IGE and Off-Campus Study: An RShiny App

CSC-324-02 Technical Documentation

Natalia Ramirez Jimenez, Ishita Sarraf, Caitlin Abreu, Dylan Hong, and Lucas Holler

2024-05-08

Table of contents

User Stories

1. As a user, I can easily use the UI to view desired visualizations.

i. There are six tabs which lead to each of the different visualizations.

ii. The user can change tabs as they please without causing any visual bugs.

2. As a user, I can use a cluster map to see how many students went where and to what program.

i. The map renders the number of visits and locations of off-campus programs correctly.

ii. The user can explore the map and cities by zooming in and out on the map.

iii. The clusters display the number of cities that are close geographically, and they expand to individual city markers.

iv. The markers have pop-ups that show the visit count and links to active off-campus programs.

3. As a user, I can use a stacked bar chart to see the ethnicities of students going to different countries

i. The bars accurately show the proportions of ethnicities per country in different colors.

ii. The user can filter the chart by region.

iii. The user can hover over the segments to see the ethnicity/count.

4. As a user, I can use a segmented bar chart to compare the costs of different off-campus programs for a semester compared to Grinnell.

i. The bars accurately show the different programs in each level of cost compared to a semester in Grinnell.

ii. Hovering over the bars shows the program and its need-based aid.

5. As a user, I can use a stacked bar chart to see how many students have travelled abroad each semester.

i. The bars accurately show the student counts per term in chronological order.

ii. The user can filter the chart by region and term.

iii. The user can hover over the segments to see the count of students abroad each term.

6. As a user, I can use a line chart to see the number of students travelling to each region overtime.

i. The lines accurately show the student counts to each region each year.

ii. The user can use the sidebar to include/exclude any particular region(s).

7. As a user, I can use a stacked bar chart to see the gender proportions of the students visiting each country.

i. The bars accurately show the gender proportions for the different countries.

Reproducibility

First, the original CSV file of the trip information (provided by IGE) is read from the github. The column names are then changed for convenience:

```
colnames(dataFile) <- c("ApplicationCycle", "Ethnicity", "AnticipatedCompletionDate",
                        "ProgramName", "CountryName", "ProgramCity", "ProgramRegion",
                        "ApplicationID", "ProgramYear", "ProgramTerm",
                        "ApplicationStatusAlias", "Gender", "PreferredGender")
```

Next, changed certain values in the dataset. For ethnicity, "Non-resident alien" was replaced with "International student" at IGE's request to be more inclusive. A couple bugs were fixed for the Grinnell in London Program, like the program name being misspelled and removing duplicate entries.

```
dataFile$Ethnicity <- ifelse(dataFile$Ethnicity == "Non-resident alien",
                             "International student", dataFile$Ethnicity)

dataFile$ProgramName <- ifelse(dataFile$ProgramName ==
                                 "*<B>Grinnell-in-London</B>",
                               "Grinnell in London", dataFile$ProgramName)

dataFile <- dataFile[dataFile$ProgramCity != "undefined", ]
```

Region names for countries were then manually changed to match IEE classification.

```
dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "Latin America",
                                 "Latin America and Caribbean",
                                 dataFile$ProgramRegion)

dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "South America",
                                 "Latin America and Caribbean",
                                 dataFile$ProgramRegion)

dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "Caribbean",
                                 "Latin America and Caribbean",
                                 dataFile$ProgramRegion)


dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "Middle East", "MENA",
                                 dataFile$ProgramRegion)

dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "Southeast Asia", "Asia",
                                 dataFile$ProgramRegion)

dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "South Asia", "Asia",
                                 dataFile$ProgramRegion)
```

```
dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "East Asia", "Asia",
                                 dataFile$ProgramRegion)

dataFile$ProgramRegion <- ifelse(dataFile$ProgramRegion ==
                                   "Central Asia", "Asia",
                                 dataFile$ProgramRegion)


dataFile$ProgramRegion <- ifelse(dataFile$CountryName ==
                                   "Tunisia", "MENA",
                                 dataFile$ProgramRegion)

dataFile$ProgramRegion <- ifelse(dataFile$CountryName ==
                                   "Cameroon",
                                 "Sub-Saharan Africa",
                                 dataFile$ProgramRegion)

dataFile$ProgramRegion <- ifelse(dataFile$CountryName ==
                                   "Kenya", "Sub-Saharan Africa",
                                 dataFile$ProgramRegion)
```

Then, we added two extra columns for academic year, since the programs were listed as calendar year. One column has the academic year in string format, and the other column has the first year of that string in numeric format.

```
assign_year <- function(app_cycle) {
  if (app_cycle == "Academic Year 2016" || app_cycle ==
      "Fall 2016" || app_cycle == "Spring 2017") {
    return("2016-2017")
  }

  else if (app_cycle == "Academic Year 2017" || app_cycle ==
           "Fall 2017" || app_cycle == "Spring 2018") {
    return("2017-2018")
  }

  else if (app_cycle == "Academic Year 2018" || app_cycle ==
           "Fall 2018" || app_cycle == "Spring 2019") {
    return("2018-2019")
  }

  else if (app_cycle == "Academic Year 2019" || app_cycle ==
           "Fall 2019" || app_cycle == "Spring 2020") {
    return("2019-2020")
  }

  else if (app_cycle == "Academic Year 2020" || app_cycle ==
           "Fall 2020" || app_cycle == "Spring 2021") {
    return("2020-2021")
  }

  else if (app_cycle == "Academic Year 2021" || app_cycle ==
```

