# Practical 3

Download the raw document here

This is the first summer that Lucia will spend in Salzburg. She is looking to visit a mountain hut where she can spend a night during the summer and do some hiking tours. **Can you help her?**

In this practical you will train the basics of data wrangling in R using the `tidyverse`.

You will also learn how to perform spatial queries using the `sf` package and perform raster-vector operations using `terra`. There are extra notes on the margins to give you more information on the functions you are using.

**Margin/aside/callout-box content, check online instructions**

## Part 1: Data import

In this section we will load a spatial dataset of mountain huts into R and clean it before using it in the next section.

We can use the `sf` package to load data into R. In the background, `sf` will use GDAL to identify the driver to properly load the data. As you will see, you can load data directly from an URL but also local files.

**Margin/aside/callout-box content, check online instructions**

```
library(sf)
```

```
Warning: package 'sf' was built under R version 4.4.3
```

```
Linking to GEOS 3.13.0, GDAL 3.10.1, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
huts = read_sf("https://github.com/loreabad6/app-dev-gis/raw/refs/heads/main/data/huts.gpkg")
```

Now we will start our data wrangling and cleaning workflows. For this we will use packages from the `tidyverse` but you can use base R or data.table if you have experience and feel more familiar with those.

**Margin/aside/callout-box content, check online instructions**

```
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.4.3
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```
Warning: package 'tibble' was built under R version 4.4.3
```

```
Warning: package 'tidyr' was built under R version 4.4.3
```

```
Warning: package 'readr' was built under R version 4.4.3
```

```
Warning: package 'purrr' was built under R version 4.4.3
```

```
Warning: package 'dplyr' was built under R version 4.4.3
```

```
Warning: package 'stringr' was built under R version 4.4.3
```

```
Warning: package 'forcats' was built under R version 4.4.3
```

```
Warning: package 'lubridate' was built under R version 4.4.3
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

`sf` is designed to work with base R but also with tidy workflows, so we can directly use `tidyverse` verbs to wrangle the data.

If we *glimpse* into the data we can have an idea of what we are dealing with.

```
huts |>
  glimpse()
```

```
Rows: 618
Columns: 193
$ osm_id                          <chr> "23785245", "25075130", "25439465", "2~
$ name                            <chr> "DAV-Haus Hammer", "Ruchenkopf-Hütte",~
$ access                          <chr> NA, "private", NA, NA, NA, NA, NA, NA,~
$ `addr:city`                     <chr> NA, NA, "Krimml", "Krimml", "Krimml", ~
$ `addr:country`                  <chr> NA, "DE", "AT", "AT", "AT", NA, NA, "A~
$ `addr:full`                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `addr:hamlet`                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `addr:housename`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `addr:housenumber`              <chr> NA, "4", "32", "27", "41", "1", "16", ~
$ `addr:place`                    <chr> NA, NA, "Oberkrimml", "Oberkrimml", "O~
$ `addr:postcode`                 <chr> NA, NA, "5743", "5743", "5743", "5611"~
$ `addr:source`                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `addr:street`                   <chr> NA, "Rotwand", NA, NA, NA, "Aubauernwe~
$ `addr:suburb`                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ aerialway                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ alt_ele                         <chr> NA, "1517", NA, NA, NA, NA, NA, NA, NA~
$ `alt_ele:source`                <chr> NA, "label on building", NA, NA, NA, N~
$ alt_name                        <chr> NA, "Hütte der Ruchenköpfler", NA, NA,~
$ amenity                         <chr> NA, NA, NA, "restaurant", NA, "restaur~
$ `at_bev:addr_date`              <chr> NA, NA, NA, NA, NA, NA, NA, "2017-10-0~
$ `at_bev:addr_date_last_checked` <chr> NA, NA, "2017-10-01", "2017-10-01", "2~
```

