Outdoor Equity App Technical Documentation

Clarissa Boyajian and Halina Do-Linh

2022-06-03

Contents

Chapter 1

About

1.1 Abstract

Outdoor recreation and access to nature have well-documented positive impacts on mental and physical well-being. Federal public land management agencies in the United States offer a variety of outdoor recreation activities to visitors. However, people from different socioeconomic and identity groups access federal public lands unequally due to historical discrimination and current inequities. This project uses data from the Recreation Information Database (RIDB) and the United States Census Bureau (US Census) to explore patterns of visitor use of reservable overnight sites (such as campgrounds, cabins, hike-in, and more). Specifically, we used 2018 reservation data and US Census data from the next available year to 2018 (i.e. 2018 median income data, 2015 language data). We created the interactive Outdoor Equity App that gives users tools to summarize data, explore relationships between RIDB and US Census variables, view maps of where visitors are coming from for reservable sites in California, and download subset data. This technical documentation includes information on metadata, application maintenance, and next steps for expanding the app to include visitor data from more locations and time periods.

1.2 About the Authors

This technical documentation for the Outdoor Equity App was created by Clarissa Boyajian and Halina Do-Linh. The app was created as the final capstone project for their Master of Environmental Data Science degrees from the University of California's Bren School of Environmental Science & Management. Both women are passionate about environmental justice, open science, the art of data visualizations, and spending time recreating outdoors. Please reach out to either of both of us with any questions.

This project could not have been completed without the support and guidance of the Bren School advisors Dr. Frank Davis and Dr. Allison Horst and our external advisors Dr. Kaitlyn Gaynor and Dr. Will Rice.

1.3 Helpful Links and Resources

The Outdoor Equity App is was created with the `shiny` package [Chang et al., 2021a] using RStudio version 1.4.1717-3. This technical documentation is hosted using GitHub Pages. The GitHub repository containing all code relating to this technical documentation can be found here and the GitHub repository containing all code relating to Outdoor Equity App can be found here.

Chapter 2

Executive Summary

Outdoor recreation and access to nature have well-documented positive impacts on mental physical well-being. Federal public land management agencies in the United States offer a wide variety of activities to visitors. However, people from different socioeconomic and identity groups access federal public lands unequally due to historical discrimination and current inequities. The multi-agency program, Recreation One Stop (R1S), oversees the operations of Recreation.gov and aims to increase access to recreation by providing online resources about nationwide recreational opportunities, allowing visitors to make reservations, and making the associated data accessible to all. The rich data on visitors that R1S collects presents an opportunity for the creation of more robust data-driven analytical tools to understand the patterns and correlations of this unequal access across the country and within individual recreation areas. Decision-makers can use these tools to explore and visualize how recreational opportunities on federal public lands are accessed.

Our overarching objective is to design and built an interactive web application that allows users to analyze patterns in the access and demand of visitors at reservable overnight sites (such as campgrounds, cabins, hike-in, and more), using data from the Recreation Information Database (RIDB) and the United States Census Bureau (US Census). These analyses will allow federal public land managers to explore relationships among attributes of recreation opportunities, reservation practices, and socioeconomic data from the regions of visitor origin. We achieved this goal through the creation of an interactive web application, the Outdoor Equity App, that allows for a wide range of visualization, metadata documentation, and subset data downloads. This technical documentation serves to document the Outdoor Equity App creation process, include information for ongoing maintenance, and provide suggestions for future use and expansion.

The app - which is implemented using the R programming language - accesses public RIDB and US Census data via direct download and application programming interfaces (API). All data and R code scripts are stored on the UCSB Taylor Server and version-controlled through GitHub. We isolated necessary variables and defined, standardized, and aggregated values in the data cleaning process. We calculated additional derived variables for each reservation, such as distance traveled and booking window, and summary statistics (e.g., mean and median) for census data at the ZIP code level. A data set that combines the US Census and RIDB data based on visitors' home ZIP code is the foundation for the Outdoor Equity App. We visualized distributions of variables and relationships between them with simple, straightforward figures. Within the app, users can subset the data to a specific overnight reservable site and visualize the distribution of a single variable, the relationship between two variables, or the visitorshed map (i.e. area from where visitors are coming) for the selected site. The app currently only includes data for California reservable sites in fiscal year 2018 due to project scope limitations.

Throughout the analysis and app creation processes, external advisors and federal public land managers have reviewed and tested the Outdoor Equity App. We incorporated feedback into all parts of the processes to ensure our data, analysis, and final products are robust. Potential future updates to the Outdoor Equity App are discussed in this technical documentation and include temporal and spatial expansions and app maintenance. The temporal expansion would include cleaning additional datasets for years from 2012 to 2021 as well as expanding the app's interface to allow for temporal selections when subsetting data. The spatial expansion would focus on updating the app structure and server hosting capabilities so the app runs smoothly with data from the full United States.

As environmental justice is increasingly recognized as a necessary lens to achieve environmental goals, equitable access to outdoor recreation is a high priority for managers. This tool assists managers to be equity-conscious decision-makers, can be a springboard for researchers who have questions about outdoor recreation, and strengthens nonprofit organizations' advocacy efforts. We also hope it will be a dynamic tool that empowers visitors to access the information and resources they need to explore outdoor recreation.

Chapter 3

Problem Statement

3.1 Background

Outdoor recreation provides critical health and well-being benefits to communities, and in the United States, federal public lands play an important role in providing access to nature. However, access is not equal for all people [Ewert and Hollenhorst, 1990]; [Flores et al., 2018], which has been recognized as an environmental injustice [Floyd and Johnson, 2002]. Many studies have shown that federally managed public land is accessed unequally due to historical discrimination and current inequities [Floyd and Johnson, 2002]; [Shelby et al., 1989]; [Xiao et al., 2021].

Many land management agencies in the U.S. are tasked with the dual mandate of providing recreational opportunities for visitors while also preserving and conserving natural resources and places [Shartaj and Suter, 2020]. For over a century, striking the balance necessary to uphold this mandate has proven a challenge for federal agencies like the National Parks Service [Meinecke, 1937]; [Sax, 1980], and the recent growth of recreation (Figure 3.1) has renewed concerns about its potential negative environmental impacts and changes to the visitor experience [Hammitt et al., 2015]; [Timmons, 2019]. The challenge now facing public land management agencies is how to allocate quality visitor experiences to a more diverse user base. Simply increasing recreation opportunities on public land is not a viable solution to this rising demand.

While managers seek to allocate existing resources (e.g. campsites) through the fairest means possible, including reservation systems, equal opportunities do not translate to equitable access [Shelby et al., 1989]. Historically, policies of segregation barred certain racial groups from using federal public lands and the legacy of these policies has perpetuated inequitable access for certain racial groups to this day [Xiao et al., 2021]. Additionally, previous and current inequities like lack of time, disposable income, access to technology, and lack of social or institutional knowledge about reservation systems impact access to federal public lands [Scott and Lee, 2018]. At present, park visitation and camping are seeing a surge in popularity, heightened even more by the COVID-19 pandemic, and this rapid increase in demand for recreation opportunities may only further these inequities.

3.2 Significance

Currently, much of our understanding about trends in recreation on public lands comes from the Integrated Resource Management Applications (IRMA) Portal, which the National Parks Service uses to monitor visitor counts over time [Bergstrom et al., 2020]. However, these data lack information on where visitors are coming from. This project leverages the Recreation Information Database (RIDB), managed by Recreation One Stop, an inter-agency partnership that provides reservation services and trip-planning tools on Recreation.gov. The RIDB is far more robust, including data from other land management agencies, and information on visitor zip codes, costs, group sizes, and dates of both reservations and recreation activities. While it is available for public download, there are few robust data-driven analytical tools to understand the patterns and relationships of these inequities within individual recreation areas.

Previous research has demonstrated the value of RIDB data in forecasting future recreation demand for single park units [Rice et al., 2019] and analyzing preferential characteristics for popular recreational facilities [Rice and Park, 2021]. A recent study summarizing RIDB data from national parks [Walls et al., 2018] also identified broad patterns in reservations. For example, campsite reservations are made far in advance, but many are canceled last minute (Figure 3.2); visitors tend to visit national parks near their homes (Figure 3.3); and the distribution of incomes of campers appears to be similar to the U.S. population as a whole (Figure 3.4). However, overall, the vast RIDB data has received limited system-level research attention to date, and this work will be the first to explore issues of equity with RIDB data.

Furthermore, much of the existing research on outdoor recreation focuses on National Park Service lands, such as Walls et al. [2018], which is only a small percentage of all federal land used by the public. The other land management agencies, including US Forest Service, Bureau of Land Management, and Army Corps of Engineers, often lack the capacity and funding to process reservation data, and are less frequently the subjects of outside research. Little is known about how patterns of access and demand vary across land management types. The RIDB includes data from all federal land management agencies, and therefore has tremendous promise to inform our understanding of patterns and trends in recreation across space and time and to inform policies for more equitable campground access for all federal public lands.

Our overarching objective for this project is to utilize data from RIDB and US Census to analyze spatial and demand patterns of visitor access at reservable overnight sites (such as campgrounds, cabins, hike-in, and more). We chose to focus on reservable sites since recent studies have shown this type of outdoor recreation to be a good proxy for visitation to federal public lands [Walls et al., 2018]. These analyses will provide federal public managers an opportunity to explore relationships between and within socioeconomic and reservation variables.

3.3 Figures

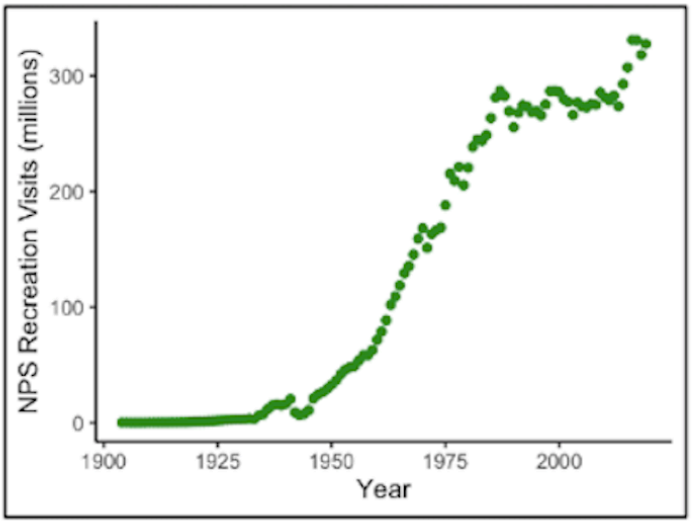

Figure 3.1: Total annual visitors to the National Park Service system, since its inception through 2020. Visitation has been rapidly increasing, particularly within the last decade. (Source: [IRMA](https://irma.nps.gov/Portal/))

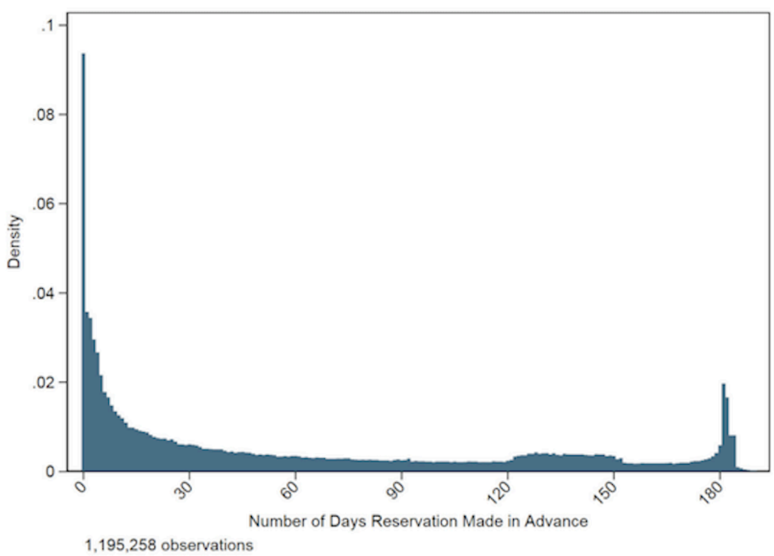

Figure 3.2: Reproduced from @Walls2018. Days in advance that National Park campsite reservations are made from 2014 to 2016. Reservations are made far in advance, but many reservations are canceled at the last minute.

