```r
knitr::opts_chunk$set(echo = TRUE)
```

Why Open Science

Open science is about making the methods, data and outcomes in your analysis available to everyone. It includes:

- Greater transparency in the experimental methodology, observation, and collection of data.
- Better public availability and reusability of data.
- Better public accessibility and transparency through better communication techniques.
- Utilizing web-based tools to facilitate collaboration.

In this tutorial, you are not going to learn all aspects of open science as listed above. However, you will learn one tool that can be used to make your workflows:

- More transparent.
- More available and accessible to the public and your colleagues.

You will learn how to document your work - by connecting data, methods and outputs in one or more reports or documents. You will learn the R Markdown file format which can be used to generate reports that connect your data, code (methods used to process the data) and outputs. You will use the rmarkdown and knitr package to write R Markdown files in Rstudio and publish them in different formats (html, pdf, etc).

About R Markdown

Simply put, `.Rmd` is a text based file format that allows you to include both descriptive text, code blocks and code output. You can run the code in R using a package called knitr (which you will learn about next). You can export the text formated .Rmd file to a nicely rendered, shareable format like pdf or html. When you knit (or use knitr), the accompanying code is executed, resulting the outputs (e.g. plots, and other figures) appearing in the rendered document.

R Markdown (.Rmd) is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R. It combines the core syntax of markdown (an easy to write plain text format) with embedded R code chunks that are run so their output can be included in the final document. R Markdown documents are fully reproducible (they can be automatically regenerated whenever underlying R code or data changes).“ RStudio documentation.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunks in your knitr markdown using:

```
1
```

There are also several option that you can add to this fucntion `{r}` to change how your code runs (e.g. `{r, include=FALSE}`).

Markdown basics

Now let's learn additional basics that you can use for creating your markdown documents.

Text

Plain text

End a line with two spaces

to start a new paragraph.

Highlighted text and special characters

italics and **bold**

`verbatim code`

sub/superscript^{2}~2~

~~strikethrough~~

####
escaped: * _ \

endash: --, emdash: ---

equation: $A = \pi * r^2$

equation block:
$$E = mc^2$$

> block quote

Header1 {#anchor}

Header 2 {#css_id}

Header 3 {.css_class}

Header 4

Header 5

Header 6

\textbf{Tex ignored in HTML}

HTML ignored in pdfs

http://www.rstudio.com

link

Jump to Header 1

image:


- unordered list
- sub-item 1
- sub-item 2
- sub-sub-item 1
- item 2
  Continued (indent 4 spaces)

1. ordered list
2. item 2
   i) sub-item 1
    A. sub-sub-item 1
   (@) A list whose numbering
   continues after
   (@) an interruption

Term 1: Definition 1

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

- slide bullet 1
- slide bullet 2
  (>- to have bullets appear on click)

horizontal rule/slide break:

A footnote [1]

```
summary(cars)
```

Including Plots

You can also embed plots, for example:

```
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Projects and Github in R Studio

To initiate your project, the first step is to establish a dedicated project folder that will serve as a centralized repository for all your data and code, facilitating efficient organization. Within your coding or development environment, you can create this project folder by selecting "File" and then "New File." If your aim is to keep your code and data organized locally, consider adding a "New Directory" within the project folder. Alternatively, for more advanced version control and collaboration, you can opt to utilize Version Control through platforms like GitHub. To do so, start by creating a GitHub account on github.com and proceed to establish a repository to host your project, enabling seamless version control and collaborative project management. This structured approach ensures that your project remains well-organized and accessible for effective development and collaboration. Create a repository by selecting "New"

- Name your repository what you like.
- Enter a description for your repository.
- Choose Public visibility.
- Select Initialize this repository with a README.
- Click Add .ignore and select R.
- Click Create repository.

With the repository created, we can get the url to it using the code button and copying it.

Now we will return to R Studio, and create a project with "Version Control", choose Git

Proceed by following the prompt to add the Repository URL for your project. This action will initiate the download of the repository you have created on GitHub to your computer. During this process, you will specify the directory and folder in which you want to store the downloaded repository. Once this download is successfully executed, you will gain access to a "git" tab within your coding or development environment. This tab will serve as your control center for managing version control, allowing you to track changes, collaborate with others, and maintain a well-organized and up-to-date project.

With the repository successfully downloaded and integrated into your project folder, any files you add or modify within this folder will be continuously monitored. This means that every change or addition you make to files, excluding those specified in the "gitignore" file, will be tracked. Moreover, you will have the option to push these changes to your GitHub repository, ensuring that your project remains up to date on the remote repository. This streamlined process simplifies version control and collaboration, allowing you to effectively manage and synchronize your project across different environments.

Create your own markdown and start committing!