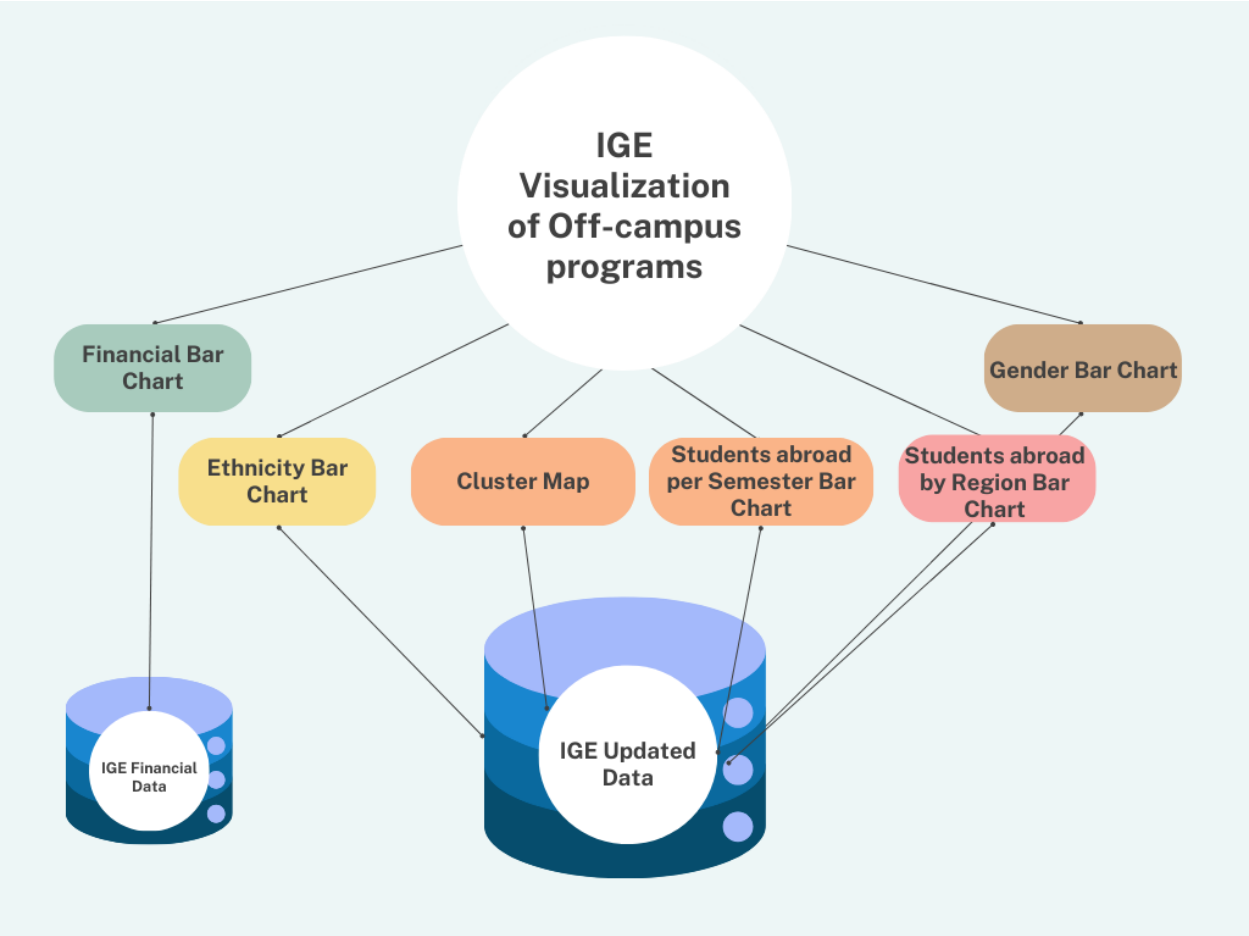
```
           "Fall 2021" || app_cycle == "Spring 2022") {
    return("2021-2022")
  }

  else if (app_cycle == "Academic Year 2022" || app_cycle ==
           "Fall 2022" || app_cycle == "Spring 2023") {
    return("2022-2023")
  }

  else if (app_cycle == "Academic Year 2023" || app_cycle ==
           "Fall 2023" || app_cycle == "Spring 2024") {
    return("2023-2024")
  } else {
    return("NA")
  }
}

for (i in 1:nrow(dataFile)) {
  dataFile$AcademicYear[i] <-
    assign_year(dataFile$ApplicationCycle[i])
}



dataFile$StartYear <-
  as.numeric(substr(dataFile$AcademicYear, 1, 4))
```

Finally, the cleaned data file is written to the local repository to be uploaded to the github.

```
write.csv(dataFile, file = "Updated OCS Data 3-7.csv", row.names = TRUE)
```

Architectural Design

We use a service-based architecture where our database is deployed to the different visualizations which each go to the user interface. More specifically, we have student trip data which goes to five different visualizations, and financial aid data which goes to one visualization. These six visualizations are then viewable by changing tabs in the UI.



