Cursos Integrados em Vigilância em Saúde

Curso

Análise espacial de dados para a Vigilância em Saúde

Instituto Todos pela Saúde

ABRASCO
ASSOCIAÇÃO BRASILEIRA DE SAÚDE COLETIVA

UNA-SUS
Universidade Aberta do SUS

FIOCRUZ

UFPEL

UFSC

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Reitor Irineu Manoel de Souza
Vice-Reitora Joana Célia dos Passos
Pró-Reitora de Pós-graduação Werner Kraus
Pró-Reitor de Pesquisa e Inovação Jacques Mick
Pró-Reitor de Extensão Olga Regina Zigelli Garcia

CENTRO DE CIÊNCIAS DA SAÚDE

Diretor Fabrício de Souza Neves
Vice-Diretora Ricardo de Souza Magini

DEPARTAMENTO DE SAÚDE PÚBLICA

Chefe do Departamento Rodrigo Otávio Moretti Pires
Subchefe do Departamento Sheila Rúbia Lindner
Coordenadora do Curso Alexandra Crispim Boing

INSTITUTO TODOS PELA SAÚDE (ITPS)

Diretor presidente Jorge Kalil (Professor titular da Faculdade de Medicina da Universidade de São Paulo; Diretor do Laboratório de Imunologia do Incor)

ASSOCIAÇÃO BRASILEIRA DE SAÚDE COLETIVA (ABRASCO)

Presidente Rosana Teresa Onocko Campos

EQUIPE DE PRODUÇÃO

Denis de Oliveira Rodrigues
Kamila de Oliveira Belo
Marcelo Eduardo Borges
Oswaldo Gonçalves Cruz
Alexandra Crispim Boing
Antonio Fernando Boing

Curso

Análise espacial de dados para a Vigilância em Saúde

Dados Internacionais de Catalogação-na-Publicação (CIP)

A532 Análise espacial de dados para a Vigilância em Saúde/ Denis de Oliveira Rodrigues, Kamila de Oliveira Belo, Marcelo Eduardo Borges, Oswaldo Gonçalves Cruz . Santa Catarina ; São Paulo ; Rio de Janeiro : UFSC ; ITPS ; Abrasco; 2022. 87p. (Cursos Integrados em Vigilância em Saúde).

Publicação Online
10.52582/curso-analise-dados-vigilancia-modulo10

1. Vigilância em saúde 2. Análise de dados I. Título

Sumário

Análise Espacial para a Vigilância em Saúde

Neste curso aprofundaremos as análises de bancos de dados de saúde construindo **mapas de interesse epidemiológico**. Na Vigilância em Saúde utilizamos o geoprocessamento como uma ferramenta que permite a análise da distribuição espacial de determinadas doenças, eventos e agravos relacionados à saúde.

Ao produzir mapas temáticos que colorem ou sombreiam um conjunto de áreas de acordo com seus valores, taxas ou coeficientes estamos construindo um mapa epidemiológico. Com ele você poderá identificar, localizar, acompanhar e monitorar populações de maneira a direcionar as ações de Vigilância em Saúde para que sejam executadas com maior efetividade e assertividade!

Ao final deste curso, você será capaz de:

1. conhecer os conceitos básicos para elaboração de mapas;
2. reconhecer dados espaciais em bancos de dados da saúde;
3. transformar e visualizar os principais tipos de dados espaciais utilizando o `R`;
4. gerar alguns dos principais mapas de interesse para a Vigilância em Saúde.

Atenção

Para seguir com este curso, você deve conhecer as ferramentas básicas para uso da linguagem `R` e do `RStudio`, além de possuir conhecimentos básicos de rotinas de análises e visualização de dados utilizando a linguagem de programação `R`. Lembre-se que você pode acessar a qualquer momento o curso **"Análise de dados para a Vigilância em Saúde – curso básico"** obtendo os códigos desejados para a confecção de seus mapas. Caso não tenha feito o curso, sugerimos fortemente que se inscreva nele.

1. Geoprocessamento para Vigilância em Saúde

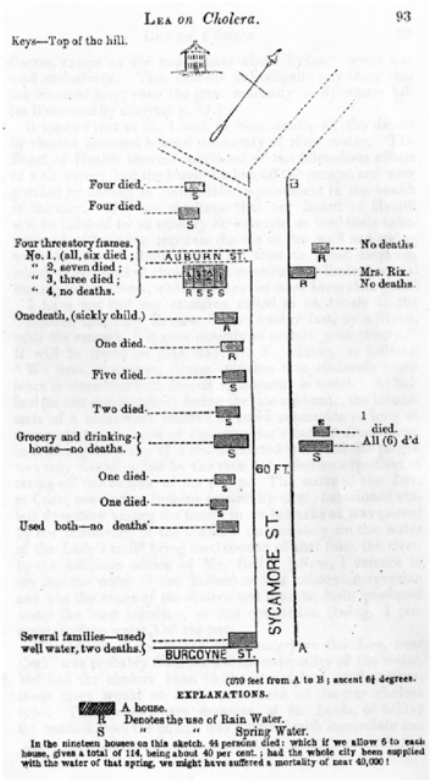
Na Vigilância em Saúde, quando necessitamos analisar o comportamento de uma doença ao longo do tempo e sua distribuição pelas regiões, podemos construir mapas que representem isso. Esses mapas são instrumentos valiosos para a análise epidemiológica quando descrevem, por exemplo, onde estão localizadas as populações mais vulneráveis para um determinado evento ou qual o risco para doenças em uma região de interesse. Além disso, podem revelar elementos do ambiente que determinam algum um agravo à saúde apontando, assim, possíveis associações entre fontes de contaminação e áreas de risco elevado.

Neste cenário, podemos utilizar técnicas voltadas para a coleta, tratamento e visualização de informações que podem ser representadas em um mapa. A esse conjunto de técnicas dá-se o nome de **geoprocessamento**. Na área da saúde, o uso do geoprocessamento tem buscado compreender, entender e estudar a distribuição geográfica de uma determinada doença, agravo ou evento.

Com auxílio das técnicas adequadas de visualização e análise, a epidemiologia teve um grande auxílio na prevenção, na vigilância e no controle de doenças. Inclusive, o uso dessas técnicas na epidemiologia é pioneiro. Você deve se lembrar do mapeamento dos casos de cólera realizados por John Lea e John Snow, em 1849 nos Estados Unidos e 1854 no Reino Unido, respectivamente. Em ambas as situações, foram produzidos mapas localizando os casos de cólera, identificando pontos de suspeita de transmissão da doença.

Figura 1: Mapas produzidos por John Lea (A) e John Snow (B) para análise dos casos de cólera em Cincinnati, Estados Unidos, e Londres, Reino Unido, respectivamente.

A



B

A utilização de técnicas de geoprocessamento é fundamental na epidemiologia, uma vez que permite uma visão abrangente da saúde dos indivíduos no contexto social, histórico, político, cultural e ambiental em que estão inseridos. Atualmente, existem vários *softwares* que apoiam as análises espaciais como o TabWin, o ArcGis, o Qgis, o Google Maps e o Google Earth. Neste curso, vamos apresentar métodos básicos para tratar dados na área da Vigilância em Saúde utilizando a linguagem de programação `R`.

Muitas vezes, no dia a dia, analisar dados de vigilância e tratá-los para inserir em um mapa, pode ser desafiador. Automatizar esta etapa qualificará sua rotina de detecção precoce de epidemias, diminuindo horas de trabalho afinco. O `R` é uma ferramenta poderosa para essas tarefas.

Mas, antes de iniciar a trabalhar com o `R`, precisamos entender um pouco mais de alguns conceitos. Vamos lá?

2. O dado espacial na Vigilância em Saúde

A análise espacial é muito utilizada para avaliação dos determinantes de saúde e doença pela Vigilância em Saúde. Organizar dados espaciais ao longo do tempo pode evidenciar e caracterizar fenômenos de saúde vinculados ao local geográfico onde ocorrem. É importante ressaltar que, para a epidemiologia, os processos de saúde e doença são invariavelmente afetados pelos determinantes socioambientais. Dessa forma, para o bom desenvolvimento das ações de vigilância, é fundamental determinar se eventos podem ter desfechos diferentes a depender da sua localização.

Para entender o contexto dos dados espaciais na Vigilância em Saúde, vamos falar um pouco sobre quais os dados espaciais disponíveis na Vigilância em Saúde, o nível de detalhamento e a sua classificação geral.

Neste curso, os termos **espaço** e **espacial** se referem aos objetos geográficos do mundo real, como ruas, hospitais, casas, rios, bairros, e sua localização. Tudo que está na superfície terrestre possui uma posição, uma referência espacial. Muitas vezes, esta posição é coletada por meio de endereços e convertidos em códigos (coordenadas) que registram a localização. Veremos mais à frente com detalhes. Não se preocupe.

2.1 Quais os dados espaciais da Vigilância em Saúde?

Uma pergunta que pode aparecer quando trabalhamos com mapas é: **"Onde estão os dados espaciais na vigilância?"**.

Na Vigilância em Saúde, os dados espaciais podem ser encontrados nas localizações dos eventos ou dos acometidos pelos eventos. Geralmente, são registrados como o endereço de ocorrência ou de residência de um caso ou como a região de nascimento e morte. É também possível o registro por meio do CEP ou da área de cobertura das ações de uma unidade básica de saúde. Você, no dia a dia de sua vivência, colabora para a produção desses dados.

Os campos de registro para estes dados estão presentes em, praticamente, todos os formulários da saúde como prontuários, fichas de notificação de casos e registros de vacinação. Também os sistemas de informação da saúde possuem campos para esses registros. Mas, no cenário atual, esses dados são preenchidos em campos abertos, sem padronização. Dessa forma, na hora da digitação, há muitas chances de inconsistências e erros, que necessitam de um árduo trabalho para serem corrigidas.

De modo geral, esses dados integram a análise de situação de saúde enquanto uma categoria para explorar padrões relativos ao espaço. Nesse sentido, a Vigilância em Saúde pode:

- realizar o mapeamento de doenças, análise clássica que compara onde os fenômenos ocorreram e busca identificar áreas de risco mais alto,
- fomentar estudos que revelam padrões que não eram previamente reconhecidos e ajudar a formular hipóteses sobre a causa dos fenômenos,
- analisar padrões na ocorrência de uma doença e investigar os mecanismos que a causam,
- coletar dados espaciais, e
- orientar políticas e ações para controle e prevenção de doenças, agravos e eventos relacionados à saúde.

2.2 Nível de detalhe dos dados espaciais que podem ser utilizados na Vigilância em Saúde

Quando citamos nível de detalhe, estamos nos referindo à disponibilidade de dados para análise conforme a necessidade do analista. É, portanto, um dos fatores mais decisivos para que alguma análise seja feita. Sem dados disponíveis, não há como a elaborar algum material que represente adequadamente o fenômeno de interesse, seja na escala espacial ou na escala temporal.

De modo geral, os microdados com alta frequência de atualização são preferíveis. Microdados são aquelas bases de dados onde cada linha se refere ao registro de indivíduo apenas, com suas características demográficas gerais e clínicas, mas devidamente anonimizado.

Alguns anos atrás, os sistemas de informação em saúde não dispunham de exportação eficiente que permitissem análises elaboradas com microdados. Além disso, os computadores eram menos potentes e restritos a equipes técnicas em grandes centros. Na grande maioria dos municípios brasileiros não havia capacidade instalada para tal pretensão.

Atualmente, há muitos avanços na área de ciência de dados que refletem diretamente na área da saúde. Dados em escalas mais finas (os microdados) são disponibilizados semanalmente e APIs oferecem uma forma de acesso ágil (API do inglês *Application Programming Interface*, que se refere a um conjunto de métodos e protocolos de comunicação via web, principalmente).

Abaixo, listamos os principais sistemas de informação com as escalas temporais e espaciais disponíveis que podem dar suporte às análises espaciais.

Tabela 1: Nível de detalhe dos principais sistemas de informação em saúde.

Sistema de Informações saúde	Descrição do evento de registro	Dados finais Disponíveis	Dados preliminares	Unidade territorial de acesso público	Unidade territorial disponível no nível local
SIM	Óbitos	1979 - 2020	2021	Microdado anonimizado, Agregado (Município, Unidade de federação, Regiões)	Microdado
SINASC	Nascidos vivos	1994 - 2020	2021	Microdado anonimizado, Agregado (Município, Unidade de federação, Regiões)	Microdado
SI-PNI	Doses aplicadas e cobertura vacinal	1994 - 2022	-	Agregado (Município, Unidade de federação, Regiões)	Microdado
SI-PNI Covid19	Doses aplicadas contra Covid-19	2021 - 2022	-	Microdado anonimizado	Microdado
CNES	Estabelecimentos, Recursos Físicos, Recursos Humanos, Equipes de Saúde	2005 - 20221[1]	-	Microdado anonimizado, Agregado (Município, Unidade de federação, Regiões)	Microdado
SIH	Internações hospitalares pagas pelo SUS	1992 - 2022	-	Microdado anonimizado	-
POPULACAO	Censo, contagem, projeções interecensitárias, estimativas populacionais	1980 - 2021	-	Setor censitário, Agregado (Município, Unidade de federação, Regiões)	-

Sistema de Informações saúde	Descrição do evento de registro	Dados finais Disponíveis	Dados preliminares	Unidade territorial de acesso público	Unidade territorial disponível no nível local
SINAN	Acidente de trabalho com material biológico, Acidente de trabalho, Acidente por Animais Peçonhentos, Atendimento Antirrabico, Botulismo, Câncer relacionado ao trabalho, Doença de Chagas Aguda, Febre de Chikungunya, Cólera, Coqueluche, Dengue, Dermatoses ocupacionais, Difteria, Esquistossomose, Febre Amarela, Febre Maculosa, Febre Tifóide, Hanseníase, Hantavirose, Hepatites Virais, Intoxicação Exógena, Influenza Pandêmica, Leishmaniose Visceral, Leptospirose, LER/ Dort, Leishmaniose Tegumentar Americana, Malária, Meningite, Transtornos mentais relacionados ao trabalho, Perda auditiva por ruído relacionado ao trabalho, Peste, Paralisia Flácida Aguda, Pneumoconioses realacionadas ao trabalho, Raiva, Sífilis Adquirida, Sífilis Congênita, Sífilis em Gestante, Tétano Acidental, Tétano Neonatal, Tuberculose, Violência doméstica, sexual e/ou outras violências, Zika Vírus	2001 - 20172	2018 - 20222	Microdado anonimizado, Agregado (Município, Unidade de federação, Regiões)	Microdado
SIVEP-Gripe	Casos e óbitos por Sindrome Respiratória Aguda Grave	2021 - 2022	-	Microdado anonimizado	Microdado
API InfoDengue	Dados de dengue, chikungunya e zika, integrados com dados de redes sociais, dados climáticos e dados epidemiológicos	2007 - 2022	-	Agregado (Município, Unidade de federação, Regiões)	-

[1] Variado conforme o recurso
[2] Variado conforme o agravo

2.3 Como os dados espaciais são classificados?

Os eventos na área da saúde que são mais usados nas análises espaciais na vigilância são, geralmente, classificados em:

- **Eventos pontuais**: são aqueles que a própria localização já é uma variável de interesse. São registrados individualmente e podem ter diversas outras características associadas (também chamados de atributos, mas não confundir com atributos de objetos do `R`). São exemplos as localizações de casos de doenças, óbitos, focos de vetores e unidades de saúde. Geralmente, somente o nível local tem o acesso a esses dados.

- **Eventos agregados em áreas territoriais**: geralmente, são aqueles dados originalmente individuais, mas que foram agregados segundo alguma unidade territorial, após um cálculo que normaliza a quantidade de eventos com a população daquela unidade. Podemos exemplificar como a taxa de incidência de uma doença por bairros de residência do paciente ou área de abrangência das equipes da Estratégia de Saúde da Família.

Um outro tipo muito comum utilizado nas análises de vigilância ambiental, são os eventos distribuídos de maneira contínua, ou seja, não restritos a um local específico. São exemplos a temperatura, a pluviosidade e a altitude. Por serem contínuos na superfície, sua medição se dá através de pontos coletados por amostra e com posterior estimativa para outros locais onde não houve coleta.

O analista da Vigilância em Saúde se depara o tempo todo com a disponibilidade de dados para esses eventos. O acesso aos dados produzidos no nível local é acessível às equipes de analistas deste nível. Mas, frequentemente, há a necessidade de integrar dados de outros setores (como os dados sobre internações).

3. Criando um mapa

Você se lembra do mapa de papel com alfinetes representando casos espalhados pelo município? Ele também é um exemplo de representação espacial. A representação espacial de evento de saúde é o processo de criar símbolos que representem as ruas, os bairros ou os casos acometidos por alguma doença e possam ser interpretados pelas pessoas. Os mapas são extremamente úteis e bem difundidos para esta tarefa.

Mas, o que seria um mapa? Mapa é um meio de comunicação gráfica, visual ou até tátil que representa algum fenômeno do mundo real. Tudo o que acontece na superfície terrestre pode ser espacializado e representado. O produto da transformação do dado espacial coletado, processado, editado e integrado em uma visualização com nível de detalhe específico é um mapa.

No `R`, a criação de mapas pode utilizar vários pacotes. Vamos praticar com os principais, como o pacote `sf`, e outros que tenham maior simplicidade para manejo, como os pacotes `mapsf` e `tmap`. Mas não se preocupe, falaremos deles mais à frente. Neste momento, precisaremos instalá-los e carregá-los. Execute o código abaixo no seu computador:

```
# Instalando e carregando os pacotes necessários

if(!require(tidyverse)) install.packages("tidyverse");library(tidyverse)
if(!require(foreign)) install.packages("foreign");library(foreign)
if(!require(readxl)) install.packages("readxl");library(readxl)
if(!require(janitor)) install.packages("janitor");library(janitor)
if(!require(stringi)) install.packages("stringi");library(stringi)
if(!require(sf)) install.packages("sf");library(sf)
if(!require(mapsf)) install.packages("mapsf");library(mapsf)
if(!require(tmap)) install.packages("tmap");library(tmap)
if(!require(ggspatial)) install.packages("ggspatial");library(ggspatial)
```

Agora vamos conhecer os elementos básicos que compõem um mapa, como importar arquivos vetoriais e visualizá-los como um mapa no `R`. Acompanhe com atenção os próximos tópicos.

O mapa é uma simplificação da realidade. Por isso, precisamos priorizar qual nível de detalhe expressa melhor a realidade que queremos representar. Esta tarefa não é trivial. É necessário o conhecimento de conceitos importantes para a melhor representação possível de um fenômeno. Um mapa errado pode levar a interpretações e decisões equivocadas, gasto irracional de dinheiro público.

3.1 Elementos básicos na composição de um mapa

Alguns elementos devem fazer parte na elaboração de mapas. Esses elementos são focados na comunicação com o leitor, por meio de informações elementares na representação dos fenômenos. De modo geral, podem ser elementos de comunicação que visam melhor explicar o que está sendo mapeado, apresentando informações sobre o assunto, mas também podem ser elementos cartográficos, para explicar as referências cartográficas utilizadas para sobre a elaboração do mapa. Os principais elementos são:

- **Título:** elemento de comunicação que deve mostrar o assunto, onde e quando o fenômeno representado aconteceu. A depender das regras de elaboração utilizadas para a montagem do mapa ou relatório que o mapa seja incluído (como por exemplo, normas da Associação Brasileira de Normas Técnicas - ABNT, de periódicos científicos ou regras de publicação definidas pelo setor de comunicação) pode conter tamanho de letra e posição específicas;
- **Legenda:** elemento de comunicação de decodificação dos símbolos, relações, cores, formas e texturas utilizadas no mapa, sendo considerado um guia para a leitura. Mais adiante no material serão discutidas formas de classificação a depender do tipo de mapa produzido;
- **Escala:** elemento cartográfico que representa a relação entre o tamanho dos elementos representados em um mapa e o tamanho real. Na área da saúde, geralmente, é representada em sua forma gráfica, por uma linha ou barra subdivida em trechos nos quais cada comprimento significa o valor correspondente na superfície;
- **Seta norte:** elemento de orientação na superfície terrestre oriundo dos pontos cardeais;
- **Fonte dos dados:** descrição da origem dos dados utilizados para a elaboração e digitalização do mapa. A depender das normas de elaboração utilizadas, podem ser apenas citados na seção de métodos de relatórios, por exemplo;

- **Sistema Geodésico de Referência:** é um conjunto de parâmetros para ajustar a forma real da Terra à geometria que melhor representa. Existem diversos sistemas geodésicos que se ajustam melhor de acordo com a região do mundo. No Brasil, desde 2015 o Sistema de Referência Geocêntrico para as Américas (SIRGAS2000) foi definido como padrão.

