



Программирование в среде R

Шевцов Василий Викторович,
директор ДИТ РУДН, shevtsov_vv@rudn.university

GeoJSON

Определение

GeoJSON — открытый формат, предназначенный для хранения географических структур данных, основан на JSON.

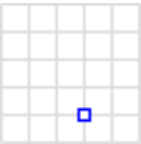
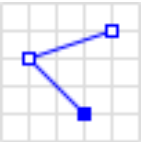
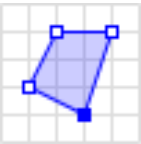
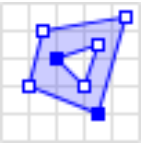
Формат может хранить примитивные типы для описания географических объектов, такие как: точки (адреса и местоположения), линии (улицы, шоссе, границы), полигоны (страны, штаты, участки земли). Также могут храниться так называемые мультитипы, которые представляют собой объединение нескольких примитивных типов.

Формат GeoJSON отличается от других стандартов ГИС тем, что он был написан и поддерживается не какой-либо организацией по стандартизации, а с помощью рабочей группы разработчиков.

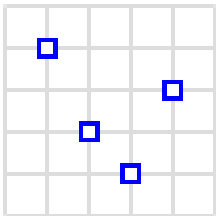
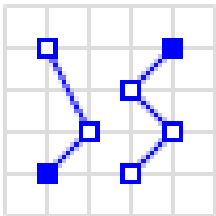
Дальнейшим развитием GeoJSON является TopoJSON, расширение GeoJSON, которое кодирует геопространственную топологию, и, как правило, обеспечивает меньший размер файлов.

Формат GeoJSON начал обсуждаться рабочей группой в марте 2007 года и окончательная спецификация стандарта была готова к июню 2008 года.

Типы GeoJSON

Типы	Примеры	
Point		<pre>{ "type": "Point", "coordinates": [30, 10] }</pre>
LineString		<pre>{ "type": "LineString", "coordinates": [[30, 10], [10, 30], [40, 40]] }</pre>
Polygon		<pre>{ "type": "Polygon", "coordinates": [[[30, 10], [40, 40], [20, 40], [10, 20], [30, 10]]] }</pre>
		<pre>{ "type": "Polygon", "coordinates": [[[35, 10], [45, 45], [15, 40], [10, 20], [35, 10]], [[20, 30], [35, 35], [30, 20], [20, 30]]] }</pre>

Мульти типы GeoJSON

Типы	Примеры	
MultiPoint		<pre>{ "type": "MultiPoint", "coordinates": [[10, 40], [40, 30], [20, 20], [30, 10]] }</pre>
MultiLineString		<pre>{ "type": "MultiLineString", "coordinates": [[[10, 10], [20, 20], [10, 40]], [[40, 40], [30, 30], [40, 20], [30, 10]]] }</pre>

Мульти типы GeoJSON

Типы	Примеры	
MultiPolygon		<pre>{ "type": "MultiPolygon", "coordinates": [[[[30, 20], [45, 40], [10, 40], [30, 20]]], [[[15, 5], [40, 10], [10, 20], [5, 10], [15, 5]]]] }</pre>
		<pre>{ "type": "MultiPolygon", "coordinates": [[[[40, 40], [20, 45], [45, 30], [40, 40]]], [[[20, 35], [10, 30], [10, 10], [30, 5], [45, 20], [20, 35]], [[30, 20], [20, 15], [20, 25], [30, 20]]]] }</pre>

Использование

GeoJSON поддерживается множеством картографических программных пакетов и ГИС, включая OpenLayers, Leaflet, MapServer, Geoforge software, GeoServer, GeoDjango, GDAL, Safe Software FME, и CartoDB. Кроме этого, можно использовать GeoJSON с PostGIS и Mapnik, оба работают с форматами с помощью библиотеки GDAL OGR. Онлайн-сервисы Bing Maps, Yahoo! и Google также поддерживают GeoJSON в своих API.

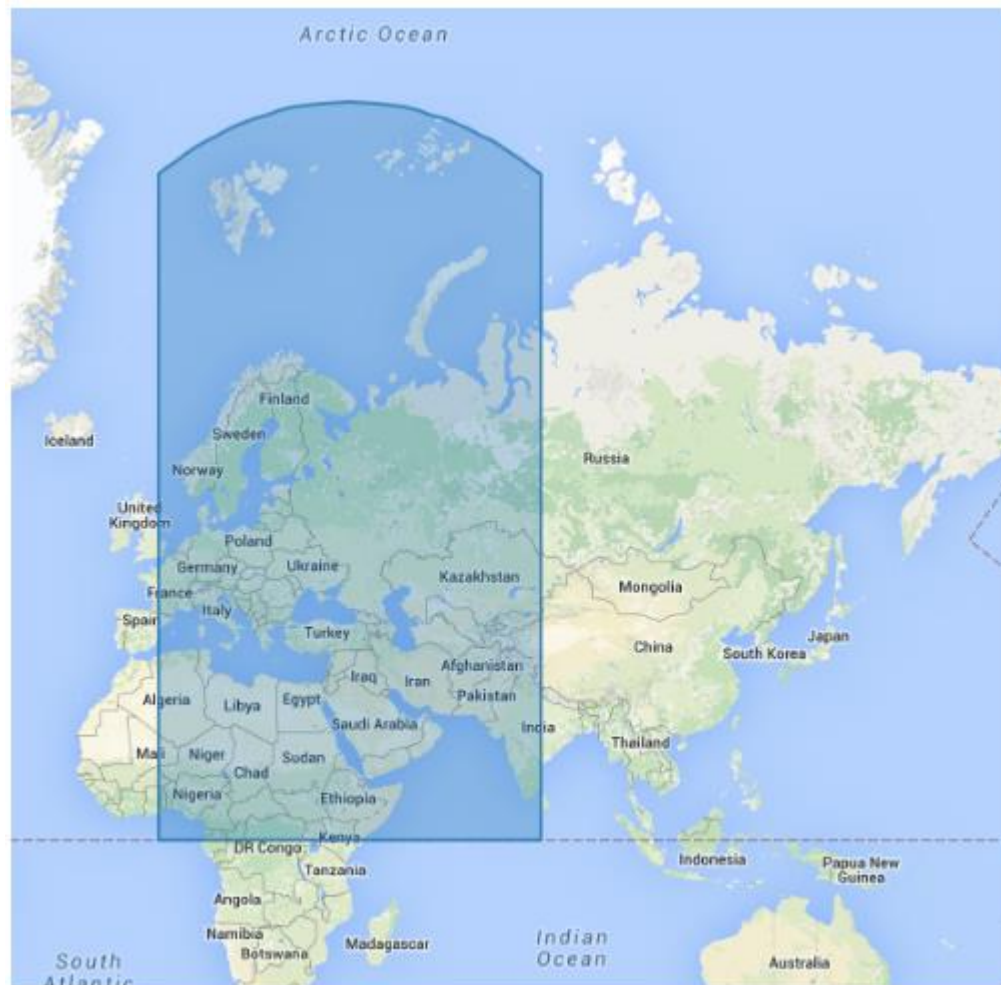
Интерфейс Javascript API v3 карт Google Maps напрямую поддерживают интеграцию слоёв данных GeoJSON с 19 марта 2014 года.

