AADA Archaeological Data Query and Map Visualisations in R

Meeli Roose

2024-05-17

1 Introduction

Purpose of the project

This markdown document presents the code used for repeating the steps involved in generating the archaeological data queries and map visualisations presented in the article by Pesonen et al 2023. The code also loads reference data for setting the map background (core map). To use the code, the datasets in XLSX format must be in the working directory.

Frequently used terms

- **AADA**: a database that contains information on archaeological artefacts from the Bronze Age, Iron Age, and Stone Age in Finland. It consists of three workbooks, respectively, with multiple sheets, and is intended to be used as an umbrella term when referring to all three datasets collectively. This document guides in binding these three datasets.
- **package**: a package in R is a collection of functions, data sets, and documentation that can be loaded into R and used to perform specific tasks.
- **library**: a library is a location where packages are stored on your computer. To use a package in R, you first need to load it into your R session using the library() function.
- **sf package**: a package used for processing vector spatial data. It provides a set of functions for reading, writing, manipulating, and visualizing spatial data using a simple feature data model. It supports common file formats like shapefiles, GeoJSON, and CSV, offering a unified way to handle spatial data regardless of the format.
- **function**: a function in R is a block of code that performs a specific task. Functions can take arguments as input, perform operations on the input, and return an output.
- **subset**: subsetting refers to the process of selecting a subset of data from a larger dataset.
- **data frame**: a data structure that is used to store tabular data, where each column can have a different data type (e.g., character, numeric, logical) and each row represents a single observation or case.
- **coordinate reference system (CRS)**: a standardized way of defining the location and orientation of spatial data in a two-dimensional or three-dimensional space. Transforming your datasets to a common coordinate system is important for ensuring that they align properly and can be accurately overlaid and analyzed together.
- **reference datasets**: data used as a background for maps or other geographic visualisations. They include geographic features, such as oceans, water bodies, as well political boundaries such as place names.

Data formats used in this project

In this project, the data is represented in two different formats: `sf` and `data frame`. A data frame is a common way of organizing data in R, and is similar to a spreadsheet with rows and columns. On the

other hand, `sf` - simple features - is a specific format used for spatial data, based on the Open Geospatial Consortium standard. In `sf`, each row in a table corresponds to a spatial feature and each column represents a specific attribute of that feature. The geometry of each feature is stored as a special column called ***geometry***, which contains information about the feature's shape, location, and other geometric properties. This project uses both formats to represent both the spatial and non-spatial aspects of the data.

2 Preparation

By following the steps outlined in this section, the data is prepared for visualisation in the subsequent sections of the project.

2.1 **Downloading R packages**: this subsection explains how to download and install the necessary R packages for the data visualisation.

2.2 **Loading an XLSX table and extracting spreadsheets**: this subsection describes how to load an XLSX table into R and extract specific spreadsheets for analysis.

2.3 **Adding spatial information to the datasets**: this subsection explains how to add spatial information to the datasets, such as latitude and longitude coordinates.

2.4 **Loading reference datasets**: this subsection describes how to load reference datasets, such as marine and land area, that are used in map visualisation.

2.5 **Performing coordinate transformations**: this subsection explains how to perform coordinate transformations, such as projecting coordinates into a different coordinate reference system.

2.6 **Defining the study area for AADA**: this subsection outlines how to define the study area for the visualisations, based on spatial boundaries.

2.1 Downloading R packages

2.2 Load an XLSX table and extract spreadsheets

Reading and saving the datasheets. 1) Set the file path and name of your Excel workbook. Note that you'll need to modify the file_path variable to match the path and name of your Excel workbook; 2) Load the entire workbook into R as a list of data frames; 3) Extract each spreadsheet into a separate data frame; 4) Naming separate data frames following AADA spreadsheets name conventions.

```
myfolder <- "C:/Users/YourUsername/Documents/R" # path to my folder
setwd("C:/Users/YourUsername/Documents/R") # use forward slashes ("/")
getwd() # check that the Excel files are in your working directory
dir() # alternatively, you can use the this function to list the files in a directory and
# confirm the file name and location
```

2.2.1 Loading Bronze Age dataset

When loading an Excel file into R, it typically reads only the first spreadsheet by default. Multiple excel sheets are read to R using `excel_sheets` function.

```
file_path <- file.path(myfolder, "AADA_Bronze_Age.xlsx")
sheet_names <- excel_sheets(file_path) # Get the names of all sheets in the workbook
sheet_data <- lapply(sheet_names, function(sheet) {
  read_excel(file_path, sheet = sheet)
})
```

The resulting `sheet_data` object is a list of data frames, where each element corresponds to a separate sheet in the workbook. The next step involves saving each sheet as a separate data frame with an assigned name. In programming, a **loop** is used to execute a set of instructions repeatedly. The code loops through each element in the `sheet_data` list and reads it as a separate data frame with the same name as in the Excel file spreadsheets.

```
for (sheet in sheet_names) {
  assign(sheet, read_excel(file_path, sheet = sheet), envir = .GlobalEnv)
}
```

2.2.2 Loading Stone Age dataset

```
file_path <- file.path(myfolder, "AADA_Stone_Age.xlsx")
sheet_names <- excel_sheets(file_path) # Get the names of all sheets in the workbook
sheet_data <- lapply(sheet_names, function(sheet) {
  read_excel(file_path, sheet = sheet)
})
for (sheet in sheet_names) {
  assign(sheet, read_excel(file_path, sheet = sheet), envir = .GlobalEnv)
}
```

2.2.3 Loading Iron Age dataset

```
file_path <- file.path(myfolder, "AADA_Iron_Age.xlsx")
sheet_names <- excel_sheets(file_path) # Get the names of all sheets in the workbook
sheet_data <- lapply(sheet_names, function(sheet) {
  read_excel(file_path, sheet = sheet)
})
for (sheet in sheet_names) {
  assign(sheet, read_excel(file_path, sheet = sheet), envir = .GlobalEnv)
}
```

2.3 Transforming data frames to spatial data

Assign geometrical component `sf` to the data frame using AADA's coordinate columns latitude (X) and longitude (Y). In other words, the regular data frame format is transformed to a spatial format, allowing e.g. spatial analysis and mapmaking. AADA's coordinate reference system (CRS) is "The World Geodetic System 1984 (WGS84)", with the EPSG identifier 4326 (https://epsg.io/4326) and Pseudo-Mercator projection.

```
BA_pottery = st_as_sf(BA_pottery, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
BA_stone = st_as_sf(BA_stone, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
BA_clay = st_as_sf(BA_clay, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
BA_bone = st_as_sf(BA_bone, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
BA_wooden = st_as_sf(BA_wooden, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
BA_bronze = st_as_sf(BA_bronze, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
BA_birch = st_as_sf(BA_birch, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
```

```
SA_pottery = st_as_sf(SA_pottery, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
SA_stone = st_as_sf(SA_stone, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
SA_clay = st_as_sf(SA_clay, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
SA_bone = st_as_sf(SA_bone, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
SA_wooden = st_as_sf(SA_wooden, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
SA_amber = st_as_sf(SA_amber, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
SA_birch = st_as_sf(SA_birch, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
```

```
IA_pottery = st_as_sf(IA_pottery, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_pottery_detailed = st_as_sf(IA_pottery_detailed, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_stone = st_as_sf(IA_stone, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_clay= st_as_sf(IA_clay, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_bone = st_as_sf(IA_bone, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_bronze = st_as_sf(IA_bronze, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_wooden = st_as_sf(IA_wooden, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_beads = st_as_sf(IA_beads, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_silver = st_as_sf(IA_silver, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_iron = st_as_sf(IA_iron, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
IA_amber = st_as_sf(IA_amber, coords = c("X", "Y"),crs = 4326, na.fail=FALSE)
```

2.4 Loading reference datasets

Reference map datasets form a core for the map visualisation. The code is loading different reference map datasets: ***region***, ***lakes***, ***ocean***, and ***cities***.

The `rnaturalearthdata` package includes several datasets that have been pre-loaded for convenience. These datasets include geometries for the world, states, waterbodies and e.g. place names.

Region

The region dataset is loaded using the `ne_countries()` function from the `rnaturalearth` package. The code first assigns a vector of country names that includes Finland and its neighboring countries. It then subsets the world data frame from `ne_countries()` to only include rows corresponding to those country names. The Aland region name is corrected to use the correct spelling of Åland.

```
# Load map data for world
world <- ne_countries(scale = "medium", returnclass = "sf")
# Subset to show only Finland and neighboring countries
countries <- c("Finland", "Sweden", "Norway", "Denmark","Russia", "Estonia","Aland")
region <- world[world$admin %in% countries,]
# Correct "Aland" special character
region$admin[region$admin == "Aland"] <- gsub("Aland", "Åland", region$admin[region$admin == "Aland"])
```

Lakes

The lakes dataset is downloaded using the `ne_download()` function from `rnaturalearth`. The `scale`, `category`, and `type` arguments are used to specify that the dataset should be at a 10m scale and include only lakes. The `returnclass` argument specifies that the dataset should be returned as an *sf* object.

```
# Load lakes dataset at 10m scale
lakes <- ne_download(scale = 10, category = "physical", type = "lakes", returnclass = "sf")
```

```
## Reading layer 'ne_10m_lakes' from data source
```

```
##   'C:\Users\meroos\AppData\Local\Temp\RtmpOwJgsc\ne_10m_lakes.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1355 features and 41 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -165.9656 ymin: -50.66967 xmax: 177.1544 ymax: 81.95521
## Geodetic CRS:  WGS 84
```

```
# Alternative download: set URL and download file
# url <- "https://www.naturalearthdata.com/http//www.naturalearthdata.com/
# download/10m/physical/ne_10m_lakes.zip"
# download.file(url, destfile = "ne_10m_lakes.zip")
# Unzip file
# unzip("ne_10m_lakes.zip")
# Read shapefile using sf
# lakes <- st_read("ne_10m_lakes.shp")
```

Ocean

The ocean dataset is downloaded in the same way as the lakes dataset, but with different arguments. The type argument is set to `geography_marine_polys` to retrieve polygons representing marine areas. The seas vector is used to subset the dataset to include only marine areas neighboring Finland.

```
ocean <- ne_download(scale = "medium", category = "physical", type = "geography_marine_polys",
                     returnclass = "sf")
```

```
## Reading layer 'ne_50m_geography_marine_polys' from data source
##   'C:\Users\meroos\AppData\Local\Temp\RtmpOwJgsc\ne_50m_geography_marine_polys.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 118 features and 37 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -85.19206 xmax: 179.9999 ymax: 90
## Geodetic CRS:  WGS 84
```

```
seas <- c("Gulf of Bothnia", "Baltic Sea", "Gulf of Finland", "Barents Sea")
ocean <- ocean[ocean$name %in% seas,]

# Alternative download: set URL and download file
# url_marine_polys <- "https://www.naturalearthdata.com/http//www.naturalearthdata.com/
# url continues: download/10m/physical/ne_10m_geography_marine_polys.zip"
# download.file(url, destfile = "ne_10m_geography_marine_polys.zip")
# unzip("ne_10m_geography_marine_polys.zip")
# ocean <- st_read("ne_10m_geography_marine_polys.shp")
# seas <- c("Gulf of Bothnia", "Baltic Sea", "Gulf of Finland", "Barents Sea")
# ocean <- ocean[ocean$name %in% seas,]
```

Cities

Downloading the point data for populated places (cities) at a "large" scale indicating a higher level of detail. This data is used in section *3.4. Iron Age* maps for advanced reference level, while zooming in the map.

```r
# Load point data for place names (populated places)
cities <- ne_download(scale = "large", category = "cultural", type = "populated_places",
                      returnclass = "sf")
```

```
## Reading layer 'ne_10m_populated_places' from data source
##   'C:\Users\meroos\AppData\Local\Temp\RtmpOwJgsc\ne_10m_populated_places.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 7342 features and 137 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -179.59 ymin: -90 xmax: 179.3833 ymax: 82.48332
## Geodetic CRS:  WGS 84
```

2.5 Creating AADA table

By combining SA, BA, and IA into a single data frame, a new data frame called AADA is created. In order to harmonize the columns, the mutate function is used, with pre-examined settings provided for your convenience. However, there is an issue with the **Subnumber** and **z** columns as they contain both integer and character data types simultaneously, which can be problematic. Following operations solve this issue.

```r
BA_pottery <- mutate(BA_pottery, Subnumber = as.integer(Subnumber))
BA_pottery <- mutate(BA_pottery, z = as.integer(z))
BA_birch <- mutate(BA_birch, z = as.integer(z))
BA_bronze <- mutate(BA_bronze, z = as.integer(z))
BA_stone <- mutate(BA_stone, z = as.integer(z))
BA_wooden <- mutate(BA_wooden, z = as.integer(z))
BA_clay <- mutate(BA_clay, z = as.integer(z))

SA_amber <- mutate(SA_amber, z = as.integer(z))
SA_birch <- mutate(SA_birch, z = as.integer(z))
SA_bone <- mutate(SA_bone, z = as.integer(z))
SA_clay <- mutate(SA_clay, z = as.integer(z))
SA_pottery <- mutate(SA_pottery, z = as.integer(z))
SA_pottery <- mutate(SA_pottery, Subnumber = as.integer(Subnumber))
SA_stone <- mutate(SA_stone, z = as.integer(z))
SA_wooden <- mutate(SA_wooden, z = as.integer(z))

IA_beads <- mutate(IA_beads, z = as.integer(z))
IA_pottery <- mutate(IA_pottery, Subnumber = as.integer(Subnumber))
IA_pottery <- mutate(IA_pottery, z = as.integer(z))
IA_pottery_detailed <- mutate(IA_pottery_detailed, Subnumber = as.integer(Subnumber))
IA_pottery_detailed <- mutate(IA_pottery_detailed, z = as.integer(z))
IA_silver <- mutate(IA_silver, z = as.integer(z))
IA_stone <- mutate(IA_stone, z = as.integer(z))
IA_iron <- mutate(IA_iron, z = as.integer(z))
IA_clay <- mutate(IA_clay, z = as.integer(z))
IA_bronze <- mutate(IA_bronze, z = as.integer(z))
IA_amber <- mutate(IA_amber, z = as.integer(z))
IA_wooden <- mutate(IA_wooden, z = as.integer(z))
IA_bone <- mutate(IA_bone, z = as.integer(z))

# Bronze Age contains 1474 observations, 47 variables
```

```
BA_all <- bind_rows(BA_pottery, BA_stone, BA_clay, BA_bronze, BA_bone, BA_wooden, BA_birch)
class(BA_all)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
# Stone Age contains 39991 observations, 41 variables
SA_all <- bind_rows(SA_pottery, SA_stone, SA_clay, SA_bone, SA_wooden, SA_amber, SA_birch)
class(SA_all)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
# Iron Age contains 6470 observations, 68 variables
IA_all <- bind_rows(IA_pottery, IA_pottery_detailed,IA_stone, IA_clay, IA_bronze, IA_bone, IA_wooden, I
class(IA_all)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

In this step, the three different AADA workbooks are merged into a single table to facilitate various analyses, such as archaeological period plotting. The bind_rows function is used to combine the SA_all, BA_all and IA_all tables, resulting in a new table.