```
library(ggplot2)
library(sf)
library(tidyverse)
```

1. Let's also grab some data here. This is spatial point dataset that I have collected as part of a project in the Open Spaces and Moutain Parks of Boulder Colorado. It consist of the points where people have taken pictures using Flickr and Panramio. We have also collected several spatial varibles that might explain why individuals might be taking photographs at these points and all other points in park. We will import the data as a sf spatial dataset.

```
boulder <- st_read("./BoulderSocialMedia/BoulderSocialMedia.shp")
boulder
```

This Here are the details of data:

Variable	Description
DB	indicates whether the point is a social media location (Flickr or Panramio) or a point in the park
extent	extent that can be viewed at each point estimated through viewshed analysis
Climb_dist	distance to nearest climbing wall
TrailH_Dis	distance to hiking trails
NatMrk_Dis	distance to natural landmark
Trails_dis	distance to walking trails
Bike_dis	distance to biking trails
PrarDg_Dis	distance to prairie dog mounds
PT_Elev	Elevation
Hydro_dis	distance to lakes, rivers and creeks
Street_dis	distance to streets and parking lots

2. We can plot these variables using ggplot2. We define the sf data using the `geom_sf` function. The different arguments control the object attributes(this can be points, lines or polygons). For example, `fill=` control the color of object outline. `alpha =` controls the opacity of the object. The final argument is a complete theme, which controls the non-data display(e.g. neatlines, gradicule title). More details can be found regarding these [themes] here(https://ggplot2.tidyverse.org/reference/ggtheme.html). Here we use `theme_bw`, which is the black and white theme. You can try other themes to explore the different options.

```
ggplot() +
    geom_sf(data =boulder,
    fill = NA, alpha = .2) +
    theme_bw()
```

3. At the moment, the projection is a bit weird. Let's project the data using an appropriate projection for Colorado. Use the epsg.io website for choosing the an appropriate projection

```
boulder = st_transform(boulder, 26753)
ggplot() +
    geom_sf(data =boulder,
    fill = NA, alpha = .2) +
    theme_bw()
```

Commit and push the changes to GitHub

Now that you have created the R Markdown document, you might want to start committing these changes.

- In RStudio click the Git tab in the upper right pane.
- ick Commit.
- In the Review changes view, check the staged box for all files.
- Add a commit message, for example Add initial code.
- Click Commit.
- Click the Pull button to fetch any remote changes(perhaps others working on the code).
- Click the Push button to push your changes to the remote repository.
- On GitHub, navigate to the Code tab of the repository to see the changes.

As you progress with your project, you can commit your changes periodically to maintain a clear version history. This practice ensures that you can easily track and revert to previous states of your project if needed. Additionally, for added convenience, you can clone your project's code onto other machines. This allows you to seamlessly continue your coding work across different environments, such as your UMICH workstation and personal machines, without any disruptions or discrepancies.

4. Now we will explore different methods for visualizing this data. We will add 'Gradient colour scales' in ggplot2. Here is the documentation of these options https://ggplot2.tidyverse.org/reference/scale_gradient.html.

```
ggplot() +
    geom_sf(data =boulder, aes(color=PT_Elev),
    fill = NA, alpha = .2) +
    theme_bw()
```

5. `ggplot2` has several gradient colour scale options. The details can be found here.

```
ggplot() +
    geom_sf(data =boulder, aes(color=PT_Elev),
    fill = NA, alpha = .2) +
  scale_colour_gradientn(colours = terrain.colors(10)) +
  theme_bw()
```

6. Let's look at the locations above 2200 meters. For this we will need to use the `ifelse()` function. The function basically means if the first argument is true (PT_Elev >= 2200), the elevation is greater than 2200 meter, then print the first varible: TRUE; if not true, print the second varible: FALSE. We use the mutate fucntion to make a new variable in our boulder dataframe. We then use ggplot to plot these locations.

