Journal of Linguistic Geography (2025), 1–15
doi:10.1017/jlg.2024.11

CAMBRIDGE UNIVERSITY PRESS

Article

Visualizing map data for linguistics using `ggplot2`: A tutorial with examples from dialectology and typology

Dana Roemling, Bodo Winter and Jack Grieve

Department of Linguistics and Communication, University of Birmingham, Edgbaston, Birmingham, UK

Abstract

Maps are important in many areas of linguistics, especially dialectology, sociolinguistics, typology, and historical linguistics, including for visualizing regional patterns in the distribution of linguistic features and varieties of language. In this hands-on tutorial, we introduce map making for linguistics using R and the popular package `ggplot2`. We walk the reader through the process of making maps using both typological data, based on the *World Atlas of Language Structures*, and dialect data, based on large corpora of language data collected from German and American social media platforms. This tutorial is intended to be of use to anyone interested in making maps of linguistic data, and more widely to anyone wanting to learn about mapping in R.

Keywords: typology; dialectology; dialectometry; geolinguistics; mapping; maps; sociolinguistics

1. Introduction

Maps are a form of data visualization instrumental to many different areas of research. In the same way that bar charts, line graphs, and other visualizations help to communicate quantitative information efficiently, maps assist the audience in understanding complex geographical information in a simple and concise way. Throughout the history of linguistics, maps have played a key role in many sub-fields of linguistics, especially dialectology, sociolinguistics, historical linguistics, and typology, but also in applied linguistics, field linguistics, language policy and planning, and research on language evolution.

Perhaps most notably, there is a long tradition in regional dialectology, where the analysis of dialect maps has often been the primary form of data analysis. Dialect atlases generally contain data on hundreds of linguistic variables from hundreds of locations distributed across a region of interest. To make sense of such complex data, dialectologists usually map the values of each linguistic variable across the region, inspecting these maps to identify individual patterns of regional linguistic variation, traditionally by manually plotting isoglosses that divide the region into sub-regions where the different values of the variable predominate (Chambers & Trudgill, 1998). Common patterns of regional variation are then identified by comparing these maps, ultimately providing a basis for identifying dialect borders. For example, Hans Kurath's *Word Geography of the Eastern United States* (1949) was based on maps of over 100 lexical alternations, each plotting the distribution of two or more variants across hundreds of locations where over 1,200 informants were interviewed. Based on these maps, Kurath plotted isoglosses for each alternation to identify underlying regional patterns in their variants. By then searching for bundles of isoglosses, he divided the eastern USA into three major dialect regions—the North, Midland, and South—which he argued corresponded to the migration of settlers originating from three different cultural hearths on the East Coast. This theory of American dialect regions, which was largely based on cartographic analysis, still dominates the field of American dialectology today (see Labov et al., 2006).

Maps have also long been used in typology and historical linguistics to classify and compare languages of the world. The *World atlas of language structures* (WALS; Dryer & Haspelmath, 2013) provides maps for a wide range of typological features measured across languages from around the world, including the distribution of phonological features like voicing, or morphosyntactic features like case and word order. Inspecting maps such as these at different levels of resolution allows for the discovery of areal patterns in the distribution of linguistic features. For example, there has long been debate over the status of Mesoamerica as a linguistic area. Based on an analysis of a wide range of features across languages of the region—including Oto-Manguean, Mixe-Zoquean, and Mayan languages, and several unaffiliated languages, including Taranscam and Xincan, as well as various languages from the surrounding area—Campbell et al. (1986) argued that Mesoamerica is an especially cohesive linguistic area, characterized by widespread linguistic similarities across different levels of linguistic analysis, including vowel harmony, relational nouns, and a vigesimal numeral system.

Despite the importance of mapping linguistic data in dialectology, historical linguistics, typology, and other fields of linguistics, making maps has traditionally been a major undertaking, often slow and laborious, requiring collaboration with geographers and cartographers, or access to expensive and

Corresponding author: Dana Roemling; Email: d.roemling@bham.ac.uk

Cite this article: Roemling D, Winter B, Grieve J. Visualizing map data for linguistics using `ggplot2`: A tutorial with examples from dialectology and typology. *Journal of Linguistic Geography* https://doi.org/10.1017/jlg.2024.11

complex commercial GIS (Geographic Information System) software. In recent years, however, it has become much easier for individual researchers in any field to produce high-quality maps using R, including with `ggplot2` (Wickham, 2016), the most widely used plotting package in R. Given the popularity of R and `ggplot2` in our field, and the importance of mapping and cartographic analysis across several sub-fields of linguistics, in this paper we provide a much-needed hands-on introduction to mapping for linguistics in R.

We start this tutorial by mapping typological data on a global scale using data from WALS. We then zoom in on mapping dialect variation in German-speaking countries and the USA based on large social media corpora. Along the way, we discuss several important concepts from cartography from a geolinguistic perspective, including map projections, map types, map resolution, class intervals, and legend construction. We make all code and data from this tutorial available to the reader to allow full replicability. Our hope is that this tutorial will be useful for anyone interested in visualizing map data, especially for linguists working with regional dialect and typological data.

2. Required R packages

First, we need to load all libraries used throughout this tutorial.[1]

```
library(tidyverse)
library(lingtypology)
library(sf)
library(rnaturalearth)
library(rnaturalearthhires)
library(mapproj)
library(maps)
library(classInt)
```

We have written this tutorial entirely using code based on the `tidyverse` (Wickham et al., 2019), which is a network of open-source R packages designed for data processing and analysis that greatly increases code complexity and legibility. A free online introduction to the tidyverse is given by Wickham and Grolemund (2016).

Most notably, we primarily use functions from the `ggplot2` package for making maps (Wickham, 2016). Within the tidyverse, `ggplot2` is the primary work horse for data visualization. `ggplot2` is very popular in academic research, including in linguistics (Winter, 2019). The package also provides efficient means for map making, as it interacts well with functions for mapping and spatial analysis from more specialized R packages. The core logic of `ggplot2` is made up of three components that any plot must have: data, aesthetic, and geoms. The data forms the substrate, which is projected onto geometric like lines, bars, points, polygons, etc. The aesthetics link specific columns of a data frame to specific features of the geoms. For a book-length introduction to `ggplot2`, we recommend Healy's *Data visualization: A practical introduction* (2018), but the code below can be followed even without having consulted `tidyverse` and `ggplot2` introductions prior to reading this tutorial.

We go into much more detail below, but for now, we want to describe the packages required for this tutorial. We use the `lingtypology` package (Moroz, 2017), which allows access to typological and other linguistic data. Over the last few years, mapping and spatial analysis in R has become integrated into the `tidyverse` with the development of the package `sf` (Pebesma & Bivand, 2022), which greatly simplifies working with spatial data in R. We use the package `sf`, which allows for spatial vector data to be analyzed in R. In addition, we use the `rnaturalearth` (Massicotte & South, 2023) and `rnaturalearthhires` (South et al., 2023) packages to map Austria, Germany, and Switzerland, but they can also be used to access global data.[2] The package `mapproj` (McIlroy, 2020) is needed in addition to the general `tidyverse` so map projections can be changed. The `maps` package (Becker et al., 2021) is used for mapping and projections. We use the final package, `classInt` (Bivand, 2020), to split data into class intervals to help visualize continuous data. The functions of all these packages interact well with the functions from the `tidyverse`. Other packages relevant to linguistic mapping are `glottospace` (Norder, 2022) and `lingtypR` (Becker, 2022). Even though we are not using them in this tutorial, they provide great resources as well.

All annotated code from this tutorial can also be found at https://github.com/danaroemling/mapping-for-linguists/ together with the linguistic data used.

3. Mapping typological features across the world

The first map we make shows word order patterns across languages of the world based on WALS. We begin by creating an empty world map. To retrieve the data for our map, we use the `map_data()` function, which is part of `ggplot2` and contains the geographical information necessary to plot the borders of countries around the world. We store this information in an R object named `world`.

```
world <- map_data("world")
```