```
$ bar                                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ beds                               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ beer_garden                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ bench                              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ biergarten                         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ bin                                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ bivac_room                         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `bivac_room:access`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ brand                              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ brewery                            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ building                           <chr> "yes", "yes", "yes", "yes", "yes", "ye~
$ `building:colour`                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `building:levels`                  <chr> NA, NA, NA, NA, NA, NA, NA, "3", NA, N~
$ `building:levels:underground`      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `building:material`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ capacity                           <chr> NA, NA, NA, "59", NA, NA, NA, NA, NA, ~
$ `capacity:beds`                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `capacity:dormitory`               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `capacity:overnight`               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `cash_withdrawal:operator`         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ changing_table                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ charge                             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ check_date                         <chr> NA, NA, NA, NA, NA, NA, NA, "2024-07-1~
$ `check_date:internet_access`       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `check_date:opening_hours`         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ construction                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:address`                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:email`                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:facebook`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:fax`                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:instagram`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:mobile`                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:phone`                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `contact:website`                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ cuisine                            <chr> NA, NA, NA, "regional", NA, "alpine_hu~
$ delivery                           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ description                        <chr> "DAV-Selbstversorgerhütte DAV-Haus Ham~
$ `description:en`                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `diet:vegan`                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `diet:vegetarian`                  <chr> NA, NA, NA, "yes", NA, NA, NA, NA, NA,~
$ dogs                               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ drinking_water                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, "yes",~
$ drive_through                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
$ ele                          <chr> "750", "1509", NA, "1620", "2328", "17~
$ `ele:source`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ electricity                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ email                        <chr> NA, NA, "richterhütte@gmx.at", NA, "in~
$ entrance                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ eve                          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ facebook                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ fax                          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ fee                          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ fireplace                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ fixme                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ height                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ heritage                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `heritage:operator`          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ highway                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ image                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ indoor_seating               <chr> NA, NA, NA, "yes", NA, NA, NA, NA, NA,~
$ inscription                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ internet_access              <chr> NA, NA, NA, NA, NA, NA, NA, "no", "no"~
$ `internet_access:access`     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `internet_access:fee`        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `internet_access:private`    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `internet_access:ssid`       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `internet_access:wlan`       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ layer                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ leisure                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ level                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ loc_name                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ locked                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ massif                       <chr> NA, NA, NA, NA, "Zillertaler Alpen", N~
$ mattress                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ microbrewery                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ mobile                       <chr> NA, NA, NA, NA, "+43 (0)664 873 22 05"~
$ month_off                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ month_on                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `name:de`                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `name:en`                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `name:etymology:wikidata`    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `name:ru`                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ note                         <chr> NA, NA, NA, NA, "Koordinaten vom ÖAV z~
$ `note:access`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `note:de`                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ old_name                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
$ opening_date                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ opening_hours                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `opening_hours:bivac_room`    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `opening_hours:signed`        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ operator                      <chr> "DAV Sektion München und Oberland", NA~
$ `operator:restaurant`         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `operator:tenant`             <chr> NA, NA, NA, NA, NA, NA, NA, NA, "NN", ~
$ outdoor_seating               <chr> NA, NA, NA, "yes", NA, NA, NA, NA, NA,~
$ `payment:american_express`    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:cards`               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:cash`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:coins`               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:contactless`         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:credit_cards`        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:debit_cards`         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:diners_club`         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:discover_card`       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:electronic_purses`   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:girocard`            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:jcb`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:maestro`             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:mastercard`          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:notes`               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `payment:visa`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ phone                         <chr> NA, NA, "+43 6564 7328", "+43 664 2612~
$ `phone:mobile`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ power_supply                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ ref                           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `ref:at:bda`                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `ref:avf`                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, "144",~
$ reg_name                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ reservation                   <chr> "members_only", "members_only", NA, NA~
$ restaurant                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `roof:colour`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `roof:height`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `roof:levels`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `roof:material`               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `roof:orientation`            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `roof:shape`                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ rooms                         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ self_service                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ service                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ shelter                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
$ shelter_type                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ short_name                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ shower                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, "yes",~
$ `shower:fee`                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `shower:hot_water`            <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ sink                          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ smoking                       <chr> NA, NA, NA, NA, NA, NA, NA, "outside",~
$ source                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `source:date`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `source:ele`                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `source:name`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `source:old_name`             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `source:operator`             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `source:outline`              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ staff                         <chr> NA, "no", NA, NA, NA, NA, NA, NA, "sea~
$ start_date                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "1~
$ takeaway                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ tenant                        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ toilets                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `toilets:access`              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `toilets:charge`              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `toilets:fee`                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `toilets:fee:conditional`     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `toilets:female`              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `toilets:male`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `toilets:wheelchair`          <chr> NA, NA, NA, NA, NA, "no", NA, NA, NA, ~
$ tourism                       <chr> "wilderness_hut", "wilderness_hut", "a~
$ url                           <chr> NA, NA, NA, NA, "http://www.zittauerhu~
$ `url:official`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, "http:~
$ website                       <chr> "https://www.alpenverein-muenchen-ober~
$ `website:booking`             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `website:menu`                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ `website:menu:en`             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ wheelchair                    <chr> NA, NA, NA, NA, NA, "no", NA, NA, "no"~
$ `wheelchair:description`      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ wikidata                      <chr> NA, NA, "Q2151300", "Q1734546", "Q2065~
$ wikimedia_commons             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ wikipedia                     <chr> NA, NA, "de:Richterhütte", NA, "de:Zit~
$ winter_room                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ wpt_description               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ wpt_symbol                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ type                          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ geom                          <POINT [°]> POINT (11.94015 47.70373), POINT~
```

Now that is a long file! You might have noticed some interesting patterns here and there, but what gives this data away is the first column: `osm_id`.

Now if this is a spatial file, where are the coordinates? Take a look at the last column of the data: `geom`.