Design Decisions

Tabs UI

Cluster Map

What: A world map displaying all off-campus programs in the form of clusters. The clusters contain markers that are geometrically (spatially) close, when a user clicks on a cluster the map will zoom in into its position and display the individual city markers. The clusters are ordered and quantitative, they display the number of markers contained.

Why: The user can `look up` (`search`) the desired city by continuously clicking on a cluster, it helps identifying the programs in a region. This incentives user interaction with the visualization.

How: We encoded the spatial attributes of the data into single markers that point to a specific city. The clusters are created based on the location of this markers, and the `hue` of the clusters will depend on how many markers are in one region. The hues vary from green to yellow, where yellow indicates a larger density of markers. The clusters are shown as a circular shape.

Ethnicity Bar Chart

What: The segmented bar chart displays proportions/frequencies of ethnicities of students that have visited countries through the study abroad program. Each bar is segmented by ethnicity, and user can filter by region or by ethnicity; users can also hover over bars to see the count of students of the selected ethnicity.

Why: The segmented bar chart is used to `compare` (`query`) the frequencies/proportions of ethnicities of studies that have visited select regions/countries. This may help the user to decide which country they want to visit, and

How: The bar chart uses filtering to reduce the number of regions shown so that the user is not overloaded with information, the `line` mark with `vertical spatial position` channel for the varying proportions of ethnicity, `horitzontal spatial position` to distinguish countries, and `hue` to differentiate ethnicities.

Financial Bar Chart

What: The segmented bar chart (`flat table`) displaying the difference between the cost of a semester off-campus vs. a semester in Grinnell for each program. Each bar is segmented for each program and upon hovering over a segment, the user can see the name of the program with the available need-based aid.

Why: The segmented bar chart is to `compare` (`query`) the difference between the cost of a semester off-campus vs. a semester in Grinnell including the need-based aid available. This visualization allows the user to compare cost of different programs in comparison to Grinnell and select the program they find the most suitable.

How: The segmented bar chart uses the `line` mark with `vertical spatial position` channel for the number of programs for that particular cost comparison to Grinnell and the `horizontal spatial position` channel for the different levels of cost comparisons to Grinnell.

Students Abroad each Semester Bar Chart

What: The segmented bar chart shows the user how many students were abroad each year, and during what term (Spring, Fall, Full Year) they were abroad. Each bar is segmented by term and hovering over a segment will display the term and frequency of students abroad, and the user can filter based on term.

Why: The segmented bar chart is to `discover` (`consume`) the amounts/proportions of students that study abroad in each term over time; this allows the student to learn about what terms students typically study abroad.

How: The visualization arranges the data chronologically, uses the `line` mark with `vertical spatial position` channel for the number of students abroad per term, `horizontal spatial position` for different academic years, and `hue` to distinguish each term.

Students Abroad in each Region Line Chart

What: The line chart is a `network` that links different years based on the number of students for each region. Each region has a different line which a user can include/exclude based on their interests.

Why: The line chart is to `present` (`analyze`) the different number of students going to a particular region for each year. The user can select their regions of interest and the line chart helps to present the data to the user.

How: The line chart uses the `line` mark with the `length` and `color` channels. The lines are of different lengths based on the number of students for each year for that region and are colored with different colors for the different regions.