```
# 47935 observations, 81 variables
AADA <- bind_rows(SA_all, BA_all, IA_all)
```

Following that, the `st_bbox()` function is utilized to calculate the bounding box for all the features in the table. This provides valuable information about the spatial extent of the data and can be used for various spatial operations. The resulting bounding box is stored in a variable called `limits_aada`.

```
class(AADA)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
AADA
```

```
## Simple feature collection with 47935 features and 80 fields (with 652 geometries empty)
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 19.55952 ymin: 59.03879 xmax: 35.59946 ymax: 70.0718
## Geodetic CRS:  WGS 84
## # A tibble: 47,935 x 81
##    Phase `Site id` Collection `Main number` Subnumber Municipality `Site name`
##  * <chr> <chr>     <chr>              <dbl>     <dbl> <chr>        <chr>
##  1 SA    porvoo 1  BM                  57.3        NA Porvoo       Ilola
##  2 SA    613010040 BM                  67.4        NA Porvoo       Böle
##  3 SA    613010040 BM                  67.4        NA Porvoo       Böle
##  4 SA    613010001 BM                  67.4        NA Porvoo       Bastäng
##  5 SA    613010002 BM                  67.4        NA Porvoo       Västanå I
##  6 SA    613010013 BM                  67.5        NA Porvoo       Henttala
##  7 SA    613010013 BM                  67.5        NA Porvoo       Henttala
##  8 SA    613010040 BM                  67.8        NA Porvoo       Böle
```

```
##  9 SA    613010040 BM                  67.8        NA Porvoo       Böle
## 10 SA    porvoo 3  BM                  68.1        NA Sipoo        Fredriksdal
## # i 47,925 more rows
## # i 74 more variables: p <dbl>, i <dbl>, z <dbl>, m1 <lgl>, m2 <lgl>, m3 <lgl>,
## #   n1 <lgl>, n2 <lgl>, n3 <lgl>, n4 <lgl>, p1 <lgl>, p2 <lgl>, r1 <lgl>,
## #   r2 <lgl>, Category <chr>, Type <chr>, Subtype <chr>,
## #   `Subtype (Finnish)` <chr>, Certainty <dbl>, `Main temper (Finnish)` <chr>,
## #   `Other tempers (Finnish)` <chr>, `Count (1-10)` <lgl>,
## #   `Count (10-100)` <lgl>, `Count (>100)` <lgl>, ...
```

```
limits_aada <- st_bbox(AADA)
limits_aada
```

```
##     xmin     ymin     xmax     ymax
## 19.55952 59.03879 35.59946 70.07180
```

3 Visualisations

This section provides guidance on creating visualisations to represent AADA data:

3.1 **Designing maps**: designing maps that communicate extracted spatial information.

3.2 **Data exploration and spatial query**: use of data exploration tools to identify patterns and trends in the data, such as finding specific features.

3.3 **Code for Pesonen et al. map visualisations**: code for replicating maps.

3.4 **Generating interactive graphics**: creating interactive graphics to explore the data in greater depth.

3.5 **Exporting visualisations**: exporting visualisations into .jpg format for use in reports or presentations.

By following the steps outlined in this section, clear and informative visualizations can be created to communicate the results of the analysis. Additionally, spatial query and data exploration tools are employed to gain further insights into the spatial patterns of the data.

To explore the data in the AADA table, following functions are used:

`colnames()`, `str()`, and `table()` to gain a quick overview of the table's structure and contents. These functions provide information on the variables (columns), their data types, and summary statistics.

3.1. Designing maps

Designing maps in R involves using various mapping packages. The `ggplot2` package is commonly used for this purpose and provides set of tools for creating maps with customized aesthetics. Key steps in designing maps include **loading map datasets**, **adding layers with geom functions** (such as `geom_sf` for spatial data), **setting fill colors and labels** with `scale` functions, **adjusting plot coordinates** with `coord` functions, and **applying themes** to fine-tune the plot appearance. Through a combination of these functions and tools, R allows creating maps with various levels of complexity and detail, making it a versatile option for data visualisation.

3.1.1 Mapping functions

In this document, the focus is on designing maps with `ggplot2`. The code chunks provided demonstrate how to create a plot using spatial data represented as `simple features (sf)`. The code uses various `ggplot2` functions to customize the plot's appearance and add various layers and labels.

Here's a breakdown of what each line of code used is doing:

- `ggplot()` initializes a new ggplot2 plot.
- `geom_sf()` adds a layer to the plot with spatial data represented as simple features (sf). The first `geom_sf` layer displays a map of the region with white fill and grey border. The second `geom_sf` layer adds a layer for lakes with a specified fill color and no border. The third `geom_sf` layer adds spearhead data with different colors based on the "Type" column.
- `scale_fill_manual()` sets the fill color for the lakes layer to a specific value
- `geom_sf_text()` adds text labels to the plot. The first ***geom_sf_text*** layer adds labels for ocean names with black text, adjusted to avoid overlap in the case of Barents Sea and Gulf of Bothnia. The second ***geom_sf_text*** layer adds labels for region admin with black text, adjusted to avoid overlap in the case of Åland. The third ***geom_sf_text*** layer adds a label for Lake Ladoga with black text.
- `ggtitle()` sets the plot title to "Mesolithic (8900-5100 calBC/AD) spearheads".
- `coord_sf()` sets the coordinates for the plot, with the x and y limits set to the limits_aada vector and the projected coordinate reference system (CRS) set to EPSG 3067.
- `theme_void()` removes the fill guide for the legend.
- `theme()` settings for the data frame.

3.1.2 Building the core of the map

In this chapter, a map background visualisation is created. The visualisation is comprised of several layers of spatial data represented as ***simple features (sf)*** to display a map of the region with lakes and text labels for 1) sea names (ocean data layer), 2) region admin, and 3) Lake Ladoga. The coordinates for the map are set explicitly, region with a white background and grey borders. The lakes and map background are filled with a blue color. The code adjusts the position of some text labels to prevent overlapping. Formatting options for the plot's background, border, and grid lines are set with `theme_void()`function. Finally, the ***scalebar*** and ***north arrow*** is added.

```
core_map <- ggplot() +
  geom_sf(data = region, fill="#E9E8E7", color="grey30") +
  geom_sf(data = lakes, fill="#eaf2f6", color="grey80") +
  geom_sf(data=ocean,fill="#F0F8FF")+
  geom_sf_text(data = st_crop(ocean, st_as_sfc(st_bbox(limits_aada))), aes(label = name),
               color="#2554C7", size = 2.5, color = "black",
               nudge_y = ifelse(ocean$name == "Barents Sea" |
                                  ocean$name == "Gulf of Bothnia", 40000,0)) +
  geom_sf_text(data=st_crop(region, st_as_sfc(st_bbox(limits_aada))), aes(label=admin),
               size=4,color = "black", nudge_y = ifelse
               (region$admin == "Åland", 0.3,0))+
  # adjust Aland position to avoid overlap
  geom_sf_text(data = lakes, aes(label = ifelse(name == "Lake Ladoga", "Lake Ladoga", "")),
               color="#2554C7", size = 2.5, color = "black") +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4])) +
  guides(fill = "none")+
  theme_void()+
  theme(panel.background = element_rect(fill = "#F0F8FF"),
        panel.border = element_rect(color = "grey30", fill = NA, linewidth = 0.25),
        panel.grid.major = element_blank(),
        legend.background = element_blank(),  # Remove legend background
        legend.key = element_blank(),          # Remove legend key background
        legend.title = element_text(color = "black"),  # Adjusting legend title color
        legend.text = element_text(color = "black"))
```

```
core_map
```


```
core_map <- core_map +
    annotation_scale(
    location = "br",  # bottom left
    width_hint = 0.4,  # Adjusting the width_hint to make the scale bar smaller
    text_cex = 0.6,
    distance = 2,
    style = "ticks"  # "bar" or "ticks" based on your preference
  ) +
  annotation_north_arrow(
    location = "br",
    which_north = "grid",
    pad_x = unit(0.4, "cm"),
    pad_y = unit(0.6, "cm"),
    height = unit(0.8, "cm"),  # Adjust the height to make it smaller
    width = unit(0.8, "cm"),   # Adjust the width to make it smaller
    style = north_arrow_fancy_orienteering(
      line_width = 1,
      line_col = "black",
      fill = c("white", "black"),
      text_col = "black",
      text_size = 5,
      text_angle = 0))
core_map
```

3.2 Data exploration

3.2.1 Basic functions

This chapter provides a set of basic functions that can be utilized for dataset exploration. These functions are essential in understanding the structure and content of a dataset.

- `str()` examining the structure of the dataset, which includes the column names and the number of observations for each column. It can be especially useful when exploring variable types, such as character or date variables.
- `class()` examining the dataset class. In this project, the data is represented `character`, `sf` and `data frame` formats. `sf` - simple features - is a specific format used for spatial data. The function is useful in mapmaking process, when one need to make sure dataset is in proper class.
- `colnames()` viewing the names of the variables or columns in a dataset. This is particularly helpful when dealing with large datasets and it is needed to quickly check if all the necessary variables are present, or if there are any variables that need to be renamed for clarity.
- `unique()` viewing the unique values present within a specific variable. This is particularly useful when examining the number of different categories or levels in a variable.
- `table()` counting the number of occurrences of different values within a variable. This can be particularly helpful when examining the frequency of specific values or comparing the frequencies of different values in a dataset.

```r
str(AADA) # structure of the dataset, e.g. character = chr, integer = int
```

```
## sf [47,935 x 81] (S3: sf/tbl_df/tbl/data.frame)
##  $ Phase                          : chr [1:47935] "SA" "SA" "SA" "SA" ...
##  $ Site id                        : chr [1:47935] "porvoo 1" "613010040" "613010040" "613010001" ...
##  $ Collection                     : chr [1:47935] "BM" "BM" "BM" "BM" ...
##  $ Main number                    : num [1:47935] 57.3 67.4 67.4 67.4 67.4 ...
##  $ Subnumber                      : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Municipality                   : chr [1:47935] "Porvoo" "Porvoo" "Porvoo" "Porvoo" ...
##  $ Site name                      : chr [1:47935] "Ilola" "Böle" "Böle" "Bastäng" ...
##  $ p                              : num [1:47935] 6705663 6702770 6702770 6700317 6700772 ...
##  $ i                              : num [1:47935] 3431506 3433160 3433160 3411819 3411950 ...
##  $ z                              : num [1:47935] NA 10 10 20 17 20 20 10 10 NA ...
##  $ m1                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ m2                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ m3                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ n1                             : logi [1:47935] FALSE TRUE FALSE FALSE FALSE TRUE ...
##  $ n2                             : logi [1:47935] TRUE FALSE TRUE FALSE FALSE FALSE ...
##  $ n3                             : logi [1:47935] FALSE FALSE FALSE TRUE TRUE FALSE ...
##  $ n4                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ p1                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ p2                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ r1                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ r2                             : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Category                       : chr [1:47935] "Pottery" "Pottery" "Pottery" "Pottery" ...
##  $ Type                           : chr [1:47935] "Late Comb Ware" "Early Comb Ware" "Typical Comb Ware
##  $ Subtype                        : chr [1:47935] "Late Comb Ware" "Sperrings 1 Ware" "Typical Comb Wa
##  $ Subtype (Finnish)              : chr [1:47935] "Ka 3" "Ka 1:1" "Ka 2" "nuora" ...
##  $ Certainty                      : num [1:47935] 1 1 1 1 2 1 1 1 1 NA ...
##  $ Main temper (Finnish)          : chr [1:47935] "orgaaninen" "kivimurska" "orgaaninen" "shamotti" ..
##  $ Other tempers (Finnish)        : chr [1:47935] NA NA NA NA ...
##  $ Count (1-10)                   : logi [1:47935] TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ Count (10-100)                 : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Count (>100)                   : logi [1:47935] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Other notes (Finnish)          : chr [1:47935] NA NA NA NA ...
##  $ geometry                       :sfc_POINT of length 47935; first list element:  'XY' num [1:2] 25.8
##  $ Subtype 2                      : chr [1:47935] NA NA NA NA ...
##  $ Subtype 2 (Finnish)            : chr [1:47935] NA NA NA NA ...
##  $ Integrity                      : logi [1:47935] NA NA NA NA NA NA ...
##  $ Rock type (Finnish)            : chr [1:47935] NA NA NA NA ...
##  $ Length (mm)                    : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Width (mm)                     : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Thickness (mm)                 : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Fragment length (mm)           : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Fragment width (mm)            : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Fragment thickness (mm)        : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Photo ID                       : chr [1:47935] NA NA NA NA ...
##  $ PhotoID                        : chr [1:47935] NA NA NA NA ...
##  $ Ajoitusperuste                 : chr [1:47935] NA NA NA NA ...
##  $ Mon1                           : logi [1:47935] NA NA NA NA NA NA ...
##  $ Mon2                           : logi [1:47935] NA NA NA NA NA NA ...
##  $ Mon3                           : logi [1:47935] NA NA NA NA NA NA ...
##  $ Mon4                           : logi [1:47935] NA NA NA NA NA NA ...
```