GitHub тоже поддерживает GeoJSON[и GeoJSON-экспорт Potrace.

Искажение

Сферическая геометрия будет казаться искаженной при визуализации на карте из-за природы проецирования трехмерной сферы, такой как земля, на плоскую плоскость.

Например, возьмем спецификацию сферического квадрата, определяемого точками долготы широты $(0,0)$, $(80,0)$, $(80,80)$, и $(0,80)$. На следующем рисунке показана область, охватываемая этим регионом:



Географические координаты

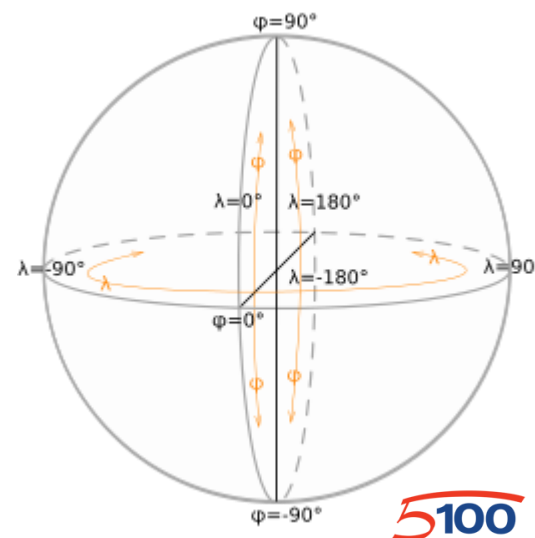
Чтобы указать положение точки на поверхности Земли можно воспользоваться:

- Широтой(latitude) — идет с севера на юг. 0 — экватор.
Изменяется от -90 до 90 градусов.
- Долготой(longitude) — идет с запада на восток. 0 — нулевой меридиан (Гринвич).
Изменяется от -180 до 180 градусов.

Нужно обратить внимание что x — это долгота, y — широта(Google Maps, Яндекс.Карты и все остальные сервисы указывают долготу первой).

Географические координаты можно перевести в пространственные — просто точка (x, y, z) . Количество знаков после запятой определяет точность:

Градусы	Дистанция
1	111 km
0.1	11.1 km
0.01	1.11 km
0.001	111 m
0.0001	11.1 m
0.00001	1.11 m
0.000001	11.1 cm



Широта и Долгота

Широта

Широтá — угол φ между местным направлением зенита и плоскостью экватора, отсчитываемый от 0° до 90° в обе стороны от экватора. Географическую широту точек, лежащих в северном полушарии, (северную широту) принято считать положительной, широту точек в южном полушарии — отрицательной. О широтах, близких к полюсам, принято говорить как о высоких, а о близких к экватору — как о низких.

Из-за отличия формы Земли от шара, географическая широта точек несколько отличается от их геоцентрической широты, то есть от угла между направлением на данную точку из центра Земли и плоскостью экватора.

Широту места можно определить с помощью таких астрономических инструментов, как секстант или гномон (прямое измерение), также можно воспользоваться системами GPS или ГЛОНАСС (косвенное измерение).

Долгота

Долготá — двугранный угол λ между плоскостью меридиана, проходящего через данную точку, и плоскостью начального нулевого меридиана, от которого ведётся отсчёт долготы. Долготу от 0° до 180° к востоку от нулевого меридиана называют восточной, к западу — западной. Восточные долготы принято считать положительными, западные — отрицательными.

Выбор нулевого меридиана произволен и зависит только от соглашения. Сейчас за нулевой меридиан принят Опорный меридиан, проходящий рядом с обсерваторией в Гринвиче, на юго-востоке Лондона. В качестве нулевого ранее выбирались меридианы обсерваторий Парижа, Кадиса, Пулкова и т. д.