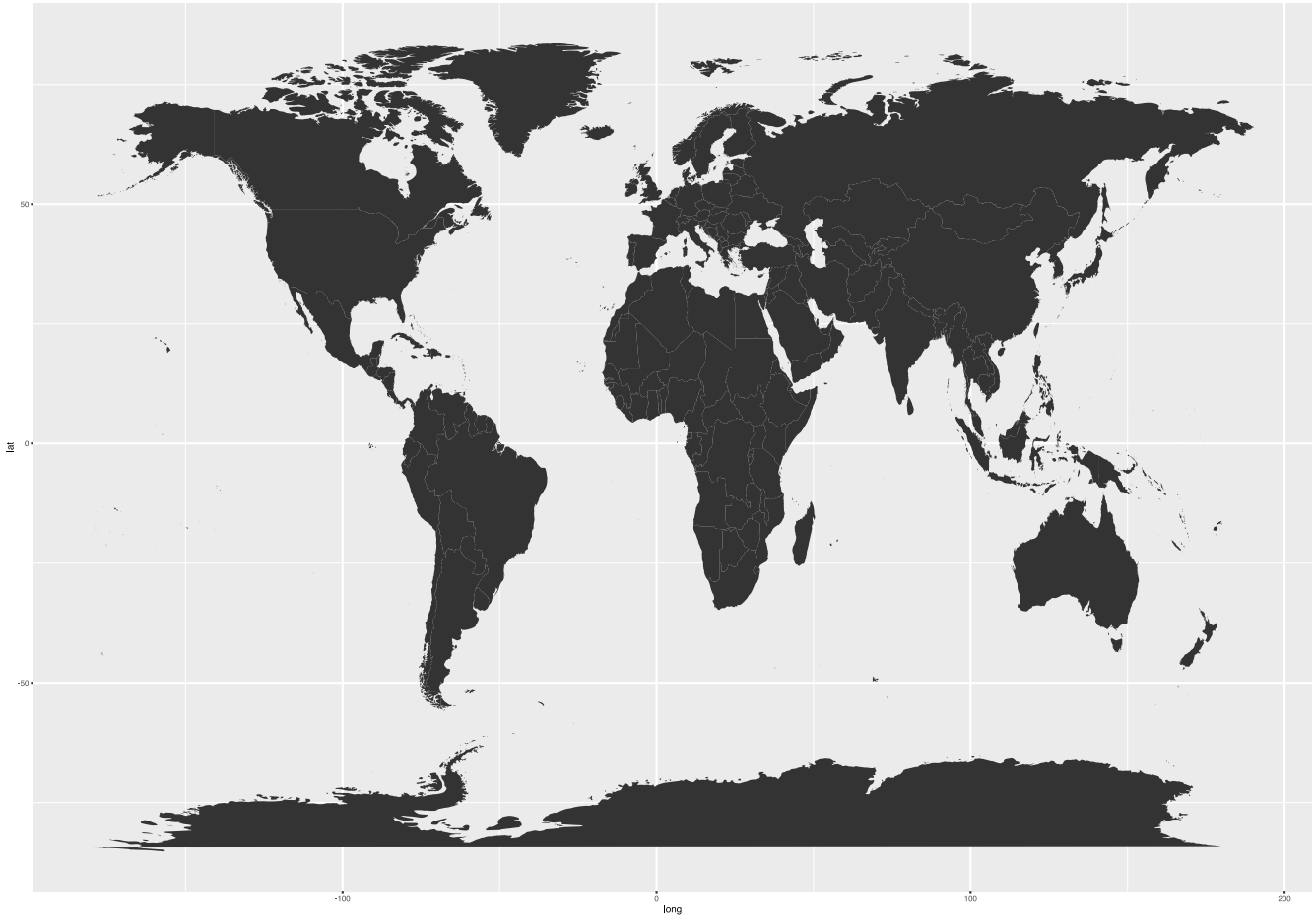
To see the data we use `head()`, to display the first six rows of this object, which contains longitude and latitude values that define part of the border of the island nation of Aruba (Caribbean Sea, north of Venezuela). When taken together, these separate longitude and latitude points create a "polygon" corresponding to the outline of the country. In this dataset, the `region` column contains information about the names of the country.

```
head(world)
```

long	lat	group	order	region	subregion
-69.89912	12.45200	1	1	Aruba	<NA>
-69.89571	12.42300	1	2	Aruba	<NA>
-69.94219	12.43853	1	3	Aruba	<NA>
-70.00415	12.50049	1	4	Aruba	<NA>
-70.06612	12.54697	1	5	Aruba	<NA>
-70.05088	12.59707	1	6	Aruba	<NA>

We can now use `ggplot` to produce a map of the world. The following complex `ggplot()` function call creates the map. We first specify that the data to be mapped is contained in the `world` data frame. Next, for `mapping`, we define the aesthetic mappings which specify what columns of the data are mapped onto which aesthetic features of the plot. Specifically, we map the `long` and `lat` columns onto the *x*- and *y*-axes, so the longitude and latitude values are projected onto the *x*- and *y*-axes. These mappings are defined inside the `aes()` function, the name of which stands for "aesthetics." The `geom_map()` function then retrieves these aesthetic mappings from the main `ggplot()` call so that it knows which columns to draw the *x*- and *y*-coordinates from to plot the polygons. Without `geom_map()`, we would get an empty grid, as geoms are the elements that populate a ggplot. `geom_map()` requires that we define which `map` should be used, so we pass the `world` data here again. We also define that we want the data points to be grouped as regions by using the `map_id()` function, so that we see countries.

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 1. Base map of the world.

```
world_plot <- ggplot(data = world,
                     mapping = aes(x = long,
                                   y = lat)) +
  geom_map(map = world,
           aes(map_id = region))
```

This ggplot is now saved in an object called `world_plot`. Map 1 is printed to RStudio's plotting device when calling this object by typing its name into the R console.

```
world_plot
```

An advantage of saving a plot in an R object like this is that it allows us to add additional features into the plot later without having to re-run code. For example, we can add linguistic data to the map, such as data from the *World Atlas of Language Structures* (Dryer & Haspelmath, 2013). The next function call imports data for feature 81 from the atlas using the `wals.feature()` function from the `lingtypology` package (Moroz, 2017). This feature contains information about the dominant word order of languages.

```
word_order <- wals.feature("81a")
```

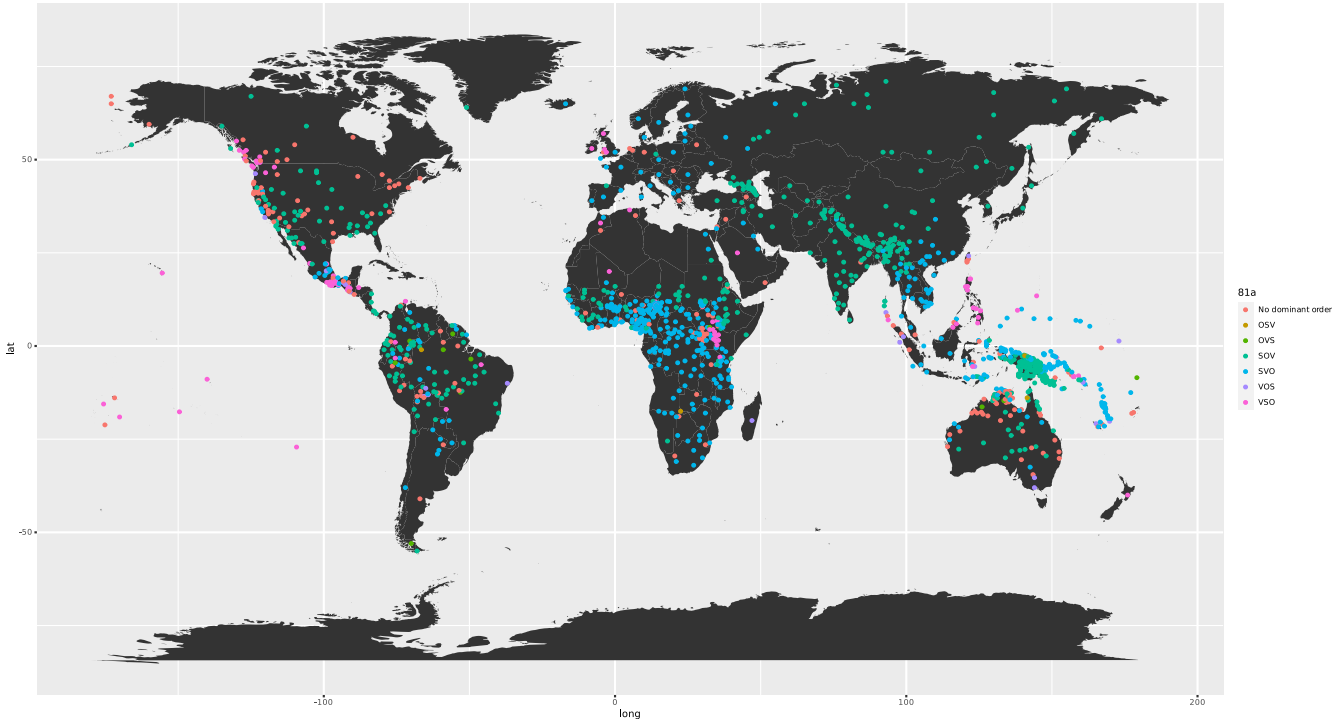
Feature 81a has seven levels, including Subject-Verb-Object (SVO), Object-Subject-Verb (OSV), or "No dominant order." For example, Finnish is classed as order Subject-Verb-Object, and Aneityum, a language spoken in Vanuatu, is classed as order Verb-Object-Subject. We can have a look using `head()` again.

```
head(word_order)
```

wals.code	81a	latitude	longitude	glottocode	language
aba	SOV	-4.00000	141.25	abau1245	Abau
abi	SVO	-29.00000	-61.00	abip1241	Abipon
abk	SOV	43.08333	41.00	abkh1244	Abkhaz
abn	SOV	-28.25000	136.25	arab1267	Arabana
abo	SOV	5.00000	36.75	arbo1245	Arbore
abu	SVO	-0.50000	132.50	abun1252	Abun

Now we add this information in another layer to our existing map that we previously stored in `world_plot`. To include data points visualizing the word order features, we add `geom_point()`. It is important to note that the data we are mapping comes from two different sources: we use the `world` object to create our base map, and we use the `word_order` object for the linguistic data. Both objects have columns that provide the necessary geolocation information. In `world` they are named `long` and `lat` and in `word_order` they are named `longitude` and `latitude`. These columns are mapped onto *x*- and *y*-coordinates of the points. Since this is an aesthetic feature of the plot that depends on the data, we have to wrap these mappings in `aes()` again. We also additionally map categorical information about the linguistic feature, contained in column 81a named after the corresponding chapter in WALS, onto the `color` aesthetic of the point geom, so that the points appear in different colors as a function of the different word order categories in the data. When referring to columns that start with numbers, we have to use two tick marks, `` `81a` ``. This code yields Map 2.