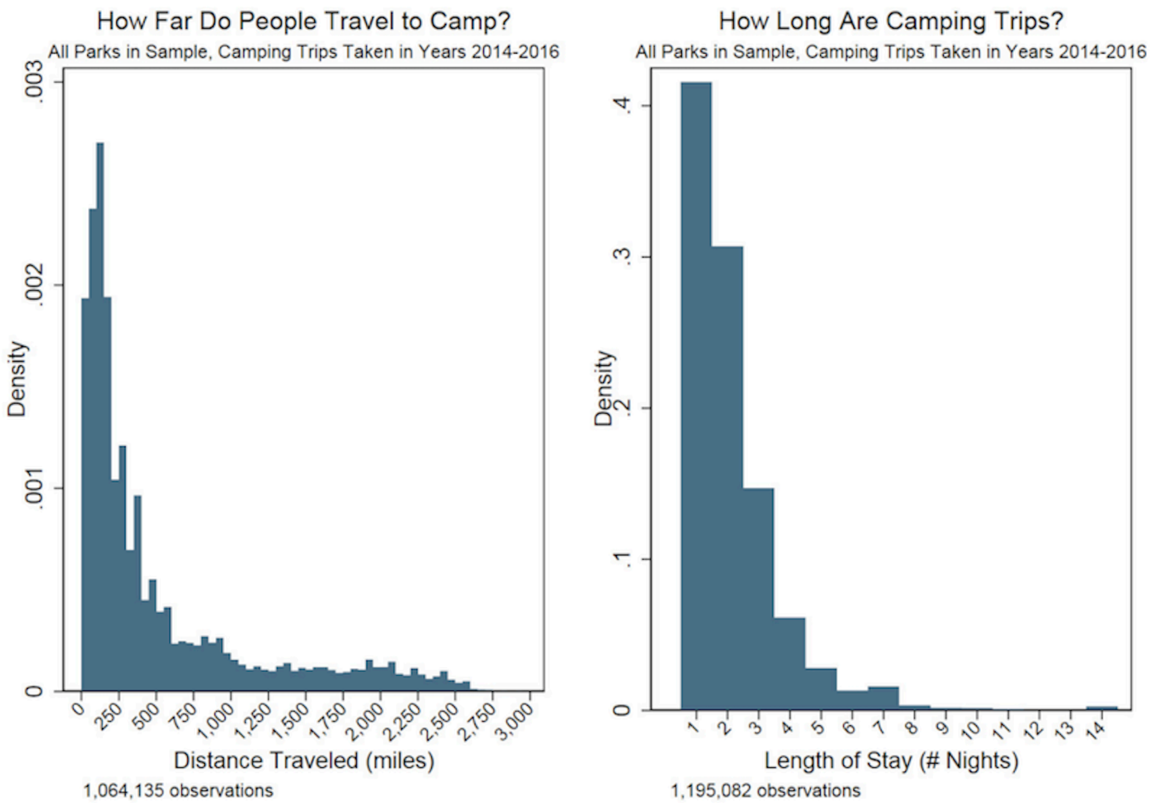

Figure 3.3: Reproduced from @Walls2018. Distance traveled and duration of stay for National Park camping visits from 2014 to 2016. Visitors tend to visit national parks near their homes and stay only two nights, and longer trips are rare.

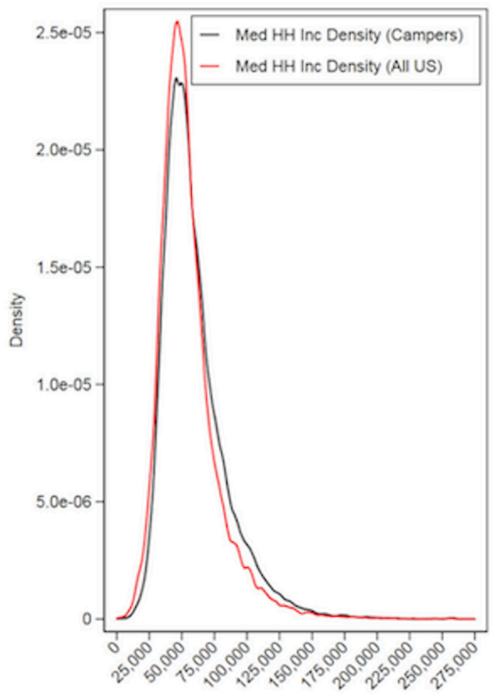

Figure 3.4: Reproduced from @Walls2018. Median household income (by zip code) for National Park campers and for US Population. The black line estimates the distribution of median household (HH) income of campers from 2014 to 2016. The red line estimates the distribution of median household for all zip codes in the U.S. using average median household income from 2014 to 2016 where each zip code is an observation.

Chapter 4

Specific Objectives

Federal lands in the United States provide important recreation opportunities to the public, but there is a **growing need to understand and mitigate inequities in access to outdoor recreation.** This project addressed this need by creating the Outdoor Equity App, an **interactive platform** for summarizing and visualizing site-specific patterns and trends in visitation volume, demand, and visitors' location of origin. The platform integrates nationwide Recreation.gov reservation data with US census data to:

- Gain insights into **demand for reservations** across different types of recreation areas.
- Analyze access to federal public lands among **historically underserved groups** in relation to recreation **site type, cost, location, and demand**.
- Clearly define all variables and values in the **metadata documentation**.
- Allow users to **download a subset of the combined data** for further analysis.

Chapter 5

Summary of Solution Design

5.1 Glossary and Definitions

Throughout this document we define "reservable sites" as traditional campgrounds, single remote campsites, overnight boat-in sites or mooring, equestrian sites, cabins, and other shelters listed in the RIDB data.

A full glossary can be found in the Appendix Section.

5.2 Access, Clean, and Wrangle Data

RIDB data and US Census American Communities Survey (ACS) data are freely available online to the public. We accessed, cleaned, and wrangled all data outside of the Outdoor Equity App using the `data_wrangle_and_clean.Rmd` document and 18 custom-made functions (Appendix Table 10.2). We downloaded RIDB data in CSV format from Recreation.gov and ACS data through API using the R package `tidycensus` [Walker and Herman, 2021]. We first subsetted RIDB and ACS datasets to include only the variables relevant to our objectives. We then normalized, aggregated, and calculated variables as necessary. Once both datasets are cleaned and wrangled, we joined them using ZIP codes as the key (common value in both datasets). Finally we wrangled the joined RIDB and ACS dataset to ready them for creating data relationship plots.

See the Appendix Section for a full table of all functions.

5.3 Analysis and Visualizations

The Outdoor Equity App features interactive maps and plots. Users of this app can select a single reservable site to create custom plots that show a data summary of a single variable or a data relationship between two variables. Visualizations of multiple reservable sites appear as separate plots. Users can also select a single site to create a visitorshed map for the full United States and for the state in which the site is located.

5.4 Outdoor Equity App

The app has a navigation bar with four tabs: About, Analysis, Metadata and Data Download. Nested under the Analysis tab are the subtabs of Data Summary, Data Relationship, and Visitorshed Maps. The app opens automatically to the About tab.

Figure 5.1: Screenshot of the About page of the Outdoor Equity App

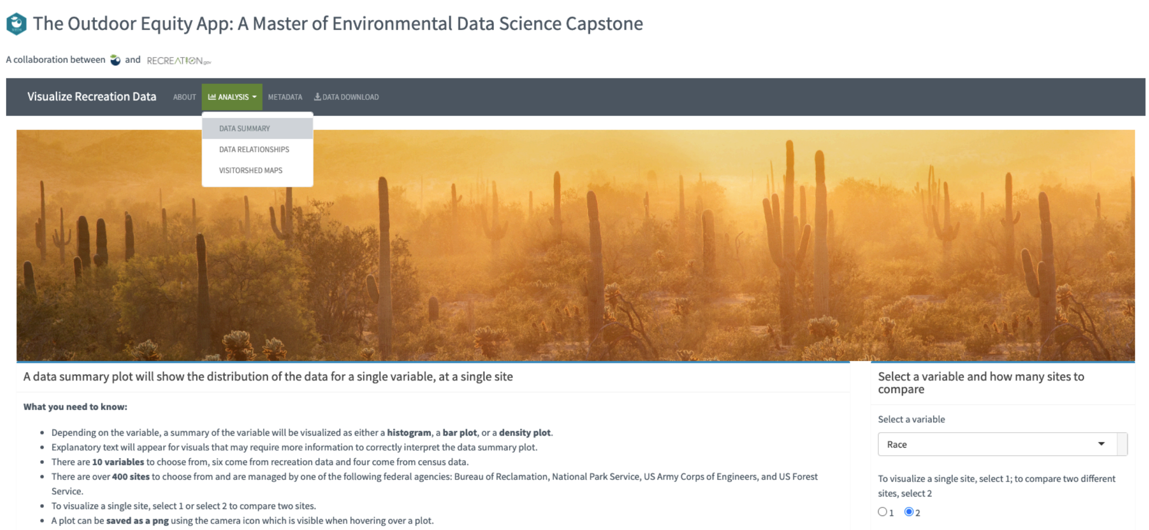

Figure 5.2: Screenshot of the Analysis page of the Outdoor Equity App

5.5 User Guide and Technical Documentation

The Outdoor Equity App includes a user guide and metadata information. The user guide section includes a quick overview of the app and helper text on how to start creating visuals.

This technical documentation is created with the `bookdown` package [Xie, 2021] and is linked in the About tab of the Outdoor Equity App. Metadata for all variables used within the app are also available on the app Metadata tab and in the Products and Deliverables Section of this document.