O sistema de coordenadas é um dos elementos do sistema geodésico. É por meio dele que é possível localizar algo na superfície terrestre. De modo geral, esse posicionamento pode ser feito por meio de dois sistemas:

- O **sistema de coordenadas geográficas**, que utiliza linhas verticais (meridianos) e linhas horizontais (paralelos) posicionadas no globo para definir a localização espacial de um objeto. Essa localização é, então, especificada em graus decimais (ou em graus, minutos e segundos) por meio da longitude e da latitude.
- O **sistema de coordenadas planas**, que utiliza uma projeção em plano no qual uma localização é representada por X e Y, especificada em metros. A projeção mais utilizada na saúde pública é a do sistema Universal Transverso de Mercator (UTM), que utiliza a divisão da Terra em fusos. Com ela, é possível aplicar funções de cálculos de distâncias entre pontos e comprimento de linhas mais facilmente.

Atenção

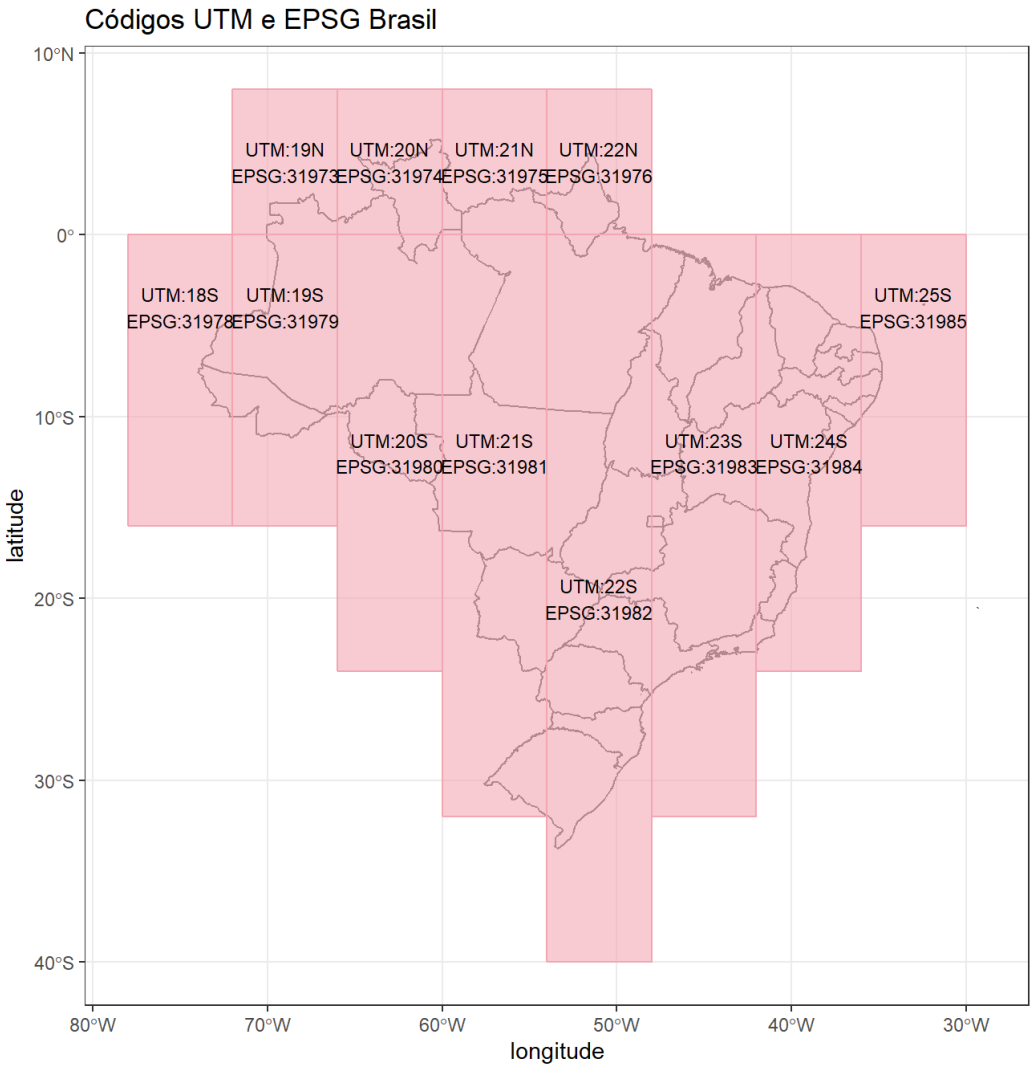
Quando lidamos com diferentes dados, os sistemas de referência podem ser diferentes. Usar referências diferentes resultará em locais em posições diferentes ou arquivos que não se sobrepõem corretamente, ou seja, não são representados adequadamente.

Logo, é uma boa prática utilizar o mesmo sistema de referência em todos os projetos e arquivos geográficos em que se estiver trabalhando.

No R, as coordenadas de ambos os sistemas citados estão indexadas pelo código **EPSG** (sigla em inglês para *European Petroleum Survey Group* que, em português, pode ser traduzido para Grupo Europeu de Pesquisa de Petróleo). Nesta indexação, o sistema geográfico adotado pelo Brasil, o SIRGAS2000, possui o código **4674**. Para o sistema de coordenadas planas UTM há vários códigos, de modo que cada região coberta por um fuso possui um código (Figura 2).

É comum um município adotar um sistema de coordenada como padrão, mas acontece de alguns estarem em áreas limites entre dois fusos. Nesta situação, para elaboração de mapas, é sugerido a adoção de um sistema de coordenadas que utiliza projeção cônica cujo código *EPSG* é 5880.

Figura 2: Limites dos fusos na projeção UTM e respectivos códigos UTM e EPSG do Brasil.



3.1.1 O uso das cores em mapas

As cores despertam respostas emocionais diferentes nas pessoas, a depender do contexto. Por exemplo, ao elaborar um mapa com área de risco para determinada doença, utilizar a cor azul pode passar uma sensação de tranquilidade, além de poder confundir o leitor, fazendo-o pensar em ser uma representação de um rio ou lago. Então a questão é: como escolher as cores adequadamente?

Ao elaborar mapas, o profissional da Vigilância em Saúde deve ter em mente sobre o que cabe destacar no mapa e como a cor contribui com o seu objetivo. Não é uma questão de gosto, mas uma forma de comunicação que utiliza um simbolismo psicológico que pode ser muito eficiente se bem utilizada. A depender do tipo de variável que se deseja utilizar para representação no mapa, o grupo de cores devem ser apropriadas.

Existem muitas maneiras de especificar as cores no `R`. Uma das formas mais usadas é o padrão *RGB* (do inglês: "Red, Green, Blue"), que representa a intensidade das cores primárias vermelho, verde e azul. Uma das codificações desse padrão é o formato chamado **hexadecimal**, formado por seis dígitos, dois para cada cor primária. Esses dois dígitos podem ser representados por números que variam de 00 a 99 e pares de letras de AA até FF (por exemplo AB, AC, CE, EF, FF).

```
R = 00 a 99 / AA a FF
G = 00 a 99 / AA a FF
B = 00 a 99 / AA a FF
```

No `R`, o símbolo (#) nesta situação indica que é um código hexadecimal. Então, um código `#000000` representa a cor preta e `#FFFFFF` representa a cor branca. Combinando as três cores básicas podemos especificar mais de 16 milhões de cores e tons de cinza!

Uma pequena amostra de cores pode ser vista na tabela abaixo, mas é possível visualizar as combinações possíveis clicando em https://www.w3schools.com/colors/colors_hexadecimal.asp. Na tabela, mostramos a cor em cada célula e, também, o código hexadecimal.

Tabela 2: Amostra de cores com codificação hexadecimal.

#FFFFFF	#FFFFEE	#FFFFDD	#FFFFAA	#FFFF50	#FFFF00
#FFDD00	#FFC000	#FFAD48	#FF9900	#D95F0E	#993404
#C7E9C0	#00DD00	#00BB00	#00AA00	#008000	#006000
#00FFFF	#00DDDD	#00AAAA	#008080	#000099	#000060
#00DDFF	#0080FF	#0080D0	#0000FF	#0000DD	#000080
#FEEBE2	#FCC5C0	#FA9FB5	#F768A1	#C51B8A	#7A0177
#FEE5D9	#FCBBA1	#E34A33	#FF0000	#800000	#600000
#FAFAFA	#F2F2F2	#D9D9D9	#BABABA	#404040	#000000

Para a representação de indicadores que expressam valores relativos (razão, taxas), deve-se seguir a lógica de ordenação (menor para maior, maior que, menor que). A escolha mais adequada é representá-los por meio de variação de cores, utilizando paletas de cores mais adequadas e intuitivas:

- sequenciais: são paletas de gradientes que, de modo geral, denotam que cores mais claras representam valores baixos e cores mais escuras representam valores altos. Podem também representar uma gradação de uma única cor, por exemplo do verde claro para o verde escuro (neste caso é chamada matiz única), ou utilizar gradação de cores diferentes, por exemplo do amarelo para o roxo (sendo chamada matiz múltipla). São muito intuitivas na representação de mapas temáticos e muito comuns de escolha em produtos gráficos na Vigilância em Saúde.
- divergentes: também são gradientes, mas possuem um ponto médio mais claro entre duas extremidades escuras destacadas. São usadas para representar variáveis com valores que transitam de um extremo para outro e, principalmente, quando a variável a ser representada possui um valor "central" como, por exemplo:

- zero: taxas de crescimentos que variam de valores negativos a positivos, passando pelo zero;
- porcentagens e proporções: valores que variam de 0 a 1 (ou 0% a 100%);
- meta: indicadores que possuem uma pactuação, uma meta a ser atingida;
- média ou mediana: taxas que apresentem médias gerais ou medianas.

Ambas as paletas podem representar uma gradação de cores contínua (também chamada de não categorizada) e discretizada (também chamada de classificada, escalonada, graduada). Na tabela abaixo, apresentamos alguns exemplos:

Tabela 3. Exemplos de tipos de paletas de cores para dados contínuos.

Dados contínuos	Não categorizados	Categorizados
Sequencial Matiz Múltipla		
Sequencial Matiz Única		
Divergente		

As paletas de cores voltadas para dados categóricos representam uma cor associada a cada categoria de uma variável que **não pode ser ordenada** ou representam dados qualitativos que não podem ser mensurados (somados ou calculados em média). Neste caso, a escolha de cor privilegia a diferenciação clara entre as categorias.

Tabela 4: Exemplo de tipo de paleta para dados qualitativos.

Dados qualitativos	
Categóricos qualitativos	Categoria 1 Categoria 2 Categoria 3 Categoria 4 Categoria 5

No menu lateral “Arquivos” do curso se encontra um arquivo em PDF com as cores e nomes nativas da linguagem `R`. Há vários pacotes que ampliam as opções de cores como `RColorBrewer` e `colorspace`.

3.2 Conhecendo e importando dados vetoriais

Lembramos há pouco dos mapas com alfinetes utilizados para representação de eventos. Atualmente, a representação é muito mais focada no meio digital. Mapas digitais são aqueles desenvolvidos utilizando um conjunto de ferramentas, incluindo programas de computador, editados e disponibilizados em formato digital em páginas online ou em relatórios enviados por *email*, por exemplo.

A representação dos objetos geográficos no meio digital é feita por dados do tipo **gráfico**. Esses dados são assim chamados porque utilizam um conjunto de geometrias para descrever, graficamente, os objetos geográficos que compõem os **arquivos vetoriais**. Três tipos básicos de geometria são utilizados: **pontos**, **linhas** e **polígonos**.

A importação de arquivos vetoriais no `R` pode ser feita utilizando o pacote `sf` (referente às palavras em inglês *simple feature*). O pacote `sf` importa e lê esses arquivos como *dataframes*, sendo as geometrias armazenadas em colunas. Dessa forma, podem ser normalmente manipuladas com pacotes como o `dplyr` e ter visualizações com pacotes como o `ggplot2`.

Vamos conhecer um pouco mais das principais geometrias e, para isso, considere que foi solicitado pelo diretor geral, um mapa contendo as localizações das unidades de saúde e as ruas de acesso até estas unidades. Considere também utilizar dados do Estado do Acre para executar esta tarefa. Acompanhe com atenção os conceitos que seguem.

a) Pontos

Vamos importar dados de pontos contendo a localização das unidades de saúde do Estado do Acre. A geometria de pontos é composta por um conjunto de pares de coordenadas que representam a posição de um evento, estrutura física ou um objeto geográfico qualquer no espaço. Em geral, são usados para indicar um local específico.

Na nossa tarefa, vamos utilizar a função `read_sf()` do pacote `sf` para importar os dados de coordenadas das unidades de saúde {`ac_unidades`}, disponível no menu lateral "Arquivos" do curso, juntamente com os arquivos auxiliares. Importante baixar todos na mesma pasta.

A função `read_sf()` possui como argumento obrigatório apenas o arquivo a ser importado. Como veremos mais à frente, essa função é genérica, pois pode ser usada para importar qualquer geometria (pontos, linhas, polígonos). Agora é com você. Execute o comando abaixo no seu `RStudio`. Lembre-se que os pacotes já devem estar instalados e carregados.

```
# Importando dados vetoriais de pontos com a função read_sf() e
# salvando no objeto `ac_unidades`
ac_unidades <- read_sf('Dados/ac_unidades_saude_m.shp')
```

Vamos aproveitar esta importação para analisar a estrutura do arquivo. Digite o nome do objeto salvo (`ac_unidades`) no Painel Console e veja o *output*:

```
# Analisando o arquivo importado
ac_unidades
```

```
#> Simple feature collection with 1008 features and 4 fields
#> Geometry type: POINT
#> Dimension:     XY
#> Bounding box:  xmin: -73.22469 ymin: -11.029 xmax: -66.64395 ymax: -7.417815
#> Geodetic CRS:  SIRGAS 2000
#> # A tibble: 1,008 × 5
#>    uf    code_state cod_mun cnes_n               geometry
#>    <chr>      <dbl>   <dbl> <chr>             <POINT [°]>
#>  1 AC            12  120035 3591018 (-72.68529 -9.309803)
#>  2 AC            12  120001 3638685      (-66.883 -9.828)
#>  3 AC            12  120001 7245890      (-66.883 -9.828)
#>  4 AC            12  120001 7026641 (-67.05353 -10.07714)
#>  5 AC            12  120001 0257184      (-66.883 -9.828)
#>  6 AC            12  120001 3382745      (-66.883 -9.828)
#>  7 AC            12  120001 3006166      (-66.883 -9.828)
#>  8 AC            12  120001 9639896      (-66.883 -9.828)
#>  9 AC            12  120001 3393984      (-66.883 -9.828)
#> 10 AC            12  120001 0153281      (-66.883 -9.828)
#> # … with 998 more rows
#> # i Use `print(n = ...)` to see more rows
```

Um sumário parecido com o apresentado acima deve ser mostrado no seu `RStudio`. O *output* contém os seguintes elementos:

1. a quantidade de registros e atributos (número de linhas e colunas);
2. o tipo de geometria;
3. os limites geográficos dos dados (também chamado de *bounding box*);
4. o sistema de referência (sigla *CRS*);
5. o objeto geográfico em si (*sf - simple feature*);
6. a coluna do tipo lista contendo a geometria (objeto de classe *sfc* - sigla para *simple feature column*);
7. a geometria individual referente ao objeto geográfico identificado pelo item 5 (objeto de classe *sfg* - sigla para *simple feature geometry*).

Reveja os elementos descritos acima identificados na Figura 3:

Figura 3: Arquivo vetorial contendo geometria de pontos importada.

```
> ac_unidades
Simple feature collection with 1008 features and 4 fields (1)
Geometry type: POINT (2)
Dimension:     XY
Bounding box:  xmin: -73.22469 ymin: -11.029 xmax: -66.64395 ymax: -7.417815 (3)
Geodetic CRS:  SIRGAS 2000 (4)
# A tibble: 1,008 × 5
                                                          (6)
   uf    code_state cod_mun cnes_n              geometry
   <chr>      <dbl>   <dbl> <chr>           <POINT [°]>
(5) 1 AC            12  120035 3591018 (-72.68529 -9.309803) (7)
  2 AC            12  120001 3638685      (-66.883 -9.828)
  3 AC            12  120001 7245890      (-66.883 -9.828)
  4 AC            12  120001 7026641 (-67.05353 -10.07714)
  5 AC            12  120001 0257184      (-66.883 -9.828)
  6 AC            12  120001 3382745      (-66.883 -9.828)
  7 AC            12  120001 3006166      (-66.883 -9.828)
  8 AC            12  120001 9639896      (-66.883 -9.828)
  9 AC            12  120001 3393984      (-66.883 -9.828)
 10 AC            12  120001 0153281      (-66.883 -9.828)
# … with 998 more rows
```

Bem fácil, não é mesmo? Importamos 1008 unidades de saúde, entre postos de saúde, consultórios e hospitais. Vamos agora, importar os demais dados para executar a nossa tarefa.

O formato de arquivo mais utilizado para armazenar dados espaciais é o *shapefile*. Este arquivo é formado por três arquivos básicos que armazenam informações diferentes e devem, obrigatoriamente, permanecerem na mesma pasta:

- ".shp": arquivo com os dados da geometria em si;
- ".shx": arquivo de índice posicional;
- ".dbf": arquivo de atributos das formas representadas pelo *shp*.

Eventualmente, também compõem o conjunto de arquivos *shapefile*:

- ".prj": arquivo de texto que contém a projeção dos dados;
- ".sbn" e ".sbx": índice espacial;
- ".shp.xml": metadados.

Mais recentemente, e devido às limitações do *shapefile*, o uso de arquivo no formato *geopackage* têm aumentado. Diferente do *shapefile*, esse formato, cuja extensão é ".gpkg", é um arquivo único, comporta várias camadas de geometrias diferentes e suporta tamanhos maiores de bases.

Também os arquivos *KML*, sigla que se refere a *Keyhole Markup Language*, têm sido muito utilizados no ambiente das secretarias de saúde. Esse arquivo possui um formato para visualização espacial em aplicativos como Google Earth e Google Maps.

Esses arquivos e muitos outros formatos podem ser lidos no `R` e, inclusive, utilizando a mesma função do pacote `sf`, que será apresentada no próximo tópico.

Para saber mais sobre os tipos de arquivos vetoriais que são aceitos pelo pacote, clique aqui.

b) Linhas

Vamos importar um arquivo vetorial de linhas, {`ac_vias`}, que representam as ruas e rodovias do Estado do Acre, disponível no menu lateral "Arquivos" do curso, juntamente com os arquivos auxiliares. Importante baixar todos na mesma pasta.

Os dados vetoriais de linhas são conceituados como conjuntos de pares de coordenadas conectados, com início e fim definido. As linhas são usadas para representar objetos geográficos que denotam algum comprimento como estradas ou ruas, por exemplo. Também é utilizada em análises para representar distâncias entre pontos, tendência de deslocamentos, entre outros.

Vamos lá, novamente, utilize a função `read_sf` para importar o arquivo {`ac_vias`}. Execute o código abaixo no seu computador.

```
# Importando dados vetoriais de linhas com a função read_sf() e
# salvando no objeto `ac_vias`
ac_vias <- read_sf('Dados/ac_vias.shp')

# Analisando o arquivo importado
ac_vias
```

```
#> Simple feature collection with 18423 features and 14 fields
#> Geometry type: MULTILINESTRING
#> Dimension:     XY
#> Bounding box:  xmin: -73.74049 ymin: -11.1454 xmax: -66.6359 ymax: -7.135078
#> Geodetic CRS:  SIRGAS 2000
#> # A tibble: 18,423 × 15
#>      fid osm_id     code fclass   name  ref   oneway maxsp…¹ layer bridge tunnel
#>    <dbl> <chr>     <dbl> <chr>    <chr> <chr> <chr>    <dbl> <dbl> <chr>  <chr>
#>  1 45637 379159318  5122 residen… Rua … <NA>  B            0     0 F      F
#>  2 45632 379159312  5122 residen… Rua … <NA>  B            0     0 F      F
#>  3 45635 379159315  5122 residen… <NA>  <NA>  B            0     0 F      F
#>  4 45618 379159294  5122 residen… Rua … <NA>  B            0     0 F      F
#>  5 45639 685642583  5122 residen… <NA>  <NA>  B            0     0 F      F
#>  6 45621 379159297  5122 residen… Rua … <NA>  B            0     0 F      F
#>  7 45623 379159299  5122 residen… Rua … <NA>  B            0     0 F      F
#>  8 45625 379159302  5122 residen… Rua … <NA>  B            0     0 F      F
#>  9 45612 379159287  5122 residen… Rua … <NA>  B            0     0 F      F
#> 10 45613 379159288  5122 residen… Rua … <NA>  B            0     0 F      F
#> # … with 18,413 more rows, 4 more variables: NM_MUN <chr>, SIGLA_UF <chr>,
#> #   cod_mun <chr>, geometry <MULTILINESTRING [°]>, and abbreviated variable
#> #   name ¹maxspeed
#> # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Podemos perceber que importamos 18423 linhas entre rodovia e ruas. Mais à frente, veremos como podemos visualizar esses dados, inclusive, sobrepondo com os demais dados importados.

Você percebeu que este arquivo vetorial de linhas possui algumas ruas sem nome? Inclusive, algumas ruas que existem podem não estar sendo representadas. Isso é comum na realidade e deve ser uma limitação a ser superada, buscando arquivos mais atualizados. Parcerias com setor de geoprocessamento do município são úteis para acesso a dados mais atuais ou com menos imperfeições.

c) Polígonos

Para finalizar a importação dos dados para executar nossa tarefa, vamos importar os limites administrativos dos municípios do Acre presentes no arquivo {`ac_municipios`}, disponível no menu lateral "Arquivos" do curso, juntamente com os arquivos auxiliares. Importante baixar todos na mesma pasta. Esses limites utilizam a geometria de um polígono para serem representados.

Polígonos são conjuntos de linhas que possuem, em geral, o mesmo ponto de início e fim, denotando uma estrutura fechada. Representam áreas, territórios, ou uma determinada região de destaque qualquer (por exemplo áreas de risco para alguma doença) ou com características similares (regiões climáticas, biomas).

Utilizando a função `read_sf`, importe o arquivo {`ac_municipios`} seguindo o código abaixo.