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 2. Map of the world showing word order categories.

```
word_order_plot <- world_plot +
  geom_point(data = word_order,
             mapping = aes(x = longitude,
                           y = latitude,
                           color = `81a`))

word_order_plot
```

Now that we have a basic map in place, we can tweak its look and feel. Adding `scale_color_brewer()` to our plot changes the discrete scale of the point colors to a "Brewer" palette, in this case specifically `Set2`, which is a colorblind accessible palette.[3] `ggtitle()` adds a custom title, and `xlab()` and `ylab()` achieve the same for the axis titles. We also add `theme_minimal()` to the plot. This particular theme is useful for mapping because it omits most other style elements except for a grid which can be helpful for orientation. We also use the function `theme()` to define our own additional plot options, such as moving the legend to the bottom of the map with the `legend.position` argument, and moving the title to the center using the `plot.title` argument. This code yields Map 3.

```
word_order_plot +
  scale_color_brewer(palette = "Set2") +
  ggtitle("Word Order in World Languages") +
  xlab("Latitude") +
  ylab("Longitude") +
  theme_minimal() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5))
```

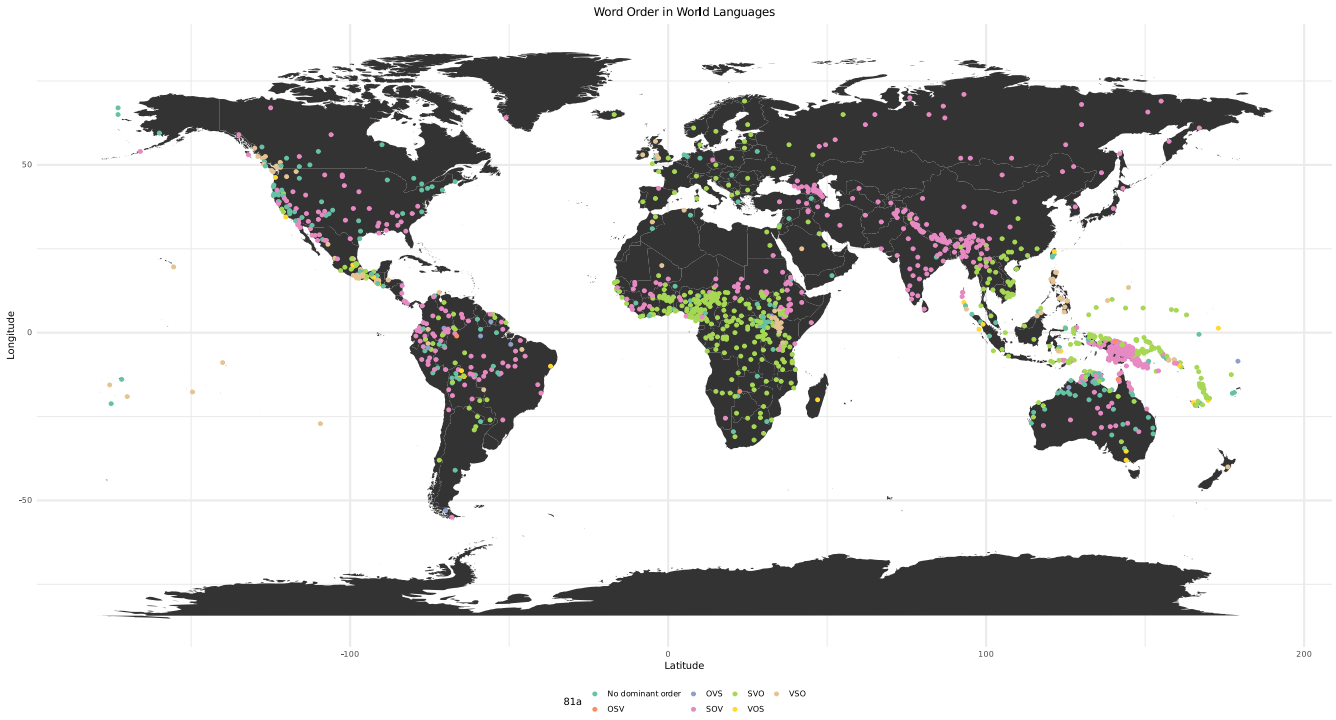
An important component of mapping is projection; the earth is a sphere, but the maps we use are two-dimensional. The coordinate system underpinning the map needs to "project" points from spherical space onto the two-dimensional map space. There are different ways of performing such "projections," each of which has ramifications for how mapped elements will appear to the human eye: for example, certain countries appear smaller or larger depending on which projection is chosen. The most widely used projection for maps of the world is the Mercator projection, which enlarges areas that are away from the equator, so, for example, Greenland appears a lot bigger than it actually is relative to other landmasses.[4] `geom_map()` uses a rectangular projection default, similar to the Mercator projection, in which longitude and latitude proportions are identical at the center of the map (cf. Map 1). Thus far, in our code, we have not provided R with details on how we wish our world map to be projected. This is not to say that it is a "wrong" way of doing it, but we have made a design choice by using the default.

For our map of word order across the world, we change not only the projection, but we center the map on the Pacific. We can shift the map by loading the data from the `maps` package and adding `wrap = c(-30, 330)`. This moves the world map so that the center meridian, often by default longitude 0° (Greenwich, UK), is now not shown as the center, but moved 30° to the left of the map. If we shifted the map by 180°, we would split up Africa and Europe.

```
world2 <- map_data("world", wrap = c(-30,330))
```

This means that rather than just changing the display in one of the plotting functions, we make the changes on the data level. Because we are now going to build our map based on the new `world2` data frame, which is centered at the Pacific, we need to ensure the `word_order` data conforms to this. This means the longitude values of `word_order` need to shift like they did for `world2`, so we add 360° to any longitude value below −30. To do this, we start by passing `word_order` at the beginning of the below function call and saving it to the same object as before. We

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 3. Map of the world showing word order with design alterations.

do this by using the |> pipe, an operator used for a sequence of actions. Here, the operator takes the `word_order` data frame and applies the `mutate()` function to it. The function `mutate()` creates a new variable (column) called `new_long` based on the outcome of the `ifelse()` function. This if-statement checks if the longitude is below −30 (first argument), and if so, it adds 360 to the longitude value (second argument); if the condition is *not* met, the initial longitude value is added (third argument).

```
word_order <- word_order |>
  mutate(new_long = ifelse(
    longitude < -30, longitude + 360, longitude))
```

If we build a new map using the new `world2` data frame and the changed `word_order` order data frame, we get Map 4.

```
centered_map <- ggplot(data = world2,
                       mapping = aes(x = long,
                                     y = lat)) +
  geom_map(map = world2,
           aes(map_id = region))  +
  geom_point(data = word_order,
             aes(x = new_long,
                 y = latitude,
                 color = `81a`)) +
  scale_color_brewer(palette = "Set2") +
  ggtitle("Word Order in World Languages") +
  xlab("Latitude") +
  ylab("Longitude") +
  theme_minimal() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5))

centered_map
```

Given this new map, we can see that even though Antarctica is a lot smaller than, for example, the Russian Federation, it appears very large in this map. This is because of how the map is projected. To change the projection we use `coord_map()` and pass the argument `projection` to it. We add this to our previous map, which yields Maps 5a and 5b. The Gilbert projection helps us see the spherical nature of the world more easily. Other projections, such as Mollweide for example, can help us see the true size of landmasses more accurately.

```
centered_map +
  coord_map(projection = "gilbert")
```

or

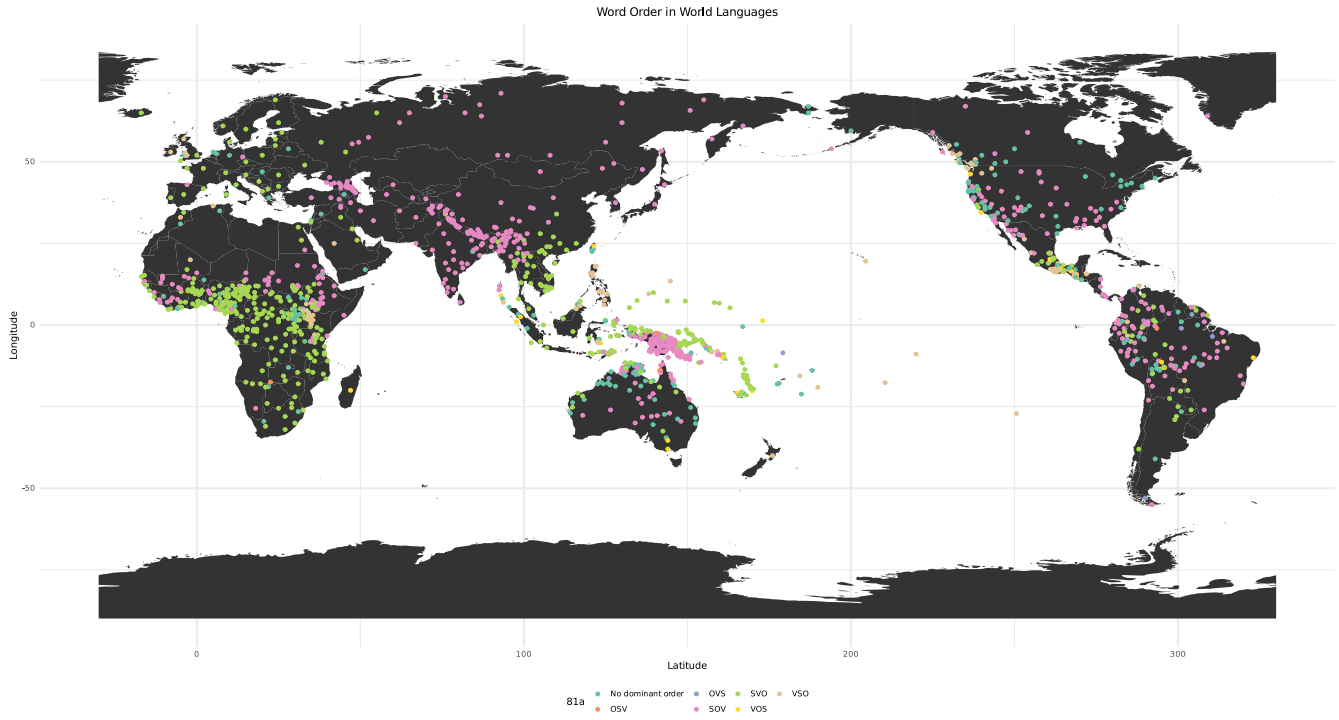
```
centered_map +
  coord_map(projection = "mollweide")
```

4. German-speaking area map

In the next part of this tutorial, we focus on visualizing dialect patterns in the German-speaking area (GSA), here simplified as Austria, Germany, and Switzerland. We load the data for this section using `read_csv()`. The original data set of social media postings, upon which this data is based, was compiled by Hovy and Purschke (2018). The example we are working with focuses on an alternation of *guck* and *schau* (Engl. 'to see').

```
gsa_data <-
read_csv("https://raw.githubusercontent.com/
danaroemling/mapping-for-linguists/main/
MAPPING_DIALECT.csv")
```

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 4. Map of the world showing word order centered on the Pacific.