```
##  $ Mon5                          : logi [1:47935] NA NA NA NA NA NA ...
##  $ Mon6                          : logi [1:47935] NA NA NA NA NA NA ...
##  $ Length                        : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Preroman IA                   : logi [1:47935] NA NA NA NA NA NA ...
##  $ Early Roman IA                : logi [1:47935] NA NA NA NA NA NA ...
##  $ Late Roman IA                 : logi [1:47935] NA NA NA NA NA NA ...
##  $ Migration period              : logi [1:47935] NA NA NA NA NA NA ...
##  $ Merovingian period            : logi [1:47935] NA NA NA NA NA NA ...
##  $ Viking period                 : logi [1:47935] NA NA NA NA NA NA ...
##  $ Crusade period                : logi [1:47935] NA NA NA NA NA NA ...
##  $ 1100-1300                     : logi [1:47935] NA NA NA NA NA NA ...
##  $ 1400-1500                     : logi [1:47935] NA NA NA NA NA NA ...
##  $ Settlement site               : logi [1:47935] NA NA NA NA NA NA ...
##  $ Cremation cemetery            : logi [1:47935] NA NA NA NA NA NA ...
##  $ Inhumation cemetery           : logi [1:47935] NA NA NA NA NA NA ...
##  $ Cairn                         : logi [1:47935] NA NA NA NA NA NA ...
##  $ Stray find                    : logi [1:47935] NA NA NA NA NA NA ...
##  $ Hoard                         : logi [1:47935] NA NA NA NA NA NA ...
##  $ Other context                 : logi [1:47935] NA NA NA NA NA NA ...
##  $ Decoration                    : logi [1:47935] NA NA NA NA NA NA ...
##  $ Wave motif                    : logi [1:47935] NA NA NA NA NA NA ...
##  $ Grid motif                    : logi [1:47935] NA NA NA NA NA NA ...
##  $ Cord impression               : logi [1:47935] NA NA NA NA NA NA ...
##  $ Decoration elements (Finnish): chr [1:47935] NA NA NA NA ...
##  $ Wall thickness 1 (mm)         : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Wall thickness 2 (mm)         : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Rim thickness 1 (mm)          : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Rim thickness 2 (mm)          : num [1:47935] NA NA NA NA NA NA NA NA NA NA ...
##  $ Crust                         : logi [1:47935] NA NA NA NA NA NA ...
##  $ Stone art (Finnish)           : chr [1:47935] NA NA NA NA ...
##  $ Type (Finnish)                : chr [1:47935] NA NA NA NA ...
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "names")= chr [1:80] "Phase" "Site id" "Collection" "Main number" ...
```

```
class(AADA) # "sf" format indicating dataset has spatial format
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
colnames(AADA) # column names
```

```
##  [1] "Phase"                         "Site id"
##  [3] "Collection"                    "Main number"
##  [5] "Subnumber"                     "Municipality"
##  [7] "Site name"                     "p"
##  [9] "i"                             "z"
## [11] "m1"                            "m2"
## [13] "m3"                            "n1"
## [15] "n2"                            "n3"
## [17] "n4"                            "p1"
## [19] "p2"                            "r1"
## [21] "r2"                            "Category"
## [23] "Type"                          "Subtype"
```

```
## [25] "Subtype (Finnish)"            "Certainty"
## [27] "Main temper (Finnish)"        "Other tempers (Finnish)"
## [29] "Count (1-10)"                 "Count (10-100)"
## [31] "Count (>100)"                 "Other notes (Finnish)"
## [33] "geometry"                     "Subtype 2"
## [35] "Subtype 2 (Finnish)"          "Integrity"
## [37] "Rock type (Finnish)"          "Length (mm)"
## [39] "Width (mm)"                   "Thickness (mm)"
## [41] "Fragment length (mm)"         "Fragment width (mm)"
## [43] "Fragment thickness (mm)"      "Photo ID"
## [45] "PhotoID"                      "Ajoitusperuste"
## [47] "Mon1"                         "Mon2"
## [49] "Mon3"                         "Mon4"
## [51] "Mon5"                         "Mon6"
## [53] "Length"                       "Preroman IA"
## [55] "Early Roman IA"               "Late Roman IA"
## [57] "Migration period"             "Merovingian period"
## [59] "Viking period"                "Crusade period"
## [61] "1100-1300"                    "1400-1500"
## [63] "Settlement site"              "Cremation cemetery"
## [65] "Inhumation cemetery"          "Cairn"
## [67] "Stray find"                   "Hoard"
## [69] "Other context"                "Decoration"
## [71] "Wave motif"                   "Grid motif"
## [73] "Cord impression"              "Decoration elements (Finnish)"
## [75] "Wall thickness 1 (mm)"        "Wall thickness 2 (mm)"
## [77] "Rim thickness 1 (mm)"         "Rim thickness 2 (mm)"
## [79] "Crust"                        "Stone art (Finnish)"
## [81] "Type (Finnish)"
```

```
colnames(BA_pottery)
```

```
##  [1] "Phase"                  "Site id"
##  [3] "Municipality"           "Site name"
##  [5] "Collection"             "Main number"
##  [7] "Subnumber"              "p"
##  [9] "i"                      "z"
## [11] "m1"                     "m2"
## [13] "m3"                     "n1"
## [15] "n2"                     "n3"
## [17] "n4"                     "p1"
## [19] "p2"                     "r1"
## [21] "r2"                     "Category"
## [23] "Type"                   "Subtype"
## [25] "Subtype (Finnish)"      "Certainty"
## [27] "Main temper (Finnish)"  "Other tempers (Finnish)"
## [29] "Count (1-10)"           "Count (10-100)"
## [31] "Count (>100)"           "Other notes (Finnish)"
## [33] "PhotoID"                "geometry"
```

```
unique(AADA$Category) # unique entries in Categories column
```

```
##  [1] "Pottery"          "Stone artefacts"  "Clay artefacts"   "Bone artefacts"
```

```
##  [5] "Wooden artefacts" "Amber artefacts"  "Birch bark tar"   "Bronze artefacts"
##  [9] "Bone artefact"    "Beads"            "Silver artefact"  "Iron artefact"
```

```
unique(BA_pottery$Type)
```

```
## [1] "Coastal Bronze Age pottery" "Säräisniemi 2 Ware"
## [3] "Sarsa-Tomitsa Ware"         "Lovozero Ware"
## [5] "Vardöy (IT) Ware"
```

```
unique(BA_pottery$Subtype)
```

```
##  [1] "Coastal Bronze Age pottery" "Luukonsaari Ware"
##  [3] "Sarsa-Tomitsa Ware"         "Sirnihta Ware"
##  [5] "Paimio Ware"                "Anttila Ware"
##  [7] "Kjelmöy Ware"               "Lovozero Ware"
##  [9] "Lausitz Ware"               "Vardöy (IT) Ware"
## [11] "Säräisniemi 2 Ware"         "Otterböte Ware"
```

```
table(IA_bone$Type)# counting of different types of pottery
```

```
##
##                Bone    Bone arrow point            Bone awl           Bone comb
##                   1                   1                   1                  41
##        Bone fitting        Bone harpoon         Bone needle        Bone pendant
##                   1                   1                   1                   1
##  Bone spindle whorl          Bone spoon Other bone artefact  Rivet setting tool
##                  13                   3                  33                   1
```

```
table(BA_pottery$Type)
```

```
##
## Coastal Bronze Age pottery             Lovozero Ware
##                        142                        77
##         Sarsa-Tomitsa Ware         Säräisniemi 2 Ware
##                        292                       337
##           Vardöy (IT) Ware
##                         13
```

```
unique(AADA$Phase) #unique periods
```

```
## [1] "SA" "BA" "IA"
```

```
table(AADA$Phase) #counting periods
```

```
##
##    BA    IA    SA
##  1474  6470 39991
```

3.2.2 Museum collections

This section is about exploring AADA museum collections information.To explore the source information of specific museum **Collection** column is explored.

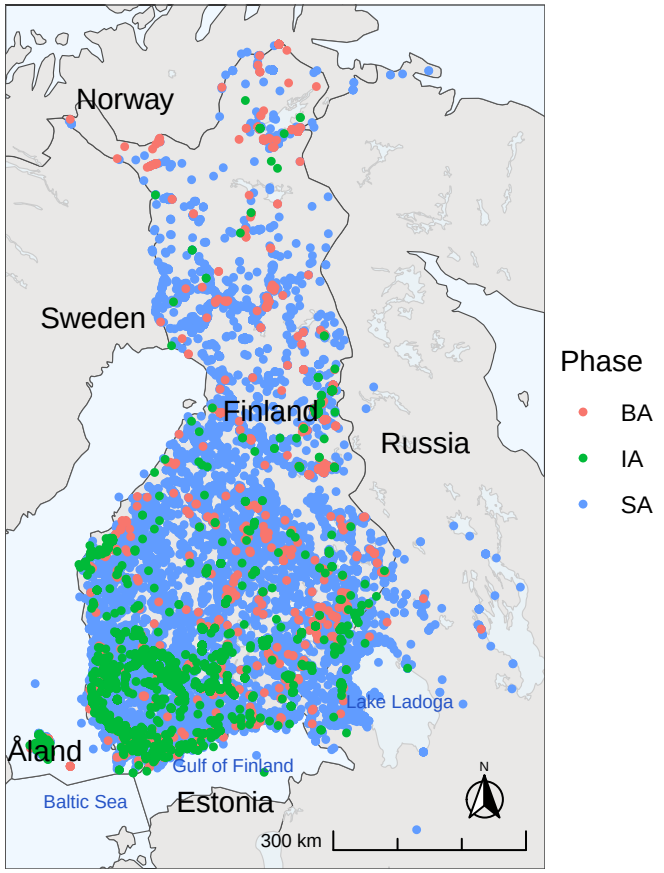
```
unique(AADA$Collection) # 32 unique entries in Collection column
```

```
##  [1] "BM"                          "EKM"
##  [3] "KHMESIE"                     "KM"
##  [5] "Nyberg"                      "SatM"
##  [7] "TMM"                         "TYA"
##  [9] "ÅM"                          "Linder"
## [11] "Per"                         "PerL"
## [13] "Hal"                         "Lauri Nautelan kok"
## [15] "Punk"                        "Lap"
## [17] "Koke"                        "Köy"
## [19] "Keik"                        "Kiik"
## [21] "Kul"                         "Suod"
## [23] "Noor"                        "Lavi"
## [25] "Noormarkun kotiseututalo" "Kank"
## [27] "Karv"                        "KARTT/VI"
## [29] "SII"                         "KIUR"
## [31] "Linder/TMM"                  "SHH"
## [33] "HM"                          NA
```

```
table(AADA$Collection) # counting of different types of museum collections,
```

```
##
##                      ÅM                      BM                     EKM
##                     622                     235                      52
##                     Hal                      HM                    Kank
##                      99                      70                       1
##                KARTT/VI                   Karv                    Keik
##                       3                       5                       5
##                 KHMESIE                    Kiik                    KIUR
##                     842                      12                      96
##                      KM                   Koke                     Kul
##                   39991                      31                       1
##                     Köy                     Lap      Lauri Nautelan kok
##                       2                       8                       2
##                    Lavi                  Linder             Linder/TMM
##                       4                     412                       4
##                    Noor Noormarkun kotiseututalo                  Nyberg
##                       7                       1                      14
##                     Per                    PerL                    Punk
##                      97                       8                       7
##                    SatM                     SHH                     SII
##                    3634                       1                      14
##                    Suod                     TMM                     TYA
##                       3                     605                    1027
```

```r
# e.g. most representative is KM - Finnish National Museum (Kansallismuseo) with 36790 artefacts.

AADA_map <- ggplot() +
  geom_sf(data = region, fill="#E9E8E7", color="grey30") +
  geom_sf(data = lakes, fill="#eaf2f6", color="grey80") +
  geom_sf(data=ocean,fill="#F0F8FF") +
  geom_sf(data = AADA, aes(color = Phase), size = 1.5, shape =20) +
  geom_sf_text(data = st_crop(ocean, st_as_sfc(st_bbox(limits_aada))), aes(label = name),
               color="#2554C7", size = 2.5, color = "black", nudge_y = ifelse(ocean$name ==
                                                                                "Barents Sea" | ocean$na
  geom_sf_text(data=st_crop(region, st_as_sfc(st_bbox(limits_aada))), aes(label=admin),
               size=4,color = "black", nudge_y = ifelse(region$admin == "Åland", -0.1,0))+
  # adjust Aland position to avoid overlap
  geom_sf_text(data = lakes, aes(label = ifelse(name == "Lake Ladoga", "Lake Ladoga", "")),
               color="#2554C7", size = 2.5, color = "black") +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4])) +
  guides(fill = "none") +
  theme_void() +
  theme(panel.background = element_rect(fill = "#F0F8FF"),
        panel.border = element_rect(color = "grey30", fill = NA, linewidth = 0.25),
        panel.grid.major = element_blank(),
        legend.background = element_blank(),  # Remove legend background
        legend.key = element_blank(),          # Remove legend key background
        legend.title = element_text(color = "black"),  # Adjusting legend title color
        legend.text = element_text(color = "black")) +  # Adjusting legend text color
  annotation_scale(
    location = "br",  # bottom left
    width_hint = 0.4,  # Adjusting the width_hint to make the scale bar smaller
    text_cex = 0.6,
    distance = 2,
    style = "ticks") +  # "bar" or "ticks" based on your preference
  annotation_north_arrow(
    location = "br",
    which_north = "grid",
    pad_x = unit(0.4, "cm"),
    pad_y = unit(0.6, "cm"),
    height = unit(0.8, "cm"),  # Adjust the height to make it smaller
    width = unit(0.8, "cm"),   # Adjust the width to make it smaller
    style = north_arrow_fancy_orienteering(
      line_width = 1,
      line_col = "black",
      fill = c("white", "black"),
      text_col = "black",
      text_size = 5,
      text_angle = 0))

# Display the plot
AADA_map
```



3.3 Stone Age

3.3.1 Mesolithic leaf-shaped slate spearheads, Figure 4a

To create a artefact map, it is needed first to subset the relevant data by determining which types of spearhead tools are available from the **Subtype** column. When working with long lists (e.g. 115 different artefact subtypes), filtering information with keywords to find patterns can be useful. For example, to find all subtypes that include the word **"spearhead"**, one can create a vector called `spearhead_subtypes` using the following code and `grep()` function.