```
huts$geom
```

```
Geometry set for 618 features
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 11.93262 ymin: 46.84716 xmax: 14.10032 ymax: 48.08579
Geodetic CRS:  WGS 84
First 5 geometries:

POINT (11.94015 47.70373)

POINT (11.95847 47.64878)

POINT (12.13402 47.12415)

POINT (12.19086 47.13893)

POINT (12.12424 47.16277)
```

These are basically the locations of our huts, and sf already knows to look for those coordinates in this column.

To have a quick view of where your data is located, you can use the `mapview()` function from the `{mapview}` package.

```
library(mapview)
mapview(huts)
```

## Part 2: Data cleaning

### Margin/aside/callout-box content, check online instructions

So now you know this is OpenStreetMap data. If you have ever worked with OSM data before you will know that their nodes have several tags with their properties attached to them. When querying the data with R, you will get in this case the queried huts in each row with all the tags in different columns forming a data frame.

Using this messy data you will be wrangling and tidying a bit so that you can work with a more manageable dataset in the next section.

> ❗ Watch out!
>
> From now on, each of the code chunks below will cause and error and/or will do the desired task incorrectly. (Even the chunks that run without error are not correct!) You will need to find the mistake, and correct it, to complete the intended action.

You will see a big amount of `NA` values in the huts data. That is because not all OSM tags are filled for every mountain hut, but if one hut has it, then it is included in the dataset.

1. Let's reduce the number of variables in the dataset. Let's narrow the dataset to:

- Name of the hut
- Elevation of the hut
- Capacity (no. of beds)
- Amenity (is it a restaurant, a bar, a self-service hut?)
- Operator of the hut (Alpenverein, Naturfreunde, etc.)
- Location of the hut (the coordinates)

```
huts_clean = huts |>
  select(name, ele, capacity, beds, amenity, operator, geom)

# Check the cleaned data
glimpse(huts_clean)
```

```
Rows: 618
Columns: 7
$ name     <chr> "DAV-Haus Hammer", "Ruchenkopf-Hütte", "Richterhütte", "Krimm~
$ ele      <chr> "750", "1509", NA, "1620", "2328", "1795", "2051", "1966", "2~
$ capacity <chr> NA, NA, NA, "59", NA, NA, NA, NA, NA, NA, NA, NA, "150", NA, ~
$ beds     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ amenity  <chr> NA, NA, NA, "restaurant", NA, "restaurant", NA, NA, NA, NA, "~
$ operator <chr> "DAV Sektion München und Oberland", NA, NA, NA, "Sektion Warn~
```

```
$ geom     <POINT [°]> POINT (11.94015 47.70373), POINT (11.95847 47.64878), P~
```

Tip

Take a close look at the result of your selection, are all the columns that you asked for there? What about the `geom` column? Did you ask for it? Is it anyway there? Let's try to get rid of it.

```
# THIS CHUNK HAS NO ERROR!!!!
huts_clean |> select(-geom)
```

```
Simple feature collection with 618 features and 6 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 11.93262 ymin: 46.84716 xmax: 14.10032 ymax: 48.08579
Geodetic CRS:  WGS 84
# A tibble: 618 x 7
   name          ele   capacity beds  amenity operator               geom
   <chr>         <chr> <chr>    <chr> <chr>   <chr>            <POINT [°]>
 1 DAV-Haus Ham~ 750   <NA>     <NA>  <NA>    DAV Sek~ (11.94015 47.70373)
 2 Ruchenkopf-H~ 1509  <NA>     <NA>  <NA>    <NA>     (11.95847 47.64878)
 3 Richterhütte  <NA>  <NA>     <NA>  <NA>    <NA>     (12.13402 47.12415)
 4 Krimmler Tau~ 1620  59       <NA>  restau~ <NA>     (12.19086 47.13893)
 5 Zittauer Hüt~ 2328  <NA>     <NA>  <NA>    Sektion~ (12.12424 47.16277)
 6 Aualm         1795  <NA>     <NA>  restau~ Richard~ (13.16634 47.26281)
 7 Passauer Hüt~ 2051  <NA>     <NA>  <NA>    <NA>     (12.75033 47.47421)
 8 Schmidt-Zabi~ 1966  <NA>     <NA>  <NA>    DAV Sek~ (12.64632 47.55062)
 9 Riemannhaus   2177  <NA>     <NA>  <NA>    Sektion~ (12.91447 47.45801)
10 Alte Traunst~ 1580  <NA>     <NA>  <NA>    DAV Sek~ (12.79342 47.62545)
# i 608 more rows
```

Oh no! It is still there! Well, that is because of how `sf` objects work. The geometry column is a "sticky" column, meaning that it cannot be dropped with tidyverse verbs. But, we want to work with spatial data, so we are not really going to remove that column.

**Margin/aside/callout-box content, check online instructions**

2. Now, let's adjust the proper variables to be numeric. We use the `mutate()` function which helps you change existing columns (if you save the result with the same column name) or to create new columns (by giving it a new column name).