This creates a data frame with cities, two features and their counts, the corresponding proportion of the features, and geolocation details. We can type `head(gsa_data)` to inspect our new dataset.

```
head(gsa_data)
```

City	Token1	Count1	Token2	Count2	Proportion	Long	Lat
<chr>	<chr>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Aachen	schau	117	guck	139	0.457	6.08	50.8
Aalen	schau	12	guck	4	0.75	10.1	48.8
Abstatt	schau	1	guck	1	0.5	9.29	49.1
Ahlen	schau	3	guck	1	0.75	7.90	51.8
Albstadt	schau	1	guck	1	0.5	9.02	48.2
Alfter	schau	2	guck	1	0.667	7.01	50.7

To map this dataset, we need a base map of the GSA first. Like the world data available in `ggplot2`, `rnaturalearth` provides us with polygons we can use to create this map, which we store below in `gsa_outline`. The list of country names could easily be changed to create new maps. We add the `returnclass` argument to change the output data format so that it is compatible with the `sf` package functions. We also add the argument `scale`, which changes the scale of the map. The larger the scale, the more detailed the borders appear. If `rnaturalearthhires` is not installed, `scale = "large"` needs to be omitted.

```
gsa_outline <- ne_countries(country = c("Austria",
                                        "Germany",
                                        "Switzerland"),
                            returnclass = "sf",
                            scale = "large")
```

After this, we can create the GSA base map as in Map 6. We import the polygons of the three countries in `sf` format, which means we use the corresponding `geom_sf()` function. `sf` stands for *simple feature* and is a format to handle spatial vector data. `geom_sf()` is a versatile function that can map points, lines, and polygons.

```
gsa <- ggplot(data = gsa_outline) +
  geom_sf()
```

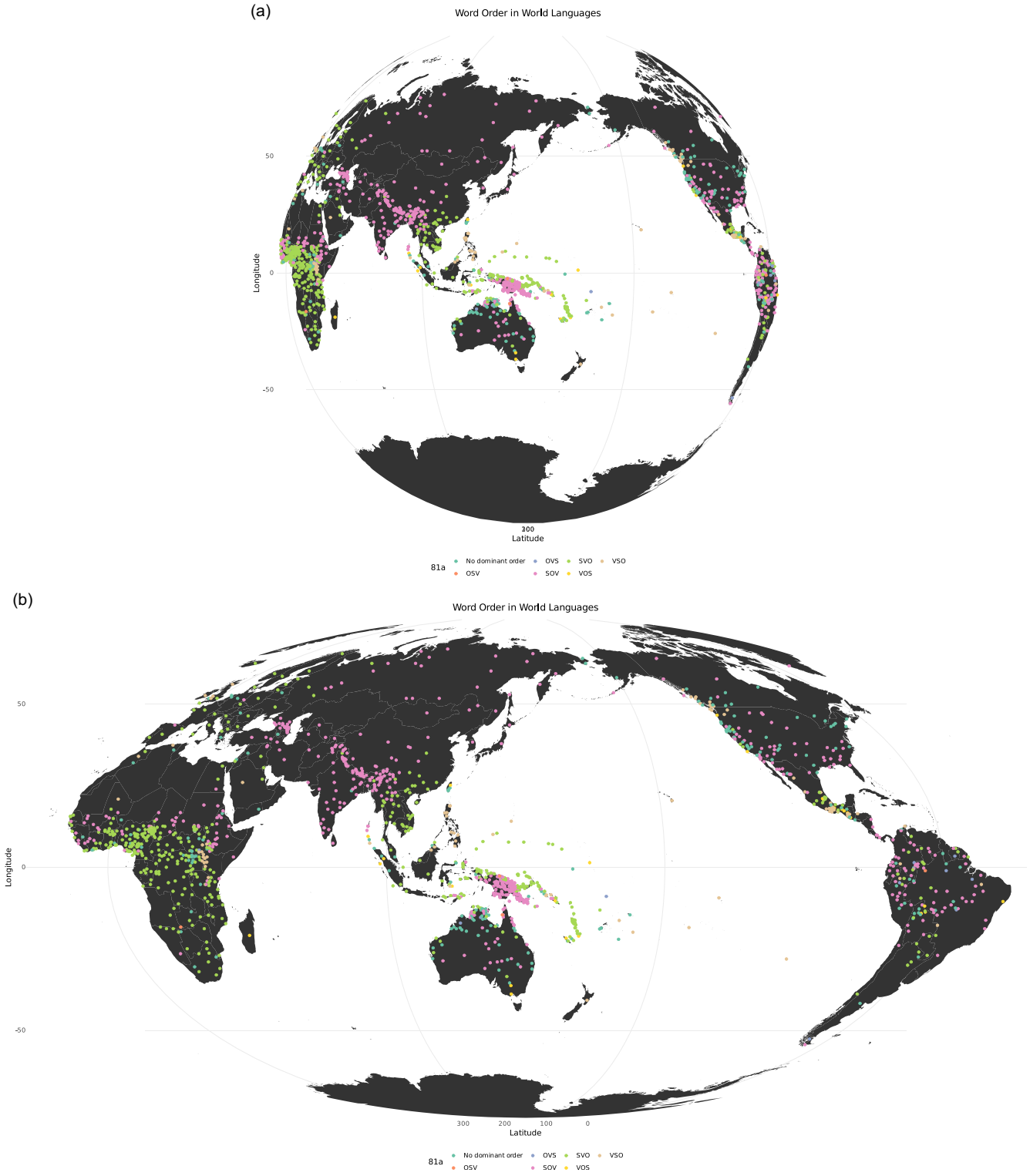
In the next step, we can combine the base layer with the data in `gsa_data`. We do this by adding them in the `geom_point()` layer we have already used. The `color` argument reflects the proportion, in this case, of the first variant (Token1), and the `size` argument reflects the combined count of our features. Here, `color` and `size` are aesthetics because they relate to columns in the data frame. The result of executing the code below is Map 7.

```
gsa_point <- gsa +
  geom_point(data = gsa_data,
             mapping = aes(x = Long,
                           y = Lat,
                           color = Proportion,
                           size = (Count1 + Count2)))

gsa_point
```

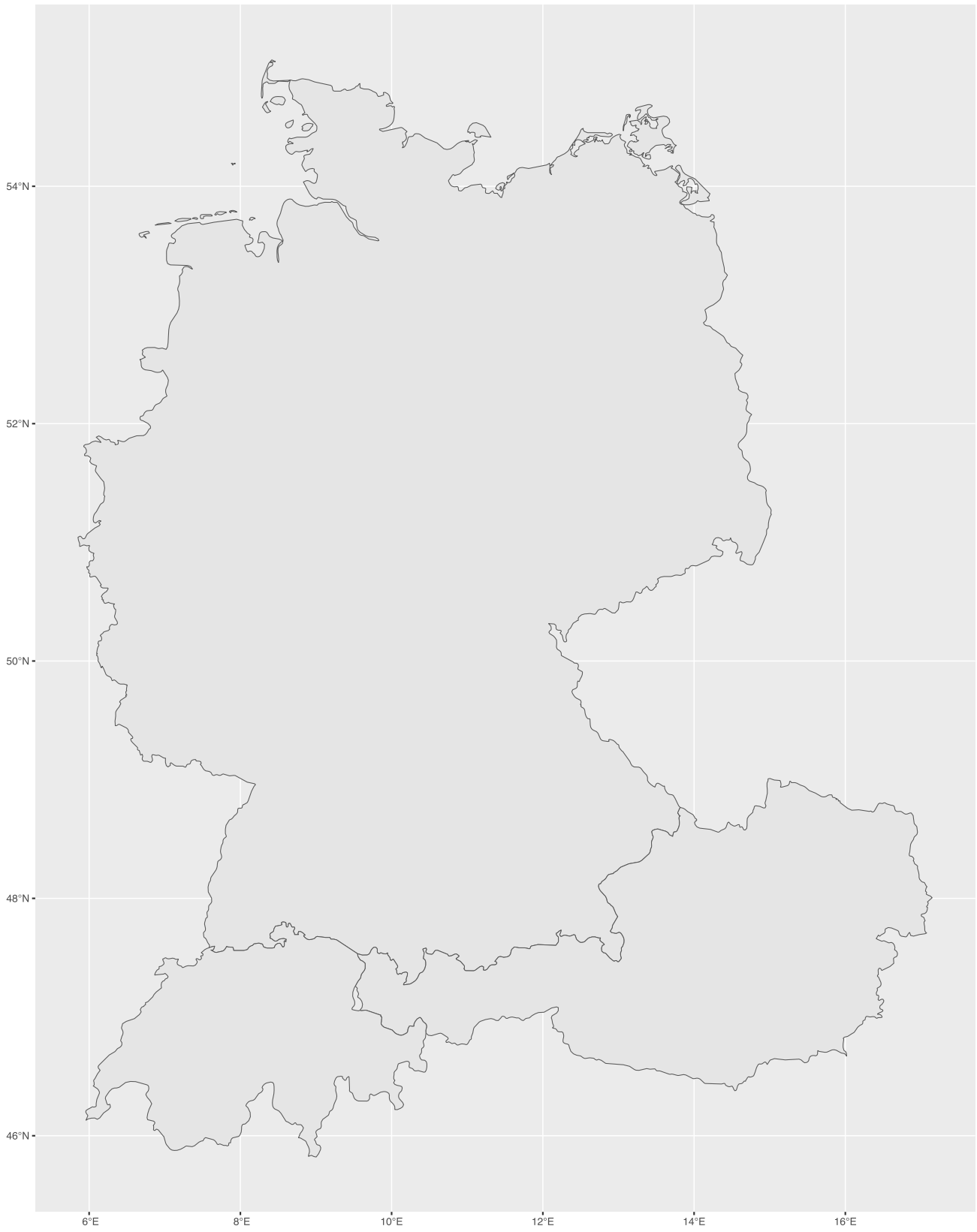
If we want to use more complex colors for the points, we can use the `scale_color_gradient()` function, with the `low` and `high` arguments set to two specific colors that form the ends of the color scale. Here, we choose a diverging color scheme going from green to purple, which means we use sequential greens and purples to show the proportion. With `guides()` we can change the display of the legend, which, in this case, allows us to omit the legend for the `size` aesthetics mapping by setting `size` to `"none"`. This leaves us with only a legend for the proportion. To change the title of the legend, we can use the `name` argument in the `scale_color_gradient()` function. This code then produces Map 8.

