summary(impacts(SLX\_94))  
rbind(Impacts = sum\_imp\_94\_SLX$mat[5,],   
 SE = sum\_imp\_94\_SLX$semat[5,])  
# Direct Indirect Total  
# Impacts -0.0128 -0.01874 -0.03151  
# SE 0.0028 0.00556 0.00611

같은 방법을 가중 공간 회귀 분석에 적용하면, 공기 오염이 주택 가치에 미치는 총 영향은 감소하지만 여전히 유의미하다.

sum\_imp\_94\_SDEMw <- summary(impacts(SDEM\_94w))  
rbind(Impacts = sum\_imp\_94\_SDEMw$mat[5,],   
 SE = sum\_imp\_94\_SDEMw$semat[5,])  
# Direct Indirect Total  
# Impacts -0.00592 -0.01076 -0.01668  
# SE 0.00269 0.00531 0.00559

전체적으로, 공기 오염 모델 출력 구역 수준으로 집계된 공간 지체 공변량만 포함하는 가중 공간 회귀 모델이 대부분의 모델 오지정 오류를 해결하는 것처럼 보이며, Bivand (2017)에서 더 자세히 논의되어 있는 바와 같이, 이는 오지정된 대체 모델들보다 훨씬 더 큰 오염 제거에 대한 지불 의향이 있다는 점을 보여준다.

sum\_imp\_94\_SLXw <- summary(impacts(SLX\_94w))  
rbind(Impacts = sum\_imp\_94\_SLXw$mat[5,],   
 SE = sum\_imp\_94\_SLXw$semat[5,])  
# Direct Indirect Total  
# Impacts -0.00620 -0.01221 -0.01842  
# SE 0.00326 0.00628 0.00629

## 17.4 예측

보스턴 구역 데이터셋에서 응답 변수인 중위 주택 가격 17개의 관측치가 제한되어 있다. 우리는 "Sarlm" 객체에 대한 predict() 메서드를 사용하여 이 값을 채울 것이다. 이 메서드는 Goulard, Laurent, and Thomas-Agnan (2017)을 기반으로 Martin Gubri에 의해 다시 작성되었으며, Laurent와 Margaretic (2021)도 참조할 수 있다. pred.type 아규먼트는 해당 논문에서 제시된 예측 전략을 지정한다.

이 예시를 SDEM 모델의 다양한 변종을 pred.type의 설정을 통해 시도하고 샘플 외 예측을 수행해 보면 차이가 있음을 알 수 있으며, 이는 연구할 만한 중요한 영역임을 시사한다. 누락 변수 처리에 대한 몇 가지 대안이 제시된바 있다(Gómez-Rubio, Bivand, and Rue 2015; Suesse 2018). 예측에 대한 관심이 증가하는 또 다른 이유는 예측이 기계 학습 접근 방식의 기본 요소라는 점이다. 기계 학습에서는 검증 및 테스트 데이터셋을 위한 예측이 모델 사양 선택을 이끈다. 의존성을 가진 공간 데이터로부터 훈련 및 기타 데이터셋을 선택하는 것은 여전히 해결되지 않은 문제로, 독립적인 데이터와는 분명히 다른 복잡성을 지니고 있다.

여기에서는 검열(censored) 트랙트 관측값에 대해 세 가지 서로 다른 예측 유형을 사용하여 예측값을 산출하고, USD 단위의 주택 중위값으로 복원하기 위해 지수 연산을 적용한다. newdata 객체의 row.names()가 공간 가중치 행렬의 "region.id" 속성과 일치하여 샘플 외 예측이 가능해진다는 점에 유의해야 한다.

nd <- boston\_506[is.na(boston\_506$median),]  
t0 <- exp(predict(SDEM\_489, newdata = nd, listw = lw\_q,  
 pred.type = "TS", zero.policy =TRUE))  
suppressWarnings(t1 <- exp(predict(SDEM\_489, newdata = nd,  
 listw = lw\_q,  
 pred.type = "KP2",  
 zero.policy = TRUE)))  
suppressWarnings(t2 <- exp(predict(SDEM\_489, newdata = nd,  
 listw = lw\_q,  
 pred.type = "KP5",  
 zero.policy = TRUE)))

우리는 INLA의 "slm" 모델을 사용하여 모델 적합 함수 호출 과정에서 누락된 반응 변수 값을 예측할 수도 있다. "slm" 모델이 아직 실험적인 상태이므로 약간의 설정 코드가 필요하다.