```
# Importando dados vetoriais de polígonos com a função read_sf() e
# salvando no objeto `ac_municipios`
ac_municipios <- read_sf('Dados/ac_municipios.shp')

# Analisando o arquivo importado
ac_municipios
```

```
#> Simple feature collection with 22 features and 5 fields
#> Geometry type: POLYGON
#> Dimension:     XY
#> Bounding box:  xmin: -73.99045 ymin: -11.14556 xmax: -66.62368 ymax: -7.111824
#> Geodetic CRS:  SIRGAS 2000
#> # A tibble: 22 × 6
#>    CD_MUN  NM_MUN          SIGLA AREA_KM2 cod_mun                      geometry
#>    <chr>   <chr>           <chr>    <dbl> <chr>                    <POLYGON [°]>
#>  1 1200013 Acrelândia      AC       1812. 120001  ((-67.13181 -9.677307, -66.62…
#>  2 1200054 Assis Brasil    AC       4979. 120005  ((-69.59034 -10.3717, -69.586…
#>  3 1200104 Brasiléia       AC       3928. 120010  ((-69.12866 -10.39925, -69.12…
#>  4 1200138 Bujari          AC       3035. 120013  ((-68.2199 -9.242753, -67.976…
#>  5 1200179 Capixaba        AC       1706. 120017  ((-67.80225 -10.19873, -67.80…
#>  6 1200203 Cruzeiro do Sul AC       8783. 120020  ((-72.62078 -7.538954, -72.58…
#>  7 1200252 Epitaciolândia  AC       1653. 120025  ((-68.6448 -10.61638, -68.643…
#>  8 1200302 Feijó           AC      27977. 120030  ((-70.05145 -7.84816, -69.444…
#>  9 1200328 Jordão          AC       5357. 120032  ((-71.56361 -8.64941, -71.560…
#> 10 1200336 Mâncio Lima     AC       5452. 120033  ((-73.1655 -7.341657, -72.900…
#> # … with 12 more rows
#> # i Use `print(n = ...)` to see more rows
```

Analisando o arquivo importado, percebemos que os 22 municípios do Acre foram importados. Agora, vamos conhecer as principais funções para visualizar os dados espaciais que importamos.

A visualização de dados espaciais é uma atividade muito comum e, às vezes, confundida com a análise espacial. Isso ocorre porque, com um bom produto de visualização, é possível realizar análises que compõem, de fato, uma análise de dados.

Algumas funções do pacote `sf` emitem *outputs* similares como os citados acima como, por exemplo a função `st_geometry()`, que tem como argumento principal apenas o arquivo vetorial a ser avaliado. Repita as linhas a seguir no seu `RStudio` e observe o resultado:

```
# Visualizando informações sobre o objeto ac_municipios
# com a função st_geometry()
st_geometry(ac_municipios)
```

```
#> Geometry set for 22 features
#> Geometry type: POLYGON
#> Dimension: XY
#> Bounding box: xmin: -73.99045 ymin: -11.14556 xmax: -66.62368
ymax: -7.111824
#> Geodetic CRS: SIRGAS 2000
#> First 5 geometries:
#> POLYGON ((-67.13181 -9.677307, -66.62689 -9.898…
#> POLYGON ((-69.59034 -10.3717, -69.58664 -10.371…
#> POLYGON ((-69.12866 -10.39925, -69.12937 -10.40…
#> POLYGON ((-68.2199 -9.242753, -67.97684 -9.3527…
#> POLYGON ((-67.80225 -10.19873, -67.80008 -10.20…
```

Para concluir nossa tarefa de visualizar as localizações das unidades de saúde e as ruas de acesso até estas unidades, vamos aprofundar um pouco mais na visualização dos dados espaciais, pois vamos precisar entender as principais funções que permitem essa visualização e quais detalhes podem melhorar a qualidade do nosso trabalho.

3.2.1 Transformação do sistema de coordenadas

Para seguir com a criação dos mapas, vamos realizar um procedimento muito comum ao trabalhar com arquivos vetoriais: a transformação do sistema de coordenadas.

A transformação do sistema de coordenadas deve ser realizada quando:

- A região geográfica da análise possui uma extensão relativamente pequena e sistema de referência do arquivo vetorial está como sistema de coordenadas geográficas e não planas,
- A análise envolver vários arquivos vetoriais e que, eventualmente, alguns estão em sistemas de coordenadas diferentes,
- Os pacotes utilizados para mapeamento possuem um melhor desempenho com sistemas de coordenadas planas.

Para regiões geográficas relativamente pequenas (extensão de um município ou um pequeno grupo de municípios), o sistema de coordenadas planas é indicado, sendo o sistema UTM o mais comum, já que esses locais estão inclusos em um único fuso.

Como já vimos anteriormente, quando a região de interesse tem uma extensão muito grande e está localizada entre dois ou mais fusos, é indicada a adoção da projeção policônica cujo código *EPSG* é **5880** (Projeção Policônica SIRGAS 2000).

Como podemos perceber na Figura 2, o Estado do Acre está localizado entre os fusos 18 e 19 e, dessa forma, indica que vamos precisar transformar o sistema de coordenadas do nosso arquivo vetorial.

Para verificar qual o sistema de coordenadas estão nossos arquivos, vamos utilizar a função `st_crs()` do pacote `sf`. Essa função possui como argumento principal o nome do arquivo vetorial. Acompanhe o *script* abaixo e reproduza-o no seu `RStudio`:

```
# Verificando o sistema de coordenadas de referência
st_crs(ac_municipios)
```

```
#> Coordinate Reference System:
#>   User input: SIRGAS 2000
#>   wkt:
#> GEOGCRS["SIRGAS 2000",
#>     DATUM["Sistema de Referencia Geocentrico para las AmericaS 2000",
#>         ELLIPSOID["GRS 1980",6378137,298.257222101,
#>             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["Horizontal component of 3D system."],
#>         AREA["Latin America - Central America and South America - onshore and
offshore. Brazil - onshore and offshore."],
#>         BBOX[-59.87,-122.19,32.72,-25.28]],
#>     ID["EPSG",4674]]
```

Perceba que na primeira linha já é mostrado o sistema de referência e na última linha mostra o código *EPSG*. O nosso arquivo, portanto, se encontra no sistema de coordenadas geográficas SIRGAS 2000, código 4674.

Para transformar do sistema de coordenadas geográficas, código 4674, para coordenadas planas, código 5880 vamos utilizar a função `st_transform()` do pacote `sf`. Os argumentos principais dessa função são:

- `x` = arquivo vetorial importado pelo pacote `sf`. Nesse caso, vamos utilizar {`ac_municipios`};
- `crs` = sistema de referência de destino. Neste argumento são aceitos códigos *EPSG*.

Acompanhe o código abaixo e reproduza no computador:

```r
# Transformando o sistema de coordenadas do objeto `ac_municipios`
# com a função st_transform()
ac_municipios_5880 <- st_transform(x = ac_municipios, crs = 5880)


# Verificando a trasnformação do
# sistema de coordenadas de referência
st_crs(ac_municipios_5880)
```

```
#> Coordinate Reference System:
#>   User input: EPSG:5880
#>   wkt:
#> PROJCRS["SIRGAS 2000 / Brazil Polyconic",
#>     BASEGEOGCRS["SIRGAS 2000",
#>         DATUM["Sistema de Referencia Geocentrico para las AmericaS 2000",
#>             ELLIPSOID["GRS 1980",6378137,298.257222101,
#>                 LENGTHUNIT["metre",1]]],
#>         PRIMEM["Greenwich",0,
#>             ANGLEUNIT["degree",0.0174532925199433]],
#>         ID["EPSG",4674]],
#>     CONVERSION["Brazil Polyconic",
#>         METHOD["American Polyconic",
#>             ID["EPSG",9818]],
#>         PARAMETER["Latitude of natural origin",0,
#>             ANGLEUNIT["degree",0.0174532925199433],
#>             ID["EPSG",8801]],
#>         PARAMETER["Longitude of natural origin",-54,
#>             ANGLEUNIT["degree",0.0174532925199433],
#>             ID["EPSG",8802]],
#>         PARAMETER["False easting",5000000,
#>             LENGTHUNIT["metre",1],
#>             ID["EPSG",8806]],
#>         PARAMETER["False northing",10000000,
#>             LENGTHUNIT["metre",1],
#>             ID["EPSG",8807]]],
#>     CS[Cartesian,2],
#>         AXIS["easting (X)",east,
#>             ORDER[1],
#>             LENGTHUNIT["metre",1]],
#>         AXIS["northing (Y)",north,
#>             ORDER[2],
#>             LENGTHUNIT["metre",1]],
#>     USAGE[
#>         SCOPE["Topographic mapping (small scale)."],
#>         AREA["Brazil - onshore and offshore. Includes Rocas, Fernando de Noronha
archipelago, Trindade, Ihlas Martim Vaz and Sao Pedro e Sao Paulo."],
#>         BBOX[-35.71,-74.01,7.04,-25.28]],
#>     ID["EPSG",5880]]
```

Pronto! É um procedimento relativamente simples, não é mesmo? Agora, vamos seguir com a elaboração dos mapas temáticos, abordando primeiramente o mapa base, a primeira camada da nossa sobreposição. Vamos lá?

3.3 Visualizando dados espaciais

Vários pacotes no `R` são úteis para visualizar dados espaciais. Alguns são mais genéricos, tendo a visualização de dados espaciais como um dos itens contemplados na visualização de dados, como é o caso do `ggplot2`. Outros possuem funções que abordam de forma mais específica a visualização de mapas com temas (mapas temáticos), como é o caso do pacote `tmap` e `mapsf`.

Atualmente, um pacote que funciona como uma extensão para o `ggplot2` chamado `ggspatial` auxilia em alguns pontos na padronização da visualização de mapas. Mas, se o objetivo é apenas uma visualização rápida, a função `plot()`, nativa do `R`, pode auxiliar bastante. Vamos conhecer alguns deles no próximo tópico.

3.3.1 Usando Função base *plot()*

Agora que já vimos os elementos básicos para um mapa e como utilizar as cores para melhorar a qualidade do nosso produto, vamos iniciar a entender como visualizar os arquivos de dados espaciais.

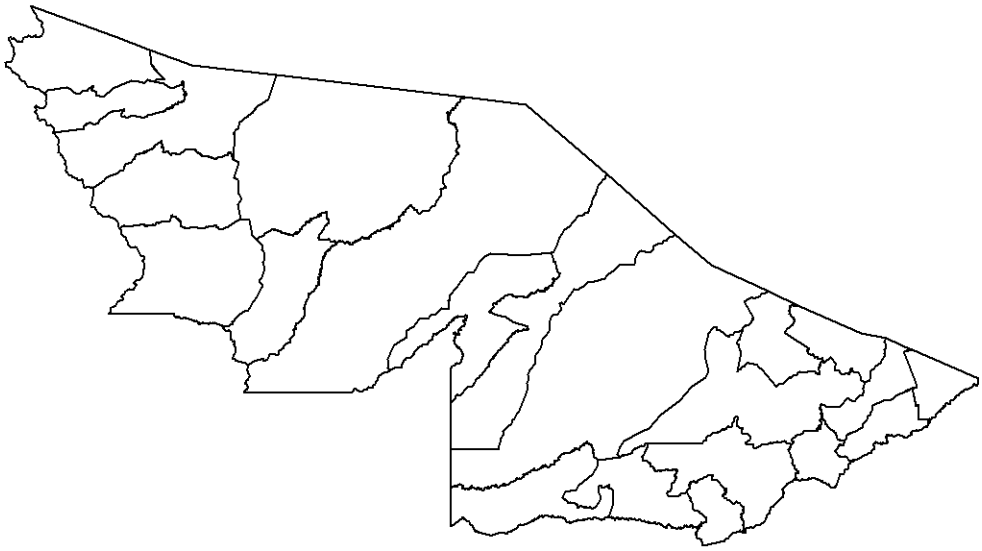
Vamos iniciar com um arquivo que importamos anteriormente, o {`ac_municipios`}, disponível no menu lateral "Arquivos" do curso, juntamente com os arquivos auxiliares. Importante baixar todos na mesma pasta.

Esse arquivo contém dados vetoriais de polígonos, os limites dos municípios do Acre. Contudo, na rotina do trabalho da vigilância no nível municipal, pode haver arquivos que representam bairros, localidades de trabalho da equipe de endemias, áreas de abrangência da estratégia de saúde da família, setor censitário, regiões administrativas ou áreas de planejamento. O funcionamento das funções apresentadas é o mesmo.

Com a função `plot()`, que vamos utilizar agora, o argumento essencial (`x`) é o objeto a ser visualizado. É possível combinar com a função `st_geometry()` do pacote `sf` para visualizar apenas a geometria, sem os outros campos. Faça a plotagem do objeto no seu `RStudio` com o comando a seguir:

```
# Plotando e visualizando o objeto `ac_municipios`
plot(x = st_geometry(ac_municipios))
```

Figura 4: Visualizando os limites dos municípios do Acre com a função `plot()`.



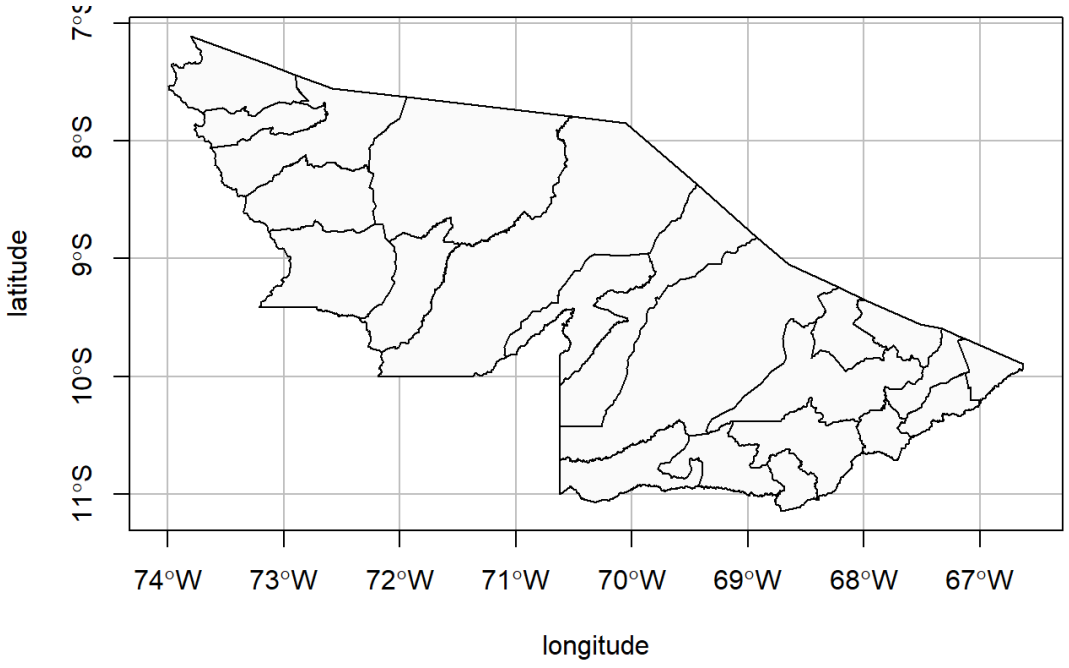
Bem fácil, não é mesmo? Contudo, essa visualização é útil para atender a objetivos que não são necessários elementos de comunicação, como os apresentados no item 3.1. Mas, para melhorar a qualidade do nosso mapa, alguns outros argumentos podem ser utilizados como, por exemplo:

- `axes`: plota as coordenadas nos eixos x e y;
- `graticule`: plota uma grade que acompanha as coordenadas;
- `col`: define a cor de preenchimento (no caso de polígonos);
- `xlab` e `ylab`: define os rótulos dos eixos;
- `main`: apresenta um título centralizado no topo do mapa.

Agora, vamos utilizar estes elementos e criar um mapa mais completo, embora ainda sem todos os elementos. Execute as linhas a seguir em seu `RStudio`:

```
# Plotando e visualizando o objeto ac_municipios
plot(

  # Definindo o objeto que será utilizado para a visualização do mapa
  x = st_geometry(ac_municipios),

  # Definindo como verdadeiro o argumento para plotagem das coordenadas
  # nos eixos do gráfico
  axes = TRUE,

  # Definindo como verdadeiro o argumento para plotagem da grade com as
  # coordenadas
  graticule = TRUE,

  # Definindo a cor de preenchimento (código hexadecimal)
  col = "#FAFAFA",

  # Definindo os rótulos dos eixos x e y
  xlab = "longitude",
  ylab = "latitude"
)
```

Figura 5: Limites dos municípios do Estado do Acre.



Na Figura 5 vemos, portanto, os municípios do Estado do Acre. Percebe-se que o Estado do Acre se localiza ao extremo oeste do País, dado seu intervalo de longitude (67º a 74º, oeste, representado pela letra W), e no norte do país, dado seu intervalo de latitude (7º a 11º sul, representado pela letra S).

Agora vamos utilizar o pacote `ggplot2` para obtermos uma melhor qualidade e finalizar nossa tarefa. Veja em seguida e acompanhe com atenção.

3.3.2 Usando o pacote ggplot2

O pacote `ggplot2` é um poderoso recurso para visualização de dados. Foi criado por Hadley Wickham e faz parte do universo do `tidyverse`, um metapacote projetado com a finalidade de apoiar as análises na área de ciência de dados. Esse pacote implementa um sistema de gráficos baseado em uma gramática própria, permitindo a criação de gráficos avançados.

Essa gramática se encarrega de diversos detalhes que precisariam ser especificados se fôssemos usar a função `plot()` do módulo base do `R`. Ou seja, é mais fácil de utilizar. Como também é estruturado para sobreposição de camadas, traduz muito bem o comportamento de *softwares* de informação geográfica. Além disso, integra muito bem com os demais pacotes do `tidyverse` e possui várias extensões que potencializam seu uso.

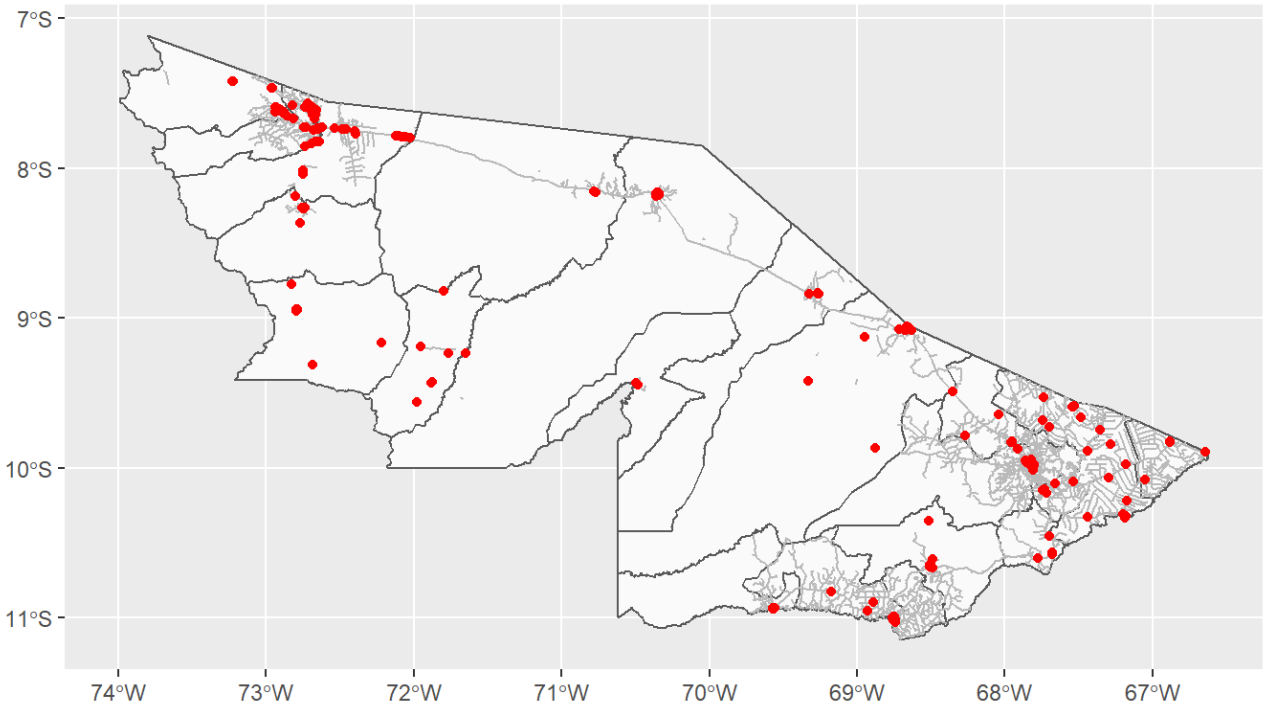
Vamos lá! Agora vamos sobrepor as camadas de arquivos importados anteriormente (unidades de saúde, ruas e limites de municípios) para completar nossa tarefa. Para isso, vamos utilizar a função `geom_sf()` que plota camadas de objetos importados pelo pacote `sf`. Essa função tem como argumento obrigatório `data`, que se refere ao objeto `sf` importado. Vamos utilizar também argumentos estéticos como:

- `fill`: cor do preenchimento quando o objeto for composto por polígonos e,
- `color`: cor das linhas e pontos, quando o objeto for compostos por estas geometrias.

Execute o seguinte código no seu computador e observe o mapa que é gerado:

```
# Utilizando a função ggplot() para plotagem dos mapas
ggplot() +

  # Plotando o objeto de polígono com os contornos do estado
  # O argumento "fill" define a cor de preenchimento
  geom_sf(data = ac_municipios, fill = "#FAFAFA") +

  # Plotando o objeto de linhas para visualização de estradas
  # O argumento "color" define a cor das linhas
  geom_sf(data = ac_vias, color = "#BABABA") +

  # Plotando o objeto de pontos para visualização das unidades de saúde
  # O argumento "color" define a cor de preenchimento
  geom_sf(data = ac_unidades, color = "#FF0000")
```

Figura 6: Exemplo de sobreposição de arquivos vetoriais utilizando o pacote `ggplot2`.



Ficou muito bom, não é mesmo? Está quase pronta nossa tarefa. Perceba que usamos códigos hexadecimais para definir as cores do preenchimento dos polígonos e das ruas. Além disso, o `ggplot2` possui um conjunto de configurações contidos em temas que padronizam a cor do fundo e das linhas da latitude e longitude. Alguns dos temas disponíveis são:

- `theme_grey()` (ou `theme_gray()`): tema padrão (como o visto acima);
- `theme_bw()`: uma variação do padrão que usa fundo branco e linhas cinzas;
- `theme_linedraw()`: um tema com apenas linhas pretas de várias larguras sobre fundo branco;
- `theme_light()`: semelhante ao anterior, mas com linhas e eixos cinza claro, para dirigir mais atenção para os dados;
- `theme_dark()`: "o primo escuro" de `theme_light()`, com linhas de larguras semelhantes, mas com um fundo escuro. Útil para fazer saltar linhas de cores finas;
- `theme_minimal()`: um tema minimalista, sem anotações ou fundo;
- `theme_classic()`: Um tema de aspecto "clássico", com linhas de eixo x e y destacado e nenhuma linha de grade (as linhas internas do gráfico);
- `theme_void()`: um tema completamente vazio, apresentando somente o mapa.

Vamos utilizar o tema `theme_bw()` no nosso próximo mapa. Acompanhe com atenção.

Antes de analisarmos o mapa, vamos seguir os padrões cartográficos apresentados neste curso. Agora vamos utilizar o pacote `ggspatial` para compor os elementos como escala e seta de norte. Para isso, vamos utilizar as funções `annotation_scale()`, que sobrepõe uma escala gráfica no mapa e `annotation_north_arrow()`, que sobrepõe uma seta de norte. Os argumentos dessas funções são:

- `location`: posição do elemento. Neste argumento podem ser usadas as siglas abaixo:
  - *tr*: top right, acima à direita;
  - *tl*: top left, acima à esquerda;
  - *br*: bottom right, abaixo à direita;
  - *bl*: bottom left, abaixo à esquerda.
- `height`: altura. Necessário utilizar a função `unit()` do pacote `ggplot2` para especificar a unidade de medida que, no nosso caso, será centímetros;
- `style`: somente para a seta. É definido um estilo para a seta.

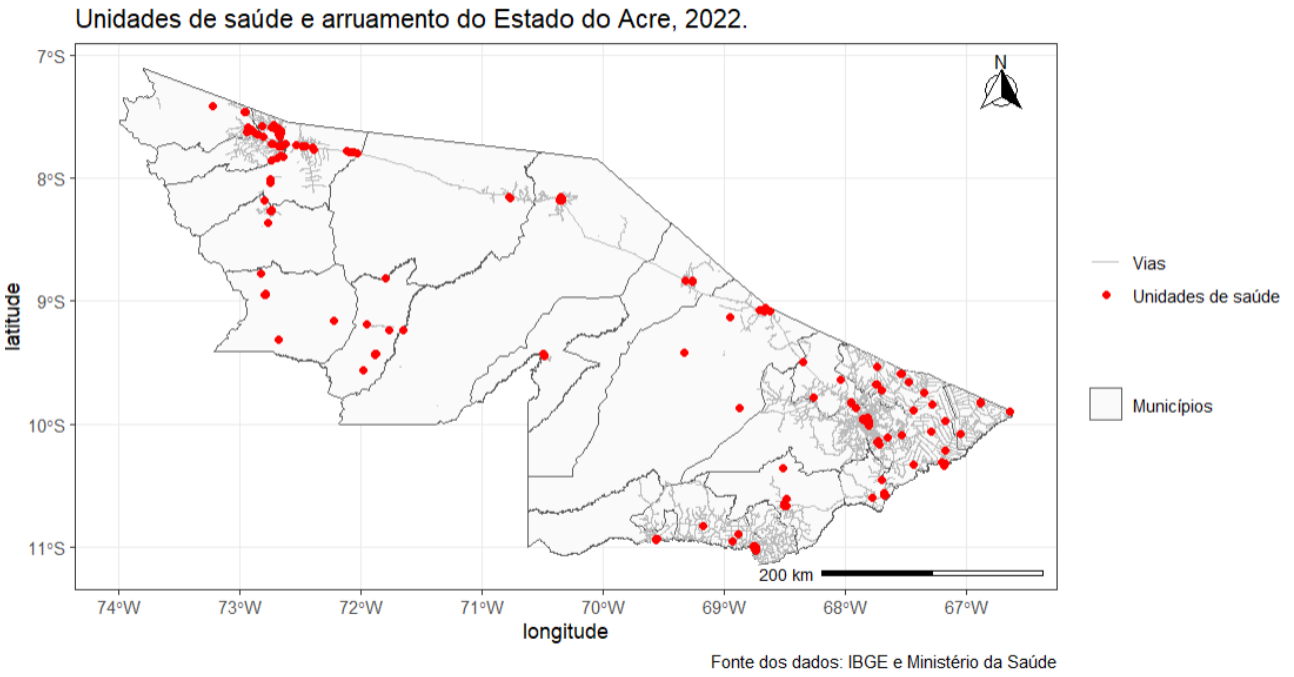
Também vamos utilizar a função `labs()` do pacote `ggplot2` para definir o texto do título (*title*), do eixo x (no caso, a longitude), do y (no caso, a latitude) e a fonte dos dados.

Para as legendas, vamos utilizar o recurso estético da função `aes()` do pacote `ggplot2` junto com as funções `scale_color_manual()` e `scale_fill_manual()`, ambas do mesmo pacote `ggplot2`. Com a função `aes()`, podemos definir um codinome para a cor que vamos utilizar nas geometrias (*fill* e *color*) e utilizar as demais para formatar rótulos e cores, propriamente ditas.

Execute o código mostrado abaixo com bastante atenção no seu `RStudio` e observe como fica a composição final do mapa:

```
# Utilizando a função ggplot() para plotagem dos mapas
ggplot()+

  # Adicionando camada com os limites poligonais do estado do AC com o
  # objeto `ac_municipios` e a função geom_sf().
  # O argumento "fill" define o codinome da cor de preenchimento
  geom_sf(data = ac_municipios, aes(fill = "cor_municipios")) +

  # Adicionando camada com objeto de linhas representando as vias com o
  # objeto `ac_vias` e a função geom_sf()
  # O argumento "color" define o codinome da cor das linhas
  geom_sf(data = ac_vias, aes(color = "cor_ruas")) +

  # Adicionando camada com objeto de pontos  representando unidades de
  # saúde com o objeto `ac_unidades` e a função geom_sf()
  # O argumento "color" define o codinome da cor de preenchimento
  geom_sf(data = ac_unidades, aes(color = "cor_unidades")) +

 # Definindo a escala gráfica do mapa com a função annotation_scale()
  annotation_scale(location = "br",
                   height = unit(.1, "cm")) +

  # Definindo a visualização de seta apontada para o Norte
  annotation_north_arrow(location = "tr",
                         height = unit(1, "cm"),
                         style = north_arrow_fancy_orienteering) +

  # Criando a legenda das linhas e pontos
  scale_color_manual(
    name = "",
    guide = guide_legend(override.aes = list(linetype = c("solid", "blank"),
                                             shape = c(NA, 16))),
    values = c('cor_ruas' = '#BABABA', 'cor_unidades' = '#FF0000'),
    labels = c("Vias", "Unidades de saúde")
  ) +

  # Criando a legenda dos polígonos
  scale_fill_manual(
    name = "",
    values = c('cor_municipios' = '#FAFAFA'),
    labels = "Municípios"
  ) +

  # Definindo o título e rótulos dos eixos do gráfico
  labs(title = "Unidades de saúde e arruamento do Estado do Acre, 2022.",
       x = "longitude",
       y = "latitude",
       caption = "Fonte dos dados: IBGE e Ministério da Saúde") +

  # Adicionando o tema
  theme_bw()
```

Figura 7: Sobreposição de arquivos vetoriais e composição de mapa com `ggplot2()`.



Muito fácil, não é mesmo? Apesar de alguns comandos que podem parecer avançados ou difíceis, com o tempo você se habituará às configurações. O importante é **seguir atentamente os códigos apresentados**.

Um detalhe que podemos analisar no mapa acima é que, embora a escala utilizada seja para um estado inteiro, impedindo de visualizar detalhes mais próximos, ainda podemos notar as unidades de saúde com uma concentração na região Sul e Sudeste do estado. Nesta região, possivelmente, localizam-se os municípios mais populosos e com maior infraestrutura de saúde disponível, incluindo a capital Rio Branco. Percebemos também que, a Oeste do estado, alguns municípios não possuem estradas de ligação às unidades de saúde. Possivelmente, pode ser unidades localizadas em zona rural ou com acesso somente de barco por rios, dadas algumas características geográficas do Acre.

Vocês perceberam que fizemos uma breve análise somente com os dados que sobrepomos? Essas informações são importantes e podem compor o relatório para apresentar ao diretor geral que solicitou o mapa. Interessante, não é mesmo? Isso demonstra que a capacidade do analista da Vigilância em Saúde pode ir além dos conhecimentos sobre a doença.

Agora vamos entender como cruzar informações de tabelas com os arquivos vetoriais.

Os dados de limites dos municípios podem ser acessados no site do IBGE. Para acessar, acompanhe os passos:

1. acesse no seu navegador www.ibge.gov.br;
2. clique no Menu, em Geociências e, em seguida, em *Downloads*;
3. siga o seguinte caminho, clicando nas setas da árvore de arquivos:

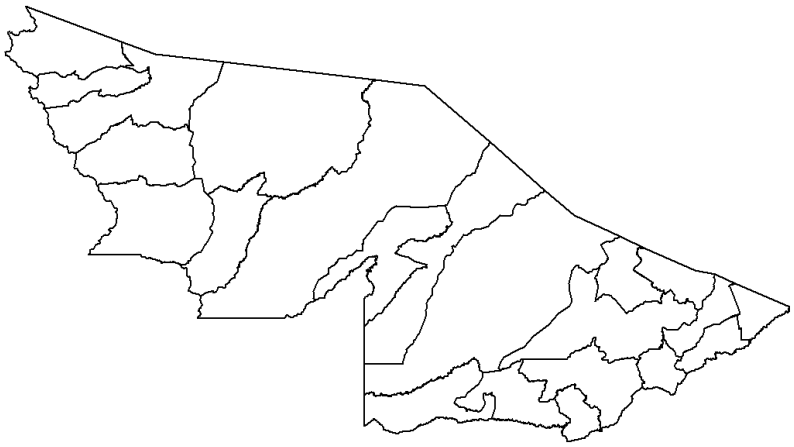
- ▶ organizacao_do_territorio
- ▶ malhas_territoriais
- ▶ malhas_municipais
- ▶ município_2021
- ▶ Brasil
- ▶ UFs

Outra forma de realizar o *download* dos limites dos municípios, limites dos estados, localização geográfica de unidades de saúde no Brasil, é utilizando o pacote `geobr`. Esse pacote foi desenvolvido pela equipe do Instituto de Pesquisa Econômica Aplicada (IPEA) e tem como objetivo possibilitar o *download* de conjuntos oficiais de dados espaciais do Brasil.

Sua utilização é bem simples:

```
# Instalando e carregando o pacote `geobr`
if(!require(geobr)) install.packages("geobr");library(geobr)
if(!require(sf)) install.packages("sf");library(sf)

# Realizando o download dos municípios do Estado do Acre
ac_municipios_geobr <- read_municipality("AC")

# Visualizando o objeto `ac_municipios_geobr`
plot(st_geometry(ac_municipios_geobr))
```

Figura 8: Exemplo de visualização utilizando dados do pacote `geobr`.



E por último, uma forma de realizar o *download* das vias é o pacote `osmextract`, que baixa dados do projeto *Open Street Map*.

```
 # Instalando e carregando o pacote `osmextract`
if(!require(osmextract)) install.packages("osmextract");library(osmextract)

# Realizando o download dos dados do Estado do Acre
# e filtrando apenas as vias
ac_via_osm <- oe_get(place = "Acre") |>
  filter(!is.na(highway))

# Visualizando o objeto `ac_via_osm`
plot(st_geometry(ac_via_osm))
```

Figura 9: Exemplo de visualização utilizando dados do pacote `osmextract`.

4. União dos dados tabulares (não-gráficos) com bases geográficas (gráficas)

Uma das formas de armazenamento dos dados espaciais no meio digital é por meio dos arquivos vetoriais, estruturados conforme a geometria que se assimila no mundo real. Mas todo dado espacial possui atributos. Por exemplo, na vigilância ambiental é comum o uso de área das localidades de trabalho para contenção de epidemias. Essas localidades possuem identificador, extensão total, número de residências, número de comércios, números de locais de grande acúmulo de focos de vetores, sua geometria e as coordenadas de localização. Todas essas informações são atributos.

Os dados que se referem à localização, tipo da geometria e demais atributos referentes à descrição gráfica do objeto a ser representado são chamados de **dados gráficos**. Por vezes, algumas destas informações não estão originalmente presentes na base geográfica. São chamados de **dados não-gráficos** aqueles que expressam características descritivas, sociais, naturais e quantitativas armazenadas em forma de tabelas (essencialmente nos formatos ".csv"; ".xlsx"). A ligação entre estes dois tipos de dados é chamada união ou relacionamento.

Várias informações podem ser armazenadas como dados não gráficos:

- dados populacionais,
- contagens de eventos (como as tabelas do Tabnet),
- taxas e razões de eventos relacionados à saúde,e
- dados ambientais.

No próximo tópico vamos aprender como é possível a união dos dados citados utilizando a linguagem `R`.

4.1 União com códigos (geocódigos)

Para unir tabelas com dados gráficos a tabelas com dados não-gráficos é necessária uma chave identificadora chamada de **geocódigo**, presente em ambas e que fará a ligação entre elas. O geocódigo pode ser uma variável de identificação da localidade ou um código do IBGE. Mas é essencial que tenha exatamente o mesmo tipo de dado (numérico, texto) e o mesmo número de caracteres.

No `R` essa ligação acontece por meio de um cruzamento entre campos chamado *join*. Na Unidade 3 do curso básico "Análise de dados para a Vigilância em Saúde" conhecemos o conceito desse cruzamento. Neste curso, vamos praticar utilizando os dados de unidades de saúde.

O arquivo de dados geográficos de unidades de saúde contém 1008 registros e 5 colunas, incluindo a geometria. Como neste arquivo cada linha representa uma unidade de saúde, vamos inspecionar a estrutura da tabela usando a função `glimpse()` do pacote `dplyr`. Utilize o código a seguir para fazer essa inspeção:

```
# Inspecionando o objeto ac_unidades com a função glimpse()
glimpse(ac_unidades)
```

```
#> Rows: 1,008
#> Columns: 5
#> $ uf         <chr> "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC",…
#> $ code_state <dbl> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,…
#> $ cod_mun    <dbl> 120035, 120001, 120001, 120001, 120001, 120001, 120001, 120…
#> $ cnes_n     <chr> "3591018", "3638685", "7245890", "7026641", "0257184", "338…
#> $ geometry   <POINT [°]> POINT (-72.68529 -9.309803), POINT (-66.883 -9.828), …
```

Percebemos que na base geográfica, só há informações a respeito da localização básica das unidades no estado, como código do município (`cod_mun`) e código da unidade de saúde (`variável cnes_n`). Contudo, a variável `cnes_n` é o código de cadastro no CNES (Cadastro Nacional de Estabelecimento de Saúde), do tipo *character*.

Agora, para conectar com esta tabela, vamos importar o arquivo {`ac_unidades_tabela.xlsx`} usando a função `read_xlsx()` do pacote `reaxl` e repetir a inspeção feita acima. Execute os seguintes códigos no seu computador:

```
# Carregando base de dados com a função read_xlsx()
tabela_unidades <- read_xlsx('Dados/ac_unidades_tabela.xlsx')

# Inspecionando o objeto tabela_unidades com a função glimpse()
glimpse(tabela_unidades)
```

```
#> Rows: 1,008
#> Columns: 7
#> $ codigo_cnes <chr> "3591018", "3638685", "7245890", "7026641", "0257184", "33…
#> $ nome        <chr> "ESF RIBEIRINHA ROSENDO RODRIGUES", "UNIDADE DE SAUDE DA F…
#> $ endereco    <chr> "RIO JURUA", "BR 364 KM 114", NA, NA, "RUA JOSE EMILIAO DE…
#> $ bairro      <chr> "VILA FOZ DO BREU", "ZONA RURAL", NA, NA, "NOSSA SENHORA R…
#> $ cod_mun     <dbl> 120035, 120001, 120001, 120001, 120001, 120001, 120001, 12…
#> $ code_state  <dbl> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12…
#> $ tipo_unida  <chr> "02 - CENTRO DE SAUDE/UNIDADE BASICA", "02 - CENTRO DE SAU…
```

Perceba que essa tabela não possui dados gráficos ou geometrias. Há somente sete variáveis que consistem, basicamente, em identificação das unidades, como nome, endereço e tipo. Também há a variável `codigo_cnes` que, como o nome da variável sugere, armazena o código do CNES de cada unidade, sendo do tipo *character*.

Muito bem, vamos utilizar as variáveis com o código da unidade de ambos os arquivos ({`ac_unidades`} e {`tabela_unidades`}). Embora essas variáveis não estejam nomeadas exatamente iguais, isso não atrapalha a união que faremos, já que as duas são compatíveis (ambas variáveis são do tipo *character* e têm o mesmo comprimento).

Agora, vamos utilizar a função `left_join()` do pacote `dplyr` para realizar a união. Assim, vamos indicar no argumento `by` da função `left_join()` a variável do objeto {`ac_unidades`} que se liga à tabela {`tabela_unidades`} utilizando a função `c()`. Acompanhe o código abaixo com atenção e replique no seu computador:

```
# Salvando o objeto `ac_unidades_completo` pela união da tabela
# `ac_unidades` com a tabela tabela_unidades
ac_unidades_completo <- ac_unidades |>

  # Atilizando a função left_join() para união.
  # A ligação é feita pela correspondência entre as
  # colunas "cnes_n" e "codigo_cnes"
  left_join(tabela_unidades, by = c("cnes_n" = "codigo_cnes"))
```

Pronto! Os arquivos agora estão unidos e podem ser utilizados normalmente. Para verificar como os campos ficaram após a união, utilize a função `glimpse()` novamente:

```
glimpse(ac_unidades_completo)
```

```
#> Rows: 1,008
#> Columns: 11
#> $ uf           <chr> "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC…
#> $ code_state.x <dbl> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 1…
#> $ cod_mun.x    <dbl> 120035, 120001, 120001, 120001, 120001, 120001, 120001, 1…
#> $ cnes_n       <chr> "3591018", "3638685", "7245890", "7026641", "0257184", "3…
#> $ geometry     <POINT [°]> POINT (-72.68529 -9.309803), POINT (-66.883 -9.828)…
#> $ nome         <chr> "ESF RIBEIRINHA ROSENDO RODRIGUES", "UNIDADE DE SAUDE DA …
#> $ endereco     <chr> "RIO JURUA", "BR 364 KM 114", NA, NA, "RUA JOSE EMILIAO D…
#> $ bairro       <chr> "VILA FOZ DO BREU", "ZONA RURAL", NA, NA, "NOSSA SENHORA …
#> $ cod_mun.y    <dbl> 120035, 120001, 120001, 120001, 120001, 120001, 120001, 1…
#> $ code_state.y <dbl> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 1…
#> $ tipo_unida   <chr> "02 - CENTRO DE SAUDE/UNIDADE BASICA", "02 - CENTRO DE SA…
```

Caso deseje entender mais sobre os cruzamentos, indicamos fortemente a realização do curso básico "Análise de dados para a Vigilância em Saúde". Além disso, é possível utilizar as funções de ajuda do `R` da seguinte forma:

```
help("left_join")
```

Da forma acima, uma página deve ser aberta no painel Help, localizada à direita do seu `RStudio`.

5. Criação de mapas temáticos

Mapas temáticos são aqueles destinados a representar quaisquer fenômenos distribuídos geograficamente, indo além da representação ou visualização do espaço em si. A representação temática busca explorar, analisar e sintetizar dados sobre um determinado assunto e, por vezes, correlacionar com outros tipos de dados. Dessa forma, utilizam-se dados gráficos e não gráficos já existentes para compor uma análise. Eles são muito utilizados nas pesquisas socioeconômicas, de saúde e ambientais.

Atualmente, com as ferramentas de análise espacial disponíveis, o mapa não é somente um produto para comunicar algo. Pelo contrário, uma representação adequada é o começo de todo processo de análise. Para a Vigilância em Saúde isso é muito poderoso, pois utilizar mapas para abordar temas de interesse, cruzando informações e analisando correlações torna-se uma das habilidades mais importantes para a saúde pública.

Nesse tópico, vamos abordar a elaboração de alguns dos principais mapas para a Vigilância em Saúde e vamos utilizar os arquivos vetoriais que já importamos e mais algumas bases de dados necessárias para cálculo de algum indicador.

Vamos lá?

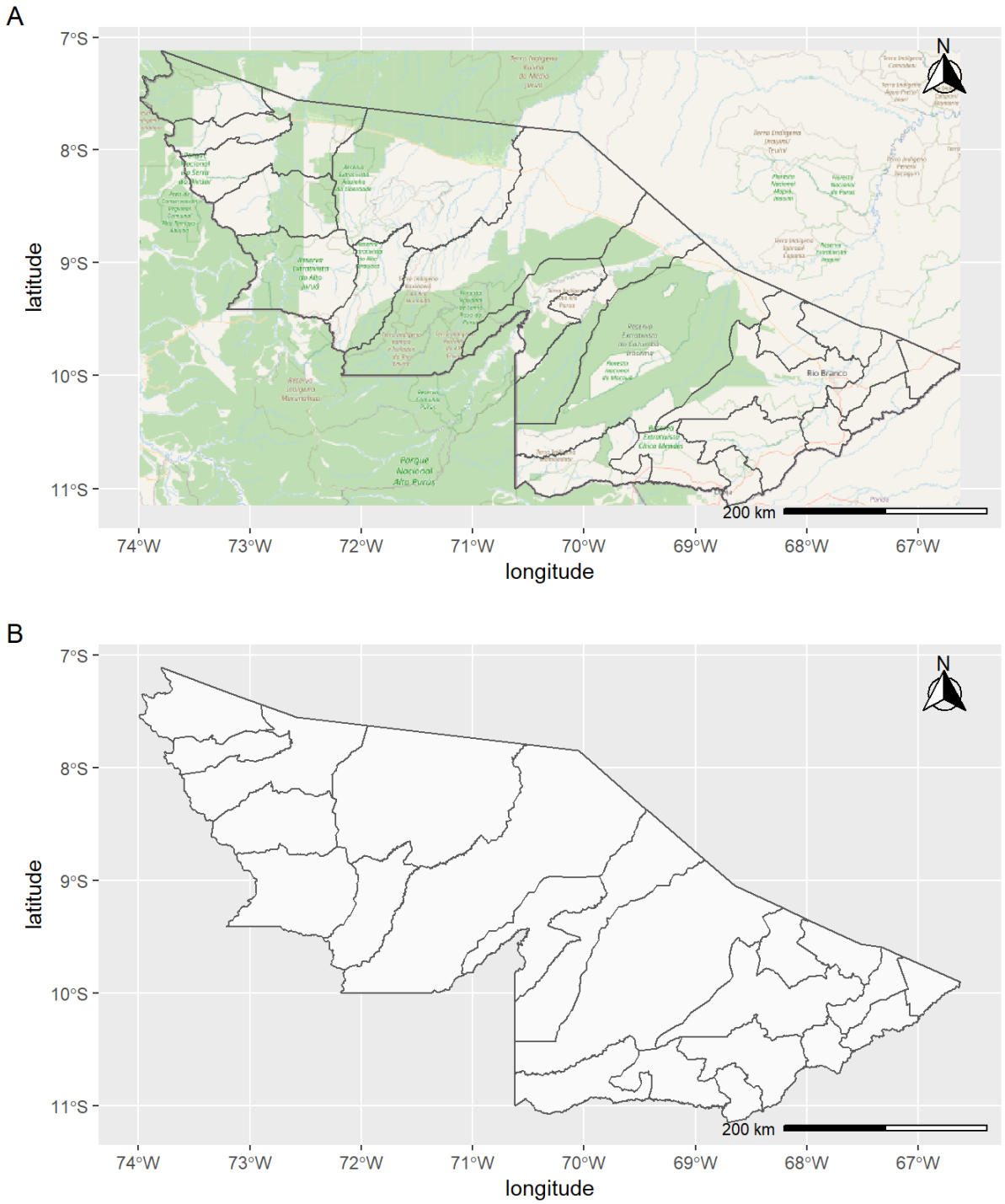
5.1 Mapa base

O mapa base é o primeiro elemento a ser adicionado na composição de um mapa. Também é chamado de mapa de fundo, e em inglês basemap e *backgroundmap*.

É o mapa que representará a área geográfica de estudo ou os limites administrativos necessários para representar o fenômeno de interesse. Por vezes, é substituído por outra camada, principalmente nos mapas temáticos que utilizam toda a área para representar um fenômeno.

Como qualquer elemento adicionado durante a composição de um mapa, a escolha do mapa base deve seguir critérios que sempre melhorem a interpretação e ajudem ao leitor do mapa a entender o que está sendo representado. Um mapa base pode ser desde a representação dos limites administrativos a imagens do terreno estudado.

Figura 10: Exemplos de mapas base utilizados para mapas temáticos.



A depender do objetivo, os mapas base podem conter diferentes informações. Neste curso, utilizaremos o mapa base que representa os municípios do Estado do Acre. Dessa forma, vamos destacar as informações que vamos sobrepor. Acompanhe os demais tipos de mapas utilizados na Vigilância em Saúde.

5.2 Mapa de símbolos proporcionais

Mapas de símbolos proporcionais representam a ocorrência de um evento em valores absolutos por meio da variação proporcional do tamanho de uma figura geométrica, geralmente círculo, sobrepostos sobre a unidade territorial. O círculo é localizado conforme a posição real do evento ou atribuído à localização de referência, como o centro geográfico (centróide) de um município ou de um bairro.

A representação de números totais por meio da variação do tamanho respeita a ideia de quão maior é um valor em relação a outro, sem envolver relações geográficas. Além disso, permite uma leitura fácil para entender as diferenças entre as unidades territoriais avaliadas. Mas, uma atenção especial deve ser dada à legenda, pois a interpretação deve ser clara. Ou seja, os símbolos proporcionais devem se destacar de forma única.

Para demonstrar este mapa, vamos utilizar os dados de população do estado do Acre, segundo os municípios. Nós utilizamos estes dados no curso básico "Análise de dados para a Vigilância em Saúde" e retornaremos a eles aqui. Primeiro, importaremos os dados, em seguida vamos uni-los à base geográfica utilizando o geocódigo do município (o código do IBGE) e, finalmente, vamos plotar o mapa.

A nossa base de população contém dados do estado do Acre de 2009 a 2021. Ao importá-la, perceba que o comando possui um argumento chamado `col_types`. Esse argumento é necessário para garantir que a primeira coluna do arquivo, chamada "Codigo", se mantenha como o formato original "*character*".

Além disso, vamos padronizar o nome das colunas (minúsculas, sem acento, sem espaço e sem colunas começando com números) utilizando a função `clean_names()` do pacote `janitor` (perceba que após corrigir o nome das colunas, aquelas que iniciavam com números, possuem a letra "x" no lugar). Execute o código:

```
# Carregando tabela com dados de população do acre para o objeto pop_ac
# utilizando a função read_csv2()
pop_ac <- read_csv2('Dados/pop_ac_09_21.csv', col_types = list("character"))
```

```
#> i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
```

```
# Padronizando o nome das colunas utilizando a função `clean_names()`
# do pacote janitor
pop_ac <- clean_names(pop_ac)
```

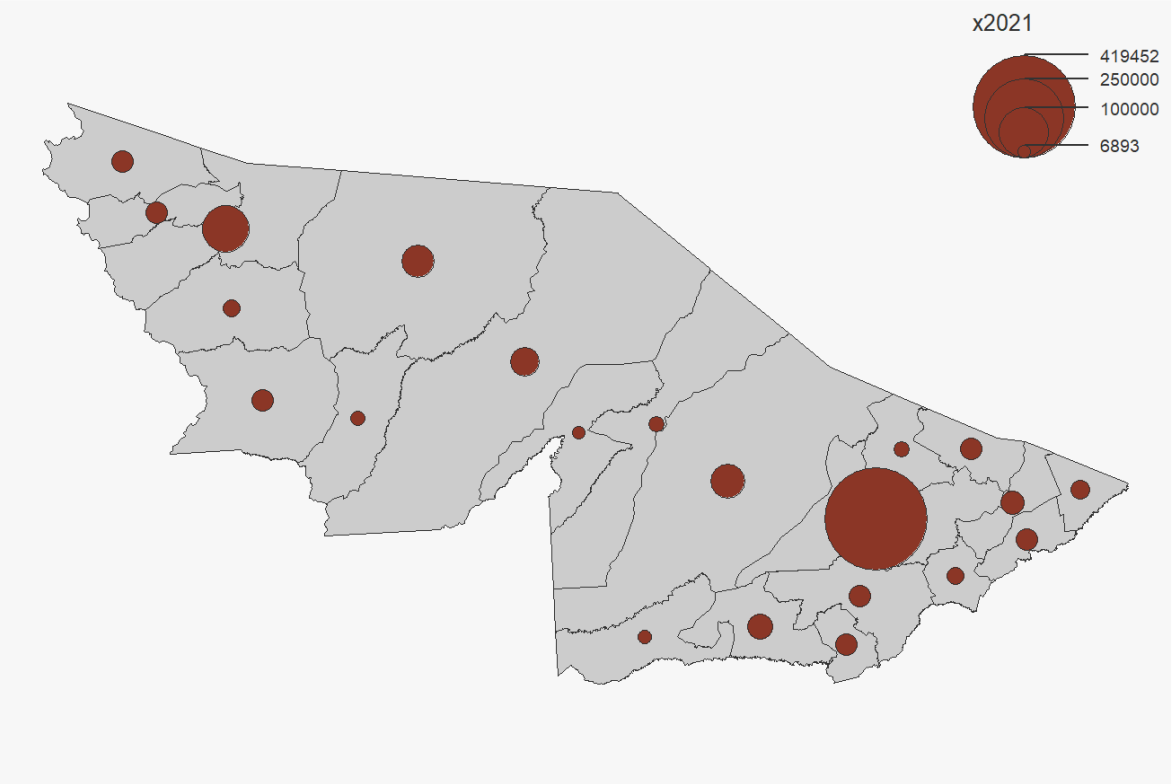
Agora, vamos unir à base geográfica de municípios que será, nesse caso, nosso mapa base. Para isso, vamos utilizar a função `left_join()` do pacote `dplyr`. Como o geocódigo possui nomes diferentes em cada base, vamos indicar no argumento `by` quais colunas serão utilizadas como chave.

```
# Unindo a tabela ac_municipios_5880 com pop_ac com a função left_join()
pop_geo_ac <- ac_municipios_5880 |>

  # A ligação é feita pela correspondência entre as colunas "cod_mun" e "codigo"
  left_join(pop_ac, by = c('cod_mun' = 'codigo'))
```

Por último, vamos plotar o mapa utilizando a coluna da população de 2021 (“x2021”) no argumento `var`. Note que ao utilizar o pacote `mapsf` temos uma redução da necessidade de várias linhas de comandos. A função `mf_prop()` já calcula o tamanho dos círculos, classifica e posiciona a legenda.

```
# Plotando o objeto pop_geo_ac com uso da função mf_map()
mf_map(pop_geo_ac) |>

  # Indicando o tamanho proporcional da população com uso da função mf_prop()
  mf_prop(var = "x2021")
```

Figura 11: Exemplo de Mapa de símbolos proporcionais.



Viu como é fácil? Apesar de poucas linhas, podemos adicionar mais complexidade, deixando o mapa com uma melhor qualidade do mapa, como adicionar os elementos cartográficos e estilizar um pouco mais. Faremos esse processo por etapas:

1. Primeiro, vamos utilizar a função `mf_theme()` que apresenta vários argumentos para criar um tema para o mapa. Neste caso vamos apenas alterar a cor do fundo para branco utilizando o argumento `bg`. Esse fundo não é o fundo do mapa, mas a área por trás do mapa.

```
# Definindo a cor de fundo por trás do mapa como "branco",
# com uso da função mf_theme()
mf_theme(bg = "white")
```

2. Em seguida, vamos adicionar o mapa base com os limites dos municípios. Vamos utilizar o argumento `col` para alterar a cor do preenchimento da área dos municípios. Perceba que usamos a codificação de cores (hexadecimal) apresentada no curso. Vamos também sobrepor a camada com os círculos proporcionais, indicando os seguintes argumentos:

- `var`: coluna com os dados a serem plotados;
- `leg_pos`: posição da legenda. Este argumento aceita as seguintes instruções:
  - *topright* : acima à direita;
  - *topleft*: acima à esquerda;
  - *bottomright*: abaixo à direita;
  - *bottomleft*: `abaixo à esquerda.`
- `leg_title`: o título da legenda;
- `col`: este argumento dentro da função `mf_prop()` se refere à cor interna dos círculos.

```
# Plotando o objeto pop_geo_ac com uso da função mf_map()
# O argumento "col" define a cor de preenchimento utilizando o código de
# cor hexadecimal
mf_map(pop_geo_ac, col = "#FAFAFA") |>

  # Indicando o tamanho proporcional da população com uso da função mf_prop()
  mf_prop(
    # Definindo a variável x2021 para cálculo do tamanho proporcional dos círculos
    var = "x2021",

    # Definindo a posição da legenda para "abaixo e à esquerda"
    leg_pos = "bottomleft",

    # Definindo o título do gráfico
    leg_title = "Populacao Acre 2021",

    # O argumento "col" define a cor de preenchimentto dos círculos utilizando
    # o código de cor hexadecimal
    col = "#FFAD48"
  )
```

3. Agora vamos inserir a seta de norte utilizando a função `mf_arrow()`. Essa função tem como principal argumento a posição da seta, indicada por `pos`. Note abaixo que indicar a posição é similar às configurações citadas acima.

```
# Definindo a posição da seta de norte para a região "superior e à direita"
# com a função mf_arrows()
mf_arrow(pos = "topright")
```

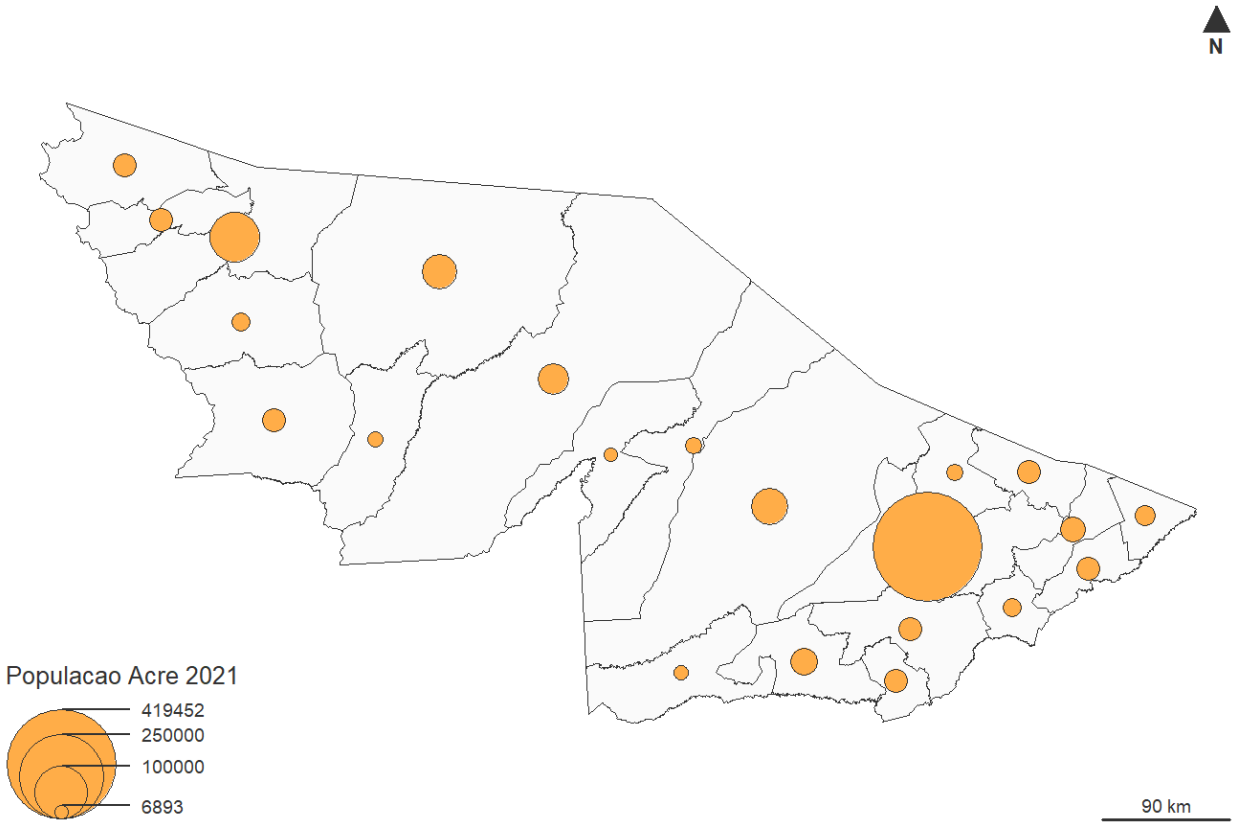
4. Por fim, vamos inserir a escala, utilizando a função `mf_scale()`. O tamanho da escala é padronizado, mas é possível alterá-lo utilizando o argumento `size`. Mas cuidado, este argumento é legível quando está sendo utilizada um sistema de coordenada plana.