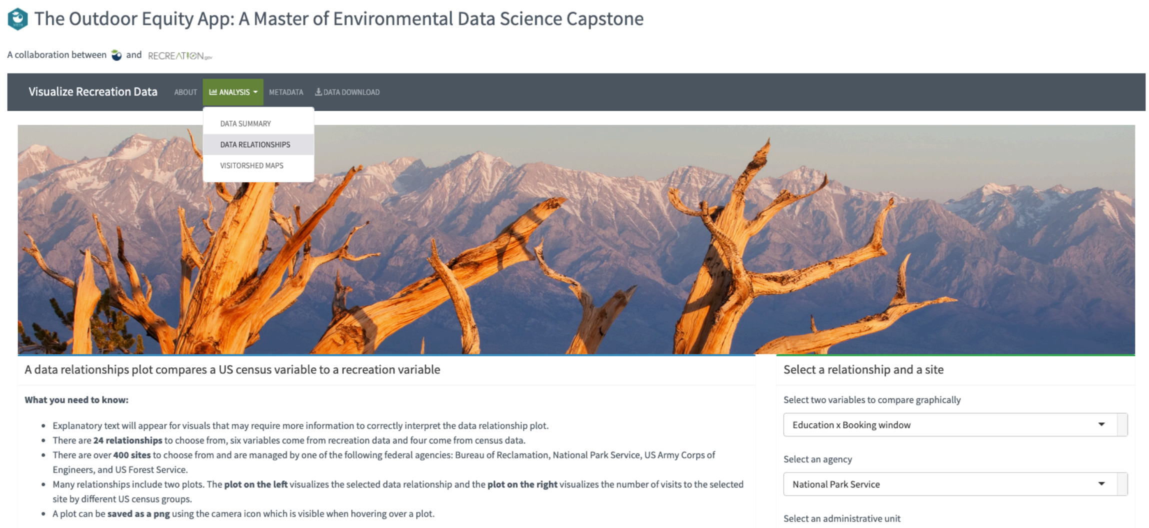

Figure 5.3: Screenshot of the Analysis page of the Outdoor Equity App

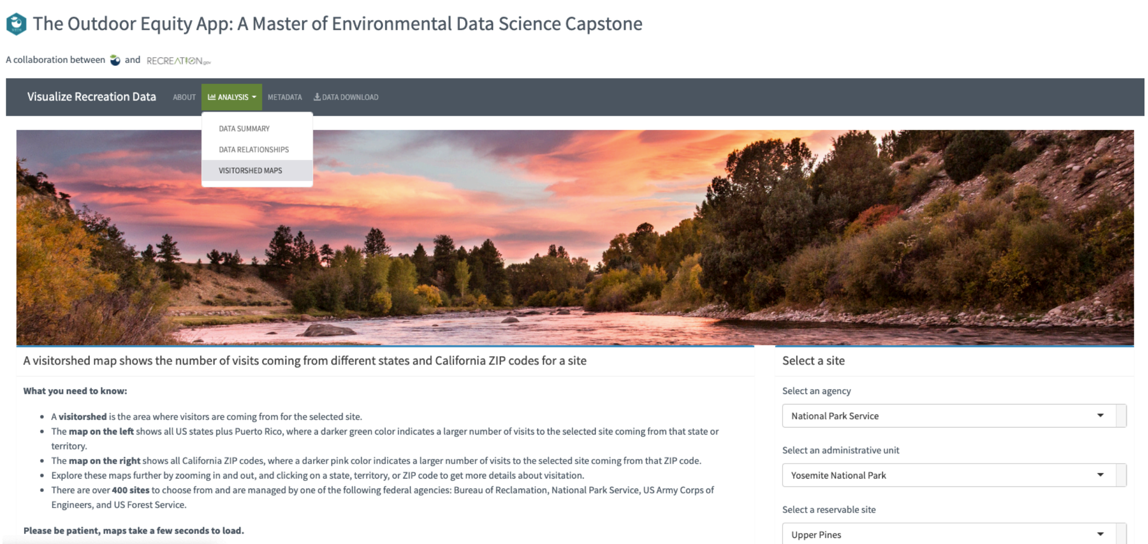

Figure 5.4: Screenshot of the Analysis page of the Outdoor Equity App

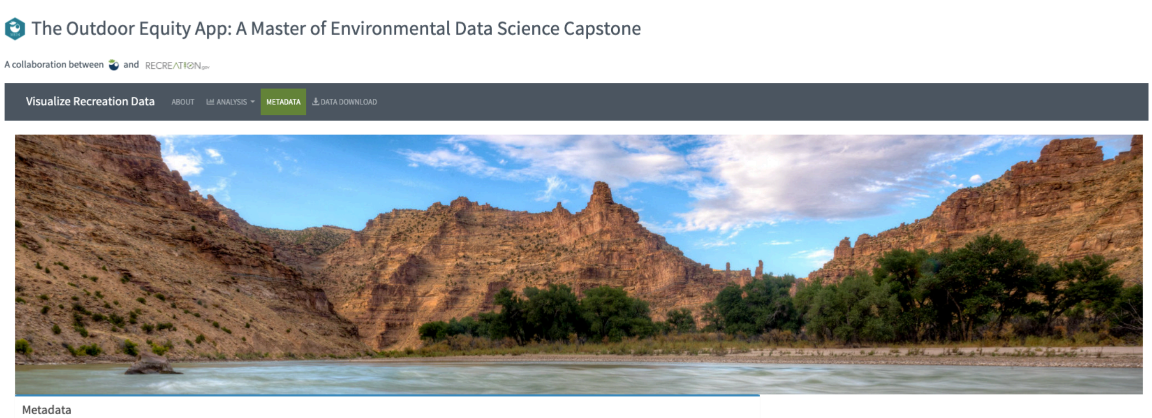

Figure 5.5: Screenshot of the Metadata page of the Outdoor Equity App

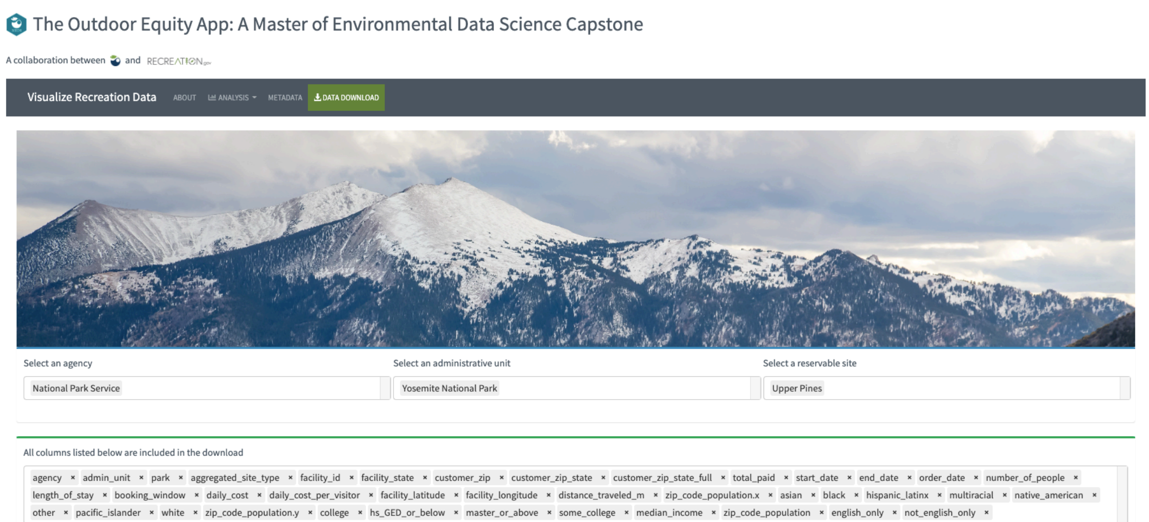

Figure 5.6: Screenshot of the Metadata page of the Outdoor Equity App

Chapter 6

Products and Deliverables

6.1 R Shiny App

The Outdoor Equity App was built using the `shiny` package [Chang et al., 2021a] and has the following functionality:

- Visualize statistical distributions of RIDB and US Census American Community Survey (ACS) variables
- Visualize relationships between RIDB and ACS variables
- Download visualizations as PNG images
- View visitorshed maps of reservable sites both nationally and within the state where the campsite is located
- Download customized subsets of the data

6.2 Metadata

A condensed list of all metadata can be found in the Appendix Section. A full list of all metadata can be found in the Outdoor Equity App on the Metadata tab.

6.3 Packages and Versions Used

See the Appendix Section for a full list of packages used.

Chapter 7

Summary of Testing

7.1 Data Integrity

We screened the data for outliers by summarizing and visualizing the raw data, and assessed whether those outliers need to be removed. We consulted other researchers who are familiar with the RIDB datasets to confirm outliers or other anomalies in the data.

Additionally, we documented the percent loss from data wrangling to ensure that our cleaning and wrangling of the data were reasonable.

7.2 Code Review

We conducted code reviews within the team, and with faculty or external advisers. We reviewed specific code chunks and scripts related to the Outdoor Equity App.

We separated our workflows so that one person created scripts, and the other reviewed them. We did this to maintain some objectivity when evaluating if our datasets were aggregating correctly. We also had a separate workflow for metadata, where one person created and wrote metadata, and the other reviewed it. This confirmed that the data matches how it is being described in the metadata. This confirmation is important as we want our client to be able to scale our product and workflows for future use.

7.3 Product Testing

We used three packages to test our R Shiny app. We used `shinytest` [Chang et al., 2021b] to ensure our app is visualized the way we expect it to using the package's snapshot-based testing strategy. We used `shinyloadtest` [Schloerke et al., 2021] to test the server hosting the R Shiny App to ensure that it responds in a reasonable amount of time based on the inputs a user provides. Similarly, we utilized the `tictoc` [Izrailev, 2021] package during our data wrangling and cleaning, and when we initially created our plots, graphs, and maps to estimate an informed guess of how long it may take the app to run our scripts. Lastly, we used the `reactlog` [Schloerke, 2020] package's diagnostic tool which creates a reactive visualizer for the app to make sure that reactive elements are working the way we expected them to. It is important to note that this diagnostic tool was not useful as our app functionality increased, as the reactive visualizer became impossible to read. There may be other options within `reactlog` to use the reactive visualizer in a different way, but we did not have enough time to research this.

We added temporary print statements to all functions in the app to ensure that the functions were working correctly were are outputting what we expect. We removed print statements from functions that were functioning with zero errors. We did this because print statements can take a long time to run and should not be left in functions or in the app permanently.

Additionally, we held multiple meetings with the Recreation One Stop team to obtain real-time and focused feedback to improve user design and experience.

7.3.1 Next steps for testing

Due to time constraints, we were not able to implement all testing methods we wanted. We recommend the following testing strategies to make the app more robust and for smoother functionality.

- Use the R package `testthat` [Wickham, 2022] to conduct unit tests on the scripts used to create Tidy datasets and for subsetted datasets for visualization. This type of testing may be important to avoid silent failures and to ensure that the datasets are aggregating correctly.
- Use `gremlin.js`), a JavaScript library used for "Monkey testing" to test the behavior of the R Shiny App. This package is compatible with `shiny` [Chang et al., 2021a] and does not require any external installation. See Chapter 11 in Engineering Production-Grade Shiny Apps for more guidance. "Monkey testing" is a type of testing where random, automated tests provide random inputs and then checks the behavior of the app (i.e. if the system or application crashes). We were able to find some finicky bugs through our own testing of random inputs, but "Monkey testing" is the formal process of testing app behavior.
- Employ user testing with federal public land managers, researchers, and those who are not familiar with RIDB data to further enhance the user experience and design.

Chapter 8

User Documentation

8.1 Purpose of the Outdoor Equity App

The Outdoor Equity App provides users with tools to explore trends at different overnight reservable sites to analyze access to these sites. The intended audience is federal public land managers and researchers as well as nonprofit organizations and recreation users.

The Recreation Information Database (RIDB) data are comprehensive when it comes to information regarding the site and reservation, but do not include information about the visitor outside of their home ZIP code. US Census American Community Survey (ACS) data is used to approximate socioeconomic demographics by joining information about the visitors' home ZIP code to the RIDB data.

8.2 How to Use the Outdoor Equity App

8.2.1 About the App

The About tab of the Outdoor Equity App includes background information about what the App is, why outdoor recreation is important, the App creators, and what data is used in the App. These sections are similar to the About Section of this technical documentation. The About tab also includes example questions that a user might explore through the different parts of the Analysis tab.

The Analysis tab of the App consists of three parts Data Summary, Data Relationship and Visitorshed Maps. Each of these pages includes a brief explanatory section of how to interpret the plot or graph, a section to subset the data to the desired campsite, and the plot or map outputs.

The Metadata tab of the App includes metadata for all variables in the combined RIDB and ACS dataset. This section mirrors the Metadata section in the Products and Deliverables Section of this document.