library(INLA)  
# Loading required package: foreach  
# Loading required package: parallel  
# Loading required package: sp  
# This is INLA\_23.04.24 built 2023-04-24 19:15:35 UTC.  
# - See www.r-inla.org/contact-us for how to get help.  
# - To enable PARDISO sparse library; see inla.pardiso()  
#   
# Attaching package: 'INLA'  
# The following object is masked \_by\_ '.GlobalEnv':  
#   
# f  
W <- as(lw\_q, "CsparseMatrix")  
n <- nrow(W)  
e <- eigenw(lw\_q)  
re.idx <- which(abs(Im(e)) < 1e-6)  
rho.max <- 1 / max(Re(e[re.idx]))  
rho.min <- 1 / min(Re(e[re.idx]))  
rho <- mean(c(rho.min, rho.max))  
boston\_506$idx <- 1:n  
zero.variance = list(prec = list(initial = 25, fixed = TRUE))  
args.slm <- list(rho.min = rho.min, rho.max = rho.max, W = W,  
 X = matrix(0, n, 0), Q.beta = matrix(1,0,0))  
hyper.slm <- list(prec = list(prior = "loggamma",   
 param = c(0.01, 0.01)),  
 rho = list(initial = 0, prior = "logitbeta",  
 param = c(1,1)))  
WX <- create\_WX(model.matrix(update(form, CMEDV ~ .),   
 data = boston\_506), lw\_q)  
SDEM\_506\_slm <- inla(update(form,   
 . ~ . + WX + f(idx, model = "slm",  
 args.slm = args.slm,  
 hyper = hyper.slm)),  
 data = boston\_506, family = "gaussian",  
 control.family = list(hyper = zero.variance),  
 control.compute = list(dic = TRUE, cpo = TRUE))  
mv\_mean <- exp(SDEM\_506\_slm$summary.fitted.values$mean[  
 which(is.na(boston\_506$median))])

INLA는 또한 예측값에 대한 주변 분포(marginal distribution)의 격자형 추정치를 제공하며, 이를 통해 예측값과 관련된 불확실성을 평가할 수 있다.

data.frame(fit\_TS = t0[,1], fit\_KP2 = c(t1), fit\_KP5 = c(t2),  
 INLA\_slm = mv\_mean, censored =   
 boston\_506$censored[as.integer(attr(t0, "region.id"))])  
# fit\_TS fit\_KP2 fit\_KP5 INLA\_slm censored  
# 13 23912 29477 28147 31112 right  
# 14 28126 27001 28516 31314 right  
# 15 30553 36184 32476 41298 right  
# 17 18518 19621 18878 21160 right  
# 43 9564 6817 7561 6830 left  
# 50 8371 7196 7383 6885 left  
# 312 51477 53301 54173 56274 right  
# 313 45921 45823 47095 46447 right  
# 314 44196 44586 45361 42805 right  
# 317 43427 45707 45442 48025 right  
# 337 39879 42072 41127 41462 right  
# 346 44708 46694 46108 45847 right  
# 355 48188 49068 48911 49138 right  
# 376 42881 45883 44966 47747 right  
# 408 44294 44615 45670 46164 right  
# 418 38211 43375 41914 43913 right  
# 434 41647 41690 42398 41551 right

공간 회귀를 위한 툴박스는 여전히 완전하지 않았으며, 빈틈을 채우는 데는 시간이 걸릴 것이다. 여러 공간 회귀 전통들이 서로의 이해와 발전을 거의 공유하지 않는 점은 여전히 아쉬운 부분이다.

## 17.5 연습문제

1. Piras와 Prucha(2014), 그리고 Raymond J. G. M. Florax, Folmer, Rey(2003)를 참조할 때, 사전 테스트 전략을 선택하면, 속성만 포함된 데이터셋과 지자체 구역 변수가 추가된 속성 데이터셋의 선형 모델에서 잔차의 공간적 의존성이 나타나는지 답하시오. 사전 테스트가 어떤 모델을 지목하는지에 대해 답하시오.
2. 지자체 부서 더미 변수의 포함하거나 지자체 부서 체제 모델을 실행하면 잔차의 공간적 의존성을 줄이는 데 도움이 될 수 있을지에 대해 답하시오.
3. 속성만 포함한 모델과 지자체 부서 변수를 추가한 모델에 대해 최대우도법을 사용하여 SEM 모델을 실행하고(GMM 코드 예시는 Bivand, Millo, and Piras (2021)를 참조), 이를 SDEM 모델로 확장하시오. SLX 모델을 실행하고, 해당 모델의 잔차에 대한 공간적 자기상관 테스트 결과를 해석하시오. SEM과 SDEM 모델에 대한 Hausman 테스트가 보여주는 매우 높은 유의성을 해석하시오.
4. 속성만 포함한 모델과 지자체 부서 변수를 추가한 모델에 대해 GNM 모델을 실행하시오. 이러한 모델을 SDM 또는 SDEM 형식으로 단순화할 수 있을지에 대해 답하시오.

# Appendix A — 오래된 공간 R 패키지

## A.1 rgdal과 rgeos 패키지의 퇴역

당신이 오랜 R 사용자, 특히 **sf**와 **stars** 같은 패키지가 개발되기 이전부터 R을 사용해 왔다면, **maptools**, **sp**, **rgeos**, **rgdal** 같은 오래된 패키지에 더 익숙할 수 있다. 기존 코드나 오래된 패키지에 의존하는 다른 R 패키지를 새롭게 갱신해야할 필요가 있는지를 궁금해하는 것은 당연한다. 그에 대한 답은 ’그렇다’이다.