```
#find all subtypes from Stone tools data sheet that include the word **"spearhead"**
spearhead_subtypes <- SA_stone$Subtype[grep("spearhead", SA_stone$Subtype)] # use "grepl" function inst
unique(spearhead_subtypes) #explore the unique spearhead subtypes
```

```
## [1] "Tapering oval spearhead"      "Leaf-shaped slate spearhead"
## [3] "Scandinavian slate spearhead"
```

Subsetting is perfomed using `filter()` function from `dplyr` to select only the rows where Subtype is **"Leaf-shaped slate spearhead"**. The resulting layer is stored in a new variable called `SA_spearhead`.

```
SA_spearhead <- SA_stone %>%
  filter(Subtype == "Leaf-shaped slate spearhead")
# 499 entries (observations)
```

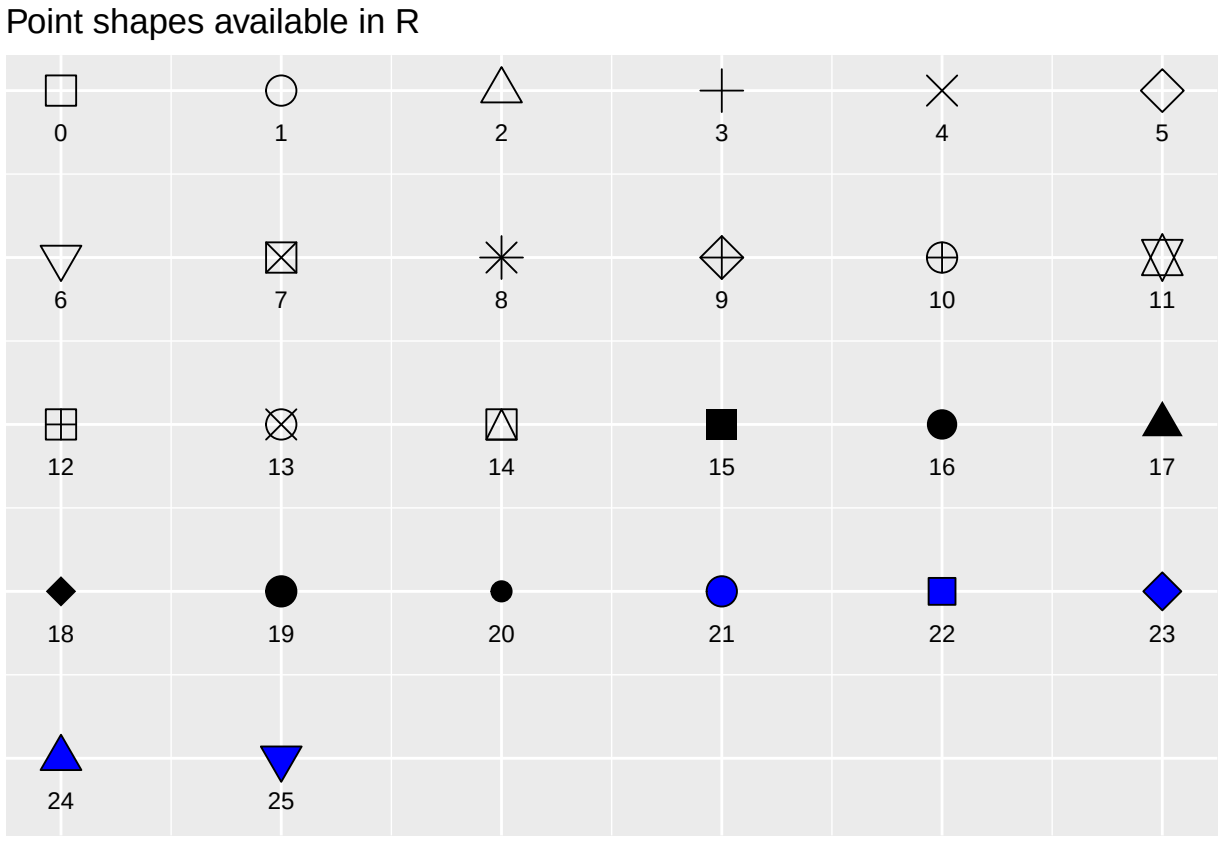
Plotting the subset to the map. This code is using the `ggplot2` package to create a plot that displays Mesolithic spearheads in the region of the Baltic Sea. The plot includes multiple layers of geographic data and annotations. By assigning the name for the resulting layer, you create "gg"-format, which is later used in exporting the figure as ***.jpg***. This section applies point shape visualisation method, which adds clarity and readability to the point data. The different symbol shapes are designed to be visually distinguishable, making it easier to interpret and compare data points in a plot. `ggplot2` package contains 25 built-in shapes.

Other different characters symbols can be used to specify the shape argument, including "+", "*",-",:","#, "%", "o".

```
install.packages("ggpubr")
```

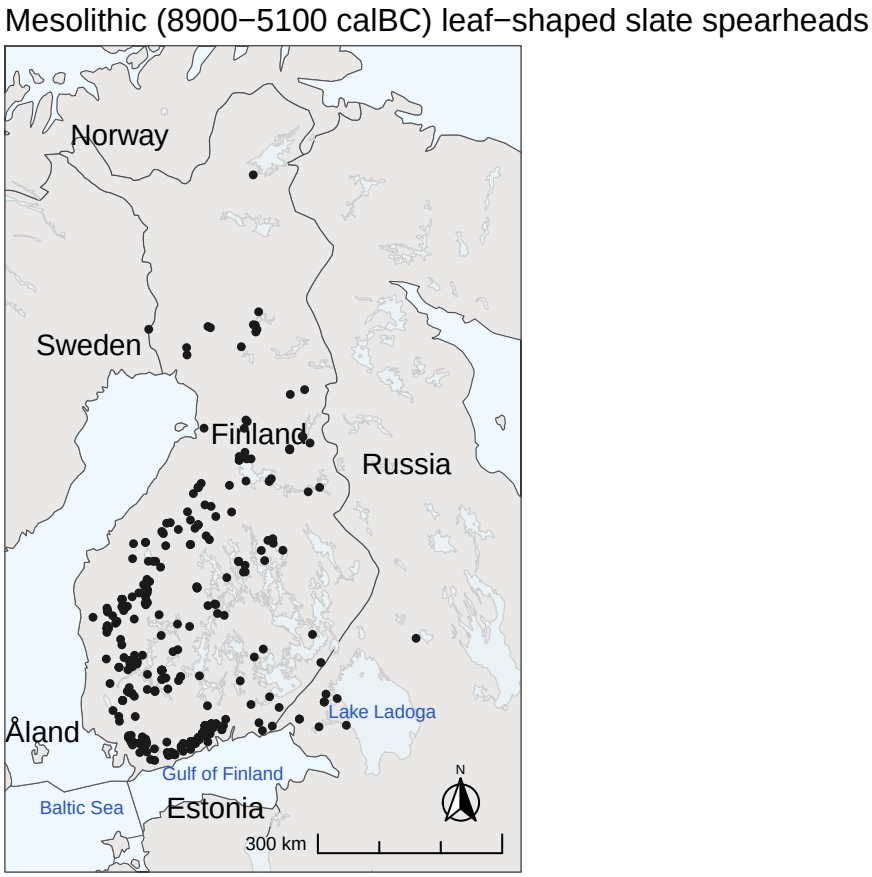
```
## package 'ggpubr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\meroos\AppData\Local\Temp\RtmpOwJgsc\downloaded_packages
```

```
library(ggpubr)
# The function below illustrates the different point shape values
ggpubr::show_point_shapes()
```




```
SA_spearheads_fig4a <- core_map +
  geom_sf(data = SA_spearhead, aes(color = Type), color="grey10", size = 1.5, shape =20) +
  ggtitle(str_wrap("Mesolithic (8900-5100 calBC) leaf-shaped slate spearheads", width = 81))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size=12))+
```

```
  # the title is set 1 mm away from the top of the plot
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))

# Display the plot
SA_spearheads_fig4a
```



3.3.2 Neolithic East Carelian adzes and chisels, Figure 4b

Specific colors are assigned by command `scale_color_manual()` to each type.

```
unique(SA_all$Subtype)
# 224 unique subtype entires
```

```
# Print shortened list, first 15 unique entries as an example
unique_values <- unique(SA_all$Subtype)[1:15]
print(unique_values)
```

```
##  [1] "Late Comb Ware"                "Sperrings 1 Ware"
##  [3] "Typical Comb Ware"             "Corded Ware"
##  [5] "Other pottery"                 "Early Asbestos Ware"
##  [7] "Pöljä Ware"                    "Kierikki Ware"
##  [9] "Säräisniemi 2 Ware"            "Luukonsaari Ware"
## [11] "Sperrings 2 Ware"              "Kierikki WareEarly Asbestos Ware"
```

```
## [13] "Jysmä Ware"                          "Pit-Comb Ware"
## [15] "Pitted Ware"
```

It is difficult to read such a long list of entries, thus it is advisable to use `grep`function. In following examples all entries containing ***adze*** and ***chisel*** are selected and shorter list of unique values observed. After shortened search, one can determine the adze and chisel types needed.

```
adze_subtypes <- SA_stone$Subtype[grep("adze", SA_stone$Subtype)]
unique(adze_subtypes)
```

```
## [1] "Stone adze"          "East Carelian adze" "Kiukainen adze"
## [4] "Jäkärlä adze"        "Miniature adze"     "Northbothnian adze"
## [7] "Double adze"         "Flint adze"
```

```
chisel_subtype <- SA_stone$Subtype[grep("chisel", SA_stone$Subtype)]
unique(chisel_subtype)
```

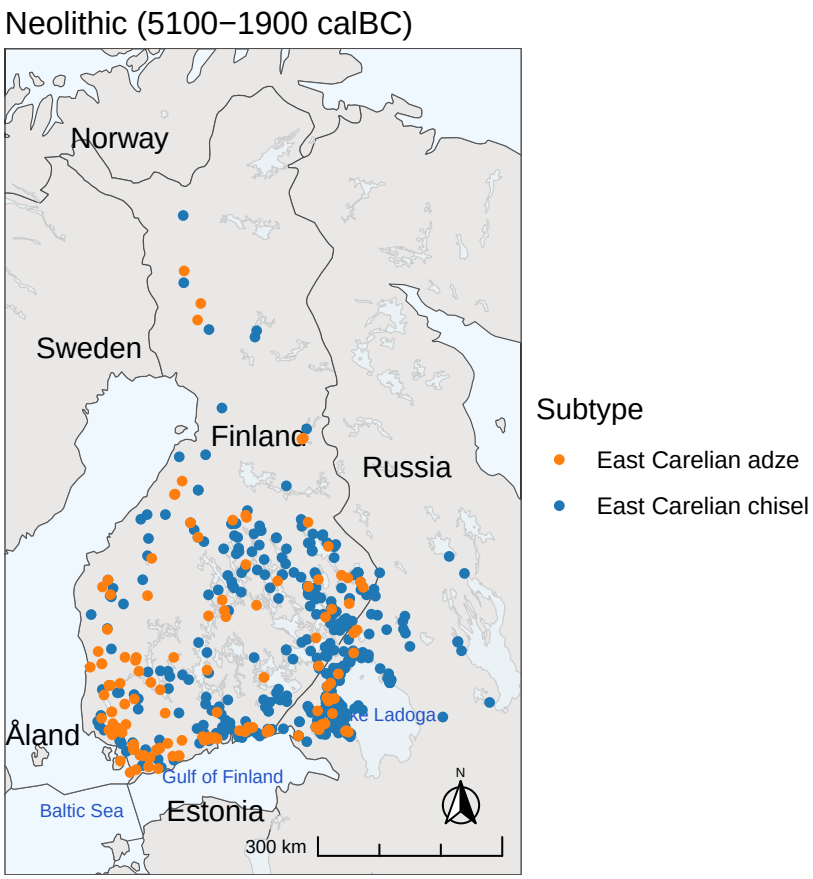
```
##  [1] "Stone chisel"          "Northbothnian chisel"  "South Finnish chisel"
##  [4] "Small chisel"          "Narrow chisel"         "East Carelian chisel"
##  [7] "Kiukainen chisel"      "Adze-chisel"           "Double chisel"
## [10] "South Carelian chisel" "Primitive chisel"      "Clawed chisel"
```

```
SA_adze <- SA_stone %>%
  filter(`Subtype` == "East Carelian adze")
# 140 obs
SA_chisel <- SA_stone %>%
  filter(`Subtype` == "East Carelian chisel")
# 493 obs

# subsetting adze and chisel to the same layer, optional
# SA_adze_chisel <- SA_stone %>%
#  filter(`Subtype` == "East Carelian adze" | `Subtype` == "East Carelian chisel")
# 633 obs
```

Utilize `aes(color = ...)` within `geom_sf` for automatic color mapping based on a variable in your dataset. Employ `scale_color_manual` when seeking manual control over color assignment, particularly for specifying specific colors for each level of a categorical variable. Consider your specific visualisation goals and tailor your approach accordingly.