Карты

Polygon maps

Polygon maps

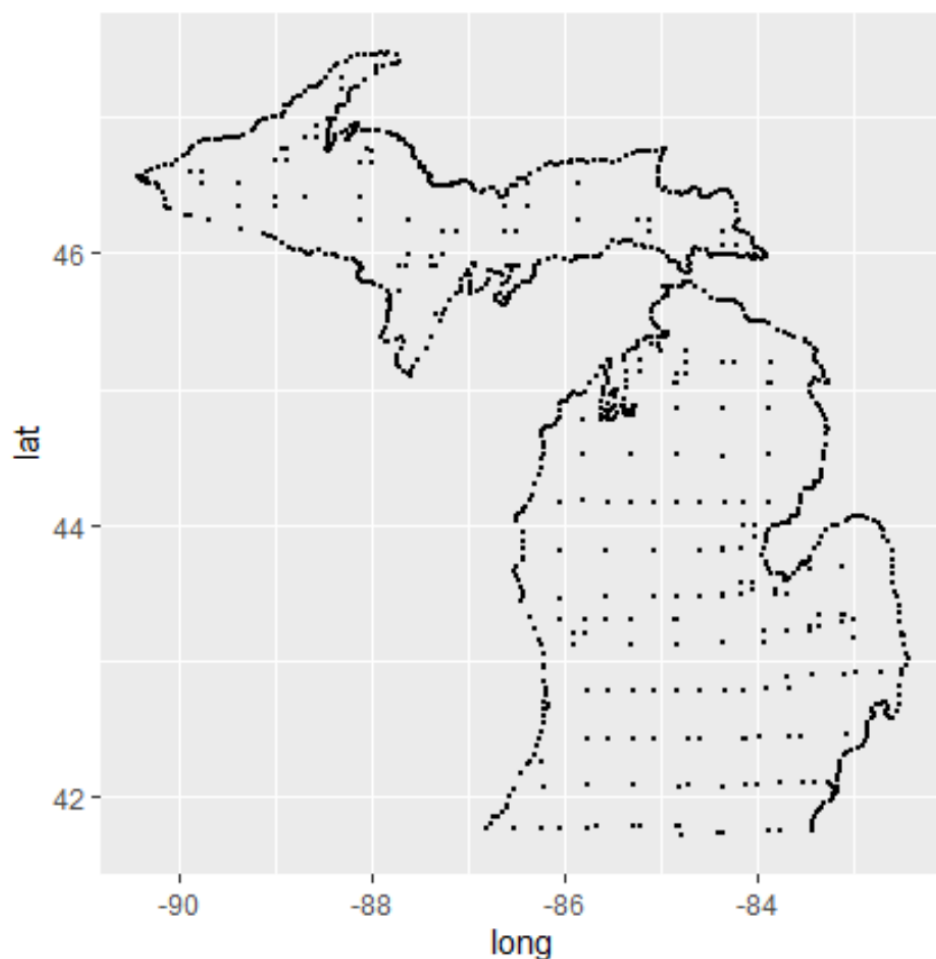
Возможно, самый простой подход к составлению карт - это использование функции `geom_polygon()` для построения границ различных регионов. Для этого примера мы берем данные из пакета `maps` с помощью функции `ggplot2::map_data()`. Пакет карт не особенно точен и современен, но он встроен в R, так что это простое место для начала. Вот набор данных, определяющий границы округа Мичиган:

```
mi_counties <- ggplot2::map_data("county", "michigan")
head(mi_counties)
```

	long	lat	group	order	region	subregion
1	-83.88675	44.85686	1	1	michigan	alcona
2	-83.36536	44.86832	1	2	michigan	alcona
3	-83.36536	44.86832	1	3	michigan	alcona
4	-83.33098	44.83968	1	4	michigan	alcona
5	-83.30806	44.80530	1	5	michigan	alcona
6	-83.30233	44.77665	1	6	michigan	alcona

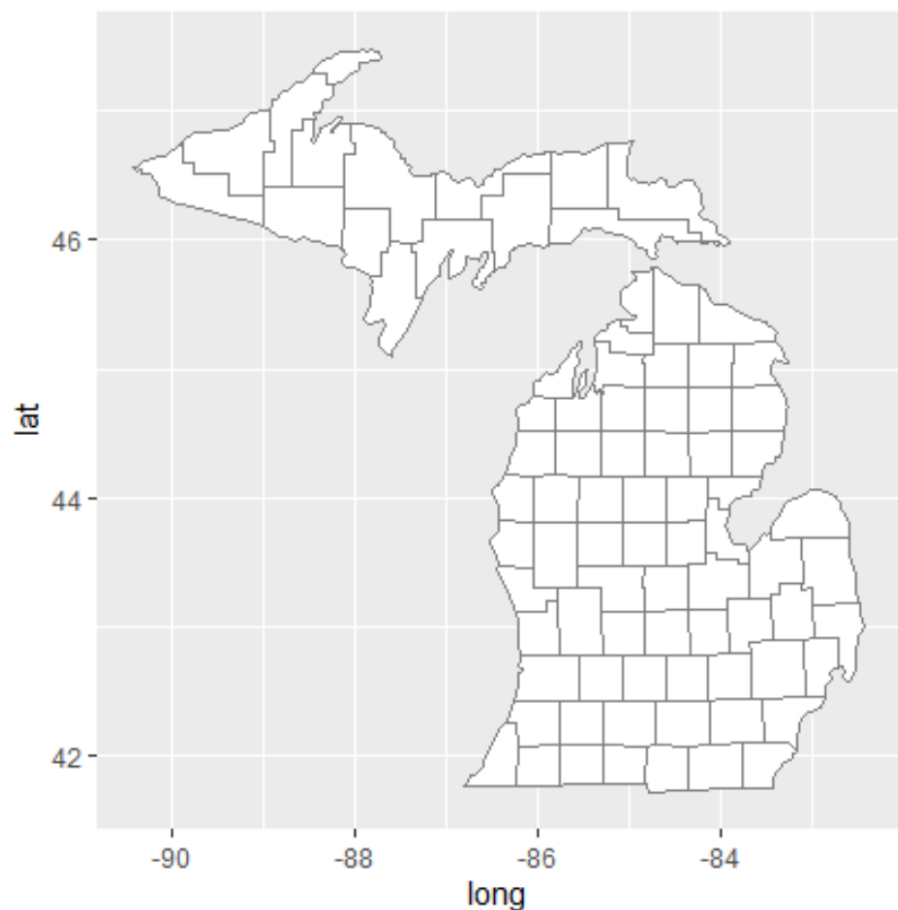
Polygon maps

```
ggplot(mi_counties, aes(long, lat)) +  
  geom_point(size = .25, show.legend = FALSE) +  
  coord_quickmap()
```



Polygon maps

```
ggplot(mi_counties, aes(long, lat, group = group)) +  
  geom_polygon(fill = "white", colour = "grey50") +  
  coord_quickmap()
```



В обоих графиках используется `coord_quickmap()` для настройки осей, чтобы гарантировать, что долгота и широта отображаются в одном масштабе.

Simple features maps

У описанного выше подхода есть несколько ограничений, не последним из которых является тот факт, что простой формат данных “долгота-широта” обычно не используется в реальном картографировании. Векторные данные для карт обычно кодируются с использованием стандарта “простые объекты”, разработанного открытым геопространственным консорциумом. Пакет **sf** (Pebesma 2018), разработанный компанией Edzer Pebesma <https://github.com/r-spatial/sf> предоставляет набор инструментов для работы с такими данными, а функции `geom_sf()` и `coord_sf()` в `ggplot2` предназначены для совместной работы с пакетом **sf**.