The Data Download section of the App allows a user to download a subset of data include as many or as few campsites and variables as they require for further analysis or use.

8.3 How to Maintain the Outdoor Equity App

8.3.1 Repository Directory Structure

Our code is version controlled in a GitHub repository. This includes the data cleaning, wrangling, and analysis, creation of plots and maps, and structure of the shiny app. An overview of the directory structure can be found in the Appendix Section.

8.3.2 Data Preparation Methods

8.3.2.1 RIDB Data

RIDB data data are available through direct download as CSV files or via the API as JSON files via Recreation.gov. API access requires creating a Recreation.gov account and requesting second tier API access via the Recreation.gov website's Contact Us page. CSV files are readily available for download via the Recreation.gov website. Data are collected each time a visitor makes a reservation through Recreation.gov. Data packages are posted annually in the spring by Recreation One Stop (R1S) and contain the previous fiscal year's reservations (ex: the 2018 package includes 2018-10-01 through 2018-09-30). Data packages are available for download from 2006 to present. Each annual data package file contains a range between 2 million and 5 million observations (or reservations) and includes variables in character, numeric, and date/time formats about each reservation. A shift in the data collection and storage processes occurred in 2019, changing what variables are available and how they are labeled. Currently, the Outdoor Equity App contains only data for reservable sites in California and reservations for fiscal year 2018 due to time limitations of this project. To see more about expanding the App temporally and spatially see the How to Expand the Outdoor Equity App Section.

We created a reproducible workflow for cleaning and wrangling data, which is employed in the `data_wrangle_and_clean.Rmd` document that sources custom functions to prepare RIDB data for joining with ACS data. All custom functions are listed in the Access, Clean, and Wrangle Data Section of this document. These functions rely heavily on functions from the `tidyverse` collection of packages [Wickham, 2021].

First, we use the `function_ridb_subset-pre2018.R` to subset the RIDB data, filtering only reservations within the selected state that are listed as "Overnight" reservations within the `use_type` variable. For the 2018 California dataset this results in a starting "raw" data frame with 521,682 reservations

```
# filter for state
filter(facility_state == state_abbrev) %>%
  # filter for use type
  filter(use_type == "Overnight")
```

We then select only the necessary variables, including information about the site (agency, park or forest name, site name, site type, and site location) and information about the reservation (home ZIP come, total paid, visit start and end dates, visit order date, and number of people in party).

```
# select variables
select(c("agency", "parent_location", "region_description",
         "park", "site_type", "facility_id", "facility_state",
         "facility_longitude", "facility_latitude",
         "customer_zip", "total_paid", "start_date",
         "end_date", "order_date", "number_of_people")) %>%
  mutate(site_type = tolower(site_type)) %>%
  filter(!site_type %in% c("historic tour", "hiking zone",
                           "group picnic area", "cave tour",
                           "management", "picnic",
                           "entry point", "trailhead"))
```

Next we normalize the customer ZIP code values. This includes filtering for only US ZIP codes and shortening all 9 digit ZIP codes to include only the first 5 digits.

```
# filter out invalid ZIP codes
filter(str_detect(string = customer_zip,
                  pattern = "^[:digit:]{5}(?!.)") |
         str_detect(string = customer_zip,
                    pattern = "^[:digit:]{5}(?=-)")) %>%
  filter(!customer_zip %in% c("00000", "99999")) %>%
  mutate(customer_zip = str_extract(string = customer_zip,
                                    pattern = "[:digit:]{5}"))
```

This function results in the removal of 24,866 reservations (or 4.77%) from the original "raw" dataset that included all reservations for reservable overnight campsites in California.

We created a second custom function `function_ridb_variable_calculate-pre2018.R` to calculate and manipulate variables of interest. We use start, end, and order dates to calculate the lengths of stay and booking windows (number of days from order to start date) of each reservation. The booking window calculations return a number of results that are negative. This is a known issue that others working with the RIDB data have encountered. This resulted in 4,530 reservations (or 0.87%) without a valid booking window, which are removed for plots that visualize the booking window variable.

```
mutate(start_date = as.Date(start_date),
       end_date = as.Date(end_date),
       order_date = as.Date(order_date),
       # calculate new variables
       length_of_stay = as.numeric(difftime(end_date, start_date),
                                   units = "days"),
       booking_window = as.numeric(difftime(start_date, order_date),
                                   units = "days"))
```

We calculate daily cost per reservation by dividing total costs by lengths of stay. We then calculated daily cost per visitor by dividing the daily cost by the number of visitors.

```
# calculate new variables
mutate(daily_cost = total_paid / length_of_stay,
       daily_cost_per_visitor = daily_cost / number_of_people)
```

We aggregate the `site_type` variable to create 7 broader site categories.

```
# aggregate site type
mutate(aggregated_site_type =
         case_when(site_type %in% c("walk to", "hike to",
                                    "group hike to", "group walk to"
         ) ~ "remote",
         site_type %in% c("cabin nonelectric", "cabin electric",
                          "yurt","shelter nonelectric"
         ) ~ "shelter",
         site_type %in% c("boat in", "anchorage") ~ "water",
         site_type %in% c("group equestrian",
                          "equestrian nonelectric"
         ) ~ "equestrian",
         site_type %in% c("rv nonelectric", "rv electric",
                          "group rv area nonelectric"
         ) ~ "rv only",
         site_type %in% c("group standard nonelectric",
                          "standard nonelectric",
                          "standard electric",
                          "group standard area nonelectric",
                          "group standard electric"
         ) ~ "rv or tent",
         site_type %in% c("tent only nonelectric",
                          "group tent only area nonelectric",
                          "tent only electric"
         ) ~ "tent only"))
```

We create an administrative unit variable by combining the `parent_location` and `region_description` variables as different federal agencies track the administrative unit information within different variables. Then we update the `agency`, `admin_uni` and `park` variable character strings using multiple functions from the `stringr` package [Wickham, 2019]. These updates expand acronyms, remove unnecessary characters (such as "–" or location codes), and normalize any discrepencies in characater strings.

```
mutate(admin_unit =
         case_when(agency == "USFS" ~ parent_location,
                   agency %in% c(
                     "NPS", "BOR", "USACE"
                   ) ~ region_description)) %>%
  # edit values
  mutate(
    # agency abbreviations to names
```

```
    agency = str_replace(string = agency,
                         pattern = "NPS",
                         replacement = "National Park Service"),
    agency = str_replace(string = agency,
                         pattern = "USFS",
                         replacement = "US Forest Service"),
    agency = str_replace(string = agency,
                         pattern = "USACE",
                         replacement = "US Army Corps of Engineers"),
    agency = str_replace(string = agency,
                         pattern = "BOR",
                         replacement = "Bureau of Reclamation"),
    # update admin_unit values (generic)
    admin_unit = str_replace(string = admin_unit,
                             pattern = paste(c("NF - FS", "NF -FS",
                                               "NF- FS", "NF-FS",
                                               "-FS", " - FS"),
                                             collapse = "|"),
                             replacement = "National Forest"),
    admin_unit = str_to_title(admin_unit),
    admin_unit = str_replace(string = admin_unit,
                             pattern = "And",
                             replacement = "&"),
    # update park values (generic)
    park = str_remove(string = park,
                      pattern = paste(c("\\(.*", " \\(.*",
                                        "---.*", " ---.*",
                                        ",.*"),
                                      collapse = "|")),
    park = str_to_title(park),
    park = str_replace(string = park,
                       pattern = "Cg",
                       replacement = "Campground"),
    park = str_replace(string = park,
                       pattern = "Nhp",
                       replacement = "National Historic Park"),
    park = str_replace(string = park,
                       pattern = "@",
                       replacement = "At"),
    park = str_replace(string = park,
                       pattern = "&",
                       replacement = "And"),
    park = str_replace(string = park,
                       pattern = paste(c("/", " / "), collapse = "|"),
                       replacement = " "),
    park = str_remove_all(string = park,
                          pattern = " \\d.*"),
```

```
    # update park values (CA specific)
    park = str_remove(string = park,
                      pattern = paste(c(" - Angeles Nf", " -Hwy"),
                                      collapse = "|")),
    park = str_replace(string = park,
                       pattern = "Tunnel Mills Il",
                       replacement = "Tunnel Mills"))
```

We calculate distance traveled by measuring the distance from the latitude and longitude coordinate facility locations to the centroid of the home ZIP code. Here we utilize both the `tidycensus` package [Walker and Herman, 2021] and the `sf` package [Pebesma, 2022].

```
# bootstrap geometries and reproject to NAD 83
df_geometries <- df %>%
  st_as_sf(coords = c("facility_longitude", "facility_latitude"),
           crs = 4326) %>%
  st_transform(crs = 4269) # using NAD83 because measured in meters

# get centroid of geometries for all US ZIP codes
df_zip_centroids_us <-
  get_acs(geography = "zcta", year = 2018,
          geometry = TRUE,
          summary_var = "B01001_001",
          survey = "acs5",
          variables = c(male = "B01001_002")) %>%
  select(NAME, geometry) %>%
  mutate(zip_code = str_sub(NAME, start = -5, end = -1)) %>%
  select(zip_code, geometry) %>%
  st_centroid()

# join data and calculate `distance_traveled` variable
df_joined_geometries <-
  left_join(x = df_geometries %>% as.data.frame(),
            y = df_zip_centroids_us %>% as.data.frame(),
            by = c("customer_zip" = "zip_code")) %>%
  st_sf(sf_column_name = 'geometry.x') %>%
  mutate(distance_traveled_m = st_distance(x = geometry.x,
                                           y = geometry.y,
                                           by_element = TRUE),
         distance_traveled_m = as.numeric(distance_traveled_m))
```

And finally, we add a variable indicating in which state or territory each customer zip code is located. This portion of the code utilizes the `zipcodeR` package [Rozzi, 2021].

```
# create df of fips and full state names
fips_list <- c(
  "01", "02", "04", "05", "06", "08", "09", "10", "11", "12",
```

```
  "13", "15", "16", "17", "18", "19", "20", "21", "22", "23",
  "24", "25", "26", "27", "28", "29", "30", "31", "32", "33",
  "34", "35", "36", "37", "38", "39", "40", "41", "42", "44",
  "45", "46", "47", "48", "49", "50", "51", "53", "54", "55",
  "56", "72")
state_list <- c(
  "AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "DC", "FL",
  "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME",
  "MD", "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH",
  "NJ", "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI",
  "SC", "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI",
  "WY", "PR")
states_names_list <- c(
  "Alabama", "Alaska", "Arizona", "Arkansas", "California",
  "Colorado", "Connecticut", "Delaware",
  "District of Columbia", "Florida","Georgia", "Hawaii",
  "Idaho", "Illinois", "Indiana", "Iowa", "Kansas",
  "Kentucky", "Louisiana", "Maine","Maryland",
  "Massachusetts", "Michigan", "Minnesota", "Mississippi",
  "Missouri", "Montana", "Nebraska", "Nevada", "New Hampshire",
  "New Jersey", "New Mexico", "New York", "North Carolina",
  "North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania",
  "Rhode Island","South Carolina", "South Dakota", "Tennessee",
  "Texas", "Utah", "Vermont", "Virginia", "Washington",
  "West Virginia", "Wisconsin","Wyoming", "Puerto Rico")
df_states_fips <- as.data.frame(list(fips = fips_list,
                                     state = state_list,
                                     state_full = states_names_list))

# loop through state df to get all ZIP codes w/in state
df_states_zip_codes <- data.frame()

for (i in seq_along(fips_list)){
  state <- zipcodeR::search_fips(state_fips = fips_list[[i]]) %>%
    select(zipcode, state)
  df_states_zip_codes <- rbind(df_states_zip_codes, state)
}

# add full state name and fips code to list of all ZIP codes for each state
df_states_fips_zip_codes <-
  left_join(x = df_states_zip_codes,
            y = df_states_fips,
            by = "state") %>%
  select(-fips) %>%
  rename(customer_zip_state = state,
         customer_zip_state_full = state_full,
         zip_code = zipcode)
```

8.3.2.2 U.S. Census Data

US Census data is publicly accessible in many ways. Our product utilizes the R package `tidycensus` [Walker and Herman, 2021] to access the necessary variables from the 2018 American Community Survey (ACS) via API. API access requires an api key, which can then be saved into your RStudio environment. Learn more about API keys and working with `tidycensus` here.

```
# API set up
# Only have to run this the first time using on a new machine
census_api <- source("private/census-api.R")
census_api_key(key = census_api[[1]], install = TRUE, overwrite = TRUE)
# run in console:
readRenviron("~/.Renviron")
```

Sample data are collected for the ACS each year and includes many variables that cover social, economic, housing, and demographic characteristics. All variable options can be easily viewed using the code below:

```
# look at variable options
View(load_variables(2018, "acs5", cache = TRUE))
```

The Outdoor Equity App utilizes the ACS 5-year data, which is an estimate representing data collected over the designated 5 year period. We used the 5-year ACS data over the 1-year ACS data because it increases "statistical reliability of the data for less populated areas and small population subgroups" [Bureau, 2022].