```
# plot the subset
SA_adze_chisel_fig4b <- core_map +
  geom_sf(data = SA_chisel, aes(color = Subtype), size = 2, shape =20) +
  geom_sf(data = SA_adze, aes(color = Subtype), size = 2, shape =20) +
  scale_color_manual(values = c("East Carelian adze" = "#ff7f0e",
                                "East Carelian chisel" = "#1f77b4")) +
  theme(panel.background = element_rect(fill = "#F0F8FF"),
        panel.border = element_rect(color = "grey30", fill = NA, linewidth = 0.25),
        panel.grid.major = element_blank(),
        legend.background = element_blank(),  # Remove legend background
        legend.key = element_blank(),          # Remove legend key background
        legend.title = element_text(color = "black"),  # Adjusting legend title color
        legend.text = element_text(color = "black")) +  # Adjusting legend text color
```

```
  ggtitle(str_wrap("Neolithic (5100-1900 calBC)", width = 48))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size=12))+
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))

# Display the plot
SA_adze_chisel_fig4b
```



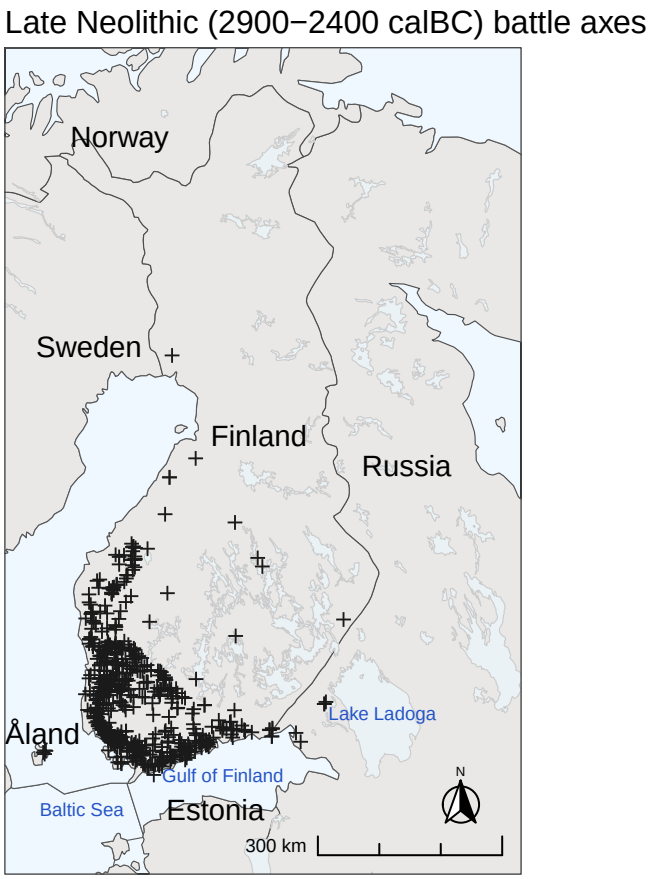
3.3.3 Battle axes of the Corded Ware culture, Figure 4c

```
# subsetting the SA_stone tools based on Type = "Battle Axe"
axe_subtypes <- SA_stone$Subtype[grep("axe", SA_stone$Subtype)]
unique(axe_subtypes)
```

```
##  [1] "Battle axe"                        "Stone axe"
##  [3] "Simple shaft-hole axe"             "Kiukainen axe"
##  [5] "Shouldered axe"                    "Northbothnian axe"
##  [7] "Sperrings axe"                     "Four-sided axe"
##  [9] "Jäkärlä axe"                       "Narrow-edged axe"
## [11] "Rocker-shaped axe"                 "Shaft-hole axe with tapering butt"
## [13] "Primitive axe"                     "Round-edged axe"
## [15] "Double axe"                        "Flint axe"
## [17] "Crooked pickaxe"                   "Knob battle axe"
```

```
## [19] "Shaft-hole axe"                    "Straight-backed shaft-hole axe"
## [21] "Ice pick -like axe"                "East Carelian axe"
## [23] "Massive axe"                       "Ilomantsi axe"
## [25] "Scandinavian shaft-hole axe"
```

```
SA_battle_axe <- SA_stone %>%
  filter(`Subtype` == "Battle axe")
# 713 obs

# plot the subset, shape value = 1, to visualize the point overlaps
SA_battle_axe_fig4c <- core_map +
  geom_sf(data = SA_battle_axe, aes(color = Subtype), color="grey10", size = 1.5, shape =3) +
  ggtitle("Late Neolithic (2900-2400 calBC) battle axes")+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size=12))+
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
# Display the plot
SA_battle_axe_fig4c
```



3.4 Bronze Age

3.4.1 Bronze Age asbestos-tempered ceramics, Figure 5a

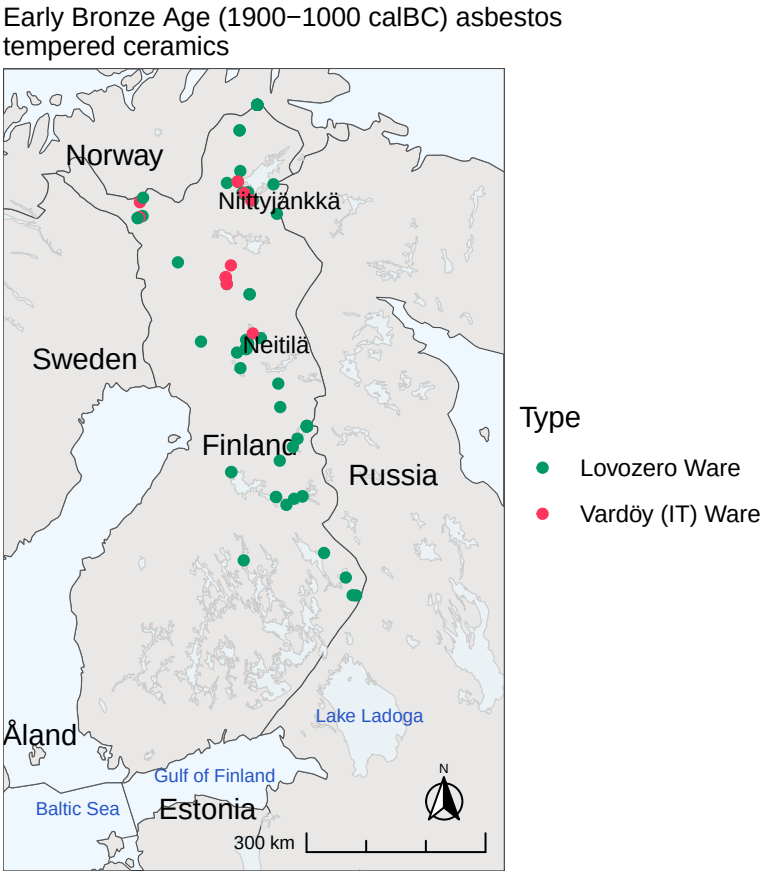
Examples of Bronze Age artefacts (c. 1900-500 calBC) plotted on the map of Finland: Early Bronze Age asbestos-tempered ceramics - Lovozero Ware and Vardöy Ware. The two classes are subsetted to the same

layer.

```
unique(BA_pottery$Type)
```

```
## [1] "Coastal Bronze Age pottery" "Säräisniemi 2 Ware"
## [3] "Sarsa-Tomitsa Ware"         "Lovozero Ware"
## [5] "Vardöy (IT) Ware"
```

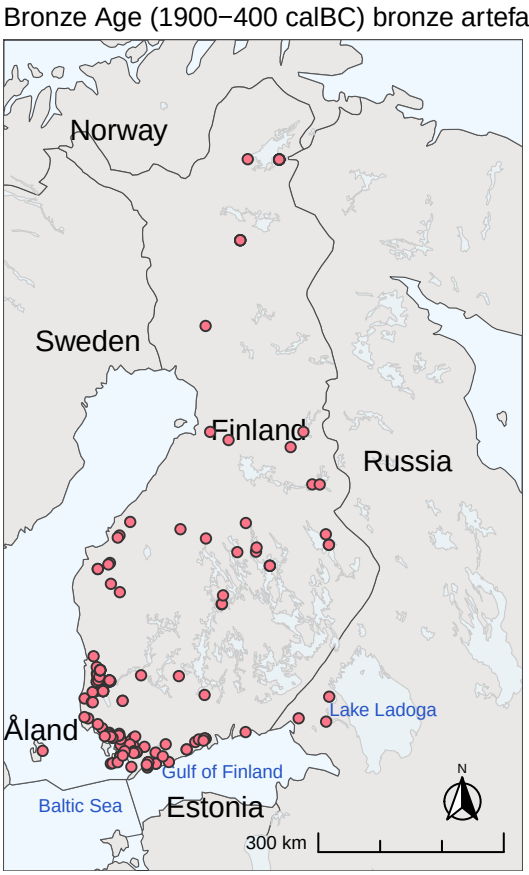
```
BA_lovozero_vardoy <- BA_pottery %>%
  filter(Type == "Lovozero Ware" | Type == "Vardöy (IT) Ware" & (p1))
# 90 obs

BA_lovozero_vardoy_fig5a <- core_map +
  geom_sf(data = BA_lovozero_vardoy, aes(fill=Type, color=Type),size = 1.5, shape = 19) +
    scale_fill_manual(values = c("Lovozero Ware" = "#009966", "Vardöy (IT) Ware" = "#ff355e")) +
  scale_color_manual(values = c("Lovozero Ware" = "#009966", "Vardöy (IT) Ware" = "#ff355e")) +
  geom_sf_text(data = BA_lovozero_vardoy %>% filter(`Main number` %in% c(16145, 26240)),
               aes(label = ifelse(`Main number` == 16145, "Neitilä",
                                  ifelse(`Main number` == 26240, "Niittyjänkkä", ""))),
               size = 3, color = "black", nudge_x = 1) +
  guides(shape = guide_legend(title = "Decoration Types and Counts"))+
  ggtitle(str_wrap("Early Bronze Age (1900-1000 calBC) asbestos tempered ceramics", width = 48))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size=10))+
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
# Display the plot
BA_lovozero_vardoy_fig5a
```



3.4.2 Bronze Age bronze artefacts, Figure 5b

The code is using the 'BA_bronze' data frame to plot Bronze Age artefacts on a previously created core map. No need for subsetting, as the whole collection is presented.

```
BA_bronze_fig5b <- core_map +
  geom_sf(data = BA_bronze, color="grey20",fill="#fe7489", size = 1.5, shape =21) +
  ggtitle(str_wrap("Bronze Age (1900-400 calBC) bronze artefacts", width = 48))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size=10))+
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))

# Display the plot
BA_bronze_fig5b
```



3.4.3 Sarsa-Tomitsa textile pottery, Figure 5c

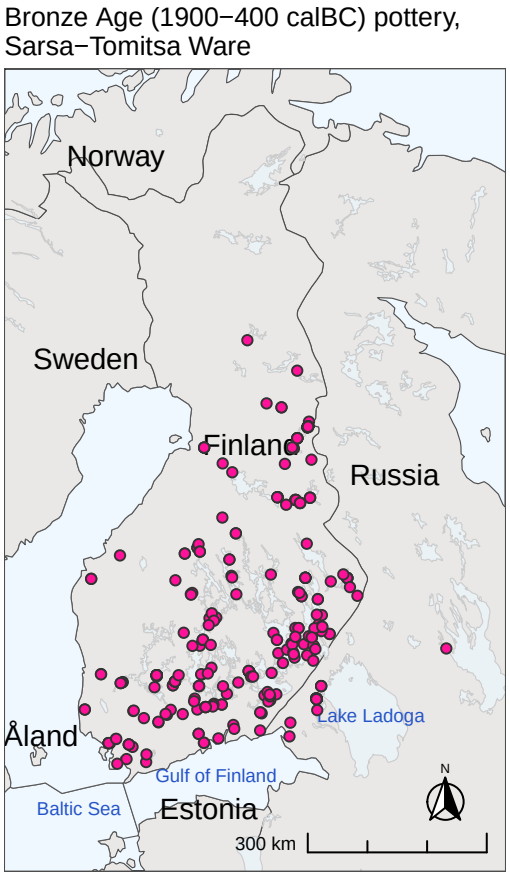
This code is determining the types of pottery present in the 'BA_pottery' data frame. Specifically, it is used to find the types of pottery.

```
# there is 5 different types of pottery
table(BA_pottery$Type)
```

```
##
## Coastal Bronze Age pottery              Lovozero Ware
##                        142                         77
##         Sarsa-Tomitsa Ware         Säräisniemi 2 Ware
##                        292                        337
##           Vardöy (IT) Ware
##                         13
```

```
# subsetting the "Sarsa-Tomitsa Ware" pottery
BA_sarsa_tomitsa <- BA_pottery %>%
  filter(Type == "Sarsa-Tomitsa Ware")
# 292 obs

BA_sarsa_tomitsa_fig5c <- core_map +
  geom_sf(data = BA_sarsa_tomitsa,color="grey20",fill="#ff1199", size=1.5, shape=21) +
  ggtitle(str_wrap("Bronze Age (1900-400 calBC) pottery, Sarsa-Tomitsa Ware", width =48))+
```

```
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size=10))+
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
# Display the plot
BA_sarsa_tomitsa_fig5c
```



3.5 Iron Age

3.5.1 Iron Age pottery decorations, Figure 6a

This code generates a map of Iron Age pottery categorized by its decoration motifs, employing specific colors and adjusted transparency. For instance, an alpha value of 80 represents approximately 50% transparency (128 out of 255), allowing for some visibility of overlapping data. The `case_when` function is utilized to establish conditions based on the values in the selected columns. If the "Wave motif" column is ***TRUE***, new "Motifs" column is set to "Wave." A similar procedure is applied for the "Grid" and "Cord" motifs. Moreover, this chapter demonstrates how to create a focused and detailed map by adjusting the visible extent based on a specific zoom level (bounding box).