Чтобы работать с этими функциями, воспользуемся пакетом **ozmaps** Майкла Самнера <https://github.com/mdsumner/ozmaps/> который предоставляет карты для границ австралийского штата, районов местного самоуправления, избирательных границ и так далее (Sumner 2019). Чтобы проиллюстрировать, как выглядит набор данных **sf**, мы импортируем набор данных, изображающий границы австралийских штатов и территорий

Simple features maps

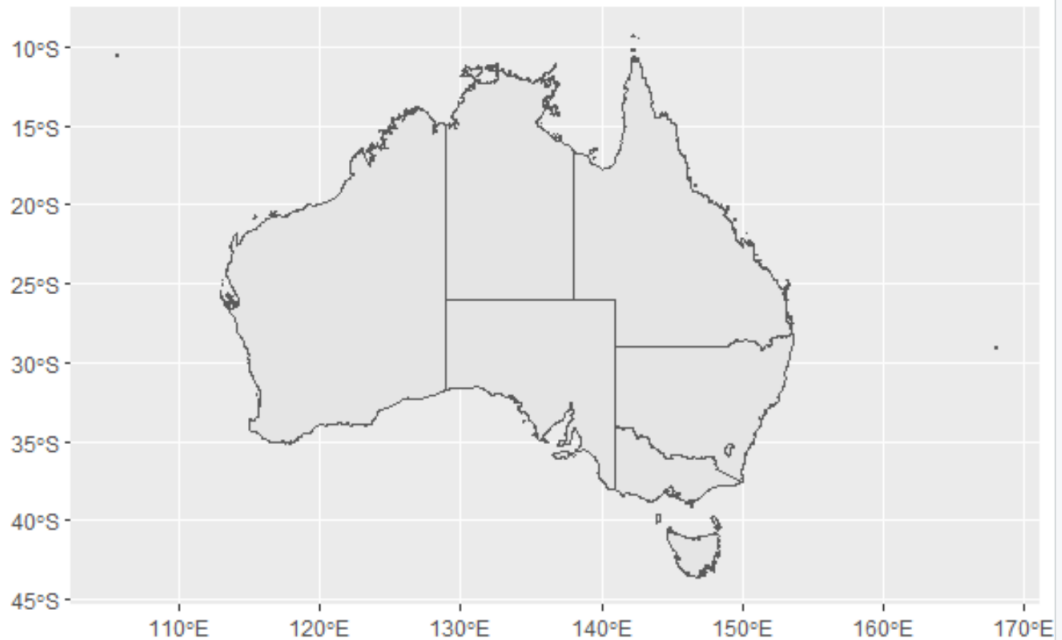
```
install.packages("ozmaps")
install.packages("sf")
library(ozmaps)
library(sf)
oz_states <- ozmaps::ozmap_states
oz_states
```

```
      <chr>                                <POLYGON [area: degree^2]>
1 New South Wales      (((150.7016 -35.12286, 150.6611 -35.11782, 150.6373 -35.14688, 150.6~
2 Victoria             (((146.6196 -38.70196, 146.6721 -38.70259, 146.6744 -38.71385, 146.6~
3 Queensland           (((148.8473 -20.3457, 148.8722 -20.37575, 148.8515 -20.38094, 148.84~
4 South Australia      (((137.3481 -34.48242, 137.3749 -34.46885, 137.3805 -34.4973, 137.36~
5 Western Australia    (((126.3868 -14.01168, 126.3625 -13.98264, 126.3765 -13.97462, 126.4~
6 Tasmania             (((147.8397 -40.29844, 147.8902 -40.30258, 147.8812 -40.32595, 147.8~
7 Northern Territory   (((136.3669 -13.84237, 136.3339 -13.83922, 136.3532 -13.81137, 136.3~
8 Australian Capital Te~ (((149.2317 -35.222, 149.2346 -35.24047, 149.2716 -35.2708, 149.3153~
9 Other Territories     (((167.9333 -29.05421, 167.9188 -29.0344, 167.9313 -29.00187, 167.96~
```

Этот вывод показывает некоторые метаданные, связанные с данными, и говорит нам, что данные по существу представляют собой таблицу с 11 строками и 4 столбцами. Сразу бросается в глаза одно преимущество данных sf: мы легко видим общую структуру данных: Австралия состоит из шести штатов, четырех территорий и острова Маккуори, который политически является частью Тасмании. Существует 11 различных географических единиц, поэтому в этой таблице есть 11 строк (сравните данные `mi_counties`, где есть одна строка на вершину полигона).

	centroid	population	area	density
1 New South Wales	((150.7016 -35.12286, 150.6611 -35.11782, 150.6373 -35.14688, 150.6~	7582956	3239473	23.43
2 Victoria	((146.6196 -38.70196, 146.6721 -38.70259, 146.6744 -38.71385, 146.6~	5628600	227446	24.74
3 Queensland	((148.8473 -20.3457, 148.8722 -20.37575, 148.8515 -20.38094, 148.84~	4753600	1700000	2.79
4 South Australia	((137.3481 -34.48242, 137.3749 -34.46885, 137.3805 -34.4973, 137.36~	1702900	95960	17.74
5 Western Australia	((126.3868 -14.01168, 126.3625 -13.98264, 126.3765 -13.97462, 126.4~	2554400	2500000	1.02
6 Tasmania	((147.8397 -40.29844, 147.8902 -40.30258, 147.8812 -40.32595, 147.8~	543200	9086	59.79
7 Northern Territory	((136.3669 -13.84237, 136.3339 -13.83922, 136.3532 -13.81137, 136.3~	237650	1380000	0.17
8 Australian Capital Te~	((149.2317 -35.222, 149.2346 -35.24047, 149.2716 -35.2708, 149.3153~	237650	2331	102.33
9 Other Territories	((167.9333 -29.05421, 167.9188 -29.0344, 167.9313 -29.00187, 167.96~	13000	13000	1.0

```
ggplot(oz_states) +  
  geom_sf() +  
  coord_sf()
```