```
# Inserindo a escala com a função mf_scale()
mf_scale()
```

Abaixo, apresentamos o código completo. Execute-o e observe o mapa gerado:

```
# Definindo a cor de fundo por trás do mapa como "branco", com uso da
# função mf_theme()
mf_theme(bg = "white")

# Plotando o objeto pop_geo_ac com uso da função mf_map()
# O argumento "col" define a cor de preenchimento utilizando o código de
#  cor hexadecimal
mf_map(pop_geo_ac, col = "#FAFAFA") |>

# Indicando o tamanho proporcional da população com uso da função mf_prop()
mf_prop(
  # Definindo a variável x2021 para cálculo do tamanho proporcional dos círculos
  var = "x2021",

  # Definindo a posição da legenda para "abaixo e à esquerda"
  leg_pos = "bottomleft",

  # Definindo o título do gráfico
  leg_title = "Populacao Acre 2021",

  # O argumento "col" define a cor de preenchimentto dos círculos utilizando
  # o código de cor hexadecimal
  col = "#FFAD48"
)

# Definindo a posição da seta de norte para a região "superior e à direita"
# com a função mf_arrows()
mf_arrow(pos = "topright")

# Inserindo a escala com a função mf_scale()
mf_scale()
```

Figura 12: Exemplo de elaboração de mapas com símbolos proporcionais.

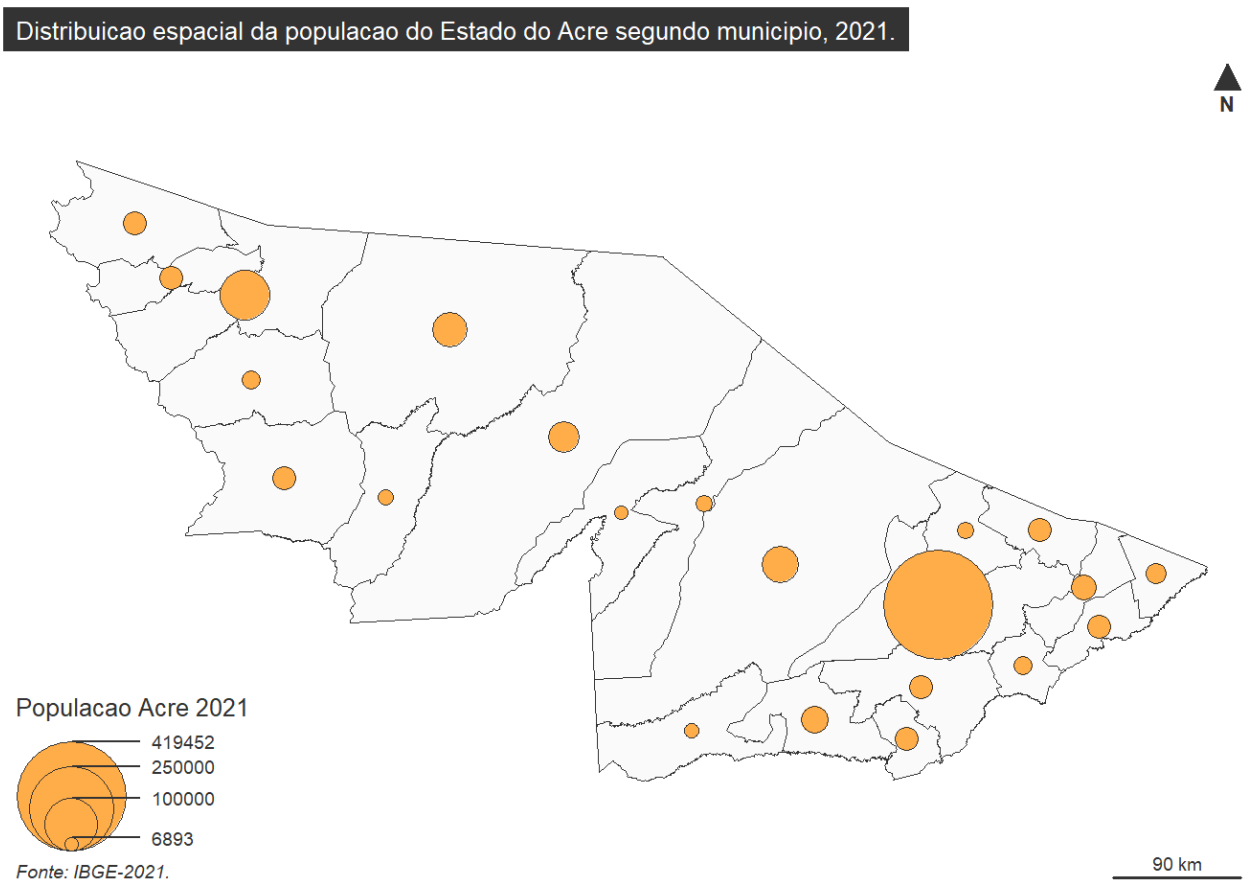


Note que não utilizamos o operador pipe em todas as situações. Para sobreposição de elementos cartográficos, como seta de norte, escala ou até mesmo alterações do layout, **o pacote `mapsf` não exige o uso do pipe em todas as situações**.

Utilizando o pacote `mapsf` também é possível inserir título e notas de rodapé, utilizando as funções `mf_title()` e `mf_credits()`. O uso dessas funções é muito similar às demais do pacote.

```
mf_theme(bg = "white")
mf_map(pop_geo_ac, col = "#FAFAFA") |>
mf_prop(
  var = "x2021",
  leg_pos = "bottomleft1",
  leg_title = "Populacao Acre 2021",
  col = "#FFAD48"
)
mf_arrow(pos = "topright")
mf_scale(pos = "bottomright")
mf_title(txt = "Distribuicao espacial da populacao do Estado do Acre segundo municipio, 2021.", cex = .8)
mf_credits(txt = "Fonte: IBGE-2021.", pos = "bottomleft")
```

Figura 13: Exemplo de elaboração de mapas com símbolos proporcionais com layout completo.



A população total do Acre estava em cerca de 900 mil habitantes em 2021. Analisando a distribuição da população, podemos notar que a população acreana está irregularmente distribuída, com uma grande concentração na capital Rio Branco. A ligação entre a região norte e noroeste do estado se dá, principalmente, pela Rodovia BR-364. Possivelmente, há uma relação entre as regiões mais povoadas, as redes de transporte (seja rodoviária, seja hidroviária) e a extensão da Floresta Amazônica.

Essa distribuição irregular pode gerar uma desigualdade na área da saúde, com concentração da infraestrutura de saúde como referência hospitalares e equipamentos de diagnósticos. Como analista da Vigilância em Saúde, estas informações podem ser ampliadas, integradas com as realidades locais, acrescidas de detalhes sobre o diagnóstico situacional e potencializadas com o uso de indicadores de saúde.

Agora, vamos conhecer o principal mapa utilizado na Vigilância em Saúde.

5.3 Mapa coroplético

Mapas coropléticos representam dados de unidades de áreas delimitadas, que podem ser bairro, áreas administrativas, distritos ou até mesmo o município. Cada área recebe uma cor de acordo com a classificação prévia dos dados, que podem ser taxas de incidências, razões de mortalidade, entre outros. Para elaborá-lo, basta que haja as taxas para cada área de análise e o arquivo vetorial com os limites administrativos.

São muito utilizados para análise do comportamento de uma doença em relação ao tempo ou para comparação entre áreas. Sua escolha se dá, muitas vezes, devido aos dados da vigilância serem divulgados por área (bairros, regiões). Além disso, como não apresenta a localização exata do fenômeno, a privacidade é mantida.

Quando utilizado para comparar comportamento de um evento entre semana epidemiológicas, meses ou anos, é necessário que a classificação dos dados deve ser a mesma para permitir uma comparação adequada.

Os valores absolutos ou totais não são adequados para representar eventos relacionados à saúde utilizando mapas coropléticos e não devem ser usados quando se referir a eventos que podem ser influenciados pelo tamanho da população em risco ou pelo tamanho da área que se origina os casos. Por exemplo, utilizar mapa coroplético para mostrar onde há mais casos de uma doença não é adequado, considerando que há uma possibilidade alta de onde tiver mais habitante, ter mais casos. Por isso, a indicação é pela adoção do uso de indicadores como taxas de incidência ou prevalência.

A composição deste tipo de mapa envolve, basicamente, quatro etapas essenciais:

- definição da área que será representada;
- cálculo das razões, frequências ou taxas da variável de interesse;
- classificação da variável;
- visualização do mapa.

Vamos praticar estas etapas com dois exemplos que seguem nos próximos tópicos.

5.3.1 Prevalência de Hanseníase no Acre

Para demonstrar a elaboração do mapa coroplético, nossa área de interesse continuará no Estado do Acre. Vamos utilizar os dados da prevalência de hanseníase entre os municípios do Acre, os quais foram detalhados na Unidade 5 do curso básico “Análise de dados para a Vigilância em Saúde”. Abaixo, vamos resgatar a linha de cálculo da prevalência, revisando os passos utilizados. Para mais detalhes, recomendados fortemente a realização do curso.

A prevalência de hanseníase estima a magnitude da endemia no Acre. Ela deve ser estudada com base na totalidade de casos em tratamento no momento da realização desta avaliação. Uma alta prevalência (valores acima de 5 casos por 10 mil habitantes) pode indicar um cenário de baixo desenvolvimento socioeconômico e falta de ações efetivas do município para o controle da doença. Por outro lado, a baixa prevalência (valores menores que 1 caso por 10 mil habitantes) pode indicar que a hanseníase não deve considerada um problema de saúde pública.

No código abaixo vamos importar os casos de hanseníase do SINAN no ano de 2021. Este banco de dados se encontra no menu lateral "Arquivos" do curso. Em seguida, vamos calcular a frequência de casos em andamento por município de residência atual e unir com a população residente. Por último, vamos calcular a prevalência para todos os municípios do estado e salvar esses cálculos no objeto {`prevalencia_hans_ac`}.

Acompanhe o código com atenção e replique-o no seu computador:

```
# Importação da base de casos do SINAN para o R, utilizando o pacote foreign.
base_hans <- read.dbf('Dados/base_hans_ac.dbf', as.is = TRUE)

casos_hans_ac_21 <- base_hans |>

  # Filtrando para os casos que o Tipo de Alta está em branco (TPALTA_N),
  # denotando os casos em acompanhamento.
  filter(is.na(TPALTA_N)) |>

  # Sumarização dos casos por município de atendimento (MUNIRESAT).
  count(MUNIRESAT)



prev_casos_ac <- casos_hans_ac_21 |>

  # Unindo a tabela de casos à tabela de população pelo geocódigo do município de atendimento.
  left_join(pop_ac, by = c("MUNIRESAT" = "codigo")) |>

  # Filtrando os municípios que não foram encontrados na união.
  filter(!is.na(MUNIRESAT))

# Calculando a prevalência por município e seleção das colunas necessárias para o mapa.
prevalencia_hans_ac <- prev_casos_ac |>
  mutate(prevalencia_2021 = (n / x2021) * 10000) |>
  select(MUNIRESAT, municipio, prevalencia_2021)

# Visualizando a tabela
prevalencia_hans_ac
```

```
#>    MUNIRESAT            municipio prevalencia_2021
#> 1     120001           Acrelândia        1.2721837
#> 2     120005         Assis Brasil        3.9220813
#> 3     120010           Brasiléia        2.9495262
#> 4     120013              Bujari        2.8376844
#> 5     120017             Capixaba        2.4429967
#> 6     120020      Cruzeiro do Sul        1.3368984
#> 7     120025       Epitaciolândia        4.7420834
#> 8     120030                Feijó        1.7149717
#> 9     120032              Jordão        1.1590172
#> 10    120033          Mâncio Lima        1.0181744
#> 11    120035 Marechal Thaumaturgo        0.5069195
#> 12    120038    Plácido de Castro        1.4890554
#> 13    120039         Porto Walter        4.0009602
#> 14    120040           Rio Branco        1.6688441
#> 15    120042      Rodrigues Alves        1.0117873
#> 16    120043  Santa Rosa do Purus        1.4507471
#> 17    120045     Senador Guiomard        2.5590719
#> 18    120050       Sena Madureira        1.4840570
#> 19    120060            Tarauacá        2.5154356
#> 20    120070              Xapuri        5.0337260
#> 21    120080           Porto Acre        2.6121937
```

A tabela final será mostrada no seu painel Console.

Agora vamos escolher o método de classificação da variável `prevalencia_2021`, presente no objeto {`prevalencia_hans_ac`} para representar no mapa coroplético.

Vários métodos podem ser calculados para a escolha da classificação da variável. Mas é sempre importante destacar: **não há método certo ou errado**. A melhor escolha é aquela que leve em consideração a forma que a variável se distribui e o número de unidades territoriais a serem representadas. A depender do método, a aparência do mapa muda. Áreas que poderiam ser consideradas prioritárias para alguma ação de prevenção, deixam de ser, simplesmente pela mudança da classificação.

Essa classificação irá criar grupos (classes) e contabilizar quantos municípios possuem a prevalência entre os intervalos classificados. Há várias maneiras de definir as classes. Vamos apresentar os principais métodos:

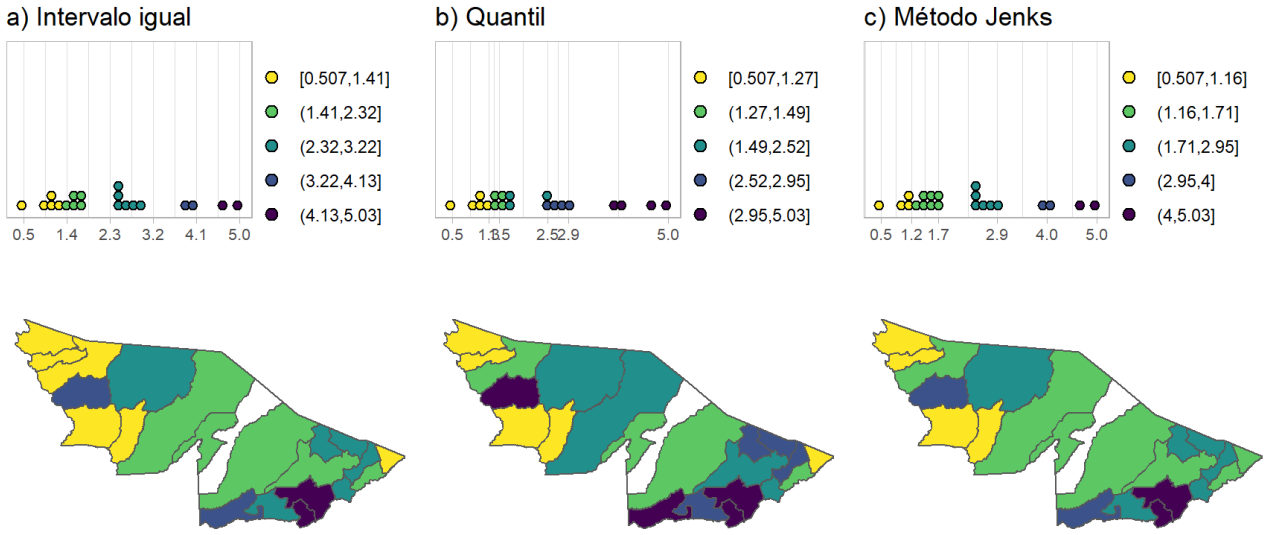
a) Intervalo igual: essa classificação define o intervalo considerando o máximo e mínimo da variável para dividir o número de classes. Cada classe possui um valor igual de diferença entre as outras. É de fácil entendimento e é adequada para distribuições uniformes e simétricas, além de poder ser usada para comparações entre mapas.

b) Quantil: essa classificação define a quantidade igual de observações em cada classe (quantis). É adequada para distribuições uniformes, simétricas e assimétricas e permite a comparação entre mapas.

c) Jenks: essa classificação utiliza o método de Jenks, que se baseia na semelhança (e diferença) de valores entre as unidades mapeadas. Na estatística, entende-se que esse método busca minimizar a variância dentro das classes e maximizar a variância entre as classes. É adequado para distribuições simétricas, mas a comparação entre mapas deve ser cautelosa.

Para ilustrar essas classificações, vamos apresentadas com os histogramas e mapas, na Figura 14:

Figura 14: Exemplo de métodos de definição de classes para mapas coropléticos.



Perceba que o mapa muda de acordo com a a classificação dos dados de prevalência utilizada. Municípios que ficariam com cores mais escuras numa classificação e, assim, representando valores mais altos, podem ser representados de forma diferente se mudar a classificação. Novamente reforçamos que a escolha será conforme o método que melhor represente o fenômeno mostrado no mapa e atenda ao objetivo do Analista.

Após termos uma introdução aos métodos de classificação, vamos visualizar a prevalência de hanseníase entre os municípios do Acre utilizando o método de quantis com o pacote `mapsf`, pois a distribuição dos dados está mais uniforme. Primeiro, vamos unir a tabela de prevalência com os municípios, utilizando como geocódigo a variável com código do IBGE em ambas as tabelas. Execute o código a seguir:

```
# Unindo a tabela `prevalencia_hans_ac` com `ac_municipios_5880`
# com a função left_join() e salvando no objeto `prevalencia_mun`
prevalencia_mun <- left_join(

  x = ac_municipios_5880,
  y = prevalencia_hans_ac,

  # A ligação é feita pela correspondência entre as colunas
  #  "cod_mun" e "MUNIRESAT"
  by = c("cod_mun" = "MUNIRESAT")
)
```

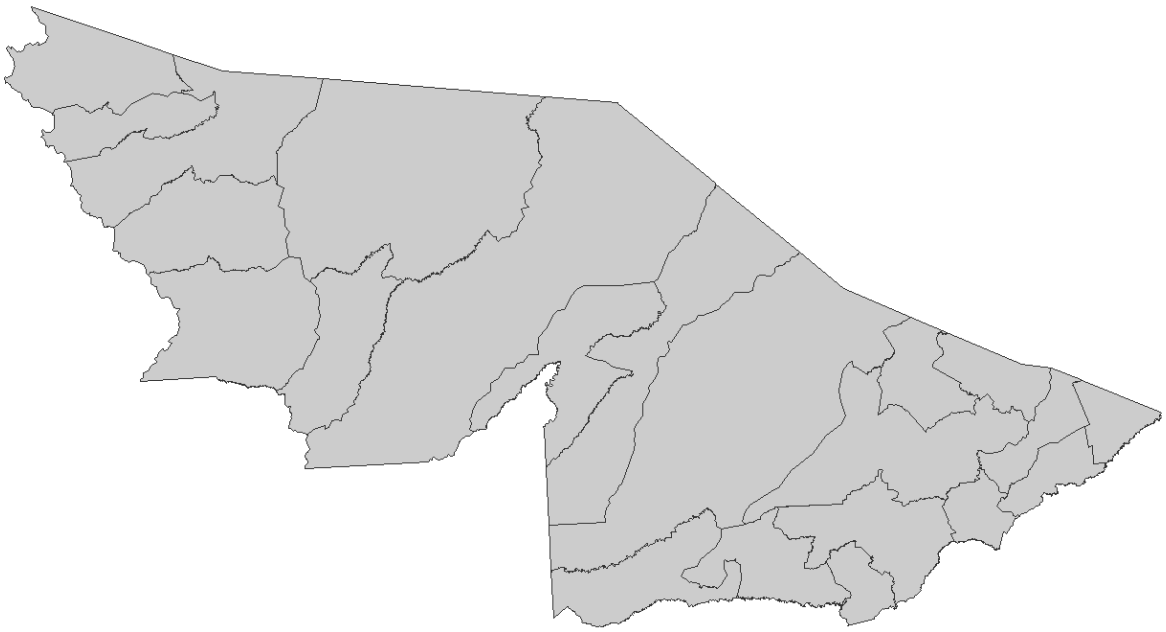
Em seguida, vamos utilizar a função `mf_choro()` para visualizar o mapa, definindo os seguintes argumentos:

- `var`: coluna com os dados a serem plotados;
- `breaks`: classificação utilizada para divisão das classes;
- `pal`: paleta de cores. Para visualização das paletas disponíveis excute a função `hcl.pals()` no console;
- `leg_pos`: posição da legenda;
- `leg_title`: o título da legenda;
- `leg_no_data`: texto a ser mostrado sempre que houver município sem dados.

Reproduza os próximos códigos com atenção e observe o resultado gerado:

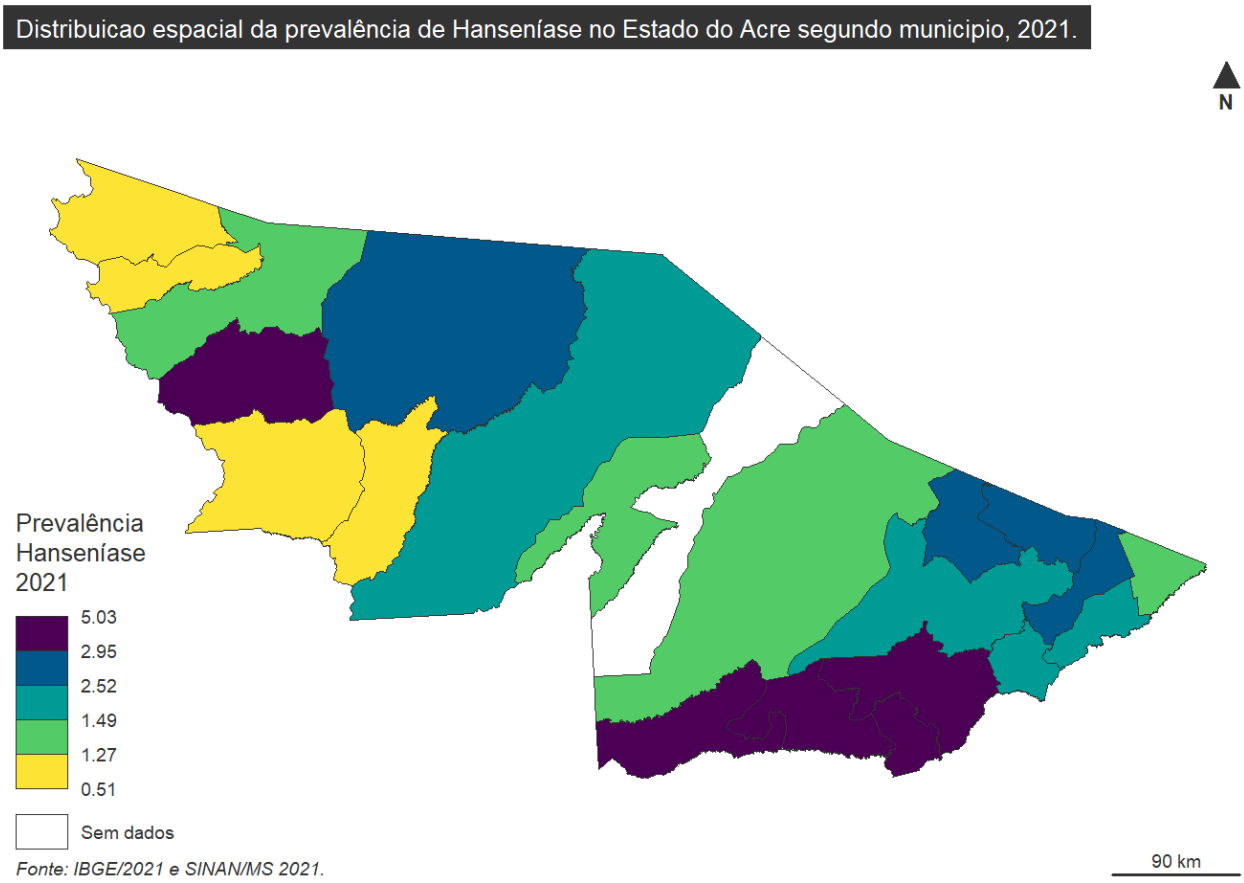
```
# Plotando o objeto prevalencia_mun com uso da função mf_map()
mf_map(prevalencia_mun) |>

# Adicionando camada de mapa coroplético com a função mf_choro()
mf_choro(

  # Definindo a variável que será utilizada para a cor de preenchimento
  var = "prevalencia_2021",

  # Definindo os intervalos de valores
  breaks = "quantile",

  # Definindo a paleta de cores para "viridis"
  pal = "Viridis",

  # Definindo a posição da legenda para a região "inferior e à esquerda"
  leg_pos = "bottomleft1",

  # Definindo o título da legenda
  #
  leg_title = "Prevalência \nHanseníase \n2021",

  # Inserindo caixa adicional para a legenda
  leg_no_data = "Sem dados"
)
```

Figura 15: Exemplo de elaboração de mapa coroplético com layout completo.