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



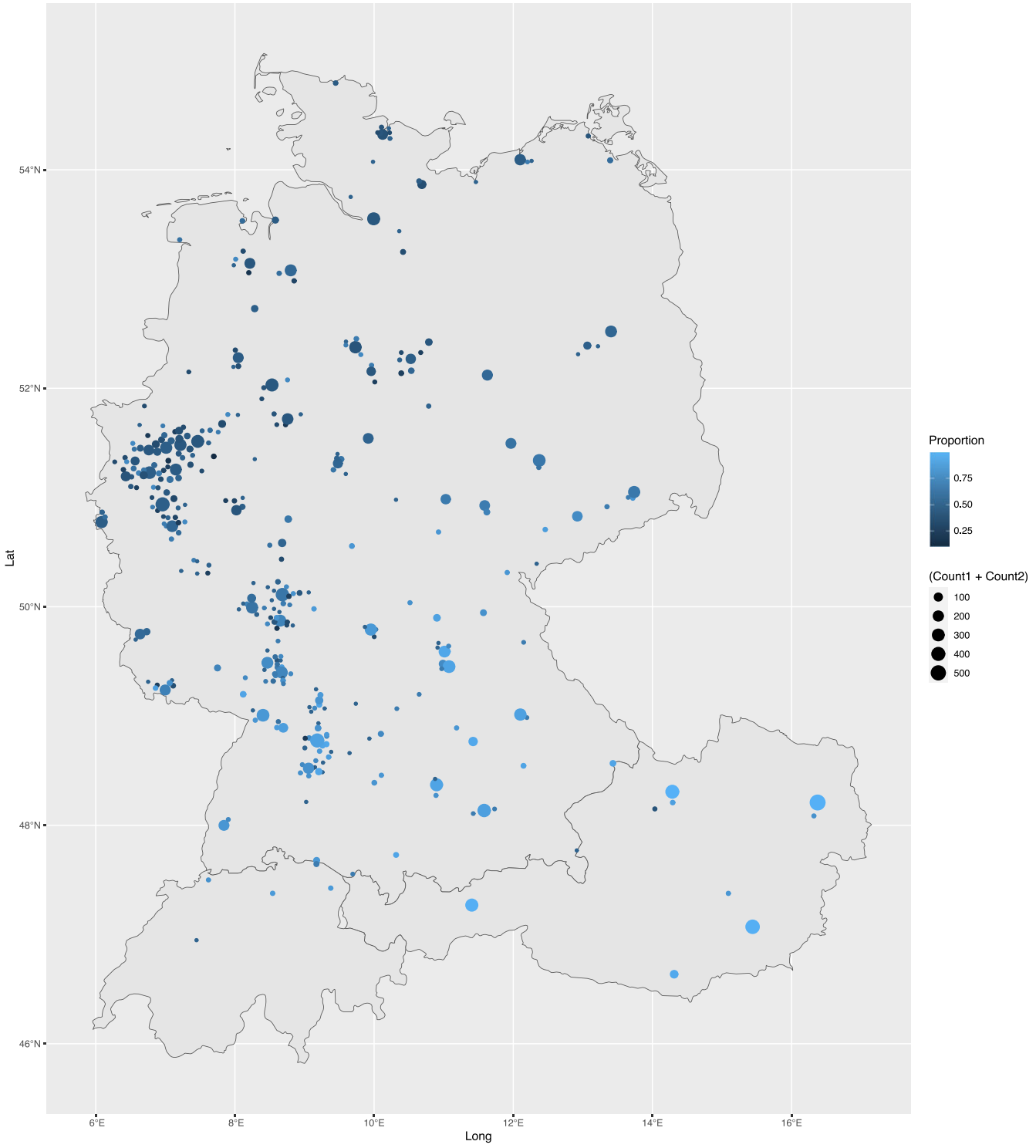
Maps 5a and 5b. Projected maps of the world centered on the Pacific in Gilbert (above) and Mollweide (below) projection.

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 6. Base map of the German-speaking area.

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 7. Dialect features mapped in the German-speaking area.

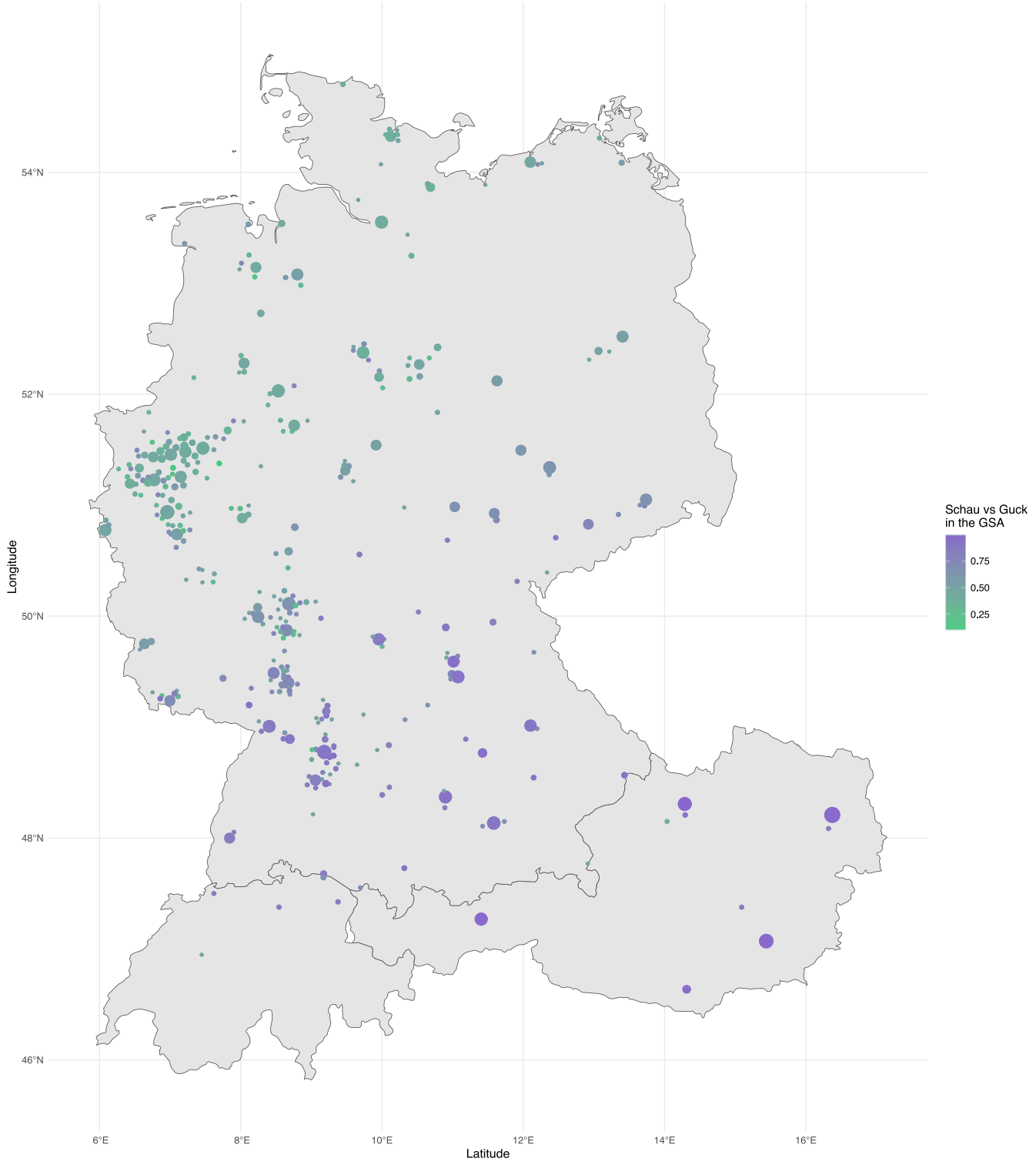
```
gsa_final <- gsa_point +
   scale_color_gradient(low = "seagreen3",
                        high = "mediumpurple3",
                        name = "Schau vs Guck \nin the GSA") +
  xlab("Latitude") +
  ylab("Longitude") +
  guides(size = "none") +
  theme_minimal()

gsa_final
```

5. US map

For the final maps of this tutorial, we focus on regional variation in the USA. We start by importing the respective geographical county information and save it as the object `us_geo` using the `st_as_sf()` function. This function allows us to transform location information into a different format, so that we can map