```
unique(IA_pottery_detailed$`Wave motif`)
```

```
## [1] FALSE  TRUE
```

```r
unique(IA_pottery_detailed$`Grid motif`)
```

```
## [1] FALSE  TRUE
```

```r
unique(IA_pottery_detailed$`Cord impression`)
```

```
## [1] FALSE  TRUE
```

```r
table(IA_pottery_detailed$`Wave motif`)
```

```
##
## FALSE  TRUE
##   784    38
```

```r
table(IA_pottery_detailed$`Grid motif`)
```

```
##
## FALSE  TRUE
##   816     6
```

```r
table(IA_pottery_detailed$`Cord impression`)
```

```
##
## FALSE  TRUE
##   746    76
```

```r
# Create new column containing concatenated value "TRUE"
# of "Wave motif", "Grid motif", "Cord impression"

library(tidyr)
# Creating a new column "Motifs" based on conditions
IA_decorations <- IA_pottery_detailed %>%
  mutate(
    Motifs = case_when(
      (`Wave motif` & `Grid motif`) | (`Wave motif` & `Cord impression`) |
        (`Grid motif` & `Cord impression`) ~ "Multiple",
      `Wave motif` ~ "Wave",
      `Grid motif` ~ "Grid",
      `Cord impression` ~ "Cord",
      TRUE ~ "NA/None"
    ))

# result is same data frame with new column "Motifs", 47 -> 48 variables

unique(IA_decorations$Motifs) # Displaying values in Decorations column
```

```
## [1] "NA/None"  "Wave"     "Grid"     "Cord"     "Multiple"
```

```r
# "NA/None"  "Wave"     "Grid"     "Cord"     "Multiple"
table(IA_decorations$Motifs) # table of counts for each unique value
```

```
##
##     Cord     Grid Multiple  NA/None     Wave
##       70        5        7      709       31
```

```r
# Creating the count of each class
IA_deco_class_counts <- table(IA_decorations$Motifs)
IA_deco_class_counts
```

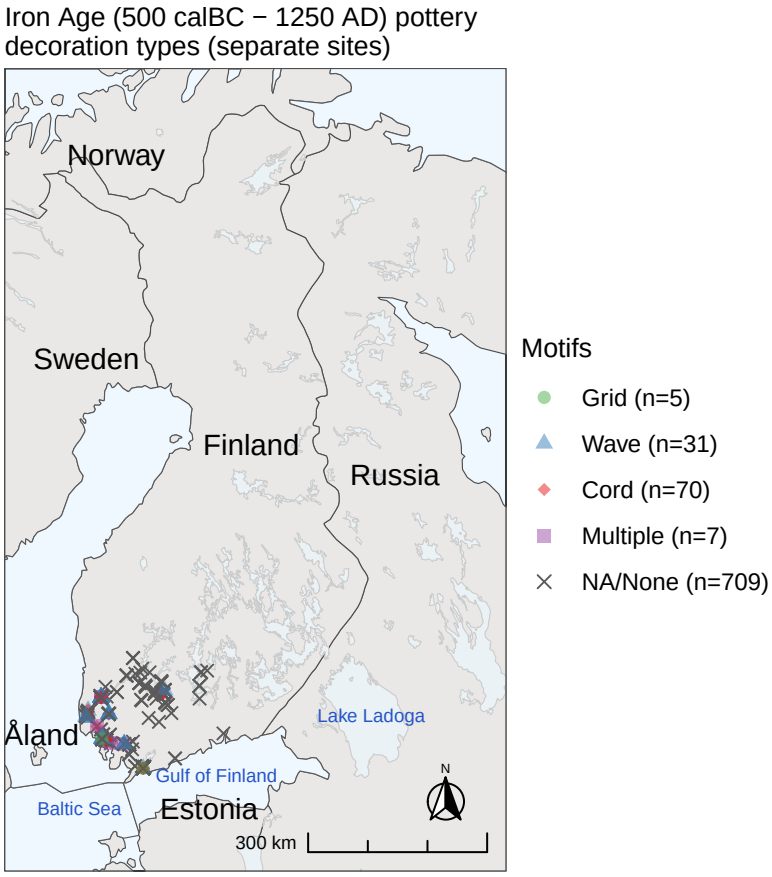
```
##
##     Cord     Grid Multiple  NA/None     Wave
##       70        5        7      709       31
```

```r
# Defining desired order of bead types in the legend
motifs_desired_order <- c("Grid","Wave","Cord","Multiple", "NA/None")

# Defining custom point shapes
motifs_point_shapes <- c(16, 17, 18,15,4)  # More types in chapter 3.3.1
# Combining class names with counts
motifs_legend_labels <- paste(
  motifs_desired_order,
  " (n=",
  as.character(IA_deco_class_counts[motifs_desired_order]),
  ")",
  sep = ""
)

# Creating and modifying the color code. The transparency of overlapping data
# is archieved by appending "80" at the end of color code.
#4DAF4A -> #4DAF4A80 (transparency of approximately 50%)
motifs_colors <- c("Grid" = "#4DAF4A80", "Wave" = "#377EB880","Cord" = "#E41A1C80",
                   "Multiple"="#984EA380",  "NA/None" = "#5a5a5a")

# Reorder the levels of the Type variable in IA_stone dataset
IA_decorations$Motifs <- factor(IA_decorations$Motifs, levels = motifs_desired_order)

IA_detailed_fig6a <- core_map +
  geom_sf(data = IA_decorations, aes(shape=Motifs, color=Motifs), size=2)+
  scale_shape_manual(motifs_desired_order,values = motifs_point_shapes,
                     labels = motifs_legend_labels) +
  scale_color_manual(values = motifs_colors) +
  ggtitle(str_wrap("Iron Age (500 calBC - 1250 AD) pottery
                   decoration types (separate sites)", width = 48)) +
  guides(shape = guide_legend(title = "Motifs",
                              override.aes = list(color = motifs_colors)),
         color = "none") +
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
    legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
# Display the plot
IA_detailed_fig6a
```

```
## Scale on map varies by more than 10%, scale bar may be inaccurate
```



Following map provides a closer view of Iron Age pottery decoration types within the defined subset, allowing for a more detailed observation compared to the entire study area. The `st_bbox` function is used to calculate the bounding box of the IA_pottery_detailed dataset, which represents the extent of the detailed subset of the Iron Age pottery data.

```
limits_IA_pottery_detailed <- st_bbox(IA_pottery_detailed) # same as in chapter 2.5 st_bbox
limits_IA_pottery_detailed # coordinates of the subset
```

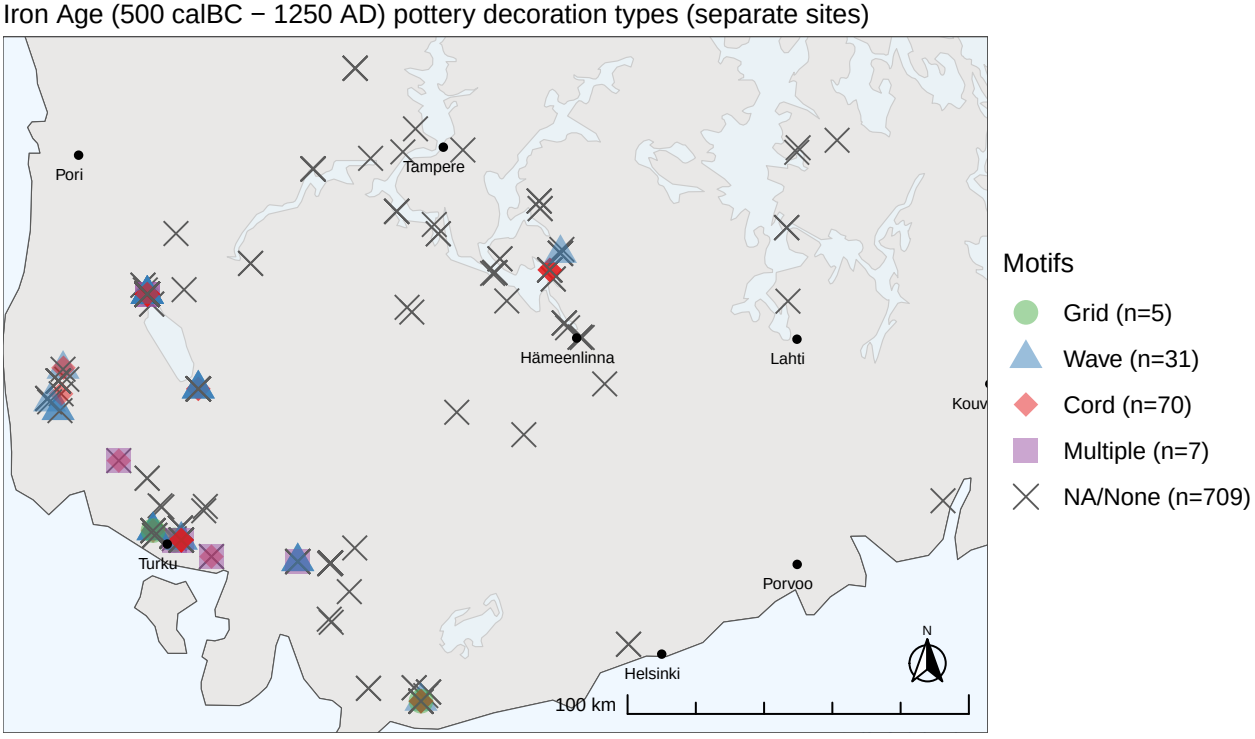
```
##     xmin     ymin     xmax     ymax
## 21.60786 60.03981 26.45570 61.70801
```

```
# Plotting the map within the specified extent
IA_detailed_fig6a_zoomed <- core_map +
  geom_sf(data = IA_decorations, aes(shape=Motifs, color=Motifs), size=4)+
  geom_sf(data = cities, size = 1)+
  geom_sf_text(data = cities, aes(label = NAME),color="black", size = 2,
```

```
              nudge_x = -0.05, nudge_y = -0.05) + # adding cities names data for reference
  scale_shape_manual(motifs_desired_order,values = motifs_point_shapes,
                     labels = motifs_legend_labels) +
  scale_color_manual(values = motifs_colors) +
  ggtitle(str_wrap("Iron Age pottery decoration types", width = 61))+
  guides(shape = guide_legend(title = "Motifs",
                              override.aes = list(color = motifs_colors)),
    color = "none")+
  ggtitle(str_wrap("Iron Age (500 calBC - 1250 AD) pottery decoration types (separate sites)",
                   width = 81))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
    legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_IA_pottery_detailed[1], limits_IA_pottery_detailed[3]), # defining the zoom
           ylim = c(limits_IA_pottery_detailed[2], limits_IA_pottery_detailed[4]))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
# Display the plot
IA_detailed_fig6a_zoomed
```



3.5.2 Iron Age swords, Figure 6b

This code generates a visual representation of Iron Age swords categorized by their types. Similarly to 3.3.1 case, we look for Type column that include the word **"sword"**, one can create a vector called `IA_swords`

using the following code and `grep()` function. In this example you are going to make keyword search and subsetting in one command.

```
IA_swords <- IA_all %>%
filter(grepl("sword", Subtype, ignore.case = TRUE))
# 124 entries (observations)

unique(IA_swords$Subtype) # explore the dataset
```

```
## [1] "Iron sword"                          "Petersen B type iron sword"
## [3] "Iron mount of a sword shaft"         "Kirpichnikov A type iron sword"
## [5] "Petersen Z type iron sword"          "Petersen V type iron sword"
## [7] "Petersen H type iron sword"          "Iron sword with Brazil nut pommel"
## [9] "Petersen C type iron sword"
```

```
table(IA_swords$Subtype)
```

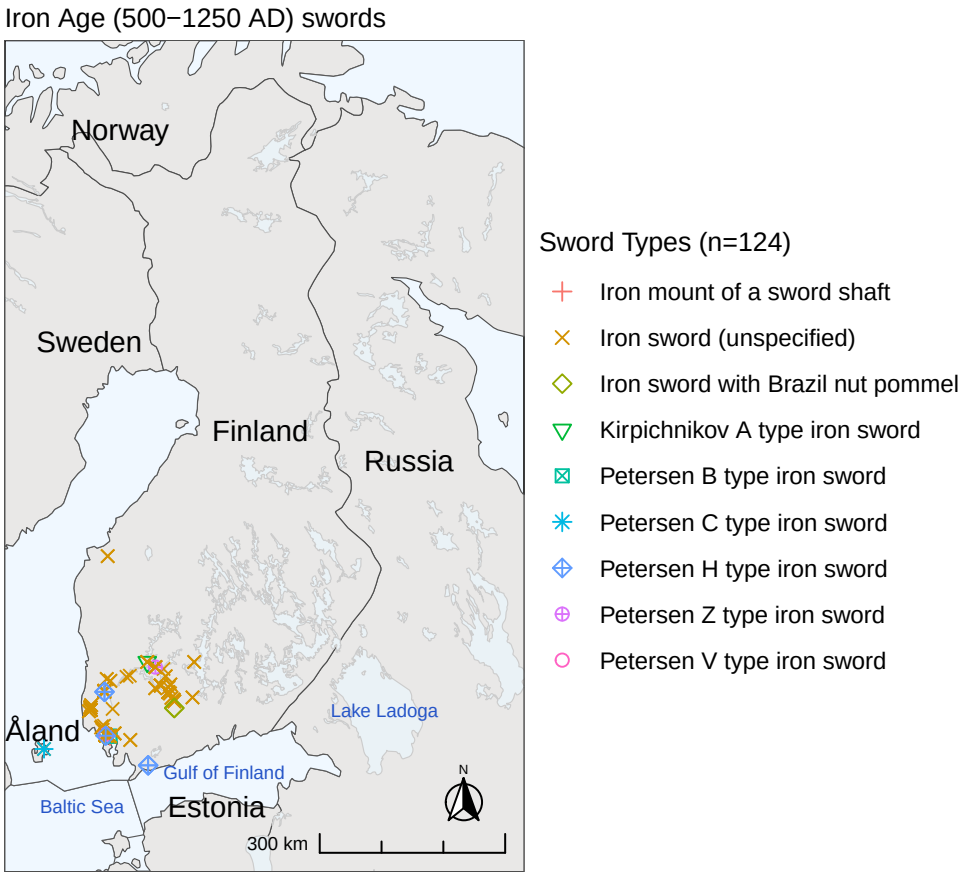
```
##
##       Iron mount of a sword shaft                         Iron sword
##                                 1                                110
## Iron sword with Brazil nut pommel    Kirpichnikov A type iron sword
##                                 2                                  1
##        Petersen B type iron sword        Petersen C type iron sword
##                                 2                                  1
##        Petersen H type iron sword        Petersen Z type iron sword
##                                 5                                  1
##        Petersen V type iron sword
##                                 1
```

```
IA_swords <- IA_swords %>%
  mutate(Subtype = ifelse(Subtype == "Iron sword", "Iron sword (unspecified)", Subtype))

# Now check the unique values
unique(IA_swords$Subtype)
```

```
## [1] "Iron sword (unspecified)"            "Petersen B type iron sword"
## [3] "Iron mount of a sword shaft"         "Kirpichnikov A type iron sword"
## [5] "Petersen Z type iron sword"          "Petersen V type iron sword"
## [7] "Petersen H type iron sword"          "Iron sword with Brazil nut pommel"
## [9] "Petersen C type iron sword"
```