**maptools**, **rgdal**, **rgeos** 패키지는 2023년 중에 퇴역할 예정이다. 퇴역의 의미는 유지보수가 중단되고, 그 결과로 해당 패키지들이 CRAN에서 아카이브로 전환된다는 것이다. 하지만 소스 코드 저장소는 R-Forge가 유지되는 한 계속 남아 있을 것이다. 퇴역의 한 가지 이유는 관리자가 은퇴했기 때문이고, 더 중요한 이유는 이들 패키지의 역할이 더 새로운 패키지들로 대체되었기 때문이다. 새로운 관리자가 R-Forge 저장소를 인계받을 가능성은 매우 낮아 보인다. 이는 부분적으로 GEOS, GDAL, PROJ 라이브러리의 발전과 함께 이들 패키지의 코드가 점진적으로 변화해 왔으며, 많은 코드가 시대에 뒤떨어진 구조를 포함하고 있어 읽기 어렵기 때문이다.

**rgeos**와 **rgdal**이 퇴역하기 전에, **sp** 패키지가 **rgdal** 및 **rgeos**와 가지고 있는 기존 연결성은 **sf** 패키지와의 연결성으로 대체될 수 있으며, 실제로 대체될 예정이다. 이는 예를 들어 좌표참조계(CRS) 식별자의 검증, 링이 내부 홀인지 외부 링인지 확인하는 작업 등을 포함한다. **maptools**의 일부 선택된 함수는 **sp**로 이전될 가능성도 있다.

## A.2 sf와 sp 패키지의 연결성 및 차별성

**sf**와 **sp**에는 여러 차이가 있다. 가장 눈에 띄는 차이는 **sp** 클래스는 정형적인 S4 클래스를 사용하는 반면, **sf**는 보다 비공식적인 S3 클래스 계층을 사용한다는 점이다. **sf** 객체는 data.frame 또는 tibble에서 파생되며, 이로 인해 기존 R 생태계, 특히 **tidyverse** 패키지 계열과 더 쉽게 연동할 수 있다. sf 객체는 기하 데이터를 리스트-컬럼에 저장하며, 이는 기하 데이터가 항상 리스트 요소로 유지됨을 의미한다. 반면, **sp**는 데이터 구조를 덜 엄격하게 사용하며, 예를 들어 SpatialPoints나 SpatialPixels의 모든 좌표를 행렬에 저장한다. 이는 특정 문제에 대해 더 높은 성능을 제공하지만 리스트-컬럼으로는 구현할 수 없는 방식이다. sf 객체 x를 **sp**로 변환하려면 다음과 같이 하면 된다.

library(sp)  
y = as(x, "Spatial")

그리고 반대로 sp 객체를 **sf**로 변환하려면 다음과 같이 하면 된다.

x0 = st\_as\_sf(y)

이러한 변환에는 몇 가지 제한 사항이 있다.

* **sp**는 LINESTRING과 MULTILINESTRING 지오메트리, 또는 POLYGON과 MULTIPOLYGON 지오메트리를 구분하지 않는다. 예를 들어, LINESTRING을 sp로 변환한 후 다시 변환하면 MULTILINESTRING으로 반환된다.
* **sp**는 GEOMETRYCOLLECTION 지오메트리를 가지고 있지 않으며, “빅 세븐” (3.1.1절) 범주에 포함되지 않는 지오메트리를 가진 sf 객체를 표현할 수 있는 구조를 가지고 있지 않다.
* sf 또는 sfc 객체 중 혼합된 지오메트리를 가진 GEOMETRY 유형인 경우, 이를 sp 객체로 변환할 수 없다.
* 속성-지오메트리 관계 속성은 **sp**로 변환할 때 손실된다.
* 두 개 이상의 지오메트리 리스트-컬럼을 가진 sf 객체는 **sp**로 변환할 때 부차적인 리스트-컬럼을 잃게 된다.

## A.3 마이그레이션을 위한 코드와 패키지

**sf**의 GitHub 사이트 위키 페이지(<https://github.com/r-spatial/sf/wiki/Migrating>)에는 **rgeos**, **rgdal**, **sp**의 메서드 및 함수와 그에 상응하는 **sf** 메서드 및 함수의 목록이 포함되어 있다. 이는 기존 코드나 패키지를 변환하는 데 도움이 될 수 있다.