The variables included in the App include median-income, race, language(s) spoken at home, and highest level of education attained. All variables are represented as estimates in numeric format for a census tract. Data are called by geographic region, in our case the ZIP code tabulation area, and include an estimated number of people that fall into each category within each ZIP code, a margin of error, and an estimated number of total people in the area. Within our custom functions `function_acs_education.R`, `function_acs_language.R`, `function_acs_median_income.R`, and `function_acs_race.R` we first imported just the necessary columns for each ACS variable. See Table 8.1 for an example of the race variable.

```
# import variables for race
race_df <-
  get_acs(
    geography = "zcta", year = 2018,
    state = "CA",
    summary_var = "B03002_001", #Estimate!!Total:
    variables = c(
      white = "B03002_003", #White alone
      black = "B03002_004", #Black or African American alone
      native_american = "B03002_005", #American Indian and Alaska Native alone
      asian = "B03002_006", #Asian alone
      pacific_islander = "B03002_007", #Native Hawaiian/Other Pacific Islander alone
      other = "B03002_008", #Some other race alone
```

Table 8.1: Example of raw ACS Race Variable.

GEOID	NAME	variable	estimate	moe	summary_est	summary_moe
90001	ZCTA5 90001	white	413	165	58975	1725
90001	ZCTA5 90001	black	5138	646	58975	1725
90001	ZCTA5 90001	native_american	37	42	58975	1725
90001	ZCTA5 90001	asian	67	38	58975	1725
90001	ZCTA5 90001	pacific_islander	0	29	58975	1725
90001	ZCTA5 90001	other	168	169	58975	1725
90001	ZCTA5 90001	multiracial	66	44	58975	1725
90001	ZCTA5 90001	hispanic_latinx	53086	1591	58975	1725
90002	ZCTA5 90002	white	223	81	53111	2031
90002	ZCTA5 90002	black	10110	876	53111	2031

```
      multiracial = "B03002_009", #Two or more races
      hispanic_latinx = "B03002_012" #Hispanic or Latino
    ))
```

We calculated percentages for the categories within each ACS variable for race, language(s) spoken in the home, and highest level of education by dividing the estimate for a category by the estimated total population for that ZIP code.

```
race_df_percent <-
  race_df %>%
  group_by(zip_code, race) %>%
  summarise(estimate = sum(estimate),
            moe = sum(moe),
            summary_est = unique(summary_est),
            summary_moe = unique(summary_moe),
            percent = estimate / summary_est)
```

For median-income, we use the estimated median-income as is, without further adjustments.

```
median_income_df <-
  get_acs(geography = "zcta", year = 2018,
          state = "CA",
          variables = c(
            median_income = "B19013_001"
          )) %>%
  clean_names() %>%
  rename(median_income = estimate) %>%
  mutate(zip_code = str_sub(name, start = -5, end = -1)) %>%
  select(median_income, zip_code)
```

We transform teh ACS dataframe by pivoting wider, which moves the different categories and percentages (ex: racial groups) to their own columns to create a single row for each ZIP code. This is necessary for joining the ACS data sets to the RIDS data set.

```
select(zip_code, summary_est, race, percent) %>%
  rename(zip_code_population = summary_est) %>%
  # create column for each percentage for each group (pivot_wider)
  # necessary to be able to left_join() with RIDB data
  pivot_wider(names_from = "race",
              values_from = "percent")
```

Finally, we use the `function_join_ridb_acs.R` custom functions to join the RIDB and ACS data to create the dataset that is available for download on the App and utilized for all plots and maps.

```
joined_df <-
  left_join(x = ridb_df,
            y = acs_df_race,
            by = c("customer_zip" = "zip_code")) %>%
  left_join(y = acs_df_education,
            by = c("customer_zip" = "zip_code")) %>%
  left_join(y = acs_df_median_income,
            by = c("customer_zip" = "zip_code")) %>%
  left_join(y = acs_df_language,
            by = c("customer_zip" = "zip_code"))
```

8.3.3 Statistical Analysis and Data Wrangling for Plots

Each type of plot and map require unique data wrangling which is explained in this section.

8.3.3.1 Data Summary

We created custom functions for data wrangling and plotting for use within the Outdoor Equity App code. Wrangling begins with filtering based on the user's choice campsite input. We then further subset the dataset to include only the necessary columns for plotting.

```
# example for booking window data summary
booking_window_rdf <- reactive({
  validate(
    need(siteInput != "",
         "Please select a reservable site to visualize.")
  ) # EO validate

  ridb_df %>%
    filter(park %in% siteInput,
           booking_window > 0,
           booking_window != "Inf") %>%
    select(park, booking_window) %>%
    filter(!is.na(booking_window))
})
```

We create histograms, bar charts, or density plots to show the distribution of the data for each variable. For ACS variables, data from reservations are plotted against data for the full California population. This allows a user to compare the distribution within the reservation data to that of the California census in order to see where reservations either under- or over-represent that variable as compared to California residents. We used the `plotly` package [Sievert et al., 2021] to allow a user to hover over the plot to view helper text. We also included additional helper text outside of the plots for more complicated plots.

```
## -- example for booking window data summary -- ##
# parameters
hist_colors <- c("#64863C", "#466C04")
quant_80_color <- c("#FACE00")
caption_color <- c("#ac8d00")

# plot for shiny app
plotly <- ggplot(
  data = booking_window_rdf()) +
  geom_histogram(
    aes(x = booking_window,
        text = paste0(percent(..count.. / nrow(booking_window_rdf()),
                              accuracy = 0.1),
                      " of all visits are reserved between ", xmin,
                      " and ", xmax,
                      " days before the start of the visit",
                      "<br>",
                      "(Total reservations to site: ",
                      comma(nrow(booking_window_rdf()), accuracy = 1),
                      ")")),
    binwidth = 7,
    center = 7 / 2,
    fill = hist_colors[[1]],
    col = hist_colors[[2]], size = 0.05) +
  labs(x = "Days in advance before visit (each bar = 1 week)",
       y = "") +
  scale_x_continuous(limits = c(0, x_max)) +
  geom_vline(xintercept = quant_80,
             linetype = "dashed", alpha = 0.5, color = quant_80_color) +
  theme_minimal() +
  theme(plot.background = element_rect("white"),
        panel.grid.major.y = element_blank())

# add 6 month / 1 year annotation if data allows
if (x_max <= 180) {
  # don't include 6 month or 1 year annotation
  plotly

} else if (x_max > 180 & x_max <= 360){
  # include 6 month annotation
```

```
  plotly <- plotly +
    geom_vline(xintercept = 180,
               linetype = "dashed", size = .3, alpha = .5) +
    annotate("text", label = "6 months",  size = 3,
             x = 180, y = 5)

} else if (x_max >= 360) {

  # include 6 month and 1 year annotation
  plotly <- plotly +
    geom_vline(xintercept = 180,
               linetype = "dashed", size = .3, alpha = .5) +
    annotate("text", label = "6 months", size = 3,
             x = 180, y = 5) +
    geom_vline(xintercept = 360,
               linetype = "dashed", size = .3, alpha = .5) +
    annotate("text", label = "1 year", size = 3,
             x = 360, y = 5)
} # EO else if for plotly

ggplotly(plotly,
         tooltip = list("text"),
         dynamicTicks = TRUE) %>%
  layout(title =
           list(text =
                  paste0('<b>', siteInput, '<br>', admin_unitInput, '</b>',
                         '<br>',
                         'Number of days from reservation to start of visit'),
                font = list(size = 15)),
         xaxis = list(separatethousands = TRUE),
         yaxis = list(separatethousands = TRUE),
         margin = list(b = 130, t = 100),
         annotations =
           list(x = x_max/2, y = -0.6,
                text =
                  paste0("80% of reservations reserve their visit less than ",
                         '<b>', quant_80, '</b>',
                         " days before the start date",
                         "<br>",
                         "(shown on plot with yellow dotted line)."),
                showarrow = F,
                xre = 'paper', yref = 'paper',
                           xanchor = 'middle', yanchor = 'auto',
                           xshift = 0, yshift = 0,
                           font = list(size = 12, color = caption_color))) %>%
  config(modeBarButtonsToRemove = list("pan", "select", "lasso2d", "autoScale2d",
                                       "hoverClosestCartesian",
```

```
                                        "hoverCompareCartesian"))
```

8.3.3.2 Data Relationships

Visualizing the relationship between RIDB and ACS variables is challenging because the socioeconomic data available are an estimate of the ZIP code population, rather than data of the individual making the reservation. In order to create simple, easy to interpret plots, we determined whether reservations were from a location with "high" percentages of a given ACS variable category (ex: one of the eight racial groups). We determined "high" as anything above the weighted 3rd quartile (weighted based on the California census). This method allows for reservations to be included in the "high" category for multiple values within one ACS variable (ex: a reservation might be from a ZIP code that falls into the "high" category for both Black and Asian folks). By allowing a reservation to appear in multiple categories within a single ACS variable, the uncertainty of the reserver's actual socioeconomic status is retained.

Example code for calculating the "high" threshold for language can be seen below:

```
# `function_acs_top_quartile_language.R` code
threshold_df <- acs_df %>%
  # filter variables of interest
  select(zip_code, english_only, not_english_only, mean_zip_code_population) %>%
  pivot_longer(cols = 2:3,
               names_to = "language",
               values_to = "language_percentage") %>%
  # filter category of interest
  filter(language == language_group) %>%
  drop_na(language_percentage)

# weighted median value (weighted based on ZIP code populations)
weighted_half <- weighted.mean(x = threshold_df$language_percentage,
                               w = threshold_df$mean_zip_code_population)

# drop rows below weighted median
df_half <- threshold_df %>% filter(language_percentage >= weighted_half)

# weighted 3rd quartile
## weighted median value of top half (weighted based on ZIP code populations)
weighted_quartile <- weighted.mean(x = df_half$language_percentage,
                                   w = df_half$mean_zip_code_population)
```

```
source("r/function_acs_top_quartile_language.R")

# language groups
language_group <- c("english_only", "not_english_only")

# calculate value of 3rd quartile for each language group
language_quants_df <-
```

Table 8.2: Threshold values for ACS language and education variables.