```
huts_clean = huts_clean |>
  mutate(
    ele = as.numeric(ele),
    capacity = as.numeric(capacity),
    beds = as.numeric(beds)
  )
# Check structure
glimpse(huts_clean)
```

```
Rows: 618
Columns: 7
$ name     <chr> "DAV-Haus Hammer", "Ruchenkopf-Hütte", "Richterhütte", "Krimm~
$ ele      <dbl> 750, 1509, NA, 1620, 2328, 1795, 2051, 1966, 2177, 1580, 1560~
$ capacity <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, NA, NA, ~
$ beds     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ amenity  <chr> NA, NA, NA, "restaurant", NA, "restaurant", NA, NA, NA, NA, "~
$ operator <chr> "DAV Sektion München und Oberland", NA, NA, NA, "Sektion Warn~
$ geom     <POINT [°]> POINT (11.94015 47.70373), POINT (11.95847 47.64878), P~
```

3. We will next create a new variable called "capacity_overall". This column will combine the columns "capacity" and "beds". When there is no capacity value, then the beds value will be taken. Otherwise the capacity value is taken. If both columns are `NA`, then the column will also have an `NA`. For this we can use the function `case_when()` inside the `mutate()` function.

```
huts_clean = huts_clean |>
  mutate(
    capacity_overall = case_when(
      is.na(capacity) ~ beds,
      TRUE ~ capacity
    )
  )

# Check result
glimpse(huts_clean)
```

```
Rows: 618
Columns: 8
$ name             <chr> "DAV-Haus Hammer", "Ruchenkopf-Hütte", "Richterhütte"~
$ ele              <dbl> 750, 1509, NA, 1620, 2328, 1795, 2051, 1966, 2177, 15~
$ capacity         <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
```

```
$ beds             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ amenity          <chr> NA, NA, NA, "restaurant", NA, "restaurant", NA, NA, N~
$ operator         <chr> "DAV Sektion München und Oberland", NA, NA, NA, "Sekt~
$ geom             <POINT [°]> POINT (11.94015 47.70373), POINT (11.95847 47.6~
$ capacity_overall <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
```

4. Considering that the huts are located in Europe, we can project the data from WGS84 to a more appropriate CRS. Let's use the European Equal Area "EPSG:3035".

#| eval: false

```
huts_clean = huts_clean |>
  st_transform(crs = 3035)

# Check CRS
st_crs(huts_clean)
```

```
Coordinate Reference System:
  User input: EPSG:3035
  wkt:
PROJCRS["ETRS89-extended / LAEA Europe",
    BASEGEOGCRS["ETRS89",
        ENSEMBLE["European Terrestrial Reference System 1989 ensemble",
            MEMBER["European Terrestrial Reference Frame 1989"],
            MEMBER["European Terrestrial Reference Frame 1990"],
            MEMBER["European Terrestrial Reference Frame 1991"],
            MEMBER["European Terrestrial Reference Frame 1992"],
            MEMBER["European Terrestrial Reference Frame 1993"],
            MEMBER["European Terrestrial Reference Frame 1994"],
            MEMBER["European Terrestrial Reference Frame 1996"],
            MEMBER["European Terrestrial Reference Frame 1997"],
            MEMBER["European Terrestrial Reference Frame 2000"],
            MEMBER["European Terrestrial Reference Frame 2005"],
            MEMBER["European Terrestrial Reference Frame 2014"],
            MEMBER["European Terrestrial Reference Frame 2020"],
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]],
            ENSEMBLEACCURACY[0.1]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4258]],
    CONVERSION["Europe Equal Area 2001",
        METHOD["Lambert Azimuthal Equal Area",
```

```
            ID["EPSG",9820]],
        PARAMETER["Latitude of natural origin",52,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",10,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["False easting",4321000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",3210000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["northing (Y)",north,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["easting (X)",east,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["Statistical analysis."],
        AREA["Europe - European Union (EU) countries and candidates. Europe - onshore and of
        BBOX[24.6,-35.58,84.73,44.83]],
    ID["EPSG",3035]]
```

5. Finally, note how we started each code chunk with: `huts_clean = huts_clean |>`.

That is very redundant and can cause you trouble if you are recreating your object over and over again.

We can pipe all these steps together to have one single workflow for data cleaning. In the code chunk below, combine the (fixed!) code.