Simple features maps

Чтобы понять, почему это работает, обратите внимание, что `geom_sf()` опирается на эстетику геометрии, которая не используется в других частях `ggplot2`. Эта эстетика может быть определена одним из трех способов:

- В простейшем случае (проиллюстрированном выше), когда пользователь ничего не делает, `geom_sf()` попытается сопоставить его со столбцом с именем `geometry`.
- Если аргумент `data` является объектом `sf`, то функция `geom_sf()` может автоматически обнаружить столбец `geometry`, даже если он не называется `geometry`.
- Можно указать отображение вручную обычным способом с помощью `aes(geometry = my_column)`. Это полезно, если в наборе данных есть несколько столбцов геометрии.

```
ggplot(oz_states) +  
  geom_sf() +  
  coord_sf()
```

Layered maps

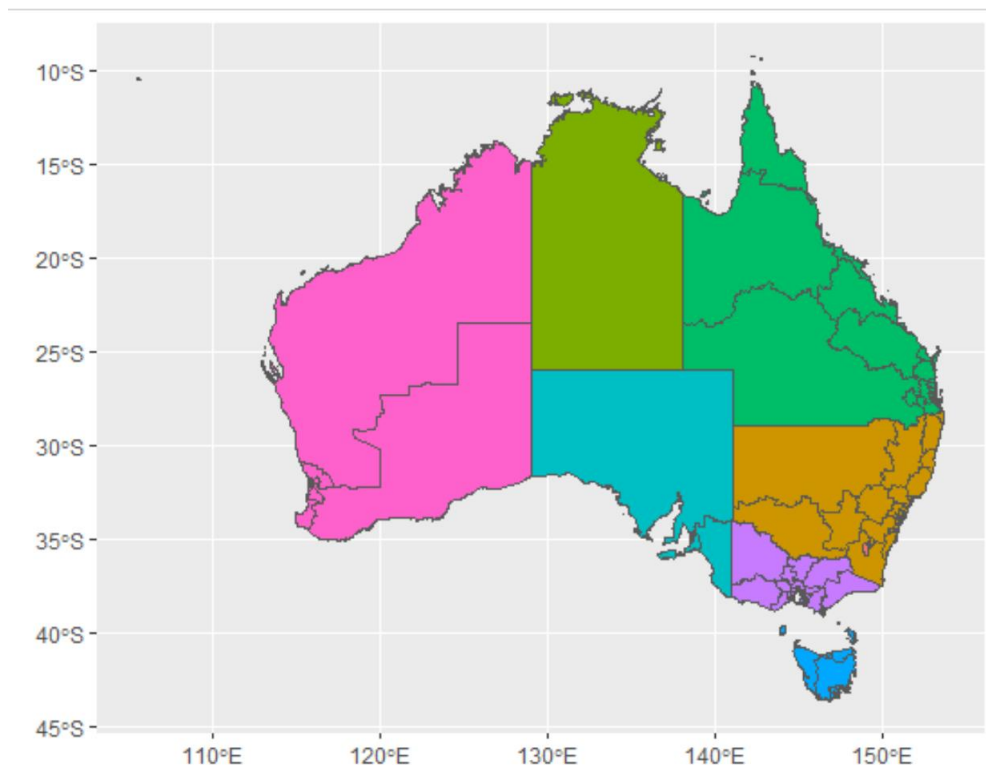
В некоторых случаях вы можете наложить одну карту поверх другой. Пакет `ggplot2` поддерживает это, позволяя добавлять несколько слоев `geom_sf()` на график. В качестве примера я использую данные `oz_states`, чтобы нарисовать австралийские Штаты в разных цветах, а также наложить этот график на границы австралийских избирательных регионов. Для этого необходимо выполнить два этапа предварительной обработки. Во-первых, я использую `dplyr::filter()` для удаления “других территорий” с государственных границ.

Приведенный ниже код рисует график с двумя слоями карты: первый использует `oz_states` для заполнения Штатов разными цветами, а второй использует `oz_votes` для рисования избирательных границ. Во-вторых, я извлеку избирательные границы в упрощенной форме с помощью функции `ms_simplify()` из пакета `rmapshaper` (Teucher and Russell 2018). Как правило, это хорошая идея, если исходный набор данных (в данном случае `ozmaps::abs_ced`) хранится с более высоким разрешением, чем требуется для вашего графика, чтобы сократить время, необходимое для визуализации графика

Layered maps

```
library(dplyr)
library(rmapshaper)
oz_states <- ozmaps::ozmap_states %>% dplyr::filter(NAME != "Other Territories")
oz_votes <- rmapshaper::ms_simplify(ozmaps::abs_ced)
ggplot() +
  geom_sf(data = oz_states, mapping = aes(fill = NAME), show.legend = FALSE) +
  geom_sf(data = oz_votes, fill = NA) +
  coord_sf()
```

Стоит отметить, что первый слой этого графика отображает эстетику заливки на переменную в данных. В этом случае переменная NAME является категориальной переменной и не передает никакой дополнительной информации, но тот же подход может быть использован для визуализации других видов метаданных области. Например, если бы у oz_states был дополнительный столбец, указывающий уровень безработицы в каждом штате, мы могли бы сопоставить эстетику заполнения с этой переменной.