```
#library(dplyer)
boulder %>%
    mutate(high_elev = ifelse(PT_Elev >= 2200, TRUE, FALSE))%>%
ggplot() +
  geom_sf(aes(color=high_elev),
    fill = NA, alpha = .2)  +
  theme_bw()
```

7. We can also plot different charts using ggplot. Let's compare the distance from roads and social media photographs. Here we `filter()` to analyze social media only. We use a box plot to compare mean distance of these photographs from the nearest road. What does this test?

```
boulder %>%
  filter(DB ==  'Pano' | DB == 'Flickr') %>%
  ggplot(aes(x=DB, y=Street_dis)) +
  geom_boxplot()
```

As you can see there is no significant relationship. The mean values and standard deviation is highly similar. There are numerous other tests and charts that you can use to investigate the relationship between locations of soical media photographs and other locations in the park.

Additional Geovis tools

We are also going to learn about two new packages that might be helpful for your data science approach. We will learn about the `library(viridis)`, which provides color palettes that are interpretable for visually impaired.

The color scale

The package viridis contains four color scales: “Viridis”, the primary choice, and three alternatives with similar properties, “magma”, “plasma”, and “inferno”.

```
library(sf)
library(ggspatial)
library(viridis)
## the function gives the hexadecimal colors
## the interger give the numbers of colors
magma(10)
```

```
boulder <- st_read("./BoulderSocialMedia/BoulderSocialMedia.shp")
ggplot() +
    geom_sf(data = boulder, aes(color=PT_Elev),
    fill = NA, alpha = .2) +
    scale_colour_gradientn(colours = magma(10))
```

We can also plot discrete values.

```
summary(boulder$DB)
```

```
p <- ggplot() +
  annotation_spatial(boulder) +
  layer_spatial(boulder, aes(col = DB))
p + scale_color_brewer(palette = "Dark2")
```

tmaps

Alternatively, we can use tmap a way to create maps using R

```
library(tmap)
## Add the data - these are specific to the vector or raster
tm_shape(boulder) +
  ## which variable, is there a class interval, palette, and other options
  tm_symbols(col='PT_Elev',
             style='quantile',
             palette = 'YlOrRd',
             border.lwd = NA,
             size = 0.1)

```

It is really easy to add cartographic elements in tmap

```
## here we are using a simple dataset of the world
# tmap_mode("plot")
data("World")
tm_shape(World) +
    tm_polygons("gdp_cap_est", style='quantile', legend.title = "GDP Per
Capita Estimate")
```

It is really easy to make an interactive map in tmap as well

```
## the view mode creates an interactive map
tmap_mode("view")