코드를 마이그레이션하는 간단한 방법은 **rgdal** 패키지의 readOGR() 함수만 사용하여 파일을 읽는 경우이다. 다른 대안은 **sf** 패키지의 read\_sf() 함수를 사용하는 것이다.

x = as(sf::read\_sf("file"), "Spatial")

그러나 readOGR() 함수를 사용할 때는 아규먼트를 보다 주의해서 다뤄야 한다. 우리는 **rgdal**, **rgeos**, **maptools** 없이 전체적으로 실행되도록, 가능하면 **sp** 없이 실행되도록, 이전 책인 *Applied Spatial Data Analysis with R* (Virgilio Gómez-Rubio, Bivand, Pebesma, Gómez-Rubio, 2013)의 모든 코드를 변환하는 작업을 진행 중이다. 해당 스크립트는 <https://github.com/rsbivand/sf_asdar2ed>에서 확인할 수 있다.

## A.4 raster와 terra 패키지

**raster** 패키지는 2010년부터 R을 사용한 래스터 데이터 분석의 핵심적인 패키지였으며, 이후 “Geographic Data Analysis and Modeling” (Hijmans 2023a) 패키지로 성장하여 모든 종류의 공간 데이터에 사용되고 있다. **raster** 패키지는 벡터 데이터를 위해 **sp** 객체를 사용하고, GDAL 라이브러리가 제공하는 형식으로 데이터를 읽고 쓸 때 **terra**를 사용한다. 후속 패키지인 **terra**는 “Spatial Data Analysis” (Hijmans 2023b)를 위한 패키지로, “raster 패키지와 매우 비슷하지만 […] 더 많은 기능을 제공하며, 사용하기 더 쉽고, […] 더 빠르다.” **terra** 패키지는 벡터 데이터를 위한 자체 클래스를 가지고 있지만 많은 **sf** 객체를 받아들인다. 물론 위에서 열거한 **sp** 변환 시에 발생하는 것과 유사한 제한 사항이 부과된다. **terra** 패키지는 GDAL, GEOS, PROJ와의 직접적인 연결을 가지고 있어, 더 이상 다른 패키지가 필요 없다.

**raster** 또는 **terra** 패키지의 래스터 레이어나 래스터 스택은 st\_as\_stars() 함수를 사용하여 stars 객체로 변환할 수 있다. **sf** 패키지의 st\_as\_sf() 함수를 통해 **terra** 패키지의 SpatVector 객체를 변환할 수 있다.

로버트 히즈먼(Robert Hijmans)이 저술한 온라인 책 *Spatial Data Science with R*(<https://rspatial.org/terra>)은 **terra**를 활용한 공간 데이터 분석 방법을 자세히 설명하고 있다. **sf**와 **stars** 패키지 및 이 책에서 다루는 여러 r-spatial 패키지들은 r-spatial GitHub 조직에 속해 있다(여기서 r과 spatial 사이에 하이픈이 있다는 점에 유의하라. 히즈먼의 조직에는 하이픈이 없다). 해당 조직은 블로그 사이트가 있으며, 이 책에 대한 링크가 <https://r-spatial.org/book>에서 제공된다.

**sf**와 **stars** 패키지, 그리고 **terra** 패키지는 많은 공통된 목표를 가지고 있지만, 약간 다른 방식으로 이를 달성하려고 하며, 데이터 분석, 소프트웨어 엔지니어링, 커뮤니티 관리에 대한 상이한 강조점을 가지고 있다. 이러한 차이점이 일부 사용자들에게 혼란을 줄 수 있지만, 우리는 이러한 차이점들이 R 패키지 생태계를 풍요롭게 하고, 사용자에게 유익하며, 다양성과 선택을 장려한다고 믿는다. 또한 이는 사용자가 R을 공간 데이터 문제에 사용할 때 새로운 아이디어를 계속 시도하도록 유도하고, 결국 R spatial의 깃발을 만천하에 더 높이는데 기여할 것으로 믿는다.

# Appendix B — R 기초

여기서는 이 책을 읽는 데 도움이 될 수 있는 최소한의 R 기초를 제공한다. R 기초에 대한 보다 자세한 내용은 Wickham (2014) 2장을 보라.

## B.1 파이프 오퍼레이터

|> (파이프) 기호는 ’*그렇다면*’으로 읽어야 한다.

a |> b() |> c() |> d(n = 10)

위의 코드는 a를 실행한 후 b를 실행하고, 그 다음에 c, d를 실행하며, n은 10인 경우와 같다는 의미이다. 이 파이프 코드는 다음과 같이 쓸 수 있다.

d(c(b(a)), n = 10)

혹은 다음과 같이 쓸 수 있다.

tmp1 <- b(a)  
tmp2 <- c(tmp1)  
tmp3 <- d(tmp2, n = 10)

많은 사람들은 이러한 파이프 형태가 읽기 쉽다고 생각할 것이다. 왜냐하면 실행 순서가 읽는 순서대로 왼쪽에서 오른쪽으로 진행되기 때문에. 위의 두 번째 구문에 나타나 있는 포섭 함수 호출처럼, 중간 결과에 대해 이름을 지정할 필요가 없게 된다. 그런데 포섭 함수 호출처럼, 예상과 다른 중간 결과를 디버깅하는 것이 어렵게 된다. 중간 결과는 메모리에 존재하므로 세 번째 구문에 비해 메모리 할당의 절약을 기대할 수는 없다는 점은 유의할 필요가 있다. 원래 파이프 오퍼레이터는 **magrittr** 패키지에서 %>%의 형태로 제시되었지만, R 4.1.0에서 네이티브 파이프로 수용되어 |> 기호를 사용하고 있다. 어느 쪽을 사용해도 상관이 없다.