```
huts_clean = huts |>
  select(name, ele, capacity, beds, amenity, operator, geom) |> # select only necessary colu
  mutate(
    ele = as.numeric(ele),                # convert to numeric
    capacity = as.numeric(capacity),
    beds = as.numeric(beds),
    capacity_overall = case_when(         # create capacity_overall column
      is.na(capacity) ~ beds,
      TRUE ~ capacity
```

```
    )
  ) |>
  st_transform(crs = 3035)                # reproject to EPSG:3035

# Check result
glimpse(huts_clean)
```

```
Rows: 618
Columns: 8
$ name             <chr> "DAV-Haus Hammer", "Ruchenkopf-Hütte", "Richterhütte"~
$ ele              <dbl> 750, 1509, NA, 1620, 2328, 1795, 2051, 1966, 2177, 15~
$ capacity         <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
$ beds             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ amenity          <chr> NA, NA, NA, "restaurant", NA, "restaurant", NA, NA, N~
$ operator         <chr> "DAV Sektion München und Oberland", NA, NA, NA, "Sekt~
$ geom             <POINT [m]> POINT (4466683 2734145), POINT (4468215 2728077~
$ capacity_overall <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
```

```
st_crs(huts_clean)
```

```
Coordinate Reference System:
  User input: EPSG:3035
  wkt:
PROJCRS["ETRS89-extended / LAEA Europe",
    BASEGEOGCRS["ETRS89",
        ENSEMBLE["European Terrestrial Reference System 1989 ensemble",
            MEMBER["European Terrestrial Reference Frame 1989"],
            MEMBER["European Terrestrial Reference Frame 1990"],
            MEMBER["European Terrestrial Reference Frame 1991"],
            MEMBER["European Terrestrial Reference Frame 1992"],
            MEMBER["European Terrestrial Reference Frame 1993"],
            MEMBER["European Terrestrial Reference Frame 1994"],
            MEMBER["European Terrestrial Reference Frame 1996"],
            MEMBER["European Terrestrial Reference Frame 1997"],
            MEMBER["European Terrestrial Reference Frame 2000"],
            MEMBER["European Terrestrial Reference Frame 2005"],
            MEMBER["European Terrestrial Reference Frame 2014"],
            MEMBER["European Terrestrial Reference Frame 2020"],
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]],
            ENSEMBLEACCURACY[0.1]],
```

```
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4258]],
    CONVERSION["Europe Equal Area 2001",
        METHOD["Lambert Azimuthal Equal Area",
            ID["EPSG",9820]],
        PARAMETER["Latitude of natural origin",52,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",10,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["False easting",4321000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",3210000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["northing (Y)",north,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["easting (X)",east,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["Statistical analysis."],
        AREA["Europe - European Union (EU) countries and candidates. Europe - onshore and of
        BBOX[24.6,-35.58,84.73,44.83]],
    ID["EPSG",3035]]
```

> ⚠ Checkpoint
>
> Up to this point, your clean dataset should have 4.6% of the number of columns in the original dataset.

```
# Write code to verify that your huts_clean dataset has 4.6%
# of the number of columns in the huts dataset

# Count the number of columns in both datasets
ncol_original = ncol(huts)
ncol_clean = ncol(huts_clean)
```

```r
# Calculate percentage
percentage = (ncol_clean / ncol_original) * 100

# Print result
cat("Percentage of columns retained:", round(percentage, 1), "%\n")
```

```
Percentage of columns retained: 4.1 %
```

### Part 3: Enrich your data

So far we have used only wrangling and cleaning functions. Now, we are going to enrich our dataset with other spatial datasets.

> **!** Watch out!
>
> In this section, you will get a series of instructions, you should implement code to fulfil the task.

1. The huts are located in different regions. You have a `regions` dataset here: https://github.com/loreabad6/app-dev-gis/raw/refs/heads/main/data/regions.gpkg. Load the data using the `sf` package.