Gender Bar Chart

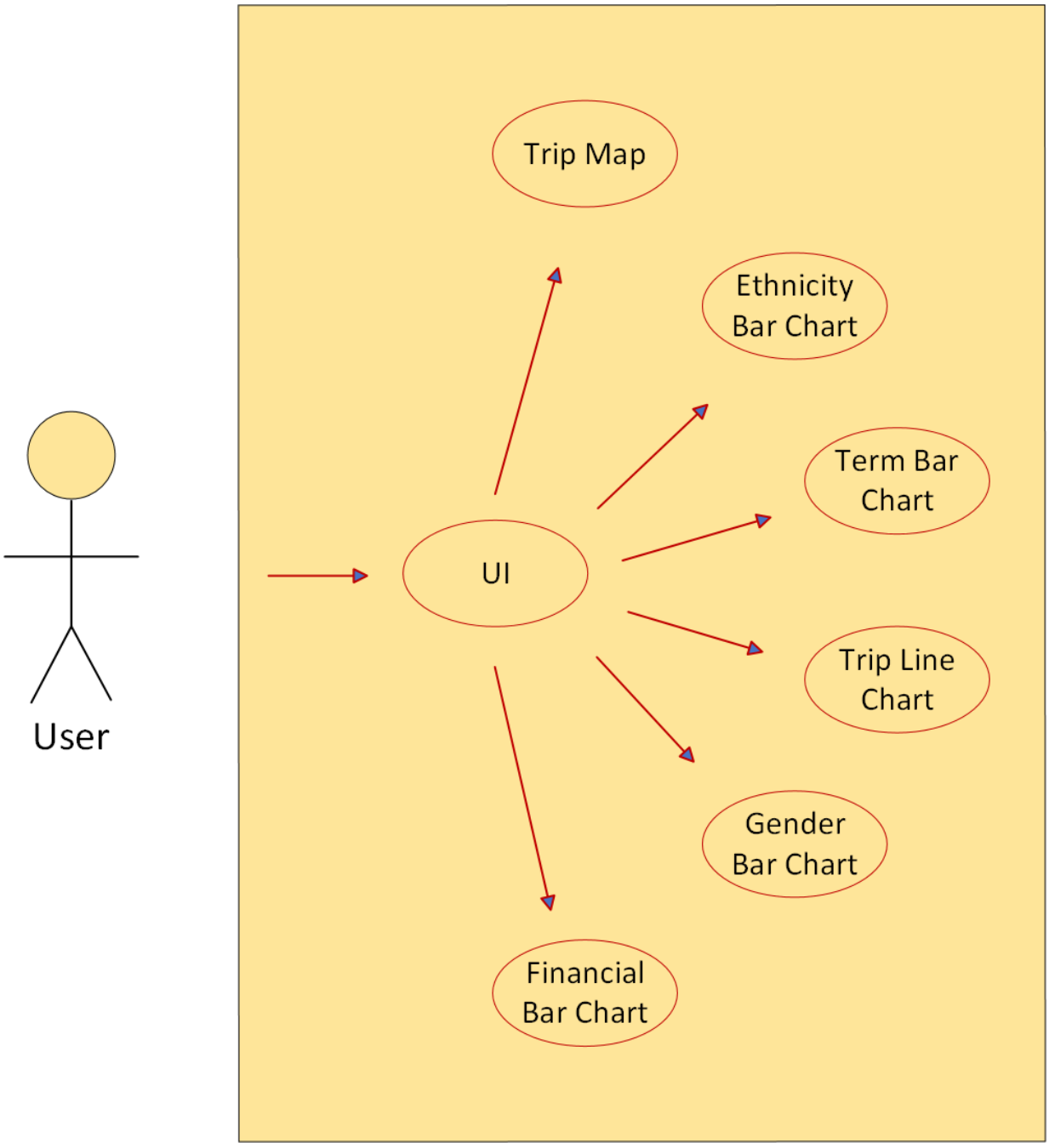
What: The gender bar chart displays proportions/frequencies of male and female students who have participated in the study abroad program, segmented by gender. Each bar is segmented by gender, and users can filter the data by region or by gender. Additionally, users can hover over the bars to see the count of students of the selected gender.

Why: The gender bar chart is used to 'compare' ('query') the frequencies/proportions of male and female students who have visited select regions/countries through the study abroad program. This visualization can help prospective students decide which country they might want to visit, providing insights into gender distribution in various study abroad destinations.

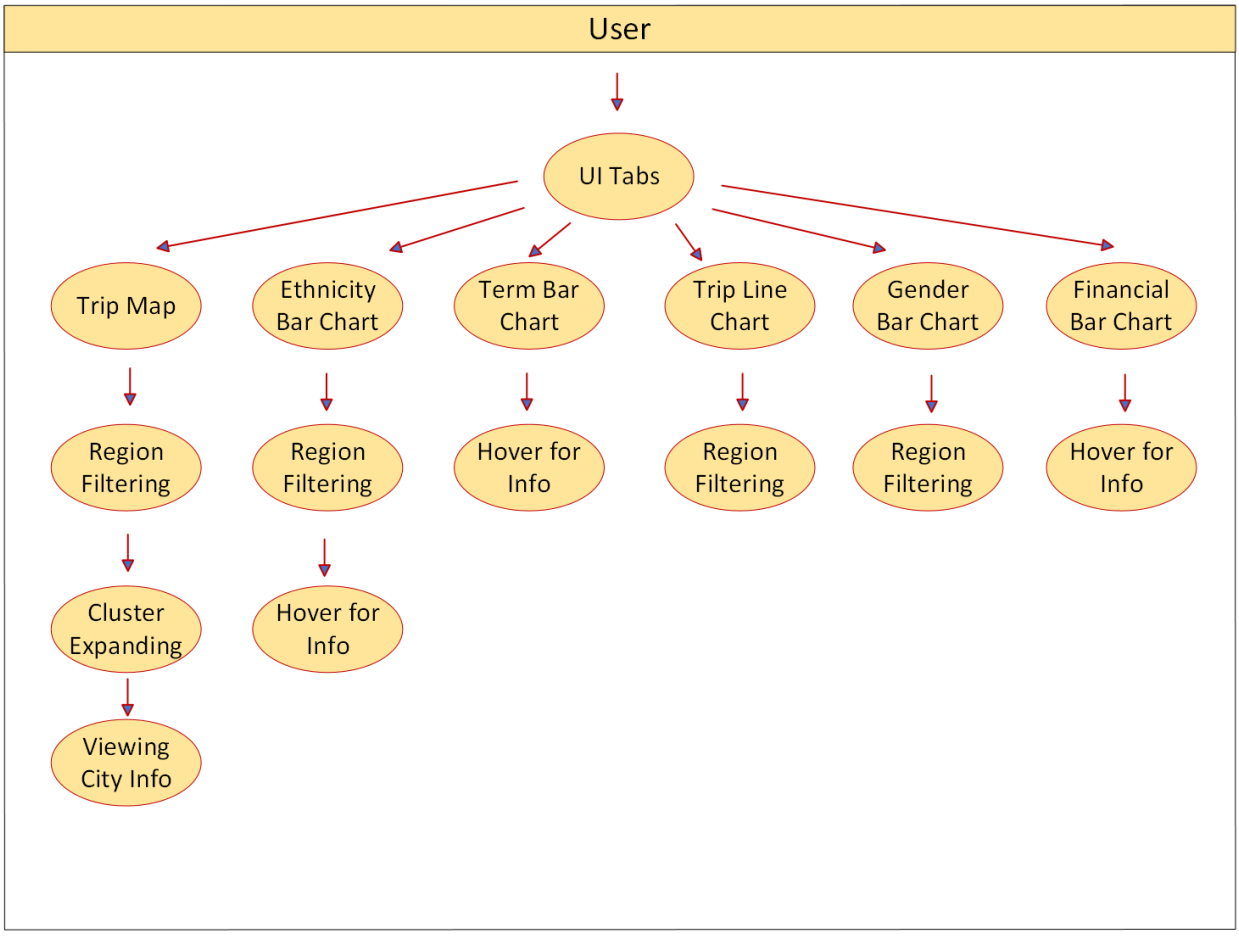
How: The bar chart employs filtering to reduce the number of regions shown, preventing information overload for the user. It uses the line mark with 'vertical spatial position' channel to display the varying proportions of genders, 'horizontal spatial position' to distinguish between countries, and 'hue' to differentiate between genders. This setup allows users to easily visualize and compare the gender makeup of students in different study abroad programs.