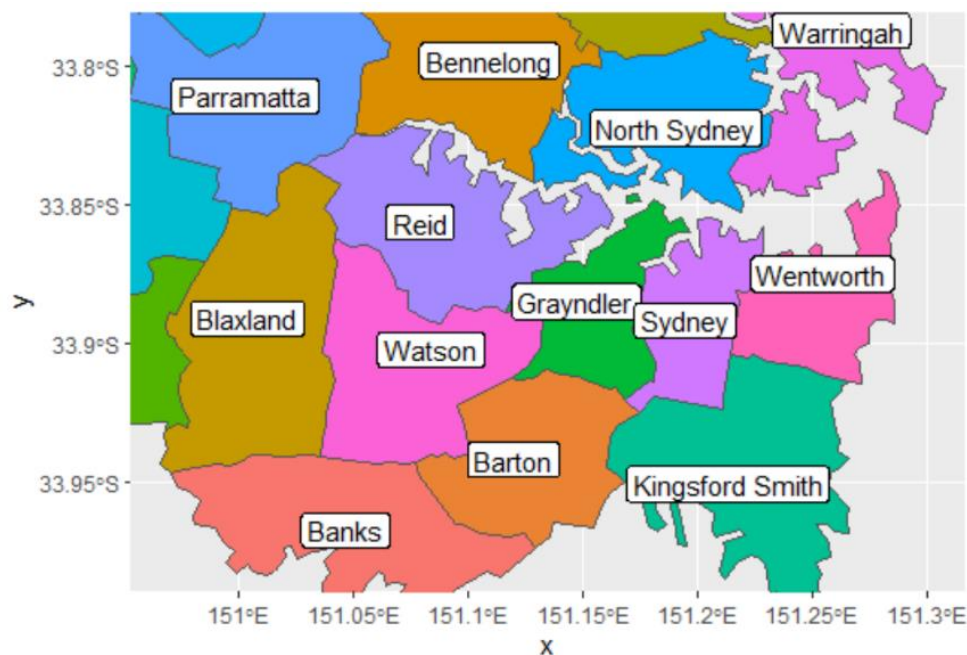
Labelled maps

Добавление меток к картам является примером аннотирования графиков и поддерживается `geom_sf_label()` и `geom_sf_text()`. Например, в то время как австралийская аудитория могла бы разумно ожидать, что она знает названия австралийских штатов, мало кто из австралийцев будет знать названия различных избирательных округов в столичном регионе Сиднея. Таким образом, чтобы нарисовать избирательную карту Сиднея, нам сначала нужно будет извлечь картографические данные для соответствующих электоратов, а затем добавить метку. Приведенный ниже график приближает область Сиднея, указывая `xlim` и `ylim` в `coord_sf()`, а затем использует `geom_sf_label()` для наложения метки на каждый электорат:

Layered maps

```
sydney_map <- ozmaps::abs_ced %>% filter(NAME %in% c(
  "Sydney", "Wentworth", "Warringah", "Kingsford Smith", "Grayndler", "Lowe",
  "North Sydney", "Barton", "Bradfield", "Banks", "Blaxland", "Reid",
  "Watson", "Fowler", "Werriwa", "Prospect", "Parramatta", "Bennelong",
  "Mackellar", "Greenway", "Mitchell", "Chifley", "McMahon"
))
```

```
ggplot(sydney_map) +
  geom_sf(aes(fill = NAME), show.legend = FALSE) +
  coord_sf(xlim = c(150.97, 151.3), ylim = c(-33.98, -33.79)) +
  geom_sf_label(aes(label = NAME), label.padding = unit(1, "mm"))
```



Layered maps

```
#> Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not  
#> give correct results for longitude/latitude data
```

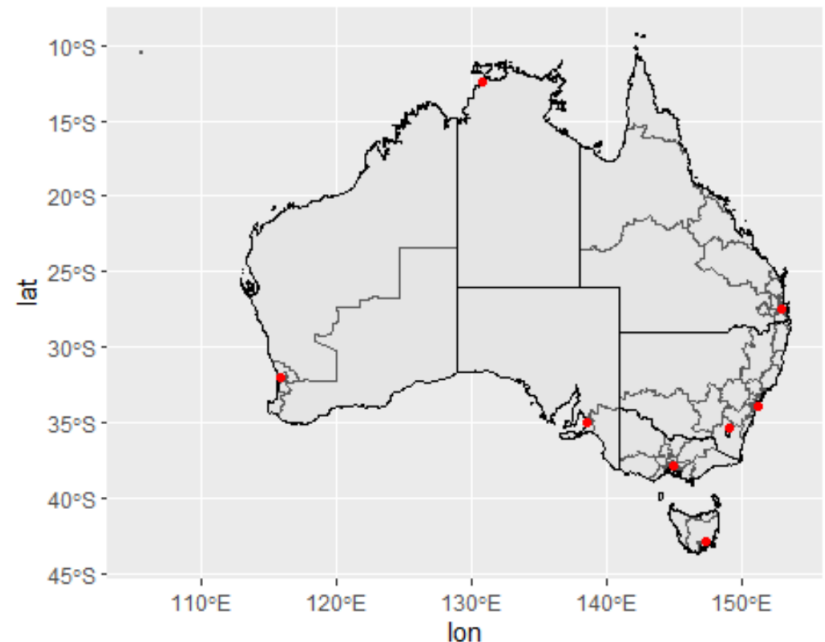
Это предупреждающее сообщение стоит отметить. Внутренне `geom_sf_label()` использует функцию `st_point_on_surface()` из пакета `sf` для размещения меток, и предупреждающее сообщение возникает потому, что большинство алгоритмов, используемых `sf` для вычисления геометрических величин (centroids, interior points), основаны на предположении, что точки лежат на плоской двумерной поверхности и параметризованы Декартовыми координатами. Это предположение не является строго обоснованным, и в некоторых случаях (например, в районах вблизи полюсов) расчеты, рассматривающие долготу и широту таким образом, дают ошибочные ответы. По этой причине пакет `sf` выдает предупреждающие сообщения, когда он полагается на это приближение.

Adding other geoms

Хотя функция `geom_sf()` в некотором смысле является особенной, она, тем не менее, ведет себя почти так же, как и любой другой `geom`, позволяя отображать дополнительные данные на карте со стандартными GEOM. Например, мы можем построить расположение австралийских столиц на карте с помощью функции `geom_point()`.

```
oz_capitals <- tibble::tribble(  
  ~city,    ~lat,    ~lon,  
  "Sydney", -33.8688, 151.2093,  
  "Melbourne", -37.8136, 144.9631,  
  "Brisbane", -27.4698, 153.0251,  
  "Adelaide", -34.9285, 138.6007,  
  "Perth",   -31.9505, 115.8605,  
  "Hobart",  -42.8821, 147.3272,  
  "Canberra", -35.2809, 149.1300,  
  "Darwin",  -12.4634, 130.8456,  
)
```

```
ggplot() +  
  geom_sf(data = oz_votes) +  
  geom_sf(data = oz_states, colour = "black", fill = NA) +  
  geom_point(data = oz_capitals, mapping = aes(x = lon, y = lat), colour = "red") +  
  coord_sf()
```



Map projections

В начале мы рисовали карты, вычерчивая долготу и широту на декартовой плоскости, как будто геопространственные данные ничем не отличались от других видов данных, которые можно было бы построить. В первом приближении это нормально, но этого недостаточно, если вы заботитесь о точности. В этом подходе есть две фундаментальные проблемы.