```r
regions = read_sf("https://github.com/loreabad6/app-dev-gis/raw/refs/heads/main/data/regions
# Check the data
glimpse(regions)
```

```
Rows: 33
Columns: 3
$ region  <chr> "Bolzano - Bozen", "Bezirk Liezen", "Bezirk Murau", "Landkreis~
$ country <chr> "Italy", "Austria", "Austria", "Germany", "Germany", "Austria"~
$ geom    <MULTIPOLYGON [°]> MULTIPOLYGON (((10.47669 46..., MULTIPOLYGON (((1~
```

```r
st_crs(regions)  # Check CRS
```

```
Coordinate Reference System:
  User input: WGS 84
  wkt:
GEOGCRS["WGS 84",
```

```
    ENSEMBLE["World Geodetic System 1984 ensemble",
        MEMBER["World Geodetic System 1984 (Transit)"],
        MEMBER["World Geodetic System 1984 (G730)"],
        MEMBER["World Geodetic System 1984 (G873)"],
        MEMBER["World Geodetic System 1984 (G1150)"],
        MEMBER["World Geodetic System 1984 (G1674)"],
        MEMBER["World Geodetic System 1984 (G1762)"],
        MEMBER["World Geodetic System 1984 (G2139)"],
        MEMBER["World Geodetic System 1984 (G2296)"],
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]],
        ENSEMBLEACCURACY[2.0]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    USAGE[
        SCOPE["Horizontal component of 3D system."],
        AREA["World."],
        BBOX[-90,-180,90,180]],
    ID["EPSG",4326]]
```

2. Now, we will perform a spatial join of the "huts_clean" data and the "regions" data. For this you can use the function `st_join()`. Remember you can check for function documentation by typing `?st_join` on the console.

*Hint:* you most likely get an error when you first try to do your join. **READ THE ERROR MESSAGE CAREFULLY!** What does it tell you?

```r
# Reproject regions to match huts_clean CRS
regions = regions |>
  st_transform(crs = st_crs(huts_clean))

# Perform spatial join
huts_enrich = huts_clean |>
  st_join(regions)
```

```
# Check result
glimpse(huts_enrich)
```

```
Rows: 618
Columns: 10
$ name             <chr> "DAV-Haus Hammer", "Ruchenkopf-Hütte", "Richterhütte"~
$ ele              <dbl> 750, 1509, NA, 1620, 2328, 1795, 2051, 1966, 2177, 15~
$ capacity         <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
$ beds             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ amenity          <chr> NA, NA, NA, "restaurant", NA, "restaurant", NA, NA, N~
$ operator         <chr> "DAV Sektion München und Oberland", NA, NA, NA, "Sekt~
$ geom             <POINT [m]> POINT (4466683 2734145), POINT (4468215 2728077~
$ capacity_overall <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
$ region           <chr> "Landkreis Miesbach", "Landkreis Miesbach", "Bezirk Z~
$ country          <chr> "Germany", "Germany", "Austria", "Austria", "Austria"~
```

**Margin/aside/callout-box content, check online instructions**

3. Now let's add some data about maximum temperature. For this you will find a `.tif` file here: https://github.com/loreabad6/app-dev-gis/raw/refs/heads/main/data/AUT_wc2.1_30s_tmax.tif. You can load this raster dataset using the `rast()` function from the `{terra}` package.

**Margin/aside/callout-box content, check online instructions**

```
library(terra)
```

```
Warning: package 'terra' was built under R version 4.4.3
```

```
terra 1.8.29
```

```

Attaching package: 'terra'
```

```
The following object is masked from 'package:tidyr':

    extract
```

```
tmax = rast("https://github.com/loreabad6/app-dev-gis/raw/refs/heads/main/data/AUT_wc2.1_30s_
tmax
```

```
class       : SpatRaster
dimensions  : 420, 960, 12  (nrow, ncol, nlyr)
resolution  : 0.008333333, 0.008333333  (x, y)
extent      : 9.5, 17.5, 46, 49.5  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : AUT_wc2.1_30s_tmax.tif
names       : AUT_w~max_1, AUT_w~max_2, AUT_w~max_3, AUT_w~max_4, AUT_w~max_5, AUT_w~max_6,
min values  :       -12.1,       -13.0,       -12.4,       -10.8,        -6.2,        -2.8,
max values  :         7.5,         9.5,        13.8,        17.6,        23.2,        26.3,
```