Language Group	Threshold
English Only	72.98%
Not English Only	62.75%

Education Group	Threshold
Hs Ged Or Below	54.21%
Some College	25.62%
College	36.85%
Master Or Above	22.02%

Table 8.3: Threshold values for ACS race variable.

Race Group	Threshold
Other	0.52%
Pacific Islander	0.87%
Multiracial	4.38%
Asian	30.29%
Black	13.16%
White	59.08%
Native American	0.93%
Hispanic Latinx	62.07%

```
  language_group %>%
  map_dbl(language_top_quartile, acs_df = data_acs_ca_all) %>%
  cbind("language_group" = language_group,
        "weighted_quartile" = .) %>%
  as.data.frame() %>%
  mutate(language_group = str_replace_all(string = language_group,
                                          pattern = "_",
                                          replacement = " "),
         language_group = str_to_title(language_group),
         weighted_quartile = percent(as.numeric(weighted_quartile),
                                     accuracy = 0.01)) %>%
  rename("Language Group" = language_group,
         "Threshold" = weighted_quartile)
```

The thresholds used within the Outdoor Equity App can be seen in Table 8.2 and Table 8.3.

We create lollipop and bar plots to show the relationships of the data for each set of variables. We used the `plotly` package [Sievert et al., 2021] to allow a user to hover over the plot to view helper text. An example of the code used to create the education compared to booking window plot is show below:

```
## -- data wrangle -- ##s
# create reactive dataframe and further subset
rdf <- reactive ({

  validate(
```

```
    need(siteInput != "",
         "Please select a reservable site to visualize.")
  ) # EO validate

  education_top_quartile_df %>%
    # filter to user site of choice
    filter(park == siteInput) %>%
    # select to variables of interest
    select(park, customer_zip,
           education, education_percentage, education_y_lab,
           booking_window) %>%
    drop_na(booking_window, education_percentage) %>%
    filter(booking_window >= 0) %>%
    # summarize to inner quartile range, median, and total reservations
    group_by(education, education_y_lab) %>%
    summarize(median_booking_window = median(booking_window),
              quartile_lower = quantile(booking_window)[[2]],
              quartile_upper = quantile(booking_window)[[4]],
              count = n())

}) #EO reactive df

validate(need(
  nrow(rdf()) > 0,
  paste0("There are no reservations to ", siteInput, ", ", admin_unitInput,
         " that come from communities in the high range for any educational",
         "categories.")
)) # EO validate

## -- create plot -- ##
# parameters
education_group_colors <-
  c(
    "HS, GED,\nor Below" = "#a6cee3",
    "Some College or\nTrade School"  = "#1f78b4",
    "Associates or\nBachelors Degree" = "#b2df8a",
    "Masters Degree\nor Above" = "#33a02c"
  )

# create plot
plotly <- ggplot(data = rdf(),
                 aes(x = median_booking_window,
                     y = education_y_lab)) +
  geom_segment(aes(xend = 0, yend = education_y_lab)) +
  geom_point(
    aes(
      color = education_y_lab,
```

```
      fill = education_y_lab,
      text = paste0(
        comma(count, accuracy = 1),
        " unique visits were made by people who live in ZIP codes with high ",
        " rates of",
        "<br>", education,
        " as the maximum level of education. ",
        "Typically these visitors reserved their visit between",
        "<br>", comma(quartile_lower, accuracy = 1), " and ",
        comma(quartile_upper, accuracy = 1),
        " days before the start of their trip, with a median booking window of ",
        comma(median_booking_window, accuracy = 1),  " days."
      )
    ),
    size = 3.5, shape = 21, stroke = 2 ) +
  scale_y_discrete(expand = c(0.45, 0)) +
  scale_fill_manual(values = education_group_colors) +
  scale_color_manual(values = education_group_colors) +
  labs(x = paste("Estimated Number of Days in Advance Site is Reserved"),
       y = "") +
  theme_minimal() +
  theme(
    plot.background = element_rect("white"),
    panel.grid.major.y = element_blank(),
    legend.position = "none")

# create plotly
ggplotly(plotly,
         tooltip = list("text")) %>%
  config(
    modeBarButtonsToRemove =
      list("zoom", "pan", "select", "lasso2d", "autoScale2d",
           "hoverClosestCartesian", "hoverCompareCartesian")) %>%
  layout(title = list(
    text = paste0( '<b>', siteInput, '<br>', admin_unitInput, '</b>', '<br>',
                   'Number of Days in Advance Site is Reserved by Visitors with",
                   "Different Levels of Education'),
    font = list(size = 15) )) %>%
  add_annotations(
    text = "Reservations from ZIP codes<br>with high proportions of:",
    x = -0.15, y = 0.9,
    font = list(size = 11),
    xref = 'paper', yref = 'paper',
    showarrow = FALSE)
```

8.3.3.3 Spatial analysis

We created a state-level visitorshed map to show the number of visitors to overnight reservable sites from each state. We use the package `zipcodeR` [Rozzi, 2021] to determine the state in which each ZIP code, and thus visitor, is located, using the RIDB data. We then create a dataframe with the necessary geometries using the `tigris` [Walker, 2022] package. And finally we simplify the geometries to lower the load time within the App using the `rmapshaper` [Teucher and Russell, 2021] package.

```r
# ZIP codes and respective states
fips_list <-
  c("01", "02", "04", "05", "06", "08", "09", "10", "11", "12",
    "13", "15", "16", "17", "18", "19", "20", "21", "22", "23",
    "24", "25", "26", "27", "28", "29", "30", "31", "32", "33",
    "34", "35", "36", "37", "38", "39", "40", "41", "42", "44",
    "45", "46", "47", "48", "49", "50", "51", "53", "54", "55",
    "56", "72")
state_list <-
  c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "DC", "FL",
    "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME",
    "MD", "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH",
    "NJ", "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI",
    "SC", "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI",
    "WY", "PR")
states_names_list <-
  c("Alabama", "Alaska", "Arizona", "Arkansas", "California",
    "Colorado", "Connecticut", "Delaware", "District of Columbia",
    "Florida","Georgia", "Hawaii", "Idaho", "Illinois", "Indiana",
    "Iowa", "Kansas", "Kentucky", "Louisiana", "Maine","Maryland",
    "Massachusetts", "Michigan", "Minnesota", "Mississippi", "Missouri",
    "Montana", "Nebraska", "Nevada", "New Hampshire","New Jersey",
    "New Mexico", "New York", "North Carolina", "North Dakota", "Ohio",
    "Oklahoma", "Oregon", "Pennsylvania", "Rhode Island","South Carolina",
    "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia",
    "Washington", "West Virginia", "Wisconsin","Wyoming", "Puerto Rico")

# create dataframe of states
df_states_fips <- as.data.frame(list(fips = fips_list,
                                     state = state_list,
                                     state_full = states_names_list))

# loop through state df to get all ZIP codes w/in state
df_states_zip_codes <- data.frame()

for (i in seq_along(fips_list)){
  state <- zipcodeR::search_fips(state_fips = fips_list[[i]]) %>%
    select(zipcode, state)
  df_states_zip_codes <- rbind(df_states_zip_codes, state)
```

```
}

# add full state name, fips code, etc. to list of all ZIP codes for each state
df_states_fips_zip_codes <-
  left_join(x = df_states_zip_codes,
            y = df_states_fips,
            by = "state") %>%
  select(fips, zipcode) %>%
  rename(zip_code = zipcode)

# get state geometries for full US
df_state_geometries_us <- tigris::states(year = 2018) %>%
  select(GEOID, STUSPS, NAME, geometry) %>%
  rename(fips = GEOID,
         state_abbrev = STUSPS,
         state = NAME) %>%
  rmapshaper::ms_simplify(keep = 0.005, keep_shapes = TRUE)
```

We then calculate the number of reservations per state for the site (as chosen by the app user) and use `tmaps` package [Tennekes, 2022] to create an interactive map that colors states based on the total number of reservations for that state.

```
## -- data wrangle -- ##
# reactive data frame for siteInput
rdf <- reactive ({

  validate(
    need(siteInput != "",
         "Please select a reservable site to visualize.")
  ) # EO validate

  ridb_df %>%
    filter(park %in% siteInput) %>%
    select(agency, admin_unit, park, customer_zip_state_full,
           customer_zip_state)
})

# value of total reservations for this park
total_reservations <- nrow(rdf())

# number of reservations and % per state
map_data <- rdf() %>%
  group_by(customer_zip_state_full, customer_zip_state) %>%
  summarize(number_reservations = n(),
            percentage_reservations =
              percent((number_reservations / total_reservations),
                                                  accuracy = 0.01)) %>%
```

```
  filter(!is.na(customer_zip_state_full))

# add state geometries
map_data_geometries <-
  state_geometries_df %>%
  left_join(y = map_data,
            by = c("state_abbrev" = "customer_zip_state")) %>%
  mutate_at(vars(number_reservations),
            ~replace(number_reservations, is.na(number_reservations), 0)) %>%
  mutate_at(vars(percentage_reservations),
            ~replace(percentage_reservations, is.na(percentage_reservations),
                     0)) %>%
  select(customer_zip_state_full, number_reservations,
         percentage_reservations, geometry)

## -- create map -- ##
tmap_mode("view")

tm_shape(map_data_geometries) +
  tm_borders(col = "grey", alpha = 0.5) +
  tm_fill(col = "number_reservations",
          title = "Number of Visits",
          palette = "YlGn",
          n = 10,
          style = "jenks",
          id = "customer_zip_state_full",
          popup.vars = c("Total Visits" = "number_reservations",
                         "Percentage of All Visits" = "percentage_reservations")
          ) +
  tm_view(set.view = c(-101.834335, 40.022356, 2)) +
  tmap_options(basemaps = 'https://server.arcgisonline.com/ArcGIS/rest/services/Canvas/World_Li
```

A ZIP-code level visitorshed map shows the number of visitors to sites for each ZIP code within California only due to computational load of calculating for all ZIP codes in the United States. We then create a dataframe with the necessary geometries using the `tigris` [Walker, 2022] package. And finally we simplify the geometries to lower the load time within the App using the `rmapshaper` [Teucher and Russell, 2021] package.

```
## -- ZIP codes and respective states -- ##
fips_list <-
  c("01", "02", "04", "05", "06", "08", "09", "10", "11", "12",
    "13", "15", "16", "17", "18", "19", "20", "21", "22", "23",
    "24", "25", "26", "27", "28", "29", "30", "31", "32", "33",
    "34", "35", "36", "37", "38", "39", "40", "41", "42", "44",
    "45", "46", "47", "48", "49", "50", "51", "53", "54", "55",
    "56", "72")
state_list <-
```

```
  c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "DC", "FL",
    "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME",
    "MD", "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH",
    "NJ", "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI",
    "SC", "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI",
    "WY", "PR")
states_names_list <-
  c("Alabama", "Alaska", "Arizona", "Arkansas", "California",
    "Colorado", "Connecticut", "Delaware", "District of Columbia",
    "Florida","Georgia", "Hawaii", "Idaho", "Illinois", "Indiana",
    "Iowa", "Kansas", "Kentucky", "Louisiana", "Maine","Maryland",
    "Massachusetts", "Michigan", "Minnesota", "Mississippi", "Missouri",
    "Montana", "Nebraska", "Nevada", "New Hampshire","New Jersey",
    "New Mexico", "New York", "North Carolina", "North Dakota", "Ohio",
    "Oklahoma", "Oregon", "Pennsylvania", "Rhode Island","South Carolina",
    "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia",
    "Washington", "West Virginia", "Wisconsin","Wyoming", "Puerto Rico")

# create dataframe of states
df_states_fips <- as.data.frame(list(fips = fips_list,
                                     state = state_list,
                                     state_full = states_names_list))

# loop through state df to get all ZIP codes w/in state
df_states_zip_codes <- data.frame()

for (i in seq_along(fips_list)){
  state <- zipcodeR::search_fips(state_fips = fips_list[[i]]) %>%
    select(zipcode, state)
  df_states_zip_codes <- rbind(df_states_zip_codes, state)
}
# add full state name, fips code, etc. to list of all ZIP codes for each state
df_states_fips_zip_codes <-
  left_join(x = df_states_zip_codes,
            y = df_states_fips,
            by = "state") %>%
  rename(state_abbrev = state,
         zip_code = zipcode) %>%
  relocate(fips, .before = 2)


## -- CA ZIP geometries -- ##
df_zip_geometries_ca <- get_acs(geography = "zcta", year = 2018,
                                geometry = TRUE,
                                state = "California",
                                summary_var = "B01001_001",
                                variables = c(male = "B01001_002")) %>%
```

```
  select(NAME, geometry) %>%
  mutate(zip_code = str_sub(NAME, start = -5, end = -1)) %>%
  select(zip_code, geometry) %>%
  rmapshaper::ms_simplify(keep = 0.005, keep_shapes = TRUE) %>%
  left_join(y = df_states_fips_zip_codes,
            by = "zip_code") %>%
  relocate(zip_code, .before = 1)
```