Первый вопрос - это форма планеты. Земля не является ни плоской плоскостью, ни совершенной сферой. Как следствие, чтобы сопоставить координатное значение (долготу и широту) с местоположением, нам нужно сделать предположения о всевозможных вещах. Насколько эллипсоидальна Земли? Где находится центр планеты? Где находится исходная точка для долготы и широты? А где же уровень моря? Как движутся тектонические плиты? Все эти вещи релевантны, и в зависимости от того, какие предположения вы делаете, одна и та же координата может быть сопоставлена с местами, которые находятся на расстоянии многих метров друг от друга. Набор предположений о форме Земли называется геодезическими данными, и хотя для некоторых визуализаций данных это может не иметь значения, для других это имеет решающее значение.

Есть несколько различных вариантов, которые можно было бы рассмотреть: если вы ориентируетесь на Северную Америку "то" североамериканские данные "(NAD83) - это хороший выбор, тогда как если ваша перспектива глобальна, то" мировая Геодезическая система " (WGS84), вероятно, лучше.

Map projections

Вторая проблема - это форма вашей карты. Земля имеет приблизительно эллипсоидальную форму, но в большинстве случаев ваши пространственные данные должны быть нарисованы на двухмерной плоскости. Невозможно сопоставить поверхность эллипсоида с плоскостью без некоторого искажения или разрезания, и вам придется сделать выбор относительно того, какие искажения вы готовы принять при рисовании карты. Это и есть задача картографической проекции.

Картографические проекции часто классифицируются с точки зрения сохраняемых ими геометрических свойств, например:

Проекции с сохранением площади гарантируют, что области одинаковой площади на земном шаре будут нарисованы с одинаковой площадью на карте.

Сохраняющие форму (или конформные) проекции обеспечивают сохранение локальной формы областей.

И, к сожалению, невозможно, чтобы какая-либо проекция сохраняла форму и сохраняла площадь.

Дополнительные сведения о картографических проекциях см. в разделе Геокомпьютеризация с помощью R <https://geocompr.robinlovelace.net/> (Lovelace, Nowosad, and Muenchow 2019).

Map projections

Взятые вместе, геодезические данные (например, WGS84), тип картографической проекции (например, Mercator) и параметры проекции (например, местоположение начала координат) определяют координатную систему отсчета, или CRS, полный набор допущений, используемых для перевода информации о широте и долготе в двумерную карту.

Объект `sf` часто включает в себя CRS по умолчанию:

```
> st_crs(oz_votes)
```

Coordinate Reference System:

User input: EPSG:4283

wkt:

```
GEOGCS["GDA94",  
  DATUM["Geocentric_Datum_of_Australia_1994",  
    SPHEROID["GRS 1980",6378137,298.257222101,  
      AUTHORITY["EPSG","7019"]],  
    TOWGS84[0,0,0,0,0,0,0],  
    AUTHORITY["EPSG","6283"]],  
  PRIMEM["Greenwich",0,  
    AUTHORITY["EPSG","8901"]],  
  UNIT["degree",0.0174532925199433,  
    AUTHORITY["EPSG","9122"]],  
  AUTHORITY["EPSG","4283"]]
```

Map projections

В ggplot2 CRS управляется функцией `coord_sf()`, которая гарантирует, что каждый слой на графике использует одну и ту же проекцию. По умолчанию `coord_sf()` использует CRS, связанный со столбцом геометрии `data1`. Поскольку данные `sf` обычно предоставляют выбор CRS, этот процесс обычно разворачивается незаметно, не требуя никакого вмешательства со стороны пользователя. Однако если вам нужно установить CRS самостоятельно, вы можете указать параметр `crs`:

```
ggplot(oz_votes) +  
  geom_sf() +  
  coord_sf(crs = st_crs("+proj=lcc +datum=WGS84"))
```

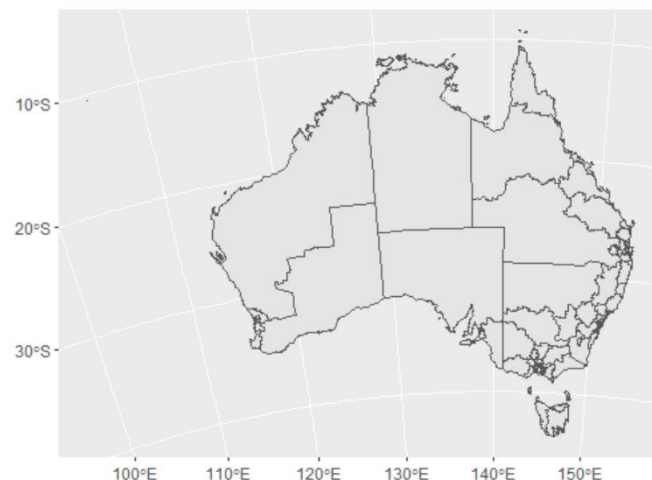


Map projections

В этой карте использованы данные WGS84 и конформную коническую проекцию Ламберта (LCC), которая часто используется в авиационных приложениях, поскольку прямые линии на карте являются приближительными “большими кругами” на земном шаре, и обычно считается хорошей проекцией для региональных карт в средних широтах. Однако карта выглядит неэстетично.