UML Diagrams

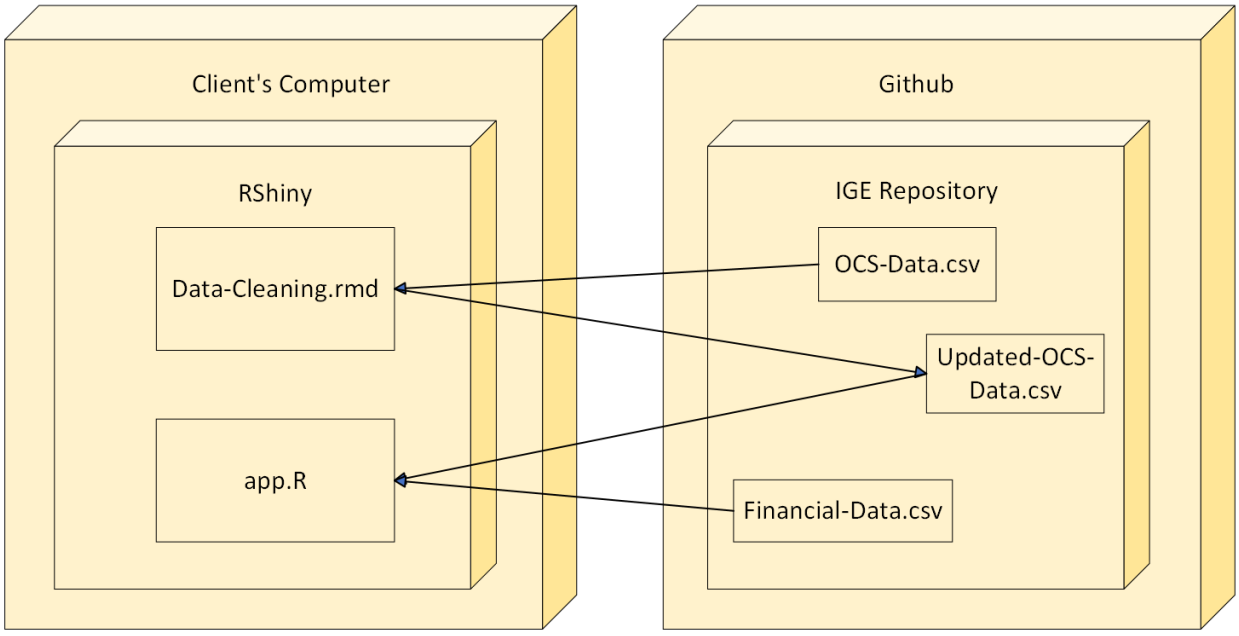
Use Case Diagram



Activity Diagram



Deployment Diagram

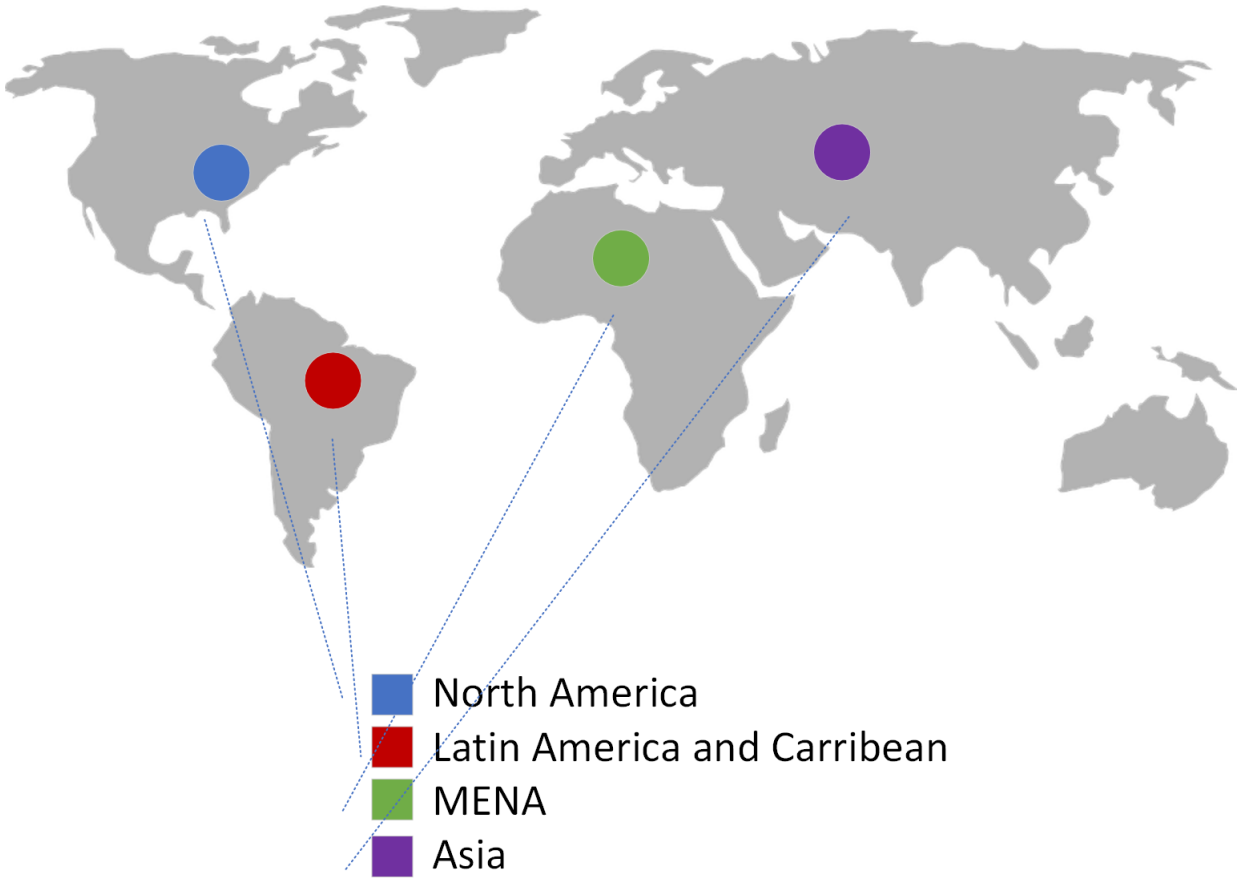


Design Patterns

Dynamic Queries

"Display environment: A display with symbols representing entities drawn from a set of multidimensional discrete data, with a set of controls that restricts