```
# Check the raster
tmax
```

```
class       : SpatRaster
dimensions  : 420, 960, 12  (nrow, ncol, nlyr)
resolution  : 0.008333333, 0.008333333  (x, y)
extent      : 9.5, 17.5, 46, 49.5  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : AUT_wc2.1_30s_tmax.tif
names       : AUT_w~max_1, AUT_w~max_2, AUT_w~max_3, AUT_w~max_4, AUT_w~max_5, AUT_w~max_6,
min values  :       -12.1,       -13.0,       -12.4,       -10.8,        -6.2,        -2.8,
max values  :         7.5,         9.5,        13.8,        17.6,        23.2,        26.3,
```

**Margin/aside/callout-box content, check online instructions**

4. Note that there are 12 layers in this dataset. These correspond to the 12 months in the year. We can change the names of the layers with:

```
# you don't need to change anything here!
names(tmax) = month.abb
```

5. We are interested in the summer months (June, July, August, September). Let's get the mean `tmax` for these months.

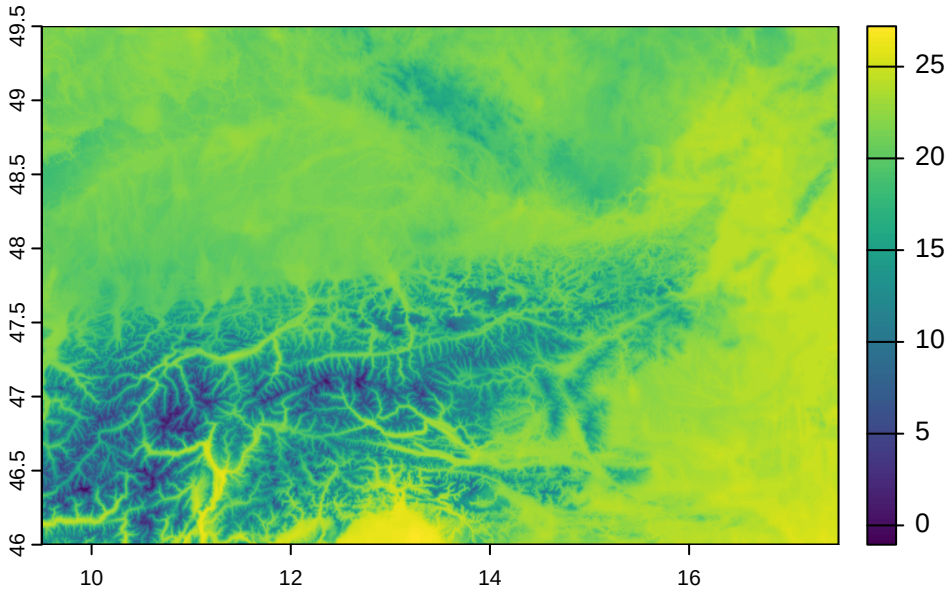
```r
library(terra)

# Calculate the mean tmax for summer months: Jun, Jul, Aug, Sep
tmax_mean = mean(tmax[[c("Jun", "Jul", "Aug", "Sep")]])

# Check the result
tmax_mean
```

```
class       : SpatRaster
dimensions  : 420, 960, 1  (nrow, ncol, nlyr)
resolution  : 0.008333333, 0.008333333  (x, y)
extent      : 9.5, 17.5, 46, 49.5  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source(s)   : memory
name        :  mean
min value   : -1.05
max value   : 27.20
```

```r
plot(tmax_mean)  # Optional: visualize
```




**Margin/aside/callout-box content, check online instructions**

6. Now, let's actually add the temperature information to the hut dataset. We can use the `terra::extract()` function to do this.

```
# Extract the mean summer temperature at each hut location
tmax_mean_huts = terra::extract(tmax_mean, huts_enrich)
```

```
Warning: [extract] transforming vector data to the CRS of the raster
```

```
# Check the result
head(tmax_mean_huts)
```

```
  ID   mean
1  1 20.050
2  2 16.250
3  3  8.400
4  4 14.925
5  5 10.400
6  6 12.325
```

If you print this data you will notice that this is a data frame with the exact number of points as the **sf** object. The order is the same as the one in your dataset. Therefore, you can add this information directly as a new column to the "huts_enrich" dataset.

**Margin/aside/callout-box content, check online instructions**

```
# Add the mean summer temperature to the huts_enrich dataset
huts_enrich = huts_enrich |>
  mutate(tmax_summer = tmax_mean_huts[[2]])

# Check result
glimpse(huts_enrich)
```

```
Rows: 618
Columns: 11
$ name             <chr> "DAV-Haus Hammer", "Ruchenkopf-Hütte", "Richterhütte"~
$ ele              <dbl> 750, 1509, NA, 1620, 2328, 1795, 2051, 1966, 2177, 15~
$ capacity         <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
$ beds             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ amenity          <chr> NA, NA, NA, "restaurant", NA, "restaurant", NA, NA, N~
$ operator         <chr> "DAV Sektion München und Oberland", NA, NA, NA, "Sekt~
$ geom             <POINT [m]> POINT (4466683 2734145), POINT (4468215 2728077~
```