```
# Definindo a posição da seta de norte para a região "superior e à direita" com a
função mf_arrows()
mf_arrow(pos = "topright")

# Inserindo a escala com a função mf_scale()
mf_scale()

# Definindo o título do mapa e definindo o tamanho do texto
# para encaixar melhor no título
mf_title(txt = "Distribuicao espacial da prevalência de Hanseníase no Estado do
Acre segundo municipio, 2021.",
         cex = 0.8)

# Definindo informações complementares
mf_credits(txt = "Fonte: IBGE/2021 e SINAN/MS 2021.", pos = "bottomleft")
```

Figura 15: Exemplo de elaboração de mapa coroplético com layout completo.



Ao utilizarmos a a divisão por quantis em cinco classes, temos que 20% dos municípios estão classificados com valores altos. Dessa forma, ao analisarmos a Figura 15, percebemos que entre os municípios que apresentaram prevalência acima de 2,93 casos por 100 mil habitantes, quatro localizam-se na região sul do Acre e um na região oeste. NO entnado, apenas um possui prevalência registrada em 5 casos por 100 mil habitantes. Os municípios que registram os 20% mais baixos encontram-se todos na região oeste do estado.

É importante que, com esses dados, o analista da vigilância, juntamente com a área técnica que coordena ações contra a hanseníase, cruze dados socioeconômicos, de infraestrutura de saúde, de saneamento básico, de demografia e de situação do atendimento de Hanseníase nos locais acometidos. **O mapa não é o final de uma análise. Sempre será o ponto de partida para várias ações de saúde**.

5.3.2 Coeficiente de Mortalidade Infantil no município de São Paulo

A taxa de mortalidade infantil representa o número de óbitos ocorridos em crianças menores de um ano de idade por mil nascidos vivos no mesmo período. Este indicador é muito importante por permitir avaliar as condições de vida e de saúde de uma dada população. Com o cálculo da sua taxa, é possível estimar o risco de uma criança morrer antes de chegar a um ano de vida.

O indicador de mortalidade infantil, em especial a mortalidade pós-neonatal, quando **elevadas refletem as precárias condições de vida e saúde e valores abaixo do nível de desenvolvimento social e econômico**. Já taxas reduzidas podem sinalizar bons indicadores sanitários e sociais, mas também podem encobrir más condições de vida em segmentos sociais específicos.

Que tal olharmos um pouco para indicadores dentro dos municípios? Nessa perspectiva, considere que nossa tarefa é comparar a distribuição espacial do coeficiente de mortalidade infantil no município de São Paulo em três anos diferentes, 2019 a 2021. Portanto, precisamos criar um mapa coroplético para cada ano.

Para esse objetivo, vamos utilizar um outro pacote para elaboração de mapas temáticos: o `tmap`. Esse pacote possui muitas semelhanças com o `mapsf`, possuindo uma grande flexibilidade para elaboração de mapas e para adequações dos elementos de comunicação e cartográficos.

Para realizar nossa tarefa, vamos importar dois arquivos:

- um arquivo com a extensão “csv” baixado do Tabnet do município de São Paulo (Clique aqui para acesso). Neste arquivo temos o coeficiente de mortalidade infantil calculado para cada distrito administrativo do município de São Paulo, cuja fonte é a Secretaria Municipal de Saúde de São Paulo (SMS-SP).

- um arquivo vetorial com a extensão "gpkg" dos distritos administrativos. O distrito administrativo é a unidade territorial padrão de localização dos eventos relacionados à saúde em todos os sistemas de informação do SUS no município de São Paulo, sendo a menor área intramunicipal disponível para *download* (Clique aqui para acesso). A fonte é o Sistema Municipal de Planejamento e Gestão Urbana da Prefeitura de São Paulo, que disponibiliza o dado pelo Geosampa, um sistema de informações georreferenciadas de São Paulo.

Como vimos anteriormente, a ligação entre os dados não-gráficos e gráficos se dá pelo geocódigo. Mas, em algumas situações, o geocódigo não está disponível. Dessa forma, uma alternativa é utilizar o nome das unidades territoriais como ligação. Então, vamos realizar o procedimento necessário para padronizar os nomes para que fiquem iguais e permitam uma conexão entre as bases. Vamos lá?

Vamos começar importando o arquivo {`cmi_sp_19_21.csv`} com os dados de mortalidade infantil do município de São Paulo entre 2019 e 2021. Este arquivo se encontra no menu lateral "Arquivos" do curso. Note que, no código abaixo, vamos utilizar o argumento `locale` na função `read_csv2()` para corrigir a codificação dos caracteres do arquivo para os padrões brasileiros. Replique o código no seu `RStudio`:

```
# Importando o arquivo{`CID-cmi_sp_19_21.csv`} para o `R`
cmi_sp_19_21 <- read_csv2("Dados/cmi_sp_19_21.csv",
                          locale = locale(encoding = "ISO-8859-1"))
```

```
#> i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
```

```
#> Rows: 98 Columns: 5
#> ── Column specification ────────────────────────────────────────────────────
#> Delimiter: ";"
#> chr (5): Dist Administrativo resid, 2019, 2020, 2021, Total
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Visualizando o objeto
cmi_sp_19_21
```

```
#> # A tibble: 98 × 5
#>    `Dist Administrativo resid` `2019` `2020` `2021` Total
#>    <chr>                       <chr>  <chr>  <chr>  <chr>
#>  1 Água Rasa                   5,05   5,34   5,13   5,17
#>  2 Alto de Pinheiros           3,23   -      15,21  5,80
#>  3 Anhanguera                  7,43   11,49  8,51   9,08
#>  4 Aricanduva                  7,27   12,84  11,29  10,43
#>  5 Artur Alvim                 4,13   13,16  10,13  8,89
#>  6 Barra Funda                 7,71   2,82   2,75   4,51
#>  7 Bela Vista                  14,08  9,42   10,40  11,51
#>  8 Belém                       6,32   11,74  17,70  11,57
#>  9 Bom Retiro                  5,78   2,12   8,44   5,46
#> 10 Brás                        19,49  10,80  26,74  18,66
#> # … with 88 more rows
#> # i Use `print(n = ...)` to see more rows
```

Podemos perceber que o arquivo importado possui 5 colunas e 98 linhas. Os nomes das colunas possuem espaços e números e uma coluna "Total" que não será necessária. Além disso, as colunas que armazenam os coeficientes de mortalidade infantil estão como character.

Então, nosso próximo passo é padronizar as colunas e, principalmente, os nomes dos distritos administrativos, retirando acentos e sinais de pontuação (hífen, vírgulas e pontos) que, eventualmente, podem estar presentes. Vamos realizar uma pivotagem das colunas, levando os dados de mortalidade infantil para apenas uma coluna, pois como nosso objetivo é comparar todos os anos. Assim, teremos que ter uma coluna com os anos e outra com os valores correspondentes.

Para realizar esses procedimentos, vamos utilizar funções dos pacotes `janitor`, `stringr` e `stringi`, instalados e carregados anteriormente. Acompanhe o código a seguir com muita atenção aos comentários, que explicam as ações que estão sendo realizadas. Também o replique no seu computador:

```
# Criando um novo objeto com as correções feitas
cmi_sp <- cmi_sp_19_21 |>

  # Padronizando o nome das colunas utilizando a função `clean_names` do
  # pacote `janitor`
  clean_names() |>

  # Retirando a coluna `total`, pois não será necessária
  select(-total) |>

  # Retirando as linhas que não corresponde a nenhum distrito administrativo,
  # utilizando a função `filter()` do pacote `dplyr`
  filter(
    dist_administrativo_resid != "Endereço não localizado" ,
    dist_administrativo_resid != "Ignorado"
  ) |>

  # Padronizando as colunas de ano para uma coluna só utilizando a função
  # `pivot_longer()` do pacote `dplyr`. Note que vamos retirar a letra "x",
  # adicionada pela função `clean_names()`
  pivot_longer(
    cols = c(x2019, x2020, x2021),
    names_to = "ano",
    values_to = "cmi",
    names_prefix = "x"
    ) |>

  # Criando as novas colunas corrigidas
  mutate(

    # Primeiro criando a coluna que receberá os nomes dos distritos
    # administrativos corrigidos (passaremos para maiúsculo utilizando a
    # função `str_to_upper()` do pacote `stringr` e retiraremos os acentos
    # utilizando a função `stri_trans_general()` do pacote `stringi`)
    ds_nome = str_to_upper(dist_administrativo_resid) |>
      stri_trans_general("latin-ascii"),

    # Depois, a coluna que receberá os coeficientes corrigidos, utilizando a
    # função `str_replace_all()` do pacote `stringr`, e transformados para
    # valores numéricos utilizando a função `as.numeric()`
    cmi_num = str_replace_all(cmi, "\\,", "\\.") |>
      str_replace_all("\\-", NA_character_) |>
      as.numeric()
  )
```

Pronto, a base está agora devidamente corrigida. Caso tenha dúvida sobre as funções utilizadas, sugerimos a Unidade 3 do curso básico "Análise de dados para a Vigilância em Saúde".

Vamos visualizar as primeiras linhas da base corrigida. Digite no seu `RStudio`:

```
# Visualizando as primeiras linhas do objeto `cmi_sp`
head(cmi_sp)
```

```
#> # A tibble: 6 × 5
#>   dist_administrativo_resid ano   cmi   ds_nome           cmi_num
#>   <chr>                     <chr> <chr> <chr>               <dbl>
#> 1 Água Rasa                 2019  5,05  AGUA RASA            5.05
#> 2 Água Rasa                 2020  5,34  AGUA RASA            5.34
#> 3 Água Rasa                 2021  5,13  AGUA RASA            5.13
#> 4 Alto de Pinheiros         2019  3,23  ALTO DE PINHEIROS    3.23
#> 5 Alto de Pinheiros         2020  -     ALTO DE PINHEIROS   NA
#> 6 Alto de Pinheiros         2021  15,21 ALTO DE PINHEIROS   15.2
```

Com as colunas devidamente corrigidas, vamos importar o arquivo vetorial dos distritos administrativos de São Paulo. O arquivo {`sp_da.gpkg`} encontra-se disponível para *download* no menu lateral "Arquivos" do curso. Vamos utilizar a função `read_sf()` para a importação e vamos salvá-lo no objeto chamado {`distrito_adm_sp`}. Acompanhe o código abaixo e reproduza-o no seu computador:

```
# Importando o arquivo{`sp_da.gpkg`} para o `R`
distrito_adm_sp <- read_sf("Dados/sp_da.gpkg")

# Visualizando o objeto salvo
distrito_adm_sp
```

```
#> Simple feature collection with 96 features and 7 fields
#> Geometry type: POLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 313389.7 ymin: 7343743 xmax: 360618.2 ymax: 7416156
#> Projected CRS: SIRGAS 2000 / UTM zone 23S
#> # A tibble: 96 × 8
#>    ds_areamt ds_subpref             ds_sigla ds_nome    ds_ar…¹ ds_cd…² ds_co…³
#>        <dbl> <chr>                  <chr>    <chr>        <dbl> <chr>   <chr>
#>  1        NA SANTANA-TUCURUVI       MAN      MANDAQUI        NA 05      51
#>  2        NA MOOCA                  MOO      MOOCA           NA 25      53
#>  3        NA CASA VERDE-CACHOEIRINHA LIM     LIMAO           NA 04      50
#>  4        NA JABAQUARA              JAB      JABAQUARA       NA 15      38
#>  5        NA CIDADE ADEMAR          CAD      CIDADE AD…      NA 16      22
#>  6        NA ITAQUERA               ITQ      ITAQUERA        NA 27      37
#>  7        NA ITAQUERA               JBO      JOSE BONI…      NA 27      47
#>  8        NA VILA PRUDENTE          SLU      SAO LUCAS       NA 29      72
#>  9        NA MOOCA                  PRI      PARI            NA 25      56
#> 10        NA ITAQUERA               PQC      PARQUE DO…      NA 27      57
#> # … with 86 more rows, 1 more variable: geom <POLYGON [m]>, and abbreviated
#> #   variable names ¹ds_areakm, ²ds_cd_sub, ³ds_codigo
#> # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Note que, ao visualizarmos o arquivo importado, percebemos que é um arquivo vetorial de 96 polígonos e no sistema de coordenadas consta UTM 23S ("Projected CRS: SIRGAS 2000 / UTM zone 23S"). Logo, o arquivo já está em um sistema de coordenadas planas, não sendo necessária qualquer transformação.

Perceba também que a variável chamada `ds_nome`, que é a variável que contém os nomes dos distritos administrativos, já está em letra maiúscula e sem acentos sendo, portanto, dispensada de quaisquer correções.

Agora vamos unir o arquivo vetorial à tabela de coeficiente de mortalidade infantil. A variável `ds_nome`, que agora está presente nos dois arquivos, será a ligação entre os dois. Acompanhe o *script* abaixo e replique-o em seu `RStudio`.

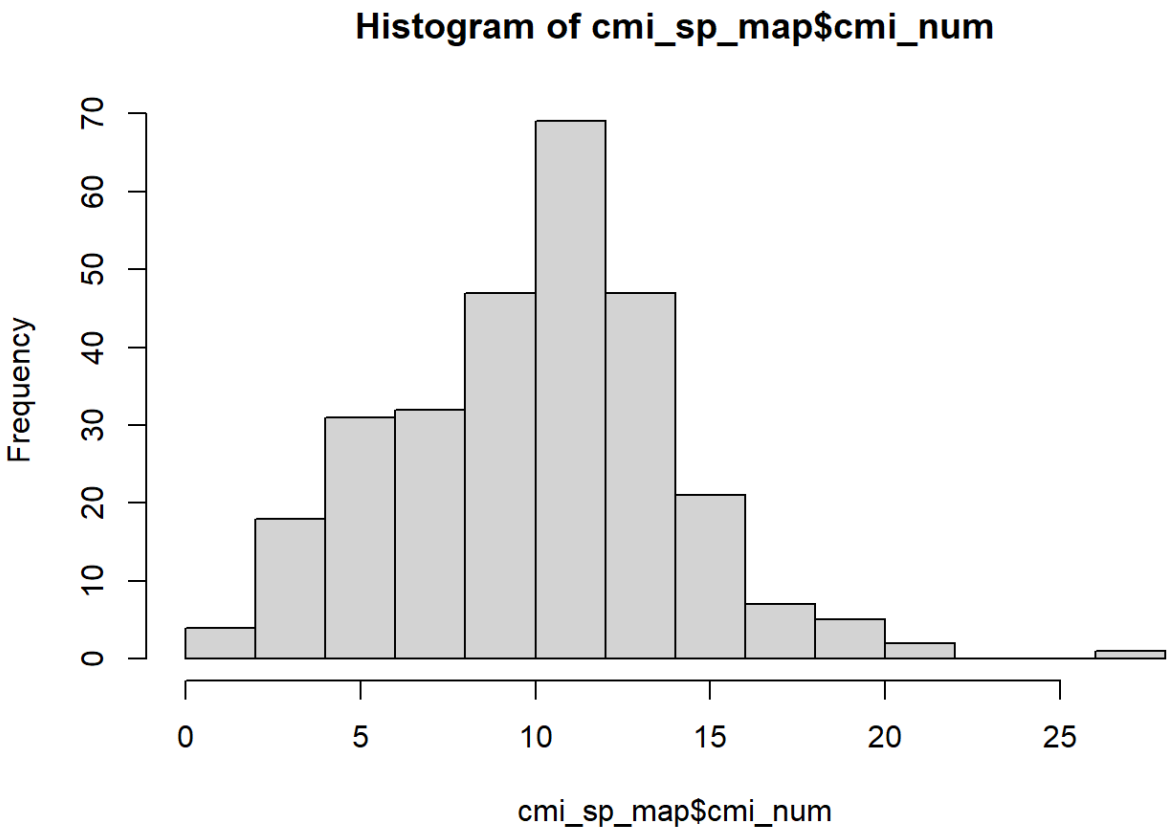
```
# Unindo a tabela `distrito_adm_sp` com `cmi_sp`
# com a função left_join() e salvando no objeto `cmi_sp_map`
cmi_sp_map <- left_join(

  x = distrito_adm_sp,
  y = cmi_sp,

  # A ligação é feita pela variável `ds_nome`
  by = "ds_nome"
)
```

Perfeito. Com a união pronta, vamos criar o mapa temático para comparar os três anos. Novamente vamos criar um mapa coroplético, destacando os distritos intramunicipais com maior coeficiente de mortalidade infantil com uma cor mais intensa e aqueles com menores valores com cores mais claras. Utilizaremos a variável "cmi_num" mas, antes, vamos visualizar como os valores estão distribuídos.

Vamos utilizar a função `hist()`, nativa da linguagem `R`, para visualizar os valores que vamos inserir no mapa. Essa função possui como argumento principal `x`, que é onde vamos explicitar a variável. Note que vamos usar o cifrão para indicar a variável "cmi_num" da base {`cmi_sp_map`}. Acompanhe o código abaixo e digite no seu computador:

```
hist(x = cmi_sp_map$cmi_num)
```

Figura 16: Visualização da distribuição de uma variável utilizando histograma.



Como vamos comparar por anos, temos que visualizar uma só variável. Dessa forma, podemos notar que há poucos valores baixos, poucos valores altos e a maioria não centro do gráfico. Com essa distribuição, vamos utilizar a classificação de Jenks para criar a legenda do mapa coroplético.

Para isso, vamos utilizar as funções `tm_shape()` e `tm_polygons()` do pacote `tmap`. A primeira insere uma camada referente ao mapa base e a segunda insere o mapa com a classificação dos valores coloridos conforme a paleta especificada. Os argumentos são muito parecidos com a função `mf_choro()`, utilizada anteriormente.