We calculate the number of reservations per ZIP code for the site (as chosen by the app user) and use `tmaps` to create an interactive map that colors ZIP codes based on the total number of reservations for that ZIP code.

```
## -- data wrangle -- ##
# reactive data frame for siteInput
rdf <- reactive ({

  validate(
    need(siteInput != "",
         "Please select a reservable site to visualize.")
  ) # EO validate

  ridb_df %>%
    select(agency, admin_unit, park, customer_zip, facility_latitude,
           facility_longitude) %>%
    filter(park %in% siteInput)
})

# number of reservations per ZIP code
map_data <- rdf() %>%
  group_by(customer_zip) %>%
  summarize(number_reservations = n())

# join with ZIP geometries
map_data_geometries <-
  zip_geometries_df %>%
  left_join(map_data,
            by = c("zip_code" = "customer_zip")) %>%
  mutate_at(vars(number_reservations),
            ~replace(number_reservations, is.na(number_reservations), 0))  %>%
  select(zip_code, number_reservations, geometry)

# value of total CA reservations for this park
total_reservations <- sum(map_data_geometries$number_reservations)

# add percentage of all CA reservations for each ZIP
map_data_geometries <- map_data_geometries %>%
  mutate(percentage_reservations =
```

```
          percent((number_reservations / total_reservations),
                  accuracy = 0.01)) %>%
  mutate_at(vars(percentage_reservations),
            ~replace(percentage_reservations, is.na(percentage_reservations),
                     0))

# calculate location of park for point on map
park_location_geom <- rdf() %>%
  group_by(park) %>%
  summarise(facility_latitude = median(facility_latitude),
            facility_longitude = median(facility_longitude)) %>%
  st_as_sf(coords = c("facility_latitude", "facility_longitude"),
           crs = 4326) %>%
  st_transform(crs = 4269) # using NAD83 because measured in meters

print(paste("The site's location has been calculated and it is:",
            park_location_geom$geometry))

## -- create map -- ##
tmap_mode("view")

tm_shape(map_data_geometries) +
  tm_fill(
    col = "number_reservations",
    title = "Number of Visits",
    palette = "PuRd",
    style = "jenks",
    n = 10,
    popup.vars = c("Total Visits" = "number_reservations",
                   "Percentage of All CA Visits" = "percentage_reservations")) +
  tm_shape(park_location_geom) +
  tm_dots(col = "darkgreen", size = 0.1) +
  tm_shape(park_location_geom) +
  tm_symbols(shape = map_site_icon,
             id = "park") +
  tm_shape(ca_cities_df) +
  tm_text(col = "black",
          text = "city") +
  tm_view(set.view = c(-119.559917, 37.061753, 6)) +
  tmap_options(basemaps = 'https://server.arcgisonline.com/ArcGIS/rest/services/Canvas/World_L
```

8.3.4 Data updates

The `data_wrangle_and_clean.Rmd` document is set up to create the joined dataset necessary for the Outdoor Equity App. To access the ACS data we use the `tidycensus` package [Walker and Herman, 2021] to access the necessary data via API, meaning data updates are not necessary outside of the `.Rmd` document. The RIDB data is accesses via direct download from the Recreation.gov website. See the How to Expand the Outdoor Equity App Section on expanding the time periods included in the App.

8.3.5 Server Hosting

The Outdoor Equity App is hosted on server owned by the Bren School at the University of California, Santa Barbara until the end of 2022. At that time the App will be moved to the shiny.io server for the duration of 2023.

8.4 How to Expand the Outdoor Equity App

The Outdoor Equity App currently only includes reservation data from California for 2018 due to server hosting capabilities. Expanding the App to include additional years and states will require a larger server and likely more computing power.

8.4.1 Temporal Expansions

To expand the years of data included in the Outdoor Equity App, users must first update the `data_wrangle_and_clean.Rmd` where all data cleaning and wrangling occurs outside of the shiny app code.

For RIDB data from 2018 and early, users can simply include new code chunks in the `.Rmd` document that use the following functions to create a joined dataset for each new year: `function_ridb_subset-pre2018.R`, `function_ridb_variable_calculate-pre2018.R`, `function_acs_race.R`, `function_acs_median_income.R`, `function_acs_education.R`, `function_acs_language.R`, and `function_join_ridb_acs.R`.

For RIDB data from 2019 and later, users must update the `function_ridb_subset-post2019.R` function. We did not continue to update this function once we decided to include only 2018 data in the App. We recommend updating this function to first select only the necessary variables, then rename the variables so that they match the 2018 and earlier naming conventions, and finally copy and paste the remaining code used in the `function_ridb_subset-pre2018.R` function. Users can then add new code chunks to the `.Rmd` document that use the following functions to create a joined dataset for each new year: `function_ridb_subset-post2019.R`, `function_ridb_variable_calculate-pre2018.R`, `function_acs_race.R`, `function_acs_median_income.R`, `function_acs_education.R`, `function_acs_language.R`, and `function_join_ridb_acs.R`.

RIDB data from 2019 and later may have additional variables available, so we also recommend exploring all available variables to decide if adding in additional variables is appropriate. If users decide to add additional variables, edits will have to be made to all custom functions within the `data_preparation` and `outdoor-equity-app` directories.

Data standards for the RIDB data is in the process of being updated as of June 2022. Users should check to see if this impacts any decisions in adding additional years to the Outdoor Equity App.

For the Outdoor Equity App consider these updates to these files:

- `global.R`:
    - Add any new functions, packages, or data
- `ui.R`:
    - Create input widgets in Analysis tabs (data summary, data relationships, and visitorsheds) that allow for a user to select a specific year or mulitple years (consider making a function for this if the input is the same across all tabs)
- `server.R`:
    - Update plot or map functions that use a reactive data frame that includes the temporal input from `ui.R`

Additionally, for the entire App consider how load time will be affected by adding more years of data. It might be helpful to consider creating separated datasets or creating functions that create datasets that contain only the necessary data needed to create the visual(s).

8.4.2 Spatial Expansions

To expand the states included in the Outdoor Equity App, users must update the `data_wrangle_and_clean.Rmd` where all data cleaning and wrangling occurs outside of the shiny app code.

The `function_ridb_variable_calculate-pre2018.R` function has a section that manipulates the character strings of the `agency`, `admin_unit`, and `park` variables. This code would likely need to be changed for additional states as the specific irregularities in character strings may differ state to state. We recommend creating a new function for each additional state. See the RIDB Data Section for a more detailed description of this function code. Once a new custom function is created for each state, code chunks can be added to the `.Rmd` document to clean and wrangle data for additional states.

For the Outdoor Equity App consider these updates to these files:

- `global.R`:
    - Add any new functions, packages or data
    - Update or create new objects that contain information about sites and agencies
    - Update agency to administrative unit dictionary, and administrative unit to park dictionary so that these key value pairs appear for states beyond California
- `ui.R`:
    - Create an input widget that allows for someone to select a year or multiple years
    - Build out an informational box that contains basic information about the site chosen
- `server. R`:
    - Update `state_visitorshed_map()` function if the time input or other data needs to be added to this function

  - Update `ca_zip_code_visitorshed_map()` function to allow for a specific state to be called and the data associated with it

Additionally, for the entire App consider load time especially because spatial data is much larger and therefore takes much longer to load. It might be helpful to consider the structure of the data and which data gets loaded into the App to make the visuals, so that only the necessary data is being used in the back end of the App. See Technical Challenges Section for more insight and suggestions.

8.4.3 Statistical Analysis

Currently the Outdoor Equity App only visualizes patterns in the data, but does not allow a user to explore why these patterns are occurring. We recommend considering the following additions to expand the statistical analysis abilities of the App:

- Simple statistical analysis (such as linear regressions, ANOVA, etc.) to accompany all plots
- Ability to combine multiple campsites in a single plot or map to explore patterns and trends across geographic areas and years

For the Outdoor Equity App consider these updates to these files:

- `global.R`:
  - Add any new functions, packages, or data
- `ui.R`:
  - Create and add a new tab to the `navbarPage()` layout using the function `tabPanel()` for the statistical analysis
  - Build out layout in the new tab. To match the layout of the other tabs use `fluidRow()` and `box()` within `tabPanel()`
  - Create necessary inputs to create visuals or run statistical tests
- `server.R`:
  - Create necessary objects to render the visuals desired (i.e. reactive data frames, render plots or tables)
  - Create necessary conditional outputs based on an input using `observeEvents()`

Because statistical analysis was beyond the scope of our project, please note that these updates are not an exhaustive list of what needs to be done to successfully add statistical analysis to the App.

Chapter 9

Additional Challenges

9.1 Data Limitations

Socioeconomic data are not available within the RIDB data for each person making a reservation. This means that we must instead use the socioeconomic data available through American Community Survey (ACS), which are estimates for an entire ZIP code. Therefore, we must characterize the racial breakdown of the ZIP code associated with a reservation (e.g., 60% white, 30% Hispanic Latinx, 5% Asian, and 5% black), rather than assign a reservation to a given racial group. This creates limitations when creating plots that visualize any ACS variables.

9.2 Technical Challenges

In the Outdoor Equity App, there are behavioral bugs with the plots where plots will stop updating or being created after either an input has been changed multiple times or if the input chosen is associated with a large amount of data. We were not able to troubleshoot this completely, but believe this has to do with the functions that create the visuals.

Load time for all plots especially maps is a challenge that we were able to partially tackle, but not completely. For example, we used the package `rmapshaper` [Teucher and Russell, 2021] to simplify the geometries of the zip code and state polygons. This helped with load time significantly, but there may be other functions in `rmapshaper` [Teucher and Russell, 2021] that may help with reducing load time in the future. This will be crucial to consider as the spatial data will grow as the data expands. For plots, we reduced load time by doing as much data wrangling as possible outside of the App so that the functions creating the plots could update more quickly. We also used the package `purr` [Henry and Wickham, 2020] in our functions to iterate faster than for loops. Refer to Data Preparation Methods Section to see examples of code we used for data wrangling. In the future, there is likely room to improve our functions so that they are more efficient and will therefore run faster and reduce load time in the App.