## B.2 데이터 구조

Chambers (2016)에 의해 지적된 바와 같이, “*R에서 존재하는 모든 것은 객체이다*”. 객체에는 언어 객체나 함수와 같이 특정한 일을 하는 객체뿐만 아니라 데이터 객체와 같은 더 기본적인 “*것들*”도 포함된다. 다음에서 몇 가지 기본적인 R 데이터 구조에 대해 논의하겠다.

### B.2.1 동질 벡터

데이터 객체는 데이터를 포함하고 있으며, 때로는 메타데이터도 포함된다. 데이터는 항상 벡터 형태로 존재하며, 벡터는 다양한 유형을 가질 수 있다. 유형은 typeof() 함수를 사용하여 확인할 수 있으며, 벡터의 길이는 length() 함수를 사용하여 확인할 수 있다. 벡터는 c() 함수를 사용하여 개별 요소들을 결합함으로써 생성된다.

typeof(1:10)  
# [1] "integer"  
length(1:10)  
# [1] 10  
typeof(1.0)  
# [1] "double"  
length(1.0)  
# [1] 1  
typeof(c("foo", "bar"))  
# [1] "character"  
length(c("foo", "bar"))  
# [1] 2  
typeof(c(TRUE, FALSE))  
# [1] "logical"

이런 종류의 벡터를 동질(homegeneous) 벡터라고 부르는데, 이는 이러한 종류의 벡터가 오로지 한 가지 유형의 데이터만을 가질 수 있기 때문이다.

벡터는 길이가 0일 수 있다는 점에 유의해야 한다.

i <- integer(0)  
typeof(i)  
# [1] "integer"  
i  
# integer(0)  
length(i)  
# [1] 0

우리는 한 벡터의 요소들을 [ 혹은 [[를 이용하여 불러올 수 있다(할당의 맥락에서는 요소들을 대체할 수 있다).

a <- c(1,2,3)  
a[2]  
# [1] 2  
a[[2]]  
# [1] 2  
a[2:3]  
# [1] 2 3  
a[2:3] <- c(5,6)  
a  
# [1] 1 5 6  
a[[3]] <- 10  
a  
# [1] 1 5 10

차이점은 [ ]는 인덱스 *범위*(또는 여러 인덱스)에 대해 작업할 수 있는 반면, [[ ]]는 단일 벡터 값에 대해 작업한다는 것이다.

### B.2.2 이질 벡터: list

두 번째 벡터 유형은 list로, 상이한 유형을 결합할 수 있다는 의미에서 이질(heterogeneous) 벡터라고 불린다.

l <- list(3, TRUE, "foo")  
typeof(l)  
# [1] "list"  
length(l)  
# [1] 3

리스트에는 [ ]와 [[ ]] 사이에 추가적인 구분이 있다. 즉, 단일 [ ]는 항상 리스트를 반환하고, [[ ]]는 리스트 요소의 *내용*을 반환한다.

l[1]  
# [[1]]  
# [1] 3  
l[[1]]  
# [1] 3

교체의 맥락에서는, 리스트를 제공할 때는 [ ]를 사용하고, 새로운 값을 제공할 때는 [[ ]]를 사용한다.

l[1:2] <- list(4, FALSE)  
l  
# [[1]]  
# [1] 4  
#   
# [[2]]  
# [1] FALSE  
#   
# [[3]]  
# [1] "foo"  
l[[3]] <- "bar"  
l  
# [[1]]  
# [1] 4  
#   
# [[2]]  
# [1] FALSE  
#   
# [[3]]  
# [1] "bar"

리스트 요소에 *이름*을 지정할 수 있다.

l <- list(first = 3, second = TRUE, third = "foo")  
l  
# $first  
# [1] 3  
#   
# $second  
# [1] TRUE  
#   
# $third  
# [1] "foo"

l[["second"]]와 같은 방식으로 이름을 사용할 수고, 좀 더 단순한 형태로 사용할 수도 있다.

l$second  
# [1] TRUE  
l$second <- FALSE  
l  
# $first  
# [1] 3  
#   
# $second  
# [1] FALSE  
#   
# $third  
# [1] "foo"

이름을 사용하는 것은 편리해 보이지만, 이름 속성에서 이름을 우선 찾아야 한다는 점도 있다(아래 참조).

### B.2.3 NULL과 리스트 요소의 제거

NULL은 R에서 널(null) 값을 표현하는 방식이다. NULL은 단순 비교에서 상식적인 결과가 도출되지 않는다는 점에서 주의를 요한다.