the range displayed on each of the data dimensions."

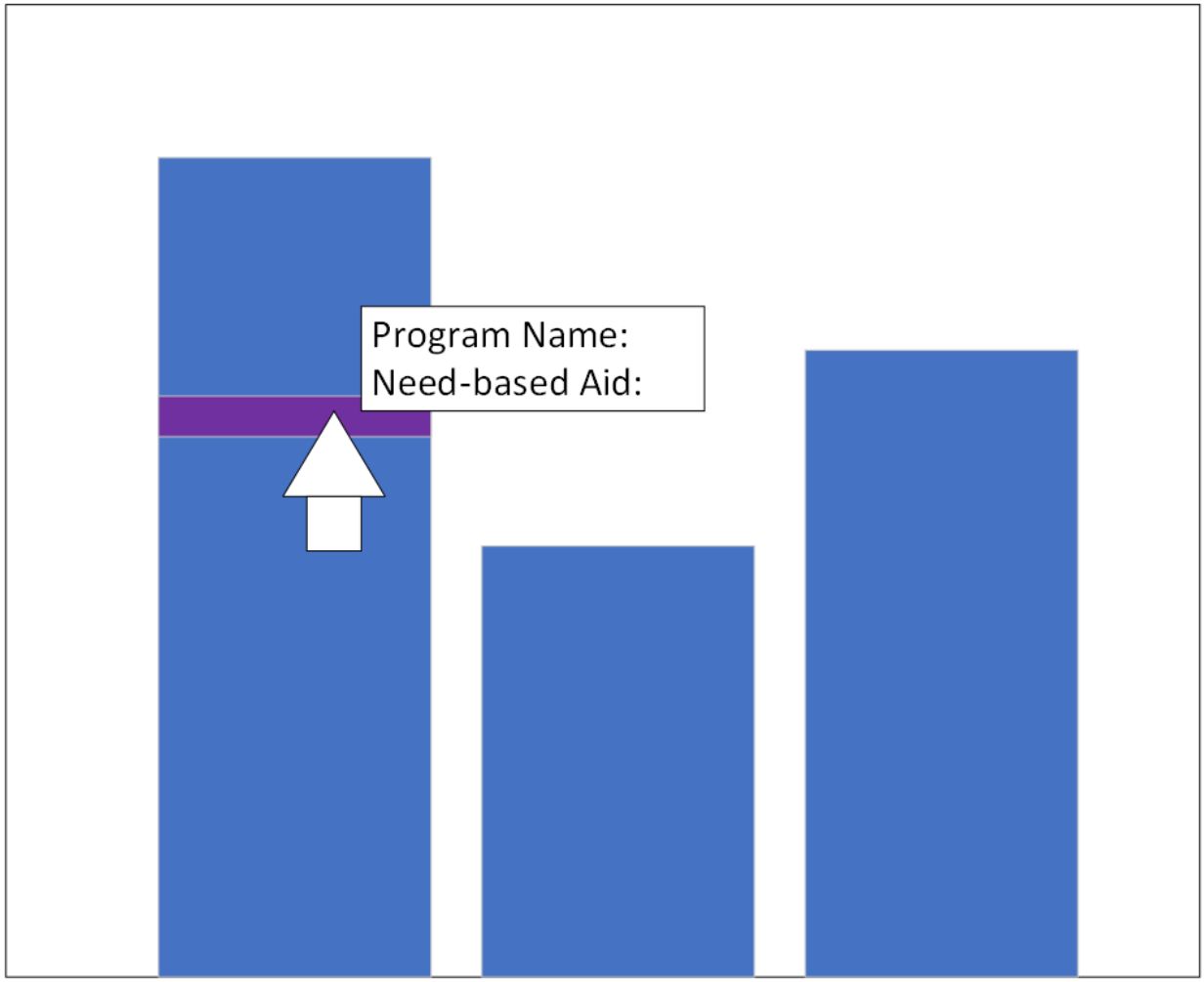
We draw the city locations of all the rows in the dataset to display the off-campus programs being offered at that specific city. The multidimensional space is a map visualization with a slider input where the user can filter out data points by region.