Центрируем карту на долготе 140 и широте -25, а также фиксируем две стандартные параллели (линии, на которых нет искажений) на широтах -18 и -36.

```
crs <- "+proj=lcc +datum=WGS84 +lat_0=-25 +lon_0=140 +lat_1=-18 +lat_2=-36"  
ggplot(oz_votes) +  
  geom_sf() +  
  coord_sf(crs = st_crs(crs))
```



Raster maps

Вместо отображения контекста с векторными границами, вы можете нарисовать традиционную карту. Это называется растровым изображением. Самый простой способ получить растровую карту заданной области - это использовать пакет `ggmap`, который позволяет получать данные из различных онлайн-картографических источников, включая OpenStreetMap и Google Maps. Загрузка растровых данных часто занимает много времени, поэтому рекомендуется кэшировать их в файле `rds`, как показано ниже:

```
if (file.exists("mi_raster.rds")) {  
  mi_raster <- readRDS("mi_raster.rds")  
} else {  
  bbox <- c(  
    min(mi_counties$lon), min(mi_counties$lat),  
    max(mi_counties$lon), max(mi_counties$lat)  
  )  
  mi_raster <- ggmap::get_openstreetmap(bbox, scale = 8735660)  
  saveRDS(mi_raster, "mi_raster.rds")  
}
```

Raster maps

Обратите внимание, что для нахождения подходящего масштаба в этом примере требуется много ручной настройки, поэтому может потребоваться некоторое усилие, чтобы получить нужные данные. Как только у вас есть растровые данные, функция `ggmap()` из пакета `ggmap` позволяет вам создать график, рисующий карту, как показано на левой панели ниже:

```
ggmap::ggmap(mi_raster)
```

```
mi_cities <- tbl_df(maps::us.cities) %>%  
  filter(country.etc == "MI") %>%  
  select(-country.etc, lon = long)
```

```
ggmap::ggmap(mi_raster) +  
  geom_point(aes(size = pop), mi_cities, colour = "red") +  
  scale_size_area()
```

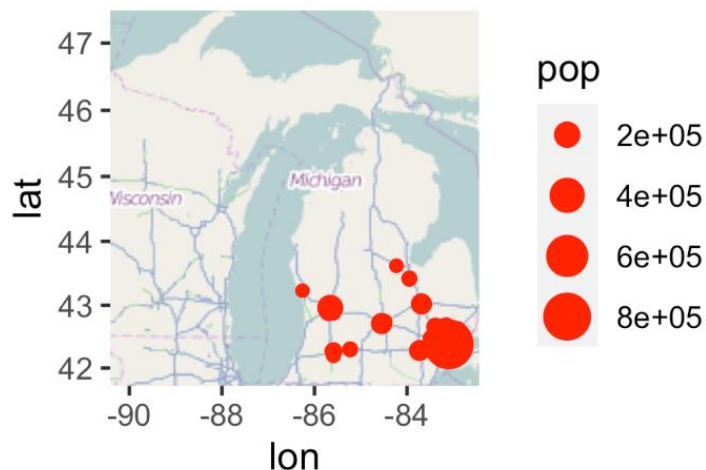
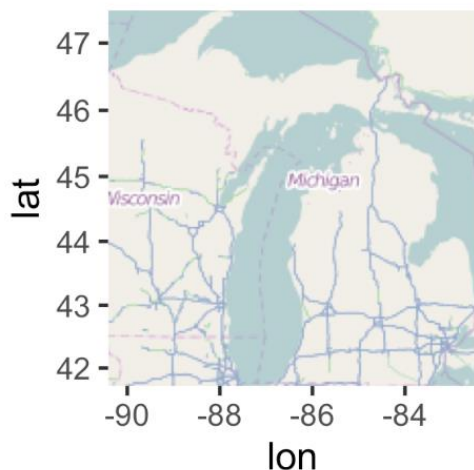
результатом выполнения `ggmap` является объект `ggplot`

Raster maps

```
ggmap::ggmap(mi_raster)
```

```
mi_cities <- tbl_df(maps::us.cities) %>%  
  filter(country.etc == "MI") %>%  
  select(-country.etc, lon = long)
```

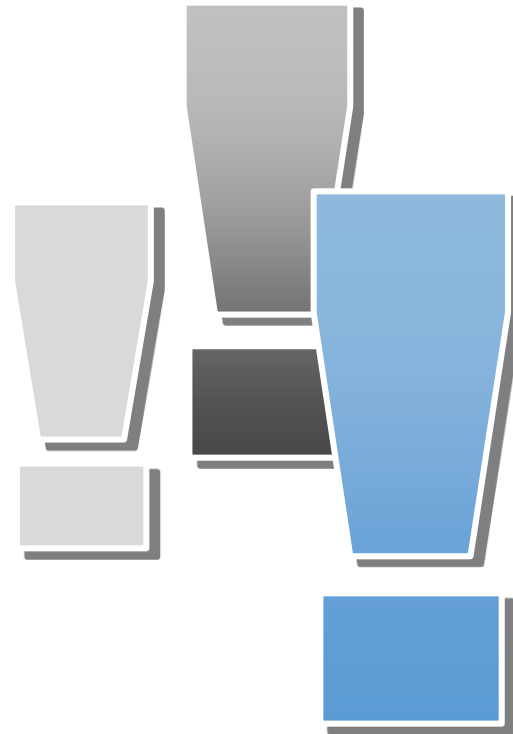
```
ggmap::ggmap(mi_raster) +  
  geom_point(aes(size = pop), mi_cities, colour = "red") +  
  scale_size_area()
```



Data sources

- The USAboundaries package, <https://github.com/ropensci/USAboundaries> contains state, county and zip code data for the US (Mullen and Bratt 2018). As well as current boundaries, it also has state and county boundaries going back to the 1600s.
- The tigris package, <https://github.com/walkerke/tigris>, makes it easy to access the US Census TIGRIS shapefiles (Walker 2019). It contains state, county, zipcode, and census tract boundaries, as well as many other useful datasets.
- The rnaturalearth package (South 2017) bundles up the free, high-quality data from <http://naturalearthdata.com/>. It contains country borders, and borders for the top-level region within each country (e.g. states in the USA, regions in France, counties in the UK).
- The osmar package, <https://cran.r-project.org/package=osmar> wraps up the OpenStreetMap API so you can access a wide range of vector data including individual streets and buildings (Eugster and Schlesinger 2010)
- If you have your own shape files (.shp) you can load them into R with `sf::read_sf()`

Спасибо за внимание!



Шевцов Василий Викторович

shevtsov_vv@rudn.university
+7(903)144-53-57