```
$ capacity_overall <dbl> NA, NA, NA, 59, NA, NA, NA, NA, NA, NA, NA, NA, 150, ~
$ region           <chr> "Landkreis Miesbach", "Landkreis Miesbach", "Bezirk Z~
$ country          <chr> "Germany", "Germany", "Austria", "Austria", "Austria"~
$ tmax_summer      <dbl> 20.050, 16.250, 8.400, 14.925, 10.400, 12.325, 13.575~
```

> ⚠ Checkpoint
>
> Up to this point, your enriched dataset should have mean "tmax" temperatures between 1.15 °C and 21.4 °C.

```r
# Write code to verify that the tmax_summer column in
# the huts_enrich dataset ranges between the above values

# Find the minimum and maximum of tmax_summer
summary(huts_enrich$tmax_summer)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.15   13.20   14.91   14.63   16.52   21.48
```

```r
# Alternatively, just check range
range(huts_enrich$tmax_summer, na.rm = TRUE)
```

```
[1]  1.150 21.475
```

```r
# Logical check
min(huts_enrich$tmax_summer, na.rm = TRUE) >= 1.15 &
  max(huts_enrich$tmax_summer, na.rm = TRUE) <= 21.4
```

```
[1] FALSE
```

### Part 4: Find the dream summer hut!

Remember Lucia? She is very excited to find the perfect hut for her. Now that you have a clean and enriched dataset, you can help her find it!

Here are her requirements:

- The hut should be at a good enough altitude to enjoy the views, she thinks **huts above 800 m** should be good enough!

- Temperature is also an important factor for Lucia. She wants to escape the heat from the valley but not freeze at the top! The **maximum temperature should be higher than 15 °C** on average over the summer months.
- She wants to stay in a small hut, nothing with too many other guests (otherwise it gets so hot!), but she doesn't want to be completely alone either. Something **between 10 and 30 overall capacity** sounds good for her.
- She needs to be sure she can actually eat at the hut, are there **huts with restaurants**?
- She will start close to the train station and doesn't want to drive long… what is the closest hut from Salzburg Hbf?

> **Tip**
>
> Hints for the last point…
>
> - You can find distances with `sf::st_distance()`. Create a `sf` object as `sbg_hbf = st_sfc(st_point(c("x", "y")), crs = "EPSG")`.
> - Don't forget to handle the CRS correctly! (Set CRS and transform).
> - You may want to create a new column first and then filter on it…

Use your `huts_enrich` object and filter for Lucia's requirements. After filtering, you should have only one hut as a result.

```
library(sf)
library(dplyr)
library(mapview)
```

```
Warning: package 'mapview' was built under R version 4.4.3
```

```
# 1. Create Salzburg Hbf point (in WGS84, then reproject to match huts_enrich)
sbg_hbf = st_sfc(st_point(c(13.038775, 47.823497)), crs = 4326) |>
  st_transform(crs = st_crs(huts_enrich))

# 2. Add distance column to huts_enrich using the proper geometry accessor
huts_enrich = huts_enrich |>
  mutate(distance_to_sbg = as.numeric(st_distance(st_geometry(huts_enrich), sbg_hbf)))

# 3. Filter based on Lucia's criteria and get the closest hut
result = huts_enrich |>
  filter(
    ele > 800,
    tmax_summer > 15,
    capacity_overall >= 10,
```

```
    capacity_overall <= 30,
    amenity == "restaurant"
  ) |>
  slice_min(order_by = distance_to_sbg, n = 1)

# 4. View result
result
```

```
Simple feature collection with 1 feature and 11 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 4508387 ymin: 2740568 xmax: 4508387 ymax: 2740568
Projected CRS: ETRS89-extended / LAEA Europe
# A tibble: 1 x 12
  name           ele capacity  beds amenity   operator              geom
  <chr>        <dbl>    <dbl> <dbl> <chr>     <chr>          <POINT [m]>
1 Hochgernhaus  1510       29    NA restaura~ Moritz ~ (4508387 2740568)
# i 5 more variables: capacity_overall <dbl>, region <chr>, country <chr>,
#   tmax_summer <dbl>, distance_to_sbg <dbl>
```

Where is the hut located? Make an interactive map!

```
mapview(result)
```

> **i** Solution
>
> **Margin/aside/callout-box content, check online instructions**

Upload the .qmd and PDF to Blackboard (don't forget to add all your teammates/your name(s)!). The first team receives an extra point each in class participation