Drill Down

"Here a user clicks on a symbol or visualization to get more information about the visualization."

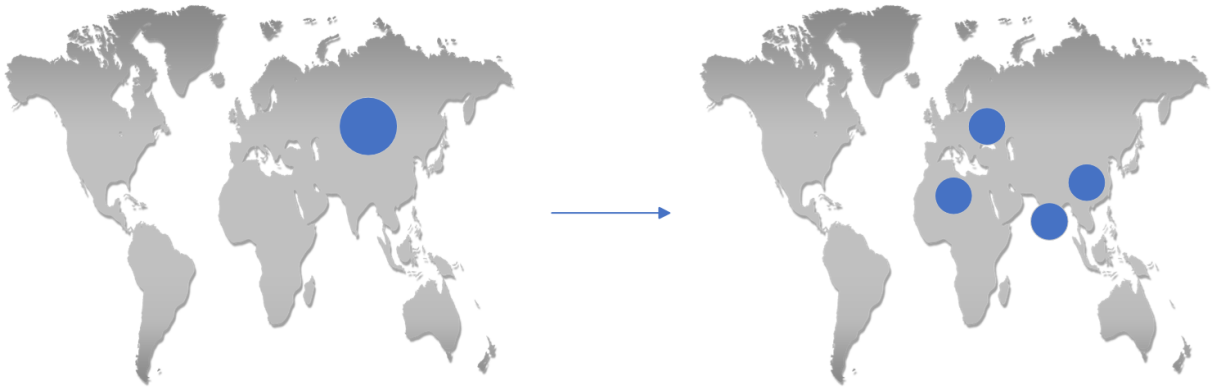
We create a financial information segmented bar chart where the user can hover over a segment of a bar (x-axis: Difference in cost of doing Off-Campus Study vs. Semester in Grinnell, y-axis: Number of Programs) to get extra information that shows program name and need-based aid available when you hover your mouse on it.



Seed then Grow

"Often a data analyst starts with a particular seed of information and then begins to gather related information. Each additional information nugget may lead to further expansion of an information tree or network. This has been called"Start with what you know, then grow"