```
# Define custom point shapes
swords_point_shapes <- c(3, 4, 5, 6, 7, 8, 9, 10, 1)

swords_default_colors <- scales::hue_pal()(9)

IA_swords_fig6b <- core_map +
  geom_sf(data = IA_swords, aes(shape=Subtype,color=Subtype),size=2) +
  scale_shape_manual(values = swords_point_shapes) +
  scale_color_manual(values = swords_default_colors) +  # Automatically use default colors
  guides(shape = guide_legend(title = "Sword Types (n=124)", override.aes =
                                list(color = swords_default_colors)),
```

```
    color = "none")+
  ggtitle(str_wrap("Iron Age (500-1250 AD) swords", width =51))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
  legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
# Display the plot
IA_swords_fig6b
```




```
limits_IA_swords <- st_bbox(IA_swords) #data extent
limits_IA_swords # coordinates of the subset
```

```
##     xmin     ymin     xmax     ymax
## 20.09178 60.03981 25.19380 63.09347
```

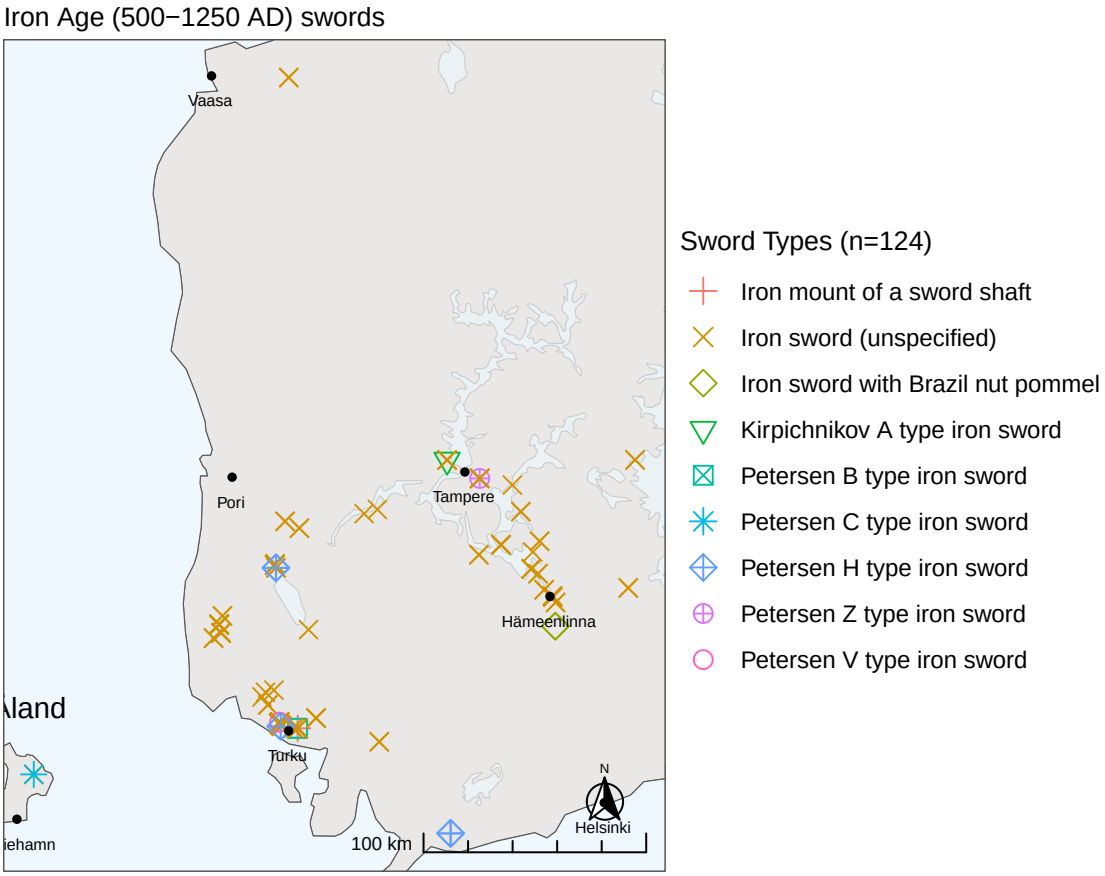
```
# Plotting the map within the specified extent
IA_swords_fig6b_zoomed <- core_map +
  geom_sf(data = IA_swords, aes(shape=Subtype,color=Subtype),size=3) +
  geom_sf(data = cities, size = 1)+
  geom_sf_text(data = cities, aes(label = NAME),color="black", size = 2,
               nudge_x = -0.01, nudge_y = -0.1) +
  scale_shape_manual(values = swords_point_shapes) +
  scale_color_manual(values = swords_default_colors) +
  guides(shape = guide_legend(title = "Sword Types (n=124)", override.aes =
```

```
                                list(color = swords_default_colors)), color = "none")+
  ggtitle(str_wrap("Iron Age (500-1250 AD) swords", width =51))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
  legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_IA_swords[1], limits_IA_swords[3]),
           ylim = c(limits_IA_swords[2], limits_IA_swords[4]))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
# Display the plot
IA_swords_fig6b_zoomed
```

```
## Scale on map varies by more than 10%, scale bar may be inaccurate
```



3.5.3 Iron Age beads, Figure 6c

This code focuses on bead artefacts where selection of predefined amount of recorded instances are mapped. It creates a advanced map visualisation, showing the bead types as different point symbols and colors on a map. Additionally, the order of class is adjusted in legend as desired and count value is added. The shape (point symbols) arrangement is achieved using `ggplot2` package function `scale_shape_manual()`.

```r
unique(IA_beads$Type) #observe 12 different bead types in the Iron Age beads dataset
```

```
##  [1] "Glass bead"          "Bronze bead"         "Stone bead"
##  [4] "Enamel bead"         "Clay bead"           "Bone bead"
##  [7] "Bead"                "Glazed clay bead"    "Glass drinking horn"
## [10] "Filigree bead"       "Porcelain bead"      "Amber bead"
```

```r
table(IA_beads$Type) # this dataset has 7 types with only 1-2 findings
```

```
##
##          Amber bead                Bead           Bone bead         Bronze bead
##                   1                   2                  18                  95
##           Clay bead         Enamel bead       Filigree bead          Glass bead
##                  36                   1                   1                 809
## Glass drinking horn    Glazed clay bead      Porcelain bead          Stone bead
##                   1                   1                   1                  26
```

```r
# Subset the dataset to filter those unique beads where there is more than two findings
IA_beads_subset <- IA_beads %>%
  group_by(Type) %>%
  filter(n() > 2) %>%
  ungroup()

# Calculate the count of each class
beads_class_counts <- table(IA_beads_subset$Type)
beads_class_counts
```

```
##
##   Bone bead Bronze bead   Clay bead  Glass bead  Stone bead
##          18          95          36         809          26
```

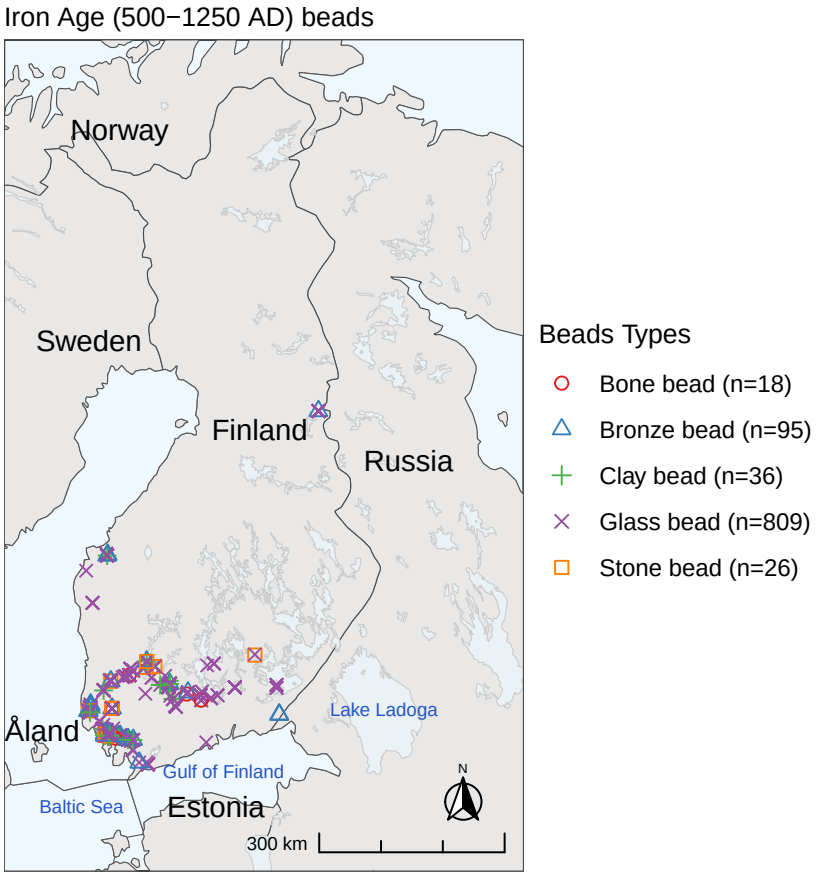
```r
# Define order of bead types in the legend, e.g. based on count from least to most represented.
beads_desired_order <- c("Bone bead", "Bronze bead", "Clay bead", "Glass bead", "Stone bead")
# Define custom point shapes
beads_point_shapes <- c(1, 2, 3, 4, 0)  # Shapes can be adjusted, more types in chapter 3.3.1
# Combine class names with counts
beads_legend_labels <- paste(beads_desired_order," (n=", beads_class_counts, ")", sep = "")

# Assuming beads has factors that need different colors
library(RColorBrewer)

# Choose a color palette from RColorBrewer
beads_default_colors <- brewer.pal(5, "Set1")


# Customize the appearance of the map
IA_beads_fig6c <- core_map +
  geom_sf(data = IA_beads_subset, aes(shape = Type, color=Type), size = 2)+ #adds selected bead types o
  scale_shape_manual(beads_desired_order,values = beads_point_shapes,
                     labels = beads_legend_labels) +
  scale_color_manual(values = beads_default_colors)+
```

```
  guides(shape = guide_legend(title = "Beads Types", override.aes = list(color = beads_default_colors))
  ggtitle(str_wrap("Iron Age (500-1250 AD) beads", width = 51)) +
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
  legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
# Display the plot
IA_beads_fig6c
```

```
## Scale on map varies by more than 10%, scale bar may be inaccurate
```




```
# Adding specified zoom level. As in chapter 3.4.1 "zooming in" the extent (subset area).
limits_IA_beads_subset <- st_bbox(IA_beads_subset) #data extent
limits_IA_beads_subset # coordinates of data subset
```

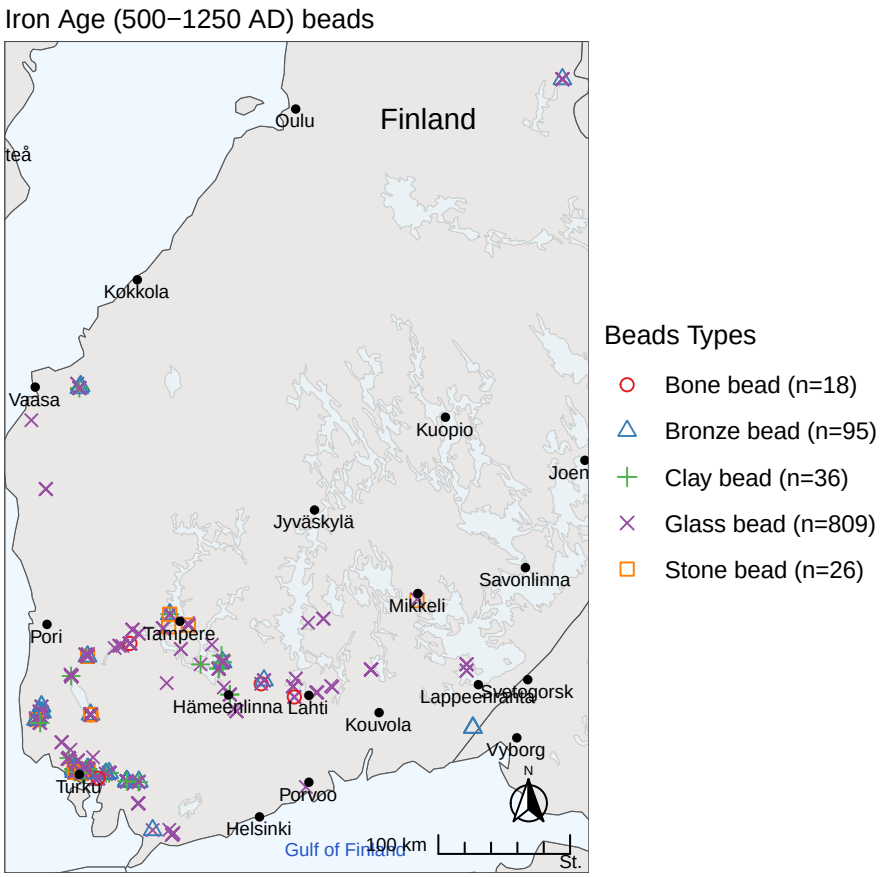
```
##     xmin     ymin     xmax     ymax
## 21.54238 60.03981 29.43283 65.20393
```

```
IA_beads_fig6c_zoomed <- core_map +
  geom_sf(data = IA_beads_subset, aes(shape=Type, color=Type),size=2) +
  geom_sf(data = cities, size = 1)+
  geom_sf_text(data = cities, aes(label = NAME),color="black", size = 2.5,
               nudge_x = -0.01, nudge_y = -0.08) +
  scale_shape_manual(beads_desired_order,values = beads_point_shapes,
                     labels = beads_legend_labels) +
  scale_color_manual(values = beads_default_colors)+
  guides(shape = guide_legend(title = "Beads Types", override.aes = list(color = beads_default_colors))
  ggtitle(str_wrap("Iron Age (500-1250 AD) beads", width =51))+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
  legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_IA_beads_subset[1], limits_IA_beads_subset[3]),
           ylim = c(limits_IA_beads_subset[2], limits_IA_beads_subset[4]))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
IA_beads_fig6c_zoomed
```