3 == NULL # not FALSE!  
# logical(0)  
NULL == NULL # not even TRUE!  
# logical(0)

따라서 특별히 취급되어야 하는데, is.null() 함수가 유용하다.

is.null(NULL)  
# [1] TRUE

리스트의 특정 요소를 제거하려면, 제거 대상을 포함하지 않는 새로운 리스트를 생성하면 된다.

l <- l[c(1,3)] # remove second, implicitly  
l  
# $first  
# [1] 3  
#   
# $third  
# [1] "foo"

혹은, 제거 대상에 NULL을 할당할 수도 있다.

l$second <- NULL  
l  
# $first  
# [1] 3  
#   
# $third  
# [1] "foo"

### B.2.4 속성

임의의 메타데이터 객체를 데이터 객체에 결합하는 것이 가능하다.

a <- 1:3  
attr(a, "some\_meta\_data") = "foo"  
a  
# [1] 1 2 3  
# attr(,"some\_meta\_data")  
# [1] "foo"

이 메타데이터를 검색하거나 교체할 수 있다.

attr(a, "some\_meta\_data")  
# [1] "foo"  
attr(a, "some\_meta\_data") <- "bar"  
attr(a, "some\_meta\_data")  
# [1] "bar"

본질적으로 객체의 속성은 이름이 지정된 리스트이며, 우리는 전체 리스트를 다음과 같이 가져오거나 설정할 수 있다.

attributes(a)  
# $some\_meta\_data  
# [1] "bar"  
attributes(a) = list(some\_meta\_data = "foo")  
attributes(a)  
# $some\_meta\_data  
# [1] "foo"

R은 여러 속성을 특별하게 처리하며, 전체 세부 사항은 ?attributes에서 확인할 수 있다. 이제 몇 가지 특별한 속성에 대해 설명하고자 한다.

#### B.2.4.1 객체 클래스와 class 속성

R의 모든 객체는 “클래스를 가진다”가 의미하는 바처럼, class(obj)는 obj의 클래스에 대한 문자 벡터를 반환한다. 일부 객체는 기본 벡터와 같은 *암시적* 클래스를 가진다.

class(1:3)  
# [1] "integer"  
class(c(TRUE, FALSE))  
# [1] "logical"  
class(c("TRUE", "FALSE"))  
# [1] "character"

클래스를 명시적으로 설정할 수도 있다. attr을 사용하거나 표현식의 왼쪽에서 class를 사용하면 된다.

a <- 1:3  
class(a) <- "foo"  
a  
# [1] 1 2 3  
# attr(,"class")  
# [1] "foo"  
class(a)  
# [1] "foo"  
attributes(a)  
# $class  
# [1] "foo"

이 경우, 새로 설정된 클래스가 이전의 암시적인 클래스를 덮어쓴다. 이렇게 하면, 클래스 이름을 메서드 이름에 추가하여 foo 클래스에 대한 메서드를 추가할 수 있다.

print.foo <- function(x, ...) {   
 print(paste("an object of class foo with length", length(x)))  
}  
print(a)  
# [1] "an object of class foo with length 3"

이러한 메서드를 제공하는 목적은 일반적으로 더 사용하기 쉬운 소프트웨어를 만드는 것이지만, 동시에 객체를 더 불투명하게 만들 수도 있다. 클래스 속성이 제거된 후 객체를 출력하여 “무엇으로 구성되어 있는지” 확인하는 것이 때때로 유용할 수 있다.

unclass(a)  
# [1] 1 2 3

좀 더 구체적인 예로, 패키지 sf 패키지를 사용하여 다각형을 만드는 경우를 고려해보자.

library(sf) |> suppressPackageStartupMessages()  
p <- st\_polygon(list(rbind(c(0,0), c(1,0), c(1,1), c(0,0))))  
p  
# POLYGON ((0 0, 1 0, 1 1, 0 0))

이것은 잘 알려진 텍스트 형식(WKT)을 출력한다. 데이터 구조가 어떤지 알아보려면, 다음과 같이 하면 된다.

unclass(p)  
# [[1]]  
# [,1] [,2]  
# [1,] 0 0  
# [2,] 1 0  
# [3,] 1 1  
# [4,] 0 0

#### B.2.4.2 dim 속성

dim 속성은 행렬 또는 어레이(array)의 차원을 설정한다.

a <- 1:8  
class(a)  
# [1] "integer"  
attr(a, "dim") <- c(2,4) # or: dim(a) = c(2,4)  
class(a)  
# [1] "matrix" "array"  
a  
# [,1] [,2] [,3] [,4]  
# [1,] 1 3 5 7  
# [2,] 2 4 6 8  
attr(a, "dim") <- c(2,2,2) # or: dim(a) = c(2,2,2)  
class(a)  
# [1] "array"  
a  
# , , 1  
#   
# [,1] [,2]  
# [1,] 1 3  
# [2,] 2 4  
#   
# , , 2  
#   
# [,1] [,2]  
# [1,] 5 7  
# [2,] 6 8