9.3 Future Challenges

Adding in multiple years and adding in more spatial data (beyond California) will require a larger server and likely more computing power. Refer to How to Expand the Outdoor Equity App Section to learn more.

Chapter 10

Appendix

Table 10.1: A table of abbreviations, their definitions, and source URLS.

Abbreviation	Definition	Source
ACS	American Community Survey	https://www.census.gov/programs-surveys/acs
BLM	Bureau of Land Management	https://www.blm.gov/
BOR	Bureau of Reclamation	https://www.usbr.gov/
MEDS	Master of Environmental Data Science	https://bren.ucsb.edu/masters-programs/master-environmental-data-science
NPS	National Park Service	https://www.nps.gov/index.htm
R1S	Recreation One Stop	https://www.recreation.gov/
UCSB	University of California, Santa Barbara	https://www.ucsb.edu/
USACE	United States Army Corps of Engineers	https://www.usace.army.mil/
USFS	United States Forest Service	https://www.fs.usda.gov/

10.1 Glossary Table

Table 10.2: A table of the cleaning and wrangling functions created for the ACS and RIDB data and functions to create data sets for visitorshed maps and data relationship plots.

Script	Purpose
function_acs_deciles_median_income.R	Calculate decile values of California census household median-income
function_acs_education.R	Call and calculate education percentages for given geographic area and state
function_acs_language.R	Call and calculate language percentages for given geographic area and state
function_acs_median_income.R	Call and calculate median-income percentages for given geographic area and state
function_acs_race.R	Call and calculate race percentages for given geographic area and state
function_acs_top_quartile_education.R	Calculate weighted third quartile value of California census education percentages
function_acs_top_quartile_language.R	Calculate weighted third quartile value of California census language percentages
function_acs_top_quartile_race.R	Calculate weighted third quartile value of California census race percentages
function_ridb_subset-pre2018.R	Subset RIDB data
function_ridb_variable_calculate-pre2018.R	Define, standardize, and aggregate values and calculated additional derived variables
function_join_ridb_acs.R	Join RIDB and ACS data
function_map_ca_data.R	Create dataset for California ZIP code visitorshed map
function_map_us_data.R	Create dataset for US State visitorshed map
function_ridb_deciles_median_income.R	Create dataset for median-income data relationship plots
function_ridb_top_quartile_education.R	Create dataset for education data relationship plots
function_ridb_top_quartile_language.R	Create dataset for language data relationship plots
function_ridb_top_quartile_race.R	Create dataset for race data relationship plots

10.2 Functions Table

10.3 Metadata Table

Table 10.3: A table of metadata for the joined RIDB-ACS dataset

Variable Name	Definition
agency	the governing body that manages a type of US public land (i.e. national park, national forest)
admin_unit	the parent location or region description that a campsite belongs within
park	the name of a campsite
aggregated_site_type	type of site at a campsite; a campsite can have multiple site types
facility_id	unique id given to a campsite
facility_state	the state that a campsite is located in
customer_zip	the numeric code of the area from where a visitor lives
customer_zip_state	state acronym for home state of visitor
customer_zip_state_full	full name of state for home state of visitor
total_paid	total amount of dollars paid for a reservation
start_date	date when booked reservation begins
end_date	date when booked reservation ends
order_date	date when reservation was booked and purchased
number_of_people	number of people reported when booking reservation
length_of_stay	the number of days a visit is; difference of end date from start date
booking_window	the number of days a reservation is made before the start of the visit; difference of start date from order date
daily_cost	the total amount paid per day for a reservation
daily_cost_per_visitor	the total amount paid per day for one person
facility_latitude	latitude of the campsite, but note this may not be the center of the campsite
facility_longitude	longitude of the campsite, but note this may not be the center of the campsite
distance_traveled_m	distance between visitor home zip code and campsite
zip_code_population.x	the zip code population when get_acs() from tidycensus pulls in data for education variable. Note we take the average of zip code population x, zip code population y, and zip code population in our data wrangling script
asian	estimated percentage of asian population in a zip code
black	estimated percentage of black population in a zip code
hispanic_latinx	estimated percentage of hispanic latinx population in a zip code

Table 10.4: A table of information on packages used to create the Outdoor Equity App.

R Package	Version	Purpose
bslib	0.3.1	Web Application (theme)
collections	0.3.5	Web Application (dictionary)
DT	0.20	Web Application (tables)
formattable	0.2.1	Web Application (tables)
googlesheets4	1.0.0	Web Application (metadata)
here	1.0.1	Data cleaning (relative file paths)
janitor	2.1.0	Data cleaning (data frame)
lubridate	1.7.10	Data cleaning (dates)
plotly	4.10.0	Data visualiation (plots)
reactlog	1.1.0	Web Application (testing)
rmapshaper	0.4.5	Web Application (load time)
rsconnect	0.8.25	Web Application (deploy app)
scales	1.1.1	Data visualiation (plots)
sf	1.0.7	Data cleaning and visualization (maps)
shiny	1.7.1	Web Application (build app)
shinycssloaders	1.0.0	Web Application (plot loader)
shinydashboard	0.7.2	Web Application (dashboard layout elements)
shinydashboardPlus	2.0.3	Web Application (box elements)
shinyjs	2.1.0	Web Application (hide and show boxes in ui)
shinyWidgets	0.6.4	Web Application (inputs)
tidycensus	1.2.1	Data acquisition (US census data)
tidyverse	1.3.1	Data cleaning, analysis, and visualization
tigris	1.6	Data acquisition (state and ZIP code geometries)
tmap	3.3.2	Data visualization (maps)
vroom	1.5.7	Data cleaning (read in data)
zipcodeR	0.3.3	Data acquisition (state info about ZIP code)

10.4 Packages Table

10.5 Repository Directory Structure

Figure 10.1: Screenshot of the Metadata page of the Outdoor Equity App

Bibliography

John C. Bergstrom, Matthew Stowers, and Scott J. Shonkwiler. What does the future hold for u.s. national park visitation? estimation and assessment of demand determinants and new projections. 45(1):38 – 55, 2020. doi: 10.22004/AG.ECON.298433.

United States Census Bureau. American community survey 5-year data (2009-2020). website, 2022. URL: https://www.census.gov/data/developers/data-sets/acs-5year.html.

Winston Chang, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. *shiny: Web Application Framework for R*, 2021a. URL https://shiny.rstudio.com/. R package version 1.7.1.

Winston Chang, Gábor Csárdi, and Hadley Wickham. *shinytest: Test Shiny Apps*, 2021b. URL https://github.com/rstudio/shinytest. R package version 1.5.1.

Alan Ewert and Steve Hollenhorst. Resource allocation: Inequities in wildland recreation. 61(8): 32 – 36, 1990. doi: 10.1080/07303084.1990.10604598.

David Flores, Gennaro Falco, Nina S. Roberts, and III Francisco P. Valenzuela. Recreation equity: Is the forest service serving its diverse publics? 116(3):266 – 272, 2018. doi: 10.1093/jofore/fvx016.

Myron Floyd and Cassandra Y. Johnson. Coming to terms with environmental justice in outdoor recreation: A conceptual discussion with research implications. 24(1):59 – 77, 2002. doi: 10.1080/01490400252772836.

William E. Hammitt, David N. Cole, and Christopher A. Monz. *Wildland Recreation: Ecology and Management*. John Wiley and Sons, Oxford, UK, 2015.

Lionel Henry and Hadley Wickham. *purrr: Functional Programming Tools*, 2020. URL https://CRAN.R-project.org/package=purrr. R package version 0.3.4.

Sergei Izrailev. *tictoc: Functions for Timing R Scripts, as Well as Implementations of Stack and List Structures*, 2021. URL https://github.com/collectivemedia/tictoc. R package version 1.0.1.

E.P. Meinecke. Recreation planning: A discussion. 35(12):1120 – 1128, 1937. doi: 10.1093/jof/35.12.1120.

Edzer Pebesma. *sf: Simple Features for R*, 2022. URL https://CRAN.R-project.org/package=sf. R package version 1.0-7.

William L. Rice and So Young Park. Big data spatial analysis of campers' landscape preferences: Examining demand for amenities. 292(15), 2021. doi: 10.1016/j.jenvman.2021.112773.

William L. Rice, So Young Park, Bing Pan, and Peter Newman. Forecasting campground demand in us national parks. 75:424 – 438, 2019. doi: 10.1016/j.annals.2019.01.013.

Gavin Rozzi. *zipcodeR: Data & Functions for Working with US ZIP Codes*, 2021. URL https://CRAN.R-project.org/package=zipcodeR. R package version 0.3.3.

Joseph L. Sax. *Mountains Without Handrails: Reflections on the National Parks*. University of Michigan Press, Ann Arbor, MI, 1980.

Barret Schloerke. *reactlog: Reactivity Visualizer for shiny*, 2020. URL https://CRAN.R-project.org/package=reactlog. R package version 1.1.0.

Barret Schloerke, Alan Dipert, and Barbara Borges. *shinyloadtest: Load Test Shiny Applications*, 2021. URL https://CRAN.R-project.org/package=shinyloadtest. R package version 1.1.0.

David Scott and KangJae Jerry Lee. People of color and their constraints to national parks visitation. 35(1):73 – 82, 2018.

Mostafa Shartaj and Jordan F. Suter. Exploring the local determinants of campground utilization on national forest land. 18(2):114 – 128, 2020. doi: 10.22004/ag.econ.308121.

Bo Shelby, Doug Whittaker, and Mark Danley. Idealism versus pragmatism in user evaluations of allocation systems. 11(1):61 – 70, 1989. doi: 10.1080/01490408909512205.

Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec, and Pedro Despouy. *plotly: Create Interactive Web Graphics via plotly.js*, 2021. URL https://CRAN.R-project.org/package=plotly. R package version 4.10.0.

Martijn Tennekes. *tmap: Thematic Maps*, 2022. URL https://github.com/r-tmap/tmap. R package version 3.3-3.

Andy Teucher and Kenton Russell. *rmapshaper: Client for mapshaper for Geospatial Operations*, 2021. URL https://github.com/ateucher/rmapshaper. R package version 0.4.5.

Abby L. Timmons. Too much of a good thing: Overcrowding at america's national parks. 94(2): 985 – 1017, 2019.

Kyle Walker. *tigris: Load Census TIGER/Line Shapefiles*, 2022. URL https://github.com/walkerke/tigris. R package version 1.5.1.

Kyle Walker and Matt Herman. *tidycensus: Load US Census Boundary and Attribute Data as tidyverse and sf-Ready Data Frames*, 2021. URL https://walker-data.com/tidycensus/. R package version 1.1.

Margaret Walls, Casey Wichman, and Kevin Ankney. Nature-based recreation: Understanding campsite reservations in national parks. Technical report, Resources for the Future, 2018.

Hadley Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2019. URL https://CRAN.R-project.org/package=stringr. R package version 1.4.0.

Hadley Wickham. *tidyverse: Easily Install and Load the Tidyverse*, 2021. URL https://CRAN.R-project.org/package=tidyverse. R package version 1.3.1.

Hadley Wickham. *testthat: Unit Testing for R*, 2022. URL https://CRAN.R-project.org/package=testthat. R package version 3.1.4.

Xiao Xiao, KangJae Jerry Lee, and Lincoln R. Larson. Who visits u.s. national parks (and who doesn't)? a national study of perceived constraints and vacation preferences across diverse populations. 53(3):404 – 425, 2021. doi: 10.1080/00222216.2021.1899776.

Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*, 2021. URL https://CRAN.R-project.org/package=bookdown. R package version 0.23.