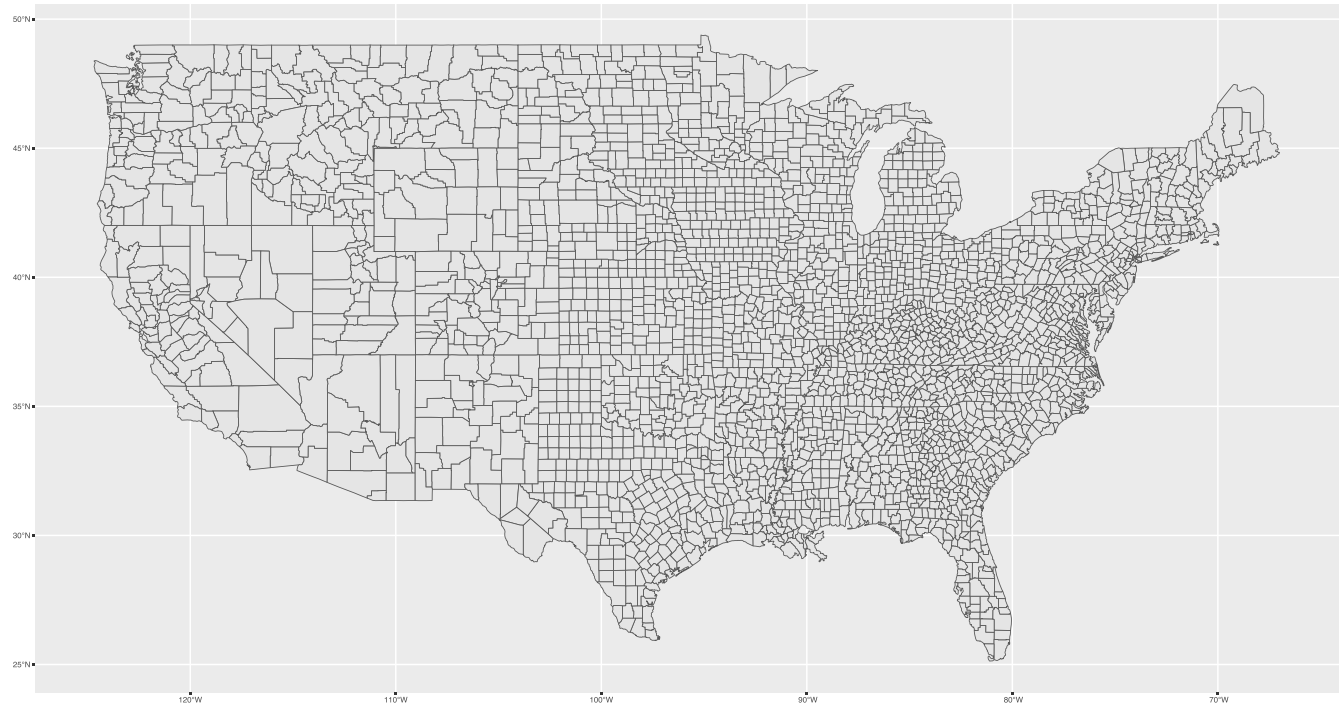
https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 8. Map with dialect features in the German-speaking area including design alterations.

features based on states or other relevant regions. Instead of having each point of longitude and latitude as its own row, we now store all location details for one county in the `geom` column. Essentially, they are grouped, here by county. The use of the double colon `::` is necessary because there are different functions in R that are called `map()`. With the prefix `maps::`, we can tell R to use the `map()` function specifically from the package `maps`, and not any of the functions with the same name from another package. More generally, the double colon notation is used whenever there are "naming conflicts" between packages loaded within the same R session, i.e. multiple functions with the same name. We have set the `plot` argument to `FALSE`, because we don't actually want to use

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 9. Base map of the USA.

the `map()` function itself to plot the data; instead, we use it to retrieve the data (without plotting it), which we then feed to `ggplot2` functions in a later step. The code also includes the argument `fill` set to `TRUE`. This means that the polygons are filled and it is not just their outlines that are used. Even though we are not plotting right now, this turns out to be important for the extraction of the polygons.

```
us_geo <- st_as_sf(maps::map(database = "county",
                             plot = FALSE,
                             fill = TRUE))
```

Now we can look at the data we have just stored:

```
head(us_geo)
```

ID	geom
alabama,autauga	MULTIPOLYGON (((-86.50517 3...
alabama,baldwin	MULTIPOLYGON (((-87.93757 3...
alabama,barbour	MULTIPOLYGON (((-85.42801 3...
alabama,bibb	MULTIPOLYGON (((-87.02083 3...
alabama,blount	MULTIPOLYGON (((-86.9578 33...
alabama,bullock	MULTIPOLYGON (((-85.66866 3...

We can now easily map the counties of the USA using the `ggplot()` function. To do this, we pass the `data` as an argument and then tell R to map it using the `geom_sf()` function, which yields Map 9.

```
ggplot(data = us_geo) +
  geom_sf()
```

The next step is importing data to map to the counties. We do this using the `read_csv()` function one more time. In this case, we import a dataset consisting of the relative frequencies of pronouns (per billion words) across the counties of the contiguous USA based on a multi-billion-word corpus of geolocated Twitter data from 2013–2014 (see Huang et al., 2016). Crucially, the data was normalized by dividing the total number of occurrences of each word in tweets originating from each county by the total number of words in those tweets. In this way, the authors controlled for variation in the amount of data observed across counties, which is closely related to population. We load the data and save it to the `us_data` object.

```
us_data <-
read_csv("https://raw.githubusercontent.com/
danaroemling/mapping-for-linguists/main/
MAPPING_PRONOUNS.csv")
```

```
head(us_data)
```

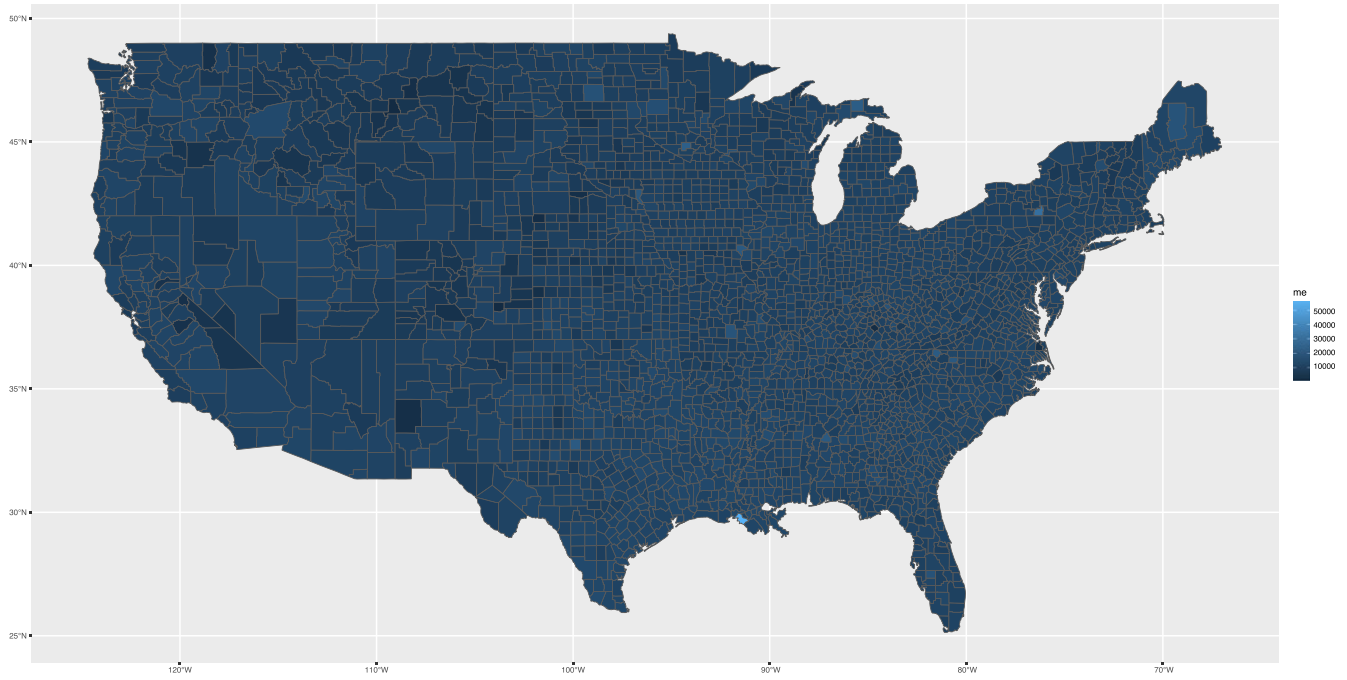
county <chr>	you <dbl>	he <dbl>	she <dbl>	it <dbl>
alabama,aut…	22019.	2699.	2326.	12973.
alabama,bal…	21465.	2471.	2174.	12354.
alabama,bar…	19131.	3370.	2494.	12132.
alabama,bibb	18699.	2513.	2127.	15910.
alabama,blo…	23783.	2623.	1979.	12696.
alabama,bul…	20630.	3736.	4071.	13520.

Conveniently, the names are already in a format that matches the data in `us_geo`. This is not always true, in which case the location names would need to be changed so they align. We now create a new object `us_geo_data` and match the information stored in the old objects by `ID` and `county`, so that the geolocation information and linguistic data are combined. We also use `na.omit()` to ensure the new data set does not have any missing values.

```
us_geo_data <- na.omit(left_join(us_geo,
                                 us_data,
                                 by = c("ID" = "county")))
```

Finally, we plot the new data frame `us_geo_data`. Here, we visualize the first person pronoun *me*. As before, we pass our data to the `ggplot()` function and add `geom_sf()`. We also pass aesthetics to `geom_sf()` to say that the `fill` is an aesthetic mapping that draws from the column called `me`.