tm_shape(World) +
    tm_polygons("gdp_cap_est", style='quantile', legend.title = "GDP Per Capita Estimate")

```

Advanced Week 1 Lab Assignment

In this week's lab, you will make an open science markdown that documents your process of data analysis and geovisualization. We will be using git to aid in version control for the code. Your assignment is to use Knitr to develop a markdown document that shows your analysis of the Boulder data (you can also use your own data if you wish). Demonstrate how you did your analysis giving step-by-step instructions with the accompanying code.

Questions

1. Discuss the advantages and challenges associated with an open data science approach. Provide an example based on this week's reading. (1-2 paragraphs)
2. Create a markdown document that showcases an analysis of this week's data or any other dataset of your choice. Include descriptive text that explains your analysis, and incorporate figures and geovisualizations.Include 1 chart and 1 map. Structure and explain your analysis with text, headings, highlights, images and other markdown basics.

Bonus: Capture a screenshot of the history of your Git commits. Share your strategy for utilizing Git in your workflow.

Here are the evaluation criteria for the geovisualizations. Questions will be worth 30% of your grade, while the geovisualization and explanation will be worth 70%.

Evaluation	***Highly well-done***	***Well-done***	***Some deficiencies***	***Several deficiencies***
Cartographic principles - 20% (title, name, date, north arrow, scale, legend, explanation symbols)	Elements present and correctly portrayed (100%)	Most elements present and correctly portrayed (99-80%)	Some elements (when appropriate) present and correctly portrayed (79-50%)	Minimal information (<50%)

Evaluation	*Highly well-done*	*Well-done*	*Some deficiencies*	*Several deficiencies*
Presentation and Legibility - 20% (readable, consistency and ease of understanding, flow of ideas consistent with cognition, clear explanation of content)	Highly legible, consistent and easy to understand (100%)	Mostly legible, consistent and easy to understand (99 -80%)	Somewhat legible, consistent and easy to understand (79-50%)	Minimally legible, consistent and poorly understandable (<50%)
Content - 20% (relevant, coherent and interesting topic, appropriate subject matter given the presented information/data, free of bias and error)	Highly relevant coherent, and interesting; consistent information free of bias and error (100%)	Mostly relevant coherent, and interesting; consistent information free of bias and error (99 -80%)	Somewhat relevant coherent, and interesting; some inconsistencies in information(79-50%)	Minimally relevant coherent, and interesting; inconsistencies in information (<50%)
Aesthetics - 20% (is the map attractive, are there objective elements that are popularly viewed as beautiful)	Highly attractive/ beautiful (100%)	Mostly attractive/ beautiful (99 -80%)	Somewhat attractive/beautiful (79-50%)	Minimally attractive beautiful (<50%)
Creativity and persuasiveness - 20% (imaginative information/data, convincing argumentation, presence of sustainability principles)	Highly imaginative; convincing of sustainability principles (100%)	Mostly imaginative; convincing of sustainability principles (99 -80%)	Somewhat imaginative; less convincing of sustainability principles (79-50%)	Minimally imaginative; not convincing of sustainability principles (<50%)

Optional steps for Hosting a HTML of your RMD as a Website on GitHubn

It is rather simple to make your html publicly available via github. Here is an example of one I made for a recent paper https://derekvanberkel.github.io/Planning-for-climate-migration-in-Great-Lake-Legacy-Cities/. Below are the step to make the knit html you make for this lab into a static website. Here is another website that give more detail https://blog.flycode.com/how-to-deploy-a-st

atic-website-for-free-using-github-pages

Create a New Repository:

1. Click on the '+' sign at the top right corner and select "New repository."

Fill in Repository Information:

1. Choose a name for your repository. This will be part of your website's URL, so choose it accordingly.
2. You can choose to make the repository public (visible to everyone) or private (restricted access).
3. Optionally, add a description for your repository.
4. Make sure the "Initialize this repository with a README" option is unchecked.
5. Click the "Create repository" button.

Add Your HTML File:

Now, you need to add your HTML file to the repository. You can do this in several ways:

- Use the GitHub web interface to upload your HTML file. Click on the "Add file" button, then select "Upload files" and follow the instructions.
- If you're comfortable with Git, you can clone your repository to your local machine, add your `index.html` file to the local folder, and push the changes back to GitHub.

Commit Changes:

After adding your HTML file to the repository, you need to commit the changes. On the GitHub website:

1. Navigate to the repository.
2. Click on the "Add file" button and select "Create a new file."
3. Name the file `index.html` and add your HTML code to it.
4. Scroll down to the "Commit new file" section.
5. Enter a "Commit summary" (e.g., "Initial commit").
6. Click the "Commit new file" button.

Configure GitHub Pages:

Once your HTML file is in the repository, go to your repository's main page.

1. Click on the "Settings" tab (located towards the right, under your repository's name).
2. Scroll down to the "GitHub Pages" section
3. Navigate to the `Pages` tab and click it
4. Under the "Source" section, click the dropdown under "Branch" and select "main" (or your repository's default branch).
5. Click the "Save" button.

Wait for Deployment:

GitHub Pages may take a few minutes to build and deploy your site. Be patient; it usually happens within 10 minutes.

Access Your Live Website:

After GitHub Pages has deployed your site, you'll find the URL associated with your website in the "GitHub Pages" section of your repository's settings. It should be something like `https://yourusername.github.io/repositoryname`.

---

1. Here is the footnote. ↩