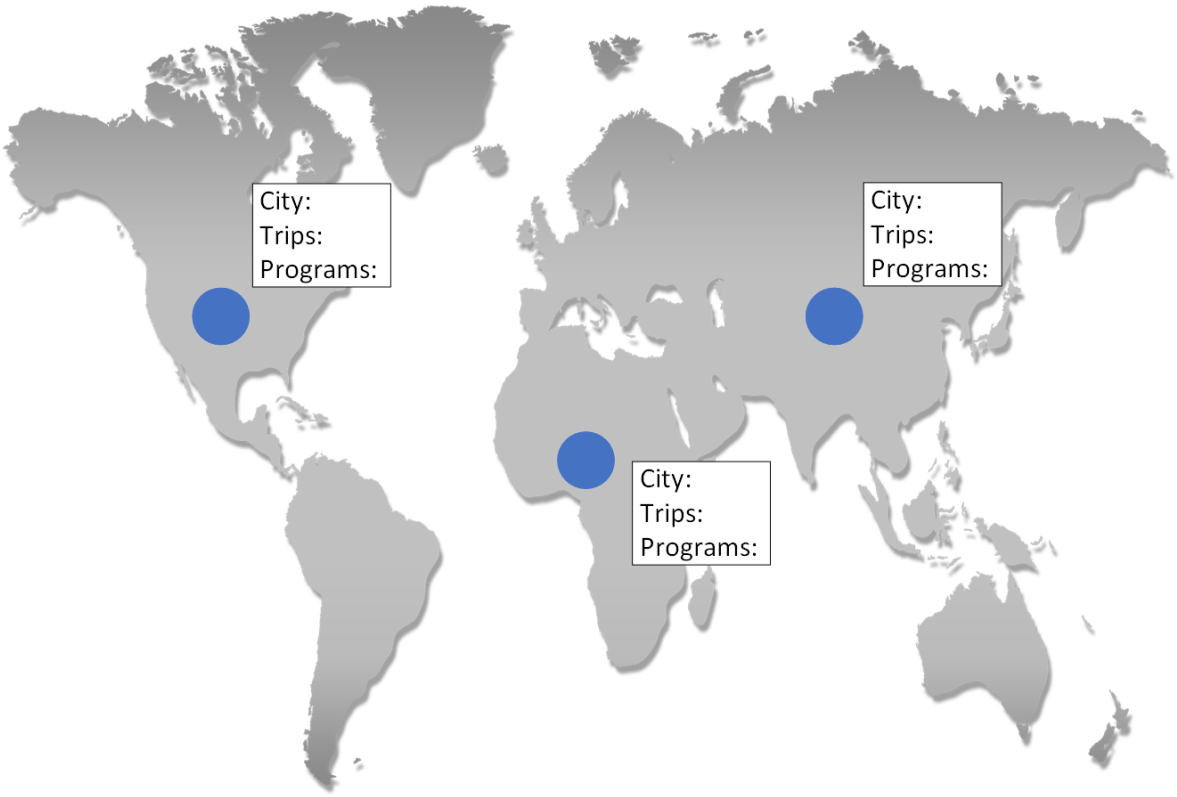
Our map visualization first displays cities where programs are located. As users continue to click on the city markers, more programs/program info is displayed in detail.



Visual Query

"Solving problem components by identifying patterns through a visual search."

For our map visualization, the user may be curious about how many and what programs are in each country, and the cluster map allows the user to look at different countries to find this information.



Coding Standards

Formatting

1. Ensure everything is properly and consistently indented (CTRL + I)

-Intent: To ensure easy readability

2. Keep each line less than 80 characters

-Intent: Maintain readability and efficiency

-Example: If there are super long lines of code, people will have to scroll.

3. No redundant code

-Intent: Keep code clean and concise

-Example: Do not read data file more than once.

4. General comments should be identified with # followed by a space

-Intent: To understand what complicated code is doing

-Example: # filter data frame by country

5. Inline comments need 2 spaces before the #

-Intent: Maintain structure and normal style

-Example: “printf()#This prints” vs. “printf() #This prints”

6. Variable names and file names should have the format: dataFile

-Intent: Maintain code context and readability

-Example: finData

7. Use meaningful and descriptive variable names

-Intent: Ensuring readability and clearness

-Example: countryCounts <- dataFile %>% group_by(ProgramRegion) %>% summarise(TripCount = n(), ProgramNames = list(ProgramName), .groups = “drop”)

8. For variable assignments, use <– not =

9. All libraries in same code block, not dispersed throughout the code

-Intent: Make libraries apparent and don’t repeat lines

10. Unique block code for reading data files

-Intent: Easily accessible data for editing

-Example:
dataFile <- read.csv(“df.csv”) dataFile$ProgramCity <- -gsub(”, . * ”, ””, dataFile$ProgramCity)

Documentation

1. Explain the use of code that produce a general functionality in the project

-Intent: To make the code easy to understand for functions

-Example: A comment that says that a function is making a side bar layout

2. Don't repeat explanations for similar lines of code, but always make clear what lines of code are for/what they're doing

-Intent: Prevent redundant comments

-Example: If you use the same filtering methods multiple times, don't re-explain the methods, but explain what you're filtering for each time.

References

[1] C. Beeley, Web Application Development with R Using Shiny, 2nd ed. Birmingham, UK: Packt Publishing, 2016.

[2] T. Munzer, Visualization Analysis & Design, New York, NY, USA: CRC Press, 2015.

[3] J. Cheng, B. Schloerke, B. Karambelkar, and Y. Xie. "Add markers to leaflet." Rstudio.github.io. Accessed: April 17, 2024. [Online.] Available: https://rstudio.github.io/leaflet/articles/markers.html

[4] Natural Earth, 2009-2024, "Admin 0- Countries," Source. [Online]. Available: https://www.naturalearthdata.com/downloads/50m-cultural-vectors/50m-admin-0-countries-2/

[5] J. F. Dooley, Software Development, Design and Coding, 2nd ed. New York, NY, USA: Apress, 2017.

[6] F. M. Fowler, Navigating Hybrid Scrum Environments, New York, NY, USA: Apress, 2019.

[7] S. Sundaramoorthy, UML Diagramming, 1st ed. Boca Raton, FL, USA: CRC Press, 2022.

[8] C. Ware. "Visual Thinking Design Patterns for Data Analysis Tools". Ccom.unh.edu. Accessed: April 15, 2024. [Online.] Available: https://ccom.unh.edu/vislab/VTDP_web_pages/VTDP_HomePage.html