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 10. First map of *me* distribution in the USA.

```
ggplot(data = us_geo_data) +
  geom_sf(mapping = aes(fill =  me))
```

The map we have just created is what is called a *choropleth map*. Essentially, this is a map where the regions are shaded to reflect the value of a variable in each region. Generally, the more distinct the color is from the background, the more interesting the value. This map, however, does not make it easy to see where the word *me* is more or less frequent. To make a better visualization, we must adjust the way the variation in value of the variable, in this case relative word frequency, is represented by the color scale. Colors are an essential part of data visualization, and they can not only drastically impact the visual appeal of a plot, but also its effectiveness in communicating data patterns. Certain colors increase interpretability of maps by making underlying spatial patterns easier or harder to appreciate.

In a choropleth map, it is especially important to consider how the values of the variable being mapped are split into color bands. There are two basic options. First, and perhaps most straightforwardly, we can split observed values of the variable into bands of equal length, which is called *equal breaks*. For example, if the values of a variable ranges from 0 to 1, we might split it into four equal-length intervals (i.e. 0–0.25, 0.26–0.50, 0.51–0.75, 0.76–1). This approach is how the color scale is defined in Map 10, although much finer intervals are used by default. Second, we can split the observed values into equal groups, which is called *quantile breaks*. For example, if the values of a variable range from 0 to 1, but most of the values are less than 0.5, we would use finer splits for lower numbers so that the same number of observed values were covered in each band (e.g. 0–0.10, 0.11–0.25, 0.26–0.50, 0.51–1). As we illustrate below, this approach often allows for patterns to be more easily discerned on maps as it maximizes distinctiveness in part of the variable's distribution where most values have been observed.

For our example of the pronoun data in the USA, we use the `classInt` package (Bivand, 2020) to split the relative frequencies of *me* by county into intervals using quantile breaks. Specifically, we use the `classIntervals()` function to split the variable `me` into five equal intervals using the `style` quantile.

```
pronoun_quantiles <- classIntervals(us_geo_data$me,
                                    n = 5,
                                    style = "quantile")
```

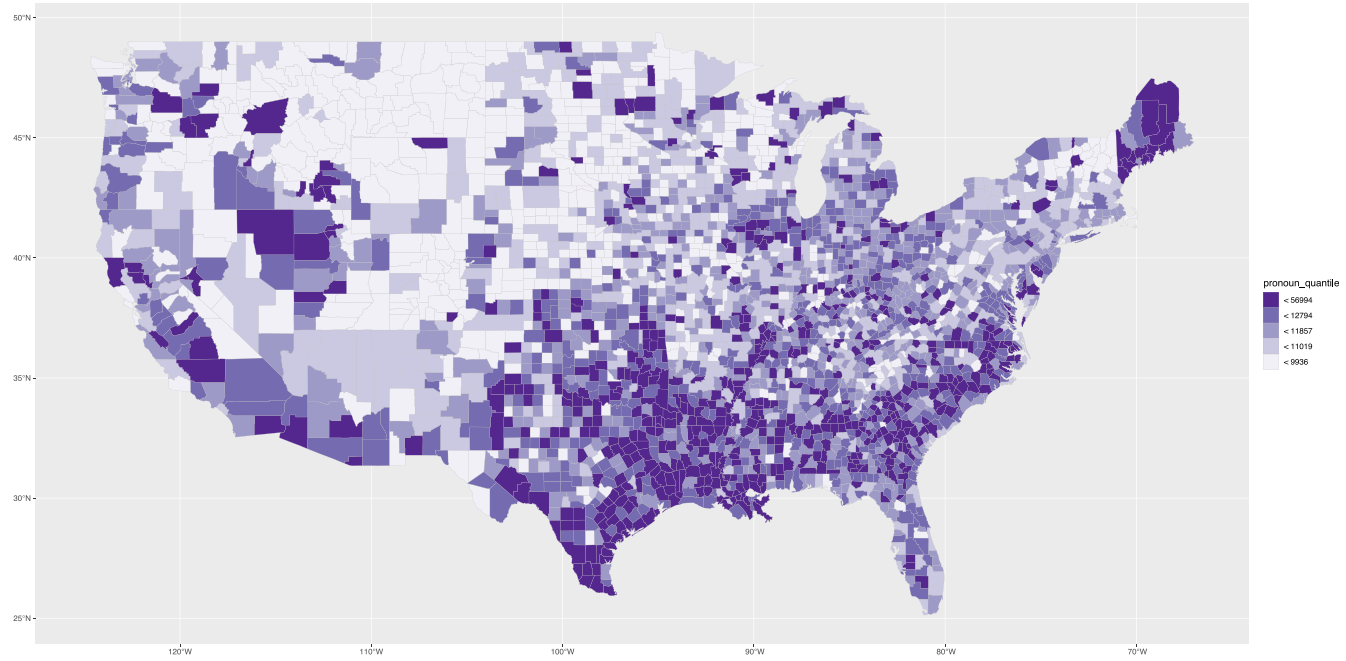
Next, we add this new information to our existing data set using the `mutate()` function, which adds the `pronoun_quantile` column to the data set `us_geo_data`. The `pronoun_quantile` column results from applying the `cut()` function to the `me` column; it returns a class interval based on the intervals we have defined previously. We also specify additional arguments in the `cut()` function: the argument `include.lowest = TRUE` ensures that no missing values are introduced, and the argument `dig.lab = 6` means that four digits for numbers are to be used, which eliminates the use of scientific notation for small numbers, which would make the legend harder to read. To change the quantile numbers into more readable breaks, we use `recode()` and exchange the numbers for just their maximum.

```
us_geo_data <- mutate(us_geo_data,
                      pronoun_quantile = cut(me,
pronoun_quantiles$brks,
                                             include.lowest =
TRUE,
                                             dig.lab=6))

us_geo_data$pronoun_quantile <-
recode(us_geo_data$pronoun_quantile,
       "[765.7,9936.44]" = "<9936",
       "(9936.44,11019.1]" = "<11019",
       "(11019.1,11856.5]" = "<11857",
       "(11856.5,12794.3]" = "<12794",
       "(12794.3,56994]" = "<56994")
```

Now we can map the variable using our new quantile class intervals. For this, we use the same code as before, but instead of passing the variable itself as a `fill` argument to the aesthetics mapping, we pass the intervals stored in `pronoun_quantile`.

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 11. Second map of *me* distribution in the USA introducing new color bands.

We also improve the design of the map to make it more readable. To allow for variation across counties, which are often quite small, we change the county border color to "grey" using the `color` argument and reduce the border width using the argument `lwd`, which is short for "line width." Since the legend is plotted in reverse, we change its order by using `guide_legend(reverse = TRUE)` in the `scale_fill_brewer()` function. Finally, we change the color palette of the `fill` by using `scale_fill_brewer()` with the specific palette "Purples". Note that, for choropleth maps, the most distinctive color traditionally contrasts with the background, so for a light background the darkest color should be the highest value of the mapped feature. That also means that choropleth maps traditionally do not use diverging color scales, like we have used for the GSA maps, so as to focus on the intensity of the color bands. Conventionally, four to eight color bands are used. The result of our code is Map 11.

```
ggplot(data = us_geo_data) +
  geom_sf(mapping = aes(fill = pronoun_quantile),
          lwd = 0.1,
          color = "grey") +
  scale_fill_brewer(palette = "Purples",
                    guide = guide_legend(reverse = TRUE))
```

As a last step, we change the projection and formatting of the map. There are conventions of which projection to use for which map, and the USA is often mapped using the Albers projection, which accounts for the border of the USA and Canada appearing curved instead of as a straight line in the Mercator and some other projections. We make this change by adding `coord_sf()`. Since we are using `geom_sf()`, we also need to pass the projection with `coord_sf()`. The argument we pass to `coord_sf()` is CRS, or Coordinate Reference System. We specify the ESRI CRS number 102003, which is the Albers projection for the contiguous USA. We change the legend title from the column name `pronoun_quantile` using the name argument in `scale_fill_brewer()`. The result is shown in Map 12.

```
ggplot(data = us_geo_data) +
  geom_sf(mapping = aes(fill = pronoun_quantile),
          lwd = 0.1,
          color = "grey") +
scale_fill_brewer(palette = "Purples",
                  guide = guide_legend(reverse = TRUE),
                  name = "'Me' Distribution \nper US
                  County") +
xlab("Latitude") +
ylab("Longitude") +
theme_minimal() +
coord_sf(crs = "ESRI:102003")
```