### B.2.5 names 속성

이름이 지정된 벡터는 names 속성에 이름을 가지고 있다. 위에서 리스트의 예를 봤고, 숫자 벡터의 예는 다음과 같다.

a <- c(first = 3, second = 4, last = 5)  
a["second"]  
# second   
# 4  
attributes(a)  
# $names  
# [1] "first" "second" "last"

다른 이름 속성으로는 행렬이나 어레이의 dimnames가 있다. 이는 차원의 이름뿐만 아니라 각 차원에 연관된 값의 레이블도 지정한다.

a <- matrix(1:4, 2, 2)  
dimnames(a) <- list(rows = c("row1", "row2"),  
 cols = c("col1", "col2"))  
a  
# cols  
# rows col1 col2  
# row1 1 3  
# row2 2 4  
attributes(a)  
# $dim  
# [1] 2 2  
#   
# $dimnames  
# $dimnames$rows  
# [1] "row1" "row2"  
#   
# $dimnames$cols  
# [1] "col1" "col2"

data.frame 객체는 행과 열을 가지며, 각 행과 열에는 이름이 있다.

df <- data.frame(a = 1:3, b = c(TRUE, FALSE, TRUE))  
attributes(df)  
# $names  
# [1] "a" "b"  
#   
# $class  
# [1] "data.frame"  
#   
# $row.names  
# [1] 1 2 3

### B.2.6 structure의 사용

프로그래밍할 때, 객체를 반환하기 전에 속성을 추가하거나 수정하는 패턴은 매우 일반적이다. 다음의 예시를 보자.

f <- function(x) {  
 a <- create\_obj(x) # call some other function  
 attributes(a) <- list(class = "foo", meta = 33)  
 a  
}

마지막 두 문장은 다음과 같이 축약할 수 있다.

f <- function(x) {  
 a <- create\_obj(x) # call some other function  
 structure(a, class = "foo", meta = 33)  
}

여기서 structure() 함수는 첫 번째 아규먼트로 전달된 객체의 속성을 추가 및 교체하거나(값이 NULL인 경우) 속성을 제거한다.

## B.3 MULTIPOLYGON 분할하기

위의 예를 사용하여 MULTIPOLYGON이 포함된 sf 객체를 여러 조각으로 분해할 수 있다. 예를 들어, nc 데이터셋을 사용한다고 가정해보자.

system.file("gpkg/nc.gpkg", package = "sf") |>   
 read\_sf() -> nc

nc의 속성에서 다음을 볼 수 있다.

attributes(nc)  
# $names  
# [1] "AREA" "PERIMETER" "CNTY\_" "CNTY\_ID" "NAME"   
# [6] "FIPS" "FIPSNO" "CRESS\_ID" "BIR74" "SID74"   
# [11] "NWBIR74" "BIR79" "SID79" "NWBIR79" "geom"   
#   
# $row.names  
# [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
# [16] 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30  
# [31] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45  
# [46] 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60  
# [61] 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75  
# [76] 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90  
# [91] 91 92 93 94 95 96 97 98 99 100  
#   
# $class  
# [1] "sf" "tbl\_df" "tbl" "data.frame"  
#   
# $sf\_column  
# [1] "geom"  
#   
# $agr  
# AREA PERIMETER CNTY\_ CNTY\_ID NAME FIPS   
# <NA> <NA> <NA> <NA> <NA> <NA>   
# FIPSNO CRESS\_ID BIR74 SID74 NWBIR74 BIR79   
# <NA> <NA> <NA> <NA> <NA> <NA>   
# SID79 NWBIR79   
# <NA> <NA>   
# Levels: constant aggregate identity

geom이라는 이름의 지오메트리 컬럼이 있다는 것을 알 수 있다. 이 컬럼만 끄집어 낼 수 있다.

nc$geom  
# Geometry set for 100 features   
# Geometry type: MULTIPOLYGON  
# Dimension: XY  
# Bounding box: xmin: -84.3 ymin: 33.9 xmax: -75.5 ymax: 36.6  
# Geodetic CRS: NAD27  
# First 5 geometries:  
# MULTIPOLYGON (((-81.5 36.2, -81.5 36.3, -81.6 3...  
# MULTIPOLYGON (((-81.2 36.4, -81.2 36.4, -81.3 3...  
# MULTIPOLYGON (((-80.5 36.2, -80.5 36.3, -80.5 3...  
# MULTIPOLYGON (((-76 36.3, -76 36.3, -76 36.3, -...  
# MULTIPOLYGON (((-77.2 36.2, -77.2 36.2, -77.3 3...