```
## Scale on map varies by more than 10%, scale bar may be inaccurate
```



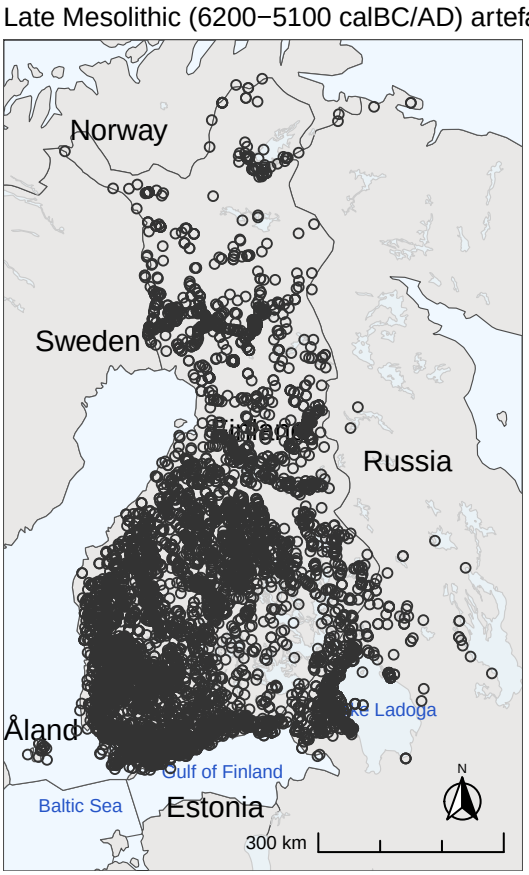
3.6 Archaeological periods

In this section, the AADA master table containing information on the Bronze Age (BA), Stone Age (SA), and Iron Age (IA) is used to make of different time period artefacts.

3.6.1 Late Mesolithic, Figure 7a

The object`m3` is created by subsetting the AADA master table to only include data from the Late Mesolithic period, which is contained in the ***m3*** column of the AADA data frame. Next, it creates a new subset named `m3_subset` by filtering for rows where the ***Category*** column is equal to "Pottery", "Clay artefacts", or "Amber artefacts" within the `m3` subset.

```
#subset Late Mesolithic which are in columns m3
m3 <- AADA %>%
  filter(m3)
# 14538 obs

latemesolithic_fig7a <- core_map +
  geom_sf(data = m3, color="grey20", size=1.4, shape=1,) +
  ggtitle("Late Mesolithic (6200-5100 calBC/AD) artefacts")+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
  legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))

# Display the plot
latemesolithic_fig7a
```



3.6.2 Early Neolithic, Figure 7b

The Early Neolithic artefacts are subsetted from the AADA table by filtering rows where the ***n1*** column is true. It then assigns this subset to the object `n1`. Next, it creates a new subset named `n1_subset` by filtering for rows where the ***Category*** column is equal to "Pottery", "Clay artefacts", or "Amber artefacts" within the `n1` subset.

```
#subset Early Neolithic which are in columns n1
n1 <- AADA %>%
  filter(n1)
# determine Pottery, Clay artefacts and Amber categories from n1 subset and
# assign them to the new subset named "n1_subset"
# 16162
table(n1$Category)
```

```
##
##  Amber artefacts   Birch bark tar   Bone artefacts   Clay artefacts
##                3               65              209              529
##          Pottery  Stone artefacts Wooden artefacts
##             1475            13874                7
```

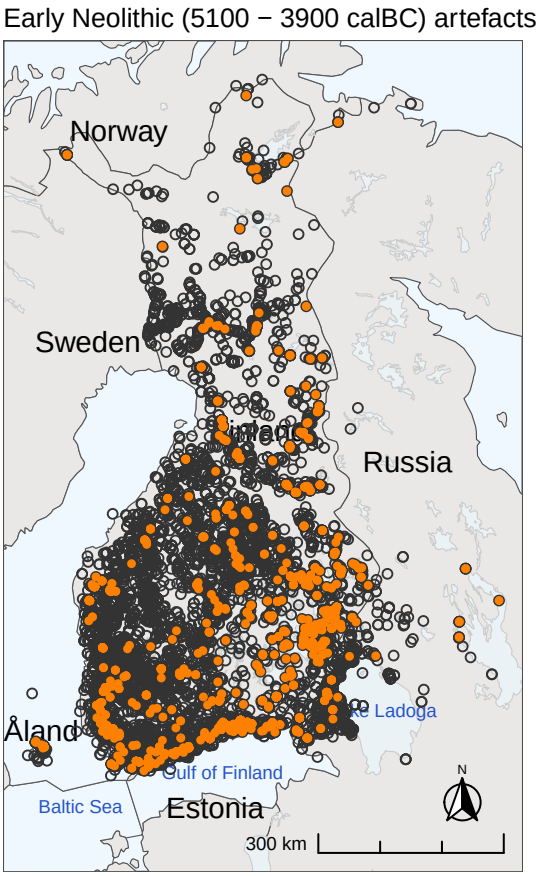
```
n1_subset <- n1 %>%
  filter(n1 & Category == "Pottery" | Category == "Clay artefacts" | Category == "Amber artefacts")
# 2007
```

```
earlyneolithic_fig7b <- core_map +
  geom_sf(data = n1, color="grey20", size=1.4, shape=1) +
  geom_sf(data = n1_subset,color = "#FE8000", size=1.5, shape=20) +
  ggtitle("Early Neolithic (5100 - 3900 calBC) artefacts")+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
  legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
# Display the plot
earlyneolithic_fig7b
```

```
## Scale on map varies by more than 10%, scale bar may be inaccurate
```



3.6.3 Middle Neolithic, Figure 7c

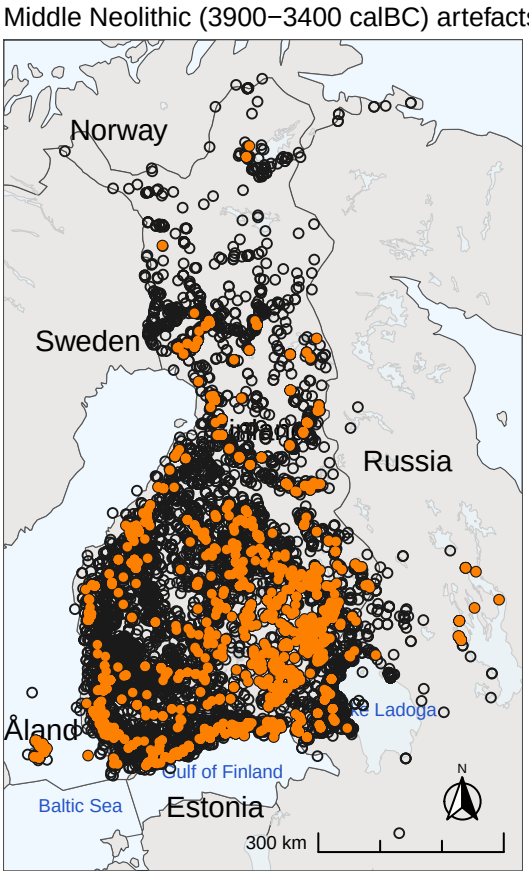
The object`n2` is created by subsetting the AADA master table to only include data from the Late Mesolithic period, which is contained in the ***n2*** column of the AADA data frame. A new subset named `n2_subset` is created by further filtering the `n2` subset for artifacts in the "Pottery", "Clay artefacts", and "Amber artefacts" categories. The image visualizes how pottery, clay and amber artefacts has increased.

```
n2 <- AADA %>%
  filter(n2)
n2_subset <- n2 %>%
  filter(n2 & Category == "Pottery" | Category == "Clay artefacts" | Category == "Amber artefacts")
# 5038 obs

middleneolithic_fig7c <- core_map +
  geom_sf(data = n2, color="grey10", size=1.4, shape=1) +
  geom_sf(data = n2_subset, color="#FE8000", size=1) +
  ggtitle("Middle Neolithic (3900-3400 calBC) artefacts")+
  theme(plot.title = element_text(margin = margin(b = 1, unit = "mm"), size = 10),
  legend.title = element_text(size = 10)) +
  coord_sf(xlim = c(limits_aada[1], limits_aada[3]),
           ylim = c(limits_aada[2], limits_aada[4]))
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```

```
# Display the plot
middleneolithic_fig7c
```

```
## Scale on map varies by more than 10%, scale bar may be inaccurate
```



3.7 Generating interactive graphics

This section demonstrates how to create interactive maps using the `mapview` package. The interactive maps allow users to explore the data more deeply and gain new insights through dynamic browsing. After making several selections through data query and subsetting, the subsets can be examined in the interactive map.

```
install.packages("mapview")
library(mapview)

# example 1, few observations
SA_spearhead %>% mapview()
mapview(SA_spearhead) + SA_battle_axe
mapview(SA_spearhead, color="red")+ SA_battle_axe
# given that data frame SA_n3_axe contains 3760 observations, it is advisable
# to utilize an interactive map viewer rather than use a static printed map
# example 2
SA_battle_axe %>% mapview()
# example 3
m3 %>% mapview() #only include data from the Late Mesolithic period, which is contained
# in the m3 column, altogether 14534 observations
```

3.8 Exporting map visualisations

This example creates a map visualisation using the `core_map` object and adds additional layers specific to Bronze Age bronze artifacts.

To export the visualisation as a JPG file, the `ggsave()` function is used. This function takes in the filename and the plot object as arguments, and the dpi parameter can be used to specify the image resolution. In this example, the visualisation is saved as "BA_bronze_fig3b.jpg" with a resolution of 300 dpi using the `ggsave()` function.

```
# Save plot as JPG file
# for printing purposes "gg" "ggplot" format of the maps are used
# save a .jpg image
# ggsave("AADA_map.jpg", plot = AADA_map, dpi = 600)
ggsave("SA_spearheads_fig4a.jpg", plot = SA_spearheads_fig4a, dpi = 600)
ggsave("SA_adze_chisel_fig4b.jpg", plot = SA_adze_chisel_fig4b, dpi = 600)
ggsave("SA_battle_axe_fig4c.jpg", plot = SA_battle_axe_fig4c, dpi = 600)
ggsave("BA_lovozero_vardoy_fig5a.jpg", plot = BA_lovozero_vardoy_fig5a, dpi = 600)
ggsave("BA_bronze_fig5b.jpg", plot = BA_bronze_fig5b, dpi = 600)
ggsave("BA_sarsa_tomitsa_fig5c.jpg", plot = BA_sarsa_tomitsa_fig5c, dpi = 600)
ggsave("IA_detailed_fig6a.jpg", plot = IA_detailed_fig6a, dpi = 600)
ggsave("IA_detailed_fig6a_zoomed.jpg", plot = IA_detailed_fig6a_zoomed, dpi = 600)
ggsave("IA_swords_fig6b.jpg", plot = IA_swords_fig6b, dpi = 600)
ggsave("IA_swords_fig6b_zoomed.jpg", plot = IA_swords_fig6b_zoomed, dpi = 600)
ggsave("IA_beads_fig6c.jpg", plot = IA_beads_fig6c, dpi = 600)
ggsave("IA_beads_fig6c_zoomed.jpg", plot = IA_beads_fig6c_zoomed, dpi = 600)
ggsave("latemesolithic_fig7a.jpg", plot = latemesolithic_fig7a, dpi = 600)
ggsave("earlyneolithic_fig7b.jpg", plot = earlyneolithic_fig7b, dpi = 600)
ggsave("middleneolithic_fig7c.jpg", plot = middleneolithic_fig7c, dpi = 600)
```

Session Info:

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17763)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Estonian_Estonia.1252  LC_CTYPE=Estonian_Estonia.1252
## [3] LC_MONETARY=Estonian_Estonia.1252 LC_NUMERIC=C
## [5] LC_TIME=Estonian_Estonia.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] RColorBrewer_1.1-3      ggpubr_0.6.0            ggspatial_1.1.9
##  [4] lubridate_1.9.2         forcats_1.0.0           purrr_1.0.1
##  [7] readr_2.1.4             tidyr_1.3.0             tibble_3.2.1
## [10] tidyverse_2.0.0         stringr_1.5.1           rnaturalearthdata_1.0.0
## [13] rnaturalearth_1.0.1     dplyr_1.1.1             ggplot2_3.5.1
## [16] sp_2.1-3                sf_1.0-16               readxl_1.4.3
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.7         jsonlite_1.8.8     carData_3.0-5      highr_0.10
##  [5] cellranger_1.1.0   yaml_2.3.8         pillar_1.9.0       backports_1.4.1
##  [9] lattice_0.20-45    glue_1.6.2         digest_0.6.30      ggsignif_0.6.4
## [13] colorspace_2.0-3   htmltools_0.5.8    pkgconfig_2.0.3    broom_1.0.5
## [17] s2_1.1.6           scales_1.3.0       terra_1.7-71       tzdb_0.3.0
## [21] timechange_0.1.1   proxy_0.4-27       generics_0.1.3     farver_2.1.1
## [25] car_3.1-2          withr_3.0.0        cli_3.4.1          magrittr_2.0.3
## [29] evaluate_0.23      fansi_1.0.3        rstatix_0.7.2      class_7.3-20
## [33] textshaping_0.3.7  tools_4.2.2        hms_1.1.3          lifecycle_1.0.4
## [37] munsell_0.5.0      compiler_4.2.2     e1071_1.7-12       systemfonts_1.0.5
## [41] rlang_1.1.0        classInt_0.4-8     units_0.8-1        grid_4.2.2
## [45] rstudioapi_0.15.0  labeling_0.4.3     rmarkdown_2.25     wk_0.9.1
## [49] gtable_0.3.4       codetools_0.2-19   abind_1.4-5        DBI_1.2.1
## [53] R6_2.5.1           knitr_1.45         fastmap_1.1.0      utf8_1.2.2
## [57] ragg_1.2.7         KernSmooth_2.23-20 stringi_1.7.8      Rcpp_1.0.12
## [61] vctrs_0.6.1        tidyselect_1.2.0   xfun_0.42
```