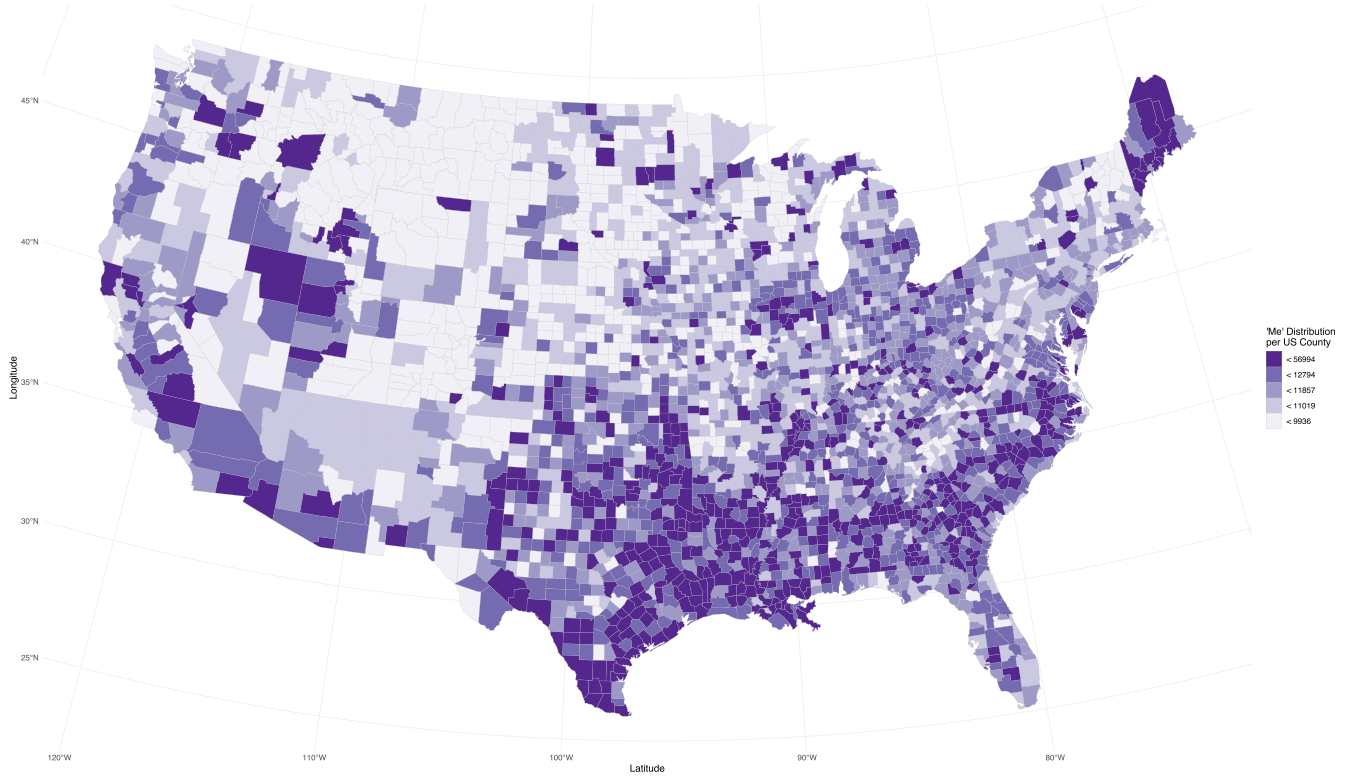
6. Next steps

Our goal in this tutorial has been to use linguistic examples from dialectology and typology to provide an introduction to mapping with R, especially with `ggplot2`, to guide linguists wishing to visualize their data. We have shown how to create maps based on different underlying map packages, and how these maps can then be combined with different kinds of linguistic data. Combining different ways of mapping into one tutorial creates an accessible walk-through, which is more intuitive given the familiar data employed, and which provides the code to help get interested readers started.

If this tutorial has sparked an interest in mapping, we recommend *Spatial data science* by Pebesma and Bivand (2022) and *Geocomputation with R* by Lovelace, Nowosad and Muenchow (2019) to explore mapping further, including outside the `tidyverse`. In particular, for a more in-depth explanation of coordinates, projection, and spheres we recommend Chapters 2 "Coordinates" and 4 "Spherical geometries" in Pebesma and Bivand (2022).

For example, in this tutorial we have not explored how to map with shapefiles. Basically, shapefiles are a way of storing the location information together with properties of those locations. If

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press



Map 12. Cleaned-up map of *me* distribution in the USA.

we take the example of US counties, we could have a shapefile with the names of the counties and the geometric information for each county. We could make maps with this shapefile just as we did with the data we extracted from the maps package. Shapefiles and `ggplot2` work well together, but they require additional steps of finding and loading in the shapefiles. They are also projected and face the same design choices we have elaborated on in this tutorial. There are also other formats of files that can be used to map and we refer to "Geographic data I/O" in Lovelace, Nowosad and Muenchow (2019).

We have also not covered any spatial statistics, instead, we focused on how to visualize the data at hand. For example, we could have explored the relationship between values of a variable and their location. This can be done using spatial autocorrelation analysis, as has been shown by Grieve (2016) for regional variation in the USA. We recommend the handbook chapter by Grieve (2017) to anyone interested in spatial statistics for dialectology. For a more introductory reading on spatial analysis in R, we recommend Brunsdon and Comber (2015), as well as Pebesma and Bivand (2022) and Lovelace et al. (2019) for more in-depth study. We also recommend consulting Winter (2019) for a hands-on general introduction to R with linguistic applications.

Finally, we would like to conclude this tutorial by reiterating that our focus has been on data visualization. That means it is supposed to provide the tools needed to make maps and include personal design choices. It is important to consider what a map is supposed to show and what features are important for a particular map. This should be reflected in the design choices. Maps are always an abstraction of the real world, no matter the design choices we make, and we always design them for a purpose.

Acknowledgments. We thank Dirk Hovy, Christoph Purschke, and Diansheng Guo for sharing their data with us, and Magnus Pharao Hansen for the typology example.

Dana Roemling was supported by the UKRI ESRC Midlands Graduate School Doctoral Training Partnership ES/P000711/1. Bodo Winter was supported by the UKRI Future Leaders Fellowship MR/T040505/1. Jack Grieve was supported by the Arts and Humanities Research Council (UK), the Economic and Social Research Council (UK), Jisc (UK) (Jisc grant reference number 3154), and the Institute of Museum and Library Services (US), as part of the Digging into Data Challenge (Round 3).

Competing interests. The authors declare none.

Notes

1 Packages can be installed using the `install.packages()` function; for example: `install.packages('tidyverse')`.

2 Installing `rnaturalearthhires` may not work with the standard `install.packages()` function. The tutorial can be used without the high resolution, or to install the larger package use:
`devtools::install_github("ropensci/rnaturalearthhires")` or `install.packages("rnaturalearthhires", repos = "https://ropensci.r-universe.dev", type = "source")`

3 To explore other colorblind accessible options, check
`library(RColorBrewer)`
`display.brewer.all(colorblindFriendly = TRUE)`
There is also a Greys color palette in this package, which can be useful for printing and publication.

4 See https://www.youtube.com/watch?v=kIID5FDi2JQ for a hands-on explanation of projections including the Mercator projection.

References

Becker, Laura. 2022. *lingtypR: Easy data manipulation for typologists* [Computer software]. https://gitlab.com/laurabecker/lingtypr

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press

Becker, Richard A., Allan R. Wilks, Ray Brownrigg, Thomas P. Minka & Alex Deckmy. 2021. *maps: Draw geographical maps* (R package version 3.4.0) [Computer software]. https://CRAN.R-project.org/package=maps

Bivand, Roger. 2020. *classInt: Choose univariate class intervals* (R package version 0.4-3) [Computer software]. https://CRAN.R-project.org/package=classInt

Brunsdon, C. & Comber, L. 2015. *An introduction to R for spatial analysis & mapping*. SAGE.

Campbell, Lyle, Terrence Kaufman & Thomas C. Smith-Stark. 1986. Meso-America as a linguistic area. *Language 62*(3). 530–570.

Chambers, J. K. & Peter Trudgill. 1998. *Dialectology*, 2nd edn. Cambridge University Press.

Dryer, Matthew S. & Martin Haspelmath (eds.). 2013. *The world atlas of language structures online*. Max Planck Institute for Evolutionary Anthropology. http://wals.info

Grieve, Jack. 2016. *Regional variation in written American English*. Cambridge University Press.

Grieve, Jack. 2017. Spatial statistics for dialectology. In Charles Boberg, John Nerbonne & Dominic Watt (eds.), *The handbook of dialectology*, 1st edn, 415–433. Wiley.

Healy, Kieran. 2018. *Data visualization: A practical introduction*. Princeton University Press.

Hovy, Dirk & Christoph Purschke. 2018. Capturing regional variation with distributed place representations and geographic retrofitting. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4383–4394. https://doi.org/10.18653/v1/D18-1469

Huang, Yuan, Diansheng Guo, Alice Kasakoff & Jack Grieve. 2016. Understanding US regional linguistic variation with Twitter data analysis. *Computers, Environment and Urban Systems 59*. 244–255.

Kurath, Hans. 1949. *A word geography of the Eastern United States*. University of Michigan Press.

Labov, William, Sharon Ash, Maciej Baranowski, Naomi Nagy, Maya Ravindranath & Tracey Weldon. 2006. Listeners' sensitivity to the frequency of sociolinguistic variables. *University of Pennsylvania Working Papers in Linguistics 12*(2). 105–129.

Lovelace, Robin, Jakub Nowosad & Jannes Muenchow. 2019. *Geocomputation with R*. CRC Press. https://geocompr.robinlovelace.net/index.html

Massicotte, Philippe & Andy South. 2023. *rnaturalearth: World map data from Natural Earth* (R package version 0.3.2) [Computer software]. https://CRAN.R-project.org/package=rnaturalearth

McIlroy, Doug. 2020. *mapproj: Map projections* (R package version 1.2.7) [Packaged for R by Ray Brownrigg and Thomas P. Minka and transition to Plan 9 codebase by Roger Bivand]. https://CRAN.R-project.org/package=mapproj

Moroz, George. 2017. *lingtypology: Easy mapping for linguistic typology* [Computer software]. https://CRAN.R-project.org/package=lingtypology

Norder, Sietze. 2022. *glottospace: Language mapping and geospatial analysis of linguistic and cultural data* (0.0.112) [Computer software]. https://github.com/SietzeN/glottospace

Pebesma, Edzer & Roger Bivand. 2022. *Spatial data science: With applications in R*. https://r-spatial.org/book/

South, Andy, Michael Schramm & Philippe Massicotte. 2023. *rnaturalearthhires: High resolution world vector map data from Natural Earth used in rnaturalearth* [Computer software]. https://github.com/ropensci/rnaturalearthhires

Wickham, Hadley. 2016. *ggplot2: Elegant graphics for data analysis* [Computer software]. New York: Springer.

Wickham, Hadley & Garrett Grolemund. 2016. *R for data science: Import, tidy, transform, visualize, and model data*. O'Reilly Media.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo & Hiroaki Yutani, H. 2019. Welcome to the Tidyverse. *Journal of Open Source Software 4*(43). 1686. https://doi.org/10.21105/joss.01686

Winter, Bodo. 2019. *Statistics for linguists: An introduction using R*, 1st edn. Routledge. https://doi.org/10.4324/9781315165547

https://doi.org/10.1017/jlg.2024.11 Published online by Cambridge University Press