해당 컬럼만을 포함한 객체가 다음의 속성을 가지고 있음을 확인할 수 있다.

attributes(nc$geom)  
# $n\_empty  
# [1] 0  
#   
# $crs  
# Coordinate Reference System:  
# User input: NAD27   
# wkt:  
# GEOGCRS["NAD27",  
# DATUM["North American Datum 1927",  
# ELLIPSOID["Clarke 1866",6378206.4,294.978698213898,  
# LENGTHUNIT["metre",1]]],  
# PRIMEM["Greenwich",0,  
# ANGLEUNIT["degree",0.0174532925199433]],  
# CS[ellipsoidal,2],  
# AXIS["geodetic latitude (Lat)",north,  
# ORDER[1],  
# ANGLEUNIT["degree",0.0174532925199433]],  
# AXIS["geodetic longitude (Lon)",east,  
# ORDER[2],  
# ANGLEUNIT["degree",0.0174532925199433]],  
# USAGE[  
# SCOPE["Geodesy."],  
# AREA["North and central America: Antigua and Barbuda - onshore. Bahamas - onshore plus offshore over internal continental shelf only. Belize - onshore. British Virgin Islands - onshore. Canada onshore - Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, Northwest Territories, Nova Scotia, Nunavut, Ontario, Prince Edward Island, Quebec, Saskatchewan and Yukon - plus offshore east coast. Cuba - onshore and offshore. El Salvador - onshore. Guatemala - onshore. Honduras - onshore. Panama - onshore. Puerto Rico - onshore. Mexico - onshore plus offshore east coast. Nicaragua - onshore. United States (USA) onshore and offshore - Alabama, Alaska, Arizona, Arkansas, California, Colorado, Connecticut, Delaware, Florida, Georgia, Idaho, Illinois, Indiana, Iowa, Kansas, Kentucky, Louisiana, Maine, Maryland, Massachusetts, Michigan, Minnesota, Mississippi, Missouri, Montana, Nebraska, Nevada, New Hampshire, New Jersey, New Mexico, New York, North Carolina, North Dakota, Ohio, Oklahoma, Oregon, Pennsylvania, Rhode Island, South Carolina, South Dakota, Tennessee, Texas, Utah, Vermont, Virginia, Washington, West Virginia, Wisconsin and Wyoming - plus offshore . US Virgin Islands - onshore."],  
# BBOX[7.15,167.65,83.17,-47.74]],  
# ID["EPSG",4267]]  
#   
# $class  
# [1] "sfc\_MULTIPOLYGON" "sfc"   
#   
# $precision  
# [1] 0  
#   
# $bbox  
# xmin ymin xmax ymax   
# -84.3 33.9 -75.5 36.6

네 번째 리스트 요소의 *내용*을 가져온다.

nc$geom[[4]] |> format(width = 60, digits = 5)  
# [1] "MULTIPOLYGON (((-76.009 36.32, -76.017 36.338, -76.033 36..."

객체의 클래스가 리스트임을 확인한다.

typeof(nc$geom[[4]])  
# [1] "list"

어떤 속성이 있는지 확인한다.

attributes(nc$geom[[4]])  
# $class  
# [1] "XY" "MULTIPOLYGON" "sfg"

그리고 length를 확인한다.

length(nc$geom[[4]])  
# [1] 3

길이는 외부 링의 개수를 나타낸다. 멀티폴리곤은 하나 이상의 폴리곤으로 구성될 수 있다. 대부분의 카운티는 하나의 폴리곤만 가지고 있음을 알 수 있다.

lengths(nc$geom)  
# [1] 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
# [32] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 2 1 1 1 1 1  
# [63] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1  
# [94] 1 2 1 1 1 1 1

멀티폴리곤은 폴리곤의 리스트이다.

typeof(nc$geom[[4]])  
# [1] "list"

네 번째 멀티폴리곤의 *첫 번째* 폴리곤은 다시 리스트이다. 왜냐하면 폴리곤은 외부 링을 가지며, 그 뒤에 여러 개의 내부 링(구멍)이 올 *수* 있기 때문이다.

typeof(nc$geom[[4]][[1]])  
# [1] "list"

해당 폴리곤이 외부 링 하나만 포함하고 있음을 알 수 있다.

length(nc$geom[[4]][[1]])  
# [1] 1

해당 폴리곤의 유형, 차원, 첫 번째 좌표 집합를 다음과 같이 출력할 수 있다.

typeof(nc$geom[[4]][[1]][[1]])  
# [1] "double"  
dim(nc$geom[[4]][[1]][[1]])  
# [1] 26 2  
head(nc$geom[[4]][[1]][[1]])  
# [,1] [,2]  
# [1,] -76.0 36.3  
# [2,] -76.0 36.3  
# [3,] -76.0 36.3  
# [4,] -76.0 36.4  
# [5,] -76.1 36.3  
# [6,] -76.2 36.4

속성을 변경할 수 있는데, 예를 들어 세 번째 좌표의 위도값을 다음과 같이 변경할 수 있다.

nc$geom[[4]][[1]][[1]][3,2] <- 36.5