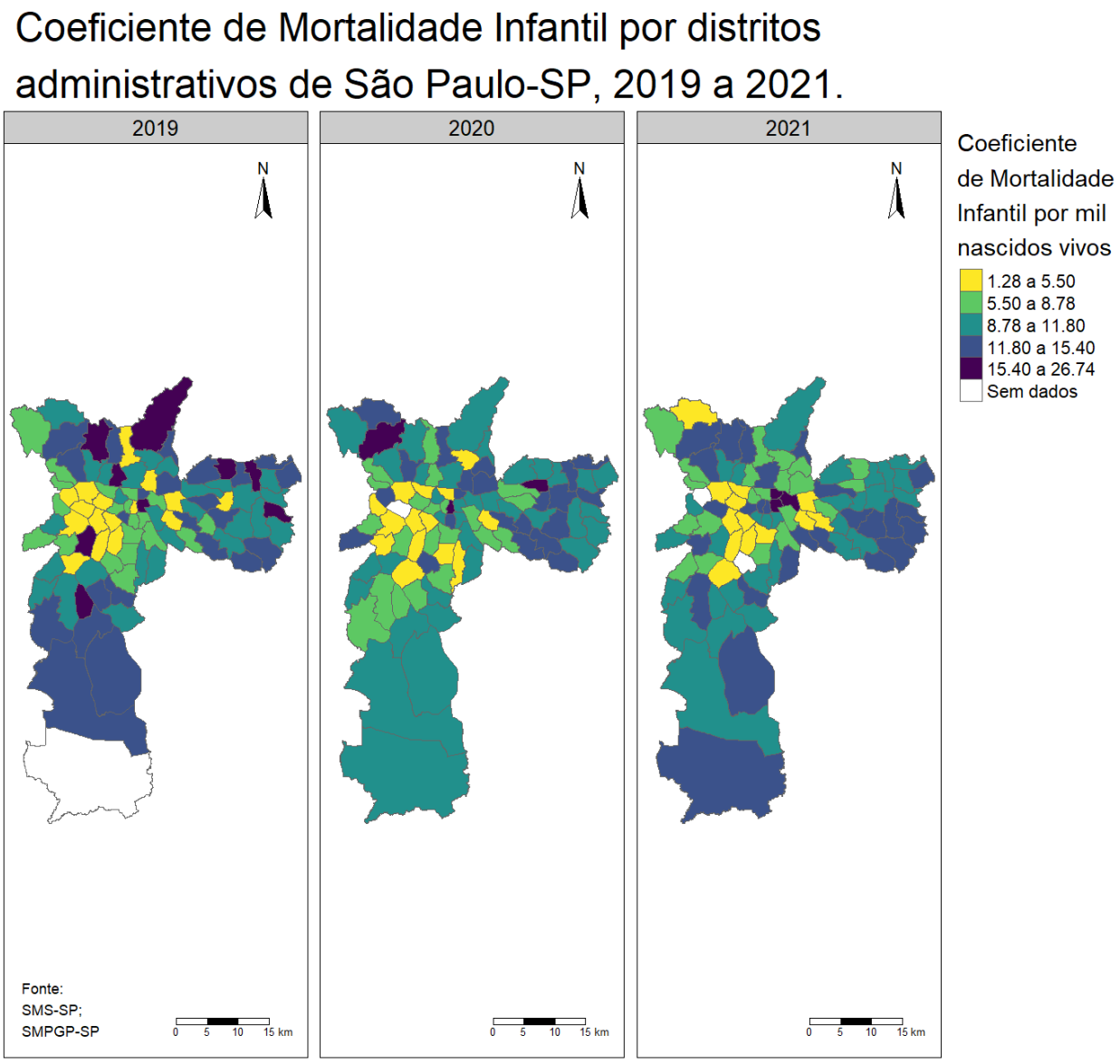
Também utilizaremos uma função bem interessante chamada `tm_facets()`, que cria um mapa para cada ano. Isso possibilitará compararmos o coeficiente de mortalidade infantil de forma única.

Para inserir o título, a fonte dos dados, a escala e a seta de norte, vamos utilizar as funções `tm_layout()`, `tm_credits()`, `tm_scale_bar()` e `tm_compass()`, respectivamente. **Note que o pacote `tmap` utiliza o símbolo de mais (+) para sobrepor as camadas e elementos definidos pelas funções**.

Acompanhe o código a seguir com especial atenção aos comentários pois apontaremos informações complementares para auxiliar no seu entendimento. Novamente replique o código no seu computador:

```
# Plotando o objeto `cmi_sp_map` com uso da função tm_shape().
# Esse será o nosso mapa base.
tm_shape(cmi_sp_map) +

  # Adicionando camada de mapa coroplético com a função tm_polygons()
  tm_polygons(

    # Definindo a variável que será utilizada para a cor de preenchimento
    col = "cmi_num",

    # Definindo os intervalos de valores e a classificação via "jenks"
    n = 5, style = "jenks",

    # Definindo a paleta de cores para "viridis". Para inverter a direção
    # das cores (do mais claro para o mais escuro), precisamos inserir
    # o símbolo de menos antes do nome da paleta.
    palette = "-viridis",

    # Definindo o título da legenda. Usamos o marcador "\n" para pular linha
    title = "Coeficiente \nde Mortalidade \nInfantil por mil \nnascidos vivos",

    # Definindo o separador dos coeficientes na legenda
    legend.format = list(text.separator = "a"),

    # Inserindo caixa adicional para a legenda onde não há dados
    colorNA = "white",  textNA = "Sem dados"
  ) +

  # Utilizando a função `tm_facets()` para criar um mapa para cada ano com o
  # argumento `by` e definindo uma linha para a disposição dos mapas com o
  # argumento `nrow`
  tm_facets(by = "ano", nrow = 1) +

  # Definindo a posição da seta de norte para a região
  # "superior e à direita" com a função tm_compass(), com
  # argumento `position`
  tm_compass(position = c("right", "top")) +

  # Inserindo a escala com a função tm_scale_bar()
  tm_scale_bar() +

  # Definindo o título do mapa e definindo o tamanho do texto
  # para encaixar melhor no título
  tm_layout(main.title = "Coeficiente de Mortalidade Infantil por distritos \nadministrativos de São Paulo-SP, 2019 a 2021.") +

  # Definindo informações complementares. Como teremos três mapas, um para
  # cada ano, inserimos um vetor contendo a fonte dos dados aparecendo
  # somente no primeiro mapa.
  tm_credits(text = c("Fonte:\nSMS-SP;\nSMPGP-SP","",""), position = c("left", "bottom"))
```

Figura 17: Exemplo de elaboração de mapa coroplético com comparação entre anos.



Muito bem, nosso mapa está pronto. Na Figura 17 é possível notar que há importantes desigualdades na ocorrência dos óbitos infantis entre as unidades administrativas de São Paulo, exigindo ações para compreender os motivos das diferenças e para eliminar as desigualdades observadas..

Percebeu como é interessante a comparação? Agora vamos entender um pouco sobre o mapa de fluxos?

5.4 Mapa de fluxos

Os mapas de fluxos representam os movimentos no espaço, onde a largura das linhas indica o volume do deslocamento de uma unidade territorial para outra, em números totais. Linhas mais grossas indicam maior quantidade de indivíduos se deslocando. No mapa apresentado, não são representadas rotas de deslocamento, mas sim os fluxos da origem e do destino.

A elaboração de mapas de fluxos não é um processo trivial, apesar de serem muito úteis na visualização do movimento em busca de atendimento de saúde. São necessários dados agregados que expressem a quantidade de deslocamentos de uma unidade para outra. A depender da ferramenta utilizada, são também necessárias as coordenadas da origem e do destino.

Para demonstrar a elaboração desse mapa, vamos utilizar variáveis selecionadas no Sistema de Informações Hospitalares (SIH), cujo diagnóstico principal foi por causas relacionadas à gravidez, parto ou puerpério, classificadas no Capítulo XV da CID-10. Realizamos um filtro das internações em que houve deslocamento da mulher entre municípios do estado do Acre, no ano de 2021.

Primeiramente, vamos importar a base que se encontra no formato "*.dbf". Para isso, vamos utilizar o pacote `foreing`. Execute o comando a seguir no seu computador:

```
# Carregando a base de dados para o objeto ac_internacao utilizando
# a função read.dbf()
ac_internacao <- read.dbf("Dados/ac_sih.dbf", as.is = TRUE)
```

Vamos visualizar apenas as primeiras linhas da tabela importada. Replique o código abaixo:

```
# Visualizando as primeiras linhas da tabela importada
head(ac_internacao)
```

```
#>    UF_ZI ANO_CMPT MES_CMPT MUNIC_RES VAL_TOT DT_INTER DT_SAIDA DIAG_PRINC
#> 1 120000     2021       01    120025  562.73 20210115 20210117       O821
#> 2 120000     2021       01    120010  109.24 20210114 20210116       O234
#> 3 120000     2021       01    120025  133.24 20210115 20210118       O234
#> 4 120000     2021       01    120010  110.24 20210117 20210118       O998
#> 5 120000     2021       01    120010  109.24 20210112 20210114       O234
#> 6 120000     2021       01    120010  180.62 20210102 20210104       O034
#>   MUNIC_MOV IDADE RACA_COR
#> 1    120010    16       03
#> 2    120010    24       03
#> 3    120010    16       03
#> 4    120010    22       03
#> 5    120010    16       03
#> 6    120010    28       01
```

Agora, para criar o mapa de fluxo, vamos considerar as variáveis que representam o município de residência (MUNIC_RES) da mulher como a origem do deslocamento e município de internação (MUNIC_MOV) como o destino do deslocamento que realizou a internação relacionada ao parto, gravidez e puerpério.

Em seguida, vamos contabilizar quantos deslocamentos ocorreram no ano de 2021 entre o par de município origem e destino. Para isso, vamos utilizar a função `count()` do pacote `dplyr`. Além disso, vamos excluir as internações onde a origem e o destino são iguais, ou seja, é o mesmo município e o paciente foi internado no município onde reside. Também vamos desconsiderar os deslocamentos com volume menor que 10. Essas operações serão salvas em um novo objeto chamado {`ac_capitulo_15`}. Acompanhe o código abaixo com atenção e replique no seu `RStudio`:

```
# Criando uma nova tabela com o nome ac_capitulo_15
ac_capitulo_15 <- ac_internacao |>

  # Contando a frequência de registros por município de residência
  # (coluna MUNIC_RES) e município de internação (coluna MUNIC_MOV)
  count(MUNIC_RES, MUNIC_MOV) |>

  # Filtrando os registros em que o município de residência difere
  #  do município de internação e com uma frequência maior ou igual a 10
  filter(MUNIC_RES != MUNIC_MOV, n >= 10)
```

Agora, com a frequência do deslocamento município de origem e município de destino calculada, vamos unir esses dados ao mapa. Dessa forma, criaremos as referências geográficas dos municípios de origem e de destino para a criação das linhas de ligação entre os municípios devidamente espacializadas. Para isso, vamos utilizar a função `mf_get_links()` do pacote `mapsf` e informar os seguintes argumentos:

- `x`: um arquivo vetorial de municípios (vamos utilizar a {`ac_municipios_5880`});
- `x_id`: a variável do arquivo vetorial a ser utilizada como o geocódigo (“cod_mun”);
- `df`: uma base de frequência de deslocamentos {`ac_capitulo_15`};
- `df_id`: um vetor contendo as variáveis de origem e de destino, respectivamente (no caso “MUNIC_RES” e “MUNIC_MOV”).

Observe e reproduza o código abaixo no seu computador:

```
# Criando as linhas de ligação com a frequência de deslocamento
ac_deslocamentos <- mf_get_links(
  x = ac_municipios_5880,             # um arquivo vetorial de municípios
  x_id = "cod_mun",                   # variável de ligação
  df = ac_capitulo_15,                # arquivo de frequência de deslocamentos
  df_id = c("MUNIC_RES", "MUNIC_MOV") # variáveis de origem e destino
  )
```

Ao visualizar as primeiras linhas do objeto `ac_deslocamentos`, você perceberá que, para cada par de município de residência e município de internação, foi calculada a frequência de deslocamentos entre estes municípios e sua devida ligação geográfica.

```
head(ac_deslocamentos)
```

```
#> Simple feature collection with 6 features and 3 fields
#> Geometry type: LINESTRING
#> Dimension:     XY
#> Bounding box:  xmin: 3250263 ymin: 8762143 xmax: 3580225 ymax: 8904710
#> Projected CRS: SIRGAS 2000 / Brazil Polyconic
#>   MUNIC_RES MUNIC_MOV   n                       geometry
#> 1    120001    120040 190 LINESTRING (3580225 8873782...
#> 2    120005    120010 122 LINESTRING (3250263 8762143...
#> 3    120005    120040  31 LINESTRING (3250263 8762143...
#> 4    120010    120040  53 LINESTRING (3337263 8770332...
#> 5    120013    120040  95 LINESTRING (3444644 8904710...
#> 6    120017    120040 163 LINESTRING (3485784 8808455...
```

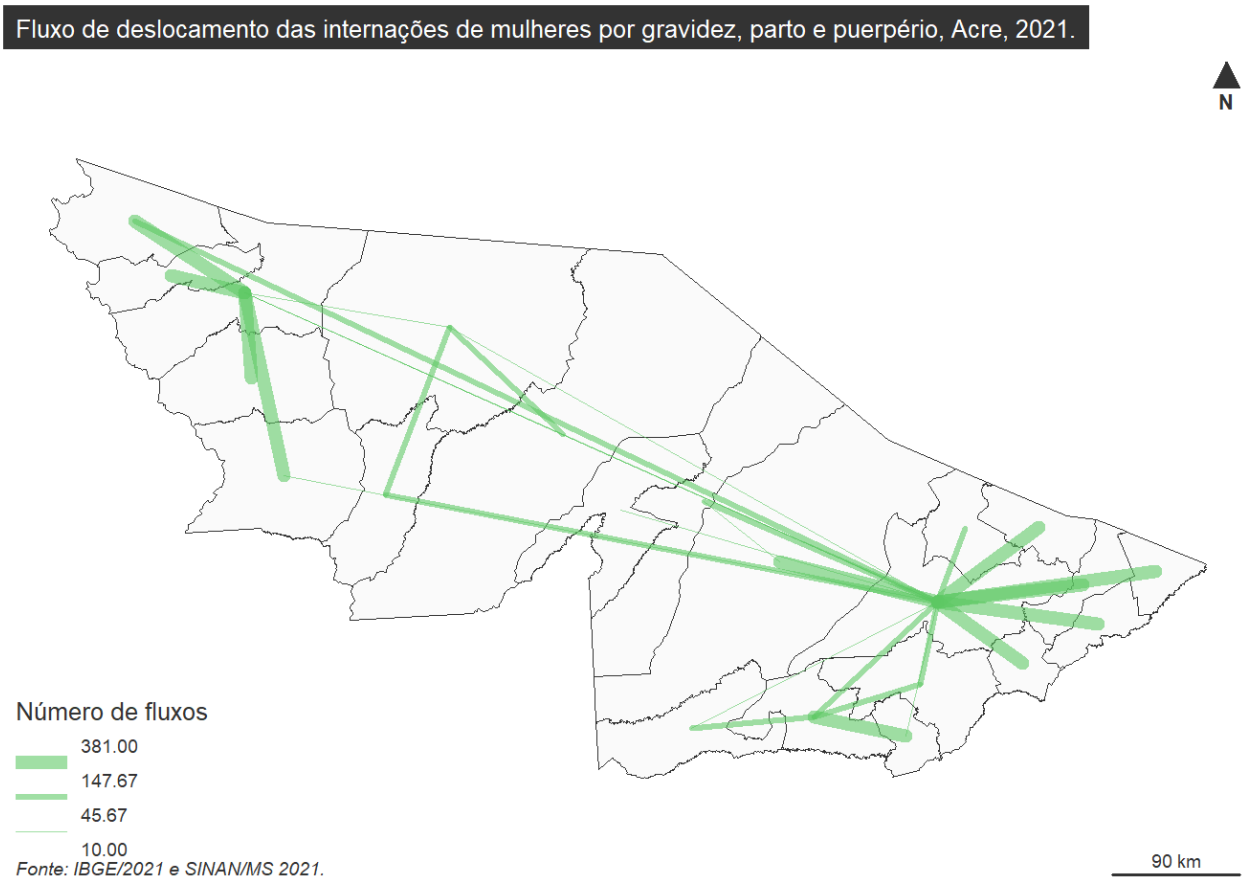
O próximo passo é definir o mapa base que, nessa prática, será {`ac_municipios_5880`}, e definir uma cor de preenchimento. Em seguida, vamos utilizar a função `mf_grad()` para representar os deslocamentos de forma gradual, aumentado a espessura da linha conforme aumenta a frequência de deslocamento entre os municípios. Vamos ainda definir os seguintes argumentos:

- `x`: a base de dados com os deslocamentos espacializados criados com a função `mf_get_links()`;
- `var`: coluna com os dados a serem plotados;
- `breaks`: classificação utilizada para divisão das classes (quantil);
- `n_breaks`: números de classes (como temos poucos registros, utilizaremos 3 classes);
- `leg_pos`: posição da legenda;
- `leg_title`: o título da legenda;
- `col`: cor das linhas de deslocamentos.

Observe o código abaixo com atenção e reproduza-o no seu `RStudio`.

```
# Plotando o objeto `ac_municipios_5880` com uso da função mf_map()
# O argumento "col" define a cor de preenchimento utilizando o código de
# cor hexadecimal
mf_map(ac_municipios_5880, col = "#FAFAFA")

# Inserindo camada indicando os descolamentos com uso da função mf_grad()
mf_grad(

  # Definindo o objeto com a frequência sobre os deslocamentos
  x = ac_deslocamentos,

  # Definindo a variável que indicará a espessura da linha conforme a frequência
  var = "n",

  # Definindo o número mínimo de intervalos para os valores de espessura
  nbreaks = 3,

  # Definindo o método de classificação dos intervalos para quantis
  breaks = "quantile",

  # Definindo a espessura da linha relativa a cada intervalo
  lwd = c(.7, 3, 7),

  # Definindo a posição da legenda para "inferior e à esquerda"
  leg_pos = "bottomleft1",

  # Definindo o título do gráfico
  leg_title = "Número de fluxos",

  # Definindo a cor da linha com o código de cor hexadecimal.
  # Para definir a opacidade, inserimos dois dígitos no final do
  # código. Dessa forma, nossas linhas ficarão mais transparentes,
  # melhorando a visualização
  col = "#5DC86395"
)

# Definindo a posição da seta de norte para a
# "superior e à direita" com a função mf_arrows()
mf_arrow(pos = "topright")

# Inserindo a escala com a função mf_scale()
mf_scale()

# Definindo o título do mapa e definindo o tamanho do texto
# para encaixar melhor no título
mf_title(txt = "Fluxo de deslocamento das internações de mulheres por gravidez, parto e puerpério, Acre, 2021.",
         cex = 0.8)

# Inserindo informações complementares
mf_credits(txt = "Fonte: IBGE/2021 e SINAN/MS 2021.", pos = "bottomleft")
```

Figura 18: Exemplo de elaboração de mapa de fluxos com layout completo.



Ao analisarmos a Figura 18 percebemos que dois municípios concentram as internações de mulheres relacionadas a gravidez, parto ou puerpério. Um na região sul e outro na região oeste do estado. Isso pode ser devido à localização da referência hospitalar nestes municípios, pactuação para atendimento feita entre gestores ou demanda espontânea das mulheres. Entretanto, é cabível avaliar se esse deslocamento gera algum tipo de impacto nas mulheres, causados pelo tempo do deslocamento, qualidade das vias de transporte utilizada ou até suporte no município de destino. Por ser um procedimento de especialidade básica, mais investigações podem ser envidadas.

Neste curso, mostramos muitos exemplos que utilizam as áreas geográficas de municípios em um estado. Mas, é totalmente possível replicar os códigos apresentados em áreas menores, como bairros de um município. O mapa pode evidenciar situações na realidade local que, quando a análise é apenas por tabelas, não é possível identificar. Com uma análise exploratória bem feita é possível enriquecer muito as análises da Vigilância em Saúde.

A linguagem `R` oferece excelentes ferramentas que facilitam desde abordagens básicas como importação e visualização de dados espaciais, até robustas técnicas de modelagem estatística espacial. Inclusive, há uma seção exclusiva para análise dados espaciais dentro do CRAN https://cran.r-project.org/web/views/Spatial.html, onde são listados dezenas de pacotes separados por tema.

Nossos cursos

Pronto, chegamos ao final deste curso! Agora você já conhece as principais ações para criar mapas epidemiológicos com o apoio da linguagem de programação `R`. Quer seguir a diante no aprendizado? Você encontrará outras etapas para aprofundamento das análises de dados em Vigilância em Saúde nos outros cursos. Aproveite e já faça sua inscrição nos cursos abaixo clicando nos *links*:

- Análises de dados para Vigilância em Saúde - curso básico.
- Visualização de dados de interesse para a Vigilância em Saúde.
- Produção automatizada de relatórios na Vigilância em Saúde.
- Construção de diagramas de controle na Vigilância em Saúde
- Linkage de bases de dados de saúde.
- Construção de painéis (dashboards) para monitoramento de indicadores de saúde.

Instituto Todos pela Saúde

ABRASCO
ASSOCIAÇÃO BRASILEIRA DE SAÚDE COLETIVA

UNA-SUS
Universidade Aberta do SUS

FIOCRUZ

UFPEL

UFSC